Université de Lille

Thèse de doctorat

# Deep learning for population genetics: inferring demography and targets of natural selection using convolutional neural networks

## Apprentissage profond pour la génétique des populations : inférer la démographie et les cibles de sélection naturelle à l'aide de réseaux de neurones convolutifs

Thèse de doctorat de l'Université de Lille préparée à l'Unitée EvoEcoPaléo (EEP)

École doctorale n°104 Science de la Matière, du Rayonnement et de l'Environnement (SMRE)

Thèse présentée et soutenue à Villeneuve d'Ascq, le 13 Décembre 2024, par

**GUILLAUME LAN-FONG**

pour obtenir le grade de Docteur en
**Biologie de l'environnement, des organismes, des populations, écologie**

Composition du Jury :

| | |
|---|---|
| Nataliya Sokolovska<br>Professeur, Sorbonne-Université | Rapportrice |
| Frédéric Austerlitz<br>Directeur de Recherche CNRS, Université Paris Cité | Rapporteur *(Président)* |
| Flora Jay<br>Chargée de Recherche CNRS, Université Paris Saclay | Examinatrice |
| Raphaël Leblois<br>Chargé de Recherche CNRS, INRAE (CBGP) | Examinateur |
| Xavier Vekemans<br>Professeur, Uniersité de Lille | Directeur de thèse |
| Camille Roux<br>Chargé de Recherche CNRS, Université de Lille | Co-encadrant de thèse |

# Contents

# Abstract

Inferring demographic history and detecting natural selection are major challenges in population genetics. Traditional methods, though effective, often rely on unrealistic assumptions, such as the absence of selection or of simultaneous demographic changes. However, in many natural populations, selection and demography affect genetic diversity at the same time. Their interaction is making evolutionary inferences more complex and is a significant obstacle. In this thesis, we explore the use of convolutional neural networks (CNNs), a deep learning technique, to overcome the limitations of traditional methods. This work focuses on two main tasks: (1) classification of genomic data based on demographic history, and (2) detecting and localizing targets of natural selection along genomes. The aim is not to develop a ready-to-use tool but to understand the necessary considerations for training CNNs on these tasks, highlighting the challenges and nuances of developing deep learning-based methods for population genetics. We compare CNN performance on simulated data with established population genetics methods, such as Approximate Bayesian Computation Random Forest (ABC-RF) for classification and SweepFinder2 for detecting selection. CNNs can achieve results comparable to or better than these existing methods, especially in localizing selection signals. Overall, our results highlight the potential of deep-learning approaches as powerful tools for population genetics, as long as careful consideration is given to input data representation. Combining CNNs with other methods may further improve the accuracy and reliability of demographic inferences and selection detection in population genetics.

# Introduction

Genetic diversity within living organisms has led to their classification within reproductively isolated groups called species. Within a species, genetic information at a given locus can either be fixed and shared among all individuals (monomorphic locus) or rather vary among individuals (polymorphic locus). Evolutionary research, and specifically population genetics, aims at defining a theoretical framework to explain the distribution of genetic diversity among loci, individuals and species, and to understand evolutionary forces shaping it. Thus, one of the main approaches in evolutionary research is to use molecular genetic data in order to make inferences about the natural and evolutionary history of populations. However, numerous forces shape the observed variation in genetic diversity and determining which are responsible for the observed patterns can be difficult when their effects are similar and confounding. This complexity complicates efforts to accurately reconstruct the evolutionary history of populations and to identify the specific genetic changes driven by adaptive forces. To address these challenges, various statistical and computational methods have been developed over the past decades and, recently, deep learning has also emerged as a powerful tool for population genetics, offering new possibilities and showing promise in improving the accuracy of evolutionary inferences.

## Genomic variation of diversity

For the past century, the study of genetic diversity has been one of the main goals of evolutionary biology. At its core, genetic diversity is a measure of the genetic variability within a population. Indeed, even within the same species, each individual will carry a specific DNA sequence and thus, specific and distinct genetic information.

As early as 1859, in "*On the Origin of Species by Means of Natural Selection*", Darwin proposed a first insight in genetic diversity, making the hypothesis that variation within species could be linked to variation between species through means of natural selection, isolation and thus, evolution. Together with Gregor Mendel's foundational work on heredity in 1865 (Mendel, 1865), these insights laid the first key concepts about genetic variation and heritability. A few decades later, during the mid-30s, new studies of evolutionary biology about genetic diversity by Ronald A. Fisher, J. B. S. Haldane and Sewall Wright (Fisher 1930; Haldane 1932; Wright 1968) were the foundations of the theory of population genetics upon which the modern evolutionary synthesis was then built by integrating both Mendelian genetics and Darwinian principles. From there stems one of the most commonly used models of population genetics: the Wright-Fisher model. It describes a simplified population with several key assumptions: no mutation, no selection, no migration, non-overlapping generations and random mating. Essentially, each generation consists of a fixed number of individuals who randomly contribute to the next generation, without considering any other evolutionary forces than genetic drift. Despite its simplicity, the Wright-Fisher model serves as a valuable tool for understanding how genetic drift operates in finite populations. Although real populations rarely adhere strictly to these assumptions, the Wright-Fisher model provides a baseline for studying how deviations from these conditions—such as incorporating mutation, selection, migration, or overlapping generations—can affect evolutionary dynamics. Indeed, within this new theory, changes in the phenotypes displayed by a given population through time are explained by the variability of genes and genome sequences.

These changes can be seen as manifestation of the balance between the apparition and the disapparition, within the population, of genetic variants (Ellegren & Galtier, 2016). Four main evolutionary forces are usually considered to explain the variability of the genetic diversity found within natural populations:

1) Mutation, which introduces new variants (alleles) within a population. It is a source of diversity and novelty, but its impact on the variation is limited.

2) Migration, which can introduce new alleles when gene flow occurs between two structured demes. Its homogenisation of allelic frequencies between demes is faster if the migration rate is higher.

3) Genetic drift, the intensity of which depends on the effective population size Ne. It will lead to stochastic variations in frequencies over time, ultimately resulting in the loss or fixation of an allele.

4) Natural selection, whose impact on the allele frequencies will be different depending on its nature. It can either lead to the fixation of an advantageous variant in a given environment (directional selection) or eliminate new deleterious variants (purifying selection).

From those four evolutionary forces, two main mechanisms affect the evolution of genetic diversity: natural selection in itself, and demography. Natural selection in itself acts like a filter on the other three forces, by influencing which genetic variants persist and spread in a population. Indeed, natural selection impacts allele frequencies in various ways depending on its form: positive directional selection will increase the frequency of the selected allele while also affect the neutral regions linked to it, generating a “valley of diversity” around the selected loci (Smith & Haigh, 1974; Barton, 2000) as observed in *Plasmodium falciparum* in response to strong selection pressure exerted by malaria treatments (Nair et al., 2003). Balancing selection will tend to maintain a greater variety of alleles in the population over long periods, resulting in increased polymorphism at linked loci (Tian et al., 2002). One such example is found in the genes responsible for plants' self-incompatibility, such as the S-locus in *Arabidopsis halleri* and *A. lyrata* (Roux et al., 2013). As for demography, it shapes the effects of genetic drift and migration. Demographic events, for example population size changes or gene flow, also leave marks on genetic diversity (Pespeni et al., 2012).

## How to study genetic diversity?

In order to study the variation of genetic diversity in genomes, one must first quantify such diversity. The most often used definitions rely on variation of allelic states or quantitative characters and range from the more simpler ones such as the allelic diversity (based on the number of alleles per locus), the allelic richness (the average number of alleles per locus) or the heterozygosity (the average proportion of loci with two different alleles within a single individual) all the way to more elaborate ones like the nucleotide diversity π (the average number of nucleotide differences per site - Nei & Li, 1979) for example (Hughes *et al*., 2008). The latter is a prime example of one of the main approaches developed to study genetic diversity: summary statistics.

Historically, summary statistics have been proposed as an answer to the need of analytical tools to study genetic diversity, numerous have been developed through the years, such as Watterson's θ estimator (Watterson, 1975), Tajima's π and Tajima's D (Tajima, 1989). Such statistics are designed to capture specific aspects of the diversity. For example, Watterson's θ estimator can be used to estimate the genetic diversity of a population based on its effective size Ne and its mutation rate per generation μ. More precisely, it measures genetic diversity as a function of the number of polymorphic sites in a given genomic region. This measure is then compared to the genetic diversity of a population evolving under Wright & Fisher where θ is expected to be equal to $2pN_e\mu$, with $p$ the ploidy of the species, $N_e$ the effective size of the population of interest, and μ the mutation rate per base (**Figure 1**).

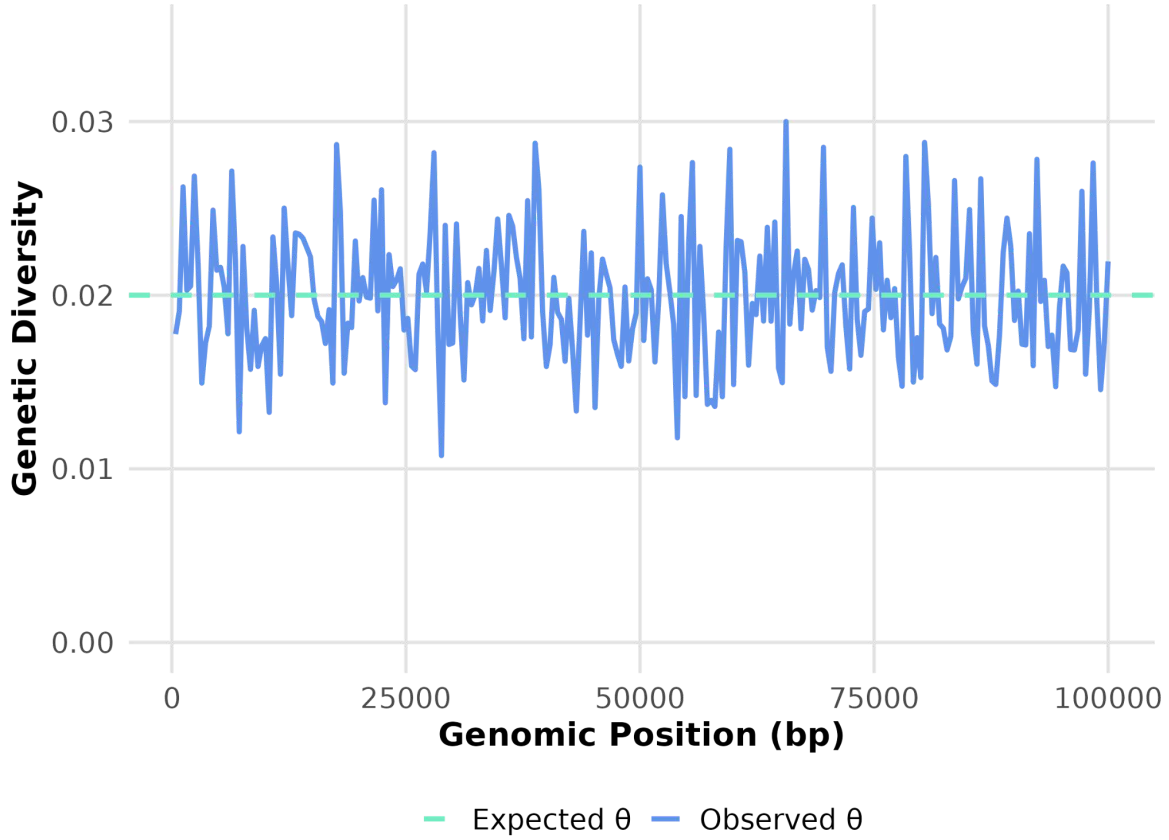



**Figure 1** - Expected and Observed genetic diversity under neutrality along a genome of 100 kbp, for a population diploid population (*p* = 2) of effective size *Ne* = 5000, with a mutation rate μ = 1e-06 and a recombination rate r = 1e-06.

The genetic diversity θ under neutrality is expected to be equal to $2pN_e\mu$ (green dashed line). However, even in absence of selection, the actual observed diversity (blue line) shows variability due to genetic drift.

The use of these statistics has largely contributed to make possible the study and analysis of genetic diversity. Such statistics are commonly used to perform such neutrality tests. These tests rely on well-established theoretical expectations for genetic variability under standard neutral models, such as the Wright-Fisher model. By comparing the observed genetic variability to these expectations, researchers can assess the degree of departure from neutrality.

Taking the example of Tajima's D (Tajima, 1989), which compares two measures of genetic diversity in a population: $n$ the average number of pairwise nucleotide differences per site between the sampled sequences and Watterson's θ which is based on the number of segregating sites. Under neutrality, both measures are expected to be roughly equal, leading to a Tajima's D value close to zero. Deviations from zero thus indicate that the population

may have experienced demographic events or selection. Thus, sudden increase of the effective size of a population for example would lead to more recent common ancestor of most lineages which will in turn lead to an excess of low-frequency variants. This excess of rare mutations lowers the value of $n$ compared to Watterson's θ causing a negative value of Tajima's D. If we consider migration rather than a population's expansion, the gene flow facilitates the exchange of alleles between demes thus enhancing the sharing of genetic diversity among the involved populations. Among the several summary statistics developed to measure this difference, many often focus on estimators of $\theta = 2pN_e\mu$, with *p* the ploidy of the sampled individuals, $N_e$ the effective size of the population and $\mu$ the mutation rate (Achaz, 2008).

However, while these estimators help determine how closely the observed data aligns with the predictions of the standard neutral model and the genomic patterns thus revealed offer insights into the evolutionary history of populations, they can often be difficult to extract. Moreover, for a long time, the high cost associated with complete sequencing constituted a major limitation to large scale accumulation and exploitation of such data for non model organisms. Fortunately, recent breakthroughs in DNA sequencing technologies (NGS) have made possible the generation of large numbers of such datasets and to an explosion in the amount of genomic data available. In turn, numerous tools and methods for interpreting observed patterns and for answering relevant evolutionary questions have been developed and many new studies have been conducted: on one hand, analyses of patterns of genomic distribution of nucleotide diversity are conducted in order to unravel and understand various mechanisms affecting genetic diversity, such as signatures of positive selection (Tajima 1989; Nair *et al*., 2003; Nielsen et al. 2005; Pavlidis et al. 2010). Other studies have been focusing on the impact of various aspects of the demographic history of natural populations ranging from population size changes (Marth et al. 2004) to patterns of gene flow and admixture (Martin et al. 2013; Corbett-Detig and Nielsen 2017; Schrider et al. 2018).

To meet the growing demand for more powerful tools capable of processing the ever-growing quantity of datasets in population genomics, new methods have emerged to answer questions in evolutionary genomics. Summary statistics are still used, but as each statistic is usually catered to highlight only one specific aspect of the information found in a genome alignment, another approach has been proposed: using large number of such summary statistics at once to maximize the amount of retrieved information (Lin et al. 2011; Schrider and Kern 2016; Sheehan and Song 2016). The focus then shifts from trying to

interpret the individual values of each statistic to trying to make sense of the overall patterns observed across the set of chosen statistics.

## The interaction of demography and selection interactions

Reconstruction of the evolutionary histories of natural populations has always been the goal of population genetics. To understand the diversification of a biological group at various levels (morphological, ecological, etc…) it is essential to take into account the historical demographic context: did the diversification take place in the presence of gene flow? In a large or small population? Over long periods of time or in a few generations? Another major question generally linked to adaptation processes is the genetic architecture associated with these processes: is this adaptation the result of the combined effect of many loci with small individual effects, or does it come from a small number of loci with large effects? All of those questions can be summarized in the two major goals of population genetics: to reconstruct the demographic history associated with a given population, and to identify the genomic targets of natural selection.

Such reconstructions, for example, have revealed evidence of Neanderthal and denisovan DNA in modern human genomes, particularly in Eurasian and Melanesian populations (Vernot et al., 2016), and the discovery of a first-generation Neanderthal-Denisovan hybrid provides compelling evidence that interbreeding was not a rare event but likely occurred frequently when these groups coexisted (Slon et al., 2018). Moreover, these geographical differences in interbreeding patterns have likely contributed to the genetic variations in modern humans, with higher Neanderthal admixture in East Asian populations and Denisovan genetic legacy persisting in Southeast Asian and Oceanian populations (Ko, 2016).These findings point to a scenario where gene flow between various hominin lineages was a regular aspect of human evolution. This is one of many examples of why the study of demographic histories plays a crucial role in understanding genetic diversity of natural populations.

Unfortunately, from there rises the major hurdle to population genetics: while it is true that demographic changes impact whole genomes, things are harder to untangled at a more local scale, where the genetic diversity is affected by the confounding effects of both demography (acting at a global scale) and selection (acting on specific targets locus and eventually their surroundings) mechanisms (Li et al., 2012).

Indeed, most neutrality tests are not robust to demographic changes. For example, a sudden increase in the effective size of a population following a Wright-Fisher model will lead to a decrease of Tajima's D values (Tajima, 1989) due to the lengthening of the branches of the coalescent, a pattern similar to the one observed in genomic regions affected by positive selection. A more detailed explanation of coalescent models is presents in the introduction of the Chapter 1 of this thesis, but to put things simply, coalescent models focus on 'coalescent times' *i.e.* the expected time for two sampled lineages to merge into one when looking at a population history backward in time. These coalescent times are tied to the population's effective size as the probability of two lineages coalescing in a given generation is approximately $1/(2.N_e)$, with $N_e$ the effective size of the population. Thus, when the population sizes increase, the probability of coalescent events to occur at a given generation decreases which results in a lengthening of the branches in the coalescent tree, as lineages remain separate for a longer period. Another example, selective sweeps and bottlenecks *i.e*. the sudden reduction of a population's effective size (Simonsen *et al.* 1995) may share similarities in the way they influence patterns of genomic diversity, but the same selective sweeps will lead to an excess of singleton reminiscent of a population undergoing an expansion (Achaz, 2008). Thus, selection negatively impacts the performance of classic demographic inferences methods, leading to biased estimates of parameters and incorrect selection of demographic models (Schrider et al. 2016). As for demography, it may also generate diversity patterns nearly indistinguishable from those of positive selection (Pavlidis et al., 2008), as explained in the following example, extracted from Koropoulis et al. (Koropoulis et al., 2020): assuming a population undergoing a bottleneck event going through three phases: (1) a population of large effective size goes through (2) a sudden decrease of population size (the bottleneck) before a restoration of the population to a large size (3). During (2), numerous coalescent events occurred in a short period of time, due to the bottleneck. However, some lineage can still take more time and only coalesce during (3). The pattern in the coalescent tree thus created is highly similar to those of a selective sweep (**Figure 2**), hardly distinguishable from one another.

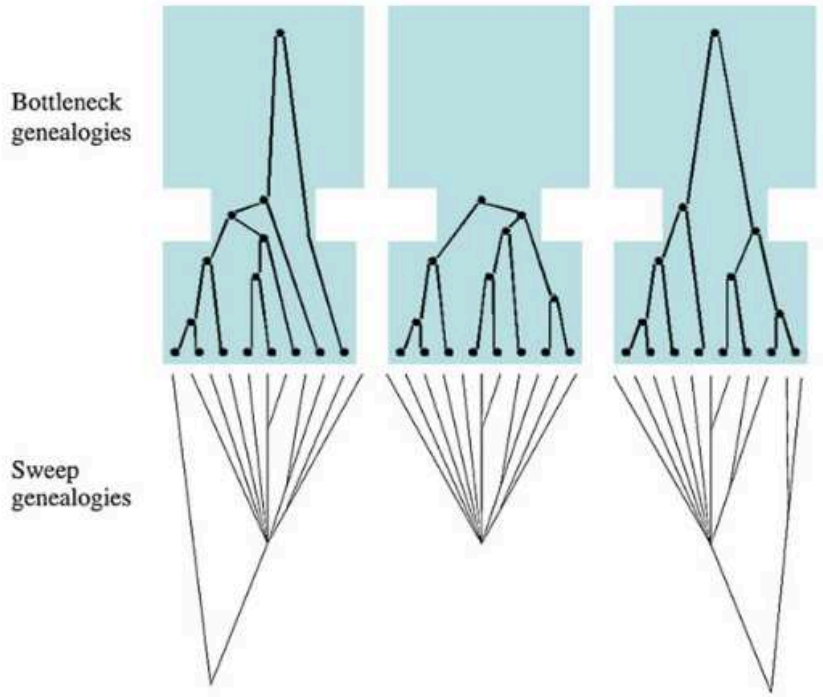



**Figure 2** - Similar tree genealogies generated from demographic events and selective sweeps.

Various bottleneck configurations can result in genealogies similar to selective sweep ones. Selective sweep genealogy presents very short coalescent trees within the region of the beneficial mutation, and trees with long internal branches as we move away from it. Bottleneck ones may present very long internal branches too as long as the ancestral population size is large. (*Figure from Koropoulis et al., 2020*)

# New promising tools for population genetics

As stated before, most historical approaches in population genetics do not account for the confounding effects and interactions between selection and demographic events. However, over the years and thanks to the advances in computational power, new methods and tools have been used more and more in population genetics, with two of them being especially suited to deal with sets of summary statistics: the approximate Bayesian computation (ABC) framework, where the idea is to approximate Bayesian inferences in cases where a proper likelihood function is unknown. This approach is applied on datasets by comparing observed sets of summary statistics with those computed on a great number of random simulations. The other method is supervised machine-learning, where a neural network is trained on a huge dataset of labeled examples before making inferences on never seen before data.

## Approximate Bayesian Computation

The interest and development of Approximate Bayesian Computation in its modern form was thanks to the works of four key papers during the early twenty-first century by Tavaré et al. (1997), Pritchard et al. (1999), Beaumont et al. (2002), and Marjoram et al. (2003). To summarize things, as stated by Beaumont (2010), Bayesian statistics are based on the following observations: creating theoretical frameworks to describe how certain patterns or data might have come about is relatively simple. Using such a model to generate artificial data is as simple as setting specific parameters values in the model. However, the reverse process is where the challenge lies: starting with actual observed data and trying to determine the exact parameter values (or even the model itself) that generated it. In other words, while creating models and generating data from them is relatively straightforward, inferring the original conditions that produced real life data is much more challenging. The Bayesian approach offers a perfect framework to tackle this kind of complex problems in modeling and inferences, as it offers a way to use probability to make inferences about unknown parameters based on observed data.

Bayesian computation involves estimating the probability of parameter values given observed data *i.e.* the conditional probability density. With θ the parameter values and x the observed data, the goal is to compute the posterior distribution $p(\theta|x)$, the probability of the parameters given the observed data, expressed by the formula:

$$p(\theta|x) = \frac{p(\theta|x)\pi(\theta)}{p(x)}$$

with $\pi(\theta)$ the prior distribution (initial knowledge or assumption about the possibles values of $\theta$ prior to any observations) and $p(x)$ the total probability of the observed data across all possible parameters values, normalizing the posterior distribution so it sums up to 1. This probability $p(x)$ is expressed as:

$$p(x) = \int p(x|\theta)\pi(\theta)d\theta$$

where the integral must sum over all possible parameter values.

However, computing this probability is the challenging part in the context of population genetics where models display lots of hidden states. Hidden states are aspects of the model that are not directly observed but that can still affect the data (old selection events influencing the ancestry of the individuals, demographic changes in the population size, interactions between individuals,...) and turn out to be a problem for the computation of the likelihood as it involves summing the probabilities across all possible configurations of these unobserved variables.

As an answer to this problem, population geneticists have developed a different methods such as composite likelihoods (Lindsay, 1988) or the so called Approximate Bayesian Computation (ABC), which has since been used in numerous scientific fields (Beaumont, 2008, Chan et al., 2014). Approximate Bayesian Computation is the idea of approximating traditional Bayesian inference using simulations instead of directly computing the likelihood, which may be too complex or impossible to evaluate. In ABC, we simulate data based on prior knowledge of the parameter values (θ) and then compare the simulated data to the actual observed data (x). To determine whether a simulation is a good match, ABC uses a measure of distance to quantify how close simulated data is to the observed data. If the distance is small enough, the parameter θ used for the simulation is considered a plausible explanation for the observed data and is then added to the pool of accepted prior samples (Raynal et al. in 2018). By repeating this simulation process it is possible to produce an approximation of the posterior distribution (**Figure 3**).

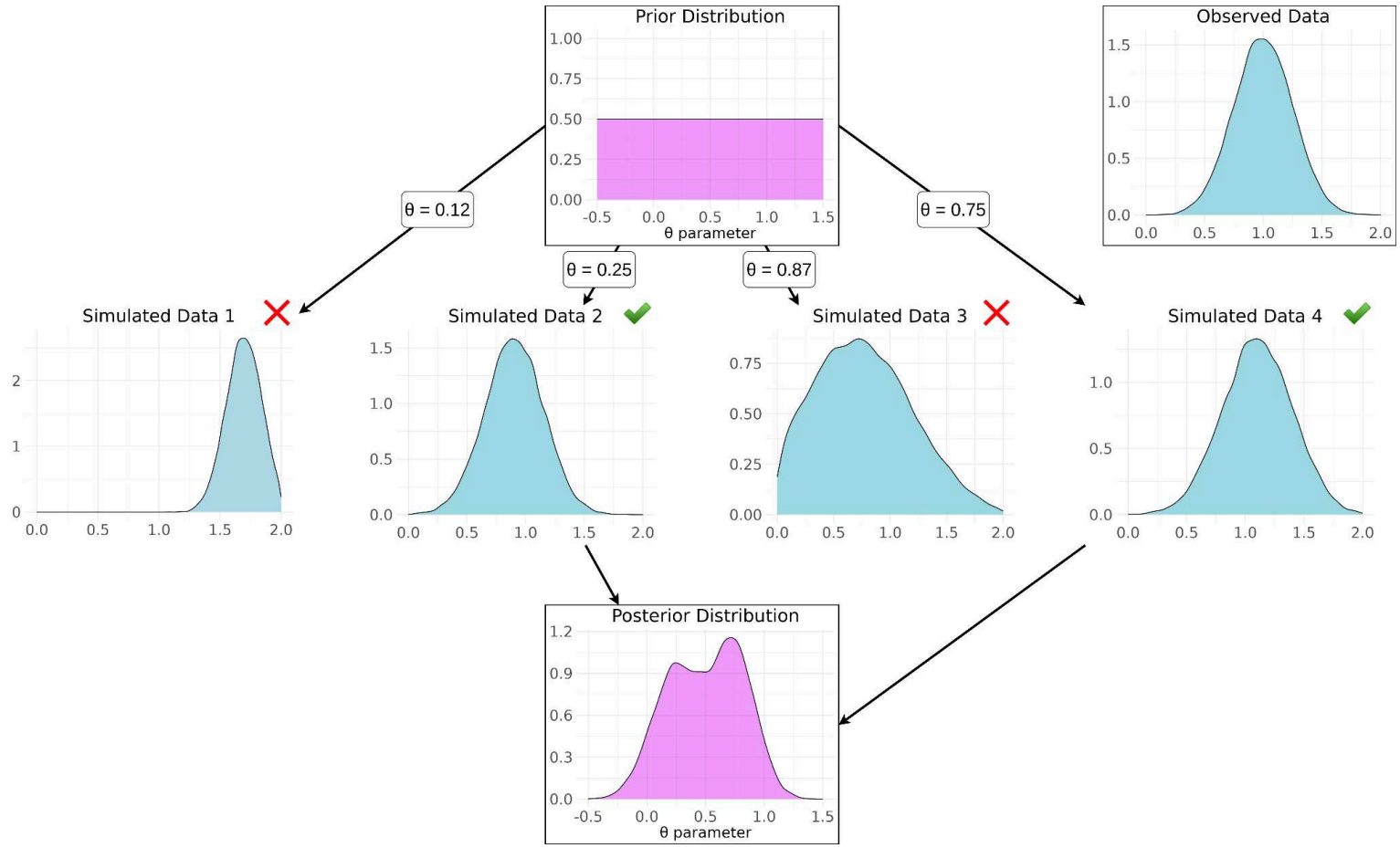



**Figure 3** - Approximation of the Posterior Distribution by ABC.

This figure illustrates the approximation procedure of the posterior distribution by an ABC. Given an observed dataset, a model is constructed and performs *n* simulations, each using a value of the parameter θ drawn from a prior distribution. The simulation results are compared to the observed data and, if the difference is below a given threshold, the corresponding θ value is added to the posterior distribution. With enough simulations resulting in simulated data close enough to the observed data, it is possible to generate an approximation of the posterior distribution.

ABC have been applied in various areas, with applications in ecology to study agent-based models (Piou et al., 2009) or to analyze species abundance data (Solow & Smith, 2009), in epidemiology to fit parameters of epidemiological model of tuberculosis transmission (Tanaka et al., 2006) or to study the evolution of HIV (Shriner et al., 2006) and of course, in population genetics with a myriad of studies making use of ABC to study alleles or haplotypes frequencies, with studies focusing on undercovering the demographic history associated with colonization events of various species, such as *Drosophila subobscura* (Pascual et al., 2007) or *Drosophila sechellia* (Legrand et al., 2009). ABC has also been applied to the study of selection, with the work of Przeworski (2003) on a method to infer

selective sweeps parameters, with works on models with recombination (Wilson et al., 2009) or to study the selection on lactase persistence (Gerbault et al., 2009; Itan et al., 2009).

Recently, more and more performant and flexible tools are being developed to make use of such methods easier for non-experts (Cornuet et al., 2014), such as DILS, an ABC framework allowing for application of ABC models with great flexibility in terms of evolutionary scenarios and type of data inputs (Fraïsse et al., 2020). This tool is in line with the recent surge of new and promising applications of machine learning (for a brief review of the history of machine learning, see **Box 1**).

**Box 1 - A quick history of Machine Learning**

The development of machine learning (ML) and deep learning (DL) find its roots back to the 1940s and 1950s. In 1943, Warren McCulloch and Walter Pitts published "*A logical calculus of the ideas immanent to nervous activity*" where they laid the foundations and earliest theoretical concepts of neural networks. In 1950, Alan Turing in "Computing Machinery and Intelligence" introduced the concept of "learning machines" and proposed to consider the question 'Can machines think?' through the lens of the "*Imitation Game*", now famously known as the "Turing Test". This paper foreshadowed the possibility of building machines that could acquire intelligence through experience. From the 1959 paper of Arthur Samuel, machine learning could then be seen as the "field of study that gives computers the ability to learn without being explicitly programmed". In this paper, Samuel developed an algorithm to play checkers that improved itself by learning from its games, one of the earliest machine learning programs. A year before, in 1958, Frank Rosenblatt introduced the first practical implementation of an artificial neural network, "The perceptron, a perceiving and recognizing automaton", but due to its limitations and drawbacks at that time, highlighted by Minsky and Papert in "Perceptrons" (1969) the field of research about neural network declined in the following years. Accompanying this decline was a shift toward statistical methods in machine learning, driven by advances in probability theory and statistics with linear and logistic regressions and decision trees becoming more and more popular. In 1986, thanks to David Rumelhart, Geoffrey Hinton and Ronald Williams, the backpropagation algorithm for training multi-layer perceptrons was rediscovered and popularized. This marks a breakthrough that in turn allowed for effective training of deep neural networks which reignited the interest for neural networks. Unfortunately, neural networks were still limited by computational constraint and the lack of datasets large enough to make use of them. In parallel, support vector machines (SVM), based on statistical learning theory, developed by Vapnik and Chervonenkis, quickly became one of

the most popular algorithms for classification tasks thanks to their ability to handle high-dimensional data and to utilize kernel functions. Finally, the beginning of a new millenia marked a turning point for ML, as the ever increasing computational power has been followed by a sharp increase in availability of large datasets. Hence, neural networks saw an unprecedented resurgence thanks to the developments of newer and more efficient algorithms such as recurrent neural networks (RNN) and convolutional neural networks (CNN). The turning point was in 2012 when Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton's deep CNN, known as AlexNet, won the ImageNet Large Scale Visual Recognition Challenge, outperforming traditional methods by a significant margin, demonstrating the power of deep neural networks in handling complex pattern recognition tasks and accelerated the adoption of deep learning techniques across various fields. In the following years, innovations such as generative adversarial networks (GANs), introduced by Ian Goodfellow et al. in 2014, and deep reinforcement learning, popularized by the success of DeepMind's AlphaGo in 2016, pushed the boundaries of what deep learning could achieve. The introduction of transformer architectures, starting with the paper "Attention is All You Need" in 2017, revolutionized natural language processing, leading to the development of highly effective language models like BERT and GPT.

## Machine Learning, Supervised Learning, Convolutional Neural Networks

Machine learning (ML), as a subset of artificial intelligence, focuses on developing algorithms capable of learning from data to perform tasks without relying on traditional, explicitly defined mathematical models. Instead, ML algorithms discover patterns within the data and use these insights to make predictions or to identify structures. The power of such methods lies in their ability to improve performance as more data becomes available, making it highly effective for tasks where large datasets are available, such as the modern area of population genetics where genomic data is more available than ever thanks to NGS.

ML methods are generally classified into two main categories: unsupervised (Ghahramani et al., 2004) and supervised learning (Kotsiantis et al., 2007), which differ primarily in the nature of the data provided to the model and the specific objectives of the learning process. Unsupervised learning focuses on identifying inherent patterns or structure within unlabeled data *i.e.* without any explicit target values provided. The objective is often to discover the underlying data distribution, group similar instances, or reduce the data dimensionality. Since there is no "correct" answer to learn from, the model optimizes based on the internal

properties of the data. A simple and commonly used example of such machine learning is principal component analysis, taking a highly dimensional matrix as input and allowing analysis of the clustering of data. On the other hand, supervised learning consists in algorithms training on huge labeled datasets, where the relationship between input features and output labels is known. The goal is to learn a mapping from inputs to outputs allowing the model to predict the expected target value of an unseen data, allowing for classification and regression inferences. Here, the learning process minimizes the discrepancy between the predicted and actual output values through iterative adjustments of the model parameters. **Figure 4** provides a very simple example of how a supervised ML approach works, while Box 2 proposes a more detailed and formally accurate breakdown (both **Figure 4** and **Box 2** are adapted from Schrider & Kern, 2018).

**Box 2 - Supervised Machine Learning**
***(adapted from "Supervised Machine Learning for Population Genetics: A New Paradigm", Schrider & Kern, 2018)***

In supervised machine learning, the goal is to learn a mapping from a given input feature vector $x \in X$, consisting of M input variable, to a target output $y \in Y$, with $X$ the input space and $Y$ the output space such that the relationship between the two can be defined as a function $f: X \rightarrow Y$. The learned function $\widehat{f}$ is an approximation of the true underlying function $f$, which represent the actual relationship between the input and output. The nature of this output determines the type of supervised machine learning problem being addressed: if this output $y$ is categorical, the task is a **classification** task whereas if $y$ is continuous, the task is a **regression** problem.

From there, the objective of supervised ML is then to optimize $\widehat{f}: x \rightarrow \widehat{y}$. In other words, the goal is to find the parameters θ that minimize the discrepancy between the predicted output $\widehat{y}$ and the actual output $y$. To do so, the model is trained using a **training dataset** which consist in a set of labeled data *i.e.* data where the responses values $y$ of each $x$ value is known. Formally, such a dataset consist of $N$ labeled examples $\{(x_1, y_1), \ldots, (x_n, y_n)$ , where pair $(x_i, y_i)$ represents an observation of the input values and the corresponding output label.

The training process uses this set to adjust the parameters θ such that the learned function can predict the outputs accurately for new, unseen data points. To do so, a **loss function** $L$ is implemented to quantify how good or how bad a given prediction is. For classification, a commonly used a easy to understand loss function is the indicator function defined as $L((f(x), y) = 1(f(x) \neq y)$ while regression tasks can use the mean squared error (MSE) $L((f(x), y) = (f(x) - y)^2$. From the loss function, the **risk function** is defined as the average value of $L$ across the entire training dataset. Thus, the goal of the training is to find the parameters θ such that the risk function is minimized.

After the training, the model is evaluated on a separate set of data, the **test dataset**, which was not used during training. This evaluation assesses the model's generalization ability *i.e.* how well it performs on new, unseen data. What is important after the training is for the $\widehat{f}$ function to have correctly learnt the overall characteristics of the problem and not the specific characteristics of the training dataset specifically, which is known as **overfitting**. In order to evaluate the models performances, **precision** and **recall**, commonly used metrics based on the rate of positive and negative predictions, as well as **confusion matrices**,

which are contingency tables of true *vs*. predicted class labels, are helpful tools in the case of classification. As for regressions, usual tools used to test for model fit such as $R^2$, or the examination of various loss functions are good approaches.

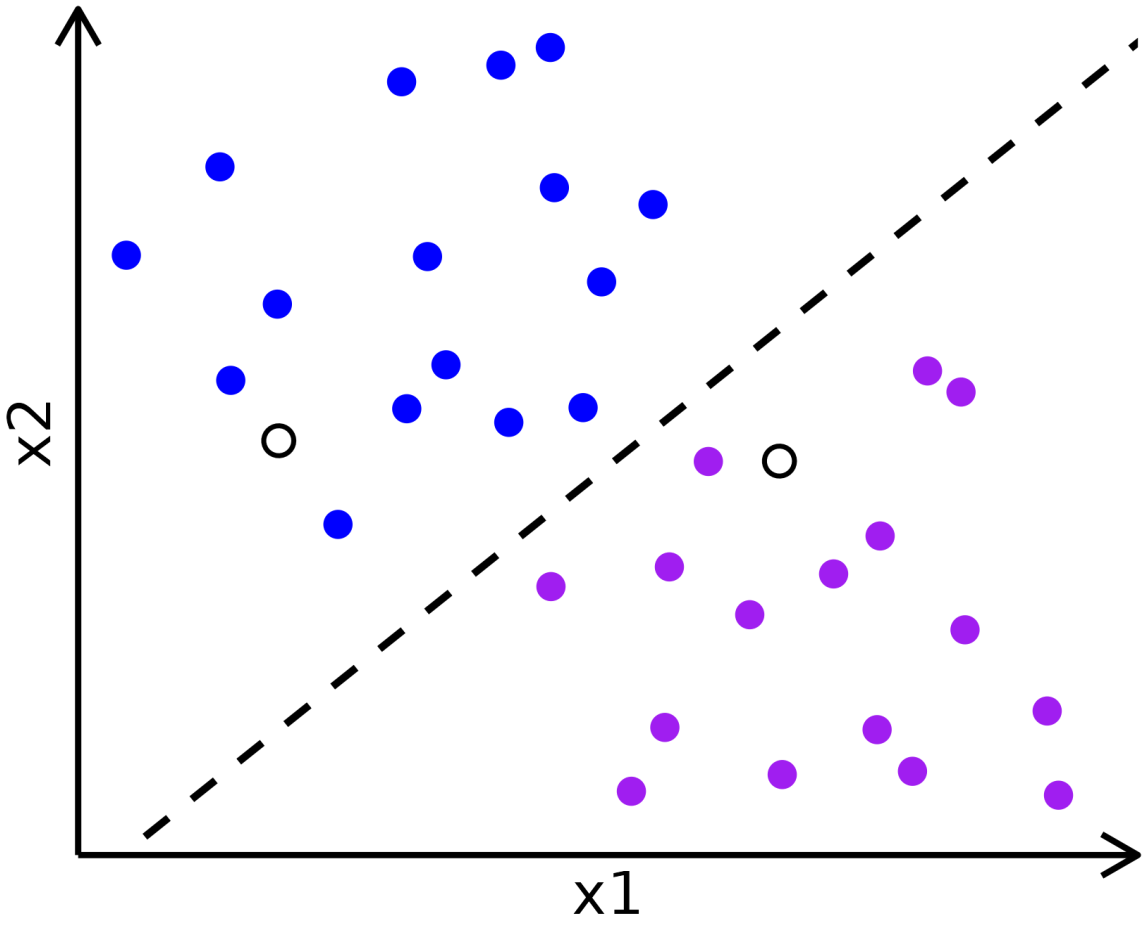



**Figure 4** - Simple example of how supervised machine learning works for a classification task.

By using the blue and purple points as training data, with x1 and x2 the two input variables of the feature vector, supervised ML can learn a function (the dashed line) that differentiate data between the two classes and be used to classify new, never seen before data points (in white). (*Figure adapted from "Supervised Machine Learning for Population Genetics: A New Paradigm", Schrider & Kern, 2018*)

As stated before, machine learning, and more specifically supervised-machine learning works by learning to match a given input with a specific desired output using huge datasets of labeled training data. Theoretically, as long as a model is possible to simulate, a machine learning model could be trained to fit it, giving to such methods an incredible power to tackle a variety of complex scenarios and evolutionary questions. In the past years, theses methods have received more and more attention (Schrider & Kern, 2018) and they have been used by numerous population genomics studies (Flagel et al. 2019; Kern & Schrider, 2018; Lin et al. 2011; Mughal & DeGiorgio, 2019; Pavlidis et al. 2010; Ronen et al. 2013;

Torada et al. 2019; Xue et al. 2021; Caldas et al. 2022) and in new tools developed to detect signatures of selection from genetic data. To name a few, Schrider and Kern (Schrider & Kern, 2016) developed a software called S/HIC" and its newer version "diploS/HIC" (Kern & Schrider, 2018) using a randomized trees classifier to differentiate various selective sweeps, linked and neutral regions. Hamid et al. (Hamid et al., 2023) propose a deep learning method using local ancestry-painted genomes to localize post-admixture adaptive variants, while Caldas et al. (Caldas et al., 2022) use supervised learning to estimate different parameters of a given selective sweep.

Many studies also succeeded to use machine learning to make inferences about and estimate parameters of the demographic histories : time to most recent common ancestor, estimates of dispersal distance from genotype data, recombination rates or even local ancestry ( Schrider et al., 2018 - Flagel et al, 2018 - Saada et al., 2023 - Smith et al., 2023). For a more detailed review of the uses of deep learning in population genetics, see Sheehan & Song "Deep Learning for Population Genetic Inference" (2016), Korfmann et al., "Deep Learning in Population Genetics" (2023) or the very readable and beginner-friendly "Supervised Machine Learning for Population Genetics: A New Paradigm" by Schrider & Kern (2018).

Among them, Convolutional Neural Networks (also known as 'CNNs' - LeCun et al., 1998) are models constructed on the basis of Artificial Neural Networks (ANNs). The structures of ANNs are inspired by biological nervous systems: a large number of 'nodes', also called neurons (in fact polynomial functions) are linked together and take a value (input) as input and return a value (output) as output (Mitchel, 1997). The classic architecture of an artificial neural network, (often interchangeably called dense feed-forward neural network), can be represented as in **Figure 5**.

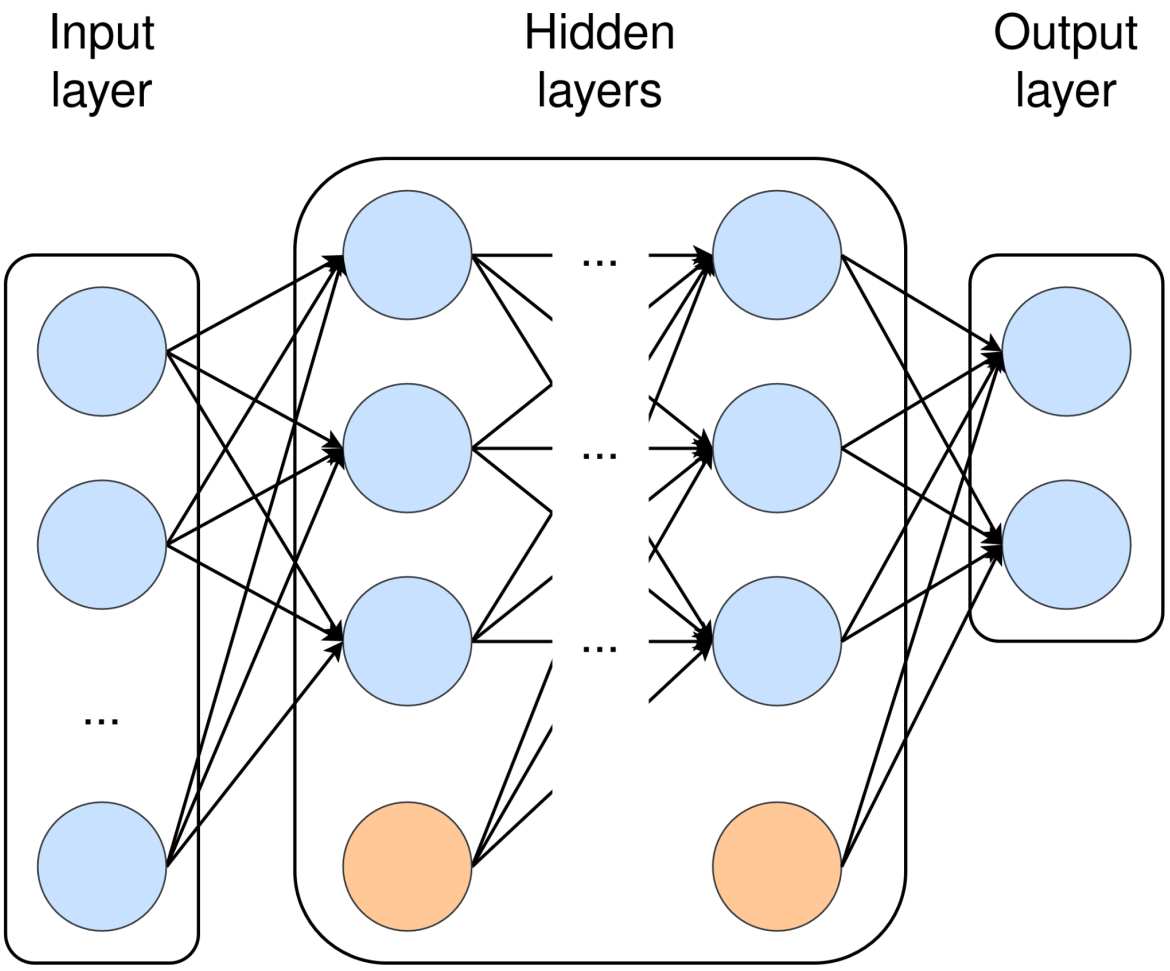



**Figure 5** - Simplified representation of a classic Artificial Neural Network architecture.

Blue circles represent either input, output or hidden neurons. Orange circles represent the bias terms. In terms of internal mechanisms, each internal neuron of the hidden layers consists of the results of a linear combination of the output of all neurons of the previous layers plus a bias term going through an activation function.

These networks can be used, for example, for image recognition: in this case, the input is an image (a matrix of pixels) and the output can be a prediction of the main color of the image or its content. The neurons making up the network are organized into successive layers: the input layer receives the data to be analyzed before passing it on to one or more hidden layers which will analyze the data. The multiplication of successive hidden layers provides access to information that is ‘deeper’ in the data: this is known as deep-learning (O'Shea, 2015). Finally, the output layer will return the ‘expected’ value after analysis by the network.

Formally, a neural network consists of the following components:

- Input layers: the input data is fed into the network, with each node representing a feature of the input.

- Hidden layers: one or more layers of neurons that perform computations on the data to generate the desired output.
- Output layer: produces the network's prediction, with each node being one of the output dimensions.

For the network to produce the output for a given input, the network goes through forward propagation. Let's consider an input vector $x$, with $n$ features. For each of layer $l$ we have :

- $z^{(l)}$: the linear combination of inputs of layer $l$
- $a^{(l)}$: the activation output of layer $l$

Each neuron of $l$ computes a weighted sum of the outputs of all neurons of the previous layers such as:

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)},$$

with $W^{(l)}$ the weight matrix for layer $l$, whose dimensions depends on the number of neurons of the previous and the current layers, $a^{(l-1)}$the activation output from the previous layers and $b^{(l)}$ the bias vector of layer $l$. The results of this sum (also known as linear transformation) is then passed to an activation function to compute the activation output of the neurons of the current layer:

$$a^{(l)} = \sigma(z^{(l)}),$$

with $\sigma()$ the activation function. A very commonly used one is the ReLU function (Rectified Linear Unit), $\sigma(z) = max(0, z)$. Once at the output layer, a last linear transformation is done, followed by an activation function appropriate for the task at hand, either a sigmoid activation for a binary classification or a softmax for a multi-class classification for example.

The particularity of CNNs (**Figure 6**) compared to neural networks conventionally used in machine learning is the use of a convolutional layer in order to identify key characteristics of the input used to carry out their inferences (LeCun et al., 1998). To build their own feature vector, additional layers, called convolutional layers, are added before the classic architecture of a standard artificial neural network. This stage is generally followed by a pooling layer, which "compresses" the information, reducing the matrix size while retaining as much of the information extracted by the convolutional layers as possible. This step (a convolutional layer followed by a pooling layer) is repeated a *n* times to only retain the important, filtered and compressed information. This information is then flattened in the form of a one-dimensional vector (LeCun et al., 2015) and passed to a classic dense neural network. To breakdown things more formally:

Considering an input image of dimensions $H \times W \times C$, with $H$ the height, $W$ the width and $C$ the number of color channels (1 for gray-scaled pictures, 3 for RGB). This input is passed to convolutional layers which perform the convolution operation *i.e.* the extraction of the features from the input. To do so, convolutional kernels, sets of learnable filters, are used to scan the input and to generate a new representation, a feature map. These newly generated feature maps then go through pooling layers which down-sample *i.e.* reduces the dimensions of the feature maps, notably to control for overfitting. This pair of steps are repeated *n* times before the output is flatten to then be fed through a classic, fully connected dense neural network which is used to make the final prediction based on the features extracted by the convolutional phase.

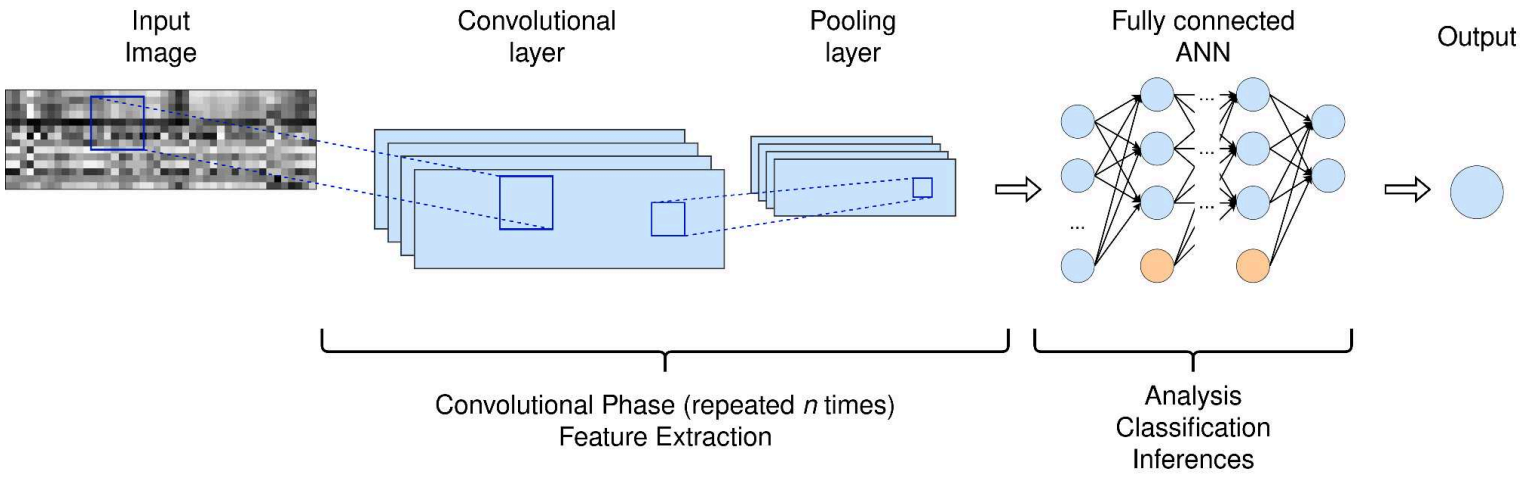



**Figure 6** - Simplified representation of a Convolutional Neural Network architecture.

For a CNN, the input is a matrix (here, a picture of a genomic alignment as the ones we use in this study) of known dimensions $H \times W \times C$ goes through a series of convolutional layers where a set of filters (kernels) scan the image to extract specific aspects and generate new representations of the input: feature maps. A pooling layer then performs a down-sampling to reduce the dimensions of the feature maps. These two steps, usually pooled together as the *convolutional phase* is repeated *n* times, before their output is flattened (step not represented here) and passed to a classic fully connected dense neural network for it to perform the analysis, classifications or inferences and generated the desired output.

# Objectives of the thesis

As seen in this introduction, numerous methods have been developed through the history of population genetics to answer the ever growing need for theoretical and computational tools to unravel the evolutionary history of natural species. However, while such methods have been successful, one major assumption remains present in all of those methods aiming at reconstructing demographic scenarios : the genomic data must have evolved under selective neutrality (Schrider et al. 2016). This represents a major issue, as a large part of natural populations are evolving under selection (Hahn, 2008) and such selection could lead to variations of the genetic diversity that could in turn be mistakenly attributed to some demographic event, while the latter are known to cause many test for selection to generate false positives (Simonsen *et al.* 1995; Jensen *et al.* 2005; Nielsen et al. 2005). This circular thinking between demography and selection has been one of the major obstacles to evolutionary inferences in population genetics. In this thesis, we propose to tackle this question and to explore possible ways of circumventing this issue through the prism of convolutional neural networks. The idea is to use CNNs for two main tasks. First, using pseudo-genomic alignments sampled from a population as input data, can the CNNs correctly identify the demographic scenario under which the aforementioned population has evolved. The second task is the neverending goal of accurately detecting selection. Here, we focus not only on detecting genomic targets of natural selection, but more so on accurately localizing the targets along the genomes. For each task, the goal is not so much to develop a ready-to-use tool, but to get a better understanding of the many choices done at each step of the development of such deep-learning methods and to try and propose an enlightened view of some of the issues that may arise while doing so. As both tasks require the same main steps, our approach can be summarized as follow (**Figure 7**):

1. Pseudo-genomic data is generated using the *msms* coalescent simulator (Ewing and Hermisson, 2010), with populations simulated under various demographic scenarios, either under selective neutrality (with only neutral mutations) or in presence of selection. In the case of the latter, the simulated population history involves a selective sweep rising from a beneficial mutation. The output data from the simulations are formatted as ms files, which contain the SNPs positions and their state, coded as either 0 (ancestral state) or 1 (derived state).

2. The simulation outputs are processed and the data is formatted to produce datasets for the training and testing of the networks. This involves converting the *ms* files into

two types of matrices: (a) matrices representing raw genetic alignments as black-and-white pixel images, or (b) matrices of summary statistics computed within sliding windows across the genome, formatted as grayscale images. Labels indicating the demographic scenario as well as the position of the simulated selective sweeps are also generated for use in supervised training. Data is then split into three datasets : *train, validation* and *test*. The training and validation sets are used during the CNN training process, where the validation set helps in monitoring the model's performance to avoid overfitting. The test set remains separate until the final evaluation, allowing an unbiased assessment of the trained model's generalization ability.

3. The CNNs are trained using the *train* and *validation* datasets. This training phase involves the choice of numerous parameters and hyper-parameters *i.e.* parameters specific to the training phase of the network. Throughout this phase, the network training is monitored thanks to the *validation* dataset in order to get a better grasp of the impact of each chosen parameter.

4. After this initial training, the performance of the CNNs is assessed using a set of unseen data, the *test* dataset. Various metrics (accuracy, precision, recall, and AP scores) are computed to evaluate the network performances. Depending on the results, the data formatting, training parameters, or network architecture may be adjusted to improve performance, followed by further rounds of training and evaluation.

5. Once the CNNs are correctly trained, their final performance is tested against established methods in population genetics. For classification of demographic scenario, the CNN results are compared with those of ABC-RF (Approximate Bayesian Computation Random Forest - Pudlo et al. 2015), while for detecting selective sweeps, the performance is evaluated against SweepFinder2 (DeGiorgio *et al.*, 2016). This step helps to validate the CNN's accuracy and robustness and provides insight into its potential advantages or limitations in comparison to existing approaches.

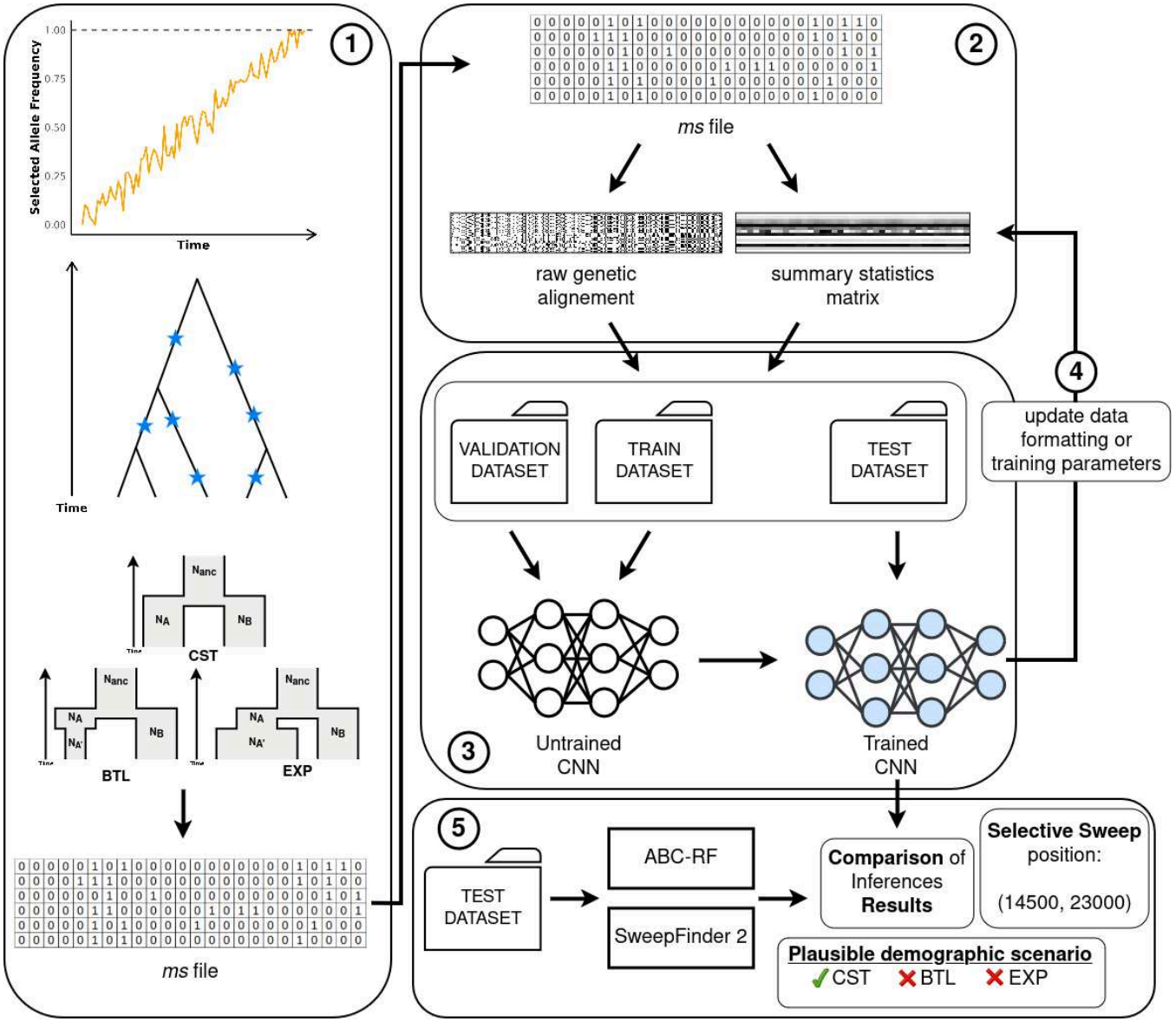



**Figure 7** - General organization of the main steps of both approaches.

Summarized organization of the main steps used for the development of both methods presented in this thesis. ① Simulation of pseudo-genomic data using the *msms* simulator. For the simulations with selection, the frequency path of the selected allele is first generated. Then, the population's histories are simulated through coalescent simulations (conditional to the frequency path in case of selection), following one of three possible demographic scenarios of constant size (CST), bottleneck (BTL) or expansion (EXP). The outputs are *ms* formatted files. ② Data processing of the simulation output. Each *ms* file is converted either directly into matrices of black and white pixels (raw genetic alignments) or summary statistics are computed on sliding windows along the genomes and then converted into gray-scaled matrices. ③ The generated matrices are splits into 3 distincts datasets : *train*, *validation* and *test* datasets. The *train* and *validation* datasets are used for the training of an untrained CNN. The CNN performances are evaluated on the *test* dataset. ④ Depending on the CNN's performances, updates and tweaks are made on the data formatting and the training parameters. ⑤ Once the CNN is correctly trained, the inferences results are compared to ABC-RF and SweepFind2, two other commonly used tools for classification and selective sweep detection.

The first chapter of this thesis focuses on a breakdown of the simulation protocol, followed by the development and test of a simple homemade CNN architecture used to classify genetic alignments between various demographic scenarios. Parallel to this homemade architecture, two pre-trained architectures are also fine-tuned and all the CNNs are compared. The best ones are tested against ABC-RF. The analysis addresses aspects such as the choice of data format (raw alignments *vs.* summary statistics), the influence of selection, the robustness to model misspecification, and the impact of unexpected gene flow. By examining these factors, this study aims to provide insights into the strengths and limitations of CNN approaches in demographic inference, for more informed methodological choices in future research.

# Chapter I

## *Classification of genomic data according to plausible demographic scenarios using Convolutional Neural Networks*

# Introduction

Knowledge about the demographic history of a population leads to a better understanding of its evolutionary history. As the effects of selection and demographic changes are often hard to distangled due to very similar patterns of diversity left on the genomes by those two mechanisms, on the most common approach is to assume that demography have genome-wide effects, while selection works at a local scale and only targets specifics loci (Li et al., 2011). This simple assumption has been largely used as a way to check for false positives in demographic inferences (Nielsen et al., 2005 ; Li & Stephan, 2006 ; Pavlidi et al., 2013). However, as soon as 1968, random drift was experimentally proven to affect selection dynamics in small populations (Frankham et al., 1968) and recent studies of sequence polymorphism indicates that things might be more complex, and that selection might shape diversity in more intricate ways (Sella et al., 2009 ; Siol et al., 2010). Thus, the development of methods able to realize correct inferences of the demographic history of natural populations from genetic data are one of the main goals of population genetics. From likelihood methods (Kuhner et al., 2000 ; Beerli & Felsenstein, 2001 ; Hey & Nielsen, 2007; Hey, 2010) to summary statistics centered ones (Becquet & Przeworski, 2007 ; Naduvilezhath et al., 2011) all the way to ABC focused ones and the myriad of other using site-frequency spectrum (SFS – Nielsen, 2000 ; Adams & Hudson, 2004 ; Marth et al. 2004 ; Gutenkunst et al., 2009 ; Naduvilezhath et al., 2011 ; Lukic & Hey, 2012), incredible advances has been made, leading to the development of powerful tools the likes of δaδi (Gutenkunst et al., 2009) or fastsimcoal2 (Excoffier et al., 2021). These methods have led to great improvements in identifying historical demographic events such as population size changes (Marth et al. 2004; Tennessen et al. 2012; Gazave et al. 2014) and genetic exchange between populations and species (Martin et al. 2013; Hellenthal et al. 2014; Sankararaman et al. 2014; Corbett-Detig and Nielsen 2017; Schrider et al. 2018; Hamid et al., 2021).

All of those methods find their roots back from one of the very first evolutionary models proposed by Hardy and Weinberg in 1908. This model shows the evolution of a biallelic locus in a panmictic population of infinite size and introduces the so-called Hardy-Weinberg equilibrium: under those conditions, and in absence of mutation or selection, the two alleles will reach an equilibrium. A decade later, a turning point occurred with the introduction of the Wright-Fisher model (Wright, 1931 ; Fisher, 1922). In this new model, evolutionary forces and demographic parameters are introduced in the form of mutations occuring into a population of finite size. In 1958, Moran (Moran, 1958) proposed a model with a new twist : overlapping generations and asexual reproduction and a few years later, Kimura's work (Kimura, 1968) marked another milestone with the neutral theory of molecular evolution. Parallel to these advances in the development of these theoretical frameworks, methods for sampling and sequencing individuals are also seeing significant progress. Hence, more and more genetic data became available to study present day populations. However, the theoretical models have all been focused on a 'forward in time' approach thus far: theoretical populations are sets considering a set of given parameters, and the models attempt to move them 'forward in time' from that state. Thus, a brand new evolutionary model was needed to understand the evolutionary evolution of populations 'backward in time' – working from the present-day observed variation and tracing the story back in time. It is Kingman, in 1982 (Kingman, 1982) that provides an answer to this need : the coalescent model (**Figure 8**).

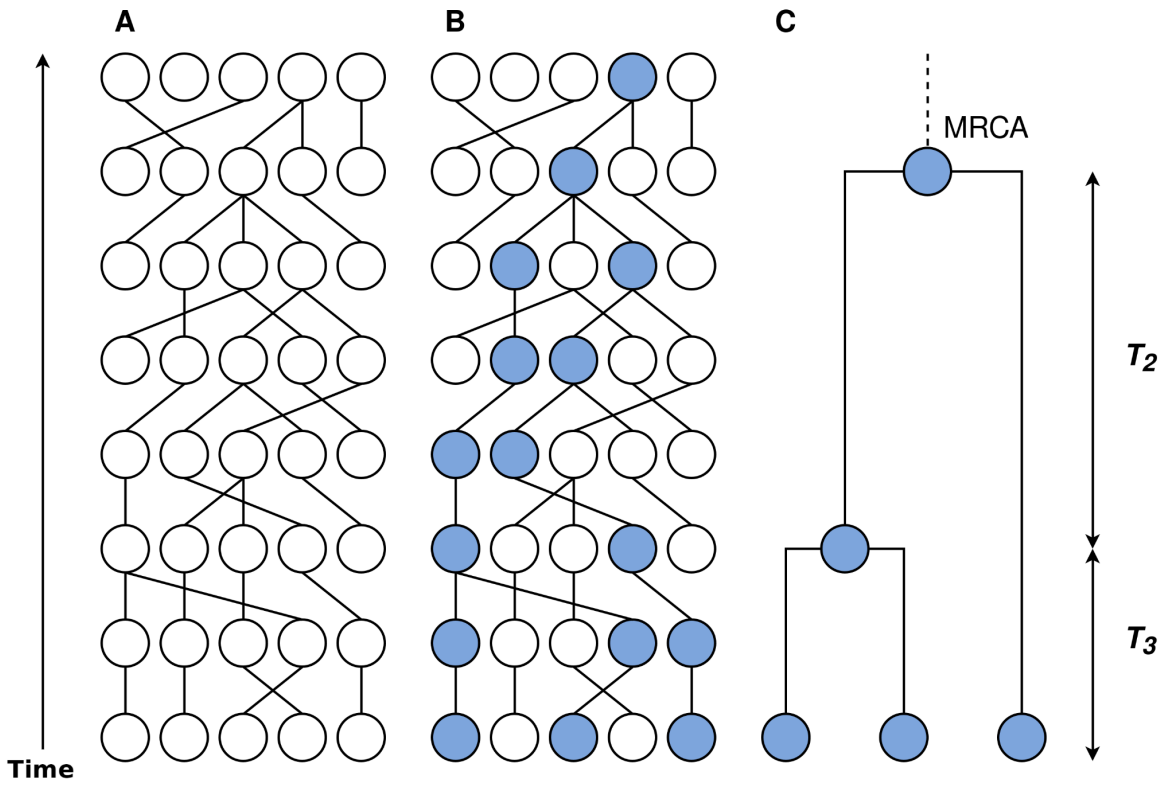



**Figure 8** - Coalescent Model

Individuals (circles) are linked together through generations by their ancestry (line). **A)** Basic representation of the genealogy of a population. **B)** Genealogy of a sample of individuals/genes of the population. By sampling 3 individuals from this population at present time, we can retrace their genealogy backward in time (blue circles). Doing so allows to ignore all other individuals that are not part of this genealogy (white circles). **C)** Simplified genealogy of the lineage of the three sampled individuals. $T_3$ and $T_2$ are the times between each coalescent event, occurring when two or more lineages merge. The last coalescence event led to the most recent common ancestor (MRCA). *Figure adapted from De Smet., 2014*

The coalescent process models the genealogy of a sample of alleles from a population by tracing their ancestry backward in time, all the way until all lineages coalesce to a single, most recent common ancestor (MRCA). This model is based on the same assumptions of a neutral Wright-Fisher idealized population: a panmictic population of constant size $N$, with discrete generations, without selection, mutation or migration. The most important aspect of this coalescent model is the time of coalescence. For any two lineage of the population, the time to coalescence is an exponentially distributed random variable with mean of $2N_e$ generations. The expected time for two lineages to coalesce is $T_k = \frac{2N_e}{k(k-1)}$ with $k$ the

number of lineages remaining at this step of the process and $T_k$ the expected time between coalescence with $k$ lineage remaining. Initially, the probability of any two lineages coalescing is higher due to the higher number of lineages. As the process continues, the number of lineages decreases leading to longer times for the remaining lineages to coalesce. The expected time for the all sampled lineages to coalesce is the sum of the expected times, which approximates to $4N_e(1 - 1/n)$ generations for a sample of size $n$.

This model has since then been expanded upon, with the incorporation of mutation, recombination and selection processes that will alter the shape of genealogies, but also variable population sizes, or even gene flow.

However, while such methods have been successful, one major assumption remains present in all of those methods aiming at reconstructing demographic scenarios : the genomic data must have evolved under selective neutrality (Schrider et al. 2016). This represents a major issue, as a large part of natural populations are evolving under selection (Hahn, 2008) and such selection could lead to variations of the genetic diversity that could in turn be mistakenly attributed to some demographic event, while the latter are known to cause many test for selection to generate false positives (Simonsen *et al.* 1995; Jensen *et al.* 2005; Nielsen et al. 2005).

As stated in the introduction of this thesis, the one major issue of such methods, despite having already shown great success, is that the studied genomic data used to reconstruct the demographic scenarios must have evolved under selective neutrality (Schrider et al., 2016). Indeed, for years, the paradigm in population genetics research has been to follow what Breiman defines as the “data modeling culture”: a statistical model is devised with the aim of replicating the effect of a natural phenomenon well enough for it to be considered a correct approximation of it. Then, independent variables (genetic or demographic parameters for example), are fed through the model to generate, hopefully, a response close enough to the one of the natural phenomenon. The model is then validated using goodness-of fit tests and residual estimations. However, the other mode of analysis, the “algorithmic modeling culture”, works quite differently. In this approach, the effects of nature are considered unknown and too complex to come up with a practically usable model. Thus, the approach is to find a function, *i.e.* an algorithm that will simply take as input any given parameter and will try to replicate as closely as possible the observed data (**Figure 9** - Brieman, 2001).

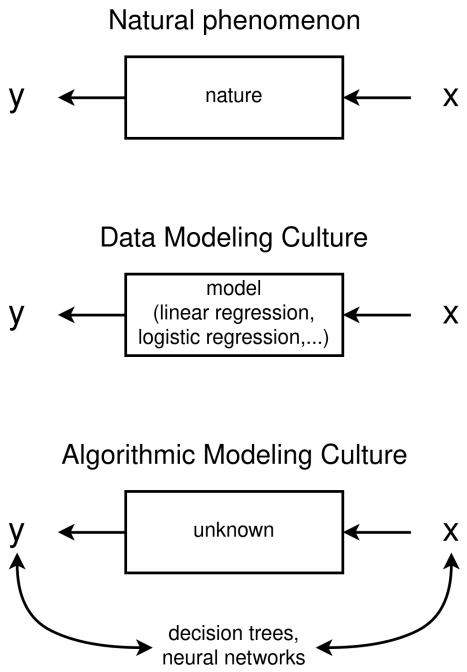



**Figure 9** - Schematisation of the data modeling and algorithmic modeling culture.

Any observed data y produced by a natural phenomenon working, akin to a black-box, on a vector of independent input variables x. To analyze such data, *i.e* to either make a prediction about the possibles responses or to extract information of how nature works, two different approaches exists: the **data modeling culture** assumes a given statistical model to be a close enough representation of nature's blackbox, and feed input variables into this model used to make predictions or to gather informations. The **algorithmic modeling culture** considers the inside of the box complex and unknown and focuses on finding a function *i.e.* an algorithm capable of taking x to generate the exact same y responses. *Figure adapted from Brieman, 2001*

This 'algorithmic modeling culture' is what is known as machine learning (ML). Over the past two decades, machine learning has made significant advancements across a variety of domains, demonstrating exceptional performance in numerous tasks: in natural language processing (machine translation, sentiment analysis or with transformer-based models like GPT) and both text (Sebastiani, 2002) and speech recognition (Hinton et al., 2012) with the development of virtual assistants, but also in the fields of computer vision and image classification (Krizhevsky et al., 2012) as well as in bioinformatics (Byvatov & Schneider, 2003; Angermueller et al., 2016; Libbrecht & Noble, 2015). Studies more specifically catered to the use of ML methods in population genetics have seen an incredible rise during the last few years (Caldas et al., 2022; Flagel et al., 2019; Fraïsse et al., 2020; Gower et al., 2021; Hamid et al., 2023; Kern & Schrider, 2018; Kittlein et al., 2022; Lopez et al., 2018; Nait Saada et al., 2023; Sanchez, 2022; Schrider & Kern, 2016; Smith et al., 2023; Yelmen et al.,

2021) and seem to bear great promises for the application of such tools in the future (see “The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inferences” by Flagel et al., 2019 and “Deep Learning in Population Genetics” by Korfmann et al., 2023 for good and very readable reviews of the subject).

In this first part, we focus on the development of one specific type of deep-learning method as a way to answer the previously highlighted issue of performing accurate predictions of demographic history from genetic data influenced by selection mechanisms. This question is studied through the lens of Convolutional Neural Networks (CNNs), used to classify genomic data according to demographic scenarios. The main goal is to highlight the important steps of such a demarche and try to help make the decisions when training a CNN to tackle such a task. To do so, we train our CNN for a common task in population genomics: determining the demographic history associated with a given genomic alignment. In our case, it boils down to a classification task between possible demographic scenarios. We propose a detailed comparison, at different levels of the development of the method, between 3 different architectures of CNN as well as a comparison to Approximate Bayesian Computation using Random Forest (ABC-RF - Pudlo et al. 2016), an already widely used tool in population genomics (Fraïsse et al, 2021; Pavinato et al. 2022). This comparison consist of four main points : 1) the choice of the type of data to use for the training and to run the inferences for the CNNs, *i.e.* the alternative choice between using a set of summary statistics or running inferences using raw genomic alignments, 2) the impact of selection on both methods performances, 3) the impact of model misspecification on the classifiers robustness and 4) the effect of unanticipated gene flow in the history of the sampled populations.

We found that pre-trained CNN architectures performed better compared to more homemade approaches in most, if not all cases. Moreover, pre-trained CNNs fine-tuned on raw alignments often outperformed architectures trained on summary statistics. Moreso, CNNs trained on pseudo-genomic data from populations simulated with a selective sweep equals or outperforms the others, without any drawbacks regarding the training. Finally, even though CNN based approaches outperformed the ABC-RF in terms of accuracy, an argument could be made in favor of the ABC-RF for their already more widespread usage making it a more straightforward and easier method to use for simple questions and for their good accuracy.

# Materials and Methods

## Pipeline overview

Our method requires the five main steps illustrated in **Figure 7**. First, pseudo-genomic data is simulated using the *msms* coalescent simulator (Ewing and Hermisson, 2010). Populations are simulated under different demographic scenarios, detailed below, in presence and absence of selection - in our case a selective sweep arising from a single new beneficial mutation increasing in frequency in the population. Second, the output of the simulations are processed by bash and python scripts to generate the datasets used to train and test the classifiers, either by providing matrices of raw genomic alignments of genomes sampled from the simulated populations, or by computing matrices of summary statistics based on such alignments. Third, the generated matrices are split into *train*, *validation* and *test* datasets and the classifiers are trained using the *train* and *validation* datasets. Both methods require different training processes, detailed in their respective sub-sections. Fourth, potential updates are done to the data formatting or the training steps. Finally, the trained classifiers (*i.e.* the various CNNs of each of the three tested architecture types), as well as the ABC-RF, are tested on a *test* dataset to evaluate their performances on new, never seen before data.

## Simulations

We simulate pseudo-genomic data for training and testing of our CNNs using the *msms* coalescent simulator under tree possible basic demographic scenarios : a population of constant effective size (labeled as **'CST** scenario' - constant size), a population where the effective size suddenly decreases (labeled as **'BTL** scenario' - bottleneck) and a population which effective size suddenly increases (labeled as **'EXP** scenario' - expansion). Each demographic scenario is simulated in presence or absence of unidirectional migration coming from a non-sampled, "CST" sister population (sometimes referred to as the "*ghost sister population*"). On top of these demographic scenarios, each simulation is run twice: once under selective neutrality, with all the mutations occurring being neutral, and a second time in presence of a selective sweep arising from a single beneficial mutation occuring at a random generation and at a random position on the genome.

## Simulated models - Demography:

The simulations follow a Wright-Fisher model. As shown in **Figure 10,** we start from an ancestral population of size $Ne_{anc}$ which splits into two sister populations A and B of equal initial size $Ne_A = Ne_B = Ne_{anc}$. After some time, a demographic change happens in population A : (1) **CST** : the population size remain constant, (2) **BTL** : the population goes through a sudden bottleneck which reduces the value of $Ne_A$ down to $Ne_{A'} < Ne_A$ or (3) **EXP** : the population goes through a sudden expansion which increases the value of $Ne_A$ up to $Ne_{A'} > Ne_A$. As stated before, each of these scenarios is also simulated in presence and absence of gene flow from population B. When present, migration is set so, on average, one migrant from population B arrives in population A every 20 generations ($4.N_A.m = 0.2$). The counterparts of the CST, BTL and EXP scenarios with migration are labeled as **MIG** (migration), **MGB** (migration-bottleneck) and **MGX** (migration-expansion).

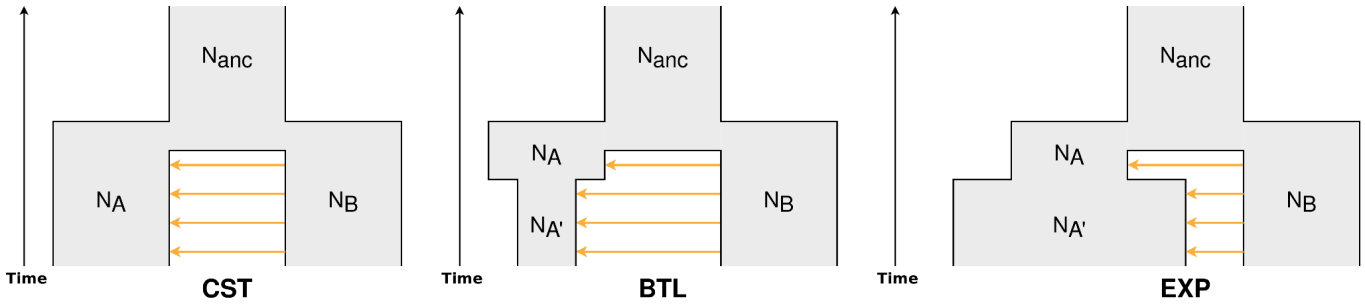



**Figure 10** - Simulated demographic scenarios

All scenarios start the same, with an ancestral population of size $N_{anc}$ that splits into two sister populations of respective sizes $N_A$ and $N_B$. The **CST** scenario is the baseline. It works as the most basic case, with the focus on a single population of constant size (population A). In the **BTL** scenario, the population of interest goes through a sudden and permanent bottleneck, drastically reducing the effective size of the population from $N_A$ to $N_{A'}$. The **EXP** scenario is the opposite, with a sudden and strong expansion permanently increasing the population size from $N_A$ to $N_{A'}$. Each of the three scenarios is simulated in presence or absence of unidirectional migration (orange arrows) from population B to population A, for a total of 6 possible demographic histories.

## Simulated models - Genetic model:

The simulated populations consist of diploid individuals represented by their genome. In fact, the individuals can be seen as chromosomes of size L = 100 00 bp, with a fixed recombination rate *r* of 1e-6 crossing over per generation per base pair. These chromosomes are either simulated under selective neutrality or in presence of a directional

positive selection. For the former, only neutral mutations are allowed to occur. Such simulations void of selection events are used for the *neutral* datasets. To test for the robustness of our method when faced with selection also impacting the shape of genetic diversity signals, the populations are also simulated in presence of a selective sweep, a process where a beneficial mutation is driven to fixation by the effect of selection. Assuming selection at a single locus, and a biallelic A/a state for the allele, fitness values for each genotype are $1 + s_{aa}$, $1 + s_{aA}$ and $1 + s_{AA}$. The resulting signature is what is called a 'selective sweep' *i.e.* the creation of a valley of diversity around a selected locus (Haigh, 1974) due to the rapid increase in frequency around it (**Figure 11**). These simulations are used for the *sweep* datasets.

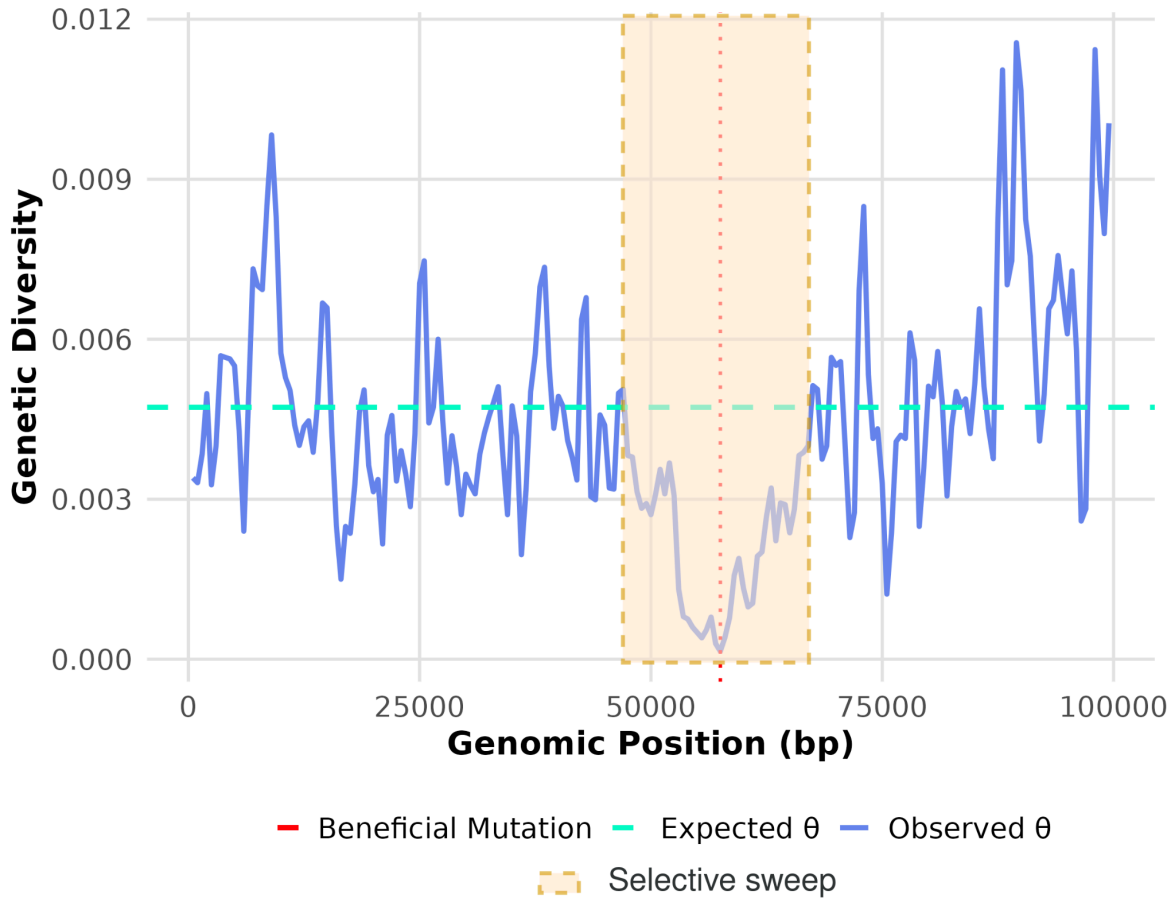



**Figure 11** - Effect of a selective sweep on genetic diversity along a genome.

Comparison of the average expected diversity θ measured along a genome evolving under neutrality (green line) *vs.* the observed genetic diversity (blue line) along a genome with a selective sweep arising from a single beneficial mutation occurring at a selected locus (red line). The x-axis represents the genomic positions. The y-axis is the measure of the genetic diversity θ computed in overlapping windows of 1kb with a step of 500 bp.

## Simulated models - Coalescent simulations:

Simulations are run using the *msms* simulator. Below is a quick overview of the simulation process, also illustrated in **Figure 12**:

1) In case of simulations involving selection, time-forward simulations generate the frequency path (*i.e.* the trajectories) of the selected allele.
2) Coalescent simulations are then run based on the provided demographic scenario to construct the genealogy of neutral loci, conditional to the possible frequency path previously simulated.
3) Neutral mutations are added to the genealogy branches, following a Poisson process, where the probability of a neutral mutation occurring on a specific branch depends on its length.

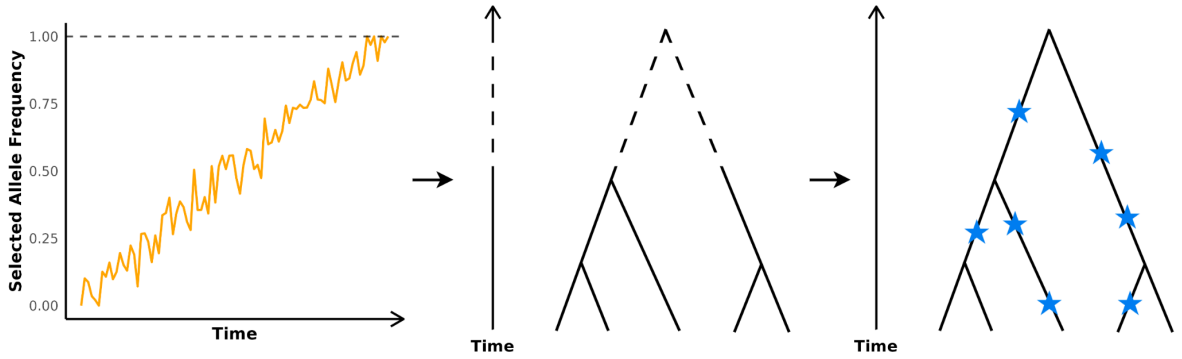



**Figure 12** - Overview of the *msms* simulation process

Considering a simulation with selection, the frequency trajectory of the selected allele is generated first. Neutral loci's genealogies are constructed using a coalescent procedure, conditional on the frequency path. Neutral mutations are added on the genealogy based on the branch's length.

To summarize the key parameters of the simulations, as they are important to better grasp the simulated data :

The effective size $\boldsymbol{Ne_A}$ and $\boldsymbol{Ne_B}$, of population A and B respectively, depend on the demographic scenario simulated. For **CST/MIG** simulations, the effective sizes of population A and B are both equal to the effective size $Ne_{anc}$ of the ancestral population, drawn in a **uniform distribution between 1 000 and 10 000 individuals**. The population sizes remain constant through time. For **BTL/MGB** and **EXP/MGX** scenarios, a parameter (***scalar_N***) is drawn in a **uniform distribution** $U(0.02,\ 0.2)$. It defines the **strength of the demographic event** to happen, in other words the factor by which the effective size of population A will

vary following the bottleneck or the expansion. For **BTL/MGB** scenarios, the population's initial sizes of the ancestral population, and of population A and B are set between 1 000 and 10 000 individuals, and **the size of population A following the bottleneck**, $Ne_{A'} = Ne_A \times scalar\_N$, so $Ne_{A'}$ **is between 20 and 2 000 individuals**. For the scenarios **EXP/MGX**, the opposite is done. The initial sizes of the populations are set between 20 and 2 000, and the **size of population A following the expansion** is $Ne_{A'} = Ne_A / scalar\_N$, for an actual number of individuals **between 1 000 and 10 000**. For the **timing of these demographic events**, as well as the **timing of the split** of the ancestral population into population A and B, to make sense, they are **randomly determined based on the effective size of population A**. The **split happened between 1 and 8 x $Ne_A$ generations ago** and the **demographic event** happened within a time frame of **0.05 to 0.5 x $t_{split}$ generations ago**, with $t_{split}$ the timing of the split.

As stated before, individuals are simulated as **genomes of length *L* = 100 000 bp** with an **homogeneous recombination rate *r*** ranging from 1e-6 recombination per generation per bp. A set number of **2 000 neutral SNPs** are simulated in each population, and, for the simulations with selection, the **position of the beneficial mutation** from which the selective sweep originated is **randomly drawn** from all possible positions on the genome. The **selective coefficient *s*** is set such that the product $Ne_A.s$ **is between 10 and 100**. The **timing of the apparition of this mutation** also depends on $t_{split}$, and the mutation **occurs between 0.05 to 0.5 x $t_{split}$ generations ago**. Hence, the **demographic event and the rise of the beneficial mutation can happen in any order during the simulations**. Finally, for the migration, **migration rate *m* is set so** $4.Ne_A.m = 0.2$ for the simulations with gene flow, and equals to **zero otherwise**. **Figure 13** below shows the example of an EXP scenario. A detailed breakdown of the parameters used for the simulations can be found in **Supplementary Table 1**.

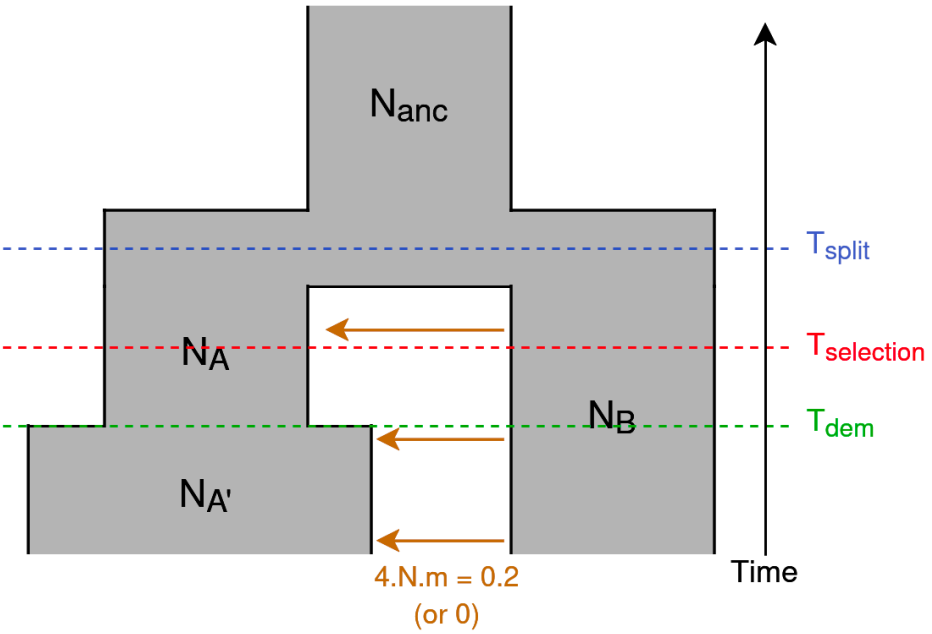



**Figure 13** - Example of an EXP scenario.

$N_{anc}$ is the ancestral population's size. $N_A$, $N_{A'}$ and $N_B$ are effective population sizes of population A before the demographic change, after the demographic change, and of population B respectively. The orange arrows indicate migration from population B to population A, with a migration rate set such as $4.N.m = 0.2$ (scenarios with migration) or $4.N.m = 0$ (scenarios without migration). $T_{split}$ is the generation at which the ancestral population splits into population A and population B. $T_{selection}$ is the generation at which the selection dynamic starts *i.e* the generation at which the mutation starts to be beneficial in population A. Finally, $T_{dem}$ is the generation at which the demographic event (*in this example the expansion*) occurs.

A total of 10 000 simulations of each of the three basic demographic scenarios, with or without migration and in presence and absence of a selective sweep are generated, for a total of 60 000 simulations with selection (the *sweep* dataset) and 60 000 simulations without selection (the *neutral* dataset). To construct the datasets, those simulations are split into 3 sub-types of datasets : 70% are used for the *train* dataset, 20% for the *validation* dataset and the remaining 10% are used for the *test* dataset. Each dataset contains an equal amount of simulation of each demographic and genetic scenario. *Train* and *valid* datasets are used during the training phase of the neural networks, while *test* datasets remain untouched until the end of the training phase and are used to have a first look at the performances on never seen before data.

# Data processing

At the end of each simulation 40 genomes are sampled from population A and a .ms formatted file is created. Such files contain the position and state (encoded as 0 or 1) of the 2 000 SNPs in the genomes of the sampled individuals (**Figure 14**). Bash and python scripts are used to process those files and generate two types of data used for the training of the CNN architectures : *'rawData'* and *'sumStats'*.

| | positions | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0.11727 | 0.35370 | 0.49882 | 0.58885 | 0.69909 | 0.70244 | 0.73685 | 0.83413 | 0.89938 | 0.95764 |
| Ind. 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| Ind. 2 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| Ind. 3 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| Ind. 4 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Ind. 5 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| Ind. 6 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| Ind. 7 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| Ind. 8 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| Ind. 9 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| Ind. 10 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

**Figure 14** - Representation of an .ms file content.

Example extract of the data contained inside a .ms formatted file. Here are 10 SNPs positions associated with the corresponding allele state (encoded as 0 or 1) of each individual of a sample of 10.

*rawData* data is in essence just the content of the .ms formatted files. Each element of such a dataset is a matrix where each line corresponds to one genome (*i.e.* one individual) and each column corresponds to a SNP. In other words, it is a representation of the states of the SNPs of the sampled individuals. Because all mutations are simulated as biallelic, the matrices can be converted into black and white pictures, with a 0 represented by a white pixel, and a 1 represented by a black pixel (**Figure 15 - A**). This way of representing input data, as well as the extra-step of sorting the chromosomes by genetic similarity, have been shown to improve the CNNs results (Flagel et *al*., 2019). Thus, a non-randomly ordered version of each matrix, referred to as "*sorted-rawData*" files, is also created : bins of simili-haplotypes of 20 consecutive SNPs are sorted by decreasing frequency along the genomes (**Figure 15 - B**).

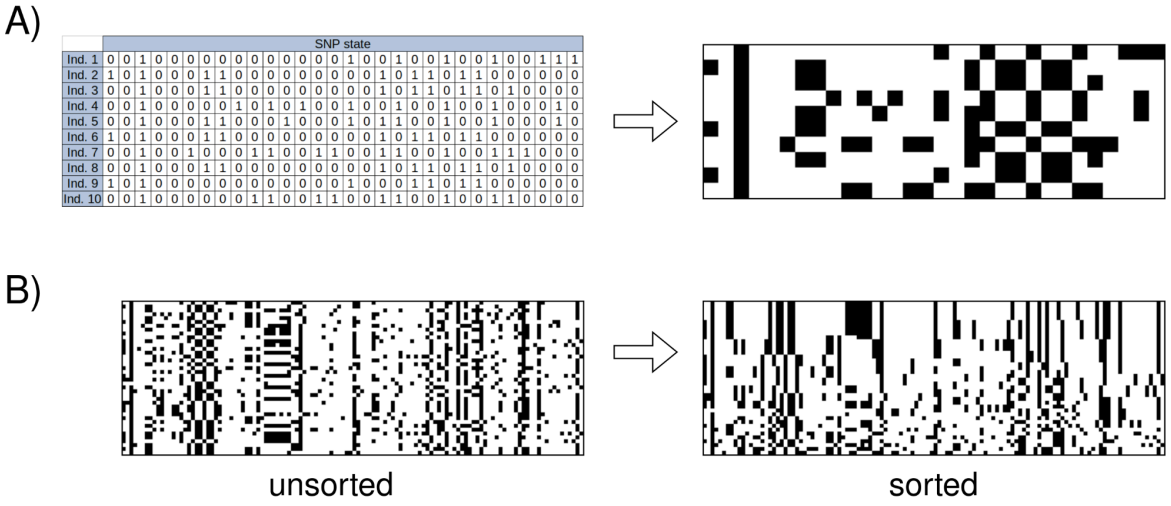



**Figure 15** - Representation of the *rawData* data inputs.

**A)** The matrices in .ms format are converted into black and white pictures. Each line corresponds to one individual, each column to one SNP position. Each pixel is thus the state of the allele at this position for the corresponding individual (0 = white, 1 = black). Represented here is a basic example matrix. **B)** Two types of *'rawData'* matrices are created. The *'unsorted*' ones are representations of the genomes in the random order they've been sampled. The '*sorted-rawData*' ones are matrices where the genomes have been splitted into bins of 20 consecutive SNPs and the resulting pseudo-haplotypes have been sorted by decreasing frequency. In that case, one line no longer corresponds to one individual.

The *sumStats* data inputs are matrices of summary statistics: we compute summary statistics in overlapping windows of 1 000 bp sliding along the genomes, by steps of 500 bp. A total of 14 summary statistics (detailed in **Supplementary Table 2**) are computed to capture various aspects of the genetic diversity. Half of them focus on nucleotide diversity: the average and standard deviation of the nucleotide diversity π (Nei & Li, 1979) computed inside each window, Watterson's estimator θ (Watterson, 1975), Tajima's D (Tajima, 1989), Achaz's Y (Achaz, 2008) and the Pearson r and p-value for π. The other half are more focused on haplotype diversity : the number of haplotypes, four different measures of haplotype's homozygosity H1, H2, H12 and H2/H1 (Garud et al., 2015 ; Harris et al., 2018) and two different measures of linkage disequilibrium, either measured by D (Lewontin, 1963) or by r2 (Hill & Robertson, 1968). The values of all these summary statistics are then normalized across the entire dataset, in order to highlight the variations inter-genomes, and converted into a matrix with each line corresponding to a specific statistics, and each column to one window (**Figure 16**). Doing so, we can work using normalized pictures while preserving the demographic and selection patterns in the genomes.


**Figure 16** - Representation of the *sumStats* data inputs.

Matrices of summary statistics computed inside 1 000 bp overlapping windows sliding along the chromosome in steps of 500 bp. From top to bottom : π (average), π (standard deviation), Watterson's θ, Tajima's D, Achaz's Y, Pearson's r for π, Pearson's p-value for π, number of haplotypes, haplotype's homozygosity H1, haplotype's homozygosity without the most common haplotype (H2), haplotype's homozygosity considering the two most common haplotype as the same (H12), ratio of H2 over H1, linkage disequilibrium measured by D and linkage disequilibrium measured by r2. Values are normalized between 0 and 1, based on the minimum and maximum values of the whole dataset, and each pixel gray-scaled accordingly.

## Convolutional Neural Networks architectures

Three different types of CNN architecture are tested for this classification task. The first type is a simple custom CNN architecture (referred to as *'simple'* CNN) consisting of a succession of *n* convolutional layers, *i.e.* a convolution followed by a ReLU activation, some dropout to avoid overfitting, a 2D batch normalization and a 2D Max Pooling, used to extract the features of the input data. The output of this convolution phase is flattened and passed to a classic dense fully connected output layer to perform the classification task (**Figure 17 - A**). The second type of architecture, referred to as *'mix*' CNNs, is, in short, just a small addition to *simple* CNNs. A basic sequential network is built to handle a newly added information: a list of the genomic positions of the SNPs. The outputs of both the convolution phase and this new small network are concatenated together and used for classification (**Figure 17 - B**). The third type of architectures are what we call '*pre-trained*' CNNs: architectures available online that have already been trained on other datasets. We focus our attention on two of them: resNet (Residual Network, He et *al*., 2015) and efficientNet (Le & Tan, 2020). resNet was created to address challenges of training deep neural networks, particularly the issues of vanishing gradients and accuracy degradation in very deep models. For each layer, instead of learning the target mapping *i.e.* the correct function that would, for any given input,

generate the desired output :

$$H(x) = y$$

with *x* the input of the layer, and *y* the output, resNet introduces residual connections (shortcut connections) that allow the network to learn a residual function :

$$F(x) = H(x) - x$$

thus reformulating the mapping as :

$$H(X) = F(x) + x$$

These shortcut connections help stabilize gradient propagation, allowing for much deeper networks while improving accuracy (He *et al.*, 2015).

As for efficientNet, it is a family of CNNs designed to optimize model scaling by balancing depth, width, and resolution through a compound scaling method. Tan and Le (2020) introduced EfficientNet-B0 as a baseline model, constructed using Mobile Inverted Bottleneck (MBConv) layers (Sandler *et al.*, 2018). This approach has demonstrated state-of-the-art performance on ImageNet, a large-scale dataset containing over 14 million images across 1 000 object categories, widely used for benchmarking image classification models, and has shown strong generalization capabilities for transfer learning tasks on both CIFAR-10 and CIFAR-100, which consist of 60 000 images of 32×32 pixels across 10 and 100 classes, respectively. Notably, efficientNet-B0 achieves comparable accuracy while being 2 to 5 times smaller than ResNet-50 in parameter count (Le & Tan, 2020).

Indeed, the number of parameters varies across different model architectures and depends on the type of input data used. Our *simple* architectures using *sumStats* data have the smallest parameter count, approximately 76 000. In contrast, *simple* CNNs using *rawData* and *sorted-rawData* reach up to 2,8 and 1,8 millions parameters, respectively, at most. *mix* architectures exhibit a comparable range, with parameter counts between 1,3 and 2,9 millions. Finally, among the evaluated models, our implementation of resNet is the most complex, with up to 11,2 millions parameters, while our implementation of efficientNet contains slightly more than 4 millions parameters. Further details regarding the exact number of parameters for each architecture are provided in **Supplementary Table 3**.

As for the input data, *rawData, sorted-rawData* and *sumStats* training datasets are used for the *simple* and *pre-trained* CNNs. *Mix* CNNs only use *rawData* and *sorted-rawData* with the

added information of the position of each SNPs along the genomes. Early attempts to use *mix* CNNs architecture with *sumStats* have yielded poor results, never surpassing random guess, and have thus been discarded. Finally, among each architecture type, different sizes for the convolution kernels have been tested, referenced as "k*n*" (with *n* the horizontal dimension of the kernel, the vertical dimension being equals to the input matrix height) for the rest of this thesis.

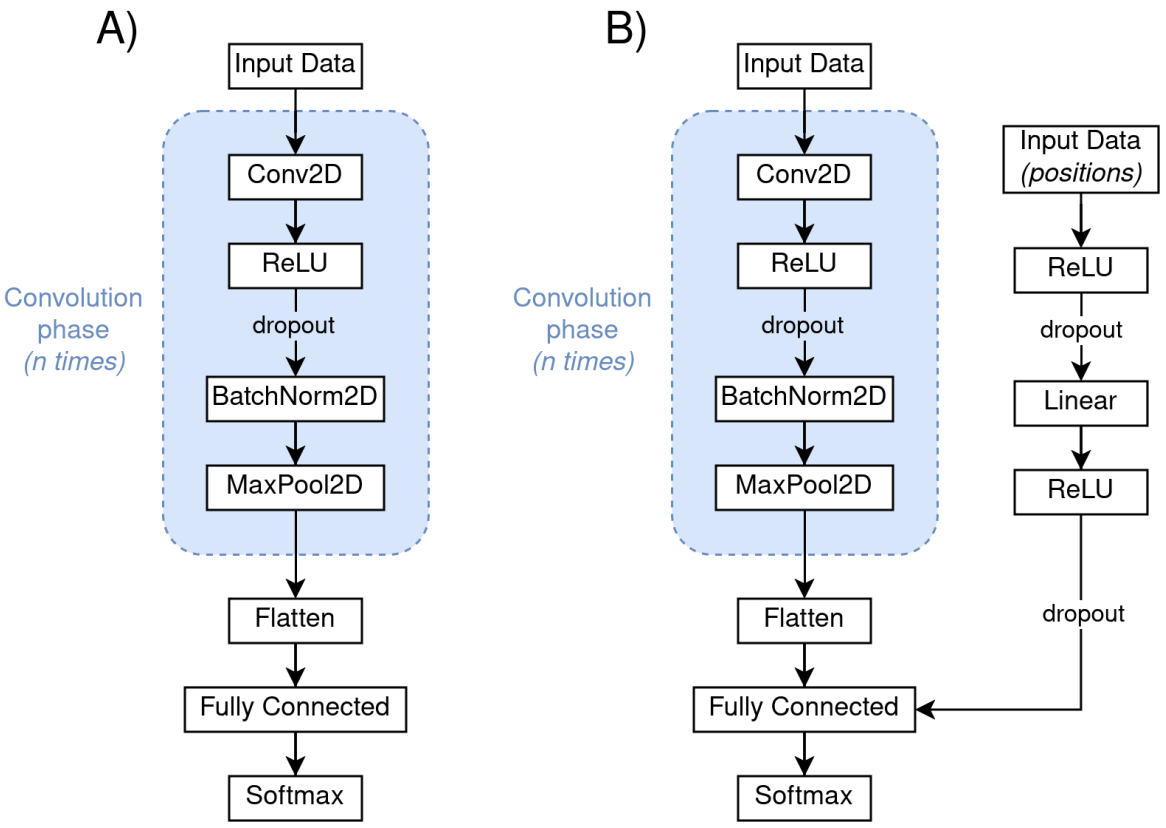



**Figure 17** - Architectures of the two homemade CNNs used for the classification task.

**A)** A simple architecture using only matrices (either *rawData* or *sumStats*) as input data. **B)** A slightly more complex architecture using both matrices and SNPs positions information as input data. Two separate networks process the two different types of input data. The outputs of each are then concatenated together before being passed to a fully connected network to perform the classification.

## Convolutional Neural Networks - Training phase

The goal of this training phase is to tune the CNN weights and biases. By going through each example of the *train* dataset multiple times during the training phase; a few examples are fed to the network which tries to produce an output. The error gradient associated with those predictions is computed and is propagated in reverse through the network using the

backpropagation algorithm (Rumelhart et *al*., 1986), in order to update the weights and biases thanks to stochastic gradient descent (Kingma & Ba, 2014). Once all the examples of the *train* dataset have been used one time, one *epoch* has passed. The network performance is evaluated using the *validation* dataset, to monitor the evolution of the accuracy and loss (a measure of the error of a network) at each step of the training. The training is complete once the network has been through a set number of epochs, or when certain criterions have been reached.

In our study, we train a total of 7 different architectures : 3 *simple* ones (CNN simple k7, CNN simple k9 and CNN simple stats, using respectively a kernel with a width of 7 or 9 on *rawData* for the two first, and working on *sumStats* for the latter) ; 2 *mix* ones (CNN mix k7 and CNN mix k9, once again working with kernels of width 7 or 9) and 2 pre-trained CNNs (resNet and efficientNet). Each of those 7 architectures are trained on the different possible combinations of input data and of genetic and demographic scenarios: *simple* and *pre-trained* CNNs are trained on *rawData*, *sorted-rawData* and *sumStats*, while *mix* CNNs are only trained on the first two input data types. While this phase constitutes the first time the *simple* CNNs are trained, the *pre-trained* architectures have already been previously trained. Indeed, the goal of using such architectures is to build on an already existing base knowledge by slightly tuning the last layers of already trained neural networks through a new training phase in order for them to learn to perform a new task. Both pre-trained architectures have been trained on the ImageNet dataset (IMAGENET1K_V1 from the PyTorch library).

Each CNN is trained using the *train* and *validation* datasets, containing respectively 7 000 and 2 000 examples of each of the three basic demographic scenarios (CST, BTL and EXP) formatted in the three types of input data (*rawData*, *sorted-rawData* and *sumStats*) for both *sweep* and *neutral* variants. All the training are done with and without data augmentation, a way to artificially increase the training example variability by slightly altering ‘on the fly’ the input data during the training phase. The networks are trained for 100 epochs, using batches of size 32, without any specific criteria to stop the training earlier. A complete list of all the parameters used for the training of each CNN, as well as accuracy and loss scores is available in **Supplementary Table 4 & 5**.

## ABC - Random Forest (ABC-RF)

As presented in the Introduction of this thesis, ABC are computational methods based on approximation of the posterior distribution of a model parameters, allowing for inferences when models are too complex for a likelihood to be computed, but when it is still possible to run simulations from this model. Two major issues arise with this method. First, a huge number of simulations are needed, thus making it computationally heavy ; which is even more of an issue with genomic data with tens of thousands of possible data points). Secondly, they require a calibration phase, often a critical phase of using such a method, where choices must be made to decide which threshold to use, or with which statistics to work. However, a new method has arisen during the last decade : a combination of ABC and a tool inspired from deep-learning, the random forests (Breiman, 2001; Pudlo et al., 2015 - “Reliable ABC model choice via random forests”)

ABC require four steps : the simulation of datasets from parameters randomly drawn from the posterior distributions, the computation of summary statistics from both the observed and the simulated datasets, the evaluation of the distance between both sets of summary statistics computed and finally, the comparison of those distances with a threshold value to accept or reject the approximation. Random Forests are methods used for classification and regression. They are based on the construction of a huge number of decision trees which will then collectively make a prediction : all individual predictions are gathered and the prediction of the forest is the one predicted by the highest number of trees.

The same *sumStats* datasets used to train the CNNs architectures are used with ABC-RF, as training it using the *rawData* datasets would be too computationally heavy and not make much sense. However, we also tried a small optimization for the ABC-RF methods : instead of only using *sumStats* datasets as is, we tried ABC-RF on extremely summarized *sumStats* datasets where only the average value of each summary statistics is used.

# Results

We propose a comparison between multiple possibilities for the set up of a supervised deep-learning approach for the classification of genomic alignments between plausible demographic scenarios. Below, we compare the results of *simple*, *mix*, and *pre-trained* CNNs architectures. We evaluate the performance of each architecture when using summary statistics or raw genomic data as input, the robustness of the CNNs in case of model misspecification or unexpected gene flow and compare the results against another classification method, the ABC-RF.

## Metrics and evaluation of model performance

A quick explanation of some technical aspects of the comparison of CNN approaches is important to understand how we decide which CNN has performed the best. Two main metrics are used to evaluate a model performance : the **accuracy** and the **loss**.

**Accuracy** depicts the fraction of prediction that the model got right out of all the predictions it does, thus ranging between 0 and 1. Formally, accuracy can be defined as :

$$accuracy = \frac{CP}{CP + WP}$$

with CP the number of correct predictions and WP the number of wrong predictions.

The **loss** is a method of evaluating if the model's predictions are far from the truth. Its values depend on the selected "loss function". While the Mean Square Error (MSE) loss function is one of the most common ones for regression problems, here we use a Cross-Entropy loss function, better suited to classification problems. The value of cross-entropy loss decreases as the predicted probability of the correct label rises. Formally, the cross-entropy loss $L$ is defined as :

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i . log(\widehat{y}_i)$$

with $m$ the number of samples in the dataset, $i$ the i-th element of the dataset, $y_i$ the expected value for the i-th element and $\widehat{y}_i$ the predicted value for the i-th element.

Another commonly used method of evaluating a model performance are **Receiver Operating Characteristic (ROC) curves**. ROC Curves show the trade-off for different thresholds, by comparing True Positive Rate (TPR) on the y-axis and False Positive Rate (FPR) on the x-axis. While typically used for binary classification, it is still possible to use ROC curves for a multi-class classifier as in our case by binarizing the output for each class, in a One-vs-Rest method *i.e.* by comparing each class (TP) against all the others (FP). The **Area Under the Curve (AUC)** helps to summarize the overall performance of the model : a higher AUC value (closer to 1) indicates better model performance in distinguishing between classes.

Finally, while accuracy and loss are good metrics to evaluate a model performance, another useful tool is a confusion matrix. For each test example, the predicted scenario is compared to the expected (true) scenario which it belongs to. It allows for an easier visualization of the proportion of correct and incorrect predictions for each class.

The choice for the best CNN for each comparison is done by comparing the accuracy and loss scores obtained on both *neutral* and *sweep test* datasets. Unless otherwise stated, a CNN architecture is considered better if its loss is closer to 0 and its accuracy closer to 1.

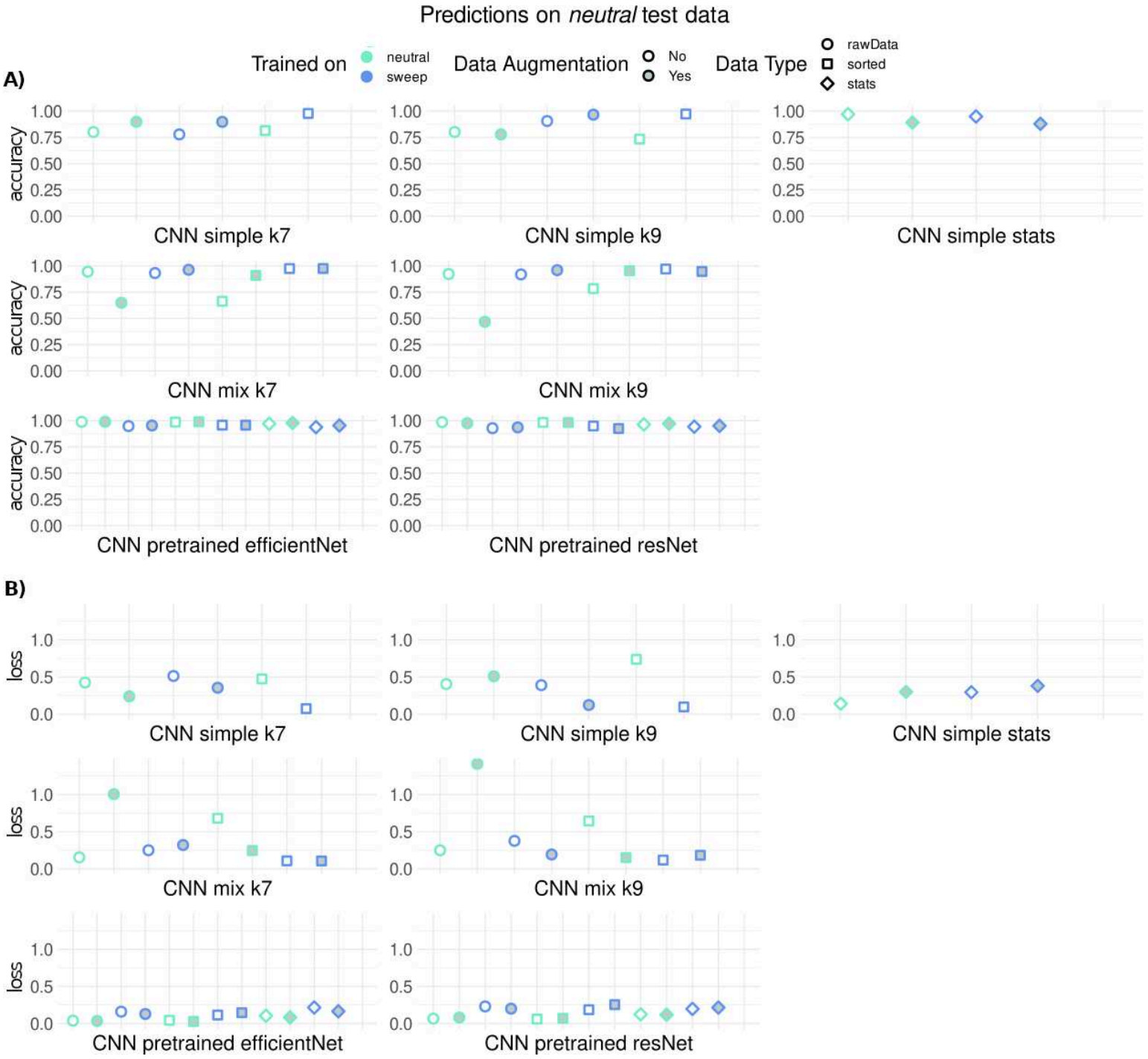



**Figure 18** - Evaluation of classification performances of the CNN tested on *neutral* datasets.

CNN trained on *neutral* datasets are displayed in green, CNN trained on *sweep* datasets are in blue. Data augmentation is implemented during training as a random chance for any input data to be slightly altered (horizontally or vertically flipped) thus increasing the diversity of training examples. Data type corresponds to the type of input data used by the network (either *rawData*, *sorted-rawData* or *sumStats*). The tested CNNs architecture names indicate the type, between *simple*, *mix* or *pre-trained* architectures. **A)** shows accuracy values and **B)** loss values.

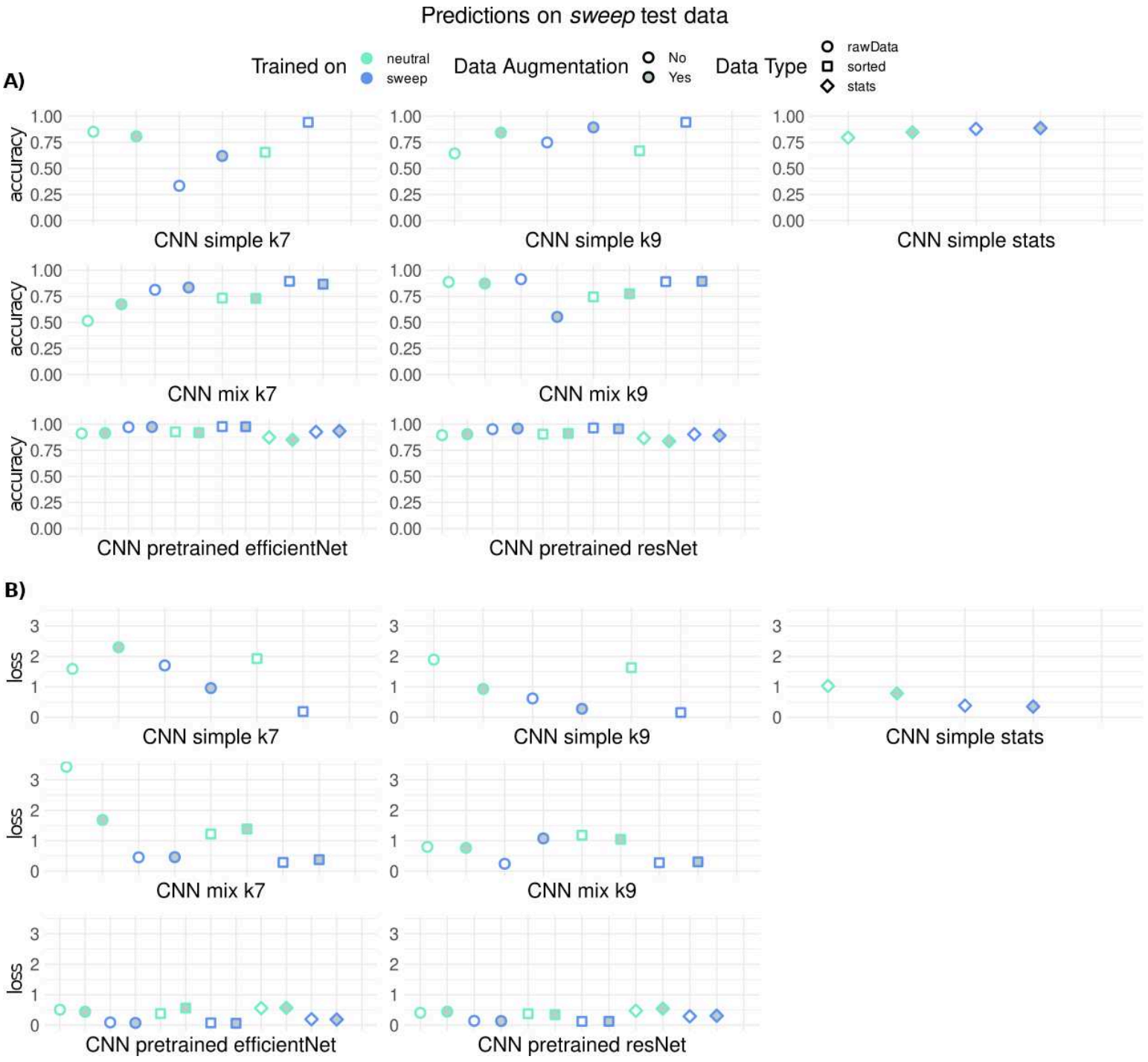



**Figure 19** - Evaluation of classification performances of the CNN tested on *sweep* datasets.

CNN trained on *neutral* datasets are displayed in green, CNN trained on *sweep* datasets are in blue. Data augmentation is implemented during training as a random chance for any input data to be slightly altered (horizontally or vertically flipped) thus increasing the diversity of training examples. Data type corresponds to the type of input data used by the network (either *rawData*, *sorted-rawData* or *sumStats*). The tested CNNs architecture names indicate the type, between *simple*, *mix* or *pre-trained* architectures. **A)** shows accuracy values and **B)** loss values.

# Comparison of the overall performances of different CNNs architectures

The performance of the three different CNN architectures on *test* datasets (containing 1 000 simulations of each scenario) is compared in order to determine a baseline ranking between them. As stated in the Material & Method part, two sets of training are conducted, one on the *neutral* datasets, and one on the *sweep* datasets, each containing balanced numbers of simulations of the three different basic demographic scenarios, BTL, CST and EXP. All three types of input data (*rawData, sorted-rawData* and *sumStats*) are tested as well. **Figure 18** shows the accuracy (**18 - A**) and loss (**18 - B**) obtained on *neutral test* dataset for all the combinations of input data, selection and data augmentation used for the training of each of the 3 types of CNN architectures and **Figure 19** shows the same for tests on *sweep* dataset.

## CNNs tested on *neutral* data

The highest accuracy and lowest loss, for the test on *neutral* data, are obtained by the same *pre-trained* CNN architecture, an efficientNet CNN train on neutral *sorted-rawData*, using data augmentation during the training. It achieves an accuracy of 0.9893 (tied with the efficientNet trained on neutral *rawData*) and a loss of 0.0299 (0.0358 for the one train on *rawData*). The lowest accuracy of 0.4677 and the highest loss of 1.4135 are also obtained by the same CNN, a *mix k9* architecture trained on neutral *rawData* using data augmentation. Overall, the *pre-trained* architectures performed better (with the highest average accuracy of 0.962 ± 0.0205 and the lowest average loss of 0.129 ± 0.0685) followed by the *simple* CNNs (with an average accuracy of 0.875 ± 0.0805 and an average loss of 0.341 ± 179) and the *mix* architectures slightly behind both in terms of accuracies (0.871 ± 0.15) and losses (0.388 ± 0.371). See **Table 1** and **Supplementary Table 4** for a more detailed breakdown of all CNN accuracy and loss values.

**Table 1** - Performance comparison of *simple, mix* and *pre-trained* CNNs architecture tested on *neutral* datasets. In bold, the highest accuracy and lowest loss.

| CNN | accuracy (average) | accuracy (range) | loss (average) | loss (range) |
|---|---|---|---|---|
| simple | 0.875 (± 0.0805) | [0.734, 0.978] | 0.341 (± 0.179) | [0.0735, 0.7379] |
| mix | 0.871 (± 0.15) | [0.468, 0.975] | 0.388 (± 0.371) | [0.1063, 1.4135] |
| pre-trained | 0.962 (± 0.0205) | [0.9247, 0.9893] | 0.129 (± 0.0685) | [0.0299, 0.2552] |

## CNNs tested on *sweep* data

Two pre-trained CNNs are nearly tied for the rank of best CNN on *sweep* test data. Both are efficientNet trained on *sweep*, *sorted-rawData* with the only difference being the use or not of data augmentation. The CNN trained with data augmentation achieves an accuracy of 0.9760 and a loss of 0.0618 while the CNN trained without data augmentation scores a slightly better accuracy of 0.9770 but a worse loss of 0.0747. Simply going with the lowest difference in accuracy or loss as the deciding factor, we consider the CNN trained with data augmentation as being the best of the two. While the worst loss of 3.4280 is once again obtained by a *mix* CNN (mix k7, trained on *neutral rawData* without data augmentation), a *simple* architecture achieves the worst accuracy of 0.333. Such an accuracy is, in fact, not different from the expected results of a random guessing, so we chose to discard entirely this specific CNN (*simple* k7, trained on *sweep rawData* without data augmentation). From there, the worst accuracy then became 0.5143 and is the accuracy of the same *mix* CNN achieving the highest loss of 3.4280. The ranking on *sweep* data remain the same than the one obtained on *neutral* data: the *pre-trained* architectures are the best ones, with the highest average accuracy (0.921 ± 0.0402) and the lowest loss (0.308 ± 0.18), followed by the *simple* CNNs (accuracy = 0.801 ± 0.109, loss = 0.998 ± 0.707) and *mix* architecture (accuracy = 0.787 ± 0.123, loss = 0.937 ± 0.804) having very similar average results. See **Table 2** and **Supplementary Table 5** for a more detailed breakdown of all CNN accuracy and loss values.

**Table 2** - Performance comparison of *simple, mix* and *pre-trained* CNNs architecture tested on *sweep* datasets. In bold, the highest accuracy and lowest loss.

| CNN | accuracy (average) | accuracy (range) | loss (average) | loss (range) |
|---|---|---|---|---|
| *simple* | 0.801 (± 0.109) | [0.6193, 0.9427] | 0.998 (± 0.707) | [0.1537, 2.2981] |
| *mix* | 0.787 (± 0.123) | [0.5143, 0.914] | 0.937 (± 0.804) | [0.2415, 3.428] |
| *pre-trained* | 0.921 (± 0.0402) | [0.8367, 0.977] | 0.308 (± 0.18) | [0.0618, 0.5678] |

## Overall best architecture on *neutral* or *sweep* datasets

To conclude this first look at the various CNN architecture's performances, we take a look at the best CNN of each type, on either *neutral* or *sweep* datasets (**Table 3** and **Supplementary Table 4 & 5**). The ROC curves obtained on *neutral* datasets and on *sweep* datasets are presented in **Supplementary Figure 1 and 2**.

On *sweep* test data, the best simple CNN is the *simple* k9, using *sweep sorted-rawData* without data augmentation, the best mix CNN is the *mix* k9, using *sweep rawData* without data augmentation, and the best pre-trained CNN, which is also the overall best CNN on *sweep* test data, is the *efficientNet* using *sweep sorted-rawData*, trained with data augmentation. On *neutral* test data, while the best CNN remains the *efficientNet* using *neutral sorted-rawData* and trained with data augmentation (also being the overall best CNN on *neutral* test data), something interesting is to be noted for both the best *simple* and the best *mix* architectures. The best simple is the *simple k7*, using *sweep sorted-rawData* without data augmentation, and the best mix is the *mix k7,* using *sweep sorted-rawData* without data augmentation. For comparison, the best pre-trained CNN trained on *sweep* data and tested on *neutral* data is the *efficientNet* using *sorted-rawData* without data augmentation, which scores an accuracy of 0.957 and a loss of 0.115, which are results not too far from the *simple* and *mix* CNNs.

**Table 3** - Comparison of the overall performance of *simple, mix* and *pre-trained* CNNs architecture on *neutral* and *sweep* test datasets. In bold, the highest accuracy and lowest loss for each category (*neutral / sweep*) of test data.

| Test Data | CNN type | Model | Input Data | Train Data | Data Augmentation | Accuracy | Loss |
|---|---|---|---|---|---|---|---|
| *neutral* | *simple* | k7 | *sorted-rawData* | *sweep* | No | 0.977 | 0.0735 |
| | *mix* | k7 | *sorted-rawData* | *sweep* | Yes | 0.975 | 0.1063 |
| | ***pre-trained*** | ***efficientNet*** | ***sorted-rawData*** | ***neutral*** | **Yes** | **0.989** | **0.0299** |
| *sweep* | *simple* | k9 | *sorted-rawData* | *sweep* | No | 0.943 | 0.154 |
| | *mix* | k9 | *rawData* | *sweep* | No | 0.914 | 0.242 |
| | ***pre-trained*** | ***efficientNet*** | ***sorted-rawData*** | ***sweep*** | **Yes** | **0.976** | **0.0618** |

## CNN architecture comparison

After the first overall comparison of performances, we now compare the CNN architectures in a more detailed manner, one element at a time. To do so, we begin by focusing on the impact of the architecture use and only focus on the CNNs trained and tested on *neutral* datasets. The best *simple* CNN is the *simple k3* using *sumStats* without data augmentation (accuracy = 0.9657, loss = 0.1422), the best *mix* CNN is the *mix k9* using *sorted-rawData* with data augmentation (accuracy = 0.9540, loss = 0.1507) and the best *pre-trained* CNN is the *efficientNet* using *sorted-rawData* with data augmentation (accuracy = 0.9893, loss = 0.0299).

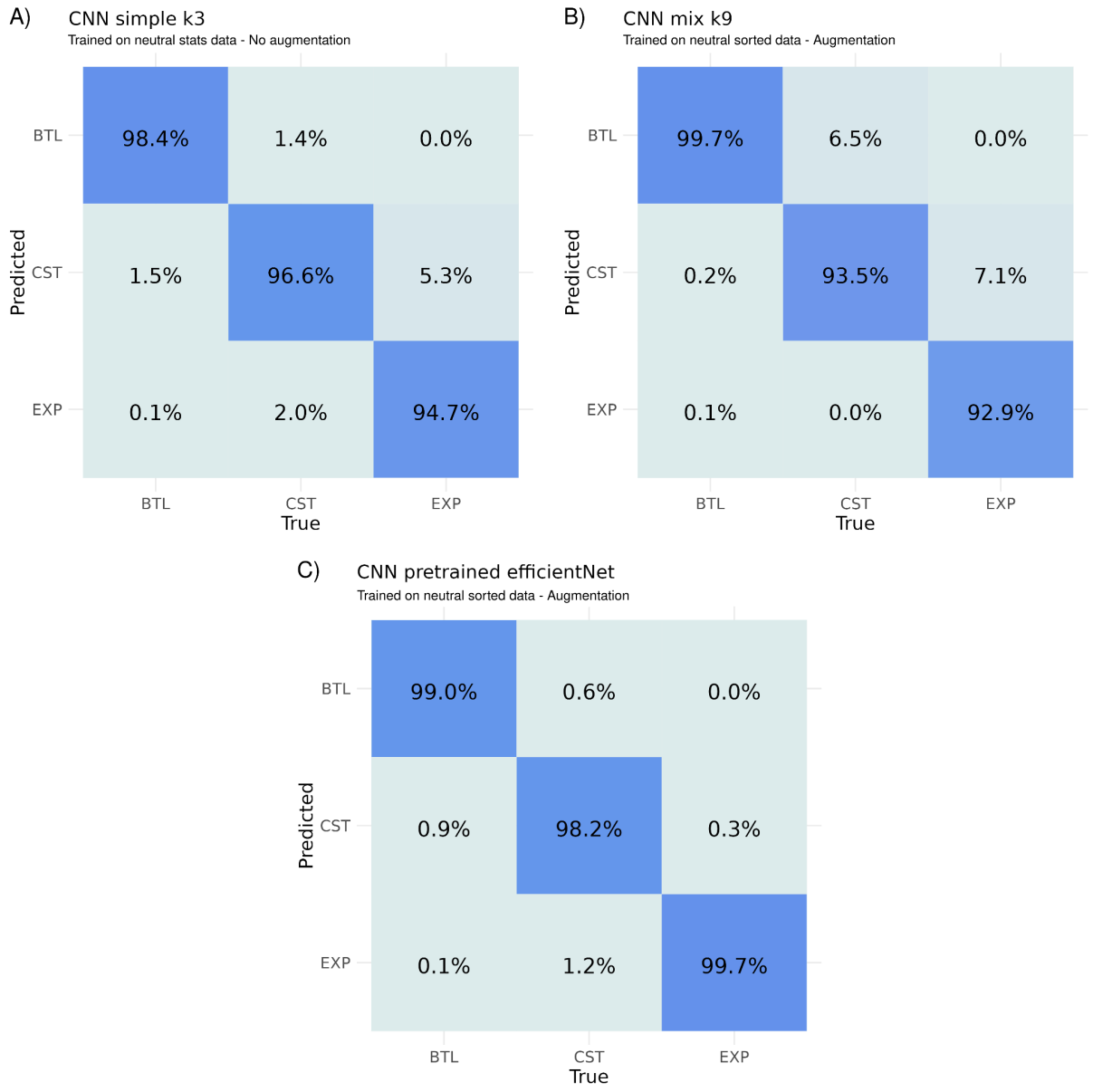


**Figure 20** - Confusion matrices of the best CNN of each type of architecture, trained and tested on *neutral* datasets.

The x-axis is the correct (true) scenario and the y-axis the predicted scenario. Predictions on the diagonal are correct predictions. The *test* datasets used contain 1 000 simulations of each of the three classes BTL, CST and EXP. **A)** Results for the best *simple* CNN architecture, **B)** results for the best *mix* CNN architecture and **C)** results for the best *pre-trained* CNN architecture.

Confusion matrices are shown in **Figure 20** for the best performing CNN for the *simple* (**20 - A**), *mix* (**20 - B**) and *pre-trained* (**20 - C**) CNN. Out of the three, the *mix* CNN displays the lowest accuracy and the worst results on the classification of CST and EXP simulations, with respectively “only” 93.5% and 92.9% of them correctly classified. The *simple* CNN is the worst at classifying BTL simulations, with “only” 98.4% of them correctly classified. The *pre-trained* CNN shows more than 98% of correct classifications regardless of the scenario. BTL or EXP scenarios are nearly never mistaken for one another, with one single BTL simulation wrongly classified as an EXP. BTL simulations are also rarely mistaken for CST simulations, with the simple CNN making this mistake the most in 1.5% of the cases. About 1.5 to 2% of the CST simulations are wrongly classified as either BTL or EXP by the *simple* CNN, and 0.6 to 1.2% by the *pre-trained* one. The *mix* CNN seems to never classify CST simulations as EXP, but up to 6.5% of them are mistaken for BTL simulations. As for the EXP simulations, no CNN classifies them as BTL, but the *simple* CNN is wrong and classifies 5.3% of them as CST and the *mix* CNN is wrong for 7.1% of them.

The same approach is applied on CNN trained and tested on *sweep* datasets. In that case, the three best CNNs are the same ones as in **Table 3**: the *simple k9* using *sorted-rawData* without data augmentation (accuracy = 0.943, loss = 0.154), the *mix k9* using *rawData* without data augmentation (accuracy = 0.914, loss = 0.242) and the *efficientNet* using *sorted-rawData* with data augmentation (accuracy = 0.976, loss = 0.0618). The corresponding confusion matrices are displayed in **Figure 21.** Once again, the *mix* CNN is the worst, with the lowest accuracy and the highest loss of the three. The *pre-trained* one is also once again the best, with the most error it does being 2.2% of the EXP simulations predicted as CST and 1.2% predicted as BTL. 1.7% of the CST simulations are predicted as EXP and the same amount of BTL are predicted as EXP too. As for the *simple* and *mix* CNN mistakes, both wrongly classify more than 10% of the BTL simulations as EXP. The *simple* CNN miss-classifies 4.1% of the CST simulations as EXP, and the *mix* classifies 9.7% of the EXP simulations as CST.

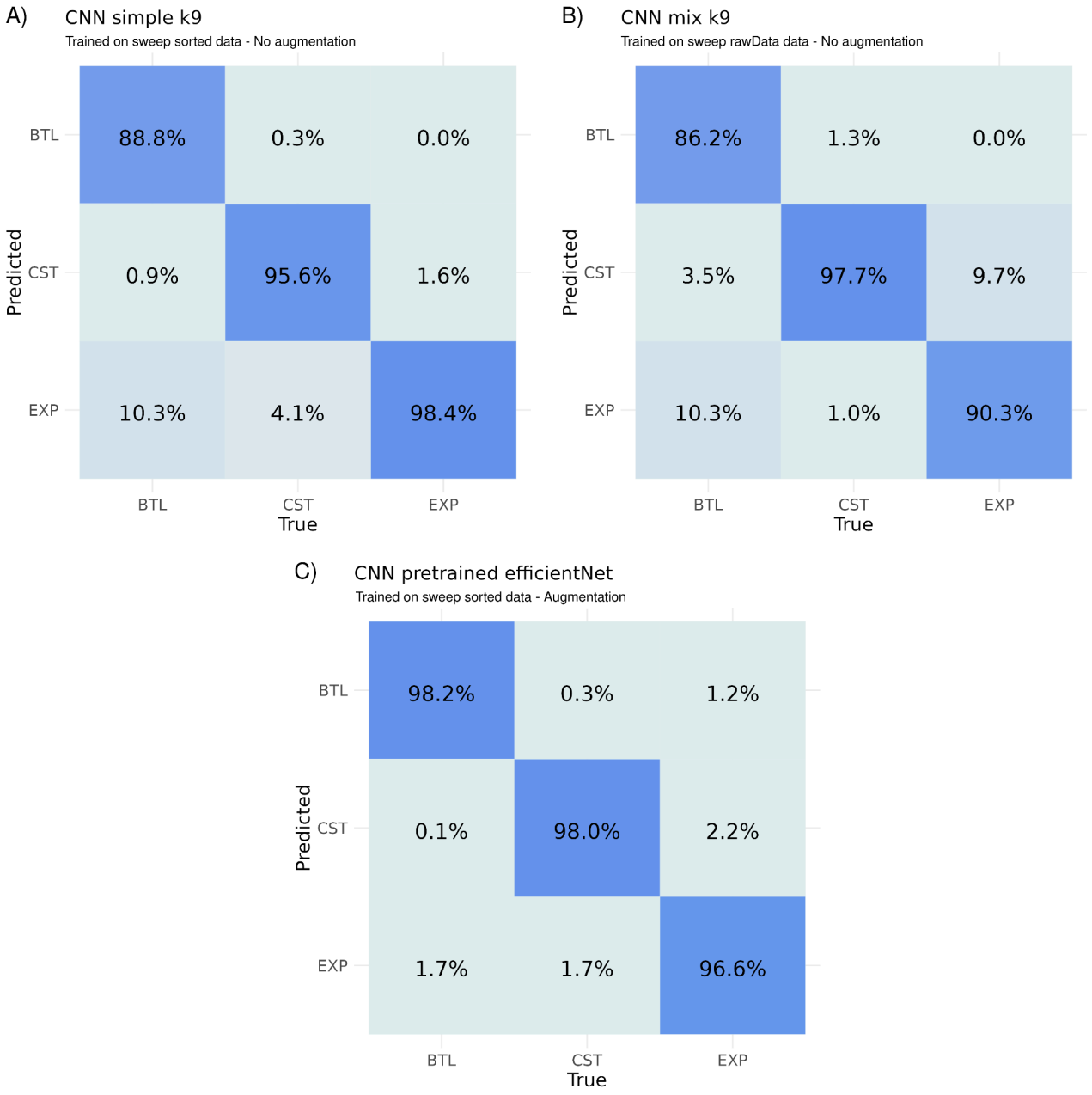



**Figure 21** - Confusion matrices of the best CNN of each type of architecture, trained and tested on *sweep* datasets.

The x-axis is the correct (true) scenario and the y-axis the predicted scenario. Predictions on the diagonal are correct predictions. The *test* datasets used contain 1 000 simulations of each of the three classes BTL, CST and EXP. **A)** Results for the best *simple* CNN architecture, **B)** results for the best *mix* CNN architecture and **C)** results for the best *pre-trained* CNN architecture.

## Input data type comparison

The comparison between the two types of input data, *rawData* (*rawData* + *sorted-rawData*) and *sumStats* is done by comparing the results between *simple* and *pre-trained* architecture only, as the idea of using *mix* architecture used with *sumStats* has been discarded early due to the very poor results of this approach. As before, the CNNs are first trained and tested on the *neutral* dataset and the results of the best of each architecture compared, before going on to do the same on the *sweep* dataset.

**Table 4** - Comparison of the effect of **input data** in *simple* and *pre-trained* CNNs architecture. In bold, the highest accuracy and lowest loss for each category.

| Test & Train Data | CNN type | Model | Input Data | Data Augmentation | Accuracy | Loss |
|---|---|---|---|---|---|---|
| *neutral* | *simple* | k7 | *rawData* | Yes | 0.899 | 0.2387 |
| | | **k3** | ***sumStats*** | **No** | **0.9657** | **0.1422** |
| | *pre-trained* | ***efficientNet*** | ***sorted-rawData*** | **Yes** | **0.989** | **0.0299** |
| | | *efficientNet* | *sumStats* | Yes | 0.9780 | 0.0836 |
| *sweep* | *simple* | **k9** | ***sorted-rawData*** | **No** | **0.943** | **0.154** |
| | | k3 | *sumStats* | Yes | 0.886 | 0.3392 |
| | *pre-trained* | ***efficientNet*** | ***sorted-rawData*** | **Yes** | **0.976** | **0.0618** |
| | | *efficientNet* | *sumStats* | Yes | 0.935 | 0.1862 |

On *neutral* datasets, the best *simple* CNN on *rawData* scores an accuracy of 0.899 and a loss of 0.2387 *vs*. 0.9657 and 0.1422 for the best *simple* CNN using *sumStats*. However, outside of this occurrence, using *sorted-rawData* as data input provide better results (higher accuracy and lower loss) for all other CNN architectures, *i.e. pre-trained* CNN on *neutral* datasets, as well as both *simple* and *pre-trained* CNNs on *sweep* datasets (**Table 4**).

The difference is quite visible when looking at the confusion matrices: focusing on the CNNs trained and tested on *neutral* datasets, while the *simple* CNN using *sumStats* display great classification performance overall, except for about 5% of the EXP simulations classified as CST (**Figure 22 - A**), the best *simple* CNN working using *rawData* mistakenly classifies up to 23% of the CST simulations as BTL (**Figure 22 - B**).

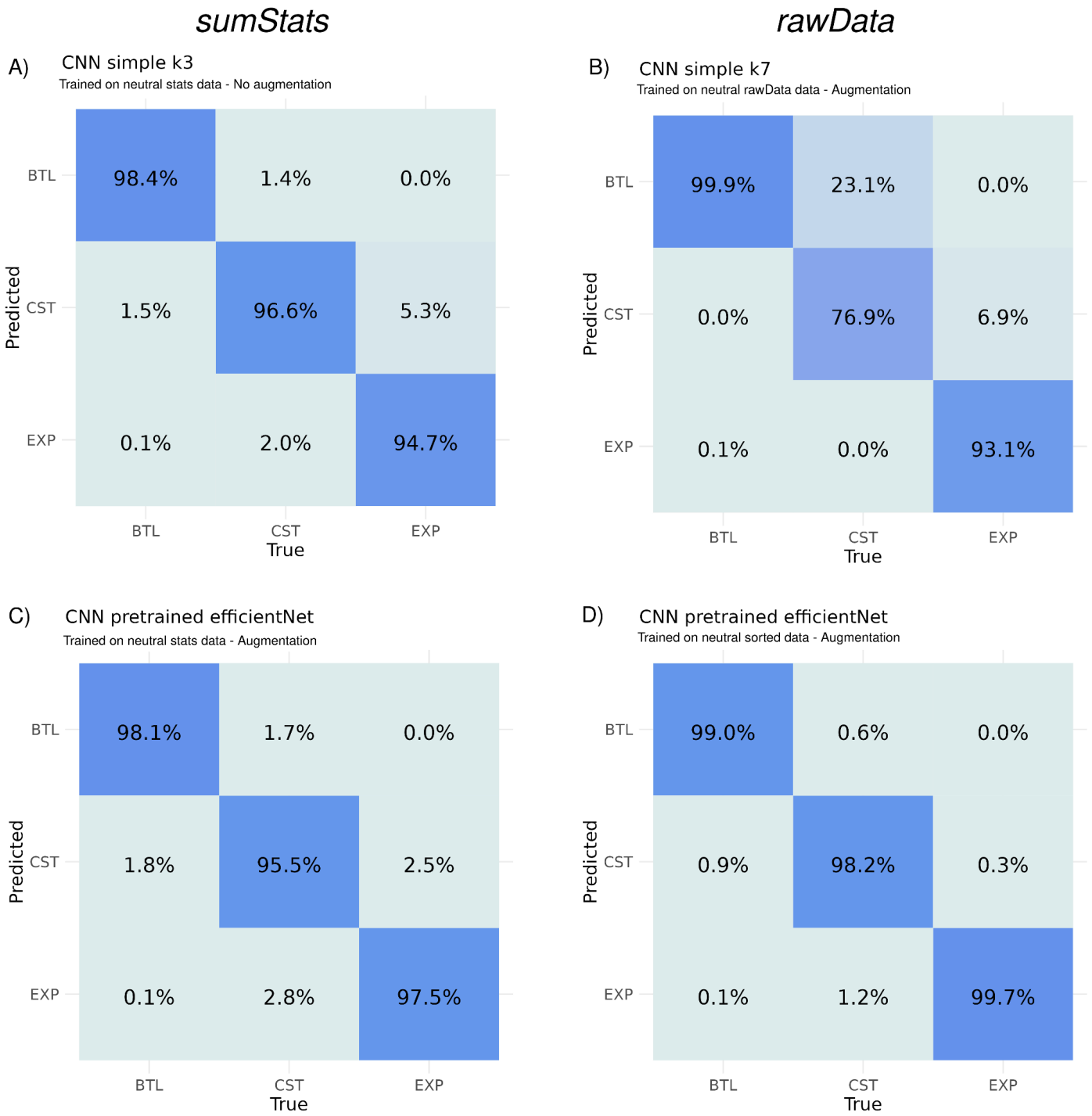



**Figure 22** - Confusion matrices of the best *simple* and *pre-trained* CNN regarding inferences on *neutral* datasets - comparisons of *rawData* vs *sumStats* input data.

**A)** best *simple* CNN architecture working on *sumStats* data, **B)** best *simple* CNN architecture working on *rawData* data, **C)** best *pre-trained* CNN architecture working on *sumStats* data and **D)** best *pre-trained* CNN architecture working on *rawData* data.

On the other hand, the results between *sumStats* and *rawData* for the *pre-trained* CNNs are very similar, despite a very slight advantage for the *(sorted-)rawData* one on the classification of all three scenarios (**Figure 22 - C & D**).

As for the CNNs trained and tested on *sweep* datasets, the same observations remain: the *simple* CNNs struggle with the classification of BTL simulations, wrongly classified as EXP in more than 10% of the cases for the *rawData* one, and up to 12% for the one using the *sumStats*. That same *sumStats simple* CNN also mistakenly classifies nearly 15% of the EXP simulations as CST, and up to 4.5% of the BTL as CST (**Figure 23 - A & B**). Meanwhile, the best *pre-trained* ones achieve more than 90% of correct classifications regardless of the type of input data or demographic scenario, with a slight advantage for the CNN trained on *sorted-rawData* (**Figure 23 - C & D**).

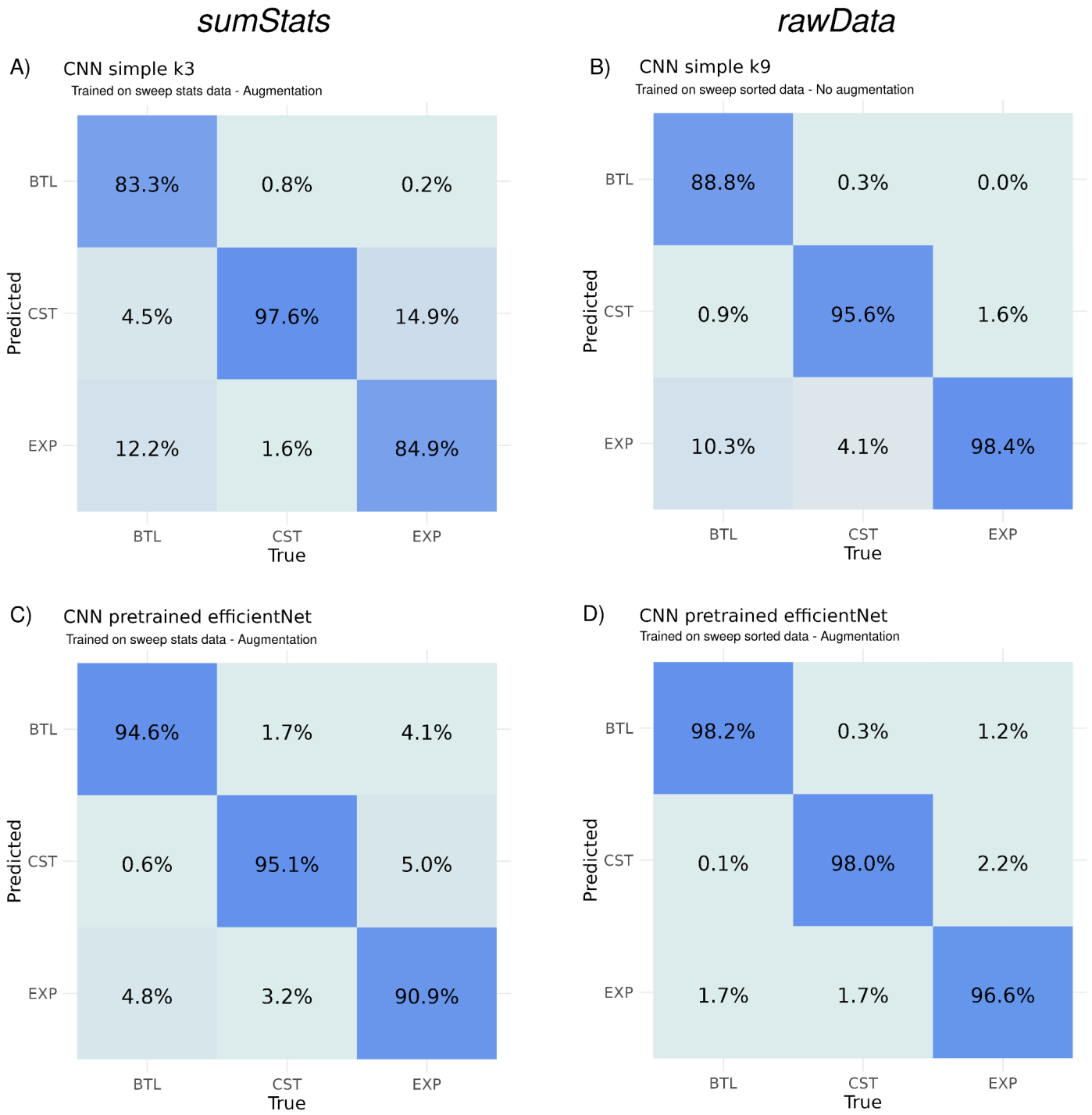



**Figure 23** - Confusion matrices of the best *simple* and *pre-trained* CNN regarding inferences on *sweep* datasets - comparisons of *rawData* vs *sumStats* input data.

**A)** best *simple* CNN architecture working on *sumStats* data, **B)** best *simple* CNN architecture working on *rawData* data, **C)** best *pre-trained* CNN architecture working on *sumStats* data and **D)** best *pre-trained* CNN architecture working on *rawData* data.

## Impact of selection and of model misspecification

We test how our 3 CNNs architecture handle miss-specification of the selection scenario. To do so, CNNs trained on *neutral* datasets are used to perform inferences on *sweep* datasets, and vice-versa. In this section, we try to give a more detailed breakdown, not only of the best scoring CNN of each category but also of the architectures in general.

Figure **24 - A** presents the confusion matrices of the best scoring CNNs of each category, trained on *neutral* data and tested on *sweep* datasets, while **Figure 24 - B** shows the results of the opposite *i.e.* CNNs trained on *sweep* datasets and tested on *neutral* datasets. **Table 5** below compares the accuracy and loss of the best CNN of each type when making inferences on the correct and the misspecified dataset. We compute Δ *accuracy* and Δ *loss* values to indicate the difference in accuracy and loss when the model is used to make inferences on a dataset with misspecified *selection* compared to its training dataset. **Supplementary Table 6** shows the same comparison as **Table 5**, but for the worst CNNs, in order to get a better understanding of what works and what does not.

**Table 5** - Comparison of the accuracy of the best CNNs architecture in case of model misspecification. Δ accuracy and Δ loss are computed as the difference in accuracy or loss when the model is used to make inferences on a dataset with misspecified *selection* compared to its training dataset. In **bold**, cases when model misspecification increases the accuracy and/or reduces the loss.

| Train on | Predict on | CNN | Input Data | Model | accuracy | loss | Δ accuracy | Δ loss |
|---|---|---|---|---|---|---|---|---|
| *neutral* | *neutral* | *simple* | *sumStats* | k3 | 0.966 | 0.142 | - | - |
| | | *mix* | *sorted-rawData* | k9 | 0.954 | 0.151 | - | - |
| | | *pre-trained* | *sorted-rawData* | *efficientNet* | 0.989 | 0.0299 | - | - |
| | *sweep* | *simple* | *rawData* | *k7* | 0.848 | 0.788 | - 0.118 | + 0.646 |
| | | *mix* | *rawData* | k9 | 0.873 | 0.764 | - 0.081 | + 0.613 |
| | | *pre-trained* | *sorted-rawData* | *efficientNet* | 0.927 | 0.380 | - 0.062 | + 0.350 |
| *sweep* | *neutral* | ***simple*** | ***sorted-rawData*** | **k7** | **0.978** | **0.0735** | **+ 0.035** | **- 0.081** |
| | | ***mix*** | ***sorted-rawData*** | **k7** | **0.975** | **0.106** | **+ 0.061** | **- 0.136** |
| | | *pre-trained* | *sorted-rawData* | *efficientNet* | 0.957 | 0.115 | - 0.019 | + 0.53 |
| | *sweep* | *simple* | *sorted-rawData* | k9 | 0.943 | 0.154 | - | - |
| | | *mix* | *rawData* | k9 | 0.914 | 0.242 | - | - |
| | | *pre-trained* | *sorted-rawData* | *efficientNet* | 0.976 | 0.0618 | - | - |

## Train on *neutral* and test on *sweep*

While the highest accuracy of the *simple* CNNs trained on *neutral* datasets but tested on *sweep* test datasets is at 0.8517 it also shows a loss way higher than the second most accurate *simple* CNN, which happens to have the lowest loss of all of them (accuracy : 0.852 vs. 0.848, loss : 1.59 vs. 0.788). Thus, the latter is considered, in this case, as the best simple CNN. The overall mean accuracy of the 8 simple CNNs put together is 0,764 ± 0,092 and the overall loss is 1,507 ± 0,543.

Two CNNs also compete for the place of best mix CNNs : two variations of the same mix CNN, trained on rawData, with or without data augmentation. The latter has a slightly higher accuracy (0,887 vs. 0,873) but also a higher loss (0,798 vs. 0,764). Considering the wider gap between the two losses the CNN trained with data augmentation is chosen as the best mix CNN. The overall mean accuracy of the 8 mix CNNs is 0,741 ± 0,117, and the overall loss is 1,439 ± 0,858.

Finally, the best pre-trained CNN architecture is an efficientNet architecture, trained on sorted-rawData without data augmentation, with an accuracy of 0,927 and a loss of 0,380. Overall, the 12 pre-trained architectures have an average accuracy of 0,893 ± 0,029, and an average loss of 0,467 ± 0,080.

**Table 6** - Comparison of the accuracy and the loss of the best and the average CNNs of each architecture when trained on *neutral* and tested on *sweep* datasets. Δ accuracy and Δ loss are the difference in accuracy and loss between the best CNN and the average of each category. Both values quantify the advantage of the best and the average CNN of a given architecture type.

| CNN | accuracy (average) | loss (average) | accuracy (best) | loss (best) | Δ accuracy | Δ loss |
|---|---|---|---|---|---|---|
| *simple* | 0,764 (± 0,092) | 1,507 (± 0,543) | 0,848 | 0,788 | -0,024 | 0,719 |
| *mix* | 0,741 (± 0,117) | 1,439 (± 0,858) | 0,873 | 0,764 | -0,023 | 0,675 |
| *pre-trained* | 0,893 (± 0,029) | 0,467 (± 0,080) | 0,927 | 0,380 | -0,034 | 0,087 |

All three architectures seem to struggle with the classification of BTL scenarios, wrongly classified as EXP in 11.3%, 12.8% and 16.7% of the cases by the *pre-trained*, *mix* and *simple* CNN respectively. The *simple* CNN also struggles to classify CST simulations, with 7.2% mistaken for BTL and nearly 13% classified as EXP, while the *mix* CNN also wrongly classifies about 9% of CST simulations as BTL, and more than 9% of the EXP simulation as

CST. As for the *pre-trained* CNN, it classifies about 6% of the EXP simulations as CST (**Figure 24 - A, B & C**).

Overall, the *pre-trained* CNNs seem to be more robust to model misspecification when trained on *neutral* and tested on *sweep* data and achieve the highest accuracy and lowest loss overall, as well as the single highest accuracy and lowest loss (**Table 6**).

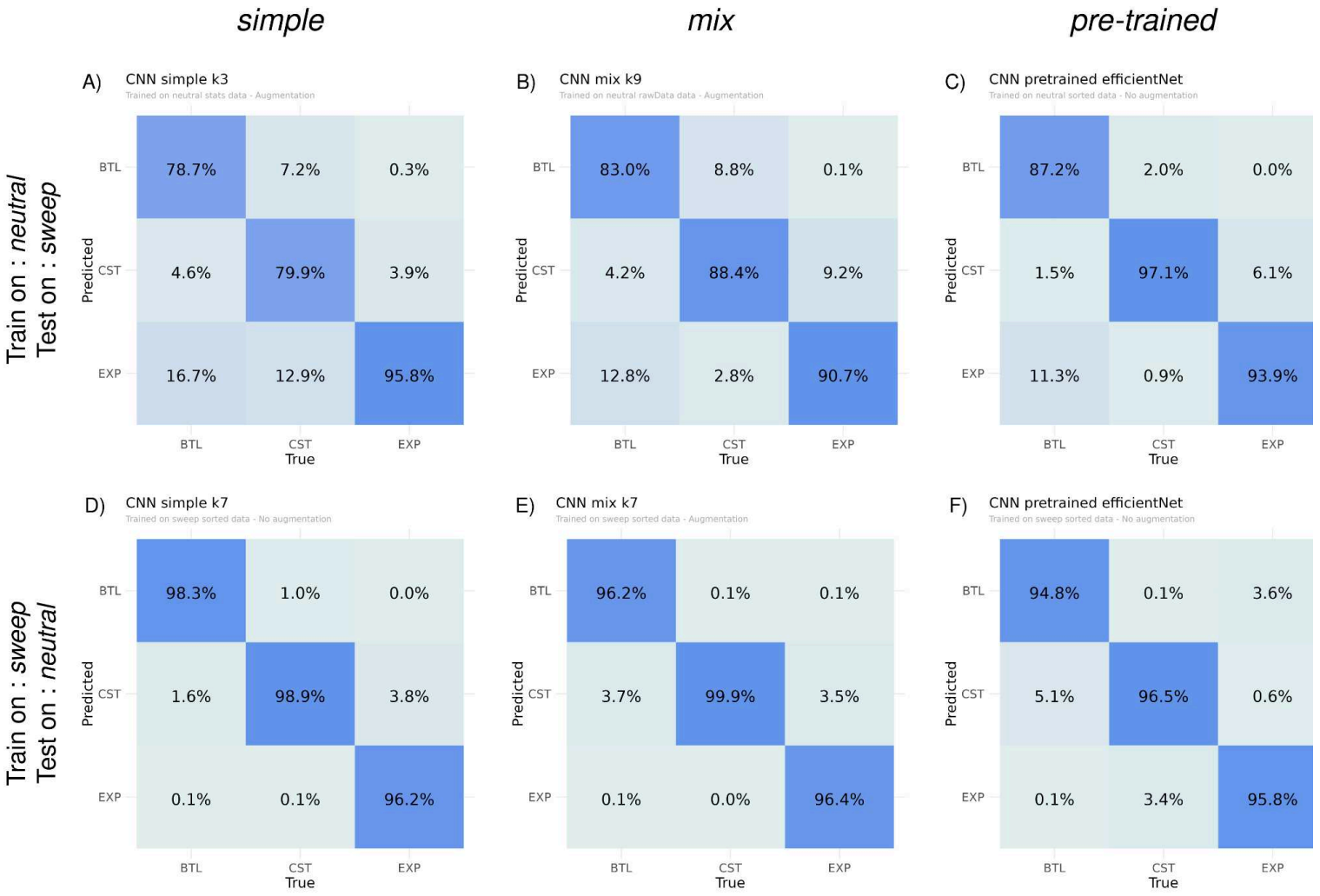



**Figure 24** - Confusion matrices comparing the 3 CNN architectures regarding inferences on a *test* dataset using the other selection model.

**A)**, **B)** and **C)** are respectively the *simple*, *mix* and *pre-trained* CNN trained on *neutral* data and tested on *sweep* test datasets. **D)**, **E)** and **F)** are the *simple, mix* and *pre-trained* CNN trained on *sweep* data and tested on *neutral* test datasets.

## Train on *sweep* and test on *neutral*

In the case of a model misspecification where the CNNs are trained on *sweep* datasets but tested on *neutral* datasets, the single best performing CNN both in regards to the accuracy or the loss is a *simple k7* CNN using *sorted-rawData* and trained without data augmentation, scoring an accuracy of 0.9777 and a loss of 0.0735. The 8 *simple* CNNs achieve an average accuracy of 0.915 ± 0.0672 and an average loss of 0.2768 ± 0.1612.

The best performing mix CNN is a mix CNN trained on *sorted-rawData* using data augmentation, with an accuracy of 0.975 and a loss of 0.106. However, the second best one is the same CNN but trained without data augmentation and it achieves an accuracy of 0.974 and a loss of 0.108. The differences being small enough for them to be due to the random sampling of the *test* dataset, it seemed worthy to mention that data augmentation doesn't seem to have a noticeable impact in this case. Together, the *mix* CNNs have an average accuracy of 0.9548 ± 0.0208 and an average loss of 0.2072 ± 0.1016.

Regarding the *pre-trained* CNNs, we have once again two CNNs performance that are quite similar, with the only difference being the use or not of data augmentation during the training phase. The *efficientNet* architecture trained on *sorted-rawData* has the higher accuracy of 0.957, and data augmentation only impacts the loss values : 0.115 without it, 0.147 with, thus making the former the best pre-trained CNN architecture. The average accuracy of all *pre-trained* CNNs is 0.945 ± 0.0109 and the average loss 0.185 ± 0.0424.

Few mistakes are done on *neutral* data by the CNN trained on *sweep* datasets. The *simple* CNN classifies about 4% of EXP simulations as CST, the *mix* CNN classifies between 3.5 to 4% of BTL and EXP simulations as CST and the *pre-trained* CNN classifies about 5% of BTL simulations as CST and 3.6% of EXP simulations as BTL (**Figure 24 - D, E & F**).

Unexpectedly, the best CNN in this case of misspecification is a *simple* CNN. However, looking at the average performances, both *mix* and *pre-trained* architectures are overall better, with a slight advantage in terms of accuracy for the *mix* ones, and in terms of loss for the *pre-trained* (**Table 7**).

**Table 7** - Comparison of the accuracy and the loss of the best and the average CNNs of each architecture when trained on *sweep* and tested on *neutral* datasets. Δ accuracy and Δ loss are the difference in accuracy and loss between the best CNN and the average of each category. Both values quantify the advantage of the best and the average CNN of a given architecture type.

| CNN | accuracy (average) | loss (average) | accuracy (best) | loss (best) | Δ accuracy | Δ loss |
|---|---|---|---|---|---|---|
| *simple* | 0,915 (± 0,0672) | 0.2768 (± 0,1612) | 0,9777 | 0,0735 | -0,0627 | 0,2033 |
| *mix* | 0,955 (± 0,0208) | 0.2072 (± 0,1016) | 0,975 | 0,106 | -0,0202 | 0,1012 |
| *pre-trained* | 0,945 (± 0,0109) | 0,185 (± 0,0424) | 0,957 | 0,115 | -0,012 | 0,07 |

## Comparison of the CNNs vs. ABC-RF approaches

As presented in the Introduction, ABC-RFs have historically been a more commonly used method of deep learning compared to CNNs. While running enough simulations for ABC to be accurate used to be an issue, using Random Forest in conjunction with them has allowed to drastically reduce this number to a point where it has become a realistically usable tool to perform accurate inferences in populations genomics (Pudlo et al., 2016 - Raynal et al., 2019). Even so, this method remains computationally heavy and in a lot of regards, quite similar to CNNs. A comparison between the two methods seems natural, and we chose to compare the best of our trained CNN against the ABC-RF. Tests have been run using the *sumStats* data of both *neutral* and *sweep* datasets. As explained in the Material & Methods section, as a way of reducing as much as possible the computation time needed to run ABC-RF, we tried to use extremely summarized summary statistics, labeled *average* (average-*sumStats*), where each of the 14 summary statistics is summarized even further into a single average value.

The ABC-RF models trained directly on the *sumStats* datasets, referred to as '*all bins'* display good results on *neutral* data, with 97% of the BTL and 100% of the CST simulations correctly classified, as well as up to 88.3% of the EXP simulations correctly classified. However, 11.7% of those EXP simulations are incorrectly classified as CST (**Figure 25 - A)**. Using *sweep* datasets for the training and tests, CST and EXP classifications display similar results but up to 17.7% of BTL simulations are incorrectly classified as EXP, which reduces the overall accuracy to 0.879 (**Figure 25 - B**). The best *pre-trained* CNNs trained and tested on the same *sumStats* datasets have similar accuracy on *neutral* for the BTL and CST

simulations (98.1 *vs.* 97.0 on BTL, 95.5 *vs.* 100 on CST) and a higher accuracy (97.5 *vs.* 88.3%) for the EXP classification compared to ABC-RF (**Figure 25 - A & C**). On *sweep* datasets, contrary to the ABC-RF, only a small fraction of BTL simulations (4.8%) are incorrectly classified as EXP and the mistakes on EXP simulations are split between incorrect classifications as CST (5%) and BTL (4.1%) rather than all of them being EXP classified as CST (**Figure 25 - D**). In terms of overall accuracy, the *pre-trained* CNNs outperformed the ABC-RF in both cases, with an accuracy of 0.970 vs. 0.950 on *neutral* and 0.935 vs. 0.888 on *sweep* datasets.

Using *average-sumStats* input data, the ABC-RF displays remarkably good results : on *neutral* data, 97% of the EXP simulations and more than 99% of both BTL and CST are correctly classified, for an overall accuracy of 0.985 (**Figure 25 - E**). While a slight decrease is observed for all classes when using *sweep* data (BTL 91.6%, CST 96.4%, EXP 94.3%), the overall accuracy remains quite high with a value of 0.944 (**Figure 25 - F**), outperforming the *pre-trained* CNNs on all but two classifications : the EXP simulations of the *neutral* dataset and the BTL simulations of the *sweep* dataset.

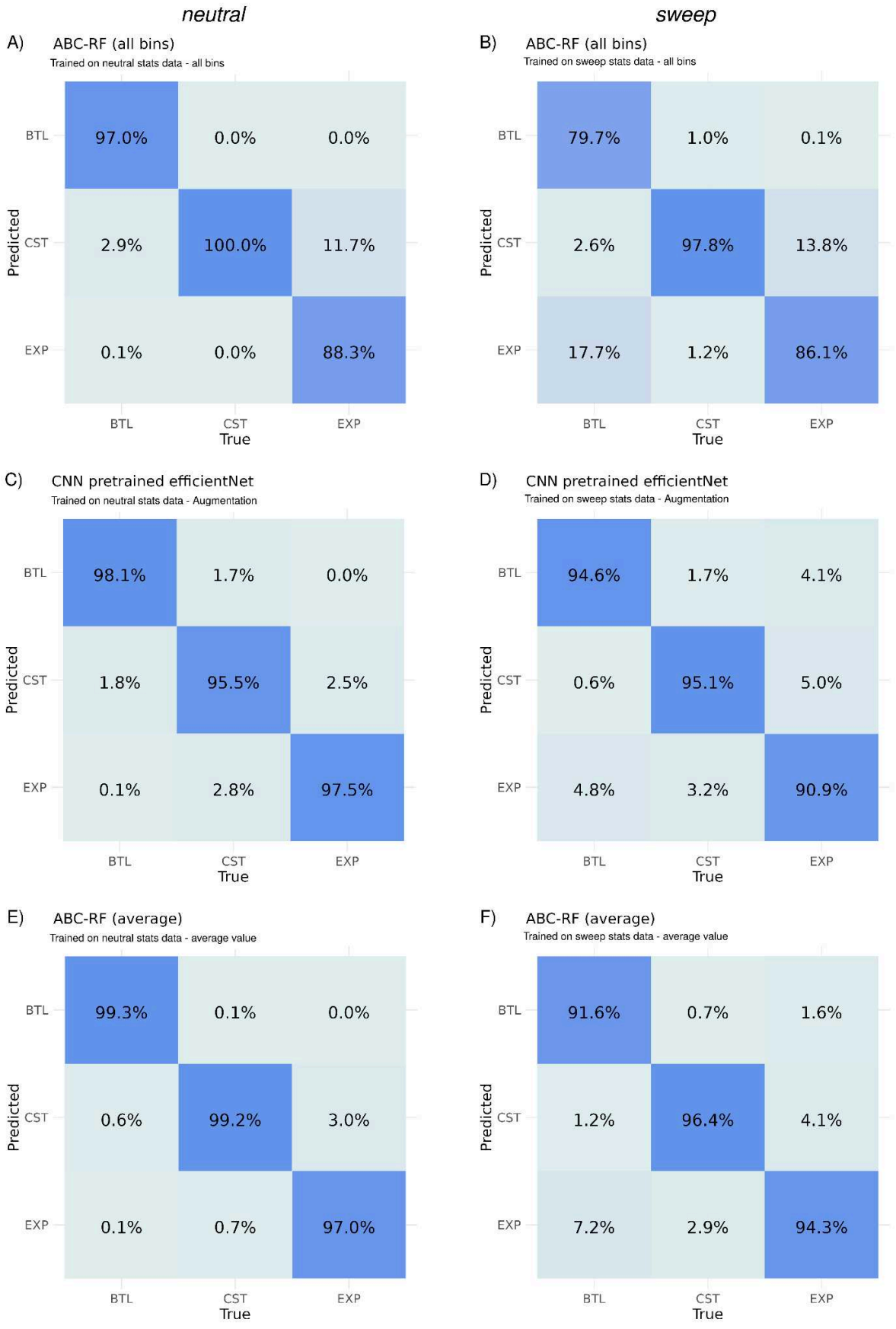

neutral
sweep
A)
ABC-RF (all bins)
Trained on neutral stats data - all bins
B)
ABC-RF (all bins)
Trained on sweep stats data - all bins
C)
CNN pretrained efficientNet
Trained on neutral stats data - Augmentation
D)
CNN pretrained efficientNet
Trained on sweep stats data - Augmentation
E)
ABC-RF (average)
Trained on neutral stats data - average value
F)
ABC-RF (average)
Trained on sweep stats data - average value
Predicted
True
BTL
CST
EXP
97.0% 0.0% 0.0%
2.9% 100.0% 11.7%
0.1% 0.0% 88.3%
79.7% 1.0% 0.1%
2.6% 97.8% 13.8%
17.7% 1.2% 86.1%
98.1% 1.7% 0.0%
1.8% 95.5% 2.5%
0.1% 2.8% 97.5%
94.6% 1.7% 4.1%
0.6% 95.1% 5.0%
4.8% 3.2% 90.9%
99.3% 0.1% 0.0%
0.6% 99.2% 3.0%
0.1% 0.7% 97.0%
91.6% 0.7% 1.6%
1.2% 96.4% 4.1%
7.2% 2.9% 94.3%

**Figure 25** - Confusion matrices comparing the ABC-RF and the best CNN architecture results.

**A)** and **B)** are the results of ABC-RF run on the *sumStats* datasets, respectively on *neutral* and *sweep*. **C)** and **D)** are the results of the *pre-trained* CNNs run on the same *sumStats*, *neutral* and *sweep* dataset respectively. **E)** and **F)** are the results of ABC-RF run on *average-sumStats*, *neutral* and *sweep* datasets *i.e*. on datasets where each summary statistics computed along the genomes have been summarized to a single average value.

## Robustness to model misspecification

We test the robustness of ABC-RF to model misspecification the same way we did for the CNNs architectures, by training models on *neutral* and performing classification on *sweep* datasets and reversely. We then compare the results of ABC-RF with the best scoring *pre-trained* CNNs using *sumStats*.

The ABC-RF trained on *neutral* and tested on *sweep* datasets correctly classifies 77.5%, 92.5% and 74.5% of the BTL, CST and EXP simulations respectively. 20.5% of the BTL simulations are mistakenly classified as EXP, and 24.9% of the EXP simulations are mistaken for CST. 5.7% of the CST are classified as BTL (**Figure 26 - A**). In comparison, the *efficientNet* CNN scores 81.3%, 82.0% and 91.9% of correct predictions on BTL, CST and EXP respectively. 16.6% of the BTL are wrongly classified as EXP but only 6.9% of EXP are classified as CST. While the same proportion of CST (5.8%) are classified as BTL, up to 12.2% of them are also mistakenly classified as EXP by the CNN (**Figure 26 - C**).

Both methods are then tested for the reverse situation : trained on *sweep* datasets, and tested on *neutral* datasets. The ABC-RF scores a perfect 100% of CST simulations classified as such, and BTL (10.8%) and EXP (12.6%) simulations are only incorrectly classified as CST but never as one another (**Figure 26 - B**). The CNN once again shows accuracies of 95.8%, 97.8% and 92.5% on BTL, CST and EXP scenarios. The CNN's mistakes are mostly BTL and EXP scenarios classified as CST (4.1% for BTL, and 5.3% for EXP), and very few mistakes otherwise, with only 2.2% of the EXP scenarios classified as BTL (**Figure 26 - D**).

In both cases, the *efficientNet* CNN slightly outperforms the ABC-RF in terms of overall accuracy (0.8506 *vs*. 0.8196 on *neutral*, 0.9537 *vs.* 0.9232 on *sweep*) . Finally, while the ABC-RF achieves near perfect classification of CST simulations, the CNN scores higher in all other classes with less mistakes overall.

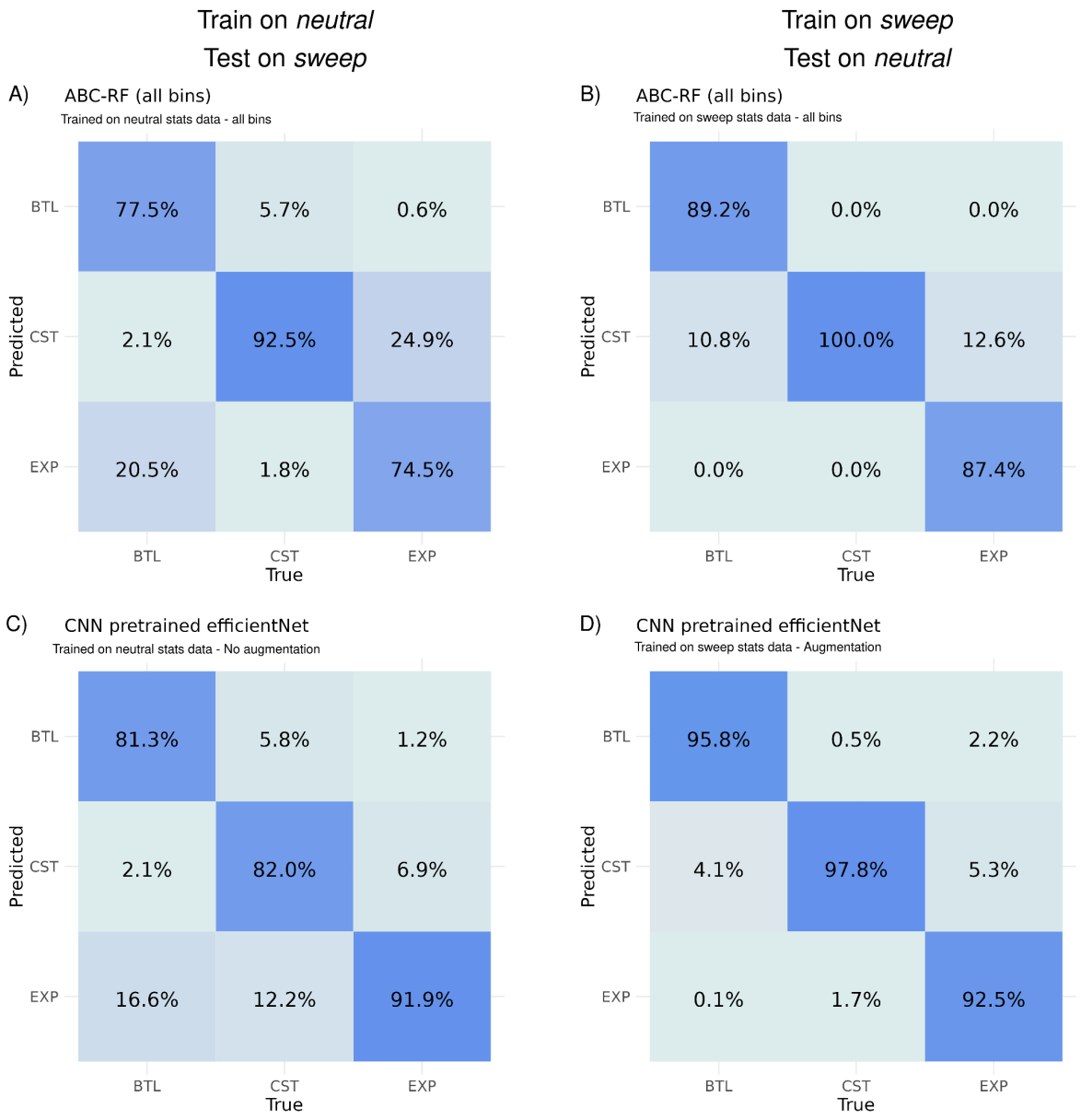



**Figure 26** - Confusion matrices comparing best CNNs *vs.* ABC-RF in case of model misspecification.

**A)** results of ABC-RF trained on *neutral* and tested on *sweep* datasets. **B)** results of ABC-RF trained on *sweep* and tested on *neutral* datasets. **C)** results of *pre-trained* CNN trained on *neutral* and tested on *sweep* datasets. **D)** results of *pre-trained* CNN trained on *sweep* and tested on *neutral* datasets.

## Effect of gene flow on CNN and ABC-RF predictions

As a last stop in our study, we wanted to test the methods robustness to migration, No migration was present in any of the datasets used until this point in our study. To do so, we use a separate set of *migration-test* datasets, composed of simulated genotypes sampled from populations following the 3 demographic scenarios and the two modes of selection, but with gene flow from population B to population A, at a rate of 1 migrant every 20 generations ($4.Ne.m = 0.2$, with $m$ the migration rate). The classifiers are trained on *non-migration* datasets, and their performances are then evaluated on the *migration-test* datasets, in order to highlight the effect of the presence of an unexpected gene flow.

First, we compare the influence of gene flow on the accuracy of the CNN architectures. To quantify the loss in accuracy caused by un-specified gene flow, we compared the accuracy of the best scoring CNNs on the migration datasets with their accuracy when tested on datasets without gene flow (**Table 8**). To do so, we compute Δ *accuracy* as the difference in accuracy between CNNs trained and tested on datasets without *migration vs*. the accuracy of the same CNNs tested on *migration-test* datasets. A negative value of Δ accuracy indicates a decrease in accuracy caused by the migration.

The best CNN of each architecture is chosen based on the highest accuracy and lowest loss obtained on the *migration-test* dataset. On *neutral* data, the best architecture is once again a *pre-trained* one, an *efficientNet* working here on *sumStats* and trained using data augmentation, achieving an accuracy of 0.887 and a loss of 0.3537 on the *neutral migration-test* dataset ; a decrease in accuracy of about 0.1 and an increase of 0.27 points in loss compared to its results on *non-migration* test data, but the *simple* CNN using *sumStats* data and trained without data augmentation shows very similar results with an accuracy of 0.8857 and a loss of 0.3168. Both the *simple* and *mix* architectures see a decrease in accuracy of the same order (between 0.08 and 0.1) but a relatively smaller increase in loss (between 0.16 and 0.175) compared to the *pre-trained* one.

On *sweep migration-test* datasets, the best architecture is the *mix k9* using *rawData* and trained without using data augmentation, with an accuracy of 0.914 but a loss of 0.5879. Due to this high loss value, an argument could be made to consider the *pre-trained resNet* architecture, using *rawData* and trained without using data augmentation as the actual best CNN here, with very similar accuracy of 0.9063 but a considerably lower loss of 0.2749. The *simple* CNN display a lower accuracy of 0.8857 and an intermediate loss of 0.371. This time, when compared to their results on *non-migration* datasets, both the *simple* and *pre-trained*

CNN display a decrease in accuracy quite similar at around 0.05~0.06, with an increase in loss of only 0.13 for the *pre-trained vs.* 0.22 for the *simple* CNN. The *mix* architecture shows the higher increase in loss at + 0.346, but no variation at all in its accuracy.

**Table 8** - Comparison of the influence of gene flow on the accuracy of CNNs architectures. Δ *accuracy* is computed as the difference in the accuracy between each type of CNNs tested on datasets without and with migration. A negative value of Δ accuracy indicates a decrease in accuracy. A positive value of Δ loss indicates an increase in the loss value. In **bold** are indicated the best scoring CNN on each dataset.

| Selection Model | Migration | CNN | Input Data | Model | Data augmentation | accuracy | loss | Δ *accuracy* | Δ *loss* |
|---|---|---|---|---|---|---|---|---|---|
| *neutral* | *No* | *simple* | *sumStats* | k3 | No | 0.9657 | 0.1422 | - | - |
| | | *mix* | *sorted-rawData* | k7 | Yes | 0.909 | 0.2464 | - | - |
| | | *pre-trained* | *sumStats* | *efficientNet* | Yes | 0.978 | 0.0836 | - | - |
| | *Yes* | *simple* | *sumStats* | k3 | No | 0.8857 | 0.3168 | - 0.08 | + 0.1746 |
| | | *mix* | *sorted-rawData* | k7 | Yes | 0.8117 | 0.4057 | - 0.0973 | + 0.1593 |
| | | ***pre-trained*** | ***sumStats*** | ***efficientNet*** | **Yes** | **0.887** | **0.3537** | **- 0.091** | **+ 0.2701** |
| *sweep* | *No* | *simple* | *sorted-rawData* | k9 | No | 0.9427 | 0.1537 | - | - |
| | | *mix* | *rawData* | k9 | No | 0.914 | 0.2415 | - | - |
| | | *pre-trained* | *rawData* | *resNet* | No | 0.952 | 0.1413 | - | - |
| | *Yes* | *simple* | *sorted-rawData* | k9 | No | 0.8857 | 0.371 | - 0.057 | + 0.2173 |
| | | *mix* | *rawData* | k9 | No | 0.914 | 0.5879 | **0** | + 0.3464 |
| | | ***pre-trained*** | ***rawData*** | ***resNet*** | **No** | **0.9063** | **0.2749** | **- 0.0457** | **+ 0.1336** |

Overall, gene flow seems to have minimal impact on the overall performance of the CNNs in terms of accuracy, despite increasing the associated loss values. Getting into the details of each scenario classification thanks to the confusion matrices (**Figure 27 & Figure 28**), two main effects can be observed. First, as before, classifications on the *sweep* dataset tend to generate mistakes where BTL simulations are classified as EXP, especially for the *simple* and *mix* architectures. Second, it appears that gene flow impacts the CNN classifications by making it so EXP simulations are mistakenly classified as CST. Indeed, across both datasets (*neutral* and *sweep*) and for all architectures, at least 10% of the EXP simulations are classified as CST; between 11 and 12.5% for *simple* CNN, between 13 and 16% for *pre-trained* CNN and up to 22 to 30% for the *mix* CNN.

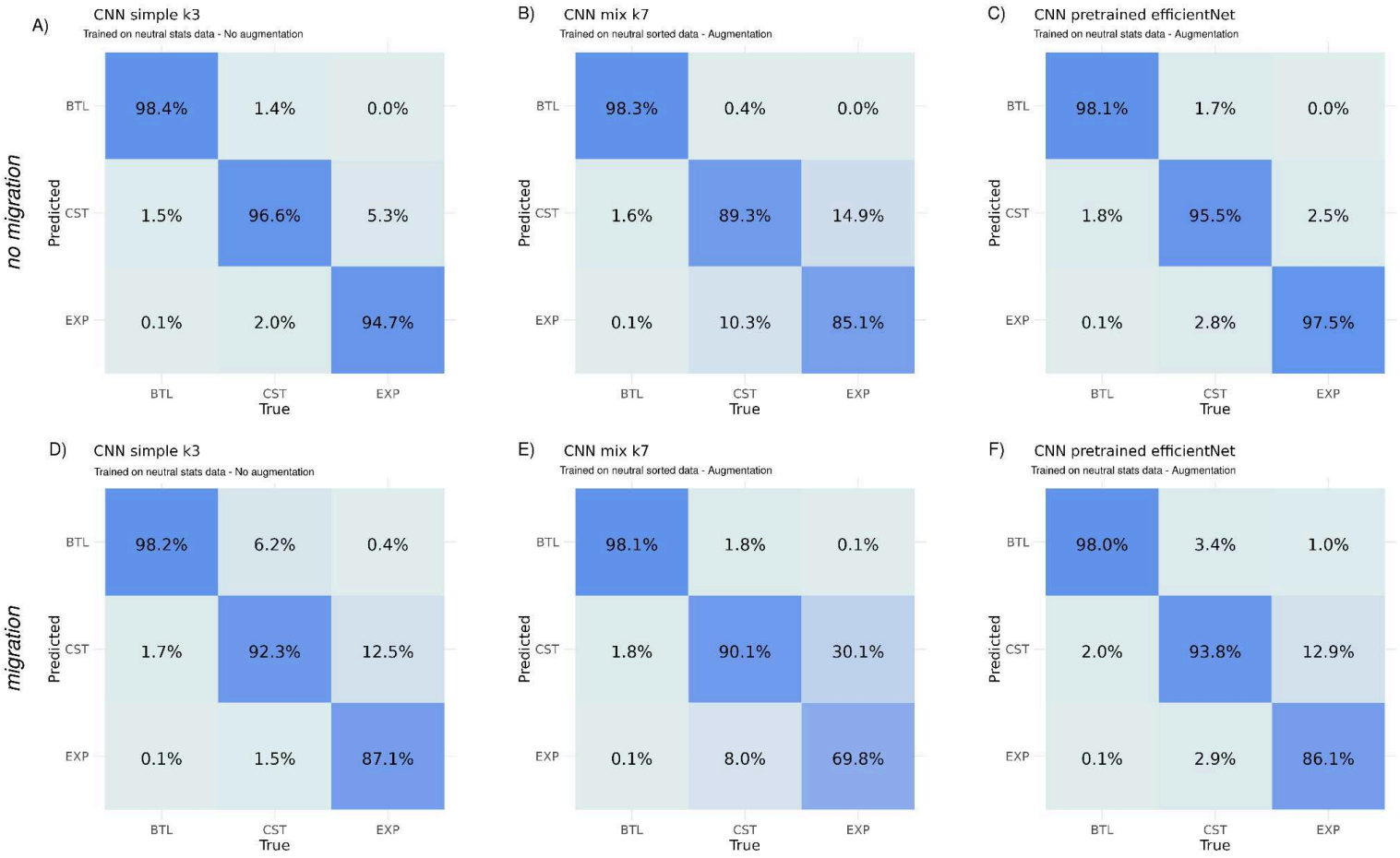



**Figure 27**- Confusion matrices comparing the results of the 3 CNNs architectures for inferences on *neutral* datasets with and without migration.

**A)**, **B)** and **C)** are the results of *simple*, *mix* and *pre-trained* CNN on datasets without migration. **D)**, **E)** and **F)** are the results of the same *simple*, *mix* and *pre-trained* CNN but tested on datasets with migration.

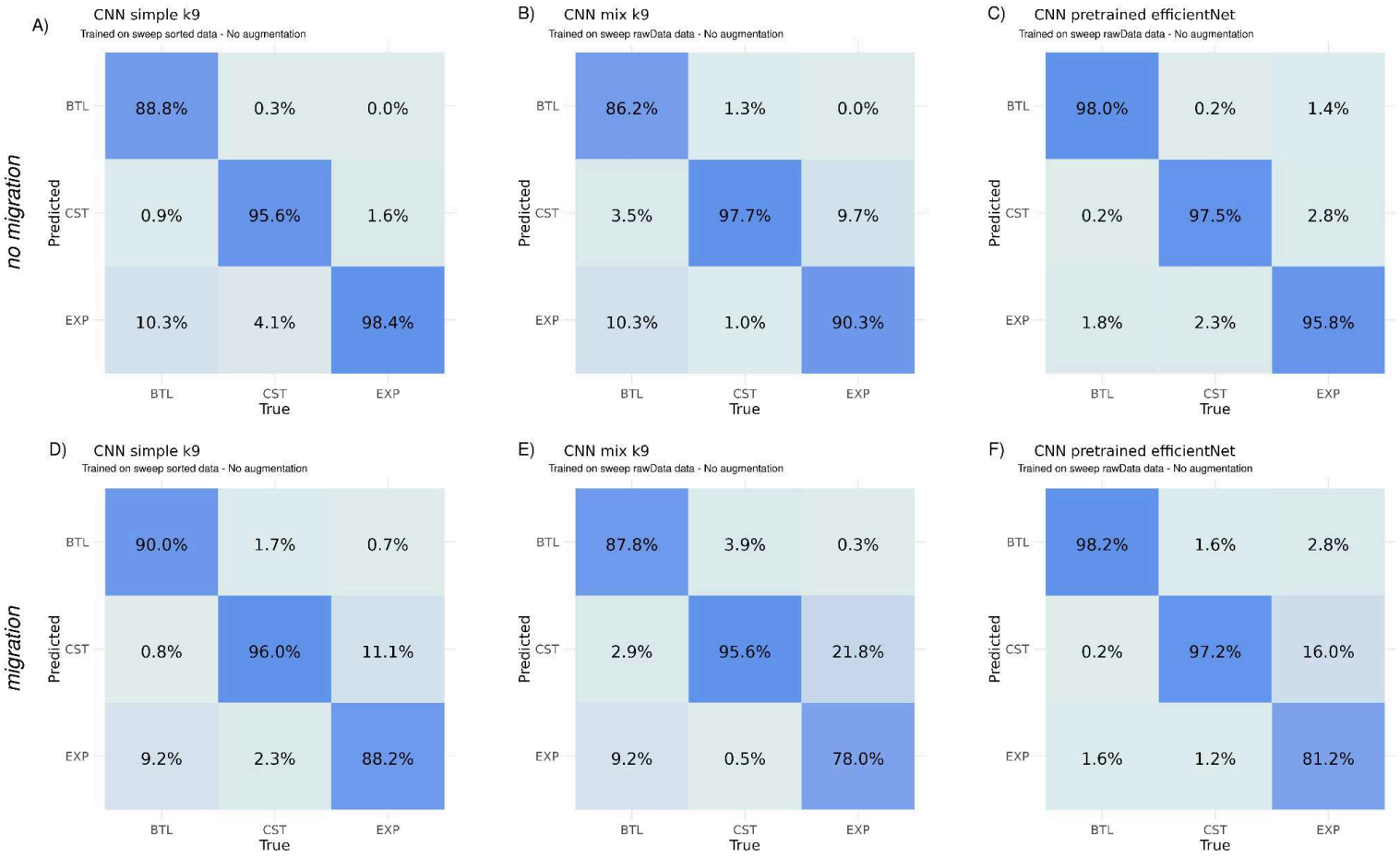



**Figure 28** - Confusion matrices comparing the results of the 3 CNNs architectures for inferences on *sweep* datasets with and without migration.

**A)**, **B)** and **C)** are the results of *simple*, *mix* and *pre-trained* CNN on datasets without migration. **D)**, **E)** and **F)** are the results of the same *simple*, *mix* and *pre-trained* CNN but tested on datasets with migration.

We follow the same method to compare the influence of gene flow on the ABC-RF results. Models are trained on datasets without migration and tested on *migration-test* datasets. The impact on the accuracy is twice as important in the case of the ABC-RF trained on the *neutral* datasets, with a diminution of about 0.1 in accuracy due to the presence of gene flow, for a reduction of only 0.056 for the *sweep* ones (**Table 9**). These values are remarkably close to the ones impacting the CNN in the same circumstances, with a reduction of around 0.08 to 0.1 for the *neutral* trained ones, and of 0.05~0.06 for the *sweep*.

**Table 9** - Comparison of the influence of gene flow on the accuracy of ABC-RFs. Δ *accuracy* is computed as the difference in the accuracy between ABC-RF tested on datasets with and without migration. A negative value of Δ accuracy indicates a decrease in accuracy.

| Selection Model | Migration | accuracy | Δ *accuracy* |
|---|---|---|---|
| neutral | *No* | 0.9503 | - |
| | *Yes* | 0.8434 | - 0.1069 |
| sweep | *No* | 0.8885 | - |
| | *Yes* | 0.8319 | - 0.0566 |

The same pattern is also found in the effect of migration on the classification of each scenario, with effects similar to those observed with the CNNs. Apart from the BTL scenarios classified as EXP for the *sweep* datasets, the noteworthy effect of gene flow on the classifications of ABC-RF is a consequent increase in the number of EXP simulations wrongly classified as CST, going from 13.8% to 30.8% on *sweep* and from 11.7% to 37.4% on *neutral* datasets (**Figure 29**).

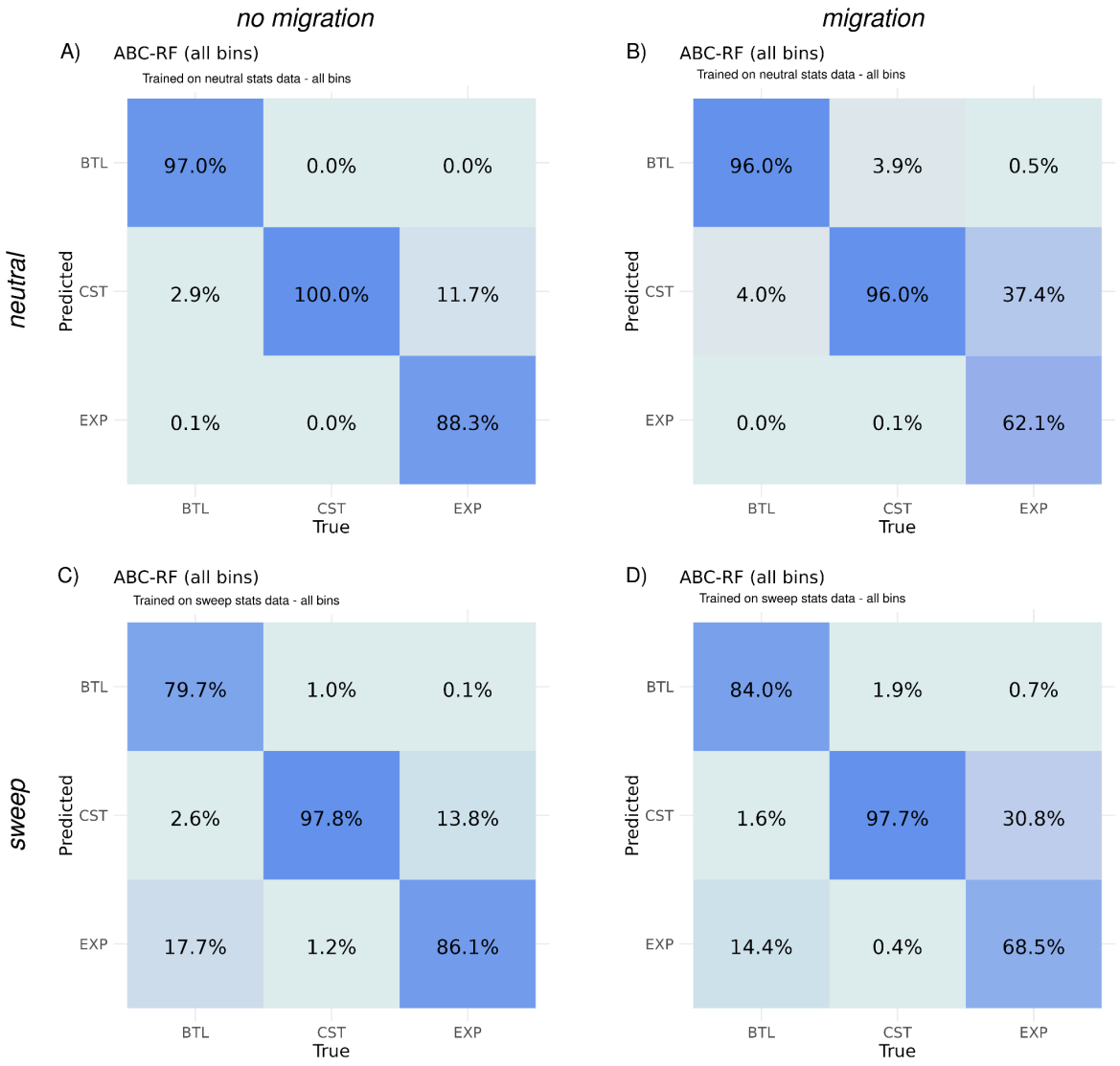



**Figure 29** - Confusion matrices comparing the ABC-RF results for inferences on migration data while being trained on simulations without it.

**A)** results of ABC-RF trained and tested on *neutral non-migration* datasets. **B)** results of ABC-RF trained on *neutral non-migration* but tested on *migration* datasets. **C)** results of ABC-RF trained and tested on *sweep non-migration* datasets. **D)** results of ABC-RF trained on *sweep non-migration* but tested on *migration* datasets.

# Discussion

The goal of this study is to propose an insight into different issues that may arise when developing a classification method of demographic history for genomic data, powered by deep-learning and specifically convolutional neural networks. One main objective was to try out a panel of different approaches, focusing on some key aspects all revolving around the same idea : the choice of the best CNN architecture for the task at hand. To do so, we tested three different network architectures, two “homemade” ones (*simple* and *mix*) and what we called “*pre-trained*” (*efficientNet* and *resNet*) architectures. One of the major aspects of our comparisons are the issues arising regarding model misspecification, both in terms of demography and selection. In this section, we will go back into each step of our comparisons, provide potential explanations of the differences observed and propose what we consider to be good practices for the implementation of such tools.

But before we dive into the discussion of our results, we want to take a few moments to discuss the type and origin of the possible mistakes of our classifiers. They can be explained in two ways: either (1) the mistakes are due to the demographic histories of the populations, meaning that the classifiers are not capable to differentiate the classes well enough, or (2) the mistakes are due to the effects of forces outside of the demographic events, in our case either *selection* or gene flow.

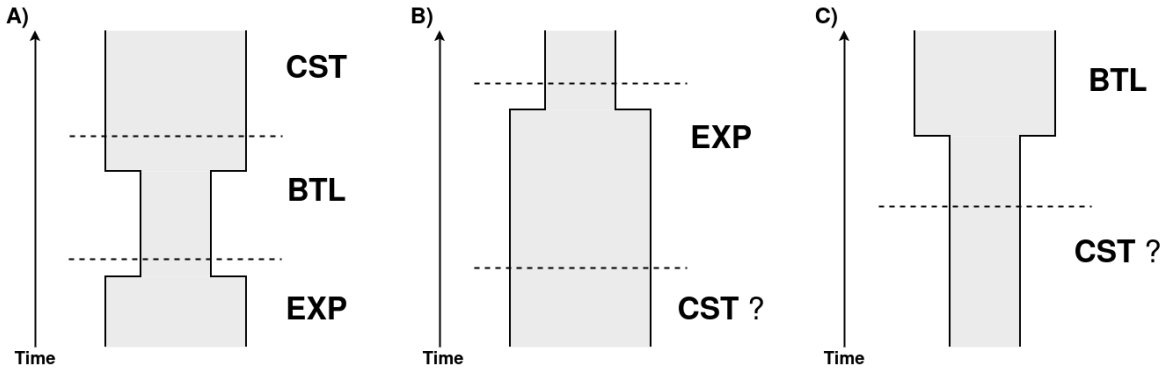



**Figure 30** - Representation of hypothetical populations.

**A)** Population experiencing an expansion following a bottleneck. Our three demographic scenarios can each be seen as one part of the demographic history of such a population. **B)** Population experiencing an expansion. If the demographic event is old enough, the population might be considered as a population of constant size. **C)** Same as B), but this time the demographic change is an old bottleneck rather than an expansion.

For the first type of mistakes, we must consider how the parameters used for our simulations affect things, specifically the effective sizes of the populations of interest. To put things simply, our three scenarios can be seen as part of the demographic history of a hypothetical population (**Figure 30 - A**). From there, it is easy to understand that the impact of old and/or weak demographic changes might be hard to detect, leading to the incorrect classification of BTL or EXP simulations as CST (**Figure 30 - B & C**).

The two major types of mistakes done by the all classifiers are BTL simulations incorrectly classified as EXP in presence of a selective sweep, and EXP simulations classified as CST in presence of gene flow. They are both mistakes due to either *selection* or *migration*, and it seems more coherent to cover them as we continue and they naturally arise into our discussion.

## All three types of CNNs correctly classify genomic data

For classifications in the most optimal and easiest case *i.e.* CNN trained and tested on *neutral* datasets, all three architectures can score accuracies above 0.95 (**Supplementary Table 4**), with the *pre-trained* CNNs being the best ones. Indeed, 9 out of the 10 best architectures on *neutral* data are *pre-trained* CNNs. Together, they achieve an average accuracy of 0.980 and an average loss of 0.0719, followed by the *simple* CNNs (average accuracy = 0.835, average loss = 0.404) and the *mix* CNNs ranking last (average accuracy = 0.787, average loss = 0.568). That means that *pre-trained* CNNs are between 15 to 20% more accurate and have a loss 5 to 8 times lower than the *simple* and *mix* CNNs. A similar pattern is observed for CNNs trained and tested on *sweep* datasets. This time, 8 out of the 10 best architectures are *pre-trained* CNNs, with an average accuracy of 0.949 and an average loss of 0.150, where *mix* CNNs have an average accuracy of 0.832 and an average loss of 0.435 and *simple* CNNs of 0.780 and 0.576. The gap between *pre-trained* CNNs performance remains about the same in terms of accuracy, with an advantage of about 15 to 20%, and a loss 2.5 to 3.5 times lower. While the best *pre-trained* CNNs display a slightly lower amount of mistakes than the other architectures on the *sweep* datasets, the only case where they seem to have a clear advantage is on the classification of BTL scenarios which are often wrongly classified as EXP by the *simple* and *mix* CNNs. This can be easily explained by the confounding effect of selective sweeps and the demographic histories of our EXP simulations. As already explained before, selective sweeps and expansions following a bottleneck (which is what our EXP scenarios can be seen as - **Figure 30 - A**) can leave very similar effects on the genomes (Pavlidis et al., 2008). Thus, as the *simple* and *mix* architectures are quite simple, it might be possible that the models are not complex enough

to learn, from the type of data provided, the necessary patterns to always differentiate the effects of a selective sweep from those of an expansion. Despite their relatively worse results, both *simple* and *mix* architectures still display good accuracies, well above the expected 0,333… of a random guess, and could be used as substitutes to the *pre-trained* architectures if needed, for example in a study focusing on neural network interpretability where simpler networks might easier to work with (see Räz, 2024 for an interesting introduction to interpretability in ML).

## CNNs performed best when trained using (*sorted*) raw genetic alignments

As explained in the Introduction, in population genomics it is common practice to use summary statistics to highlight a specific aspect of a genomic sequence. However, these statistics have certain disadvantages. They can be influenced by the confounding effects of other processes than the one they have been developed for. To recall the same example, a summary statistic such as Tajima's D which has been developed to detect departure from neutrality (Tajima, 1989) will be impacted by both selection mechanisms and demographic events. The statistic will be positive in presence of an excess of high-frequency mutations which could be either the sign of a population contraction or of a process of balancing selection. Moreover, since they summarize information, these summary statistics may miss some of the information present in genomes. Thus, two methods are commonly used to circumvent these issues. The first one is to use a large number of summary statistics at the same time, each of which focusing on a specific aspect of the information contained in the alignments. Another possibility is to use the alignments directly, without calculating summary statistics. In both cases, it was possible here to represent the data directly in the form of matrices, used as input data for the CNNs.

The choice of the representation of the input data is one of the most important aspects of any machine learning approach (Mughal & DeGiorgio, 2019). Thus, each architecture was trained then tested on different types of input data : matrices of raw (*rawData*) or sorted (*sorted-rawData*) genotype alignments, or matrices of a combination of summary statistics (*sumStats*). These choices are based of previous studies where such sets of summary statistics have been used as inputs with similar CNN approaches (Schrider & Kern, 2016; Caldas, Clark & Messer, 2022), while others have tried to directly use raw alignments with SNP encoded as black and white pixels (Flagel et al., 2019; Hamid et al., 2023). However, to

our knowledge, this study is the first one to directly compare these different methods of representation of input data against each other on a given task.

One could expect the summary statistics to highlight information within the alignments and thus, to reduce noise in the data, making it that much easier for the neural networks to extract the most important features. However, *rawData* and more specifically *sorted-rawData* appears to be the input data type achieving the best results in most cases, with only the *simple* CNN working on *neutral* data achieving better performance using *sumStats* than its *rawData* counterpart. As for the *pre-trained* CNNs, while already obtaining higher accuracies and lower losses scores than the two other architectures, even on *sumStats* data, they performed even better on *rawData*. One explanation could be that, due to their more intricate architecture, such CNNs are able to better leverage all the information contained in the raw alignments. This is, in itself, another notable difference in regards to the cases of use of each of those architectures. If for whatever reason, summary statistics can't be computed and raw alignments must be used as is, *pre-trained* CNN architectures could be a better tool to use. Moreso, if possible, the small step of sorting the genomic alignments appears to be beneficial and should, in most cases, be easy enough to implement into an analysis pipeline.

## CNNs trained on genomic data containing patterns of selection display advantages over CNNs trained on *neutral* data

Overall, all of the three tested CNNs architectures showed great robustness to model misspecification, with the biggest decrease in accuracy being of only - 0.114 . Few mistakes are done here and are the expected ones: CNNs trained on *neutral* data mistakenly *sweep* simulations as EXP. Indeed, models trained on *neutral* data are slightly less accurate on *sweep* data, which is to be expected, as *sweep* data contains patterns of selection that could resemble that of demography. As those models never learn to recognize such patterns, a decrease in accuracy is to be expected. As for models trained on *sweep* data and tested on *neutral* data, one could also expect a decrease in accuracy. Few errors are made, with the most common ones being confusions between CST and any of the other two scenario, which are to be expected as the CST scenario is a 'middleground' scenario and cases are bound to happen when demographic and/or selection events are old enough or were so soft that most if not all of their signals are gone by the time the population is sampled. However, it appears that *simple* and *mix* CNN trained on *sweep* data but used to make inferences on *neutral* datasets outperformed their "correctly specified" counterparts. An explanation could be that,

due to the important range for the parameters used for the simulations of the *sweep* datasets, some *sweep* simulations are bound to display old and degraded signals of selection. In such cases, one could argue that those *sweep* simulations are virtually indiscernible from an equivalent *neutral* simulation. Thus, it is possible that *sweep* trained CNNs managed to learn not only to differentiate demographic scenarios in presence of selective sweeps, but also in absence of it. As *sweep* datasets are more difficult to classify, due to the confounding effects of selection and demography, it appears coherent that CNNs trained only on the more “easy” *neutral* dataset struggle when working on *sweep* data, while CNNs trained on the more “difficult” *sweep* dataset manage to be more generalists, as they could have also learn to classify genomic alignments with no “visible” signal of natural selection.

Unexpectedly, the overall best CNN architecture in this case of model misspecification is the *simple k7* CNN architecture, trained on *sorted-rawData*, followed by the *mix* CNN and finally the *pre-trained* CNN. Even though we do not currently have any proper explanation as to how and why such a simple architecture managed to outperformed the more intricate *pre-trained* CNNs (it might just be due to the random nature of neural network training, but this explanation does not sit right in our minds), another interesting result should be noted: the worst performing CNN in case of selection model misspecification is the *simple k7* CNN train on *rawData*. This shows the utmost importance of the choice of representation of the input data.

## CNNs slightly outperform ABC-RF when using matrices of summary statistics as input data

ABC-RF trained on *sumStats* datasets showed overall results slightly inferior to those of the *pre-trained* CNN. Noticeably, on both *neutral* and *sweep* datasets, they generate 2 to 3 times more incorrect classifications than the CNN, with up to 13,8 % of EXP scenarios mistaken for CST. They also seem to have more issues with dealing with the effect of selective sweeps, with up to 17,7 % of the BTL simulations classified as EXP, 3 to 4 times more than the 4,8 % of the CNN. Both of these errors are the same as the ones observed before with the *simple* and *mix* CNN architectures and thus, it appears that the ABC-RF approach remains closer to those more simple architectures rather than to the *pre-trained* CNNs.

Regarding model misspecification, the results indicate that both ABC-RF and the *efficientNet* CNN are robust to model misspecification, though the CNN demonstrated a slight edge in

overall accuracy and consistency. The same behavior as with the CNNs in case of misspecification is observed with ABC-RFs: models trained on *neutral* data struggle on *sweep* data and lose some accuracy, while models trained on *sweep* datasets tends to be highly robust to model misspecification and even display a slight increase in accuracy in some cases (BTL classification going from 0.955 to 0.96, CST classification going from 0.953 to 0.975) when used on *neutral* datasets. The same explanation should hold here too: *sweep* datasets most likely contain enough *neutral*-like simulations for the models to learn to classify the input data in presence and absence of actual selection signal.

Finally, as a simple test to reduce the computational cost of the ABC-RF, we tried to run ABC-RF on *average-sumStats*: “extremely summarized summary statistics”. ABC-RF trained using this approach outperformed *pre-trained* CNN working on *sumStats*, by 0.006 points of accuracy on *neutral* data, and 0.006 points on *sweep* data. Such a small difference does not seem like much, but the ABC-RF managed to reach this accuracy while working on only 14 variables against the thousands of parameters used by the CNNs.

The strong performance of this strategy may be attributed to its inherent simplicity: demographic events tend to exert effects at a genome-wide scale. By averaging the summary statistics across the genome, the noise and local effects of processes like selection are effectively smoothed out, leaving the more global demographic signals intact. As a result, classifying the samples based on their demographic history may be more straightforward using this type of input. While not being a proper comparison to the CNN, this is once more an indication of the huge effect that a change of the representation of data used as input can have on machine-learning approaches.

## The homogenizing effects of Gene Flow lead to more common misclassification of simulations as CST

Gene flow significantly impacts species evolutionary trajectories and thus demographic inferences. It can lead to overestimates of population sizes, as it works as a force that counteracts the differentiation caused by genetic drift, mutation, and local adaptation.

CNN architectures, particularly *pre-trained* models, maintained a high level of overall accuracy with only a slight drop in performance. For instance, when tested on *neutral* datasets with migration, the *pre-trained efficientNet* achieved an accuracy of 0.887 and a loss of 0.3537, showing only a 0.1 decrease in accuracy and a 0.27-point increase in loss compared to the results on *non-migration* datasets, and similar results are observed on

*sweep* datasets. Thus, gene flow seems to have minimal impact on the overall performance of the CNNs in terms of accuracy, despite increasing the associated loss values. However, the confusion matrices revealed specific biases in the classification outcomes. Gene flow consistently led to higher misclassification rates where EXP simulations are categorized as CST across both *neutral* and *sweep* datasets. This trend was observed in all CNN architectures, but the *mix* architecture was particularly impacted, with up to 22-30% of this type of mistake.

The analysis of gene flow's impact on ABC-RF results reveals a consistent pattern with the effects observed in CNNs, with, however, a more notable decline in accuracy when models trained on *non-migration* datasets are tested on *migration* datasets. For the ABC-RF trained on *neutral* datasets, gene flow causes a substantial reduction in accuracy by approximately 0.107, while for *sweep*-trained models, the decrease is smaller at around 0.057. These results are comparable to those seen in CNNs, where the accuracy drop ranges from 0.08 to 0.1 for *neutral*-trained models and from 0.05 to 0.06 for *sweep*-trained ones. In more details, gene flow leads to an increase in the misclassification of EXP simulations as CST across both datasets. The proportion of EXP scenarios misclassified as CST nearly doubled, from 13.8% to 30.8% in *sweep* datasets and from 11.7% to 37.4% in *neutral* datasets. This suggests once again that the homogenizing effect of gene flow disrupts the signals left by demographic events, making it difficult also for ABC-RF to differentiate between EXP populations and those with a constant size.

Indeed, gene flow can result in the appearance of larger effective population sizes due to the influx of "CST-like" migrants from the ghost-sister population, which blurred the distinction between the CST and the other scenarios. This homogenizing effect appears to make it difficult for the models to detect the distinct patterns typically associated with EXP population, leading to this bias in classification.

# Conclusion

In this study, we explored the effectiveness of convolutional neural networks (CNNs) and Approximate Bayesian Computation Random Forest (ABC-RF) in classifying demographic histories from genomic data. Our comparative approach, evaluating different CNN architectures and input data representations, provided insights into the strengths and limitations of such models in population genomics. By testing a range of CNN architecture types (*simple*, *mix*, and *pre-trained* CNNs), we highlighted the importance of architecture choice, data representation, and model training for achieving good results at this classification task. Additionally, we assessed the impact of factors like model

misspecification and gene flow on classification accuracy, further elucidating how different evolutionary processes can influence the performance of machine learning models. Our results indicate that pre-trained CNNs consistently outperform simpler architectures. Their superior accuracy across different scenarios suggests that these more sophisticated models can better capture complex patterns in the data. However, the *simple* and *mix* CNNs still achieve satisfactory performance, especially when trained on *sweep* data. This implies that even simpler models can be effective and provide a valuable trade-off between model performance and complexity and interpretability. The choice of data representation plays a crucial role, as seen with the numerous instances where a simple change in type of data input heavily impacts the models accuracy. Specifically, it appears that using sorted raw alignments over summary statistics improved the accuracy of the CNN models. Model misspecification analyses revealed that CNNs trained on sweep datasets exhibited better generalization capabilities compared to those trained on neutral data. This likely reflects the ability of *sweep*-trained models to learn a broader range of signals, encompassing both selection and neutral demographic patterns. The robustness observed in both CNNs and ABC-RF in the face of model misspecification further highlights the potential of these approaches for demographic inference, despite the increased misclassification rates, especially when distinguishing between expansion and constant-size populations observed in presence of gene flow.

Overall, fine-tuned *pre-trained* CNNs, particularly *efficientNet* architectures, prove to be the most efficient approaches for this classification task. They consistently outperformed the other architectures in most comparisons and worked well whether used on raw alignments or summary statistics matrices. Moreover, they are robust to model misspecification and maintain high performance even in cases where gene flow was not explicitly accounted for in the training data. They also outperformed the widely used ABC-RF method in all but the most straightforward and ideal cases. Their performances, availability as well as ease of use increasing gradually with the passing years might make their use slowly outshines the currently more popular methods or more homemade CNN architectures.

To conclude this first part, we want to propose a short summary of the key elements highlighted by this work:

1) Even very simple CNNs architectures, the likes of the *simple* and *mix* CNNs presented in this study, show very good results in the classification of genomic data according to demographic scenarios. The best ones are the *pre-trained* architectures.

They are easily available online, but are more complex architectures and often less interpretable compared to more homemade ones.

2) Between the three tested types of representation for our input data, the *sorted-rawData* outperformed the simple *'non-sorted-rawData*' as well as the *sumStats*. Given how small of a step this sorting the alignments is, and that it seems to really enhance the models performances, we are convinced that it really is a worthy extra step to incorporate into any pre-processing of genomic alignments with similar goals as this study.

3) Classifiers trained only on *neutral* data are bound to display a decrease in accuracy when used on data impacted by *selection*. However, if CNNs trained on *sweep* datasets show remarkable resistance to model misspecification and sometimes even perform better on data not impacted by selection. In other words, making sure that the training datasets contain as much variability as possible plays a major role in the generalization capabilities of the networks.

4) CNNs results are on par or better than those of ABC-RF when using the same type of data input, and both methods react the same to model-misspecification. The same can be said about the effect of gene flow: both methods display a decrease in accuracy for the specific classification of EXP scenarios, but are otherwise quite robust to the presence of migration.

These four aspects are at the core of this study, but one stands out above the rest: the significance of choosing the right data representation. In case of selection model misspecification, the best CNN is not a *pre-trained* but a *simple* one, using *sorted-rawData* while the worst CNN is actually the exact same architecture with the only difference between them being the use of *rawData* instead. A similar pattern occurs with the ABC-RF method, where employing the *average-sumStats* representation allows it to surpass the *pre-trained* CNNs. In both instances, altering the way the data is represented significantly affects the classifiers' performance. This observation is, in our view, the most critical takeaway - while the other points discussed are important, one of the most important decisions when developing this kind of deep-learning approach for population genomics lies in the choice of a specific way to represent the data and understanding the far-reaching consequences of that choice.

We like to think that this kind of result could help push forward the use of deep learning in population genetics. Efforts have been made during the last years to make deep-learning approaches as user-friendly as possible for researchers with various backgrounds in population genetics. Specifically, some work has been done to propose clear and well documented workflows (Whitehouse & Schrider, 2022) and *pre-trained* architecture specifically catered to population genetics have started to emerge (Hamid et al., 2023). However, one of the remaining issues that clouds the use of deep-learning is the interpretability of such tools. For the time being, most of them are akin to a black-box and we lack a clear understanding of the inner workings of their decision process, an understanding that could lead population geneticists to better grasps or even uncover new and complex evolutionary mechanisms or processes (Korfmann et al., 2023). That is why, we hope that the results presented in the first chapter of the thesis could help to highlight two main points – first, more and more efforts are made to make them as accessible as established tools for demographic inference, and *pre-trained* architectures are both easy to implement and boast great results. Second, while we believe that pre-trained architectures are likely to become the standard in the coming years, the most critical factor may not be the aspects often highlighted in some AI discussions – such as the ever growing number of parameters or the increasing complexity of networks – but rather the thoughtful choices and decisions made regarding the specific datasets and types of data employed.

# Chapter II

## *Detection and localization of selective sweeps using Convolutional Neural Networks*

# Introduction

In the previous chapter, we focused on the training of various CNNs for the classification of genomic alignments according to demographic scenarios. Doing so, we had to generate pseudo-genomic data in order to train and test the networks, and we have highlighted the importance of model architecture and data representation. In this next section, our focus will shift to the detection of selective sweeps. Given the promising results of CNNs in demographic classification, we wanted to explore their capabilities in recognizing the specific patterns associated with selective sweeps. By developing methodologies aimed specifically at the identification and localization of selective sweeps associated with populations of various demographic histories, we aim to further illustrate the applicability of deep learning in uncovering complex evolutionary signals within genomic data.

As stated in the Introduction of this thesis, a central goal of evolutionary research is to leverage genetic data to unravel the forces shaping genetic diversity and trace the natural history of populations. Both natural selection and demographic events leave distinctive patterns on genetic variation which can be used to make these inferences. Selection can influence allele frequencies in various ways: positive directional selection will increase the frequency of the selected allele, but also affect the neutral regions linked to it, and generate the “valley of diversity” around the selected loci (Smith & Haigh, 1974) characteristics of a selective sweep, whereas balancing selection will tend to increase polymorphism at linked loci (Smith & Haigh, 1974). Similarly, demographic events like population size changes or gene flow can significantly shape patterns of genetic diversity, leaving distinctive signatures in the genome (Pespeni et al., 2011). Deciphering these patterns provides valuable insights into a population's evolutionary history and the recent advances in DNA sequencing technologies have led to the rapid growth of available genomic data, leading to a drastic increase in its availability.

As a result, more powerful methods and tools have been developed to investigate the signatures of evolutionary processes. One such example is the detection and identification of recent positive selection, with more and more studies proposing ways to detect the molecular target of such positive selection *i.e.* to search for patterns associated with selective sweeps (Tajima 1989; Nielsen et al. 2005; Boitard et al., 2009 & 2012; Pavlidis et al. 2010; Chen et al., 2010 - Wollstein & Stephan, 2015 for a recent review). Such methods search for the genetic signature of selective sweep already discussed multiple times in this thesis: following the rapid fixation of a beneficial allele, a “valley of diversity” is created at, and around the selected loci (Smith and Haigh, 1974). Such mechanism also tends to produce an excess of low and high frequency derived alleles and, while linkage disequilibrium is not affected across the flanks of the valley, it is increased on either side of the selective sweep (Kim & Nielsen, 2004). However, by moving away from the target of selection, polymorphism is recovered by the effect of recombination that breaks apart neutral variants linked together by the sweep (Smith & Haigh, 1974).

From there, four main methods have been developed for the detection of selective sweeps (Koropoulis et al., 2020) : (1) detection based on the reduction in genetic diversity, largely used in the study of the model species *D. melanogaster, D.simulans and D. ananassae* (Aguade & Langley, 1989 & 1994; Begun & Aquadro, 1991, Miyashita, 1990) for example, (2) detection of selective sweeps based on the SFS signature *i.e.* the shift of the SFS toward high and low frequency derived variants (Fay & Wu, 2000; Kim & Stephan, 2002), (3) detection of selective sweeps based on the signature of linkage-disequilibrium (Kim & Nielsen, 2004) *i.e.* the increase levels of LD on each side of the selected loci and (4) detection of selective sweeps using machine learning methods. Regardless of the method, tools used to detect signatures of selection on the genomes usually work through two possible approaches : using summary statistics, or detecting sweeps from whole genome data.

Historically, the use of summary statistics is the first approach proposed to fill the need of analytical tools to study genetic diversity, with various statistics developed through the years (Watterson's θ estimator, Watterson, 1975; Tajima's pi or Tajima’s D, Tajima, 1989; etc.). Such statistics are used to perform neutrality tests by comparing their values to expected values under neutrality. However, as each statistic is usually catered to highlight only one specific aspect of the information found in a genome alignment, another approach has been proposed : using large number of such summary statistics at once to maximize the amount of retrieved information (Lin et al. 2011; Schrider and Kern 2016; Sheehan and Song 2016).

Thus, the focus shifts from trying to interpret the individual values of each statistic to trying to make sense of the overall patterns observed across the set of chosen statistics. On the other hand, detection of selective sweeps using whole genomes has been made possible thanks to the advent of next generation sequencing (NGS). Some of those methods, such as SweepFinder (Nielsen et al., 2005), SweepFinder2 (DeGiorgio et al., 2016) as well as SweeD (Pavlidis et al., 2013) works using CLR (Composite Likelihood Ratio) tests while other, such as OmegaPlus (Alachiotis et al., 2012), are built using the ω-statistic (Kim & Nielsen, 2004) and search specific patterns of LD.

However, demographic history complicates the interpretation of these tests, as population events like bottlenecks can mimic the genetic signatures of selective sweeps (Koropoulis et al., 2020). For instance, a bottleneck reduces effective population size, causing a rapid coalescence of lineages and producing genealogies that resemble those shaped by positive selection (**Figure 31**).

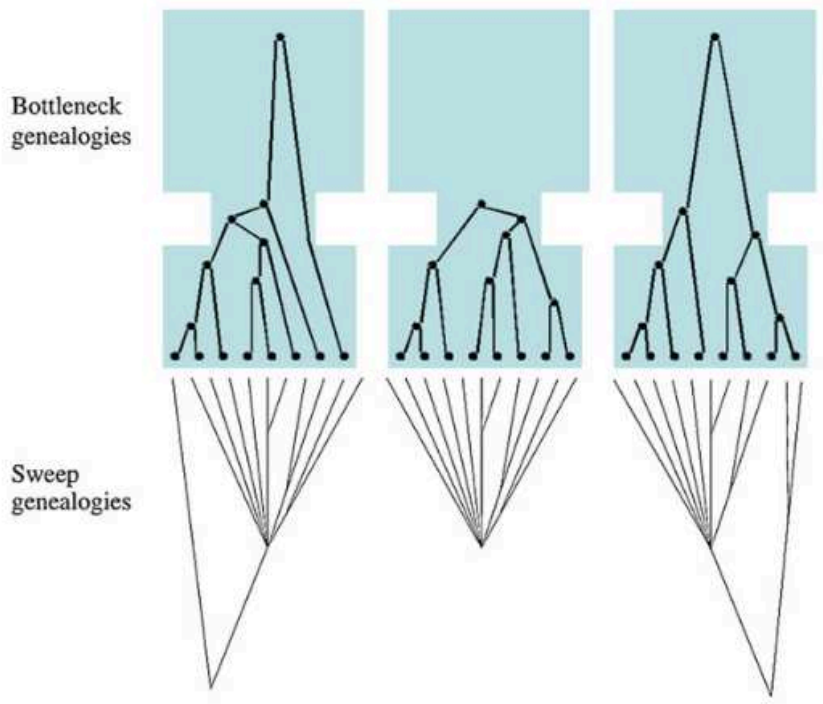



**Figure 31** - Various bottleneck configurations can result in genealogies similar to selective sweep ones.

Selective sweep genealogy presents very short coalescent trees within the region of the beneficial mutation, and trees with long internal branches as we move away from it. Bottleneck ones may present very long internal branches too as long as the ancestral population size is large. (*Figure from Koropoulis et al., 2020*)

Most current approaches do not account for such confounding effects and often struggle to disentangle the overlapping signals of demography and selection (Pavlidis et al., 2008). That is why some precautions should be taken regarding the assumptions about the demographic history of the data use. However, a new method for the detection of selective sweeps has started to grab some more attention : machine learning methods.

In recent years, machine learning has emerged as a promising approach to overcoming these challenges in population genetics, thanks to its capacity to learn complex patterns from large datasets. First applied by Pavlidis et al. (2010), machine learning methods, particularly supervised approaches, have gained traction for detecting selective sweeps. Notable tools, like S/HIC (Schrider & Kern, 2016) and diploS/HIC (Kern & Schrider, 2018) use machine learning to distinguish between different types of sweeps and neutral regions, while other recent approaches employ deep learning to detect post-admixture adaptation (Hamid et al., 2023) or infer selective sweep parameters (Caldas et al., 2022).

In this study, we focus on one such method. More specifically, we propose to observe the possibilities of using convolutional neural networks (CNNs) to perform an object detection task: the detection and localisation of selective sweeps within genomes sampled from populations simulated under six different demographic histories. To do so, we fine tune a pre-trained CNN architecture – a FasterRCNN model (Ren et al., 2016) with a ResNet-50-FPN backbone – explore various parameters and made comparisons of the different possible choices during the training phase : (1) the choice of the type of data to use for the training and to run the inferences for the CNNs, *i.e.* the alternative choice between using a set of summary statistics or running inferences using raw genomic alignments, (2) the impact of the demographic scenario under which the training data has been simulated, (3) the impact of data augmentation during the training phase and (4) the number of backbone layers retrained. We then compare the results of the best version of our CNN against the results obtained using SweepFinder2 (SF2), focusing on 3 main points : (1) the number of False Positive (FP) *i.e.* sweeps detected where no sweep is present, (2) the number of True Positive (TP) *i.e.* sweeps present in the dataset that are actually detected and (3) the distance between the predicted position and the actual position of the beneficial mutation.

The best versions of our trained CNNs end up being the ones trained using *objDet* summary statistics (more on this in Materials and Methods) as input data. Overall, models trained on *BTL* (bottleneck) simulations outperform the other, but also present the highest variation in

terms of performance. Compared to SF2, while generating more false positives, the CNNs also detect more targets and predict positions that are significantly closer to the actual position of the beneficial mutation.

# Materials and Methods

The same datasets generated for Chapter 1 are used here too. The only extra steps are the generation of labels files containing the bounding boxes coordinates of the selective sweep, as well as the use of an alternative way of representing the summary statistics data. Indeed, in an effort to provide the CNN input data in the most efficient format, we tried a slight variation of the basic '*sumStats*' format. Instead of normalizing the statistics values of each simulation across the entire dataset, values are only normalized using the values within the same simulation *i.e*. with the *min* and *max* values of the *sweep* and *neutral* runs of this specific simulation. This way, the intra-genome variance is maximized which should, hopefully, help to highlight the sweep signature. This new representation of the summary statistics data is referred to as *objDet*.

## Detection of selective sweeps using Convolutional Neural Networks

### Object Detection Model and Training

In this study, we use the PyTorch implementation of a FasterRCNN model (Ren et al., 2016) with a ResNet-50-FPN backbone, a resNet making use of a feature pyramid network (Lin *et al*., 2017), known to provide better results for object detection, with about 25.6 M parameters. The CNN is pre-trained on the COCO dataset (Lin *et al*., 2014), a large-scale dataset of over 330 000 pictures of common objects in real life environments, widely used in computer vision research for tasks such as object detection, segmentation, and image captioning. In order for it to be able to localize and detect selective sweeps in genomes, a new training phase on this specific task is necessary. We chose to explore multiple possibilities for this training and four main parameters vary between different versions of our trained CNNs : (1) the type of data use as input (*sorted-rawData*, *sumStats* or *objDet* data), (2) the demographic scenario of the simulations used for training (BTL, CST or EXP), (3) the use or not of data augmentation, *i.e.* slight alterations of the training data *on the fly* during the training to virtually increase the variability of training examples and (4) the number of backbone layers retrained (4 or 5) *i.e.* how deep the training should impact the already trained layers of the model, for a total of 32 different versions. Each model is trained for 150 epochs - one epoch consisting of going through all the training examples one time - with batches of size 8. We use an initial learning rate of $1 \times 10^{-4}$ and a weight decay of $1 \times 10^{-3}$. A

detailed breakdown of all the parameters used for the training of these CNNs is available in **Supplementary Table 7**.

## Targets, bounding boxes and prediction score

For the simulation containing selective sweeps, matching labels files are generated with the coordinates of the bounding boxes (*bbox*) of the spawn of the selective sweep. We defined those bbox as the range, in each direction, starting from the position of the beneficial mutation all the way until the observed genetic diversity gets back to an expected diversity equals to the average diversity observed in the corresponding “twin simulation” evolving under neutrality *i.e*. the neutral simulation launched with the same prior parameters. Bounding box coordinates are first expressed as [class, x_center, y_center, width, height] (**Figure 32**), with the coordinates expressed as normalized values between 0 and 1. Due to the varying sizes of the different types of input data, specific label files are generated for each type of input data. The ‘class’ value in our case is consistent with a value of 1 across all label files as every object to detect is a selective sweep. From there, for a given input data, the desired output of our CNN is a vector containing the corresponding class value (always 1 for us), the predicted bbox coordinates together with an associated prediction score. This score indicates the network’s own confidence in this particular prediction, but it is in no means a probability of the prediction to be correct but rather a self-evaluation of the network.

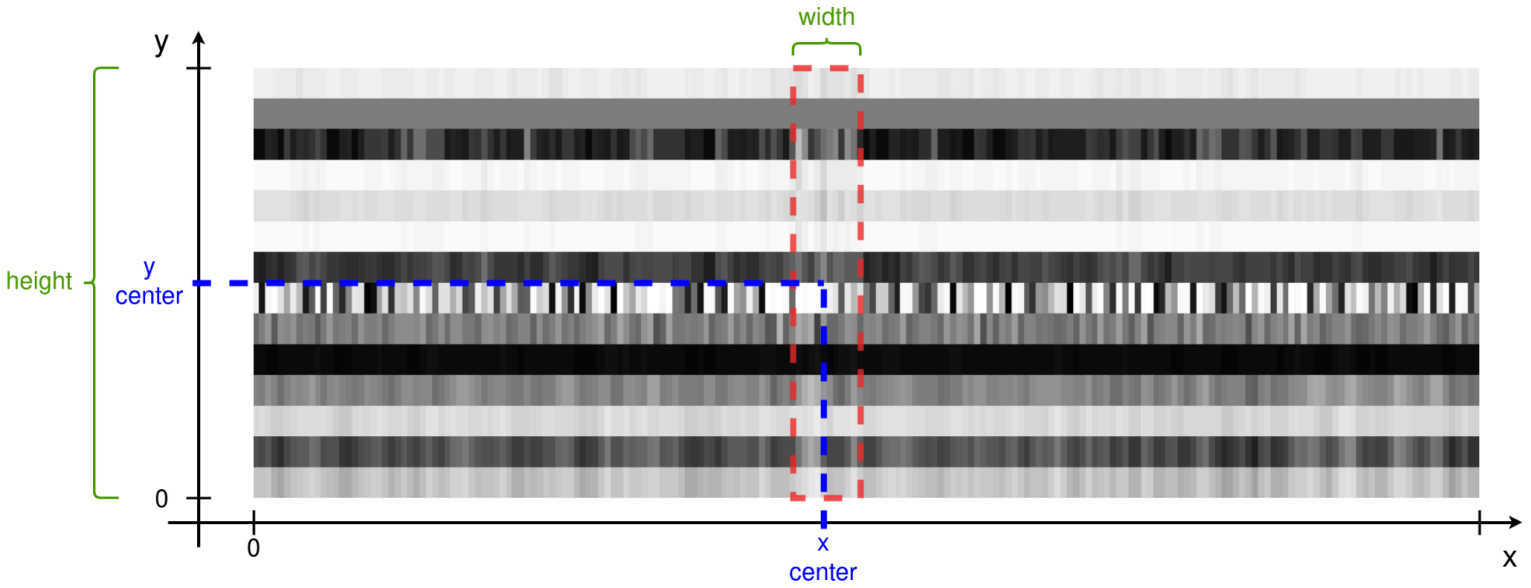


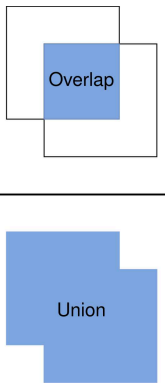

**Figure 32** - Representation of the different parameters of a bounding box on an example *sumStats* matrix.

Bounding boxes (in red) are defined based on their width, their height (both in green) and the x and y coordinates of their center (in blue). These four values are first expressed in pixels and then normalized by the total matrix width and height to be between 0 and 1. Hence, the height of all bounding boxes is equal to the number of rows of the matrix (14 for *sumStats* and *objDet*, 40 for *rawData*) and the y-center coordinate is also constant (*7* for *sumStats* and *objDet*, 20 for *rawData*).

## Evaluating model performance : Correct and False Predictions, Bounding Boxes Position, IoU, Precision and Recall, Average Precision

In order to consider a prediction as *'correct'*, different methods can be used. One way is to simply consider a prediction correct if the prediction is done on an input data where a selective sweep is present (in that case, we are more interested in detection rather than localization). If localization is the focus, one can check the distance between the center of the predicted bbox and the actual position of the beneficial mutation and consider a prediction correct if it is within a certain range of the actual target. Another more commonly used way to determine if a prediction is correct in computer vision is through the computation of IoU, Intersection over Union. IoU is defined as the ratio of the area of overlap (between the predicted and the ground-truth bbox of the target) and the area of union (area that encompassed both the predicted and the ground-truth bbox).

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} =$$




An IoU score above 0.5 is usually considered as a “correct” prediction, as it means that more than half of the predicted bbox overlaps with the ground-truth bbox, but higher scores can be used if an emphasis is put on an accurate localization of the target (Ren *et al*., 2016 used an IoU threshold at 0.7 with the FasterRCNN for example). From there, predictions can be classified as Positive (correct prediction) or Negative (false prediction) and we can compute two metrics to evaluate our models’ performances : **Precision** and **Recall**. Precision is defined as the proportion of predictions that are actually correct, while recall is the proportion of targets that have been found by the model. The former is more used to get a sense of how correct are the predictions made, when the latter focus more on the ability of the model to find the targets. Formally :

$$Precision = \frac{TP}{Nb\ Positive\ Predictions} \qquad Recall = \frac{TP}{Nb\ Targets}$$

Precision and recall are then used to compute **Average Precision (AP)**, an additional metric used to evaluate a model performance as it gives a measure of a model’s ability to balance between correctly identifying targets while minimizing false positives. To compute AP, the Precision-Recall (PR) curve is needed. This curve plots precision against recall for different thresholds and allows one to visualize how a model’s precision reacts to different values of recall. Formally, AP is defined as the Area Under the Precision-Recall Curve (**Figure 33**) :

$$AP = \int_0^1 p(r)d(r),$$

with *p* the precision and *r* the recall.

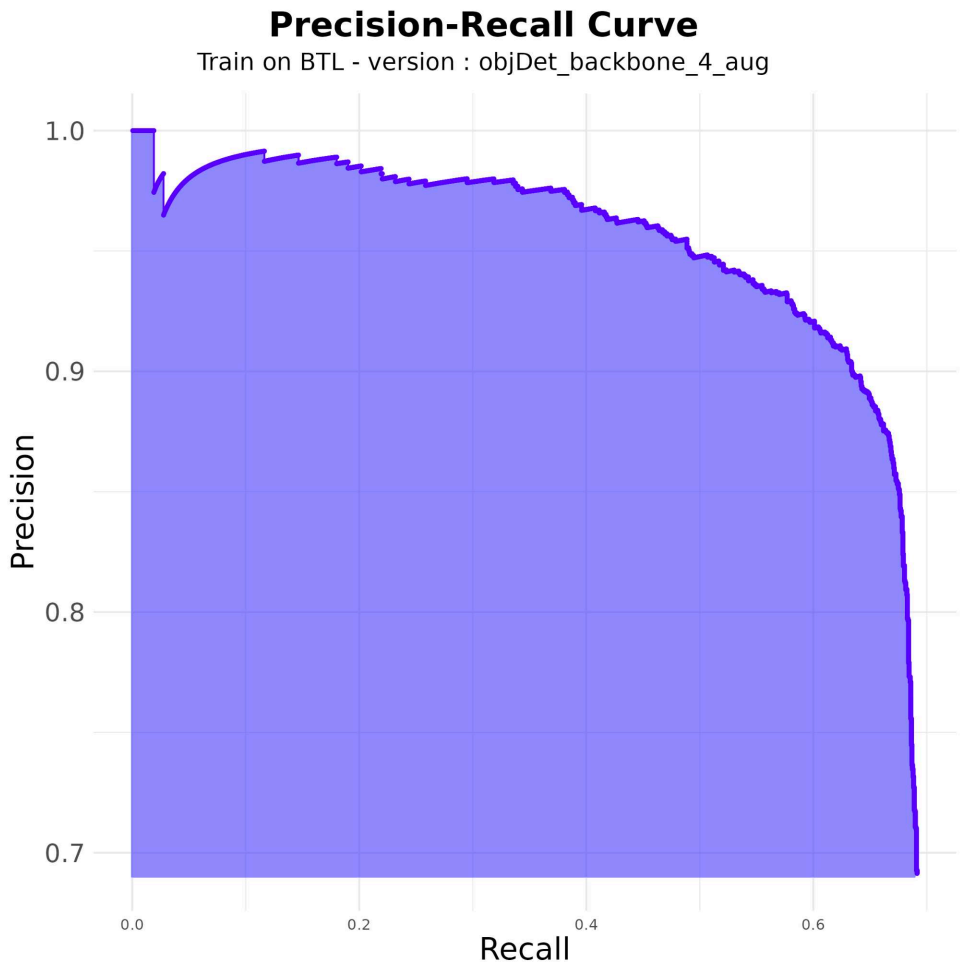



**Figure 33 -** Example of a Precision-Recall curve.

The blue line corresponds to the evolution of Precision-Recall values *i.e.* the precision of the model for a specific value of recall. The blue area is the AUC (Area Under the Curve), in other words the Average Precision.

In this study, we use AP as the main method to evaluate a CNN model's performance. We first compare the '*overall* AP value' (*i.e.* in regards to the 3 basic demographic scenarios) depending on the type of input data used : *sorted-rawData*, *sumStats* or *objDet*. The same comparison is done in regards to the demographic scenario used to train the CNNs, to the number of backbone layers retrained and finally in regards to the use of data augmentation. Finally, we also compute AP values independently for test data of each of the 3 demographic scenarios, in order to get 4 different APs : an "overall AP", an "AP on CST", "AP on BTL" and "AP on EXP". Doing so the different versions of the CNN can be ranked either by "overall AP" or by AP regarding predictions on a specific demographic scenario.

# Comparison of CNN versions

Using the Average Precision, it is possible to evaluate the performance of each trained CNN to get an idea of the impact of the various decisions made during the training phase. We begin by a comparison of the general AP of all CNNs either on specific demographic scenarios, or in the entirety of the test datasets. Then, we focus on the four aforementioned parameters : the demographic scenario associated with the data used for the training, the type of data used as input, the use of data augmentation during the training and finally, the number of backbone layers re-trained.

## Best AP on each scenario

To evaluate the ability of convolutional neural networks (CNNs) to detect selective sweeps under different demographic conditions, we trained separate CNN models on simulated datasets generated under the three distinct demographic scenarios: constant population size (CST), population bottleneck (BTL), and population expansion (EXP). Each trained CNN was then tested on datasets corresponding to all three demographic scenarios (CST, BTL, EXP) to assess its generalization ability across different population histories. The models performance was quantified using Average Precision (AP) scores.

## Best Overall AP

To assess the generalization ability of CNN models trained on genomic data from one demographic scenario to the other, we evaluated their performance across all demographic histories at once. Individual CNN models were trained on simulated data from either CST, BTL or EXP scenarios. These models were then tested across all three demographic scenarios to measure their overall ability to detect selective sweeps, regardless of the specific demographic history used for training. We also compare the use of the three distinct types of input data (*sumStats*, *objDet* and *sorted-rawData*) as well as the number of retrained backbone layers (4 or 5), and the use of data augmentation.

Once again, to quantify performance across all demographic scenarios simultaneously, we computed the overall Average Precision (AP) for each CNN. These comparisons allowed for a detailed examination of the CNNs' robustness and the effect of each of those choices on a model’s performances.

# Convolutional Neural Networks *vs.* SweepFinder2

In order to get a better idea of the CNNs performance, we compare the results of the best version of our CNN, *i.e.* the version which scores the highest overall AP, against SweepFinder2 (SF2 - DeGiorgio *et al*., 2016), another software used to detect selective sweeps from genomic data. SF2 operates similarly to SweepFinder (Nielsen et al., 2005) by analyzing the site frequency spectrum (SFS) and computing a composite likelihood ratio to compare expectations under a null hypothesis (neutral drift) versus those under positive selection leading to a selective sweep. SweepFinder identifies deviation from neutrality by comparing the local SFS, computed within a sliding window along the genome, to the neutral SFS expected under neutrality. At each genomic position, the probability of the observed allele frequencies is calculated under both a neutral and a sweep model. A likelihood ratio test (LRT) is then performed to compare the likelihood of the observed SFS under a sweep model to that under a neutral model.

## False Positive Rate on *neutral* simulations

To compare the two methods, we begin by testing each of them for False Positive (FP) on a test dataset of 6 000 simulated chromosomes, 1 000 of each of the six possible demographic scenarios (CST / BTL / EXP, MIG / MGB / MGX). Here, a FP is defined as a *neutral* simulation (*i.e.* without selection) where a method incorrectly predicts the presence of a selective sweep. The results are separated in 4 classes : simulations where a sweep is detected by both methods, simulations where a sweep is only found by the CNN, simulations where a sweep is only found by SF2 and finally, simulations where no sweep is detected. Comparisons are done at various levels of confidence scores for the CNN prediction, from 0 to 0.3 to 0.5, to get a better sense of the confidence of the CNN on false predictions. It should be noted here that we do not use the IoU score as a way to evaluate if a prediction is correct or not. Rather, any prediction made for a given input is considered, regardless of its IoU score.

## Detection of Selective Sweeps

The methods are then tested for true positives : the same comparison is done with a dataset of 6 000 *sweep* simulations *i.e.* chromosomes containing selective sweeps. The number of sweeps found by each method is compared in the same fashion, with 4 classes : simulations where a sweep is found by both methods, simulations where a sweep is only found by the CNN, simulations where a sweep is only found by SF2 or simulations where no sweep is found at all. To keep things consistent, we once again do not take into account the IoU

scores in this comparison. Thus, a True Positive (TP) is defined here as any simulation where a selective sweep is detected, regardless of the IoU score of the prediction. We also increased the prediction score threshold of the CNN from 0 to 0.3 and to 0.5, in order to investigate how the CNN's confidence affects the accuracy of sweep detection. For each threshold level, we compared the number of sweeps detected by both methods. Additionally, we examined prediction scores associated with sweeps and neutral simulations to assess the reliability of the CNN's predictions between selective sweeps and neutral simulations.

## Localization of Selective Sweeps

We have not yet considered the localization aspect of the sweep detection methods so far - ignoring the IoU scores in the evaluations of both False Positives and True Positives. However, not only do we aim for methods able to detect the presence of a selective sweep within genomic data, our goal is also to precisely localize the position of the beneficial mutation from which the sweep occurs. However, SF2 does not provide any equivalent of the IoU score of a CNN. Thus, to evaluate the ability of both methods to accurately localize selective sweeps, we compared the models based on the predicted sweep positions. For the CNN, the predicted location is represented by the center of the bounding box surrounding the sweep, whereas for SF2, it is the predicted position of the mutation responsible for positive selection.

First, we defined a "*distance to mutation*" method, where a sweep as "correctly localized", *i.e.* a True Positive, when the distance between the predicted position (from SF2 or the center of the CNN's bounding box) and the actual position of the beneficial mutation is less than 24,186 base pairs (bp), the average sweep size in our test dataset.

We also tested a secondary criterion, "*within true bbox*", where a prediction is considered a TP if the predicted position fell within the true bounding box of the sweep present within the simulation. This alternative approach, mimicking in essence the IoU score, allowed us to further compare the performance of the CNN and SF2, offering insights into how each method fares into different tasks: precisely localizing the exact position of the beneficial mutation or predicting a position at the very least impacted by a selective sweep. In both cases, we made the assumption that the beneficial mutation is roughly in the center of the bounding-box and that its effects are roughly equal in both directions.

## Predictions Distance to the Beneficial Mutation

As a final step in our comparisons, we ought to evaluate the localization accuracy of the predicted selective sweeps by both the CNN and SF2 models. We compared the predicted positions to the position of the actual beneficial mutation. For the CNN, we measure the distance in two different ways : either the distance between the center of the predicted bounding box and the position of the mutation, or as the distance between the centers of predicted and ground-truth bounding boxes. For SF2, we compare the predicted position directly with the true position.

## Exploration of Parameters Space

Finally, to investigate whether certain simulation parameters influenced the ability of either method to detect sweeps, we conducted an exploration of the simulation's parameter space. First, we evaluated key variables (population effective size, the age of demographic events, the timing and duration of selective sweeps, …) with a focus on detecting patterns in FP occurrence on the *neutral* dataset. We then explored how the simulation parameters influenced the actual detection of selective sweeps by analyzing in greater details the simulations parameters of the *sweep* dataset, for each method of determining a correct prediction (*sweep found*, *distance to mutation* and *prediction within true bbox*), especially the duration of selective sweeps (*i.e.*, the number of generations between the emergence of a beneficial mutation and the sampling time) and the width of the ground-truth bounding box (*i.e.*, the sweep size) as these parameters appeared to be the most relevant in regards to sweep detection success. In both cases, only the more interesting plots are present. It should also be noted that the duration of sweeps is expressed in this study in units of time scaled on the effective size of the population.

# Results

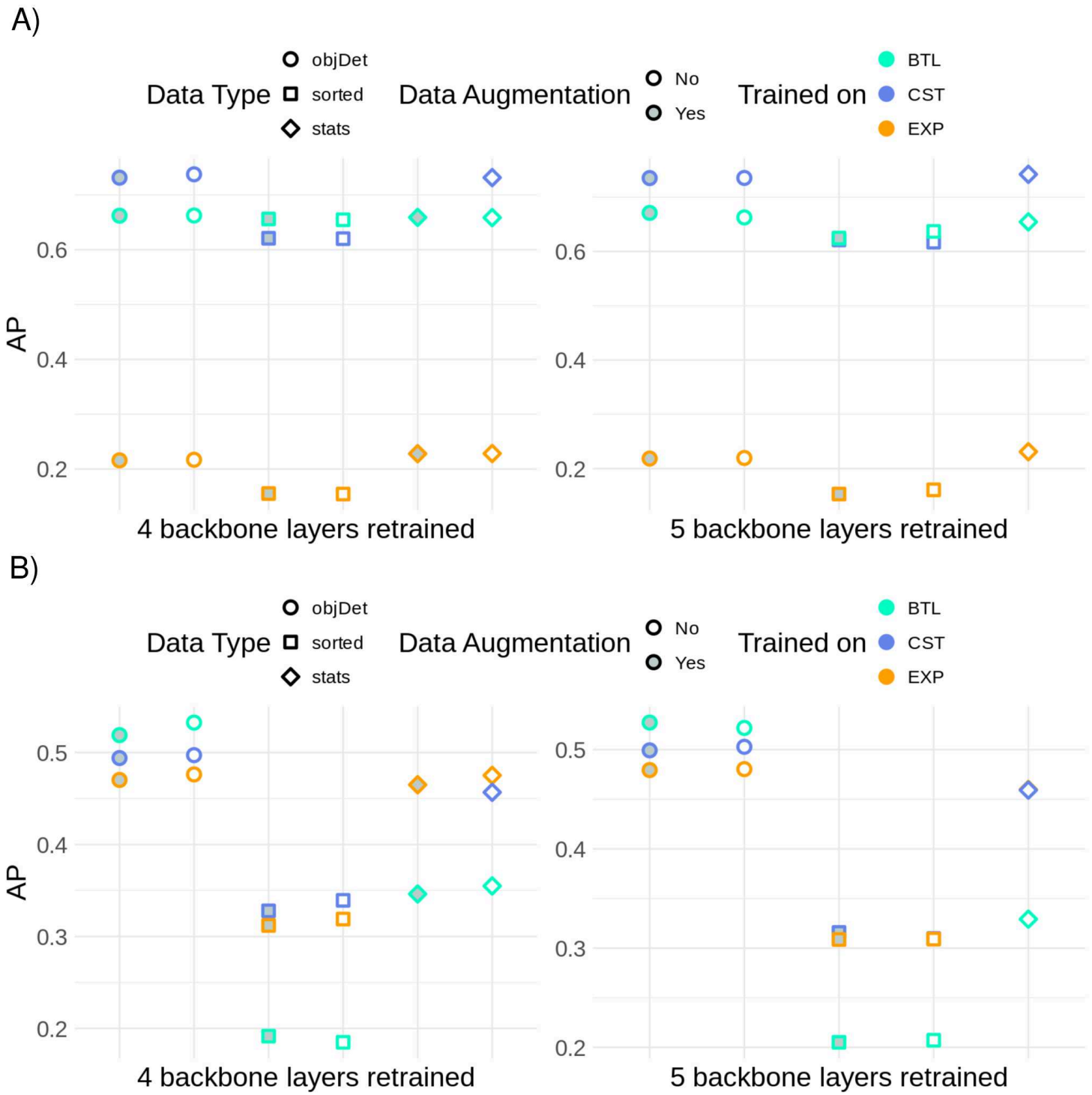



**Figure 34** - Comparison of the Average Precision (AP) of all 32 different versions of the trained CNNs.

Average Precision computed using **A)** only on the simulations sharing the same basic demographic scenario as the one used for the training phase or **B)** using all 6 000 *test* simulations.

# Comparison of CNN versions

## Best AP on each scenario

To assess the ability of our CNNs to detect selective sweeps across different demographic histories, we train individual CNNs on simulations corresponding to three distinct demographic scenarios: constant population size (CST), population bottleneck (BTL), and population expansion (EXP). We test each CNN on *test* data corresponding to each demographic scenario, calculating Average Precision (AP) scores for each case (**Figure 34 - A, Figure 35** - see **Supplementary Table 8** for detailed values of mAP and mAR (mean Average Recall) for each CNNs). This setup allowed us to directly compare the ability of CNNs to generalize across different demographic contexts.

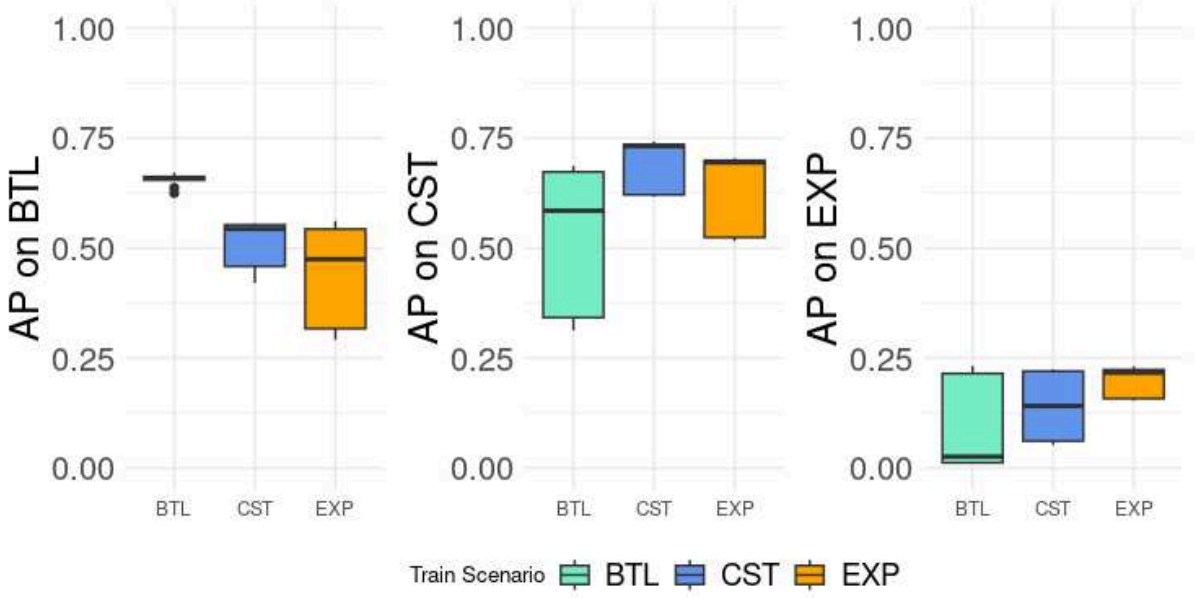



**Figure 35** - Average Precision (AP) computed on CST, BTL or EXP *test* data.

AP values are organized according to the demographic scenario of the simulation used to train the CNNs. The x-axis is the scenario of the simulations used for the training of the CNN.

For the CST scenario, the best performance was observed for CNNs trained on CST data. The top-performing CNN, which achieved an AP score of 0.742, was trained on *sumStats* data, with five retrained backbone layers and no data augmentation. Other top CNNs trained on CST data, including those using *objDet* data and varying configurations of backbone layers, achieved comparable AP values (ranging from 0.735 to 0.738).

In the BTL scenario, CNNs trained on BTL data clearly outperformed those trained on CST or EXP data. The top four CNNs, all trained on *objDet* data, achieved AP scores ranging from 0.662 to 0.671, with minimal variance in performance across models. In contrast, CNNs trained on CST and EXP scenarios showed significantly lower and more variable AP scores, with the worst-performing CNN trained on EXP data scoring an AP of 0.562, approximately 10% lower than the weakest BTL-trained model.

In the EXP scenario, the overall performance of all CNNs was lower compared to the CST and BTL scenarios, with no model achieving an AP score above 0.25. The highest-performing model, trained on BTL data using *objDet* input with four retrained backbone layers, scored an AP of 0.232, slightly outperforming the best EXP-trained CNN, which achieved an AP of 0.231.

## Best overall AP

To further evaluate the generalization ability of our CNN models, we tested whether a CNN trained on data from a single demographic scenario could perform well on genomic data originating from populations with different demographic histories. To do this, we computed the overall AP of each model across all scenarios simultaneously (**Figure 34 - B and Figure 36**).

### Effect of Train Scenario

When examining the effect of the training scenario on overall performance, CNNs trained on CST and EXP simulations achieved similar average AP values. CNNs trained on CST had an average overall AP of 0.420 ± 0.085, while those trained on EXP scored 0.418 ± 0.064. In contrast, CNNs trained on BTL data had a lower average overall AP of 0.356 ± 0.147 and exhibited the highest variance in performance. However, the BTL-trained models also produced both the worst and best overall AP scores. The highest AP of 0.533 was achieved by a BTL-trained CNN using *objDet* data with four retrained backbone layers and no data augmentation. The lowest AP of 0.185 came from a BTL-trained CNN using sorted data, also with four retrained backbone layers and no data augmentation.

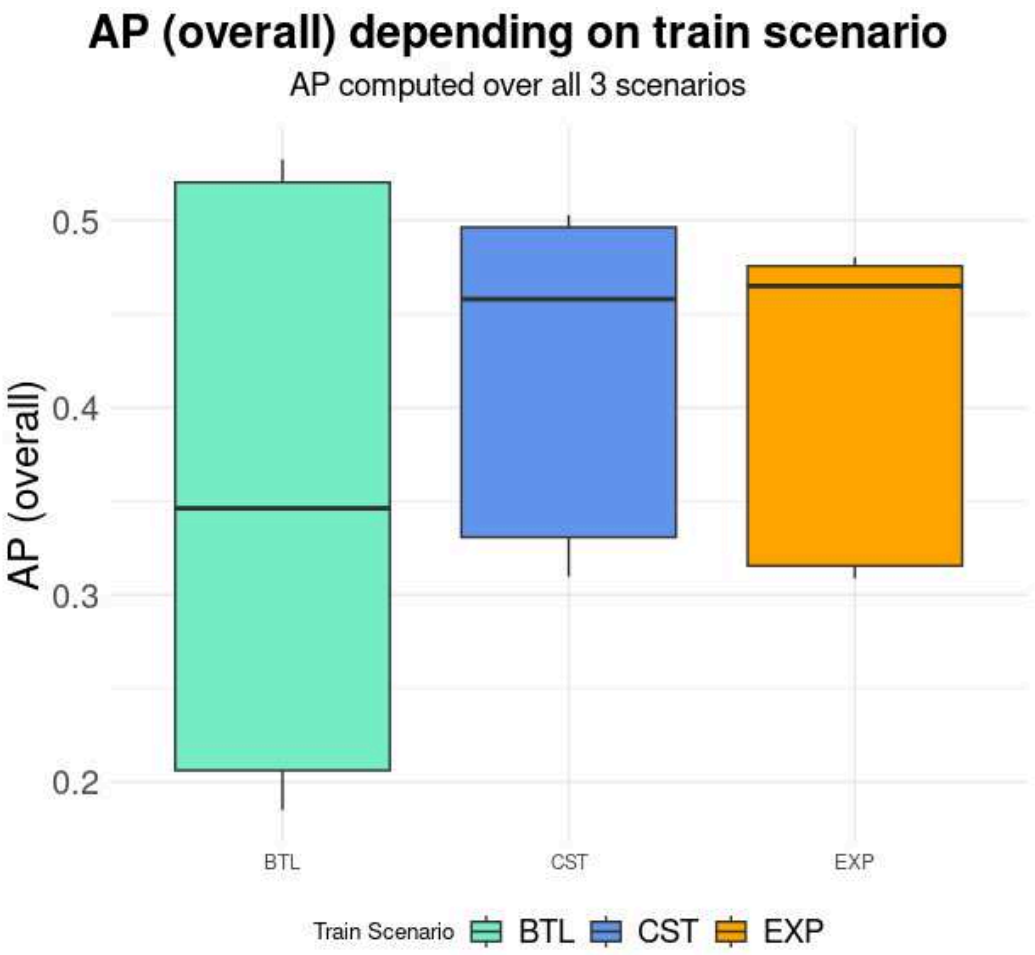



**Figure 36** - Comparison of the overall AP (computed over all 3 demographic scenarios) between CNNs trained on BTL, CST or EXP simulations. The x-axis is the scenario of the simulations used for the training of the CNN.

**Effect of Data Input Type**

When comparing the performance of CNNs based on the type of input data, a clear ranking emerged across all scenarios (**Figure 37 - A**). CNNs trained on *objDet* data outperformed the other data types, achieving an average overall AP of 0.500 ± 0.021. CNNs trained on summary statistics (*sumStats*) data followed with an average AP of 0.418 ± 0.063, while those trained on *sorted-rawData* performed the worst, with an AP of 0.277 ± 0.060. These results suggest that *objDet* data provides the most informative input for CNN training in this context.

### Effect of Data Augmentation

The impact of the number of retrained backbone layers on overall performance was minimal. CNNs with four retrained backbone layers had an average overall AP of 0.397 ± 0.109, while those with five retrained layers showed a similar AP of 0.385 ± 0.112 (**Figure 37 - B**). This indicates that increasing the number of retrained backbone layers from 4 to 5 did not significantly improve model performance.

### Effect of the Number of Retrained Backbone Layers

Similarly, data augmentation had little effect on the overall AP values. CNNs trained with data augmentation scored an average overall AP of 0.401 ± 0.108, while those trained without augmentation achieved an AP of 0.390 ± 0.116 (**Figure 37 - C**). This suggests that, for the demographic scenarios tested, data augmentation did not substantially enhance the generalization ability of the CNNs.

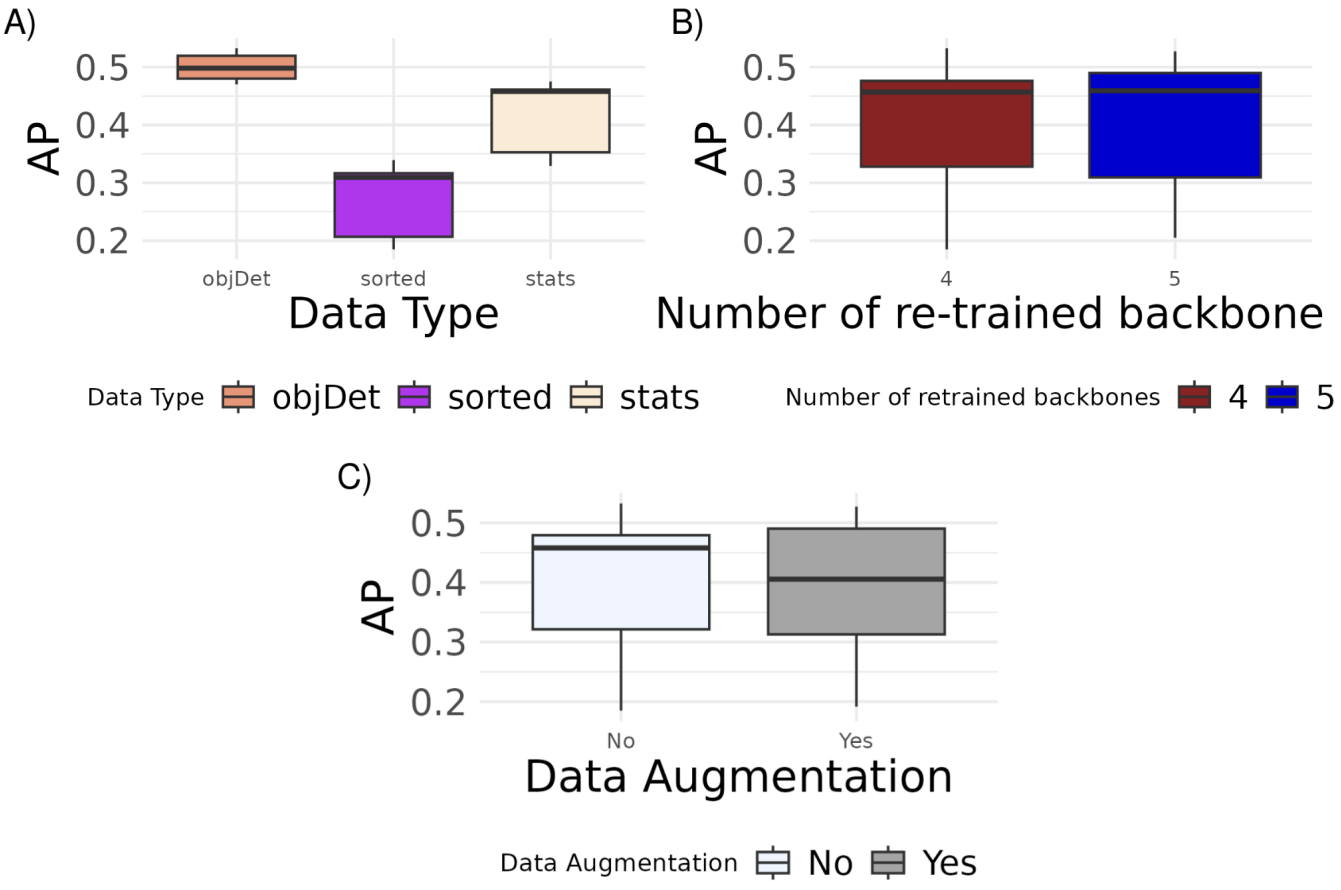



**Figure 37** - Comparison of the effects of different types of data inputs, number of re-trained backbones and the use or not of data augmentation on the Average Precision (*computed over all 3 demographic scenarios*)

AP are compared between CNNs **A**) using either *sorted-rawData*, *sumStats* or *objDet* as input data, **B**) with either 4 or 5 re-trained backbone layers and **C**) using or not data augmentation.

# Convolutional Neural Networks vs. SweepFinder2

In our previous comparisons, the CNN model that performed best in terms of overall Average Precision (AP) was the one trained on BTL simulations using *objDet* as input data, with four retrained backbone layers and no data augmentation. This model achieved the highest AP on BTL simulations (0.662) and EXP simulations (0.232), and still performed well on CST simulations, with an AP of 0.687, slightly above the average of 0.673 across all versions. Given its strong performance across various scenarios, this CNN model was selected for comparison with the widely-used SweepFinder2 (SF2) software.

## False Positive Rate on *neutral* simulations

To begin this comparison, we focused on the false positive (FP) rate, defined as the number of simulations where at least one selective sweep prediction is generated by either method when applied to a *neutral* test dataset. While accurate identification of sweeps (maximizing recall) is crucial, precision is also important to avoid excessive false positives, where predictions incorrectly label neutral regions as sweep events.

When using a prediction score threshold of 0 for the CNN – meaning all predictions are considered regardless of confidence (**Figure 38 - A**) – the number of false positives generated exclusively by the CNN ranged from 300 to 400 on BTL, MGB, MIG, and EXP scenarios, similar to the number of FPs generated by both methods combined. On CST simulations, the CNN produced 476 false positives, nearly twice the 277 FPs found by both methods. For MGX simulations, the CNN produced 281 false positives compared to 610 found by both methods. Across all scenarios, SweepFinder2 alone produced relatively few FPs, ranging from 145 to 50 depending on the scenario. Overall, the CNN displayed a False Positive Rate (FPR) of 0.783 *vs.* 0.476 for SF2.

When we increased the CNN's prediction score threshold to 0.3, thereby discarding less confident predictions, the number of false positives dropped considerably, with the CNN generating no more than 60 to 170 FPs at worst. Incidentally, the number of FPs generated by both methods together also decreased, while the number produced by SweepFinder2 alone increased proportionally (**Figure 38 - B**). With this new threshold, the CNN FPR = 0.318.

At a threshold of 0.5 (**Figure 38 - C**), the number of FP produced by the CNN is around 80 on bottleneck scenarios (88 on BTL and 78 on MGB), while both methods combined generated 170 FPs on each. Meanwhile, SweepFinder2 alone generated approximately 300

FPs. On CST, MIG, and EXP simulations, the CNN generated between 15 and 33 FPs, while SweepFinder2 alone produced significantly more (329, 426, and 428 FPs, respectively). On MGX simulations, the CNN generated 65 FPs, while both methods combined produced 199, and SweepFinder2 alone accounted for 474 FPs. Here, the CNN's FPR drops to 0.151.

Raising the threshold further, up to 0.9 or 1, would eliminate all false positives from the CNN's predictions. However, setting the threshold this high would not be practical for any realistic use, where a balance between precision and recall must be maintained. Additional comparisons of FP rates at different thresholds (in 0.1 increments) can be found in **Supplementary Figures 3 to 12**.

## Detection of selective sweeps

We next evaluated the detection of actual selective sweeps using a test dataset of 6,000 simulations. Initially, with a prediction score threshold of 0 – considering all proposed predictions by the CNN (**Figure 39 - A**) – both the CNN and SF2 identified between 600 and 880 sweeps in each scenario. CST simulations had the highest overlap, with 878 sweeps detected by both methods, while the lowest overlap was in MGX simulations with 602 sweeps. The CNN significantly outperformed SF2 in detecting sweeps on BTL simulations (329 vs. 6), MGB (318 vs. 2), CST (85 vs. 29), MIG (171 vs. 16), and MGX (332 vs. 34). However, on EXP simulations, SF2 identified more unique sweeps than the CNN (160 vs. 103). Overall, the CNN achieves here a True Positive Rate (TPR) of 0.947 *vs*. 0.765 for SF2.

With a prediction score threshold of 0.3 (**Figure 39 - B**), both methods detected between 600 and 700 sweeps on BTL, MGB, CST, and MIG simulations (e.g., 588 for BTL, 616 for MGB, 684 for CST, and 654 for MIG). However, the number of shared detections dropped to around 400 on EXP (397) and MGX (407) simulations. While the CNN continued to outperform SF2 on BTL and MGB scenarios (297 vs. 78 for BTL, 288 vs. 65 for MGB), the number of sweeps exclusively found by SF2 increased in other scenarios. Indeed, SF2 found 223 sweeps not found by the CNN in CST, 168 in MIG, 483 in EXP, and 229 in MGX. The higher threshold also led to an increase in missed sweeps, with both methods now failing to detect between 30 and 70 sweeps across most scenarios, and up to 165 sweeps missed in MGX. Using this threshold on the predictions scores of the CNN, its TPR = 0.685, a bit less than the 0.765 of SF2. It should be noted that using a threshold of 0.25 instead, the CNN TPR = 0.770.

At a threshold of 0.5 (**Figure 39 - C**), the number of sweeps jointly identified by the two methods remained high (500-550 for BTL, MGB, CST, and MIG scenarios), but decreased to 242 in EXP and 277 in MGX. Although the CNN continued to find more sweeps in BTL (273) and MGB (267), the gap widened in other scenarios, with SF2 identifying 379 sweeps not found by the CNN in CST, 307 in MIG, 359 in MGX, and up to 638 in EXP. Once again, the CNNs TPR drops, here to 0.580. As expected, as the threshold increases, more and more sweeps that were initially detected only by the CNN are now classified as missed by both methods and its TPR decreases.

Again, increasing the prediction score threshold for the sake of it does not make much sense here, as a selective sweep detected with a perfect prediction score is highly unlikely and discarding even predictions with a medium prediction score would most likely lead to missing targets. However, for the sake of completion, such comparison at different thresholds (in 0.1 increments) can still be found in **Supplementary Figures 13 to 22**.

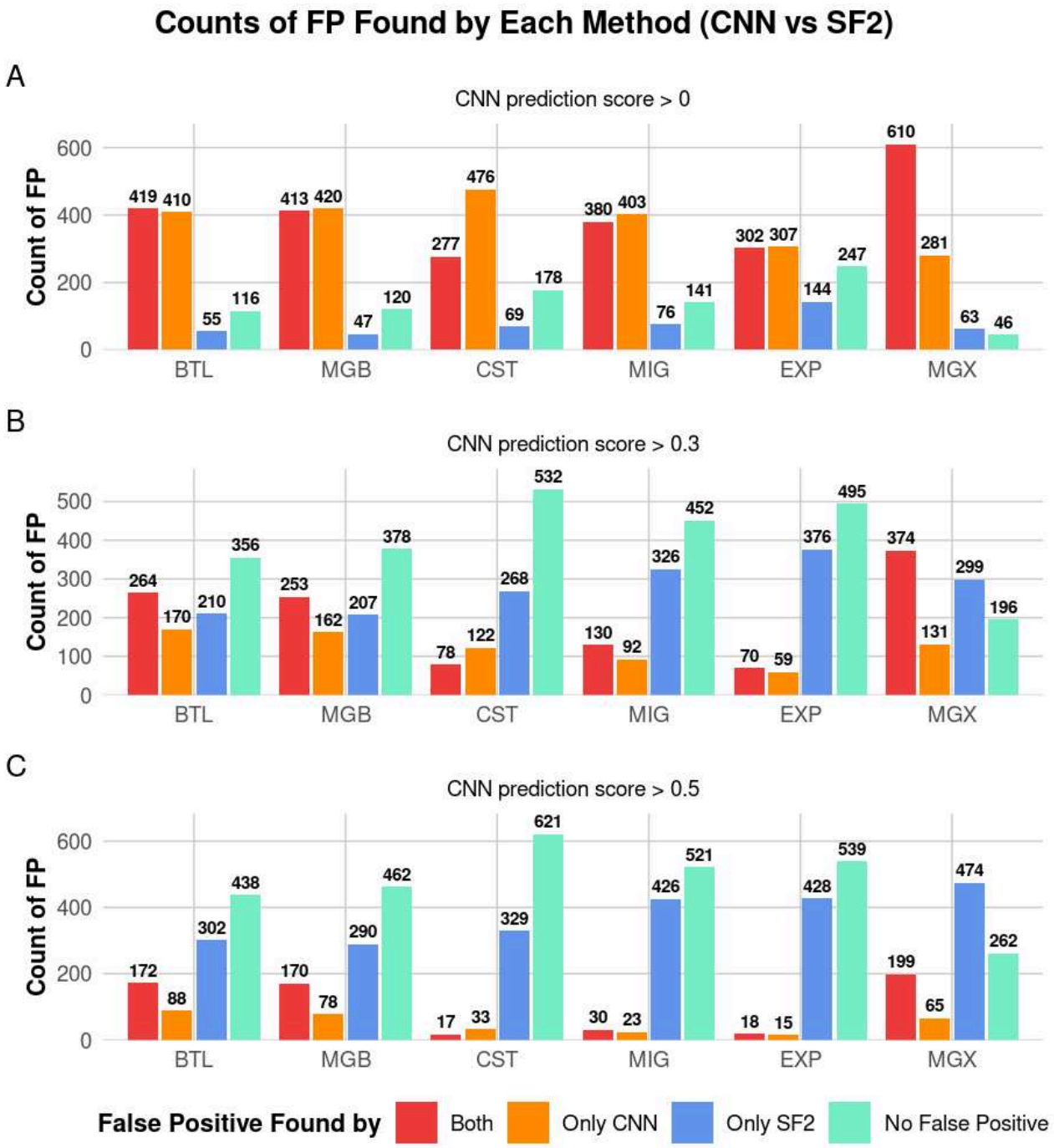



**Figure 38** - Number of **False Positive (FP) generated** by each method.

Predictions are classified as FP found by both methods (red), only by the CNN (orange), only by SF2 (blue) or no FP (green). Various threshold of prediction scores for the CNN : **A)** prediction score > 0 (all predictions), **B)** prediction score > 0.3 and (**C**) prediction score > 0.5.

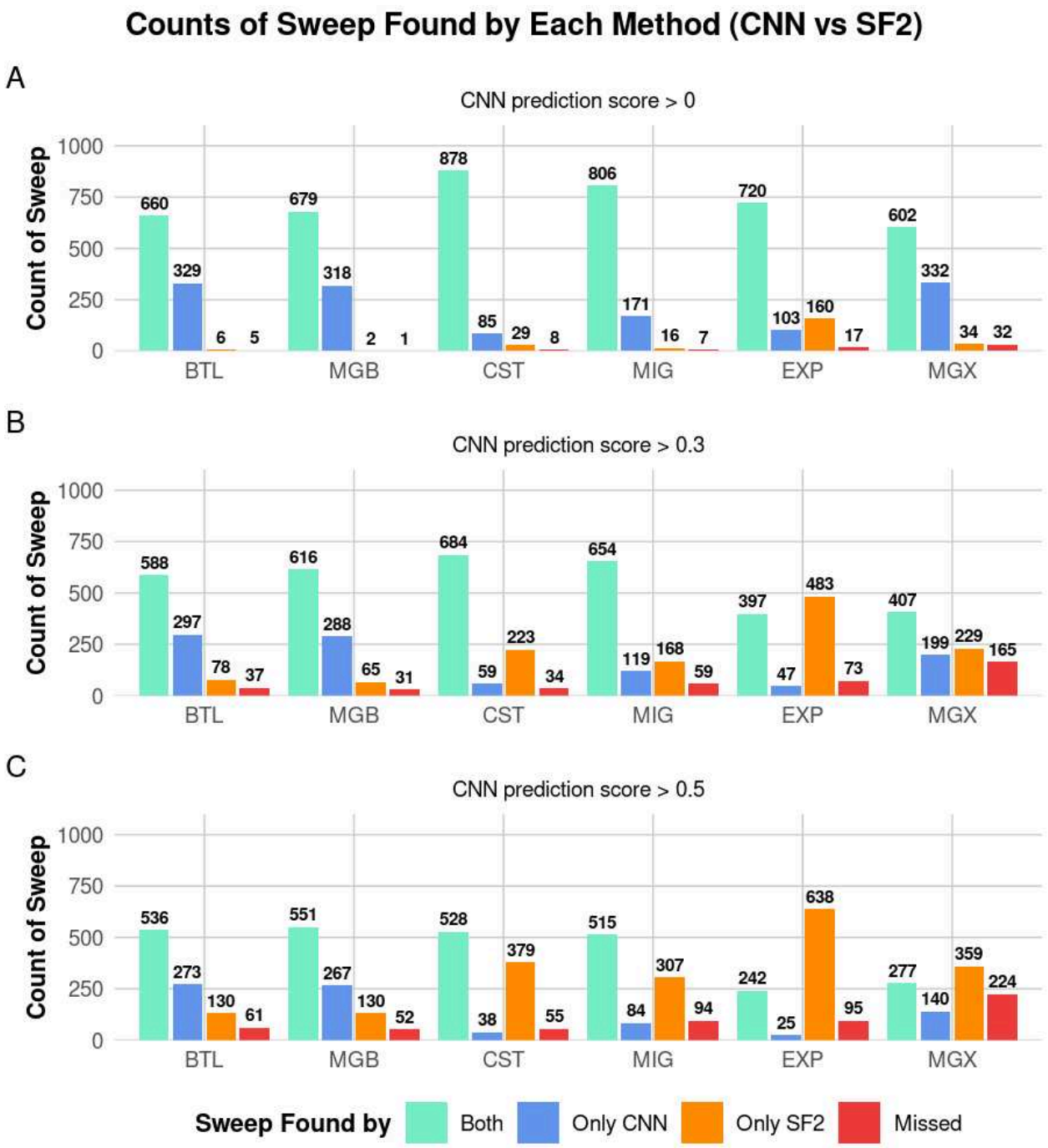



**Figure 39** - Number of **sweeps found** by each method.

Predictions are classified as sweeps found by both methods (green), only by the CNN (blue), only by SF2 (orange) or sweeps missed by both methods (red). Various threshold of prediction scores for the CNN : **A)** prediction score > 0 (all predictions), **B)** prediction score > 0.3 and **C)** prediction score > 0.5.

As a wrap up to this comparison of the FP and TP, it is worth noting that the prediction scores associated with detected sweeps were significantly higher than those associated with neutral data (average sweep score = 0.629 ± 0.315 vs. neutral score = 0.325 ± 0.215, Wilcoxon test p-value < 2.2e-16), indicating better model confidence for true positive sweeps (**Figure 40**).

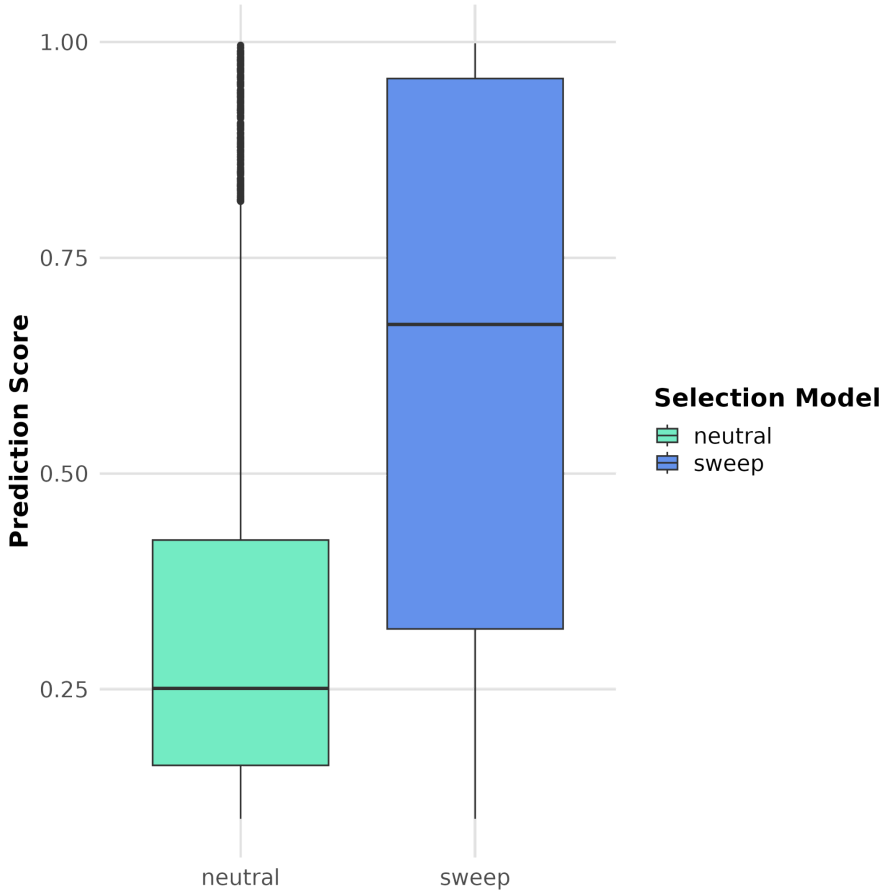



**Figure 40** - Distribution of prediction scores associated with the CNN predictions.

Prediction scores are values outputted along the CNN predicted bbox coordinates indicating how confident the network is in a given prediction. Prediction scores are computed on *neutral* (green) or *sweep* (blue) test datasets.

## Localization of Selective Sweeps

Since both the CNN and SF2 are not only capable of detecting selection, but are more importantly capable of predicting the position of the selective sweeps. We evaluated these methods based on their ability to accurately localize the sweep or the mutation responsible for positive selection.

We first defined a sweep as "found", so a "correct prediction", when the distance between the predicted position (for SF2) or the center of the CNN's predicted bounding box and the actual position of the beneficial mutation was less than 24 186 bp, the average sweep size in our test dataset (**Figure 41 - A**). Under this criterion, the CNN consistently outperformed SF2 on BTL simulations, detecting 451 sweeps not found by SF2, compared to just 71 found only by SF2. On MGB simulations, the CNN detected 429 sweeps not found by SF2, while SF2 found 81 not found by the CNN. CST and MIG results were similar, with around 600 sweeps detected by both methods, and the CNN finding 260-330 sweeps uniquely, while SF2 identified fewer (34 on CST, 23 on MIG). For EXP simulations, the CNN detected 252 "unique" sweeps compared to 115 by SF2. Finally, on MGX, the CNN found 401 sweeps exclusively, while SF2 detected only 77. Overall, the CNN ended up with 4335 TP for 1348 FP and 317 sweeps not found, so more than 75% of correct predictions out of the 5683 sweep predicted, and about 5% of the sweeps totally missed. As for SF2, he scores 2718 TP for 1874 FP and 1408 sweeps missed. In other words, about 23% of the sweeps are missed, and among the sweeps predicted, about 40% are too far from the target for the prediction to be considered correct.

We then considered a second criterion : a sweep was considered "found" if the predicted position fell within the actual bounding box of the sweep (**Figure 41 - B**). Under this definition, results were similar for BTL and MGB simulations. However, on CST, MIG, EXP, and MGX, the number of missed sweeps by both methods increased, especially in the EXP and MGX scenarios, with 498 and 580 sweeps missed, respectively. Interestingly, using this method increased the number of sweeps found exclusively by the CNN in CST and MIG scenarios, showing its potential advantage in these cases. Using this method, the CNN achieves 3490 TP, 2193 FP (and missed the same 317 sweeps), so a bit more than 60% of its predictions are within the true bbox. For SF2, 1748 predictions are TP for 2844 FP (and missed the same 1408 sweeps), which is about 40% of correct predictions.

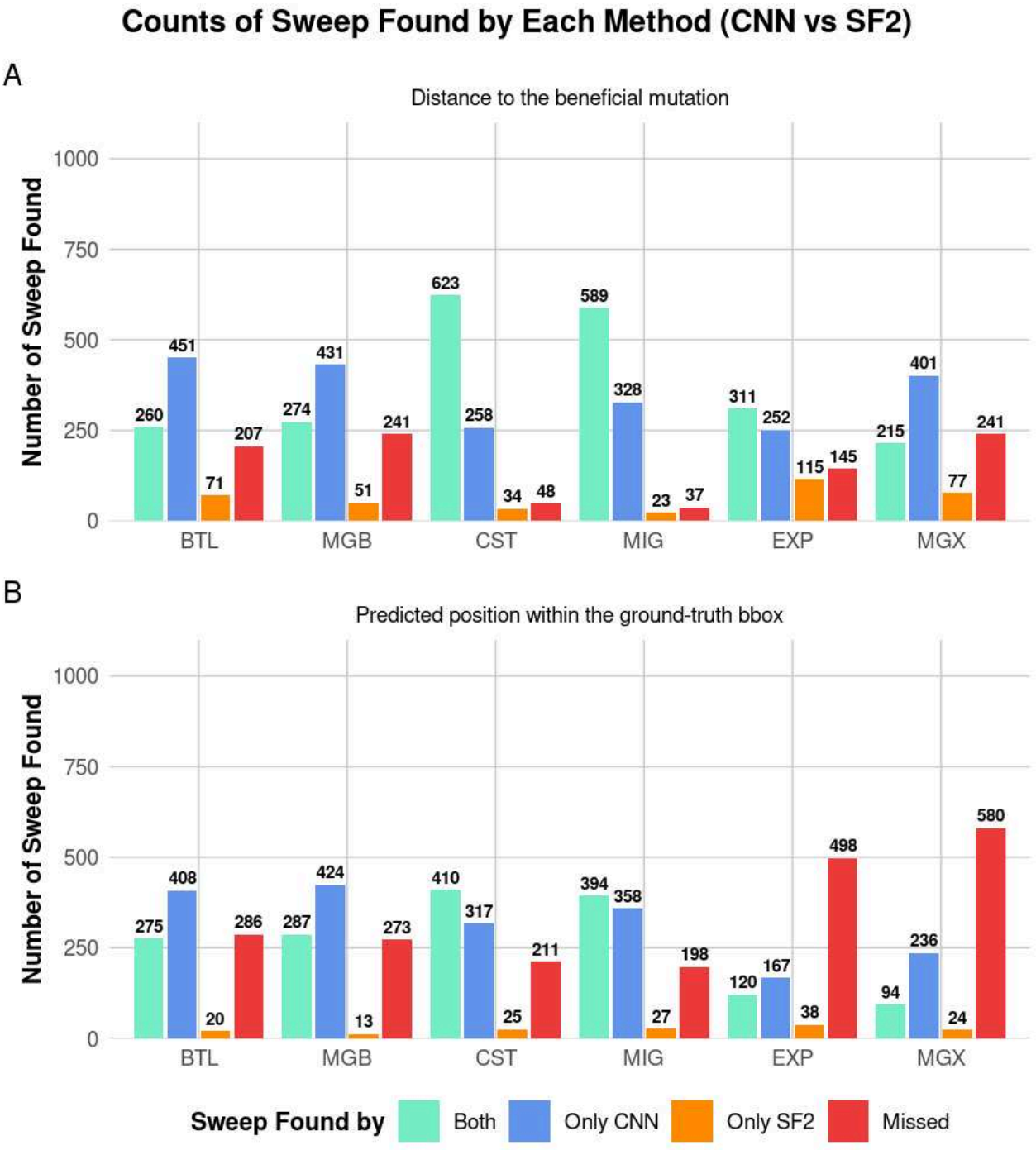



**Figure 41** - Number of **sweeps found** by each method depending on the **criterion used to define a prediction** as correct.

Predictions are classified as sweeps found by both methods (green), only by the CNN (blue), only by SF2 (orange) or sweeps missed by both methods (red). Sweeps are considered detected if : **A)** the **distance between the predicted position and the beneficial mutation** is less than the average size of a selective sweep (average computed within the test dataset) or **B)** the **predicted position is within the limits of the bounding-box** of the sweep.

# Exploration of Simulation Parameter Space

Before delving into the analysis of the parameters associated with FP and correct detection of selective sweeps, It is important to remember that due to the way our simulations are set up, the timing of the split of the ancestral population, as well as the timing of demographic events are expressed in regards to the current population size of the corresponding population. Thus, smaller values for BTL and MGB simulations compared to the other scenarios are to be expected. Moreso, as the next section is dedicated to the analysis of the parameters for the detection on *sweep* data, it is also important to remember that older splits widens the prior from which we draw the timing of onset of the selection, and therefore, produces on average older sweeps that are more eroded over time.

## Parameter Space of predictions on *neutral* data

We first focused on the false positives (FPs) generated by the models on the neutral dataset, testing for significant differences in the distributions of current and ancestral population sizes, as well as the ages of demographic events and ancestral population splits, between simulations where an FP was generated versus those where no incorrect prediction occurred (see **Supplementary Table 9** for Wilcoxon tests p-values and results).

For the CNN, FPs were primarily observed in BTL, MGB, and MIG simulations with smaller population sizes, older demographic events, and older population splits. In the EXP scenario, FPs were associated with simulations featuring smaller current population sizes, while in MGX, FPs were linked to older population splits (**Figure 42**). The pattern for SF2's FPs was more complex. In BTL and MGB simulations, FPs occurred in simulations with smaller populations and older demographic events and splits. However, in CST and MIG scenarios, while FPs also occurred in simulations with older demographic events and splits, they were associated with larger population sizes (**Figure 43**). Finally, we compared the FPs generated by both models on the same simulations. Most FPs were found in BTL, MGB, CST, and MIG simulations with smaller population sizes, older demographic events, and older splits; in EXP simulations with smaller populations and more recent demographic changes; and in MGX simulations with older splits (**Figure 44**).

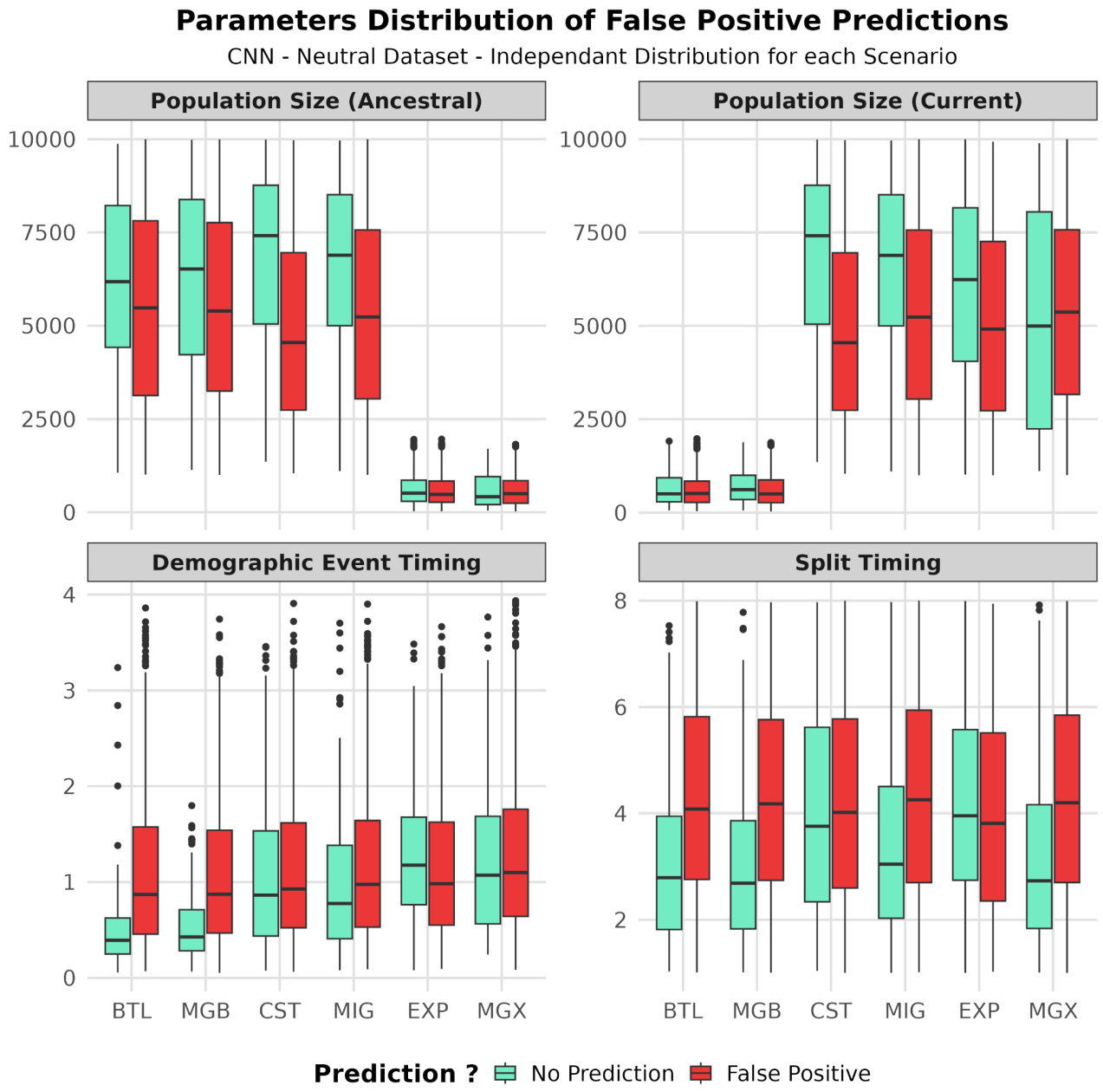



**Figure 42** - Distribution of Parameters for the **False Positive generated by the CNN on *neutral* test dataset**.

Population sizes are expressed in number of individuals. Demographic Event and Split Timings are expressed in regards to the current population size of the corresponding simulation.

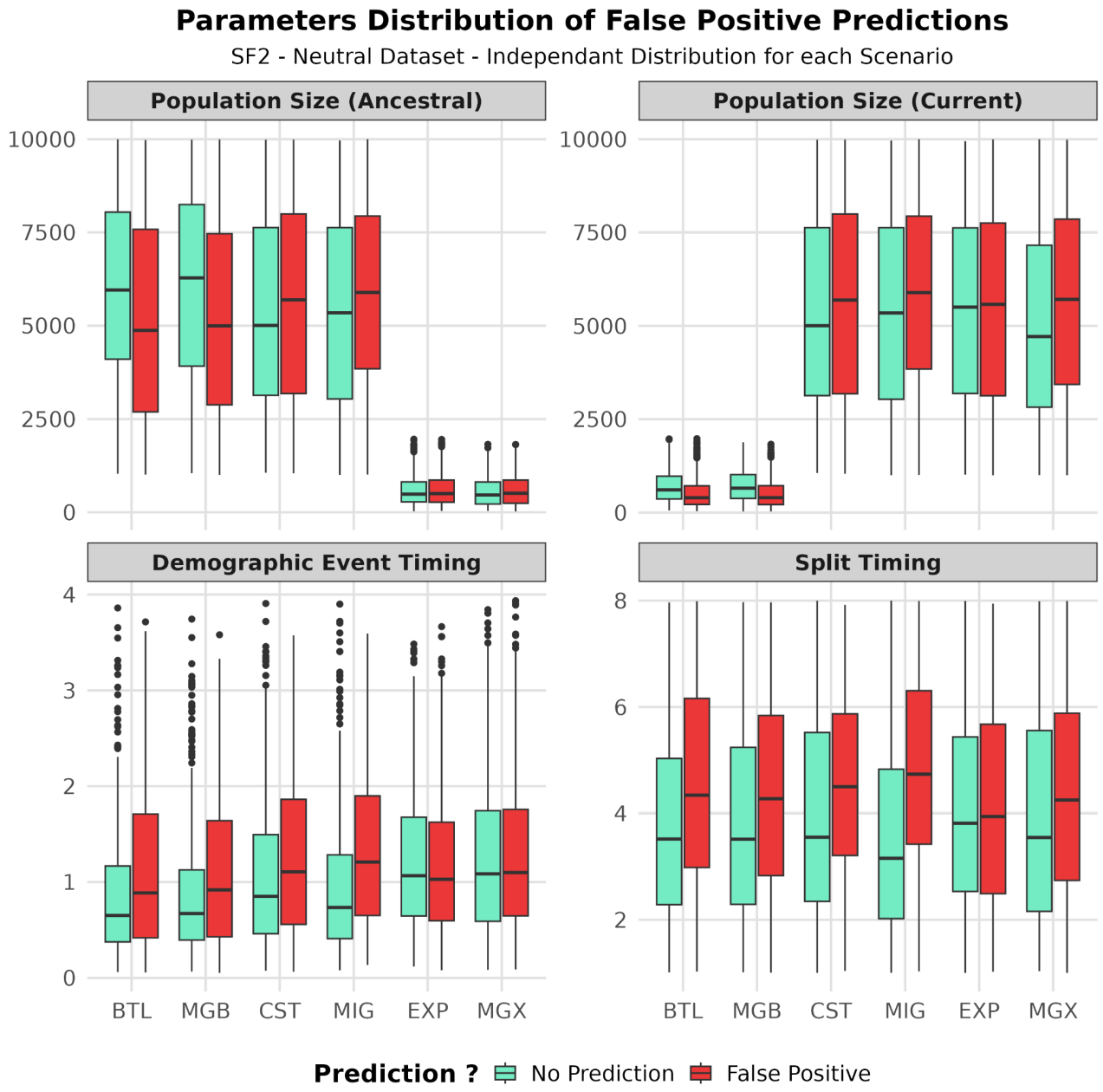



**Figure 43** - Distribution of Parameters for the **False Positive generated by SF2 on *neutral* test dataset**.

Population sizes are expressed in number of individuals. Demographic Event and Split Timings are expressed in regards to the current population size of the corresponding simulation.

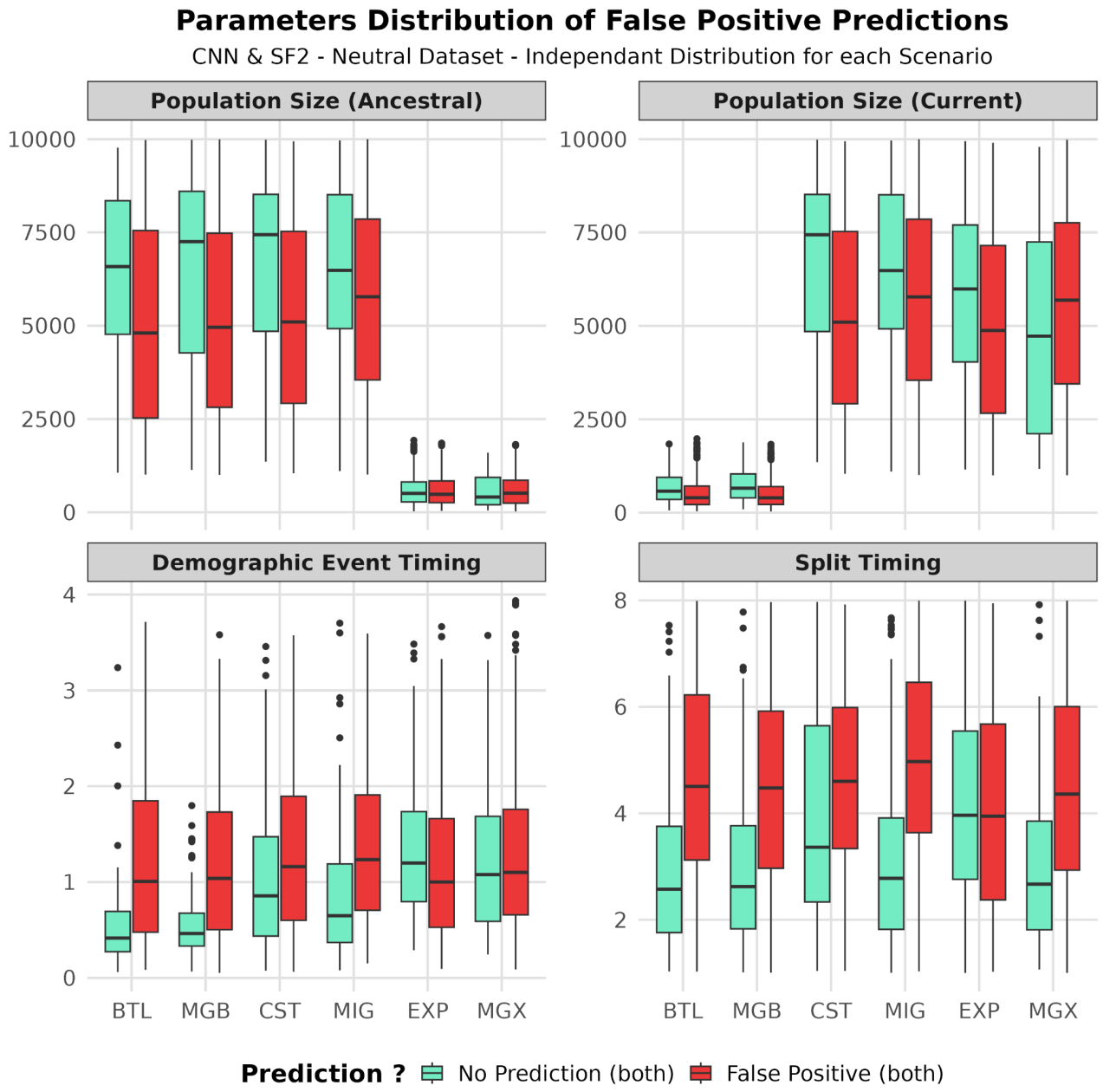



**Figure 44** - Distribution of Parameters for the **False Positive generated by both models (CNN and SF2) on *neutral* test dataset**.

Population sizes are expressed in number of individuals. Demographic Event and Split Timings are expressed in regards to the current population size of the corresponding simulation.

## Parameter Space of predictions on *sweep* data

Now, as for the detection of selective sweeps on *sweep* dataset we tested for significant differences in the distributions of the same demographic parameters (current and ancestral population sizes, ages of demographic events and ancestral population splits) as well as for selection parameters such as the age of the onset of selection, the timing at which the beneficial mutation reaches a frequency of 0.99 in the population as a proxy of the timing of fixation of the sweep, the selective coefficient, the width of the sweep bounding box and the duration of the sweep (see **Supplementary Table 10** for Wilcoxon tests p-values and results). However, it should be noted that due to the overall great performance of the CNN on this particular task, FPs parameters distributions had to be done using very few data points for BTL (11), MGB (3), CST (37), MIG (23) and MGX (66) scenarios. Thus, while such comparisons are still interesting to be carried, results must be observed with caution.

Focusing first on the CNN predictions (**Figure 45**) and specifically on the demography-related parameters, in CST, MIG, and EXP scenarios, sweeps were more likely to be missed in simulations with larger current or ancestral population sizes. In MIG and EXP, older population splits also contributed to missed sweeps, while in BTL and EXP, the timing of demographic events played a role, with younger events leading to missed sweeps in BTL and older events causing misses in EXP. Selection-related parameters also influenced sweep detection. On CST, MIG, EXP, and MGX scenarios, sweeps associated with older selection events as well as sweeps that were fixed for a longer time were missed more frequently. Additionally, in CST and MIG scenarios, smaller selective coefficients resulted in missed sweeps. Smaller true bounding box widths were particularly challenging for detection across all demographic scenarios. Similarly, shorter sweep durations were linked to missed detections in all but the CST and MIG scenarios.

As for SF2 predictions (**Figure 46**), on BTL and MGB scenarios, sweeps were more frequently missed in simulations with smaller current and ancestral population sizes. Older splits and demographic changes were a challenge across BTL, MGB, CST, and MIG, where sweeps associated were more likely to be missed. Regarding selection parameters, missed sweeps were linked to older selection events and older fixation times across all scenarios. The effect of the selection coefficient was less clear, with missed sweeps in BTL and MGB potentially linked to higher values in the former, and lower values in the latter. The width of the true bounding box influenced results differently across scenarios: larger sweeps were missed in BTL and MGB, while smaller sweeps were missed in MIG, CST, and MGX. Additionally, longer sweep durations and higher selective coefficients led to missed

detections in BTL and MGB simulations. As expected, the position of the selected allele showed no significant impact on sweep detection accuracy.

Finally, comparing sweeps detected by both CNN and SF2 against those missed by at least one model (**Figure 47**). Given that the CNN consistently detected significantly more sweeps than SF2 across all scenarios, it should be noted that most of the "missed sweeps" here resulted from sweeps missed by SF2. Missed sweeps occurred more frequently in smaller populations for BTL and MGB, whereas in EXP, sweeps were missed in larger populations. Similarly, smaller ancestral population sizes were associated with missed detections in BTL and MGB. Missed sweeps are also more common in simulations featuring older splits (BTL, MGB, CST, MIG, and MGX) and older demographic changes (BTL, MGB, CST, and MIG). Regarding selection, sweeps associated with older selection events and older fixation times are more likely to be missed across all scenarios. Sweeps with larger bounding boxes were missed in BTL and MGB, while smaller sweeps were more often missed in CST, MIG, EXP, and MGX. Sweep duration also affected detection: longer sweeps were missed in BTL and MGB, whereas shorter sweeps were more likely to be missed in EXP. Once again, missed sweeps in BTL and MGB are linked to higher selective coefficients.

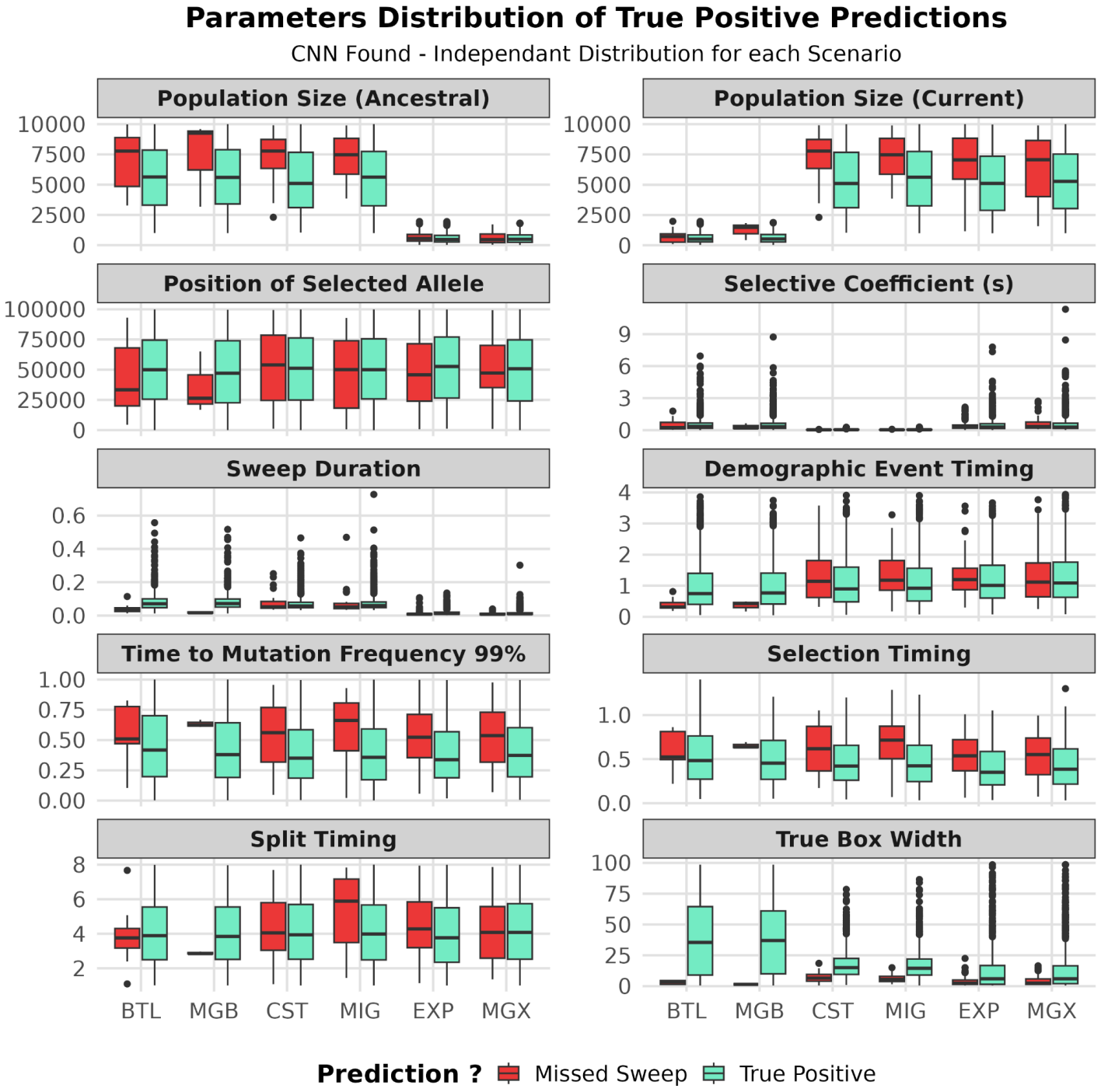



**Figure 45** - Distribution of Parameters for the True Positive predictions of the CNN on *sweep* test dataset.

Population sizes are expressed in number of individuals. Demographic Event and Split Timings, as well as Time to Mutation Frequency 99%, Selection Timing and Split Timing are expressed in regards to the current population size of the corresponding simulation. True Box Width is expressed in kbp.

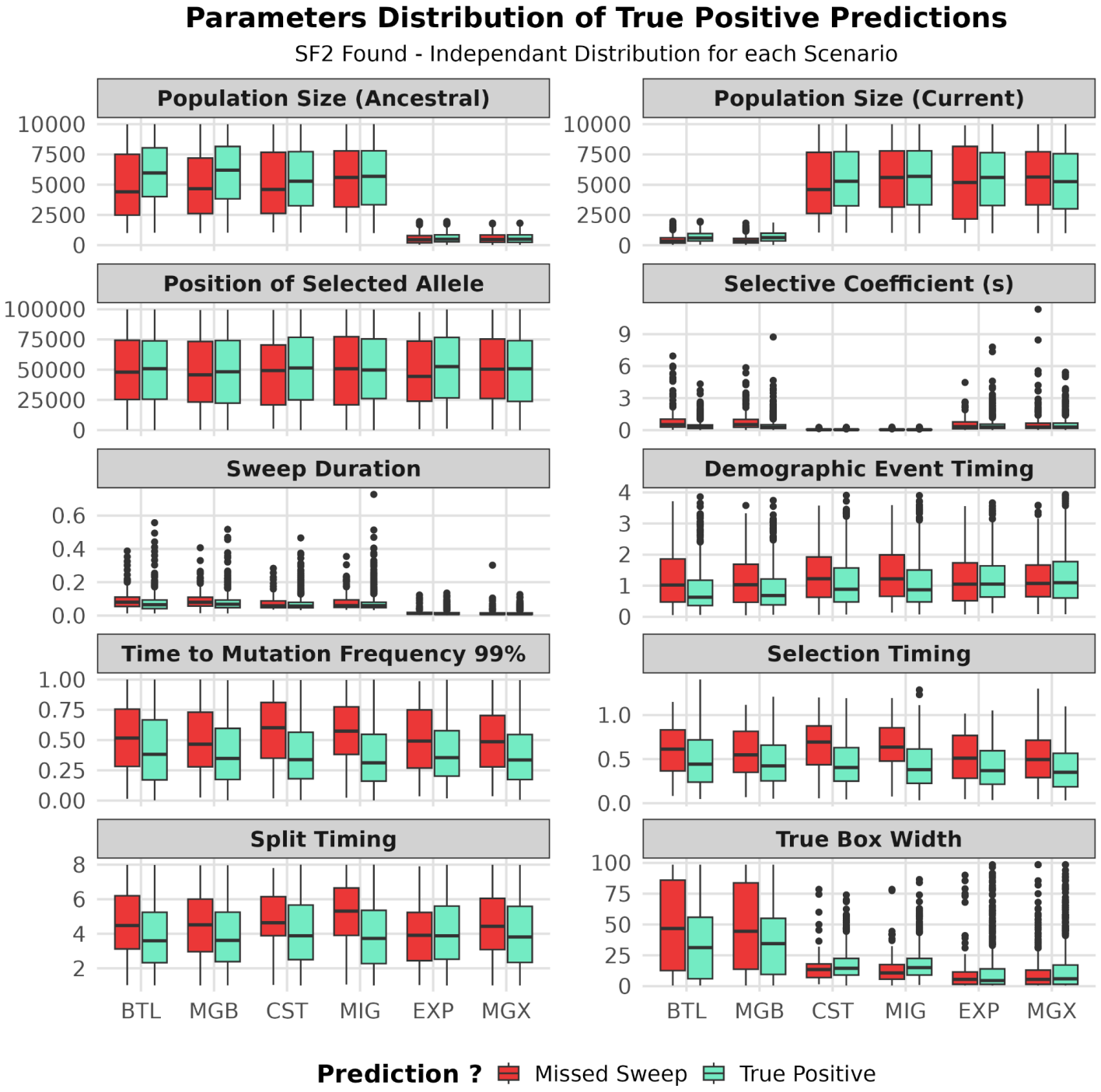



**Figure 46** - Distribution of Parameters for the True Positive predictions of SF2 on *sweep* test dataset.

Population sizes are expressed in number of individuals. Demographic Event and Split Timings, as well as Time to Mutation Frequency 99%, Selection Timing and Split Timing are expressed in regards to the current population size of the corresponding simulation. True Box Width is expressed in kbp.

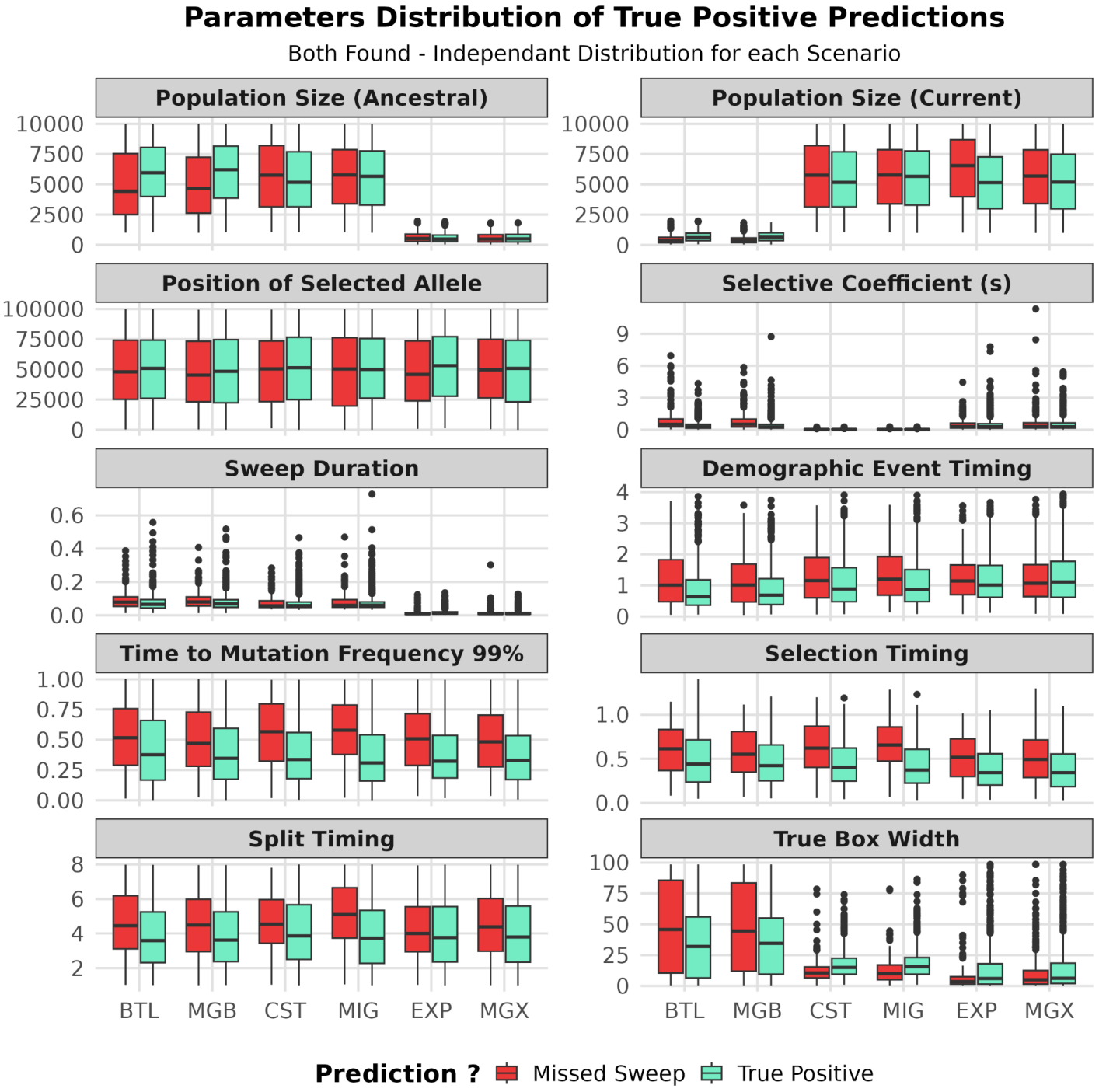



**Figure 47** - Distribution of Parameters for the True Positive predictions found by both models on *sweep* test dataset.

Missed Sweeps are sweeps missed by at least one model Population sizes are expressed in number of individuals. Demographic Event and Split Timings, as well as Time to Mutation Frequency 99%, Selection Timing and Split Timing are expressed in regards to the current population size of the corresponding simulation. True Box Width is expressed in kbp.

Exploration of the parameters revealed interesting trends. Two main parameters emerged as influencing the sweep detection results: the duration of the selective sweep (*i.e.* the number of generations between the appearance of the beneficial mutation and the sampling time) and the width of the bounding box associated with the sweep. Sweeps that were detected by both CNN and SF2, as well as those detected only by the CNN, showed a wide range of durations. In contrast, sweeps missed by both methods or detected solely by SF2 tended to be shorter in duration, with the width of the ground-truth bounding box being particularly small (**Figure 48**).

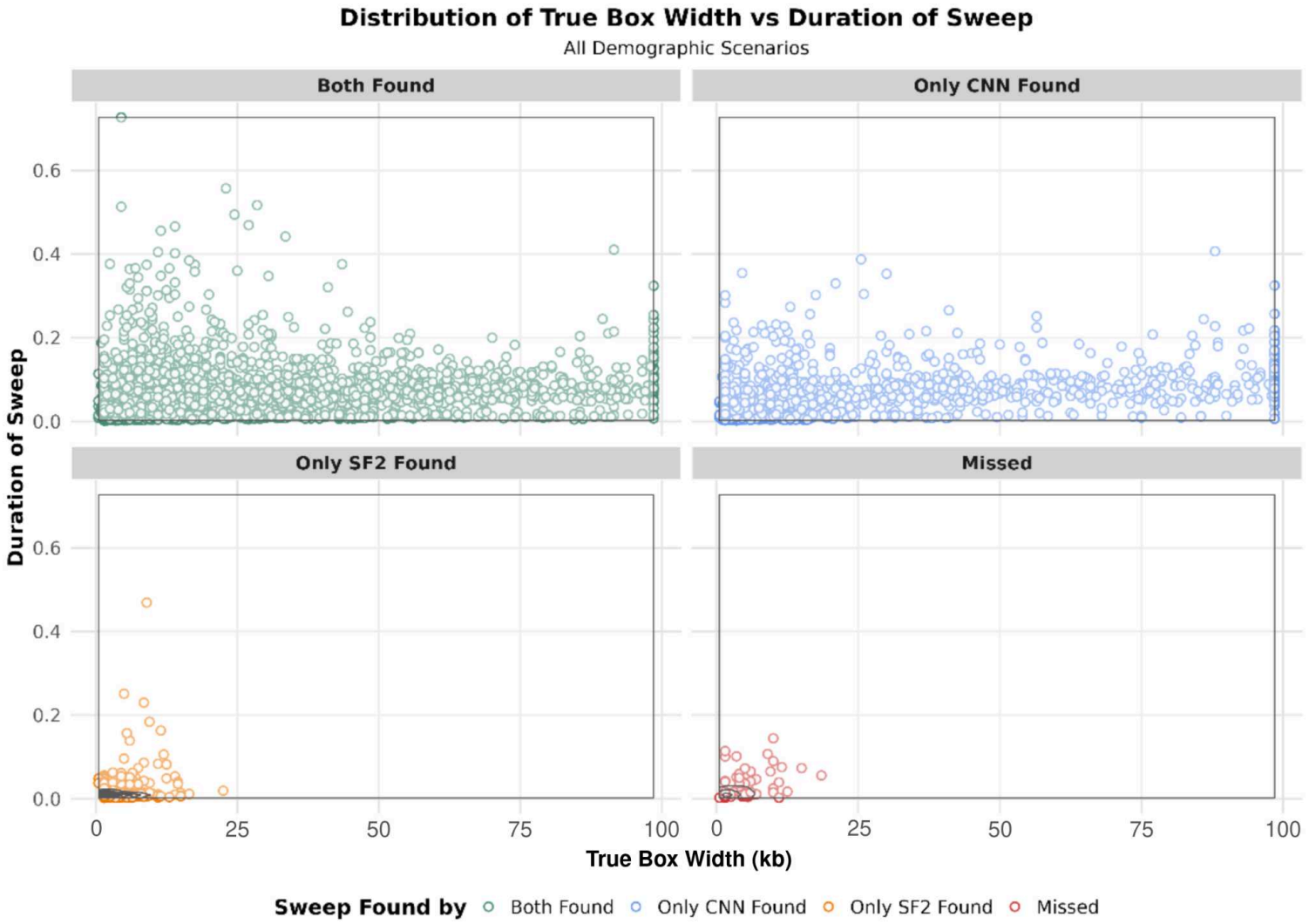



**Figure 48** - Distribution of found selective sweeps within the 2d density representation of the duration of the sweep compared to the size of the associated true bounding box.

Duration of sweep is (y-axis is expressed regarding the current effective size N of the associated population compared to the size of the associated true bounding box, *i.e.* the spawn of the area of effect of the selective sweep (x-axis). Sweeps found by both CNN and SF2 are on green, only found by CNN on blue, only found by SF2 are orange and simulations where the selective sweep have been missed are on red.

Thus, we focused on the analysis of those parameters. **Figure 49** shows the distribution of sweep duration compared between the 3 possible outcomes for a prediction (TP / FP / Not Found) across all six demographic scenarios for both CNN and SF2. Overall, no specific pattern appears, with the exception of the sweeps not found by the CNN, for the BTL and

MGB scenarios, which seemed to have a slightly lower duration. No such difference is present for the predictions done by SF2.

However, a clear demarcation can be seen with the distribution of the ground-truth bounding-box width (‘*true bbox width*’ – **Figure 50**) : the vast majority of FP or sweeps not found by the CNN have a true bbox width especially small ; graphically, the threshold is at about 25 pixels (**Figure 50**, dashed red line); the quantile 90 for the FP is at 28 and at 19 for the sweeps not found for an average of about 23.5 px *i.e.* about 1 175 bp. While not as good for SF2’s prediction, the same thresholds still works quite well here, with a majority of the FP and sweeps missed having a true bbox width smaller when considering a correct prediction as a prediction within the true bbox (which is to be expected, as a smaller bbox means that it is harder for the prediction to land within). The only two scenarios for which this threshold does not seem to hold, for both CNN and SF2 predictions, are the BTL and MGB scenarios.

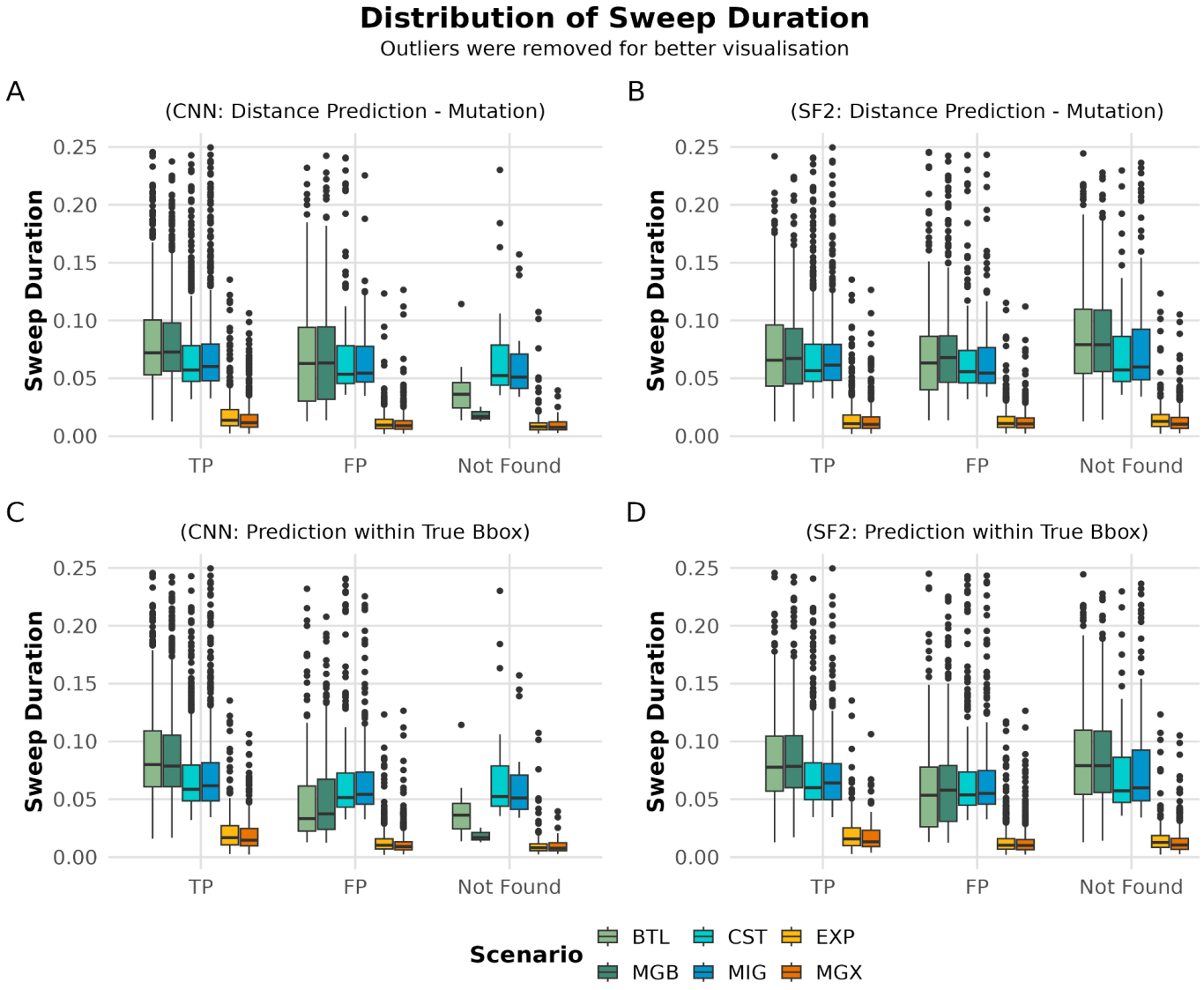



**Figure 49** - Distribution of duration of sweeps.

Sweep durations (y-axis) are expressed in regards to the current population size of the corresponding simulation, classified (x-axis) by True Positive (TP), False Positive (FP) or if the sweep was Not Found. **A)** and **C)** are the results for the CNN, **B)** and **D)** are the results for SF2. BTL/MGB scenarios are in green, CST/MIG scenarios are in blue and EXP/MIG scenarios are in orange. Outliers with sweep duration above quantile 90 were removed for better visualization of the distributions.

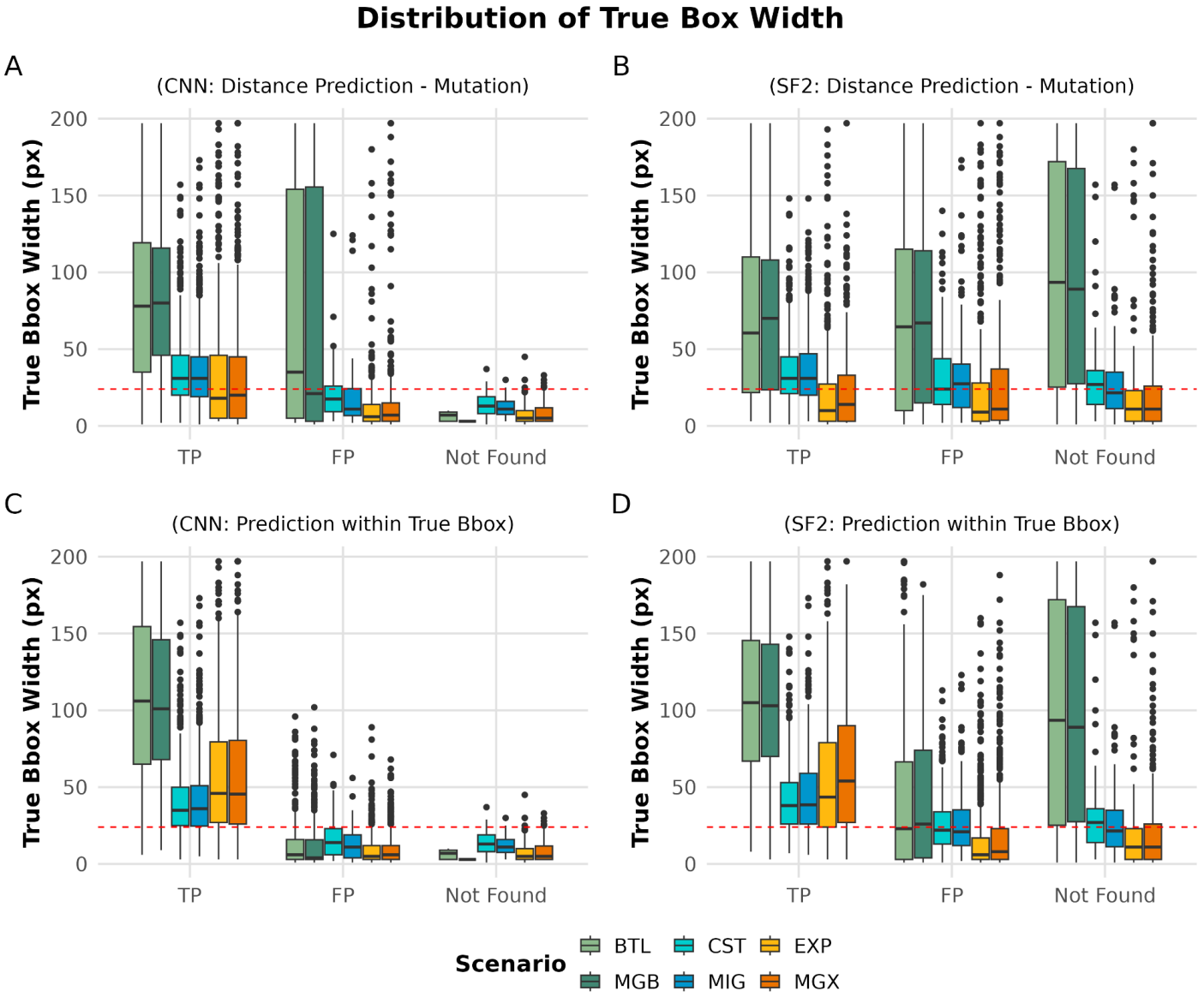



**Figure 50** - Distribution of true bounding-boxes widths among the demographic scenarios.

The width of the true bounding-boxes (y-axis) are expressed in pixels (1 px = 500 bp) and are classified (x-axis) by True Positive (TP), False Positive (FP) or if the sweep was Not Found. **A)** and **C)** are the results for the CNN, **B)** and **D)** are the results for SF2. The red dashed line indicates a graphically chosen threshold below which most FP and sweep not found by the CNN seem to be. BTL/MGB scenarios are in green, CST/MIG scenarios are in blue and EXP/MIG scenarios are in orange.

**Figure 51** shows the distribution of the true bbox widths for each prediction class among the six demographic scenarios for the CNN and **Figure 52** shows the same for SF2 predictions. In each case, the red line corresponds to the size below which there is 90 % of the FP and sweeps not found, without considering the BTL and MGB scenarios in the computation. Once again, these two scenarios are put aside as they appear to have a bimodal distribution

with lots of very small and very wide true bboxes. For the CNN's predictions, most FP and sweeps not found have an unusually small true bbox width compared to the TP ones. This still partially holds true on BTL and MGB scenarios, but there is also a number of very large bbox associated with the FP on those scenarios. The same pattern can't really be observed for SF2 predictions. Indeed, the distributions true bbox width for the TP, FP and sweeps not found all match quite closely on all scenarios, with again the exception of BTL and MGB where lots of missed sweeps here have a very large true bbox.

Smaller sweeps are harder to detect for the CNN; they may not produce the same strong selective signal, making them harder to detect reliably, but sweeps spawning the whole length of the chromosome are also harder to detect due to the absence of certain characteristics expected patterns, for example if the "valley of diversity" is so wide that it encompasses the whole chromosome. Overall, sweeps detected by both methods covered a broad range of widths, while those missed or only found by SF2 tended to be smaller in scope. This indicates that larger, more pervasive sweeps were more consistently identified by both approaches, whereas smaller or more localized sweeps posed a greater detection challenge.

## Predictions Distance to the Beneficial Mutation

As a final step to evaluate our models, we wanted to compare the localization accuracy of the predictions. To do so, we analyzed the distance between the actual position of the beneficial mutation and the predicted sweep positions for both methods. In general, the results showed that SF2 predictions were further from the true beneficial mutation compared to the CNN, but the gap varied depending on the demographic scenario and the method used for calculating the distance. Across all demographic scenarios, the average distance of SF2 predictions to the actual beneficial mutation was approximately 24 433 ± 22 119 bp. In contrast, CNN predictions, when considering the distance from the center of the predicted bounding box to the mutation's true position, averaged 15 820 ± 18 154 bp. This difference was even more pronounced when computing the distance as the difference between the center of the predicted and ground-truth bounding boxes, with CNN predictions averaging 11 958 ± 18 721 bp (**Table 10**). These results suggest that CNN's localization performance is generally superior to SF2's, with a significant reduction in the average distance to the target mutation.

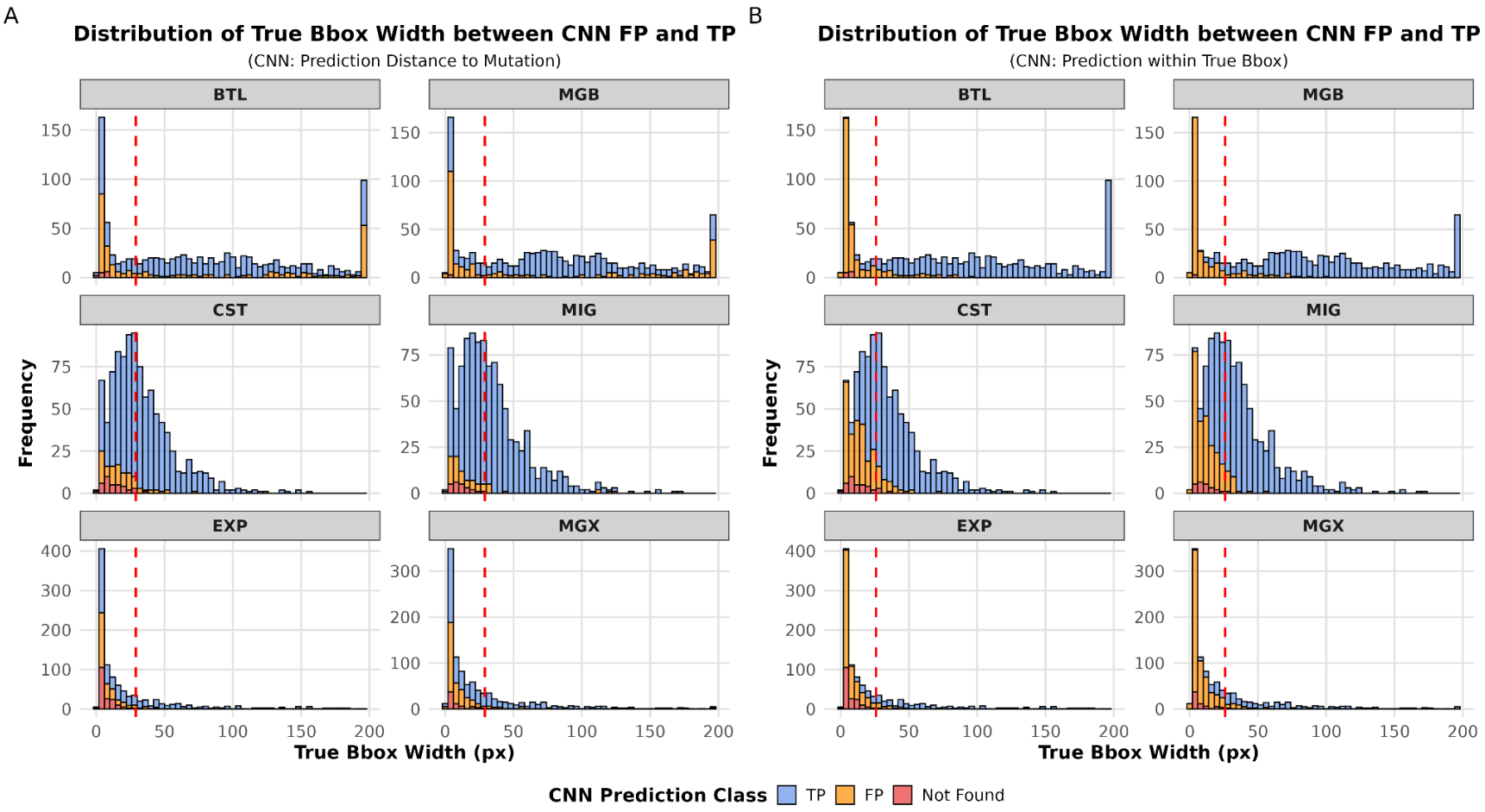



**Figure 51** - Distribution of the widths of true bbox among the six demographic scenarios for the CNN, classified according to the CNN predictions.The dashed-red line corresponds to the size below which there is 90 % of the FP and sweeps *not found* (without considering the BTL and MGB scenarios in the computation). **A**) Predicted positions are inside the true bbox and **B)** within close to the mutation.

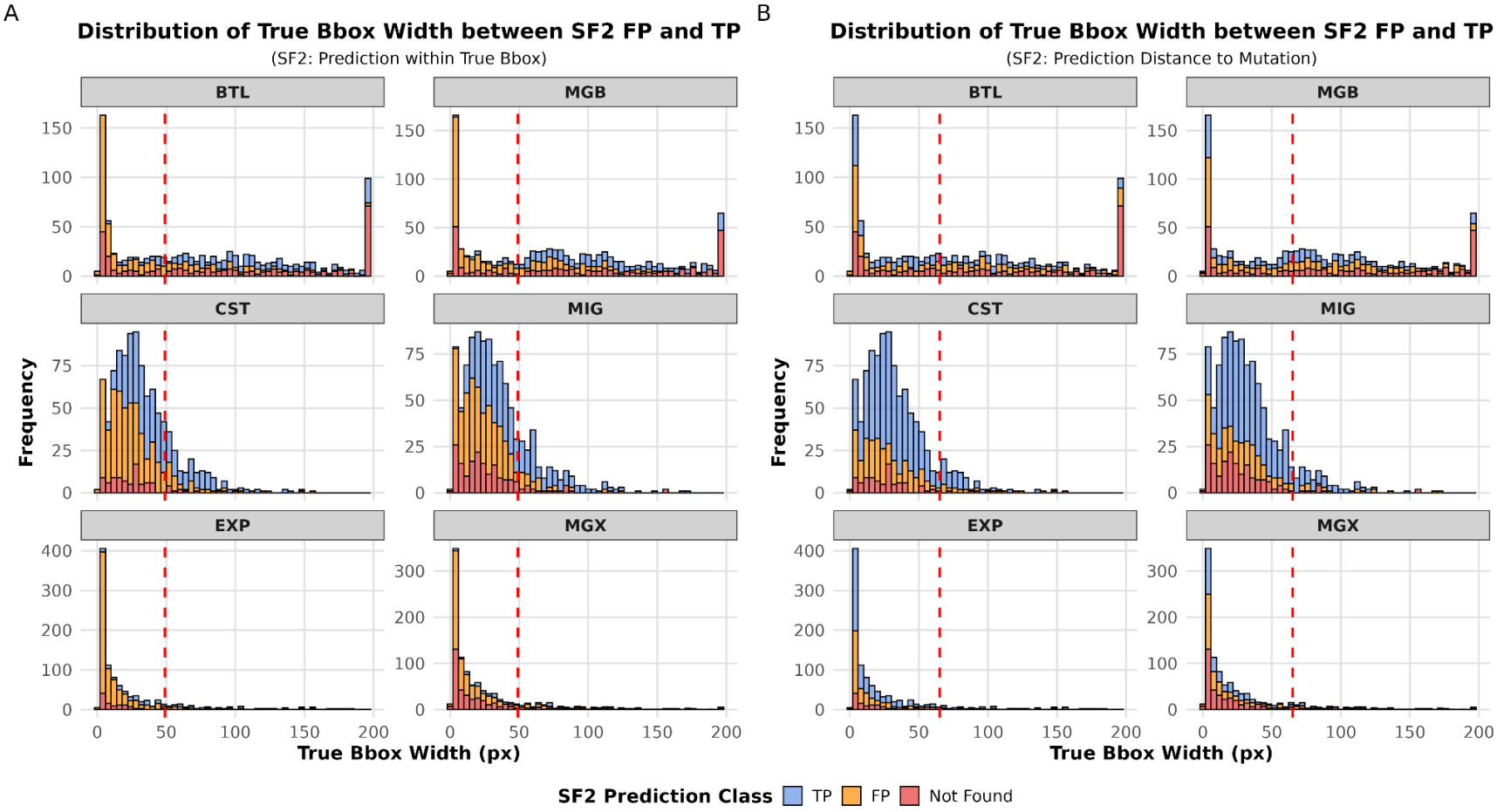



**Figure 52** - Distribution of the widths of true bbox among the six demographic scenarios for the CNN, classified according to the CNN predictions. The dashed-red line corresponds to the size below which there is 90 % of the FP and sweeps *not found* (without considering the BTL and MGB scenarios in the computation). **A**) Predictions inside the true bbox and **B)** predictions close to the mutation.

When breaking down these results by specific demographic scenarios (**Table 10 & Figure 53**), we found the following patterns. On CST and MIG simulations, both CNN and SF2 predictions were relatively close to the actual beneficial mutation. For the CNN, the distance averaged around 8 000 to 6 500 bp depending on the method of measurement, while SF2 predictions were consistently farther, with an average distance of 17 700 bp. On BTL and MGB scenarios, the CNN predictions were again closer (approximately 11 000 bp up to 19 000 bp when considering the actual position of the mutation). In contrast, SF2 predictions were about 30 000 bp away, regardless of the method used for distance calculation. On EXP and MGX simulations, the CNN's distance averaged 20 000 bp, while SF2 predictions were at a distance of 24 000 bp for EXP simulations and approximately 30 000 bp for MGX simulations. Across all scenarios, the standard deviation of prediction distances remained relatively high, indicating substantial variability in prediction accuracy within each method. This variability was often comparable to or even exceeded the average distance, underscoring the inherent challenges in accurately localizing selective sweeps in complex demographic settings.

**Table 10** - Comparison of mean and standard deviation of the distance between the predicted position of the selective sweep and the position of the actual beneficial mutation. Distances are expressed in bp.

| Demographic Scenario | Distance Center pred. bbox - mutation (CNN) | | Distance Center pred. bbox - Center true bbox (CNN) | | Distance Predicted position - mutation (SF2) | |
|---|---|---|---|---|---|---|
| | mean | sd | mean | sd | mean | sd |
| BTL | 19245 | 17929 | 11286 | 18792 | 30185 | 22132 |
| MGB | 19076 | 18555 | 11325 | 19689 | 30872 | 23676 |
| CST | 8600 | 13842 | 6696 | 14278 | 17880 | 19809 |
| MIG | 7309 | 11424 | 5371 | 11691 | 17615 | 19933 |
| EXP | 20476 | 20056 | 19024 | 20776 | 24133 | 20530 |
| MGX | 20962 | 20289 | 19432 | 20935 | 30089 | 23111 |

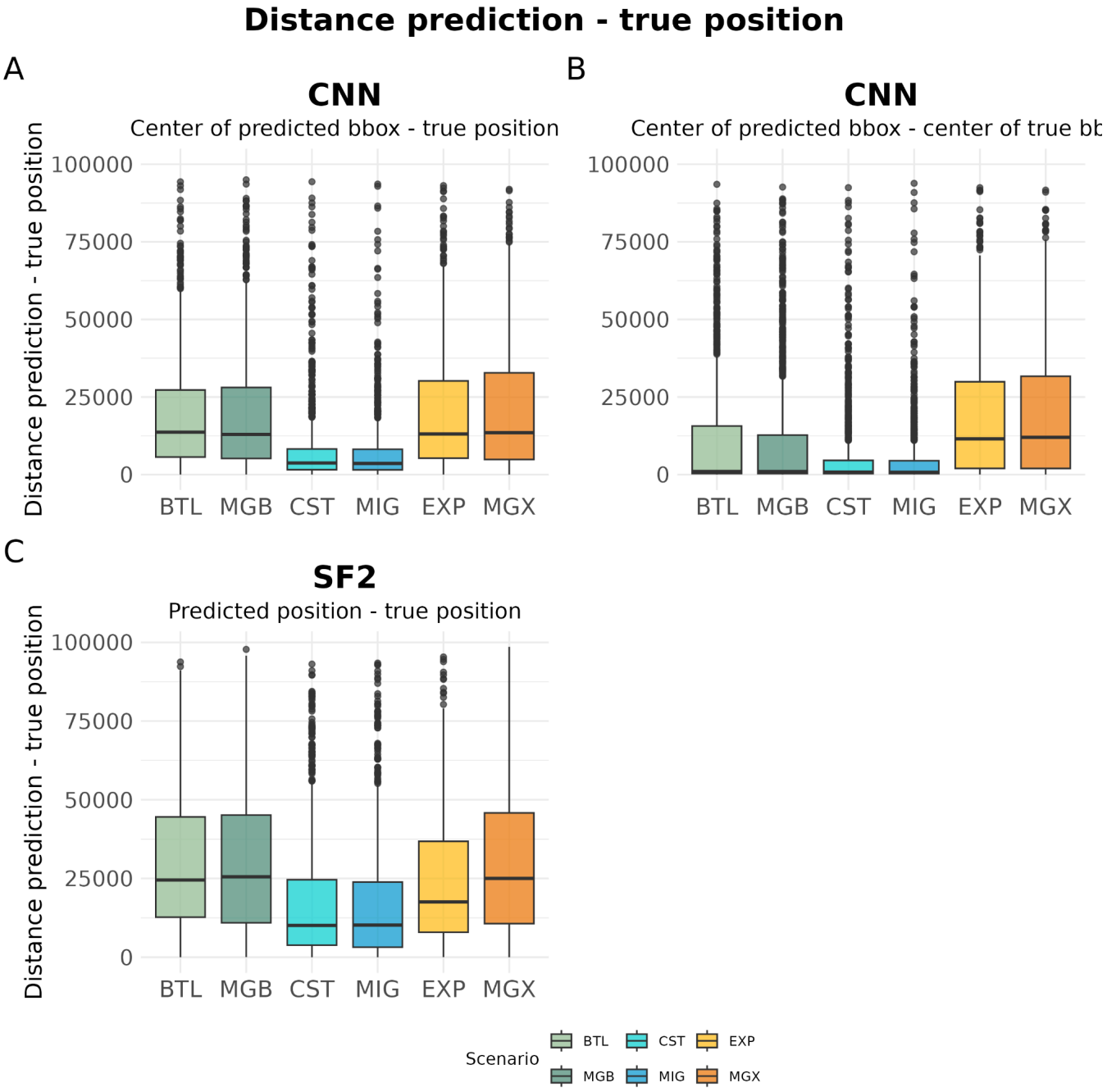



**Figure 53** - Distribution of the distance between the predictions and the position of the actual beneficial mutation.

Distances are computed in two different ways for the CNN predictions : either by comparing the position of the center of the predicted bounding box to the position of the beneficial mutation, or by comparing the position of the center of both the predicted and the ground truth bounding box. For SF2, distances are simply computed as the distance between the predicted and the true position of the mutation.

# Discussion

Thanks to the recent advances of DNA sequencing technologies leading to a drastic increase in the amount of genetic data available, lots of new methods to study and detect positive selection have been developed. Lots of studies leverage the power of summary statistics, and some have also started to apply machine learning to this type of problem (Schrider et al., 2015, Schrider & Kern, 2016, Pavlidis et al., 2010, Lin et al., 2011). Our work here follows this same trend, and focuses on the development of a machine learning approach using a resNet-FasterRCNN to detect but also to localize quite precisely targets of natural selection from genetic data. More precisely, our CNN is able to detect and localize the position and the approximate range of effect of a selective sweep by predicting a bounding box associated with the sweep. The predictions are done after a fine-tuning, in other words an additional training, of a pre-trained CNN is realized for it to learn how to realize such a task. In this study, we chose to compare a set of 32 different trainings to test and find good practices when implementing such methods before testing our newly trained CNN against the SweepFinder2 tool. Thanks to these comparisons, we tried to gather information about the important parameters to consider for the training dataset used by the CNN.

## CNNs performed best when undergoing Scenario Specific Training

CNNs trained on one particular demographic scenario performed best on *test* data from the same scenario, highlighting the importance of scenario-specific training. However, CNNs trained on more complex demographic histories, such as BTL and EXP, also showed a great ability to generalize to CST scenarios, where sweep signals are less confounded by demographic effects. More precisely, CNNs trained on EXP data also performed well on CST simulations, suggesting that they learned to recognize some general patterns of selective sweeps that can be transferred between scenarios. On the same simulations, CNNs trained on BTL data displayed the highest variance : their performance, while still good overall, was less consistent. On BTL simulations, the strong performance associated with a low variance of BTL trained CNNs data suggests that they are highly specialized in detecting selective sweeps in data shaped by demographic contractions, where genetic signals may be more distinct due to reduced diversity. Finally, all CNNs struggled with EXP data, likely due to the confounding signals generated by the interaction of selective sweeps with the demographic expansion, which can resemble the patterns seen in populations recovering from a

bottleneck. This poor performance on EXP simulations highlights the challenge of detecting sweeps in expanding populations, where coalescent signals become more diffuse.

EXP and CST trained CNNs showed the same performance when tested on all demographic scenarios at once, with an average AP at around 0,420 and standard deviation of about 0,07. BTL trained CNNs on the other hand, displayed a slightly lower overall average AP, but with such a variance that they also achieved both the lowest and the highest overall AP. A possible explanation comes from the demographic scenarios used for the training. Indeed, as shown before, CST trained CNNs seemed to be the ones able to generalize the best, with all models retaining good APs compared to the ones of the other CNNs. The information EXP trained CNNs have been able to learn from the EXP simulations allow them to reach AP equivalent to the CST trained ones on both CST and BTL simulations while outperforming them on EXP. This specialization comes at the cost of some stability, as they displayed an increased variance on both BTL and CST simulations compared to CST. BTL trained CNNs displayed this even more, with the highest APs and lowest variance on BTL simulations, but with very variable results on both CST and EXP simulations. Interestingly, despite this overspecialization of most BTL trained CNNs, the best CNNs when comparing average overall AP still happened to be a BTL trained one. Indeed, it appears that, using the correct training parameters, such a specialized model is able to outperform the more generalized CST trained CNNs.

## Using specifically formatted data inputs such as *objDet* increase the CNNs performances

It comes with no surprise that this best BTL trained CNN was trained on *objDet* data as input. Indeed, the type of input data plays a crucial role in CNN performance. CNNs trained on *objDet* data significantly outperformed those trained on plain summary statistics (sumStats) or raw data (sorted-rawData), achieving an average overall AP of 0.500 compared to 0.418 and 0.277, respectively. This shows again the importance of selecting the right representation of genomic data, with *objDet* data likely capturing more relevant information for sweep detection. These inputs are, at their core, matrices of numerous summary statistics which were already proven to be an efficient way of analyzing genetic data for detecting selection patterns with tools such as S/HIC (Schrider and Kern, 2016). The normalization of each statistics using the maximum and minimum values obtained on the simulation itself also allows to maximize the intra-genome variance highlighting even more the selective sweep signal. The comparatively lower performance of CNNs trained on

*sorted-rawData* data suggests that this input type is most likely not as effective at highlighting the patterns necessary for accurate sweep detection using the resNet-FasterRCNN architecture we've used in this study.

To wrap things up regarding the training parameters, neither the number of retrained backbone layers nor the use of data augmentation significantly affected the overall performance of the CNNs. Models with four retrained backbone layers performed similarly to those with five, and data augmentation did not yield noticeable improvements. This is interesting because it suggests that we can reduce computational times without sacrificing accuracy. By not requiring data augmentation and using only four retrained layers, we can decrease the computational resources needed, leading to faster training times and lower energy consumption.

## CNNs detects more sweeps but also tend to generate more false positive

As stated before, we choose to compare the results of the best CNN we obtained against the popular sweep detection tool, SweepFinder2. This comparison reveals several important insights regarding their relative performance across the different demographic scenarios. The CNN consistently outperforms SF2 when it comes to detecting selective sweeps and accurately localizing the beneficial mutation, while SF2 shows more conservative predictions with fewer false positives on a *neutral* dataset. Indeed, when comparing the false positive rate of both methods on a test datasets of 6 000 simulations of population evolving under selective neutrality, the CNN generates substantially more FPs than SF2, up to 3 to 5 times more, if no filter is applied to the CNN's predictions. However, a threshold value even as low as 0,3 applied on the prediction scores  is enough to even the number of unique FPs generated by the two methods. Hence, even though SF2 appears more conservative at first glance, it is also important to remember that both methods are vastly different in their approach, and that it might sometimes be hard to find a comparison protocol efficient to evaluate one method against another. In our case, it is easy for us to set multiple different thresholds on the prediction scores of the CNN, but no such score exists for SF2 as setting thresholds on the CLR values seemed too far off from what we wanted.

In detecting selective sweeps, the CNN consistently outperforms SF2 across most scenarios, particularly on BTL, MGB, CST and MIG simulations. The CNN identifies significantly more sweeps than SF2 on BTL (329 vs. 6), MGB (318 vs. 2), and MIG (171 vs.

16) simulations. If we use the same threshold of 0.3, the CNN maintains a clear advantage in BTL and MGB scenarios, detecting 297 unique sweeps on BTL and 288 on MGB, compared to 78 and 65 found exclusively by SF2, respectively and the results of both methods are comparable on MIG and MGX scenarios. However, on EXP and CST simulations, SF2 then finds more unique sweeps. This suggests that SF2 may be better suited to detecting sweeps in certain rapidly growing or expanding populations where our implementation of CNN struggles to maintain the same level of performance. Nonetheless, both methods remain good even in these scenarios, highlighting their overall versatility.

## Older and more eroded selection signals are harder to detect for both methods

Out of all the parameters significantly impacting the models ability to detect a selective sweep, three of them are consistent across all scenarios : the age of the selection event, the age of the beneficial mutation fixation and the width of the bounding box *i.e.* the size of the selective sweep. Indeed, if the beneficial mutation, as well as its fixation, are older, then other evolutionary forces such as the recombination will have time to erode the selective sweep signature. Thus, with enough time, the observable effects of the selective sweep on the genetic diversity will decay, which will in turn reduce the observable width of the selective sweep. All in all, these three variables will erode the sweeps signal, making them harder to detect by our models. It should also be noted that, while no significant difference was found for the CST and MIG scenarios, the sweep duration also impacted the models performances, especially for the BTL and MGB scenarios, where shorter sweeps were harder to find for the CNN, but easier to find by SF2. As the time spent between the apparition of the beneficial mutation and its fixation is shorter, shorter sweeps usually display a low haplotype diversity surrounding the beneficial mutation. However, it’s also important to remember that very few sweeps were missed by the CNN so, while the differences were deemed statistically significant, they were computed using no more than 11 BTL and 3 MGB simulation parameters.

## CNNs localize the selective sweeps position more accurately

The CNN's ability to localize the beneficial mutation is one of its major advantages over SF2. When defining a sweep as “found” if the predicted position is within a range equal to the average sweep size of the true mutation the CNN performs substantially better across most

scenarios showing that the CNN has a greater capacity for identifying sweeps with greater positional accuracy. A more stringent criterion, which considers a sweep “found” only if the predicted position falls within the actual bounding box of the sweep, further highlights the CNN's strengths in CST and MIG scenarios, where it identifies more unique sweeps than SF2. However, this method also reveals that both models struggle with certain scenarios, particularly EXP and MGX, where they miss a considerable number of sweeps (498 and 580, respectively). These findings highlight the importance of scenario-specific performance differences and suggest that while the CNN excels at localizing mutations, neither method is infallible across all demographic conditions.

# Conclusion

In this study, we showed that while CNNs demonstrate superior accuracy in most simulated scenarios, particularly in terms of detection and accurate localization of the sweeps, it struggles with high FP rates on *neutral* simulations if no threshold is put on the predictions. On the other hand, SF2, while more conservative and less prone to generating false positives, also misses a considerable amount of sweeps found by the CNN. Overall, the CNN approach demonstrated great performance in detecting sweeps across most scenarios, often identifying more sweeps than SF2. It also showed remarkable accuracy in localizing the beneficial mutations, outperforming SF2 in this regard. However, despite its overall strong performance, the CNN struggled in expansion (EXP) and growth (MGX) simulations, where SF2 showed relative strengths, detecting more unique sweeps in these scenarios. This suggests that, while CNN-based methods hold great potential for improving the identification of selective sweeps in populations with complex demographic histories, scenario-specific strategies may be necessary to optimize sweep detection performance, as neither method performs uniformly across all population histories.

As a conclusion to this second part, we would like to also highlight the following key points:

1) The *pre-trained* architecture tested in this study (FasterRCNN model using a ResNet-50-FPN backbone) performed best when trained on data following the same demographic scenario as the test data. However, models trained on more complex histories (BTL or EXP) show better generalizability. Hence, a more diverse training dataset (containing all possible expected demographic scenarios) could constitute a huge advantage.

2) Among the tested input formats, *objDet* data substantially improves CNN performance, achieving higher average AP scores than sumStats or raw data. This structured input highlights selective sweep signals more effectively. This highlight one of the major results of Chapter 1: the importance of data representation for the use of deep-learning tools.

3) Other tested training parameters, such as the number of retrained backbone layers or the use of data augmentation did not seem to affect the performance of the CNNs.

4) Compared to the traditional SweepFinder2 (SF2) method, CNNs demonstrate greater sensitivity in sweep detection but tend to produce more false positives in neutral datasets. Specifically, CNNs excel in accurately localizing selective sweep positions compared to SF2.

5) Both CNNs and SF2 face difficulties in detecting older or eroded sweeps. Key factors impacting detection accuracy include the sweep's age, fixation time, and the size of the selective sweep (bounding box width), with older events being more challenging due to the erosion of the selective signature.

In conclusion, while the CNN outperforms SF2 in most cases, particularly in terms of sweep detection and localization accuracy, SF2 remains a valuable tool in its ability to minimize false positives in *neutral* scenarios and perform well in certain demographic contexts like population expansion. This comparison highlights the importance of choosing the right tool depending on the specific demographic history being studied and suggests that hybrid approaches, combining the strengths of both models, could further enhance the accuracy and reliability of selective sweep detection in population genetics research.

Furthermore, these results emphasize, as were the findings of Chapter 1, the importance of the choice of data representation to achieve optimal performance. A well-constructed dataset - representing a broad spectrum of the expected variability of real data, and formatted with thoughtful input structure - substantially enhances model accuracy. We also would like to take a moment to highlight the limited influence observed when increasing the number of retrained layers or using data augmentation. This suggests that CNNs may achieve high performance without excessive computational overhead. While thoroughly testing parameter combinations is essential for developing broadly applicable deep-learning tools, we believe a balanced approach can yield efficient, high-performing models that also reduce computational demand and environmental impact. This is not a critique of extensive

exploration, but rather a call to reconsider the “high complexity” often marketed as a selling point in AI tools. Quite reassuringly, recent studies has focused on developing more efficient deep learning methods to address the challenges of increasing parameter counts and aim to create more parameter-efficient models without sacrificing performance (see Menghani (2024) for a comprehensive survey of techniques aimed to make deep-learning models more efficient).

# Discussion

Our understanding of evolutionary processes and, more specifically, of genetic diversity has greatly evolved throughout the centuries. From the early Greek philosophers' view of variation as "imperfection of an ideal form," the study of genetic diversity evolved through foundational contributions, notably Darwin's *On the Origin of Species* (1859), which introduced the concept of natural selection, and Mendel's principles of heredity (1865). These pivotal works laid the groundwork for a better understanding of genetic diversity, later expanded by evolutionary theoriticians like Fisher, Haldane, and Wright, who further defined the genetic mechanisms underlying variation within and between species. Thanks to huge advancements in sequencing technology during the mid 1960's and the revolution sparked by Motoo Kimura and its neutral theory of molecular evolution (Kimura, 1968) the field evolved into what is nowadays perceived as the study of genetic diversity. The study of the evolutionary history of species through genetic diversity has historically used neutrality tests, the likes of Tajima's D (Tajima, 1989) and researchers continue to push further our understanding of this theoretical framework with, for example, the work of Guillaume Achaz in unifying all neutrality tests under a general model (Achaz, 2009). However, the confounding effects of demographic events and selection processes have always been an issue, as illustrated by numerous works throughout the recent years (Smith & Haigh, 1974 ; Tajima, 1989 ; Fu & Li, 1993 ; Simonsen et al., 1995 ; Fay & Wu, 2000 ; Galtier et al., 2000 ; Kim & Stephan, 2002 ; Jensen et al., 2005 ; Hahn, 2008). While various approaches have been proposed to distinguish the effects of selection from those of demography, such as the use of genome-wide data (Nielsen et al., 2005 ; Li & Stephan, 2006), most if not all of these works have been using what Brieman characterize as 'data modeling culture' (Brieman, 2001). The recent years have seen the rise of methods based on the 'algorithmic modeling culture' : machine learning. Armed with this new approach, numerous studies have started to apply machine-learning models to population genetics (Caldas et al., 2022; Flagel et al., 2019; Fraïsse et al., 2020; Gower et al., 2021; Hamid et al., 2023; Kern & Schrider, 2018; Kittlein et al., 2022; Lopez et al., 2018; Nait Saada et al., 2023; Sanchez, 2022; Schrider & Kern, 2016; Smith et al., 2023; Yelmen et al., 2021). However, many population geneticists are unfamiliar with deep-learning tools, making the deployment, use and interpretation sometimes challenging. Recent efforts to provide detailed workflows (Whitehouse & Schrider, 2022) and the multiplication of pre-trained architectures catered to tackle specific population genetics analyzes (Hamid et al., 2022; Sanchez et al., 2023) have largely helped

to ensure reproducibility and to reduce the need for retraining complex models as well as reducing the carbon footprint of machine learning application (Grealey et al., 2022).

In this thesis, we have proposed a focus on the deployment of deep-learning methods to tackle two aspects of the issue of the intertwined effects of demography and selection : the classification of a given genomic sample according to plausible demographic scenarios in Chapter 1, and the detection and localization of signals of selective sweeps in Chapter 2. Both parts focus on each particular problem through the lens of Convolutional Neural Networks, a specific type of deep-learning networks that mimics the way the human brain processes visual information, making them particularly effective for analyzing complex patterns in pictures.

Chapter 1 examines the performance of convolutional neural networks (CNNs) and Approximate Bayesian Computation Random Forest (ABC-RF) in classifying demographic histories from genomic data. We specifically try to propose an insight into the various challenges encountered in developing a deep-learning-based classification method for demographic history using genomic data, with a particular focus on the choices made at various steps. By evaluating several CNN architectures (three different network architectures – two custom designs (simple and mix) and two pre-trained models, efficientNet (Tan & Le, 2020) and resNet (He et al., 2015) ) and different input data representations (matrices of raw genomic alignments, sorted or not, or matrices of summary statistics), we aimed to find optimal (or at least better) configurations and assess challenges, such as model misspecification and gene flow impacts, that might affect classification accuracy. A key finding is the importance of data representation: CNNs trained on sorted-rawData significantly outperformed those trained on raw or summary statistics data, indicating that a seemingly small change in data pre-processing can substantially improve model performance. Additionally, networks trained on data from sweep scenarios showed greater robustness than those trained on neutral data, highlighting the importance of including data containing enough variability in training sets to maximize the networks generalization abilities. Most notably, Chapter 1 of this work emphasizes this point even more as even a simple CNN architecture can outperform more complex classifiers when trained on the appropriate data and using correct data representation. A simple CNN trained on sweep sorted raw data performed exceptionally well, rivaling and sometimes exceeding the performance of even pre-trained architectures. This suggests that success in demographic classification depends less on model complexity and more on careful decisions regarding data preparation and training dataset composition.

Chapter 2 proposes to compare the effectiveness of CNNs and the more traditional SweepFinder2 (SF2) method in detecting selective sweeps within genomic data across various demographic scenarios. Overall, CNNs demonstrated superior performance in identifying sweeps, often surpassing SF2 in terms of accuracy and localization. However, the CNN model exhibited higher false-positive rates in neutral simulations, whereas SF2 proved to be more conservative, minimizing false positives but also missing sweeps detected by the CNN. Notably, CNNs excelled in accurately pinpointing the location of beneficial mutations, a task where SF2 was less precise. While this comparison is in itself quite interesting, we would like to focus more on the comparison of the different possibilities explored for the CNN architecture. The architecture chosen for this study, a pre-trained FasterRCNN (Ren & al., 2016) model with a ResNet-50-FPN backbone, performed best when trained on data closely matching the test data's demographic context. Nevertheless, models trained on more complex demographic histories, such as bottleneck (BTL) or expansion (EXP), showed greater flexibility and generalizability, indicating that a more diverse training set encompassing varied scenarios might greatly enhance CNN robustness and performances. Interestingly, some training parameters, such as the number of retrained backbone layers and the use of data augmentation techniques, showed little influence on CNN performance. While an extensive parameter search remains crucial for the development of generalizable deep-learning applications, this suggests that a more modest approach can still yield high-performing models while also reducing computational demand and environmental impact. Given the recent attention on parameter efficiency in deep learning, these insights align with current efforts to refine models, prioritizing effectiveness over unnecessary complexity (Menghani, 2024). Additionally, input data formatting emerged once again as a critical factor, with the *objDet* data format (a way of representing our genomic data thought specifically to try and enhance the visualization of selective sweep signals) yielding significantly higher accuracy scores than basic summary statistics (sumStats) or raw data matrices. This underscores again the importance of well-chosen data representation, an insight similarly highlighted in Chapter 1, suggesting that input structure is vital to optimizing deep learning performance in population genetics.

## Take Home Message

CNNs provide a powerful and flexible tool for population genetics, as exemplified in this thesis with applications in both demographic classification and selective sweep detection. The results underscore CNNs' adaptability, showing that they can effectively classify complex demographic scenarios and detect selective sweeps with high accuracy, often surpassing traditional methods in certain tasks like localization. However, the effectiveness

of CNNs does not hinge on network complexity alone; rather, performance is closely tied to the quality of the training data and the chosen input representation. This study highlights how even simpler CNN architectures, when paired with well-represented and structured datasets, can deliver strong results, sometimes outperforming more intricate, pre-trained models. This is essential for future applications, as it suggests that accessible, interpretable CNN models could be used widely, given the right data preparation. If we had to summarize this thesis into a single message, it would be the following: the success of machine learning in population genetics – whether for demographic inference, selective sweep localization or any of the other thrilling questions among the myriad of topics in population genetics – is tied to the careful and thoughtful representation of the input data. While model parameters and architectures are crucial, the representation of data used as inputs remains a primary driver of model success, as CNNs – but all deep-learning methods speaking more broadly – rely on these representations to effectively extract meaningful patterns from genomic information. While not a new discovery by any means, we consider it an important subject to bring to the forefront of this conclusion. We could not think of a better way to underscore this fact than by citing the insightful work of Yoshua Bengio, Aaron Courville, and Pascal Vincent, Representation Learning: A Review and New Perspective (2014): "*The performance of machine learning methods is heavily dependent on the choice of data representation (or features) on which they are applied. For that reason, much of the actual effort in deploying machine learning algorithms goes into the design of preprocessing pipelines and data transformations that result in a representation of the data that can support effective machine learning.*"

This insight captures the fundamental takeaway of our work, and we would like to conclude this thesis by emphasizing a topic we touched on only briefly: the incredible and, as of now, barely tapped potential of deep learning for population genetics. As stated by Lex Flagel, Yaniv Brandvain and Daniel R. Schrider in their paper ‘The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inference’ (Flagel et al., 2018), “CNNs have enormous potential for population genomic inference.”. This statement has been proved accurate over the past years, with considerable advancements in the field attributed to the expanding use of deep learning. Similarly, Korfmann et al., in ‘Deep Learning in Population Genetics’ (Korfmann et al., 2023) emphasize the importance of making these methods more accessible and inclusive to population geneticists to allow a broader community of researchers to work with them.

We wholeheartedly adhere to this point of view, and are convinced that the numerous successes of deep-learning applications in population genetics seen during the past few

years are only the beginning. As deep-learning tools become more available, understood, and broadly used, we believe they will not only provide solutions to existing challenges but will also allow us to tackle new questions and widen our way of considering problems in population genomics.

# Bibliography


Abbott, S., & Fairbanks, D. J. (2016). Experiments on Plant Hybrids by Gregor Mendel. *Genetics*, *204*(2), 407–422. https://doi.org/10.1534/genetics.116.195198

Achaz, G. (2008). Testing for Neutrality in Samples With Sequencing Errors. *Genetics*, *179*(3), 1409–1424. https://doi.org/10.1534/genetics.107.082198

Achaz, G. (2009). Frequency Spectrum Neutrality Tests: One for All and All for One. *Genetics*, *183*(1), 249–258. https://doi.org/10.1534/genetics.109.104042

Adams, A. M., & Hudson, R. R. (2004). Maximum-Likelihood Estimation of Demographic Parameters Using the Frequency Spectrum of Unlinked Single-Nucleotide Polymorphisms. *Genetics*, *168*(3), 1699–1712. https://doi.org/10.1534/genetics.104.030171

Aguade´ M, Langley CH (1994) Polymorphism and divergence in regions of low recombination in Drosophila. In: Non-neutral evolution. Springer, Boston, pp 67–76. https://doi.org/10.1007/978-1-4615-2383-3_6

Aguade M, Miyashita N, Langley CH (1989) Reduced variation in the yellow-achaete-scute region in natural populations of Drosophila melanogaster. Genetics 122(3):607–615 https://doi.org/10.1093/genetics/122.3.607

Alachiotis, N., Stamatakis, A., & Pavlidis, P. (2012). OmegaPlus: A scalable tool for rapid detection of selective sweeps in whole-genome datasets. *Bioinformatics*, *28*(17), 2274–2275. https://doi.org/10.1093/bioinformatics/bts419

Angermueller, C., Pärnamaa, T., Parts, L., & Stegle, O. (2016). Deep learning for computational biology. *Molecular systems biology*, *12*(7), 878. https://doi.org/10.15252/msb.20156651

Barton, N. H. (2000). Genetic hitchhiking. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, *355*(1403), 1553–1562. https://doi.org/10.1098/rstb.2000.0716

Beaumont, M. A. (2010). Approximate Bayesian Computation in Evolution and Ecology. *Annual Review of Ecology, Evolution, and Systematics*, *41*(1), 379–406. https://doi.org/10.1146/annurev-ecolsys-102209-144621

Beaumont, M. A., Zhang, W., & Balding, D. J. (2002). Approximate Bayesian Computation in Population Genetics. *Genetics*, *162*(4), 2025–2035. https://doi.org/10.1093/genetics/162.4.2025

Becquet, C., & Przeworski, M. (2007). A new approach to estimate parameters of speciation models with application to apes. *Genome Research*, *17*(10), 1505–1519. https://doi.org/10.1101/gr.6409707

Beerli, P., & Felsenstein, J. (2001). Maximum likelihood estimation of a migration matrix and effective population sizes in *n* subpopulations by using a coalescent approach. *Proceedings of the National Academy of Sciences*, *98*(8), 4563–4568. https://doi.org/10.1073/pnas.081068098

Begun DJ, Aquadro CF (1991) Molecular population genetics of the distal portion of the x chromosome in Drosophila: evidence for genetic hitchhiking of the yellow-achaete region. Genetics 129(4):1147–1158. https://doi.org/10.1093/genetics/129.4.1147

Bengio, Y., Courville, A., & Vincent, P. (2014). *Representation Learning: A Review and New Perspectives* (arXiv:1206.5538). arXiv. http://arxiv.org/abs/1206.5538

Boitard, S., Schlötterer, C., & Futschik, A. (2009). Detecting Selective Sweeps: A New Approach Based on Hidden Markov Models. *Genetics*, *181*(4), 1567–1578. https://doi.org/10.1534/genetics.108.100032

Boitard, S., Schlotterer, C., Nolte, V., Pandey, R. V., & Futschik, A. (2012). Detecting Selective Sweeps from Pooled Next-Generation Sequencing Samples. *Molecular Biology and Evolution*, *29*(9), 2177–2186. https://doi.org/10.1093/molbev/mss090

Breiman, L. (2001). "Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author)." Statist. Sci. 16 (3) 199 - 231, August 2001. https://doi.org/10.1214/ss/1009213726

Byvatov E, Schneider G. Support vector machine applications in bioinformatics. Applied Bioinformatics. 2003 ;2(2):67-77. PMID: 15130823. 10.1214/ss/1009213726

Caldas, I. V., Clark, A. G., & Messer, P. W. (2022). *Inference of selective sweep parameters through supervised learning*. https://doi.org/10.1101/2022.07.19.500702

Chan, J., Perrone, V., Spence, J. P., Jenkins, P. A., Mathieson, S., & Song, Y. S. (2018). *A Likelihood-Free Inference Framework for Population Genetic Data using Exchangeable Neural Networks* [Preprint]. Evolutionary Biology. https://doi.org/10.1101/267211

Charlesworth, B. (2022). Fisher's historic 1922 paper *On the dominance ratio*. *Genetics*, *220*(3), iyac006. https://doi.org/10.1093/genetics/iyac006

Chen, H., Patterson, N., & Reich, D. (2010). Population differentiation as a test for selective sweeps. *Genome Research*, *20*(3), 393–402. https://doi.org/10.1101/gr.100545.109

Corbett-Detig, R., & Nielsen, R. (2017). A Hidden Markov Model Approach for Simultaneously Estimating Local Ancestry and Admixture Time Using Next Generation Sequence Data in Samples of Arbitrary Ploidy. *PLOS Genetics*, *13*(1), e1006529. https://doi.org/10.1371/journal.pgen.1006529

Cornuet, J.-M., Santos, F., Beaumont, M. A., Robert, C. P., Marin, J.-M., Balding, D. J., Guillemaud, T., & Estoup, A. (2008). Inferring population history with *DIY ABC*: A user-friendly approach to approximate Bayesian computation. *Bioinformatics*, *24*(23), 2713–2719. https://doi.org/10.1093/bioinformatics/btn514

Darwin, C. (1859). On the origin of species by means of natural selection (Vol. 167). John Murray, London.

DeGiorgio, M., Huber, C. D., Hubisz, M. J., Hellmann, I., & Nielsen, R. (2016). SweepFinder 2: Increased sensitivity, robustness and flexibility. *Bioinformatics*, *32*(12), 1895–1897. https://doi.org/10.1093/bioinformatics/btw051

Ellegren, H., & Galtier, N. (2016). Determinants of genetic diversity. *Nature Reviews Genetics*, *17*(7), 422–433. https://doi.org/10.1038/nrg.2016.58

Ewing, G., & Hermisson, J. (2010). MSMS: A coalescent simulation program including recombination, demographic structure and selection at a single locus. *Bioinformatics*, *26*(16), 2064–2065. https://doi.org/10.1093/bioinformatics/btq322

Excoffier, L., Marchi, N., Marques, D. A., Matthey-Doret, R., Gouy, A., & Sousa, V. C. (2021). *fastsimcoal2*: Demographic inference under complex evolutionary scenarios. *Bioinformatics*, *37*(24), 4882–4885. https://doi.org/10.1093/bioinformatics/btab468

Fay, J. C., & Wu, C. I. (2000). Hitchhiking Under Positive Darwinian Selection, *Genetics*, Volume 155, Issue 3, 1 July 2000, Pages 1405–1413, https://doi.org/10.1093/genetics/155.3.1405

Fisher, R. A. (1999). The genetical theory of natural selection: a complete variorum edition. Oxford University Press.

Flagel, L., Brandvain, Y., & Schrider, D. R. (2019). The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inference. *Molecular Biology and Evolution*, *36*(2), 220–238. https://doi.org/10.1093/molbev/msy224

Fraïsse, C., Popovic, I., Mazoyer, C., Spataro, B., Delmotte, S., Romiguier, J., Loire, É., Simon, A., Galtier, N., Duret, L., Bierne, N., Vekemans, X., & Roux, C. (2021). DILS: Demographic inferences with linked selection by using ABC. *Molecular Ecology Resources*, *21*(8), 2629–2644. https://doi.org/10.1111/1755-0998.13323

Frankham, R., Jones, L. P., & Barker, J. S. F. (1968). The effects of population size and selection intensity in selection for a quantitative character in *Drosophila*: I. Short-term response to selection. *Genetical Research*, *12*(3), 237–248. https://doi.org/10.1017/S0016672300011848

Galtier, N., Depaulis, F., & Barton, N. H. (2000). Detecting Bottlenecks and Selective Sweeps From DNA Sequence Polymorphism. *Genetics*, *155*(2), 981–987. https://doi.org/10.1093/genetics/155.2.981

Garud, N. R., & Rosenberg, N. A. (2015). Enhancing the mathematical properties of new haplotype homozygosity statistics for the detection of selective sweeps. *Theoretical Population Biology*, *102*, 94–101. https://doi.org/10.1016/j.tpb.2015.04.001

Gazave, E., Ma, L., Chang, D., Coventry, A., Gao, F., Muzny, D., Boerwinkle, E., Gibbs, R. A., Sing, C. F., Clark, A. G., & Keinan, A. (2014). Neutral genomic regions refine models of recent rapid human population growth. *Proceedings of the National Academy of Sciences*, *111*(2), 757–762. https://doi.org/10.1073/pnas.1310398110

Gerbault, P., Moret, C., Currat, M., & Sanchez-Mazas, A. (2009). Impact of Selection and Demography on the Diffusion of Lactase Persistence. *PLoS ONE*, *4*(7), e6369. https://doi.org/10.1371/journal.pone.0006369

Gower, G., Picazo, P. I., Fumagalli, M., & Racimo, F. (2021). Detecting adaptive introgression in human evolution using convolutional neural networks. *eLife*, *10*, e64669. https://doi.org/10.7554/eLife.64669

Grealey, J., Lannelongue, L., Saw, W.-Y., Marten, J., Méric, G., Ruiz-Carmona, S., & Inouye, M. (2022). The Carbon Footprint of Bioinformatics. *Molecular Biology and Evolution*, *39*(3), msac034. https://doi.org/10.1093/molbev/msac034

Gutenkunst, R. N., Hernandez, R. D., Williamson, S. H., & Bustamante, C. D. (2009). Inferring the Joint Demographic History of Multiple Populations from Multidimensional SNP Frequency Data. *PLoS Genetics*, *5*(10), e1000695. https://doi.org/10.1371/journal.pgen.1000695

Hahn, M. W. (2008). TOWARD A SELECTION THEORY OF MOLECULAR EVOLUTION. *Evolution*, *62*(2), 255–265. https://doi.org/10.1111/j.1558-5646.2007.00308.x

Haldane, J. B. (1990). The causes of evolution (Vol. 5). Princeton University Press.

Hamid, I., Korunes, K. L., Beleza, S., & Goldberg, A. (2021). Rapid adaptation to malaria facilitated by admixture in the human population of Cabo Verde. *eLife*, *10*, e63177. https://doi.org/10.7554/eLife.63177

Hamid, I., Korunes, K. L., Schrider, D. R., & Goldberg, A. (2023). Localizing post-admixture adaptive variants with object detection on ancestry-painted chromosomes. *Molecular Biology and Evolution*, *40*(4), msad074. https://doi.org/10.1093/molbev/msad074

Hardy, G. H. (1908). Mendelian proportions in a mixed population. Science, 28(706), 49-50.

Harris, A. M., Garud, N. R., & DeGiorgio, M. (n.d.). Detection and Classification of Hard and Soft Sweeps from Unphased Genotypes by Multilocus Genotype Identity, *Genetics*, Volume 210, Issue 4, 1 December 2018, Pages 1429–1452, https://doi.org/10.1534/genetics.118.301502

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition* (arXiv:1512.03385). arXiv. http://arxiv.org/abs/1512.03385

Hellenthal, G., Busby, G. B. J., Band, G., Wilson, J. F., Capelli, C., Falush, D., & Myers, S. (2014). A Genetic Atlas of Human Admixture History. *Science*, *343*(6172), 747–751. https://doi.org/10.1126/science.1243518

Hey, J. (2010). Isolation with Migration Models for More Than Two Populations. *Molecular Biology and Evolution*, *27*(4), 905–920. https://doi.org/10.1093/molbev/msp296

Hey, J., & Nielsen, R. (2007). Integration within the Felsenstein equation for improved Markov chain Monte Carlo methods in population genetics. *Proceedings of the National Academy of Sciences*, *104*(8), 2785–2790. https://doi.org/10.1073/pnas.0611164104

Hill, W. G., & Robertson, A. (1968). Linkage disequilibrium in finite populations. *Theoretical and Applied Genetics*, *38*(6), 226–231. https://doi.org/10.1007/BF01245622

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., & Kingsbury, B. (2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97. https://doi.org/10.1109/MSP.2012.2205597

Itan, Y., Powell, A., Beaumont, M. A., Burger, J., & Thomas, M. G. (2009). The Origins of Lactase Persistence in Europe. *PLoS Computational Biology*, *5*(8), e1000491. https://doi.org/10.1371/journal.pcbi.1000491

Jensen, J. D., Kim, Y., DuMont, V. B., Aquadro, C. F., & Bustamante, C. D. (2005). Distinguishing Between Selective Sweeps and Demography Using DNA Polymorphism Data. *Genetics*, *170*(3), 1401–1410. https://doi.org/10.1534/genetics.104.038224

Kern, A. D., & Schrider, D. R. (2018). diploS/HIC: An Updated Approach to Classifying Selective Sweeps. *G3 (Bethesda, Md.)*, *8*(6), 1959–1970. https://doi.org/10.1534/g3.118.200262

Kim, Y., & Nielsen, R. (2004). Linkage Disequilibrium as a Signature of Selective Sweeps. *Genetics*, *167*(3), 1513–1524. https://doi.org/10.1534/genetics.103.025387

Kim, Y., & Stephan, W. (2002). Detecting a Local Signature of Genetic Hitchhiking Along a Recombining Chromosome. *Genetics*, *160*(2), 765–777. https://doi.org/10.1093/genetics/160.2.765

Kimura, M. (1968). Evolutionary Rate at the Molecular Level. *Nature*, *217*(5129), 624–626. https://doi.org/10.1038/217624a0

Kingma, D. P., & Ba, J. (2017). *Adam: A Method for Stochastic Optimization* (arXiv:1412.6980). arXiv. http://arxiv.org/abs/1412.6980

Kittlein, M. J., Mora, M. S., Mapelli, F. J., Austrich, A., & Gaggiotti, O. E. (2022). Deep learning and satellite imagery predict genetic diversity and differentiation. *Methods in Ecology and Evolution*, *13*(3), 711–721. https://doi.org/10.1111/2041-210X.13775

Ko, K. H. (2016). Hominin interbreeding and the evolution of human variation. *Journal of Biological Research-Thessaloniki*, *23*(1), 17. https://doi.org/10.1186/s40709-016-0054-7

Korfmann, K., Gaggiotti, O. E., & Fumagalli, M. (2023). Deep learning in population genetics. *Genome Biology and Evolution*, *15*(2), evad008. https://doi.org/10.1093/gbe/evad008

Koropoulis, A., Alachiotis, N., & Pavlidis, P. (2020). Detecting Positive Selection in Populations Using Genetic Data. In J. Y. Dutheil (Ed.), *Statistical Population Genomics* (Vol. 2090, pp. 87–123). Springer US. https://doi.org/10.1007/978-1-0716-0199-0_5

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, *160*(1), 3-24. https://informatica.si/index.php/informatica/article/viewFile/148/140

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Kuhner, M. K., Yamato, J., & Felsenstein, J. (2000). Maximum Likelihood Estimation of Recombination Rates From Population Data. *Genetics*, *156*(3), 1393–1401. https://doi.org/10.1093/genetics/156.3.1393

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Legrand, D., Tenaillon, M. I., Matyot, P., Gerlach, J., Lachaise, D., & Cariou, M.-L. (2009). Species-Wide Genetic Variation and Demographic History of *Drosophila sechellia* , a Species Lacking Population Structure. *Genetics*, *182*(4), 1197–1206. https://doi.org/10.1534/genetics.108.092080

Lewontin, R. C. (1964). THE INTERACTION OF SELECTION AND LINKAGE. I. GENERAL CONSIDERATIONS; HETEROTIC MODELS. *Genetics*, *49*(1), 49–67. https://doi.org/10.1093/genetics/49.1.49

Li, H., & Stephan, W. (2006). Inferring the demographic history and rate of adaptive substitution in Drosophila. *PLoS genetics, 2(10), e166.* *https://doi.org/10.1371/journal.pgen.0020166*

Li, J., Li, H., Jakobsson, M., Li, S., Sjödin, P., & Lascoux, M. (2012). Joint analysis of demography and selection in population genetics: Where do we stand and where could we go? *Molecular Ecology*, *21*(1), 28–44. https://doi.org/10.1111/j.1365-294X.2011.05308.x

Li, Y., Xie, S., Chen, X., Dollar, P., He, K., & Girshick, R. (2021). Benchmarking Detection Transfer Learning with Vision Transformers (arXiv:2111.11429). arXiv. https://doi.org/10.48550/arXiv.2111.11429

Libbrecht, M. W., & Noble, W. S. (2015). Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, *16*(6), 321–332. https://doi.org/10.1038/nrg3920

Lin, K., Li, H., Schlötterer, C., & Futschik, A. (2011). Distinguishing Positive Selection From Neutral Evolution: Boosting the Performance of Summary Statistics. *Genetics*, *187*(1), 229–244. https://doi.org/10.1534/genetics.110.122614

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015). *Microsoft COCO: Common Objects in Context* (arXiv:1405.0312). arXiv. http://arxiv.org/abs/1405.0312

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 936–944. https://doi.org/10.1109/CVPR.2017.106

Lindsay, B. G. (1988). Composite likelihood methods. *Comtemporary Mathematics*, *80*(1), 221-239.

Lopez, C., Tucker, S., Salameh, T., & Tucker, C. (2018). An unsupervised machine learning method for discovering patient clusters based on genetic signatures. *Journal of Biomedical Informatics*, *85*, 30–39. https://doi.org/10.1016/j.jbi.2018.07.004

Lukić, S., & Hey, J. (2012). Demographic Inference Using Spectral Methods on SNP Data, with an Analysis of the Human Out-of-Africa Expansion. *Genetics*, *192*(2), 619–639. https://doi.org/10.1534/genetics.112.141846

Marjoram, P., Molitor, J., Plagnol, V., & Tavaré, S. (2003). Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, *100*(26), 15324–15328. https://doi.org/10.1073/pnas.0306899100

Marth, G. T., Czabarka, E., Murvai, J., & Sherry, S. T. (2004). The Allele Frequency Spectrum in Genome-Wide Human Variation Data Reveals Signals of Differential Demographic History in Three Large World Populations. *Genetics*, *166*(1), 351–372. https://doi.org/10.1534/genetics.166.1.351

Martin, S. H., Dasmahapatra, K. K., Nadeau, N. J., Salazar, C., Walters, J. R., Simpson, F., Blaxter, M., Manica, A., Mallet, J., & Jiggins, C. D. (2013). Genome-wide evidence for speciation with gene flow in *Heliconius* butterflies. *Genome Research*, *23*(11), 1817–1828. https://doi.org/10.1101/gr.159426.113

Mendel, G. (1996). Experiments in plant hybridization (1865). *Verhandlungen des naturforschenden Vereins Brünn.) Available online: www. mendelweb. org/Mendel. html*

Menghani, G. (2023). Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better. *ACM Computing Surveys*, *55*(12), 1–37. https://doi.org/10.1145/3578938

Mitchell, T. M., & Mitchell, T. M. (1997). *Machine learning* (Vol. 1, No. 9). New York: McGraw-hill.

Miyashita NT. Molecular and phenotypic variation of the Zw locus region in Drosophila melanogaster. Genetics. 1990 Jun;125(2):407-19. doi: 10.1093/genetics/125.2.407. PMID: 1974224; PMCID: PMC1204029.

Moran, P. A. P. (1958). THE RATE OF APPROACH TO HOMOZYGOSITY. *Annals of Human Genetics*, *23*(1), 1–5. https://doi.org/10.1111/j.1469-1809.1958.tb01436.x

Mughal, M. R., & DeGiorgio, M. (2019). Localizing and Classifying Adaptive Targets with Trend Filtered Regression. *Molecular Biology and Evolution*, *36*(2), 252–270. https://doi.org/10.1093/molbev/msy205

Naduvilezhath, L., Rose, L. E., & Metzler, D. (2011). Jaatha: A fast composite-likelihood approach to estimate demographic parameters: ESTIMATION OF DEMOGRAPHIC PARAMETERS. *Molecular Ecology*, *20*(13), 2709–2723. https://doi.org/10.1111/j.1365-294X.2011.05131.x

Nair, S. (2003). A Selective Sweep Driven by Pyrimethamine Treatment in Southeast Asian Malaria Parasites. *Molecular Biology and Evolution*, *20*(9), 1526–1536. https://doi.org/10.1093/molbev/msg162

Nait Saada, J., Tsangalidou, Z., Stricker, M., & Palamara, P. F. (2023). Inference of Coalescence Times and Variant Ages Using Convolutional Neural Networks. *Molecular Biology and Evolution*, *40*(10), msad211. https://doi.org/10.1093/molbev/msad211

Nei, M., & Li, W. H. (1979). Mathematical model for studying genetic variation in terms of restriction endonucleases. *Proceedings of the National Academy of Sciences*, *76*(10), 5269–5273. https://doi.org/10.1073/pnas.76.10.5269

Nielsen, R. (2000). Estimation of Population Parameters and Recombination Rates From Single Nucleotide Polymorphisms. *Genetics*, *154*(2), 931–942. https://doi.org/10.1093/genetics/154.2.931

Nielsen, R. (2005). Genomic scans for selective sweeps using SNP data. *Genome Research*, *15*(11), 1566–1575. https://doi.org/10.1101/gr.4252305

O'Shea, K., & Nash, R. (2015). *An Introduction to Convolutional Neural Networks* (arXiv:1511.08458). arXiv. http://arxiv.org/abs/1511.08458

Pascual, M., Chapuis, M. P., Mestres, F., Balanyà, J., Huey, R. B., Gilchrist, G. W., Serra, L., & Estoup, A. (2007). Introduction history of *Drosophila subobscura* in the New World: A microsatellite-based survey using ABC methods. *Molecular Ecology*, *16*(15), 3069–3083. https://doi.org/10.1111/j.1365-294X.2007.03336.x

Pavinato, V. A. C., De Mita, S., Marin, J.-M., & De Navascués, M. (2022). Joint inference of adaptive and demographic history from temporal population genomic data. *Peer Community Journal*, *2*, e78. https://doi.org/10.24072/pcjournal.203

Pavlidis, P., Hutter, S., & Stephan, W. (2008). A population genomic approach to map recent positive selection in model species. *Molecular Ecology*, *17*(16), 3585–3598. https://doi.org/10.1111/j.1365-294X.2008.03852.x

Pavlidis, P., Jensen, J. D., & Stephan, W. (2010a). Searching for Footprints of Positive Selection in Whole-Genome SNP Data From Nonequilibrium Populations. *Genetics*, *185*(3), 907–922. https://doi.org/10.1534/genetics.110.116459

Pavlidis, P., Jensen, J. D., & Stephan, W. (2010b). Searching for Footprints of Positive Selection in Whole-Genome SNP Data From Nonequilibrium Populations. *Genetics*, *185*(3), 907–922. https://doi.org/10.1534/genetics.110.116459

Pavlidis, P., Živković, D., Stamatakis, A., & Alachiotis, N. (2013). SweeD: Likelihood-Based Detection of Selective Sweeps in Thousands of Genomes. *Molecular Biology and Evolution*, *30*(9), 2224–2234. https://doi.org/10.1093/molbev/mst112

Pespeni, M. H., Garfield, D. A., Manier, M. K., & Palumbi, S. R. (2012). Genome-wide polymorphisms show unexpected targets of natural selection. *Proceedings of the Royal Society B: Biological Sciences*, *279*(1732), 1412–1420. https://doi.org/10.1098/rspb.2011.1823

Piou, C. (2015). Information criteria and approximate Bayesian computing for agent-based modelling in ecology: New tools to infer on Individual-level processes. *Biomath Communications*, *2*(1). https://doi.org/10.11145/511

Pritchard, J. K., Stephens, M., & Donnelly, P. (2000). Inference of Population Structure Using Multilocus Genotype Data. *Genetics*, *155*(2), 945–959. https://doi.org/10.1093/genetics/155.2.945

Przeworski, M. (2003). Estimating the Time Since the Fixation of a Beneficial Allele. *Genetics*, *164*(4), 1667–1676. https://doi.org/10.1093/genetics/164.4.1667

Pudlo, P., Marin, J.-M., Estoup, A., Cornuet, J.-M., Gautier, M., & Robert, C. P. (2016). Reliable ABC model choice via random forests. *Bioinformatics*, *32*(6), 859–866. https://doi.org/10.1093/bioinformatics/btv684

Räz, T. (2024). ML interpretability: Simple isn't easy. Studies in history and philosophy of science, 103, 159-167. https://doi.org/10.1016/j.shpsa.2023.12.007

Raynal, L., Marin, J.-M., Pudlo, P., Ribatet, M., Robert, C. P., & Estoup, A. (2019). ABC random forests for Bayesian parameter inference. *Bioinformatics*, *35*(10), 1720–1728. https://doi.org/10.1093/bioinformatics/bty867

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497 [Cs]*. http://arxiv.org/abs/1506.01497

Ronen, R., Udpa, N., Halperin, E., & Bafna, V. (2013). Learning Natural Selection from the Site Frequency Spectrum, *Genetics*, Volume 195, Issue 1, 1 September 2013, Pages 181–193, https://doi.org/10.1534/genetics.113.152587

Roux, C., Pauwels, M., Ruggiero, M.-V., Charlesworth, D., Castric, V., & Vekemans, X. (2013). Recent and Ancient Signature of Balancing Selection around the S-Locus in Arabidopsis halleri and A. lyrata. *Molecular Biology and Evolution*, *30*(2), 435–447. https://doi.org/10.1093/molbev/mss246

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986). https://doi.org/10.1038/323533a0

Sanchez, T. (2022). *Reconstructing our past: deep learning for population genetics* (Doctoral dissertation, Université Paris-Saclay).

.

Sanchez, T., Bray, E. M., Jobic, P., Guez, J., Letournel, A.-C., Charpiat, G., Cury, J., & Jay, F. (2023). dnadna: A deep learning framework for population genetics inference. *Bioinformatics*, *39*(1), btac765. https://doi.org/10.1093/bioinformatics/btac765

M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. -C. Chen (2018). "MobileNetV2: Inverted Residuals and Linear Bottlenecks" doi: 10.1109/CVPR.2018.00474

Sankararaman, S., Mallick, S., Dannemann, M., Prüfer, K., Kelso, J., Pääbo, S., Patterson, N., & Reich, D. (2014). The genomic landscape of Neanderthal ancestry in present-day humans. *Nature*, *507*(7492), 354–357. https://doi.org/10.1038/nature12961

Schrider, D. R., Ayroles, J., Matute, D. R., & Kern, A. D. (2018). Supervised machine learning reveals introgressed loci in the genomes of Drosophila simulans and D. sechellia. *PLOS Genetics*, *14*(4), e1007341. https://doi.org/10.1371/journal.pgen.1007341

Schrider, D. R., & Kern, A. D. (2016). S/HIC: Robust Identification of Soft and Hard Sweeps Using Machine Learning. *PLOS Genetics*, *12*(3), e1005928. https://doi.org/10.1371/journal.pgen.1005928

Schrider, D. R., & Kern, A. D. (2018a). Supervised Machine Learning for Population Genetics: A New Paradigm. *Trends in Genetics*, *34*(4), 301–312. https://doi.org/10.1016/j.tig.2017.12.005

Schrider, D. R., & Kern, A. D. (2018b). Supervised Machine Learning for Population Genetics: A New Paradigm. *Trends in Genetics*, *34*(4), 301–312. https://doi.org/10.1016/j.tig.2017.12.005

Schrider, D. R., Shanku, A. G., & Kern, A. D. (2016). Effects of Linked Selective Sweeps on Demographic Inference and Model Selection. *Genetics*, *204*(3), 1207–1223. https://doi.org/10.1534/genetics.116.190223

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, *34*(1), 1–47. https://doi.org/10.1145/505282.505283

Sella, G., Petrov, D. A., Przeworski, M., & Andolfatto, P. (2009). Pervasive Natural Selection in the Drosophila Genome? *PLoS Genetics*, *5*(6), e1000495. https://doi.org/10.1371/journal.pgen.1000495

Sheehan, S., & Song, Y. S. (2016). Deep Learning for Population Genetic Inference. *PLOS Computational Biology*, *12*(3), e1004845. https://doi.org/10.1371/journal.pcbi.1004845

Shriner, D., Liu, Y., Nickle, D. C., & Mullins, J. I. (2006). Evolution of Intrahost Hiv—1 Genetic Diversity During Chronic Infection. *Evolution*, *60*(6), 1165–1176. https://doi.org/10.1111/j.0014-3820.2006.tb01195.x

Simonsen, K. L., Churchill, G. A., & Aquadro, C. F. (1995). Properties of statistical tests of neutrality for DNA polymorphism data. *Genetics*, *141*(1), 413–429. https://doi.org/10.1093/genetics/141.1.413

Siol, M., Wright, S. I., & Barrett, S. C. H. (2010). The population genomics of plant adaptation. *New Phytologist*, *188*(2), 313–332. https://doi.org/10.1111/j.1469-8137.2010.03401.x

Slon, V., Mafessoni, F., Vernot, B., De Filippo, C., Grote, S., Viola, B., Hajdinjak, M., Peyrégne, S., Nagel, S., Brown, S., Douka, K., Higham, T., Kozlikin, M. B., Shunkov, M. V., Derevianko, A. P., Kelso, J., Meyer, M., Prüfer, K., & Pääbo, S. (2018). The genome of the offspring of a Neanderthal mother and a Denisovan father. *Nature*, *561*(7721), 113–116. https://doi.org/10.1038/s41586-018-0455-x

De Smet, Y. (2020). Through the fog: evolutionary insights provide novel genus-and species-level boundaries in tribe Hydrangeeae and genus Hydrangea (Doctoral dissertation, Ghent University).

Smith, C. C. R., & Kern, A. D. (2023). disperseNN2: A neural network for estimating dispersal distance from georeferenced polymorphism data. *BMC Bioinformatics*, *24*(1), 385. https://doi.org/10.1186/s12859-023-05522-7

Smith, J. M., & Haigh, J. (1974). The hitch-hiking effect of a favourable gene. *Genetical Research*, *23*(1), 23–35. https://doi.org/10.1017/S0016672300014634

Solow, A. R., & Smith, W. K. (2009). Estimating species number under an inconvenient abundance model. *JABES* **14**, 242–252 (2009). https://doi.org/10.1198/jabes.2009.0015

Tajima, F. (1989). Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics*, *123*(3), 585–595. https://doi.org/10.1093/genetics/123.3.585

Tan, M., & Le, Q. V. (2020). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* (arXiv:1905.11946). arXiv. http://arxiv.org/abs/1905.11946

Tanaka, M. M., Francis, A. R., Luciani, F., & Sisson, S. A. (2006). Using Approximate Bayesian Computation to Estimate Tuberculosis Transmission Parameters From Genotype Data. *Genetics*, *173*(3), 1511–1520. https://doi.org/10.1534/genetics.106.055574

Tavaré, S., Balding, D. J., Griffiths, R. C., & Donnelly, P. (1997). Inferring Coalescence Times From DNA Sequence Data. *Genetics*, *145*(2), 505–518. https://doi.org/10.1093/genetics/145.2.505

Tennessen, J. A., Bigham, A. W., O'Connor, T. D., Fu, W., Kenny, E. E., Gravel, S., McGee, S., Do, R., Liu, X., Jun, G., Kang, H. M., Jordan, D., Leal, S. M., Gabriel, S., Rieder, M. J., Abecasis, G., Altshuler, D., Nickerson, D. A., Boerwinkle, E., … on behalf of the NHLBI Exome Sequencing Project. (2012). Evolution and Functional Impact of Rare Coding Variation from Deep Sequencing of Human Exomes. *Science*, *337*(6090), 64–69. https://doi.org/10.1126/science.1219240

Tian, D., Araki, H., Stahl, E., Bergelson, J., & Kreitman, M. (2002). Signature of balancing selection in Arabidopsis. *Proceedings of the National Academy of Sciences*, *99*(17), 11525–11530. https://doi.org/10.1073/pnas.172203599

Torada, L., Lorenzon, L., Beddis, A., Isildak, U., Pattini, L., Mathieson, S., & Fumagalli, M. (2019). ImaGene: A convolutional neural network to quantify natural selection from genomic data. *BMC Bioinformatics*, *20*(S9), 337. https://doi.org/10.1186/s12859-019-2927-x

Turing, A. M. (1950). Computing machinery and intelligence (pp. 23-65). Springer Netherlands.

Vernot, B., Tucci, S., Kelso, J., Schraiber, J. G., Wolf, A. B., Gittelman, R. M., Dannemann, M., Grote, S., McCoy, R. C., Norton, H., Scheinfeldt, L. B., Merriwether, D. A., Koki, G., Friedlaender, J. S., Wakefield, J., Pääbo, S., & Akey, J. M. (2016). Excavating Neandertal and Denisovan DNA from the genomes of Melanesian individuals. *Science*, *352*(6282), 235–239. https://doi.org/10.1126/science.aad9416

Watterson, G.A. (1975), "On the number of segregating sites in genetical models without recombination.", Theoretical Population Biology, 7 (2): 256–276. doi:10.1016/0040-5809(75)90020-9, PMID 1145509

Weinberg, W., (1908) "Über den Nachweis der Vererbung beim Menschen. Jahresh. Ver. Vaterl. Naturkd. Wu¨rttemb". 64: 369–382 (English translations in Boyer 1963 and Jameson 1977)

Whitehouse, L. S., & Schrider, D. R. (2023). Timesweeper: accurately identifying selective sweeps using population genomic time series, *Genetics*, Volume 224, Issue 3, July 2023, iyad084, https://doi.org/10.1093/genetics/iyad084

Wilson, D. J., Gabriel, E., Leatherbarrow, A. J. H., Cheesbrough, J., Gee, S., Bolton, E., Fox, A., Hart, C. A., Diggle, P. J., & Fearnhead, P. (2009). Rapid Evolution and the Importance of Recombination to the Gastroenteric Pathogen Campylobacter jejuni.

*Molecular Biology and Evolution*, *26*(2), 385–397. https://doi.org/10.1093/molbev/msn264

Wollstein, A., & Stephan, W. (2015). Inferring positive selection in humans from genomic data. *Investigative Genetics*, *6*(1), 5. https://doi.org/10.1186/s13323-015-0023-1

Wright, S. (1931). Evolution in Mendelian Populations. *Genetics*, *16*(2), 97–159.

Xue, A. T., Schrider, D. R., Kern, A. D., Ag1000g Consortium, Della Torre, A., Kern, A., Caputo, B., Kabula, B., White, B., Godfray, C., Edi, C., Wilding, C., Neafsey, D., Schrider, D., Conway, D., Weetman, D., Ayala, D., Kwiatkowski, D., Sharakhov, I., … Antão, T. (2021). Discovery of Ongoing Selective Sweeps within *Anopheles* Mosquito Populations Using Deep Learning. *Molecular Biology and Evolution*, *38*(3), 1168–1183. https://doi.org/10.1093/molbev/msaa259

Yelmen, B., Decelle, A., Ongaro, L., Marnetto, D., Tallec, C., Montinaro, F., Furtlehner, C., Pagani, L., & Jay, F. (2021). Creating artificial human genomes using generative neural networks. *PLOS Genetics*, *17*(2), e1009303. https://doi.org/10.1371/journal.pgen.100930

# Acknowledgements

Je voudrais commencer par exprimer ma plus profonde gratitude à toutes les personnes qui ont participé à la réalisation de cette thèse. Sans le soutien, les conseils et les encouragements de nombreuses personnes et organisations, ce travail n'aurait pas été possible.

## À ceux qui m’ont financé

Je tiens à remercier la fondation I-SITE ULNE et l’Université de Lille d’avoir financé ma thèse au travers du programme “*AI_PhD@Lille*” porté par Mme Clarisse Dhaenens, ainsi que l’école doctorale SMRE pour son soutien. Mes remerciements vont également à l’équipe Évolution et Écologie pour m’avoir accompagné et avoir financé mes déplacements et missions, ainsi que pour m’avoir apporté un soutien financier pour les derniers mois de ma thèse. C’est grâce à tous ces acteurs que j’ai pu réaliser ma thèse dans d’excellentes conditions, et je leur en suis profondément reconnaissant.

## À mes encadrants de thèse

Camille, je tiens à te remercier de m’avoir proposé de m’embarquer dans l’aventure qu’a été cette thèse pour moi. Merci d’avoir toujours continué, pendant plus de trois ans, à me transmettre tes connaissances et à partager ton expérience avec moi. Merci d'avoir su trouver les mots, dans mes périodes de doutes et les moments les plus difficiles, pour me remotiver et me pousser à persévérer. Enfin, merci pour ces petits moments de détente à la fin des réunions, à discuter de tes dernières meilleures (ou pires) lectures et anecdotes. Sans toi, je ne serais pas là aujourd’hui. Merci pour tout Camille.

Xavier, je souhaite te remercier d’avoir toujours su te rendre disponible, tout au long de ma thèse, malgré toutes tes obligations entre la direction du laboratoire, à l’école doctorale, avec les étudiants et j’imagine que j’en oublie plus de la moitié. Ton regard, posé et toujours bienveillant, a apporté une précieuse perspective à ma thèse. Pour moi qui ne suis pas toujours à l’aise avec les aspects administratifs, ta réactivité sur ces questions a été une aide inestimable. Merci d’avoir accepté de diriger cette thèse, qui n’aurait certainement pas eu la même saveur sans toi.

# À tout ceux qui m’ont accompagné et aidé sur les aspects techniques

Le travail présenté dans cette thèse n’aurait pas pu être possible sans une aide et des ressources précieuses qui ont été partagées avec moi. Je tiens tout particulièrement à remercier :

- L’Institut Français de Bioinformatique (IFB) pour l’accès à leur clusters de calcul et leur aide pour leur utilisation.

- Le CNRS, l’Université Grenoble Alpes et le MIAI pour la formation FIDLE, qui a été une aide incroyable pour moi qui n’avait aucune formation préalable en deep-learning.

- Mathieu Genete et Clément Mazoyer, du plateau bioinformatique de l’équipe Évolution et Écologie, pour leur soutien et leur conseils (je ne lancerais pas de jobs en cascade, promis).

- Caroline de Pourtalès, du Réseau des ingénieurs CNRS du Programme national de recherche en intelligence artificielle (PNRIA), pour son aide inestimable dans le développement et l’utilisation du pipeline d’entraînement et d'inférence des CNNs. Ton arrivée et ta présence dans ce projet ont sans aucun doute permis à cette thèse d’arriver à son terme de manière sereine.

# À toutes les personnes qui ont participé à mon parcours

J’ai une pensée particulière pour l’équipe d’enseignement, et la promo 2014-2015 de la classe préparatoire aux grandes écoles CPGE du lycée Roland Garros de l’île de la Réunion (et un remerciement spécial à toi PN). Mes deux années au sein de cette classe préparatoire, bien que n’ayant pas abouties pour moi sur une admission au concours Véto, m’ont profondément marquées et font partie des expériences qui m’ont le plus influencées et permis d’avoir, je pense, la mentalité nécessaire pour poursuivre mes études jusqu’au Doctorat.

J’ai également une grande reconnaissance envers le laboratoire “Schrider Lab” de m’avoir accueilli lors d’un séjour de quelques semaines outre-Atlantique. Ce court séjour m’a énormément aidé à avoir davantage confiance en mes connaissances et ma compréhension en deep-learning et son utilisation en génétique des populations. Un remerciement tout particulier à Amjad, Logan and Dan, ainsi qu’à tous les autres membres du Schrider Lab, pour leur accueil.

Je souhaite également remercier Marina Voinson ainsi que Thomas Lesaffre de m’avoir accueillie en tant que stagiaire et de m’avoir accompagné durant mes premiers pas dans le monde de la recherche. Au travers de ces stages et de nos échanges, vous avez tous les deux su (ainsi que Sylvain qui n’était jamais très loin) cultiver chez moi l’envie de poursuivre dans ce domaine.

Étant un ancien élève de Licence et du Master lié au laboratoire, j'ai eu la chance de pouvoir évoluer au sein de l'équipe Évolution et Écologie, et de voir certains de mes enseignants devenir des collègues. À tous ceux qui m'ont connu en tant qu'étudiant, Anne, Céline, Eléonore, Isabelle, Jean-François, Nina, Sylvain,... et tous les autres que je n'ai pas cité. Merci de m'avoir accueilli à bras ouverts dans l'équipe EE et d'avoir continué à me soutenir à mesure que j'avançais dans mes études et mon parcours.

Enfin, je veux également te remercier une fois de plus Camille, de m'avoir offert l'opportunité de travailler avec toi sur les prémices de ce qui allait devenir mon futur sujet de thèse dès mon stage de fin de Master, de m'avoir accompagné dans la recherche d'un financement, et d'avoir été là avec moi dans cette aventure depuis bientôt 4 ans.

## Aux permanents, "*les grands*"

J'ai également pu rencontrer tous les autres membres du laboratoire, qui ont tous contribué à me faire passer trois ans (un peu plus !) dans un environnement chaleureux et accueillant. Particulièrement,

- Merci infiniment à toi Sylvain. C'est en partie toi qui m'a convaincu que je voulais et surtout pouvais continuer dans la recherche. Tu m'as transmis cette volonté de me poser les bonnes questions et m'a fait prendre conscience de l'importance de prendre du recul (et de dire non à la méthode du doigt mouillé !).

- Merci Eléonore pour ta présence à certains moments difficiles, et pour ces petites fenêtres de calme et de tendresse quand nos discussions finissaient presque toujours sur les dernières nouvelles de Cassiopée.

- Merci encore à vous deux, ainsi qu'à Anne et Nina de m'avoir accompagné dans ma découverte du monde de l'enseignement avec ces longues heures de TP ECAD (souvent aux pires horaires possibles).

- Merci à toi Christelle, d'avoir souvent prit de mes nouvelles lorsqu'on se croisait au hasard d'une pause ou dans les couloirs, et pour tes tous premiers conseils qui remontent à l'époque où je soumettais ma demande de financement (ton document avec toutes tes questions à l'époque m'avait beaucoup marqué).

- Jef' et Isabelle, merci d'avoir été là pendant mes études. Vous avez tous les deux su me donner l'envie de continuer dans ce cursus au moment où je me posais le plus de questions. Et vous avez toujours été, durant ma thèse, des soutiens discrets mais réconfortants.

- Merci Cécile, Christelle, Laurence et Anne-Cat' pour vos sourires dès que vous me croisiez - même si nous n'avons pas eu l'occasion de travailler ensemble, votre bonne humeur lorsque l'on se croisait était toujours un moment agréable dans ma journée.

- Merci pour tout ce que tu fais pour le laboratoire Pierre, pour ta présence et ton savoir sans lesquels le bâtiment tout entier se serait certainement déjà effondré !

- Enfin, merci à toutes les personnes avec qui j'ai pu travailler ou qui ont simplement pu me croiser au laboratoire, et je n'ai pas cité ici. Vous avez tous été les raisons qui ont rendu ces quelques années passées au sein du laboratoire aussi agréables pour moi.

## Aux non-permanents, "*les jeunes*"

Vous êtes sans aucun doute ce qui m'a permis de tenir jusqu'au bout de ma thèse, et rien n'aurait été pareil sans vous, sans votre bonne humeur, sans votre soutien et sans ces pauses thé et ces soirées à discuter autour d'une petite bière et d'une énième partie de jeu de société. Merci à ceux qui étaient déjà là pour m'accueillir quand je suis arrivé, merci à celles avec qui j'ai eu la chance de démarrer ma thèse et merci à tous les "petits" nouveaux. Citer tout le monde et toutes les raisons dans ce manuscrit serait trop long, mais vous êtes tous un peu la raison pour laquelle j'ai été heureux de passer ces trois ans au labo.

Agathe, chère Reine (actuelle) des doctorants, merci pour toute ton aide pour l'organisation et les longues discussions qu'on a eu lors de ce voyage en Écosse, ainsi que pour ton soutien constant depuis qu'on s'est rencontré durant mon stage de Master ! Et merci pour ces souvenirs géniaux de nos (tentatives) d'enseignement des ACP aux étudiants !

Arthur, je ferais bref parce que c'est toi qui dans quelques années aura à écrire tant de choses, mais je tenais à te remercier pour ton amitié et ta présence qui ont été un grand soutien dès ton arrivée et jusqu'à la fin de ma thèse. Bon courage pour ta thèse cher "petit stagiaire" ! T'es quelqu'un de génial, continue tu es sur la bonne voie !

Achille, merci pour ton soutien, et merci d'être une des rares personnes qui me comprend quand je parle de la douceur de la vie (et le piquant des plats !) sur les îles où on peut profiter du Soleil ! Merci pour ces petites bières où on discutait de la vie, en espérant qu'il y en aura encore quelques-unes à l'avenir.

Flavia, merci infiniment pour tout. Merci d'avoir été ma petite lueur aussi bien dans les moments où tout allait bien que dans les périodes les plus difficiles. Tu es un concentré de bonne humeur, de joie et de folie. Merci de m'avoir aidé à persévérer, de m'avoir autant soutenu et d'avoir autant partagé avec moi (et de m'avoir traumatisé dès mon troisième jour dans le 206 !). Ma thèse n'aurait pas pu avoir la même saveur sans toi.

Pour finir, j'ai une pensée toute particulière pour ce fameux bureau 206, qui a été une des choses les plus importantes pour moi pendant cette thèse. Moi qui suis d'un naturel timide et réservé, vous avez su me faire me sentir chez moi, et à ma place dans un petit noyau qu'a été le 206 pendant quelques années. Claire, Flavia, François, merci du fond du cœur pour tous ces souvenirs et ces moments partagés ensemble.

# À mes amis

Alex, merci d'être là pour moi depuis toutes ces années. C'est en partie "à cause" de toi que je suis venu m'installer à Lille à la base, et tu n'as cessé d'être une de ces rares constantes dans ma vie depuis qu'on se connaît. Merci pour tout, pour ces longues soirées "sirop à la menthe" à la maison, pour ce week-end 'barre de cake', pour les longs débats sur Bleach, les soirées JdR, bref… Merci de m'avoir soutenu au travers de toutes ces épreuves, depuis tout ce temps.

Laurent, Justine, merci d'être mes "presque Parisiens" préférés. On n'a pas l'occasion de se voir aussi souvent qu'on le voudrait, mais les occasions que l'on passe ensemble sont toujours des moments qui me font beaucoup de bien. Merci d'avoir toujours été présent dans les moments difficiles au cours de ces dernières années et, je l'espère, pour les années à venir (et une pensée particulière pour ces… 27 ans depuis qu'on se connaît, Laurent ?).

Louis, merci de t'être dit que c'était une bonne idée de me demander si j'avais envie de faire du baseball. Merci de t'être dit que c'était une bonne idée de venir me parler de Vocaloid et d'anime. Merci d'être qui tu es, et d'être devenu mon ami. Désolé d'autant me confier à toi quand j'ai des moments de mou, mais merci du fond du cœur de toujours répondre présent.

Guillaume, merci d'avoir été là quand je suis venu m'installer sur Lille, pour ta présence pour "veiller sur mini-Larx" qui venait d'arriver, de m'avoir mis une bastos de sniper dans le plus grand des calmes au milieu d'une conversation, et de m'avoir aidé à y voir plus clair dans les moments difficiles ces dernières années (et pour les diagnostiques "Tu survivras." !). Quand je passerais vous voir maintenant, on pourra se dire "Bonjour Dr ? Bonjour Dr.".

Et une petite pensée également pour vous, Anne et Pierre, qui êtes encore pour l'instant loin des yeux, mais qui restez toujours près de mon cœur. Merci d'avoir été là depuis toutes ces années, et d'avoir été présents (et si doux !) chaque fois que je rentrais à la Réunion pour passer du temps avec moi.

# À ma famille

Finalement, ma plus grande reconnaissance va à mes parents, et à mon frère.

Clément, tu as été présent dans ma vie depuis toujours. C'est grâce à toi que j'ai pu venir m'installer en Métropole, que j'ai réussi à me ressaisir dans mes études et que je sais que j'aurais toujours quelqu'un qui fera de son mieux pour essayer de m'aider et me soutenir. Je n'ai pas la place de m'étendre ici, donc je vais faire simple, parce que j'espère que c'est suffisant. Merci Clem. Merci pour tout. Peut-être que me mettre à la poubelle dès la naissance nous aurait épargné quelques inquiétudes. Mais ça y est, Clem, je suis docteur.

"Next, it's your turn."

Maman, merci pour tout l'amour que tu me portes depuis même avant ma naissance. Tu as toujours été l'îlot de douceur dans ma vie et tu m'a toujours soutenu inconditionnellement, dans mes études mais aussi dans ma vie. Tu me manques énormément, et saches que tu es et sera toujours ma petite maman. Merci Maman.

Papa, merci pour tout l'amour que tu m'as donné et tous les sacrifices que tu as fait pour moi. Merci pour tes conseils, merci de prendre soin de Ioshi, et merci d'être mon vieux sage dans la forêt. Je suis désolé pour toute l'inquiétude que j'ai pu te causer depuis toutes ces années. J'espère sincèrement que j'ai réussi à te rendre fier Papa.

Je ne serais pas la personne que je suis aujourd'hui sans vous trois.
Merci du plus profond du cœur.

# Supplementary

**Supplementary Table 1** - parameters used for the simulations of pseudo genomic data. General parameters are used for all scenarios, while the parameters under the **CST**, **BTL** or **EXP** sections are used for simulating the corresponding scenario.

| Parameter | Use | 'Formula' |
|---|---|---|
| **General Parameters** | | |
| **N_sampled** | Used to define the population size | Drawn |
| **samp** | Number of sampled haploid genomes | User defined |
| **L** | Genome length (in bp) | User defined |
| **r** | Recombination rate (in recombination per generation per bp) | User defined |
| **nb_SNP** | Number of SNP simulated by the simulator | User defined |
| **Ns_sampled** | Used to define the selection coefficient | Drawn |
| **position** | Position of the beneficial mutation | Drawn between [1, L] |
| **4.N.m** | Migration rate, define as 4.N.m | User defined |
| **scalar_T_split** | Used to define the generation of the split | Drawn |
| **scalar_T_event** | Used to define the generation of demographic/selection events | Drawn |
| **scalar_N** | Used to define the population size in BTL and EXP scenarios | Drawn |
| **CST** | | |
| **$N_{anc}$** | Effective size of the ancestral population | $N_{anc}$ = N_sampled |
| **$N_A$** | Effective size of the population A | $N_A$ = N_sampled |
| **T_split** | Generation of the split | scalar_T_split * $N_A$ |
| **T_dem** | Generation of the demographic change | scalar_T_event * T_split |
| **T_selection** | Generation of the onset of the positive selection | scalar_T_event * T_split |
| **s** | Selection coefficient of the beneficial mutation | Ns_sampled / $N_A$ |
| **BTL** | | |
| **$N_{anc}$** | Effective size of the ancestral population | $N_{anc}$ = N_sampled |
| **$N_A$** | Effective size of the population A | $N_A$ = N_sampled * scalar_N |
| **T_split** | Generation of the split | scalar_T_split * $N_A$ |
| **T_dem** | Generation of the demographic change | scalar_T_dem * $N_A$ |
| **T_selection** | Generation of the onset of the positive selection | scalar_T_selection * $N_A$ |
| **s** | Selection coefficient of the beneficial mutation | Ns_sampled / $N_A$ |

| EXP | | |
|---|---|---|
| **$N_{anc}$** | Effective size of the ancestral population | $N_{anc}$ = N_sampled * scalar_N |
| **$N_A$** | Effective size of the population A | $N_A$ = N_sampled |
| **T_split** | Generation of the split | scalar_T_split * $N_A$ |
| **T_dem** | Generation of the demographic change | scalar_T_dem * $N_A$ |
| **T_selection** | Generation of the onset of the positive selection | scalar_T_selection * $N_A$ |
| **s** | Selection coefficient of the beneficial mutation | Ns_sampled / $N_A$ |

---

**Supplementary Table 2** - summary statistics computed along the genomes, organized in the same order as they are in the *sumStats* matrices. The first seven statistics (in gray) are mainly focused on nucleotidic diversity while the seven other statistics focus on haplotype diversity.

| Summary Statistic | Description / Usage |
|---|---|
| **Nucleotide diversity π (average)** | Measure the polymorphism within the population (pairwise average number of nucleotide differences per site) |
| **Nucleotide diversity π (standard deviation)** | The variability in nucleotide diversity across loci; Measure the polymorphism within the population |
| **Watterson's estimator θ (average)** | Estimates genetic diversity based on mutation rates and the number of segregating sites |
| **Tajima's D** | Neutrality test comparing observed genetic variation to expected values under neutral evolution, identify sequences which do not fit the neutral theory model at equilibrium between mutation and genetic drift |
| **Achaz's Y** | Neutrality tests similar to Tajima's D but immune to sequencing errors (if singletons) - see Achaz, 2008 |
| **Pearson r for π** | Correlation coefficient of the nucleotide diversity π |
| **Pearson's p-value for π** | Probability to find the same result if the correlation coefficient were zero (null hypothesis) |
| **Number of haplotypes** | Total number of distinct haplotypes within the sample, reflecting genetic variation |
| **Haplotype's homozygosity (H1)** | Probability that two random haplotypes are identical, measuring genetic uniformity |
| **Haplotype's homozygosity without the most common haplotype (H2)** | Similar to H1, excluding the most common haplotype, highlighting more rare haplotype diversity |
| **Haplotype's homozygosity considering the two most common haplotype as the same (H12)** | Probability that two random haplotypes are identical, while considering the two most common as the same |
| **H2 over H1** | Ratio reflecting the relative frequency of the most common haplotype |
| **Linkage disequilibrium measured by D** | Quantifies non-random association between alleles at two loci, measured by the deviation of haplotype frequencies from expected values based on gene frequencies. |
| **Linkage disequilibrium measured by r2** | Square of the correlation coefficient between "the presence or absence of a particular allele at a locus" and "the presence or absence of a particular allele at the second locus" |

**Supplementary Table 3** - number of parameters of each CNN architecture used for the classification of demographic scenarios.

| CNN type | model | data type | # of parameters |
|---|---|---|---|
| *simple* | k7 | rawData | 2 799 267 |
| *simple* | k9 | rawData | 2 788 435 |
| *simple* | k7 | sorted | 1 783 283 |
| *simple* | k9 | sorted | 1 270 387 |
| *simple* | k3 | stats | 75 859 |
| *mix* | k7 | rawData | 2 883 803 |
| *mix* | k9 | rawData | 2 872 611 |
| *mix* | k7 | sorted | 1 867 459 |
| *mix* | k9 | sorted | 1 354 563 |
| *pretrained* | efficientNet | rawData | 4 010 815 |
| *pretrained* | efficientNet | sorted | 4 010 815 |
| *pretrained* | efficientNet | stats | 4 010 815 |
| *pretrained* | resNet | rawData | 11 171 779 |
| *pretrained* | resNet | sorted | 11 171 779 |
| *pretrained* | resNet | stats | 11 171 779 |

**Supplementary Table 4** - detailed breakdown of the different CNNs parameters used for the classification task. Accuracy and loss values are obtained on ***neutral* test** datasets.

**CNN type** is the type of architecture (*simple, mix* or *pre-trained*); **model** is either the width of the convolutional kernel used (*simple* and *mix*) or the type of pre-trained architecture used (*efficientNet* or *resNet*); **data type** is the type of data input used to train the CNN (*rawData*, sorted for *sorted-rawData* or stats for *sumStats*); **train on** indicates whether the training was on *sweep* (in blue) or *neutral* (in green) dataset; **data aug** indicates if data augmentation is used during the training; **batch size** is the number of input data used in each batch during the training; **nb workers** is the number of parallel processes launched; **ini learning rate** is the initial value of the learning rate at the beginning of the training; **weight decay** is the value $\lambda$ used to prevent an overfitting by reducing some of the bigger weights; **scheduler steps** is the number of epochs before a modification of the learning rate; **scheduler gamma** is a value used to progressively educe the learning rate as the training progress; **epochs** is the number of epochs of the training; **test acc** is the accuracy obtained on the *test* dataset; **test loss** is the accuracy obtained on the *test* dataset; **test mig acc** is the accuracy obtained on the *mig-test* dataset; **test mig loss** is the loss obtained on the *mig-test* dataset.

| CNN type | model | data type | train on | data aug | batch size | nb workers | init learning rate | weight decay | scheduler steps | scheduler gamma | epochs | test acc | test loss | test mig acc | test mig loss |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *mix* | k7 | rawData | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9627 | 0.3207 | 0.6747 | 1.1947 |
| *mix* | k7 | rawData | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9317 | 0.2505 | 0.8247 | 0.5215 |
| *mix* | k9 | rawData | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.959 | 0.1932 | 0.7607 | 0.7918 |
| *mix* | k9 | rawData | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9183 | 0.3769 | 0.8043 | 0.7999 |
| *mix* | k7 | sorted | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.975 | 0.1063 | 0.7453 | 1.0697 |
| *mix* | k7 | sorted | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9743 | 0.1078 | 0.7493 | 0.9381 |
| *mix* | k9 | sorted | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.947 | 0.1834 | 0.823 | 0.6266 |
| *mix* | k9 | sorted | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9703 | 0.1187 | 0.7703 | 0.8872 |
| *pretrained* | efficientNet | rawData | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.953 | 0.1294 | 0.813 | 0.9552 |
| *pretrained* | efficientNet | rawData | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9477 | 0.1607 | 0.817 | 0.8455 |
| *pretrained* | efficientNet | sorted | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9567 | 0.1466 | 0.7963 | 1.169 |
| *pretrained* | efficientNet | sorted | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.957 | 0.115 | 0.7963 | 1.1429 |
| *pretrained* | efficientNet | stats | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9537 | 0.1658 | 0.8393 | 0.4614 |
| *pretrained* | efficientNet | stats | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9377 | 0.2167 | 0.841 | 0.4466 |
| *pretrained* | resNet | rawData | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9357 | 0.201 | 0.785 | 0.6907 |
| *pretrained* | resNet | rawData | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.928 | 0.2307 | 0.7923 | 0.6379 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *pretrained* | resNet | sorted | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9247 | 0.2552 | 0.7887 | 0.7012 |
| *pretrained* | resNet | sorted | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9487 | 0.187 | 0.8017 | 0.6307 |
| *pretrained* | resNet | stats | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.95 | 0.2158 | 0.836 | 0.4559 |
| *pretrained* | resNet | stats | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.942 | 0.1978 | 0.839 | 0.4791 |
| *simple* | k7 | rawData | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.898 | 0.3549 | 0.7777 | 0.6345 |
| *simple* | k7 | rawData | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.7783 | 0.513 | 0.5163 | 1.5994 |
| *simple* | k9 | rawData | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9663 | 0.1222 | 0.814 | 0.8381 |
| *simple* | k9 | rawData | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9057 | 0.3899 | 0.784 | 0.8382 |
| *simple* | k7 | sorted | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9777 | 0.0735 | 0.7363 | 1.2785 |
| *simple* | k9 | sorted | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9727 | 0.097 | 0.7417 | 1.0098 |
| *simple* | k3 | stats | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.875 | 0.3745 | 0.8477 | 0.457 |
| *simple* | k3 | stats | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9477 | 0.2897 | 0.8693 | 0.4242 |
| *mix* | k7 | rawData | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.649 | 1.0052 | 0.7383 | 0.7243 |
| *mix* | k7 | rawData | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9447 | 0.1548 | 0.6713 | 1.8246 |
| *mix* | k9 | rawData | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.4677 | 1.4135 | 0.496 | 1.3064 |
| *mix* | k9 | rawData | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9223 | 0.2498 | 0.6523 | 1.327 |
| *mix* | k7 | sorted | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.909 | 0.2464 | 0.8117 | 0.4057 |
| *mix* | k7 | sorted | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.663 | 0.6811 | 0.4477 | 2.3693 |
| *mix* | k9 | sorted | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.954 | 0.1507 | 0.7133 | 1.3242 |
| *mix* | k9 | sorted | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.7837 | 0.6441 | 0.5213 | 3.3728 |
| *pretrained* | efficientNet | rawData | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9893 | 0.0358 | 0.7597 | 1.113 |
| *pretrained* | efficientNet | rawData | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.988 | 0.04 | 0.7787 | 1.0353 |
| *pretrained* | efficientNet | sorted | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9893 | 0.0299 | 0.7797 | 1.1952 |
| *pretrained* | efficientNet | sorted | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9863 | 0.0444 | 0.774 | 1.2269 |
| *pretrained* | efficientNet | stats | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9703 | 0.1048 | 0.8823 | 0.3386 |
| *pretrained* | efficientNet | stats | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.978 | 0.0836 | 0.887 | 0.3537 |
| *pretrained* | resNet | rawData | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9753 | 0.082 | 0.8157 | 0.7109 |
| *pretrained* | resNet | rawData | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9857 | 0.0668 | 0.7683 | 0.8682 |
| *pretrained* | resNet | sorted | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9823 | 0.0709 | 0.7873 | 0.6879 |
| *pretrained* | resNet | sorted | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.983 | 0.061 | 0.785 | 0.8415 |
| *pretrained* | resNet | stats | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.971 | 0.1191 | 0.8813 | 0.3543 |
| *pretrained* | resNet | stats | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9633 | 0.1248 | 0.835 | 0.4325 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *simple* | k7 | rawData | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.899 | 0.2387 | 0.6273 | 2.1858 |
| *simple* | k7 | rawData | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.801 | 0.4251 | 0.589 | 1.4878 |
| *simple* | k9 | rawData | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.7783 | 0.5088 | 0.559 | 2.2713 |
| *simple* | k9 | rawData | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.8017 | 0.4045 | 0.578 | 1.9388 |
| *simple* | k7 | sorted | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.8147 | 0.4739 | 0.572 | 2.9566 |
| *simple* | k9 | sorted | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.734 | 0.7379 | 0.5107 | 3.7589 |
| *simple* | k3 | stats | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.8833 | 0.3037 | 0.6637 | 0.8692 |
| *simple* | k3 | stats | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9657 | 0.1422 | 0.8857 | 0.3168 |

---

**Supplementary Table 5** - detailed breakdown of the different CNNs parameters used for the classification task. Accuracy and loss values are obtained on ***sweep* test** datasets.

**CNN type** is the type of architecture (*simple, mix* or *pre-trained*); **model** is either the width of the convolutional kernel used (*simple* and *mix*) or the type of pre-trained architecture used (*efficientNet* or *resNet*); **data type** is the type of data input used to train the CNN (*rawData*, sorted for *sorted-rawData* or stats for *sumStats*); **train on** indicates whether the training was on *sweep* or *neutral* dataset; **data aug** indicates if data augmentation is used during the training; **batch size** is the number of input data used in each batch during the training; **nb workers** is the number of parallel processes launched; **ini learning rate** is the initial value of the learning rate at the beginning of the training; **weight decay** is the value $\lambda$ used to prevent an overfitting by reducing some of the bigger weights; **scheduler steps** is the number of epochs before a modification of the learning rate; **scheduler gamma** is a value used to progressively reduce the learning rate as the training progress; **epochs** is the number of epochs of the training; **test acc** is the accuracy obtained on the *test* dataset; **test loss** is the accuracy obtained on the *test* dataset; **test mig acc** is the accuracy obtained on the *mig-test* dataset; **test mig loss** is the loss obtained on the *mig-test* dataset.

| CNN type | model | data type | train on | data aug | batch size | nb workers | init learning rate | weight decay | scheduler steps | scheduler gamma | epochs | test acc | test loss | test mig acc | test mig loss |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *mix* | k7 | rawData | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.8343 | 0.4602 | 0.8343 | 0.9688 |
| *mix* | k7 | rawData | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.8123 | 0.4544 | 0.8123 | 0.7387 |
| *mix* | k9 | rawData | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.553 | 1.0758 | 0.5530 | 1.9506 |
| *mix* | k9 | rawData | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.914 | 0.2415 | 0.9140 | 0.5879 |
| *mix* | k7 | sorted | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.8667 | 0.3772 | 0.8666 | 1.3358 |
| *mix* | k7 | sorted | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.8943 | 0.2869 | 0.8943 | 1.1805 |
| *mix* | k9 | sorted | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.894 | 0.3057 | 0.8940 | 1.1528 |
| *mix* | k9 | sorted | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.8907 | 0.2775 | 0.8906 | 1.0419 |
| *pretrained* | efficientNet | rawData | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9733 | 0.0729 | 0.846 | 0.6407 |
| *pretrained* | efficientNet | rawData | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9713 | 0.089 | 0.873 | 0.5403 |
| *pretrained* | efficientNet | sorted | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.976 | 0.0618 | 0.8753 | 0.676 |
| *pretrained* | efficientNet | sorted | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.977 | 0.0747 | 0.8773 | 0.6878 |
| *pretrained* | efficientNet | stats | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.935 | 0.1862 | 0.8847 | 0.3404 |
| *pretrained* | efficientNet | stats | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9263 | 0.1935 | 0.8687 | 0.3841 |
| *pretrained* | resNet | rawData | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9587 | 0.1348 | 0.8903 | 0.3412 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *pretrained* | resNet | rawData | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.952 | 0.1413 | 0.9063 | 0.2749 |
| *pretrained* | resNet | sorted | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9563 | 0.1264 | 0.888 | 0.3476 |
| *pretrained* | resNet | sorted | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.964 | 0.1248 | 0.9063 | 0.2804 |
| *pretrained* | resNet | stats | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.892 | 0.3081 | 0.8477 | 0.4163 |
| *pretrained* | resNet | stats | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9033 | 0.2903 | 0.863 | 0.4187 |
| *simple* | k7 | rawData | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.6193 | 0.9568 | 0.6447 | 0.8472 |
| *simple* | k7 | rawData | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.3333 | 1.7008 | 0.3333 | 2.3553 |
| *simple* | k9 | rawData | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.8933 | 0.2748 | 0.7727 | 0.6473 |
| *simple* | k9 | rawData | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.749 | 0.6159 | 0.5567 | 1.385 |
| *simple* | k7 | sorted | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.942 | 0.1845 | 0.8263 | 0.5235 |
| *simple* | k9 | sorted | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.9427 | 0.1537 | 0.8857 | 0.371 |
| *simple* | k3 | stats | *sweep* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.886 | 0.3392 | 0.8407 | 0.4356 |
| *simple* | k3 | stats | *sweep* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.873 | 0.3799 | 0.8043 | 0.4992 |
| *mix* | k7 | rawData | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.674 | 1.6838 | 0.7757 | 1.11825 |
| *mix* | k7 | rawData | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.5143 | 3.428 | 0.4443 | 5.176 |
| *mix* | k9 | rawData | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.873 | 0.7638 | 0.766 | 1.1998 |
| *mix* | k9 | rawData | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.887 | 0.7983 | 0.7757 | 1.2016 |
| *mix* | k7 | sorted | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.7303 | 1.3871 | 0.582 | 2.3825 |
| *mix* | k7 | sorted | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.7337 | 1.2214 | 0.593 | 2.254 |
| *mix* | k9 | sorted | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.7747 | 1.0467 | 0.6207 | 2.0649 |
| *mix* | k9 | sorted | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.7443 | 1.1806 | 0.5913 | 2.346 |
| *pretrained* | efficientNet | rawData | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.915 | 0.4411 | 0.7997 | 0.998 |
| *pretrained* | efficientNet | rawData | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.911 | 0.5079 | 0.7933 | 1.0061 |
| *pretrained* | efficientNet | sorted | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9193 | 0.5607 | 0.8187 | 1.2256 |
| *pretrained* | efficientNet | sorted | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.927 | 0.3801 | 0.82 | 1.1307 |
| *pretrained* | efficientNet | stats | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.8737 | 0.5589 | 0.829 | 0.6997 |
| *pretrained* | efficientNet | stats | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.8507 | 0.5678 | 0.8307 | 0.6256 |
| *pretrained* | resNet | rawData | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.904 | 0.4424 | 0.8547 | 0.585 |
| *pretrained* | resNet | rawData | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.8947 | 0.4038 | 0.8317 | 0.6344 |
| *pretrained* | resNet | sorted | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.912 | 0.3455 | 0.8543 | 0.5216 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *pretrained* | resNet | sorted | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.9043 | 0.3747 | 0.8453 | 0.5388 |
| *pretrained* | resNet | stats | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.8367 | 0.5433 | 0.797 | 0.615 |
| *pretrained* | resNet | stats | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 5 | 0.1 | 100 | 0.866 | 0.472 | 0.836 | 0.5691 |
| *simple* | k7 | rawData | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.807 | 2.2981 | 0.838 | 1.8419 |
| *simple* | k7 | rawData | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.8517 | 1.5857 | 0.733 | 1.8493 |
| *simple* | k9 | rawData | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.8427 | 0.9281 | 0.7073 | 1.5898 |
| *simple* | k9 | rawData | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.6433 | 1.8901 | 0.5237 | 3.0616 |
| *simple* | k7 | sorted | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.6547 | 1.9256 | 0.5263 | 2.9769 |
| *simple* | k9 | sorted | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.669 | 1.6282 | 0.5377 | 3.1418 |
| *simple* | k3 | stats | *neutral* | Yes | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.848 | 0.7876 | 0.815 | 0.7593 |
| *simple* | k3 | stats | *neutral* | No | 32 | 10 | 0.00001 | 0.005 | 3 | 0.8 | 100 | 0.7923 | 1.0148 | 0.738 | 1.0848 |

**Supplementary Table 6** - comparison of the accuracy of the worst CNNs architecture in case of model misspecification.

Δ accuracy and Δ loss are computed as the difference in accuracy or loss when the model is used to make inferences on a dataset with misspecified *selection* compared to its training dataset. In bold, cases when model misspecification increases the accuracy and/or reduces the loss.

| Train on | Predict on | CNN | Input Data | Model | accuracy | loss | Δ accuracy | Δ loss |
|---|---|---|---|---|---|---|---|---|
| *neutral* | *neutral* | *simple* | *sorted-rawData* | k9 | 0.734 | 0.738 | - | - |
| | | *mix* | *rawDAta* | k9 | 0.468 | 0.141 | - | - |
| | | *pre-trained* | *sumStats* | *resNet* | 0.963 | 0.125 | - | - |
| | *sweep* | ***simple*** | ***rawData*** | ***k7*** | **0.778** | **0.513** | **+ 0.044** | **- 0.225** |
| | | ***mix*** | ***rawData*** | **k9** | **0.918** | **0.377** | **+ 0.45** | **- 0.236** |
| | | *pre-trained* | *sorted-rawData* | *resNet* | 0.925 | 0.255 | - 0.038 | + 0.13 |
| *sweep* | *neutral* | *simple* | *rawData* | k9 | 0.643 | 1.89 | + 0.31 | + 0.19 |
| | | *mix* | *rawData* | k7 | 0.514 | 3.43 | - 0.039 | + 2.35 |
| | | *pre-trained* | *sumStats* | *resNet* | 0.837 | 0.543 | - 0.055 | + 0.235 |
| | *sweep* | *simple* | *rawData* | k7 | 0.333 | 1.7 | - | - |
| | | *mix* | *rawData* | k9 | 0.553 | 1.08 | - | - |
| | | *pre-trained* | *sumStats* | *resNet* | 0.892 | 0.308 | - | - |

**Supplementary Table 7** - detailed breakdown of the different CNNs parameters used for the object detection task.

**CNN Name** is the name of architecture, composed of: the scenario, the type of data used, the number of retrained backbone layers and finally the use or not of data augmentation; **scenario** is the demographic scenario of the simulations used for the CNN training (CST, BTL or EXP); **data type** is the type of data input used to train the CNN (sorted for *sorted-rawData*, stats for *sumStats* or *objDet* for *sumStats* scaled on the variability intra-simulation); **backbone layers** indicates the number of backbone layers that are unfrozen and fine-tuned during the training (4 or 5); **data aug** indicates if data augmentation is used during the training; **batch size** is the number of input data used in each batch during the training; **nb workers** is the number of parallel processes launched; **ini learning rate** is the initial value of the learning rate at the beginning of the training; **weight decay** is the value $\lambda$ used to prevent an overfitting by reducing some of the bigger weights; **scheduler steps** is the number of epochs before a modification of the learning rate; **scheduler gamma** is a value used to progressively reduce the learning rate as the training progress; **epochs** is the number of epochs of the training; **test acc** is the accuracy obtained on the *test* dataset; **test loss** is the accuracy obtained on the *test* dataset; **test mig acc** is the accuracy obtained on the *mig-test* dataset.

| CNN Name | scenario | data type | backbone layers | data aug | batch size | nb workers | init learning rate | weight decay | scheduler steps | scheduler gamma | epochs |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CST_objDet_backbone_4_aug | CST | *objDet* | 4 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| CST_objDet_backbone_4_noaug | CST | *objDet* | 4 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| CST_objDet_backbone_5_aug | CST | *objDet* | 5 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| CST_objDet_backbone_5_noaug | CST | *objDet* | 5 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| CST_sorted_backbone_4_aug | CST | *sorted* | 4 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| CST_sorted_backbone_4_noaug | CST | *sorted* | 4 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| CST_sorted_backbone_5_aug | CST | *sorted* | 5 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| CST_sorted_backbone_5_noaug | CST | *sorted* | 5 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| CST_stats_backbone_4_noaug | CST | *stats* | 4 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| CST_stats_backbone_5_noaug | CST | *stats* | 5 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| BTL_objDet_backbone_4_aug | BTL | *objDet* | 4 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| BTL_objDet_backbone_4_noaug | BTL | *objDet* | 4 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| BTL_objDet_backbone_5_aug | BTL | *objDet* | 5 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| BTL_objDet_backbone_5_noaug | BTL | *objDet* | 5 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| BTL_sorted_backbone_4_aug | BTL | *sorted* | 4 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| BTL_sorted_backbone_4_noaug | BTL | *sorted* | 4 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BTL_sorted_backbone_5_aug | BTL | *sorted* | 5 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| BTL_sorted_backbone_5_noaug | BTL | *sorted* | 5 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| BTL_stats_backbone_4_aug | BTL | *stats* | 4 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| BTL_stats_backbone_4_noaug | BTL | *stats* | 4 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| BTL_stats_backbone_5_noaug | BTL | *stats* | 5 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| EXP_objDet_backbone_4_aug | EXP | *objDet* | 4 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| EXP_objDet_backbone_4_noaug | EXP | *objDet* | 4 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| EXP_objDet_backbone_5_aug | EXP | *objDet* | 5 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| EXP_objDet_backbone_5_noaug | EXP | *objDet* | 5 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| EXP_sorted_backbone_4_aug | EXP | *sorted* | 4 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| EXP_sorted_backbone_4_noaug | EXP | *sorted* | 4 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| EXP_sorted_backbone_5_aug | EXP | *sorted* | 5 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| EXP_sorted_backbone_5_noaug | EXP | *sorted* | 5 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| EXP_stats_backbone_4_aug | EXP | *stats* | 4 | Yes | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| EXP_stats_backbone_4_noaug | EXP | *stats* | 4 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |
| EXP_stats_backbone_5_noaug | EXP | *stats* | 5 | No | 8 | 2 | 0.0001 | 0.001 | 3 | 0.9 | 150 |

**Supplementary Table 8** - detailed breakdown of the different CNNs parameters used for the object detection task.

**CNN Name** is the name of architecture, composed of: the scenario, the type of data used, the number of retrained backbone layers and finally the use or not of data augmentation; **map50** is the mean average precision (mAP) when the Intersection over Union (IoU) threshold is set at 0.5; **map75** is similar to map50 but with a higher IoU threshold of 0.75; **mar** is the mean average recall (mAR), computing the average recall over multiple IoU thresholds (between 0.5 to 0.95 in increments of 0.05).

| CNN Name | BTL | | | CST | | | EXP | | | MGB | | | MIG | | | MGX | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | map50 | map75 | mar | map50 | map75 | mar | map50 | map75 | mar | map50 | map75 | mar | map50 | map75 | mar | map50 | map75 | mar |
| BTL_objDet_backbone_4_aug | 0,6459 | 0,6067 | 0,6189 | 0,6591 | 0,5551 | 0,5910 | 0,2084 | 0,1655 | 0,2119 | 0,6769 | 0,6361 | 0,6426 | 0,6804 | 0,5475 | 0,5933 | 0,2105 | 0,1535 | 0,2305 |
| BTL_objDet_backbone_4_noaug | 0,6445 | 0,5963 | 0,6159 | 0,6564 | 0,5616 | 0,5921 | 0,2131 | 0,1587 | 0,2101 | 0,6733 | 0,6362 | 0,6438 | 0,6840 | 0,5665 | 0,6018 | 0,2147 | 0,1428 | 0,2232 |
| BTL_objDet_backbone_5_aug | 0,6459 | 0,5939 | 0,6172 | 0,6470 | 0,5411 | 0,5811 | 0,2004 | 0,1575 | 0,2103 | 0,6862 | 0,6403 | 0,6465 | 0,6698 | 0,5579 | 0,5845 | 0,2074 | 0,1519 | 0,2163 |
| BTL_objDet_backbone_5_noaug | 0,6470 | 0,6067 | 0,6182 | 0,6566 | 0,5598 | 0,5905 | 0,1976 | 0,1521 | 0,1969 | 0,6754 | 0,6284 | 0,6401 | 0,6843 | 0,5645 | 0,5998 | 0,2153 | 0,1647 | 0,2312 |
| BTL_sorted_backbone_4_aug | 0,5739 | 0,4664 | 0,5045 | 0,2818 | 0,1139 | 0,2052 | 0,0167 | 0,0112 | 0,0323 | 0,5986 | 0,4851 | 0,5199 | 0,3499 | 0,1540 | 0,2376 | 0,0213 | 0,0094 | 0,0540 |
| BTL_sorted_backbone_4_noaug | 0,5698 | 0,4672 | 0,5036 | 0,2679 | 0,1063 | 0,2014 | 0,0152 | 0,0113 | 0,0306 | 0,5918 | 0,4906 | 0,5207 | 0,3218 | 0,1417 | 0,2273 | 0,0240 | 0,0126 | 0,0533 |
| BTL_sorted_backbone_5_aug | 0,5658 | 0,4633 | 0,5009 | 0,2724 | 0,1137 | 0,2050 | 0,0158 | 0,0115 | 0,0323 | 0,5744 | 0,4837 | 0,5082 | 0,3318 | 0,1400 | 0,2330 | 0,0246 | 0,0127 | 0,0551 |
| BTL_sorted_backbone_5_noaug | 0,5713 | 0,4693 | 0,5113 | 0,2983 | 0,1169 | 0,2203 | 0,0170 | 0,0117 | 0,0335 | 0,5876 | 0,4941 | 0,5214 | 0,3590 | 0,1628 | 0,2457 | 0,0227 | 0,0097 | 0,0564 |
| BTL_stats_backbone_4_aug | 0,6396 | 0,6003 | 0,6309 | 0,5588 | 0,3146 | 0,4746 | 0,0228 | 0,0098 | 0,0638 | 0,6718 | 0,6500 | 0,6629 | 0,6052 | 0,3442 | 0,4876 | 0,0363 | 0,0147 | 0,0915 |
| BTL_stats_backbone_4_noaug | 0,6482 | 0,6027 | 0,6343 | 0,5589 | 0,3433 | 0,4784 | 0,0301 | 0,0146 | 0,0691 | 0,6728 | 0,6486 | 0,6589 | 0,6116 | 0,3661 | 0,4989 | 0,0481 | 0,0214 | 0,0955 |
| BTL_stats_backbone_5_noaug | 0,6388 | 0,6006 | 0,6334 | 0,5398 | 0,3170 | 0,4655 | 0,0270 | 0,0161 | 0,0651 | 0,6711 | 0,6479 | 0,6645 | 0,6089 | 0,3629 | 0,4894 | 0,0403 | 0,0102 | 0,0899 |
| CST_objDet_backbone_4_aug | 0,5163 | 0,3798 | 0,4758 | 0,7164 | 0,6456 | 0,6628 | 0,2192 | 0,1595 | 0,2210 | 0,5551 | 0,4198 | 0,5128 | 0,7438 | 0,6537 | 0,6755 | 0,2147 | 0,1473 | 0,2383 |
| CST_objDet_backbone_4_noaug | 0,5071 | 0,3646 | 0,4643 | 0,7185 | 0,6516 | 0,6715 | 0,2239 | 0,1609 | 0,2228 | 0,5670 | 0,4030 | 0,4976 | 0,7441 | 0,6581 | 0,6798 | 0,2198 | 0,1541 | 0,2397 |
| CST_objDet_backbone_5_aug | 0,5219 | 0,3921 | 0,4870 | 0,7153 | 0,6559 | 0,6683 | 0,2192 | 0,1572 | 0,2183 | 0,5790 | 0,4310 | 0,5182 | 0,7435 | 0,6656 | 0,6824 | 0,2187 | 0,1610 | 0,2352 |
| CST_objDet_backbone_5_noaug | 0,5331 | 0,3924 | 0,4912 | 0,7160 | 0,6496 | 0,6681 | 0,2185 | 0,1603 | 0,2166 | 0,5664 | 0,4179 | 0,5129 | 0,7428 | 0,6529 | 0,6820 | 0,2201 | 0,1648 | 0,2392 |
| CST_sorted_backbone_4_aug | 0,4277 | 0,2786 | 0,3567 | 0,6005 | 0,4195 | 0,4761 | 0,0720 | 0,0351 | 0,1002 | 0,4735 | 0,3227 | 0,3880 | 0,6219 | 0,3915 | 0,4826 | 0,0611 | 0,0306 | 0,1026 |
| CST_sorted_backbone_4_noaug | 0,4470 | 0,2872 | 0,3567 | 0,5997 | 0,4258 | 0,4783 | 0,0867 | 0,0395 | 0,1036 | 0,4899 | 0,3242 | 0,3902 | 0,6196 | 0,4013 | 0,4781 | 0,0727 | 0,0380 | 0,1092 |
| CST_sorted_backbone_5_aug | 0,3990 | 0,2749 | 0,3408 | 0,6106 | 0,4237 | 0,4829 | 0,0617 | 0,0298 | 0,0929 | 0,4582 | 0,3210 | 0,3780 | 0,6091 | 0,3973 | 0,4756 | 0,0604 | 0,0257 | 0,1028 |
| CST_sorted_backbone_5_noaug | 0,3932 | 0,2848 | 0,3414 | 0,6071 | 0,4180 | 0,4799 | 0,0630 | 0,0317 | 0,0858 | 0,4473 | 0,3271 | 0,3778 | 0,6162 | 0,4024 | 0,4776 | 0,0572 | 0,0343 | 0,1031 |
| CST_stats_backbone_4_noaug | 0,5386 | 0,4466 | 0,5068 | 0,7160 | 0,6727 | 0,6951 | 0,1377 | 0,0682 | 0,1511 | 0,5686 | 0,4646 | 0,5359 | 0,7407 | 0,6632 | 0,6889 | 0,1582 | 0,0893 | 0,1769 |
| CST_stats_backbone_5_noaug | 0,5248 | 0,4227 | 0,5037 | 0,7357 | 0,6856 | 0,7042 | 0,1351 | 0,0719 | 0,1493 | 0,5749 | 0,4653 | 0,5363 | 0,7429 | 0,6656 | 0,6953 | 0,1543 | 0,0876 | 0,1790 |
| EXP_objDet_backbone_4_aug | 0,5250 | 0,3567 | 0,4612 | 0,6959 | 0,5671 | 0,6255 | 0,2335 | 0,1855 | 0,2414 | 0,5533 | 0,3627 | 0,4752 | 0,7060 | 0,5376 | 0,6165 | 0,2161 | 0,1420 | 0,2191 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXP_objDet_backbone_4_noaug | 0,5347 | 0,3467 | 0,4497 | 0,6915 | 0,5886 | 0,6313 | 0,2298 | 0,1829 | 0,2377 | 0,5684 | 0,3535 | 0,4626 | 0,7105 | 0,5533 | 0,6183 | 0,2199 | 0,1440 | 0,2184 |
| EXP_objDet_backbone_5_aug | 0,5412 | 0,3860 | 0,4703 | 0,6996 | 0,5818 | 0,6345 | 0,2383 | 0,1937 | 0,2426 | 0,5877 | 0,3889 | 0,4938 | 0,6885 | 0,5647 | 0,6171 | 0,2160 | 0,1429 | 0,2245 |
| EXP_objDet_backbone_5_noaug | 0,5455 | 0,3823 | 0,4636 | 0,6837 | 0,5745 | 0,6298 | 0,2349 | 0,1933 | 0,2384 | 0,5780 | 0,3801 | 0,4805 | 0,6980 | 0,5506 | 0,6210 | 0,2201 | 0,1477 | 0,2269 |
| EXP_sorted_backbone_4_aug | 0,2940 | 0,1670 | 0,2525 | 0,5229 | 0,3006 | 0,3947 | 0,1724 | 0,1041 | 0,1600 | 0,3326 | 0,1868 | 0,2854 | 0,5274 | 0,3231 | 0,4028 | 0,1425 | 0,0925 | 0,1415 |
| EXP_sorted_backbone_4_noaug | 0,3135 | 0,1727 | 0,2618 | 0,5273 | 0,2938 | 0,3908 | 0,1690 | 0,1081 | 0,1631 | 0,3507 | 0,1983 | 0,2918 | 0,5225 | 0,3079 | 0,4018 | 0,1444 | 0,0879 | 0,1423 |
| EXP_sorted_backbone_5_aug | 0,2886 | 0,1602 | 0,2504 | 0,5098 | 0,3017 | 0,3923 | 0,1727 | 0,1060 | 0,1585 | 0,3238 | 0,1874 | 0,2841 | 0,5233 | 0,3136 | 0,4007 | 0,1406 | 0,0867 | 0,1432 |
| EXP_sorted_backbone_5_noaug | 0,2799 | 0,1576 | 0,2568 | 0,5223 | 0,2963 | 0,3899 | 0,1752 | 0,1054 | 0,1582 | 0,3162 | 0,1837 | 0,2856 | 0,5173 | 0,3102 | 0,3969 | 0,1476 | 0,0942 | 0,1444 |
| EXP_stats_backbone_4_aug | 0,4592 | 0,3597 | 0,4044 | 0,6931 | 0,5545 | 0,5823 | 0,2231 | 0,1727 | 0,2239 | 0,4802 | 0,3731 | 0,4253 | 0,7118 | 0,5722 | 0,5969 | 0,2401 | 0,1733 | 0,2313 |
| EXP_stats_backbone_4_noaug | 0,4832 | 0,3795 | 0,4289 | 0,7043 | 0,5559 | 0,5885 | 0,2286 | 0,1829 | 0,2254 | 0,5222 | 0,4168 | 0,4580 | 0,7034 | 0,5460 | 0,5946 | 0,2380 | 0,1768 | 0,2273 |
| EXP_stats_backbone_5_noaug | 0,4621 | 0,3677 | 0,4102 | 0,6902 | 0,5348 | 0,5810 | 0,2330 | 0,1728 | 0,2228 | 0,4744 | 0,3705 | 0,4357 | 0,7006 | 0,5278 | 0,5814 | 0,2404 | 0,1711 | 0,2306 |

**Supplementary Table 9** - results of Wilcoxon test for significant difference between *neutral* simulations parameters with or without FP.

In **bold**, parameters where a significant difference is found (p-value < 0.05). In ***bold italic***, parameters where 0.01 < p-value < 0.05. ***N_current*** is the population's current effective size; ***N_ancestral*** is the ancestral population's effective size; ***T_split*** is the age of the ancestral population's split; ***T_demography*** is the age of the demographic change.

**A) CNN FP predictions** on *neutral* dataset

| Scenario | Variable | p-value | Significance (5%) | Significance (1%) |
|---|---|---|---|---|
| BTL | N_current | 4,77E-01 | No | No |
| | **N_ancestral** | **8,99E-04** | **Yes** | **Yes** |
| | **T_split** | **3,34E-14** | **Yes** | **Yes** |
| | **T_demography** | **4,64E-26** | **Yes** | **Yes** |
| MGB | N_current | 5,00E-02 | No | No |
| | **N_ancestral** | **1,05E-03** | **Yes** | **Yes** |
| | **T_split** | **2,96E-15** | **Yes** | **Yes** |
| | **T_demography** | **8,95E-20** | **Yes** | **Yes** |
| CST | **N_current** | **1,96E-25** | **Yes** | **Yes** |
| | **N_ancestral** | **1,96E-25** | **Yes** | **Yes** |
| | T_split | 5,80E-02 | No | No |
| | T_demography | 2,14E-01 | No | No |
| MIG | **N_current** | **1,40E-11** | **Yes** | **Yes** |
| | **N_ancestral** | **1,40E-11** | **Yes** | **Yes** |
| | **T_split** | **1,66E-09** | **Yes** | **Yes** |
| | **T_demography** | **4,50E-04** | **Yes** | **Yes** |
| EXP | **N_current** | **2,72E-09** | **Yes** | **Yes** |
| | N_ancestral | 1,88E-01 | No | No |
| | T_split | 5,15E-02 | No | No |
| | **T_demography** | **1,03E-04** | **Yes** | **Yes** |
| MGX | N_current | 8,09E-01 | No | No |
| | N_ancestral | 7,28E-01 | No | No |
| | **T_split** | **6,82E-08** | **Yes** | **Yes** |
| | T_demography | 4,11E-01 | No | No |

**B) SF2 FP predictions** on *neutral* dataset

| Scenario | Variable | p-value | Significance (5%) | Significance (1%) |
|---|---|---|---|---|
| BTL | N_current | **5,58E-13** | **Yes** | **Yes** |
| | **N_ancestral** | **1,94E-06** | **Yes** | **Yes** |
| | **T_split** | **4,40E-10** | **Yes** | **Yes** |
| | **T_demography** | **5,86E-06** | **Yes** | **Yes** |
| MGB | N_current | **2,98E-16** | **Yes** | **Yes** |
| | **N_ancestral** | **5,68E-07** | **Yes** | **Yes** |
| | **T_split** | **4,15E-06** | **Yes** | **Yes** |
| | **T_demography** | **6,96E-06** | **Yes** | **Yes** |
| CST | ***N_current*** | ***3,30E-02*** | ***Yes*** | No |
| | ***N_ancestral*** | ***3,30E-02*** | ***Yes*** | No |
| | T_split | **5,90E-07** | **Yes** | **Yes** |
| | T_demography | **6,70E-05** | **Yes** | **Yes** |
| MIG | **N_current** | **2,84E-03** | **Yes** | **Yes** |
| | **N_ancestral** | **2,84E-03** | **Yes** | **Yes** |
| | **T_split** | **7,91E-26** | **Yes** | **Yes** |
| | **T_demography** | **1,15E-16** | **Yes** | **Yes** |
| EXP | **N_current** | 8,86E-01 | No | No |
| | N_ancestral | 8,18E-01 | No | No |
| | T_split | 4,43E-01 | No | No |
| | **T_demography** | 2,17E-01 | No | No |
| MGX | N_current | **2,34E-03** | **Yes** | **Yes** |
| | N_ancestral | 2,26E-01 | No | No |
| | **T_split** | **2,09E-04** | **Yes** | **Yes** |
| | T_demography | 7,63E-01 | No | No |

**C) FP predictions made by both models** on *neutral* dataset

| Scenario | Variable | p-value | Significance (5%) | Significance (1%) |
|---|---|---|---|---|
| BTL | **N_current** | **6,32E-05** | **Yes** | **Yes** |
| | **N_ancestral** | **1,38E-06** | **Yes** | **Yes** |
| | **T_split** | **9,51E-18** | **Yes** | **Yes** |
| | **T_demography** | **6,57E-18** | **Yes** | **Yes** |
| MGB | N_current | **1,16E-08** | **Yes** | **Yes** |
| | **N_ancestral** | **7,98E-07** | **Yes** | **Yes** |
| | **T_split** | **4,30E-15** | **Yes** | **Yes** |
| | **T_demography** | **8,50E-16** | **Yes** | **Yes** |
| CST | **N_current** | **1,88E-09** | **Yes** | **Yes** |
| | **N_ancestral** | **1,88E-09** | **Yes** | **Yes** |
| | **T_split** | **1,89E-05** | **Yes** | **Yes** |
| | **T_demography** | **1,23E-03** | **Yes** | **Yes** |
| MIG | **N_current** | **3,49E-04** | **Yes** | **Yes** |
| | **N_ancestral** | **3,49E-04** | **Yes** | **Yes** |
| | **T_split** | **1,73E-23** | **Yes** | **Yes** |
| | **T_demography** | **9,33E-12** | **Yes** | **Yes** |
| EXP | **N_current** | **1,33E-04** | **Yes** | **Yes** |
| | N_ancestral | 4,97E-01 | No | No |
| | T_split | 4,03E-01 | No | No |
| | **T_demography** | **7,32E-04** | **Yes** | **Yes** |
| MGX | N_current | 1,46E-01 | No | No |
| | N_ancestral | 5,14E-01 | No | No |
| | **T_split** | **7,79E-06** | **Yes** | **Yes** |
| | T_demography | 5,48E-01 | No | No |

**Supplementary Table 10** - results of Wilcoxon test for significant difference between *sweep* simulations parameters where a selective sweep is predicted or not.

In **bold**, parameters where a significant difference is found (p-value < 0.05). In ***bold italic***, parameters where 0.01 < p-value < 0.05. (A) Predictions by the CNN, (B) prediction by SF2 and (C) predictions of a selective sweep on the same simulation by both models. ***N_current*** is the population's current effective size; ***N_ancestral*** is the ancestral population's effective size; ***T_split*** is the age of the ancestral population's split; ***T_demography*** is the age of the demographic change; ***T_selection*** is the age of the onset of the selection event; ***T_freq99*** is used as a proxy of the age of the fixation of the sweep; ***position_selected_allele*** is the position of the beneficial mutation on the chromosome; ***selective_coefficient_s*** is the selective coefficient of the beneficial mutation; ***tb_width*** is the width of the bounding box of the selective sweep; ***sweep_duration*** is the duration of the selective sweep.

**A) CNN predictions** on *sweep* dataset

| Scenario | Variable | p-value | Significance (5%) | Significance (1%) |
|---|---|---|---|---|
| BTL | N_current | 6,20E-01 | No | No |
| | N_ancestral | 7,00E-02 | No | No |
| | T_split | 7,19E-01 | No | No |
| | **T_demography** | **2,09E-03** | **Yes** | **Yes** |
| | T_selection | 3,27E-01 | No | No |
| | T_freq99 | 2,05E-01 | No | No |
| | position_selected_allele | 5,88E-01 | No | No |
| | selective_coefficient_s | 3,77E-01 | No | No |
| | **tb_width** | **3,25E-05** | **Yes** | **Yes** |
| | **sweep_duration** | **1,71E-03** | **Yes** | **Yes** |
| MGB | N_current | 8,30E-02 | No | No |
| | N_ancestral | 2,60E-01 | No | No |
| | T_split | 2,81E-01 | No | No |
| | T_demography | 8,73E-02 | No | No |
| | T_selection | 2,41E-01 | No | No |
| | T_freq99 | 1,46E-01 | No | No |

| | | | | |
|---|---|---|---|---|
| | position_selected_allele | 4,82E-01 | No | No |
| | selective_coefficient_s | 5,41E-01 | No | No |
| | **tb_width** | **2,98E-02** | **Yes** | No |
| | **sweep_duration** | **5,82E-03** | **Yes** | **Yes** |
| CST | **N_current** | **2,16E-06** | **Yes** | **Yes** |
| | **N_ancestral** | **2,16E-06** | **Yes** | **Yes** |
| | T_split | 3,10E-01 | No | No |
| | T_demography | 8,48E-02 | No | No |
| | **T_selection** | **7,64E-03** | **Yes** | **Yes** |
| | **T_freq99** | **4,93E-03** | **Yes** | **Yes** |
| | position_selected_allele | 9,92E-01 | No | No |
| | **selective_coefficient_s** | **2,53E-04** | **Yes** | **Yes** |
| | **tb_width** | **2,19E-11** | **Yes** | **Yes** |
| | sweep_duration | 2,31E-01 | No | No |
| MIG | **N_current** | **2,34E-03** | **Yes** | **Yes** |
| | **N_ancestral** | **2,34E-03** | **Yes** | **Yes** |
| | T_split | 2,21E-02 | **Yes** | No |
| | T_demography | 1,14E-01 | No | No |
| | **T_selection** | **1,93E-03** | **Yes** | **Yes** |
| | **T_freq99** | **1,84E-03** | **Yes** | **Yes** |
| | position_selected_allele | 6,87E-01 | No | No |
| | selective_coefficient_s | 4,56E-02 | **Yes** | No |
| | **tb_width** | **3,47E-08** | **Yes** | **Yes** |
| | sweep_duration | 1,33E-01 | No | No |
| EXP | **N_current** | **1,31E-14** | **Yes** | **Yes** |
| | **N_ancestral** | **7,33E-03** | **Yes** | **Yes** |
| | **T_split** | **8,45E-04** | **Yes** | **Yes** |
| | **T_demography** | **1,13E-02** | **Yes** | No |
| | **T_selection** | **2,85E-10** | **Yes** | **Yes** |
| | **T_freq99** | **4,18E-11** | **Yes** | **Yes** |
| | position_selected_allele | 2,72E-01 | No | No |
| | selective_coefficient_s | 9,63E-01 | No | No |
| | **tb_width** | **7,00E-15** | **Yes** | **Yes** |

| | | | | |
|---|---|---|---|---|
| | **sweep_duration** | **1,16E-14** | **Yes** | **Yes** |
| MGX | **N_current** | **2,09E-03** | **Yes** | **Yes** |
| | N_ancestral | 8,18E-01 | No | No |
| | T_split | 8,97E-01 | No | No |
| | T_demography | 7,40E-01 | No | No |
| | **T_selection** | **1,74E-03** | **Yes** | **Yes** |
| | **T_freq99** | **9,89E-04** | **Yes** | **Yes** |
| | position_selected_allele | 8,48E-01 | No | No |
| | selective_coefficient_s | 4,71E-01 | No | No |
| | **tb_width** | **1,71E-06** | **Yes** | **Yes** |
| | **sweep_duration** | **3,50E-04** | **Yes** | **Yes** |

**B) SF2 predictions** on *sweep* dataset

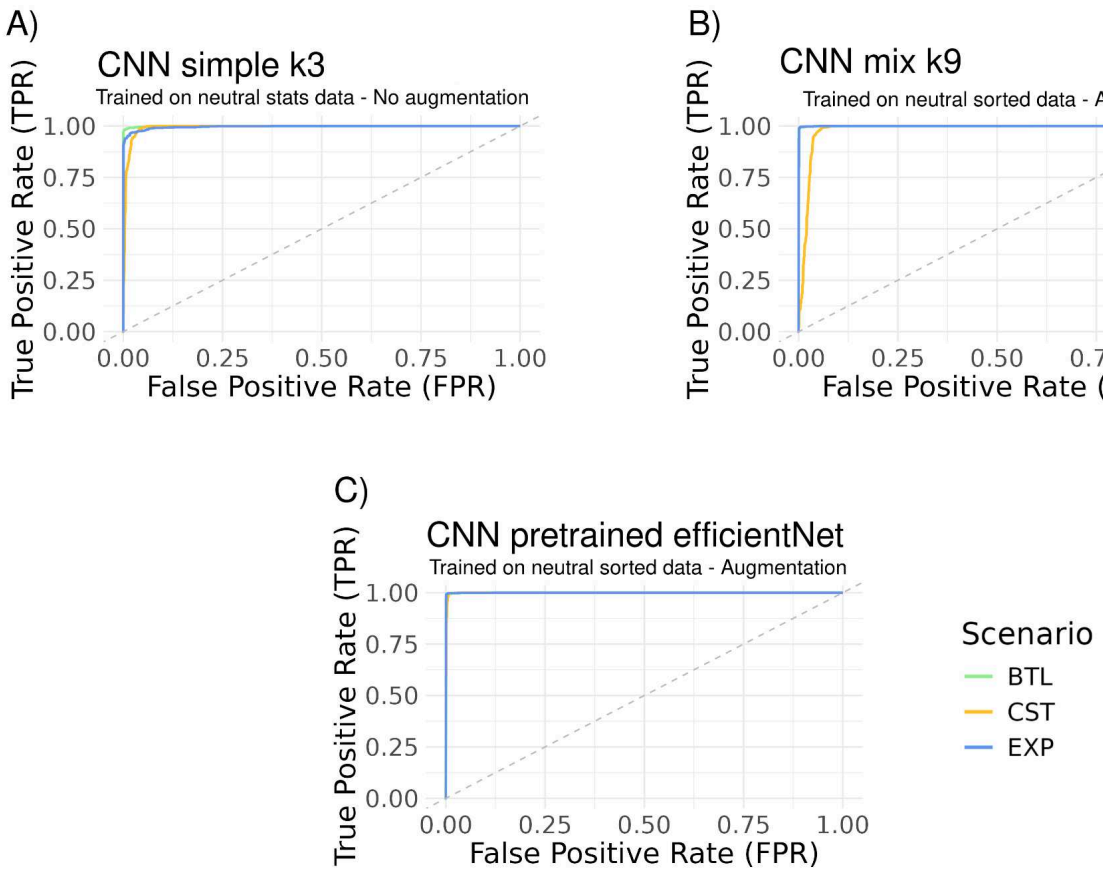
| Scenario | Variable | p-value | Significance (5%) | Significance (1%) |
|---|---|---|---|---|
| BTL | N_current | 6,20E-01 | No | No |
| | N_ancestral | 7,00E-02 | No | No |
| | T_split | 7,19E-01 | No | No |
| | **T_demography** | **2,09E-03** | **Yes** | **Yes** |
| | T_selection | 3,27E-01 | No | No |
| | T_freq99 | 2,05E-01 | No | No |
| | position_selected_allele | 5,88E-01 | No | No |
| | selective_coefficient_s | 3,77E-01 | No | No |
| | **tb_width** | **3,25E-05** | **Yes** | **Yes** |
| | **sweep_duration** | **1,71E-03** | **Yes** | **Yes** |
| MGB | N_current | 8,30E-02 | No | No |
| | N_ancestral | 2,60E-01 | No | No |
| | T_split | 2,81E-01 | No | No |
| | T_demography | 8,73E-02 | No | No |
| | T_selection | 2,41E-01 | No | No |
| | T_freq99 | 1,46E-01 | No | No |
| | position_selected_allele | 4,82E-01 | No | No |
| | selective_coefficient_s | 5,41E-01 | No | No |
| | **tb_width** | **2,98E-02** | **Yes** | No |
| | **sweep_duration** | **5,82E-03** | **Yes** | **Yes** |
| CST | **N_current** | **2,16E-06** | **Yes** | **Yes** |
| | **N_ancestral** | **2,16E-06** | **Yes** | **Yes** |
| | T_split | 3,10E-01 | No | No |
| | T_demography | 8,48E-02 | No | No |
| | **T_selection** | **7,64E-03** | **Yes** | **Yes** |
| | **T_freq99** | **4,93E-03** | **Yes** | **Yes** |
| | position_selected_allele | 9,92E-01 | No | No |
| | **selective_coefficient_s** | **2,53E-04** | **Yes** | **Yes** |
| | **tb_width** | **2,19E-11** | **Yes** | **Yes** |

| | | | | |
|---|---|---|---|---|
| | sweep_duration | 2,31E-01 | No | No |
| MIG | **N_current** | **2,34E-03** | **Yes** | **Yes** |
| | **N_ancestral** | **2,34E-03** | **Yes** | **Yes** |
| | T_split | 2,21E-02 | **Yes** | No |
| | T_demography | 1,14E-01 | No | No |
| | **T_selection** | **1,93E-03** | **Yes** | **Yes** |
| | **T_freq99** | **1,84E-03** | **Yes** | **Yes** |
| | position_selected_allele | 6,87E-01 | No | No |
| | selective_coefficient_s | 4,56E-02 | **Yes** | No |
| | **tb_width** | **3,47E-08** | **Yes** | **Yes** |
| | sweep_duration | 1,33E-01 | No | No |
| EXP | **N_current** | **1,31E-14** | **Yes** | **Yes** |
| | **N_ancestral** | **7,33E-03** | **Yes** | **Yes** |
| | **T_split** | **8,45E-04** | **Yes** | **Yes** |
| | **T_demography** | **1,13E-02** | **Yes** | No |
| | **T_selection** | **2,85E-10** | **Yes** | **Yes** |
| | **T_freq99** | **4,18E-11** | **Yes** | **Yes** |
| | position_selected_allele | 2,72E-01 | No | No |
| | selective_coefficient_s | 9,63E-01 | No | No |
| | **tb_width** | **7,00E-15** | **Yes** | **Yes** |
| | **sweep_duration** | **1,16E-14** | **Yes** | **Yes** |
| MGX | **N_current** | **2,09E-03** | **Yes** | **Yes** |
| | N_ancestral | 8,18E-01 | No | No |
| | T_split | 8,97E-01 | No | No |
| | T_demography | 7,40E-01 | No | No |
| | **T_selection** | **1,74E-03** | **Yes** | **Yes** |
| | **T_freq99** | **9,89E-04** | **Yes** | **Yes** |
| | position_selected_allele | 8,48E-01 | No | No |
| | selective_coefficient_s | 4,71E-01 | No | No |
| | **tb_width** | **1,71E-06** | **Yes** | **Yes** |
| | **sweep_duration** | **3,50E-04** | **Yes** | **Yes** |

**C) Predictions generated by both models** on *sweep* dataset

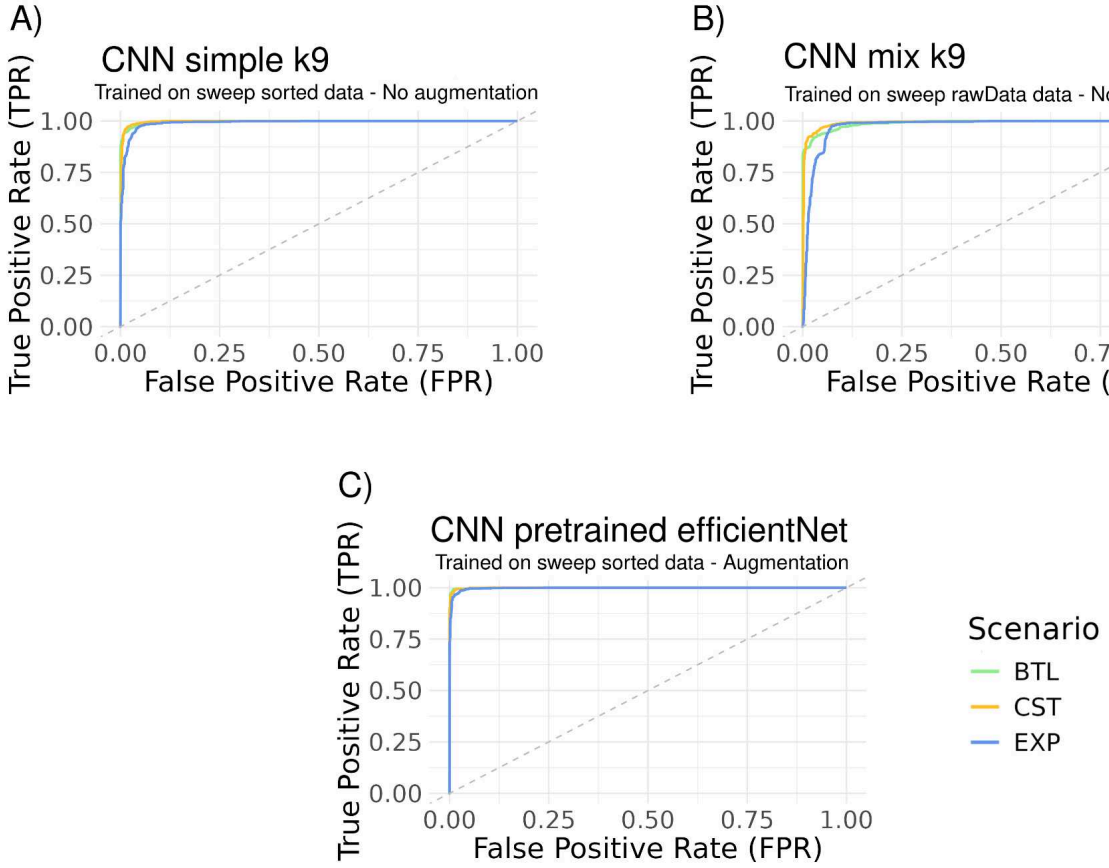
| Scenario | Variable | p-value | Significance (5%) | Significance (1%) |
|---|---|---|---|---|
| BTL | **N_current** | **3,04E-18** | **Yes** | **Yes** |
| | **N_ancestral** | **5,36E-08** | **Yes** | **Yes** |
| | **T_split** | **3,78E-08** | **Yes** | **Yes** |
| | **T_demography** | **2,60E-10** | **Yes** | **Yes** |
| | **T_selection** | **9,70E-10** | **Yes** | **Yes** |
| | **T_freq99** | **2,18E-06** | **Yes** | **Yes** |
| | position_selected_allele | 9,85E-01 | No | No |
| | **selective_coefficient_s** | **3,01E-19** | **Yes** | **Yes** |
| | **tb_width** | **6,99E-07** | **Yes** | **Yes** |
| | **sweep_duration** | **4,09E-06** | **Yes** | **Yes** |
| MGB | **N_current** | **5,14E-22** | **Yes** | **Yes** |
| | **N_ancestral** | **4,30E-09** | **Yes** | **Yes** |
| | **T_split** | **2,79E-06** | **Yes** | **Yes** |
| | **T_demography** | **6,75E-07** | **Yes** | **Yes** |
| | **T_selection** | **4,15E-09** | **Yes** | **Yes** |
| | **T_freq99** | **2,39E-07** | **Yes** | **Yes** |
| | position_selected_allele | 3,71E-01 | No | No |
| | **selective_coefficient_s** | **5,59E-20** | **Yes** | **Yes** |
| | **tb_width** | **5,24E-08** | **Yes** | **Yes** |
| | **sweep_duration** | **1,80E-05** | **Yes** | **Yes** |
| CST | N_current | 3,69E-01 | No | No |
| | N_ancestral | 3,69E-01 | No | No |
| | **T_split** | **2,34E-03** | **Yes** | **Yes** |
| | **T_demography** | **1,76E-03** | **Yes** | **Yes** |
| | **T_selection** | **9,12E-11** | **Yes** | **Yes** |
| | **T_freq99** | **2,24E-10** | **Yes** | **Yes** |
| | position_selected_allele | 5,26E-01 | No | No |
| | selective_coefficient_s | 2,27E-01 | No | No |
| | **tb_width** | **2,85E-07** | **Yes** | **Yes** |

| | | | | |
|---|---|---|---|---|
| | sweep_duration | 9,73E-01 | No | No |
| MIG | N_current | 5,49E-01 | No | No |
| | N_ancestral | 5,49E-01 | No | No |
| | **T_split** | **1,98E-13** | **Yes** | **Yes** |
| | **T_demography** | **8,35E-07** | **Yes** | **Yes** |
| | **T_selection** | **1,04E-21** | **Yes** | **Yes** |
| | **T_freq99** | **1,66E-20** | **Yes** | **Yes** |
| | position_selected_allele | 3,66E-01 | No | No |
| | selective_coefficient_s | 2,54E-01 | No | No |
| | **tb_width** | **1,26E-10** | **Yes** | **Yes** |
| | sweep_duration | 7,70E-01 | No | No |
| EXP | **N_current** | **7,20E-08** | **Yes** | **Yes** |
| | N_ancestral | 1,88E-01 | No | No |
| | T_split | 5,79E-02 | No | No |
| | T_demography | 2,34E-01 | No | No |
| | **T_selection** | **7,35E-12** | **Yes** | **Yes** |
| | **T_freq99** | **1,59E-12** | **Yes** | **Yes** |
| | position_selected_allele | 1,00E-01 | No | No |
| | selective_coefficient_s | 3,77E-01 | No | No |
| | **tb_width** | **1,36E-08** | **Yes** | **Yes** |
| | **sweep_duration** | **3,51E-08** | **Yes** | **Yes** |
| MGX | N_current | 9,69E-02 | No | No |
| | N_ancestral | 8,54E-01 | No | No |
| | **T_split** | **1,21E-04** | **Yes** | **Yes** |
| | T_demography | 8,31E-01 | No | No |
| | **T_selection** | **9,90E-14** | **Yes** | **Yes** |
| | **T_freq99** | **3,93E-14** | **Yes** | **Yes** |
| | position_selected_allele | 6,95E-01 | No | No |
| | selective_coefficient_s | 6,15E-01 | No | No |
| | **tb_width** | **1,39E-03** | **Yes** | **Yes** |
| | sweep_duration | 2,21E-01 | No | No |




**Supplementary Figure 1 -** ROC curves for the best CNN of each type of architecture, trained and tested on *neutral* datasets.

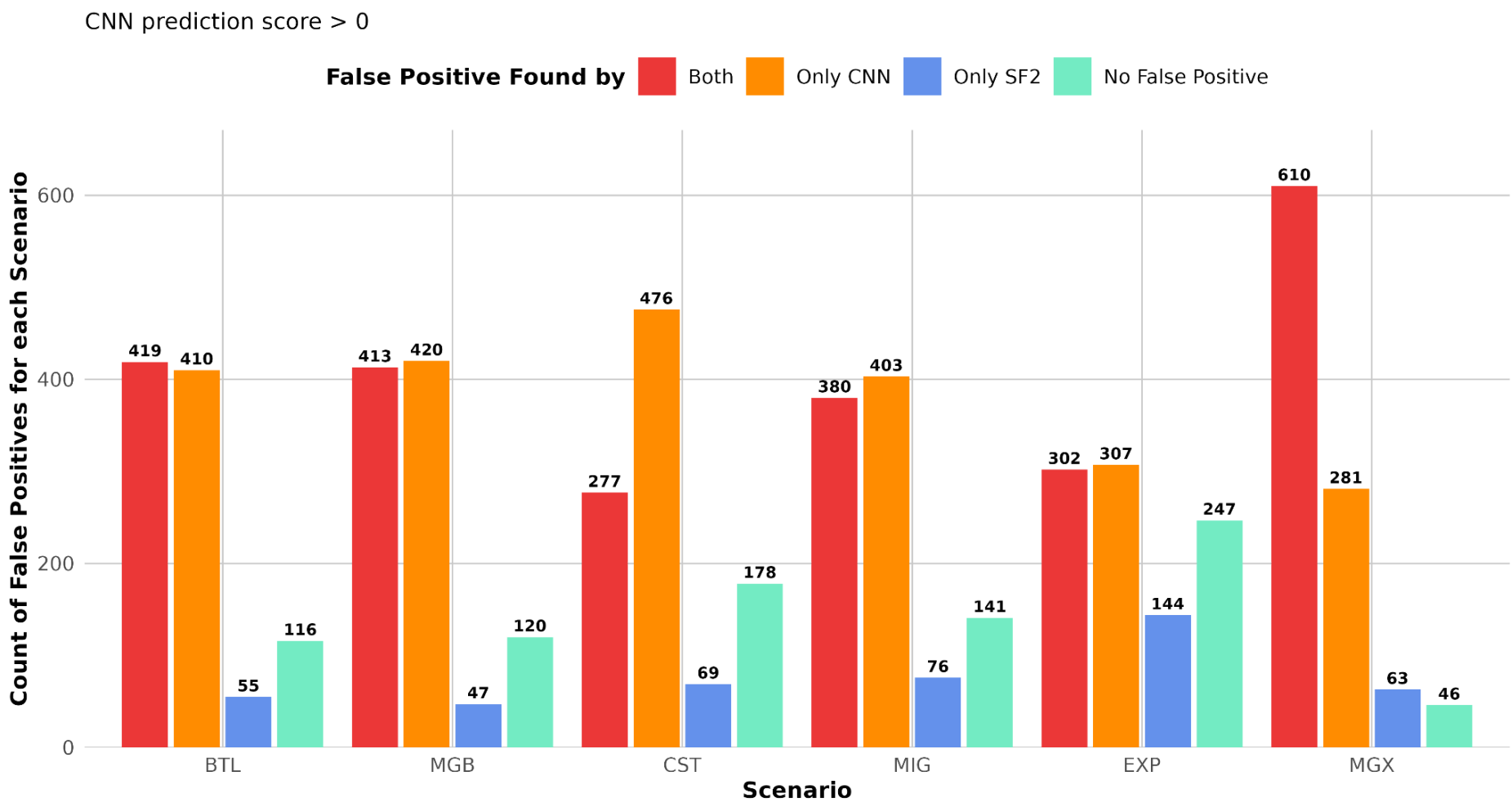
The dotted line represents the expected value for a random guess. Each curve corresponds to a One-vs-All comparison between the corresponding scenarios and the other two. A) Results for the best *simple* CNN architecture, B) results for the best *mix* CNN architecture and C) results for the best *pre-trained* CNN architecture.




**Supplementary Figure 2 -** ROC curves for the best CNN of each type of architecture, trained and tested on *sweep* datasets.

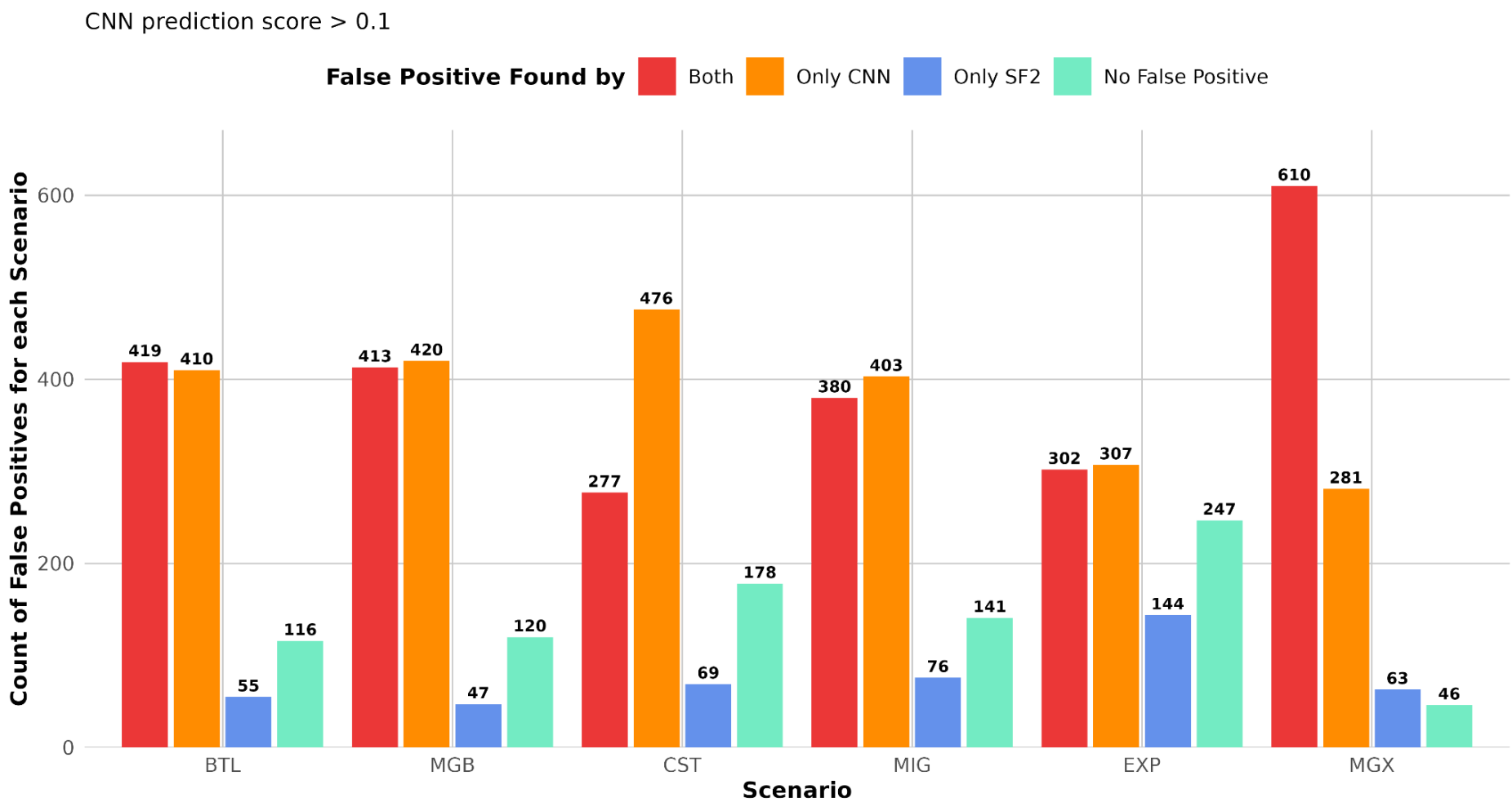
The dotted line represents the expected value for a random guess. Each curve corresponds to a One-vs-All comparison between the corresponding scenarios and the other two. A) Results for the best 'simple' CNN architecture, B) results for the best 'mix' CNN architecture and C) results for the best 'pre-trained' CNN architecture.

**Supplementary Figures 3 to 12** - Counts of FP generated on *neutral* data by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



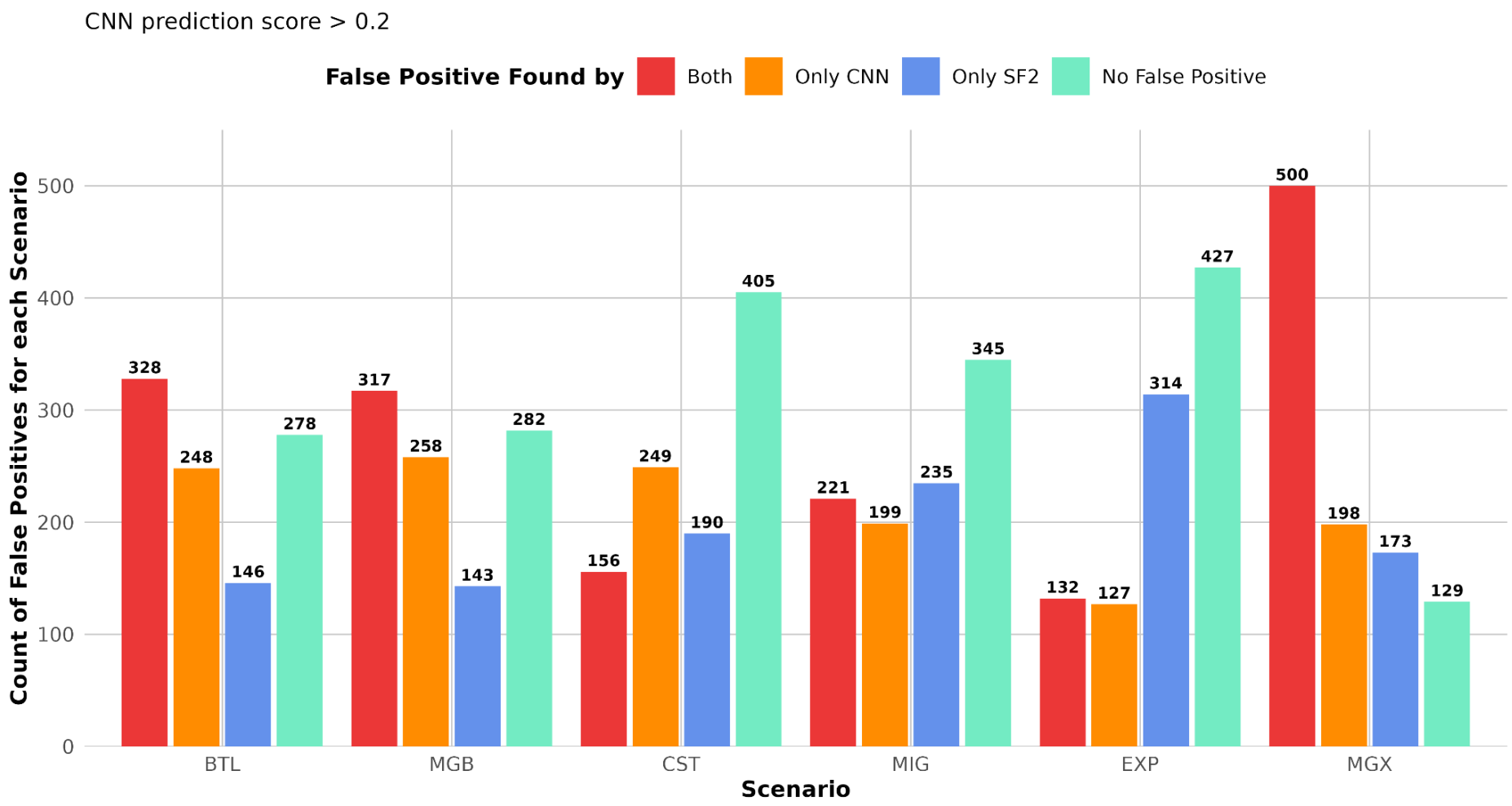

**Supplementary Figure 3** - Count of FP by CNN and/or SF2 (without CNN prediction score threshold)

**Supplementary Figures 3 to 12** - Counts of FP generated on *neutral* data by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



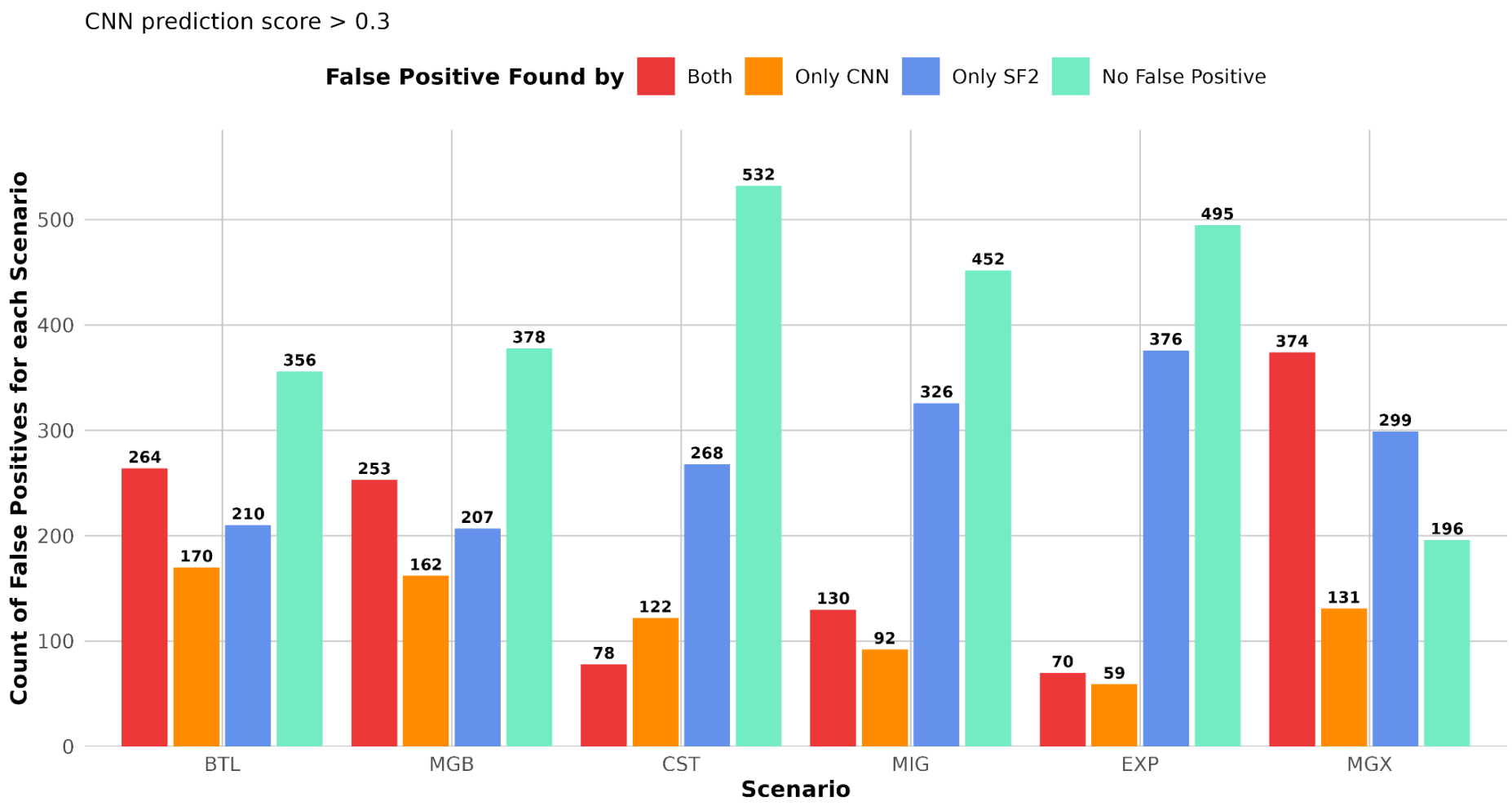

**Supplementary Figure 4** - Count of FP by CNN and/or SF2 (CNN prediction score threshold > 0.1)

**Supplementary Figures 3 to 12** - Counts of FP generated on *neutral* data by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



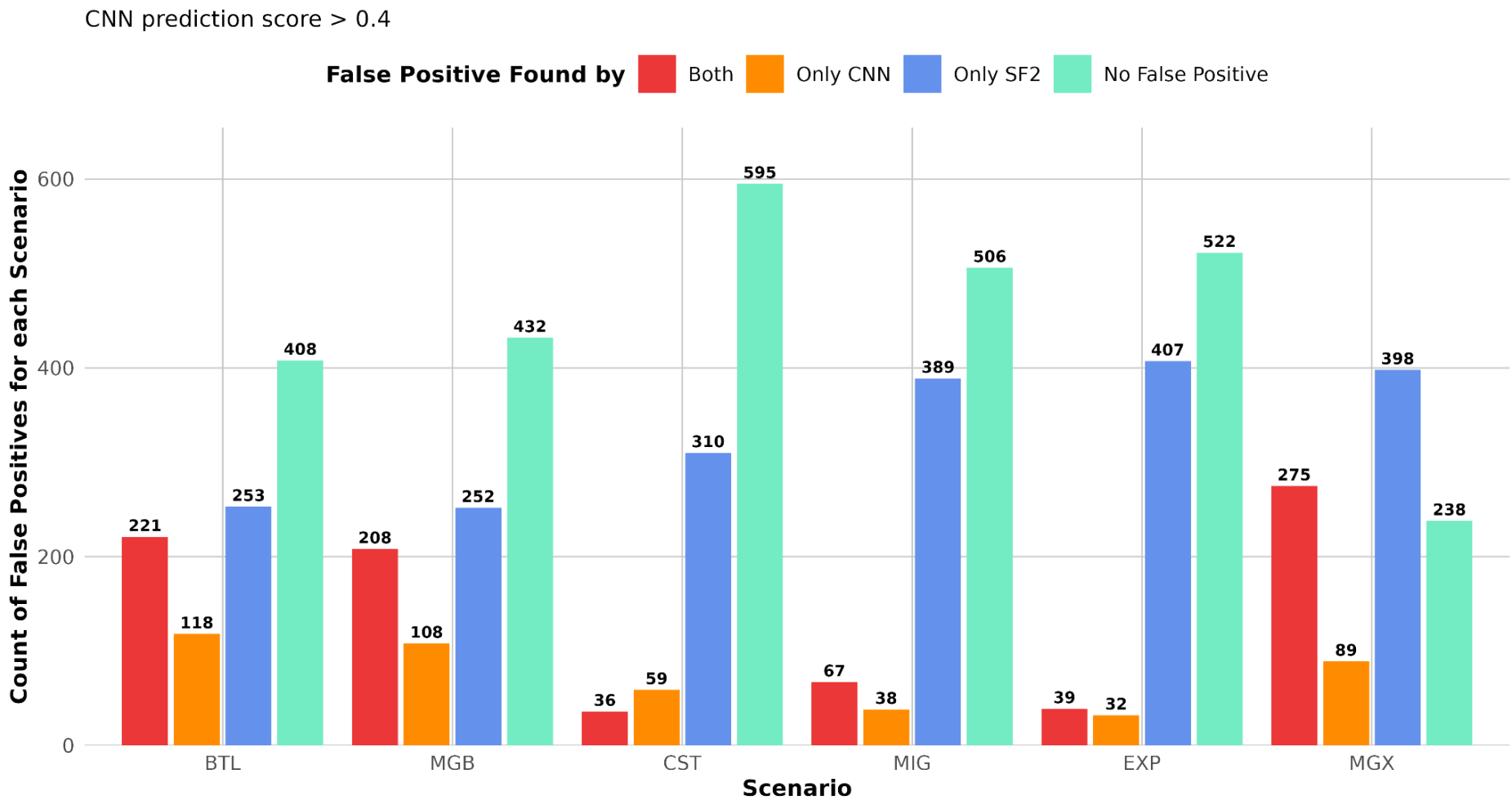

**Supplementary Figure 5** - Count of FP by CNN and/or SF2 (CNN prediction score threshold > 0.2)

**Supplementary Figures 3 to 12** - Counts of FP generated on *neutral* data by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



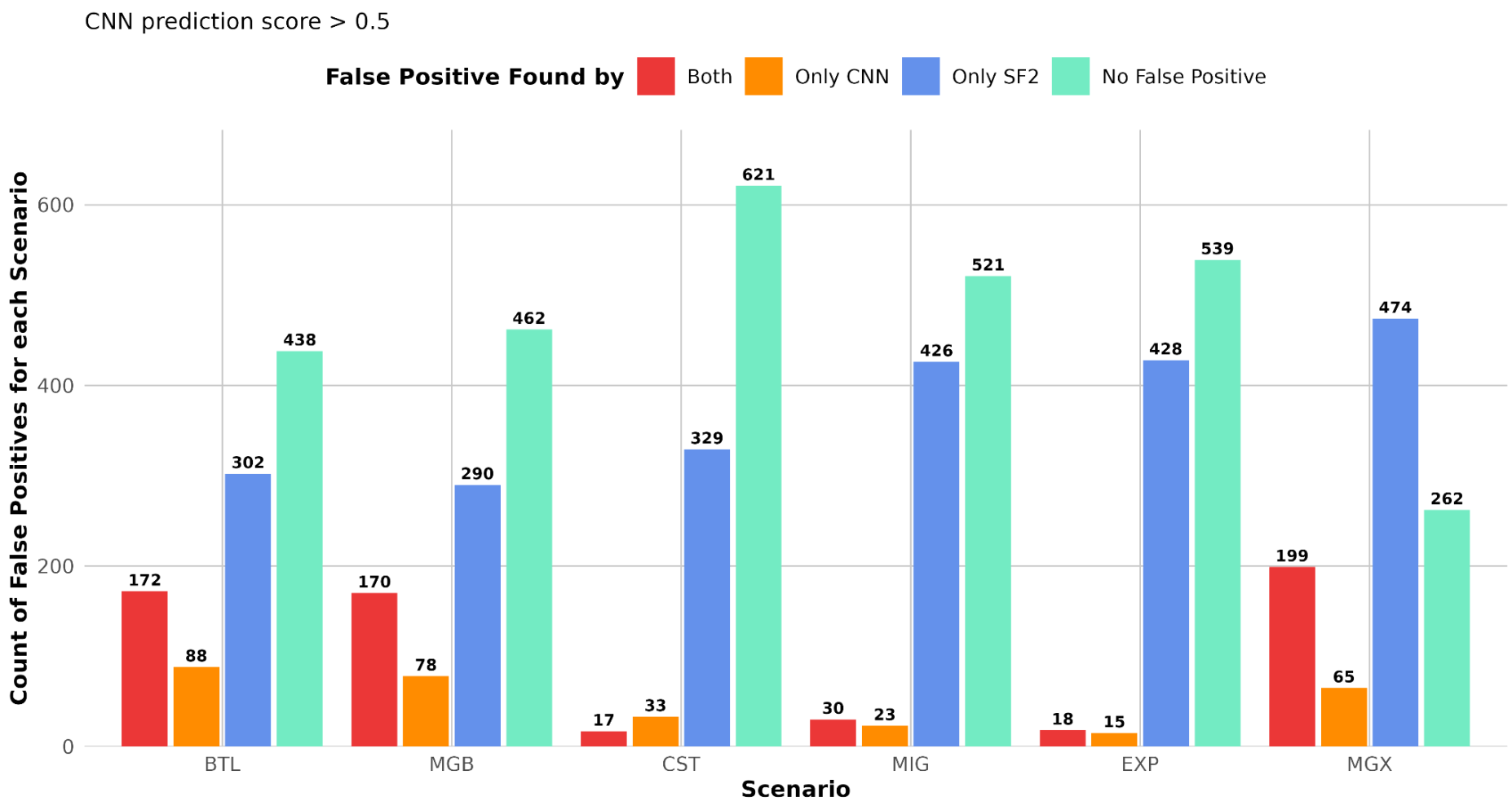

**Supplementary Figure 6** - Count of FP by CNN and/or SF2 (CNN prediction score threshold > 0.3)

**Supplementary Figures 3 to 12** - Counts of FP generated on *neutral* data by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



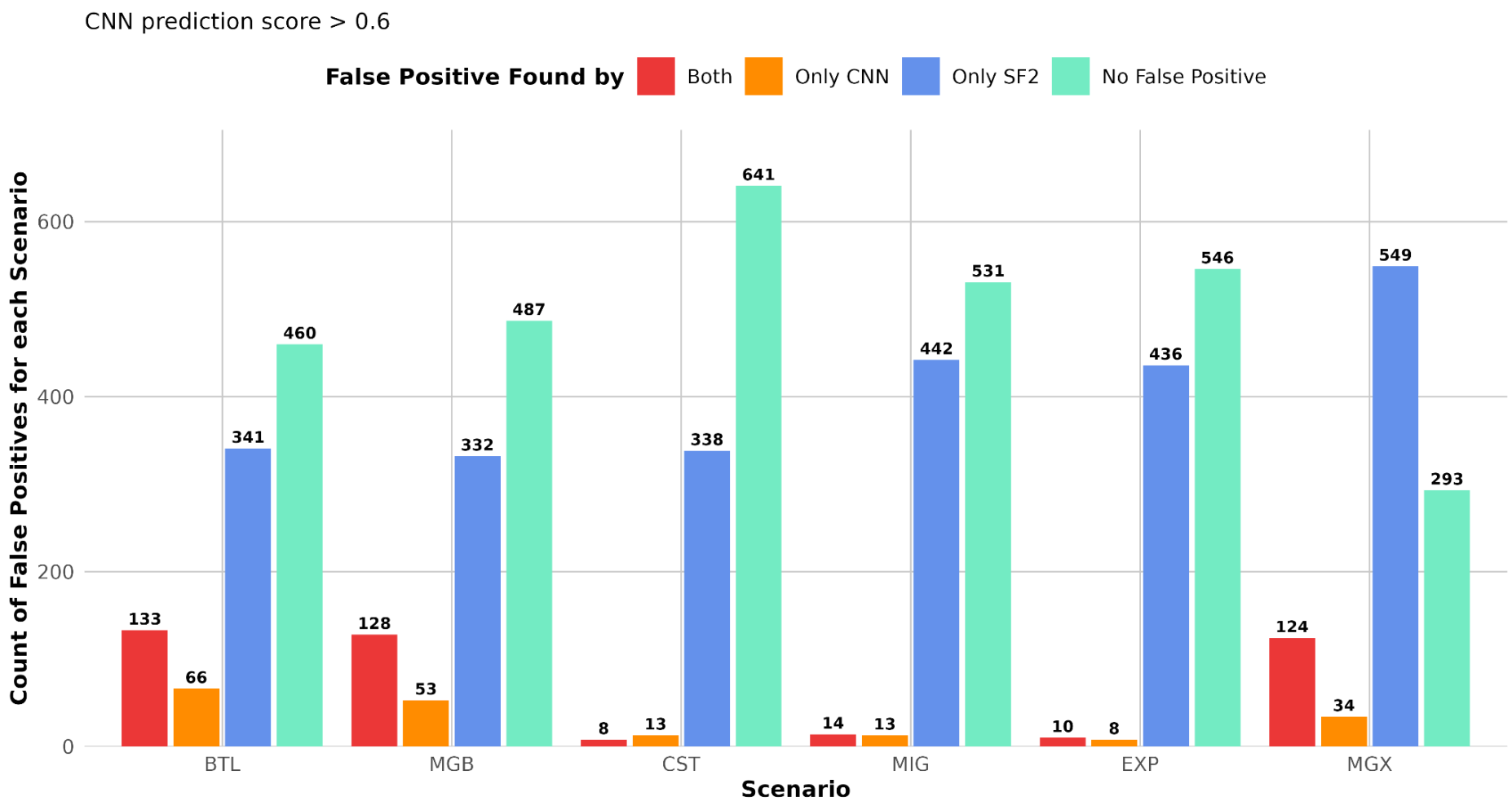

**Supplementary Figure 7** - Count of FP by CNN and/or SF2 (CNN prediction score threshold > 0.4)

**Supplementary Figures 3 to 12** - Counts of FP generated on *neutral* data by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



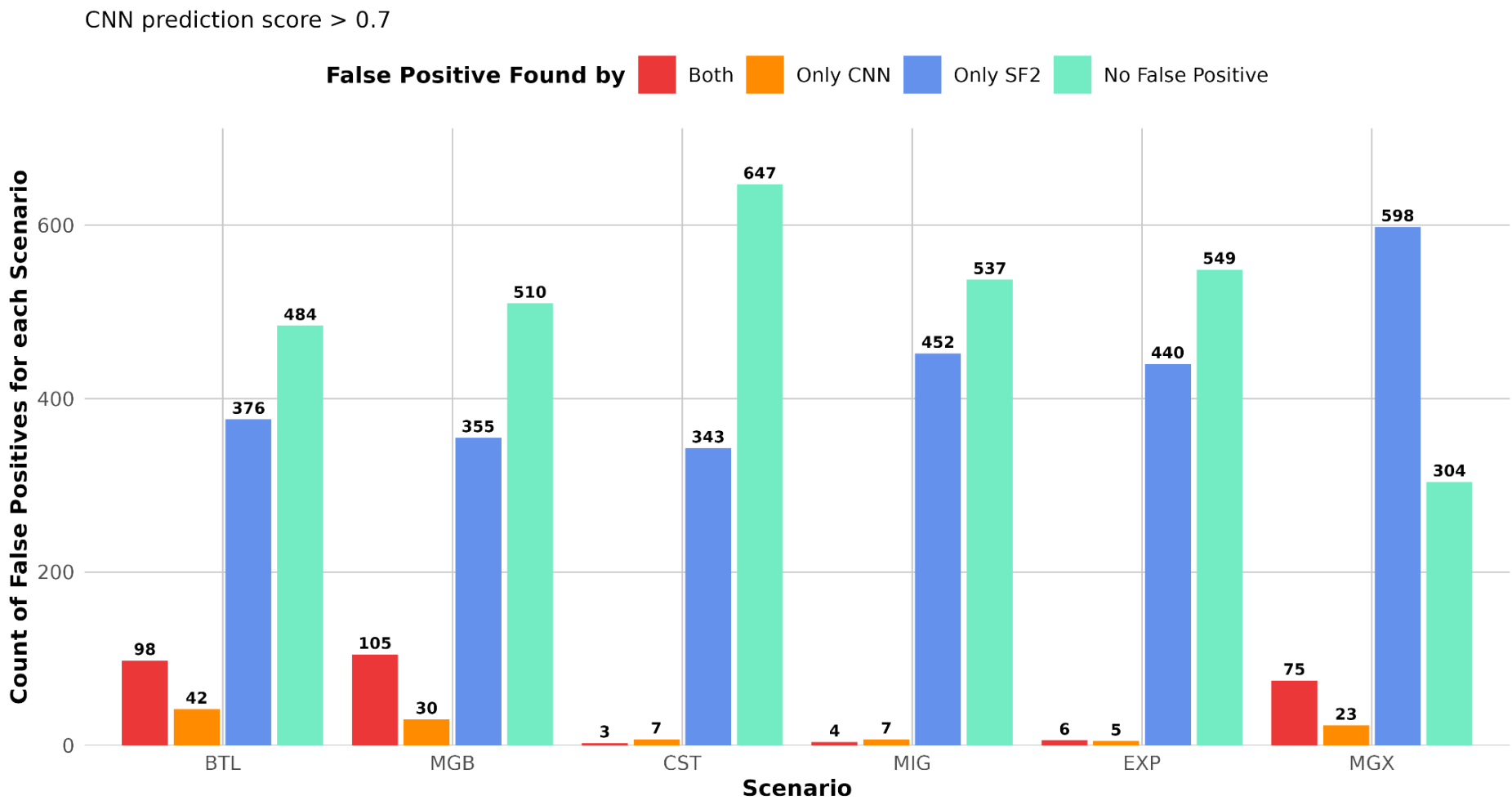

**Supplementary Figure 8** - Count of FP by CNN and/or SF2 (CNN prediction score threshold > 0.5)

**Supplementary Figures 3 to 12** - Counts of FP generated on *neutral* data by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



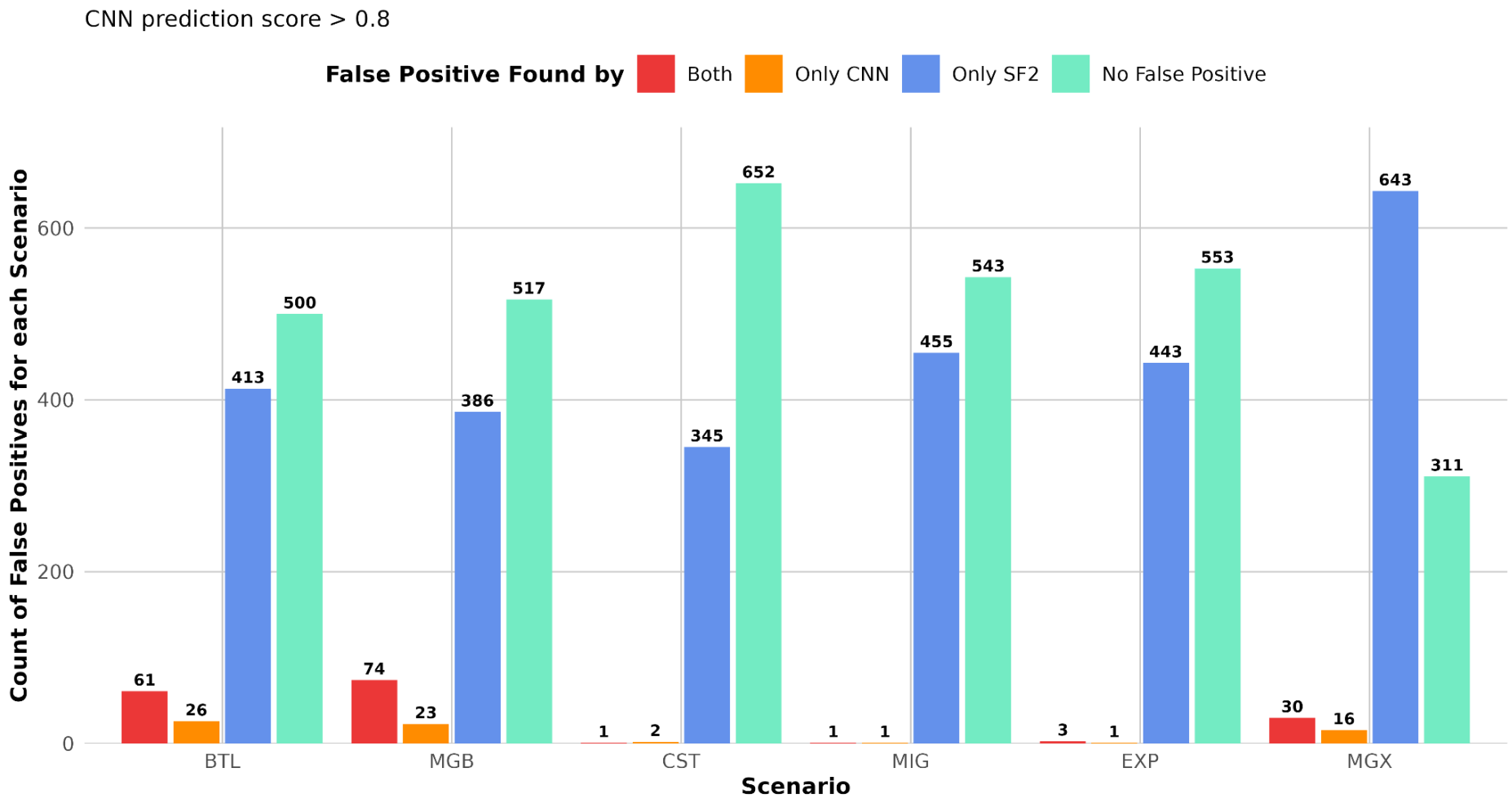

**Supplementary Figure 9** - Count of FP by CNN and/or SF2 (CNN prediction score threshold > 0.6)

**Supplementary Figures 3 to 12** - Counts of FP generated on *neutral* data by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



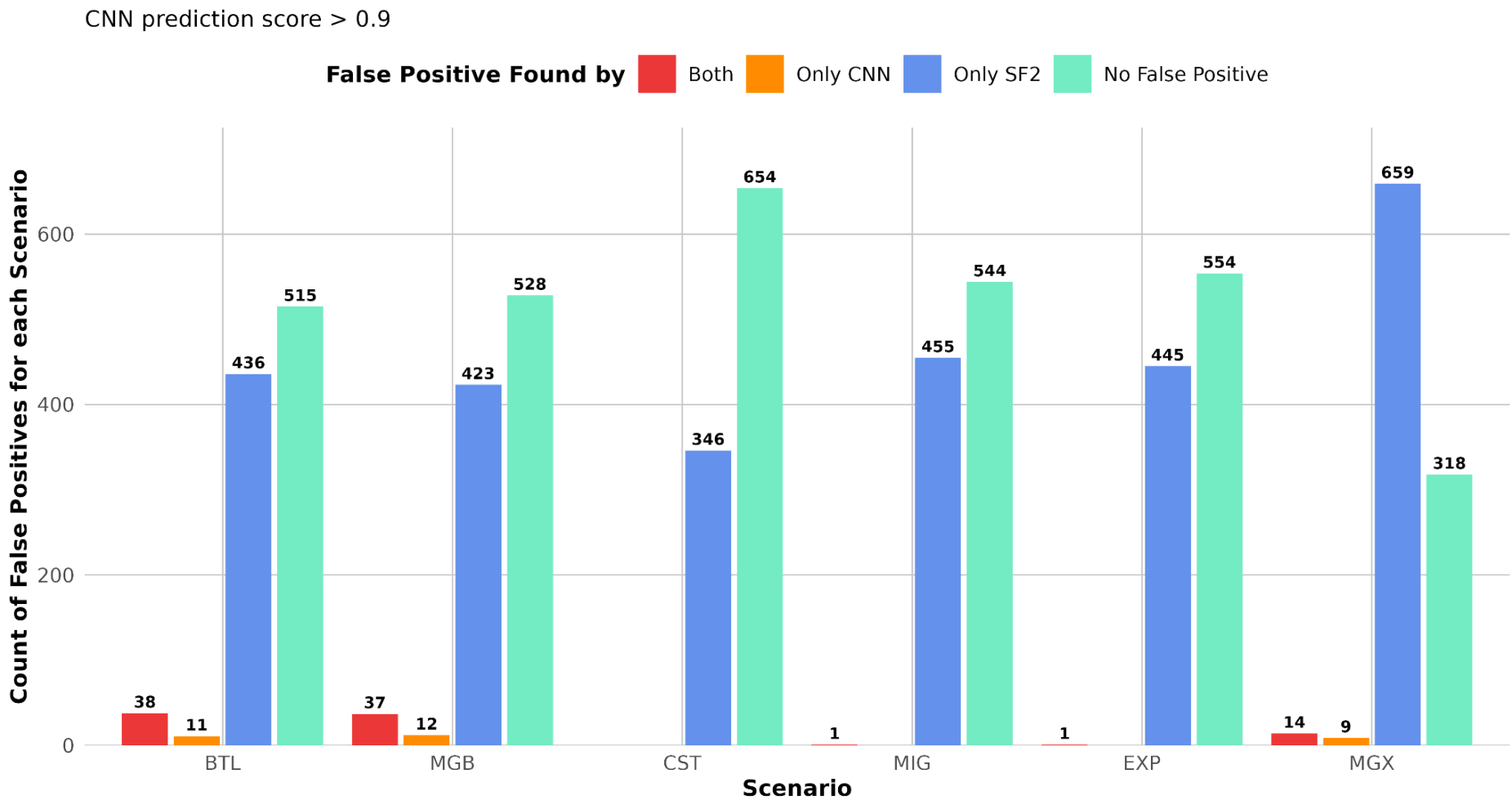

**Supplementary Figure 10** - Count of FP by CNN and/or SF2 (CNN prediction score threshold > 0.7)

**Supplementary Figures 3 to 12** - Counts of FP generated on *neutral* data by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



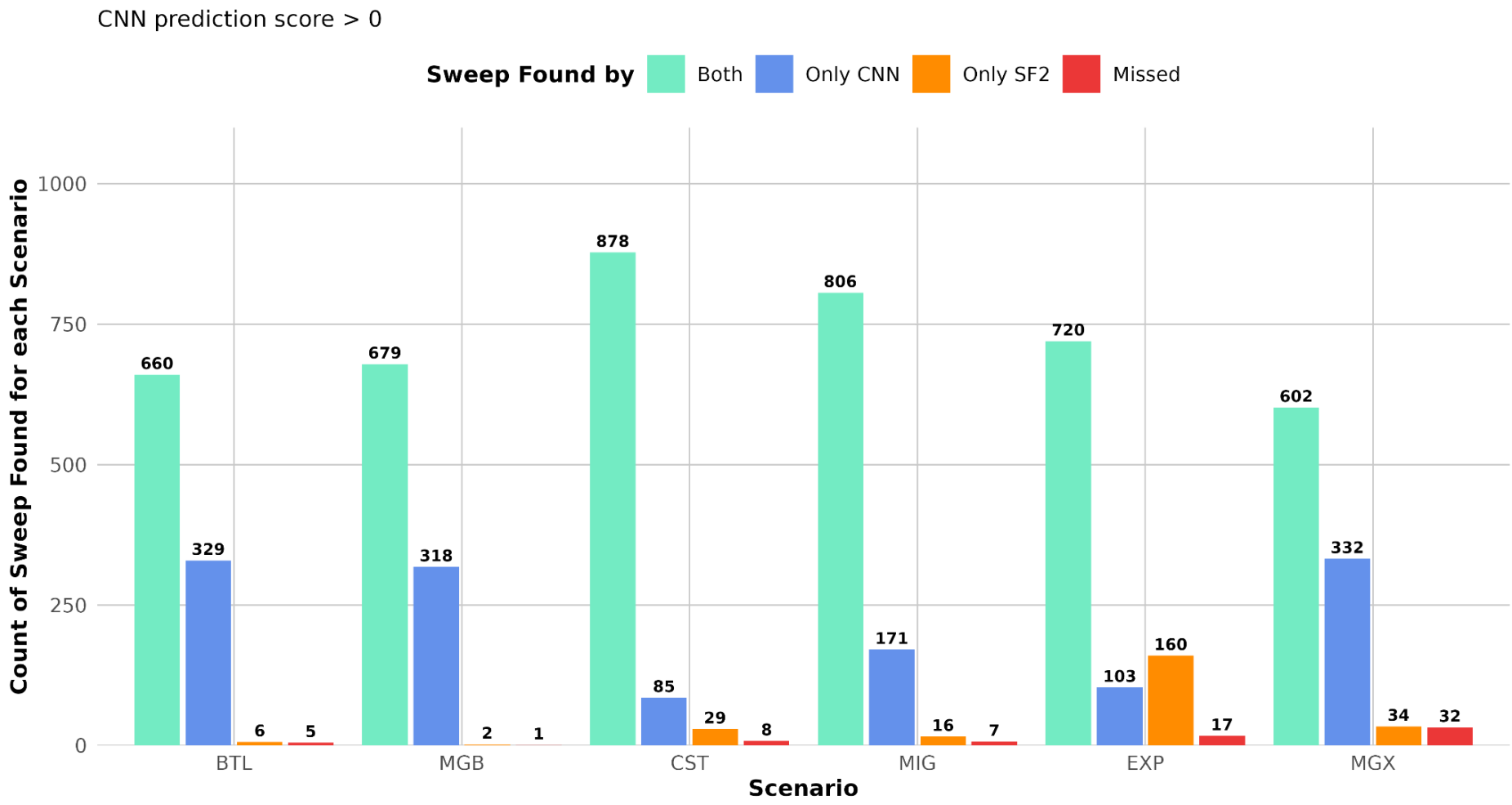

**Supplementary Figure 11** - Count of FP by CNN and/or SF2 (CNN prediction score threshold > 0.8)

**Supplementary Figures 3 to 12** - Counts of FP generated on *neutral* data by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



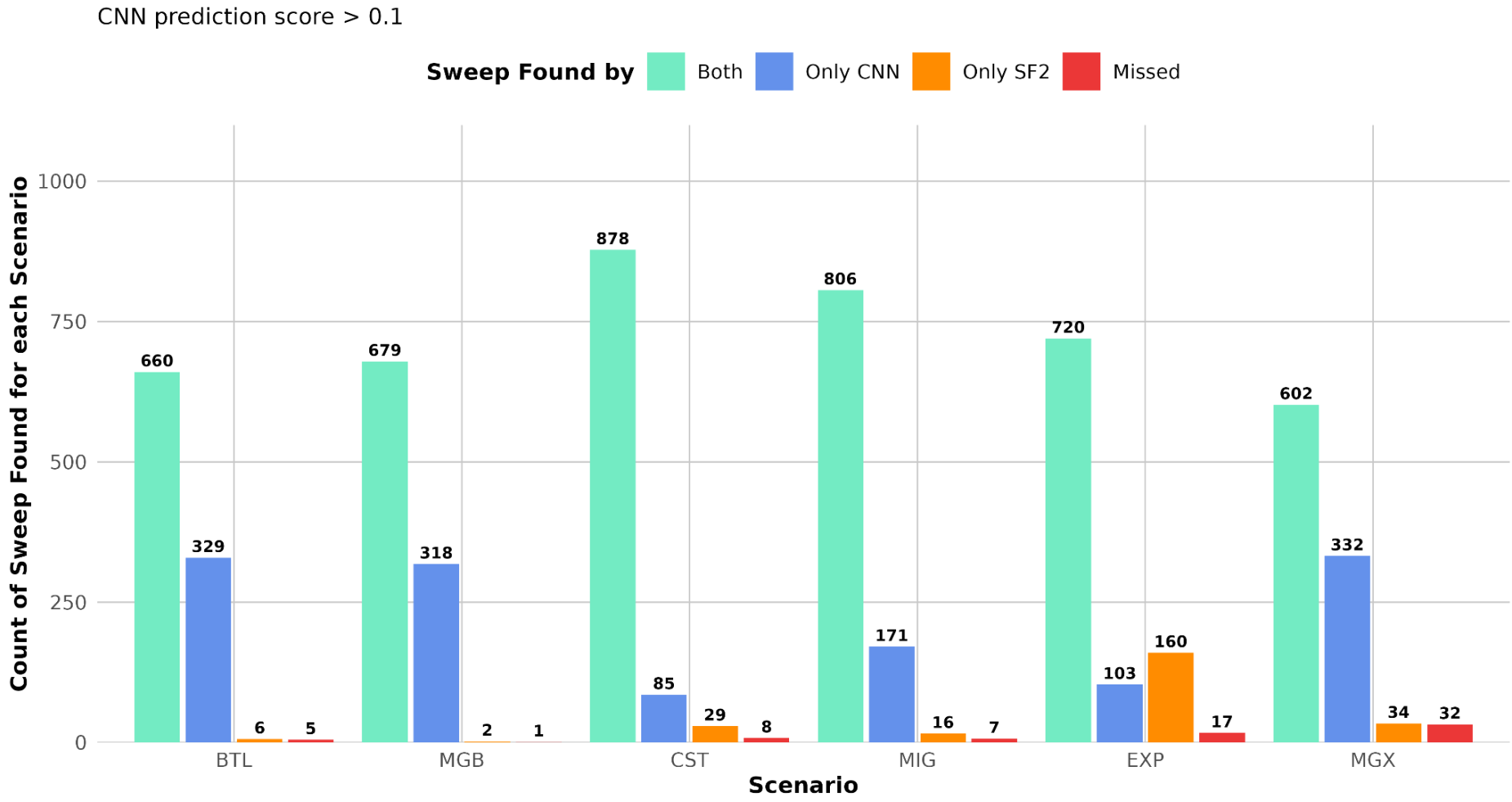

**Supplementary Figure 12** - Count of FP by CNN and/or SF2 (CNN prediction score threshold > 0.9)

**Supplementary Figures 13 to 22** - Counts of found selective sweeps by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



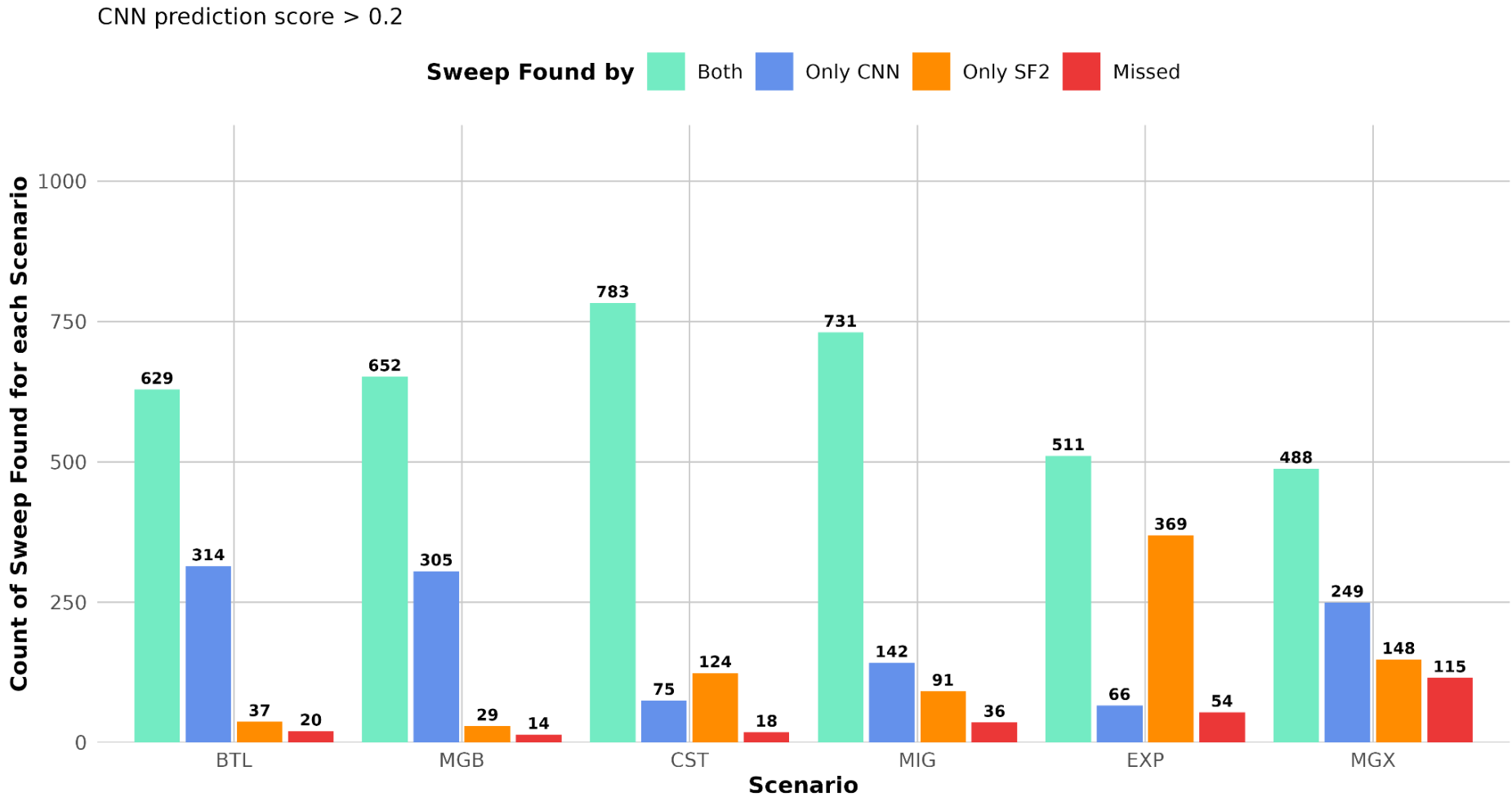

**Supplementary Figure 13** - Count of found selective sweeps by CNN and/or SF2 (without CNN prediction score threshold

**Supplementary Figures 13 to 22** - Counts of found selective sweeps by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



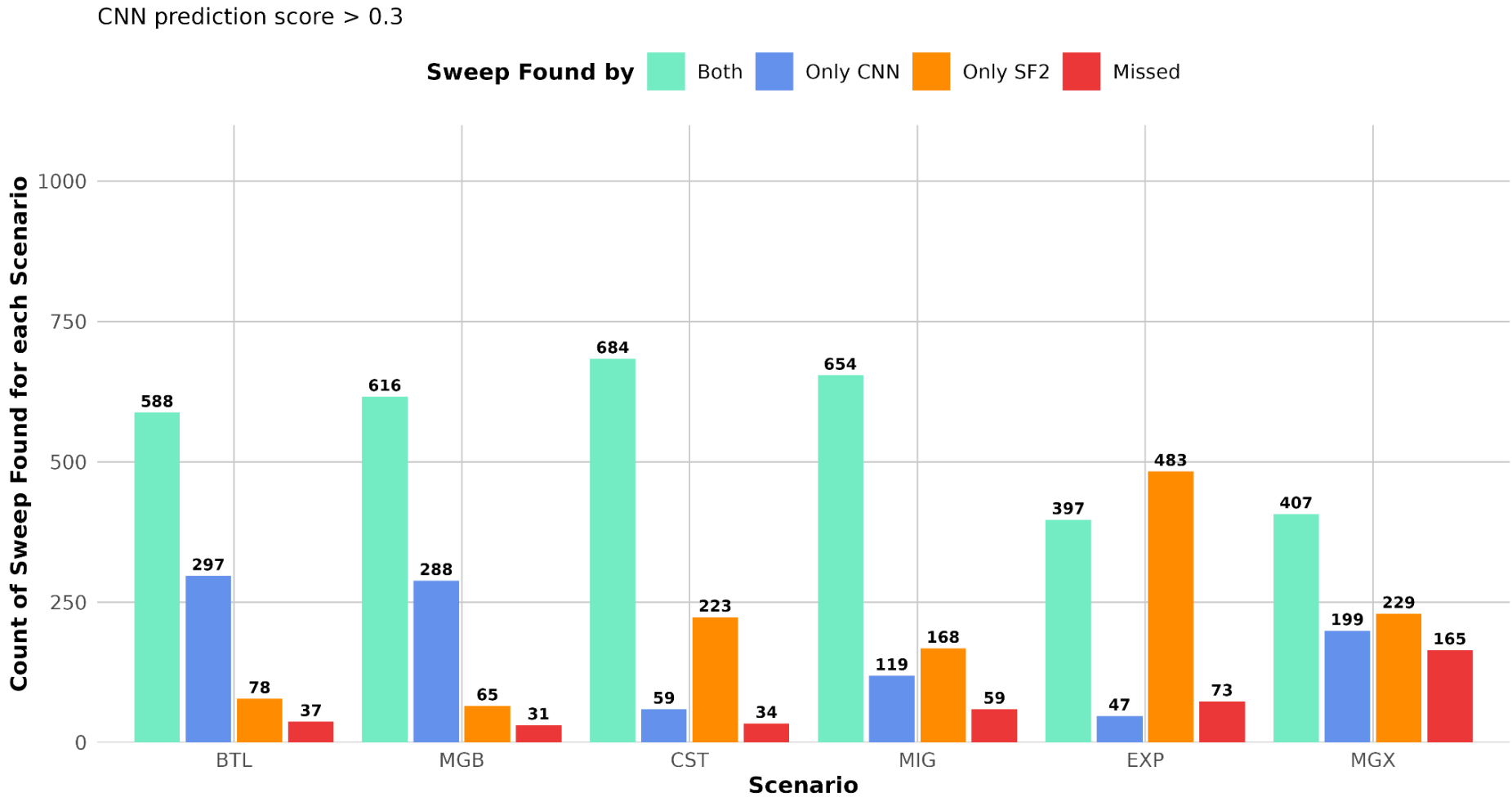

**Supplementary Figure 14** - Count of found selective sweeps by CNN and/or SF2 (CNN prediction score threshold > 0.1)

**Supplementary Figures 13 to 22** - Counts of found selective sweeps by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



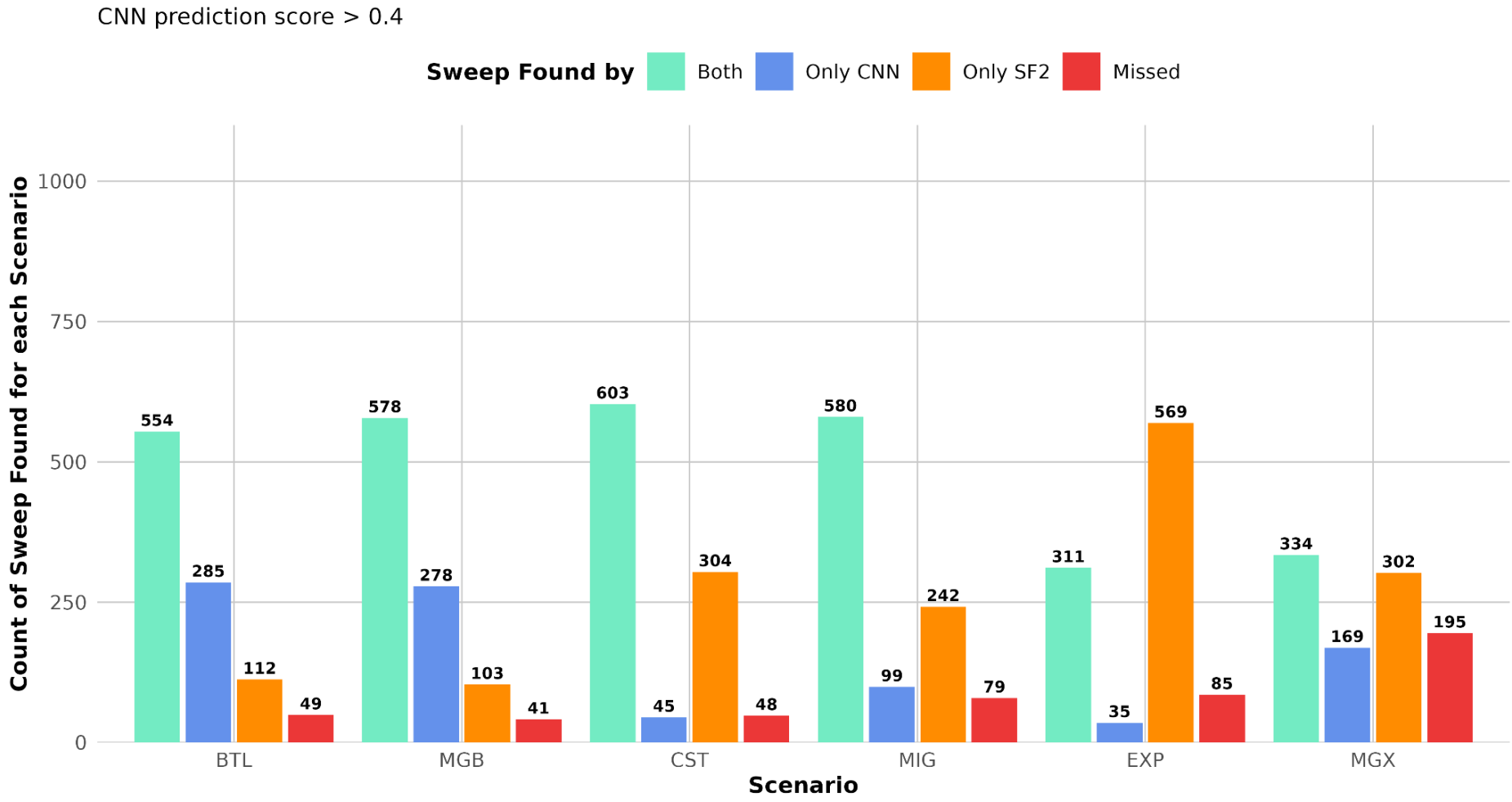

**Supplementary Figure 15** - Count of found selective sweeps by CNN and/or SF2 (CNN prediction score threshold > 0.2)

**Supplementary Figures 13 to 22** - Counts of found selective sweeps by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



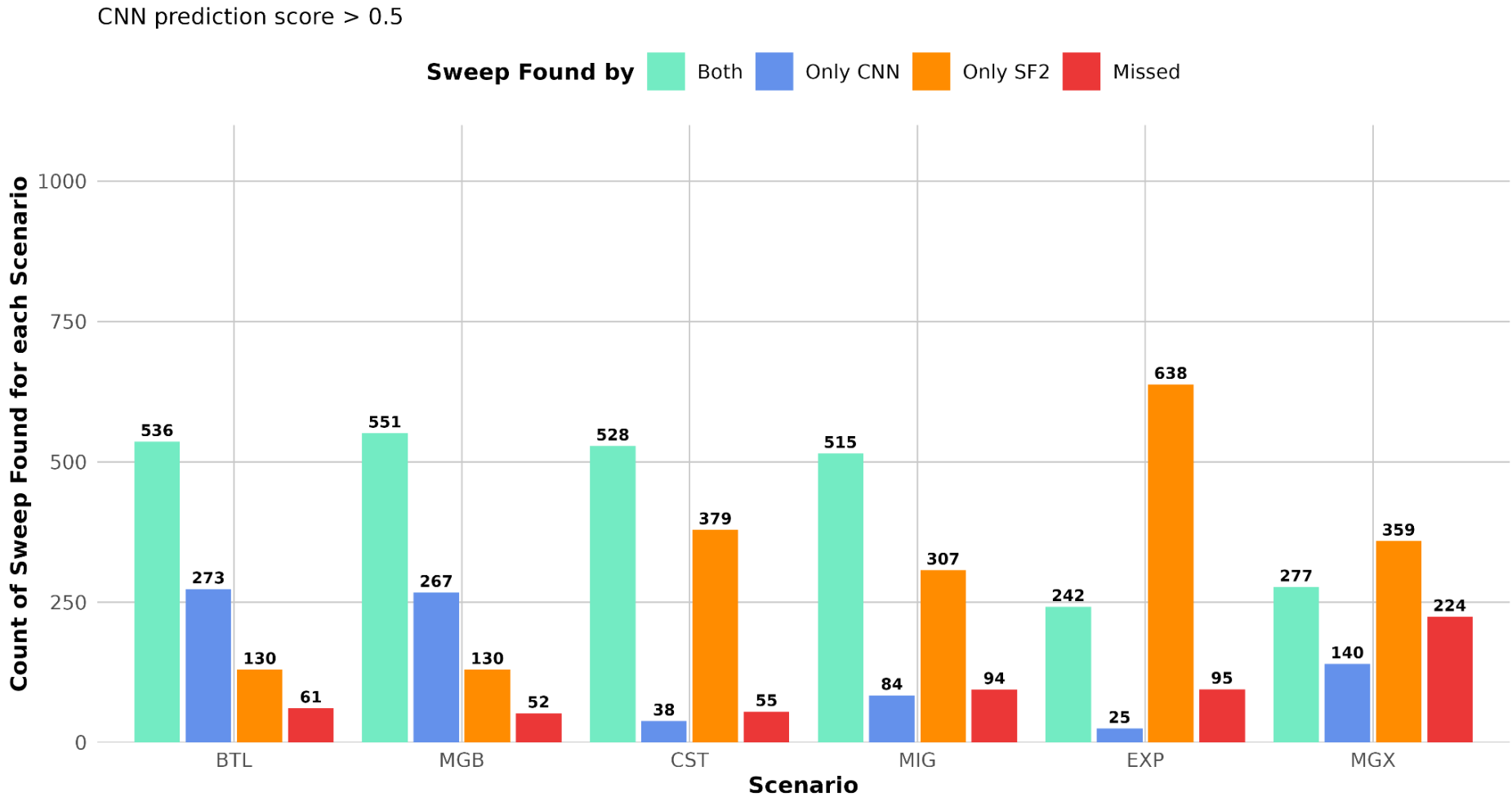

**Supplementary Figure 16** - Count of found selective sweeps by CNN and/or SF2 (CNN prediction score threshold > 0.3)

**Supplementary Figures 13 to 22** - Counts of found selective sweeps by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



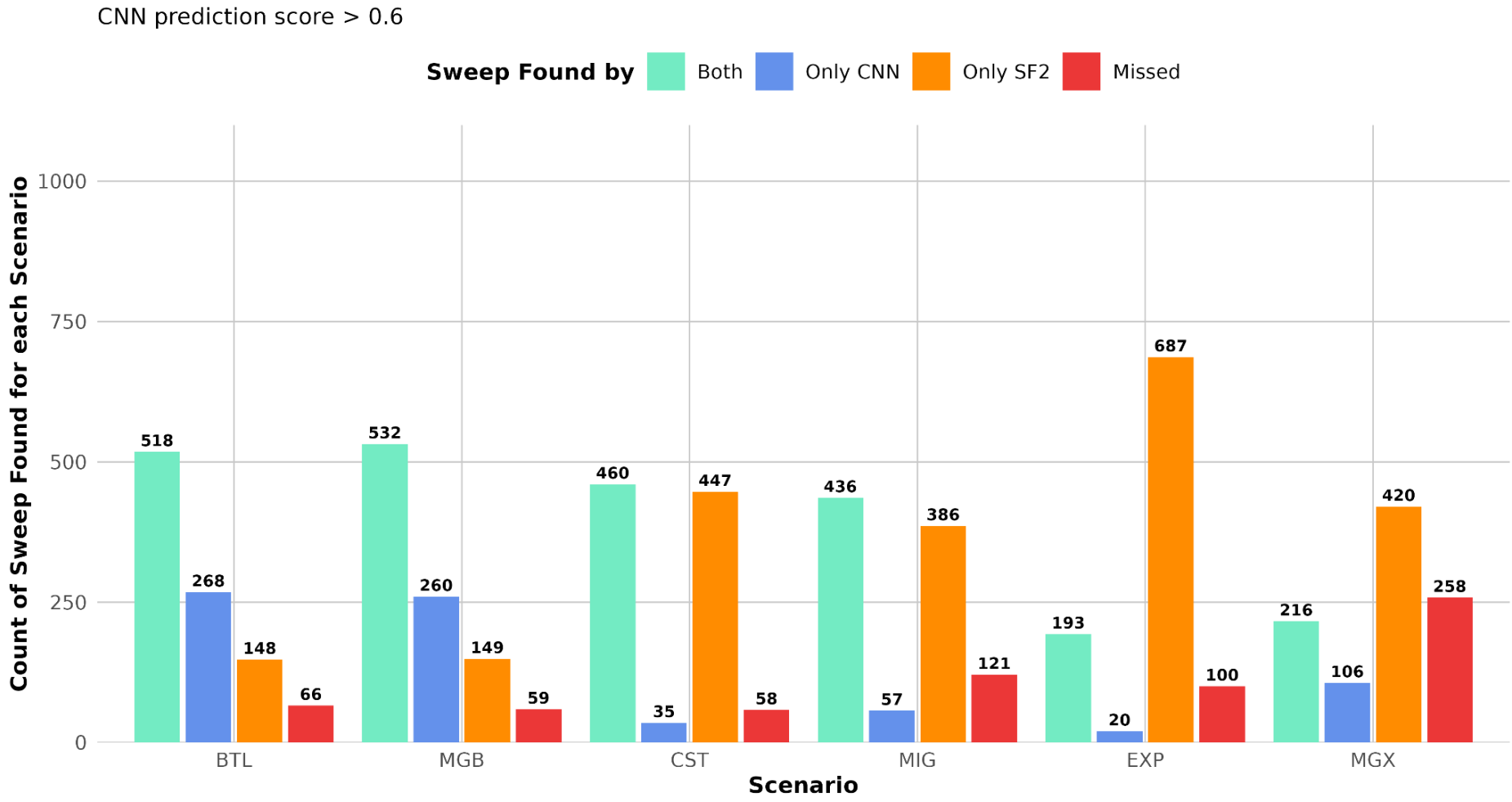

**Supplementary Figure 17** - Count of found selective sweeps by CNN and/or SF2 (CNN prediction score threshold > 0.4)

**Supplementary Figures 13 to 22** - Counts of found selective sweeps by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



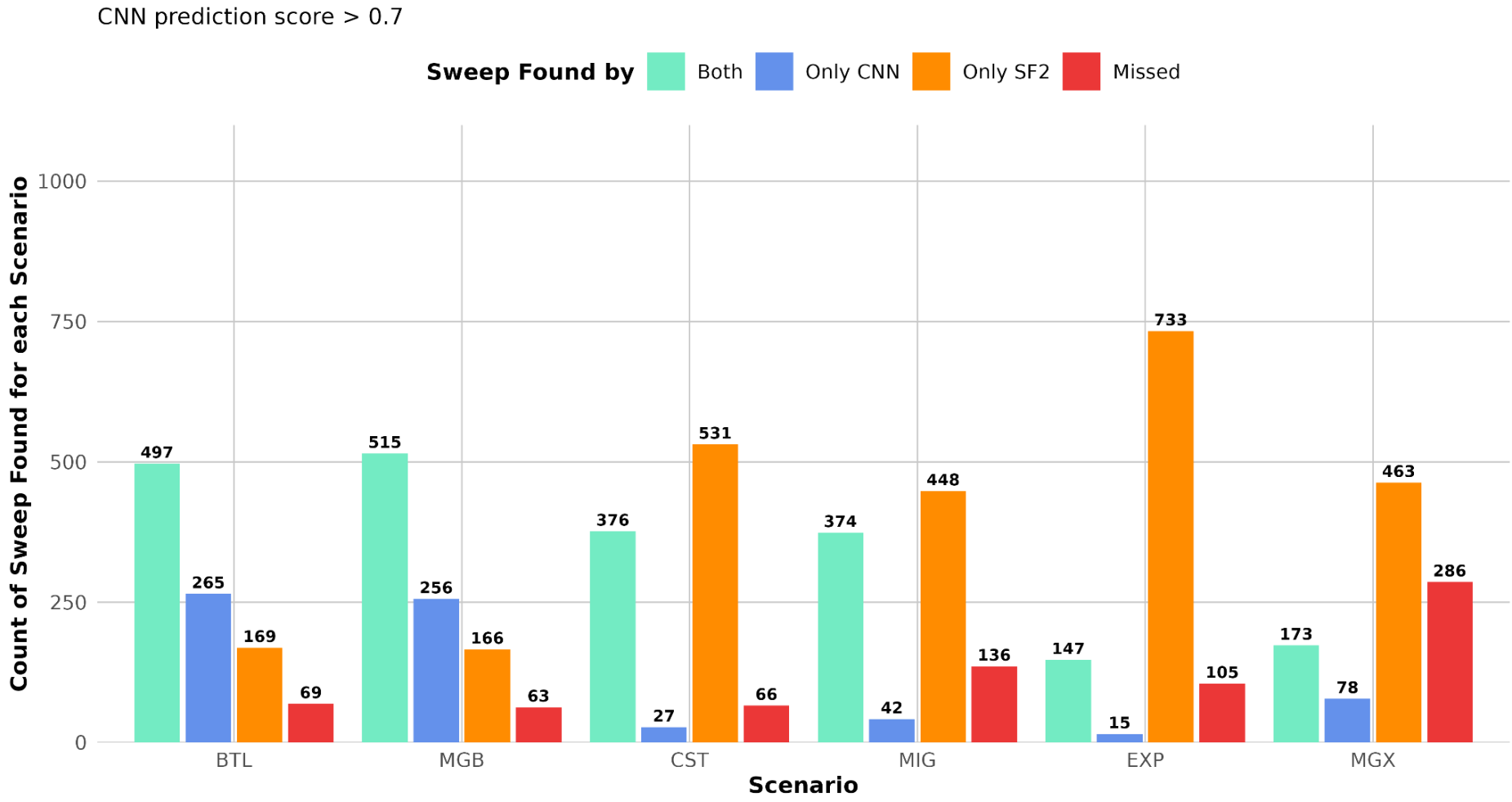

**Supplementary Figure 18** - Count of found selective sweeps by CNN and/or SF2 (CNN prediction score threshold > 0.5)

**Supplementary Figures 13 to 22** - Counts of found selective sweeps by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.



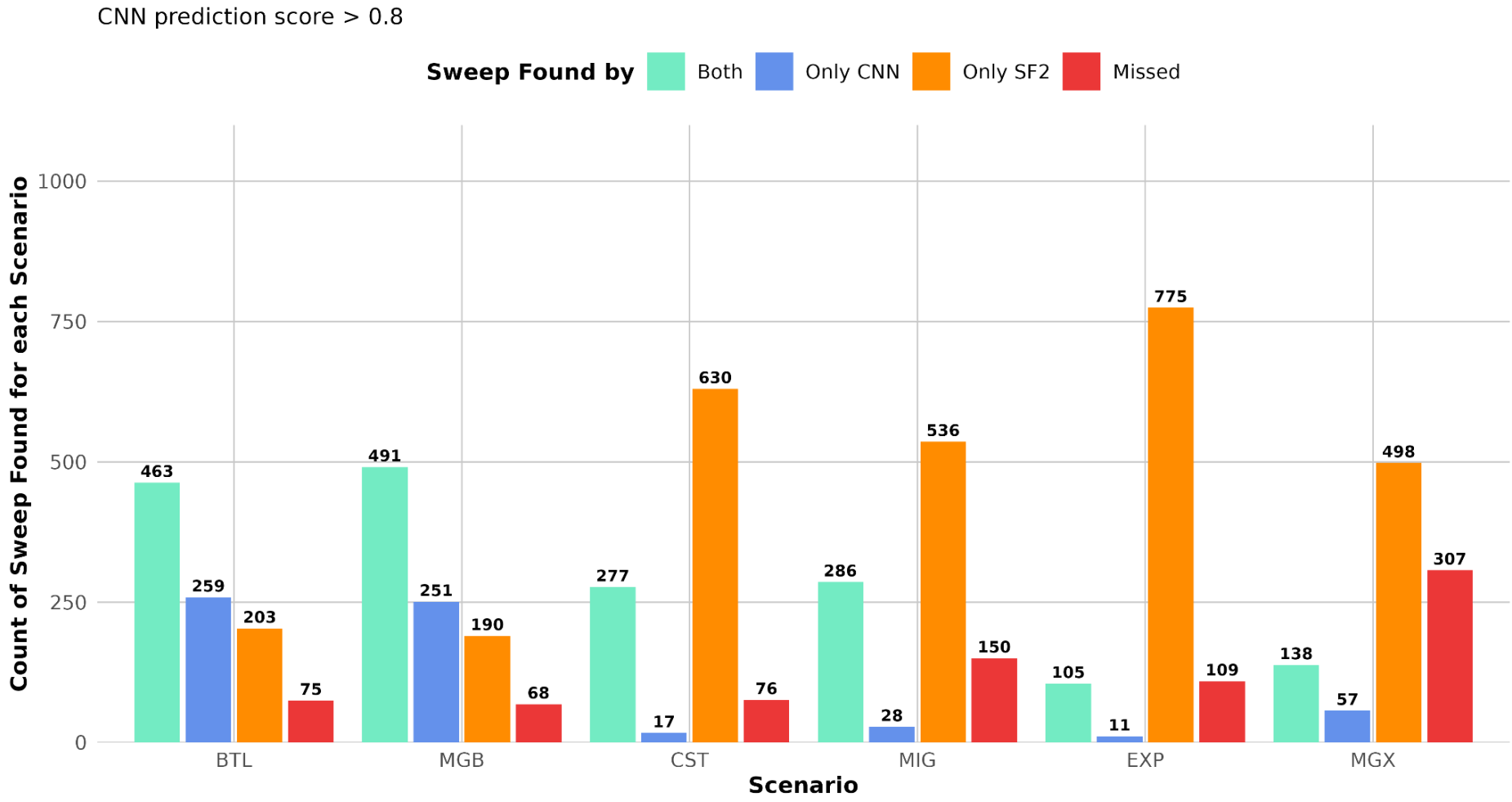

**Supplementary Figure 19** - Count of found selective sweeps by CNN and/or SF2 (CNN prediction score threshold > 0.6)

**Supplementary Figures 13 to 22** - Counts of found selective sweeps by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.




**Supplementary Figure 20** - Count of found selective sweeps by CNN and/or SF2 (CNN prediction score threshold > 0.7)

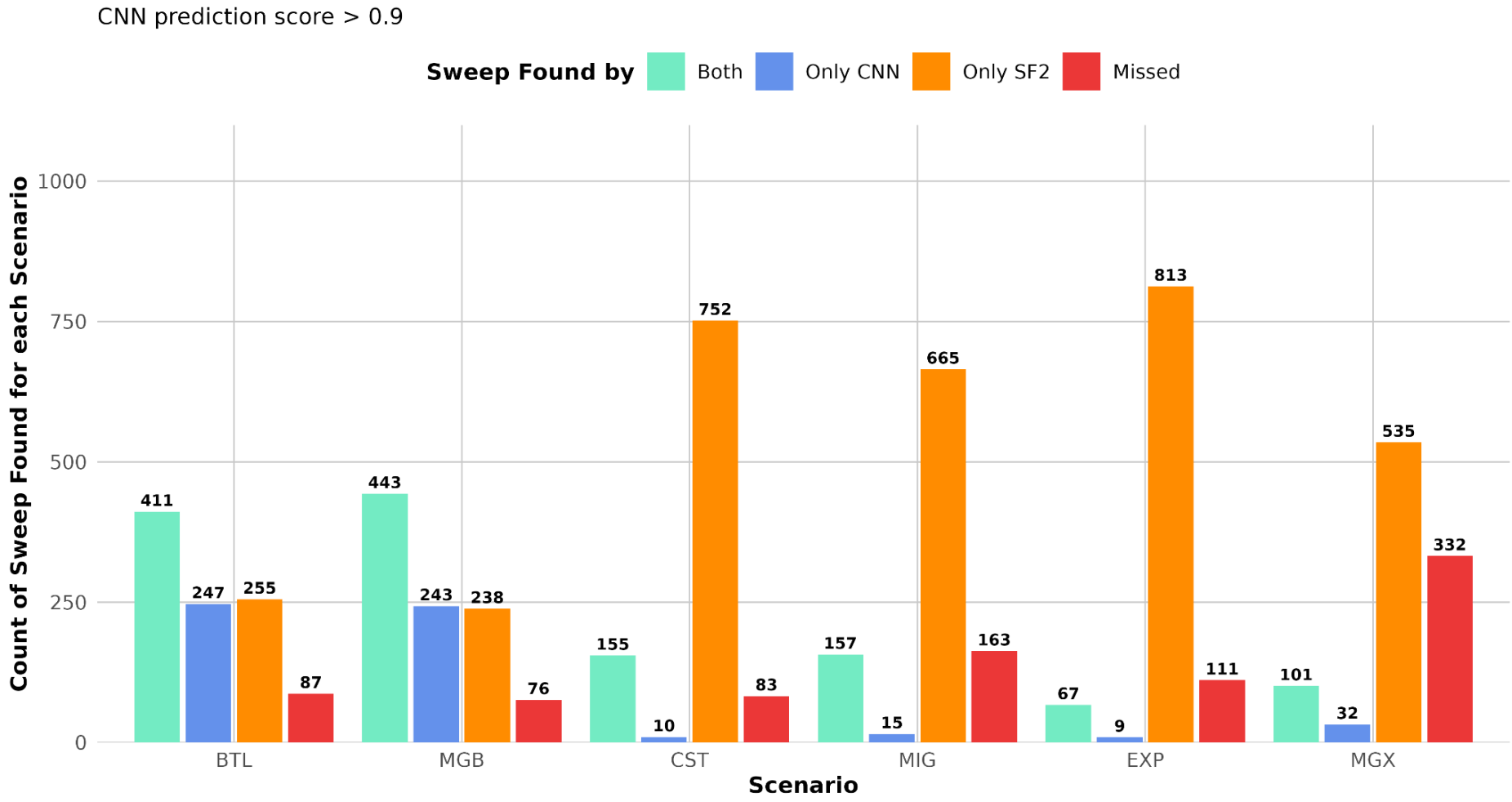
**Supplementary Figures 13 to 22** - Counts of found selective sweeps by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.




**Supplementary Figure 21** - Count of found selective sweeps by CNN and/or SF2 (CNN prediction score threshold > 0.8)

**Supplementary Figures 13 to 22** - Counts of found selective sweeps by CNN and/or SF2 for CNN prediction score threshold between 0.0 and 0.9, by increments of 0.1.

---




**Supplementary Figure 22** - Count of found selective sweeps by CNN and/or SF2 (CNN prediction score threshold > 0.9)

**Apprentissage profond pour la génétique des populations : inférer la démographie et les cibles de sélection naturelle à l'aide de réseaux de neurones convolutifs**

**Résumé** :
Inférer l'histoire démographique et détecter la sélection naturelle sont des défis majeurs en génétique des populations. Les méthodes traditionnelles, bien qu'efficaces, reposent souvent sur des hypothèses irréalistes (absence de sélection ou de changements démographiques). Or, dans les populations naturelles, la sélection et la démographie influencent simultanément la diversité génétique, complexifiant les inférences évolutives. Cette thèse explore l'utilisation des réseaux de neurones convolutionnels (CNN), une technique de deep learning, pour surmonter les limites des méthodes traditionnelles. Nos travaux se concentrent sur deux tâches principales : (1) la classification des données génomiques selon l'histoire démographique et (2) la détection et localisation des cibles de sélection naturelle le long des génomes. L'objectif n'est pas de développer un outil prêt à l'emploi mais de comprendre les considérations nécessaires à l'entraînement des CNNs pour ces tâches, en soulignant les défis et spécificités de l'élaboration de méthodes de deep learning en génétique des populations. Nous comparons les performances des CNNs sur des données simulées avec celles de méthodes établies, telles que les ABC-RF pour la classification et SweepFinder2 pour la détection de sélection. Les CNNs montrent des résultats comparables ou supérieurs, notamment pour localisation des signaux de sélection, mettant en évidence le potentiel du deep learning pour la génétique des populations, sous réserve d'une représentation des données d'entrée réfléchie et adaptée.

**Mots clés**: génétique des population, balayages sélectifs, inférences démographiques, apprentissage profond, réseaux de neurones convolutifs, détection d'objets

---

**Deep learning for population genetics: inferring demography and targets of natural selection using convolutional neural networks**

**Abstract**:
Inferring demographic history and detecting natural selection are major challenges in population genetics. Traditional methods, though effective, often rely on unrealistic assumptions (absence of selection or of demographic changes). However, in natural populations, selection and demography affect genetic diversity at the same time. Their interaction makes evolutionary inferences more complex and is a significant obstacle. In this thesis, we explore the use of convolutional neural networks (CNNs), a deep learning technique, to overcome the limitations of traditional methods. Our work focuses on two main tasks: (1) classification of genomic data based on demographic history, and (2) detecting and localizing targets of natural selection along genomes. The aim is not to develop a ready-to-use tool but to understand the necessary considerations for training CNNs on these tasks, highlighting the challenges and nuances of developing deep learning-based methods for population genetics. We compare CNN performance on simulated data with established population genetics methods, such as ABC-RF for classification and SweepFinder2 for detecting selection. CNNs achieve results comparable to or better than these existing methods, especially in localizing selection signals. This highlights the potential of deep-learning approaches for population genetics, as long as careful consideration is given to input data representation.

**Key words**: population genetics, selective sweeps, demographic inferences, deep learning, convolutional neural networks, object detection

Université de Lille

EDSMRE