DEVELOPPEMENT DE MODÈLES MULTIMODAUX INTÉRACTIFS POUR L'APPRENTISSAGE DU LANGUAGE DANS DES ENVIRONEMENTS VISUELS

Thèse présentée et soutenue par

Florian STRUB

Pour obtenir le grade de

Docteur de l'Université de Lille *spécialité* **Informatique**

École doctorale, ED SPI 074 : Sciences pour l'Ingénieur

Devant le jury de thèse composé de:

Directeur de thèse:	**Prof. Olivier PIETQUIN**	Google Research, Brain Team
Directeur de thèse:	**Dr. Jérémie MARY**	Université de Lille, Criteo AI Lab
Président du jury:	**Prof. Patrick GALLINARI**	Sorbonne Université, Criteo AI Lab
Rapporteur:	**Prof. Verena RIESER**	Heriot-Watt University
Rapporteur:	**Dr. Patrick PÉREZ**	Valeo AI
Examinateur:	**Prof. Aaron COURVILLE**	Mila, Université de Montréal
Examinateur:	**Prof. Michèle SEBAG**	CNRS
Examinateur:	**Dr. Angeliki LAZARIDOU**	DeepMind

le 28 Janvier 2020 à Villeneuve d'Ascq, France

Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL), UMR 9189
Équipe SequeL, 59650, Villeneuve d'Ascq, France

Multimodal and Interactive Models for Visually Grounded Language Learning

Submitted by

Florian Strub

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy *in* **Computer Science**

University of Lille
Doctoral School SPI 074 : Engineering Sciences

Presented to the Doctoral Committee

Thesis Supervisor:	**Prof. Olivier Pietquin**	Google Research, Brain Team
Thesis Co-Supervisor:	**Dr. Jérémie Mary**	University of Lille, Criteo AI Lab
Committee Chair:	**Prof. Patrick Gallinari**	Sorbonne University, Criteo AI Lab
Thesis Referee:	**Prof. Verena Rieser**	Heriot-Watt University
Thesis Referee:	**Dr. Patrick Pérez**	Valeo AI
Thesis Jury:	**Prof. Aaron Courville**	Mila, University of Montreal
Thesis Jury:	**Prof. Michèle Sebag**	CNRS
Thesis Jury:	**Dr. Angeliki Lazaridou**	DeepMind

on the 28th of January, 2020 at Villeneuve d'Ascq, France

Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL), UMR 9189
Équipe SequeL, 59650, Villeneuve d'Ascq, France

Abstract

While our representation of the world is shaped by our perceptions, our languages, and our interactions, they have traditionally been distinct fields of study in machine learning. Fortunately, this partitioning started opening up with the recent advents of deep learning methods, which standardized raw feature extraction across communities. However, multimodal neural architectures are still at their beginning, and deep reinforcement learning is often limited to constrained environments. Yet, we ideally aim to develop large-scale multimodal and interactive models towards correctly apprehending the complexity of the world. As a first milestone, this thesis focuses on visually grounded language learning for three reasons (i) they are both well-studied modalities across different scientific fields (ii) it builds upon deep learning breakthroughs in natural language processing and computer vision (ii) the interplay between language and vision has been acknowledged in cognitive science. More precisely, we first designed the GuessWhat?! game for assessing visually grounded language understanding of the models: two players collaborate to locate a hidden object in an image by asking a sequence of questions. We then introduce modulation as a novel deep multimodal mechanism, and we show that it successfully fuses visual and linguistic representations by taking advantage of the hierarchical structure of neural networks. Finally, we investigate how reinforcement learning can support visually grounded language learning and cement the underlying multimodal representation. We show that such interactive learning leads to consistent language strategies but gives raise to new research issues.

Résumé

Alors que nous nous représentons le monde au travers de nos sens, de notre langage et de nos interactions, chacun de ces domaines a été historiquement étudié de manière indépendante en apprentissage automatique. Heureusement, ce cloisonnement tend à se défaire grâce aux dernières avancées en apprentissage profond, ce qui a conduit à l'uniformisation de l'extraction des données au travers des communautés. Cependant, les architectures neuronales multimodales n'en sont qu'à leurs premiers balbutiements et l'apprentissage par renforcement profond est encore souvent restreint à des environnements limités. Idéalement, nous aimerions pourtant développer des modèles multimodaux et interactifs afin qu'ils puissent correctement appréhender la complexité du monde réel. Dans cet objectif, cette thèse s'attache à la compréhension du langage combiné à la vision pour trois raisons : (i) ce sont deux modalités longuement étudiées aux travers des différentes communautés scientifiques (ii) nous pouvons bénéficier des dernières avancées en apprentissage profond pour les modèles de langues et de vision (iii) l'interaction entre l'apprentissage du langage et notre perception a été validé en science cognitives. Ainsi, nous avons conçu le jeu GuessWhat?! (KéZaKo) afin d'évaluer la compréhension de langue combiné à la vision de nos modèles : deux joueurs doivent ainsi localiser un objet caché dans une image en posant une série de questions. Nous introduisons ensuite le principe de modulation comme un nouveau module d'apprentissage profond multimodal. Nous montrons qu'une telle approche permet de fusionner efficacement des représentations visuelles et langagières en prenant en compte la structure hiérarchique propre aux réseaux de neurones. Enfin, nous explorons comment l'apprentissage par renforcement permet l'apprentissage de la langue et cimente l'apprentissage des représentations multimodales sous-jacentes. Nous montrons qu'un tel apprentissage interactif conduit à des stratégies langagières valides mais donne lieu à de nouvelles problématiques de recherche.

Acknowledgement

I would like to thank my advisors Prof. Olivier Pietquin and Jérémie Mary for their priceless help during this project. I thank Olivier for his insightful research perspectives that bootstrapped this full thesis and paved the way for fructuous collaborations. I would like to point out his patience and resilience towards me, and I acknowledge him for being right every time I was wrong. He makes me enjoy even more research, Belgium beers and medium-rare juicy steaks. I thank Jeremie for its constant refreshing and sharp view on machine learning alongside with its out-of-the-box ideas. He also makes me discovering Crocs shoes, whose I would have preferred to remain unaware. I also deeply thank both of them to have allowed me to come back to academia after working in the industry, which is a rare opportunity in France.

I am truly grateful for Prof. Philippe Preux for making SequeL such a talented place for research. He is an amazing and accessible team leader who keeps promoting the work of his team members. There would be far less machine learning and GPUs at Lille without him. I also would like to express my gratitude and my deep respect to Aaron Courville for his collaboration and constant help during this thesis. He is a supportive and trustworthy deep learning researcher; he even who woke up at 4 am to review one of our paper even if he may have already forgotten about it!

A special thank to my co-author and friend Harm de Vries. We would have never been that successful without each other support. We had many laughs and desperate experiments together, and I am happy to have met with such a gifted associate. I wish him all the best in finding a steady girlfriend in the upcoming years!

I have a special thought to the poor other Ph.D., Post-Doc, and senior folks that supported me during my lifespan in SequeL, namely: Julien Perolat (Please do not die while free diving), Alexandre Bérard (GoT is over!), Jean-Bastien Grill (He will graduate... someday), Nicolas Carrara (Pro Cuda installer), Merwan Barlier (The IronMan!), Daniele Calandriello (Life is not that over-complicated!), Ronan Fruit (He is sooo fruitful), Lilian Besson (Teacher to be), Mathieu Seurin (Keep Calm, drink coffee), Xuedong Shang (Grammar fixing lover), Yannis Flet-Berliac (The artist), Omar Darwiche (Still debugging TrailBlazer), Gergely Neu (Love Birds and Hard-Rock), Matteo Pirotta (Poor [but nice] italian guy), Pierre Ménard (Brulerie 1am as usual?), Bilal Piot (The only vegetarian wrestler), Michal Valko (who loves research and red wine), Alessandro Lazaric (Peaceful and wise italian guy), Romaric Gaudel (the North remembers you!), Odalric Maillard (Two-state MDP are yet to be solved) and Emilie Kaufmann (and her long expected duck cakes!).

Another thanks to other researchers I crossed along my Ph.D.: Ethan Perez, Vincent Dumoulin, Hado Van Hasselt, Matteo Hessel and Aja Huang.

This Ph.D. was funded through the European CHIST-ERA project IGLU, bravely supported by Prof. Jean Rouat. He spent quite some energy keeping us all together!

I would like to acknowledge my friends that prevent me from falling in complete insanity while I was rushing for never-ending deadlines: Jordan Loeuillier, Romain Laby, Veronique et Mathieu Chambefort, Moon, Marie-Agathe Charpagne, Romain Levy, Jeremy Gaudin, Sophie Villerot, Laurine Roblot, Florian Sauvestre and Thanh-Quy Nguyen.

Finally, I thank my Mammy, Daddy, Grand Daddies, and Grand Mammies, even if some of them are unfortunaly missing today.

Oh... I nearly forgot: I thank *you*, anonymous reader, that is brave enough to read this thesis. Please write your name here ___________________________. It will make me happy! By the way, if I forgot to acknowledge you, please forgive me and add your name to the previous empty slot to fix this misfortune. I would not forget you in my next Ph.D. manuscript(s)!

Contents

Introduction

Today, artificial systems are neither conscious nor dreaming of electrical sheep, but we expect them to understand their environment and consistently interact with it. Machine understanding thus became an acceptable compromise between the ancient dream of general artificial intelligence and industrial requirements; besides, recent deep learning successes seem like machine understanding are within research reach. Within a few years, we witnessed dramatic improvements over notoriously hard visual and linguistic challenges: object classification now matches human performances (Russakovsky et al., 2015) and fake news articles can be written from scratch (Radford et al., 2019). Unfortunately, the reality may be a bit more complex; as warned before the second Artificial Intelligence (AI) winter: "It is perilously easy to conclude that, because one has a program which works (in some sense), its representation of its knowledge must be more or less correct (in some sense)." (Hayes, 1985). Albeit unarguable accomplishments, deep networks may be facing this pitfall: error artifacts suggest that state-of-the-art models still lack a form of understanding and are no more than advanced but foolish automatons. Despite distinguishing hundreds of dog breeds, image classifiers can be fooled by altering a few pixels (Goodfellow et al., 2015); conversational agents may write a story about unicorns, but they can contradict themselves within a few sentences (Gao et al., 2019), revealing the agent understanding trickery.

A recurrent hypothesis is that models are conceived over a restricted representation of the world, and therefore cannot grasp its underlying complexity: "if we really want computers to understand us, we need to give them the ability to use more knowledge." Winograd (1971). When training agents over unimodal datasets or task-specific scenarios, the models may catch conceptual shortcuts and could fail to grasp high-level concepts that are required to develop abstract reasoning. In practice, neural networks are indeed reported to excel at exploiting cognitive bias and spurious correlations, like using image background to discriminate objects (Ribeiro et al., 2016) or indifferently answering questions after identifying keywords (Agrawal et al., 2016). The initial hypothesis is further supported by the so-called embodiment theory from cognitive science, which claims that our reasoning, language, and thoughts are inextricably shaped by our perception and actions (Barsalou, 2008; Gibbs Jr, 2005; Wilson and Foglia, 2011). When specifically dealing with language modeling, Harnad (1990) identified the symbol grounding problem, showing that pure linguistic models are always be inherently limited in their language understanding. This symbol grounding problem states that it is merely impossible to capture the meaning of a word if it is only defined by other words. As those words also need to be defined, it eventually results in circular and meaningless definitions. Therefore, a minimum subset of words has to be grounded by external experiences and representations to initiate language understanding. Following this idea, we here pursue the joint objective of developing multimodal models to ground concepts, and interactive agents to structure their representation.

Yet, the AI community has mostly been scattered into solving domain-specific tasks due to computational constraints or divergent research paradigms; each community has been working in partial isolation on unimodal problems such as vision, language or planning, with domain-specific solutions such as image feature descriptors, or computational grammar. Even recent deep learning breakthroughs have not entirely escaped this path as models are still trained on large unimodal datasets specific to each community (Krizhevsky et al., 2012; Sordoni et al., 2015). Nonetheless, deep learning greatly reduced the gap between communities by partially unifying learning methods and feature representations, and it revives the opportunity for learning from multiple perceptions. Furthermore, the neural hierarchical architecture offers new freedom to structure and process multimodal knowledge representations. Concurrently, reinforcement learning has matured toward learning original and complex policies from experience, paving the way for developing large scale and interactive agents.

In this manuscript, we investigate the necessary deep learning machinery to learn consistent multimodal representation and to cement it through interactions with reinforcement learning methods. More precisely, we concentrate our efforts on visually grounded language tasks as those two modalities provide the necessary components to assess our research hypotheses. On the cognitive side, language is inherently interactive, and perception is a natural modality to start tackling the symbol grounding problem (Mooney, 2008). On the technical side, we can build upon deep learning breakthroughs in natural language processing and computer vision. As detailed below, we decompose our research work into two steps. First, we contribute towards developing new multimodal architectures for visually grounded language tasks. Second, we extend reinforcement learning algorithms to natural language, and we assess whether interactive agents manage to ground language to visual cues.

When starting the thesis, perception and language just emerged as a fertile ground to study multimodal neural architecture (Malinowski, 2017; Strub et al., 2017a). Within a few years, we witnessed a surge toward visually grounded language tasks ranging from image captioning (Lin et al., 2014), visual question answering (Antol et al., 2015) to text-driven image generation (El-Nouby et al., 2018). This bloom came in pair with novel neural blocks to jointly process textual and visual information. However, the multimodal paradigm was to fuse the visual and language modalities at the last stages of neural processing (Wu et al., 2017). In this thesis, we argue that this late-stage aggregation fails to grasp the full complexity that entangles the modalities together, allowing for conceptual shortcuts at the early processing stages of neural networks. We here formalize the so-called neural modulation mechanism to take advantage of the deep and hierarchical nature of networks by interleaving sensory inputs at the different processing levels. We study whether this change of design paradigm allows for more comprehensive multimodal representations to emerge, to which we hope to be one of the cornerstones of the symbol grounding problem.

Although those neural design efforts are necessary, deep networks are solely trainable receptacles, and merely enhancing their machinery is meaningless without the proper training signals. Until now, supervised learning has been the dominant training paradigm for language models (Devlin et al., 2018; Kiros et al., 2015). We argue that such approaches may miss the interactive component in which language acquisition often takes place. Therefore, we explore whether reinforcement learning is a suitable framework for training natural language models in the second part of the thesis. We investigate whether the agent learns to consistently manipulate perceptual language by interacting on top of visual features, bridging the gap between multimodal and interactive learning.

In summary, we explore two original research directions towards tackling the symbol grounding problem: (i) how can we improve multimodal learning representation by interleaving visual and language cues over the full neural pipeline (ii) how can we take advantage of reinforcement learning to cement this multimodal representation to acquire language understanding. To that purpose, we first design an interactive visual and linguistic game, namely GuessWhat?!, as a testbed for our experiments. We then focus on visually grounded language understanding by introducing the modulation paradigm for multimodal architectures. Finally, we shift towards visually grounded language generation by casting the natural language acquisition problem into a reinforcement learning problem.

Thesis Outline

Technical background and state-of-the-art descriptions are developed in Chapters 1, 2 and 3. It includes work that have been published up to May 2019. The first chapter deals with machine learning techniques including basic supervised learning techniques, an overview of deep learning methods and reinforcement learning concepts. The second chapter shift towards domain-specific methods and history, iteratively examining computer vision, natural language processing. In each section, the thesis motivations are further developed with examples and evidences from other communities. The third chapter deals with multimodal learning, and provides a brief survey of visually grounded language tasks, methods and datasets.

This thesis begins by introducing the GuessWhat?! dataset in Chapter 4, describing the game rules, the dataset collection and statistics before establishing the task baselines. The GuessWhat?! game is then used as a recurrent benchmark throughout the thesis to evaluate the core difficulties of multimodal and interactive learning. Hence, GuessWhat?! helped to diagnose some of the weaknesses of previous state-of-the-art methods on visual and language understanding.

Chapters 5, 6 and 7 introduce the concept of feature modulation for deep neural networks and assess this new conditioning mechanism on visually grounded language understanding tasks. In Chapter 5, we first highlight the limitations of the dominant multimodal network architectures which fuse modalities at the uppermost neural processing layers. We then propose to deviate from this architecture paradigm by conditioning the entire visual processing pipeline by language, and show the benefits of this original approach. In Chapter 6, we analyze further the machinery behind modulation by using a synthetic dataset for visual reasoning; we also propose a simple but scalable and efficient modulation layer. Finally, we refine further this modulation layer and demonstrate its ability to work on large scale and natural inputs in Chapter 7.

Afterward, we explore how reinforcement learning may benefit visually grounded language generation in Chapter 8. We thus cast the GuessWhat?! game into a reinforcement learning problem, and train the agents in an interactive fashion, examining how their language and strategy may evolve by learning thought interaction.

The thesis finishes with a discussion of the future potential directions for both multimodal learning and interactive language learning considering our research observations. We briefly mention new leads for diagnosis tasks and potentially promising research area to adjoin towards improving machine understanding.

Contributions

The major thesis contributions can be summarized as follow:

- The collection of the first large-scale visually grounded dialogue dataset to assess visual understanding and language generation in an interactive manner.
- The formalization, analysis and spread of modulation mechanism for deep multimodal learning with the development of three specific neural modules.
- The first proof of concept that reinforcement learning can be used to train an agent to generate natural language and this method can outperform supervised learning approach in a goal-oriented dialogue setting
- The consolidation of the visually grounded language community by organizing multiple topic-related workshops

These contributions were performed in collaborations with other researchers. As it often goes with research, no-one can fully claim the full ownership of these contributions. Having said that, I tried to roughly approximate my day to-day involvement over published papers in Appendix A to not claim some of the work of my co-authors. However, I have always performed and/or reproduce *all* the experiments in this manuscript, and the chapters are updated version of the research papers.

List of Publications

Papers presented in this thesis

- H. de Vries, F. Strub, S. Chandar, O. Pietquin, H. Larochelle, and A. Courville. GuessWhat?! Visual object discovery through multi-modal dialogue. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017
- F. Strub, H. De Vries, J. Mary, B. Piot, A. Courville, and O. Pietquin. End-to-end optimization of goal-driven and visually grounded dialogue systems. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2017b
- H. De Vries*, F. Strub*, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017
- E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 2018
- F. Strub, M. Seurin, E. Perez, H. De Vries, J. Mary, P. Preux, O. Pietquin, and A. Courville. Visual reasoning with multi-hop feature modulation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018b

Workshop preprints are not included in this list.

Other published papers

- F. Strub, R. Gaudel, and J. Mary. Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems at RecSys*, 2016
- J. Pérolat, F. Strub, B. Piot, and O. Pietquin. Learning nash equilibrium for general-sum markov games from batch data. In *Proceedings of Artificial Intelligence and statistics (AIStats)*, 2016
- S. Brodeur, E. Perez, A. Anand, F. Golemo, L. Celotti, F. Strub, J. Rouat, H. Larochelle, and A. Courville. Home: A household multimodal environment. In *Visually-Grounded Interaction and Language workshop (ViGIL) at NeurIPS*, 2017
- V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. d. Vries, A. Courville, and Y. Bengio. Feature-wise transformations. *Distill*, 2018. https://distill.pub/2018/feature-wise-transformations
- H. Van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil. Deep reinforcement learning and the deadly triad. In *Deep Reinforcement Learning Workshop at NeurIPS*, 2018
- M.-A. Charpagne*, F. Strub*, and T. M. Pollock. Accurate reconstruction of ebsd datasets by a multi-modal data approach using an evolutionary algorithm. *Materials Characterization*, 150:184–198, 2019

Workshop organization

- F. Strub, H. de Vries, A. Das, S. Kottur, S. Lee, M. Malinowski, O. Pietquin, D. Parikh, D. Batra, A. C. Courville, et al. Visually grounded interaction and language workshop (vigil). In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS), Workshop Schedule Highlights*, 2017a
- F. Strub, H. de Vries, E. Wijmans, S. Datta, E. Perez, S. Lee, P. Anderson, M. Malinowski, D. Batra, A. Courville, O. Pietquin, D. Parikh, J. Mary, C. Hori, A. Cherian, and T. Marks. Visually grounded interaction and language workshop (vigil). In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS), Workshop Schedule Highlights*, 2018a
- M. Pirotta, R. Fruit, M. Seurin, and F. Strub. European workshop of reinforcement learning (ewrl). In *Workshop Schedule Highlights*, 2018

Source code The full source code, experiment hyperparameters and neural checkpoints are available here: `https://github.com/GuessWhatGame`

Part I

Background and State-of-the-Art

No Ph.D. students have the same perspective while writing their manuscript. No readers have the same expectation while reading a Ph.D. thesis. In this cacophony, it may be worth to explicit my objectives to manage expectations. Thus, please let me introduce the four core directives I had while writing this state-of-the-art section.

First of all, I wanted the state-of-the-art section to be self-contained. My goal was to allow new Ph.D. students to quickly (and hopefully) build upon the work we have been doing over the past four years. I assumed that the reader has only some rudimentary machine learning knowledge, and I decided to define basic concepts before digging deeper into the literature. The experienced reader may thus easily skip the first sections to target more advanced topics directly.

Secondly, the state-of-the-art section aims to give a global overview of various topics, drawing a link between them. It does not aim at providing a complete overview of each subject with a comprehensive review of the underlying works. My research topic lead me to cover many themes, exploring deep learning, reinforcement learning, computer vision, natural language processing, multimodal learning and dialogue systems. It is impossible to cover all these topics in details without writing an extensively long state of the art section. As a result, I focused on the main component of each domain area, cherry-picking the tools and intuitions that I have been using along with my work. Thus, I did not write a survey on a specific subset of tasks and algorithms. However, I have always been pointing out the curious reader to complete studies or books when necessary.

I emphasized the historical and philosophical aspect of machine learning in each chapter. Although I have mostly been studying deep learning models, it is always healthy to remember the thought and motivation of our predecessors. First, it gives an understanding of the current state of the art. Then, it reminds us that most of our ideas are far from being new; in several cases, it occurs that we are barely rediscovering some old ideas by using new (but more powerful) tools. On the other hand, some "obvious" contemporary ideas (e.g., embodiment theory) result of decades of debate. Finally, checking old works help us to avoid the pitfall that equally bright people faced in the past.

The state-of-the-art section includes numerous references to cognitive science. Although it is imperative to not blindly copy human behaviour for designing machine learning models, it has been a constant source of inspiration for the development of AI. More generally, a change of paradigm in cognitive science has often led to a modification of machine learning methods in the upcoming decades. The resurgence of empiricism favoured the abandon of expert systems towards statistical approaches, and I believe that the same change is happening with the embodiment theory and the success of deep learning. Again, one must be extremely careful while extrapolating cognitive science conclusions to machine learning, yet, it would be a waste to not benefit from the reflection of the other communities.

I now hope that my writing directions are aligned with your reading expectations. More concretely, the first chapter covers statistical learning approaches including machine learning basics, deep learning, and reinforcement learning techniques. This section is mostly technical, and the experienced reader can easily skip it. The second chapter explores the computer vision and natural language processing before investigating *why* and *how* both research topics can benefit from each other. Each section starts with an historical perspective, followed by the study of deep learning models, and it ends with a discussion for future research directions supported by philosophical and cognitive arguments.

I now wish you a pleasant reading!

Chapter 1

Technical Background

«PRINT "Hello World!" END »

Language basics of BASIC language

Contents

A commonly accepted definition of machine learning was introduced by Mitchell (1997) as follows:

> "A computer program is said to learn from an experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Today, this definition mostly refers to statistical models which are updated by computing statistics from observed data (Bishop, 2006; Manning et al., 1999; Mitchell, 1997); it is worth noticing that it also includes other approaches such as dynamic expert systems (Hayes-Roth, 1985; Robinson, 1965) that update a set of rules from experience or case-based reasoning (Aamodt and Plaza, 1994) that were developed in the past. Statistical machine learning is broadly divided into three sub-tasks, where experience E is composed of input data and a potential extra signal. Supervised learning composed with input data with target labels to replicate, and it is briefly summarized in Sec. 1.1.1. Reinforcement learning maximizes (or minimizes) the sum of qualitative signals over a sequence of inputs data, and it is discussed in Sec. 1.3. Finally, unsupervised learning learns to directly extract patterns from the raw input data without explicit supervision, but it is not explored in this manuscript.

1.1 Supervised Learning

1.1.1 Definition

Supervised learning consists in learning a function that maps an input to a label from annotated data. Formally, given a dataset $\mathcal{D} = [\boldsymbol{x}_n, \boldsymbol{t}_n]_{n=1}^{N} = [\boldsymbol{X}, t]$ composed of N samples, where each sample is a tuple whose first element $\boldsymbol{x}_n \in \mathcal{X}$ is the input data, and the second element is the label/target output $\boldsymbol{t}_n \in \mathcal{T}$, the goal is to find the function $f^* : \mathcal{X} \rightarrow \mathcal{T}$ that minimizes the risk defined by the loss (or error) function $Loss : \mathcal{X} \times \mathcal{T} \rightarrow \mathbb{R}$ through an optimization procedure.

$$f^* = \arg\min_f \mathbb{E}[Loss(f(\boldsymbol{X}), \boldsymbol{t})], \tag{1.1}$$

$$= \arg\min_f \frac{1}{N} \sum_{n=1}^{N} Loss(f(\boldsymbol{x}_n), t_n). \tag{1.2}$$

When the target is qualitative (e.g., classifying image), it is a classification problem. When the objective is quantitative (e.g., predicting an income), it is a regression problem. A common regression loss is the Mean Square Error (MSE) where the goal is to find the function f that minimizes the quadratic error between the predicted output $f(\boldsymbol{x})$ and the target as follows:

$$Loss_{MSE} = (f(\boldsymbol{x}_n) - t_n)^2. \tag{1.3}$$

In a classification problem, f defines a probability distribution over C classes, and it is common to minimize the Cross-Entropy (CE) error:

$$Loss_{CE} = -\sum_{c=1}^{C} \mathbb{1}_{t=c}[\log(f_c(\boldsymbol{x}_n))]. \tag{1.4}$$

Until now, we have not specified any constraint on the function f. In practice, the model can either be parametric or non-parametric. Parametric functions assume that the function f is solely characterized by a finite set of parameters $\boldsymbol{\theta}$, independently of the size of the dataset, such as linear functions, neural networks or Gaussian processes, etc. (Bishop, 2006; Russell and Norvig, 2009). In parallel, non-parametric functions do not make strong assumptions about the form of the mapping function f such as kernel methods or trees, etc. (Bishop, 2006; Mitchell, 1997). In theory, non-parametric functions are more flexible but they may be slower to learn, and they generally scale with the number of data points.

1.1.2 Generalization and Training Procedure

The goal of statistical learning is to train a model to perform accurate prediction on new data. This objective is called generalization. In supervised learning, a model is trained on a training dataset before its generalization abilities are assessed on a test dataset with new samples. This protocol requires both datasets to be disjoint and drawn from a similar distribution (Bishop, 2006; Goodfellow et al., 2016; Mitchell, 1997).

At training time, the same loss is computed on the two datasets. Whenever the gap between the training loss and the testing loss becomes too important, it is a symptom of overfitting. It means that the model is over-specializing on the training data while losing its generalization abilities. Overfitting may arise because of lack of data, over-parametrization of a model, noisy labelling and other reasons. Several methods can be used to reduce over-fitting such as increasing the size of the dataset or using regularization methods as explored in Sec 1.2.3 for deep learning.

In many cases, overfitting cannot be entirely avoided; it is then crucial to stop the training process before the model is losing too much generalization abilities. One approach, called early stopping, consists in using a third validation dataset drawn from the initial data distribution. At training time, both the training and validation losses are computed. When the validation loss starts to increase, the training is stopped, and the test loss is calculated and reported. Therefore, it allows detecting overfitting symptoms to stop the training procedure while save-guarding the final evaluation protocol.

1.1.3 Linear Regression

Linear regression is the standard parametric method to perform supervised learning. Despite its simplicity, it is still a well-spread method as it both scales and performs well on real-world problems (Agarwal et al., 2014). A linear model is a linear combination of a fixed function of the input data. Formally, it is defined by a set of $M+1$ parameters $\boldsymbol{\theta} = [\theta]_{m=0}^{M}$, and a set of arbitrary basis functions $\boldsymbol{\phi} = [\phi]_{m=0}^{M}$ where $\phi_0(x) = 1$ such as:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \sum_{m=1}^{M} \theta_m \phi_m(\boldsymbol{x}) = \boldsymbol{\theta}^t \boldsymbol{\phi}(\boldsymbol{x}). \tag{1.5}$$

Classic basis functions include the identity function, polynomials function e.g. $\phi_m(x) = x^m$, spline functions e.g. $\phi_m(x) = e^{-\frac{(x-\nu_m)^2}{2\sigma_m}}$, where ν_m and σ_m are hyperparameters (Bishop, 2006). The linear model can easily be extended to match high dimensional multiple outputs $\boldsymbol{t}$ by writing the parameters $\boldsymbol{\theta}$ as a matrix $\boldsymbol{\Theta}$. Finally, θ_0 is often called the bias parameter as it does not depend on the input $\boldsymbol{x}$.

Then, a linear regression consists in finding the optimal set of parameters $\boldsymbol{\theta}^*$ such as:

$$f_{\boldsymbol{\theta}^*} = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} E(f_{\boldsymbol{\theta}}(\boldsymbol{x}_n), t_n). \tag{1.6}$$

When the dataset N is not too big, the optimal parameters $\boldsymbol{\theta}$ can be computed by using a least-square methods:

$$\boldsymbol{\theta}^* = \left(\boldsymbol{\phi}(\boldsymbol{X})^t \boldsymbol{\phi}(\boldsymbol{X})\right)^{-1} \boldsymbol{\phi}(\boldsymbol{X})^t \boldsymbol{t}. \tag{1.7}$$

When N increases, it becomes intractable to directly compute the inverted matrix $\phi(\boldsymbol{X})^t\phi(\boldsymbol{X}))^{-1}$, and other optimization techniques has been developed. For instance, Broyden-Fletcher-Goldfarb-Shanno (BFGS) (and its variants) iteratively approximates the inverted matrix by using gradient evaluations (Liu and Nocedal, 1989). Stochastic Gradient Descent (SGD) reduces the computational burden further by randomly drawing samples from the dataset $(\boldsymbol{x}, t) \sim \mathcal{D}$ to estimate update the model parameters until a convergence criterion is reached (Bottou, 2010). A learning rate η is defined to regulate the speed of convergence and the convergence is guaranteed if $\sum_{h=1}^{\infty} \eta = \infty$ and $\sum_{h=1}^{\infty} \eta^2 < \infty$.

$$\begin{aligned} \boldsymbol{\theta}_{h+1} &= \boldsymbol{\theta}_h - \eta \nabla Loss(f_{\boldsymbol{\theta}_h}(\boldsymbol{x}), t), && (1.8)\\ &= \boldsymbol{\theta}_h - \eta \nabla (\boldsymbol{\theta}_h{}^t \boldsymbol{\phi}(x) - t)^2, && (1.9)\\ &= \boldsymbol{\theta}_h - 2\eta \boldsymbol{\theta}_h (\boldsymbol{\theta_h}{}^t \boldsymbol{\phi}(\boldsymbol{x}) - t), && (1.10) \end{aligned}$$

where h is the training step. Finally, a linear regression can be turned into a logistic regression by turning the output into a probability distribution. It thus requires using the adequate training loss e.g. cross-entropy and to add a final non-linearity to the output e.g. $\sigma(x) = \frac{1}{1+e^x}$ for binary classification or $\mathrm{softmax}(\boldsymbol{x})_c = \frac{e^{x_c}}{\sum_{c'=1}^{C} e^{x_{c'}}}$ where $\boldsymbol{x} \in \mathbb{R}^C$ for multi-class classification.

1.1.4 Feed-Forward Networks

Unfortunately, linear (and logistic) regressions are inherently limited as they require the input data to be linearly separable tasks to perform well in classification tasks. However, raw input information (e.g., image pixels) can be too entangled to directly perform a linear combination to solve the task at hand. As a result, raw data must be pre-processed to extract key input features before training the algorithm. This operation may require careful engineering and domain expertise which severely limits the practical applicability of linear methods (LeCun et al., 2015).

A successful approach to alleviate this limitation is to allow the basis functions to be adaptive (Bishop et al., 1995; Goodfellow et al., 2016). In other words, the basis functions are learned at training time while trying to solve the task, allowing to disentangle the data without expert supervision. Following this intuition, Feed-forward Neural Networks or Multi-Layer Perceptron (MLP) consists in building a stack of K parametrized basis functions $[\boldsymbol{\phi}^k]_{k=1}^K$ where each basis function is itself a linear combination of the inputs followed by a non-linearity as defined as follows:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = g^K(\boldsymbol{\Theta}^K \boldsymbol{\phi}^K(\boldsymbol{x}) + \boldsymbol{b}^K), \tag{1.11}$$

where

$$\boldsymbol{\phi}^{(k)}(\boldsymbol{x}) = \sigma^{k-1}(\boldsymbol{\Theta}^{k-1}\boldsymbol{\phi}^{k-1}(\boldsymbol{x}) + \boldsymbol{b}^{k-1}) \qquad \text{and} \qquad \boldsymbol{\phi}^{(1)}(\boldsymbol{x}) = \boldsymbol{x}, \tag{1.12}$$

where $\boldsymbol{\Theta}^1, \ldots, \boldsymbol{\Theta}^K$ are the weight matrices, $\boldsymbol{b}^1, \ldots, \boldsymbol{b}^k$ are the bias vectors, and $g^{(k)}(.)$ are non-linear functions, also called activation functions. Classic hidden activation functions are the sigmoid, the hyperbolic tangent $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ and the Rectified Linear Unit (ReLU) $ReLU(x) = \max(0, x)$ (Nair and Hinton, 2010). A more intuitive formalization of neural networks is to define them as a stack of K layers where each layer $l^k(\boldsymbol{x})$ is the composition of an affine transform $\boldsymbol{a}^k = \boldsymbol{\Theta}^k \boldsymbol{x} + \boldsymbol{b}^k$ followed by a non-linearity σ^k:

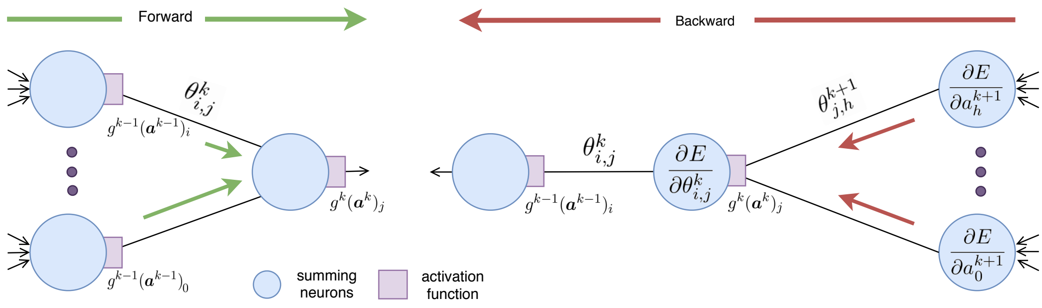


Figure 1.1: Illustration of the forward and backward pass in neural networks. In the forward pass, the neurons compute the weighed sum of the input. In the backward pass, the neurons compute the weighted sum of the backpropagated errors.

$$l^k(\boldsymbol{x}) = g^k(\boldsymbol{\Theta}^k \boldsymbol{x} + \boldsymbol{b}^k), \tag{1.13}$$

and the full stack of layers is the composition of each of the layers:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = l^K \circ l^{K-1} \cdots \circ l^1(\boldsymbol{x}), \tag{1.14}$$

where the $\circ$ is the function composition operator. Similar to linear regression, neural networks are trained with SGD (LeCun et al., 1998b),

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h - \eta \nabla E(f_{\boldsymbol{\theta}_h}(\boldsymbol{x}), \boldsymbol{t}), \tag{1.15}$$

where E is the error/loss function, $\boldsymbol{\theta}$ is the concatenation of all weight matrices and bias vectors of the MLP. However, the hierarchical nature of neural networks has two significant consequences. First, the training loss becomes non-convex; thus, SGD is not guaranteed to converge to the optimal solution. Second, the computation of the gradient is not straightforward and requires using the chain rule to be computed. This operation is also known as the error backpropagation in the neural network literature (Rumelhart et al., 1985) and it is illustrated in Fig 1.1. Backpropagation is decomposed into two passes: the forward pass consists in evaluating the network output from the input layers $l_1(.)$ to the uppermost layers $l_K(.)$ using Eq. 1.14; the backward pass computes the gradient error at each layer in a recurrent fashion, from the uppermost layer $l_K(.)$ to the input layer $l_1(.)$. Formally, given a stack of two layers parametrized by a weight matrix $\boldsymbol{\Theta}^k \in \mathbb{R}^{I \times J}$ and $\boldsymbol{\Theta}^{k+1} \in \mathbb{R}^{J \times H}$ (we include the bias into the matrices as in Eq. 1.5) and the inner neuron activation, $\boldsymbol{a}^k = \boldsymbol{\Theta}^k g^{k-1}(\boldsymbol{a}^{k-1})$, the backward pass estimates the gradient weight errors $\frac{\partial E}{\partial \theta^k_{i,j}}$ with the chain rule as follows:

$$\frac{\partial E}{\partial \theta^k_{i,j}} = \frac{\partial E}{\partial a^k_j} \frac{\partial a^k_j}{\partial \theta^k_{i,j}}, \tag{1.16}$$

where,

$$\frac{\partial a^k_j}{\partial \theta^k_{i,j}} = \frac{\partial (\boldsymbol{\Theta}^k g^{k-1}(\boldsymbol{a}^{k-1}))}{\partial \theta^k_{i,j}} = g^{k-1}(\boldsymbol{a}^{k-1})_i, \tag{1.17}$$

and,

$$\frac{\partial E}{\partial a_j^k} = \sum_{h=0}^{H} \frac{\partial E}{\partial a_h^{k+1}} \frac{\partial a_h^{k+1}}{\partial a_j^k}, \tag{1.18}$$

$$= \sum_{h=0}^{H} \frac{\partial E}{\partial a_h^{k+1}} \frac{\partial (\boldsymbol{\Theta}^{k+1} g^k(\boldsymbol{a}^k))}{\partial a_j^k}, \tag{1.19}$$

$$= \nabla g^k(\boldsymbol{a}^k)_j \sum_{h=0}^{H} \theta_{j,h}^{k+1} \frac{\partial E}{\partial a_h^{k+1}}. \tag{1.20}$$

We here observe the key recursion between $\frac{\partial E}{\partial a_j^k}$ and the activation error of the upper layer $\frac{\partial E}{\partial a_h^{k+1}}$, which is the actual backpropagated error in the backward pass. We compute the first recursion term $\frac{\partial E}{\partial a^K}$ by estimating the gradient of the loss function according to the final output. For instance, given the sample pair $(\boldsymbol{x}, t)$, and no final activation such as $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = a_K$, the MSE initializes the backpropagated error with $\frac{\partial E}{\partial a^K} = 2(a^K - t)$. Finally, we obtain the weight error by combining Eq. 1.17 and Eq. 1.20 as follows:

$$\frac{\partial E}{\partial \theta_{i,j}^k} = g^{k-1}(\boldsymbol{a}^{k-1})_i \nabla g^k(\boldsymbol{a}^k)_j \sum_{h=0}^{H} \theta_{j,h}^{k+1} \frac{\partial E}{\partial a_h^{k+1}}. \tag{1.21}$$

Note that equation Eq 1.21 encourages using the non-linear activation function whose gradients are easy to compute, e.g. $\nabla \tanh(\boldsymbol{x}) = 1 - \tanh^2(\boldsymbol{x})$. As stated before, the neural weights are updated with SGD according Eq. 1.15 once the error is computed. A classic trick is to accumulate the error over a mini-batch of samples to reduce the number of gradient updates (LeCun et al., 1998b).

From a theoretical point of view, neural networks are powerful but cursed models. Neural networks are universal approximator, which means that a sufficiently large MLP can approximate any function, given that it has enough capacity, i.e., that it has enough parameters (Cybenko, 1989; Hornik et al., 1989). However, there exists no optimization methods that guarantees to minimize the training loss of a neural network. In practice, SGD can be stuck in local minima, diverge, or the backpropagated error can be ill-distributed along the network, leading to vanishing/exploding gradients (Glorot and Bengio, 2010). In the next section, we explore the different methods that have been developed over the past twenty years to train neural networks successfully.

1.2 Deep Learning Background

In the previous section, we briefly introduced statistical learning basics, linear regression, and feed-forward neural networks. In conclusion, we observed that neural networks are powerful function approximators, but this ability comes at a cost: there is no training procedure that mathematically guarantees the convergence of a neural network to an optimal solution. On the other hand, recent empirical successes demonstrated the viability of neural networks to tackle large scale problems (Krizhevsky et al., 2012; LeCun et al., 1998a, 2015; Silver et al., 2017b; Sutskever et al., 2014; Tesauro, 1995; Van den Oord et al., 2016a; Vinyals et al., 2015b). Nowadays, the methods dealing with neural networks, e.g. optimizer, architecture, training tricks, are gathered under the name of Deep Learning (DL).

Several factors lead to the recent success of deep learning models. For instance, the development of fast Graphics Processing Units (GPUs) (Raina et al., 2009) alongside with their specialized

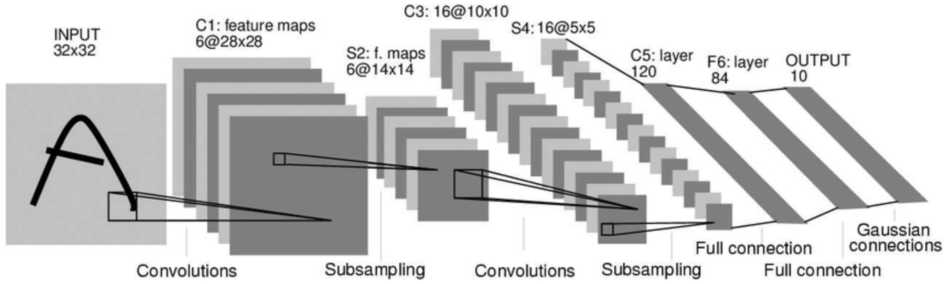


Figure 1.2: LeNet5 is a typical example of ConvNets relying on convolution layers. The figure is imported from the original LeNet5 paper (LeCun et al., 1998a)

drivers (Chetlur et al., 2014) sped-up the training of large neural networks by several order of magnitudes compared to classic Central Processing Units (CPUs). In parallel, the democratization of internet in the '00s granted access to a nearly infinite pool of data that was latter used to train deep networks (Lowe et al., 2015; Ordonez et al., 2011; Russakovsky et al., 2015). Finally, the development of auto-differential tools greatly alleviates the burden of implementing new complex neural architectures (Abadi et al., 2016; Bergstra et al., 2010; Collobert et al., 2002; Jia et al., 2014; Paszke et al., 2017). Consequently, deep learning somehow drifts away from pure statistical modelling to empirical research; instead of looking for convergence guarantees or error/sample bounds, deep learning research have developed new neural blocks to stack, improved training procedures or designed original experimental protocol to analyze the network behaviors. Unsurprisingly, this manuscript is anchored within this new empirical paradigm!

Although providing an extensive overview of deep learning methods is beyond the scope of this chapter, we explore here the deep learning tools that are used along with this thesis. For further details, we encourage the reader to prospect the deep learning book (Goodfellow et al., 2016), and to explore (Bishop et al., 1995) to better grasp the methodology evolution of neural networks within two decades. In the following, we first describe the fundamental deep learning architectural blocks. We then explore some deep learning training tools such as regularization or successful optimization approaches.

1.2.1 Spatial Layers

In their primary form, standard feed-forward networks are not well-suited to learn spatial representation. If we want to train a classifier on raw Red, Green and Blue (RGB) images, the input consists of an image $\mathcal{I}$ of width W, height H with C channels, e.g. RGB with $C = 3$. Thus, the input layer requires $C \times W \times H \times$ weights per output unit, which quickly becomes intractable, e.g. an image 64×64 with 128 hidden units requires more 1.5 million parameters. Besides, such networks only accept images with the same dimension. Finally, a standard MLP equally connects all the image pixels, ignoring that nearby pixels are more correlated than distant ones.

Convolution Layers Convolutional layers (LeCun et al., 1989, 1998a) were developed to extract patterns found within local regions of the input images. The convolutional layer generates feature maps by applying linear convolutional filters (also called kernels) followed by non-linear activation functions. Formally, given a spatial input $\boldsymbol{x}$ of dimension $C \times W \times H$, the output of a single convolution filter of size $K \times K'$ and parametrized by $(\boldsymbol{\theta}, b)$ is defined as follows:

$$f_{w,h} = \sigma\left(\sum_{k=0}^{K-1}\sum_{k'=0}^{K'-1} \boldsymbol{\theta}_{k,k'}{}^{t}\boldsymbol{x}_{w+k,h+k'} + b\right), \tag{1.22}$$

where w and h are spatial indices and σ is a non-linear activation function. After convoluting the input, the filter output $\boldsymbol{f}$ is of size $(W - K + 1) \times (H - K' + 1)$; yet a stride s can also be applied to reduce further the size of the output. The convolution window is then moved with a step of size s and $\boldsymbol{f}$ only contains the values $f_{s.w,s.h}$, which divides the feature map size by a coefficient s. Finally, we concatenate all the single convolution filters $\boldsymbol{f}$ into a single feature map tensor $\boldsymbol{F}$.

Convolution layers rely on three concepts that make them a valuable layer to deal with spatial inputs: it enforces a strong spatial inductive bias by only looking at local features, it is parameter efficient by using small shared convolutions filters, it does not fix the input size, it is translation invariant, and it is robust to small spatial transformation such as translation (Bishop et al., 1995; LeCun et al., 1998a)

Pooling Layers Pooling layers are non-trainable convolution layers with predefined operations, e.g. mean-pooling or max-pooling over the sliding windows. In practice, pooling layers often have a stride s to reduce the size of the spatial input, allowing cost-effective down scaling (Scherer et al., 2010).

Differently, global pooling layers aim to reduce the dimension of an input by making the convolution windows size embracing the full input space. For instance, a tensor of feature maps of size $W \times H \times C$ is turned into a vector of dimension C by applying a global mean-pooling over the spatial dimensions; as a result, each vector element is the mean values of the individual feature maps.

Convolution Neural Networks Convolution Neural Networks (CNNs), or ConvNets, stack several convolutional and pooling layers into a single pipeline to process spatial inputs. This neural architecture exploits the property that images have a compositional hierarchy, in which higher-level features are obtained by composing lower-level ones (LeCun et al., 2015). In its first version, CNN was the first neural architecture to have 99% accuracy on the MNIST (Lecun, 1998), where the models must classify images with digit numbers as illustrated Fig. 1.2 before being refined in subsequent work as explored in Sec 2.1.1. We also empirically observe that the bottom ConvNet layers do learn to detect low-level features such as colours, edges, etc. while upper layers are sensitive to more complex elements such as shapes, objects, etc. (Krizhevsky et al., 2012; Zeiler and Fergus, 2014; Zhou et al., 2015). However, CNN are not limited to images and have also been used to detect phonemes in speech (Waibel et al., 1990; Zhang et al., 2016b), perform natural text processing (Collobert and Weston, 2008; Zhang et al., 2015) or translation systems (Gehring et al., 2017).

1.2.2 Sequential Layers

MLP are also quickly limited while dealing with sequences of data. First, MLP have a predefined input size, which prevents them from dealing with sequences of undefined length. They are also insensitive to the ordering of the input by default, which is troublesome while dealing with series. Although sequential data can be preprocessed with handcrafted features to feed a MLP (Bishop et al., 1995), this approach quickly becomes limited and misses the original motivations of deep learning.

Recurrent Neural Networks RNNs are a family of neural architectures designed to efficiently process sequential data (Rumelhart et al., 1985). To do so, RNNs are based on a retro-action loop; the network handles the current input and the previous network output at each timestep in a recursive

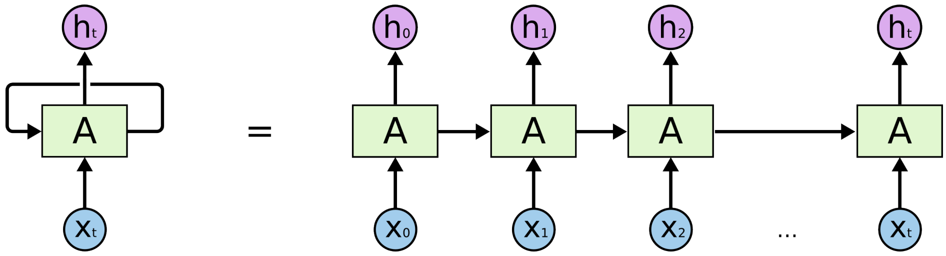


Figure 1.3: Sketch of a Recurrent Neural Network (RNN) unit (noted as A) with its recursive loop or when it is unrolled. The figure is imported from (Christopher, 2015)

fashion as shown in Fig 1.3. Formally, given a sequence of inputs $[\boldsymbol{x}_t]_{t=1}^T$, a transition function f parametrized by $\boldsymbol{\theta}$, a RNN produces a sequence $[\boldsymbol{h}_t]_{t=1}^T$ as follow:

$$\boldsymbol{h}_t = f_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}), \tag{1.23}$$

where $\boldsymbol{h}_0 = 0$. In its most basic form, RNN is a single MLP where the input is a concatenation of the $\boldsymbol{x}_t$ and the previous state $\boldsymbol{h}_{t-1}$. Such recursive structures allow RNNs to process arbitrary long chains while relying on a fixed number of parameters. Unfortunately, vanilla RNN are empirically hard to train and perform poorly on long sequences (Bengio et al., 1994; Pascanu et al., 2013).

Long Short-Term Memory Long Short-Term Memory (LSTM) networks are RNNs specifically designed to correctly process long and short term dependencies in sequential inputs (Hochreiter and Schmidhuber, 1997). The LSTM is composed of two internal memory cells, a context cell $\boldsymbol{c}_t$ and an output cell $\boldsymbol{h}_t$, that are updated thought a gating mechanism to either memorize, update and forget information from a stream of data as shown in Fig. 1.4a. Formally, a LSTM is defined by the following equations:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_i) \qquad \text{Compute the input gate,} \tag{1.24}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_f) \qquad \text{Compute the forget gate,} \tag{1.25}$$

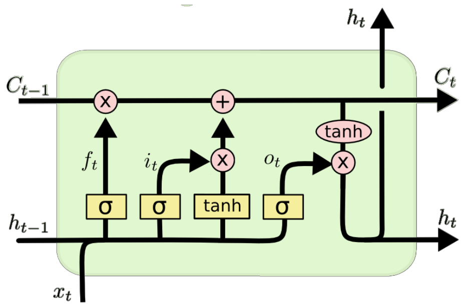
$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_o) \qquad \text{Compute the output gate,} \tag{1.26}$$

$$\boldsymbol{c}_t = \boldsymbol{i}_t \odot \tanh(\boldsymbol{W}_c[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_c) + \boldsymbol{f}_t \odot c_{t-1} \qquad \text{Update the context cell,} \tag{1.27}$$

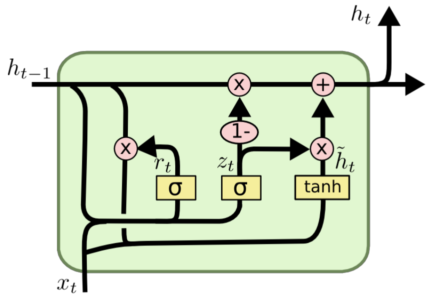
$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \qquad \text{Update the output cell,} \tag{1.28}$$

where $\boldsymbol{W}_i, \boldsymbol{W}_f, \boldsymbol{W}_o, \boldsymbol{W}_c$ are weight matrices, $\boldsymbol{b}_i, \boldsymbol{b}_f, \boldsymbol{b}_o, \boldsymbol{b}_c$ are bias vectors, $[.;.]$ is the concatenation operator, $\odot$ is the Hadamard product (or element-wise multiplication), and initial states are $\boldsymbol{h}_0 = 0$ and $\boldsymbol{c}_0 = 0$. LSTM has successfully been applied to a large variety of language tasks (Bahdanau et al., 2015; Graves, 2013; Sordoni et al., 2015; Sundermeyer et al., 2012), speech recognition (Graves et al., 2013) or even image generation (Gregor et al., 2015).

Gated-Recurrent Units Another successful and light weighted RNN architecture is the Gated-Recurrent Unit (GRU) (Chung et al., 2014) illustrated in Fig. 1.4b. GRUs are also designed to handle long and short term dependencies, while aiming to be simpler and easier to train than historical LSTMs. The recurrent network composes with the two gates and a single memory cell, halving the number of parameters. Formally, a GRU is defined by the following equations:



(a) Sketch of a LSTM



(b) Sketch of a GRU

Figure 1.4: Sketch of different RNN architectures. The figures are imported from (Christopher, 2015)

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_z) \quad \text{update the gate vector,} \tag{1.29}$$

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_r) \quad \text{Compute the reset gate,} \tag{1.30}$$

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_h[\boldsymbol{x}_t; \boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}] + \boldsymbol{b}_h) \quad \text{Update the output cell (1),} \tag{1.31}$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t \quad \text{Update the output cell (2),} \tag{1.32}$$

where $\boldsymbol{W}_z, \boldsymbol{W}_r, \boldsymbol{W}_h$ are weight matrices, $\boldsymbol{b}_z, \boldsymbol{b}_r, \boldsymbol{b}_h$ are bias vectors. Other LSTM and GRU variants have been developed to either improve RNN abilities (Dey and Salemt, 2017; Gers and Schmidhuber, 2001) or incorporate inductive bias such as hierarchical structures (Tai et al., 2015) or recurrent convolution layers (Cooijmans et al., 2017; Xingjian et al., 2015).

Bi-directionnal Recurrent Networks In the previous paragraph, we implicitly assumed that RNNs are fed with a stream of inputs $[\boldsymbol{x}_t]_{t=1}^T$ whereas we sometimes have access to the full sequences of data. In this scenario, one idea is to reverse the input stream to obtain an additional representation of the sequence. Thus, Bidirectional Recurrent Networks (Bi-RNN) combine two RNNs: one forward RNN, producing hidden states $\overrightarrow{\boldsymbol{h}}_t$ by running from $\boldsymbol{x}_1$ to $\boldsymbol{x}_T$, and a second backward RNN, producing states $\overleftarrow{\boldsymbol{h}}_t$ by running from $\boldsymbol{x}_T$ to $\boldsymbol{x}_1$. We then concatenate both unidirectional RNN states $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t; \overleftarrow{\boldsymbol{h}}_t]$ at each step t to have the final output.

1.2.3 Regularization

Deep learning architectures often require dozens of millions of parameters, e.g. Krizhevsky et al. (2012) implemented a large scale CNN with 60M parameters and Sutskever et al. (2014) used a stack of LSTMs leading to 380M weights. While it allows a network to approximate a large variety of functions, this overparametrization may also create severe overfitting problems. For instance, modern CNN architectures can perfectly classify 1.2 million images with random labelling (Zhang et al., 2017). Regularization methods, which aims at reducing the generalization gap, are de-facto crucial tools to train large neural networks effectively. In the following, we explore the regularization methods that have been used along with this thesis.

Parameter Norm Penalty In machine learning, a common overfitting symptom is to observe parameters with very high values (Bishop, 2006). Thus, a common regularization technique consists in adding a parameter norm penalty $\Omega(.)$ in the training loss. Formally, given a parametric function $f_{\boldsymbol{\theta}}(\boldsymbol{x})$, the regularized training loss is defined by:

(a) (b)

Figure 1.5: Standard MLP before (a), after (b) applying dropout. The figures are imported from (Srivastava et al., 2014).

$$Loss_{reg}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), t) = Loss(f_{\boldsymbol{\theta}_h}(\boldsymbol{x}), t) + \lambda\Omega(\boldsymbol{\theta}), \tag{1.33}$$

where $\lambda \in \mathbb{R}^+$ is a positive hyperparameter controlling the norm penalty term. For instance, the L^2 norm penalty $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{\theta}||_2^2$ prevents parameters to diverge as it would increase the training loss. This regularization is called weight decay in the neural network literature although, this equivalence seems incorrect when using adaptive gradient methods (Loshchilov and Hutter, 2019). Differently, the L^1 norm penalty $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_1 = \sum_i |w|_i$ sparsifies the model parameters (Tibshirani, 1996), performing feature selection and may mitigate potential over-parametrization effect.

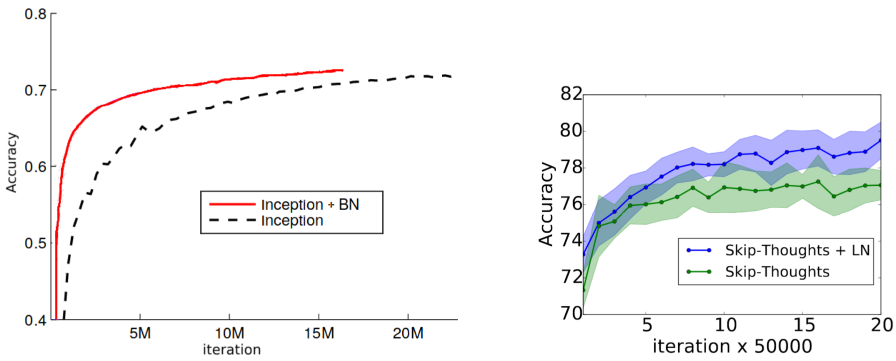
Ensemble Methods and Dropout Dietterich (2000) detailed how an ensemble of models can be combined to improve the final prediction and compensate individual overfitting issues. The expected generalization error of the ensemble performs then at least as well as any of its members. For instance, Bootstrap aggregating (Bagging) trains the same model with different training splits, altering the dataset by copying and removing random samples (Breiman, 1996). In deep learning, trained models are likely to differ because of random weight initialization or SGD. Thus, ensemble methods are often well-suited to reduce the generalization gap (Goodfellow et al., 2016).

Dropout is a stochastic regularization procedure which artificially simulates an ensemble of networks (Srivastava et al., 2014). Dropout layers apply a random activation mask over the neurons at each training step, producing new thin network on the fly as shown in Fig 1.5. Formally, the activation mask $\boldsymbol{m}$ is sampled from a Bernoulli distribution parametrized by p such as:

$$Dropout(\boldsymbol{x}) = \begin{cases} \boldsymbol{m} \odot \boldsymbol{x} \text{ where } \boldsymbol{m} \sim Bernouilli(p) & \text{at training time} \\ \frac{1}{p}\boldsymbol{x} & \text{at evaluation time.} \end{cases} \tag{1.34}$$

Another intuition behind dropout is to prevent overfitting by forcing neurons to be redundant rather than relying on the activity of specific units. Dropout was among the first methods which allows to train large CNN networks (Krizhevsky et al., 2012). When carefully applied, dropout can also regularize stacks of RNNs (Pham et al., 2014; Zaremba et al., 2015) and RNN hidden states (Gal and Ghahramani, 2016).

Batch-Normalization Batch Normalization (BN) is a deep learning technique that was initially designed to accelerate the training of neural networks by reducing the internal co-variate shift (Ioffe and Szegedy, 2015). Succinctly, the internal co-variate shift refers to the change in the distribution of layer inputs caused by updating the preceding layers at training time (Shimodaira, 2000), negatively



(a) (b)

Figure 1.6: (a) Advanced ConvNet (Inception Szegedy et al. (2015)) trained on ImageNet (Russakovsky et al., 2015), with/out BN. The figure comes from (Szegedy et al., 2015) and was manually edited to remove other baselines. (b) Advanced RNN network (Skip-Thought Vectors (Kiros et al., 2015)) trained on movie review sentiment (Pang and Lee, 2005), with/out LN. The figure comes from (Ba et al., 2016) and was manually edited to remove other baselines.

impacting the training. BN aims at reducing this distribution shift by normalizing and centring neural activation at each input according to the mini-batch statistics. Formally, given a mini-batch of features $\mathcal{B} = [\boldsymbol{x}_n]_{n=1}^N$ of N samples where $\boldsymbol{x}$ is an arbitrary input with C features (or channels) of unknown dimension, BN normalizes the activations at training time as follows:

$$BN(\boldsymbol{x}_{n,c,\cdot}) = \gamma_c \frac{\boldsymbol{x}_{n,c,\cdot} - \mathbb{E}_{\mathcal{B}}[\boldsymbol{x}_{\cdot,c,\cdot}]}{\sqrt{\text{Var}_{\mathcal{B}}[\boldsymbol{x}_{\cdot,c,\cdot}] + \epsilon}} + \beta_c, \tag{1.35}$$

where ϵ is a constant damping factor for numerical stability, and γ_c and β_c are trainable scalars introduced to keep the representational power of the original network. At inference time, the batch mean $\text{E}_{\mathcal{B}}$ and variance $\text{Var}_{\mathcal{B}}$ are replaced by the population mean μ and variance σ^2, often estimated by an exponential moving average over the batch at training time. In computer vision, $\boldsymbol{x}$ is often a tensor of feature maps $\boldsymbol{F}$ of size $C \times W \times H$ where the mean and variance are computed over both the batch and spatial dimensions, such that each location in the feature map is normalized in a similar way.

BN eventually seems to not affect the co-variate shift issue (Santurkar et al., 2018), but it unarguably speeds up the training of deep CNN and seems to have a regularization effect by improving generalization (Ioffe and Szegedy, 2015; Luo et al., 2019) as highlighted in Fig 1.6a. There were also some attempts to apply BN to RNN, but it resulted in mitigated successes (Amodei et al., 2016; Cooijmans et al., 2017; Laurent et al., 2016).

Layer-Normalization Layer Normalization (LN) is an adaptation of BN that was successfully applied to RNN (Ba et al., 2016). Like BN, the neural activations are rescaled with two $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ parameters. However, LN replaces the mini-batch statistics and the moving average and variance by the global mean and variance of the input for each single sample, making the regularization independent of the batch size. Formally, the LN is defined as follows:

$$\tilde{\boldsymbol{h}}_{n,t,\cdot} = \frac{\boldsymbol{\gamma}}{\text{Var}[\boldsymbol{x}_{n,t,\cdot}]} \odot (\boldsymbol{x}_{n,t,\cdot} - \mathbb{E}[\boldsymbol{x}_{n,t,\cdot}]) + \boldsymbol{\beta}. \tag{1.36}$$

When dealing with RNNs, LN rescales the hidden state $\boldsymbol{h}_{n,t,.}$ with the non-linear activation function $g(.)$ as follow $\tilde{\boldsymbol{h}}_{n,t} = g(LN(\boldsymbol{h_{n,t,c}}))$. In addition, it also normalizes the backpropagated gradient along the time dimension, reducing potential vanishing and exploding gradient in the RNN (Ba et al., 2016). In the end, it is generally recommended to use LN with RNN, and BN with CNN.

1.2.4 Optimization for Deep Learning

Training neural networks requires to minimize a non-linear and non-convex loss function, loosing convergence guarantees as explained in Sec 1.1.4. At training time, deep networks get stuck in poor local minima or plateaus (Bengio et al., 2007). "Getting a neural network to work well, or to work at all, was more of an art than a science" (LeCun et al., 1998b). Once Krizhevsky et al. (2012) demonstrated the potential of deep networks, there was a surge to adapt optimization methods to reduce the neural training difficulty (Goodfellow et al., 2016). In the following, we explore some common practices that have been used during this thesis.

Network Initialization As everyone knows, the best way to initialize a model is to start from the optimal solution. Unfortunately, it greatly reduces the fun of machine learning, and it is a bit tiresome to manually type a few millions of parameters for every new neural network. A compromise was found by designing initialization heuristics, but we have to be careful as starting from a poor parametrization may have catastrophic consequences while optimizing non-convex losses with SGD.

In practice, neural weights are sampled according to distributions conditioned on the activation function (sigmoid, $\tanh$, ReLU, etc.) and/or the neural architecture (e.g., MLP, CNN, RNN). As a rule of thumb, weight initialization is defined such that each layer inputs has a zero mean and a standard deviation close of one (LeCun et al., 1998b). It prevents non-linear activations, e.g. $\tanh$ or sigmoid, to saturate and alleviate gradient vanishing/explosion at the beginning of the training (Glorot and Bengio, 2010). Formally, given a MLP with n_{in} inputs and n_{out} outputs parametrized by Θ, a classic initialization is defined as follows (Glorot and Bengio, 2010):

$$\theta \sim U\Big[-4\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}; -4\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}\Big] \quad \text{with sigmoid activations,} \tag{1.37}$$

$$\theta \sim U\Big[-\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}; -\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}\Big] \quad \text{with tanh activations,} \tag{1.38}$$

$$\tag{1.39}$$

where $U[a;b]$ is a uniform distribution between a and b. Later, He et al. (2015) derived the following initialization procedure for CNN and ReLU activation by specifically addressing the ReLU non-linearity:

$$\theta \sim \mathcal{N}\Big(0, \frac{\sqrt{2}}{\sqrt{n_{in}}}\Big) \quad \text{with ReLU activations,} \tag{1.40}$$

where $\mathcal{N}(a,b)$ is a Gaussian distribution of mean a and standard deviation b. Note that initialization is sometimes referred to as a regularization procedure as it can help the final generalization. Another common pre-initialization approach is to pretrained the network on an auxiliary task (e.g., unsupervised training) and to reuse a subset of the weights in another final tasks (Bengio et al., 2007; Erhan et al., 2009), as further explored in Sec 2.1.3.

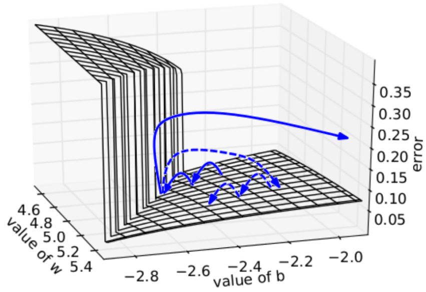


Figure 1.7: Plot of an error surface of a single hidden unit recurrent network highlighting the existence of high curvature walls. The solid lines depict standard trajectories that gradient descent might follow. Using the dashed arrow, the diagram shows what would happen if the gradients are rescaled to a fixed size when its norm is above a threshold. The figure and caption are extracted from Pascanu et al. (2013).

Adam Optimizer As discussed in the previous section, neural networks weights were historically updated by using vanilla SGD $\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h - \eta \nabla E(f_{\boldsymbol{\theta}_h}(\boldsymbol{x}), t)$ where the learning rate η conditions the update size. However, this hyperparameter often requires careful finetuning at training time by manually dismissing its value when the validation error plateaus (LeCun et al., 1998a), or by including a learning decay (Bishop et al., 1995) . Besides, SGD is notoriously known to be slowly converge on large networks (Ruder, 2016).

Adaptive learning rates were thus proposed to automatically increase or decrease the learning rate for each parameter based on individual weight errors (Duchi et al., 2011; Kingma and Ba, 2015; Tieleman and Hinton, 2012; Zeiler, 2012). Among the (very) vast literature of optimization methods, Adaptive Moment estimation (Adam) (Kingma and Ba, 2015) have empirically been very successful on a large variety of tasks although having some inherent theoretical limitations (Loshchilov and Hutter, 2019; Reddi et al., 2018).

Following (Tieleman and Hinton, 2012; Zeiler, 2012), Adam uses exponentially decaying average of both past gradients $\boldsymbol{m}$ and past squared gradients $\boldsymbol{v}$ to compute parameter-wise learning rates. Formally, given an individual gradient weight $g_{h,i} = \nabla E(f_{\boldsymbol{\theta}_h}(\boldsymbol{x}), \boldsymbol{t})|_{\theta_{h,i}}$, the adaptive gradient update rule is defined as follows:

$$\boldsymbol{m}_h = \beta_1 \boldsymbol{m}_{h-1} + (1-\beta_1)\boldsymbol{g}_h \quad \text{Compute gradient first order moment,} \tag{1.41}$$

$$\boldsymbol{v}_h = \beta_2 \boldsymbol{v}_{h-1} + (1-\beta_2)\boldsymbol{g}_h^2 \quad \text{Compute gradient second order moment,} \tag{1.42}$$

$$\tilde{\boldsymbol{m}}_h = \frac{\boldsymbol{m}_h}{1-\beta_1^t} \quad \text{Debias gradient first-order moment,} \tag{1.43}$$

$$\tilde{\boldsymbol{v}}_h = \frac{\boldsymbol{v}_h}{1-\beta_2^t} \quad \text{Debias gradient second-order moment,} \tag{1.44}$$

$$\theta_{h+1,i} = \theta_{h,i} - \frac{\eta}{\sqrt{\tilde{v}_{h,i}+\epsilon}} \tilde{m}_{h,i} \quad \text{Perform gradient descent,} \tag{1.45}$$

$$\tag{1.46}$$

where η is the initial learning rate, β_1 and β_2 are respectively decay rates for first-order and second-order moments of the gradient and ϵ is a constant damping factor for numerical stability. They are generally set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$ following the author recommendation (Kingma and Ba, 2015).

Gradient Clipping Exploding gradients are a major issue while training RNN (Bengio et al., 1994). A simple and efficient way to alleviate this issue is to rescale the gradient when its norm reaches a predefined threshold (Pascanu et al., 2013). Formally, given some arbitrary gradient $\boldsymbol{g}$ and a threshold δ, the new clipped gradient $\tilde{\boldsymbol{g}}$ is defined as follow:

$$\tilde{\boldsymbol{g}} = \frac{\min(\delta, ||\boldsymbol{g}||_2)}{||\boldsymbol{g}||_2}\boldsymbol{g}. \tag{1.47}$$

As highlighted in Fig 1.7, a geometrical intuition of clipping is to prevent the gradient from abruptly updating the weight while dealing with error surface with high curvature.

1.3 Reinforcement Learning Background

Reinforcement Learning (RL) is a training procedure to teach an agent to interact within an environment by maximizing the (discounted) sum of reward signals along its course. Hence, it belongs to the family of sequential decision-making methods. While supervised learning aims at imitating expert labeling, RL methods collect positive and negative signals within an environment to update the agent behavior. Reinforcement learning originates from research in animal psychology with the seminal work of Thorndike (1898) and Skinner (1938). The authors studied how animals could be conditioned through trial-and-error experiments. The goal was to enforce a specific behavior by either giving positive or negative stimuli to the animals. This training intuition was later cast as a mathematical problem with dynamic programming (Bellman, 1957; Bertsekas and Tsitsiklis, 1996; Puterman, 2014), and reinforcement learning (Sutton and Barto, 1998)

In the following subsections, we first define the mathematical framework of reinforcement learning before exploring the actual training algorithms. Finally, we examine how deep learning and reinforcement learning can be interconnected to create deep reinforcement learning algorithms. For more details on reinforcement learning methods, we invite the reader to look at Sutton and Barto (1998)

1.3.1 Markov Decision Process

A reinforcement learning process can be described as follows: at each time t, the agent selects an action u_t based on this current state $\boldsymbol{x}_t$, to which the environment responds with a reward r_{t+1} and the agent moves to a new state $\boldsymbol{x}_{t+1}$ as highlighted in Fig 1.8. Formally, the environment can be modelled as an Markov Decision Process (MDP) (Bellman, 1957; Bertsekas and Tsitsiklis, 1996; Howard, 1960; Puterman, 2014), where a MDP [1] is a 5-tuple $(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{R}, \boldsymbol{P}, \gamma)$ whose elements are:

- $\boldsymbol{X}$ is a finite set of states $\boldsymbol{x}$
- $\boldsymbol{U}$ is a finite set of actions u_t that can be done in state $\boldsymbol{x}$ at time t
- $\boldsymbol{R} : \boldsymbol{X} \times \boldsymbol{U} \rightarrow \mathbb{R}$ is the reward function returning the reward r_t for taking action u_t in state $\boldsymbol{x}_t$
- $\boldsymbol{P} : \boldsymbol{X} \times \boldsymbol{U} \times \boldsymbol{X} \rightarrow [0; 1]$ is the transition kernel where $P(\boldsymbol{x}' = \boldsymbol{x}_{t+1} | \boldsymbol{x} = \boldsymbol{x}_t, u = u_t)$ is the probability of reaching state $\boldsymbol{x}'_t$ if the agent takes the action u_t in state $\boldsymbol{x}_t$
- $\gamma \in [0; 1[$ is the discount factor that encodes how to weight long-term rewards.

The MDP is said to be episodic if it is guaranteed to finish after a finite number of steps. In such cases, the discount factor γ can be equal to one. Differently, a MDP is said to be non-stationary (vs.

[1] We here use the control notation to prevent some confusion with the GuessWhat?! notation defined later in the thesis

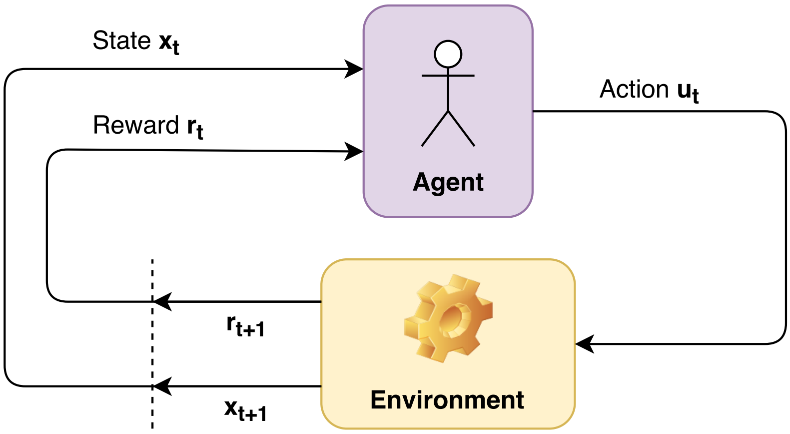


Figure 1.8: Sketch of a Markov decision process.

stationary) if the transition kernel (or any other element of the MDP) depends on the timestep t. In the following, we assume that we are dealing with stationary and episodic environments.

The agent is modelled by a policy $\boldsymbol{\pi} : \boldsymbol{X} \rightarrow \boldsymbol{U}$ which maps a probability distribution over the actions $\boldsymbol{u}$ for every state $\boldsymbol{x}$. The policy is said to be deterministic if for all states $\boldsymbol{x}$, there is a single action u with probability one; otherwise, the policy is said to be stochastic. Similar to MDPs, a policy is said to be non-stationary if it depends on the timestep t.

In the following subsections, we explore different methods to evaluate and optimize (or control) the policy $\boldsymbol{\pi}$. Some of those methods assume to have a full knowledge of the MDP, i.e. the transition kernel $P(\boldsymbol{x}, u, \boldsymbol{x}')$ is explicitly defined; they are referred as Dynamic Programming algorithms (Bellman, 1957; Bertsekas and Tsitsiklis, 1996; Howard, 1960; Puterman, 2014). Differently, algorithms that do not have access to the exact transition kernel and sample the transition kernel by interacting with the environment, are referred to as reinforcement learning methods (Sutton and Barto, 1998). We first analyze both approaches indifferently as they rely on the same core equations before focusing on reinforcement learning algorithms.

1.3.2 Evaluation with State-Value Functions

In RL and Dynamic Programming (DP), the ultimate goal is to find the policy $\boldsymbol{\pi}^*$ that maximizes the cumulative expected rewards $\mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R(\boldsymbol{x}_k, u)]$ for every $\boldsymbol{x}$ where $u \sim \boldsymbol{\pi}$. As a first step, we need to estimate the quality of a policy in order to improve it. We therefore define the value of a state $\boldsymbol{x}$ under a policy $\boldsymbol{\pi}$ to be the expected return of the trajectories generated by the policy when starting from state $\boldsymbol{x}$. Formally, the state-value function for policy $\boldsymbol{\pi}$ can be written as follows:

$$V^\pi(\boldsymbol{x}) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R(\boldsymbol{x}_{t+k}, u_{t+k}) \,\middle|\, \boldsymbol{x}_t = \boldsymbol{x} \right] \qquad \forall \boldsymbol{x} \in \boldsymbol{X}. \tag{1.48}$$

The state-value function also satisfies the Bellman equation as follows (Bellman, 1957):

$$V^{\pi}(\boldsymbol{x}) = \mathbb{E}_{\pi}\Big[\sum_{k=0}^{\infty}\gamma^{k}R(\boldsymbol{x}_{t+k}, u_{t+k})\ \Big|\ \boldsymbol{x}_t = \boldsymbol{x}\Big], \tag{1.49}$$

$$= \mathbb{E}_{\pi}\Big[R(\boldsymbol{x}_t, u_t) + \sum_{k=1}^{\infty}\gamma^{k}R(\boldsymbol{x}_{t+k}, u_{t+k})\ \Big|\ \boldsymbol{x}_t = \boldsymbol{x}\Big], \tag{1.50}$$

$$= \mathbb{E}_{\pi}\Big[R(\boldsymbol{x}_t, u_t) + \gamma\sum_{k=0}^{\infty}\gamma^{k}R(\boldsymbol{x}_{t+k+1}, u_{t+k+1})\ \Big|\ \boldsymbol{x}_t = \boldsymbol{x}\Big], \tag{1.51}$$

$$= \mathbb{E}_{\pi}\Big[R(\boldsymbol{x}_t, u_t) + \gamma V^{\pi}(\boldsymbol{x}_{t+1})\ \Big|\ \boldsymbol{x}_t = \boldsymbol{x}\Big]. \tag{1.52}$$

If we estimate the expectation by following the policy $\boldsymbol{\pi}$ given the transition kernel P, we obtain:

$$V^{\pi}(\boldsymbol{x}) = \sum_{u\in U}\pi(\boldsymbol{x}, u)\sum_{x'\in X}P(\boldsymbol{x}, u, \boldsymbol{x}')\big(R(\boldsymbol{x}, u) + \gamma V^{\pi}(\boldsymbol{x}_{t+1})\big). \tag{1.53}$$

The Bellman relation (Eq. 1.53) highlights the link between the value of a state and the values of the other states; this relation is the cornerstone of RL and DP algorithms. For instance, it guarantees the existence of a single state-value function for a policy. In dynamic programming, if we respectively define $\boldsymbol{P}^{\pi}$ and $\boldsymbol{R}^{\pi}$ as the transition kernel and reward function reweighted by the policy distribution $\boldsymbol{\pi}$ and $\boldsymbol{V}^{\pi}$ the value-state function column vector, the state-value function can be computed by either recursively applying the Bellman operation to the value state $\boldsymbol{V}^{\pi}_{k+1} = \boldsymbol{R}^{\pi} + \gamma\boldsymbol{P}^{\pi}\boldsymbol{V}^{\pi}_{k}$ or using linear algebra $\boldsymbol{V}^{\pi} = (\mathcal{I} - \gamma\boldsymbol{P}^{\pi})^{-1}\boldsymbol{R}^{\pi}$ where $\mathcal{I}$ is the identity matrix (Bertsekas and Tsitsiklis, 1996; Puterman, 2014). In reinforcement learning, we first sample a batch of trajectories $\tau = [(\boldsymbol{x}_t, u_t, r_{t+1}, \boldsymbol{x}_{t+1})]_{t=1}^{T}$ by following the policy $\boldsymbol{\pi}$ before estimating the value-state function with Monte-Carlo methods or temporal difference $V^{\pi}_{k+1}(\boldsymbol{x}_t) = V^{\pi}_{k}(\boldsymbol{x}_t) + \eta(r_{t+1} + \gamma V^{\pi}_{k}(\boldsymbol{x}_{t+1}) - V^{\pi}_{k}(\boldsymbol{x}_t))$ where η is a learning rate (Sutton, 1988; Sutton and Barto, 1998).

As a second step, we can define an ordering of the policies by using the state-value function. A policy $\boldsymbol{\pi}$ is said to be better than another policy $\boldsymbol{\pi}'$ if its state-value function $\boldsymbol{V}^{\pi}$ is greater than the other state-value function $\boldsymbol{V}^{\pi'}$ for every state $\boldsymbol{x} \in \boldsymbol{X}$. Formally,

$$\boldsymbol{\pi} \geq \boldsymbol{\pi}' \quad \Leftrightarrow \quad V^{\pi}(\boldsymbol{x}) \geq V^{\pi'}(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \boldsymbol{X}. \tag{1.54}$$

It exists a unique optimal state-value function $\boldsymbol{V}^{*}$ that is greater that all the others $\boldsymbol{V}^{\pi}$ for a given MDP. Therefore, all policies $\boldsymbol{\pi}^{*}$ that achieve the same state-value $\boldsymbol{V}^{\pi^{*}} = \boldsymbol{V}^{*}$ (Puterman, 2014) are said to be optimal.

1.3.3 Control with Action-Value Functions

While the state-value function gives a clear definition of optimal policies, it does not directly provide the optimal policy itself. Thus, we define the action-value function Q^{π} (or Q-value function) which encodes the expected return of taking the action u in state $\boldsymbol{x}$ while following $\boldsymbol{\pi}$. Intuitively, it gives a direction about which action should we take in each state if we want to maximize our reward. It is defined as follow:

$$Q^{\pi}(\boldsymbol{x}, u) = \mathbb{E}_{\pi}\Big[\sum_{k=0}^{\infty}\gamma^{k}R(\boldsymbol{x}_{t+k}, u_{t+k})\ \Big|\ \boldsymbol{x}_t = \boldsymbol{x}, u_t = u\Big] \qquad \forall \boldsymbol{x} \in \boldsymbol{X}, \forall u \in \boldsymbol{U}. \tag{1.55}$$

The state-value function and action-value function are linked with the following equation:

$$V^{\pi}(\boldsymbol{x}) = \sum_{u \in \boldsymbol{U}} \boldsymbol{\pi}(\boldsymbol{x}, u) Q^{\pi}(\boldsymbol{x}, u). \tag{1.56}$$

Thus, the Q-value function also follows the Bellman equation such as:

$$\begin{aligned} Q^{\pi}(\boldsymbol{x}, u) &= \mathbb{E}_{\boldsymbol{\pi}}\left[R(\boldsymbol{x}_t, u_t) + \gamma Q^{\pi}(\boldsymbol{x}_{t+1}, u_{t+1}) \,\middle|\, \boldsymbol{x}_t = \boldsymbol{x}, , u_t = u\right], && (1.57) \\ &= R(\boldsymbol{x}, u) + \gamma \sum_{\boldsymbol{x}' \in \boldsymbol{X}} P(\boldsymbol{x}, u, \boldsymbol{x}') \sum_{u' \in \boldsymbol{U}} \boldsymbol{\pi}(\boldsymbol{x}', u') Q^{\pi}(\boldsymbol{x}', u'). && (1.58) \end{aligned}$$

Similarly, the action-value function can also be used to define an ordering:

$$\boldsymbol{\pi} \geq \boldsymbol{\pi}' \quad \Leftrightarrow \quad Q^{\pi}(\boldsymbol{x}, u) \geq Q^{\pi'}(\boldsymbol{x}, u) \quad \forall \boldsymbol{x} \in \boldsymbol{X}, u \in \boldsymbol{U}, \tag{1.59}$$

and it exists a unique optimal action-value function $\boldsymbol{Q}^*$ that is greater than action-value functions from all other policies. Besides, all the optimal policies share the same optimal action-value function $\boldsymbol{Q}^{\pi^*} = \boldsymbol{Q}^*$. Finally, there is always a deterministic optimal policy in the episodic case that can be defined by maximizing $\boldsymbol{\pi}^*$ over $\boldsymbol{Q}^*$:

$$\boldsymbol{\pi}^*(u|\boldsymbol{x}) = \begin{cases} 1 & \text{if } u = \arg\max_{u' \in \boldsymbol{U}} Q^*(\boldsymbol{x}, u') \\ 0 & \text{otherwise.} \end{cases} . \tag{1.60}$$

Finding an optimal policy therefore requires computing the maximal action-value function $\boldsymbol{Q}^*$, which follows the Bellman optimality equation (Bellman, 1957):

$$\begin{aligned} Q^*(\boldsymbol{x}, u) &= \mathbb{E}_{\boldsymbol{\pi}}\left[R(\boldsymbol{x}_t, u_t) + \gamma \max_{u' \in U} Q^{\pi}(\boldsymbol{x}_{t+1}, u') \,\middle|\, \boldsymbol{x}_t = \boldsymbol{x}, , u_t = u\right], && (1.61) \\ &= R(\boldsymbol{x}, u) + \gamma \sum_{\boldsymbol{x}' \in \boldsymbol{X}} P(\boldsymbol{x}, u, \boldsymbol{x}') \max_{u'} Q^*(\boldsymbol{x}', u'). && (1.62) \end{aligned}$$

In DP, an approximate solution of this equation can be estimated with policy iteration and value iteration algorithms (Bertsekas and Tsitsiklis, 1996; Puterman, 2014; Sutton and Barto, 1998)). On the other hand, reinforcement learning is subdivided into model-free methods, that directly estimate the policy or value functions without trying to figure the transition kernel, e.g. SARSA (Sutton and Barto, 1998), Q-learning (Watkins and Dayan, 1992), REINFORCE (Williams, 1992), and model-based methods that first approximate the transition kernel to either apply dynamic programming methods, e.g. UCRL (Auer and Ortner, 2007; Jaksch et al., 2010), or re-applied model-free algorithms, e.g. Dyna-Q (Silver et al., 2008; Sutton, 1991).

More specifically, value-based model-free algorithms learn the optimal policy by performing a training loop. Given an initial state $\boldsymbol{x}_1$ and an initial Q-value function, these algorithms sample a new transition $(\boldsymbol{x}_t, u_t, r_{t+1}, \boldsymbol{x}_{t+1})$ from the environment, update the Q-value function, update the policy and repeat the operation until a final state is reached. Then, new trajectories are generated until the optimal policy is obtained, i.e., the policy stops changing.

An algorithm is said to perform on-policy learning when the policy $\boldsymbol{\pi}$ is trained from trajectories sampled from $\boldsymbol{\pi}$ itself. On the other hand, an algorithm is said to be off-policy when the policy $\boldsymbol{\pi}$ is trained from trajectories sampled from another behavior policy $\boldsymbol{\pi}'$. Both families of algorithms have their pros and cons: while on-policy are well-understood and stable but sample-inefficient, off-policy algorithms suffer from weaker convergence guarantees and high-variance (Geist and Scherrer, 2014; Munos et al., 2016; Van Hasselt et al., 2018) but they can re-use previously generated trajectories (Lin, 1993; Schaul et al., 2015b) or even use human trajectories (Hester et al., 2018). Besides, off-policy can natively use exploratory policies which is critical point as we will see in the next subsection.

1.3.4 Exploration vs Exploitation Dilemma

As stated above, value-based and model-free control algorithms alternate between updating the Q-value function and the policy. Although the action-value function update differs from one algorithm to another, policy updates perform a greedy improvement over the action-value function:

$$\boldsymbol{\pi}'(u|\boldsymbol{x}) = \begin{cases} 1 & \text{if } u = \arg\max_{u' \in \boldsymbol{U}} Q(\boldsymbol{x}, u') \\ 0 & \text{otherwise.} \end{cases} \tag{1.63}$$

Intuitively, a greedy policy may become trapped in a sub-space of the MDP, it would keep producing the same sub-optimal trajectories at training time. Somehow, we need to force the algorithm to vary its actions to have diverse trajectories. In other words, the agent must explore its environment. Formally, this intuition is known as the Greedy in the Limit with Infinite Exploration (GLIE); model-free algorithms converge to the optimal policy only if all the state-action pairs are explored infinitely many times and the policy converges to a greedy policy. In other words, GLIE states that the algorithm needs to explore all the states of the space before being able to discover an optimal deterministic policy. Therefore, a simple exploration policy is the ϵ-greedy policy defined by:

$$\boldsymbol{\pi}'(u|\boldsymbol{x}) = \begin{cases} \epsilon/m_t + 1 - \epsilon & \text{if } u = \arg\max_{u' \in \boldsymbol{U}} Q(\boldsymbol{x}, u') \\ \epsilon/m_t & \text{otherwise,} \end{cases} \tag{1.64}$$

where ϵ is the probability to explore a random non-greedy action and m_t is a decay factor decreasing over time. ϵ-greedy is guaranteed to explore the full action-state space and converge to a deterministic policy when m goes to infinity. Unfortunately, this can take an exponentially number of timesteps before exploring all states (Kakade et al., 2003).

More generally, exploration suffers from the exploration-exploitation dilemma. Should the agent keep investigating its environment to potentially improve future rewards? or should it take action based on its current knowledge of the environment to maximize its reward, risking converging to sub-optimal policies? This dilemma is still an open-problem in reinforcement learning, and several methods have been proposed to perform efficient exploration. It ranges from smart action-value function initialization (Wiewiora, 2003), to intrinsic motivation methods that provide an extra reward signal to explore new states (Chentanez et al., 2005; Schmidhuber, 1990) which includes "pseudo-count" methods (Bellemare et al., 2016; Brafman and Tennenholtz, 2002; Ostrovski et al., 2017) or optimism in the face of uncertainty (Brafman and Tennenholtz, 2002; O'Donoghue et al., 2017; Strehl and Littman, 2008) and numerous other methods that go far beyond the scope of this thesis (Conti et al., 2018; Fortunato et al., 2018; Osband et al., 2016).

1.3.5 Value-Function Approximation

Until now, we implicitly assumed the action-value functions to be stored inside a single vector representation $Q^{\pi}(\boldsymbol{x}, u)$. In practice, this representation quickly becomes computationnally intractable and prevents from dealing with large scale problems. For instance, "simple" games such as Backgammon have 10^{20} states and Go have 10^{170} states. Furthermore, discrete representation prevents from

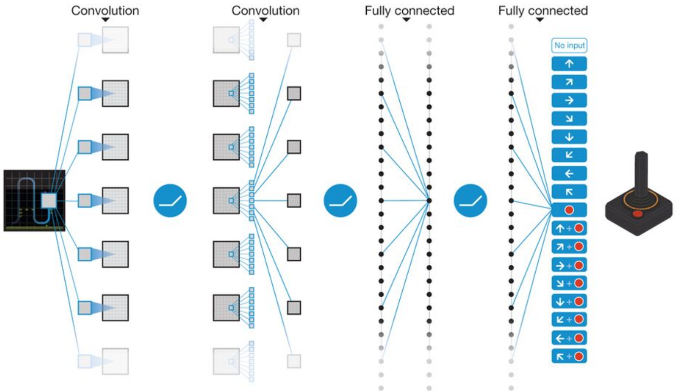


Figure 1.9: DQN architecture. The inputs are stacked screenshots while the action space is the combination of the buttons of the Atari-2600 console. The figure comes from (Mnih et al., 2015)

studying continuous action and state space. If we ignore the storage requirement, GLIE also requires exploring all those states at least a few times before converging to an optimal policy, which is again computationally unfeasible.

A classic solution is to learn to generalize across states and actions to avoid storing every single state and action. Instead of representing the value functions with giant look-up tables, they are estimated with function approximation. These function approximators can either be non-parametric functions such as trees (Ernst et al., 2005), nearest neighbours (de Lope et al., 2011), or parametric functions such as linear models (Bertsekas and Tsitsiklis, 1996; Tsitsiklis and Van Roy, 1997), Gaussian processes (Engel et al., 2003), neural networks (Riedmiller, 2005; Tesauro, 1995). For instance, following Eq. 1.5, the value-based function $\tilde{V}^{\pi}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $\tilde{Q}^{\pi}_{\boldsymbol{\theta}}(\boldsymbol{x}, u)$ can be linearly approximated by:

$$\tilde{V}^{\pi}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \sum_{m=1}^{M} \theta_m \phi_m(\boldsymbol{x}), \tag{1.65}$$

$$\tilde{Q}^{\pi}_{\boldsymbol{\theta}}(\boldsymbol{x}, u) = \theta_0 + \sum_{m=1}^{M} \theta_m \phi_m(\boldsymbol{x}, u), \tag{1.66}$$

where the state is represented as a vector $\boldsymbol{x}$ and M is the number of basis functions. As a counter aspect, function approximations lead to residual approximation errors, that may explode when bootstrapping the value function (Baird, 1995; Bertsekas and Tsitsiklis, 1996; Tsitsiklis and Van Roy, 1997). Therefore, several RL algorithms lose their theoretical convergence guarantees in the linear case while combining function approximation, off-policy learning, and bootstrapping (Baird, 1995; Boyan and Moore, 1995; Sutton and Barto, 1998; Van Hasselt et al., 2018).

Nonetheless, reinforcement learning has witnessed impressive empirical successes since the development of deep learning methods. For instance, Deep Q-Networks (DQN) (Mnih et al., 2013, 2015) first showed that RL could learn complex policy from raw features on Atari-2600 games (Bellemare et al., 2013). To do so, they extended fitted-Q methods (Ernst et al., 2005; Riedmiller, 2005), and use the neural architecture depicted in Fig. 1.9 as the function approximator. Since then, there has been a continuous improvement over Atari games by continuously improving neural fitted-Q approaches (Hasselt et al., 2016; Hessel et al., 2018; Mnih et al., 2016; Wang et al., 2016b). Other, major successes include defeating professional players at board games such as Go (Silver et al., 2016, 2017a,b) or video games such as StarcraftII (Vinyals et al., 2019) or Dota2 (Chan et al., 2019). Li

(2017) provide an extensive list of successful applications of deep RL that ranges from robotics, to natural language processing, business management, finance, healthcare, smart grid and other various topics.

1.3.6 Policy Gradient Theorem

In the previous subsections, the Q-functions are used as a proxy to compute the policy. Hence, we estimate the action-value function of the policy, then update the policy by being greedy over the action-value function and repeat the process until converging to the optimal policy.

An alternative solution is to directly search for the optimal policy in the space of policy, discarding the need of computing intermediate value-functions. Q-value functions have inherent issues such as overestimation (Thrun and Schwartz, 1993), instability due to the greedy-policy update (Bertsekas and Tsitsiklis, 1996) or delusional bias (Lu et al., 2018). More precisely, the larger is the action-space, the bigger is the action-value overestimation (Thrun and Schwartz, 1993), requiring additional tricks to mitigate this effect such as having two Q-value estimates (Hasselt, 2010; Hasselt et al., 2016), use ad-hoc regularization (Bahdanau et al., 2017), perform action elimination at each state (Zahavy et al., 2018). Q-functions are also ill-defined when dealing with continuous action spaces. Finally, value-based methods are oriented toward finding an optimal deterministic policy, while we may desire to have a stochastic but optimal policy. On the other hand, the space of policies is sometimes easier to learn than using convoluted value functions like in robotics (Peters and Schaal, 2006). It also provides a natural representation for continuous actions.

The underlying idea is to directly approximate the policy by parametrizing it, e.g. with neural networks, and to optimize those parameters towards finding the optimal policy. Therefore, the goal is to find a policy $\pi_{\boldsymbol{\theta}}(u|\boldsymbol{x})$ that maximizes the expected return, also known as the mean value:

$$J(\boldsymbol{\theta}) = \int d_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{x}) V^{\pi_{\boldsymbol{\theta}}} d\boldsymbol{x}, \tag{1.67}$$

where $d_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{x})$ is the stationary probability state distribution induced by the policy $\boldsymbol{\pi}$, i.e, the probability of being in a state $\boldsymbol{x}$ following the policy $\boldsymbol{\pi_{\theta}}$. Following the gradient policy theorem (Sutton et al., 2000), the policy is improved by updating parameters in the direction of the gradient of the mean value:

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \eta_h \nabla_\theta J|_{\theta=\theta_h}, \tag{1.68}$$

where h denotes the training time-step and η_h is a learning rate such that $\sum_{h=1}^{\infty} \eta_h = \infty$ and $\sum_{h=1}^{\infty} \eta_h^2 < \infty$. Following (Sutton et al., 2000), the gradient of the mean value can be estimated from a batch of trajectories $\mathcal{T}_h$ sampled from the current policy $\pi_{\boldsymbol{\theta}_h}$ by:

$$\nabla J(\boldsymbol{\theta}_h) = \left\langle \sum_{t=1}^{T} \sum_{u_t \in V} \nabla_{\boldsymbol{\theta}_h} \log \pi_{\boldsymbol{\theta}_h}(u_t|\boldsymbol{x}_t)(Q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t) - b) \right\rangle_{\mathcal{T}_h}, \tag{1.69}$$

where b is some arbitrary baseline function which can help to reduce the variance of the estimation of the gradient. Intuitively, the policy gradient follows the direction of the Q-value function, which is a less drastic change than being greedy over the action-value function. Inspired by the REINFORCE algorithm (Williams, 1992), it is possible to estimate the Q-function by using Monte-Carlo rollouts. Thus, the inner sum of actions is estimated by using the actions from the trajectory, simplifying Eq. (1.69) as follow:a

$$\nabla J(\boldsymbol{\theta}_h) = \left\langle \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}_h} \log \pi_{\boldsymbol{\theta}_h}(u_t|\boldsymbol{x}_t)(Q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t) - b) \right\rangle_{\mathcal{T}_h}. \tag{1.70}$$

Unfortunately, this gradient has very high variance and it can require a lot of samples before converging. We can partly alleviate this issue by analytically computing the optimal baseline given a rollout (Peters and Schaal, 2006). Another approach is to reduce the variance by using value-based methods to better estimate the action-value function. Those algorithms are known as actor-critic and rely on the second set of parameters $\boldsymbol{\phi}$ to estimate the Q-value function:

$$\nabla J(\boldsymbol{\theta}_h) = \left\langle \sum_{t=1}^{T} \sum_{u_t \in V} \nabla_{\boldsymbol{\theta}_h} \log \pi_{\boldsymbol{\theta}_h}(u_t|\boldsymbol{x}_t)(Q_{\boldsymbol{\phi}}^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t) - b) \right\rangle_{\mathcal{T}_h}, \tag{1.71}$$

where the parameters $\boldsymbol{\phi}$ are generally optimized using temporal difference errors. In addition to historical actor-critic methods (Grondman et al., 2012), deep reinforcement learning led to various successful deep actor-critic architectures e.g., Asynchronous Actor-Critic Agents (A3C) with n-step returns (Mnih et al., 2016), IMPALA and REACTOR, two distributed actor-critic architectures with off-policy corrections (Espeholt et al., 2018; Gruslys et al., 2018), or Soft-Actor Critic (SAC) that jointly optimizes the expected reward and entropy of the policy (Haarnoja et al., 2018). Other algorithms are also derived from the policy gradient theorem such as Deep Deterministic Policy Gradients (DPG) for continuous action (Lillicrap et al., 2015; Silver et al., 2014) or Proximal Policy Optimization (PPO) and Trust Region Policy Optimization (TRPO) which aims to approximate natural gradient updates of the policy (Schulman et al., 2015, 2017). Although those algorithms are crucial in modern deep reinforcement learning, we do not extensively use them in this thesis, and, we invite the reader to look at Lapan (2018) for a more complete overview.

Chapter 2

Re-Uniting Vision and Language

«You can't cram the meaning of a whole %&!$# sentence into a single $&!# vector! »*

Raymond J. Mooney

$$\begin{pmatrix} -0.976 & -1.166 & \cdots & 0.183, \\ -0.334 & -0.916 & \cdots & -0.785 \\ \vdots & \vdots & \ddots & \vdots \\ 1.647 & 0.313 & \cdots & .42 \end{pmatrix}$$

The Meaning of Vectors, Chap.4, Sec.3
by Hal

Contents

As far as the roots of AI go, vision and language have always been among the major components of any AI system. For instance, Bush and Wang (1945) imagined the memex, an intelligent system that cleverly collects and browses visual and textual information. However, a holistic approach of AI was quickly abandoned, favouring one modality over the other. An interesting example is the Turing test, where intelligence is assessed through a textual dialogue; yet, vision (and sounds) are excluded to avoid any form of bias in the evaluation process (Turing, 1950). A few years later emerged the first promising results in both Natural Language Processing (NLP), with a rule-based translation system (Dostert, 1955), and Computer Vision (CV), with a statistical model for image recognition (Rosenblatt, 1958). Although both works were highly controversial (Hutchins, 2003; Olazaran, 1996), they also reflect the common mathematical paradigms (rule-based vs. statistical) that would drive and cluster each research community in the next three decades. As a result, the few multimodal AI systems that were produced decoupled the language and vision into different submodules, ignoring potential low-level interconnection between modalities (Herzog and Wazinski, 1994; McDonald and Conklin, 1982). In parallel, cognitive science had been exploring how the processing of language and vision is coupled, inviting the AI community to broaden their thinking on language understanding (Harnad, 1990; Jackendoff, 1987; Miller and Johnson-Laird, 1976). However, we have to wait until the late '90s and the spread of NLP statistical approaches to witness the first appearance of hybrid methods (Barnard et al., 2003; Mooney, 2008; Roy, 2002). Finally, language-vision tasks and algorithms blossomed in the '10s (Darrell and Mooney, 2011; Gargett et al., 2010), and the research topic was further anchored with the success of deep learning that standardized NLP and CV representation learning (Malinowski et al., 2015; Vinyals et al., 2015b).

In light of this recent evolution, we study the modern deep learning background and history in both CV and NLP in this chapter. Every time, we end by examining the philosophical, practical, and cognitive motivations for better integrating vision and language in AI systems. As a second step, we define the multimodal learning paradigm and we explore how deep learning components can be designed to perform multimodal learning with vision and language.

2.1 Deep Learning and Computer Vision

Computer Vision (CV) is the construction of explicit, meaningful description of physical objects from images (Ballard and Brown, 1982). A major challenge of computer vision is visual recognition, which consists of recognizing (and labelling) an object in a given image by machines. For instance, image classification involves categorizing an image into a set of K pre-defined categories, image detection consists of locating and labelling a specific set of objects inside the image, and image segmentation aims at grouping pixels by either their semantics (car, person, etc.) and/or their instances (e.g., car1, car2) as highlighted in Fig 2.1. Historical methods relied on hand-crafted feature detectors and descriptors that respectively search for salient locations and features in objects (Dalal and Triggs, 2005; Lowe, 2004). More generally, the goal of the feature descriptors was to find the best image representation that can solve the task at hand.

In 2010, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was introduced to benchmark computer vision algorithms. In the initial version of the competition, the goal was to identify the main objects present in images. For each photograph, the competitors must produce a list of at most five object categories in the descending order of confidence. ILSVRC used 1.2 million raw labelled images from the russakovsky2015imagenet dataset (Russakovsky et al., 2015), which was several orders of magnitude bigger than other computer vision datasets (Everingham et al., 2015; Fei-Fei et al., 2007). As a result, ILSVRC quickly became the golden benchmark in computer vision. At the beginning of the competition, the best image classifiers were built on top of hand-crafted features descriptors; they reached 26% top-5 error accuracy.

2.1.1 Vision Neural Architectures

The ILSVRC challenge turned out to be a fertile ground for deep learning, leading to a massive shift of the vision community towards neural networks. Here, we briefly present the dominant visual deep learning architectures that emerged during this competition, and depict them in Fig 2.2 and Tab 2.1.

AlexNet The work that rekindled interest in CNN was AlexNet (Krizhevsky et al., 2012), which nearly halved the previous best top-5 error accuracy from 26% to 15.3% at ILSVRC in 2012. The

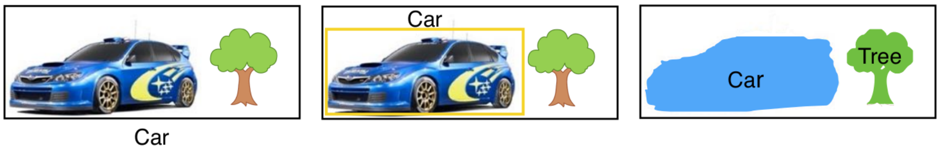


Figure 2.1: From left to right: image classification, detection, and segmentation tasks. The image classification is about assigning a category to an image, detection classifies and localizes an object, while segmentation asks for a more detailed scene representation. The figure is imported from (Malinowski, 2017)

network was composed of five convolution layers with ReLU and three fully connected layers with dropout. Krizhevsky et al. (2012) already stated that deep learning model would perform better by increasing computationally power and the amount of data, also highlighting the importance of network depth and overparameterization.

VGG-Net Two years later, Visual Geometry Group Network (VGGNet) managed to reduce the top-5 error accuracy to 7.3% by stacking sixteen convolution layers with small kernel size (Simonyan and Zisserman, 2015). This network became very popular inside the computer vision community as several pretrained networks were openly available.

GoogleLeNet The same year, GoogleLeNet (or Inception Network) won the ILSVRC challenge with 6.7% top-5 error accuracy (Szegedy et al., 2015). The authors stacked twenty-two convolution layers with mixed kernel size and removed the final fully connected layers, making the network very light-weighted. Besides, Szegedy et al. (2015) used newly developed training methods, incorporating batch-normalization (Ioffe and Szegedy, 2015) and using adaptive learning rates (Tieleman and Hinton, 2012).

ResNet The next year, Residual Neural Network (ResNet) supplanted both VGGNets and GoogleLeNet with 3.6% top-5 error accuracy (He et al., 2016), out-performing human level on image classification. The authors implemented residual connections to bypass the input from one convolution layers to another. As a result, they could stack up to 152 convolution layers with batch normalization. As ResNet plays an important role in this thesis, we further examine this architecture in the next subsection 2.1.2.

Name	Year	Depth	Params	Top-5 Error
Pre-CNN	2011	n/a	n/a	26.2%
AlexNet	2012	8	62M	15.3%
VGG-Net	2014	19	138M	7.3%
GoogleNet	2014	22	7M	6.7%
ResNet-152	2015	152	60M	3.2%
Squeeze-and-Excitation	2017	152	68M	2.3%

Table 2.1: Evolution of the ILSVRC top-5 accuracy error alongside with some network properties. Canziani et al. (2016) performed an extended comparison of those networks.

After ResNet, deep learning architecture kind of reached a plateau and subsequent neural architectures such as ResNetXt (Xie et al., 2017), Squeeze-and-Excitation Networks (Hu et al., 2017a) mostly explored variants of residual connections or (Tan and Le, 2019) optimize the width, depth and image

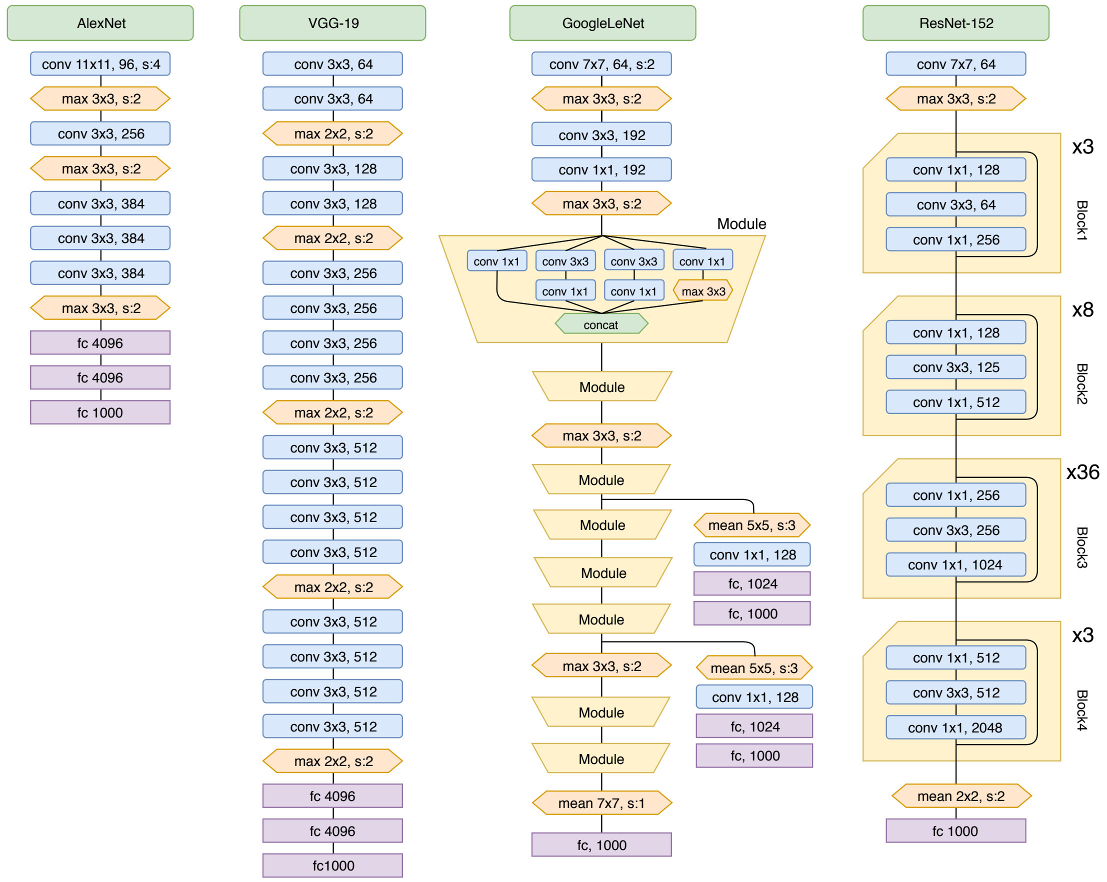


Figure 2.2: A sketch of some traditional visual networks, showing the increase of network complexity and depth along the years. Regularization layers (e.g., dropout, batch-normalization, and local response normalization) and activation functions are removed for clarity. The convolutions are described by their kernel size, the number of kernels and the stride size (default is one); fc stands for fully connected layers with the number of units. We invite the reader to look at Szegedy et al. (2015)'s paper to have the number of kernel dimension to each inception module. Note that the first convolution layer in each ResNet block has a stride of size 2.

size to better leverage ResNet abilities. By the end of the ILSVRC challenge in 2017, deep learning has become the de-facto frameworks for computer vision algorithms. Vision competitions moved on more large-scale complex vision recognition tasks such as object detection, object segmentation or pose-estimation (Lin et al., 2014). Large-scale video datasets have also started to emerge (Abu-El-Haija et al., 2016). ILSVRC was finally stopped, and the organizers are said to be collecting a 3D image dataset as a next challenge. As illustrated by this thesis, there was also a renew of interest for multimodal datasets both with newly collected images alongside multimodal annotations such as the Visual Genome (Krishna et al., 2017) or MS Coco (Chen et al., 2015), or the enhancement of previously existing CV dataset with new task-specific information such as visually grounded questions or dialogues (Antol et al., 2015; Das et al., 2017a; de Vries et al., 2017; Kazemzadeh et al., 2014).

2.1.2 Residual Networks

As described above, ResNet won the ILSVRC 2015 classification competition by changing the flow of image processing inside the network. While previous convolution networks (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; Szegedy et al., 2015) constructed a new neural representation

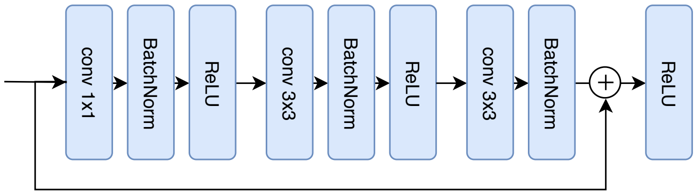


Figure 2.3: A residual function $R(.)$ for ImageNet for ResNet (He et al., 2016). A classic variant of the ResBlock is to put the ReLU inside the residual function.

at each layer, ResNet iteratively refines a representation by adding residuals connection. This modification enables to train very deep convolution networks without suffering as much from the vanishing gradient problem. More specifically, ResNets are built from Residual Blocks (ResBlocks),

$$\boldsymbol{F}^{k+1} = \mathrm{ReLU}(\boldsymbol{F}^k + R(\boldsymbol{F}^k)), \tag{2.1}$$

where F^k denotes the outputted feature map. We refer to $F_{n,c,w,h}$ to denote the n^{th} input sample of the c^{th} feature map at location (w, h). The residual function $R(F^k)$ is composed of three convolutions layers (with a kernel size of 1, 3 and 1, respectively) with batch-normalization followed by ReLU activation as depicted in Fig 2.3.

ResBlocks are stacked to form a stage (or block) in which the representation dimensionality stays identical. A general ResNet architecture starts with a single convolutional layer followed by four stages of computation as shown in Fig. 2.2. The transition from one stage to another is achieved through a projection layer that halves the spatial dimensions and doubles the number of feature maps. There are several pretrained ResNets available, including ResNet-50, ResNet-101 and ResNet-152 that differ in the number of residual blocks per stage.

2.1.3 A Word on Transfer Learning

Why would one care about a specific neural architecture for object recognition if one wants to perform object segmentation? One of the unexpected benefits of neural networks is the adaptability of the learned features across different datasets and even different tasks (Castrejon et al., 2016; Yosinski et al., 2014). As pointed out in Sec. 1.2.1, the first layers of convolution networks tend to learn features that resemble either Gabor filters or colour blobs while training on images. Such behavior is observed across different vision tasks ranging from classic object recognition (Krizhevsky et al., 2012) to unsupervised learning (Lee et al., 2009). Intuitively, it suggests that there is little need to retrain those layers for new tasks as the network would converge again to the same feature extractors. The next question is to know which layers can be re-used, and which must be re-trained.

In computer vision, it turns out that the full stack of pretrained convolution layers can be kept as a general image feature extractor. This operation, known as transfer learning, consists in training a first base network on a base dataset and task before reusing a subset of this base network to train a second target network on a target dataset and task (Yosinski et al., 2014). For example, a network is first trained on large vision dataset (e.g., ImageNet), the classification layer is then removed, and the network is used as a preprocessing unit to perform another computer vision task. Such pipeline is used in modern object localization methods; we first extract image features with an arbitrary pretrained ConvNet, and then output bounding boxes with object categories by only using the extracted features, discarding the initial raw images in the second processing round (He et al., 2017; Lin et al., 2017; Redmon and Farhadi, 2017; Ren et al., 2015b). In these examples, the ConvNet networks are

(a) Panda (57.7% conf.) + noise = Gibbon (99.3% conf.) (b) A Husky misclassified as a wolf

Figure 2.4: (a) Example of image perturbation that lead to error classification without changing the global appearance of the images (Goodfellow et al., 2015). (b) Detecting salient activation while misclassifying a Husky into a wolf (Ribeiro et al., 2016).

pretrained on ImageNet (Russakovsky et al., 2015) and the localization layers are trained on different datasets such as MS Coco (Lin et al., 2014) or VS Pascal (Everingham et al., 2015). Funnily, some modern neural architectures are based on networks pretrained on object localization, which were themselves based on object classification pretrained networks (Anderson et al., 2018b).

On a different note, we also observe a clear correlation between the performance on pretrained networks on ImageNet and the subsequent performance on the transferred task (Kornblith et al., 2018). It is sometimes beneficial to also finetune pretrained networks to tackle the new task at hand better. A classic finetuning approach involves backpropagating the target task error into the upper-most layers of the pretrained networks. More original techniques include fine-tuning specific network parameters (e.g., batch-normalization parameters as explored in Chapter 5) or adding small adaptive layers inside pretrained networks (Houlsby et al., 2019).

Transfer learning methodology has unarguably been part of the success of deep learning. However, its current application may also become a limiting factor to scale up to complex problems. For instance, multimodal tasks may require low-level interaction between modalities to be solved; thus, naively stacking pretrained network may be suboptimal. In this thesis, we explore this potential limitation in visually grounded language tasks, examining how transfer learning can be tempered with multimodal tasks in Chapter 5.

2.1.4 Language as a Natural Extension of Computer Vision

The Impact of Language over Perception

In the previous subsections, we examined the recent successes of deep learning in image recognition tasks. Yet, we may argue that we are still far from the original holy grail of computer vision: developing models that have a complete understanding of visual scenes (Geman et al., 2015; Krishna et al., 2017; Malinowski, 2017). As mentioned in the introduction, we still witness inconsistencies while classifying images, e.g., labelling human faces as gorilla (Jackyalciné, 2015), or generating images, e.g. applying zebra stripping onto a horse rider (Zhu et al., 2017a). More surprisingly, we can generate image noise that would adversely trick a deep neural network into changing the object labelling without altering the global image perception (Goodfellow et al., 2015) as shown in Fig. 2.4a. Although the gorilla misclassification and zebra stripping were due to biased datasets, and the image noise was tailored to deceive the network, they nevertheless reflect the lack of visual concept understanding. In practice, we observe that image classifiers sometimes use improper discriminant features such as background colours (Ribeiro et al., 2016; Selvaraju et al., 2017) as highlighted in Fig. 2.4b. Besides, the networks are improperly biased towards image texture rather than object shapes Geirhos et al. (2019), which are known to be robust visual features to frame human understanding (Landau et al., 1988). In other words, neural networks would use spurious data correlation rather than composing and reasoning over robust visual features. Considering cognitive evidence, we here explore some intuitions arguing in favour of integrating linguistic cues to alleviate visual understanding, and how they could complement other research directions such as adversarial training (Elsayed et al., 2018), or architectural changes.

On a human cognitive level, there is a global consensus that language eases image disambiguation, category learning, and visual reasoning during infancy (Dessalegn and Landau, 2008, 2013; Ferguson and Waxman, 2017). For instance, Waxman and Booth (2001); Waxman and Markow (1995) point out the complex interconnections between the acquisition of new words and the acquisition of a hierarchy of visual concepts. They observe that infant learn to consistently differentiate and cluster object categories (e.g., animal) and object attribute (e.g., colours) in pair with the understanding of nouns and adjectives. Dessalegn and Landau (2008, 2013) show that 4-year children have inherent difficulties to simultaneously visually reason over colours and object locations. However, the children are more successful when hearing linguistic description specifying colours and directions (e.g., "the red is on the left"), suggesting that language helps bridging visual concepts together.

A subset of the cognitive science community even argues that our language may affect the way human may perceive the world, which is also known as the Sapir-Whorf hypothesis (Kay and Kempton, 1984; Whorf et al., 1956). In its extreme variants, this assumption, or "pop-up" effect, supposes that one can only see the rainbow colours that it can name. Put differently, learning to name a new colour would pop-up the perception of this colour in the rainbow. Although the long-term impact of language over perception is highly controversial (Firestone and J. Scholl, 2014; Klemfuss et al., 2012), some neuroscience evidence that language temporally affects visual processing by setting visual priors, altering how incoming information is processed (Boutonnet and Lupyan, 2015; Kok et al., 2014; Lupyan and Ward, 2013). Simply put, we start perceiving objects after hearing their descriptions, like turning clouds into dogs in the sky.

Albeit there is still an open debate to which extend language may alter perception, linguistic cues do help visual processing learning in humans, and these observations can be carefully extended to machine learning. In that spirit, He and Peng (2017) use natural language descriptions to help to find discriminating parts or characteristics for each image within artificial neural networks. Rupprecht et al. (2018) learn to refine image segmentation of a pretrained network by using textual hints at evaluation time, dynamically altering the weight and activation of a ConvNet. In Chapter 5, we similarly explore how to alter visual processing for a given linguistic query to extract more comprehensive multimodal representations. As a parallel of category learning, Redmon and Farhadi (2017) leverage word taxonomy from WordNet (Miller, 1995) to retrieve semantic relationships between ImageNet classification labels (Russakovsky et al., 2015) and the MS Coco object detection labels (Lin et al., 2014). Thus, they could link 80 MS Coco labels such as "Dog" to the 9000 categories in ImageNet such as "Norfolk Terrier" or "Golden Retriever". The authors then design a joint loss training, allowing the model to locate and classify objects over ImageNet categories by using the annotation from MS Coco. Finally, Elhoseiny et al. (2013); Frome et al. (2013) perform zero-shot learning by generating a classification layer conditioned on textual descriptions, where the classifier learns to capture and correlate attributes from text input and visual features.

Expressing Vision Understanding with Language

In their current form, neural networks have limited abilities to express their visual understanding. Object recognition mostly consists of computing the most likely class labels from predefined sets, which grants little space for the models to indicate novelties, potential ambiguities, or partial solutions. As a next step, we want to train models to demonstrate their ability to understand an image in a less engineered fashion. For instance, humans can talk about what they see by describing an image within a few words. They can also answer questions about images, discuss about relevant details, or look for an underlying meaning of an image (Farhadi et al., 2010; Zitnick and Parikh, 2013). On a different perspective, humans can draw pictures from a textual description, and alter them given specific instruction. Following this intuition, a natural improvement of computer vision algorithms would be to integrate a linguistic module to assess their visual understanding. In this spirit, Anne Hendricks et al. (2018) train the neural network to explain its prediction, e.g. bird breed, with short descriptions, e.g. listing key attributes, although the authors faced some issues with the automatic evaluation process. Reciprocally Ling and Fidler (2017) guide the network to correctly describe an image by providing textual hints. As further explored in Sec. 3.1, we recently witnessed a renew of interest in this important line of research with image captioning tasks (Lin et al., 2014; Ordonez et al., 2011), or visual question answering (Antol et al., 2015; Malinowski, 2017), which have been depicted as a new visual Turing test. In theory, we also limit the risk of spurious relationship by complexifying the visual tasks, even though the reality may be a bit more nuanced (Agrawal et al., 2016). More generally, enhancing computer vision with language may have significant application in robotics (Chai et al., 2016; Tellex et al., 2011) and human-machine interaction (Dumas et al., 2009; Jaimes and Sebe, 2007; Pantic and Rothkrantz, 2003; Zeng et al., 2009).

To summarize, we argue that language is a natural extension of computer vision tasks. First, linguistic cues may refine visual representation by several means: they can guide the learning, pinpoint discriminant features, enhance category learning, or disambiguate image content. Second, language provides a fine-grained framework to assess image understanding and potentially improve visual representation. Finally, the linguistic channel seems to be a natural evolution for CV algorithms for the improvement of human-machine interaction.

2.2 Natural Language Processing

Natural Language Processing (NLP) aims to extract representations of textual information to read and make sense of human languages in a valuable manner. Classic NLP tasks range from automatic translation, text summarization, question answering to dialogue systems. From its early days, NLP has been heavily influenced by two linguistic approaches: rationalism and empiricism (Manning et al., 1999). The rationalist approach assumes that language possesses an underlying structure that must be discovered, conjecturing that human language is innate and can be modelled like physic laws. The empiricist approach suggests that language is a cognitive process that can be learned through experimentation, advocating to explore learning mechanisms rather than linguistic models. In the 50s, Chomsky (1957) released his work on transformational generative grammars, which marked a milestone in the NLP community; this mathematical theory of language defines a grammar through rule-based descriptions of syntactic structures and uses it to parse and transform texts. Generative grammars quickly gained popularity and became the dominant NLP paradigm. It led to some remarkable rule-based systems such as ELIZA (Weizenbaum, 1966), a conversation agent impersonating a psychologist by smartly translating patients' answers into open-questions, or SHRDLU (Winograd, 1971), an interactive interface that followed textual human instructions to operate inside a virtual toy world as shown in Fig 2.5; However, it later turned-out that rule-based agents require an exponentially growing number of rules to handle complex tasks, drastically limiting their scope of application. In parallel, there was a resurgence of empiricism theory in the AI community that learns statistical mod-

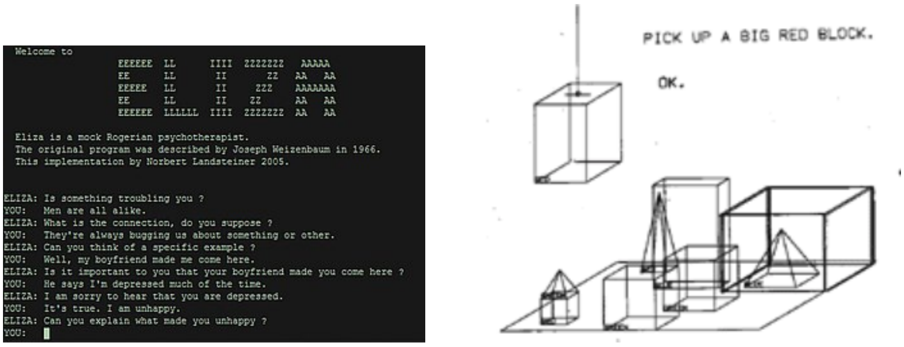


Figure 2.5: (left) ELIZA interface. Impressively, ELIZA manages to give an illusion of understanding with only a pages of handcrafted-rules (Weizenbaum, 1966). (right) Representation of the block world of SHRDLU. The agent follows instruction to move and stack geometrical figures.

els of language by parsing and analyzing corpora of text. This experimental protocol has been inspired by distributional semantics hypotheses, which assume that words occurring in the same contexts tend to purport similar meanings (Firth, 1957; Harris, 1954). Helped by access to new computing and data resources, this approach had slowly spread within the NLP community, and statistical models had slowly supplanted and finally outperformed pure rule-based systems. Statistical algorithms first tackle NLP tasks by combining handcrafted linguistic features and machine learning methods (e.g., trees, support vector machine, etc.) (Manning et al., 1999). It paved for some major successes such as Watson, a conversational agent who won the Jeopardy! Game by guessing a potential question when given an initial answer (Ferrucci, 2012). Later, deep learning models started outperforming classic statistical models (Bengio et al., 2003; Collobert and Weston, 2008; Collobert et al., 2011) by learning linguistic features (nearly) from scratch, and neural networks have become the mainstream paradigm in NLP as illustrated by today's translation systems (Wu et al., 2016).

In the following, we first introduce linguistics basics to make the reader familiar with the classic NLP terminology. We then examine neural-based architectures that we are using in this thesis. Finally, we explore how grounded language learning theory would argue towards integrating vision for the improvement of NLP.

2.2.1 Syntax, Semantics, and Pragmatics

Linguistics breaks down NLP into several concepts to potentially ease the representation learning: syntax, semantics, and pragmatics. Each of these topics entails different NLP objectives and tasks to help to disambiguate natural language. The syntax describes the structure of allowable sentences in the language, independently of the meaning of the phrases. Syntax examines the grammar rules of languages, defining nouns, verbs, adjectives, etc., and how to compose them. On the other hand, semantics associate a meaning to the syntactic elements and draw the relationship between entities. Semantics links a noun to a pronoun, looking for the plausible sentence understanding. For instance, *with eats Ana salad Bob a.* is not syntactically correct but *Ana eats a salad with Bob* is. The syntax decomposes *Ana eats a salad with Bob* as a noun followed by a verb and two nominal groups. Semantics suggest that Ana is either eating the salad with the company of Bob, or the salad contains some pieces of Bob... Pragmatics refers to practical aspects of language, analyzing the objective and/or consequence of a given phrase. It disambiguate language by using both language and external context as illustrated in Fig. 2.6. In other words, it looks at the consequence of a sentence and the intended meaning of sentences in a given context. If you ask the waiter for food, you expect to receive some

Figure 2.6: Comic strip illustrating contextual ambiguities that can be studied by pragmatics.

food. Pragmatic also makes you start sympathizing for Bob...

In NLP, syntactic tasks include word tokenization, Part-Of-Speech Tagger (POS Tagger) which associates a grammatical label to each word, or chunking, that factorizes words into grammatical groups; semantic tasks include word-sense disambiguation, role labelling, named entity extraction, and anaphora resolution; A classic natural language understanding pipeline iteratively applies syntactic and semantic algorithms to extract and translate sentences into a useful and formatted representation; this representation is then used to perform the task at hand such as translation, chatbot, etc. We invite the reader to look at (Manning et al., 1999) for further details on statistical NLP tasks and models. From the 90s', Elman (1991); Hinton (1986) advocated that neural networks are capable of learning complex language representations, without linguistic decomposition. Despite some notable works (Bengio et al., 2003; Schwenk, 2007), it took twenty years before Collobert and Weston (2008); Collobert et al. (2011) managed to train a single neural architecture that performs well on several NLP tasks without any linguistic engineering. There now exists a large variety of neural networks that deal with specific NLP issues as detailed by (Young et al., 2018).

2.2.2 Word embedding

Neural networks represent words as high dimensional floating vectors, also called word embeddings (Collobert et al., 2011; Mikolov et al., 2013a,b; Schwenk, 2007). For each task, a fixed vocabulary V of words is first defined, and each word is associated with a unique word index $\{w_i\}_{i=0}^{I}$. For instance, the word *car* may have the index 1, *blue* the index 2 etc. Plurals, conjugation or any word variants also entail new indices, e.g., *car* and *cars* are seen as two different words. Words are then processed through a look-up table, where each word index w_i is associated with a unique trainable weight vector $\boldsymbol{e}_{w_i}$. This word embedding is then used as an input to the neural networks, such as convolution networks (Collobert and Weston, 2008; Gehring et al., 2017) or recurrent networks (Cho et al., 2014b; Mikolov et al., 2010). While training neural networks on arbitrary NLP tasks, the back-propagated errors go up to the look-up table and update the word embedding.

Collobert and Weston (2008); Mikolov et al. (2013c) showed that the resulting word embedding capture meaningful syntactic and semantic regularities. More precisely, word embeddings define a continuous space representation where semantically related words have similar embeddings, and similar grammatical properties have similar vector offset. For instance, if we use standard metrics (e.g., Euclidean distance or cosine similarity), words are clustered by their topic in this latent space. As illustrated in Fig 2.7, word embedding referring to countries are grouped together, idem for colours, animals, etc. Differently, vector offset encodes semantic properties, such as gender, conjugation or plurals, e.g. $\boldsymbol{w}_{apples} - \boldsymbol{w}_{apple} \simeq \boldsymbol{w}_{cars} - \boldsymbol{w}_{car}$, or $\boldsymbol{w}_{man} - \boldsymbol{w}_{women} \simeq \boldsymbol{w}_{king} - \boldsymbol{w}_{queen}$. Those

FRANCE 454	JESUS 1973	XBOX 6909	REDDISH 11724	SCRATCHED 29869
SPAIN	CHRIST	PLAYSTATION	YELLOWISH	SMASHED
ITALY	GOD	DREAMCAST	GREENISH	RIPPED
RUSSIA	RESURRECTION	PSNUMBER	BROWNISH	BRUSHED
POLAND	PRAYER	SNES	BLUISH	HURLED
ENGLAND	YAHWEH	WII	CREAMY	GRABBED
DENMARK	JOSEPHUS	NES	WHITISH	TOSSED
GERMANY	MOSES	NINTENDO	BLACKISH	SQUEEZED
PORTUGAL	SIN	GAMECUBE	SILVERY	BLASTED
SWEDEN	HEAVEN	PSP	GREYISH	TANGLED
AUSTRIA	SALVATION	AMIGA	PALER	SLASHED

Figure 2.7: Word similarities by using word embedding of dimension 50 over a dictionary size of 30, 000 words. For each column the queried word is followed by its index in the dictionary (higher means more rare) and its 10 nearest neighbors (Euclidean distance). The table comes from (Collobert and Weston, 2008)

generic representations enable a form of linguistic transfer learning, where word embedding can be reused to new, and potentially unrelated, tasks.

Word2Vec are shallow two-layers neural networks that were designed to efficiently compute accurate word embedding (Mikolov et al., 2013a). Their training procedure is a direct application of the distributional hypothesis, which states that words that appear in the same contexts have similar semantic properties. Instead of solving a supervised NLP task, Word2Vec are unsupervised networks that iterate over text corpora to predict each word given its n neighbouring words (CBOW architecture) or to predict the n surrounding words given the current word (Skip-Gram architecture). This predictive method later turned out to be similar to other distributional algorithms that are based on word co-occurrences, such as co-word occurrence matrix factorization (Levy and Goldberg, 2014; Pennington et al., 2014). Common precomputed word embeddings include Word2Vec (Mikolov et al., 2013b), FastText (Joulin et al., 2017), or GloVe (Pennington et al., 2014). More recently, context-wise word embedding have been developed by first parse the input the full sentence, before actually computing the current embedding (Peters et al., 2018).

2.2.3 Neural Networks for Language Generation and Understanding

The Workshop on Statistical Machine Translation (WMT) has been to NLP what the ILSVRC challenge has been to computer vision; it has been a fertile ground for designing new linguistic neural architectures. More precisely, the WMT14 dataset contains several translation corpora, and the biggest one contains 36M French-English sentence pairs (Bojar et al., 2014). The algorithms are benchmarked with the BiLingual Evaluation Understudy (BLEU) score (Papineni et al., 2002), which compares a candidate translation of a text to several reference translations using n-grams. As a result, the following neural architectures were first assessed on translation tasks before being extended to more heterogeneous problems which are explored in this thesis.

In language, the dominant neural paradigm is the encoder-decoder architecture (Cho et al., 2014b), which borrows architectural concepts and terminologies from autoencoder networks (Kramer, 1991). As its name suggests, the encoder-decoder architecture is decomposed into two blocks: the encoder converts an initial linguistic input into a fixed size representation, and the decoder generates a new sequence of words based on this representation. In a French-English translation setting, the encoder first encodes the French sentence into a vector, and the decoder generates the English sentence in a second step.

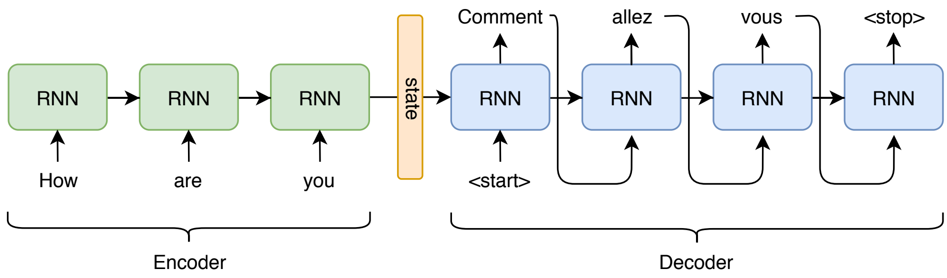


Figure 2.8: Sketch of a Seq2Seq models. The $<start>$ and $<stop>$ are used to initiate and stop the sampling procedure. The last state of the encoder is used as the initial state of decoder.

Sequence-to-Sequences The most famous language encoder-decoder architecture is the Sequence-to-Sequence (Seq2Seq) model (Cho et al., 2014a,b; Sutskever et al., 2014). In its simplest form, the encoder is a RNN (e.g. LSTM or GRU) that iterates over the input word embeddings. The decoder is another RNN whose initial state is initialized with the final state of the encoder. At each time step, the decoder outputs an embedding that is fed to a softmax layer to return the related word index. This word is then re-injected as a new input to the decoder before outputting the next token as illustrated in Fig 2.8. Besides, the word vocabulary is enhanced with two extra ad-hoc tokens $\{<start>, <stop>\}$ to control the sampling procedure. On the first step, the decoder is fed with the $<start>$ token, and word generation is stopped when the decoder outputs the $<stop>$ token. By using an intermediate fixed size representation, Seq2Seq models may output a sequence of words that differs from the input one.

Training/Evaluation procedure In practice, Seq2Seq models estimate the conditional probability $p(y_{1'}, \dots, y_{T'}|x_1, \dots, x_T)$ where $[x]_{t=1}^{T}$ and $[y]_{t'=1}^{T'}$ are the sequence of input and output words. This conditional probability is estimated by:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t'=1}^{T'} p(y_{t'}|\boldsymbol{x}, y_{1'}, \dots, y_{t'-1}), \tag{2.2}$$

At training time, we minimize the negative conditional log-likelihood to obtain the encoder-decoder parameters $\boldsymbol{\theta}^*$ such as:

$$p_{\boldsymbol{\theta}^*} = \arg\min_{\boldsymbol{\theta}} -\frac{1}{N}\sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}^n|\boldsymbol{x}^n) \tag{2.3}$$

$$= \arg\min_{\boldsymbol{\theta}} -\frac{1}{N}\sum_{n=1}^{N} \log \prod_{t=1}^{T'_n} p_{\boldsymbol{\theta}}(y_t^n|\boldsymbol{x}^n, y_1^n, \dots, y_{t-1}^n) \tag{2.4}$$

$$= \arg\min_{\boldsymbol{\theta}} -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T'_n} \log p_{\boldsymbol{\theta}}(y_t^n|\boldsymbol{x}^n, y_1^n, \dots, y_{t-1}^n), \tag{2.5}$$

where $(\boldsymbol{x}^n, \boldsymbol{y}^n)_{n=1}^{N}$ are the input and output sequence pairs from the training set. At training time, the ground truth tokens $\boldsymbol{y}^n$ are are fed back in the decoder at each time step, discarding the tokens generated by the decoder $\hat{\boldsymbol{y}}^n$. This process is also known as teacher forcing (Williams and Zipser, 1989). At inference time, the ground-truth sequence $\boldsymbol{y}^n$ is not available, and the generated tokens $\hat{\boldsymbol{y}}^n$ are used as a decoder input. However, it entails a small compounding error prediction as the generated tokens may slowly diverge from the training distribution. A few methods were developed to alleviate this discrepancy such as schedule sampling, which randomly feeds back the generated tokens at training time (Bengio et al., 2015), or professor forcing, that relies on Generative Adversarial Networks (GAN) to reduce the discrepancy between training and inference hidden states distribution (Lamb et al., 2016).

On a different note, several linguistic tasks (e.g., translation) often rely on a specific language scoring such a BLEU score or edit distance. However, those scores are often non-differential, and the log-likelihood thus acts as a training surrogate. A different approach involves using the policy gradient theorem to compute a gradient from a non-differential metric, e.g., setting the BLEU score as a reward (Sutton et al., 2000; Williams, 1992). Unfortunately, the resulting gradient has a too high variance to learn an encoder-decoder from scratch, and this approach is only used to finetune neural networks (Bahdanau et al., 2017; Paulus et al., 2018; Ranzato et al., 2016). Besides, it also tends to overfit on the scoring metric, resulting in perfect but meaningless word alignments.

At inference time, we want to generate the most likely sequence of words $p(\boldsymbol{y}|\boldsymbol{x})$ from the Seq2Seq models. However, it is not possible to directly retrieve the optimal sequence from Seq2Seq models: picking the argmax of the probability distribution at each time step does not guarantee to compute the optimal trajectory:

$$\prod_{t'=1}^{T'} \max_{y_{t'}} p(y_{t'}|\boldsymbol{x}, y_{1'}, \dots, y_{t'-1}) \leq \max_{\boldsymbol{y}} \prod_{t'=1}^{T'} p(y_{t'}|\boldsymbol{x}, y_{1'}, \dots, y_{t'-1}), \tag{2.6}$$

As a result, searching for the optimal trajectory requires to explore an exponentially large number of sequences. Beam Search (BSearch) is a breadth-first search heuristic that looks for the most likely sequence of words by exploring a subset of sentences and keeping the most likely one. At each time step t', BSearch keeps the K sequences with the highest normalized probability $\frac{1}{t'}\prod_{t=0}^{t'} p(y_t|y_1, \dots, y_{t-1})$. Then, for each of the top-K sequences, it computes the probability distribution of the next words, and repeats the operations until the $<stop>$ is reached for all the K-sequences. In generative tasks, BSearch has been reported to improve the final accuracy (Sutskever et al., 2014); yet, it also suffers from severe computation latencies, and some variants are sometimes explored (Freitag and Al-Onaizan, 2017).

Attention One potential limitation of encoder-decoder architecture relies in their ability to compress the necessary information of an input sequence into a fixed-length vector. Bahdanau et al. (2015) introduced an attention mechanism that allows the network to attend to previously generated RNN states, alleviating the burden of compressing all the information in the last hidden state. More formally, given the sequence of hidden states of the encoder $[\boldsymbol{h}_t^{enc}]_{t=1}^{T}$, a context cell $\boldsymbol{c}_{t'}$ is computed at each decoding time step t based on the currrent decoder hidden states $[\boldsymbol{h}_{t'}^{dec}]_{t'=1}^{T'}$:

$$\boldsymbol{e}_{t,t'} = g(\boldsymbol{h}_t^{enc}, \boldsymbol{h}_{t'}^{dec}) \qquad \alpha_{t,t'} = \frac{\exp(e_{t,t'})}{\sum_{i,j}\exp(e_{i,j})} \qquad \boldsymbol{c}_{t'} = \sum_{t=1}^{T} \alpha_{t,t'}\boldsymbol{h}_t^{enc}, \tag{2.7}$$

where $g(.)$ is an arbitrary differential function (e.g. concatenation followed by a some non-linear projections). The context cell $\boldsymbol{c}_{t'}$ is then appended to the decoder states before generating the next token, $\boldsymbol{h}_{t'}^{dec} = RNN(y_{t'-1}, \boldsymbol{h}_{t'-1}^{dec}, \boldsymbol{c}_{t'-1}^{dec})$. A few other variants of attention mechanisms were also developed for Seq2Seq models to tackle different settings, e.g., Luong et al. (2015) explores a local attention mechanism that does iterate over the full sequence of encoder states at each step. Differently, Vinyals et al. (2015a) designed an attention-based pointer networks to handle absolute token position in sequence for Seq2Seq models.

Attention-Based Models Although attention first sticked out as a component of Seq2Seq models, an attention mechanism can be described as a generic neural block. More generally, attention can be defined as a function mapping a query and a set of key-value pairs to an output (Vaswani et al., 2017). In that spirit, we can rewrite Eq. 2.7 as follows,

$$\boldsymbol{E} = g(\boldsymbol{q}, \boldsymbol{K}), \qquad \alpha_i = \frac{\exp(e_i)}{\sum_j \exp(e_j)}, \qquad \boldsymbol{o}_{t'} = \sum_{i=1}^{T} \alpha_i \boldsymbol{v}_i, \tag{2.8}$$

where a query $\boldsymbol{q}$ is compared to a set of keys $\boldsymbol{K}$ to select a weighted linear combination of values $\boldsymbol{v}$, $\boldsymbol{E}$ is tensor output and $g(.)$ is a matching function. For instance, memory networks use an attention mechanism to answer a question $\boldsymbol{q}$ by reasoning over a linguistic knowledge base $\boldsymbol{K}$ (Kumar et al., 2016; Sukhbaatar et al., 2015; Weston et al., 2014). The matching function is the scalar product between the query and the keys $g(\boldsymbol{q}, \boldsymbol{K}) = \boldsymbol{K}\boldsymbol{q}$, and the output $\boldsymbol{o}$ is either used to answer the initial question or encode a second query vector to perform multi-hop reasoning over the linguistic knowledge base. Similarly, Neural Turing Machines combine a RNN with an external memory bank (Graves et al., 2014, 2016), and use an attention mechanism to select, read, and write over memory slots. Finally, transformers networks push attention mechanism a step further by entirely encoding and decoding sentence with a self-attention mechanism. In such cases, tokenized words are simultaneously query vectors and knowledge base components. Noticeably, this later approach completely discards RNN modules in the encoder and decoders, and it recently became the new state of the art model in multiple NLP tasks (Devlin et al., 2018; Radford et al., 2019; Vaswani et al., 2017).

Transfer Learning Although transfer learning has been very successful in computer vision, it is still an open-issue in NLP. Word embedding can admittedly be transferred from one task to another; it is often equally efficient to learn them from scratch on new tasks. On the other hand, there were several attempts to learn generic sentence embedding through unsupervised learning; it includes skip-thought vectors (Kiros et al., 2015), doc2vec (Le and Mikolov, 2014), or weighted sum of word embedding (Arora et al., 2017). Although these methods did provide generic acceptable sequence embedding, they were not as ground-breaking as VGGNet (Simonyan and Zisserman, 2015) or ResNet (He et al., 2016) in computer vision. Again, it is often easier and as efficient to retrain language models from scratch. Very recent works suggest that transformer networks are an adequate neural architecture to learn transferable high-quality sequence embedding (Devlin et al., 2018; Radford et al., 2019; Vaswani et al., 2017), but more experiments must still be done before drawing further conclusions. Another hypothesis is that the relative success of transfer is symptomatic that current models still miss some syntactic, semantic or pragmatic properties; it thus advocates for investigating for new training objectives or experimental protocols. In this spirit, we explore why multimodal learning may be a promising direction for improving semantics in the following sections.

2.2.4 Grounded Language Learning

The dominant paradigm in modern natural language understanding is to learn statistical language models from text-only corpora. As previously discussed in Sec 2.2.2, this approach is founded on a distributional notion of semantics, where the meaning of a word is based only on its relationship to other words. This approach has been very successful by leading to numerous breakthroughs in translation systems (Wu et al., 2016), automatic text generation (Radford et al., 2019), or dialogue systems (Serban et al., 2016; Sordoni et al., 2015). However, several linguistic artifacts demonstrate that those systems still lack the understanding of the words they manipulate (Gao et al., 2019). For

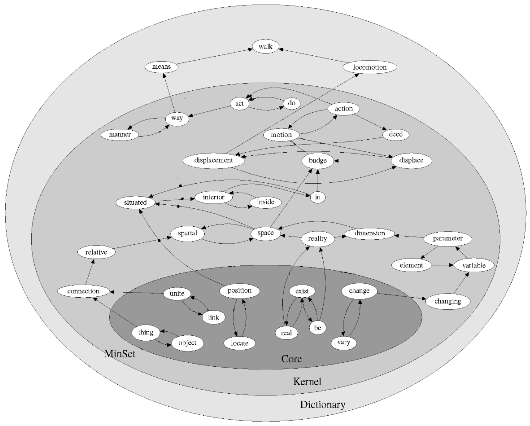


Figure 2.9: Illustration of a dictionary graph using data for a tiny (but complete) mini-dictionary (Vincent-Lamarre et al., 2016). Arrows are from defining words to defined words. The core includes self-defined words and the kernel is the minimum set that allow to define all the other words in a recurrent fashion.

instance, a dialogue system may contradict itself during a conversation, changing ages or residencies during a dialogue (Li et al., 2016a). At a lower level, word embedding tends to fail to capture some basic object properties such that bananas are yellow (Kiela, 2017), or implicit interactions, e.g., eating involves looking at the food (Kottur et al., 2016). Distributional models are still struggling to capture complete word semantics despite parsing millions of web pages. We here argue that these models miss the multimodal and interactive environment in which communication often takes place. From a cognitive perspective, we suspect distributional models to be facing the symbol grounding problem (Harnad, 1990).

The symbol grounding problem states that it is not possible to capture the meaning of a symbol if it is defined only through other symbols (Harnad, 1990). In other words, there is a problematic circularity in distributional learning: words are defined by other words, that are themselves defined by other words, etc., ending in a solipsistic form of training. To better explicit this problem, Harnad (1990) describes the Chinese dictionary problem, as an extension of the famous Chinese room argument (Searle, 1980, 1984). Imagine that you want to learn Chinese, but you only have access to a Chinese-Chinese dictionary, would you be able to learn Chinese? In theory, you have enough information to learn Chinese as dictionaries are self-sufficient, but it sounds merely impossible to start understanding Chinese if you do not have some basic understanding of initial key words. In other words, the meanings of some symbols should be learned by other means than a pure dictionary look-up. To verify this hypothesis, Vincent-Lamarre et al. (2016) transformed two English dictionaries into graphs to examine how words are interconnected, and which words are more likely to be learned through grounding as illustrated in Fig 2.9. First, the authors observed that 10% of the words suffice to define the remaining 90% words, and refer it as the kernel; similarly, 7% of the words build a self-sufficient and core set where every word is defined with the other words in this subset, pointing out the circular nature of language definition. As a second step, Vincent-Lamarre et al. (2016) compared the age of acquisition of 30k words (Kuperman et al., 2012) and remarked that the core words are acquired in average at a younger age than the kernel words, which are themselves learned later than other the words. These observations comfort the idea that language understanding is built upon both distributional and grounding approaches, where some words must be anchored in another modality before helping to define other words. This intuition has been the basis for learning multimodal language embedding, assessing whether multimodal embedding may outperform their distributional counterpart (Kiela, 2017; Lazaridou et al., 2015b), or study transfer and zero-shot learning (Kiela and

Clark, 2015; Lazaridou et al., 2014).

Language grounding learning is tightly coupled with the so-called embodiment theory (Gibbs Jr, 2005; Wilson, 2002; Wilson and Foglia, 2011). Embodiment theory states that our reasoning, language and thoughts are inextricably shaped by our perceptions and actions. It differs from Cartesian approaches which advocate studying language as an independent and amodal phenomena (Chomsky, 1957; Kintsch, 1998). In the recent years, several new elements in both cognitive science and neuroscience tend to support the embodiment theory, e.g., our semantics has been built on top of our visual perception, and sensorimotor experiences (Barsalou, 2008; Glenberg, 2015; Pulvermüller and Fadiga, 2010), and we cannot understand language without actually experiencing the physical words. Thereby, perception was shown to be crucial in semantics acquisition and understanding of shapes (Landau et al., 1988), category learning (Landau et al., 1998; Smith, 2003; Waxman and Markow, 1995), spatial relationship (Dessalegn and Landau, 2008, 2013) or nouns-adjective differentiation (Waxman and Booth, 2001). Hauk et al. (2004) highlighted the tied interconnection between sensorimotor inputs and language by showing that specific areas of the motor cortex are activated when hearing specific words. For instance, "lick" triggered areas of motor cortex that control the mouth, "kick" triggered the foot area, "pick", the hand, etc. Huth et al. (2016) later exhaustively mapped the representation of the meaning in narrative language, and the authors observe the semantics information is spread across the entire cerebral cortex with different specialized areas, suggesting a decentralized semantics understanding. More surprisingly, embodiment is not limited to the acquisition of concrete concepts, but can also be a support for abstract understanding (Borghi et al., 2017). As a result, language grounding learning is not only one solution to the symbol grounding problem, it is also the cornerstone of language understanding according to the embodiment theory; if we want our model to understand language, we also need them to understand our world (Hayes, 1978, 1985).

Embodiment and language grounding theory have been gaining strength in the machine learning community over the past ten years (Anderson et al., 2018a; Darrell and Mooney, 2011; Kiela, 2017; Mooney, 2006). Unsurprisingly, there have been direct application of embodiment in robotics (Borghi and Cangelosi, 2014; Cangelosi, 2010), for e.g., category/attribute learning (Alomari et al., 2017; Chai et al., 2016; Liu and Chai, 2015), or human-robot interaction (Landsiedel et al., 2017). Closer to the topic of this thesis, we recently witnessed the development of virtual embodiment (Kiela et al., 2016), where interactive virtual worlds are designed for language grounding (Anderson et al., 2018c; Brodeur et al., 2017; Savva et al., 2017; Wu et al., 2018c) with tasks in category learning (Hermann et al., 2017), question answering (Das et al., 2018b,c), instruction following tasks (Anderson et al., 2018c; Savva et al., 2017). Although embodiment theory is an attractive theory, it also requires to tackle new machine learning problems. As described at the beginning of the chapter, NLP, CV, and interactive models have been mainly developed as separate fields; and despite the recent success of deep learning, it is still an open question to train a holistic agent that sees, speaks and interacts with its environment. In this thesis, we thus study two intermediate milestone towards this objective. First, we focus on visually grounded language understanding through deep multimodal learning; we motivate this choice as visual perception is among the most studied modality in deep learning while being a major component of embodiment. In the next chapter, we examine dialogue systems as a testbed for learning language through interaction; in this spirit, we explore how to train an agent to learn language by uttering words in a trial-error fashion through reinforcement learning algorithms.

Chapter 3

Multimodal Learning in Practice

«All our knowledge begins with the senses, proceeds then to the understanding, and ends with reason. »

Kant, Critique of Pure Reason

Contents

In the previous chapter, we examined deep learning methods for computer vision and natural language processing. In both cases, we argued that machine learning models would benefit from interleaving the two modalities. In computer vision, language provides a powerful interface for assessing image understanding; it can also support demanding vision tasks such as image disambiguation or category learning. In natural language processing, integrating visual cues is a first step towards tackling the symbol grounding problem while aiming for a holistic language understanding. In this line of research, we explore deep multimodal learning architectures to process visual and language information into joint representations. Thus, we first rigorously define multimodal learning before highlighting the visually grounded language tasks present in this thesis. Finally, we define state-of-the-art methods that were developed to tackle these tasks, decomposing neural models into independent deep learning blocks.

3.1 Multimodal Learning

We apprehend the world around us through our senses: we observe what surrounds us, listen to noises and smell odours; our body perceives our moves, and our thoughts are projected through language. In machine learning, these individual perceptions are also referred to as modalities, and multimodal learning consists of aggregating (or disambiguating) several modalities at once. However, each modality is characterized by distinct statistical properties which are encoded through different representations, e.g., visual cues are represented by pixel images, audio signals are digitized waveforms, and words are discretized into tokens. Those heterogeneous statistics entail the significant difficulty of multimodal learning: how do we discover the relationship between modalities while disentangling or fusing them?

The research about the interplay between human perception mostly came together in the 70s with seminal works in psychotherapy (Lazarus, 1973) and cognitive science (Miller and Johnson-Laird, 1976), and later lead for the previously stated embodiment theory (Barsalou, 2008). The authors were opposed to the dominant archetype that human behaviour can be decomposed into independent concepts, such as perception and thoughts, when analyzing psychological disorders or language semantics. As a demonstration, McGurk and MacDonald (1976) showed that the consonants we hear are paired with our visual perception; the human brain perceives different sounds when observing different lips movement[1]. Those observations would motivate a first wave of multimodal learning research in AI such as Audio-Visual Speech Recognition (ASVR) systems (Dupont and Luettin, 2000; Ngiam et al., 2011; Petajan, 1984; Potamianos et al., 2003). In the '00s, multimodal learning move towards studying human-machine interaction by integrating sound, gesture, or visual cues to improve communication protocols (Dumas et al., 2009; Jaimes and Sebe, 2007; Zeng et al., 2009). Multimodal models were then developed to recognize affective states, create artificial avatars, or design intuitive interfaces (Pantic and Rothkrantz, 2003; Zeng et al., 2009). The development of the internet also raised new multimodal challenges to process and query heterogeneous data (Atrey et al., 2010; Lew et al., 2006).

Since the deep learning achievements, multimodal learning focused on studying how to combine disjoint raw inputs into high-level joint representations, irrespectively of the underlying tasks or modalities. Although historical multimodal challenges still exist, e.g., ASVR (Ringeval et al., 2018) or emotion recognition (Dhall et al., 2019), multimodal learning is now somehow perceived as an sub-constituent of deep learning research, and architectural changes are transferred across application domains as it occurred for new regularizers or optimizers. As further detailed in Sec 3.2, deep multimodal learning has been applied for intersecting vision, language, speech or even olfactory modalities (Kiela et al., 2015). Yet, multimodal neural blocks are also used to guide deep generative models towards generating sound, text or image with GAN (Mirza and Osindero, 2014; Radford et al., 2015) or auto-regressive models (Van den Oord et al., 2016a,b). Deep reinforcement learning can also be cast as a multimodal learning problem since the Q function composes with the state and action spaces for computing the policy (Mnih et al., 2013; Riedmiller, 2005). Different multimodal reinforcement learning challenges also include goal conditioning (Barreto et al., 2017; Schaul et al., 2015a) or scaling to large-scale multiplayer games with heterogeneous inputs (Chan et al., 2019; Vinyals et al., 2019).

3.1.1 Multimodal Training Objectives

Modalities may intersect differently from one problem to another, leading to different training settings. We thus decompose multimodal learning tasks into three categories, namely translation, fusion, and alignment, to formally express the interaction between the model inputs and outputs. We finally depict a summary sketch in Fig. 3.1.

[1] https://www.youtube.com/watch?v=G-lN8vWm3m0

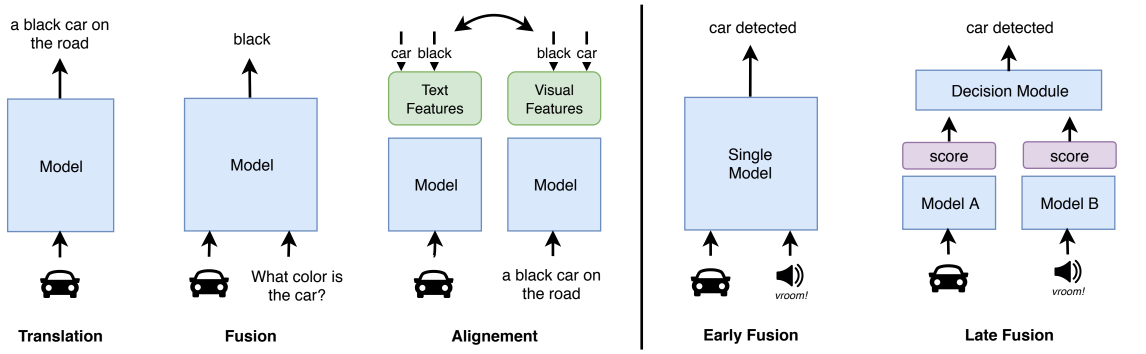


Figure 3.1: (left) Illustration of classic multimodal tasks. Translation maps one modality to another, fusion combines modality to solve a prediction task, alignment enforces feature correspondence between modalities.

Translation Translation consists in mapping one modality into another, and it generally relates to generative models. It includes topics such as speech synthesis, speech recognition (Van den Oord et al., 2016a), image-captioning (Vinyals et al., 2015b; Xu et al., 2015), or generating ambient sounds from image (Owens et al., 2016). Formally, given two modalities $\mathcal{X}$ and $\mathcal{Y}$, translation defines the function $f : \mathcal{X} \to \mathcal{Y}$ that projects the modality $\mathcal{X}$ into the space $\mathcal{Y}$.

Fusion Fusion aims at combining several input modalities to predict an outcome. It relates modalities together by retrieving input correlation to build a joint representation that tackle the task at hand. It includes topics such as ASVR (Potamianos et al., 2003), Visual Question Answering (VQA) (Antol et al., 2015; Malinowski, 2017) or instruction following (Anderson et al., 2018c; Savva et al., 2017). Given two modalities $\mathcal{X}$ and $\mathcal{Y}$, and an output space $\mathcal{Z}$, fusion aggregates both modalities at the input level, learning the function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$.

Alignment Alignment explicitly fetches the correspondence between modalities towards understanding input features. It generally involves a similarity score for guiding the training procedure, but it does not fuse both modalities into a joint representation. It often relies on (un/semi)supervised learning methods. It includes topics such as matching a specific text to visual scenes (Srivastava and Salakhutdinov, 2012; Tapaswi et al., 2015), performing text-based image retrieval (Liu et al., 2007), locating sounds in image (Arandjelovic and Zisserman, 2018), or performing zero-shot and transfer learning (Kiela, 2017; Lazaridou et al., 2015a). Given two input modalities $\mathcal{X}$ and $\mathcal{Y}$, alignment learns two feature extractor functions $f : \mathcal{X} \to \mathbb{R}^n$ and $g : \mathcal{Y} \to \mathbb{R}^n$ to enforce a consistent similarity score $sim : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ between both modalities.

Finally, we refer the reader to (Baltrušaitis et al., 2019) for a more exhaustive (but slightly different) multimodal taxonomy. Noticeably, the authors argue that multimodal problems are implicitly decomposed into multimodal subtasks, which should be tackled by specific neural blocks. For instance, neural attention mechanisms perform implicit data alignment to fuse two modalities.

3.1.2 Early and Late Conditioning in Deep Learning

The fusion procedure is traditionally split into two distinct schemes: early vs late fusion (Atrey et al., 2010; Bruni et al., 2014; Hall and Llinas, 1997; Snoek et al., 2005). Following Snoek et al. (2005), early fusion integrates unimodal features into a single model to directly perform the final task. On the other hand, late fusion first learns independent scores from external tasks before combining those

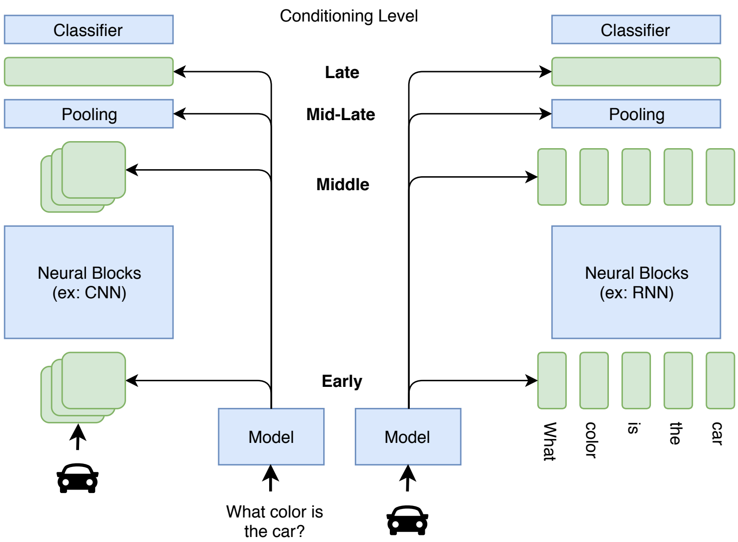


Figure 3.2: Sketch illustrating the level of multimodal conditioning in CNN and RNN neural blocks.

scores to perform the final prediction. In other words, early fusion works at the feature level while late fusion works at the scoring level (Bruni et al., 2014). Albeit legitimate, these two categories were designed for models with handcrafted features, and they fail to describe the variety of fusion schemes that emerges due to the hierarchical nature of deep learning models. In the following, we thus describe a classic deep learning scheme to define distinct feature-wise fusing mechanisms.

Deep neural networks can be seen as representation-learning methods with multiple levels of representation. Concretely, deep networks are designed towards decomposing raw input into different representation stages ranging from low-level features to and high-level ones (LeCun et al., 2015). In a similar spirit, we propose to define distinct conditioning (or fusion) mechanisms level by identifying the representation stage where both modalities are merged together. For instance, early conditioning mechanism would fuse modalities while learning low-level features, middle conditioning for middle-level features, late conditioning for high-level features etc. When aggregating two modalities, we highlight that conditioning definition is asymmetric as the modalities may encode different feature levels; therefore, we respectively refer as the *conditioned* modality, the representation with the lowest feature level, and the *conditioning* modality, the representation with the highest feature level.

Although, feature-levels are somehow acknowledged (Bau et al., 2017; Zhou et al., 2014), they remain a pure human abstraction. We thus propose arbitrary classification levels, which is based on the empirical observation that deep learning architectures are decomposed into sub-modules that entail different representation stages. In a few words, neural networks are often based on core pipeline that processes raw inputs, a pooling mechanism to induce shape invariance and a final head, which acts as feature discriminator as illustrated in Fig 3.2. Formally, given a neural block $f(.)$, a pooling mechanism $g(.)$ that turns nD-tensors into 1D-vectors, a conditioned modality $\boldsymbol{x}$, and a conditioning modality $\boldsymbol{y}$, we define conditioning-levels as follows:

- **Early Conditioning.** Early conditioning fuses $\boldsymbol{x}$ and $\boldsymbol{y}$ *before the neural block* $f(.)$. Early conditioning does not change the dimension of the input as it works at the very raw input level. A common example is to inject visual features while learning word embedding (Lazaridou et al., 2015a), or append extra visual channel to an image such as depth (Das et al., 2018a), or unit life points (Vinyals et al., 2019).

- **Middle Conditioning.** Middle conditioning fuses $\boldsymbol{x}$ and $\boldsymbol{y}$ *before the pooling mechanism* $g(.)$. Similar to early fusion, middle conditioning does not change the dimension of the input, but it fuses two processed inputs. It includes methods such as bias conditioning (Van den Oord et al., 2016b), or tilling mechanism (Malinowski and Doersch, 2018), and the so-call modulation layers that are introduced in this thesis.

- **Mid-late Conditioning.** Mid-late conditioning fuses $\boldsymbol{x}$ and $\boldsymbol{y}$ *inside the pooling mechanism* $g(.)$. Mid-late conditioning projects the conditioned modality into a lower-dimensional space guided by the other modality. More generally, we refer as middle conditioning every process that is neither in the original input space, e.g., spatial or temporal localization, nor in a linearly separable space with $1D$ vector representation. For instance, mid-late conditioning includes the vast literature of attention-pooling mechanism (Fukui et al., 2016; Jiasen et al., 2016), or object-feature attention (Anderson et al., 2018b).

- **Late Conditioning.** Late conditioning fuses $\boldsymbol{x}$ and $\boldsymbol{y}$ *after the pooling mechanism* $g(.)$. Late conditioning works with $1D$ modality representation by enforcing high-level alignment between modalities. It includes vector concatenation, projections, element-wise product, etc. As opposed to other mechanisms, late conditioning is often symmetric as there is often no discrepancy between modality representations.

Again, we acknowledge the limitation of this classification as simple reshaping may blur the lines between the different conditioning levels. Other neural architectures also have intermediate pooling layers. In any case, this categorization still provides insightful comparison during this thesis, and highlights how complementary are multimodal neural modules, and how they can be composed.

3.1.3 Conditioning Mechanism

Since we decompose conditioning levels, we here describe the mathematical and implementation details of the underlying mechanisms. We first start with high-level conditioning methods before exploring mid-late and middle conditioning.

Vanilla late conditioning In its simplest forms, late conditioning merges two vector representations by simple mathematical operations. Formally, given two modalities $\boldsymbol{x} \in \mathbb{R}^N$ and $\boldsymbol{y} \in \mathbb{R}^M$ and the resulting multimodal representation $\boldsymbol{z} \in \mathbb{R}^K$, vanilla conditioning includes concatenation $\boldsymbol{z} = [\boldsymbol{x}; \boldsymbol{y}]$, element-wise sum $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{y}$ and element-wise multiplication (or Hadamard product) $\boldsymbol{z} = \boldsymbol{x} \odot \boldsymbol{y}$. Those operations are generally followed by a projection to merge and reduce the multimodal representation. Besides, element-wise operations may require to project both input modalities to match their dimensions and ease feature alignments. Interestingly, there exist some redundancies between those late conditioning operations. For instance, linearly projecting both modalities before summing them is equivalent as concatenating modalities before linearly projecting them! In practice, the projections are not linear and each mechanism induce different inductive biases. It also common to combine the three mechanisms in a single step $\boldsymbol{z} = [\boldsymbol{x}; \boldsymbol{y}; \boldsymbol{x} + \boldsymbol{y}; \boldsymbol{x} \odot \boldsymbol{y}]$.

Bilinear transformation Bilinear models are a recurrent multimodal mechanism that interleaves two vector representations through an intermediate tensor. Formally, a bilinear transformation defines the relationship between two modalities $\boldsymbol{x} \in \mathbb{R}^N$ and $\boldsymbol{y} \in \mathbb{R}^M$ and the multimodal representation $\boldsymbol{z} \in \mathbb{R}^K$ through the tensor $\boldsymbol{W} \in \mathbb{R}^{N \times M \times K}$ such as:

$$z_k = \boldsymbol{x}^T \boldsymbol{W}_k \boldsymbol{y} \quad \Leftrightarrow \quad \boldsymbol{z} = \boldsymbol{W}(\boldsymbol{x} \otimes \boldsymbol{y}), \tag{3.1}$$

where $\otimes$ is the outer product $\boldsymbol{x}\boldsymbol{y}^T$. Tenenbaum and Freeman (1997) introduced bilinear models to disentangle latent perceptual factors in computer vision. The authors sought to separate an image style from its content, arguing that classic linear models were not rich enough to extract such complex interaction. They demonstrate the effectiveness of their approach by applying it to spoken vowel identification or zero-shot font classification. Bilinear models were then extended to several applications such facial animation by separating key facial features from visual emotions (Chuang et al., 2002; Vlasic et al., 2005) or recommendation systems to interleave users tastes and items features (Chu and Park, 2009; Koren et al., 2009; Yang et al., 2011).

In its original form, the inner tensor prevents bilinear models from scaling to high-dimensional input modalities. As mentioned by Fukui et al. (2016), language-vision tasks often entail input features $N = M = 2048$, and a multimodal representation of $K = 3000$ requires 12.5 billions parameters, which is computationally intractable. Therefore, an active multimodal research direction has been to approximate such bilinear transformation within deep networks. For instance, Fukui et al. (2016) decompose the bilinear transformation into two steps: the authors first perform a dimension reduction through a count sketch projection (Charikar et al., 2002)[2], and compute the bilinear product by performing a convolution in the Fourier domain, $\boldsymbol{z} = \text{FFT}^{-1}(\text{FFT}(\boldsymbol{x}) \odot \text{FFT}(\boldsymbol{y}))$ where $FFT(.)$ are Fast Fourier Transforms. Differently, Kim et al. (2017a) estimate a low-rank bilinear transformation by learning a linear projection for both modalities followed by an element-wise product as follows:

$$
\begin{aligned}
z_k &= \boldsymbol{x}^T \boldsymbol{W}_k \boldsymbol{y} && (3.2)\\
&\simeq \boldsymbol{x}^T \boldsymbol{U}_k \boldsymbol{V}_k^T \boldsymbol{y} = \mathbb{1}^T (\boldsymbol{U}_k^T \boldsymbol{x} \odot \boldsymbol{V}_k^T \boldsymbol{y}), && (3.3)
\end{aligned}
$$

where $\odot$ denotes an element-wise product and $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$ are low rank decomposition of $\boldsymbol{W}_k$. By keeping K very small, Yu et al. (2018c) manage to keep the computation tractable by using simple matrix multiplication, vector reshaping and sum-pooling operations. Kim et al. (2017a, 2018) then further reduce this equality by sharing the parameters of $\boldsymbol{U}$ and $\boldsymbol{V}$ over the k dimension, falling back to vanilla late fusion mechanisms. Although a very rough approximation, it turns out to work very well in practice when carefully tuned. Finally, Ben-Younes et al. (2017, 2019) either performed a low-rank tensor factorization of $\boldsymbol{W}$ through a Tucker decomposition, or a more general block-term decomposition which split $\boldsymbol{W}$ into a sum of sparse block-tensors. The authors then show that the bilinear transformation could be approximated by a sequence of outer products in a low dimensional space.

Spatial Attention Mechanisms Spatial attention extends machine translation attention mechanisms (Bahdanau et al., 2015; Luong et al., 2015) introduced in Sec 2.7 to computer vision. While the original attention module computes a weighted sum of RNN hidden states, spatial attention computes weighted sum of pixel-wise activations as shown in Fig. 3.3. More formally, given a conditioning embedding $\boldsymbol{x} \in \mathbb{R}^N$ and the image feature map $\boldsymbol{F}_{w,h,c}$ where w, h, c are the width, height, and channel indices, we obtain a final visual embedding $\boldsymbol{z}$ as follows:

$$
\xi_{w,h} = MLP(g(\boldsymbol{F}_{w,h,\cdot}, \boldsymbol{x})) \;; \quad \alpha_{w,h} = \frac{\exp(\xi_{w,h})}{\sum_{w',h'} \exp(\xi_{w',h'})} \quad ; \; \boldsymbol{z} = \sum_{w,h} \alpha_{w,h} \boldsymbol{F}_{w,h,\cdot}, \tag{3.4}
$$

[2] given the input $\boldsymbol{x} \in \mathbb{R}^N$, the count sketch projection outputs a vector $\bar{\boldsymbol{x}} \in \mathbb{R}^D$ where $N \gg D$. It first initializes two random vectors $\boldsymbol{s} \sim U\{1, -1\}^N$ and $\boldsymbol{h} \sim U[0, \ldots, D-1]^N$, then uses $\boldsymbol{h}$ a look-up table to compute $\bar{\boldsymbol{x}}$ as follows: $\bar{x}[h[i]] = \bar{x}[h[i]] + s[i]x[i]$ where $i \in [0..N-1]$ and $\bar{\boldsymbol{x}}$ is initialized as a zero vector.

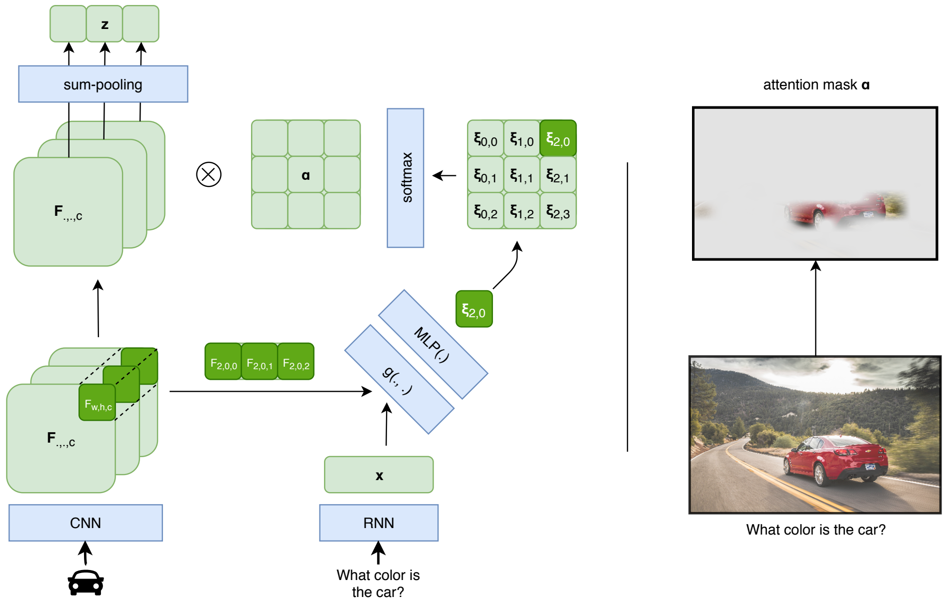


Figure 3.3: Spatial attention mechanism. Pixel-wise features $\boldsymbol{F}_{w,h,\cdot}$ are first merged with the linguistic embedding to generate a mask $\boldsymbol{\xi}$. This mask is then normalized thought a softmax to get $\boldsymbol{\alpha}$ before being applied to the feature-maps F. Finally, a sum-pooling is performed for each feature map over the spatial dimension. In the literature, the attention mechanism differs in the definition of $g(.,.)$ and the MLP to a lesser extent.

where $MLP(.)$ is a multi-layer perceptron and $g(.,.)$ is an arbitrary late-fusion mechanism (concatenation, element-wise product, bilinear transformation, etc.). For instance, Multimodal Compact Bilinear (MCB) attention (Fukui et al., 2016) defines $g(.,.)$ as follows:

$$g(\boldsymbol{F}_{w,h,\cdot}, \boldsymbol{x}) = \mathrm{FFT}^{-1}(\mathrm{FFT}(h(\boldsymbol{F}_{w,h,\cdot})) \odot \mathrm{FFT}(h(\boldsymbol{x}))), \tag{3.5}$$

where $h(.)$ is a sketch projection, and the joint representation is followed by a two-layer $MLP(.)$ with ReLU hidden activation. Similarly, Multimodal Low-rank Bilinear (MLB) attention (Kim et al., 2017a) appends $\tanh$ non-linearity to Eq. 3.3 and defines $g(.,.)$ as follows:

$$g(\boldsymbol{F}_{w,h,\cdot}, \boldsymbol{x}) = \tanh(\boldsymbol{U}^T \boldsymbol{F}_{w,h,\cdot}) \odot \tanh(\boldsymbol{V}^T \boldsymbol{x}), \tag{3.6}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are trainable weight matrices, and the subsequent MLP is a single layer with $\tanh$ activation. We also list below other remarkable spatial attention variants:

- Stacked Attention Networks (SAN) perform several spatial attention hops to iteratively refine the attention mask $\boldsymbol{\alpha}$ (Yang et al., 2016). Given the query $\boldsymbol{x}$, a first attention pass is performed over the image feature $\boldsymbol{F}$ to retrieve the visual embedding $\boldsymbol{z}$, the query $\boldsymbol{x}$ and the visual embedding $\boldsymbol{z}$ are then merged to generate a new query vector $\boldsymbol{x}'$, and the attention process is repeated to obtain a new visual embedding $\boldsymbol{z}'$ etc.

- Hard-Attention computes a sparse weighted sum of pixel-wise activations by setting some $\alpha_{w,j}$ to zero through thresholding or sum truncation. Hard-attention improves computational efficiency by only processing the most relevant information at the expense of creating a non-differentiable neural block. As a result, it either requires to optimize a second loss with REINFORCE (Xu et al., 2015), or using adaptive ReLU (Malinowski et al., 2018b).

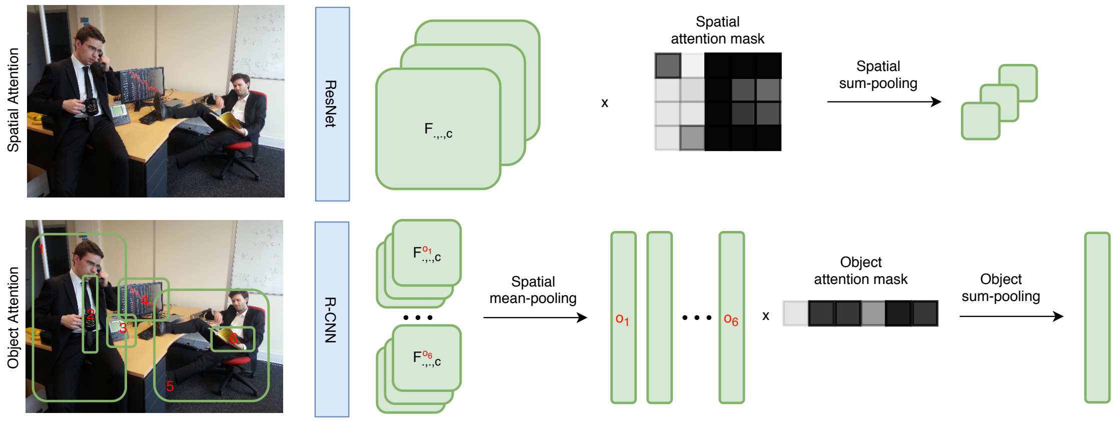


Figure 3.4: Difference between spatial attention and Object-Feature attention. (top) In spatial attention, image features are extracted from a classification network (e.g. ResNet), and the attention mask is performed over the spatial dimension. (bottom) In object-feature attention, object visual features are extracted from a object detection network (e.g. R-CNN), and are directly pooled into a $1D$-dimensional representation before applying an attention mask over the object dimension.

- Relational Networks attend to every pair of pixels for a query vector, the resulting vectors are then summed to have a fixed-length representation. Formally, relational networks introduce a function $g(.,.,.)$ whose two first terms iterate over the spatial location, and the third is fed with the query vector. Despite a polynomial complexity, relational vectors successfully deal with several spatial reasoning tasks (Santoro et al., 2017; Zambaldi et al., 2019).

In visually-grounded language tasks, the query vector $\boldsymbol{x}$ is generally the last hidden state of a RNN (Kim et al., 2017a; Yang et al., 2016) or the result of a self-attention mechanism (Yu et al., 2018c). However, it is also possible to perform a co-attention mechanism that jointly attends to the full linguistic sequences $[\boldsymbol{x}_t]_{t=1}^{T}$ and the feature maps $\boldsymbol{F}$. In a seminal work, Xu et al. (2015) perform image captioning by computing a new spatial attention mask for every new generated word. In VQA tasks, Jiasen et al. (2016) examine a parallel attention mechanism that first computes a joint query embedding by fusing the linguistic and visual embedding with mean-pooling before attending to both modalities with the resulting query vector. In a second experiment, the authors study a co-attention mechanism that first attends to the language embedding to generate a textual query, followed by a spatial attention to generate a low-dimensional visual representation. Several follow-up works extend such co-attention mechanisms to visual dialogue settings, where language embedding are naturally broken down into multiple representations (Gan et al., 2019; Wu et al., 2018b; Zhuang et al., 2018).

Finally, some neural architectures are only composed of multimodal attention modules such as visual memory networks (Xiong et al., 2016), or the Memory,Attention, and Composition (MAC) networks (Hudson and Manning, 2018), which rely on several attention blocks to attend to linguistic, visual and internal memory representations.

Object-Feature Attention Object-Feature attention builds upon object detection networks, which output a list of salient image regions as pooled convolutional feature vectors, and computes weighted sum of the object embeddings as depicted in Fig. 3.4.

In more details, a pretrained object detection networks (e.g. Faster R-CNN (Ren et al., 2015b)) first returns a pool of K object visual features ($K < 100$). In a second step, we compute the weighted sum of object features conditioned on a query vector. Noticeably, the first step is sometimes referred as bottom-up attention while the latter is named as top-down attention (Anderson et al., 2018b). Formally, given a conditioning embedding $\boldsymbol{x}$, a list of I visual object embedding $[\boldsymbol{o}_i]_{i=0}^{I-1}$ where $\boldsymbol{o}^i \in \mathbb{R}^N$, we obtain a final visual embedding $\boldsymbol{z}$ following Eq. 3.7:

$$\xi_i = MLP(g(\boldsymbol{o}_i, \boldsymbol{x})) \; ; \quad \alpha_i = \frac{\exp(\xi_i)}{\sum_{i'} \exp(\xi_{i'})} \quad ; \; \boldsymbol{z} = \sum_i \alpha_i \boldsymbol{o}_i, \tag{3.7}$$

where $g(.,.)$ is a late-fusing mechanism similar to the ones described in spatial attention mechanism. For instance, Anderson et al. (2018b) use a simple concatenation followed by a non-linear projection with $\tanh$ activation, and Kim et al. (2018) extend low-rank bilinear approximation to object-feature attention.

Differently, Shrestha et al. (2019) replace the weighted-sum by using a bidirectionnal GRU to process the object embedding. In more details, the authors concatenate object-features with a linguistic embedding, fuse them with several non-linear projections with residual connection, and use the last RNN states as a joint embedding.

Many state-of-the-art models in language-vision tasks rely on R-CNN features, e.g. the VQA winners (Anderson et al., 2018c; Jiang et al., 2018), object localization models (Yu et al., 2018b), or visual dialogue architectures (Gan et al., 2019; Kang et al., 2019; Kim et al., 2018), and R-CNN features are slowly supplanting ResNet-like features.

Modulation Mechanism When starting the Ph.D, middle conditioning mechanisms have drawn little attention in the context of deep learning architecture as defined in Sec 3.1.2. One of the main contributions of this thesis is thus to design and formalize this so-called modulation mechanism to condition deep network's middle stage. We thus thoughtfully explore this concept in Chapters 5, 6 and 7.

Hypernetworks Hypernetworks are an original conditioning mechanism that recently emerged in the literature (Ha et al., 2016). While previous methods design neural blocks to fuse modalities into a joint space, hypernetworks use the first modality to generate the weights of another network that processes the second modality. Formally, given two modalities $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$, a multimodal representation $\boldsymbol{z} \in \mathcal{Z}$, we define the primal network as $f_{\boldsymbol{\theta}} : \mathcal{Y} \rightarrow \mathcal{Z}$ which is parametrized by a vector $\boldsymbol{\theta} \in \Theta$ and the hypernetwork as $h_{\phi} : \mathcal{X} \rightarrow \Theta$ which is parametrized by a trainable weight vector ϕ. The hypernetwork first predicts the parameters $\boldsymbol{\theta}$ which are then used by the primal network, such as:

$$f_{h_{\phi}(\boldsymbol{x})}(\boldsymbol{y}) = \boldsymbol{x} \tag{3.8}$$

The attentive reader would notice that this mathematical decomposition is an artificial composition of functions which can be factorized in a classic fusion mechanism where $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{Z}$; as a result, the gradient can be backpropagated through both networks.

From a historical perspective, hypernetworks were first designed to reduce the memory footprint of large networks (Ha et al., 2016; Schmidhuber, 1992) or to quickly generalize over a set of tasks in a meta-learning fashion (Bertinetto et al., 2016; Ravi and Larochelle, 2017; Schmidhuber, 1987). For example, Bertinetto et al. (2016) perform one-show classification by training a network to produce the weights of a binary-classifier given one single example, entrusting the newly generated network to detect similar samples.

To the best of our knowledge, hypernetworks have only recently been explored in multimodal learning. For instance, adaptive CNN predicts various convolution filters as a function of auxiliary inputs like camera perspective, level of noise, etc. (Kang et al., 2017). In language-vision tasks, Ba et al. (2016) perform zero-shot learning by predicting convolutional filters and classifiers weights based on textual descriptions of object classes. The same idea was applied to VQA to either produce the last hidden layer of the classifier (Noh et al., 2016), attention modules Seo et al. (2017), or intermediate convolution filters (Gao et al., 2018) conditioned on the ongoing question. Finally, modulation can be assimilated to hypernetworks as the batch normalization parameters are conditioned by linguistic features instead of being trained by backpropagation as we explore in the manuscript.

3.2 Major Visually-Grounded Tasks

Until now, we have been studying multimodal learning from a generic perspective, we now specifically focus on vision-language tasks. Thus, we first outline the prominent recent tasks and datasets that have blossomed over the past five years shown in Fig. 3.5, before mentioning the deep neural models that were subsequently developed.

Image captioning As the name suggests, image captioning consists of generating coherent and factual descriptive statements about images, e.g., *"Two elephants crossing a road in the forest"*, *"A boy and a girl watching a sport TV-show"*. This translation task has witnessed numerous research works over the years, slowly drifting from specific topic descriptions (e.g., house picture descriptions (McDonald and Conklin, 1982) to key-word descriptions (Barnard et al., 2003) up to free-form captions (Feng and Lapata, 2010; Vinyals et al., 2015b). Similar to traditional computer vision tasks, the first captioning datasets only contained a few thousand image-caption pairs (Everingham et al., 2015; Müller et al., 2012), and internet paves for the emergence (and the need) for large-scale datasets and image-captioning models (Feng and Lapata, 2010; Lew et al., 2006). Soon after, Ordonez et al. (2011) released the Im2Text large-scale dataset containing 1 million images from captions collected on Flickr[3]. However, the dataset turned out to be as large as it was noisy despite the best efforts of the authors. Chen et al. (2015); Lin et al. (2014) and Young et al. (2014) later released the MS Coco and the Flickr30 dataset, which "only" consists of 160k and 30k images with 800k and 150k captions. To do so, the MS Coco team hired mechanical turkers to obtain multiple high-quality captions per image. Those large datasets also used acknowledged linguistics scores to automatically and standardized benchmark models, e.g. BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005). In a second step, these metrics were supplanted by new captioning scores that better correlate with human judgment such as CIDEr (Vedantam et al., 2015) or SPICE (Anderson et al., 2016). Recently, the Visual Genome (Krishna et al., 2017) and Flickr30 entities (Plummer et al., 2015) datasets shift the classic image captioning task from global descriptions to region image description (also called dense captioning), containing 100k and 30k images and 5M and 276k region captions. Deep Learning models quickly lead the way, and Karpathy and Fei-Fei (2015); Vinyals et al. (2015b) designed a new deep multimodal encoder-decoder approach inspired by the recent successes in machine translation (Cho et al., 2014b; Sutskever et al., 2014). In this scenario, a pretrained ConvNet first extracts a fixed-size image representation, and a second RNN translates it into a textual description in a Seq2Seq fashion. This approach was then refined by integrating attention mechanisms (Anderson et al., 2018b; Lu et al., 2017b; Xu et al., 2015), exploring different decoder architectures (Aneja et al., 2018; Wang et al., 2016a), integrating R-CNN encoders (Anderson et al., 2018b) or using rein-

[3] https://www.flickr.com/

Image captioning:
∗ Two businessmen having a conference call in an office
∗ Two Ph.D students acting like businessmen in a room
VQA:
∗ Is the man on the right wearing a tie? *No*
∗ How many people are standing? *2*
ReferIt:
∗ The man having a phone call
∗ The man on the left holding a coffee
GuessWhat?!:
- Is it on the office table? *No*
- Is he a person? *Yes*
- The one drinking coffee? *Yes*

VisDiag:
Two traders working in a meeting room
- What are they wearing? *white shirt and black suit*
- Are they both sitting? *No, one is standing in the front*
- What is he doing? *Drinking a coffee*

Figure 3.5: Examples of language and vision-tasks, where the target object is surrounded by a green bounding box (GuessWhat?! and ReferIt). Image captioning provide global descriptions of the image, VQA asked open-form question, ReferIt returns an ambiguous object description in the image, GuessWhat?! aims at locating the person and VisDiag aims at asking questions to imagine the image content. In VQA and GuessWhat?!, the answer module performs a classification tasks. In image captioning and VisDiag, the answers module produces free form-text. Finally, ReferIt is either a classification task (selection within a set of objects), or free-form spatial localisation (bounding box, object segmentation etc.)

forcement learning to fine-tune the network (Liu et al., 2017; Rennie et al., 2017). As studied in this thesis, those captioning models were extended to other image-to-text translation tasks, and we use them while tackling visual dialogue systems in Chapter 4.

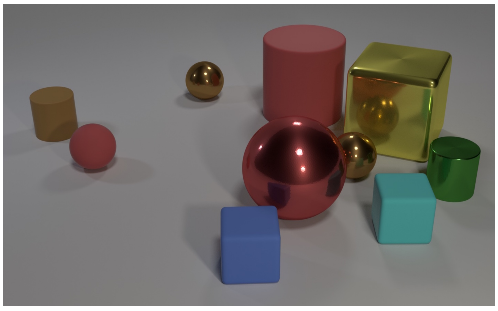
Visual Question Answering VQA requires answering free-form questions given an image (e.g., *"How many zebras are there in the picture?"*, *"Is it raining outside?"*). This fusion task has often been depicted as a Visual Turing Test as it requires a global understanding of the image to answer questions (Gao et al., 2015; Geman et al., 2015; Malinowski, 2017). Recently, the VQA challenge (Antol et al., 2015; Goyal et al., 2017) has provided a new dataset far bigger than previous attempts (Geman et al., 2015; Malinowski and Fritz, 2014). In its original version, the VQA dataset contains 750k open-ended questions on 250k different images from the MS Coco dataset, followed by several variants. Gao et al. (2015) concurrently released a VQA Chinese/English dataset containing 150k images and 310 questions-answer pairs. (Antol et al., 2015; Zhang et al., 2016a) developed a cartoon engine to generate subtle changes in scene representations leading to high semantic consequences (e.g., flipping a running character would change a caption from *running away* to *cheering*). Visual7W (Zhu et al., 2016) uses object localization to guide the mechanical turkers towards asking object-grounded questions. FVQA extends the visual questions with textual facts to integrate common sense reasoning (Wang et al., 2018). Finally, the Visual Genome dataset (Krishna et al., 2017) released 1.7M question-answer pairs over 100k images, scaling up further the VQA field. An extensive body of deep learning models has followed, largely building on the image captioning literature (Fukui et al., 2016; Jiasen et al., 2016; Kim et al., 2017a; Yang et al., 2016). However, models were observed to report the same answer to a question irrespective of the image, suggesting that they largely exploit predictive correlations between questions and answers present in the dataset (Agrawal et al., 2016; Kafle and Kanan, 2017). This observation led to the second generation of VQA datasets that tried to anneal those biases, trying to re-balance the dataset distribution. Goyal et al. (2017) collected complementary images to balance the image-answer distribution of the original VQA dataset. C-VQA reshuffled the VQAv1 training/testing dataset to enforce a mismatch between the answer distribution in both sets; this modification allowed the authors to detect models that mostly learn language biases (Agrawal et al., 2018). TDIUC carefully merged several VQA datasets and generated new artificial questions from visually annotated images (Kafle and Kanan, 2017).

Current state-of-the-art systems often use the following computational pipeline: they first extract image features from a pretrained convolutional network such as classication networks (e.g. ResNet, VGGNet), or a object detection networks (e.g. R-CNN). In parallel, a language embedding is extracted with a RNN over word embeddings. Both modalities are then fused with multiple conditioning methods: mid-late conditioning involves spatial attention (Fukui et al., 2016; Gao et al., 2018; Jiasen et al., 2016; Kim et al., 2017a; Malinowski et al., 2018a; Yu et al., 2018c) or R-CNN feature-map selection (Anderson et al., 2018b; Cadene et al., 2019; Kim et al., 2018; Shih et al., 2016; Shrestha et al., 2019; Wu et al., 2018a). Finally, a late conditioning is performed through simple projections, concatenations and element-wise products (Antol et al., 2015; Jiasen et al., 2016; Kim et al., 2017a; Malinowski et al., 2015), or by approximating bilinear mechanisms (Ben-Younes et al., 2017; Cadene et al., 2019; Fukui et al., 2016). We also investigate modulation for middle conditioning architectures in Chapter 5, 6 and 7. While the mentioned models were detailed in Sec. 3.1.3, we encourage the reader to look at the VQA challenge winners feedback for low-level training tricks (Jiang et al., 2018; Teney et al., 2018). Unfortunately, the VQA challenge had also unexpected damaging side effects by sometimes insidiously driving heavy engineering over model novelties.

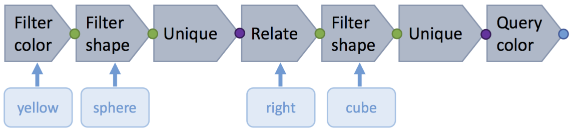
Artificial Visual Question Answering Synthetic VQA datasets were concurrently developed to alleviate some of the described VQA constraints. Besides removing the burden of collecting large scale datasets, synthetic tasks are designed to anneal potential human cognitive bias and allow to assess original research directions better. Synthetic datasets are also less sensitive to a state-of-the-art escalation that sometimes prevailed in competition. However, they are also hard to design, and the underlying task difficulty is equally hard to estimate; it is easy to end with unfeasible scenarios, degenerate solutions, or to result in problem-solving researches that overfit on toy problems.

The CLEVR dataset Johnson et al. (2017a) has been the most studied (and successful) artificial dataset in visually-grounded language tasks by a large margin. It is a synthetic dataset of 700K (image, question, answer, program) tuples whose images contain 3D-rendered objects of various shapes, materials, colors, and sizes, and questions are multi-step and compositional in nature. They range from counting questions (*"How many green objects have the same size as the green metallic block?"*) to comparison questions (*"Are there fewer tiny yellow cylinders than yellow metal cubes?"*). Similar datasets include 2D-image rendering-based datasets, e.g, the Sort-of-CLEVR (Santoro et al., 2017), SHAPES (Andreas et al., 2016b), ShapeWorld (Kuhnle et al., 2018), SQOOP (Bahdanau et al., 2019b), abstract scenes (Zhang et al., 2016a) or 3D world with Minecraft scenes (Yi et al., 2018). Recently, GQA bridge the gap between artificial and natural datasets by generating synthetic language on top of natural images (Hudson and Manning, 2019). The authors built upon the Visual Genome images (Krishna et al., 2017), and construct filling question-templates through annotated object properties and relationships.

The CLEVR task spurred new deep learning approaches that focused on visual reasoning. Partly inspired by Neural Module Network (NMN) (Andreas et al., 2016a,b), a first approach integrates (and generates) structured program operations into a neural reasoning architecture to guide the learning (Hu et al., 2017b; Johnson et al., 2017b; Suarez et al., 2018). For instance, Johnson et al. (2017b) implements a sequence-to-sequence Program Generator, which takes in a question and outputs a sequence corresponding to a tree of composable neural modules. This tree of neural modules is assem-



- Are there an equal number of large things and metal spheres?
- There is a sphere with the same size as the metal cube; is it made of the same material as the small red sphere?
- How many objects are either small cylinders or red things?



- What color is the cube to the right of the yellow sphere?

Figure 3.6: Questions in CLEVR test various aspects of visual reasoning including attribute identification, counting, comparison, spatial relationships, and logical operations. Illustration are from Johnson et al. (2017a). The fourth question is accompanied with the underlying reasoning program generated by the CLEVR-engine. This engine was extended to the ReferIt (Liu et al., 2019a) and VisDiag (Kottur et al., 2019) setting.

bled to form an execution engine that predicts an answer from the image. Other approaches discard the program supervision and train networks without the need for handcrafted intermediate representations. The resulting leading CLEVR methods include the MAC networks (Hudson and Manning, 2018; Marois et al., 2018), Relational Networks (Santoro et al., 2017), the use of hyper-networks for spatial attention (Gao et al., 2018), graph networks (Haurilet et al., 2019), modulation mechanisms as explored in Chapter 6 and subsequent multi-hop variants (Yao et al., 2018). As both approaches lead to similar performance, it is still an open issue whether the program supervision is required to learn visual reasoning with deep learning models. Concurrently, some of the dominant CLEVR architectures are observed to not scale-up to real images (Shrestha et al., 2019; Suhr et al., 2018), and simple but highly competitive CLEVR models were discovered (Malinowski and Doersch, 2018), pointing out again the limitation of synthetic datasets. In light of these observations, the CLEVR task is now perceived as a necessary but not sufficient benchmark to assess the generality of visual reasoning methods, and following synthetic tasks were released to palliate this limitation and answer other research questions (Hudson and Manning, 2019).

ReferIt Visual referring expression task, aka ReferIt, is a cooperative two-player game at the crossroad between image-captioning and VQA (Gkatzia et al., 2015; Kazemzadeh et al., 2014). The first player selects an object in a rich visual scene, for which they must generate an expression that refers to it (e.g., *the person eating ice cream*). Based on this expression, the second player selects an object within the image. Four ReferIt datasets co-exist in the literature: RefClef (Kazemzadeh et al., 2014), RefCOCO/RefCOCO+ (Yu et al., 2016a) and RefCOCOg (Mao et al., 2015). The original RefClef dataset uses the IMAGEClef dataset (Escalante et al., 2010) with 30K references over 20K images, while the three recent extensions are built on top of MS COCO (Lin et al., 2014) with respectively 142k, 142k and 86k references over 20k, 20k and 27k images. Subtle differences exist between datasets e.g., RefCOCO+ forbids certain words to prevent object references from being too simplistic, while RefCOCOg only relies on images containing 2-4 objects from the same category. More recently, a synthetic version of ReferIt has been released on top of the CLEVR game engine (Liu et al., 2019a). Finally, the Visual Query Detection dataset extends the ReferIt game to multi-choice object detection, allowing sentences such as *the red-cap children* (Acharya et al., 2019).

From a modeling perspective, the object retrieval task is divided into the multimodal fusion and the object selection parts. In the fusion step, the historical strategy incorporates hand-crafted visual modules that are tailored for ReferIt, and combines within a scoring fusion mechanism (Conti et al., 2018; Hu et al., 2017c; Kazemzadeh et al., 2014; Nagaraja et al., 2016; Yu et al., 2018b). For instance, Modular Attention Network (MAttNet) (Yu et al., 2018b) combines the scores of: the subject module specialized in categories and attributes perception, the localization module specialized in processing absolute and relative object position, and the relationship module that learns how objects visually refer to each other. Other fusing strategies involve a classic VQA pipeline that merges language and image features into a single network through concatenation (Yu et al., 2016a), RNN (Hu et al., 2016b), co-attention (Rohrbach et al., 2016; Zhuang et al., 2018), or modulation as in Chapter 7. The object selection step alternates between ranking object crops (Conti et al., 2018; Yu et al., 2018b; Zhuang et al., 2018), outputting spatial coordinates (Rohrbach et al., 2016), and directly segmenting the target object (Hu et al., 2016a; Li et al., 2018). The sentence generation is mostly based on the image-captioning literature (Johnson et al., 2016), where the loss is sometimes modified to discriminate referring expressions within the same image (Mao et al., 2015). Differently, Luo and Shakhnarovich (2017); Yu et al. (2016b) use reinforcement learning to train the object retrieval and description generation models iteratively. Finally, there have been some efforts to cast the ReferIt game into a semi-supervised and unsupervized setting (Mao et al., 2015; Rohrbach et al., 2016).

Visually-grounded dialogues Visual dialogues are goal-oriented dialogues where the agent acquires natural language by discussing over visual cues. When starting the Ph.D., visually grounded language tasks were mostly static as they align one sentence, e.g., questions or captions, with an image; we thus start developing visual dialogues to integrate the dynamic and interactive facets of language learning. In practice, visual dialogue may cast the historical language-vision tasks such as image-text retrieval, VQA, or ReferIt into a dialogue setting. From a vision perspective, such an approach provides a fine-grained interaction with visual systems, making an extra step toward visual comprehension. From a dialogue perspective, visual cues are an alternative to the (non-differentiable) knowledge-base, which are the backbone of various conversational agents. The interactive nature of visual dialogues also encourages exploring reinforcement learning techniques for language. Finally, visual dialogues are an ideal test-bed for assessing visually grounded language learning theories. Note that we further discuss this line of research in Chapter 4 while introducing the GuessWhat?! visual dialogue task.

Other tasks There exists a large panel of other vision-language tasks that we do not explore in this thesis but share similar research reflections. For instance, visually grounded semantic (Baroni, 2016) aims at learning transferable language embedding by incorporating visual cues, learning multimodal representation at the word level (Baroni et al., 2014; Kiela, 2017; Kottur et al., 2016; Lazaridou et al., 2015b; Utsumi, 2018), or sentence level (Kiela et al., 2018). There is also an effort towards generating tasks for assessing visual and linguistic model understanding e.g., Natural Language Visual Reasoning (NLVR) (Suhr et al., 2017, 2018) determine whether a caption is right with regard to a pair of images, phrase grounding explicits the alignment between image components and caption words (Anne Hendricks et al., 2018; Datta et al., 2019; Dogan et al., 2019), and the well-design FoilIt! Dataset involves detecting and fixing incorrect image captions (Ling and Fidler, 2017; Shekhar et al., 2017). Visual Storytelling Dataset (Huang et al., 2016) contains annotated sequences of images with individual image captioning, contextualized image descriptions, and entertaining stories within the sequence of images. Static image analysis has also been extended to dynamic visual perception by summarizing videos (Rohrbach et al., 2017; Sanabria et al., 2018; Xu et al., 2016), performing video question answering (Alamri et al., 2019; Jang et al., 2017; Lei et al., 2018), and designing virtual interactive world for embodied question answering (Das et al., 2018b,c; Gordon et al., 2018) or instruction following (Anderson et al., 2018c; Gargett et al., 2010; Savva et al., 2017). These tasks and datasets often have some variants with different versions, and even some competitors, but it would be

Sisyphean work to fully enumerate the new vision-language tasks that have blossomed in a couple of years. On a more polemic perspective, this profusion may even harm the community as it takes the research away from either practical applications or more fundamental questions by focusing on difficult but still toy problems. Yet, this gentle point has also been a recurrent pitfall of the AI community over the years[4] and we are confident that it should decant as time goes.

[4] *[the paper proposal] is not to create a problem-solving program, or a natural language comprehension system with the representation as target. It is tempting to make such demonstrations from time to time. (They impress people; and it is satisfying to have actually made something which works, like building model railways; and one's students can get Ph.D.'s that way.) But they divert attention from the main goal. In fact, I believe they have several more dangerous effects. It is perilously easy to conclude that, because one has a program which works (in some sense), its representation of its knowledge must be more or less correct (in some sense). Now this is true, in some sense. But a representation may be adequate to support a limited kind of inference, and completely unable to be extended to support a slightly more general kind of behaviour. It may be wholly limited by scale factors, and therefore tell us nothing about thinking about realistically complicated worlds. Images as internal pictures and the STRIPS representation of actions by add and delete lists are two good examples.* (Hayes, 1978)

Part II

Multimodal Learning for Visually Grounded Language

Chapter 4

Visually Grounded Dialogue: GuessWhat?!

«Bip Bip Bip Bip »

R2D2

Contents

As mentioned in the previous chapters, we explore the interactive visual dialogue setting as a playground for visually grounded language learning. In this chapter, we thus introduce GuessWhat?!, a two-player guessing game whose goal is to locate an unknown object in a rich image scene by asking a sequence of questions. Higher-level image understanding, like spatial reasoning and language grounding, is required to solve the proposed task. Our key contribution is the collection of a large-scale dataset consisting of 150K human-played games with a total of 800K visual question-answer pairs on 66K images. We explain our design decisions in collecting the dataset and introduce the

Oracle and questioner tasks that are associated with the two players of the game. We finally prototype deep learning models to establish initial baselines of the introduced tasks, which will be used alongside the manuscript.

4.1 Introducing GuessWhat?!

4.1.1 Underlying Motivations

People use natural language as the most effective way to communicate, including when it comes to describe the visual world around them. They often need only a few words to refer to a specific object in a rich scene. Whenever such expressions *unambiguously* point to one object, we speak of a referring expression (Krahmer and Deemter, 2012). However, uniquely identifying the referred object is not always possible, as it depends on the listener's state of mind and the context of the scene. Many real life situations, therefore, require multiple exchanges before it is clear what object is referred to:

- Did you see that dog?
* You mean the one in the corner?
- No, the one that's running.
* Yes, what's up with that?

A computer vision system able to hold conversations about what it *sees* would be an important step towards intelligent scene understanding. Such systems would be more transparent and interpretable because humans may naturally interact with them, for example by asking clarifying questions about what it perceives. Still, a fundamental challenge remains: how to create models that understand natural language descriptions and ground them in the visual world.

As detailed in Chapter 3.2, the last few years have seen an increasing interest from the computer vision community in tasks towards this goal. Thanks to advances in training deep neural networks (Goodfellow et al., 2016) and the availability of large-scale classification datasets (Krishna et al., 2017; Lin et al., 2014; Russakovsky et al., 2015; Zhou et al., 2014), automatic object recognition has now reached human-level performance (LeCun et al., 2015). As a result, attention has been shifted toward tasks involving higher-level image understanding.

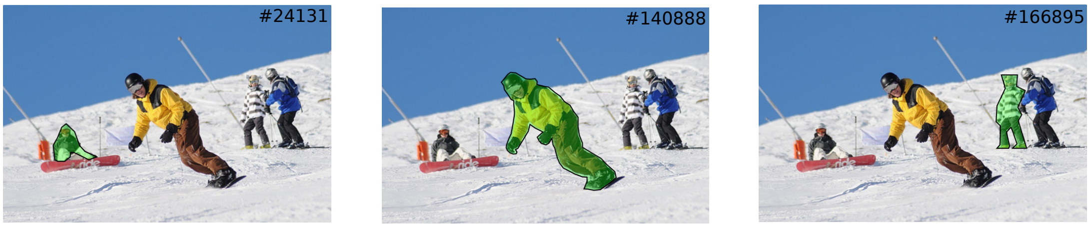
On the other hand, there has been a renewed interest in dialogue systems (Lemon and Pietquin, 2012; Serban et al., 2018), inspired by the success of data-driven approaches in other areas of natural language processing (Cho et al., 2014b; Vinyals and Le, 2015). Traditionally, dialogue systems have been built through heavy engineering and hand-crafted expert knowledge, despite machine learning attempts for almost two decades (Levin and Pieraccini, 1997; Singh et al., 1999). One of the difficulties comes from the lack of automatic evaluation as – contrary to machine translation – there is no evaluation metric that correlates well with human evaluation (Liu et al., 2016). A promising alternative is goal-directed dialogue tasks (Lemon and Pietquin, 2012; Singh et al., 1999; Wen et al., 2016; Weston et al., 2016) where agents converse to pursue a goal rather than casually chit-chat. The agent's success rate in completing the task can then be used as an automatic evaluation metric. Many tasks have recently been introduced, including the bAbI tasks (Weston et al., 2016) for testing an agent's ability to answer questions about a short story, the movie dialog dataset (Dodge et al., 2016) to assess an agent's capabilities regarding personal movie recommendation and a Wizard-of-Oz framework (Wen et al., 2016) to evaluate an agent's performance for assisting users in finding restaurants.

Is it a person? *No*
Is it a cat? *Yes*
Is it in the hands of these girls? *No*
The cat in the right side of the image? *Yes*

Is it human? *Yes*
Are they five humans visible? *Yes*
Is it leftmost? *No*
Is it in the middle? *Yes*
Is it the third one from the left? *Yes*

Is it an ear? *No*
Is it a horn? *No*
Is it a cow? *Yes*
Is it the one at the back ? *No*
Does it have a horn? *Yes*

Figure 4.1: Three examples of our dataset.



Is it a person? *Yes*
Is it in the foreground? *No*
Is he wearing blue shirt? *No*
Is he wearing black shirt? *Yes*

Is it a person? *Yes*
Is it standing? *Yes*
Is it in yellow? *Yes*

Is it a person? *Yes*
One in yellow? *No*
One with white pants? *No*
One with blue pants in the top? *No*

Figure 4.2: Three example of our dataset where different objects are picked in the same image.

In this chapter, we bring these two fields together and propose a novel goal-directed task for multi-modal dialogue. The two-player game, called GuessWhat?!, extends the ReferIt game (Kazemzadeh et al., 2014) to a dialogue setting. To succeed, both players must understand the relations between objects and how they are expressed in natural language. From a machine learning point of view, the GuessWhat?! challenge is the following: learn to acquire natural language by interaction on a visual task. Previous attempts in that direction (Akinator, 2007; Wen et al., 2016) do not ground natural language to their immediate environment; instead they rely on an external database through which a conversational agent searches.

The key contribution of this chapter is the introduction of the GuessWhat?! dataset that contains 160,745 dialogues composed of 821,889 question/answer pairs on 66,537 images extracted from the MS COCO dataset (Lin et al., 2014). We define three sub-tasks that are based on the GuessWhat?! dataset and prototype deep learning baselines to establish their difficulty. The chapter is organized as follows. First, we explain the rules of the GuessWhat?! game in Sec. 4.1.2. Then, Sec. 4.1.3 describes how GuessWhat?! relates to previous work. In Sec. 4.2.1 we highlight our design decisions in collecting the dataset, while Sec. 4.2.2 analyses many aspects of the dataset. Sec. 4.3 introduces the questioner and Oracle tasks and their baseline models. Finally, Sec. 4.5 provides a final discussion of the GuessWhat?! game.

4.1.2 GuessWhat?! game

GuessWhat?! is a cooperative two-player game in which both players see the picture of a rich visual scene with several objects. One player – the **Oracle** – is randomly assigned an object (which could be a person) in the scene. This object is not known by the other player – the **questioner** – whose goal it is to locate the hidden object. To do so, the questioner can ask a series of yes-no questions which are answered by the Oracle as shown in Fig. 4.1. Note that the questioner is not aware of the list of allowed objects, they can only see the whole picture. Once the questioner has gathered enough evidence to locate the object, they notify the Oracle that they are ready to guess the object. We then reveal the list of objects, and if the questioner picks the right object, we consider the game successful. Otherwise, the game ends unsuccessfully. We also include a small penalty for every question to encourage the questioner to ask informative questions. Figs 4.9 and 4.10 in the supplementary materials 4.5.1 display a full game from the perspective of the Oracle and questioner, respectively.

The Oracle role is a form of visual question answering where the answers are limited to *Yes*, *No* and *N/A* (not applicable). The *N/A* option is included to respond even when the question being asked is ambiguous or an answer simply cannot be determined. For instance, one cannot answer the question "*Is he wearing glasses?*" if the face of the selected person is not visible. The role of the questioner is much harder. They need to generate questions that progressively narrow down the list of possible objects. Ideally, they would like to minimize the number of questions necessary to locate the object. The optimal policy for doing so involves a binary search: eliminate half of the remaining objects with each question. Natural language is often very effective at grouping objects in an image scene. Such strategies depend on the picture, but we distinguish the following types:

Spatial reasoning We group objects spatially within the image scene. One may use absolute spatial information – *Is it on the bottom left of the picture?* – or relative spatial location – *Is it to the left of the blue car?*.

Visual properties We group objects by their size – *Is it big?*, shape – *Is it square?* – or color – *Is it blue?*.

Object taxonomy We can use the hierarchical structure of object categories, i.e. taxonomy, to group objects e.g. *Is it a vehicle?* to refer to both cars and trucks.

Interaction We group objects by how we interact with them – *Can you drive it?*.

The goal of the GuessWhat?! task is to enable machines to understand natural descriptions and ground them into the visual world. Note that such higher-level reasoning only occurs when the scene is rich enough i.e. when there are enough objects in the scene. People otherwise tend to fall back to a linear search strategy by simply enumerating objects (often by their category names).

4.1.3 Related works

The GuessWhat?! game and the data collected from it present opportunities for the visually-grounded learning research community. In the following, we describe previous works in these areas and relate them to the new challenges that GuessWhat?! brings to the language-vision ecosystem. We refer the reader to Sec. 3.2 for extensive details regarding the mentioned tasks.

ReferIt:
∗ Woman in red jacket with green bag
∗ Left woman in red coat

GuessWhat?!:
- Is it a person? ***Yes***
- One of the people with the stroller on the right? ***No***
- One of the two people crossing the street towards us? ***No***
- The woman in red? ***Yes***

ReferIt:
∗ Guy with hat bottom right front
∗ Guy sitting with hat bottom right

GuessWhat?!:
- Is it a person? ***Yes***
- Are they standing? ***No***
- Are they touching the frisbee ? ***No***
- Are they holding a square thing? ***Yes***
- Black cap ? ***Yes***

Figure 4.3: Samples illustrating the difference between GuessWhat?! and ReferIt games. As both datasets are constructed on top of MS COCO, we picked identical objects (and images).

Image captioning Our work builds on top of the MS COCO dataset (Lin et al., 2014) which consists of 120k images with more than 800k object segmentations. While image captioning research uncovered successful approaches to automatically generate coherent, factual statements about images (Feng and Lapata, 2010), GuessWhat?! instead requires to model the process of asking useful questions about images. Although the questioner may be assimilated to an image captioning model, the linguistic pragmatic differs as the questioner intent is to ask a sequence of questions rather than describing the image.

VQA datasets VQA tasks form another well known extension of the captioning task (Antol et al., 2015; Goyal et al., 2017; Lei et al., 2018). They instead require *answering* a open-form question given a picture instead of describing it. The GuessWhat?! game and dataset attempt to circumvent these issues. Because of the questioner's aim to locate the hidden object, the generated questions are different in nature: they naturally favour spatial understanding of the scene and the attributes of the objects within it, making it more valuable to consult the image. Besides, it only contains binary questions, whose answers we find to be balanced and has twice more questions on average per picture. The GuessWhat?! contains questions with co-references requiring several reasoning steps issued from the dialogue nature of the task, which is absent from the VQA tasks.

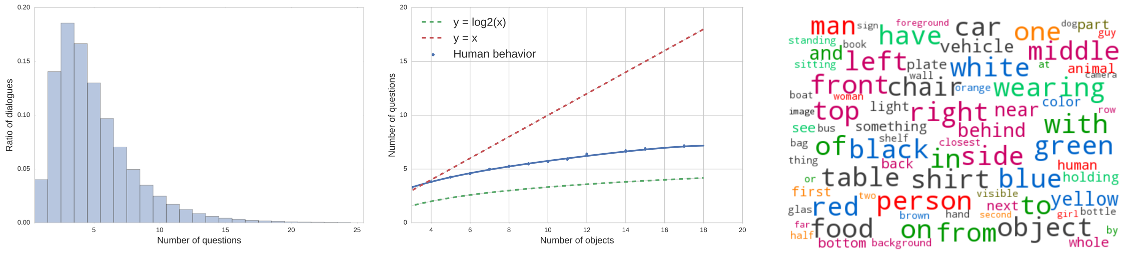
ReferIt Probably closest to our work is the ReferIt game (Bartie et al., 2016; Kazemzadeh et al., 2014; Mao et al., 2015; Yu et al., 2016a). In this game, one player observes an annotated object in a scene, for which they need to generate an expression that refers to it (*e.g. ïhe man wearing the white t-shirt¨*). The other player then receives this expression and subsequently clicks on the location of the object within the image. On a data collection perspective, the recent ReferIt databases select images with only $2-4$ objects of the *same* category. In contrast, GuessWhat?! picks images with $3-20$ objects without further restrictions on the object class, and thus contains three times more images than the ReferIt dataset. To further investigate the difference between ReferIt and GuessWhat?!, we compare two samples for the *same* selected object in Fig 4.3. While ReferIt directly locates the object with a single expression, GuessWhat?! iteratively narrows down the object by means of positive and *negative* feedback on questions. We also observe that GuessWhat?! dialogues favor more abstract concepts, such as "*Is it edible?*" or "*Is it on oval plate?*" than ReferIt.

Eye Spy The robotics community has explored variants of the "Eye Spy" (Parde et al., 2015; Thomason et al., 2016; Vogel et al., 2010) game for grounded language acquisition. In one of the scenarios, the robot is first shown a set of objects that it can poke through some predefined actions (grasp, hold, look, etc.) while recording several modalities (VGGNet features, sounds, joint motor positions, etc.). As a second step, the human describes one of the objects of his choice, and the robot must guess the referenced object. However, the robotic constraints force the game to remain limited to small numbers of objects (4 32), and the training is mainly done in an online fashion with no released dataset. As a result, GuessWhat?! is a promising dataset for transferring knowledge into real-case scenarios (Tremblay et al., 2018).

Dialogue systems A broad range of datasets co-exist in the dialogue system community (Serban et al., 2018). Gigantic datasets are composed of spontaneous and unconstrained dialogues that are extracted from chat logs (Lowe et al., 2015) or micro-blogging platforms (Sordoni et al., 2015). However, it is difficult to assess the quality of such dialogue systems (Liu et al., 2016; Schatzmann et al., 2005) and existing metrics do not always correlate with human evaluations (Elliott and Keller, 2014). Besides, those corpora lack some form of grounding, and the resulting models are prone to inconsistencies, and poor language understanding (Gao et al., 2019; Li et al., 2016b). GuessWhat?! tackles those two difficulties by casting the problem into a goal-oriented setting with visually grounded questions. To be fair, the GuessWhat?! close-form question also limit the range of potential language generalization.

Goal-directed dialogue GuessWhat?! is also relevant to the goal-directed dialogue research community. Such systems are aimed at collaboratively achieving a goal with a user, such as retrieving information or solving a problem. Although goal-directed dialogue systems are appealing, they remain hard to design. Thus, they are usually restricted to specific domains such as train ticket sales, tourist information or call routing (Pietquin and Dutoit, 2006; Singh et al., 1999; Young et al., 2013). Besides, existing dialogue datasets are either limited to fewer than 100k example dialogues (Dodge et al., 2016), unless they are generated with template formats (Dodge et al., 2016; Schulz et al., 2017; Wei et al., 2018; Wen et al., 2016; Weston et al., 2016) or simulation (Pietquin and Hastie, 2013; Schatzmann et al., 2006) in which case they do not reflect the free-form of natural conversations. Finally, recent work on end-to-end dialogue systems fail to handle dynamic contexts. For instance, (Wen et al., 2016) intersects a dialogue with an external database to recommend restaurants. Other well-known game-based dialogue systems (20 Questions, 1988; Akinator, 2007) also rely on static databases, and are neither contextual nor use natural language. In contrast, GuessWhat?! dialogues are heavily grounded by the images. The resulting dialogue is highly contextual and must be based on the content of the current picture rather than an external database. Thus, to the best of our knowledge, the GuessWhat?! dataset marks an important step for dialogue research, as it is the first large scale dataset that is both goal-oriented and multimodal.

Human computation games GuessWhat?! is in line with Von Ahn and Dabbish (2004); Von Ahn et al. (2006)'s seminal work on human computation games who showed that games are an effective way to gather labeled data. The first ESP game (Von Ahn and Dabbish, 2004) was developed to collect image tags and was later extended to Peekaboom (Von Ahn et al., 2006) to gather object segmentation. These games were developed more than a decade ago when object recognition was in its infancy and served a different purpose than GuessWhat?!.



(a) (b) (c)

Figure 4.4: (a) Number of questions per dialogue (b) Number of questions per dialogue vs the number of objects within the picture (c) Word cloud of GuessWhat?! vocabulary with each word proportional to its frequency. Words are colored based on a hand-crafted clustering. Uninformative words such as *"it"*, *"is"* are manually removed.

Visual Dialog Concurrent to our work, the Visual Dialog dataset casts the VQA task into a dialogue setting, where an agent asks a sequence of questions to picture an image (Das et al., 2017a). The underlying dataset contains 133k dialogue, with ten open-form questions by game. Note that we describe further this dataset and compare it with GuessWhat?! in Sec 4.4 to include contemporary works.

4.2 GuessWhat?! Dataset

4.2.1 Data collection

Images We use a subset of the training and validation images and objects of the MS COCO dataset (Lin et al., 2014). We first discard objects that are too small (area $< 500\text{px}^2$) to be decently located by a human observer. Then, we only keep images containing three to twenty objects, to avoid trivial or overly complicated images. In total, we keep 77,973 images with 609,543 objects. We verified that this selection does not significantly alter the original dataset distribution as depicted in Fig 4.13 and Fig 4.14 in Appendix 4.5.2.

Amazon Mechanical Turk The data collection was crowd-sourced on Amazon Mechanical Turk (AMT) (Buhrmester et al., 2011). We created two separate tasks – known as HITs on AMT – for the questioner and Oracle roles, and rewarded the questioner slightly more than the Oracle. We ensured the quality of the data collection by several means. First, the workers had to go through a qualification round which consisted of successfully completing 10 games while producing fewer than 4 mistakes or disconnects. After qualification, HITs continue to consist of a batch of 10 successful games. We incentivize the worker to produce as many successful dialogues in a row by providing bonuses for making fewer mistakes. Secondly, players could report on each other and players were banned after a certain number of reports. Thus, players were incentivized to cooperate. In the end, we only kept dialogues from qualified people and successful dialogues from the qualification round. In contrast to traditional dataset collection, our game requires an interactive session between two players. Fortunately, we found that the GuessWhat?! game was highly engaging. A total of more than 10K people participated in our HITs, and our top ten participants played over 2,000 games each. Since questions were manually typed, they could contain spelling mistakes. Thus, we retrieved all questions containing words that do not occur in an English dictionary and manually corrected the 1000 most common words. For the remaining 30k questions, we created two HITs to correct the spelling mistakes. See Figure 4.11 in Appendix 4.5.1 for further details.

	Full	Finished	Success
# dialogues	160,745	152,000	135,400
# questions	821,889	780,391	672,940
# words	3,985,368	3,788,167	3,254,793
# voc. size	11,464	11,259	10,637
# voc. size (3+)	5,444	5,324	5,013
# images	66,537	66,161	63,642
# segmented objects	535,723	531,847	505,599
# selected objects	134,073	131,415	117,513

Table 4.1: GuessWhat?! statistics split by dataset types.

4.2.2 Data analysis

In the following, we explore properties of the data we collected using the GuessWhat?! game. We provide global statistics, examine the vocabulary used by the questioners and highlight the relationship between properties of objects to guess and the odds of having a successful dialogue.

Dataset statistics The raw GuessWhat?! dataset is composed of 160,745 dialogues containing 821,889 question/answer pairs on 66,537 unique images with 1,385,197 objects and 134,073 unique selected objects. The answers are respectively 52.2% *no*, 45.6% *yes* and 2.2% *N/A*. On average, there are 5.2 questions per dialogue and 2.3 dialogues per image. The dialogues contain 3,985,368 word tokens in total, making up 11,464 different words with at least one occurrence and 5,444 words with at least 3 occurrences. Moreover, 84.2% of the dialogues are successful, 10.3% are unsuccessful and 5.5% are not completed (disconnection, timeout etc.). Thus, different subsets co-exist in the GuessWhat?! dataset, we will refer to the dataset as full, finished and successful when we include all the dialogues, all finished dialogues (successful and unsuccessful) or only successful dialogues, respectively. The previous statistics are broken down into dataset types in Tab 4.1. In practice, we only use to the successful dataset while training and benchmarking the models as it turns out to be less noisy.

Question distributions To get a better understanding of the GuessWhat?! games, we show the number of questions within a dialogue and the average number of questions given the number of objects within a image in Fig 4.4. First, the number of questions within a dialogue decreases exponentially, as players tend to shorten their dialogues to speed up the game (and therefore maximize their gains). More interestingly, we observe that the average number of questions given the number of objects within an image appears to follow a function that grows at a rate between logarithmically and linearly. A questioning strategy of simply listing objects (e.g. "*is it the chair*", etc.) implies linear growth in the number of questions, while the optimal binary search strategy only requires logarithmic growth. Thus the human questioners seem to imply a strategy that is somewhere in between. We conjecture three reasons why humans do not achieve the optimal search strategy. First, the questioner does not have access to the ground truth list of objects in the picture, and might, therefore, overestimate the number of objects. Second, some humans tend to favor a linear search strategy. Finally, the questioner may ask additional questions to confirm that he has located the right object. This can be important in the presence of possible Oracle errors.

Vocabulary To gain insight into the vocabulary used by the questioner, we compute the frequency of words in the GuessWhat?! corpus and display the most frequent words as a word cloud in Fig 4.4c. Several key words clearly stand out. As explained in Sec. 4.1.2, some of those key words refer to

(a) (b) (c)

Figure 4.5: (a-b) Histogram of absolute/relative successful dialogues with respect to the number of objects and the size of the objects, respectively. (c) Evolution of answer distribution clustered by the dialogue length

abstract object properties such as *person* or *object*, spatial locations such as *right/left* or *side* and visual features such as *red/black/white*. Furthermore, prepositions are also heavily used to express relationships between objects. We also compute a co-occurrence matrix of words and show these correlations in Fig 4.12 in Appendix 4.5.2. To better understand the sequential aspect of the questions, we study the evolution of the vocabulary at each question round and look at the occurrence difference to the previous round in Tab. 4.5. We observe that questioners use abstract object properties such as *human/object/furniture* only at the beginning of the dialogues, and quickly switch to either spatial or visual terms such as *left/right*, *white/red* or *table,chair*. This can be highlighted by applying a Dynamic Topic Model (Blei and Lafferty, 2006) to study the evolution of topics over the course of the dialogue as shown in Fig 4.16 in Appendix 4.5.2. As a consequence, there are fewer words introduced over the course of a dialogue as shown in Fig 4.15 in Appendix 4.5.2.

Elements of success To investigate whether certain object properties favour success, we compute the success ratio of dialogues relative to: the size of the unknown objects in Fig 4.5b, the number of objects within the images in Fig 4.5a, the object category, the location of objects within the images and the size of the dialogues in Fig 4.17, Fig 4.18 in Appendix 4.5.2, respectively. As one may expect, the more complex the scene is, the lower the success rate is. When there are only 3 objects, the questioner has 95% success rate, while this ratio drops to around 70% with 20 objects. Similarly, big objects are almost always found while the smallest one are only found 60% of the time. Questioners easily find objects in the middle of the picture but have more difficulties to find them on the border. Finally, objects from categories that are often grouped together, e.g. *bananas* or *books*, have a lower success rates.

Miscellaneous In Fig 4.5c we break down the ratio of yes-no answers within the dialogues. While the first yes-no answers are balanced for small dialogues, they often terminate with a final *yes*. In contrast, long dialogues often start with a higher proportion of negative answers which slowly decrease during the exchange.

4.2.3 Dataset release

We split the GuessWhat?! dataset by randomly assigning 70%, 15% and 15% of the *images* and its corresponding dialogues to the training, validation and test set. This way of dividing the data ensures that we evaluate performance on images not seen during training. The GuessWhat?! dataset and the source code is available at https://guesswhat.ai/download.

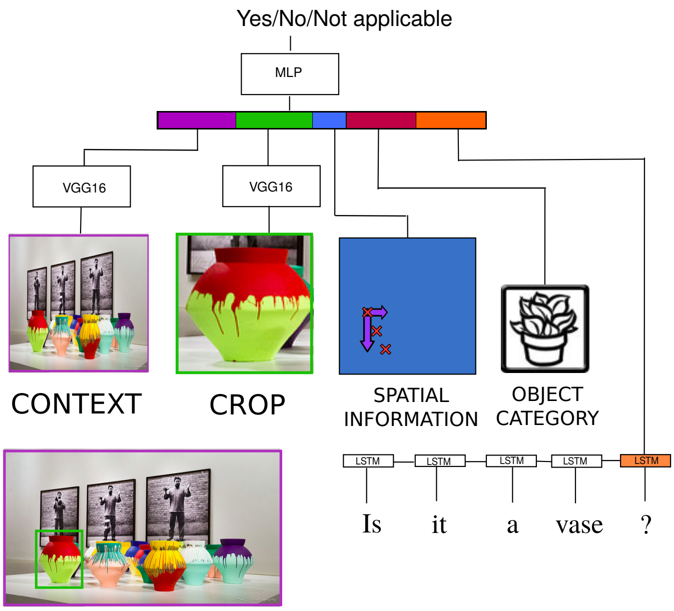


Figure 4.6: An schematic overview of the "Image + Question + Crop + Spatial + Category" oracle model.

4.3 Model Descriptions

We now empirically investigate the difficulty of the Oracle and questioner tasks. To do so, we trained reasonable baselines for each task and measured their performance.

4.3.1 Game Notation

Formally, a GuessWhat?! game revolves around an image $\mathcal{I} \in \mathbb{R}^{W \times H}$ containing a set of K segmented objects $O = \{o_1, \ldots, o_K\}$. Each object o_k is assigned an object category $c_k \in \{1, \ldots, C\}$ and has a pixel-wise segmentation mask $\mathcal{S}_k \in \{0, 1\}^{W \times H}$ to specify its location and size. The game further consists of a sequence of J questions $\boldsymbol{q}$ and answer a, which defines a dialogue $\mathcal{D} = [(q_j, a_j)]_{j=1}^{J} = [(q_1, a_1), \ldots, (q_J, a_J)]$, produced by the questioner and Oracle. We will use $\boldsymbol{q}_{j<}$ and $\boldsymbol{a}_{j<}$ to refer to the first $j-1$ questions and answers, respectively. Each question $\boldsymbol{q}_j$ contains a sequence of I_j tokens, i.e. $\boldsymbol{q}_j = [w_i^j]_{i=1}^{I} = [w_1^j, \ldots, w_{I_j}^j]$, where w_i^j is taken from a vocabulary $\mathcal{V}$ and represents the token at position i in question j. Each answer is either *Yes*, *No* or *N/A*, i.e. $a_j \in \{<yes>, <no>, <na>\}$. Finally, the Oracle has access to the identity of the correct object o^*, and the prediction of the questioner is denoted as o^{predict}. Note that the previous notation will be used thought the full manuscript.

4.3.2 Oracle baselines

The Oracle task requires to produce a yes-no answer for any object within a picture given a natural language question. We first introduce our model and then outline its results to get a better understanding of the GuessWhat?! dataset.

Model We propose a simple neural network based approach to this model, illustrated in Fig 4.6. Specifically, we use an appropriate neural network architecture to embed each of the following information: the image $\mathcal{I}$, the cropped object from $\mathcal{S}$, its spatial information, its category c and the current question $\boldsymbol{q}$. These embeddings are then concatenated as a single vector and fed as input to a single hidden layer MLP that outputs the final answer distribution using a softmax layer. Finally, we minimize the cross-entropy error during the training and report the classification error at evaluation time.

To embed the full image, it is rescaled to a 224 by 224 image and is passed through a pretrained VGGNet to obtain its fc8 features (last VGGNet layer). As for the selected object, it is first cropped by finding the smallest rectangle that encapsulates it, based on its segmentation mask. We then rescale the crop to a 224 by 224 square, before obtaining its fc8 features from the pretrained VGGNet. Although we could use the mask to drop out pixels around the selected object, we keep the crop as is since pretrained VGGNets are exposed to such background noise during their training.

We also embed the spatial information of the crop, to help locate the cropped object within the whole image. To do so, we follow the approach of (Hu et al., 2016b; Yu et al., 2016a) and extract an 8-dimensional vector of the location of the bounding box:

$$\boldsymbol{x}_{spatial} = [x_{min}, y_{min}, x_{max}, y_{max}, x_{center}, y_{center}, w_{box}, h_{box}] \tag{4.1}$$

where w_{box} and h_{box} denote the width and height of the bounding box, respectively. We normalize the image height and width such that coordinates range from -1 to 1, and place the origin at the center of the image. As for the object category, we convert its one-hot class vector into a dense category embedding using a learned look-up table. Finally, the embedding of the current natural language question $\boldsymbol{q}$ is computed using an LSTM (Hochreiter and Schmidhuber, 1997) where questions are first tokenized by using the word punct tokenizer from the python nltk toolkit (Bird et al., 2009). For simplicity, we decided to ignore the question-answer pairs history $\boldsymbol{q}_{<t}$ in our Oracle baseline.

Training setting We train all Oracle models on the successful dataset. During training, we keep the parameters of the VGGNet fixed, and optimize the LSTM, object category/word look-up tables and MLP parameters by minimizing the negative log-likelihood of the correct answer. We use Adam (Kingma and Ba, 2015) for optimization and train for at most 15 epochs. We use early stopping on the validation set, and report the train, valid and test error.

Results We report results for several Oracle models using a different set of inputs in Table 4.2. We name the model after the input we feed to it. For instance, (Question+Category+Spatial+Image) refers to the network fed with the question $\boldsymbol{q}$, the object category c, the spatial features $\boldsymbol{x}_{spatial}$ and the full image $\mathcal{I}$. The results of all subsets are reported in Table 4.6 in Appendix 4.5.2. As the GuessWhat?! dataset is fairly balanced, simply outputting the most common answer in the training set – No – results in a high 50.8% error rate. Solely providing the image or crop features barely improves upon this result. Only using the question slightly improves the error rate to 41.2%. We speculate that this small bias comes from questioners that refer to objects that are never segmented or over-represented categories. As hoped, we observe that the error rate significantly drops ($< 31\%$) when we finally feed information on the object to guess (crop, spatial or category) to the model. We find that crop and category information are redundant: the (Question+Category) and (Question+Crop) model achieve respectively 29.2% and 25.7% error, while the combined model (Question+Category+Crop) achieves 24.7%. In general, we expect the object crop to contain additional information, such as color information, beside the object class. However, we find that the object category outperforms the object crop embedding. This might be partly due to the imperfect feature extraction from the crops. We further investigate advanced models that successfully integrate visual cues in Chapter 5

Model	Train err	Val err	Test err
Majority Baseline (no)	47.4%	46.2%	50.9%
Question	40.2%	41.7%	41.2%
Image	45.7%	46.7%	46.7%
Crop	40.9%	42.7%	43.0%
Question + Crop	22.3%	29.1%	29.2%
Question + Image	37.9%	40.2%	39.8%
Question + Category	23.1%	25.8%	25.7%
Question + Spatial	28.0%	31.2%	31.3%
Question + Category + Spatial	17.2%	21.1%	**21.5%**
Question + Category + Crop	20.4%	24.4%	24.7%
Question + Spatial + Crop	19.4%	26.0%	26.2%
Question + Category + Spatial + Crop	16.1%	21.7%	22.1%
Question + Spatial + Crop + Image	20.7%	27.7%	27.9%
Question + Category + Spatial + Image	19.2%	23.2%	23.5%

Table 4.2: Classification errors for the Oracle baselines on train, valid and test set. The best performing model is "Question + Category + Spatial" and refers to the MLP that takes the question, the selected object class and its spatial features as input.

Model	Train acc	Val acc	Test acc
LSTM	27.9%	37.9%	**38.7%**
HRED	32.6%	38.2%	39.0%
LSTM+VGG	26.1%	38.5%	39.5%
HRED+VGG	27.4%	38.4%	39.6%

Table 4.3: Classification errors for the guesser baselines on train, valid and on successful set.

and Chapter 7. Finally, our best performing baseline model combines object category and its spatial features along with the question.

4.3.3 Questioner baselines

Given an image, the questioner must ask a series of questions and guess the correct object. We separate the questioner task into two different sub-tasks that are trained independently:

Guesser Given an image $\mathcal{I}$ and a sequence of J questions and answers $(\boldsymbol{q}, a)_{\leq J}$, the Guesser predicts the correct object o^* from the set of all objects O.

Question Generator Given an image $\mathcal{I}$ and a sequence of J questions and answers $(\boldsymbol{q}, a)_{\leq J}$, the Question Generator produce a new question $\boldsymbol{q}_{J+1}$.

In general, one also needs a module to determine when to start guessing the object (and stop asking questions). In our baseline, we bypass this issue by fixing the number of questions to 5 for the question generator model. However, this constraint was later removed by adding either a stop dialogue tokens in Chapter 8, or by adding a logistic stopping flag (Shekhar et al., 2018).

Guesser The role of the guesser model is to predict the correct object. To do so, the guesser has access to the image, the dialogue and the list of objects in the image. We encode the image by extracting its fc8 features from VGGNet. A dialogue of a GuessWhat?! game is a sequence on

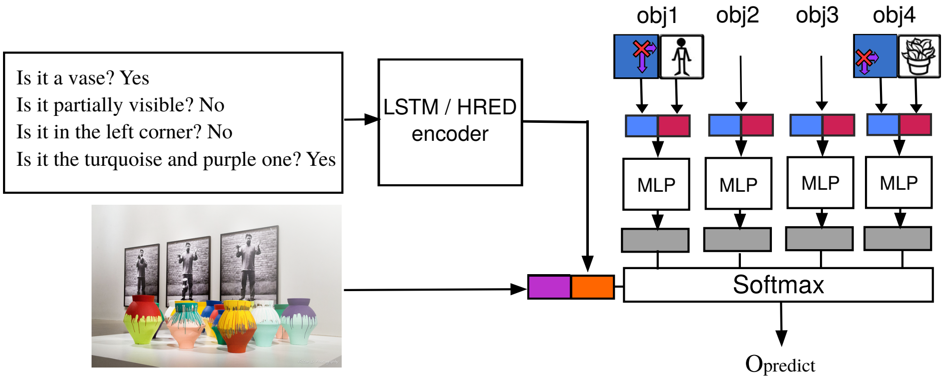


Figure 4.7: Overview of the guesser model for an image with 4 segmented objects. The weights are shared among the MLPs, this allows for an arbitrary number of objects.

two different levels: there is a variable number of question-answer pairs where each question in turn consists of a variable-length sequence of tokens. This can be encoded into a fixed size vector by using either an LSTM encoder (Hochreiter and Schmidhuber, 1997) or an Hierarchical recurrent encoder decoder (HRED) encoder (Serban et al., 2016). While the LSTM encoder considers the dialogue as one flat sequence, HRED explicitly models the hierarchy by two different RNN. First, an encoder RNN creates a fixed-size representation of a question or answer by reading in its tokens and taking the last hidden state of the RNN. This representation is then processed by the context RNN to obtain a representation of the current dialogue *state*. For both models, we concatenate the image and dialogue features and do a dot-product with the embedding for all the objects in the image, followed by a softmax to obtain a prediction distribution over the objects. Given the best performance of the "Question+Category+Spat" Oracle model, we represent objects by their category and their spatial features. More precisely, we concatenate the 8-dimensional spatial representation (see Eq. 4.1) and the object category look-up and pass it through an MLP layer to get an embedding for the object. Note that the MLP parameters are shared to handle the variable number of objects in the image. See Fig 4.7 for an overview of the guesser with HRED and LSTM.

Table 4.3 reports the results for the guesser baselines using human-generated dialogues. As a first baseline, we report the performance of a random guesser which does not use the dialogue information. We split the guesser results based on whether they use the VGGNet features or not. In general, we find that including VGGNet features does not improve the performance of the HRED and LSTM models. We hypothesize that the VGGNet features are a too coarse representation of the image scene, and that most of the visual information is already encoded in the question and the object features. Surprisingly, we find LSTMs to perform slightly better than the sophisticated HRED models. Empirical evidence would suggest that HRED tend to overfit faster than LSTM partly because of the additional parameters that are introduced. In the long run, we discard the HRED architecture as the ratio between training complexity and performance is not favorable.

Question Generator The question generation task is hard for several reasons. First, it requires high-level visual understanding to ask meaningful questions. Second, the generator should be able to handle long-term context to ask a sequence of relevant questions, which is one of the most challenging problems in dialogue systems. Additionally, we evaluate the question generator using the imperfect Oracle and imperfect guesser, which introduces compounding errors.

HRED (Serban et al., 2016) is the current state of the art method for natural language generation tasks. We extend this model by conditioning on the VGGNet features of the image as illustrated in

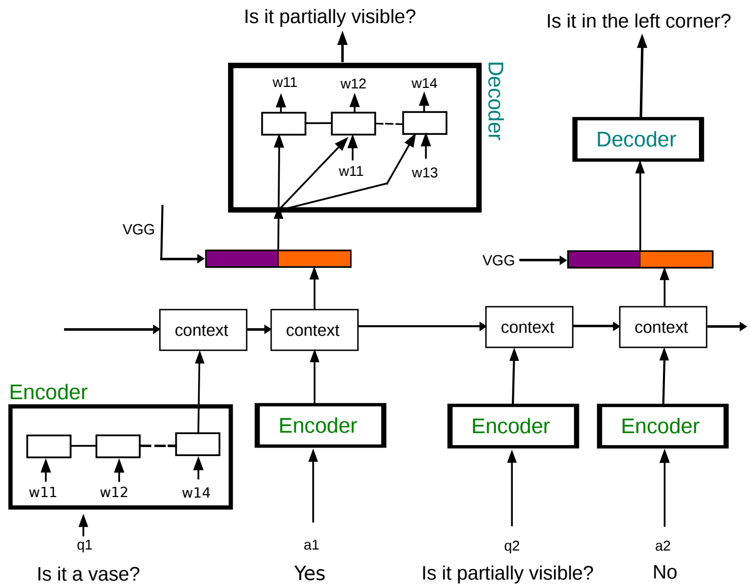


Figure 4.8: HRED model conditioned on the VGGNet features of the image. To avoid clutter, we here only show the part of the model that defines a distribution over the third question given the first two questions, its answers and the image $P(q_2|q_{<2}, a_{<2}, \mathcal{I})$. The complete HRED model models the distribution over all questions.

Fig 4.8. Finally, we train our proposed model by maximizing the conditional log-likelihood:

$$\log P(\boldsymbol{q}_j|\boldsymbol{q}_{<j}, \boldsymbol{a}_{<j}, \mathcal{I}) = \log \prod_{j=1}^{J} P(\boldsymbol{q}_j|\boldsymbol{q}_{<j}, \boldsymbol{a}_{<j}, \mathcal{I}) \tag{4.2}$$

$$= \log \prod_{j=1}^{J} \prod_{i=1}^{N_j} P(w_i^j|\boldsymbol{w}_{<i}^j, \boldsymbol{q}_{<j}, \boldsymbol{a}_{<j}, \mathcal{I}) \tag{4.3}$$

with respect to the described parameters. At test time, we use a beam-search to approximately find the most probable question $\boldsymbol{q}_j$. Evaluating the questioner model requires a pretrained Oracle and a pretrained guesser model. We use our questioner model to first generate a question which is then answered by the Oracle model. We repeat this procedure 5 times to obtain a dialogue. We then use the best performing guesser model to predict the object and report its accuracy as the metric for the QGen model in Tab. 4.4. A guesser based on human generated dialogues achieves 61.3% accuracy. The Question Generator models achieve reasonable performance which lies in between the random performance and the performance of the guesser on human dialogues. Note that some of these baseline scores were later improved by fine-tuning both the three models. A few generated dialogues are shown in Fig. 4.20a and 4.20b.

Model	Accuracy
Human generated dialogue	61.3%
QGen+Oracle	34.0%
QGen+Oracle (Strub et al., 2017b)	44.8%
Random	17.1%

Table 4.4: Test error for the question generator models (QGen) based on VGG+HRED(FT) guesser model. We here report the accuracy error of the guesser model fed with the questions from the QGen model.

4.4 Visual Dialogues Task: Concurrent and subsequent works

In this section, we dig into the literature that follows the release of GuessWhat?!. We also compare our dataset with the concurrent Visual Dialog task that was released at the same time. We here notice that despite apparent similarities, both tasks focus on different research facets of interactive and visually grounded language challenges, making them highly complementary.

GuessWhat?! As mentioned at the beginning, GuessWhat?! extends the ReferIt task into the dialogue setting: it is a cooperative two-player game in which both players see the image of a rich visual scene with several objects.

More generally, the oracle and guesser models may be seen as extensions of VQA models where object features are appended to the scene representation, and dialogue co-occurence must be disambiguated. Later works then naturally explore new neural architectures to enhance the GuessWhat?! performance such as attention mechanisms that jointly attend to individual question and objects (Deng et al., 2018; Zhuang et al., 2018), memory network architectures (Abbasnejad et al., 2018; Han et al., 2017a; Zhao and Tresp, 2018b), or R-CNN modules (Bani et al., 2018).

On the questioner side, the model may be assimilated to basic image-captioning models whose visual-features are first enhanced with dialogue history. The models are first trained on the GuessWhat?! dataset, before being finetune with RL as latter explored in Chapter 8. Most of the succeeding researches focused on adapting RL to the visual dialogue setting, using reward shaping (Zhang et al., 2018a), approximating model-based RL to plan over potential oracle answers (Abbasnejad et al., 2018; Lee et al., 2018b), incorporating replay buffer mechanisms (Zhao and Tresp, 2018a), and benchmarking exploration strategies, e.g., varying the softmax-policy temperature (Zhao and Tresp, 2018b) or using bayesian Dropout (Abbasnejad et al., 2019).

In a different perspective, Han et al. (2017b) assessed a brute-force strategy to spatially localize objects with rule-based agents. Zhu et al. (2017b) explore how GuessWhat?! models can learn to cheat when fully trained with reinforcement learning, highlighting the limitation of accuracy while benchmarking models. Shekhar et al. (2018) incorporates a decision head to learn when to stop a dialogue. Finally, Tremblay et al. (2018) embeds a questioner modules into a robot to perform an entertaining interactive demonstration of GuessWhat?!.

VisDiag VisDiag is also a two-player cooperative game where one of the player is to depict a hidden image by asking a sequence of open questions. The first player – the A-Bot – is assigned a natural image, while the second player – the Q-Bot – is assigned a brief image caption before querying the oracle. After a predefined number of steps, the Q-Bot generates a visual description of the image (Das et al., 2017b), which is compared to the ground truth. The VisDiag dataset contains 133k dialogue, with 10 rounds with distinct images. Two synthetic VisDiag were also released: the MNIST-Dialogue dataset performs the VisDiag task over an array of nine colored MNIST number, while the CLEVR-dialogue dataset use the CLEVR engine with a hand-crafted but diverse grammar (Kottur et al., 2019)

The A-Bot models are VQA models that were enhanced to deal with dialogue co-occurrences, the A-Bot can either be a generative model by producing sequence of words, or discriminative by scoring a set of potential answers (Das et al., 2017a). The A-Bot models include memory networks (Das et al., 2017a; Seo et al., 2017), NMN extensions to deal with dialogue co-references (Kottur et al., 2018) or different co-attention mechanisms with multi-step reasoning (Gan et al., 2019; Kang et al., 2019; Wu et al., 2018b; Yang et al., 2019). Differently, Jain et al. (2018) change the network conditioning mechanism to score answer candidates. Finally, (Lu et al., 2017a; Wu et al., 2018b) introduce a GAN-like procedure to reject poorly generated answers.

The Q-Bots are pure linguistic agents as they do not observe the image; they are first trained on the VisDiag dataset, before being finetuned with RL by playing with a A-Bot (Das et al., 2017b). However, VisDiag has no clear reward signal upon task completion. Thus, Das et al. (2017b) train the Q-Bot to output an expected visual representation $\hat{\boldsymbol{y}}$ which is compared with the ground truth image representation $\boldsymbol{y}^{gt}$. At every dialog turn t, the authors define the reward as the relative similarity improvement (Euclidean distance) toward the target image, $r_t(.) = ||\boldsymbol{y}^{gt} - \hat{\boldsymbol{y}}_{t-1}||_2^2 - ||\boldsymbol{y}^{gt} - \hat{\boldsymbol{y}}_t||_2^2$. The Q-Bot performance is computed by assessing whether the target image is within the pool of the top 5% images closest to the predicted visual representation (among a set of 10k candidates). However, Das et al. (2017b) observe that reinforcement learning has little effects on the success ratio and that the models mostly rely on the caption rather than asking discriminate questions. Mironenco et al. (2017) also notice that randomly changing A-Bot answers does not impact the Q-Bot success. Zhang et al. (2018b) ease the Q-Bot difficulty by selecting pre-generated questions, and learning when to stop asking questions. Although the authors obtain good results by using a pool of 200 questions over 20 images, the model has mediocre performance when dealing with new images. In the end, it is still an open question whether the Q-bot limitations come from the evaluation protocol, the complex reward, the continuous action space (to generate the visual embedding), or the game design which does not entail truly goal-oriented dialogue (Zhang et al., 2018b).

In a different research perspective, Agarwal et al. (2019) explore how to reduce language drift in Visual Dialog by using a multi-agent setting; the authors jointly train several Q-Bots and A-Bots, forcing them to align their language understanding and generating to each other, and therefore slowing down the language drift. Finally, Sharma et al. (2018) learn to generate a complete image by conditioning a GAN with Q-bot dialogues.

Game comparison In the end, GuessWhat?! and VisDiag lead to different research directions despite apparent similarities. GuessWhat?! research papers would focus on question generation side, with an emphaze on RL methods while the VisDiag community would mostly extend the VQA task to the dialogue setting, designing more flexible answer modules. In some senses, GuessWhat?! tends to embody vision into a language setting, while VisDiag would include language into a computer vision challenge (which can also be linked from the background of the authors). We also note that few researchers work on both datasets (Lee et al., 2018b, 2019) despite several joint events between the GuessWhat?! and VisDiag teams (Strub et al., 2017a, 2018a).

Following Visual Dialogue Datasets We here list a few other visually grounded dialogue challenges that have been explored over the past two years. For instance, Guo et al. (2018) train a model to retrieve a shoe picture by conversing with a user-simulator browsing a shoe catalogue (Berg et al., 2010). Chattopadhyay et al. (2017) introduce GuessWhich?! within the intersection of GuessWhat?! and VisDiag; an agent interacts with a pretrained A-Bot model to identify a secret image from a visible pool of candidate images. Differently, Talk-the-Walk (de Vries et al., 2018) casts an instruction following task into a dialogue where a tourist describes his street view allowing a guide to locate him on a map and instruct him the path to follow. Finally, Lee et al. (2018b) introduce the co-draw task where two agents converse to let the first player replicating a picture displayed to the second player with high-level drawing action. Although very promising dialogue tasks, both Talk-the-Walk and Co-Draw dataset contain around 10k games, which turns out to be insufficient to solve these complex problems. In the end, there exists no truly satisfactory goal-oriented visual dialogue dataset: there are either too small, have hand-crafted reward signals, or rely on closed-form questions.

4.5 Discussion

We introduced the GuessWhat?! game, a novel framework for multi-modal dialogue. At publication time, it was the first large-scale dataset involving images and dialogue. A wide range of challenges has arisen from this union as they rely on different fields of machine learning such as natural language understanding, generative models or computer vision. GuessWhat?! turns out to be an engaging game that greatly decreases the cost for collection of a big dataset required for modern algorithms. This chapter also introduces three agents based on the questioner and Oracle role. In each case, we prototype a neural architecture, we analyze these results and, we presents a quantitative description of the GuessWhat?! dataset. We believe GuessWhat?! could allow for a myriad of other applications that may either be based on the game itself or extending the database to other tasks. Differently, GuessWhat?! could be a test bed for one-shot learning (Li et al., 2006) of guessing new object categories, transfer learning on line-drawing images (Castrejon et al., 2016) or using questions from another language. Thus, the GuessWhat?! dataset offers an opportunity to develop original machine learning tasks upon it, and it still remains a challenging and rich dataset three years after its release.

4.5.1 User interface

Figures 4.9, 4.10 present the instructions for the Oracle and questioner before they started their first game.

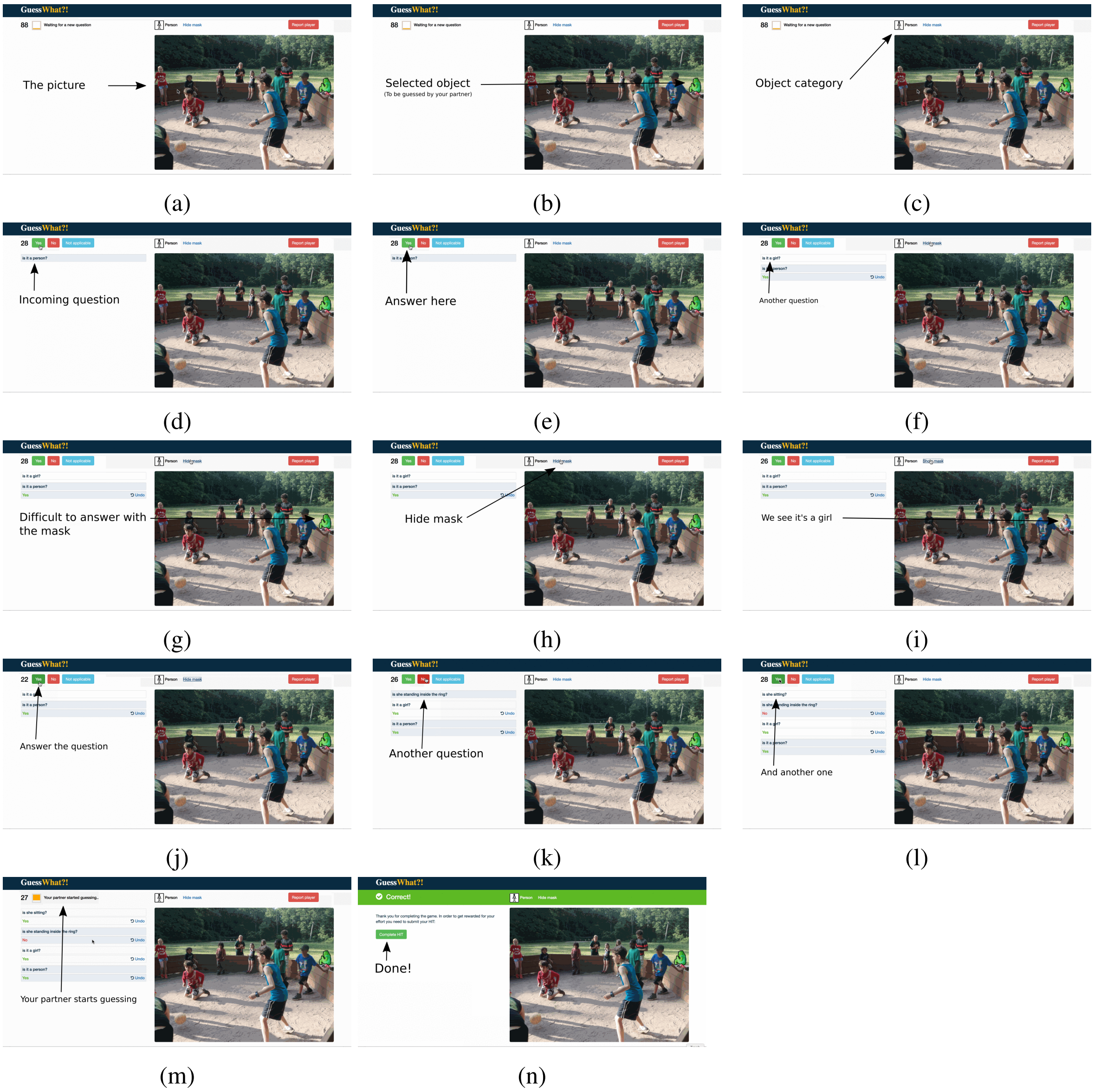


Figure 4.9: An example game from the perspective of the Oracle. Shown from left to right and top to bottom.

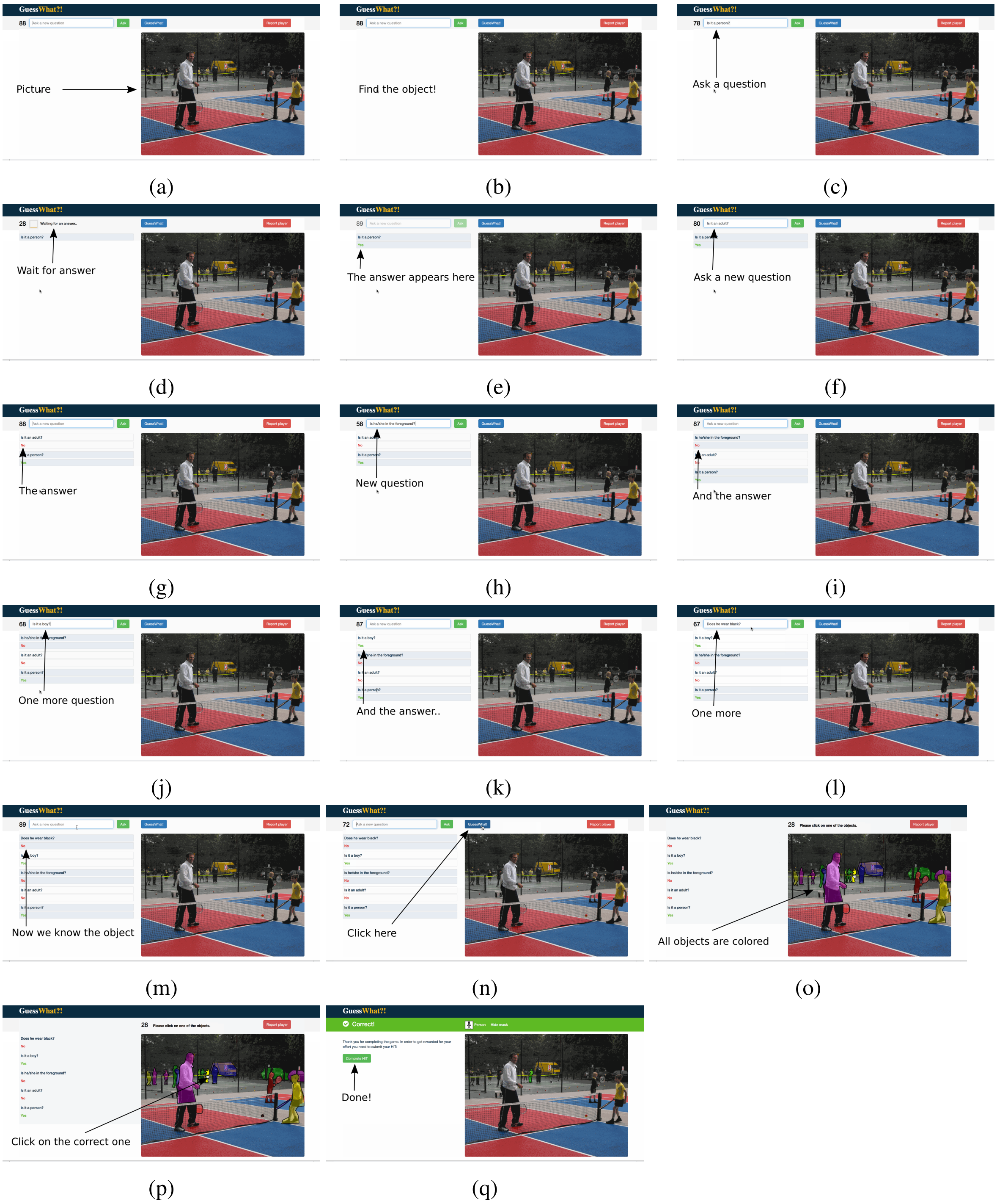
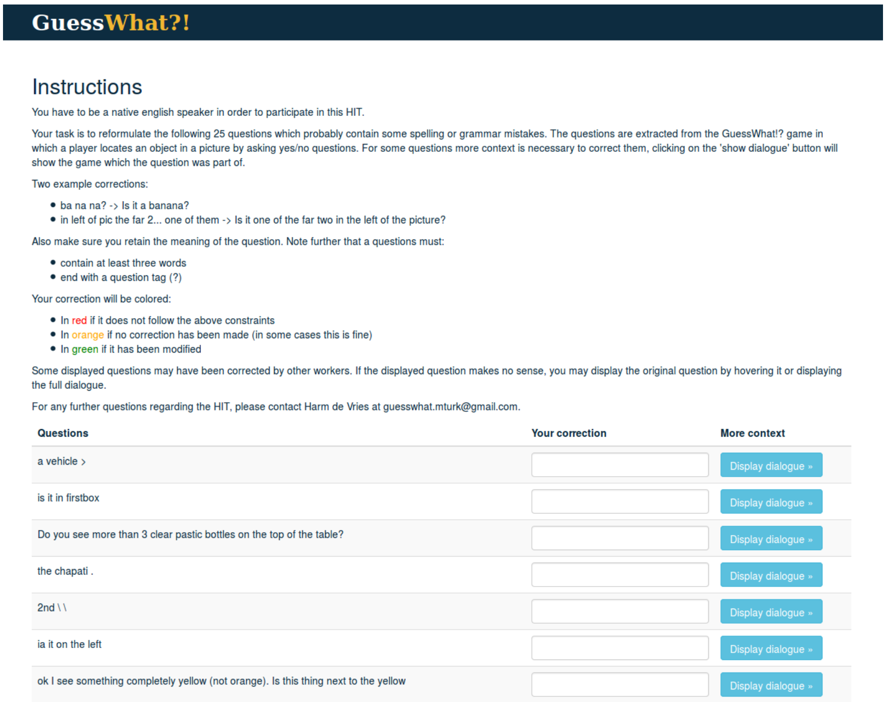
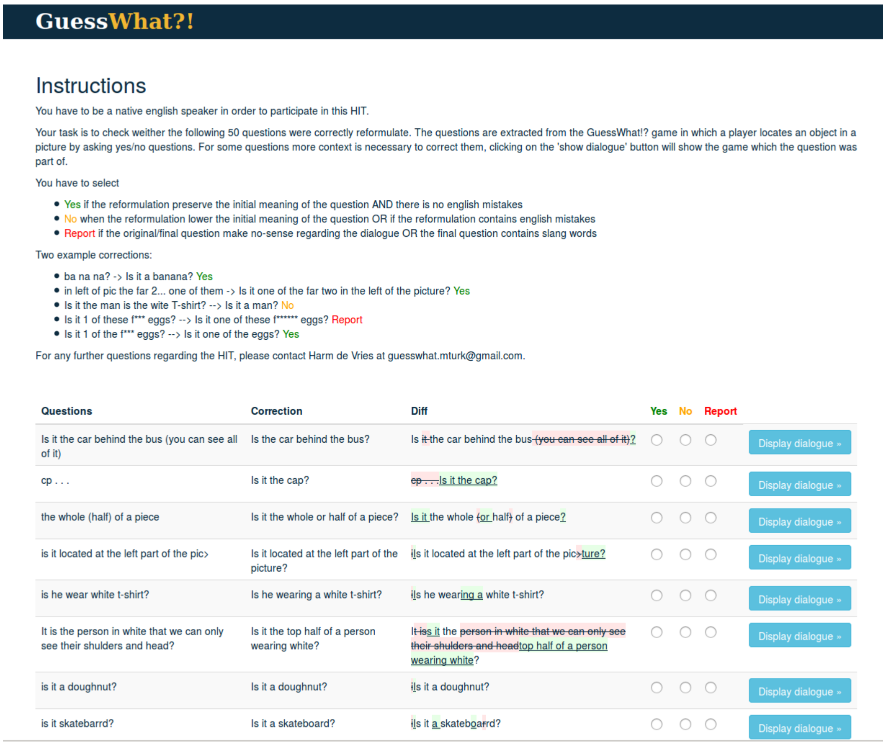


Figure 4.10: An example game from the perspective of the questioner. Shown from left to right and top to bottom.



(a) Interface to fix ill-formatted questions



(b) Interface to validate the fix ill-formatted questions

Figure 4.11: In the first task, we ask workers to correct mistakes in the questions. We then ask workers to validate the proposed correction by showing the difference between the original question and its correction. We alternate both tasks till all questions are corrected and validated.

4.5.2 Additional database statistics

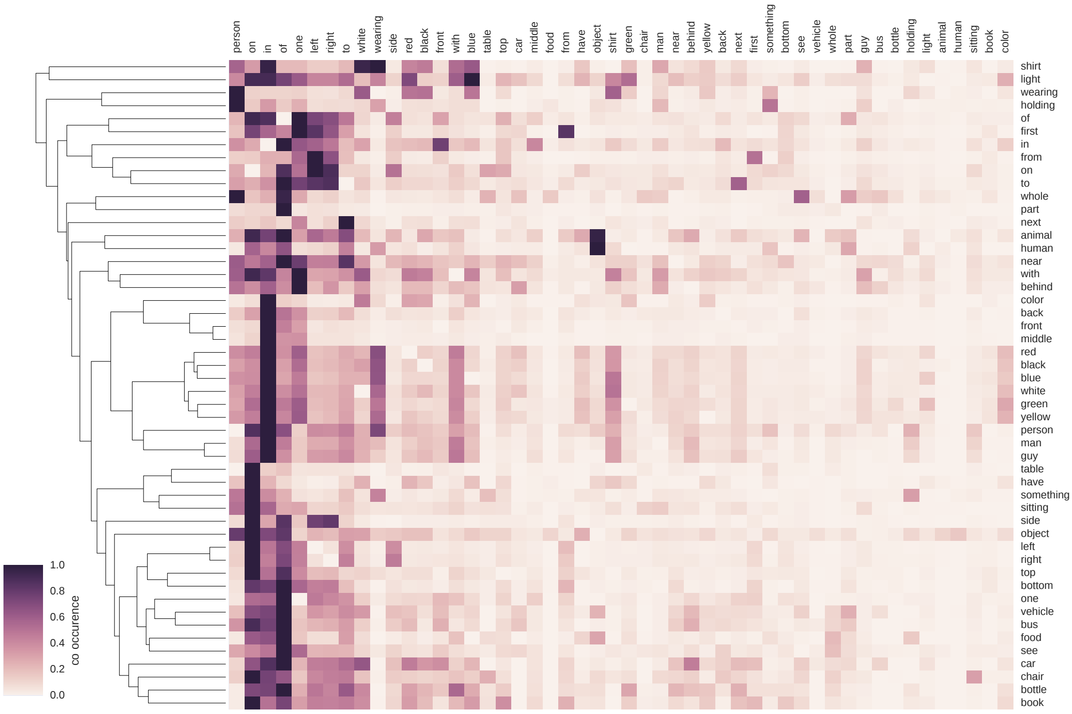


Figure 4.12: Co-occurrence matrix of words. Only the 50 most frequent words are kept. Rows are first normalized before being sorted thanks a hierarchical clustering with an Euclidean distance.

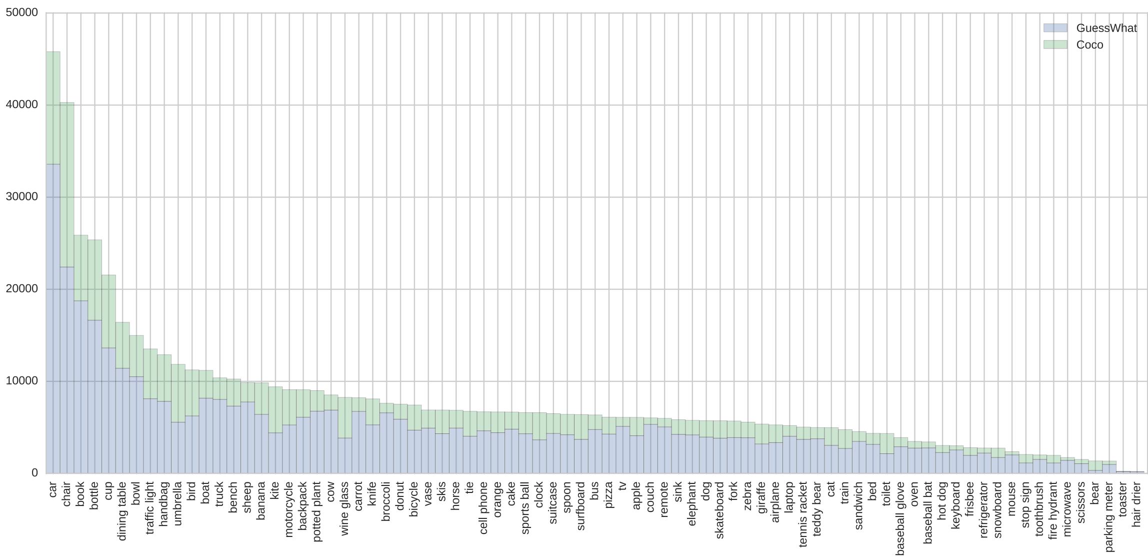


Figure 4.13: Visualization of the object category distribution of MS COCO and GuessWhat?! dataset. The person category was removed for clarity (resp. 273469 and 188204).

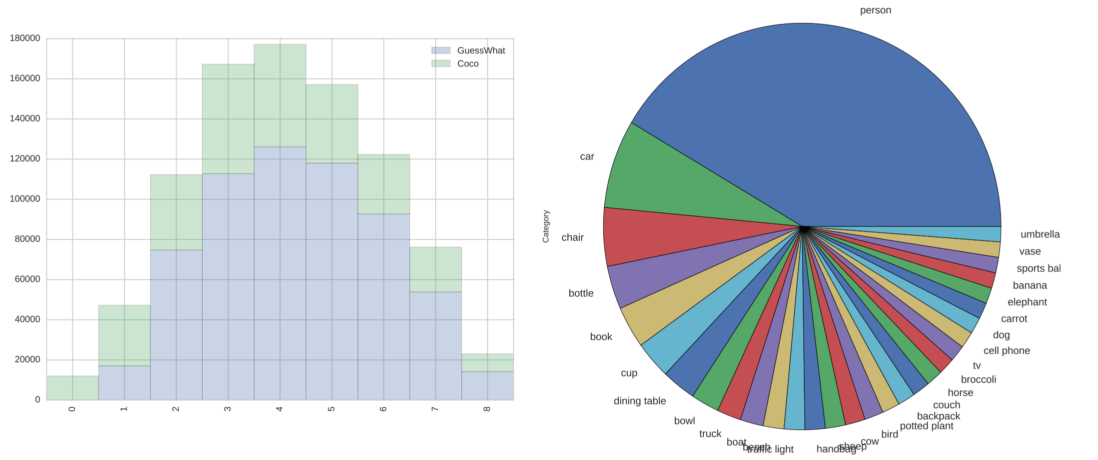


Figure 4.14: (left) Visualization of the object size distribution of MS COCO and GuessWhat?! dataset. (right) Distribution of the the 30 (out of 80) prominent object categories in the GuessWhat?! (71.3% of the objects).

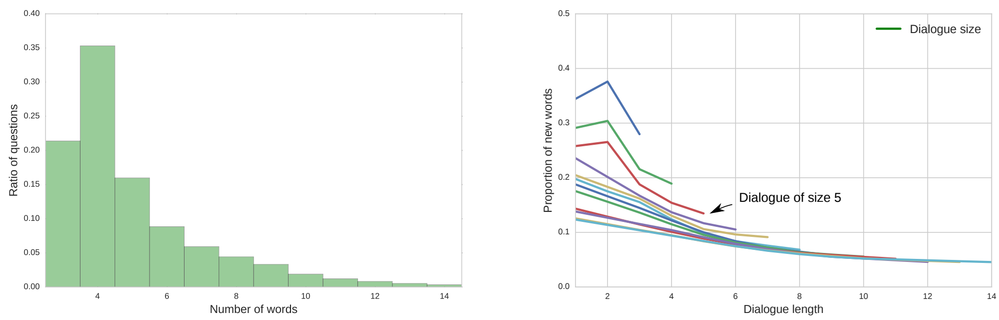


Figure 4.15: (left) Number of words per question. The question length follows a Poisson-like distribution, a finding which is in line with other datasets (Antol et al., 2015). (right) Percentage of apparition of new words along a dialogues. Questioner tends keep using the same words during the dialogues.

Topic 1	Topic 2
Abstract words	Descriptive words
person	left
food	one
vehicle	right
human	wearing
car	side
one	white
object	red
animal	table

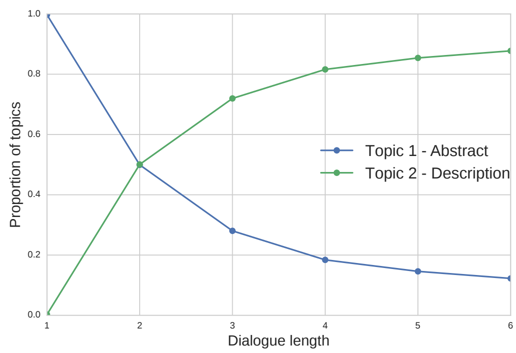


Figure 4.16: Relative evolution of topics during a dialogue of size 6. We applied Data Topic Models (DTM) (Blei and Lafferty, 2006) with the python framework (Rehurek and Sojka, 2010) on our dataset. The table reports the two prominent detected topics with their respective key words while the figure display their relative evolution during the dialogue. The topic titles are manually picked.

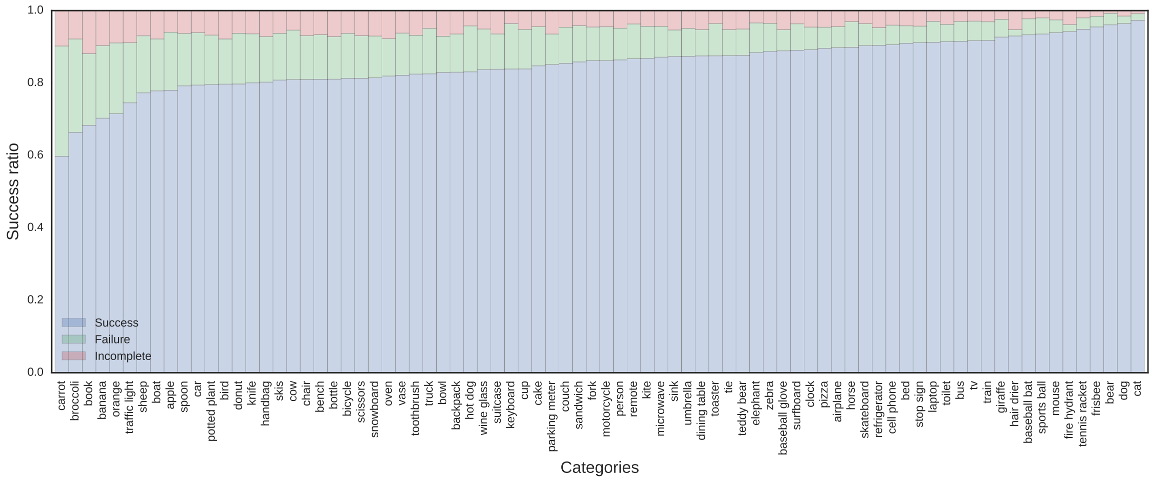


Figure 4.17: Histogram of success ratio broken down per object category.

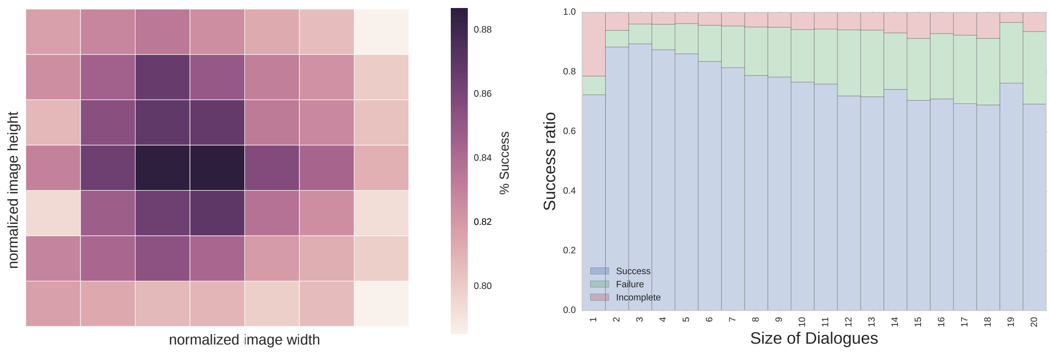


Figure 4.18: (left) Heatmap of the success ratio with respect to the spatial location within the picture. (right) Histogram of the success ratio relative to the dialogue length.

Is it a person?	*No*
Is it on the shelves?	*No*
Is it on the floor?	*Yes*
Is it a cup?	*No*
Is it blue?	*No*
Is it wood?	*No*
Is it the bed?	*No*
Is it in the lower half of the image?	*Yes*
Is it in the lower left corner?	*No*
Is it near the blue tray with two cups?	*No*
Is it near the green and blue toy?	*No*
Is it near the boy laying on the floor?	*Yes*
Is it something he is touching?	*Yes*
Is it the remote?	*Yes*
Is it the one in his right hand (close to the wooden box)?	*No*

Figure 4.19: A long dialogue example in a very rich environment.

person	14.48	person	3.20	=	left	2.95	↗
food	1.29	left	1.69		right	2.32	↗
animal	1.16	wearing	1.20	new	person	2.28	↘
human	1.03	right	1.02	new	wearing	1.66	↘
object	0.77	front	0.97	new	whole	1.58	new
car	0.60	white	0.91	new	white	1.56	=
vehicle	0.57	red	0.77	new	red	1.26	=
cat	0.41	car	0.64	↘	black	1.19	↗
alive	0.37	black	0.60	new	front	1.14	↘
dog	0.35	blue	0.59	new	blue	1.10	=

(a) Dialogues having 3 questions

person	8.20	person	1.98	=	left	1.69	↗	left	1.92	=	left	2.13	=
food	1.03	left	1.03	↗	person	1.41	↘	right	1.77	↗	right	2.04	=
human	0.56	right	0.66	new	right	1.26	=	person	1.20	↘	white	1.28	↗
animal	0.46	front	0.59	new	white	0.84	↗	white	1.12	=	person	1.26	↘
vehicle	0.45	car	0.51	↗	wearing	0.82	↗	wearing	0.93	=	black	0.90	↗
object	0.42	white	0.48	new	side	0.67	↗	black	0.79	↗	wearing	0.85	↘
car	0.36	wearing	0.48	new	red	0.62	↗	red	0.72	=	red	0.80	=
furniture	0.24	side	0.43	new	front	0.58	↘	side	0.69	↘	whole	0.80	new
left	0.24	red	0.39	new	black	0.55	new	blue	0.65	↗	blue	0.75	=
edible	0.20	vehicle	0.39	↘	blue	0.54	new	front	0.58	↘	front	0.73	=

(b) Dialogues having 5 questions

person	5.89	person	1.44	=	left	1.08	↗	left	1.26	=	left	1.33	=	left	1.42	=	left	1.65	=
food	0.74	left	0.73	↗	person	0.96	↘	right	1.08	↗	right	1.22	=	right	1.39	=	right	1.54	=
human	0.38	right	0.42	new	right	0.89	=	person	0.82	↘	white	0.81	↗	white	0.88	=	white	0.96	=
vehicle	0.30	table	0.37	↗	side	0.57	↗	white	0.67	↗	person	0.80	↘	person	0.84	=	person	0.90	=
object	0.28	front	0.36	new	white	0.50	↗	side	0.60	↘	wearing	0.59	↗	black	0.63	↗	red	0.65	↗
car	0.26	food	0.35	↘	wearing	0.48	↗	wearing	0.54	=	side	0.57	↘	red	0.57	↗	black	0.63	↘
animal	0.26	side	0.35	new	red	0.41	new	red	0.49	=	red	0.54	=	wearing	0.56	↘	blue	0.57	↗
furniture	0.20	car	0.31	↘	table	0.39	↘	table	0.41	=	black	0.51	↗	blue	0.54	↗	wearing	0.52	↘
left	0.14	wearing	0.28	new	front	0.38	↘	black	0.41	↗	blue	0.49	↗	side	0.53	↘	next	0.51	new
boat	0.14	something	0.28	new	car	0.37	↘	blue	0.37	new	front	0.42	↗	front	0.45	=	side	0.51	↘

(c) Dialogues having 7 questions

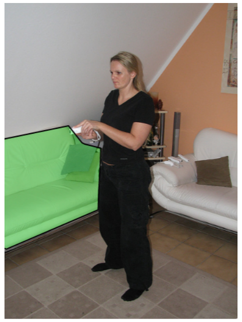
Table 4.5: Proportions of the ten most common words for each depth of questions sorted by the size of the dialogues

Model	Train err	Valid err	Test err
Dominant class (no)	47.4%	46.2%	50.9%
Category	43.0%	42.8%	43.1%
Question	40.2%	41.7%	41.2%
Crop	40.9%	42.7%	43.0%
Image	45.7%	46.7%	46.7%
Spatial	43.9%	44.1%	44.3%
Category + Spatial	41.6%	41.7%	42.1%
Question + Crop	22.3%	29.1%	29.2%
Question + Image	37.9%	40.2%	39.8%
Question + Category	23.1%	25.8%	25.7%
Question + Spatial	28.0%	31.2%	31.3%
Spatial + Crop	41.8%	42.4%	42.8%
Crop + Image	41.6%	42.1%	42.4%
Spatial + Image	42.2%	44.1%	44.2%
Category + Crop	41.0%	41.7%	42.3%
Category + Image	42.3%	42.7%	43.0%
Category + Crop + Image	40.6%	41.5%	41.8%
Category + Spatial + Crop	40.6%	41.6%	42.1%
Question + Category + Spatial	17.2%	21.1%	**21.5**%
Question + Crop + Image	23.7%	29.9%	30.0%
Category + Spatial + Image	40.4%	42.0%	42.2%
Question + Category + Image	23.4%	27.1%	27.4%
Question + Spatial + Image	28.4%	32.5%	32.5%
Spatial + Crop + Image	41.6%	42.1%	42.5%
Question + Category + Crop	20.4%	24.4%	24.7%
Question + Spatial + Crop	19.4%	26.0%	26.2%
Question + Category + Spatial + Crop	16.1%	21.7%	22.1%
Question + Spatial + Crop + Image	20.7%	27.7%	27.9%
Category + Spatial + Crop + Image	40.3%	41.4%	41.8%
Question + Category + Spatial + Image	19.2%	23.2%	23.5%
Question + Category + Crop + Image	20.0%	25.3%	25.5%
Question + Category + Spatial + Crop + Image	17.8%	23.2%	23.3%

Table 4.6: Classification errors for all Oracle baselines.

	Generated	Groundtruth
	Is it a person? No Is it the kite? No Is it the kite? No Is it the chair? No Is it the boat? Yes	Is it in the sky? No Is it the umbrella? No Is it the ocean? No Is it the boat? Yes
	Is it a person? No Is it a skateboard? No Is it a car? Yes Is it the one on the right? No Is it the one on the right? No	Is it an object? Yes Do you wear it? No Do you ride it? No Is it metal? Yes
	Is it a person? Yes Is it the one in the front? No Is it the one in the middle? Yes Is it the one in the middle? Yes Is it the whole person? Yes	Is it a person? Yes Is he in the foreground? No Is he wearing blue? Yes

(a) Three samples of for which the correct object was predicted.

	Generated	Groundtruth
	Is it the cat? No Is it the cat? No Is it the chair? No Is it the book? No Is it the book? No	Is it an animal? No Is it a device? Yes Is it silver in color? Yes
	Is it a person? No Is it a remote? No Is it the chair? Yes Is it the one on the right? No Is it the one next to the right? No	Is it a person? No Is it a couch? Yes Does the couch have two pillows on it? Yes
	Is it a person? Yes Is it the guy in the front? No Is it the guy in the middle? No Is it the guy in the middle? No Is it the guy in the middle? No	Is it a person? Yes Is it in the foreground? No Is it on a screen? Yes

(b) Three dialogue samples for which the wrong object was predicted.

Chapter 5

Conditional Batch-Normalization

«Do Androids Dream of Electric Sheep? »

Philip K. Dick

«I am guessing is a sheep? A: Yes »

GuessWhat?!

Contents

In the previous chapter, we observe the poor performance of multimodal learning architecture in the GuessWhat?! oracle/guesser tasks. At this point, there was little interest in trying to improve the visually grounded language generation as neural networks failed already failed at correctly interleaving visual and language modalities in simple classification tasks. We thus start examining the multimodal learning architecture to improve inductive neural biases. We quickly observe that there was an implicit division between language and vision representations. It was commonly assumed that language refers to high-level visual concepts, while computer models focused on the low-level features, and there no interaction between them. This view dominated the literature in computational models for language-vision tasks, where visual and linguistic inputs are mostly processed independently before being fused into a single representation, as mentioned in Sec 3.1.2. In our research, we thus deviate from this classic pipeline and propose to modulate the *entire visual processing* by a linguistic input, highlighting the need for middle-conditioning mechanisms for deep learning architecture. In this chapter, we introduce Conditional Bacth Normalization (CBN) as an efficient mechanism to modulate convolutional feature maps by a linguistic embedding. We apply Conditional Bacth Normalization (CBN) to a pretrained Residual Neural Network (ResNet), leading to the MODulatEd ResNet (ModeRn) architecture, and show that this significantly improves strong baselines on two visual question answering tasks. Our ablation study confirms that modulating from the early stages of visual processing is beneficial.

5.1 Introduction

Human beings combine the processing of language and vision with apparent ease. For example, we can use natural language to describe perceived objects and we are able to imagine a visual scene from a given textual description. Developing intelligent machines with such impressive capabilities remains a long-standing research challenge with many practical applications.

Towards this grand goal, we have witnessed an increased interest in tasks at the intersection of computer vision and natural language processing. Developing computational models for language-vision tasks is challenging, especially because of the open question underlying all these tasks: how to fuse/integrate visual and textual representations? To what extent should we process visual and linguistic inputs separately, and at which stage should we fuse them? And equally important, what fusion mechanism to use?

In this chapter, we restrict our attention to the domain of visual question answering which is a natural testbed for fusing language and vision. The VQA task concerns answering open-ended questions about images and has received significant attention from the research community (Antol et al., 2015; Fukui et al., 2016; Goyal et al., 2017; Malinowski et al., 2015). Current state-of-the-art systems often use the following computational pipeline (Ben-Younes et al., 2017; Malinowski et al., 2015; Ren et al., 2015a) illustrated in Fig 5.1. They first extract *high-level* image features from an ImageNet pretrained convolutional network (e.g. the activations from a ResNet network (He et al., 2016)), and obtain a language embedding using a RNN over word-embeddings. As fully developed in Sec 3.1, these two high-level representations are then fused by concatenation (Malinowski et al., 2015), element-wise product (Jiasen et al., 2016; Kim et al., 2016, 2017a; Malinowski et al., 2015), Tucker decomposition (Ben-Younes et al., 2017) or compact bilinear pooling (Fukui et al., 2016), and further processed for the downstream task at hand. Attention mechanisms (Xu et al., 2015) are often used to have questions attend to specific spatial locations of the extracted higher-level feature maps.

There are two main reasons for why the recent literature has focused on processing each modality independently. First, using a pretrained convnet as feature extractor prevents overfitting; Despite a large training set of a few hundred thousand samples, backpropagating the error of the downstream task into the weights of all layers often leads to overfitting. Second, the approach aligns with the dominant view that language interacts with high-level visual concepts. Words, in this view, can be thought of as "pointers" to high-level conceptual representations. To the best of our knowledge, this work is the first to fuse modalities at the very early stages of the image processing.

In parallel, the neuroscience community has been exploring to what extent the processing of language and vision is coupled (Ferreira and Tanenhaus, 2007). More and more evidence accumulates that words set visual priors which alter how visual information is processed from the very begin-

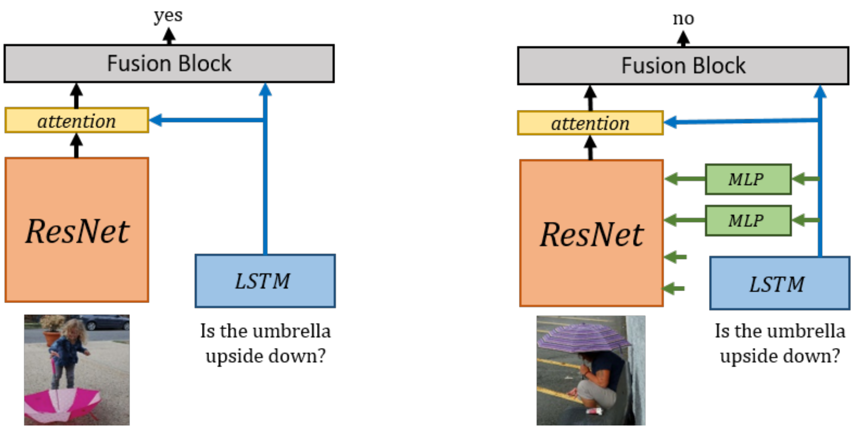


Figure 5.1: An overview of the classic VQA pipeline (left) vs ours (right). While language and vision modalities are independently processed in the classic pipeline, we propose to directly modulate ResNet processing by language.

ning (Boutonnet and Lupyan, 2015; Kok et al., 2014; Thierry et al., 2009). More precisely, it is observed that P1 signals, which are related to low-level visual features, are modulated while hearing specific words (Boutonnet and Lupyan, 2015). The language cue that people hear ahead of an image activates visual predictions and speed up the image recognition process. These findings suggest that independently processing visual and linguistic features might be suboptimal, and fusing them at the early stage may help the image processing.

In this chapter, we introduce a novel approach to have language modulate the *entire* visual processing of a pretrained convnet. We propose to condition the batch normalization (Ioffe and Szegedy, 2015) parameters on linguistic input (e.g., a question in a VQA task). Our approach, called Conditional Bacth Normalization (CBN), is inspired by recent work in style transfer (Dumoulin et al., 2017). The key benefit of CBN is that it scales linearly with the number of feature maps in a convnet, which impacts less than 1% of the parameters, greatly reducing the risk of over-fitting. We apply CBN to a pretrained Residual Network, leading to a novel architecture to which we refer as MODulatEd ResNet (ModeRn). We show significant improvements on two VQA datasets, VQAv1 (Antol et al., 2015) and GuessWhat?! (de Vries et al., 2017), but stress that our approach is a general fusing mechanism that can be applied to other multimodal tasks. To summarize, our contributions are three fold:

- We propose conditional batch normalization to modulate the entire visual processing by language from the early processing stages,
- We condition the batch normalization parameters of a pretrained ResNet on linguistic input, leading to a new network architecture: ModeRn,
- We demonstrate relative improvements on acknowledged strong baseline for two VQA tasks and show the contribution of this modulation on the early stages.

5.2 Modulated Residual Networks

In this section we introduce conditional batch normalization, and show how we can use it to modulate a pretrained ResNet. As a brief reminder from Sec 1.2.3, Batch Normalization (BN) is a technique that was originally designed to accelerate the training of neural networks by reducing the internal

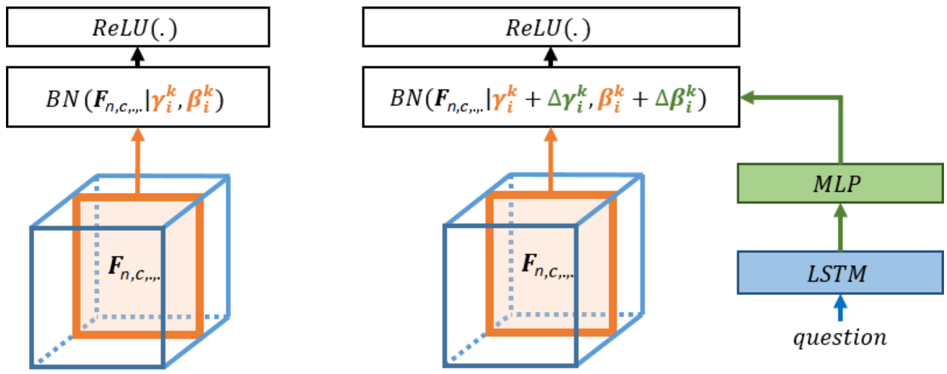


Figure 5.2: An overview of the computation graph of batch normalization (left) and conditional batch normalization (right). Best viewed in color.

co-variate shift (Ioffe and Szegedy, 2015). Given a mini-batch $\mathcal{B} = \{F_{n,\cdot,\cdot,\cdot}\}_{n=1}^{N}$ of N examples, BN normalizes the feature maps at training time as follows:

$$BN(F_{n,c,h,w}|\gamma_c, \beta_c) = \gamma_c \frac{F_{n,c,w,h} - \mathrm{E}_{\mathcal{B}}[F_{\cdot,c,\cdot,\cdot}]}{\sqrt{\mathrm{Var}_{\mathcal{B}}[F_{\cdot,c,\cdot,\cdot}] + \epsilon}} + \beta_c, \tag{5.1}$$

where ϵ is a constant damping factor for numerical stability, and γ_c and β_c are trainable scalars introduced to keep the representational power of the original network.

In the proposed model, the key idea is to predict the $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ of the batch normalization from a language embedding $\boldsymbol{e}_l$. In practice, these parameters must be close to the pretrained ResNet values when starting the training as a poor initialization could significantly deteriorate performance. Unfortunately, it is difficult to initialize a network to output the pretrained $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. For these reasons, we propose to predict a change $\Delta\beta_c$ and $\Delta\gamma_c$ on the frozen original scalars, for which it is straightforward to initialize a neural network to produce an output with zero-mean and small variance.

To do so, we use a one-hidden-layer MLP to predict these deltas from the question embedding $\boldsymbol{e}_l$ for all feature maps within the layer:

$$\boldsymbol{\Delta\beta} = MLP(\boldsymbol{e}_l) \qquad \boldsymbol{\Delta\gamma} = MLP(\boldsymbol{e}_l) \tag{5.2}$$

So, given a feature map with C channels, these MLPs output a vector of size C. We then add these predictions to the $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ parameters:

$$\hat{\beta}_c = \beta_c + \Delta\beta_c \qquad \hat{\gamma}_c = \gamma_c + \Delta\gamma_c \tag{5.3}$$

Finally, these updated $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\gamma}}$ are used as parameters for the batch normalization: $BN(F_{n,c,h,w}|\hat{\gamma}_c, \hat{\beta}_c))$, which results in the following equation:

$$CBN(F_{n,c,h,w}, \boldsymbol{e}_l|\gamma_c, \beta_c) = \Big(\gamma_c + \Delta\gamma_c(\boldsymbol{e}_l)\Big) \frac{F_{n,c,w,h} - \mathrm{E}_{\mathcal{B}}[F_{\cdot,c,\cdot,\cdot}]}{\sqrt{\mathrm{Var}_{\mathcal{B}}[F_{\cdot,c,\cdot,\cdot}] + \epsilon}} + \Big(\beta_c + \Delta\beta_c(\boldsymbol{e}_l)\Big), \tag{5.4}$$

We stress that we freeze all ResNet parameters, including the original $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, during training. In Fig. 5.2, we visualize the difference between the computational flow of the original batch normalization and our proposed modification. As explained in Sec. 2.1.2, a ResNet consists of four stages

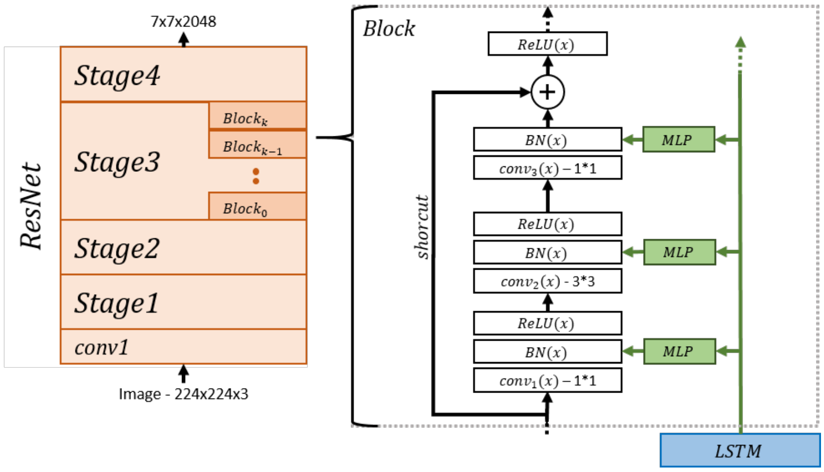


Figure 5.3: An overview of the ModeRn architecture conditioned on the language embedding. ModeRn modulates the batch norm parameters in all residual blocks.

of computation, each subdivided in several residual blocks. In each block, we apply CBN to the three convolutional layers, as highlighted in Fig. 5.3.

CBN is a computationally efficient and powerful method to modulate neural activations; It enables the linguistic embedding to manipulate entire feature maps by scaling them up or down, negating them, or shutting them off, etc. As there only two parameters per feature map, the total number of BN parameters comprise less than 1% of the total number of parameters of a pretrained ResNet. This makes CBN a very scalable method compared to conditionally predicting the weight matrices (or a low-rank approximation to that). Although, we here focus on visual feature maps and linguistic representations, we point out that CBN layers can be extended to any multimodal learning pipeline.

5.3 Experimental setting

We evaluate the proposed conditional batch normalization on the first VQA dataset and the GuessWhat?! Oracle tasks. In the next section, we outline the neural architectures we use for our experiments. Noticeably, ModeRn modulates the entire visual processing pipeline and therefore backpropagates through all convolutional layers. This requires much more GPU memory than using extracted features. To feasibly run such experiments on today's hardware, we conduct all experiments in this chapter with a ResNet-50 except when mention otherwise.

VQA As mentioned several time, VQA consists of open-ended questions about real images (Antol et al., 2015). Our baseline architecture first obtains a question embedding $\boldsymbol{e}_l$ by an LSTM. For the image, we extract the feature maps $\boldsymbol{F}$ of the last layer of ResNet-50 (before the pooling layer). For input of size $224x224$ these feature maps are of size $7x7$, and we incorporate a spatial attention mechanism, conditioned on the question embedding $\boldsymbol{e}_l$, to pool over the spatial dimensions. We use an MLP with one hidden layer and ReLU activations whose parameters are shared along the spatial dimensions. In a few words, we are using the original VQA models with an extra spatial attention mechanism. As for our training procedure, we select the 2k most-common answers from the training set, and use a cross-entropy loss over the distribution of provided answers. We train on the training set, do early-stopping on the validation set, and report the accuracies on the test-dev using the evaluation script provided by (Antol et al., 2015).

	Answer type	Yes/No	Number	Other	**Overall**
224x224	Baseline	79.45%	36.63%	44.62%	58.05%
	Ft Stage 4	78.37%	34.27%	43.72%	56.91%
	Ft BN	80.18%	35.98%	46.07%	58.98%
	ModeRn	81.17%	37.79%	48.66%	60.82%
448x448	MLB (Kim et al., 2017a) with ResNet-50	80.20%	37.73%	49.53%	60.84%
	MLB (Kim et al., 2017a) with ResNet-152	80.95%	38.39%	50.59%	61.73%
	MUTAN + MLB (Ben-Younes et al., 2017)	82.29%	37.27%	48.23%	61.02%
	MCB + Attention (Fukui et al., 2016) with ResNet-50	60.46%	38.29%	48.68%	60.46%
	MCB + Attention (Fukui et al., 2016) with ResNet-152	-	-	-	62.50%
	ModeRn with ResNet-50	81.38%	36.06%	51.64%	62.16%
	ModeRn + MLB (Kim et al., 2017a) with ResNet-50	82.17%	38.06%	52.29%	**63.01%**

Table 5.1: VQA accuracies trained with train set and evaluated on test-dev.

CBN applied to	Valid. accuracy	Relative Improv.
$\emptyset$	56.12%	-
Stage 4	57.68%	+1.56%
Stages $3-4$	58.29%	+0.61%
Stages $2-4$	58.32%	+0.03%
All	**58.56%**	+0.24%

(a) VQA, higher is better

CBN applied to	Test error	Relative Improv.
$\emptyset$	29.92%	-
Stage 4	26.42%	+3.50%
Stages $3-4$	25.24%	+1.18%
Stages $2-4$	25.31%	-0.07%
All	**25.06%**	+0.25%

(b) GuessWhat?!, lower is better

Table 5.2: Ablation study to investigate the impact of leaving out the lower stages of ResNet. The last column compute the relative improvement in percentage points by modulating an extra-layer.

GuessWhat?! In this chapter, we focus on the Oracle task, which is a form of visual question answering in which the answers are limited to <yes>, <no>and <na>. Specifically, the Oracle may take as an input the incoming question $\boldsymbol{q}$, the image $\mathcal{I}$ and the target object $o*$. This object is described with its category, its spatial location and the object crop. We here use the crop baseline architecture from chapter 4, where we merely replace the VGGNet features with ResNet features and we append a spatial attention mechanism over the crop.

Baselines For VQA, we report the results of two concurrent state-of-the-art architectures from the VQA challenge 2016 and 2017, namely, MCB (Fukui et al., 2016) (and MUTAN (Ben-Younes et al., 2017). Both approaches employ an (approximate) bilinear pooling mechanism to fuse the language and vision embedding by respectively using a random projection and a tensor decomposition. In addition, we re-implement two spatial attention mechanisms in our models: a standard attention that concatenates $\boldsymbol{e}_l$ and $\boldsymbol{e}_v$, and the MLB attention mechanism (Kim et al., 2017a) described in Eq 3.3 in Sec. 3.1.3. When benchmarking state-of-the-art models, we train our architecture on the training set, proceed early stopping on the validation set and report accuracy on the test set (test-dev in the case of VQA.)

5.3.1 Results

VQA We report the best validation accuracy of the outlined methods on the VQA task in Tab. 5.1. Note that we use input images of size $224x224$ when we compare ModeRn against the baselines (as well as for the ablation study presented in Tab. 5.2a. Our initial baseline achieves 58.05% accuracy, and we find that finetuning the last layers (*Ft Stage 4*) does not improve this performance (56.91%). Interestingly, just finetuning the batch norm parameters (Ft BN) significantly improves the accuracy to

	Raw ft	MLB	ft stage4	Ft BN	CBN
Crop	29.92%	30.15%	27.48%	27.94%	25.06%
Crop + Spatial + Category	22.55%	22.95%	22.68%	22.42%	**19.52%**
Spatial + Category	21.5%				

Table 5.3: GuessWhat?! test errors for the Oracle model with different embeddings. Lower is better.

58.98%. We see another significant performance jump when we condition the batch normalization on the question input (*ModeRn*), which improves our baseline with almost 2 accuracy points to 60.82%.

Because state-of-the-art models use images of size 448x448, we also include the results of the baseline architecture on these larger images. As seen in Tab. 5.1, this nearly matches the state of the art results with a 62.15%. As ModeRn does not rely on a specific attention mechanism, we then combine our proposed method with MLB (Kim et al., 2017a) architecture, and observe that outperforms the historical state-of-the-art MCB model (Fukui et al., 2016) by half a point. Please note that we select MLB (Kim et al., 2017a) over MCB (Fukui et al., 2016) as the latter requires fewer weight parameters and is more stable to train.

Note that the presented results use a ResNet-50 while other models rely on extracted image embedding from a ResNet-152. For sake of comparison, we run the baseline models with extracted image embedding from a ResNet-50. Also for the more advanced MLB architecture, we observe performance gains of approximately 2 accuracy points.

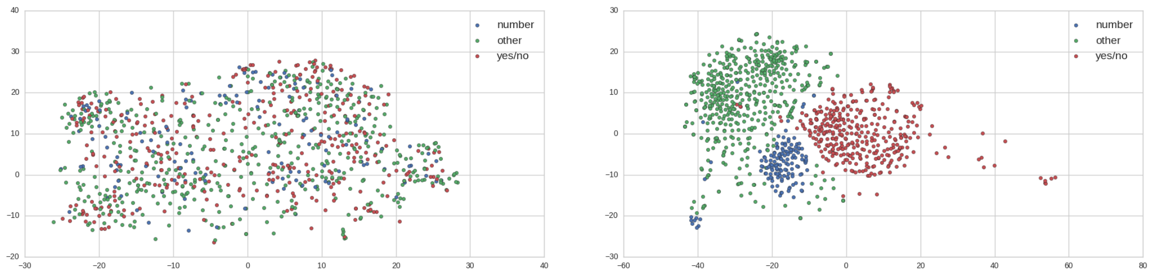
GuessWhat?! We report the best test errors for the outlined method on the Oracle task of GuessWhat?! in Tab. 5.3. We first compare the results when we only feed the crop of the selected object to the model. We observe the same trend as in VQA: with an error of 25.06%, CBN performs better than than either fine-tuning the final block (27.48% error) or the batch-norm parameters (27.94% error), which in turn improve over just using the raw features (29.92% error). Interestingly, the MLB spatial attention does not significantly improve the Oracle score over raw features with an accuracy of XX%. Note that the relative improvement (5 error points) for CBN is much bigger for GuessWhat?! than for VQA. We also investigate the performance of the methods when we include the spatial and category information. We observe that finetuning the last layers or BN parameters does not improve the performance, while ModeRn improves the best reported test error with 2 points to 19.52% error.

5.3.2 Discussion

By analyzing the results from both VQA and GuessWhat?! experiments, it is possible to have a better insight regarding ModeRn capabilities.

MODERN vs Fine tuning In both experiments, ModeRn outperforms Ft BN. Both methods update the same ResNet parameters so this demonstrates that it is important to condition on the language representation. ModeRn also outperforms Ft Stage 4 on both tasks which shows that the performance gain of ModeRn is not due to the increased model capacity, but to the conditioning mechanism.

Conditional embedding In the provided baselines of the Oracle task of GuessWhat?! (de Vries et al., 2017), the authors observed that the best test error (21.5%) is obtained by only providing the object category and its spatial location. For this model, including the raw features of the object crop actually deteriorates the performance to 22.55% error. This means that this baseline fails to extract relevant information from the images which is not in the handcrafted features. Therefore the Oracle



(a) Feature map projection from raw ResNet (b) Feature map projection from ModeRn

Figure 5.4: t-SNE projection of feature maps (before attention mechanism) of ResNet and ModeRn. Points are colored according to the answer type of VQA. Whilst there are no clusters with raw features, ModeRn successfully modulates the image feature towards specific answer types.

can not answer correctly questions which requires more than the use of spatial information and object category. In the baseline model, the embedding of the crop from a generic ResNet does not help even when we finetune stage 4 or BN. In contrast, applying ModeRn helps to better answer questions as the test error drops by 2 points.

Ablation study We investigate the impact of only modulating the top layers of a ResNet. We report these results in Tab. 5.2. Interestingly, we observe that the performance slowly decreases when we apply CBN exclusively to later stages. We stress that for best performance it is important to modulate all stages, but if computational resources are limited we recommend to apply it to the two last stages. This crucial observation would pave the way for the Feature Wise Linear Modulation (FiLM) architecture explored in the next chapter.

Modulation vs Attention In VQA, the attention and modulation mechanisms seem to be complementary as they independently increase the model accuracy, and can be combined to improve the final performance. In GuessWhat?!, spatial attention has little impact while modulation turns out to be very successful. In both cases, it suggests that both procedures would fuse the language and modalities differently. It is unclear whether this difference comes from the nature of the mechanism itself (with different mathematical equations), or whether from the conditioning level. As mentioned in Sec. 3.1.2, attention is a mid-late conditioning mechanism, fusing high-level features while CBN is a middle conditioning mechanism that impacts lower-level concepts. However, CBN is the most impactful while modulating the uppermost stages of the ResNet.

Visualizing the representations In order to gain more insight into our proposed fusion mechanism, we compare visualizations of the visual embeddings created by our baseline model and ModeRn. We first randomly picked 1000 unique image/question pairs from the *validation set* of VQA. For the trained ModeRn model, we extract image features just before the attention mechanism of ModeRn, which we will compare with extracted raw ResNet-50 features and finetune ResNet-50 (Block4 and batchnorm parameters). We first decrease the dimensionality by average pooling over the spatial dimensions of the feature map, and subsequently apply t-Distributed Stochastic Neighbor Embedding (t-SNE) (van G. der and Hinton, 2008) to these set of embeddings. We color the points according to the answer type provided by the VQA dataset, and show these visualizations for both models in Fig 5.4 and Fig 5.7 in the supplementary materials. Interestingly, we observe that all answer types are spread out for raw image features and finetuned features. In contrast, the representations of ModeRn are cleanly grouped into three answer types. This demonstrates that ModeRn successfully disentangles the images representations by answer type which is likely to ease the later fusion process. While finetuning models does cluster features, there is no direct link between those clusters and the answer type. These results indicate that ModeRn successfully learns representation that differs from classic finetuning strategies. In Fig. 5.5, we visualize the feature disentangling process stage by stage. It is possible to spot some sub-clusters in the t-SNE representation, as in fact they correspond to image and question pairs which are similar but not explicitly tagged in the VQA dataset. For example, the we highlight pairs where the answer is a color in Fig. 5.6.

5.4 Discussion

5.4.1 Related Work

Modulation was initially introduced in the context of image stylization (Dumoulin et al., 2017; Ghiasi et al., 2017). The authors aimed to condense multiple image style transfer networks into a single model by finetuning a minimum subset of parameters for each style. There are notable differences with our work. First, Dumoulin et al. (2017) uses a non-differentiable table lookup for the normalization parameters while we propose a differentiable mapping from the question embedding. Second, we predict a change on the normalization parameters of a pretrained convolutional network while keeping the convolutional filters fixed. More generally, we here show that modulation is a more generic framework than initially thought, and we exhibit modulation efficiency through language-vision tasks. Subsequent works then alleviate the generality of CBN to other multimodal tasks: Delbrouck and Dupont (2017) performed multimodal translation by modulating the late-stage of the visual pipeline for each new generated word. CBN layers have been used in the GAN litterature to conditioned the image generation (Almahairi et al., 2018; Brock et al., 2019; Miyato and Koyama, 2018).

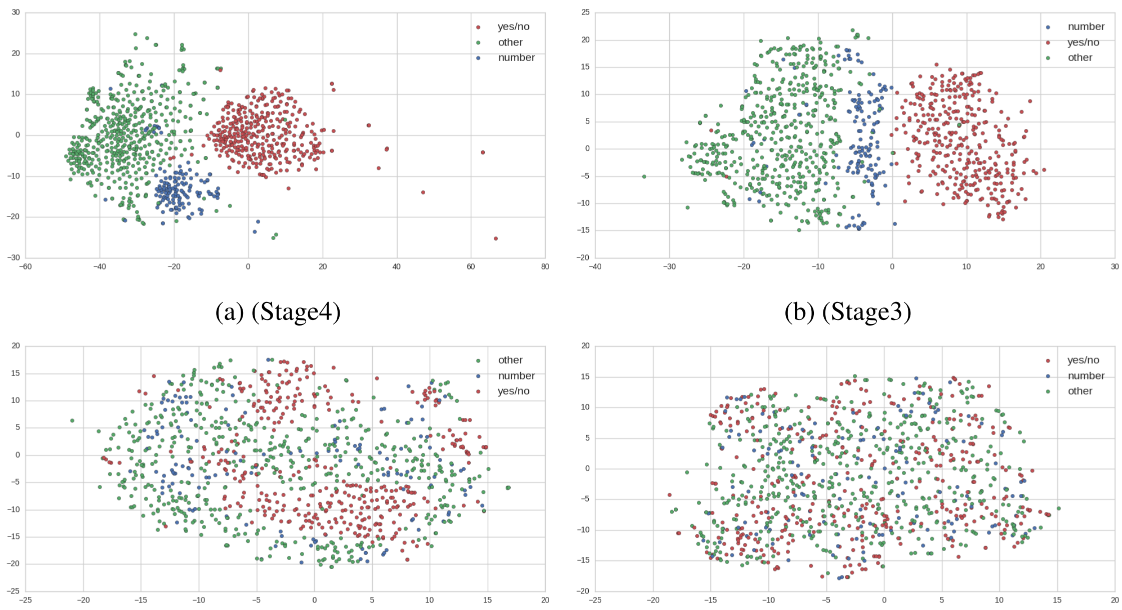
5.4.2 Conclusion

In this chapter, we introduce Conditional Bacth Normalization (CBN) as a novel fusion mechanism to modulate all layers of a visual processing network. Specifically, we applied CBN to a pretrained ResNet, leading to the proposed ModeRn architecture. Our approach is motivated by recent evidence from neuroscience suggesting that language influences the early stages of visual processing. One of the strengths of ModeRn is that it can be incorporated into existing architectures, and our experiments demonstrate that this significantly improves the baseline models. We also found that it is important to modulate the entire visual signal to obtain maximum performance gains.

While this chapter focuses on text and images, ModeRn can be extended to neural architecture dealing with other modalities such as sound or video. More broadly, CBN can could also be applied to modulate the internal representation of any deep network with respect to any embedding regardless of the underlying task. For instance, signal modulation through batch norm parameters may also be beneficial for reinforcement learning, natural language processing or adversarial training tasks.

		GuessWhat?!	VQA
Question	word embedding size	300	300
	number of LSTM	1	2
	number of LSTM hidden units	1024	1024
	use Glove	False	True
Object category	number of categories	90	N/A
	category look-up table dimension	512	N/A
Image	image size	224x224x3 (crop 1.1)	224x224x3
	attention units	512 (MLB only)	512
CBN	selected blocks	all	all
	number of MLP hidden units	512	512
	ResNet	ResNet-50v1	ResNet-50v1
Fusion block	fusion embedding size	N/A	1024
	number of MLP hidden units	512	512
	number of answers	3	2000
Optimizer	Name	Adam	Adam
	Learning rate	1e-4	2e-4
	Clip value	3	5
	number of epoch	10	20
	batch size	32	32

Table 5.4: GuessWhat?! Oracle hyperparameters



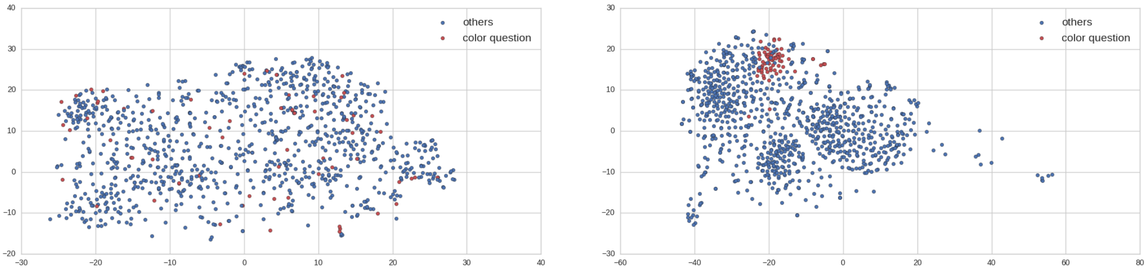
(a) (Stage4)

(b) (Stage3)

(c) (Stage2)

(d) (Stage1)

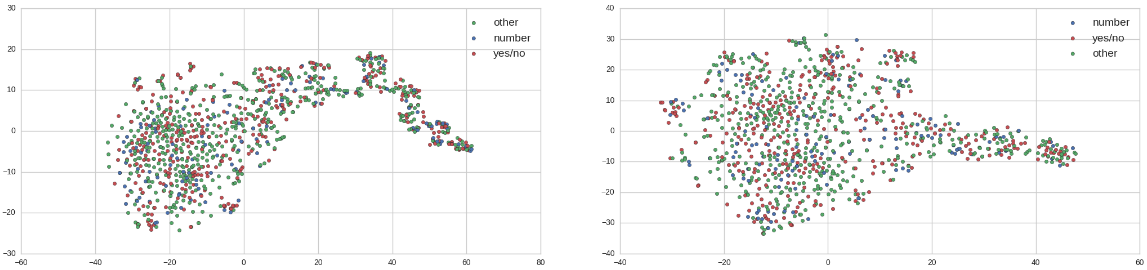
Figure 5.5: t-SNE projection for ModeRn for each stage of ModeRn. The visual features are slowly clustered blocks after blocks.



(a) Raw ResNet

(b) ModeRn

Figure 5.6: t-SNE projection of feature maps of Reset and ModeRn by coloring. Points are colored according to the question type (here, colors) of the image/question pair from the VQA dataset.



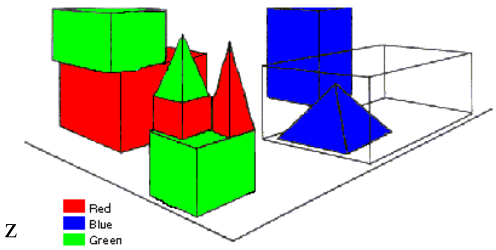
(a) ResNet + Block4 Ft

(b) ResNet + BatchNorm ft

Figure 5.7: t-SNE projection of finetune ResNet feature maps (before attention mechanism). Points are colored according to the answer type of VQA. No answer-type clusters can be observed in both cases.

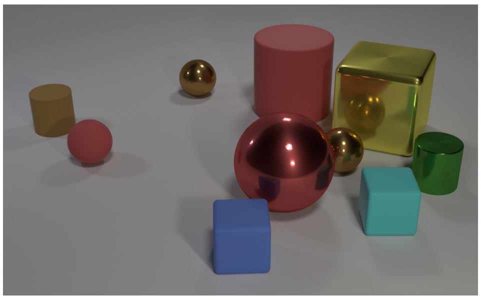
Chapter 6

Visual Reasoning with a General Modulation Layer



«Is there a large block behind a green pyramid? »

SHRDLU by Winograd (1971)



«Is there a large block behind the green cylinder? »

CLEVR by Johnson et al. (2017a)

Contents

In the previous chapter, we introduce an original conditioning mechanism, referred to as modulation, by predicting the batch normalization parameters of a pretrained ResNet. The results were promising, but several questions were left open in this initial work. Is the modulation tied to batch-normalization? Can we get rid of the heavy ResNet? Can we jointly train the modulating and modulated blocks from scratch? How does modulation operate? If we wanted to support our claim that modulation was a new generic and multimodal neural fusing block, we had to answer those questions first. The recently released CLEVR dataset allowed us to answer our interrogations without the actual burden of natural images and dialogues. Besides, spatial attention mechanisms were shown to perform on this visual reasoning task poorly. Thus, if we could improve the state-of-the-art by only using modulation layers, this would provide additional clues that both mechanisms are complementary. In practice, the results were far better than we first expected.

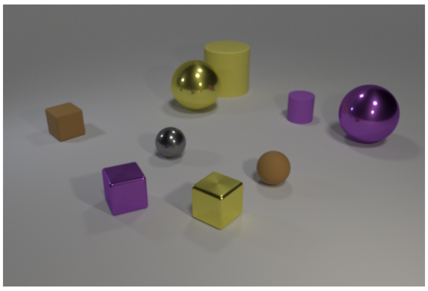
In this chapter, we thus introduce a general-purpose conditioning method for neural networks called Feature Wise Linear Modulation (FiLM). FiLM layers influence neural network computation via a simple, feature-wise affine transformation based on conditioning information. We show that FiLM layers are highly effective for visual reasoning — answering image-related questions which require a multi-step, high-level process — a task which has proven difficult for standard deep learning methods that do not explicitly model reasoning. Specifically, we show on visual reasoning tasks that FiLM layers 1) halve state-of-the-art error for the CLEVR benchmark, 2) modulate features in a coherent manner, 3) are robust to ablations and architectural modifications, and 4) generalize well to challenging, new data from few examples or even zero-shot.

6.1 Introduction

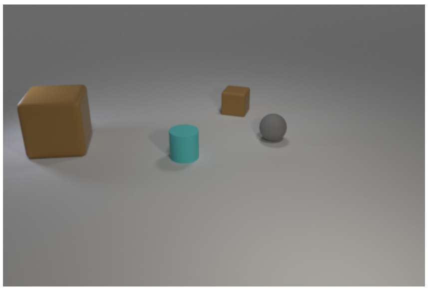
The ability to reason about everyday visual input is a fundamental building block of human intelligence. Some have argued that for artificial agents to learn this complex, structured process, it is necessary to build in aspects of reasoning, such as compositionality (Hu et al., 2017b; Johnson et al., 2017b) or relational computation (Santoro et al., 2017). However, if a model made from general-purpose components could learn to visually reason, such an architecture are likely to be more widely applicable across domains.

To understand if such a general-purpose architecture exists, we take advantage of the recently proposed CLEVR dataset (Johnson et al., 2017a) that tests visual reasoning via question answering. Examples from CLEVR are shown in Figure 6.1. Visual question answering, the general task of asking questions about images, has its own line of datasets (Antol et al., 2015; Geman et al., 2015; Malinowski and Fritz, 2014) which generally focus on asking a diverse set of simpler questions on images, often answerable in a single glance. From these datasets, a number of effective, general-purpose deep learning models have emerged for visual question answering (Anderson et al., 2018b; Jiasen et al., 2016; Malinowski et al., 2015; Yang et al., 2016). However, tests on CLEVR show that these general deep learning approaches struggle to learn structured, multi-step reasoning (Johnson et al., 2017a). In particular, these methods tend to exploit biases in the data rather than capture complex underlying structure behind reasoning (Agrawal et al., 2016; Goyal et al., 2017).

In this work, we show that a general model architecture can achieve strong visual reasoning with a method we introduce as Feature Wise Linear Modulation (FiLM). This modulation layer carries out a simple, feature-wise affine transformation on a neural network's intermediate features, conditioned on an arbitrary input. In the case of visual reasoning, it enable a RNN over an input question to influence



Q: *What number of cylinders are small purple things or yellow rubber things?* **A:** ***2***



Q: *What color is the other object that is the same shape as the large brown matte thing?* **A:** ***Brown***

Figure 6.1: CLEVR examples and FiLM model answers.

CNN computation over an image. This process adaptively and radically alters the CNN's behavior as a function of the input question, allowing the overall model to carry out a variety of reasoning tasks, ranging from counting to comparing, for example. FiLM can be thought of as a generalization of conditional normalization, which has proven highly successful for image stylization (Dumoulin et al., 2017; Ghiasi et al., 2017; Huang and Belongie, 2017), speech recognition (Kim et al., 2017b), and visual question answering as in Chapter 5, demonstrating modulation's broad applicability.

In this chapter, we show that FiLM is a strong conditioning method by showing the following on visual reasoning tasks:

- It achieves state-of-the-art across a variety of visual reasoning tasks, often by significant margins.
- It operates in a coherent manner. It learns a complex, underlying structure and manipulates the conditioned network's features in a selective manner. It also enables the CNN to properly localize question-referenced objects.
- It is robust; several ablated model still outperform prior state-of-the-art. Notably, we find there is no close link between normalization and the success of a conditioned affine transformation, a previously untouched assumption. Thus, we relax the conditions under which this method can be applied.
- It models learn from little data to generalize to more complex and/or substantially different data than seen during training. We also introduce a novel FiLM-based zero-shot generalization method that further improves and validates the modulation layer's generalization capabilities.

6.2 Model Modulation

Our model processes the question-image input using FiLM, illustrated in Figure 6.2. We start by explaining the modulation procedure and then describe our particular model for visual reasoning.

6.2.1 Feature-wise Linear Modulation

FiLM learns to adaptively influence the output of a neural network by applying an affine transformation to the network's intermediate features, based on some input. More formally, this layer learns two functions $f(.)$ and $h(.)$ which output γ and β as a function of input $\boldsymbol{q}$:

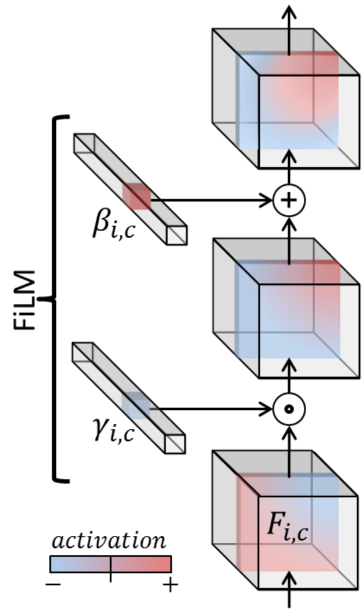


Figure 6.2: A single FiLM layer for a CNN. The dot signifies a Hadamard product. Various combinations of γ and β can modulate individual feature maps in a variety of ways.

$$\gamma_c = f_c(\boldsymbol{x}) \qquad\qquad \beta_c = h_c(\boldsymbol{x}), \tag{6.1}$$

where $\boldsymbol{x}$ is some arbitrary input, γ_c and β_c modulate a neural network's activations $\boldsymbol{F}_c$, whose subscripts refer to the i^{th} input's c^{th} feature or feature map, via a feature-wise affine transformation:

$$FiLM(\boldsymbol{F}_c, \boldsymbol{x}|\gamma_c, \beta_c) = \gamma_c(\boldsymbol{x})\boldsymbol{F}_c + \beta_c(\boldsymbol{x}). \tag{6.2}$$

f and h can be arbitrary functions such as neural networks. Modulation of a target neural network's processing can be based on the same input to that neural network or some other input, as in the case of multi-modal or conditional tasks. For CNNs, f and h thus modulate the per-feature-map distribution of activations based on $\boldsymbol{x}_n$, agnostic to spatial location. In practice, it is easier to refer to f and h as a single function that outputs one $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ vector, since, for example, f and h can be a joint MLP. We refer to this single function as the FiLM-generator. We also refer to the network to which FiLM layers are applied as the modulated (or FiLM) network network.

FiLM layers empower the FiLM-generator to manipulate feature maps of the modulated network by scaling them up or down, negating them, shutting them off, selectively thresholding them (when followed by a ReLU), and more. Each feature map is conditioned independently, giving the FiLM-generator moderately fine-grained control over activations at each FiLM layer. As the modulation process only requires two parameters per modulated feature map, it is a scalable and computationally efficient conditioning method, and it is not impacted by the image resolution.

6.2.2 Modulated Residual Blocks

Our model consists of a linguistic pipeline and a modulated visual pipeline as depicted in Figure 6.3. The FiLM-generator processes a question $\boldsymbol{q}_n$ using a GRU network (Chung et al., 2014) with 4096 hidden units that takes in learned, 200-dimensional word embeddings. The final GRU hidden state

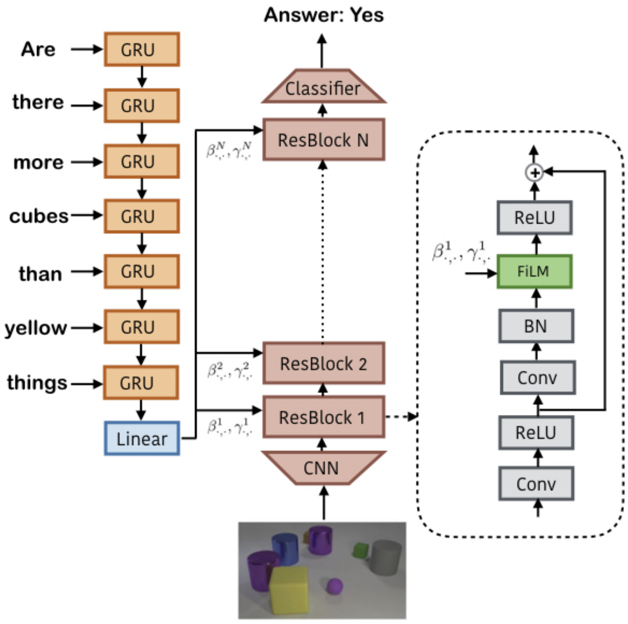


Figure 6.3: The FiLM generator, FiLM network, and residual block architecture of our model.

is a question embedding, from which the model predicts $(\gamma_{n,\cdot}^{k}, \beta_{n,\cdot}^{k})$ for each k^{th} residual block via affine projection.

The visual pipeline extracts 128 14×14 image feature maps from a resized, 224×224 image input using either a CNN trained from scratch or a fixed, pretrained feature extractor with a learned layer of 3×3 convolutions. The CNN trained from scratch consists of 4 layers with 128 4×4 kernels each, ReLU activations, and batch normalization, similar to prior work on CLEVR (Santoro et al., 2017). The fixed feature extractor outputs the *conv4* layer of a ResNet-101 (He et al., 2016) pretrained on ImageNet (Russakovsky et al., 2015) to match prior work on CLEVR (Johnson et al., 2017a,b). Image features are processed by several — 4 for our model — modulated Residual Block (ResBlock) with 128 feature maps and a final classifier. The classifier consists of a 1×1 convolution to 512 feature maps, global max-pooling, and a two-layer MLP with 1024 hidden units that outputs a softmax distribution over final answers.

Each modulated ResBlock starts with a 1×1 convolution followed by one 3×3 convolution with an architecture as depicted in Figure 6.3. We turn the parameters of batch normalization layers that immediately precede FiLM layers off. Drawing from prior work on CLEVR (Hu et al., 2017b; Santoro et al., 2017) and visual reasoning (Watters et al., 2017), we concatenate two coordinate feature maps indicating relative x and y spatial position (scaled from -1 to 1) with the image features, each ResBlock's input, and the classifier's input to facilitate spatial reasoning.

We train our model end-to-end from scratch with Adam (Kingma and Ba, 2015) (learning rate $3e^{-4}$), weight decay ($1e^{-5}$), batch size 64, and batch normalization and ReLU throughout FiLM network. Our model uses only image-question-answer triplets from the training set without data augmentation. We employ early stopping based on validation accuracy, training for 80 epochs maximum. Further model details are in the appendix. Empirically, we found FiLM had a large capacity, so many architectural and hyperparameter choices were for added regularization.

We stress that our model relies *solely* on feature-wise affine conditioning to use question information influence the visual pipeline behavior to answer questions; and no late-fusion mechanism

Model	**Overall**	Count	Exist	Compare Numbers	Query Attribute	Compare Attribute
Human (Johnson et al., 2017b)	92.6	86.7	96.6	86.5	95.0	96.0
Q-type baseline (Johnson et al., 2017b)	41.8	34.6	50.2	51.0	36.0	51.3
LSTM (Johnson et al., 2017b)	46.8	41.7	61.1	69.8	36.8	51.8
CNN+LSTM (Johnson et al., 2017b)	52.3	43.7	65.2	67.1	49.3	53.0
Stacked Attention Network (Santoro et al., 2017)	76.6	64.4	82.7	77.4	82.6	75.4
Neural Module Network* (Hu et al., 2017b)	83.7	68.5	85.7	84.9	90.0	88.7
PG+EE (9K prog.)* (Johnson et al., 2017b)	88.6	79.7	89.7	79.1	92.6	96.0
PG+EE (700K prog.)* (Johnson et al., 2017b)	96.9	92.7	97.1	98.7	98.1	98.9
Relational Network†‡ (Santoro et al., 2017)	95.5	90.1	97.8	93.6	97.9	97.1
CNN+BN+Sum ⋄ (Malinowski and Doersch, 2018)	95.5	91.0	98.5	84.7	98.4	98.7
CMM ⋄ (Yao et al., 2018)	98.6	96.8	99.2	97.7	99.4	99.1
MAC ⋄ (Hudson and Manning, 2018)	**98.9**	**97.1**	**99.5**	**99.1**	**99.5**	**99.5**
FiLM	97.7	94.3	99.1	96.8	99.1	99.1
FiLM‡	97.6	94.3	99.3	93.4	99.3	99.3

Table 6.1: CLEVR accuracy (overall and per-question-type) by baselines, competing methods, and FiLM. (*) denotes use of extra supervision via program labels. (†) denotes use of data augmentation. (‡) denotes training from raw pixels. (⋄) denotes model posterior to FiLM

is performed. This approach differs from classical visual question answering pipelines which fuse image and language information into a single embedding via element-wise product, concatenation, attention, and/or more advanced methods (Anderson et al., 2018b; Jiasen et al., 2016; Yang et al., 2016).

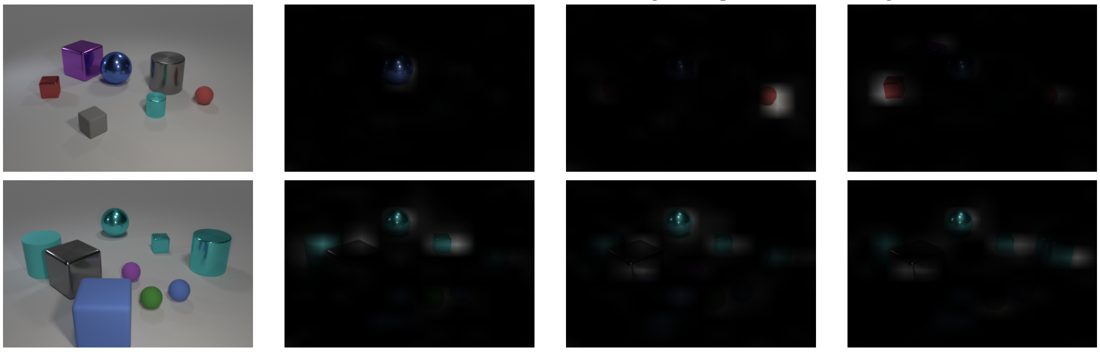
6.3 Experimenting FiLM on CLEVR

First, we test our model on visual reasoning with the CLEVR task and use trained FiLM models to analyze what FiLM learns. Second, we explore how well our model generalizes to more challenging questions with the CLEVR-Humans task. Finally, we examine how FiLM performs in few-shot and zero-shot generalization settings using the CLEVR Compositional Generalization Test. In the appendix, we provide an error analysis of our model. Two version of the code co-exist at `https://github.com/ethanjperez/FiLM` (pytorch) and `https://github.com/GuessWhatGame/clevr` (tensorflow).

6.3.1 CLEVR Task

As mentioned in Sec 3.2, CLEVR is a synthetic dataset of 700K (image, question, answer, program) tuples (Johnson et al., 2017a). Images contain 3D-rendered objects of various shapes, materials, colors, and sizes. Questions are multi-step and compositional in nature, as shown in Figure 6.1. They range from counting questions (*"How many green objects have the same size as the green metallic block?"*) to comparison questions (*"Are there fewer tiny yellow cylinders than yellow metal cubes?"*) and can be 40+ words long. Answers are each one word from a set of 28 possible answers. Programs are an additional supervisory signal consisting of step-by-step instructions, such as `filter_shape[cube]`, `relate[right]`, and `count`, on how to answer the question.

Q: *What shape is the...* | *...purple thing?* **A:** *cube* | *...red thing right of the blue thing?* **A:** *sphere* | *...red thing left of the blue thing?* **A:** *cube*



Q: *How many cyan things are...* | *...left of the small cube?* **A:** *2* | *...right of the gray cube and left of the small cube?* **A:** *1* | *...right of the gray cube or left of the small cube?* **A:** *4* (**P:** *3*)

Figure 6.4: Visualizations of the distribution of locations which the model uses for its globally max-pooled features which its final MLP predicts from. FiLM correctly localizes the answer-referenced object (top) or all question-referenced objects (bottom), but not as accurately when it answers incorrectly (rightmost bottom). Questions and images used match (Johnson et al., 2017b).

Baselines We compare our approach against very various visual reasoning models discussed in Sec 3.2 and Sec 3.1.3. As a naive baseline, we refer to Q-type while predicting the answer based on a question's category, LSTM when dealing only with the question and CNN+LSTM while concatenating mean-pooled visual features and question features. We benchmark spatial attention by using SAN Yang et al. (2016) networks with two spatial attention hops, and relational networks that perform pairwise comparisons over spatial locations Santoro et al. (2017). We also add NMN (Hu et al., 2017b) and (PG+EE) (Johnson et al., 2017b) that use the CLEVR program to learn separate specialized neural modules and dynamically assemble them into a question-dependent network. Posterior to our work, Multimodal Core (CNN+BN+Sum) tiles the LSTM features over the spatial dimensions before processing them with batch-normalization and a convolutional layer (Malinowski and Doersch, 2018). MAC network decomposes the network into a reading, writing and controlling units to impose structural constraints to perform visual reasoning (Hudson and Manning, 2018). Finally, Cascaded Mutual Modulation (CMM) (Yao et al., 2018) is a variant of Multi-hop FiLM (Multi-hop FiLM) Strub et al. (2018b), and computes a distinct FiLM embedding for each ResBlock.

Results When first released, FiLM achieved a new overall state-of-the-art on CLEVR, outperforming humans and previous methods, including those using explicit models of reasoning, program supervision, and/or data augmentation. A few subsequent works later significantly outperforms our approach (Hudson and Manning, 2018; Yao et al., 2018), but the improvement plateaued since then as shown in Tab. 6.1. For methods not using extra supervision, FiLM roughly halves state-of-the-art error (from 4.5% to 2.3%). Note that using pretrained image features as input can be viewed as a form of data augmentation in itself but that FiLM performs equally well using raw pixel inputs. Finally, we train on CLEVR for 80 epochs, which takes 4 days using 1 NVIDIA TITAN Xp GPU when learning from image features.

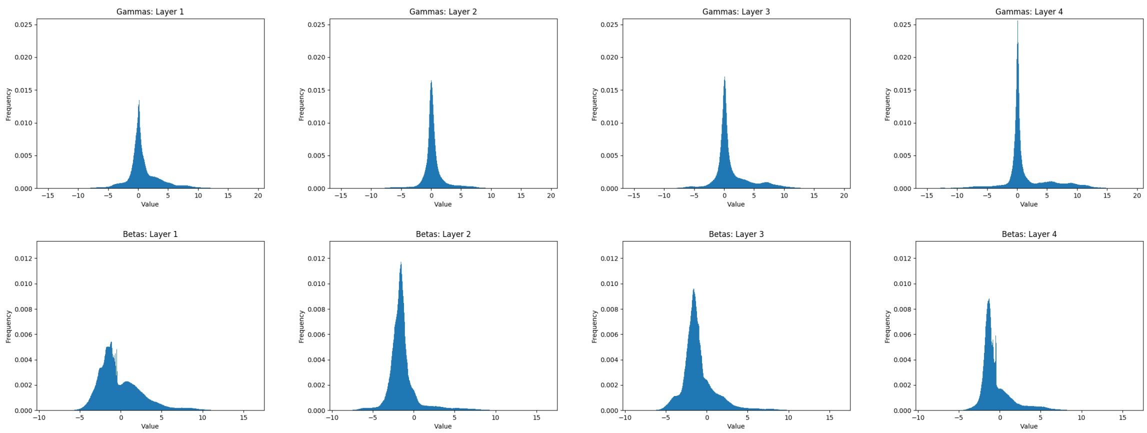


Figure 6.5: Histograms of $\gamma_{n,c}$ (top) and $\beta_{n,c}$ (bottom) values for each FiLM layer (layers 1-4 from left to right) computed on CLEVR's validation set. Plots are scaled identically. FiLM layers appear gradually more selective and higher variance.

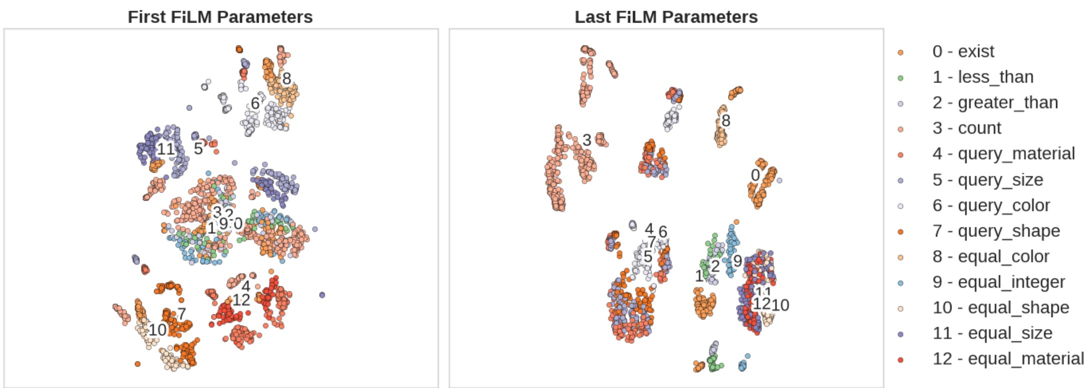


Figure 6.6: t-SNE plots of (γ, β) of the first (left) and last (right) FiLM layers of a 6-FiLM layer Network. FiLM parameters cluster by low-level reasoning functions in the first layer and by high-level reasoning functions in the last layer.

6.3.2 What Do FiLM Layers Learn?

We here describe experiments we performed to get a better insight over the modulation process.

Activation Visualizations Figure 6.4 visualizes the distribution of locations responsible for the globally-pooled features which the MLP in the model's final classifier uses to predict answers. These images reveal that the FiLM model predicts using features of areas near answer-related or question-related objects. This finding highlights that appropriate feature modulation indirectly results in spatial modulation, as regions with question-relevant features will have large activations while other regions will not. Figure 6.4 also suggests that the FiLM network carries out reasoning throughout its pipeline. In the top example, the modulated network has localized the answer-referenced object alone before the MLP classifier. In the bottom example, the FiLM network retains, for the MLP classifier, features on objects that are not referred to by the answer but are referred to by the question. The latter example provides evidence that the final MLP itself carries out some reasoning, using FiLM to extract relevant features for its reasoning.

FiLM Parameter Histograms To analyze at a lower level how FiLM uses the question to condition the visual pipeline, we plot γ and β values predicted over the validation set, as shown in Figure 6.5. γ and β values take advantage of a sizable range, varying from -15 to 19 and from -9 to 16, respectively. γ values show a sharp peak at 0, showing that FiLM learns to use the question to shut off or significantly suppress whole feature maps. Simultaneously, FiLM learns to upregulate a much more selective set of other feature maps with high magnitude γ values. Furthermore, a large fraction (36%) of γ values are negative; since our model uses a ReLU after FiLM, $\gamma < 0$ can cause a significantly different set of activations to pass the ReLU to downstream layers than $\gamma > 0$. Together, these findings suggest that FiLM learns to selectively upregulate, downregulate, and shut off feature maps based on conditioning information.

FiLM Parameters t-SNE Plot Similar to the previous chapter, we visualize FiLM parameter vectors (γ, β) for 3,000 random validation points with t-SNE in Fig 6.6. We analyze the deeper, 6-ResBlock version of our model, which has a similar validation accuracy as our 4-ResBlock model, to better examine how FiLM layers in different layers of a hierarchy behave. First and last layer FiLM (γ, β) are grouped by the low-level and high-level reasoning functions necessary to answer CLEVR questions, respectively. For example, FiLM parameters for `equal_color` and `query_color` are close for the first layer but apart for the last layer. The same is true for shape, size and material questions. Conversely, `equal_shape`, `equal_size`, and `equal_material` FiLM parameters are grouped in the last layer but split in the first layer — likewise for other high level groupings such as integer comparison and querying. These findings suggest that FiLM layers learn a sort of function-based modularity without an architectural prior. Simply with end-to-end training, modulation learns to handle not only different types of questions differently, but also different types of question subparts differently; the model works from low-level to high-level processes as is the proper approach. For models with fewer FiLM layers, such patterns also appear, but less clearly.

6.3.3 Ablation Studies

Using the validation set, we conduct an ablation study on our best model to understand how FiLM learns visual reasoning. We show results for test time ablations in Fig. 6.7, for architectural ablations in Tab. 6.2, and for varied model depths in Tab. 6.2. Without hyperparameter tuning, most architectural ablations and model depths outperform prior state-of-the-art on training from only image-question-answer triplets, supporting FiLM's overall robustness.

Effect of γ and β To test the effect of γ and β separately, we trained one model with a constant $\gamma = \mathbf{1}$ and another with $\beta = \mathbf{0}$. With these models, we find a 1.5% and $.5\%$ accuracy drop, respectively; FiLM can learn to condition the CNN for visual reasoning through either biasing or scaling alone, albeit not as well as conditioning both together. This result also suggests that γ is more important than β. To further compare the importance of γ and β, we run a series of test time ablations (Figure 6.7) on our best, fully-trained model. First, we replace β with the mean β across the training set. This ablation in effect removes all conditioning information from β parameters during test time, from a model trained to use both γ and β. Here, we find that accuracy only drops by 1.0%, while the same procedure on γ results in a 65.4% drop. This large difference suggests that, in practice, FiLM largely conditions through γ rather than β. Next, we analyze performance as we add increasingly more Gaussian noise to the best model's FiLM parameters at test time. Noise in gamma hurts performance significantly more, showing FiLM's higher sensitivity to changes in γ than in β and corroborating the relatively greater importance of γ.

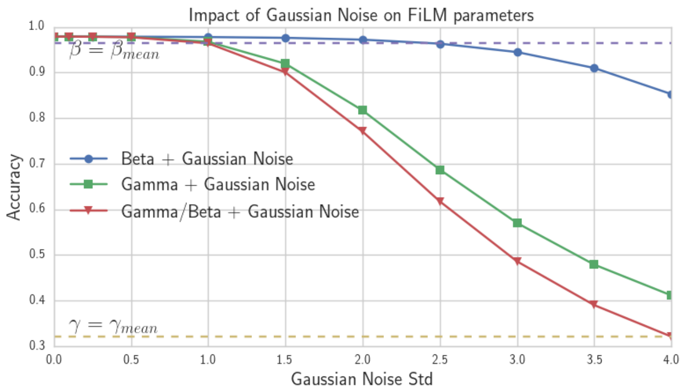


Figure 6.7: An analysis of how robust FiLM parameters are to noise at test time. The horizontal lines correspond to setting γ or β to their respective training set mean values.

Restricting γ To understand what aspect of γ is most effective, we train a model that limits γ to $(0, 1)$ using sigmoid, as many models which use feature-wise, multiplicative gating do. Likewise, we also limit γ to $(-1, 1)$ using $tanh$. Both restrictions hurt performance, roughly as much as removing conditioning from γ entirely by training with $\gamma = \mathbf{1}$. Thus, FiLM's ability to scale features by large magnitudes appears to contribute to its success. Limiting γ to $(0, \infty)$ with exp also hurts performance, validating the value of FiLM's capacity to negate and zero out feature maps.

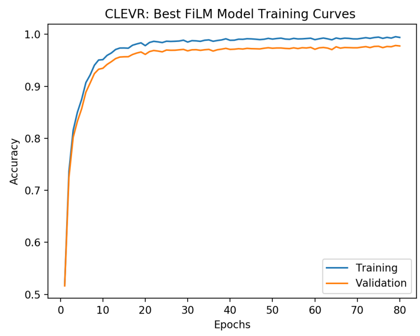
Batch-Normalization We perform an ablation study on the placement of modulaton layers to evaluate the relationship between normalization and FiLM that was implicit assumed with CBN. We moved the FiLM layers, separating it from the batch-normalization procedure, and we find no substantial performance drop. By demonstrating this conditioning mechanism is not closely connected to normalization, we open the doors to applications other settings in which normalization is less common, such as RNNs and reinforcement learning.

Repetitive Conditioning To understand the contribution of repetitive conditioning towards FiLM model success, we train the models with successively fewer modulating layers. Models with fewer modulation layers, even a single FiLM layer, do not deviate far from the best model's performance, revealing that the model can reason and answer diverse questions successfully by modulating features even just once. This observation highlights the capacity of even one FiLM layer.

Spatial Reasoning To examine how FiLM models approach spatial reasoning, we train a version of our best model architecture, from image features, with only 1×1 convolutions and without feeding coordinate feature maps indicating relative spatial position to the model. Due to the global max-pooling near the end of the model, this model cannot transfer information across spatial positions. Notably, this model still achieves a high 95.3% accuracy, indicating that the models are able to reason about space simply from the spatial information contained in a single location of fixed image features.

Residual Connection Removing the residual connection causes one of the larger accuracy drops. Since there is a global max-pooling operation near the end of the network, this finding suggests that the best model learns to primarily use features of locations that are repeatedly important throughout lower and higher levels of reasoning to make its final decision. The higher accuracies for models with FiLM modulating features inside residual connections rather than outside residual connections supports this hypothesis.

Model	Overall
Restricted γ or β	
FiLM with $\beta := \mathbf{0}$	96.9
FiLM with $\gamma := \mathbf{1}$	95.9
FiLM with $\gamma := \sigma(\gamma)$	95.9
FiLM with $\gamma := tanh(\gamma)$	96.3
FiLM with $\gamma := exp(\gamma)$	96.3
Moving FiLM within ResBlock	
FiLM after residual connection	96.6
FiLM after ResBlock ReLU-2	97.7
FiLM after ResBlock Conv-2	97.1
FiLM before ResBlock Conv-1	95.0
Removing FiLM from ResBlocks	
No FiLM in ResBlock 4	96.8
No FiLM in ResBlock 3-4	96.5
No FiLM in ResBlock 2-4	97.3
No FiLM in ResBlock 1-4	21.4
Miscellaneous	
1×1 conv only, with no coord. maps	95.3
No residual connection	94.0
No batch normalization	93.7
Replace image features with raw pixels	97.6
Best Architecture	97.4±.4



Model	Overall
1 ResBlock	93.5
2 ResBlocks	97.1
3 ResBlocks	96.7
4 ResBlocks	97.4
5 ResBlocks	97.4
6 ResBlocks	**97.7**
7 ResBlocks	97.4
8 ResBlocks	97.6
12 ResBlocks	96.9

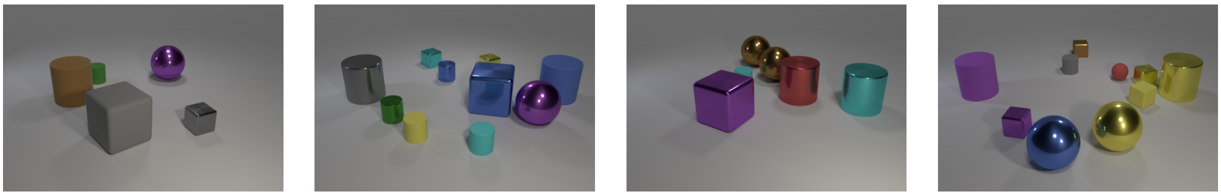
Table 6.2: (left) CLEVR val accuracy for ablations, trained with the best architecture with only specified changes. We report the standard deviation of the best model accuracy over 5 runs. (top-right) Best model training and validation curves. (bottom-right) CLEVR val accuracy by FiLM model depth.

Model Depth Table 6.2 shows model performance by the number of ResBlocks. FiLM is robust to varying depth but less so with only 1 ResBlock, backing the earlier theory that the FiLM network reasons throughout its pipeline.

Error Analysis As illustrated by Fig. 6.8, many model errors are due to partial occlusion (Kuhnle et al., 2018), which can be alleviate by scaling up the images to a higher resolution. The remaining errors mostly deal with counting mistakes, but they are off-by-one errors in the majority case, showing some structures in the prediction. Yet, the model sometimes makes unexpected reasoning mistakes as shown in Fig. 6.9, suggesting that our model still lacks some visual and/or linguistic understanding. For example, we find a case where our model correctly counts one gray object and two cyan objects but simultaneously answers that there are the same number of gray and cyan objects.

6.3.4 CLEVR-Humans: Human-Posed Questions

To assess how well visual reasoning models generalize to more realistic, complex, and free-form questions, the CLEVR-Humans dataset was introduced (Johnson et al., 2017b). This dataset contains human-posed questions on CLEVR images along with their corresponding answers. The number of samples is limited — 18K for training, 7K for validation, and 7K for testing. The questions were collected from Amazon Mechanical Turk workers prompted to ask questions that were likely *hard for a smart robot to answer*. As a result, CLEVR-Humans questions use more diverse vocabulary and complex concepts as shown in Fig. 6.10.



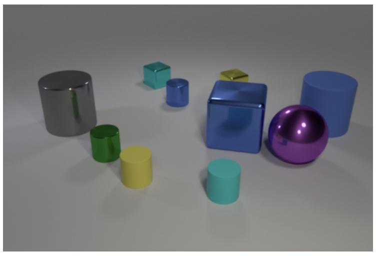
Q: *Is there a big brown object of the same shape as the green thing?* **A:** *Yes* (**P:** *No*)

Q: *What number of other things are the same material as the big gray cylinder?* **A:** *6* (**P:** *5*)

Q: *What shape is the big metal thing that is the same color as the small cylinder?* **A:** *Cylinder* (**P:** *Sphere*)

Q: *How many other things are the same material as the tiny sphere?* **A:** *3* (**P:** *2*)

Figure 6.8: Some image-question pairs where our model predicts incorrectly. Most errors we observe are due to partially occluded objects, as highlighted in the three first examples.



Question	Answer
How many gray things are there?	1
How many cyan things are there?	2
Are there as many gray things as cyan things?	**Yes**
Are there more gray things than cyan things?	No
Are there fewer gray things than cyan things?	Yes

Figure 6.9: A failure (bold) example where our model counts correctly but compares counts erroneously.

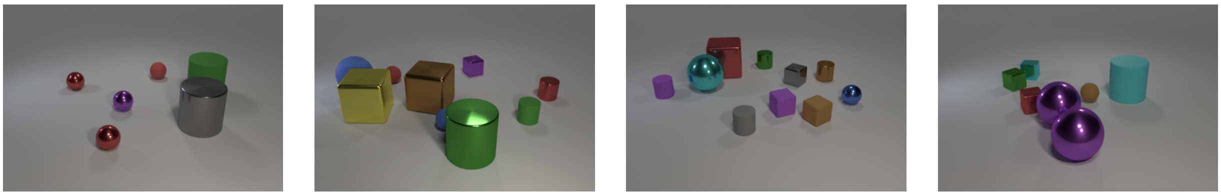
To test FiLM on CLEVR-Humans, we take our best CLEVR-trained FiLM model and fine-tune its FiLM-generating linguistic pipeline alone on CLEVR-Humans. Similar to prior work (Johnson et al., 2017b), we do not update the visual pipeline on CLEVR-Humans to mitigate overfitting to the small training set.

Our model achieves competitive accuracy in CLEVR-Humans as shown in Tab. 6.3. Before fine-tuning, FiLM outperforms prior methods by a smaller margin. After fine-tuning, FiLM reaches a considerably improved final accuracy. In particular, the *gain* in accuracy made by FiLM upon fine-tuning is more than 50% greater than those made by other models. Notably, FiLM surpasses the prior state-of-the-art method, Program Generator + Execution Engine (PG+EE), after fine-tuning by 9.3%. Prior work on PG+EEs explains that this neural module network method struggles on questions which cannot be well approximated with the model's module inventory (Johnson et al., 2017b). In contrast, FiLM seems to quickly update the feature-modulation to unseen scenario, which is in line with the hypernetwork literature for fast domain adaptation (Bertinetto et al., 2016; Ravi and Larochelle, 2017; Schmidhuber, 1987).

6.3.5 CLEVR Compositional Generalization Test

To test how well models learn compositional concepts that generalize, CLEVR-CoGenT was introduced (Johnson et al., 2017a). This dataset is synthesized in the same way as CLEVR but contains two conditions: in Condition A, all cubes are gray, blue, brown, or yellow and all cylinders are red, green, purple, or cyan; in Condition B, cubes and cylinders swap color palettes. Both conditions contain spheres of all colors. CLEVR-CoGenT thus indicates how a model answers CLEVR questions: by memorizing combinations of traits or by learning disentangled or general representations.

Results We train our best model architecture on Condition A and report accuracies on Conditions A and B, before and after fine-tuning on B, in Figure 6.11. Surprisingly, FiLM learns better com-



Q: *What object is the color of grass?* **A:** *Cylinder*

Q: *Which shape objects are partially obscured from view?* **A:** *Sphere*

Q: *What color is the matte object farthest to the right?* **A:** *Brown*

Q: *If all cubical objects were removed what shaped objects would there be the most of?* **A:** *Sphere* (**P:** *Rubber*)

Figure 6.10: Examples from CLEVR-Humans, which introduces new words (underlined) and concepts. After fine-tuning on CLEVR-Humans, a CLEVR-trained model can now reason about obstruction, superlatives but still struggles with hypothetical scenarios (rightmost). It also has learned human preference to primarily identify objects by shape (leftmost).

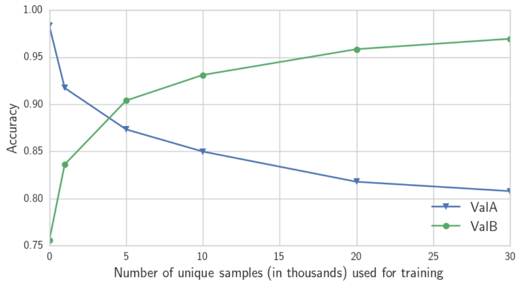
Model	Train CLEVR	Train CLEVR, fine-tune human
LSTM	27.5	36.5
CNN+LSTM	37.7	43.2
SAN	50.4	57.6
PG+EE (18K prog.)	**54.0**	**66.6**
FiLM	56.6	75.9

Table 6.3: CLEVR-Humans test accuracy, before (left) and after (right) fine-tuning on CLEVR-Humans data

positional generalization even than PG+EE, which explicitly models compositionality and is trained with program-level supervision that specifically includes filtering colors and filtering shapes. In addition, we show sample efficiency and forgetting curves in Figure 6.11 where FiLM achieves prior state-of-the-art accuracy with $1/3$ as much fine-tuning data. However, our FiLM model still suffers from catastrophic forgetting after fine-tuning.

Zero-Shot Generalization FiLM's accuracy on Condition A is much higher than on B, suggesting FiLM has memorized attribute combinations to an extent. For example, the model learns a bias that cubes are not cyan, as learning this training set bias helps minimize training loss. To overcome this bias, we develop a novel FiLM-based zero-shot generalization method. Inspired by word embedding manipulations, *e.g.* *"King"* - *"Man"* + *"Woman"* = *"Queen"* (Mikolov et al., 2013b), we test if linear manipulation extends to reasoning with FiLM. We compute (γ, β) for *"How many cyan cubes are there?"* via the linear combination of questions in the FiLM parameter space: *"How many cyan spheres are there?"* $+$ *"How many brown cubes are there?"* $-$ *"How many brown spheres are there?"*. With this (γ, β), our model can correctly count cyan cubes.

We evaluate this method on validation B, using a parser to automatically generate the right combination of questions. We test previously reported CLEVR-CoGenT FiLM models with this method and show results in Fig. 6.11. With this method, there is a 3.2% overall accuracy gain when training on A and testing for zero-shot generalization on B. Yet this method could only be applied to $1/3$ of questions in B. For these questions, model accuracy starts at 71.5% and jumps to 80.7%, demonstrating a true computationality, which is actually hidden while taking the full CLEVR-CoGenT dataset



Method	Train A A	Train A B	Fine-tune B A	Fine-tune B B
CNN+LSTM+SA	80.3	68.7	75.7	75.8
PG+EE (18K prog.)	96.6	73.7	76.1	92.7
CNN+GRU+FiLM	**98.3**	75.6	80.8	**96.9**
CNN+GRU+FiLM 0-Shot	**98.3**	**78.8**	**81.1**	**96.9**

Figure 6.11: CoGenT results. FiLM ValB accuracy reported on ValB without the 30K fine-tuning samples (Figure). Accuracy before and after fine-tuning on 30K of ValB (Table).

as in Fig. 6.11. As implemented, this method has many limitations, but it does highlight the potential of modulation. Thus, FiLM may benefit methods developed for word embeddings, representation learning, and zero-shot learning.

6.4 Discussion

6.4.1 How CLEVR is the FiLM model? Subsequent Meta-analysis

FiLM has also been used in several subsequent works, and it has spread to large variety of tasks including acoustic reasoning (Abdelnour et al., 2018), GAN conditioning (Almahairi et al., 2018; Brock et al., 2019; Miyato and Koyama, 2018), image segmentation to integrate visual priors (Yang et al., 2018) or linguistic cues (Rupprecht et al., 2018), speech recognition (Kim et al., 2017b), instruction following (Bahdanau et al., 2019a), zero and few-shot learning (Jiang et al., 2019; Oreshkin et al., 2018; Prol et al., 2018) or reinforcement learning (Vinyals et al., 2019).

Following this success, several meta-analyses have emerged to assess modulation learning and understanding potential (Bahdanau et al., 2019b; Dumoulin et al., 2018; Kuhnle and Copestake, 2018; Kuhnle et al., 2018). For instance, Kuhnle et al. (2018) run a broad range of experiments on a 2D CLEVR-like layout with more complex linguistic structures, exploring relational visual reasoning (darker, lighter, smaller, bigger), superlative, and negation. They observe that FiLM requires a form of iterative learning procedure to perform well on the most sophisticated reasoning tasks. For instance, FiLM struggles to perform relational reasoning if it is neither jointly trains (or pretrained) on existential questions or simple logical questions first. Besides, the authors observe that FiLM was very sensitive to mismatches in the data distribution, and suggest some intrinsic over-fitting issues. In a second paper, Kuhnle and Copestake (2018) study how FiLM models can approximate large quantity, and observe that the FiLM model error distributions seem to follow a realistic cognitive pattern, suggesting that an approximate number system is learned. Bahdanau et al. (2019b) examine the generalization capabilities of different CLEVR models including FiLM, NMN, Relational Networks, and MAC under the following assumption: *A good model should be able to reason about all possible object combinations despite being trained on a very small subset of them*. Similar to (Kuhnle et al., 2018), they observe that modulation is insufficient to generalize well while dealing with relational reasoning, or when using unbalanced dataset. NMN were far more efficient for this kind of generalization, and a few experiments suggest that MAC outperform FiLM by better attending to the linguistic input. However, FiLM is the neural module that was successfully applied to natural data, and other machine learning fields. One of the takeaways of these meta-analyses is that FiLM, and the modulation is not a silver-bullet for visual understanding as the CLEVR accuracies may first suggest.

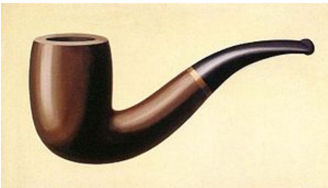
It is worth to study modulation as a mechanism, but it is misleading to reduce a model to its attention, modulation or any other conditioning mechanisms as sometimes observed in the literature.

6.4.2 Conclusion

We show that a model can achieve strong visual reasoning using general-purpose Feature-wise Linear Modulation layers. By efficiently manipulating a neural network's intermediate features in a selective and meaningful manner using FiLM layers, a RNN can effectively use language to modulate a CNN to carry out diverse and multi-step reasoning tasks over an image. Our ablation study suggests that modulation is resilient to architectural modifications, test time ablations, and even restrictions on FiLM layers themselves. Notably, we provide evidence that FiLM's success is not closely connected with normalization, as previously introduced in Chapter 5. As opposed to ModeRn models, FiLM based models do not need to embed a full ResNet at evaluation time, making it a valuable and versatile module. Besides, we show that modulation and CNN layers can be learned jointly from scratch without the need for pretrained networks. Our findings also suggest that modulation models can generalize better, more sample efficiently, and even zero-shot to challenging data that classic neural approaches.

Chapter 7

Multi-Hop FiLM



$+ \epsilon$

«This is not a Pipe »

The Treachery of Image Classification

Contents

Before starting our research on multimodal architecture, we observe that previous state-of-the-art multimodal architectures failed at improving fundamental GuessWhat?! baselines, highlighting their intrinsic limitations. Hence, we started exploring modulation blocks for multimodal learning in Chapters 5 and 6. In this chapter, we finally aim to assess our methods on the GuessWhat?! task. However, we noticed that CBN required too many computational resources to be effective, and we only tested FiLM on artificial tasks so far. Besides, we also observed that FiLM required a huge RNN to process the language jointly, and generate modulation parameters in CLEVR. We thus here explore how to correctly balance language processing capacity to deal with GuessWhat?! natural language.

To do so, we here propose to generate the parameters of FiLM layers going up the hierarchy of a convolutional network in a multi-hop fashion rather than all at once. By alternating between attending to the language input and generating FiLM layer parameters, this approach would better scale-up to settings with long sequences of inputs, e.g., dialogues. Such Multi-hop FiLM (Multi-hop FiLM) architecture achieves state-of-the-art for the short input sequence task ReferIt— on-par with single-hop FiLM generation — while significantly outperforming prior state-of-the-art and single-hop FiLM generation on the GuessWhat?! visual dialogue task.

7.1 Introduction

In the previous chapters, we introduced FiLM layers as a promising approach for vision-and-language tasks. These layers apply a per-channel scaling and shifting to a convolutional network's visual features, conditioned on an external input such as language, e.g., captions, questions, or full dialogues. Such feature-wise affine transformations allow models to dynamically highlight the key visual features for the task at hand. The parameters of FiLM layers which scale and shift features or feature maps are determined by a separate network, the so-called *FiLM generator*, which predicts these parameters using the external conditioning input.

However, the best way to design the FiLM generator is still an open question. For visual question-answering and visual reasoning, prior work uses single-hop FiLM generators that predict all FiLM parameters at once as in Chapter 5 and 6. That is, a RNN sequentially processes input language tokens and then outputs all FiLM parameters via a MLP. In this chapter, we argue that using a *Multi-hop FiLM Generator* is better suited for tasks involving longer input sequences and multi-step reasoning such as dialogue. Even for shorter input sequence tasks, single-hop FiLM generators can require a large RNN to achieve strong performance; on the CLEVR visual reasoning task (Johnson et al., 2017a) which only involves a small vocabulary (>30 words) and templated questions, the FiLM generator in Chapter 6 uses an RNN with 4096 hidden units that comprises almost 90% of the model's parameters. Models with Multi-hop FiLM Generators may thus be easier to scale to more difficult tasks involving human-generated language involving larger vocabularies and more ambiguity.

As an intuitive example, consider the dialogue in Fig. 7.1 through which one speaker localizes the second girl in the image, the one who does not "have a blue frisbee." For this task, a single-hop model must determine upfront what steps of reasoning to carry out over the image and in what order; thus, it might decide in a single shot to highlight feature maps throughout the visual network detecting either non-blue colors or girls. In contrast, a multi-hop model may first determine the most immediate step of reasoning necessary (i.e., locate the girls), highlight the relevant visual features, and then determine the next immediate step of reasoning necessary (i.e., locate the blue frisbee), and so on. While it may be appropriate to reason in either way, the latter approach may scale better to longer language inputs and/or or to ambiguous images where the full sequence of reasoning steps is hard to determine upfront, which can even be further enhanced by having intermediate feedback while processing the image.

In this chapter, we therefore explore several approaches to generating FiLM parameters in multiple hops. These approaches introduce an intermediate context embedding that controls the language and visual processing, and they alternate between updating the context embedding via an attention mechanism over the language sequence (and optionally by incorporating image activations) and predicting the FiLM parameters. We evaluate Multi-hop FiLM generation on ReferIt (Kazemzadeh et al., 2014) and GuessWhat?!, two vision-and-language tasks illustrated in Fig. 7.1. We show that Multi-hop FiLM models significantly outperform their single-hop counterparts and prior state-of-the-art for the longer input sequence, dialogue-based GuessWhat?! task while matching the state-of-the-art performance of other models on ReferIt. Our best GuessWhat?! model only updates the context embedding using the language input, while for ReferIt, incorporating visual feedback to update the context embedding improves performance.

ReferIt	*GuessWhat?!*	
- The girl with a sweater	Is it a person?	Yes
- The fourth person	Is it a girl?	Yes
- The girl holding a white frisbee	Does she have a blue frisbee?	No

Figure 7.1: The ReferIt task identifies a selected object (in the bounding box) using a single expression, while in GuessWhat?!, a speaker localizes the object with a series of yes or no questions.

In summary, this chapter makes the following contributions:

- We introduce the Multi-hop FiLM architecture and demonstrate that our approach matches or significantly improves state-of-the-art on the GuessWhat?! Oracle task, GuessWhat?! Guesser task, and ReferIt Guesser task.
- We show Multi-hop FiLM models outperform their single-hop counterparts on vision-and-language tasks involving complex visual reasoning.
- We find that updating the context embedding of Multi-hop FiLM Generator based on visual feedback may be helpful in some cases, such as for tasks which do not include object category labels like ReferIt.

7.2 Multi-hop FiLM

In this section, we introduce the Multi-hop FiLM architecture (shown in Fig. 7.2) to predict the parameters of FiLM layers in an iterative fashion, to better scale to longer input sequences such as in dialogue. Another motivation was to better disantangle the linguistic reasoning from the visual one by iteratively attending to both pipelines.

The Multi-hop FiLM architecture composes with a context vector $\boldsymbol{c}$ that acts as a controller for the linguistic and visual pipelines. This context vector is used to either attend over a sequence of language embeddings $\{\boldsymbol{e}_{l,t}\}_{t=1}^{T}$ and to modulate the visual feature maps $\boldsymbol{F}$. Given a pipeline of $\mathcal{K}$ modulated ResBlock, we therefore compute $\mathcal{K}+1$ distinct context vectors to perform κ reasoning hops. We first initialize the context vector $\boldsymbol{c}_0$ with the final state of a bidirectional RNN $\boldsymbol{e}_{l,T}$ and repeat the following procedure for each of the FiLM layers in sequence (from lowest to highest convolutional layer): first, the context vector is updated by performing attention over RNN states (extracting relevant language information), and second, the context is used to predict a layer's FiLM parameters (dynamically modulating the visual information). Note that the original FiLM architecture is equivalent to re-use the initial context vector $\boldsymbol{c}_0$ at each modulation step. More formally, the context vector is computed as follows:

$$\begin{cases} \boldsymbol{c}^0 = \boldsymbol{e}_{l,T} \\ \boldsymbol{c}^\kappa = LN(\boldsymbol{c}^{\kappa-1} + \sum_t \alpha_t^\kappa \boldsymbol{e}_{l,t}) \end{cases} \tag{7.1}$$

where:

$$\alpha_t^k(\boldsymbol{c}^{\kappa-1}, \boldsymbol{e}_{l,t}) = \frac{\exp(\xi_t^\kappa)}{\sum_t \exp(\xi_t^\kappa)} \quad ; \quad \xi_t^\kappa(\boldsymbol{c}^{\kappa-1}, \boldsymbol{e}_{l,t}) = MLP_{Attn}(g'(\boldsymbol{c}^\kappa, \boldsymbol{e}_{l,t})), \tag{7.2}$$

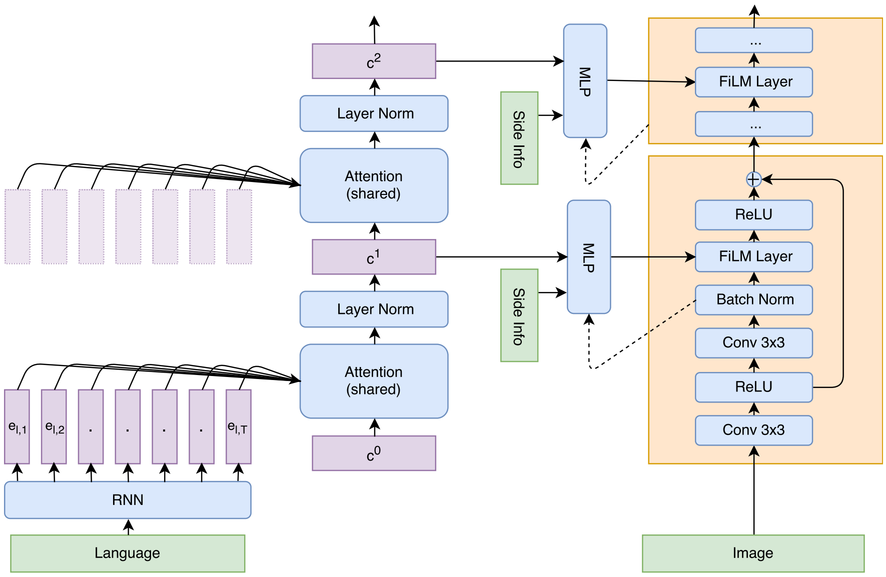


Figure 7.2: The Multi-hop FiLM architecture, illustrating inputs (green), layers (blue), and activations (purple). In contrast, Single-hop FiLM models predict FiLM parameters directly from $\boldsymbol{e}_{l,T}$.

where the dependence of ξ_t^κ and α_t^κ on $(\boldsymbol{c}^{\kappa-1}, \boldsymbol{e}_{l,t})$ are omitted to simplify notation. MLP_{Attn} is a network (shared across layers) which aids in producing attention weights. g' can be any fusion mechanism that facilitates selecting the relevant context to attend to; here we use a simple dot-product following (Luong et al., 2015), so $g'(\boldsymbol{c}^\kappa, \boldsymbol{s}_t) = \boldsymbol{c}^\kappa \odot \boldsymbol{s}_t$. Finally, FiLM is carried out using a layer-dependent neural network MLP^κ_{FiLM}:

$$[\,\boldsymbol{\gamma}^\kappa\ ;\ \boldsymbol{\beta}^\kappa\,] = MLP^\kappa_{FiLM}(\boldsymbol{c}^\kappa) \qquad ; \qquad \hat{\boldsymbol{F}}^\kappa_{.,.,c} = \gamma^\kappa_c \boldsymbol{F}^\kappa_{.,.,c} + \beta^\kappa_c. \tag{7.3}$$

As a regularization, we append a normalization-layer (Ba et al., 2016) on top of the context vector after each attention step. A straightforward improvement is to use a transformer block to update $\boldsymbol{e}_{l,t}$ at each reasoning hop (Vaswani et al., 2017), but leave it to future work.

External information Some tasks provide additional information which may be used to further improve the visual modulation. For instance, GuessWhat?! provides spatial features of the ground truth object to models which must answer questions about that object. Our model incorporates such features by concatenating them to the context vector before generating FiLM parameters.

Visual feedback Inspired by the co-attention mechanism (Jiasen et al., 2016; Zhuang et al., 2018), we also explore incorporating visual feedback into the Multi-hop FiLM architecture. To do so, we first extract the image or crop features $\boldsymbol{F}^k$ (immediately before modulation) and apply a global mean-pooling over spatial dimensions. We then concatenate this visual state into the context vector $\boldsymbol{c}^k$ before generating the next set of FiLM parameters.

7.3 Experiments

In this section, we describe our overall Multi-hop FiLM architecture before evaluating on the ReferIt and GuessWhat?! tasks.

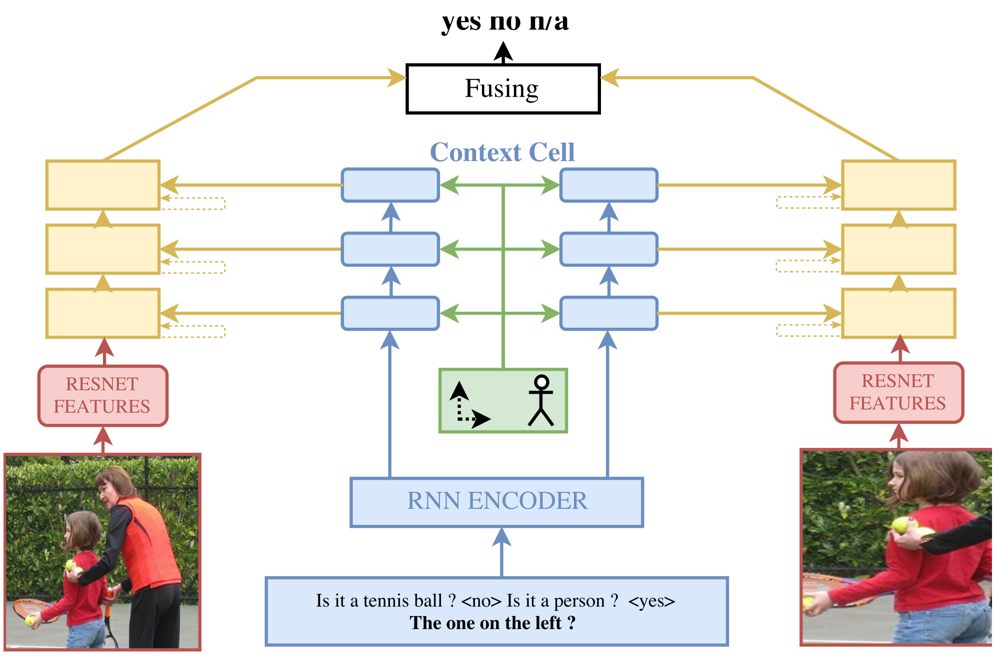


Figure 7.3: Overall model, consisting of a visual pipeline (red and yellow) and linguistic pipeline (blue) and incorporating additional contextual information (green).

7.3.1 Task Descriptions

Dataset As mentioned in Sec. 3.2, ReferIt (Kazemzadeh et al., 2014; Yu et al., 2016a) is a cooperative two-player game. The first player (the Oracle) selects an object in a rich visual scene, for which she must generate an expression that refers to it (e.g., "the person eating ice cream"). Based on this expression, the second player (the Guesser) must then select an object within the image. There are four ReferIt datasets: RefClef, RefCOCO, RefCOCO+ and RefCOCOg where RefClef relies on the ImageClef dataset Müller et al. (2012), while the three other datasets are based the MScoco dataset Lin et al. (2014). Each dataset enforces specific rules while collecting the data, but as a rule of thumb, datasets have more and more long and complex sentences. For instance, RefCOCOg also contains 8.4 words while RefCOCO only have 3.5 words in average. In our experiments, we also use the GuessWhat?! dataset which is fully detailed in Chapter 4

Notation Both games consist of quadruplets $(\mathcal{I}, \mathcal{D}, O, o^*)$, where $\mathcal{I} \in \mathbb{R}^{3\times W\times H}$ is an RGB image of width W and height H containing a set of K objects $O = \{o_1, \ldots, o_K\}$ and $\mathcal{D}$ is a language input (i.e., a series of words) describing an target object $o^* \in O$ present in the image $\mathcal{I}$. Each object o^k is assigned an object category $c_k \in \{1, \ldots, C\}$, a pixel-wise segmentation mask $\mathcal{S}_k \in \{0,1\}^{W\times H}$, an RGB crop within $\mathcal{I}$ and spatial information $\boldsymbol{x}_{spatial} = [x_{min}, y_{min}, x_{max}, y_{max}, x_{center}, y_{center}, w_{box}, h_{box}]$

The Oracle task Given the image $\mathcal{I}$, an object o, a question $\boldsymbol{q}$, and a sequence of j previous question-answer pairs $(\boldsymbol{q}, a)_{j<}$ where $a \in \{\text{Yes}, \text{No}, \text{N/A}\}$, the Oracle's task is to produce an answer a that correctly answers the question $\boldsymbol{q}$.

The Guesser task Given the image $\mathcal{I}$, the list of objects $\{o_1, \ldots, o_K\}$, the target object $o^* \in O$ and the dialogue $\mathcal{D}$, the guesser needs to output a probability σ_k that each object o_k is the target object o^*. Following (Hu et al., 2016b), the Guesser is evaluated by selecting the object with the highest probability of being correct. Note that even if the individual probabilities σ_ϕ are between 0 and 1, their sum can be greater than 1. More formally, the Guesser loss and error are computed as follows:

$$L_{Guesser} = \frac{-1}{N_{games}} \sum_{n}^{N_{games}} \frac{1}{K^n} \sum_{k}^{K} \log(p(o^*|\mathcal{I}^n, o_k^n, \mathcal{D}^n)) \tag{7.4}$$

$$E_{Guesser} = \frac{-1}{N_{games}} \sum_{n}^{N_{games}} \mathbb{1}(o^* \neq o_{\arg\max_\phi \sigma_\phi^n}) \tag{7.5}$$

where $\mathbb{1}$ is the indicator function and N the number of samples in a batch of samples.

7.3.2 Model

We use similar models for both ReferIt and GuessWhat?! and provide its architectural details in this subsection, and we depict the network in Fig. 7.3.

Object Embedding The object category is fed into a dense look-up table $\boldsymbol{e}_{cat}$, and the spatial information is scaled to [-1;1] before being up-sampled via non-linear projection to $\boldsymbol{e}_{spat}$. We do not use the object category in ReferIt models.

Visual Pipeline We first resized the image and object crop to 448×448 before extracting $14 \times 14 \times 1024$ dimensional features from a ResNet-152 (He et al., 2016) (block3) pretrained on ImageNet (Russakovsky et al., 2015). As in Chapter 6, we feed these features to a 3×3 convolution layer with BN (Ioffe and Szegedy, 2015) and ReLU. We then stack four modulated residual blocks (shown in Fig 7.2), each producing a set of feature maps $\boldsymbol{F}^\kappa$ via (in order) a 1×1 convolutional layer (128 units), ReLU activations, a 3×3 convolutional layer (128 units), and an untrainable Batch Normalization layer. The residual block then modulates $\boldsymbol{F}^\kappa$ with a FiLM layer to get $\hat{\boldsymbol{F}}^\kappa$, before again applying ReLU activations. Lastly, a residual connection sums the activations of both ReLU outputs. After the last residual block, we use a 1×1 convolution layer (512 units) with BN and ReLU followed by MLB attention (Kim et al., 2017a) (256 units and 1 glimpse) to obtain the final embedding $\boldsymbol{e}_v$. Note our model uses two independent visual pipeline modules: one to extract modulated image features $\boldsymbol{e}_v^{img}$, one to extract modulated crop features $\boldsymbol{e}_v^{crop}$.

To incorporate spatial information, we concatenate two coordinate feature maps indicating relative x and y spatial position (scaled to $[-1, 1]$) with the image features before each ResBlock. In addition, the pixel-wise segmentations $S \in \{0, 1\}^{M \times N}$ are rescaled to 14×14 floating point masks before being concatenated to the feature maps.

Linguistic Pipeline We compute the language embedding by using a word-embedding look-up (200 dimensions) with dropout followed by a Bi-GRU (512×2units) with LN (Ba et al., 2016). As described in Section 7.2, we initialize the context vector with the last RNN state $\boldsymbol{c}^0 = \boldsymbol{s}_T$. We then attend to the other Bi-GRU states via an attention mechanism with a linear projection and ReLU activations and regularize the new context vector with LN.

FiLM Parameter Generation We concatenate spatial information $\boldsymbol{e}_{spat}$ and object category information $\boldsymbol{e}_{cat}$ (GuessWhat?! only) to the context vector. In some specified experiments, we concatenate a fourth embedding consisting of intermediate visual features $\boldsymbol{F}^\kappa$ after mean-pooling. Finally, we use a linear projection to map the embedding to FiLM parameters.

Final Layers We generate our final embedding by concatenating the output of the visual pipelines $\boldsymbol{e}_{final} = [\boldsymbol{e}_v^{img}\,; \boldsymbol{e}_v^{crop}]$ before applying a linear projection (512 units) with ReLU and a softmax layer.

Referit Split by Report on	Valid	RefCOCO (unc) TestA	TestB	Valid	RefCOCO+ (unc) TestA	TestB	RefCOCOg (google) Val	(umd) Test
MMI (Nagaraja et al., 2016)	-	71.7%	71.1%	-	58.4%	51.2%	59.3%	-
visDif + MMI (Yu et al., 2016a)	-	74.6%	76.6%	-	59.2%	55.6%	64.0%	-
NEG Bag (Nagaraja et al., 2016)	-	75.6%	78.0%	-	-	-	68.4%	68.4%
Joint-SLR (Yu et al., 2016b)	78.9%	78.0%	80.7%	61.9%	64.0%	59.2%	-	71.9%
PLAN (Zhuang et al., 2018)	81.7%	80.8%	81.3%	64.2%	66.3%	61.5%	69.5	-
MAttNet (Yu et al., 2018b)	85.7%	85.3%	84.6%	71.0%	75.1%	66.2%	**73.1%**	78.1%
CM-Att-Erase (Liu et al., 2019b)	**87.5%**	**88.1%**	**86.3%**	**73.7%**	77.6	**68.9%**	-	**80.4%**
Baseline NN+MLB	77.6%	79.6%	77.2%	60.8%	59.7%	66.2%	63.1%	-
Single-hop FiLM	83.4%	85.8%	80.9%	72.1%	77.3%	63.9%	67.8%	-
Multi-hop FiLM	83.5%	86.5%	81.3%	73.4%	77.7%	64.5%	69.8%	-
Multi-hop FiLM (+img)	84.9%	**87.4%**	83.1%	**73.8%**	**78.7%**	65.8%	71.5%	-

Table 7.1: ReferIt Guesser Error.

Training Process We train our model end-to-end with Adam (Kingma and Ba, 2015) (learning rate $3e^{-4}$), dropout (0.5), weight decay ($5e^{-6}$) for convolutional network layers, and a batch size of 64. We report results after early stopping on the validation set with a maximum of 15 epochs.

7.3.3 Baselines

In our experiments, we re-implement several baseline models to benchmark the performance of our models. The standard **Baseline NN** simply concatenates the image and object crop features after mean pooling, the linguistic embedding, and the spatial embedding and the category embedding (GuessWhat?! only), passing those features to the same final layers described in our proposed model. We refer to a model which uses the MLB attention mechanism to pool the visual features as **Baseline NN+MLB**. We also implement a **Single-hop FiLM** mechanism which is equivalent to setting all context vectors equal to the last state of the Bi-GRU $\boldsymbol{e}_{l,T}$. Finally, we experiment with injecting intermediate visual features into the FiLM Generator input, and we refer to the model as **Multi-hop FiLM (+img)**. We also refer the reader to Sec. 3.2 for more details on the other state-of-the art benchmarks.

7.3.4 Results

ReferIt Guesser We report the best test error of the outlined methods on the ReferIt Guesser task in Tab. 7.1 and Tab. 7.5. Note that RefCOCO and RefCOCO+ split test sets into TestA and TestB, only including expression referring towards people and objects, respectively. Our initial baseline achieves 77.6%, 60.8%, 63.1%, 73.4% on the RefCOCO, RefCOCO+, RefCOCOg, RefClef datasets, respectively, performing comparably to state-of-the-art models. We observe a significant improvements using a FiLM-based architecture, jumping to 84.9%, 87.4%, 73.8%, 71.5%, respectively, and outperforming most prior methods and achieving comparable performance with the concurrent MAttNet (Yu et al., 2018b) model. Interestingly, MAttNet and Multi-hop FiLM are built in two different manners; while the former has three specialized reasoning blocks, our model uses a generic feature modulation approach. These architectural differences surface when examining test splits: MAttNet achieves excellent results on referring expression towards objects while Multi-hop FiLM performs better on referring expressions towards people, suggesting that an assemble models would greatly perform.

Oracle Models	Quest.	Dial.	Object	Image	Crop	Test Error
Dominant class ("no")	✗	✗	✗	✗	✗	50.9%
Question only Chapter 4	✓	✗	✗	✗	✗	41.2%
Image only Chapter 4	✗	✗	✗	✓	✗	46.7%
Crop only Chapter 4	✗	✗	✗	✗	✓	43.0%
No-Vision (Quest.) Chapter 4	✓	✗	✓	✗	✗	21.5%
No-Vision (Dial.)	✗	✓	✓	✗	✗	20.6%
Baseline NN (Quest.)	✓	✗	✓	✓	✓	23.3%
Baseline NN (Dial.)	✗	✓	✓	✓	✓	22.4%
Baseline NN + MLB (Quest.)	✓	✗	✓	✓	✓	21.8%
Baseline NN + MLB (Dial.)	✗	✓	✓	✓	✓	21.1%
MODERN Chapter 5	✓	✗	✓	✗	✓	19.5%
Single-hop FiLM (Quest.)	✓	✗	✓	✓	✓	17.8%
Single-hop FiLM (Dial.)	✗	✓	✓	✓	✓	17.6%
Multi-hop FiLM	✗	✓	✓	✓	✓	**16.9%**
Multi-hop FiLM (+img)	✗	✓	✓	✓	✓	17.1%

Table 7.2: GuessWhat?! Oracle Error by Model and Input Type.

Guesser Error	Test
Random	82.9%
LSTM	38.7%
LSTM + Img	39.5%
PLAN (Zhuang et al., 2018)	36.6%
Base NN + MLB (crop)	38.3%
Single-hop FiLM	35.6%
Multi-hop FiLM	**30.5%**

Guesser Error	Crop	Image	Crop+Img
Baseline NN	38.3%	40.0%	45.1%
Single-hop FiLM	35.3%	35.7%	35.6%
Multi-hop FiLM	32.3%	35.0%	**30.5%**

Table 7.3: GuessWhat?! Guesser Error on the successful dataset.

GuessWhat?! Oracle We report the best test error of several variants of GuessWhat?! Oracle models in Tab. 7.2. First, we looked at naive baselines which discard visual or language cues by predicting the Oracle's target answer using only the image (46.7% error) or the question (41.1% error). As first reported Chapter 4, we observe that the baseline methods perform worse when integrating the image and crop inputs (21.1%) rather than solely using the object category and spatial location (20.6%). On the other hand, concatenating previous question-answer pairs to answer the current question is beneficial in our experiments. Finally, using Single-hop FiLM reduces the error to 17.6% and Multi-hop FiLM further to 16.9%, outperforming the previous best model by 2.4%.

GuessWhat?! Guesser We provide the best test error of the outlined methods on the GuessWhat?! Guesser task in Tab. 7.3. As a baseline, we find that random object selection achieves an error rate of 82.9%. Our initial model baseline performs significantly worse (38.3%) than concurrent models (36.6%), highlighting that successfully jointly integrating crop and image features is far from trivial. However, Single-hop FiLM manages to lower the error to 35.6%. Finally, Multi-hop FiLM architecture outperforms other models with a final error of 30.5%.

7.4 Discussion

Single-hop FiLM *vs.* Multi-hop FiLM In the GuessWhat?! task, Multi-hop FiLM outperforms Single-hop FiLM by 6.1% on the Guesser task but only 0.7% on the Oracle task. We think that the

Oracle Model	Test Error
Baseline NN+MLB	26.7%
Single-hop FiLM	19.5%
Multi-hop FiLM	18.9%
Multi-hop FiLM (+img)	**18.4%**

Guesser Model	Crop	Image	Crop/Img
PLAN (Zhuang et al., 2018)	-	-	40.3%
Multi-hop FiLM	35.3%	39.8%	33.9%
Multi-hop FiLM (+img)	34.3%	40.1%	**33.2%**

Table 7.4: (top) GuessWhat?! Oracle (top) and Guesser (down) test error without object category label on the successful dataset.

small performance gain for the Oracle task is due to the nature of the task; to answer the current question, it is often not necessary to look at previous question-answer pairs, and in most cases this task does not require a long chain of reasoning. On the other hand, the Guesser task needs to gather information across the whole dialogue in order to correctly retrieve the object, and it is therefore more likely to benefit from multi-hop reasoning. The same trend can be observed for ReferIt. Single-hop FiLM and Multi-hop FiLM perform similarly on RefClef and RefCOCO, while we observe 1.3% and 2% gains on RefCOCO+ and RefCOCOg, respectively. This pattern of performance is intuitive, as the former datasets consist of shorter referring expressions (3.5 average words) than the latter (8.4 average words in RefCOCOg), and the latter datasets also consist of richer, more complex referring expressions due e.g. to taboo words (RefCOCO+). In short, our experiments demonstrate that Multi-hop FiLM is better able to reason over complex linguistic sequences.

Reasoning mechanism We conduct several experiments to better understand our method. First, we assess whether Multi-hop FiLM performs better because of increased network capacity. We remove the attention mechanism over the linguistic sequence and update the context vector via a shared MLP. We observe that this change significantly hurts performance across all tasks, e.g., increasing the Multi-hop FiLM error of the Guesser from 30.5 to 37.3%. Second, we investigate how the model attends to GuessWhat?! dialogues for the Oracle and Guesser tasks, providing more insight into how to the model reasons over the language input. We first look at the top activation in the (crop) attention layers to observe where the most prominent information is. Note that similar trends are observed for the image pipeline. As one may expect, the Oracle is focused on a specific word in the last question 99.5% of the time, one which is crucial to answer the question at hand. However, this ratio drops to 65% in the Guesser task, suggesting the model is reasoning in a different way. If we then extract the top 3 activations per layer, the attention points to $<yes>$ or $<no>$ tokens (respectively) at least once, 50% of the time for the Oracle and Guesser, showing that the attention is able to correctly split the dialogue into question-answer pairs. Finally, we plot the attention masks for each FiLM layer to have a better intuition of this reasoning process in Fig. 7.4.

Crop *vs.* Image. We also evaluate the impact of using the image and/or crop on the final error for the Guesser task 7.3. Using the image alone (while still including object category and spatial information) performs worse than using the crop. However, using image and crop together unarguably gives the lowest errors, though prior work has not always used the crop due to architecture-specific GPU limitations as in Chapter 5.

Visual feedback We explore whether adding visual feedback to the context embedding improves performance. While it has little effect on the GuessWhat?! Oracle and Guesser tasks, it improves the accuracy on ReferIt by 1-2%. Note that ReferIt does not include class labels of the selected object, so the visual feedback might act as a surrogate for this information. To further investigate this

Figure 7.4: Guesser (left) and Oracle (right) attention visualizations for the visual pipeline which processes the object crop.

hypothesis, we remove the object category from the GuessWhat?! task and report results in Tab. 7.4. In this setup, we indeed observe a relative improvement 0.4% on the Oracle task, further confirming this hypothesis.

7.5 Related Work

The ReferIt game (Kazemzadeh et al., 2014) has been a testbed for various vision-and-language tasks over the past years, including object retrieval (Luo and Shakhnarovich, 2017; Nagaraja et al., 2016; Yu et al., 2016a,b, 2018b; Zhuang et al., 2018), semantic image segmentation (Hu et al., 2016a; Rohrbach et al., 2016), and generating referring descriptions (Luo and Shakhnarovich, 2017; Yu et al., 2016a,b). To tackle object retrieval, (Nagaraja et al., 2016; Yu et al., 2016a, 2018b) extract additional visual features such as relative object locations and (Luo and Shakhnarovich, 2017; Yu et al., 2016b) use reinforcement learning to iteratively train the object retrieval and description generation models. Closer to our work, (Hu et al., 2016b; Zhuang et al., 2018) use the full image and the object crop to locate the correct object. While some previous work relies on task-specific modules (Yu et al., 2016a, 2018b), our approach is general and can be easily extended to other vision-and-language tasks.

The GuessWhat?! game can be seen as a dialogue version of the ReferIt game, one which additionally draws on visual question answering ability. As further studied in the next chapter, Lee et al. (2018b); Zhu et al. (2017b) make headway on the dialogue generation task via reinforcement learning. However, these approaches are bottlenecked by the accuracy of Oracle and Guesser models, despite existing modeling advances in the previous chapters and concurrent work (Zhuang et al., 2018); accurate Oracle and Guesser models are crucial for providing a meaningful learning signal for dialogue generation models, so the Multi-hop FiLM architecture may facilitate high quality dialogue generation as well.

There are other notable models that decompose reasoning into different modules. For instance, Neural Turing Machine (NTM) (Graves et al., 2014, 2016) divide a model into a controller composed of read&write neural units. Memory networks use an attention mechanism to answer a query by reasoning over a linguistic knowledge base (Sukhbaatar et al., 2015; Weston et al., 2014) or image features (Xiong et al., 2016). A memory network updates a query vector by performing several attention hops over the memory before outputting a final answer from this query vector. Although Multi-hop FiLM computes a similar context vector, this intermediate embedding is used to predict FiLM parameters rather than the final answer. Thus, Multi-hop FiLM includes a second reasoning step over the image. The Multi-hop FiLM reasoning module shares several similarities with the attention encoder from Transformer networks (Devlin et al., 2018; Vaswani et al., 2017), but transformer aims to learn an intermediate language representation that is latter decoded while Multi-hop FiLM discards the intermediate linguitic representations once the modulation parameters are predicted. Yet, it would be worth investigating to improve Multi-hop FiLM in light of the recent successes of the transformers architecture.

Closer to our work, Hudson and Manning (2018) designed networks composed of Memory,Attention, and Composition (MAC) cells to perform visual reasoning. Similar to NTM, each MAC cell is composed of a control unit that attends over the language input, a read unit that attends over the image and a write unit that fuses both pipelines. Though conceptually similar to Multi-hop FiLM models, compositional attention networks differ structurally, for instance they use a dynamic neural architecture, and the core neural blocks are based on spatial attention rather than modulation layers. While MAC has been unarguably superior to FiLM-ed networks on complex but artificial tasks (Bahdanau et al., 2019b; Suhr et al., 2017), it faces strong overfitting issues while dealing with natural images (Shrestha et al., 2019; Suhr et al., 2017). An interesting line of research is to compose with both approaches to bring the best of the two worlds in a single model.

7.6 Conclusion

In this chapter, we introduce a new way to exploit FiLM layers for vision-and-language tasks. Our approach generates the parameters of modulation layers going up the visual pipeline by attending to the language input in multiple hops rather than all at once. We show Multi-hop FiLM Generator architectures are better able to handle longer sequences than their single-hop counterparts. We outperform state-of-the-art vision-and-language models significantly on the long input sequence GuessWhat?! tasks, while maintaining state-of-the-art performance for the shorter input sequence ReferIt task. Finally, this Multi-hop FiLM Generator approach uses few problem-specific priors, and thus we believe it can extended to a variety of vision-and-language tasks, particularly those requiring complex visual reasoning.

Referit	RefClef (berkeley) Test
SCRC (Hu et al., 2016b)	72.7%
Baseline NN+MLB	74.6%
Single-hop FiLM	84.0%
Multi-hop FiLM	84.3%
Multi-hop FiLM +(img)	**85.1%**

Table 7.5: ReferIt Guesser Test Error.

Figure 7.5: The crop pipeline Oracle's attention. In most of the cases, the attention mask focus on the last question, but it sometimes require solving co-occurrences and attend to other questions.

Figure 7.6: The crop pipeline Guesser's attention.

Part III

Learning Language by interactions, Visually grounded Dialogue system

Chapter 8

Learning Language by Self-Play

«The cake is a lie. »

Introduction to Reinforcement Learning by GlaDOS

Contents

End-to-end dialogue systems has recently become a popular research topic thanks to powerful tools such as encoder-decoder architectures for sequence-to-sequence learning. Yet, most current approaches cast human-machine dialogue management as a supervised learning problem, aiming at predicting the next utterance of a participant given the full history of the dialogue. This vision may fail to correctly render the planning problem inherent to dialogue as well as its contextual and grounded nature. In this chapter, we introduce a deep reinforcement learning method to optimize visually grounded task-oriented dialogues, based on the policy gradient algorithm. We assess out approach on the GuessWhat?! game, and examine whether the agent can learn to consistently manipulating language, and correctly integrating multimodal concepts. We provide encouraging results toward learning consistent dialogue strategy, and uncover new interactive training difficulties, opening the spectrum of research directions for goal-oriented dialogues.

8.1 Introduction

Ever since the formulation of the Turing Test (Turing, 1950), building systems that can meaningfully converse with humans has been a long-standing goal of Artificial Intelligence (AI). Practical dialogue systems have to implement a management strategy that defines the system's behavior, for instance to decide when to provide information or to ask for clarification from the user. Although traditional approaches use linguistically motivated rules (Weizenbaum, 1966), recent methods are data-driven and make use of Reinforcement Learning (RL) (Lemon and Pietquin, 2007). Significant progress in NLP via deep neural networks (Bengio et al., 2003) made neural encoder-decoder architectures a promising way to train conversational agents (Serban et al., 2016; Sordoni et al., 2015; Vinyals and Le, 2015). The main advantage of such end-to-end dialogue systems is that they make no assumption about the application domain and are simply trained in a supervised fashion from large text corpora (Lowe et al., 2015).

However, there are many drawbacks to this approach. First, encoder-decoder models cast the dialogue problem into one of supervised learning, predicting the distribution over possible next utterances given the discourse so far. As with machine translation, this may result in inconsistent dialogues and errors that can accumulate over time. As the action space of dialogue systems is vast, and existing datasets cover only a small subset of all trajectories, it is difficult to generalize to unseen scenarios (Mooney, 2006). Second, the supervised learning framework does not account for the intrinsic planning problem that underlies dialogue, *i.e.* the sequential decision making process, which makes dialogue consistent over time. This is especially true when engaging in a task-oriented dialogue. As a consequence, reinforcement learning has been applied to dialogue systems since the late 90s (Levin et al., 1997; Singh et al., 1999) and dialogue optimization has been generally more studied than dialogue generation. Finally, it is unclear whether encoder-decoder supervised training efficiently integrates external contexts (larger than the history of the dialogue) that is most often used by dialogue participants to interact. This context can be their physical environment, a common task they try to achieve, a map on which they try to find their way, a database they want to access *etc*. These contexts are all the more important as they are part of the so called *Common Ground*, well studied in the discourse literature (Clark and Schaefer, 1989). Over the last decades, the field of cognitive psychology has also brought empirical evidence that human representations are grounded in perception and motor systems (Barsalou, 2008). These theories imply that a dialogue system should be grounded in a multi-modal environment in order to obtain human-level language understanding (Kiela et al., 2016).

On the other hand, RL approaches could handle the planning and the non-differentiable metric problems but require online learning (although batch learning is possible but difficult with low amounts of data (Pietquin et al., 2011)). For that reason, user simulation has been proposed to explore dialogue strategies in RL settings (Eckert et al., 1997; Pietquin and Hastie, 2013; Schatzmann et al., 2006). It also requires the definition of an evaluation metric which is most often related to task completion and user satisfaction (Walker et al., 1997). Without such a goal-achievement metric, it is difficult to correctly evaluate dialogues (Liu et al., 2016). In addition, successful applications of the RL framework to dialogue often rely on a predefined structure of the task, such as slot-filling

Is it a person?	*No*	Is it a cow?	*Yes*
Is it an item being worn or held?	*Yes*	Is it the big cow in the middle?	*No*
Is it a snowboard?	*Yes*	Is the cow on the left?	*No*
Is it the red one?	*No*	On the right ?	*Yes*
Is it the one being held by the person in blue?	*Yes*	First cow near us?	*Yes*

Figure 8.1: Two example games of the GuessWhat?! dataset. The correct object is highlighted by a green mask.

tasks (Williams and Young, 2007) where the task can be cast as filling in a form.

In this chapter, we present an architecture for end-to-end RL optimization of a task-oriented question generator of a dialogue system and its application to a multimodal task, grounding the dialogue in a visual context. To do so, we start from a corpus of 150k human-human dialogues collected via the introduced GuessWhat?! game. The goal of the game is to locate an unknown object in a natural image by asking a series of questions. This task is hard since it requires scene understanding and, more importantly, a dialogue strategy that leads one to rapidly identify the target object. From this data, we first build a supervised agent and a neural training environment. It serves to train a Deep RL agent online which is able to solve the task. We then quantitatively and qualitatively compare the performance of our system to a supervised approach on the same task. In short, our contributions are to propose a visually grounded goal-directed dialogue system optimized via Deep RL; and to achieve 15% improvement on task completion over a supervised learning baseline, and expose the benefice and limitation of our approach.

8.2 GuessWhat?! Game

As a first step, we briefly recap the GuessWhat?! rules, notation and models from Chapter 5 for self-consistency; the accustomed reader can safely bypass this section.

8.2.1 Rules

GuessWhat?! is a cooperative two-player game in which both players see the image of a rich visual scene with several objects. One player – the oracle – is randomly assigned an object (which could be a person) in the scene. This object is not known by the other player – the questioner – whose goal is to locate the hidden object. To do so, the questioner can ask a series of yes-no questions which are answered by the oracle as shown in Fig 8.1. Note that the questioner is not aware of the list of objects and can only see the whole image. Once the questioner has gathered enough evidence to locate the object, he may choose to guess the object. The list of objects is revealed, and if the questioner picks the right object, the game is considered successful.

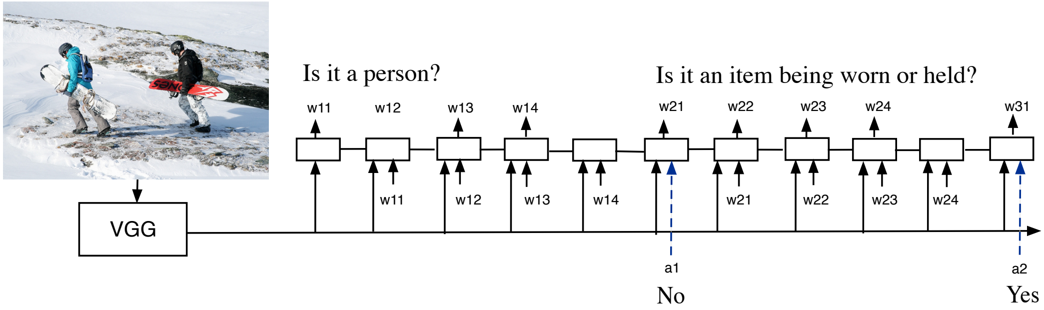


Figure 8.2: Question generation model designed by a single LSTM.

8.2.2 Notation

We briefly recap the GuessWhat?! notation from Chapter 4 with a small change in the vocabulary construction. A game is defined by a tuple $(\mathcal{I}, \mathcal{D}, O, o^*)$ where $\mathcal{I} \in \mathbb{R}^{W \times H}$ is an image of height H and width W, $\mathcal{D}$ a dialogue with J question-answer pairs $\mathcal{D} = (\boldsymbol{q}_j, a_j)_{j=1}^{J}$, O a list of K objects $O = (o_k)_{k=1}^{K}$ and o^* the target object. Moreover, each question $\boldsymbol{q}_j = (w_i^j)_{i=1}^{I_j}$ is a sequence of length I_j with each token w_i^j taken from a predefined vocabulary $\mathcal{V}$. The vocabulary $\mathcal{V}$ is composed of a predefined list of words, a question tag $<?>$ that ends a question and a stop token $<stop>$ that ends a dialogue. An answer is restricted to be either yes, no or not applicable *i.e.* $a_j \in \{<yes>, <no>, <na>\}$. For each object k, an object category $c_k \in \{1, \dots, C\}$ and a pixel-wise segmentation mask $\mathcal{S}_k \in \{0, 1\}^{H \times W}$ are available.

8.2.3 Training Environment

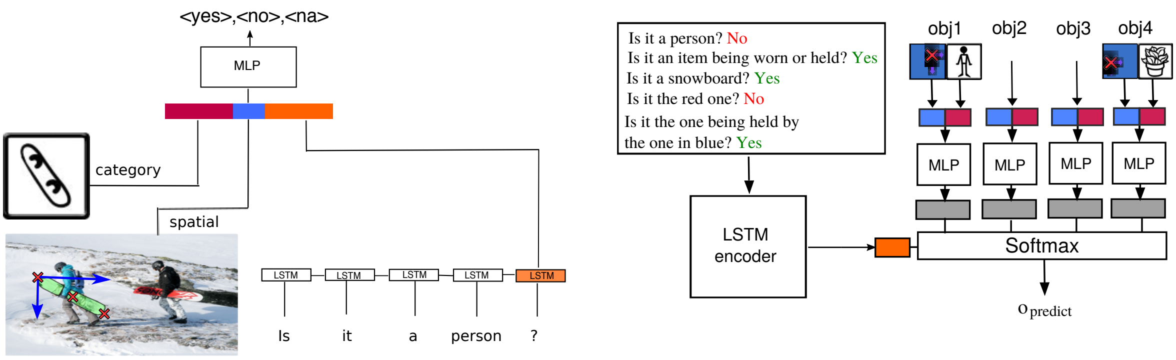
We briefly describe the QGen, oracle and guesser models that are used throughout this chapter. More precisely, we here reproduce the best baseline architectures from Chapter 4 and the more advanced Multi-hop FiLM models from Chapter 7.

Question generation architecture As mentioned in Chapter 4, we split the questioner's job into two different tasks: one for asking the questions and another one for guessing the object. The question generation task requires to produce a new question $\boldsymbol{q}_{j+1}$, given an image $\mathcal{I}$ and a history of J questions and answers pairs $(\boldsymbol{q}, a)_{<j}$.

We model the question generator with an LSTM, which constructs a distribution over tokens w_i^j from vocabulary $\mathcal{V}$. In the case of GuessWhat?!, this output distribution is conditioned on all previous questions and answers tokens as well as the image $\mathcal{I}$:

$$p(w_i^j | w_{i-1<}^j, (\boldsymbol{q}, a)_{j-1<}, \mathcal{I}). \tag{8.1}$$

We condition the model on the image by concatenating VGGNet features (fc8) to the input embedding at each step, as depicted in Fig. 8.2. We train the model by minimizing the conditional negative log-likelihood:



(a) Oracle Model

(b) Guesser Model

Figure 8.3: Oracle and guesser models are conditioned on handcrafted and pre-extracted visual features such as object category and spatial location. Only the Multi-hop FiLM model uses raw image feartures.

$$-\log p(\boldsymbol{q}_{J<}|a_{J<}, \mathcal{I}) = -\log \prod_{j=1}^{J} p(\boldsymbol{q}_j|(\boldsymbol{q}, a)_{j-1<}, \mathcal{I}), \tag{8.2}$$

$$= -\sum_{j=1}^{J}\sum_{i=1}^{I_j} \log p(w_i^j|w_{i-1<}^j, (\boldsymbol{q}, a)_{j-1<}, \mathcal{I}). \tag{8.3}$$

At test time, the samples $p(\boldsymbol{q}_j|(\boldsymbol{q}, a)_{j-1<}, \mathcal{I})$ are sampled as follows. Starting from the state $\boldsymbol{s}_1^j$, we sample a new token w_i^j from the output distribution and feed the embedded token $\boldsymbol{e}_{w_i^j}$ back as input to the RNN. We repeat this loop till we encounter an end-of-sequence token. To approximately find the most likely question, $\max_{\boldsymbol{q}_j} p(\boldsymbol{q}_j|(\boldsymbol{q}, a)_{1<j}, \mathcal{I})$, we use beam-search procedure. This heuristics aims to find the most likely sequence of words by exploring a subset of all questions and keeping the K-most promising candidate sequences at each time step as described in Section 2.2.3.

Oracle The oracle task requires to produce a yes-no answer for any object within an image given a natural language question. In a few words, we embed the spatial information of the crop by extracting an 8-dimensional vector of the location of the bounding box. Second, we convert the object category c^* into a dense category embedding using a learned look-up table. When using the original GuessWhat?! baseline, we use an LSTM to encode the current question $\boldsymbol{q}_j$. We then concatenate all three embeddings into a single vector and feed it as input to a single hidden layer MLP with ReLU activations. The model outputs the answer distribution $p(a|q, c^*, x^*_{spatial})$ using a softmax layer as shown in Fig 8.3a.

In the advanced Multi-hop FiLM model, we use a bi-GRU to encode the current question $\boldsymbol{q}_j$ and the dialogue history $(\boldsymbol{q}, a)_{j<}$. The resulting embedding is then used to jointly modulate the image and crop pipelines with a multi-hop architecture as described in Sec 7.3.2. The two visual embedding are finally concatenated and projected through the same classification layers than the baseline model.

Guesser The guesser model takes an image $\mathcal{I}$ and a sequence of questions and answers $(\boldsymbol{q}, a)_{J<}$, and predicts the correct object o^* from the set of all objects. The baseline model considers a dialogue as one flat sequence of question-answer tokens and uses the last hidden state of the LSTM encoder as our dialogue representation. We perform a dot-product between this representation and the embedding for all the objects in the image, followed by a softmax to obtain a prediction distribution over the objects. The object embeddings are obtained from the categorical and spatial features. More precisely, we concatenate the 8-dimensional spatial representation and the object category look-up and pass it through an MLP layer to get an embedding for the object. The MLP parameters are shared to handle the variable number of objects in the image. See Fig 8.3b for an overview of the guesser.

In the advanced Multi-hop FiLM model, we use the oracle model, and change the evaluation and training procedure to turn it into a guesser. For each new, we iterate over the list of objects O and compute the probability that a dialogue matches with the object such as $p(o = o^*|(\boldsymbol{q}, a)_{\leq J}, \mathcal{I}, \mathcal{C}, c, x_{spatial})$. Finally, we pick the object with the highest probability score.

8.2.4 Generation of Full Games

With the question generation, oracle and guesser model we have all components to simulate a full game. Given an initial image $\mathcal{I}$, we generate a question $\boldsymbol{q}_1$ by sampling tokens from the question generation model until we reach the question-mark token. Alternatively, we can replace the sampling procedure by a beam-search to approximately find the most likely question according to the generator. The oracle then takes the question $\boldsymbol{q}_1$, the object category c^* and $x^*_{spatial}$ as inputs, and outputs the answer a_1. We append $(\boldsymbol{q}_1, a_1)$ to the dialogue and repeat generating question-answer pairs until the generator emits a stop-dialogue token or the maximum number of question-answers is reached. Finally, the guesser model takes the generated dialogue $\mathcal{D}$ and the list of objects O and predicts the correct object.

8.3 GuessWhat?! from RL Perspective

One of the drawbacks of training the QGen in a supervised learning setup is that its sequence of questions is not explicitly optimized to find the correct object. Such training objectives miss the planning aspect underlying (goal-oriented) dialogues. In this chapter, we propose to cast the question generation task as an RL task. More specifically, we use the training environment previously described and consider the oracle and the guesser as part of the RL agent environment. In the following, we first formalize the GuessWhat?! task as an MDP so as to apply a policy gradient algorithm to the QGen problem.

8.3.1 GuessWhat?! as a Markov Decision Process

As mentioned in Sec.1.3.1, an RL environment can be modeled as an MDP (Bellman, 1957; Bertsekas and Tsitsiklis, 1996; Howard, 1960; Puterman, 2014). An MDP is a 5-tuple $(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{R}, \boldsymbol{P}, \gamma)$, whose elements are respectively, the state space, the action space, the reward function, the transition matrix and the discount factor. In the case of GuessWhat?!, we define the state $\boldsymbol{x_t} \in \boldsymbol{X}$ as the status of the game at step t. Specifically, we define $\boldsymbol{x_t} = ((w^j_1, \dots, w^j_i), (\boldsymbol{q}, a)_{j-1<}, \mathcal{I})$ where $t = \sum_{j=1}^{j-1} I_j + i$ corresponds to the number of tokens generated since the beginning of the dialogue. An action $u_t \in \boldsymbol{U}$ corresponds to select a new word w^j_{i+1} in the vocabulary $\mathcal{V}$. The transition to the next state $\boldsymbol{P}$ depends on the selected action:

- If $w^j_{i+1} = <stop>$, the full dialogue is terminated.
- If $w^j_{i+1} = <?>$, the ongoing question is terminated and an answer a_j is sampled from the oracle. The next state is $\boldsymbol{x}_{t+1} = ((\boldsymbol{q}, a)_{j<}, \mathcal{I})$ where $\boldsymbol{q}_j = (w^j_1, \dots, w^j_i, <?>)$.
- Otherwise the new word is appended to the ongoing question and the next state is $\boldsymbol{x}_{t+1} = ((w^j_1, \dots, w^j_i, w^j_{i+1}), (\boldsymbol{q}, a)_{j-1<}, \mathcal{I})$

Questions are arbitrarily terminated after I_{max} words. Similarly, dialogues are terminated after J_{max} questions. Furthermore, a reward $r(\boldsymbol{x}, u)$ is defined for every state-action pair. A trajectory $\tau = (\boldsymbol{x}_t, u_t, \boldsymbol{x}_{t+1}, r(\boldsymbol{x}_t, u_t))_{T<}$ is a finite sequence of tuples of length T which contains a state, an action, the next state and the reward where $T \leq J_{max} * I_{max}$. Thus, the game falls into the episodic RL scenario as the dialogue terminates after a finite sequence of question-answer pairs. Finally, the QGen output can be viewed as a stochastic policy $\pi_{\boldsymbol{\theta}}(u|\boldsymbol{x})$ parametrized by $\boldsymbol{\theta}$ which associates a probability distribution over the actions (i.e. words) for each state (*i.e.* intermediate dialogue and image).

8.3.2 Training the QGen with Policy Gradient

While several approaches exist in the RL literature, we opt for policy gradient methods for they empirically scale well to large action spaces (Chan et al., 2019; Silver et al., 2016; Vinyals et al., 2019), and value-based methods such as Q-learning are proned to over-estimation and instabilities while increasing the action space (Bertsekas and Tsitsiklis, 1996; Thrun and Schwartz, 1993; Zahavy et al., 2018). This action state size constraint is all the more important as the GuessWhat?! vocabulary contains thousands of words. As detailed in Sec. 1.3.6, the policy parameters can be updated in the direction of the gradient of the mean value. We then estimate it over a batch of trajectories $\mathcal{T}_h$ sampled from the current policy $\pi_{\boldsymbol{\theta}_h}$ as follows:

$$\nabla J(\boldsymbol{\theta}_h) = \left\langle \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}_h} \log \pi_{\boldsymbol{\theta}_h}(u_t|\boldsymbol{x}_t)(Q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t) - b) \right\rangle_{\mathcal{T}_h}, \tag{8.4}$$

where $\gamma \in [0, 1]$ is the discount factor, T is the length of the trajectory and $Q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}, u)$ is the state-action value-function $Q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}, u)$ approximated by Monte-Carlo rollouts (Williams, 1992). Following GuessWhat?! game notation, the policy gradient for the QGen can be written as follow:

$$\begin{aligned}\nabla J(\boldsymbol{\theta}_h) = \Bigg\langle \sum_{j=1}^{J} \sum_{i=1}^{I_j} \nabla_{\boldsymbol{\theta}_h} \log \pi_{\boldsymbol{\theta}_h}(w_i^j | w_{i-1<}^j, (\boldsymbol{q}, a)_{j-1<}, \mathcal{I}) \\ (Q^{\pi_{\boldsymbol{\theta}_h}}((w_{i-1<}^j, (\boldsymbol{q}, a)_{j-1<}, \mathcal{I}), w_i^j) - b) \Bigg\rangle_{\mathcal{T}_h}.\end{aligned} \tag{8.5}$$

8.3.3 Reward Function

One tedious aspect of RL is to define a correct and valuable reward function. As the optimal policy is the result of the reward function, one must be careful to design a reward that would not change the expected final optimal policy (Ng et al., 1999). Therefore, we put a minimal amount of prior knowledge into the reward function and construct a zero-one reward depending on the guesser's prediction:

$$r(\boldsymbol{x}_t, u_t) = \begin{cases} 1 & \text{If } \operatorname{argmax}_o [Guesser(\boldsymbol{x}_t)] = o^* \text{ and } t = T \\ 0 & \text{Otherwise} \end{cases}. \tag{8.6}$$

In other words, we give a reward of one if the correct object is found from the generated questions, and zero otherwise. Note that the reward function requires the target object o^* while it is not included in the state $\boldsymbol{x} = ((\boldsymbol{q}, a)_{J<}, \mathcal{I})$. This breaks the MDP assumption that the reward should be a function of the current state and action. However, policy gradient methods with Monte-Carlo rollout are still applicable if the MDP is partially observable (Williams, 1992).

Algorithm 1 Training of QGen with Policy Gradient

Require: Pretrained QGen,Oracle and Guesser
Require: Batch size K
for Each update **do**
    # Generate trajectories $\mathcal{T}_h$
    for $k = 1$ to K **do**
        Pick Image $\mathcal{I}_k$ and the target object $o_k^* \in O_k$
        # Generate question-answer pairs $(\boldsymbol{q}, a)_{1:j}^k$
        for $j = 1$ to J_{max} **do**
            $q_j^k = QGen(\boldsymbol{q}, a)_{1:j-1}^k, \mathcal{I}_k)$
            $a_j^k = Oracle(\boldsymbol{q}_j^k, o_k^*, \mathcal{I}_k)$
            if $<stop> \in \boldsymbol{q}_j^k$ **then**
                delete $(q, a)_j^k$ and break;
            end if
        end for
        $p(o_k|\cdot) = Guesser((q, a)_{1:j}^k, \mathcal{I}_k, O_k)$
        $r(\boldsymbol{x}_t, u_t) = \begin{cases} 1 & \text{If } \operatorname{argmax}_{o_k} p(o_k|\cdot) = o_k^* \\ 0 & \text{Otherwise} \end{cases}$
    end for
    Define $\mathcal{T}_h = ((q, a)_{1:j_k}^k, \mathcal{I}_k, r_k)_{1:K}$
    Evaluate $\nabla J(\boldsymbol{\theta}_h)$ with Eq. (8.5) with $\mathcal{T}_h$
    SGD update of QGen parameters $\boldsymbol{\theta}$ using $\nabla J(\boldsymbol{\theta}_h)$
    Evaluate $\nabla L(\boldsymbol{\phi}_h)$ with Eq. (8.7) with $\mathcal{T}_h$
    SGD update of baseline parameters using $\nabla L(\boldsymbol{\phi}_h)$
end for

8.3.4 Full Training Procedure

We use the QGen, oracle and guesser model architectures outlined in Sec 8.2.3. We first independently train the three models with a cross-entropy loss. We then keep the oracle and guesser models fixed, while we train the QGen in the described RL framework. It is important to pretrain the QGen to kick-start training from a reasonable policy as the size of the action space is simply too big to converge from a random policy.

In order to reduce the variance of the policy gradient, we implement the baseline $b_\phi(\boldsymbol{x}_t)$ as a function of the current state, parameterized by $\boldsymbol{\phi}$. Specifically, we use a one layer MLP which takes the LSTM hidden state of the QGen and predicts the expected reward. We train the baseline function by minimizing the Mean Square Error (MSE) between the predicted reward and the discounted reward of the trajectory at the current time step:

$$L(\boldsymbol{\phi_h}) = \left\langle \left[b_{\phi_h}(\boldsymbol{x}_t) - \sum_{t'=t}^{T} \zeta^{t'} r_{t'} \right]^2 \right\rangle_{\mathcal{T}_h} \tag{8.7}$$

We summarize our training procedure in Algorithm 1.

		New Objects	New Pictures
Supervised Learning	Greedy	43.5%	40.8%
	BSearch	47.1%	44.6%
RL + simple environment	Greedy	60.3%	58.4%
	BSearch	60.2%	58.4%
RL + advanced environment	Greedy	63.5%	60.9%
	BSearch	63.3%	60.9%
Human		84.4%	
Random		18,1%	

Table 8.1: Object retrieval success ratio after generating dialogues with wither the pretrained QGen, or RL agents after interacting with oracle/guesser baselines (simple environment) or Multi-hop FiLM oracle/guesser (advanced environment). New objects refers to uniformly sampling objects within the training set, while new pictures refer to sampling objects from the test set. When sampling for new objects, we obtain a standard deviation below 0.2 accuracy point over 5 runs.

8.4 Experiments

8.4.1 Training Details

We pre-train the three networks as described in Sec. 8.2.3. After training, the oracle and guesser baseline networks respectively obtain 21.5% and 36.2% error, and the Multi-hop FiLM oracle and guesser decrease their respective errors to 17.0% and 29.5%. In the following, we refer to the baseline and Multi-hop FiLM models as the simple and advanced environments.

We then initialize our environment with the pretrained models and train the QGen with policy gradient for 80 epochs with plain stochastic gradient descent (SGD) with a learning rate of 0.001 and a batch size of 64. For each epoch, we sample each training images once, and randomly choose one of its objects as the target. We simultaneously optimize the baseline parameters ϕ with SGD with a learning rate of 0.001. Finally, we set the maximum number of questions to 5 and the maximum number of words to 12.

8.4.2 Reinforcement Learning outperforms Supervised Learning...

Accuracy We report the accuracies of the supervised baseline and the RL agents trained with either the simple or advanced models in Tab. 8.1. We compare sampling objects from the training set (New Objects) and test set (New Images) i.e., unseen images. On the test set, the supervised model obtains 40.8% accuracy while sampling with a greedy policy; this success ratio is improved up to 44.6% with beam search. The RL agents increase the accuracy up to 58.4% when trained with the oracle and guesser baselines and 60.9% with the advanced models, achieving a significant boost of 16% accuracy points over the supervised models. In the following, we indifferently consider both RL agents (except stated otherwise) as their overall behavior is very similar. In light of those results, reinforcement learning has the upper hand over pure supervised learning as the generated dialogues seem to resolve the task better.

Beam Search vs Greedy While beam search is far more effective than a greedy sampling for the supervised model, it has no impact when the models are finetuned by policy gradient. This discrepancy can be explained by the shift of distribution issued by reinforcement learning: cross-entropy optimizes for the most likely next token, while policy gradient optimizes for the trajectory with the highest expected cumulative reward. As a result, the most likely sequence of actions naturally becomes the greedy policy when trained with policy gradient, making the beam search heuristic useless. This further highlights that a supervised learning framework misses the intrinsic planning problem that underlies dialogue as it requires external heuristic to perform well.

Samples We qualitatively compare the two methods by analyzing a few generated samples, as shown in Tab. 8.2. As expected, the supervised models mostly generate human-like questions that are grammatically correct. The table also highlights that the supervised baselines sometimes generate visually relevant but incoherent sequences of questions. For instance, asking "Is it the one to the right of the girl in?" is not a very logical follow-up of "Is it the one in the middle with the red umbrella?". Besides, we observe that the supervised models often keep repeating the same questions, as can be seen in the top two examples in Tab. 8.2. This behavior mainly occurs on the test set *i.e.* when confronted with unseen pictures, which suggests some generalization issues. In the end, we observe a lack of consistency in the generated dialogues, and the QGen mostly asks generic questions that sometimes turn out to sufficient to identify the target object.

The models trained with policy gradient seem to implement a more relevant strategy: they would query the most likely categories "Is it a cat?", before localizing the object "is it in left?", "in middle?". Albeit limited, this strategy makes sense while showing that the agents do learn a consistent policy. Besides, the agents never indefinitely repeat the same question despite sometimes generating duplicates; it would indicate that the agents learn to explore the state space better, and partially recover from unseen scenarios. However, the questions are often ill-formatted and contain grammatical mistakes such as "Is it in middle?". They are also far less diverse than the supervised learning models as examined in the next subsection.

8.4.3 ... but Give Raise to New Problems

Policy gradient unarguably enhances the QGen behavior: the final scores are better, the resulting policies are more consistent than the supervised ones, and some dialogue artifacts, e.g., repeating the same questions, are dramatically reduced. At first sight, reinforcement learning successfully tackles the key difficulties of goal-oriented dialogues: it handles the planning problem that underlies dialogue, and it alleviates the generalization burden of large state space by exploring the environment instead of relying on a static dataset. However, a careful analysis of the generated games highlights the emergence of new issues and limitations.

Dialogue length We observe that supervised models correctly learn to stop asking questions when using greedy sampling; it results in a well-distributed dialogue length over the games. Noticeably, the beam-search heuristic discards the $<stop>$ since it induces short questions that are heavily penalized. However, the policy gradient models completely unlearn to stop. In the current setting, the reward signal does not favor short dialogues over long ones, and the agents learn that prematurely stopping the game may hurt the final performance. A potential solution is to add a small negative reward after each question, as shown by (Shekhar et al., 2018; Zhang et al., 2018a). Yet, this reward shaping induces an extra-engineering layer that does not exist in the supervised framework, and may have unexpected effects on the final policy (Ng et al., 1999).

Question length We observe that the supervised beam-search models generate longer questions (7.1 tokens on average) compared to the policy gradient models (4.0 tokens on average). It suggests that the RL agents produce questions with poor semantic structures since four words neither allow for compositionality nor elaborate descriptions. This qualitative difference is visible in the bottom

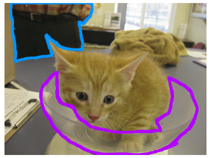
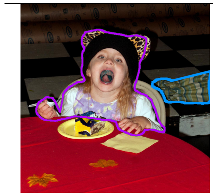
Image	Human	Beam Search	RL + simple environment	RL + advanced environment
	Is it a person ? no Is it being worn ? no Is it a bat ? no Is it a fence ? no Is it dark red ? yes Success (red chair)	Is it a person ? no Is it a ball ? no Is it a ball ? no Is it a ball ? no Is it a ball ? no Failure (blue bat)	Is it a person ? no Is it a ball ? no Is it in left ? no Is it in middle ? no On a person? no Is it on on far right? yes Success (red chair)	Is it person ? no Is it bat ? no Is it ball ? no Is it in left ? no Is in middle ? no
	is it a cat ? no is it a person ? no is it bowl ? yes is the cat in it ? yes Success (bowl)	Is it a cat ? no Is it a book ? no Is it a book ? no Is it a book ? no Is it a book ? no Failure (person)	Is it a cat ? no Is it a table ? no Is it a table ? no Is it in left ? no In middle ? yes Success (bowl)	Is it dog ? no Is it person ? no Is it in left ? no Is in middle ? yes Is it front front ? yes
	Is it a person ? yes Success (girl)	Is it a person ? yes Is it the one in front ? yes Is it the one in the middle with the red umbrella ? yes Is it the one to the right of the girl in ? no ... Failure (umbrella)	Is it a person ? yes Is it in foreground ? yes Is it in left ? yes Is it in middle ? yes Success (girl)	Is it food ? no Is it person ? yes Is it in left ? no Is in middle ? yes
	Is it a primary color ? yes Is it red ? no is it blue ? no Is it on the car ? no is it the house ? no ... Success (most left bag)	Is it a bag ? yes Is it red ? no Is it the one in the middle ? no Is it the one on the far right ? no Is it the one with the blue bag ? yes Success (most left bag)	Is it a suitcase? yes Is it in left ? yes Is it in front ? no Is it in middle ? no Is it in middle ? no Failure (left bag)	Is it car ? no Is it person ? no Is it in left ? yes Is it front ? yes Is it front front ? yes

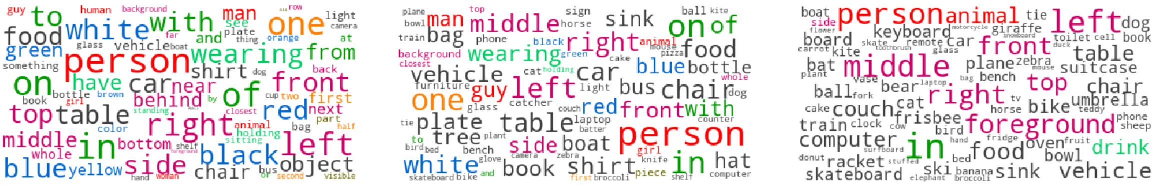
Table 8.2: Samples extracted from the test set. The blue (resp. purple and orange) box corresponds to the object picked by the guesser for the beam-search (resp. RL with greedy policy) dialogue. The small verbose description is added to refer to the object picked by the guesser. Beam search asks diverse questions, but it may indefinitely repeat itself or generate meaningless sentences. RL agents algorithms produce consistent dialogues, but their have a poor language. Surprisingly, improving the agent environment at training time does not increase the diversity of questions.

examples of 8.2, e.g., the supervised model produces precise questions "Is it the one of the far right" while the RL agent remains vague "is it in left?". In other words, the RL agents fail to grasp the composition of language, and therefore simplify the questions to their most basic forms to better compose them throughout the dialogue.

Vocabulary Size We compute the vocabulary size and observe that the number of words dramatically collapses after finetuning the model with policy gradients. While the initial vocabulary contains unique 4.9k words, the supervised models only output around 500 words with greedy sampling, and the policy gradient agents use fewer than a hundred words. More precisely, the higher is the performance, the smaller is the vocabulary size at training time. We assume that it is easier for the agent to over-optimize on a small portion of the environment by slowly discarding the words. In other words, policy gradient converges to highly peaky policies when dealing with large action spaces; it thus reduces the exploration space and eases policy improvement.

Strategy diversity We then explore the vocabulary distribution to grasp which strategies are favored by the policy gradient agents; we display the most frequent words as a cloud word in Fig. 8.4. As first observed in Tab. 8.2, both RL agents either use categorical or spatial words while asking questions. Numbers, colors, genders, or spatial prepositions are discarded from the vocabularies of the agents. When the QGen are trained by interacting with the simple environment, it is somehow expected that the agent tailors its strategy towards such specific features. Indeed, the oracle and guesser baselines do not have access to the image; they only rely on spatial and categorical information of the objects. Therefore, the models are missing key reasoning modalities such as the shape or the colors. However, when the QGen are trained by interacting with the Multi-hop FiLM oracle and guesser, the three models have access to the full spectrum of modalities. It is thus surprising (and disappointing) that the QGen does not take advantage of the new dialogue strategy opportunities. We formulate several hypotheses that are left to future works. For instance, the current strategy involves asking for the most likely object categories before splitting the image into six parts (right/left/middle and top/down), and the agent may quickly degenerate to this unsatisfying but good policy. As previously mentioned, RL agents tend to reduce optimization difficulty by reducing the exploration space, and once the modality is lost, the agent is unlikely to recover it. Therefore, applying an entropy regularization, applying negative rewards on key spatial words, or performing other exploration strategies may force the RL agents to assimilate other modalities correctly. Differently, the Multi-hop FiLM oracle and guesser may be too good at answering categorical and spatial information, naturally biasing the QGen learning toward this specific direction. As an interesting experiment, we can remove hand-crafted object features from the input models. It would increase the errors towards related question-answer pairs, forcing the QGen to use additional visual cues to become robust to the new environment mistakes. On the long run, we expect diverse language strategies to naturally emerge by adapting new RL methods to the language constraint while concurrently improving the environments.

Grammatical Drift As observed in Tab 8.2, the RL agents may produce ill-formatted questions with intriguing anomalies. For instance, one of the RL agents alternates between producing questions with "is it in front ?", "is it front front ?" and "is it front front front ?", suggesting subtle differences between the three formulations. We also observe that determinant, pronouns and grammatical structure are vanishing along the training, leading to question such as "Is it middle", "Is it dog?". Although these inconsistencies are likely induced by overfitting, they also reveal that policy gradient may disregard syntax as long as the core semantics is preserved. As RL aims to optimize the task completion, there is no signal towards preserving grammar structure, and linguistic safeguards must be injected in



(a) Human Generated (b) Pretrained network (c) RL + Baseline environment

Figure 8.4: Word cloud of GuessWhat?! vocabulary with each word proportional to its frequency. Words are colored based on a hand-crafted clustering. Uninformative words such as *"it"*, *"is"* are manually removed. The RL + advanced environment cloud of words is removed as it has a similar to (c).

the training process. Otherwise, if we assumed the strategy diversity to be fixed, we may end producing questions such as "red suitcase left blue" instead of "Is it the red suitcase on the left of the blue one?".

8.5 Related Works

Outside of the dialogue literature, RL methods have been applied to encoder-decoder architectures in machine translation (Bahdanau et al., 2017; Paulus et al., 2018; Ranzato et al., 2016) and image captioning (Liu et al., 2017). In those scenarios, the BLEU score (and its variants) is used as a reward signal to fine-tune a network trained with a cross-entropy loss. However, the BLEU score is a surrogate for human evaluation of naturalness, so directly optimizing this measure does not guarantee an improvement in the translation/captioning quality. In contrast, our reward function encodes task completion, and optimizing this metric is what we aim for. In other words, the BLEU score enforces the word alignment independently of the task at hand. Finally, the BLEU score can only be used in a batch setting because it requires the ground-truth labels from the dataset. In GuessWhat?!, the computed reward is independent of the *human* generated dialogue.

Schatzmann et al. (2006) already reported that RL agents tend to overfit to their environment in goal-oriented dialogues. However, the authors mainly report occurrences over simple simulated users and did not explore natural language generation. In this chapter, we tend to alleviate the first constraint by pretraining the environment over a large dataset, allowing our models to generalize to a large variety of dialogues, whereas previous attempts often use hand-crafted bots that are more prone to overfitting.

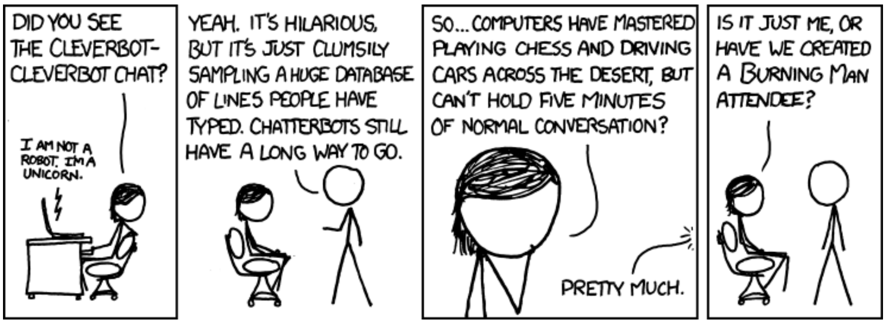
Following this work, Zhu et al. (2017b) examine the impact of jointly train the GuessWhat?! models with policy gradient by sharing the reward signal among the agents. The authors observe that the QGen accuracy would skyrocket while the generated dialogues would become meaningless. In a second step, they analyze whether the oracle's answers are consistent whenever the question can be deciphered, but they observe that the answers seems too erratic. In light of the high accuracy, Zhu et al. (2017b) assume that the oracle successfully encodes the object location within its answers, allowing the guesser to solve the game. In the end, the authors demonstrate further the requirement to constraint the training process to avoid some language drifting, as it results in the emergence of non-understandable communication protocol. Similar results were concurrently observed in other tasks (Lewis et al., 2017), and would lead to the study of language drift in machine learning (Lee et al., 2018a). In this emergent line of research, Zahavy et al. (2018) explore how to reduce the vocabulary degenerescence by dynamically tailoring the word space with contextual bandits. Agarwal et al. (2019) examine how a multi-agent setting may reduce language drift in visually grounded dialogue tasks. Finally, (Shekhar et al., 2019) excellently points out the limitation of purely optimizing tasks completions on GuessWhat?!. The authors introduce original diversity, and quality metrics that are relevant to study further new QGen models.

8.6 Conclusion

In this chapter, we propose to use policy gradient as a novel approach to train deep generative conversational agents, and to examine whether interactive learning may impact visually grounded language generation. We evaluate our approach by casting GuessWhat?! as a reinforcement learning problem, and we observe encouraging improvement over supervised baselines. On the one hand, we notice the emergence of a consistent and efficient dialogue strategy, but on the other hand, the agents tend to produce grammatically mediocre questions and fail to take advantage of the multimodal nature of the task. We then identify several training pitfalls, e.g., vocabulary degenerescence and grammar drift, and that must be fixed first before further assessing visually grounded language understanding.

Chapter 9

Discussion



«Because every Ph.D manuscripts needs its XKCD. »

Ph.D Students

Contents

I am part of the second generation of deep (reinforcement) learning Ph.D. that followed ILSVRC (Krizhevsky et al., 2012) and DQN (Mnih et al., 2013). While the first generation matured deep learning methods alongside the spread of large scale datasets, GPUs, and open-source libraries, our second-generation benefited from their breakthroughs towards exploring new machine learning boundaries. In this thesis, we advocate shifting from unimodal and domain-specific models to multimodal and interactive agents for the betterment of machine understanding. We motivate this perspective in several ways: we pinpoint the limitation and error artifacts of neural networks, we provide correspondences with cognitive science and philosophy, we study new multimodal neural architecture for visual reasoning, and explore reinforcement learning for language.

9.1 Thesis Summary

In Chapter 5, we described the GuessWhat?! game as a support for exploring our research directions, and it perfectly fulfilled its primary purpose: it highlighted the limitation of historical multimodal learning networks and uncovered reinforcement learning pitfalls while generating language. Hence, it made us reconsider the neural fusing mechanisms that independently process each modality before projecting them in a joint representation. In Chapter 3, we thus formalized the conditioning levels to better take into account the hierarchical nature of deep neural networks, and we demonstrated the power of interleaving sensory inputs from the lowest neural processing levels up to the uppermost layers in Chapter 5. To do so, we introduced CBN to modulate a pretrained ResNet to extract multi-modal representation for visual question answering. After this initial proof of concepts, we tailored the newly introduced modulation concept and proposed the FiLM layer as a scalable and straightforward transformation for middle-conditioning in Chapter 6. As a key contribution, FiLM has slowly become a standard new module in the deep learning toolbox. In Chapter 7, we significantly improved the GuessWhat?! models with Multi-hop FiLM by better balancing the computational burden between the visual and linguistic modules. Later, we assembled a multimodal GuessWhat?! environment to apply reinforcement learning for training a conversational agent in a self-play fashion in Chapter 8. Although we showed that the agent could learn an effective strategy and the feasibility of our approach, we also uncovered several forms of language degenerescence by learning through interaction. As a result, we could not benefit from the advance in multimodal knowledge representation developed during the thesis, but we opened new research perspectives for the upcoming years.

In the end, we believe in having provided the following answers during this thesis:

- We obtain higher quality multimodal representation when taking advantage of the hierarchical nature of deep networks
- Interactive conversational agents learn more efficient (albeit unsatisfactory) strategies by interacting with their environment rather than solely generalizing from datasets.

Other questions remain unsolved such as the symbol grounding problem. It is still an open-issue whether multimodal and interactive artificial agents can correctly ground concepts towards apprehending the complexity of the world. Although, we have not yet assess this statement, we hope that our contributions are among the many necessary milestones towards validating (or rebutting) this research direction.

9.2 Future directions

Machine understanding is a large and transverse problem, and we explore some of its many components in this thesis: multimodal and interactive learning. We here list several research directions that naturally follow from this manuscript towards assessing our research goal. They are more or less sorted from the low-hanging fruit proposals to long term research perspectives.

Keep Improving Modulation After the release of attention layers, a large literature emerged to improve this mechanism. The new variants either focus on looking more effective attention modules (Luong et al., 2015), extending attention to new modalities (Xu et al., 2015), or designing original neural architectures (Hudson and Manning, 2018; Sukhbaatar et al., 2015). Similar work could be pursued with modulation: we could use a dynamic stack of modulated ResBlocks, alter it to deal with graph networks, combining it with transformers (Vaswani et al., 2017), assessing with new modalities (Nguyen et al., 2019) etc. As a practical example, most of the state-of-the-art models for visual question answering combine R-CNN features with attention mechanisms (Anderson et al., 2018b), yet, there was no tentative to apply modulation to extract R-CNN features.

Adapting Reinforcement Learning for Language The current NLP literature rely on basic vanilla policy gradient algorithms for tuning generative language models (Das et al., 2017b; Ranzato et al., 2016; Strub et al., 2017b). However, it exists many works that reduce the policy gradient variance (Mnih et al., 2016), enforce stronger policy regularization or smooth policy updates (Schulman et al., 2015, 2017) or the policy exploration (Fortunato et al., 2018; Osband et al., 2016). We believe that a rigorous benchmark over language setting could dramatically improve the current state-of-the-art. As a next step, we could examine how to better integrate the latent structure of language into RL algorithms: classic RL environments are designed to perform well on small action spaces with long-term planning (Bellemare et al., 2013; Mnih et al., 2013), whereas conversational agent can produce short sequences of words within a large vocabulary space. Another interesting problem to tackle is the language drift that occurs while performing self-play. In every cases, linguistic properties could be used to alleviate the training (and exploration) difficulties: syntax encodes a hierarchical structure over language which can be used for Monte-Carlo Tree Search (MCTS), pragmatic is a form of goal conditioning and word2vec enforces distance between actions. In other words, we believe that injecting inductive language bias in the RL algorithms may ease the training of generative language models. On the other hand, language properties, e.g., compositionality, compactness, can be use as a backbone to alleviate policy training.

Virtual embodiment The development of advanced artificial agents is likely to come in pair with the sophistication of tasks to solve. In this thesis, we shift from pure unimodal and static training procedure to multimodal and partially interactive scenarios. However, visually grounded dialogues are still far from the physical reality stated in the embodiment theories (Barsalou, 2008). Ideally, we want to extend machine learning to actual agents such as robots, but the machine learning algorithms are not yet matured, e.g., they are not sample-efficient, and actual robots may either be too limited, too expensive or too hard to program. As an intermediate compromise Kiela et al. (2016), we advocate for developing artificial worlds where multimodal interactions may occur. Such virtual embodied worlds would incorporate basic physics, sounds, visual cues with multiple agents, and various objectives. For instance, we designed Household Multimodal Environment (HoME) as a comprehensive platform to include these features. Other interactive embodied tasks have also blossomed in the past years (Anderson et al., 2018c; Chen et al., 2018; Das et al., 2017b, 2018a; de Vries et al., 2018; Gordon et al., 2018; Hermann et al., 2017; Juliani et al., 2019; Mirowski et al., 2018; Qi et al., 2019; Xia et al., 2018) in physically simulated environments such as DeepMind Lab (Beattie et al., 2016), Baidu XWorld (Yu et al., 2018a), Matterport3D (Chang et al., 2017), GIBSON (Xia et al., 2018), MINOS (Savva et al., 2017), AI2-THOR (Kolve et al., 2017), StreetLearn (Mirowski et al., 2018), AI Habitat (Manolis Savva and Batra, 2019), Unity (Juliani et al., 2018). Although such open-worlds are unarguably necessary in the long-run, it is still unclear whether current virtual worlds are fertile ground for new algorithms, or if we are merely limited to engineer complex ad-hoc agents on artificial toy tasks (Das et al., 2018d; Oh et al., 2017; Wijmans et al., 2019).

Dialogues as Markov Games Goal oriented-dialogues are often designed as two-player cooperative games by casting the training as either an MDP or Partially Observable Markov Decision Process (POMDP) (Williams and Young, 2007). This design choice dramatically eases the underlying theoretical complexity; however, it requires to keep one of the conversational agent statics. For instance, the ticket booking scenario involves training a ticket seller (Weston et al., 2016), GuessWhat?!, and Visual Dialog optimize the question generator (Das et al., 2017b; de Vries et al., 2017), etc. However, this approach quickly becomes limited as it ignores the potential co-adaptation between the conversational agents (Chandramohan et al., 2012), it limits the training to two agents, and it enforces specific reward structures. In practice, natural dialogues may entail numerous agents (Bard et al., 2019), which can be partially co-operative (El Asri et al., 2014) and keep evolving. In this manuscript, we have been promoting interactive learning by pointing out the limitation of controlled environments. Therefore, it is a natural follow-up to enhance the interactive framework for language learning. Following Barlier et al. (2015), we thus advocate to explore general Markov Games (MG) as a training framework for conversational agents (Littman, 1994). MGs may involve multiple players with unstructured rewards, and look for individual strategies toward reaching different equilibria, e.g., Nash Equilibrium. On the other hand, this equilibrium search is far more difficult to reach, e.g., value-function methods are insufficient to find a stationary Nash Equilibrium in MG (Zinkevich et al., 2006), and there is no clear consensus about the equilibrium to reach (Shoham et al., 2007). In this line of research, we explore how to learn an Epsilon Nash Equilibrium in General Markov Games[1] from batch data (Pérolat et al., 2016) as a potential extension for training RL conversational agent from past dialogues (Pietquin et al., 2011). However, there exist few deep extensions of MG algorithms (Heinrich and Silver, 2016; Lanctot et al., 2017; Pérolat et al., 2018), and this research topic is still in its early days.

Rethinking Transfer Learning As discussed in Chapter 5, transfer learning usually involves extracting unimodal features from pretrained networks. Hence, we reduce the computational footprint of embedding large deep networks while training more complex multimodal networks. However, biological visual systems have never been solely trained on unimodal inputs, but jointly evolve alongside other stimuli by counterbalancing each other's defects (Dominy et al., 2004); every human sensor is even interconnected at several perceptive levels (Boutonnet and Lupyan, 2015; Huth et al., 2016), suggesting that we cannot solely rely on late-conditioning to perform transfer learning. Should unimodal transfer learning be avoided? Despite our best efforts, we may not be able to train a model from a single holistic task where the agent may develop its sensors from ground up. We here advocate to smartly compose with independent pretrained network blocks, and slowly rewired them together to increase the model capacity. We may develop internal connections between pretrained unimodal models (Moon et al., 2015), finetuning specific constituents, design non-intrusive neural residual adapters (De Vries* et al., 2017; Rebuffi et al., 2017). In the end, we would like to retrieve unimodal blocks (if possible) after they are being enhanced with multimodal and interactive learning. After reading between the lines, it may require to perform multitask learning as a finetuning procedure. In other words, instead of stacking blocks to solve one specific task, we should be merging blocks for solving multiple problems. We acknowledge the resulting computational issues, but it may be the price for improving the final agent representation.

Final words We would like to thank the brave reader for reaching this final sentence!

[1] An Epsilon Nash Equilibrium is a Nash Equilibrium, where none of the agents can improve its expected reward by an epsilon margin

Chapter 10

References

20 Questions. 20 Questions. http://www.20q.net/, 1988. Accessed: 2019-01. 72

A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59, 1994. 12

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the USENIX Conference on Operating Systems Design and Implementation*, 2016. 17

E. Abbasnejad, Q. Wu, I. Abbasnejad, J. Shi, and A. v. d. Hengel. An active information seeking model for goal-oriented vision-and-language tasks. *arXiv preprint arXiv:1812.06398*, 2018. 81

E. Abbasnejad, Q. Wu, J. Shi, and A. v. d. Hengel. What's to know? uncertainty as a guide to asking goal-oriented questions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 81

J. Abdelnour, G. Salvi, and J. Rouat. Clear: A dataset for compositional language and elementary acoustic reasoning. In *ViGIL*, 2018. 118

S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 36

M. Acharya, K. Jariwala, and C. Kanan. Vqd: Visual query detection in natural scenes. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019. 61

A. Agarwal, O. Chapelle, M. Dudík, and J. Langford. A Reliable Effective Terascale Linear Learning System. *Journal of Machine Learning Research (JMLR)*, 15(1):1111–1133, 2014. 13

A. Agarwal, S. Gurumurthy, V. Sharma, M. Lewis, and K. Sycara. Community regularization of visually-grounded dialog. In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019. 82, 149

A. Agrawal, D. Batra, and D. Parikh. Analyzing the Behavior of Visual Question Answering Models. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. 1, 40, 59, 106

A. Agrawal, D. Batra, D. Parikh, and A. Kembhavi. Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 60

Akinator. Akinator. en.akinator.com/, 2007. Accessed: 2019-01. 69, 72

H. Alamri, V. Cartillier, A. Das, J. Wang, S. Lee, P. Anderson, I. Essa, D. Parikh, D. Batra, A. Cherian, et al. Audio-visual scene-aware dialog. *arXiv preprint arXiv:1901.09107*, 2019. 62

A. Almahairi, S. Rajeswar, A. Sordoni, P. Bachman, and A. Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. In *Proceedings of International Conference on Machine Learning (ICML)*, 2018. 101, 118

M. Alomari, P. Duckworth, D. C. Hogg, and A. G. Cohn. Natural language acquisition and grounding for embodied robotic systems. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 2017. 48

D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016. 22

P. Anderson, B. Fernando, M. Johnson, and S. Gould. Spice: Semantic propositional image caption evaluation. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 58

P. Anderson, A. Das, and Q. Wu. Connecting language and vision to actions. *Proceedings of ACL 2018, Tutorial Abstracts*, 2018a. 48

P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b. 38, 53, 57, 58, 60, 106, 110, 152

P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018c. 48, 51, 57, 62, 153

J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Learning to compose neural networks for question answering. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2016a. 60

J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016b. 60

J. Aneja, A. Deshpande, and A. G. Schwing. Convolutional image captioning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 58

L. Anne Hendricks, R. Hu, T. Darrell, and Z. Akata. Grounding visual explanations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 40, 62

S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, Z. Lawrence, and D. Parikh. Vqa: Visual question answering. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015. 2, 36, 40, 51, 59, 60, 71, 88, 94, 95, 97, 106

R. Arandjelovic and A. Zisserman. Objects that sound. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 51

S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, 2017. 46

P. K. Atrey, M. A. Hossain, A. El Saddik, and M. S. Kankanhalli. Multimodal fusion for multimedia analysis: a survey. *Multimedia systems*, 16(6):345–379, 2010. 50, 51

P. Auer and R. Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2007. 28

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 22, 23, 58, 124, 126

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015. 19, 45, 54

D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. An actor-critic algorithm for sequence prediction. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017. 31, 45, 149

D. Bahdanau, F. Hill, J. Leike, E. Hughes, P. Kohli, and E. Grefenstette. Learning to understand goal specifications by modelling reward. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019a. 118

D. Bahdanau, S. Murty, M. Noukhovitch, T. H. Nguyen, H. de Vries, and A. Courville. Systematic generalization: What is required and can it be learned? In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019b. 60, 118, 131

L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995. 30

D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall Professional Technical Reference, 1st edition, 1982. ISBN 0131653164. 34

T. Baltrušaitis, C. Ahuja, and L.-P. Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443, 2019. 51

S. Banerjee and A. Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005. 58

G. Bani, D. Belli, G. Dagan, A. Geenen, A. Skliar, A. Venkatesh, T. Baumgärtner, E. Bruni, and R. Fernández. Adding object detection skills to visual dialogue agents. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2018. 81

N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes, et al. The hanabi challenge: A new frontier for ai research. *arXiv preprint arXiv:1902.00506*, 2019. 154

M. Barlier, J. Perolat, R. Laroche, and O. Pietquin. Human-machine dialogue as a stochastic game. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2015. 154

K. Barnard, P. Duygulu, D. Forsyth, N. d. Freitas, D. M. Blei, and M. I. Jordan. Matching words and pictures. *Journal of Machine Learning Research (JMLR)*, 3:1107–1135, 2003. 34, 58

M. Baroni. Grounding distributional semantics in the visual world. *Language and Linguistics Compass*, 10(1):3–13, 2016. 62

M. Baroni, G. Dinu, and G. Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2014. 62

A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 50

L. W. Barsalou. Grounded cognition. *Annu. Rev. Psychol.*, 59:617–645, 2008. 1, 48, 50, 138, 153

P. Bartie, W. Mackaness, D. Gkatzia, and V. Rieser. The REAL corpus: A crowd-sourced corpus of human generated and evaluated spatial references to real-world urban scenes. In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*, 2016. 71

D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 52

C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016. 153

M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 29

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. 30, 153

R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, N.J, 1957. 25, 26, 28, 142

H. Ben-Younes, R. Cadene, M. Cord, and N. Thome. Mutan: Multimodal tucker fusion for visual question answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 54, 60, 94, 98

H. Ben-Younes, R. Cadene, N. Thome, and M. Cord. Block: Bilinear superdiagonal fusion for visual question answering and visual relationship detection. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 2019. 54

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 45

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on Neural Networks*, 5(2):157–166, 1994. 19, 25

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research (JMLR)*, 3(Feb):1137–1155, 2003. 41, 42, 138

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2007. 23

T. L. Berg, A. C. Berg, and J. Shih. Automatic attribute discovery and characterization from noisy web data. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2010. 82

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: A cpu and gpu math compiler in python. In *Proceedings of the Python in Science Conference*, 2010. 17

L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 57, 116

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition, 1996. ISBN 1886529108. 25, 26, 27, 28, 30, 31, 142, 143

S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. O'Reilly Media, Inc., 2009. 77

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg, 2006. 12, 13, 20

C. M. Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995. 14, 17, 18, 24

D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2006. 75, 88

O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, M. Macháček, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, and L. Specia. Ninth workshop on Statistical Machine Translation. `http://www.statmt.org/wmt14/translation-task.html`, 2014. Accessed: 2019-01. 43

A. M. Borghi and A. Cangelosi. Action and language integration: From humans to cognitive robots. *Topics in Cognitive Science*, 6(3):344–358, 2014. 48

A. M. Borghi, F. Binkofski, C. Castelfranchi, F. Cimatti, C. Scorolli, and L. Tummolini. The challenge of abstract concepts. *Psychological Bulletin*, 143(3):263, 2017. 48

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. 14

B. Boutonnet and G. Lupyan. Words jump-start vision: A label advantage in object recognition. *Journal of Neuroscience*, 35(25):9329–9335, 2015. 39, 95, 154

J. A. Boyan and A. W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 1995. 30

R. I. Brafman and M. Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 3:213–231, 2002. 29

L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 21

A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019. 101, 118

S. Brodeur, E. Perez, A. Anand, F. Golemo, L. Celotti, F. Strub, J. Rouat, H. Larochelle, and A. Courville. Home: A household multimodal environment. In *Visually-Grounded Interaction and Language workshop (ViGIL) at NeurIPS*, 2017. 48

E. Bruni, N.-K. Tran, and M. Baroni. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49: 1–47, 2014. 51, 52

M. Buhrmester, T. Kwang, and S. Gosling. Amazon's Mechanical Turk a new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5, 2011. 73

V. Bush and J. Wang. As we may think. *Atlantic Monthly*, 176:101–108, 1945. 33

R. Cadene, H. Ben-Younes, N. Thome, and M. Cord. Murel: Multimodal Relational Reasoning for Visual Question Answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 60

A. Cangelosi. Grounding language in action and perception: from cognitive agents to humanoid robots. *Physics of Life Reviews*, 7(2):139–151, 2010. 48

A. Canziani, A. Paszke, and E. Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016. 35

L. Castrejon, Y. Aytar, C. Vondrick, H. Pirsiavash, and A. Torralba. Learning aligned cross-modal representations from weakly aligned data. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 37, 83

J. Y. Chai, R. Fang, C. Liu, and L. She. Collaborative language grounding toward situated human-robot dialogue. *AI Magazine*, 37(4):32–45, 2016. 40, 48

B. Chan, R. Jonathan, S. Szymon, D. Przemysław, F. David, C. Dennison, H. Pondé, F. Wolski, J. Pachocki, G. Brockman, R. Józefowicz, J. Tang, M. Petrov, S. Zhang, D. Luan, J. Schulman, D. Yoon, J. Clark, S. Hashme, I. Sutskever, Q. Fischer, E. Sigler, J. Schneider, L. Schiavo, P. Christiano, S. Gray, C. Hesse, C. Berner, and A. Radford. OpenAI Five. https://blog.openai.com/openai-five/, 2019. Accessed: 2019-01. 30, 50, 143

S. Chandramohan, M. Geist, F. Lefevre, and O. Pietquin. Behavior specific user simulation in spoken dialogue systems. In *Speech Communication; 10. ITG Symposium*, pages 1–4. VDE, 2012. 153

A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niebner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *Proceding of the International Conference on 3D Vision (3DV)*, 2017. 153

M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, 2002. 54

M.-A. Charpagne*, F. Strub*, and T. M. Pollock. Accurate reconstruction of ebsd datasets by a multimodal data approach using an evolutionary algorithm. *Materials Characterization*, 150:184–198, 2019.

P. Chattopadhyay, D. Yadav, V. Prabhu, A. Chandrasekaran, A. Das, S. Lee, D. Batra, and D. Parikh. Evaluating visual conversational agents via cooperative human-ai games. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 2017. 82

H. Chen, A. Shur, D. Misra, N. Snavely, and Y. Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Visually-Grounded Interaction and Language workshop (ViGIL) at NeurIPS*, 2018. 153

X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015. 36, 58

N. Chentanez, A. G. Barto, and S. P. Singh. Intrinsically motivated reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2005. 29

S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014. 17

K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of the Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014a. 44

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014b. 42, 43, 44, 58, 68

N. Chomsky. *Syntactic structures*. The Hague: Mouton, 1957. 40, 48

O. Christopher. Understanding LSTMs. http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015. Accessed: 2019-01. 19, 20

W. Chu and S.-T. Park. Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of International Conference on World Wide Web (WWW)*, 2009. 54

E. S. Chuang, F. Deshpande, and C. Bregler. Facial expression space learning. In *Proceedings of IEEE pacific conference on computer graphics and applications*, 2002. 54

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Deep Learning Workshop at NIPS*, 2014. 19, 108

H. H. Clark and E. F. Schaefer. Contributing to discourse. *Cognitive science*, 13(2):259–294, 1989. 138

R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2008. 18, 41, 42, 43

R. Collobert, S. Bengio, and J. Marithoz. Torch: A modular machine learning software library, 2002. 17

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*, 12:2493–2537, 2011. 41, 42

E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. Stanley, and J. Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 29, 61, 62

T. Cooijmans, N. Ballas, C. Laurent, Ç. Gülçehre, and A. Courville. Recurrent batch normalization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017. 20, 22

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4): 303–314, 1989. 16

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 34

T. Darrell and K. Mooney, Raymond J. amnd Saenko. Workshop on integrating language and vision. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS), Schedule Highlights*, 2011. 34, 48

A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. Moura, D. Parikh, and D. Batra. Visual Dialog. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017a. 36, 73, 82

A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017b. 81, 82, 153

A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied question answering. In *Proceedings of Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018a. 52, 153

A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied Question Answering. In *Proceedings of Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b. 48, 62

A. Das, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Neural Modular Control for Embodied Question Answering. In *Proceedings of conference on robot learning (CoRL)*, 2018c. 48, 62

A. Das, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Neural modular control for embodied question answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018d. 153

S. Datta, K. Sikka, A. Roy, K. Ahuja, D. Parikh, and A. Divakaran. Align2ground: Weakly supervised phrase grounding guided by image-caption alignment. *arXiv preprint arXiv:1903.11649*, 2019. 62

J. de Lope, D. Maravall, et al. Robust high performance reinforcement learning through weighted k-nearest neighbors. *Neurocomputing*, 74(8):1251–1259, 2011. 30

H. de Vries, F. Strub, S. Chandar, O. Pietquin, H. Larochelle, and A. Courville. GuessWhat?! Visual object discovery through multi-modal dialogue. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 36, 95, 99, 153, II

H. De Vries*, F. Strub*, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 154, II

H. de Vries, K. Shuster, D. Batra, D. Parikh, J. Weston, and D. Kiela. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*, 2018. 82, 153

J.-B. Delbrouck and S. Dupont. Modulating and attending the source image during encoding improves multimodal translation. *Visually-Grounded Interaction and Language Workshop (NeurIPS)*, 2017. 101

C. Deng, Q. Wu, Q. Wu, F. Hu, F. Lyu, and M. Tan. Visual grounding via accumulated attention. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 81

B. Dessalegn and B. Landau. More than meets the eye: The role of language in binding and maintaining feature conjunctions. *Psychological Science*, 19(2):189–195, 2008. 39, 48

B. Dessalegn and B. Landau. Interaction between language and vision: It's momentary, abstract, and it develops. *Cognition*, 127(3):331–344, 2013. 39, 48

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2, 46, 131

R. Dey and F. M. Salemt. Gate-variants of gated recurrent unit (gru) neural networks. In *Proceedings of International Midwest Symposium on the Circuits and Systems (MWSCAS)*, 2017. 20

A. Dhall, R. Goecke, and T. Gedeon. Emotion recognition in the wild challenge (emotiw). In *Proceedings of International Conference on Multimodal Interaction (ICMI), Workshop Schedule Highlights*, 2019. 50

T. G. Dietterich. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, 2000. 21

J. Dodge, A. Gane, X. Zhang, A. Bordes, S. Chopra, A. Miller, A. Szlam, and J. Weston. Evaluating prerequisite qualities for learning end-to-end dialog systems. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2016. 68, 72

P. Dogan, L. Sigal, and M. Gross. Neural sequential phrase grounding (seqground). *arXiv preprint arXiv:1903.07669*, 2019. 62

N. J. Dominy, C. F. Ross, and T. D. Smith. Evolution of the special senses in primates: past, present, and future. *The Anatomical Record Part A: Discoveries in Molecular, Cellular, and Evolutionary Biology: An Official Publication of the American Association of Anatomists*, 281(1):1078–1082, 2004. 154

L. E. Dostert. The georgetown–i.b.m. experiment. Technical report, Locke and Booth, 1955. 33

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011. 24

B. Dumas, D. Lalanne, and S. Oviatt. Multimodal interfaces: A survey of principles, models and frameworks. In *Human machine interaction*, pages 3–26. Springer, 2009. 40, 50

V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017. 95, 101, 107

V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. d. Vries, A. Courville, and Y. Bengio. Feature-wise transformations. *Distill*, 2018. https://distill.pub/2018/feature-wise-transformations. 118

S. Dupont and J. Luettin. Audio-visual speech modeling for continuous speech recognition. *IEEE transactions on multimedia*, 2(3):141–151, 2000. 50

W. Eckert, E. Levin, and R. Pieraccini. User modeling for spoken dialogue system evaluation. In *Proceedings of Automatic Speech Recognition and Understanding, Workshop Track*, 1997. 138

L. El Asri, R. Laroche, and O. Pietquin. DINASTI: Dialogues with a Negotiating Appointment Setting Interface. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2014. 154

A. El-Nouby, S. Sharma, H. Schulz, D. Hjelm, L. E. Asri, S. E. Kahou, Y. Bengio, and G. W. Taylor. Keep drawing it: Iterative language-based image generation and editing. In *Visually-Grounded Interaction and Language workshop (ViGIL) at NeurIPS*, 2018. 2

M. Elhoseiny, B. Saleh, and A. Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2013. 39

D. Elliott and F. Keller. Comparing automatic evaluation measures for image description. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL), Short Paper*, 2014. 72

J. L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3): 195–225, 1991. 42

G. Elsayed, S. Shankar, B. Cheung, N. Papernot, A. Kurakin, I. Goodfellow, and J. Sohl-Dickstein. Adversarial examples that fool both computer vision and time-limited humans. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 39

Y. Engel, S. Mannor, and R. Meir. Bayes meets bellman: The gaussian process approach to temporal difference learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2003. 30

D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Artificial Intelligence and Statistics*, 2009. 23

D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005. 30

H. J. Escalante, C. A. Hernández, J. A. Gonzalez, A. López-López, M. Montes, E. F. Morales, L. E. Sucar, L. Villaseñor, and M. Grubinger. The segmented and annotated iapr tc-12 benchmark. *Computer Vision and image Understanding*, 114(4):419–428, 2010. 61

L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of International Conference on Machine Learning (ICML)*, 2018. 32

M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision (IJCV)*, 111(1):98–136, 2015. 34, 38, 58

A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth. Every picture tells a story: Generating sentences from images. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2010. 40

L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007. 34

Y. Feng and M. Lapata. How many words is a picture worth? automatic caption generation for news images. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2010. 58, 71

B. Ferguson and S. Waxman. Linking language and categorization in infancy. *Journal of child language*, 44(3):527–552, 2017. 39

F. Ferreira and M. K. Tanenhaus. Introduction to the special issue on language–vision interactions. *Journal of Memory and Language*, 57(4):455–459, 2007. 94

D. A. Ferrucci. Introduction to “this is watson”. *IBM Journal of Research and Development*, 56(3.4):1–1, 2012. 41

C. Firestone and B. J. Scholl. “top-down” effects where none should be found: The el greco fallacy in perception research. *Psychological Science*, 25(1):38–46, 2014. 39

J. R. Firth. A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis (special volume of the Philological Society)*, 1952-59:1–32, 1957. 41

M. Fortunato, M. Gheshlaghi Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg. Noisy networks for exploration. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018. 29, 153

M. Freitag and Y. Al-Onaizan. Beam search strategies for neural machine translation. In *Proceedings of Workshop on Neural Machine Translation*, 2017. 45

A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2013. 39

A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. 53, 54, 55, 59, 60, 94, 98, 99

Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 21

Z. Gan, Y. Cheng, A. E. Kholy, L. Li, J. Liu, and J. Gao. Multi-step reasoning via recurrent dual attention for visual dialog. *arXiv preprint arXiv:1902.00579*, 2019. 56, 57, 82

H. Gao, J. Mao, J. Zhou, Z. Huang, L. Wang, and W. Xu. Are you talking to a machine? dataset and methods for multilingual image question. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 59

J. Gao, M. Galley, L. Li, et al. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298, 2019. 1, 46, 72

P. Gao, H. Li, S. Li, P. Lu, Y. Li, S. C. Hoi, and X. Wang. Question-guided hybrid convolution for visual question answering. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 58, 60, 61

A. Gargett, K. Garoufi, A. Koller, and K. Striegnitz. The give-2 corpus of giving instructions in virtual environments. In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*, 2010. 34, 62

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2017. 18, 42

R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019. 38

M. Geist and B. Scherrer. Off-policy learning with eligibility traces: A survey. *Journal of Machine Learning Research (JMLR)*, 15(1):289–333, 2014. 29

D. Geman, S. Geman, N. Hallonquist, and L. Younes. Visual turing test for computer vision systems. *Proceedings of the National Academy of Sciences*, 112(12):3618–3623, 2015. 38, 59, 106

F. A. Gers and E. Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001. 20

G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. In *Proceedings of the British Machine Vision Conference*, 2017. 101, 107

R. W. Gibbs Jr. *Embodiment and cognitive science*. Cambridge University Press, 2005. 1, 48

D. Gkatzia, V. Rieser, P. Bartie, and W. Mackaness. From the virtual to the RealWorld: Referring to objects in real-world spatial scenes. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2015. 61

A. M. Glenberg. Few believe the world is flat: How embodiment is changing the scientific understanding of cognition. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 69(2):165, 2015. 48

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of International Conference Artificial Intelligence and Statistics (AIStats)*, 2010. 16, 23

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016. 13, 14, 17, 21, 23, 68

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015. 1, 38

D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. Iqa: Visual question answering in interactive environments. In *Proceedings of Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 62, 153

Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 59, 71, 94, 106

A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 19

A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013. 19

A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014. 46, 130

A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538 (7626):471, 2016. 46, 130

K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. In *Proceedings of International Conference on Machine Learning (ICML)*, 2015. 19

I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42 (6):1291–1307, 2012. 32

A. Gruslys, W. Dabney, M. Gheshlaghi Azar, B. Piot, M. Bellemare, and R. Munos. The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018. 32

X. Guo, H. Wu, Y. Cheng, S. Rennie, G. Tesauro, and R. Feris. Dialog-based interactive image retrieval. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 82

D. Ha, A. Dai, and Q. V. Le. Hypernetworks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2016. 57

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of International Conference on Machine Learning (ICML)*, 2018. 32

D. L. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, 1997. 51

C. Han, Y. Heo, W. Kang, J. Jun, and B.-T. Zhang. Attention memory for locating an object through visual dialogue. In *VQA Challenge Workshop at CVPR*, 2017a. 81

C. Han, S.-W. Lee, Y. Heo, W. Kang, J. Jun, and B.-T. Zhang. Criteria for human-compatible ai in two-player vision-language tasks. In *Linguistic and Cognitive Approaches to Dialogue Agents Workshop at IJCAI*, 2017b. 81

S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346, 1990. 1, 34, 47

Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954. 41

H. V. Hasselt. Double q-learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2010. 31

H. v. Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 2016. 30, 31

O. Hauk, I. Johnsrude, and F. Pulvermüller. Somatotopic representation of action words in human motor and premotor cortex. *Neuron*, 41(2):301–307, 2004. 48

M. Haurilet, A. Roitberg, and R. Stiefelhagen. It's not about the journey; it's about the destination: Following soft paths under question-guidance for visual reasoning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 61

P. J. Hayes. The naive physics manifesto. In *Proceedings of Expert Systems in the Micro-Electronic Age*, 1978. 48, 63

P. J. Hayes. The second naive physics manifesto. In R. S. . J. Bransford, editor, *Formal Theories of the Commonsense World*, 1985. 1, 48

F. Hayes-Roth. Rule-based systems. *Commun. ACM*, 28(9):921–932, 1985. 12

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 23

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 35, 37, 46, 94, 109, 126

K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017. 37

X. He and Y. Peng. Fine-grained image classification via combining vision and language. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 39

J. Heinrich and D. Silver. Deep reinforcement learning from self-play in imperfect-information games. In *Deep Reinforcement Learning Workshop at NeurIPS*, 2016. 154

K. M. Hermann, F. Hill, S. Green, F. Wang, R. Faulkner, H. Soyer, D. Szepesvari, W. M. Czarnecki, M. Jaderberg, D. Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017. 48, 153

G. Herzog and P. Wazinski. Visual translator: Linking perceptions and natural language descriptions. *Artificial Intelligence Review*, 8(2-3):175–187, 1994. 34

M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 2018. 30

T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al. Deep q-learning from demonstrations. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 2018. 29

G. E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Conference of the Cognitive Science Society*, volume 1. Amherst, MA, 1986. 42

S. Hochreiter and J. Schmidhuber. Long short-term memory. In *Neural computation*, volume 9, pages 1735–1780. MIT Press, 1997. 19, 77, 79

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. 16

N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. *arXiv preprint arXiv:1902.00751*, 2019. 38

R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960. 25, 26, 142

J. Hu, L. Shen, and G. Sun. Squeeze-and-Excitation Networks. In *ILSVRC Workshop at CVPR*, 2017a. 35

R. Hu, M. Rohrbach, and T. Darrell. Segmentation from natural language expressions. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016a. 62, 130

R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell. Natural language object retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016b. 62, 77, 125, 130, 131

R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017b. 60, 106, 109, 110, 111

R. Hu, M. Rohrbach, J. Andreas, T. Darrell, and K. Saenko. Modeling relationships in referential expressions with compositional modular networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017c. 61

T.-H. K. Huang, F. Ferraro, N. Mostafazadeh, I. Misra, J. Devlin, A. Agrawal, R. Girshick, X. He, P. Kohli, D. Batra, et al. Visual storytelling. In *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*, 2016. 62

X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of international conference on learning representations (ICLR), Workshop Track*, 2017. 107

D. A. Hudson and C. D. Manning. Compositional attention networks for machine reasoning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018. 56, 61, 110, 111, 131, 152

D. A. Hudson and C. D. Manning. Gqa: a new dataset for compositional question answering over real-world images. *arXiv preprint arXiv:1902.09506*, 2019. 60, 61

J. Hutchins. Alpac: the (in) famous report. *Readings in Machine Translation*, 14:131–135, 2003. 33

A. G. Huth, W. A. de Heer, T. L. Griffiths, F. E. Theunissen, and J. L. Gallant. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532(7600):453, 2016. 48, 154

S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of International Conference on Machine Learning (ICML)*, 2015. 21, 22, 35, 95, 96, 126

R. Jackendoff. On beyond Zebra: The relation of linguistic and visual information. *Cognition*, 26(2):89–114, 1987. 34

Jackyalciné. Tweet: *Google Photos, y'all fucked up. My friend's not a gorilla.* `https://twitter.com/jackyalcine/status/615329515909156865/`, 2015. Accessed: 2019-01. 38

A. Jaimes and N. Sebe. Multimodal human–computer interaction: A survey. *Computer vision and image understanding*, 108(1-2):116–134, 2007. 40, 50

U. Jain, S. Lazebnik, and A. G. Schwing. Two can play this game: Visual dialog with discriminative question generation and answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 82

T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010. 28

Y. Jang, Y. Song, Y. Yu, Y. Kim, and G. Kim. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 62

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, 2014. 17

X. Jiang, M. Havaei, F. Varno, G. Chartrand, N. Chapados, and S. Matwin. Learning to learn with conditional class dependencies. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019. 118

Y. Jiang, V. Natarajan, X. Chen, M. Rohrbach, D. Batra, and D. Parikh. Pythia v0. 1: the winning entry to the vqa challenge 2018. *VQA Challenge Workshop at CVPR*, 2018. 57, 60

J. Jiasen, J. Yang, D. Batra, and D. Parikh. Hierarchical question-image co-attention for visual question answering. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 53, 56, 59, 60, 94, 106, 110, 124

J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 62

J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017a. 60, 61, 105, 106, 109, 110, 116, 122

J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017b. 60, 106, 109, 110, 111, 115, 116

A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. In *Proceedings of Conference of the European Chapter of the Association for Computational*, 2017. 43

A. Juliani, V.-P. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018. 153

A. Juliani, A. Khalifa, V.-P. Berges, J. Harper, H. Henry, A. Crespi, J. Togelius, and D. Lange. The Obstacle Tower: A generalization challenge in vision, control, and planning. In *Workshop on Games and Simulations for Artificial Intelligence at AAAI*, 2019. 153

K. Kafle and C. Kanan. An analysis of visual question answering algorithms. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 59, 60

S. M. Kakade et al. *On the sample complexity of reinforcement learning*. PhD thesis, University College London, 2003. 29

D. Kang, D. Dhar, and A. Chan. Incorporating side information by adaptive convolution. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 58

G.-C. Kang, J. Lim, and B.-T. Zhang. Dual attention networks for visual reference resolution in visual dialog. *arXiv preprint arXiv:1902.09368*, 2019. 57, 82

A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 58

P. Kay and W. Kempton. What is the sapir-whorf hypothesis? *American anthropologist*, 86(1):65–79, 1984. 39

S. Kazemzadeh, V. Ordonez, M. Matten, and T. Berg. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. 36, 61, 69, 71, 122, 125, 130

C. A. Kiela, Douwe and, A. Jabri, and M. Nickel. Learning visually grounded sentence representations. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018. 62

D. Kiela. Deep embodiment: grounding semantics in perceptual modalities. Technical report, University of Cambridge, Computer Laboratory, 2017. 47, 48, 51, 62

D. Kiela and S. Clark. Multi-and cross-modal semantics beyond vision: Grounding in auditory perception. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015. 47

D. Kiela, L. Bulat, and S. Clark. Grounding semantics in olfactory perception. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), Short Paper*, 2015. 50

D. Kiela, L. Bulat, A. L. Vero, and S. Clark. Virtual embodiment: A scalable long-term strategy for artificial intelligence research. In *Machine intelligence workshop at NeurIPS*, 2016. 48, 138, 153

J.-H. Kim, S.-W. Lee, D. Kwak, M.-O. Heo, J. Kim, J.-W. Ha, and B.-Y. Zhang. Multimodal Residual Learning for Visual QA. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 94

J.-H. Kim, K. W. On, W. Lim, J. Kim, J.-W. Ha, and B.-T. Zhang. Hadamard Product for Low-rank Bilinear Pooling. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017a. 54, 55, 56, 59, 60, 94, 98, 99, 126

J.-H. Kim, J. Jun, and B.-T. Zhang. Bilinear attention networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 54, 57, 60

T. Kim, I. Song, and Y. Bengio. Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition. In *Proceedings of European Conference on Speech Communication and Technologies (Interspeech)*, 2017b. 107, 118

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015. 24, 77, 109, 127

W. Kintsch. *Comprehension: A paradigm for cognition*. Cambridge university press, 1998. 48

R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 2, 22, 46

N. Klemfuss, B. Prinzmetal, and R. B. Ivry. How does language change perception: a cautionary note. *Frontiers in psychology*, 3:78, 2012. 39

P. Kok, M. Failing, and F. de Lange. Prior expectations evoke stimulus templates in the primary visual cortex. *Journal of Cognitive Neuroscience*, 26(7):1546–1554, 2014. 39, 95

E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 153

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. 54

S. Kornblith, J. Shlens, and Q. V. Le. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*, 2018. 38

S. Kottur, R. Vedantam, J. M. Moura, and D. Parikh. Visual word2vec (vis-w2v): Learning visually grounded word embeddings using abstract scenes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 47, 62

S. Kottur, J. M. Moura, D. Parikh, D. Batra, and M. Rohrbach. Visual coreference resolution in visual dialog using neural module networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 82

S. Kottur, J. M. Moura, D. Parikh, D. Batra, and M. Rohrbach. Clevr-dialog: A diagnostic dataset for multi-round reasoning in visual dialog. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 61, 81

E. Krahmer and K. V. Deemter. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218, 2012. 68

M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2): 233–243, 1991. 43

R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision (IJCV)*, 123:32–73, 2017. 36, 38, 58, 59, 60, 68

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2012. 2, 16, 18, 20, 21, 23, 34, 35, 36, 37, 151

A. Kuhnle and A. Copestake. The meaning of" most" for visual question answering models. *arXiv preprint arXiv:1812.11737*, 2018. 118

A. Kuhnle, H. Xie, and A. Copestake. How clever is the film model, and how clever can it be? In *Proceedings of European Conference on Computer Vision (ECCV)*, 2018. 60, 115, 118

A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016. 46

V. Kuperman, H. Stadthagen-Gonzalez, and M. Brysbaert. Age-of-acquisition ratings for 30,000 english words. *Behavior research methods*, 44(4):978–990, 2012. 47

A. M. Lamb, A. G. ALIAS PARTH GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 45

M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4190–4203, 2017. 154

B. Landau, L. B. Smith, and S. S. Jones. The importance of shape in early lexical learning. *Cognitive development*, 3(3): 299–321, 1988. 38, 48

B. Landau, L. Smith, and S. Jones. Object perception and object naming in early development. *Trends in cognitive sciences*, 2(1):19–24, 1998. 48

C. Landsiedel, V. Rieser, M. Walter, and D. Wollherr. A review of spatial reasoning and interaction for real-world robotics. *Advanced Robotics*, 31(5):222–242, 2017. 48

M. Lapan. *Deep Reinforcement Learning Hands-On: Apply Modern RL Methods, with Deep Q-Networks, Value Iteration, Policy Gradients, TRPO, AlphaGo Zero and More*. Expert insight. Packt Publishing, 2018. 32

C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio. Batch normalized recurrent neural networks. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016. 22

A. Lazaridou, E. Bruni, and M. Baroni. Is this a wampimuk? cross-modal mapping between distributional semantics and the visual world. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014. 48

A. Lazaridou, G. Dinu, A. Liska, and M. Baroni. From visual attributes to adjectives through decompositional distributional semantics. *Transactions of the Association for Computational Linguistics*, 3:183–196, 2015a. 51, 52

A. Lazaridou, N. T. Pham, and M. Baroni. Combining language and vision with a multimodal skip-gram model. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2015b. 47, 62

A. A. Lazarus. Multimodal behavior therapy: Treating the" basic id.". *Journal of Nervous and Mental Disease*, 1973. 50

Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of International Conference on Machine Learning (ICML)*, 2014. 46

Y. Lecun. MNIST Dataset. http://yann.lecun.com/exdb/mnist/, 1998. Accessed: 2019-01. 18

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 17

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998a. 16, 17, 18, 24

Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 9–50. Springer-Verlag, 1998b. 15, 16, 23

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 14, 16, 18, 52, 68

H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of International Conference on Machine Learning (ICML)*, 2009. 37

J. Lee, K. Cho, and D. Kiela. Countering language drift via grounding. In *Emergent Communication Workshop at NeurIPS*, 2018a. 149

S.-W. Lee, Y.-J. Heo, and B.-T. Zhang. Answerer in questioner's mind: Information theoretic approach to goal-oriented visual dialog. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018b. 81, 82, 83, 130

S.-W. Lee, T. Gao, S. Yang, J. Yoo, and J.-W. Ha. Large-scale answerer in questioner's mind for visual dialog question generation. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019. 82

J. Lei, L. Yu, M. Bansal, and T. L. Berg. Tvqa: Localized, compositional video question answering. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. 62, 71

O. Lemon and O. Pietquin. Machine learning for spoken dialogue systems. In *Proceedings of European Conference on Speech Communication and Technologies (Interspeech)*, 2007. 138

O. Lemon and O. Pietquin. *Data-Driven Methods for Adaptive Spoken Dialogue Systems*. Springer, 2012. 68

E. Levin and R. Pieraccini. A stochastic model of computer-human interaction for learning dialogue strategies. In *Proceedings of European Conference on Speech Communication and Technologies (Interspeech)*, volume 97, 1997. 68

E. Levin, R. Pieraccini, and W. Eckert. Learning dialogue strategies within the markov decision process framework. In *Workshop on Automatic Speech Recognition and Understanding Proceedings*, 1997. 138

O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 43

M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):1–19, 2006. 50, 58

M. Lewis, D. Yarats, Y. Dauphin, D. Parikh, and D. Batra. Deal or no deal? end-to-end learning of negotiation dialogues. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017. 149

F.-F. Li, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006. 83

J. Li, M. Galley, C. Brockett, G. Spithourakis, J. Gao, and B. Dolan. A persona-based neural conversation model. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2016a. 47

J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao. Deep reinforcement learning for dialogue generation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016b. 72

R. Li, K. Li, Y.-C. Kuo, M. Shu, X. Qi, X. Shen, and J. Jia. Referring image segmentation via recurrent refinement networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 62

Y. Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017. 30

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 32

C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 2004. 58

L.-J. Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993. 29

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2014. 2, 36, 38, 39, 40, 58, 61, 68, 69, 71, 73, 125

T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017. 37

H. Ling and S. Fidler. Teaching machines to describe images via natural language feedback. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5075–5085. Curran Associates Inc., 2017. 40, 62

M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings*, pages 157–163. Elsevier, 1994. 154

C. Liu and J. Y. Chai. Learning to mediate perceptual differences in situated human-robot dialogue. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 2015. 48

C. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*, 2016. 68, 72, 138

D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989. 14

R. Liu, C. Liu, Y. Bai, and A. Yuille. Clevr-ref+: Diagnosing visual reasoning with referring expressions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019a. 61

S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy. Improved image captioning via policy gradient optimization of spider. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 59, 149

X. Liu, Z. Wang, J. Shao, X. Wang, and H. Li. Improving referring expression grounding with cross-modal attention-guided erasing. *arXiv preprint arXiv:1903.00839*, 2019b. 127

Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma. A survey of content-based image retrieval with high-level semantics. *Pattern recognition*, 40(1):262–282, 2007. 51

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019. 21, 24

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2): 91–110, 2004. 34

R. Lowe, N. Pow, I. Serban, and J. Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*, 2015. 17, 72, 138

J. Lu, A. Kannan, J. Yang, D. Parikh, and D. Batra. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017a. 82

J. Lu, C. Xiong, D. Parikh, and R. Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017b. 58

T. Lu, D. Schuurmans, and C. Boutilier. Non-delusional q-learning and value-iteration. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 31

P. Luo, X. Wang, W. Shao, and Z. Peng. Towards understanding regularization in batch normalization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019. 22

R. Luo and G. Shakhnarovich. Comprehension-guided referring expressions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 62, 130

T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015. 45, 54, 124, 152

G. Lupyan and E. J. Ward. Language can boost otherwise unseen objects into visual awareness. *Proceedings of the National Academy of Sciences*, 110(35):14196–14201, 2013. 39

M. Malinowski. *Towards Holistic Machines: From Visual Recognition To Question Answering About Real-World Images*. PhD thesis, Saarland University, 2017. 2, 35, 38, 40, 51, 59

M. Malinowski and C. Doersch. The visual qa devil in the details: The impact of early fusion and batch norm on clevr. In *Proceedings of European Conference on Computer Vision (ECCV), Workshop Track*, 2018. 53, 61, 110, 111

M. Malinowski and M. Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 59, 106

M. Malinowski, M. Rohrbach, and M. Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 34, 60, 94, 106

M. Malinowski, C. Doersch, A. Santoro, and P. Battaglia. Learning visual question answering by bootstrapping hard attention. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2018a. 60

M. Malinowski, C. Doersch, A. Santoro, and P. Battaglia. Learning visual question answering by bootstrapping hard attention. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018b. 55

C. D. Manning, C. D. Manning, and H. Schütze. *Foundations of statistical natural language processing*. MIT press, 1999. 12, 40, 41, 42

O. Y. Z. E. W. B. J. J. S. J. L. V. K. J. M. D. P. Manolis Savva , Abhishek Kadian and D. Batra. Habitat: A platform for embodied ai research. *arXiv preprint arXiv:1904.01201*, 2019. 153

J. Mao, J. Huang, A. Toshev, O. Camburu, A. Yuille, and K. Murphy. Generation and comprehension of unambiguous object descriptions. *arXiv preprint arXiv:1511.02283*, 2015. 61, 62, 71

V. Marois, T. Jayram, V. Albouy, T. Kornuta, Y. Bouhadjar, and A. S. Ozcan. On transfer learning using a mac model variant. In *Visually-Grounded Interaction and Language workshop (ViGIL) at NeurIPS*, 2018. 61

D. D. McDonald and E. J. Conklin. Salience as a simplifying metaphor for natural language generation. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 1982. 34, 58

H. McGurk and J. MacDonald. Hearing lips and seeing voices. *Nature*, 264(5588):746, 1976. 50

T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Proceedings of Conference of the International Speech Communication Association (ISCA)*, 2010. 42

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of International Conference on Learning Representations (ICLR), Workshop Track*, 2013a. 42, 43

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2013b. 42, 43, 117

T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2013c. 42

G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 39

G. A. Miller and P. N. Johnson-Laird. *Language and perception.* Havard University Press, 1976. 34, 50

M. Mironenco, D. Kianfar, K. Tran, E. Kanoulas, and E. Gavves. Examining cooperation in visual dialog models. In *Visually-Grounded Interaction and Language workshop (ViGIL) at NeurIPS*, 2017. 82

P. Mirowski, M. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, A. Zisserman, R. Hadsell, et al. Learning to navigate in cities without a map. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 153

M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 50

T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. 11, 12, 13

T. Miyato and M. Koyama. cGANs with projection discriminator. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018. 101, 118

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 30, 50, 151, 153

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015. 30

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016. 30, 32, 153

S. Moon, S. Kim, and H. Wang. Multimodal transfer deep learning with applications in audio-visual recognition. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS), Workshop Track*, 2015. 154

R. J. Mooney. Learning language from perceptual context: A challenge problem for AI. In *Proceedings of Conference on Artificial Intelligence (AAAI), Fellows Symposium*, 2006. 48, 138

R. J. Mooney. Learning to connect language and perception. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 2008. 2, 34

R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare. Safe and efficient off-policy reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 29

H. Müller, P. Clough, T. Deselaers, and B. Caputo. *ImageCLEF: Experimental Evaluation in Visual Information Retrieval*. Springer, 2012. 58, 125

V. K. Nagaraja, V. I. Morariu, and L. S. Davis. Modeling context between objects for referring expression understanding. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 61, 127, 130

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of International Conference on Machine Learning (ICML)*, 2010. 14

A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of International Conference on Machine Learning (ICML)*, 1999. 143, 146

J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2011. 50

D. T. Nguyen, S. Sharma, H. Schulz, and L. E. Asri. From film to video: Multi-turn question answering with multi-modal context. *Dialog System Technology Challenges at AAAI*, 2019. 152

H. Noh, P. Hongsuck Seo, and B. Han. Image question answering using convolutional neural network with dynamic parameter prediction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 58

B. O'Donoghue, I. Osband, R. Munos, and V. Mnih. The uncertainty bellman equation and exploration. *arXiv preprint arXiv:1709.05380*, 2017. 29

J. Oh, S. Singh, H. Lee, and P. Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2017. 153

M. Olazaran. A sociological study of the official history of the perceptrons controversy. *Social Studies of Science*, 26(3): 611–659, 1996. 33

V. Ordonez, G. Kulkarni, and T. L. Berg. Im2text: Describing images using 1 million captioned photographs. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2011. 17, 40, 58

B. Oreshkin, P. Rodríguez López, and A. Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 118

I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 29, 153

G. Ostrovski, M. G. Bellemare, A. v. d. Oord, and R. Munos. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*, 2017. 29

A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman. Visually indicated sounds. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 51

B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005. 22

M. Pantic and L. J. M. Rothkrantz. Toward an affect-sensitive multimodal human-computer interaction. *Proceedings of the IEEE*, 91(9):1370–1390, 2003. 40, 50

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002. 43, 58

N. Parde, A. Hair, M. Papakostas, K. Tsiakas, M. Dagioglou, V. Karkaletsis, and R. D. Nielsen. Grounding the meaning of words through vision and interactive gameplay. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2015. 72

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of International Conference on Machine Learning (ICML)*, 2013. 19, 24, 25

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *AutoDiff Workshop at NeurIPS*, 2017. 17

R. Paulus, C. Xiong, and R. Socher. A deep reinforced model for abstractive summarization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018. 45, 149

J. Pennington, R. Socher, and C. Manning. Glove: Global Vectors for Word Representation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. 43

E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 2018. II

J. Pérolat, F. Strub, B. Piot, and O. Pietquin. Learning nash equilibrium for general-sum markov games from batch data. In *Proceedings of Artificial Intelligence and statistics (AIStats)*, 2016. 154

J. Pérolat, B. Piot, and O. Pietquin. Actor-critic fictitious play in simultaneous move multistage games. In *Proceedings of Artificial Intelligence and statistics (AIStats)*, 2018. 154

E. D. Petajan. *Automatic lipreading to enhance speech recognition (speech reading)*. PhD thesis, University of Illinois at Urbana-Champaign, 1984. 50

J. Peters and S. Schaal. Policy gradient methods for robotics. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems,(RSJ)*, 2006. 31, 32

M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Long Papers)*, 2018. 43

V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. Dropout improves recurrent neural networks for handwriting recognition. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014. 21

O. Pietquin and T. Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):589–599, 2006. 72

O. Pietquin and H. Hastie. A survey on metrics for the evaluation of user simulations. *The knowledge engineering review*, 28(1):59–73, 2013. 72, 138

O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3):7, 2011. 138, 154

M. Pirotta, R. Fruit, M. Seurin, and F. Strub. European workshop of reinforcement learning (ewrl). In *Workshop Schedule Highlights*, 2018.

B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 58

G. Potamianos, C. Neti, G. Gravier, A. Garg, and A. W. Senior. Recent advances in the automatic recognition of audiovisual speech. *Proceedings of the IEEE*, 91(9):1306–1326, 2003. 50, 51

H. Prol, V. Dumoulin, and L. Herranz. Cross-modulation networks for few-shot learning. In *MetaLearn Workshop at NeurIPS*, 2018. 118

F. Pulvermüller and L. Fadiga. Active perception: sensorimotor circuits as a cortical basis for language. *Nature reviews neuroscience*, 11(5):351, 2010. 48

M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. 25, 26, 27, 28, 142

Y. Qi, Q. Wu, P. Anderson, M. Liu, C. Shen, and A. v. d. Hengel. Rerere: Remote embodied referring expressions in real indoor environments. *arXiv preprint arXiv:1904.10151*, 2019. 153

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 50

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners, 2019. 1, 46

R. Raina, A. Madhavan, and A. Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of International Conference on Machine Learning (ICML)*, 2009. 16

M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2016. 45, 149, 153

S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017. 57, 116

S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 154

S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018. 24

J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 37, 39

R. Rehurek and P. Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer, 2010. 88

M. Ren, R. Kiros, and R. Zemel. Exploring models and data for image question answering. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2015a. 94

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2015b. 37, 56

S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 59

M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the International Conference on Knowledge Discovery and Data mining (SIGKDD)*, 2016. 1, 38

M. Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2005. 30, 50

F. Ringeval, B. Schuller, M. Valstar, R. Cowie, H. Kaya, M. Schmitt, S. Amiriparian, N. Cummins, D. Lalanne, A. Michaud, et al. Avec 2018 workshop and challenge: Bipolar disorder and cross-cultural affect recognition. In *Proceedings of Audio/Visual Emotion Challenge and Workshop*. ACM, 2018. 50

J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965. 12

A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele. Grounding of textual phrases in images by reconstruction. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 62, 130

A. Rohrbach, A. Torabi, M. Rohrbach, N. Tandon, C. Pal, H. Larochelle, A. Courville, and B. Schiele. Movie description. *International Journal of Computer Vision (IJCV)*, 123(1):94–120, 2017. 62

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. 33

D. Roy. Learning visually grounded words and syntax for a scene description task. *Computer speech & language*, 16(3): 353–385, 2002. 34

S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 24

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ. San Diego La Jolla Inst. for Cognitive Science, 1985. 15, 18

C. Rupprecht, I. Laina, N. Navab, G. D. Hager, and F. Tombari. Guide me: Interacting with deep networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 39, 118

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1, 17, 22, 34, 38, 39, 68, 109, 126

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 3rd edition, 2009. 12

R. Sanabria, O. Caglayan, S. Palaskar, D. Elliott, L. Barrault, L. Specia, and F. Metze. How2: A large-scale dataset for multimodal language understanding. In *Visually-Grounded Interaction and Language workshop (ViGIL) at NeurIPS*, 2018. 62

A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 56, 60, 61, 106, 109, 110, 111

S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How does batch normalization help optimization? In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 22

M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017. 48, 51, 62, 153

J. Schatzmann, K. Georgila, and S. Young. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proceedings of SIGdial Meeting on Discourse and Dialogue (SIGDIAL), Workshop Track*, 2005. 72

J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(2):97–126, 2006. 72, 138, 149

T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *Proceedings of International Conference on Machine Learning (ICML)*, 2015a. 50

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015b. 29

D. Scherer, A. Müller, and S. Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2010. 18

J. Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Master's thesis, Technische Universitat Munchen, Germany, 1987. 57, 116

J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the International Conference on Simulation of Adaptive Behavior on From Animals to Animats*, 1990. 29

J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992. 57

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, 2015. 32, 153

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 32, 153

H. Schulz, J. Zumer, L. El Asri, and S. Sharma. A frame tracking model for memory-enhanced dialogue systems. In *Proceedings of Workshop on Representation Learning for NLP*, 2017. 72

H. Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007. 42

J. R. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417–424, 1980. 47

J. R. Searle. *Minds, brains and science*. Harvard University Press, 1984. 47

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 38

P. H. Seo, A. Lehrmann, B. Han, and L. Sigal. Visual reference resolution using attention memory for visual dialog. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 58, 82

I. Serban, R. Lowe, L. Charlin, and J. Pineau. Generative Deep Neural Networks for Dialogue: A Short Review. *Learning Methods for Dialogue Workshops at NeurIPS*, 2016. 46, 79, 138

I. V. Serban, R. Lowe, P. Henderson, L. Charlin, and J. Pineau. A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue & Discourse*, 9(1):1–49, 2018. 68, 72

S. Sharma, D. Suhubdy, V. Michalski, S. E. Kahou, and Y. Bengio. Chatpainter: Improving text to image generation using dialogue. In *Proceedings of International Conference on Learning Representations (ICLR), Workshop Track*, 2018. 82

R. Shekhar, S. Pezzelle, Y. Klimovich, A. Herbelot, M. Nabi, E. Sangineto, and R. Bernardi. Foil it! find one mismatch between image and language caption. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2017. 62

R. Shekhar, T. Baumgärtner, A. Venkatesh, E. Bruni, R. Bernardi, and R. Fernández. Ask no more: Deciding when to guess in referential visual dialogue. In *Proceedings of International Conference on Computational Linguistics (CICLing)*, 2018. 78, 81, 146

R. Shekhar, A. Venkatesh, T. Baumgärtner, E. Bruni, B. Plank, R. Bernardi, and R. Fernández. Beyond task success: A closer look at jointly learning to see, ask, and GuessWhat. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019. 150

K. J. Shih, S. Singh, and D. Hoiem. Where to look: Focus regions for visual question answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 60

H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000. 21

Y. Shoham, R. Powers, and T. Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, 2007. 154

R. Shrestha, K. Kafle, and C. Kanan. Answer them all! toward universal visual question answering models. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 57, 60, 61, 131

D. Silver, R. S. Sutton, and M. Müller. Sample-based learning and search with permanent and transient memories. In *Proceedings of International Conference on Machine Learning (ICML)*, 2008. 28

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of International Conference on Machine Learning (ICML)*, 2014. 32

D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 30, 143

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017a. 30

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017b. 16, 30

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015. 35, 36, 46

S. P. Singh, M. J. Kearns, D. J. Litman, and M. A. Walker. Reinforcement learning for spoken dialogue systems. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 1999. 68, 72, 138

B. Skinner. The behavior of organisms: an experimental analysis. *Journal of General Psychology*, 1938. 25

L. B. Smith. Learning to recognize objects. *Psychological Science*, 14(3):244–250, 2003. 48

C. G. M. Snoek, M. Worring, and A. W. M. Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the ACM International Conference on Multimedia*, 2005. 51

A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*, 2015. 2, 19, 46, 72, 138

N. Srivastava and R. R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2012. 51

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 21

A. L. Strehl and M. L. Littman. An of model-based interval estimation for markov decision processes. *J. Comput. Syst. Sci.*, 74(8):1309–1331, 2008. 29

F. Strub, R. Gaudel, and J. Mary. Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems at RecSys*, 2016.

F. Strub, H. de Vries, A. Das, S. Kottur, S. Lee, M. Malinowski, O. Pietquin, D. Parikh, D. Batra, A. C. Courville, et al. Visually grounded interaction and language workshop (vigil). In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS), Workshop Schedule Highlights*, 2017a. 2, 82

F. Strub, H. De Vries, J. Mary, B. Piot, A. Courville, and O. Pietquin. End-to-end optimization of goal-driven and visually grounded dialogue systems. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2017b. 81, 153, II

F. Strub, H. de Vries, E. Wijmans, S. Datta, E. Perez, S. Lee, P. Anderson, M. Malinowski, D. Batra, A. Courville, O. Pietquin, D. Parikh, J. Mary, C. Hori, A. Cherian, and T. Marks. Visually grounded interaction and language workshop (vigil). In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS), Workshop Schedule Highlights*, 2018a. 82

F. Strub, M. Seurin, E. Perez, H. De Vries, J. Mary, P. Preux, O. Pietquin, and A. Courville. Visual reasoning with multi-hop feature modulation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018b. 111, II

J. Suarez, J. Johnson, and F.-F. Li. Ddrprog: A clevr differentiable dynamic reasoning programmer. *arXiv preprint arXiv:1803.11361*, 2018. 60

A. Suhr, M. Lewis, J. Yeh, and Y. Artzi. A corpus of natural language for visual reasoning. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL), Short Paper*, 2017. 62, 131

A. Suhr, S. Zhou, I. Zhang, H. Bai, and Y. Artzi. A corpus for reasoning about natural language grounded in photographs. In *Visually-Grounded Interaction and Language workshop (ViGIL) at NeurIPS*, 2018. 61, 62

S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end memory networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 46, 131, 152

M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *Proceedings of Conference of the International Speech Communication Association (ISCA)*, 2012. 19

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 16, 20, 44, 45, 58

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988. 27

R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, 1991. 28

R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. 25, 26, 27, 28, 30

R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2000. 31, 45

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 22, 35, 36

K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015. 20

M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 35

M. Tapaswi, M. Bäuml, and R. Stiefelhagen. Aligning plot synopses to videos for story-based retrieval. *International Journal of Multimedia Information Retrieval*, 4(1):3–16, 2015. 51

S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 2011. 40

J. B. Tenenbaum and W. T. Freeman. Separating style and content. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 1997. 54

D. Teney, P. Anderson, X. He, and A. van den Hengel. Tips and tricks for visual question answering: Learnings from the 2017 challenge. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 60

G. Tesauro. Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68, 1995. 16, 30

G. Thierry, P. Athanasopoulos, A. Wiggett, B. Dering, and J.-R. Kuipers. Unconscious effects of language-specific terminology on preattentive color perception. *Proceedings of the National Academy of Sciences*, 106(11):4567–4570, 2009. 95

J. Thomason, J. Sinapov, M. Svetlik, P. Stone, and R. J. Mooney. Learning multi-modal grounded linguistic semantics by playing" i spy". In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2016. 72

E. L. Thorndike. Animal intelligence: An experimental study of the associative processes in animals. *The Psychological Review: Monograph Supplements*, 2(4):i, 1898. 25

S. Thrun and A. Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the Connectionist Models Summer School*, 1993. 31, 143

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996. 21

T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. 24, 35

J. P. Tremblay, I. Balafrej, F. Labelle, F. Martel-Denis, É. Matte, J. Chouinard-Beaupré, A. Létourneau, A. Mercier-Nicol, S. Brodeur, F. Ferland, et al. A cooperative visually grounded dialogue game with a humanoid robot. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS), demonstration track*, 2018. 72, 81

J. N. Tsitsiklis and B. Van Roy. Analysis of temporal-diffference learning with function approximation. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 1997. 30

A. M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, 1950. 33, 138

A. Utsumi. A distributional semantic model of visually indirect grounding for abstract words. In *Visually-Grounded Interaction and Language workshop (ViGIL) at NeurIPS*, 2018. 62

A. Van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016a. 16, 50, 51

A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016b. 50, 53

L. M. van G. der and Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research (JMLR)*, 9(Nov): 2579–2605, 2008. 100

H. Van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil. Deep reinforcement learning and the deadly triad. In *Deep Reinforcement Learning Workshop at NeurIPS*, 2018. 29, 30

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017. 46, 124, 131, 152

R. Vedantam, C. Lawrence Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 58

P. Vincent-Lamarre, A. B. Massé, M. Lopes, M. Lord, O. Marcotte, and S. Harnad. The latent structure of dictionaries. *Topics in cognitive science*, 8(3):625–659, 2016. 47

O. Vinyals and Q. Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015. 68, 138

O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2015a. 45

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015b. 16, 34, 51, 58

O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, T. Ewalds, D. Horgan, M. Kroiss, I. Danihelka, J. Agapiou, J. Oh, V. Dalibard, D. Choi, L. Sifre, Y. Sulsky, S. Vezhnevets, J. Molloy, T. Cai, D. Budden, T. Paine, C. Gulcehre, Z. Wang, T. Pfaff, T. Pohlen, D. Yogatama, J. Cohen, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, C. Apps, D. Kavukcuoglu, Korayand Hassabis, and D. Silver. AphaStar. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/, 2019. Accessed: 2019-01. 30, 50, 52, 118, 143

D. Vlasic, M. Brand, H. Pfister, and J. Popović. Face transfer with multilinear models. In *ACM transactions on graphics (TOG)*, volume 24, pages 426–433. ACM, 2005. 54

A. Vogel, K. Raghunathan, and D. Jurafsky. Eye spy: Improving vision through dialog. In *Proceedings of Conference on Artificial Intelligence (AAAI), Symposium Track*, 2010. 72

L. Von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of of the SIGCHI Conference on Human Factors in Computing Systems*, 2004. 72

L. Von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. In *Proceedings of of the SIGCHI Conference on Human Factors in Computing Systems*, 2006. 72

A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. In *Readings in Speech Recognition*, pages 393–404. Elsevier, 1990. 18

M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella. Paradise: A framework for evaluating spoken dialogue agents. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 1997. 138

C. Wang, H. Yang, C. Bartz, and C. Meinel. Image captioning with deep bidirectional lstms. In *Proceedings of the ACM International Conference on Multimedia*, 2016a. 58

P. Wang, Q. Wu, C. Shen, A. Dick, and A. van den Hengel. Fvqa: Fact-based visual question answering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(10):2413–2427, 2018. 59

Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016b. 30

C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992. 28

N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 109

S. R. Waxman and A. E. Booth. Seeing pink elephants: Fourteen-month-olds' interpretations of novel nouns and adjectives. *Cognitive Psychology*, 43(3):217–242, 2001. 39, 48

S. R. Waxman and D. B. Markow. Words as invitations to form categories: Evidence from 12-to 13-month-old infants. *Cognitive Psychology*, 29(3):257–302, 1995. 39, 48

W. Wei, Q. Le, A. Dai, and J. Li. Airdialogue: An environment for goal-oriented dialogue research. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. 72

J. Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, 1966. ISSN 0001-0782. 40, 41, 138

T.-H. Wen, D. Vandyke, N. Mrksic, M. Gasic, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, and S. Young. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*, 2016. 68, 69, 72

J. Weston, S. Chopra, and B. Antoine. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014. 46, 131

J. Weston, A. Bordes, S. Chopra, A. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2016. 68, 72, 153

B. L. Whorf, J. B. Carroll, S. C. Levinson, and P. Lee. *Language, thought, and reality*. The MIT Press Cambridge, Mass, 1956. 39

E. Wiewiora. Potential-based shaping and q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003. 29

E. Wijmans, S. Datta, O. Maksymets, A. Das, G. Gkioxari, S. Lee, I. Essa, D. Parikh, and D. Batra. Embodied question answering in photorealistic environments with point cloud perception. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 153

J. Williams and S. Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007. 139, 153

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 28, 31, 45, 143

R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989. 44

M. Wilson. Six views of embodied cognition. *Psychonomic bulletin & review*, 9(4):625–636, 2002. 48

R. A. Wilson and L. Foglia. Embodied cognition. *Stanford Encyclopedia of Philosophy*, 2011. 1, 48

T. Winograd. Procedures as a representation of data in a computer program for understanding natural language. Technical report, PhD thesis, Massachusets Institute of Technology, January 1971. 1, 40, 105

C. Wu, J. Liu, X. Wang, and X. Dong. Chain of reasoning for visual question answering. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018a. 60

Q. Wu, D. Teney, P. Wang, C. Shen, A. Dick, and A. van den Hengel. Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding*, 163:21–40, 2017. 2

Q. Wu, P. Wang, C. Shen, I. Reid, and A. van den Hengel. Are you talking to me? reasoned visual dialog generation through adversarial learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b. 56, 82

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. 41, 46

Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian. Building generalizable agents with a realistic and rich 3d environment. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2018c. 48

F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 153

S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 35

S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 20

C. Xiong, S. Merity, and R. Socher. Dynamic memory networks for visual and textual question answering. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016. 56, 131

J. Xu, T. Mei, T. Yao, and Y. Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 62

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of International Conference on Machine Learning (ICML)*, 2015. 51, 55, 56, 58, 94, 152

L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos. Efficient video object segmentation via network modulation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 118

S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha. Like like alike: joint friendship and interest propagation in social networks. In *Proceedings of International Conference on World Wide Web (WWW)*, 2011. 54

T. Yang, Z.-J. Zha, and H. Zhang. Making history matter: Gold-critic sequence training for visual dialog. *arXiv preprint arXiv:1902.09326*, 2019. 82

Z. Yang, X. He, J. Gao, L. Deng, and A. Smola. Stacked attention networks for image question answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 55, 56, 59, 106, 110, 111

Y. Yao, J. Xu, F. Wang, and B. Xu. Cascaded mutual modulation for visual reasoning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. 61, 110, 111

K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 60

J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 37

P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014. 58

S. Young, M. Gašić, B. Thomson, and J. D. Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013. 72

T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018. 42

H. Yu, X. Lian, H. Zhang, and W. Xu. Guided feature transformation (gft): A neural language grounding module for embodied agents. In *Proceeding if the Conference on Robot Learning (CoRL)*, 2018a. 153

L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg. Modeling context in referring expressions. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016a. 61, 62, 71, 77, 125, 127, 130

L. Yu, H. Tan, M. Bansal, and T. L. Berg. A joint speakerlistener-reinforcer model for referring expressions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016b. 62, 127, 130

L. Yu, Z. Lin, X. Shen, J. Yang, X. Lu, M. Bansal, and T. L. Berg. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b. 57, 61, 62, 127, 130

Z. Yu, J. Yu, C. Xiang, J. Fan, and D. Tao. Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):5947–5959, 2018c. 54, 56, 60

T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 31, 143, 149

V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart, M. Shanahan, V. Langston, R. Pascanu, M. Botvinick, O. Vinyals, and P. Battaglia. Deep reinforcement learning with relational inductive biases. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019. 56

W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015. 21

M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 24

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2014. 18

Z. Zeng, M. Pantic, G. I. Roisman, and T. S. Huang. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1):39–58, 2009. 40, 50

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017. 20

J. Zhang, Q. Wu, C. Shen, J. Zhang, J. Lu, and A. van den Hengel. Goal-oriented visual question generation via intermediate rewards. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2018a. 81, 146

J. Zhang, T. Zhao, and Z. Yu. Multimodal hierarchical reinforcement learning policy for task-oriented visual dialog. In *Proceedings of SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*, 2018b. 82

P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh. Yin and yang: Balancing and answering binary visual questions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016a. 59, 60

X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 18

Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. Laurent, Y. Bengio, and A. Courville. Towards end-to-end speech recognition with deep convolutional neural networks. In *Proceedings of European Conference on Speech Communication and Technologies (Interspeech)*, 2016b. 18

R. Zhao and V. Tresp. Efficient visual dialog policy learning via positive memory retention. In *Visually-Grounded Interaction and Language workshop (ViGIL) at NeurIPS*, 2018a. 81

R. Zhao and V. Tresp. Learning goal-oriented visual dialog via tempered policy gradient. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*, 2018b. 81

B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 52, 68

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015. 18

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017a. 38

Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei. Visual7w: Grounded question answering in images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 59

Y. Zhu, S. Zhang, and D. Metaxas. Interactive reinforcement learning for object grounding via self-talking. In *Visually-Grounded Interaction and Language workshop (ViGIL) at NeurIPS*, 2017b. 81, 130, 149

B. Zhuang, Q. Wu, C. Shen, I. Reid, and A. van den Hengel. Parallel attention: A unified framework for visual object discovery through dialogs and queries. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 56, 62, 81, 124, 127, 128, 129, 130

M. Zinkevich, A. Greenwald, and M. L. Littman. Cyclic equilibria in markov games. In *Advances in Neural Information Processing Systems*, pages 1641–1648, 2006. 154

C. L. Zitnick and D. Parikh. Bringing semantics into focus using visual abstraction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3009–3016, 2013. 40

Appendix A

Contributions

As this thesis include joint works, we here provide some estimate to point out my involvement in the research papers justly. The underlying goal is to not claim the work of our colleagues by better highlighting the workload. To do so, we first enumerate some research contributions aspects and roughly score my involvement over them. Again, those are very ad-hoc and subjective scores, and they should be taken with caution. The other co-authors have also validated those estimates. In the end, we come up with the following contribution facets:

- *Brainstorming:* it defines the impact over the initial reflection step before initiating the research work
- *Codebase:* it defines the amount of code that has been implemented to run the baselines and complementary experiments.
- *Experiments:* it defines the time spend on running experiments, tuning the hyperparameters, etc.
- *Analysis:* it defines the amount of analysis, experience proposals that have been done, and how this reflection has been maintained in final work.
- *Writing:* it defines the amount of text that has been written in the published paper.

In rare cases, complementary aspects have been added to provide a better overview of the workload.

Each contribution facet is scored between 1 and 5, where 1 stands for minor involvement, and 5 stands for being the core contributor. When several authors unarguably shared the contributions, the maximum score is automatically clipped to be 4. As a piece of intuition, the scores may be roughly described as follows:

5. *Sole contributor:* run all the experiments, write most of the paper, write most of the codebase, initiate the research direction, etc.

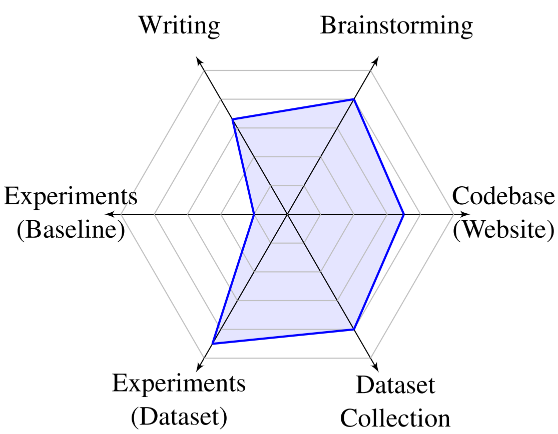
4. *Principal and joint contributor:*. run a large set of experiments, write several sections, write large portions of code, co-initiate the research directions, etc.

3. *Active and decisive contributions:* code and/or run complementary experiments present in the paper, write at least one section, impact the research directions, etc.

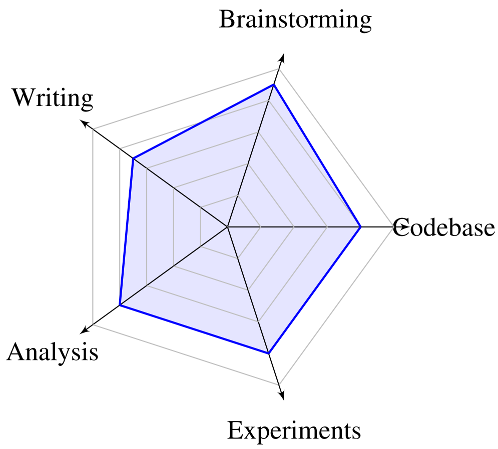
2. *Active but non-decisive contributions:* run complementary experiments not present in the paper, reformulate paper sections, refactor some code, alter some research conclusions, etc.

1. *Minor contributions and reviews:* code review, fix paper typos, provide research remarks, etc.

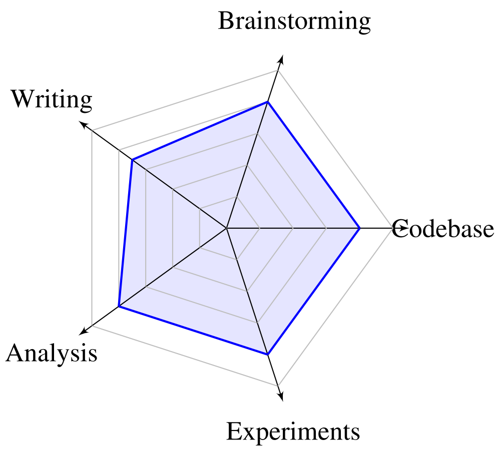
Note that the scores does not sum to 5 if we combine all authors' contributions! These scores reflect the degree of involvement, and many authors can be equally involved in one contribution!



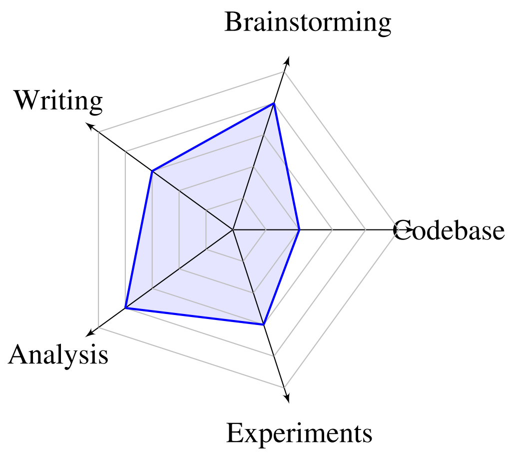
GuessWhat?! Visual object discovery through multimodal dialogue (de Vries, Strub, Chandar, Pietquin, Larochelle, and Courville, 2017)



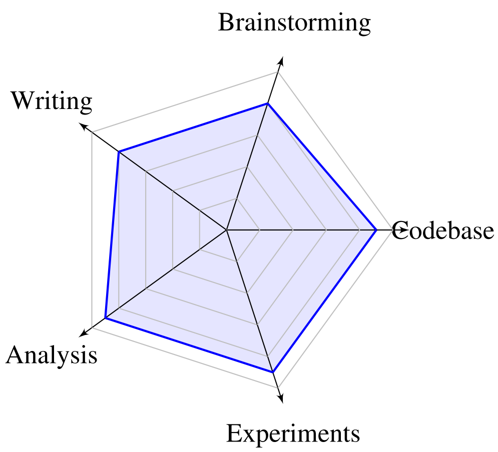
End-to-end optimization of goal-driven and visually grounded dialogue systems. (Strub, De Vries, Mary, Piot, Courville, and Pietquin, 2017b)



Modulating early visual processing by language. (De Vries*, Strub*, Mary, Larochelle, Pietquin, and Courville, 2017)



FiLM: Visual reasoning with a general conditioning layer. (Perez, Strub, De Vries, Dumoulin, and Courville, 2018)



Visual reasoning with multi-hop feature modulation. (Strub, Seurin, Perez, De Vries, Mary, Preux, Pietquin, and Courville, 2018b)

Appendix B

List of Acronyms

A3C Asynchronous Actor-Critic Agents. 32

Adam Adaptive Moment estimation. 24, 77, 109, 127

AI Artificial Intelligence. 1, 33, 34, 40, 50

ASVR Audio-Visual Speech Recognition. 50, 51

Bagging Bootstrap aggregating. 21

BFGS Broyden-Fletcher-Goldfarb-Shanno. 14

Bi-RNN Bidirectional Recurrent Networks. 20

BLEU BiLingual Evaluation Understudy. 43, 45, 58, 149

BN Batch Normalization. 21–23, 95–97, 99, 126

BSearch Beam Search. 45

CBN Conditional Bacth Normalization. 94, 95, 97, 99–101, 114, 121, 152

CE Cross-Entropy. 12

CMM Cascaded Mutual Modulation. 111

CNN Convolution Neural Network. 18, 20–23, 34, 52, 58, 107–109, 111, 119

CPU Central Processing Unit. 17

CV Computer Vision. 33, 34, 36, 40, 48

DL Deep Learning. 16

DP Dynamic Programming. 26–28

DPG Deep Deterministic Policy Gradients. 32

DQN Deep Q-Networks. 30, 151

POS Tagger Part-Of-Speech Tagger. 42

PPO Proximal Policy Optimization. 32

ReLU Rectified Linear Unit. 14, 23, 35, 37, 55, 97, 108, 109, 126, 141

ResBlock Residual Block. 37, 109, 111, 113, 115, 123, 126, 152

ResNet Residual Neural Network. 35–37, 46, 60, 94–101, 103, 106, 126, 152

RGB Red, Green and Blue. 17

RL Reinforcement Learning. 25–27, 30, 81, 82, 138, 139, 142–149, 153, 154

RNN Recurrent Neural Network. 18–23, 25, 44–46, 52, 54, 56–58, 60, 62, 79, 94, 106, 114, 119, 121–123, 126, 141

SAC Soft-Actor Critic. 32

SAN Stacked Attention Networks. 55, 111

Seq2Seq Sequence-to-Sequence. 44–46, 58

SGD Stochastic Gradient Descent. 14–16, 21, 23, 24, 145

TRPO Trust Region Policy Optimization. 32

t-SNE t-Distributed Stochastic Neighbor Embedding. 100, 102, 103, 112, 113

VGGNet Visual Geometry Group Network. 35, 46, 60, 72, 77–80, 98, 140

VQA Visual Question Answering. 51, 56, 58–62, 71, 73, 81, 82, 94, 95, 97–100, 103

WMT Workshop on Statistical Machine Translation. 43

Appendix C

List of symbols

a is the answer of some arbitrary question. In GuessWhat?!, an answer is either yes, no or not-applicable.. 76, 125

C is the number of channel in a feature map F. 17, 140

$\mathcal{D}$ is a GuessWhat?! dialogue composed of J question-answer pairs $(\boldsymbol{q}, \boldsymbol{a})_{j\leq}$. 76, 125, 140, 142

F is a 3-dimensional tensor that encodes some feature map of dimensions $W \times H \times C$.. 37

γ is the discount parameter in a MDP. 25, 142, 143

H is the height of an Image $\mathcal{I}$ or feature map F where each vertical spatial location is indexed by h. 17, 125, 140

I is the number of words in a sentence where each word is indexed by i. In a dialogue $\mathcal{D}$, I_j can be conditioned on the current question J. 140

$\mathcal{I}$ is an arbitrary RGB image of dimension $H \times W \times 3$. 17, 76–78, 98, 125, 140–142

J is the number of questions in a dialogue $\mathcal{D}$ where each question is indexed by j. 76, 78, 140

K is the number of objects in a GuessWhat?! game where each object is indexed by k. 76, 125, 140

$\mathcal{K}$ is the number of resblock or layers in stack of neural modules where each layer/resblock is indexed by κ. 123

$\mathcal{S}$ is segmented mask for object.. 76, 125, 140

O is a list of objects $O = [o_1, \ldots, o_K]$ within an image $\mathcal{I}$. Each object can be defined by some features such as his category, his spatial location etc. 76, 78, 125, 140, 142

$\boldsymbol{q}$ is a question. A question is composed of I words $\boldsymbol{q} = [w_0, \ldots, w_i, \ldots, w_I]$ where each word comes from a predefined vocabulary V. 76

$\mathcal{V}$ is some vocabulary. It consists of finite set of tokens or words.. 76, 140, 142

W is the width of an Image $\mathcal{I}$ or feature map $\boldsymbol{F}$ where each horizontal spatial location is indexed by w. 17, 125, 140