université
PARIS-SACLAY

Reconstructing our past: deep learning for population genetics

Reconstruire notre passé : apprentissage statistique profond pour la génétique des populations

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique
Graduate School : Informatique et sciences du numérique,
Référent : Faculté des sciences d'Orsay

Thèse préparée dans l'unité de recherche Laboratoire interdisciplinaire des sciences du numérique (Université Paris-Saclay, CNRS), sous la direction de Marc SCHOENAUER, directeur de recherche INRIA, le co-encadrement de Guillaume CHARPIAT, chargé de recherche INRIA et le co-encadrement de Flora JAY, chargée de recherche CNRS.

Thèse soutenue à Paris-Saclay, le 18 mars 2022, par

Théophile SANCHEZ

Composition du jury

Blaise HANCZAR Professeur des universités, IBISC, Université Paris-Saclay, Université d'Évry	Président
Aurélien TELLIER Professor, HDR, WZW, Technische Universität München	Rapporteur & Examinateur
Jean-Philippe VERT Research scientist, Google Brain, entreprise	Rapporteur & Examinateur
Audrey DURAND Professeure adjointe, Université Laval	Examinatrice
Stéphanie MANEL Professeure des universités, CEFE, École Pratique des Hautes Études	Examinatrice
Marc SCHOENAUER Directeur de Recherche, LISN, INRIA, Université Paris-Saclay	Directeur de thèse

THESE DE DOCTORAT
NNT : 2022UPASG032

Remerciements

Je tiens tout d'abord à remercier mes encadrants Flora Jay et Guillaume Charpiat pour m'avoir donné l'opportunité de réaliser ce doctorat et m'avoir accordé leur confiance. J'espère que ce travail reflète leur bienveillance et la quantité inestimable de connaissances qu'ils m'ont transmises durant toutes ces années.

Merci à Jean Cury, Erik Madison Bray et Pierre Jobic pour être derrière un grand nombre des idées et des lignes de code présentées dans cette thèse.

Merci aux rapporteurs de ces travaux, Aurélien Tellier et Jean-Philippe Vert, ainsi qu'aux examinateurs Audrey Durand, Blaise Hanczar et Stéphanie Manel pour m'avoir fait l'honneur de participer au jury et d'avoir pris le temps d'évaluer mon travail.

Un très grand merci à tous mes collègues de l'université Paris-Saclay, des équipes TAU et BioInfo pour tous ces moments partagés autour d'un café, d'une bière ou bien plus encore. Adrien, Aris, Armand, Aurélien, Burak, Corentin, Cyril, Diviyan, Éléonore, Étienne, Fanny, François, Giancarlo, Jérémy, Joël, Laurent, Loris, Manon, Marc, Marine, Nilo, Olivier, Raphaël, Sarah, Saumya, Tamon, Victor et Zhengying, ce fut un plaisir de travailler à vos côtés.

Merci à Marc Schoenauer et à Michèle Sebag d'avoir créé une équipe de recherche avec une ambiance aussi formidable.

Merci à tous mes amis caladois, lyonnais et parisiens pour m'avoir supporté dans tous les sens du terme. Merci tout particulièrement à Agnès, Audrey, Auriane, Aziliz, Grégoire, Julien, Lucas, Marianne, Noëlie, Paul, Rosa, Samuel, Thibault, Thomas, Théo, Valentina et Vincent, sans le savoir, vous m'avez aidé pendant des étapes importantes de ma thèse.

Merci aux membres de ma famille pour votre soutien indéfectible et pour m'avoir transmis le gène de la recherche s'il existe.

Et enfin, merci à vous qui prenez le temps de lire ces pages.

Résumé / Abstract

Résumé

Avec l'explosion des technologies de séquençage, de plus en plus de données génomiques sont disponibles, ouvrant la voie à une connaissance approfondie des forces évolutives en œuvre et en particulier de l'histoire démographique des populations. Toutefois, extraire l'information intéressante de ces données massives de manière efficace reste un problème ouvert. Compte tenu de leurs récents succès en apprentissage statistique, les réseaux de neurones artificiels sont un candidat sérieux pour mener à bien une telle analyse. Ces méthodes ont l'avantage de pouvoir traiter des données ayant une grande dimension, de s'adapter à la plupart des problèmes et d'être facilement mis à l'échelle des moyens de calcul disponibles. Cependant, leur performance dépend fortement de leur architecture qui requiert d'être en adéquation avec les propriétés des données afin d'en tirer le maximum d'information. Dans ce cadre, cette thèse présente de nouvelles approches basées sur l'apprentissage statistique profond, ainsi que les principes permettant de concevoir des architectures adaptées aux caractéristiques des données génomiques. L'utilisation de couches de convolution et de mécanismes d'attention permet aux réseaux présentés d'être invariants aux permutations des haplotypes échantillonnés et de s'adapter à des données de dimensions différentes (nombre d'haplotypes et de sites polymorphes). Les expériences conduites sur des données simulées démontrent l'efficacité de ces approches en les comparant à des architectures de réseaux plus classiques, ainsi qu'à des méthodes issues de l'état de l'art. De plus, la possibilité d'assembler les réseaux de neurones à certaines méthodes déjà éprouvées en génétique des populations, comme l'*approximate Bayesian computation*, permet d'améliorer les résultats et de combiner leurs avantages. La praticabilité des réseaux de neurones pour l'inférence démographique est testée grâce à leur application à des séquences génomiques complètes provenant de populations réelles de *Bos taurus* et d'*Homo sapiens*. Enfin, les scénarios obtenus sont comparés aux connaissances actuelles de l'histoire démographique de ces populations.

Abstract

Constant improvement of DNA sequencing technology that produces large quantities of genetic data should greatly enhance our knowledge of evolution, particularly demographic history. However, the best way to extract information from this large-scale data is still an open problem. Neural networks are a strong candidate to attain this goal, considering their recent success in machine learning. These methods have the advantages of handling high-dimensional data, adapting to most applications and scaling efficiently to available computing resources. However, their performance dependents on their architecture, which should match the data properties to extract the maximum information. In this context, this thesis presents new approaches based on deep learning, as well as the principles for designing architectures adapted to the characteristics of genomic data. The use of convolution layers and attention mechanisms allows the presented networks to be invariant to the sampled haplotypes' permutations and to adapt to data of different dimensions (number of haplotypes and polymorphism sites). Experiments conducted on simulated data demonstrate the efficiency of these approaches by comparing them to more classical network architectures, as well as to state-of-the-art methods. Moreover, coupling neural networks with some methods already proven in population genetics, such as the *approximate Bayesian computation*, improves the results and combines their advantages. The practicality of neural networks for demographic inference is tested on whole genome sequence data from real populations of *Bos taurus* and *Homo sapiens*. Finally, the scenarios obtained are compared with current knowledge of the demographic history of these populations.

Contents

List of Figures

List of Tables

Introduction

The study of population histories is an interdisciplinary field of research that involves archaeology, palaeontology, linguistic or cultural history studies to reconstruct the demographic evolution and spatio-temporal dynamic of populations. Although the latter mainly focuses on human populations, the reconstruction of demographic histories can also be tackled from another angle by studying the genetic variation at population scales. Indeed, genetic variation is a product of multiple phenomenons, including demographic events such as admixtures, population split or size changes. Therefore, one can reverse this process by mapping genetic variation back to demographic history. Thanks to the advances made in genomics since the first complete sequencing of a human genome (Collins et al., 2004), they are now more and more whole genome sequencing (WGS) datasets including several individuals from the same population. This type of datasets primarily focused on human populations, with the 1000 Genomes Project (Consortium et al., 2010) being one of the first large scale WGS collection which targeted seven populations from four continents originally sequenced with an average coverage of $3.6\times$ (it also includes six samples with $42\times$ coverage). This dataset was later extended for improved world coverage and data quality (Bergström et al., 2020; Consortium et al., 2015; Leitsalu et al., 2014; Mallick et al., 2016; Pagani et al., 2016). Datasets of non-human species are also available, such as *Bos taurus* with the 1000 Bull Genomes Project (Daetwyler et al., 2014) or chimpanzees and gorillas with the Great Apes Genome Project (Prado-Martinez et al., 2013) and many other species that were previously difficult to study when relying solely on archaeological and historical data. Therefore, methods based on genetic variation open the path of demographic reconstruction to all species, from bacteria to vertebrates.

They are many cases where inferring the past demography of population is useful. For instance, as changes in population sizes are often related to specific events, it helps to support and date discoveries made by archaeologists and historians such as cultural transitions, ancient migrations, climatic, geological or disease outbreak events. In conservation biology, such information could help to identify whether a species is endangered thanks to the sequencing of a very small subset of its individuals (Kerdoncuff et al., 2020) and furthermore could help to determine if a decline was caused by anthropogenic activities or natural factors (Lorenzen et al., 2011). In addition, knowing more precisely the demographic history of a population enables to build more realistic neutral models, and thus to infer more accurately non-neutral signals due to the process of selection. In the field of health, precisely inferring the demographic histories of pathogens could help to monitor the impact of health policies. From a different point of view, processing whole genome sequence datasets including multiple individuals is a very challenging task itself. Therefore, developing methods able to efficiently process

this large quantity of data benefits not only to demographic inference but also to the myriad of problematics in population genetics and genomics. For instance, methods for predicting binding sites, protein structure, or biomarkers in precision medicine, all rely on similar datasets and are only a subset of the domains that could benefit from developing new methods treating whole genome sequences.

Demographic inference from genomic data is difficult since evolution is a stochastic process and only a few present-day individuals are sequenced. Mutation rate, recombination rate, genetic drift and natural selection also drive genetic variation in populations, which, added to the phasing, sequencing or genotyping errors that can arise during sample collection, introduce a lot of noise in the data. Thus, the signal left by the demographic history into the genome is very blurred, which makes it even harder to decipher between different demographic histories. Moreover, previous methods that used to rely on mtDNA, microsatellite data or SNP chip with a limited amount of markers are difficult to scale to data produced by next generation sequencing (NGS) technics and more recent SNP chips with millions of markers. The important number of dimensions in these data makes them hard to handle computationally and statistically for most methods because of the curse of dimensionality (Blum, 2010).

Thankfully, a new branch of machine learning has shown impressive results in many fields of science, solving complex problems based on high-dimensional data. Called deep learning, these methods share the common feature of being able to automatically learn complex non-linear projections of the data. Based on the concept of artificial neural network (ANN), these algorithms are easily scalable and flexible, as they can be applied to almost any task after some adaptations to its characteristics. The main objective of this thesis is to leverage the potential of deep learning to infer the past demography of populations from genomic data by focusing on the inference of population size history.

Outline

This thesis is structured around four chapters:

Chapter 1 presents the state-of-the-art of demographic inference in population genetics. It describes the evolutionary models that the community developed in order to explain how genetic variation is related to demography. It also presents the simulators that will be later used to train the deep learning architectures. These simulators exploit the evolutionary models to generate genome sequences according to demographic scenarios specified by parameters such as the mutation rate, recombination rate, effective size or selection. This chapter also includes a review of the state-of-the-art in demographic inference with methods that do not rely on deep learning (except for some very specific cases of approximate Bayesian computation (ABC)).

Chapter 2 introduces the field of deep learning and reviews its applications to genetic data, with a focus on methods answering population genetic questions. ANNs are very modular, meaning that the deep learning community has developed many ideas that can be combined to build networks tailored to the task of interest. This chapter introduces some basic concepts that are common to most ANNs such as backpropagation or neuron activations, and others designed to leverage specific data features

such as attention mechanisms. The deep learning concepts presented here have been used by methods from the state-of-the-art of deep learning applied to genomics that we reviewed in the second part of this chapter. Deep learning applications for population genetics being fairly new, this review gives a thorough overview of the tasks, data formats and ANN architectures tackled in this field, and serves as inspiration for the methods that we developed and present in the next chapter.

Chapter 3 describes the materials and methods used to develop our ANN architectures and how we trained them to tackle population size inference on real dataset. It starts by presenting the two genomic datasets of real cattle and human populations and explain how we generated their simulated counterpart in order to evaluate and train the methods tested during this thesis. Then, the chapter presents the methods we included in the baseline to evaluate the two main deep learning architectures we developed. The first one, called sequence position informed deep learning architecture (SPIDNA), tries to take into account most of the data features specific to population genetic data. The second architecture, called mixed attention SPIDNA (MixAttSPIDNA), is build upon the first and integrates multiple attention mechanisms in order to improve the overall computing capabilities of the network without losing the adaptability to the data features implemented in SPIDNA. This chapter also presents a method based on canonical correlation analysis (CCA) that has been created during this thesis to interpret ANNs in the context of demography inference. Finally, the chapter presents a python package that we developed to help the population geneticists to develop and distribute new deep learning architectures.

Chapter 4 discusses the performance of baseline methods, SPIDNA and MixAttSPIDNA on both simulated and real datasets. It also presents experiments aimed at dissecting the robustness of the designed ANNs, the hyperparameters' influence and the optimization process. It finally concludes on the positive and negative impacts of the mechanisms introduced in our architectures and methods used to train them.

The conclusion of this thesis includes perspectives that could be explored to improve further the ANNs' accuracy in the context of demographic inference and to improve their interpretability.

To summarize, the main contributions of this thesis are:

- a review of the state-of-the-art of deep learning for demographic inference,
- the development of the SPIDNA architecture, its benchmark to other methods and application to a dataset of cattle genomic data (Sanchez et al., 2021a),
- the development of the MixAttSPIDNA architecture, its comparison to SPIDNA and application to the HGDP dataset of Human genomic data,
- the elaboration of a set of guidelines to build ANNs tailored to genomic data and
- the release of a package to facilitate the usage of deep learning by the population genetic community (Sanchez et al., 2021a).

Chapter 1
Demographic inference

Contents

Population genetics is a subfield of evolutionary biology that seeks to understand how evolution shapes genetic variation at the scale of a population. The emergence of this discipline can be traced back to the early 20th century, decades before the discovery of the DNA structure in the forties and the availability of large scale genomic data of today. Therefore, this domain used to rely primarily on mathematics with the goal of finding models that describe the genetic variation of a population. One of the first evolutionary model has been published independently by Hardy (1908) and Weinberg (1908). It describes the allele frequencies of a biallelic locus in a panmictic population of infinite size, without mutation or selection. This model shows the counterintuitive notion that two allele frequencies reach an equilibrium (the Hardy-Weinberg equilibrium) under these assumptions, independently of their dominant or recessive state. Later, Fisher (1958); Wright (1931) proposed a model with mutations in a finite size population, making this model one of the first to take into account an evolutionary force and a demographic parameter, opening the path to more realistic models that relax these simplifying assumptions. Moran (1958) introduced a model with overlapping generations and asexual reproduction, where at each step an individual is cloned and another one dies. Similarly, Kimura (1964) introduced a model based on partial differential equations to highlight genetic drift and developed the neutral theory of molecular evolution. This theory argues that genetic drift is the main evolutionary force that explains

variation, as opposed to natural selection. These models are prospective, meaning that population changes are modelled by moving forward in time.

As genetic data of present-day individuals became available, it marked a turning point because a new evolutionary model was needed to understand, retrospectively, how evolution shaped genomes. Hence, a model called the coalescent has been introduced by Kingman (1982). The principal idea of this model is to start from a sample of present-day individuals and trace their lineages backward in time until all lineages coalesce (i.e., until they merge in a single lineage) to the most recent common ancestor (MRCA). The quantities of interest here are the coalescence times, i.e., the number of generations that two or more individuals took to coalesce. For simulation purpose, this model is often more computationally efficient than prospective ones, because it only takes into account the lineages that lead to present-day individuals sampled, as illustrated in Figure 1.1. The coalescent has been complexified through the years in order to relax some demographic assumptions to allow for population size changes and population structure, or introduce other evolutionary forces such as recombination.

One of the main goal of population genetics is to study the genetic variations of a population in order to infer demographic parameter values describing the population size changes, their structuring, admixture and migration events. This first chapter gives an intuition of the link between genetic variation and population demography, and presents the problematic of demographic inference. Then, it shows how previously mentioned evolutionary models are integrated into methods for demographic inference. Section 1.2 presents the sequentially Markovian coalescent, an evolutionary model that approximates the coalescent via a hidden Markov chain, on which numerous inference methods rely. Section 1.3 presents methods based on summary statistics of the genomic data. Finally, this chapter will present the data simulators used during this thesis.

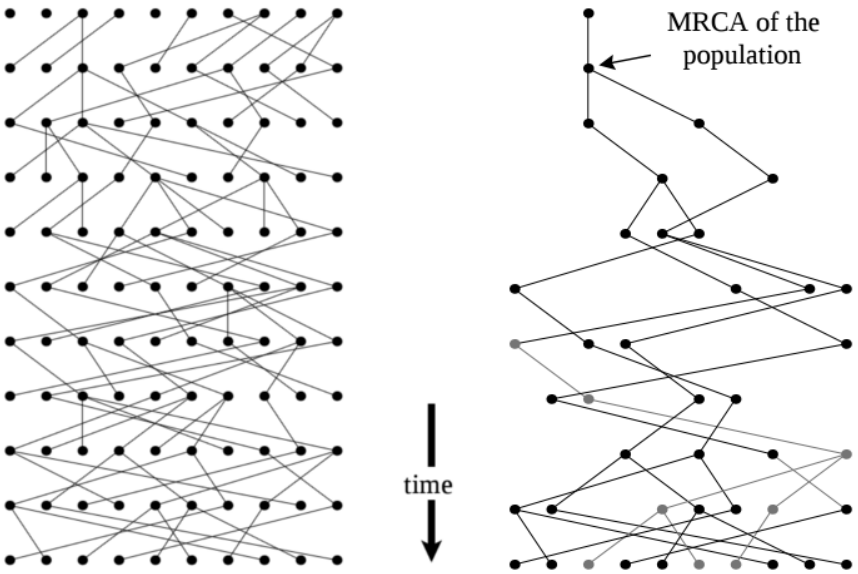

Figure 1.1: **Genealogy of the same population modelled prospectively (left) and retrospectively (right).** Figure from Nordborg (2001). MRCA stands for most recent common ancestor of the individuals of the last generation.

1.1 Insights into the demographic inference problem

As stated in the introduction, the main challenge of demographic inference in population genetics is to infer the parameters values describing a population demographic history from its genomic data. The demography of a population is described by changes in its size, structure, and the occurrence of admixture or migration events. The models cited earlier depict the effects of these demographic parameters on the evolution of genomic variation. Therefore, it is possible to reverse this process in the light of evolutionary models to infer these parameter values by studying the genomic variation. Before defining the inference problem studied during this thesis, let us better understand the link between demography and genomic variation through some examples.

1.1.1 Examples of demographic history effects on genomic variation

This section gives an intuition about why genetic variation within a population carry signals of the past demography. The first example shows how an expansion increases the genomic diversity in the coalescent evolutionary model. The second example of a bottleneck shows the reduction of the population size decrease its genetic diversity. The third example shows how an event of admixture can be dated when the recombination rate per generation is known.

Figure 1.2 illustrates how coalescent time is related to the size of the population. The population with a growing rate N_a/N_c has more coalescence events than expected in a constant size case, when the population is small (close to N_a) and the branches (i.e., the coalescence times) are shorter during this period. Intuitively, two lineages are more likely to coalesce at a time when the population is small. The total sum of the lineage branches being longer for the expansion scenario than a scenario with a small constant population size, new mutations have more chances to occur, resulting in a higher genomic diversity in terms of number of polymorphism sites.

Now consider the scenario of a population with a high genetic diversity in the past, that then undergoes a bottleneck, i.e., a sharp reduction of the population size, and finally a size expansion. During the bottleneck, the genetic diversity of the population is reduced and even if the population recovers its previous size during the expansion phase, it will not quickly recover its level of genetic diversity. From that, we can infer that a large population with small genetic diversity is likely to have encountered a bottleneck in the past followed by an expansion (Gattepaille et al., 2013).

Another example is how an event of admixture (e.g., the introduction of individuals of different ancestry (migrants) into the studied population) can be dated from genetic data. Right after a migrant individual enters a population, his first generation offspring will carry one entire chromosome originating from each population, leading to long genomic regions that can be assigned to a single ancestry. However, recombination occurs at each new generation, breaking apart these regions and mixing segments of different ancestries. Analysing the length of the admixed regions of the genome informs about the number of recombinations that happened since the admixture, and thus allows to approximate the date of the event (assuming admixture happened in a

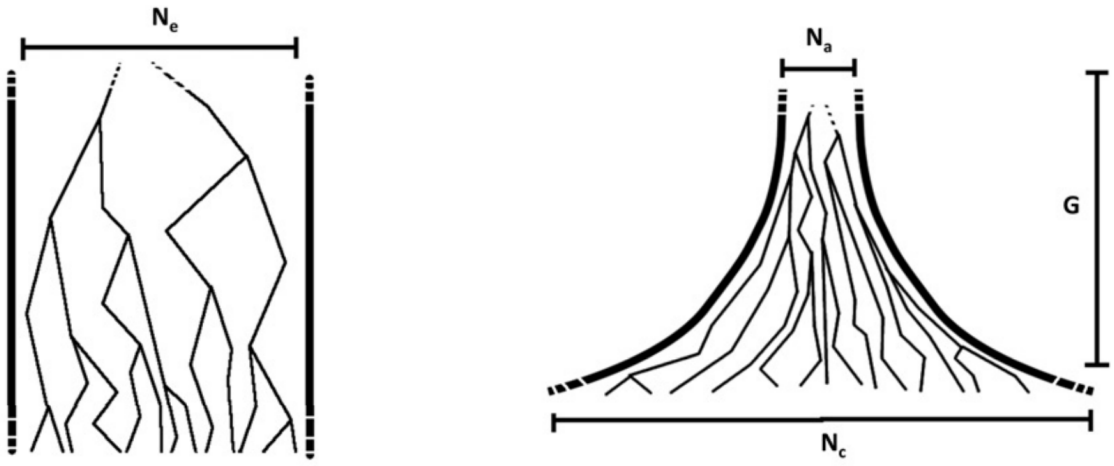

Figure 1.2: **Effect of expansion on coalescence.** Left: scenario with constant population size N_e. Right: scenario with expanding population size at a growing rate $^{N_a}/_{N_c}$ during G generations. Figure from Palamara et al. (2012).

single pulse) (Gravel, 2012; Liang and Nielsen, 2014; Verdu et al., 2014).

Yet, following the described logic to infer demographic events is valid only in the case of an idealized population, where no other process disturbs the signal left by the population size changes in the two first examples and the admixture in the third. In these cases, it is fairly easy to derive the parameter values describing these demographic events from present-day data. However, real populations can undergo many demographic events and changes in the evolutionary process at the same time. For instance, demographic events such as the presence of complex population size changes, multiple events of admixture, migrations or non-random mating can completely distort the demographic parameters inferred by simple derivations. Moreover, as pinpointed in the introduction, changes in mutation rate, recombination rate, natural selection and errors added during the data collection are other causes that break the simplest models to the point that inferred parameters are outside the acceptable margin of error.

1.1.2 Definition of the demographic inference problem

When inferring past demography, one can try to infer all demographic parameters at once, or choose to focus either on the inference of the population size history or its structure when considering no admixture or migration events. As shown by previous studies, these two families of demographic parameters are particularly difficult to disentangle for most methods, as the effects on genomic variation of one also translate very well into the other (Mazet et al., 2016). This thesis focuses solely on the inference of population size histories of a panmictic population and does not consider the split, merging and admixture events that populations may undergo, but we will discuss in the next sections methods doing the opposite, i.e., ignoring population size changes and inferring population structure through time. Another point is that inference can be performed either as a regression task or a classification task. Classification is particularly useful when the population demography studied is known enough to formulate

competitive hypotheses. For instance in human populations, if it is already know that two populations had encounter in the past thanks to archaeological evidences, one can seek for the classification between different scenarios including admixture, non-admixture or replacement of one population by the other. Here, each scenario would be modelled under a demographic model with predefined parameter values, and the classification task goal would seek which scenario is the most likely based on the real population data.

During this thesis, we will focus on a regression task with the goal of inferring detailed population size histories. These histories will be represented by 21 population sizes parameters at fixed time steps. We chose to perform regression of a fairly complex demographic scenario over classification because little prior information is known about the population, and it covers a much greater range of possible scenarios. Hybrid approaches can also be considered, with a first classification task inferring a general scenario (e.g., the presence of an expansion or not) and a second regression task aimed at refining the general scenario inferred (e.g., finding the date and rate of expansion if that is what has been inferred by the classification).

We denote the demographic parameters of interest θ. Regression methods seek either for the set of θ that maximizes the likelihood $p(X \mid \theta)$ of the observed data X, or the posterior distribution $p(\theta \mid X)$ of θ. The data X are the observed (potentially preprocessed) sequencing data. The real evolutionary process that led to the observed data is approximated by a model such as the Wright-Fisher or the coalescent ones, which will include demographic information. However, these evolutionary models are complex probabilistic models, also referred to as implicit models (Cranmer et al., 2020), where the likelihood is often not explicitly defined, an approximation or the likelihood under a simpler model. The real likelihood is hardly tractable because it would require to integrate the posterior probability over all the possible mapping from θ to X in the model, which is not possible because these mapping are too numerous in the case of complex probabilistic model. Moreover, the demographic parameter inferred can be very different than the true values because of the assumptions made by the model. In the case of population sizes, the inferred values are called "effective sizes" by opposition to the true "census sizes". We often consider the effective population sizes to be linearly proportional to the census population sizes, which allows drawing conclusions on the overall dynamic of the population size history.

The next sections present three different families of methods for demographic inference: methods based on an approximation of the coalescent with hidden Markov models, methods using the likelihood computed on summary statistics (summarizing the data) rather than the complete observed data X and finally, the approximate Bayesian computation, a method that can approximate the posterior distribution, but also requires the computation of summary statistics.

1.2 Sequentially Markov coalescent (SMC) methods

The sequential Markov coalescent (McVean and Cardin, 2005) is an evolutionary model derived from the sequential coalescent (Wiuf and Hein, 1999), itself derived from the coalescent with recombination (Hudson, 1983). It allows demographic inference by

adding few simplifications over the sequential coalescent. The main idea behind these models is that recombination events of a population sample can be represented as transitions between the marginal genealogies along the genome. To understand this concept, we can take a look at a type of representation called the ancestral recombination graphs (ARG) (Figure 1.3) where each node represents either a coalescence or a recombination event. The sequential coalescent associates each locus of the sample alignment to a genealogy that is embedded in the ARG. A recombination event in the ARG can translate into a change of genealogy along the genome. After inferring the marginal genealogies of each locus, a distribution of the coalescence times can be computed from them and finally transformed into demographic parameters such as the effective population sizes.

Inference under the coalescent with recombination is particularly difficult because the state-space of possible ARG is huge. Indeed, the number of recombination is unbounded, and they can occur on haplotype segments that are not present in the sampled haplotypes. Moreover, there are usually too few mutations to characterize locally a precise genealogy. Therefore, there are numerous possible ARGs, each contributing a little to the likelihood. SMC and its derivatives reduce this state-space by introducing the simplifying assumption that recombination events only occur in segments that lead to coalescence events in the sample. In other words, an MRCA haplotype can only contribute to one continuous fragment in the sequences sampled (in reality, multiple fragments could have the same MRCA, but it is unlikely after many recombinations). The changes in the genealogy are represented as transitions along the genome sequence that follows a Markov chain dynamic, meaning that the probability of a genealogy at a given position now depends only on the previous genealogy (directly on its left in the sequence). Thus, a likelihood approximation can be computed by using hidden Markov model (HMM) inference methods. In the HMM framework of the SMC, the hidden states are the marginal genealogies, which are simplified into coalescence times between haplotypes pairs by subsequent SMC based inference methods. The observed values are the sample states at each locus.

The SMC and its improved version SMC' (Marjoram and Wall, 2006) are the underlying model for many inference methods including PSMC, MSMC and SMC++ (Li and Durbin, 2011; Schiffels and Durbin, 2014; Terhorst et al., 2017). These methods differ by the way they construct their hidden and observed states. For instance, the pairwise sequential Markov coalescent (PSMC) uses two haplotypes (i.e., it only requires the sequence of a diploid individual) and its possible hidden states are the coalescence times of each locus. PSMC is limited when it comes to inferring recent effective population sizes because two haplotypes have few recent coalescent events, and thus less information about recent population demography. It is difficult to leverage information from more than two haplotypes due to the difficulty of expressing the marginal genealogies and coalescence times of multiple haplotypes. The multiple sequential Markov coalescent (MSMC) circumvents this issue by computing all the pairwise recombination time between multiple sample haplotypes and retaining the most recent one. Another method called SMC++ also leverages the information in multiple haplotypes. It uses an observed state similar to PSMC (i.e., the alleles of two haplotypes), but adds to it a summary statistic called the site frequency spectrum (SFS) of the remaining n-2 haplotypes.

These SMC based methods have the advantage of using the information along the

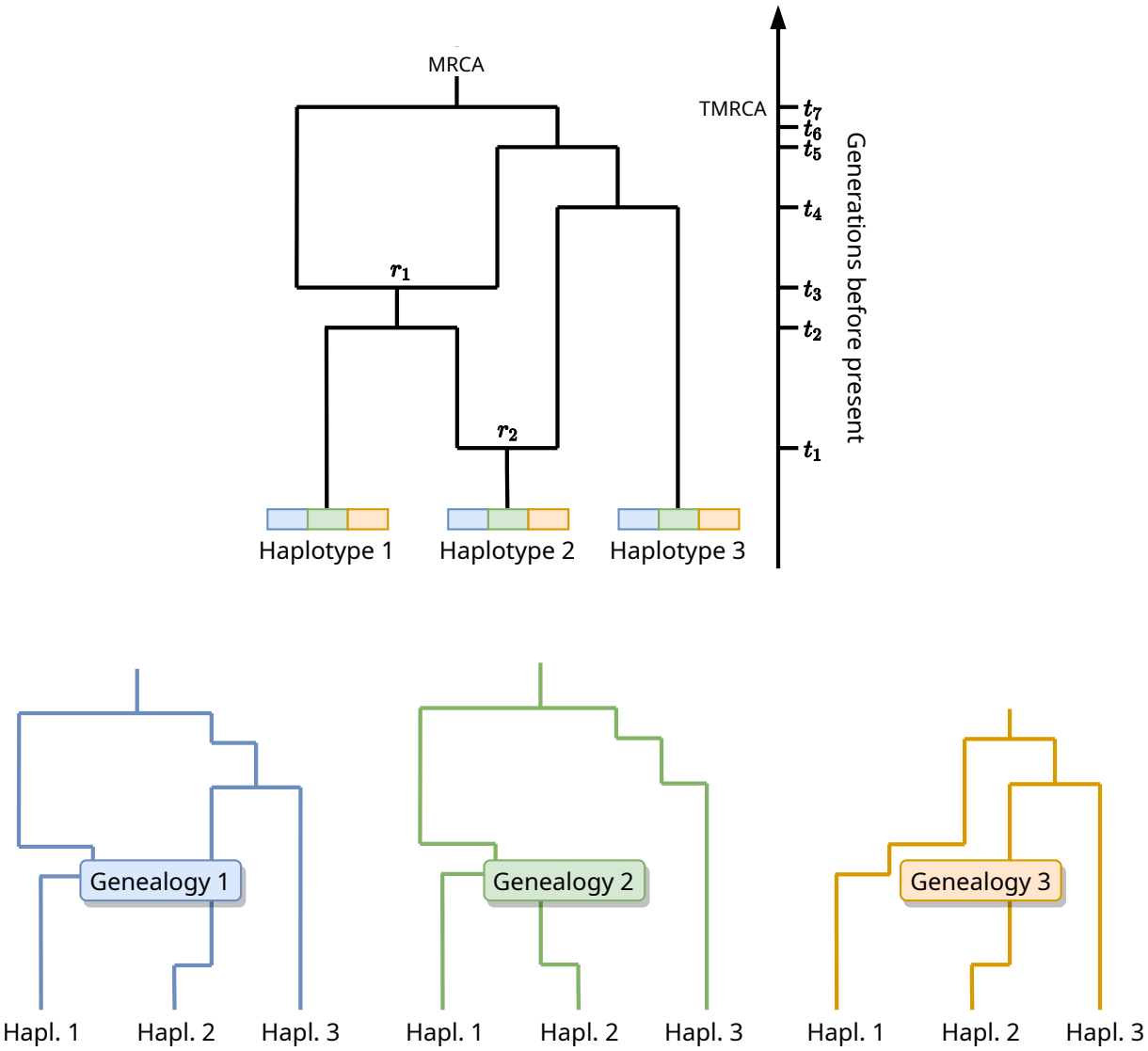

Figure 1.3: **Example of ancestral recombination graph (ARG) between three haplotypes with the marginal genealogies associated.** Each node of the ARG corresponds either a coalescence event or a recombination event (nodes r_1 and r_2). The lineages of the haplotypes are represented until they all coalesce to the most recent common ancestor (MRCA). Each marginal genealogy (genealogy 1, 2 and 3) represents the coalescence events between the non-recombined subsequences of the same colour.

sequence (usually by windows of 100 bp) and leverage linkage information between sites. However, in addition to the simplifications discussed earlier, these methods have other disadvantages due to the approximations introduced to limit their computational cost. For instance, the possible hidden states (i.e., the possible coalescence times at a loci) are discretized, which may reduce the accuracy. Moreover, they use phased data, which require greater sequencing coverage. Therefore, researches in population genetics have developed other methods based on different types of input data and evolutionary models. The next sections introduce them, alongside their strengths and weaknesses.

1.3 Summary statistics based inference

Instead of processing genomes sequentially like SMC-HMM based methods, summary statistics based methods circumvent the high dimensionality of genomic data by focusing on a subset of statistics computed from the sequences. Many of such methods have been developed by the community in population genetics through the years, and this section highlights some of them to illustrate the different ways summary statistics are used for demographic inference.

1.3.1 Inference from site frequency spectrum (SFS)

We saw in the previous sections that, compared to other SMC based methods, SMC++ uses the additional information of the site frequency spectrum (SFS) to integrate data from more than two haplotypes. The SFS is the distribution of the number of n-ton (singleton, doubleton, tripleton, etc...) polymorphic sites in the population sample. ∂a∂i from Gutenkunst et al. (2009) is an example of a method that utilizes such statistics to infer effective sizes, splitting events and migrations between up to three populations. More precisely, this method seeks for the demographic parameter values that maximizes the likelihood of the multi-population SFS under a diffusion model of evolution. The multi-population SFS, which is the joint site frequency spectrum (SFS) between multiple populations, is a matrix with as many dimension than populations in the sample. Each element of the matrix represents the shared number of polymorphic sites between the populations. For instance, the $[2, 0]$ entry of a 2 dimensional matrix records the number of polymorphic sites where the derived allele is present twice (i.e., a doubleton) in the first population, but not present in the second population. In the same way as the SFS, the multi-population SFS contains all the information of the data of multiple populations if the polymorphic sites are completely independent, and it scales easily to the full genome length.

The stairway plot (Liu and Fu, 2015) is another method that optimizes observed SFS likelihood using a genetic algorithm. The algorithm searches for the population scaled mutation rates that are then converted into population sizes at different time point. This method showed better estimations of recent effective population sizes than PSMC over predefined simulated scenarios.

The fastsimcoal2 algorithm proposed by Excoffier et al. (2013, 2021) approximates the likelihood of the SFS by using simulations. Here, the demographic parameters are

optimized thanks to an extension of the expectation-maximization (EM) algorithm. At each cycle, the algorithm generates simulations from a set of demographic parameters and computes the likelihood of the demographic model (by calculating the probability of the observed SFS based on the simulated one). This likelihood is then used by the EM algorithm to compute the next parameter values. The procedure usually simulates 100,000 SFS at each step for 20 to 40 cycles. Between 20 and 40 independent procedures are launched in parallel with different starting demographic parameter values in order to better explore the parameter space and avoid local optima.

1.3.2 Inference from identity by state (IBS) and identity by descent (IBD)

Identity by state (IBS) and identity by descent (IBD) are two other types of summary statistics that have the advantage of taking into account linkage information between sites. IBS describes sequence fragments that are identical, whereas IBD describes sequence fragments that are similar because they are inherited from a common ancestor without any recombination events breaking the lineage from this ancestor. IBD does not always imply identical fragments because new mutations can still occur after the MRCA. Following the coalescent, the length of IDB fragments can be translated into coalescence times to finally infer population sizes. Intuitively, small populations have shorter coalescence times and thus, fewer recombination events that can break up shared sequences, resulting in longer IBD fragments. One caveat of IBD based methods is that they require a first inference step to identify the IBD fragments before performing the demographic inference itself, contrary to IBS fragments that can be computed directly from the observed data (if their quality permits).

Harris and Nielsen (2013) thus compute the likelihood of the IBS segment lengths based on the SMC evolutionary model and optimize the demographic parameters of interest with quasi-Newton BFGS algorithm. This strategy enabled them to infer the parameters of a demographic model including population size changes, divergence events and admixture pulses. They obtained accurate predictions for simulated scenarios on which ∂a∂i was not able to converge, but had trouble leveraging long IBS tracts in applications to real data, as those were interrupted by sequencing errors.

In their study, Browning and Browning (2015) focused on the inference of recent effective population sizes based on IBD fragments longer than a particular threshold. Rather than having a direct expression of the likelihood, they express the relationship between effective population sizes and IDB fragments based on a Wright-Fisher discrete-generation evolutionary model. Effective population size values are updated through an expectation-maximization (EM) algorithm by computing iteratively the estimated and expected amount of IBD from a set of effective population sizes.

Except for fastsimcoal2 that could potentially use different simulators, the methods presented here are tied to the evolutionary model they use to express the relation between the summary statistics and the demographic parameters of interest. The next section presents the approximate Bayesian computation (ABC) that, similarly to the method underlying fastsimcoal2, is not tied to any evolutionary model, but has the advantage of being able to leverage many type of summary statistics (rather than solely the joint SFS).

1.3.3 Approximate Bayesian computation (ABC)

Approximate Bayesian statistical inference is a method that allows to estimate the posterior distribution of model parameters given a set of observed data. To better understand how this method works, let us remind the Bayes' theorem:

$$p(\theta \mid X) = \frac{p(X \mid \theta)p(\theta)}{p(X)} \tag{1.1}$$

where $p(\theta \mid X)$ is the posterior of parameter θ (i.e., the probability distribution of θ given the observed data X), $p(X \mid \theta)$ is the likelihood (i.e., probability distribution of the observations of the data X given θ), $p(\theta)$ is the prior knowledge about the distribution of θ without information from the data X and $p(X)$ is called the marginal likelihood, which is the distribution of all data that could be observed for all values of θ. The marginal likelihood is a normalizing constant that can be ignored because it cancels out when comparing two posterior probabilities.

As explained in Section 1.1.2 observed data X are derived from sequencing data and θ are the parameters of interest in the context of demographic inference. One caveat of ABC, is that it suffers from the curse of dimensionality (Blum, 2010) because it computes a distance in the space of X that can be high-dimensional. Simply put, data points become very sparse in the data space as the number of dimension increases, which makes them appear equidistant to each other, preventing the usage of a relevant distance metric. Because sequencing data are typically high-dimensional, they are often converted into a smaller number of commonly used summary statistics with a function s to give a vector of summary statistics observed $\vec{s^\star}$. A set of parameters θ_i are randomly drawn from the prior $p(\theta)$ and used to generate simulated data X_i. Then, a vector of simulated summary statistics $\vec{s_i}$ is computed for each simulated data X_i thanks to the function s.

The ABC algorithm allows finding an approximation of the posterior distribution $p(\theta \mid X)$ of the demographic parameters θ from the observed data X with the following formula:

$$p(\theta \mid X) \approx p(\theta_i \mid d(\vec{s_i}, \vec{s^\star}) < \epsilon) \tag{1.2}$$

with d a distance measure (usually the Euclidean distance) and ϵ the tolerance threshold. In other words, the posterior is estimated by the distribution of demographic parameters θ_i for which the corresponding summary statistics $\vec{s_i}$ have a distance $d(\vec{s_i}, \vec{s^\star})$ inferior to the tolerance threshold ϵ. Before constituting the estimated posterior, a step of correction can be applied by fitting a regression model locally in order to improve the inference of the demographic parameters in the vicinity of the observed data and help to cope with the dimensionality of the set of summary statistics. This regression can be performed with, for instance, lasso regression, ridge regression or a simple artificial neural network (Boitard et al., 2016b). ABC steps are represented in Figure 1.4.

The first applications of this method for demographic inference date from the late nineties. Tavaré et al. (1997) used it to infer the coalescence times from sequence data and Pritchard et al. (1999) used the variations of eight human Y chromosome microsatellite loci summarized into 3 summary statistics to infer the parameters of a model of exponential growth. Nowadays, researchers are addressing more and more

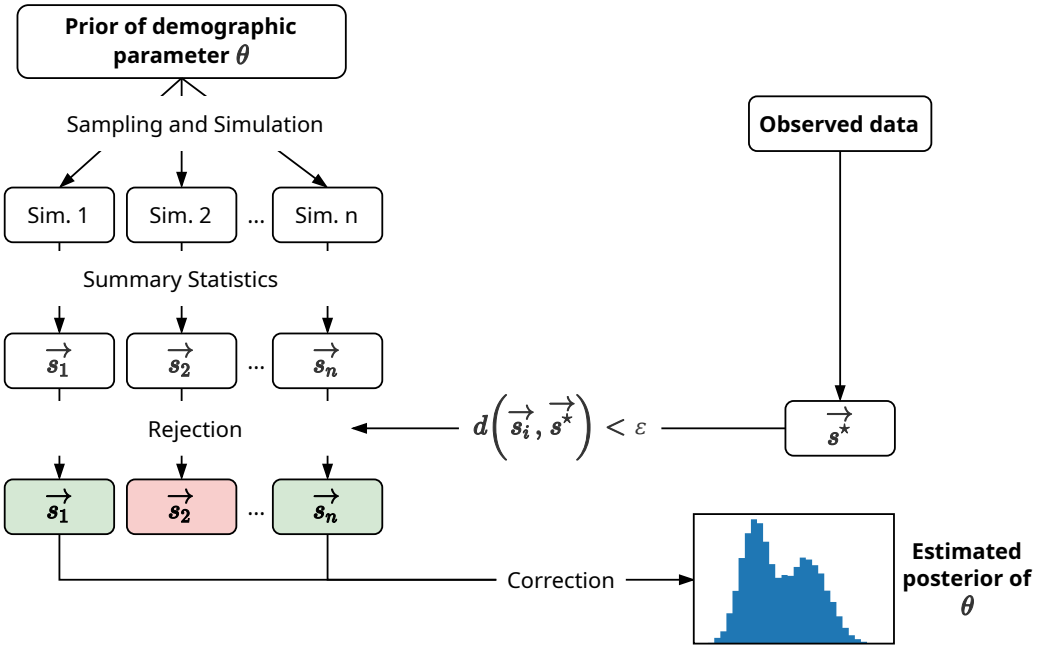

Figure 1.4: **Diagram of the approximate Bayesian computation algorithm (ABC).** Demographic parameters are randomly drawn from the prior $p(\theta)$ and fed to a simulator based on an evolutionary model. Simulated and observed data are then converted into summary statistics. Then simulated summary statistics $\vec{s_i}$ that are too far from the observed summary statistics $\vec{s^\star}$ according to the distance measure d and the tolerance threshold ϵ are rejected. The demographic parameter associated to the remaining simulated summary statistic can then pass through a step of correction to finally constitute the estimated posterior of the observed data.

complex tasks thanks to the availability of WGS data which are summarized by numerous statistics inspired by population genetic theory in order to minimize the information lost from the sequencing data. Summary statistics commonly used are the site frequency spectrum (SFS) and its summaries (e.g., Tajima D), linkage disequilibrium (LD) and statistics based on shared segments that are identical-by-state (IBS) or identical-by-descent (IBD) (Boitard et al., 2016b; Gladstein and Hammer, 2019; Jay et al., 2019; Sheehan and Song, 2016; Smith and Flaxman, 2019). Yet, they are not guaranteed to be sufficient and including too many statistics can impact the performance of standard ABC by falling back to the curse of dimensionality. An active research topic in the ABC community is thus the development of methods addressing this issue by (i) selecting the best subset of summary statistics according to some information-based criteria, (ii) integrating machine learning steps into ABC to handle a larger number of summary statistics (e.g., kernel methods, random forests), (iii) constructing summary statistics using linear and non-linear models based on candidate statistics or on the original data when feasible (Aeschbacher et al., 2012; Blum et al., 2013; Fearnhead and Prangle, 2012; Jiang et al., 2017; Nakagome et al., 2013; Raynal et al., 2018).

Therefore, ABC has some drawbacks: it is sensitive to the curse of dimensionality and cannot handle directly raw genomic data, it needs many simulated data due to its rejection algorithm, it is difficult to interpret its output in terms of which statistics were the most informative for each demographic descriptor, and it does not naturally handle correlated or weakly informative summary statistics, which can add noise to the data (Sheehan and Song, 2016) (to the exception of ABC random forests (Raynal et al., 2019), which on the other hand lose the ability to estimate posteriors). However, its ability to estimate posteriors and therefore, to compute credible intervals, gives it a great advantage over other methods, such as most deep learning frameworks. During this thesis, we used ABC as a baseline to compare to the architectures we developed. The results of this comparison should also give us an idea of how our methods should perform against other methods already tested against ABC, such as MSMC (Boitard et al., 2016b). Finally, we also combined ABC to our SPIDNA architecture in order to benefit from the advantages of the two methods, as explained in Section 3.3.3.

1.4 Simulators

Numerous simulators have been developed and can be used with inference methods that are not tied to any underlying evolutionary model, such as the ABC. They are often categorized either into forward simulators or backward simulators. Forward simulations, are based on prospective models, meaning that all individuals of a population are usually followed one generation after the other, while backward simulations follow the ancestors of only a small subset of the population. For instance, the SLiM program (Haller and Messer, 2019) proposes two different forward simulators, one based on the Wright-Fisher model and one non-Wright-Fisher (nonWF). The nonWF model is particularly flexible because it controls mating at the individual level. Thus, it can relax many of Wright-Fisher assumptions to allow introducing new mating, recombination or selection mechanisms, and more easily adapt the model to non-model organisms (Cury et al., 2021). However, these models are particularly time-consuming compared

to backward simulators which are preferred for inference methods that require a lot of simulated data, such as ABC and the deep learning methods developed during this thesis. We will see in Chapter 3 that we used three similar coalescent based simulators, ms, msms and msprime (Ewing and Hermisson, 2010; Hudson, 2004; Kelleher et al., 2016), to test the robustness of our method to change of simulator (see Section 4.2.2). We choose to use principally msprime to perform simulations to train and evaluate our methods because it is the most computationally efficient of the three.

Demographic inference methods based on simulations require generating a lot of simulated data under different scenarios that are drawn from prior distributions. As explained in Section 3.1, we designed our priors to be as close as possible to the knowledge on the two real datasets studied during this thesis.

1.5 Chapter conclusion

This chapter gave a brief overview of the state-of-the-art in demographic inference from genomic data. The different methods rely on evolutionary models that describe the influence of demography on population genetic variations with the idea of finding the demographic parameter values that maximize the likelihood of some observed data. These methods are able to handle genomic data either by analysing polymorphic sites or by reducing the data into relevant summary statistics. By comparing these methods, we can see that there is often a tradeoff between the complexity of the evolutionary model, the complexity of the demographic model, the computational efficiency and the amount of information used from the data. For instance, ABC can be used with any evolutionary model that allows for simulation, but is not particularly efficient as it requires a lot of simulations, especially when testing a wide variety of demographic models and parameters. Moreover, it also requires to carefully choose the summary statistics used in order to lose the minimum of information from the genomic data. On the other hand, we saw that some methods, such as SMC, despite analysing the full sequences, were tied to evolutionary models. Hence, we can imagine some specifications for the design of a new demographic inference method. It should be preferably agnostic to the demographic and evolutionary models, able to handle directly genomic data and have competitive prediction errors while being computationally efficient. The next chapter presents the deep learning framework and why it is a good candidate to answer these requests.

Chapter 2
Deep learning for genomic data

Contents

The main challenge of this thesis is to leverage the recent advance in Deep Learning to solve the population genetic problem of inferring past demography. This chapter first introduces deep learning and describes the fundamental building blocks of artificial neural networks (ANN) by focusing on one of the most basic ANN: the multilayer perceptron (MLP). Then, the training phase and other deep learning concepts relevant for the ANNs developed through this thesis are introduced. Finally, this chapter discusses the various applications of deep learning to genomic data with emphasis on the field of population genetics.

2.1 Introduction to deep learning

The history of deep learning can be traced back to the first models of an artificial neuron developed in the forties, which has since led to increasingly more complex networks of these artificial neurons through the years. But it is not until the development of back-propagation, a simple scheme of training based on chain rule, and the increase of computational power and availability of large datasets that ANNs became powerful enough

to solve real world tasks such as handwritten digit recognition (LeCun et al., 1989). A new deep learning craze started in 2012, after Krizhevsky et al. (2012) achieve unprecedented results during the ImageNet LSVRC-2010 image recognition contest. Fuelled by the availability of specialized hardware such as graphical processing units (GPUs) and the development of dedicated deep learning libraries, this craze has led to numerous applications of deep learning in almost all branches of experimental sciences and inspired the advancement of deep learning theory.

The deep learning family of methods consists in a large set of algorithms that share the same fundamental features: numerous learnable parameters organized in nested functions called layers, an evaluation function defined to solve a specific task and an optimization procedure aimed at finding the best parameter values over a set of data called the training set. In other words, an ANN is a complex function with learnable parameters that can be automatically optimized in order to approximate a target function that solves a task. Artificial neural networks are trained iteratively as each data sample from the training set is fed to the network, then its outputs are evaluated to guide the optimization of the learnable parameters. In the best case scenario, these two steps are repeated until convergence to an optimum, but in reality a lot of factors can deviate the training from this goal, and it is the work of the deep learning practitioner to find the right architecture and settings that yield to a good solution.

2.1.1 Multilayer perceptron

This section presents one of the most basic types of deep learning architecture which is called multilayer perceptron (MLP) or fully-connected network. Presenting this simple network will help to define most of the building blocks of deep learning architectures and how the general optimization process of deep learning works.

MLP architecture

The multilayer perceptron (Figure 2.1) is an ANN where each neuron (also referred as "hidden units") is connected to all the neurons of the upper and lower layer, hence the name of fully-connected (or dense) layers. The first layer of this model, the input layer, is a vector $\vec{x}$ representing one data point of the dataset. After the input layer, several hidden layers are added sequentially. The neurons activations of the hidden layer l of size N^l are denoted as a vector $\vec{a}^l \in \mathbb{R}^{N^l}$. The MLP has a total of L hidden layers. The weights between layers l of size N^l and $l+1$ of size N^{l+1} are denoted by the matrix $W^l \in \mathbb{R}^{N^{l+1} \times N^l}$. A bias vector $\vec{b}$ is added at each hidden layer to allow the network to shift the activation and thus, increases the capacity of the network to approximate functions. Simple non-linear functions denoted σ are inserted between layers to introduce non-linearities inside the network. The first layer thus computes the function $\vec{a}^1 = \sigma(W^1\vec{x} + \vec{b}^1)$ and each other layer computes a function of the form $\vec{a}^l = \sigma(W^l\vec{z}^{l-1} + \vec{b}^l)$. The last layer outputs the network predictions denoted by the vector $h_{W,\vec{b}}(\vec{a}^{L-1}) = \vec{\hat{y}} \in \mathbb{R}^P$. This set of weights $W = \{W^1, W^2 \cdots W^L\}$ and biases $\vec{b} = \{\vec{b}^1, \vec{b}^2, \cdots \vec{b}^L\}$ are optimized during training and therefore are part of the learnable parameters.

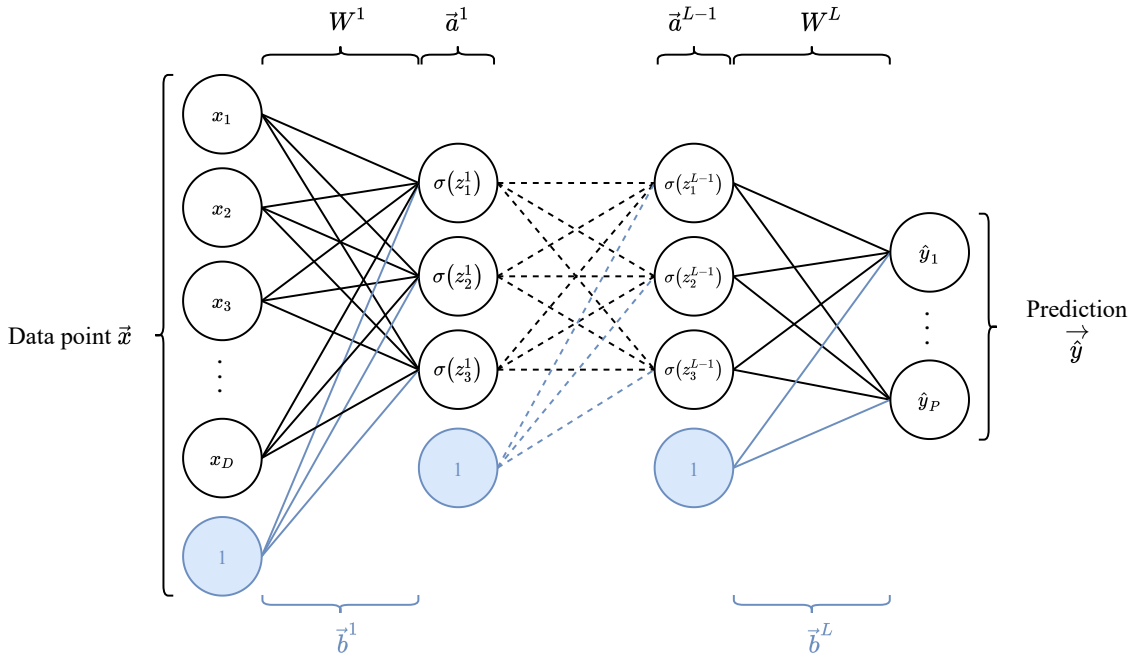

Figure 2.1: **Diagram representing a multilayer perceptron (MLP).** A data point $\vec{x}$ passes through a series of hidden layers defined by their weights W^l and bias $\vec{b}^l$. Each output $\vec{z}^l$ of an hidden layer pass through an activation function σ. Last layer output the prediction of the network $\vec{\hat{y}}$.

Activation function

Many activation functions with different proprieties have been used in Deep Learning. For instance, the Rectified Linear Unit (ReLU) is defined as:

$$\sigma_{ReLU}(z) = \begin{cases} z & \text{for} \quad z > 0 \\ 0 & \text{for} \quad z \leq 0 \end{cases} \tag{2.1}$$

ReLU has multiple advantages over other activation functions: (1) it avoids the vanishing gradient problem, (2) it gives a sparse network if weights are initialized around 0 and (3) it is efficient to compute. The vanishing gradient problem rises when the activation function has a gradient restrained to a small range (functions such as sigmoid or tanh that are asymptotically flat). These gradients are multiplied together during the backpropagation, which might lead to an extremely small (vanishing) gradient being backpropagated to the first layers. Vanishing gradients prevent changes/updates in these first layers' weights. This issue often appears when the network has many layers, i.e., when the network is deep.

Another type of activation function, the scaled exponential linear units (SELU) from Klambauer et al. (2017), mitigates the vanishing gradient by design. It is defined as:

$$\sigma_{SELU}(z) = \gamma \begin{cases} \alpha(e^z - 1) & \text{for} \quad z < 0 \\ z & \text{for} \quad z \geq 0 \end{cases} \tag{2.2}$$

with $\alpha \approx 1.673$ and $\gamma \approx 1.051$, it has been proven to keep layer activations to 0 mean and 1 variance. In practice, as it requires to carefully initialize the weights and normalize the data, simpler and less computationally expensive activation functions like ReLU combined with batch normalization (described in the next section) are often preferred.

Weight initialization

Prior to training, network weights need to be randomly initialized to ensure the smooth functioning of the optimization process and to benefit from overparametrization optimization properties. For instance, in a MLP setting, two weights with the same inputs and initialized with the same value could be updated the same way through training and remain identical, which is undesirable as it would reduce the overall expressive power of the network. Careful weight initialization is also important to avoid exploding or vanishing gradients and to start the optimization process near the optimum. To this extent, numerous initialization schemes have been developed such as the popular Kaiming He initialization (He et al., 2015) which is defined by:

$$W^l \sim \mathcal{N}(0, \frac{2}{N^{l-1}}) \tag{2.3}$$

When using ReLU activation functions, this initialization guarantees that the output variance of each layer is equal to 1. However, the assumptions made to obtain this guaranty can break with more complex architectures, e.g., for networks with residual connections, attention mechanisms or unstandardized inputs and targets. Thus, networks using these types of mechanisms are harder to train and typically use another mechanism, called batch normalization, that allows a less careful initialization.

The universal approximation theorem

The universal approximation theorem (Cybenko, 1989; Hornik, 1991) shows the computational power of ANN by proving that neural networks with bounded depth and arbitrary width can approximate any continuous function over a compact:

Theorem 1 *Let σ be any continuous sigmoidal function. Then finite sums of the form:*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j)$$

are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\epsilon > 0$, there is a sum, $G(x)$, of the above form, for which:

$$|G(x) - f(x)| < \epsilon \text{ for all } x \in I_n$$

Where ϵ can be arbitrary small, I_n is the n-dimensional unit cube and $C(I_n)$, the space of continuous functions on I_n. This theorem says that the number of units $N \in \mathbb{N}^*$, real constants $\alpha_j, y_j \in \mathbb{R}$ and real vectors $\theta_j \in \mathbb{R}^m$ exist such that G is an approximation of the target function f. It has since been extended to unbounded activation functions σ

such as ReLU and to networks with arbitrary depth but bonded width (Lin and Jegelka, 2018).

2.1.2 Training artificial neural networks

The previous section defined the MLP, the weight initialization step and its universal approximation property. This section will describe how ANN weights are actually optimized to approximate the objective function. This process called training is done iteratively by alternating between forward of the data and backward pass of the loss in the network. During the forward pass, the network makes predictions on a set of data called the training set. Then, during the backward pass, the loss gradient is backpropagated to update the network weights and biases.

Forward pass

During the forward pass, a data $\vec{x}$ is sent to the network. The activation of the neurons $\vec{a}$ of the first hidden layer $l = 1$ of a MLP is defined by:

$$\vec{a}^1 = \sigma(\vec{z}^1) = \sigma(W^1\vec{x} + b^1) \tag{2.4}$$

where the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is applied to each coordinate of its input vector independently. Then, activations of hidden layer $l > 2$ are computed as the following:

$$\vec{a}^l = \sigma(\vec{z}^l) = \sigma(W^l\vec{a}^{l-1} + b^l) \tag{2.5}$$

The output layer is computed as follows:

$$\vec{\hat{y}} = W^L\vec{a}^L + b^L \tag{2.6}$$

Here there is no activation function in the last layer because this network is aimed at solving a regression task with multiple parameters to predict, but it is also possible to adapt this layer to other tasks. For instance, by adding a softmax activation function, this last layer would be suited for a classification task where classes to predict are encoded as one-hot vectors.

Training and backpropagation

During the training phase, batches of inputs $\vec{x}$ are passed though the network and predictions $\vec{\hat{y}} = h_{W,\vec{b}}(\vec{x})$ are compared to the targeted outputs $\vec{y}$. For the regression case, the L^2 norm, also called mean squared error (MSE) can be used to measure the prediction error (also called loss):

$$J(W, \vec{b}, \vec{\hat{y}}, \vec{y}) = \frac{1}{m}\sum_{j=1}^{m}|\vec{\hat{y}}_j - \vec{y}_j|^2 \tag{2.7}$$

The method used to tune the weights of the networks developed in this thesis is Adaptive Moment Estimation (Adam) (Kingma and Ba, 2014), a method derived from

the stochastic gradient descent (SGD) (Bottou, 2010). SGD is an optimization technique that can be applied to ANN using backpropagation. We first compute $\frac{\partial J}{\partial w}$ to know how a small change of a weight $w = W^l_{i,j}$ affects the loss, and then update the weight with $w \to w' = w - \eta\frac{\partial J}{\partial w}$ with η the learning rate. For weights w of the last hidden layer L, we compute $\frac{\partial J}{\partial w}$ using the chain rule:

$$\frac{\partial J}{\partial w^L} = \frac{\partial J}{\partial \vec{\hat{y}}^L_i}\frac{\partial \vec{\hat{y}}^L_i}{\partial \vec{z}^L_i}\frac{\partial \vec{z}^L_i}{\partial w^L} \tag{2.8}$$

For weights w in another layer $q < L$, we have:

$$\frac{\partial J}{\partial w^q} = \frac{\partial J}{\partial \vec{\hat{y}}^L}\frac{\partial \vec{\hat{y}}^L}{\partial \vec{z}^L}\prod_{l=L}^{q+1}\left(\frac{\partial \vec{z}^l}{\partial \vec{a}^{l-1}}\frac{\partial \vec{a}^{l-1}}{\partial \vec{z}^{l-1}}\right)\frac{\partial \vec{z}^q}{\partial w^q} \tag{2.9}$$

The optimal solution would be to compute the gradient over the full training set, however this would lead to updating the weights only once per epoch and hence requires a very large number of epochs (and computing time) to converge. Instead, the network is trained with mini-batches, meaning that the weights are updated only after computing and accumulating (summing) the gradient over a batch of data randomly sampled from the dataset instead of accumulating over the whole dataset. This technique improves the speed of the gradient descent compared to passing the data one by one and thus, improves the convergence time. It also adds stochasticity which leads to noisier gradients that are better capable to escape local minima and prevent overfitting.

Gradient descent

The difference between Adam and SGD is that Adam stores m, the average of the exponentially decaying past gradient and v, the average of the exponentially decaying past squared gradients. With β_1 and β_2 the decay rates, at the training iteration t:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\frac{\partial J}{\partial w} \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2)\left(\frac{\partial J}{\partial w}\right)^2 \tag{2.10}$$

Then, the weight update rule becomes:

$$w \to w' = w - \frac{\eta}{\sqrt{v_t/1-\beta_2^t} + \epsilon}\frac{m_t}{1-\beta_1^t} \tag{2.11}$$

where ϵ is a smoothing term that avoids division by zero. Adam adds an adaptive momentum to the gradient descent that improves convergence speed and stability by speeding up or slowing down the gradient descent when needed during the optimization.

Learning rate

The learning rate η plays an important role in the overall training procedure. A high learning rate can prevent any convergence to an optimum or, worse, cause numerical error explosions. Contrariwise, a small learning rate can lead to a very slow training and a convergence to the nearest local optimum without any exploration of other potentially better optimums. For these reasons, η is often included into a hyperparameter search procedure. It can also be useful to control η during the training phase thanks to a learning rate decay rule based either on a number of iterations of the training loop or on a metric that detects when convergence reaches a plateau.

Generalization

Overfitting is a common issue that can arise with over-parametrized ML models able to learn features specific to each training data without any capacity of generalizing to data that are not in the training set. ANNs having up to more than billions of learnable parameters, it is fair to assume that they can be subject to the same issue. Nonetheless, it has been show experimentally and theoretically (Geiger et al., 2020; Zhang et al., 2017) that ANNs often behave differently by exploiting at first the features shared among the training data and then dedicate the remaining computational power to exploiting the features specific to each data. This makes overfitting less of an issue for ANNs but still requires monitoring overfitting by using another set of data called the validation set. The loss 2.7 computed over the validation set, on which the network has not been trained, gives the mean error expected on any data point as long as it is randomly sampled in the same distribution. Finally, in the same way that the validation set is used to monitor generalization during the optimization of the learnable parameters, a third dataset called the test set is used to monitor generalization after optimization of the hyperparameters (hyperparameters optimization is described in the Section 2.1.4).

2.1.3 Towards more complex networks

The universal approximation theorem shows that under certain conditions, any function can be approximated with a deep or large enough MLP, but it does not provide a way to find the tuned parameters of such network. In practice, computational and optimization constraints prevent to get close to this idealized network. To circumvent this issue, deep learning researchers have developed through the years network designs relying on multiple ways of organizing neurons of ANNs in order to make them easier to train and more adapted to the task. This section describes some of these improvements used in the architectures developed for this thesis or in the population genetic community. One of the first improvements made over MLPs is the development of convolution layers that allow to reuse weights to process different parts of a same data sample, and thus greatly reduce the number of weights needed as well as the required dataset size. This section also presents recurrent neural networks (RNNs), a type of network that is suited to temporal data. Finally, this section shows how attention mechanism is a very helpful method to associate layer inputs by using criteria learned during the training.

Convolution layer

Intuitively, a convolution layer consists in applying the same mask of small size A to submatrices of B of the corresponding size through a sliding window process. This allows to reuse the weights of A multiple times over different entries of a layer and exploit the spatial relationship between them. The stride describes how the filter is shifted over B. For example, a stride of 3 tells that the filter is shifted by step of 3 elements after each application of A on B.

A convolution in two dimensions (Figure 2.2) is defined as a matrix $A \in \mathbb{R}^{N \times M}$ that convolves a bigger matrix $B \in \mathbb{R}^{P \times Q}$ with $P \geq N$ and $Q \geq M$. The notation $B[i_1, i_2; j_1, j_2]$ denotes the submatrix of B consisting of the intersection of rows i_1 through i_2 and columns j_1 through j_2. With a stride of 1, the convolution filter is applied to all submatrices of size $N \times M$ of B. With a stride of s_1 along the row axis and s_2 along the column axis, the filter is applied to all submatrices $B[k \times s_1, N - 1 + k \times s_1; l \times s_2, M - 1 + l \times s_2]$, $k \in \{0, \cdots, \frac{P-N}{s_1}\}$, $l \in \{0, \cdots, \frac{Q-M}{s_2}\}$ by moving A by s_1 and s_2 steps over B. The convolution operation itself is the element-wise sum of the Hadamard product between A and the submatrix $B[i_1, i_2; j_1, j_2]$:

$$conv(A, B[i_1, i_2; j_1, j_2]) = sum(A \odot B[i_1, i_2; j_1, j_2]) \tag{2.12}$$

It can be performed on $\frac{P-N}{s_1} + 1 \times \frac{Q-M}{s_2} + 1$ submatrices of B and will result in a new matrix of dimension $\frac{P-N}{s_1} + 1 \times \frac{Q-M}{s_2} + 1$.

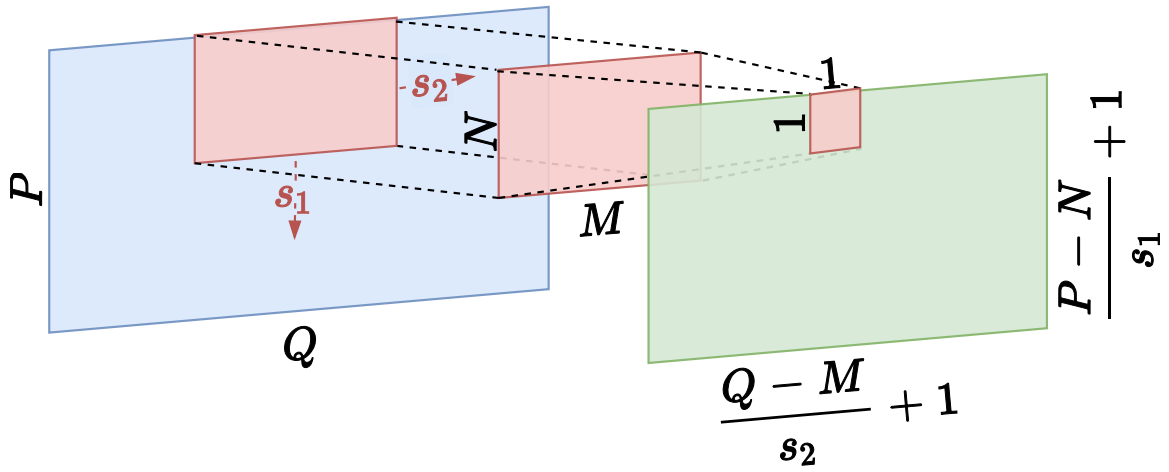

Figure 2.2: **Diagram of a convolution mask in two dimensions**. The mask (in red) is applied over a matrix (in blue), resulting in the green matrix. Each element of the green matrix is computed by performing the sum of the Hadamard product between the mask and the corresponding window over the blue input matrix. s_1 and s_2 denote the striding factor of the mask in the two dimensions.

This mechanism is the core component of convolutional neural networks (CNNs). In practice, CNNs include multiple masks per layer for which their result are concatenated over a third dimension called the feature dimension. Therefore, the data take the form of three-dimensional tensors through the network, which are processed with three-dimensional masks.

Recurrent neural networks (RNN)

Recurrent neural networks (RNN) is a type of ANN that has been introduced in the 80s to process sequential data such as temporal series or text. It consists of a cell that contains a more or less complicated ANN which is repeated over the sequence of data. For a given part of the sequence, the cell outputs a prediction, which is then passed to the next cell along the sequence. RNNs are usually trained with backpropagation through time (BPTT) which is simply the backpropagation algorithm described above, but because the same cell is repeated along the sequence, the gradient is summed up for each sequence step. RNNs are in theory capable of handling sequences of arbitrary length by repeating the cell to cover its entire length, but for long sequences, it is subject to a strong exploding or vanishing gradient effects (Hochreiter et al., 2001). To solve this issue, Hochreiter and Schmidhuber (1997) introduced long short-term memory networks (LSTMs), with cells that not only output a prediction, but also a "cell state" that can be linearly altered by the two other inputs of the cell (the prediction from the previous cell and the current input from the data sequence). This mechanism is similar to residual connections in residual networks (He et al., 2016) as layers can be skipped by a part of the data flowing through the network. This prevents gradient from vanishing in some cases, helps to find longer range dependencies in the data sequence and overall improves training (Balduzzi et al., 2017; Li et al., 2017a). Nonetheless, the long-range dependencies that can be found by an LSTM are limited by the size of the "cell state" and it is primarily for this reason that LSTMs have been later foreshadowed by attention mechanisms.

Attention mechanism

Although the "cell state" of LSTM (Hochreiter and Schmidhuber, 1997) can be considered as a form of such mechanism, the term "attention mechanism" was first introduced in the Transformer architecture (Vaswani et al., 2017) to tackle natural language processing (NLP) problems. Bigger architectures based on similar mechanisms such as Bert (Devlin et al., 2019) and GPT-3 (Brown et al., 2020) have shown impressive results on NLP tasks. Attention mechanism has also been applied to other domains such as image sampling and reconstruction with Image GPT (Chen et al., 2020), image generation with DALL-E (Ramesh et al., 2021) and protein folding with Alphafold (Jumper et al., 2021). Similarly to RNNs, attention mechanism is aimed at treating a sequence, but with the advantages of parallelizing the computation between all sequence elements instead of processing them sequentially. It also connects the sequence elements in the same way, which prevents the loss of information between distant elements and, thus, better handles long range dependencies. Although the architectures previously mentioned are fairly complex and integrate other mechanisms, the core attention mechanism (which is often called self-attention) can be described as follows: for each element i of a sequence of length n embedded as a vector $\vec{x_i}$, three weight vectors are learned: the queries $\vec{q_i}$, the values $\vec{v_i}$ and the keys $\vec{k_i}$. The attention mechanism will compute, for each element i, a linear combination of values $\vec{v_j}$, with weights depending on the affinity between element i and elements j, expressed as the similarity between the query $\vec{q_i}$ and the keys $\vec{k_j}$. For each $\vec{x_i}$, the dot-product attention vector $\vec{\alpha_i} =< \vec{q_i} \cdot \vec{k_1}, \vec{q_i} \cdot \vec{k_2}, \cdots, \vec{q_i} \cdot \vec{k_n} >$ is computed. To limit the risk of vanishing gradients, $\vec{\alpha_i}$ is then scaled by $\sqrt{d_k}$, d_k being

the dimension of the vectors $\vec{q}$ and $\vec{k}$ (in practice, the dimension d_v of the vector $\vec{v_i}$ is often chosen equal to d_k), and passed through a softmax function: $\vec{\beta_i} = \text{softmax}(\frac{\vec{\alpha_i}}{\sqrt{d_k}})$. This vector $\vec{\beta_i}$ can be interpreted as a measure of affinity between $\vec{x_i}$ and the other embedded elements of x. Finally, the output of the attention mechanism for $\vec{x_i}$ is the sum of the value vectors v of all sequence elements scaled by the affinities $\vec{\beta_i}$, that is, $\sum_{j=1}^{n} \beta_{ij}\vec{v_j}$. If the vectors $\vec{q_i}$, $\vec{k_i}$ and $\vec{v_i}$ of all elements are concatenated in matrices $Q \in \mathbb{R}^{n \times d_k}$, $K \in \mathbb{R}^{n \times d_k}$ and $V \in \mathbb{R}^{n \times d_v}$ the previous steps can be written as follows:

$$\text{attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.13}$$

In order to perform more than a single attention at once and thus increase the network expressivity, Vaswani et al. (2017) also introduced the multi-head attention scheme by projecting h times Q, K and V with h linear layers of learnable parameters W_i^Q, W_i^K and W_i^V. The output of all heads is then concatenated and a last W^O linear layer project the outputs back to the original dimension.

$$\text{multihead}(Q, K, V) = \text{concat}(head_1, \cdots, head_h)W^O \tag{2.14}$$

$$\text{where } head_i = \text{attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.15}$$

The complexity per layer of attention for a sequence of length n and a representation dimension d is $\mathcal{O}(n^2 \cdot d)$. This quadratic term arises because attention works on all pairs of elements in the sequence, which can be problematic for long sequences. To circumvent this issue, alternative attention mechanisms have been developed, such as the ones described in the agglomerative attention paper (Spellings, 2019) or the one developed for our MixAttSPIDNA architecture presented in Section 3.4.1.

2.1.4 Technical points

This section presents tools that can be applied to most types of architectures to facilitate their development. It describes how layer normalization circumvents common issues of ANN optimization. This section also introduces some hyperparameter optimization methods commonly used, how ANNs are implemented in practice, and how specialized hardware can greatly improve computation times.

Layer normalization

Layer normalization is a powerful tool to avoid internal covariate shift, a shift of activation distribution that can lead to weights having infinite values and thus, vanishing or exploding gradients. Adding layers of normalization allows being less sensitive about weight initialization and input data normalization. From the earlier batch normalization (Ioffe and Szegedy, 2015) to the more general Group normalization (Wu and He, 2018), several methods exist with the same goal of keeping the activation of each layer to zero mean and unit variance. For instance, for a batch of activations $B = a_1 \dots a_m$,

the activation after the batch normalization a'_i is given by the following process:

$$a'_i = \gamma \hat{a}_i + \beta \tag{2.16}$$

where:

$$\hat{a}_i = \frac{a_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2.17}$$

$$\mu_B = \frac{1}{m} \sum_{i=1}^{m} a_i \tag{2.18}$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (a_i - \mu_B)^2 \tag{2.19}$$

β and γ are learned with backpropagation and are frozen after training. During validation and test phases, mean μ_B and variance σ_B^2 are replaced by mean and variance computed on the fly during training. Although batch normalization is sufficient in most cases, variants of the SPIDNA architecture that handle data of varying size (see SPIDNA Section 3.3.2 for more information) have to rely on other normalization procedures. Indeed, data of varying sizes cannot be collated into the same tensor, which in practice, makes the batch dimension inaccessible to the batch normalization layers. Figure 2.3 shows three other layer normalizations that operate similarly to batch normalization without relying on the batch dimension.

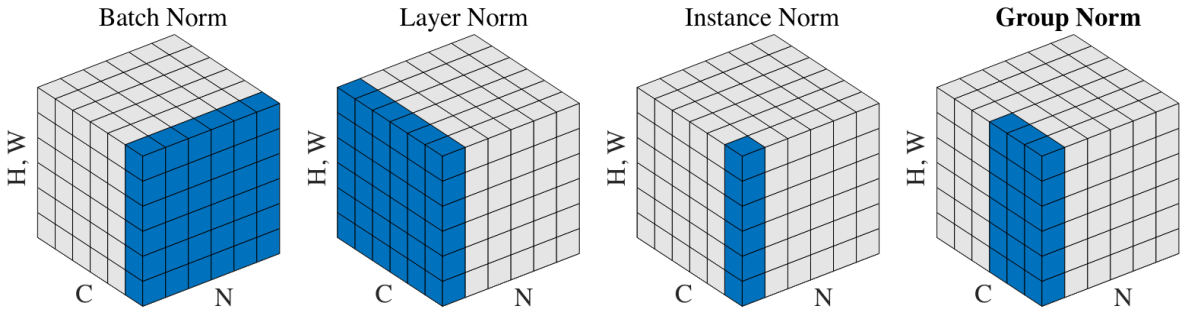

Figure 2.3: **Figure from Wu and He (2018) of the different normalization methods.** From Wu and He (2018): *Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.*

Pooling layers

Pooling layers are typically used in CNNs to quickly augment the scope of neurons over the input of a network. A pooling layer consists of a filter over the layer inputs, similar to a convolutional layer but without any weights, that performs an operation and return a smaller output. The most commonly used pooling filters return either the max, the min, or the average of the inputs covered by the filter. The size of the pooling filters determines how the scope of the input will increase in the next layer. For instance, a pooling filter with a size of 3 over one dimension will multiply by 3 the scope on the input

covered by the neurons in the next layer. One can also reduce the data flow dimension by increasing the stride of convolution filters, but with the risk of not covering part of the input if the stride is greater than the filter size.The size reduction offered by pooling layers at almost no cost (since no weight needs to be learned) is also convenient at the end of a network, in between the convolution and the fully connected layers. Indeed, it allows a drastic reduction in input number, and thus weight number, of the first fully connected layer. Finally, adaptive pooling can be used to consistently output tensors of the same size even when inputs are of different sizes, by adapting the pooling filter size to the input size.

Hyperparameters optimization

Some ANN architectures seem to perform well on very different tasks, for instance, slightly modified versions of GPT-3 have been used to tackle various natural language processing (NLP) problems (Brown et al., 2020) but also image generation (Chen et al., 2020) and text-to-image translation (Ramesh et al., 2021). However, until now, there is not a single architecture that ensures to perform well on any type of dataset. Thus, one of the challenge of applying ANN to a new dataset is to defined its architecture. ANNs distinguish themselves from other machine learning methods by their large number of hyperparameters, i.e., parameters that are not optimized during the gradient descent optimization procedure. These hyperparameters can include, similarly to other ML methods, parameters of the training process such as the learning rate, weight decay or the batch size, but they also include parameters defining the computational graph of the ANN such as the number of layers, their types (fully connected, recurrent, convolution, etc.) or the number of hidden units in each of them. The amount of hyperparameters being potentially infinite and the training of an architecture being time-consuming, the development of deep learning architectures often relies on the experience and intuition of the practitioner in a try-and-repeat process instead of on an automatic process. Nevertheless, it is possible to optimize some hyperparameters automatically depending on computational resources. Grid search and random search are two methods commonly used to explore the parameter space uniformly, but other methods based on Bayesian optimization are more suited to the computational cost of training ANNs. For instance, HpBandSter is a package that implements the HyperBand (Li et al., 2017b) algorithm to run many hyperparameter trials on a smaller resource budget (i.e., few epochs) and runs the most promising trials on a greater budget. Combined with BOHB (Falkner et al., 2018a), a Bayesian optimization procedure that models the expected improvement of the joint hyperparameters, this method provides more guided and faster search of the hyperparameter space. At each step, BOHB draws a new combination of hyperparameter values to be tested according to the expected improvement and to a predefined prior.

Deep learning libraries and hardware

The recent advance in deep learning can be attributed in part to the development of deep learning libraries and hardware that allow the implementation of complex architecture with minimal hassles. Libraries such as Pytorch (Paszke et al., 2017) or Tensor-

flow (Abadi et al., 2015) allow defining the forward computational graph of ANNs and perform automatic differentiation for gradient computation. They also implement basic building blocks of ANNs and tools such as data loaders, convolution layers, learning rate decay or layer normalization. All architectures developed through this thesis have been implemented with Pytorch.

Graphics processing units (GPUs) were at first intended to provide fast and parallelized creation of images, but they are now also used for ANNs as they are more adapted to operation between tensors than central processing units (CPUs). One important feature of GPUs is their dedicated memory called video random access memory (VRAM). In practice, VRAM availability often limits the ANN size, as the network needs to be fully loaded on it to exploit GPU computational speed without bottlenecks. To circumvent the limits of VRAM, one can use multiple GPUs by copying different parts of the ANN on the different GPUs, but this scheme introduces a bottleneck because in this case they are bonded to run sequentially and need to communicate data with each other. Another strategy that does not allow building bigger networks but still greatly increases the computational speed is to split each mini-batch across GPUs which permits bigger mini-batches without necessitating more VRAM.

2.2 Deep learning applications in genetics

Although complex evolutionary models of the genome such as the coalescent have been developed during the past decades, it is only recently that large genome wide sequencing datasets of multiple individuals have been made publicly available (Bergström et al., 2020; Consortium et al., 2015; Daetwyler et al., 2014). Hence, one of the new challenges in population genetics is to fill the gap between theoretical models of genome evolution and experimental observations, which requires suited algorithms capable to process genomic data. Because the diploid human genome being approximately 6.4 billions base pairs, handling it is very challenging for most statistical methods. This is because most of them are subject to the curse of dimensionality, an issue that arises while treating data that have many dimensions, making samples sparse in the data space. To circumvent this issue, the data dimension is often reduced by computing handcrafted summary statistics. Unfortunately, information relevant to solve the task can be lost, thus leading to poor performances.

Computer vision is a domain that was subject to the same limitations due to the dimensionality of high resolution images. Prior to the introduction of deep learning in this field, many algorithms have been developed based on summary statistics designed more or less by hand, but until the recent advances in deep learning, no method had the capacity to automatically learn any type of summary statistics possible on images. The main advantage of deep learning is that the search for summary statistics is jointly optimized with the task to perform, ensuring that the first ANN layers compute data features that are relevant for the task. Thus, deep learning has completely superseded previous methods in this field and lead to impressive results, even beating humans for some tasks such as image recognition (Krizhevsky et al., 2012), object detection (Redmon et al., 2016) or face recognition (Schroff et al., 2015).

Inspired by its success in computer vision, deep learning has also been applied to

genetic data because they share similarities with images: they both have high dimensionality, characteristic patterns and long-range dependencies. But genetic data also have their own characteristics that requires to adapt existing ANNs, which will be discussed in more details in the Section 3.3 detailing the SPIDNA architecture. For instance, permutations of sequences in multiple sequence alignments (MSA) are equivalent and represent the same data. SNP matrices, the favourite object of study of many population geneticists, can have a variable number of SNPs and haplotypes, preventing the use of some types of ANNs. Nonetheless, deep learning has already shown great success at predicting effects of noncoding variants with a CNN called DeepSEA (Zhou and Troyanskaya, 2015), at detecting alternative splicing sites (Jaganathan et al., 2019), at predicting phenotype markers (Ma et al., 2018), at predicting sequence specificities of DNA and RNA-binding proteins with another CNN (Alipanahi et al., 2015) or more recently, at predicting the protein 3D structures with Alphafold, an architecture that includes attention mechanisms and takes MSAs as inputs (Jumper et al., 2021).

This section is a non-exhaustive review of the numerous applications of deep learning in the field of population genetics, with a focus on the architectures and data used in each case. It will also discuss whether it seems possible to find a universal ANN that could tackle all these different applications.

2.2.1 Inference from genomic data

Most deep learning applications to population genetics target inference problems, meaning that ANNs are trained to take as input DNA sequences of a population and infer parameter values linked to the population evolutionary parameters. The genetic diversity of a population is driven by the interaction between mutation, genetic drift, recombination, natural selection and demography. Therefore, it is in theory possible to estimate the parameter values of each of these phenomena based on its observed diversity, yet it is in practice difficult to disentangle them because they can have similar and joint effects on the genome. In most settings, the ANN is trained in a supervised manner, which requires a dataset of genomic data and their associated evolutionary parameters associated to them. In practice, creating such dataset with real data would require to precisely monitor the population parameters over many generations, for many populations and in a setting close to what happens in nature if the goal is to study wild populations. Although some controlled experiments have been conducted with model organisms such as *E. coli* (Good et al., 2017), the differences between real and controlled populations could introduce important biases.

For all the above reasons, population geneticists turned towards simulation-based inference and relies for this on the evolutionary models described in the second chapter. Numerous population parameters are generated by drawing them from prior distributions, which are designed to contain the real evolutionary parameters of studied populations. To reduce the dimension space of these parameters, it is possible to fix some of them while focusing on the inference on the parameters of interest. For instance, if the ANNs are trained to infer demographic parameters, the simulations can neglect the effect of selection by removing it or fixing a simple rule that approximates its effect on the population. Once these parameters are drawn, they are fed to the simulator, which produces the corresponding genomic data used to train, validate and test

the network. Different steps can be added to this scheme: raw genomic data can be preprocessed and even drastically summarized into handcrafted summary statistics.

Data, either real or simulated, genotyped or sequenced, can usually be represented as single-nucleotide polymorphism (SNP) matrices where each row represents an individual or a haplotype, and each column represents a SNP which is a locus with a variation at a single base pair present in at least one of the sampled sequences. The term SNP is often misused and could be replaced by single-nucleotide variant (SNV), as its original definition indicates that the less frequent allele should be present in at least a certain percentage of the population, which is not verified in most population genetic studies. This matrix usually contains zeros and ones that represent the two possible, with zeros for the alleles with the smallest frequency in the sample or for the ancestral alleles when this information is available. A variety of alternative representations can also be used depending on the information available. For instance, diploid unphased data can be represented with zeros and twos for homozygous ancestral and derived loci and ones for heterozygous. SNPs with more than two alleles are often removed from the SNP matrix because they are very rare in reality and are often an artefact due to sequencing errors. Moreover, a vector containing the positions of the SNPs is associated to the SNP matrix to locate them in the original genome. Positions are either encoded by an integer that represents the absolute or relative number of base pairs, or a float between zero and one for scaled positions. Encoding the positions into floats requires to also store the full length of the sequence. Figure 2.4 shows how this format is related to the MSA format and that the only information lost is the precise nucleotide value. The matrix can cover from a small window to the full length of a contig in the sequence. One important feature of these type of data is that for the same length of DNA sequence covered, the number of columns in SNP matrices varies across samples. The number of rows also varies depending on the number of individuals sequenced, which is another feature that should be taken into account when designing methods based on this type of data.

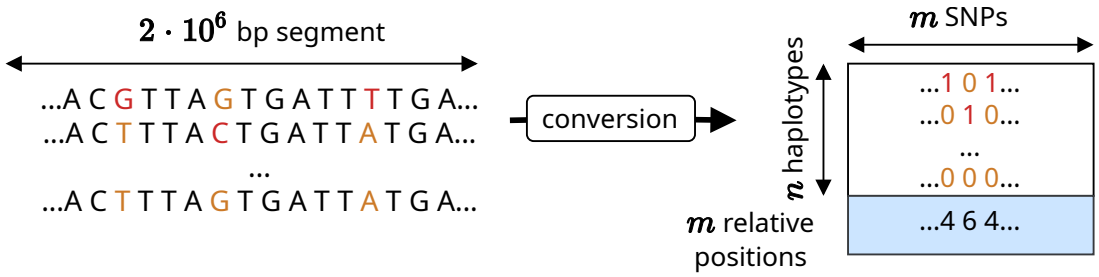

Figure 2.4: **Example of conversion of a multiple sequence alignment of 2.10^6 bp (left) into a SNP matrix (right).** Here, major alleles (in orange) in terms of frequency in the sample are encoded by ones and minor alleles (in red) by zeros. The SNP matrix has a relative position vector that encodes the distance of each SNP to its right neighbour.

2.2.2 Methods based on summary statistics

This section presents methods based on ANNs exploiting summary statistics, a set of statistics often designed by hand and tied to the population genetic theory, that are

computed on raw aligned sequences to reduce the data dimensionality.

Sheehan and Song (2016) developed one of the first approach for demography inference using deep learning. In this paper, the authors trained a MLP to jointly infer selection and demography of an African *Drosophila melanogaster* population. All simulations follow a simple demography with three piecewise constant population sizes that overall represents a bottleneck (i.e., a sudden reduction of population size later followed by a sudden increase). Along these three population sizes, a selection parameter is also drawn to include either a hard sweep, a soft sweep, balancing or no selection at the center of the genomic region simulated. Other simulation parameters such as generation time, recombination rate and mutation rate are fixed. After generating 400,000 genomic alignments of 100 haplotypes with msms (Ewing and Hermisson, 2010), summary statistics are computed for tree subregions, each of 100kb (only the middle region encompasses the SNP under selection, yet the left and right neighbouring regions can be affected more or less strongly by the selection effect). These statistics include the number of segregating sites, the Tajima's D statistic (Tajima, 1989), the folded site frequency spectrum (SFS), the length distribution between segregating sites, the identity-by-state (IBS), the linkage disequilibrium (LD) and finally H1, H12, and H2 statistics from Garud et al. (2015). This deep learning method achieved very low relative error over simulated data for the first two population sizes and a high accuracy for most type of selection except for hard sweeps, which they discovered to be mostly due to incomplete sweeps that are confused with balancing selection. Although the algorithm achieved good performances overall, the underlying demographic model is simple and it would be interesting to expand it thanks to additional population size parameters to test the robustness of the bottleneck discovered in the *Drosophila* population.

In their paper, Villanea and Schraiber (2019) used a MLP with dropout to classify the interaction between neandertal and two sapiens population (european and east asian) from the 1000 Genomes Project between five models of admixture. Simulations were performed using msprime (Kelleher et al., 2016) and translated to joint fragment frequency spectrum (FSS), a summary statistics that is able to capture the number of introgressed sites from a population to another.

Approximate Bayesian Computation (ABC) is a simple but powerful method to infer demographic parameters values and their posterior probabilities, but one of its drawback is its sensitivity to the curse of dimensionality. To circumvent this issue, Mondal et al. (2019) added a MLP to generate summary statistics from the site frequency spectrum (SFS). These summary statistics are later used with ABC to infer the posterior probabilities of the demographic parameters of a model of Neanderthal and Denisova introgression into Eurasian populations. They also used deep learning to primarily classify eight introgression models and retained the most probable ones for parameter inference. All simulations used to train the ANNs have been performed with FastSimcoal2 (Excoffier et al., 2013). Here, the main advantage of adding an ABC step to deep learning is that it is a simple way to add posterior inference. This scheme has also been tested for other methods, such as SPIDNA (Sanchez et al., 2021b) which will be discussed in details in the next chapter and Lorente-Galdos et al. (2019) which used it to study gene flow between archaic, African and Eurasian populations. The last chapter will discuss how this scheme could be replaced by methods relying solely on deep learning, such as generative artificial networks (GANs) or invertible networks.

Xue et al. (2019) introduced another deep learning method based on diploS/HIC (Kern and Schrider, 2018) called partialS/HIC to detect selection types in genome sequences from *Anopheles* Mosquito populations. This method is much different from the previous ones, as it computes 89 summary statistics for 11 subwindows of 55 kb long genomic sequence, resulting in two-dimensional matrices for each data, that are then processed using a convolutional neural network (CNN). Therefore, spatial information is introduced into the ANN and this is believe to improve inference because different types of selection have different effects on the genomic spatial structure. This method intermediary solution between methods relying on summary statistics computed on the full sequence fragment length and methods relying on SNP matrices presented in the next section.

2.2.3 Methods based on SNP matrices

The earliest application of deep learning to population genetic has been developed by Bridges et al. (2011) to classify between three populations from genotypic data. They used windows size of 20, 50 and 100 SNPs and trained the ANN with a subset of the real data. They were able to classify populations that were not differentiable with principal component analysis (PCA) and showed that ANNs perform similarly to support vector machine (SVM) for this task. One should notice that the size and complexity of the MLP they used was constrained by the technical limitations at the time, and it is expected that a CNN over the full genotypic data could significantly improve the classification error.

Flagel et al. (2018) used ANNs to tackle four tasks: detecting introgression, estimating locus-wide recombination rate for phased haplotype and autotetraploid genomes, detecting and classifying selection and finally, inferring population size histories. Flagel et al. (2018) simulated different datasets for each task, so this paragraph will focus on the inference of population size histories, which has also been tackle during this thesis. Similarly to Sheehan and Song (Sheehan and Song, 2016), they trained an ANN to infer the parameters of a demographic model with three piecewise constant population sizes and two parameters representing the time of the population size changes. They simulated 100,000 alignments of 1.5 Mb regions with ms (Hudson, 2004) and used 80,000 of them to train their network. The network is a CNN with two branches: the first one has four convolutional layers with 128 filters with SNP matrices as input and the second has one fully-connected layer with 32 hidden units with position vector as input. The outputs of the two layers are concatenated and passed to two fully connected layers with 256 hidden units and 5 hidden units (one for each parameter to infer). Multiple pooling layers are present throught the network. In order to reduce the size of simulations, the mutation rate has been divided by 10 which is equivalent to downsampling the number of SNPs by 10. Simulated SNP matrices have been padded with zeros on their left side to match the biggest matrix that contains 1,201 SNPs. They tested different shapes for their convolution filters (1×2, 1×4, 1×6, 1×8, 1×10, 2×2, 4×4, 6×6, 8×8 and 10×10) and two alleles encoding (-1/1 and 0/-1) to finally obtain the best results with 1×2 filters and 0/-1 encoding. This method is further detailed in Section 2.1.4 and has been compared to the SPIDNA architecture developed during this thesis.

Because the rows of SNP matrices represent haplotypes or individuals, a permuta-

tion of the matrix rows represent the same information as the non-permutated matrix. For this reason, Chan et al. (2018) introduced a new CNN with outputs invariant to the permutations of input rows to detect recombination hotspots in the genome. In order to make their CNN invariant to permutation by design, they used convolution filters of 1×5 dimension in the first two convolution layers and used the mean of the element-wise top decile as an invariant function afterwards. The outputs of the last convolution layer is then passed through two fully connected layers with 128 hidden units. The advantages of introducing this notion of invariance into the design of the ANN will be discussed in more details in the next chapter. The position vector has been added to the SNP matrix by duplicating it and adding it to the input tensor as a third dimension. This CNN is trained to handle windows of 20 SNPs over the input matrix and CNN predictions for all windows are compiled to obtain a posterior. Here, simulation were performed using msprime (Kelleher et al., 2016). Another important contribution of this paper is the introduction of the *simulation-on-the-fly* scheme where data are simulated during training so that each data is seen only once by the CNN. They obtained better predictions compared to classical scheme with multiple epochs. They also demonstrated that this paradigm guarantees that the posterior is calibrated and showed this experimentally. This last point is important because most deep learning applications lack of posterior and focus on point estimate, although other popular methods such as approximate Bayesian computation (ABC) already allow approximating posteriors. The perspective section (Section 5.1) of this thesis will discuss other methods to obtain posterior and how they could be used to update simulation prior to further improve predictions.

ImaGene (Torada et al., 2019) is another software that utilizes simulations generated *on-the-fly* with msms (Ewing and Hermisson, 2010). Simulations take the form of SNP matrices without position information, that are resized to 128×128 with an algorithm primarily used for images. These data are then used to estimate the positive selection coefficient for a given genomic region by training a CNN with three convolutional layers with 32, 64 and 128 filters of size 3×3. For the last layer, the authors chose a strategy that differs from most ANNs addressing regression tasks. Indeed, they transformed the regression task into a classification task by dividing the range of possible values in 11 bins and uses a softmax layer with outputs interpreted as a distribution of probabilities. The authors also showed that they obtained better results by systematically ordering the rows and columns of the input SNP matrices according to a predefined ordering function. By doing so, the algorithm is invariant to permutations, like the one presented by Chan et al. (2018). However, this ordering might be sensitive to small changes in the matrix because there is no ordering in high dimensional space that is stable with respect to perturbations (Qi et al., 2016).

In their paper, Adrion et al. (2019) used simulation from msprime to train a recurrent neural network to infer the per-base recombination rate along the genome of a population. Here, two ANNs based on gated recurrent unit (GRU) a special type of bidirectional recurrent neural networks (RNNs) have been developed. The first one using SNP matrices with position and alleles encoded as -1, 1 and 0 (for missing or padded data). The second uses data in the form of a matrix with two rows, the allele frequencies and the positions, to match real data obtained with Pool-seq.

Deelder et al. (2021) developed another CNN to classify between four different se-

lection sweep types and neutral evolution in *Plasmodium* genomes. They compared different CNN architectures and obtained a final model with one convolutional layer with two 40×9 convolutional filters and two fully-connected layers. Their simulations were performed using SFS_Code (Hernandez, 2008), data columns (SNP dimension) are sorted by shared haplotype length and the matrix is then compressed using an image processing algorithm.

2.2.4 Generative models

The advances in deep learning also introduced new generative models that have been recently applied to population genetics. In their study, Yelmen et al. (2019) used two types of ANNs called generative adversarial network (GAN) (Goodfellow et al., 2014) and restricted Boltzman machine (RBM) (Smolensky, 1986; Teh and Hinton, 2001) to create artificial genomic datasets that closely match a real dataset. GAN consists of two neural networks called the generator and the discriminator that compete in a zero-sum game, meaning that the discriminator is optimized to distinguish between real and generated data, and the generator is optimized to fool the discriminator. The input of the generator is randomly drawn from coordinates in a latent space. RBM is another kind of ANN that comprises a visible and a hidden layer that are trained by encoding and decoding inputs and updating the network weights until the decoded inputs closely match them. These two models have been compared to other generators (a simple Bernouilli generator, markov chain, HAPGEN2 (Su et al., 2011) and coalescent simulations) with different quality metrics such as the similarity between real and generated summary statistics and principal component analysis (PCA) projections. They also computed privacy metrics and searched for real haplotypes present in the generated genomes to ensure that the generated data were effectively different from the real ones. Here, real data consist in multiple subsets of SNPs from individuals of the 1000 Genomes Project and the Estonian Biobank (Leitsalu et al., 2015) and the discriminator and generator for the GAN are MLPs.

In their paper, Wang et al. (2020) utilized the GAN framework to perform inference of demographic parameters. The motivation behind this approach is to perform simulation "on-the-fly" like previous approaches, but also to optimize the set of demographic parameters used by the generator until convergence to an estimation of the real parameter values. This approach is different from classical GANs because here, the generator is an evolutionary simulator (msprime (Kelleher et al., 2016)) and not an ANN. Therefore, the demographic parameters inputted to the generator are optimized using simulated annealing (because it is not possible to backpropagate the gradient through the evolutionary simulator), while the discriminator is a CNN based on Chan et al. (2018) and optimized with stochastic gradient descent from a cross entropy loss. GANs are notoriously difficult to train because an important imbalance between the performance of the generator and the discriminator can make them rapidly diverge. Due to this issue and the difference of nature between their generator and discriminator, Wang et al. (2020) introduced a pre-training phase for their discriminator, by training it with real and data simulated from msprime with random demographic parameter values. Then, these parameters are optimized with the simulated annealing procedure. This setting has been used to infer the demographic parameters of one and two-population models

of populations from the 1000 Genomes Project.

Another type of generative ANN called variationnal autoencoders (VAEs) has been introduced both for data generation (Montserrat et al., 2019) and to replace the tools used for visualizing population structure such as principal component analysis (PCA) (Battey et al., 2021).

2.2.5 Recent works

The previous sections are intended to provide an overview of the various applications of deep learning to population genetics by describing the various problems addressed and architectures developed. However, this is a very active research area and the number of papers published each year is constantly increasing. Therefore, these sections are not completely exhaustive, but several recent publications that will be not discussed in details are worth mentioning. For instance, new deep learning methods have been applied to genomic data for the detection of selection and adaptive introgression (Fadja et al., 2021; Gower et al., 2021; Isildak et al., 2021) and one uses the ancestral recombination graph as input of its network (Hejase et al., 2021). A MLP has been recently trained to infer the mutation rate from the site frequency spectrum (SFS) with robustness to the recombination rate (Burger et al., 2021). Finally, Kirschner et al. (2022) developed a CNN paired with ABC inspired by our work (Sanchez et al., 2021b) to examine the demography of European steppe biota.

2.3 Chapter conclusion

In conclusion, deep learning is a powerful framework that has shown results surpassing classical machine learning and statistic predictive performance in most fields, but sometimes at the expanse of interpretability and computing resources. These successes can be attributed to the flexibility of ANNs that makes them adaptable to most types of data, independently of their dimensionality and the task. ANNs are also very efficient at automatically finding relevant features in the data through optimization. Moreover, they can be easily scaled to use the full computing power available and attempt to increase the network expressivity, by simply adding more weights or reusing them on more inputs, as in convolution layers. Yet, deep learning has only been recently applied to genomic data and a few papers developed ANNs specifically for population genetics. The first ANNs targeted for demographic inference using raw genomic data are even more recent and have been published during the course of this thesis. This may be due to two facts: first, an ANN needs large labelled datasets in the context of supervised training and second, developing deep learning methods for a new task is not trivial, as a lot of choices in the design of the methods are left to the practitioner. As illustrated by the short review presented in this chapter, most studies have overcome this first issue by using recently released datasets and simulators. The second point still requires more research, as the diversity of approaches present in this review shows that no clear consensus has emerged on what are the best practices to address genomic related tasks with ANNs.

This chapter presented the different building blocks that have been used during this thesis, but they are only a small subset of the wide variety of the possible deep learning tools and approaches. There is no rule of thumb for developing ANNs and most choices are made based on interpretations of the network behaviours that often have no theoretical ground. Therefore, the next chapter will show how we tailored our ANNs to the demographic inference task by taking into account the data features and the objective. It will also present the baselines we used to compare and evaluate our methods.

Chapter 3

Methodological development for demographic inference

Contents

This chapter will describe the methods developed through this thesis to infer demography from genomic data, as well as the methods used as comparison baselines. We trained and evaluated all methods to perform one objective: inferring the detailed histories of effective population sizes using genomic data from a sample of individuals. Based on whole sequences of multiple individuals from a single population, the different methods aimed at predicting 21 population size parameters, each corresponding to a fixed time window.

We simulated two datasets to comply with real data from the 1,000 genome bull project (Daetwyler et al., 2014) (mentioned as *cattle dataset*) and high-coverage HGDP-CEPH human genome sequences (Bergström et al., 2020) (mentioned as *HGDP dataset*). We also generated simulations with selection to investigate the impact of this confounding factor on inference. We constituted a collection of baselines that include two methods based on summary statistics: an ABC and a MLP, and three methods based on SNP matrices: a MLP, a CNN previously developed by Flagel et al. (2018) and a *custom* CNN that has been developed during the preliminary work of this thesis. These baselines are later compared to the two main architectures developed here: sequence position informed deep learning architecture (SPIDNA) and mixed attention SPIDNA (MixAttSPIDNA). This chapter will show how we refined these two architectures after multiple iterations to improve inference and handle the key features of SNP matrices. It will also show how we trained the different ANNs and the various automatic hyperparameter optimization scheme that we used. Section 3.6 presents a preliminary work on the interpretation of deep neural networks with cannonical correlation analysis (CCA). Finally, we introduced a python package called dnadna, which is aimed at facilitating the development and sharing of deep learning architectures for the population genetics community.

We published the methods for the baselines and the SPIDNA architecture, as well as their comparison on simulated data and their application to the cattle dataset in *Molecular Ecology Resources* (Sanchez et al., 2021b):

- *Deep learning for population size history inference: Design, comparison and combination with approximate Bayesian computation.* - Théophile Sanchez, Jean Cury, Guillaume Charpiat and Flora Jay, Molecular Ecology Resources 21, no. 8, 2021.

The development of the MixAttSPIDNA architecture and its application to the cattle and HGDP dataset has been done in collaboration with our intern Pierre Jobic. This work has not been published.

We presented the dnadna package in a preprint (Sanchez et al., 2021a):

- *dnadna: Deep Neural Architecture for DNA - A deep learning framework for population genetic inference.* - Théophile Sanchez, Erik Madison Bray, Pierre Jobic, Jérémy Guez, Anne-Catherine Letournel, Guillaume Charpiat, Jean Cury and Flora Jay, 2021.

The results obtained with the methods described in this chapter will be discussed in the next chapter (Chapter 4).

3.1 Data

The two datasets studied in this thesis are a collection of cattle (*Bos taurus*) sequencing data from the 1,000 genome bull project (Daetwyler et al., 2014) and a collection of modern humans (*Homo sapiens*) sequencing data from the HGDP-CEPH human genome sequences (Bergström et al., 2020). They both include high coverage genome-wide sequencing of many individuals from different populations and thus, require computationally efficient methods in order to perform inference in reasonable time. Aside from the quality and amount of data in these datasets, they have also been chosen because they come from species that have been well studied from an archaeological and historical viewpoint, therefore allowing to compare the inferred population size histories to our current insights on the true demographic histories. For instance, the domestication of cattle that happened approximately 10,000 years ago should translate into a decrease of the effective population sizes because most cattle breeds today are descendant of a few individuals selected for farming (Consortium et al., 2009). Another example for human populations is the Out of Africa hypothesis of the origins of non-African populations, that should translate into a bottleneck at around 50,000 up to 100,000 years ago (Nielsen et al., 2017).

Most methods compared in this thesis are trained in a supervised fashion, and thus require simulated genetic data with labels. These simulated data are also needed to compare the performances of all methods, either by evaluating predictions on specific scenarios (constant, decline, expansion, bottleneck, and zigzag) or by comparing an evaluation metric averaged over many scenarios. To this end, the data are split into three sets: the train set used by ABC and all ANNs, the validation set used to evaluate performances for each hyperparameter's settings and to monitor overfitting while optimizing ANNs, and finally, the test set consisting of data never seen during optimization to compare methods while preventing overfitting (caused by the hyperparameter optimization).

The data format has already been described in the previous chapter (Section 2.2.1) and consists of a SNP matrix and its associated position vector encoded as distances between SNPs. The matrices contain zeros and ones to encode ancestral and derived alleles or minor and major frequency alleles, depending on the information available for the real dataset. As described in the next section about the cattle dataset, the real cattle haplotypes had to be collapsed into genotypes (instead of haplotypes) with zeros, ones and twos encoding homozygous and heterozygous loci. Simulations are generated thanks to two evolutionary simulators, msprime (Kelleher et al., 2016) for neutral simulations and msms (Ewing and Hermisson, 2010) for simulations that include selection, both based on the evolutionary models described in the chapter 1.

Simulating sequences using the previously mentioned evolutionary models requires indicating the different parameter values that describe the demography of the population simulated. Parameters that describes the demographic scenario and cofounding factors are drawn from priors that are designed to be as close as possible to the knowledge on the real population studied. There is a compromise between making a prior large enough to ensure that the true values for parameters of the real population are included and making it small enough so that the simulated demographic scenarios are sufficiently dense in the space defined by the priors.

The following sections will give an overview of both datasets, how they have been processed to be handled by the different methods and how each set of priors have been defined for each set of simulations.

3.1.1 Cattle dataset

The cattle dataset from Daetwyler et al. (2014) consists of whole genome sequences of 234 individuals from four cattle breeds (Angus, Holstein, Fleckvieh and Jersey) with an 8.3 fold coverage on average. The samples were sequenced using Illumina sequencing-by-synthesis technology (Bentley et al., 2008) and preprocessed before being aligned to the UMD3.1 reference genome. In their paper, Boitard et al. (2016b) used this dataset to infer the population size history of the cattle breed with their method based on approximate Bayesian computation called PopSizeABC.

First, the Jersey population was removed from the analysis because it only includes 15 individuals compared to the 25 diploid sequences (i.e., 50 haplotypes) used to train the ANNs. As the data of real cattle sequence are prone to phasing and sequencing errors, they were converted from haplotype to genotype with a minimum allele frequency (maf) of 0.2, as suggested by Boitard et al. (2016b). 25 diploid sequences were randomly sampled from each population, split into 2Mb segment and then, segments comprising centromeres were removed, leaving 1,213 segments. A similar number of SNPs was obtained for the three breeds: Angus (average: 4,536 SNPs, maximum: 22,391 and minimum: 775), Fleckvieh (average: 4,837 SNPs, maximum: 24,896 and minimum: 896) and Holstein (average: 4,732 SNPs, maximum, 24,098 and minimum: 1,212).

Neutral simulations

The demographic parameters were set up by following similar rules as Boitard et al. (2016b): $I = 21$ time windows $[t_i, t_{i+1}]$ were defined from present to ancient periods with $t_i = \frac{1}{a}\left((1+aT)^{i/(I-1)} - 1\right)$ generations, i going from 0 to $I-1$, $T = 130,000$, $a = 0.06$ and $t_I = +\infty$. These values of T and a were chosen by Boitard et al. (2016b) to capture important periods of cattle history. They could be modified to describe more precisely specific parts of the history by playing with the ratio between the length of recent versus old time windows. By increasing exponentially the time windows as we go further in the past, the scenarios became more detailed for recent times. Generation time for cattle are assumed to be about 5 years. Each demographic scenario is generated by drawing a first population size N_0 between 10 and 100,000 from a uniform distribution which corresponds to the most recent time window. The population sizes of the next time windows follow $N_i = N_{i-1} \times 10^{\beta}$ for i in $[1, 21]$, with β sampled uniformly between -1 and 1. β is redrawn if it gives a population size out of $]10; 100,000[$. 50,000 scenarios have been randomly drawn from this prior distribution and 100 independent 2Mb-long segments of 50 haploid individuals have been simulated for each scenario using the msprime coalescent simulator version 0.6.1 (Kelleher et al., 2016). A total of 5,000,000 SNP matrices X of size $M = 50$ haplotypes $\times$ S SNP sites, each associated with a vector of size S that contains the distances between SNPs (in bp) were obtained. Ancestral and derived alleles are encoded with 0 and 1. The mutation rate is set to 10^{-8} as in MacLeod et al. (2013). The recombination rate is sampled uniformly

between 10^{-9} and 10^{-8} for each scenario to be consistent with the estimations in cattle breeds (Sandor et al., 2012).

After simulation, scenarios producing fewer than 400 SNPs in any 2Mb regions were removed. This threshold could be changed by modifying the networks or simulating longer regions. However, the real cattle dataset has on average 4,357 SNPs across a 2Mb-long region, so these scenarios were far outside the plausible posterior distribution. That reduced the dataset to 18,461 scenarios (i.e., 1,846,100 SNP matrices) out of the 50,000 scenarios simulated with an average of 2,486 SNPs and a maximum of 17,839 SNPs. This dataset is split into a validation set of 500 scenarios (i.e., 50,000 validation SNP matrices overall) and a training set with the remaining 17,961 scenarios (i.e., 1,796,100 training SNP matrices). In order to check for hyperparameter overfitting, we have also simulated a test set from the same prior distribution. Hence, we randomly drew 2,000 scenarios and kept the 767 scenarios with more than 400 SNPs which gives 76,700 test SNP matrices. Training, validation and test set demographic parameters were all standardized using mean and variance from the training set.

Simulations with selection

To investigate the robustness of the different approaches, an extra set of data was simulated under demographic changes and selective pressure. *msms* (Ewing and Hermisson, 2010) was used to simulate scenarios including positive selection with additive fitness using varying values of selection coefficient (s in 2Ne units: 100, 200, 400 or 800), selection starting time (T_{sel}: 200, 1000 or 2000 generations ago) and initial frequency of the beneficial allele (f_0: 0.1%, 1%, 5%). The SNP under selection was located at the centre of the region. The mutation rate was set to 10^{-8}, the recombination rate to $5 \cdot 10^{-9}$, the number of haplotypes to 50 and the region length to 2Mb. 16×100 replicates were simulated for each of the 36 selection parameter combinations (s, T_{sel}, f_0) and 30×100 replicates with no selection under three demographic scenarios (constant, declining or expanding size) leading to a total of 181,800 SNP matrices. Inference methods requiring a fixed input size processed the 400 successive central SNPs (i.e., 200 before and 200 after the SNP under selection).

Summary statistics

This thesis investigates two methods based on summary statistics, the ABC approach and a MLP. To this end, site frequency spectrum and the linkage disequilibrium have been computed for this dataset. For each group of 100 segments corresponding to one scenario, the site frequency spectrum and the linkage disequilibrium have been computed as a function of the distance between SNPs averaged over 19 distance bins for a total of 68 summary statistics. This python script is partly based on the scikit-allel python module (Miles et al., 2019).

3.1.2 HGDP dataset

The HGDP dataset (Bergström et al., 2020) consists of 929 whole genomes from 54 populations, with 6 to 46 individuals per population (Figure 3.1 shows the number of indi-

viduals sampled per population). Sequencing has been performed using Illumina technology with an average coverage of 35x and reads have been mapped to the GRCh38 reference assembly. Despite having 1575 fewer sequenced genomes than the 1000 Genomes Project, Bergström et al. (2020) were able to identify a number of SNPs of the same order thanks to their high-coverage and the diversity of their sample.

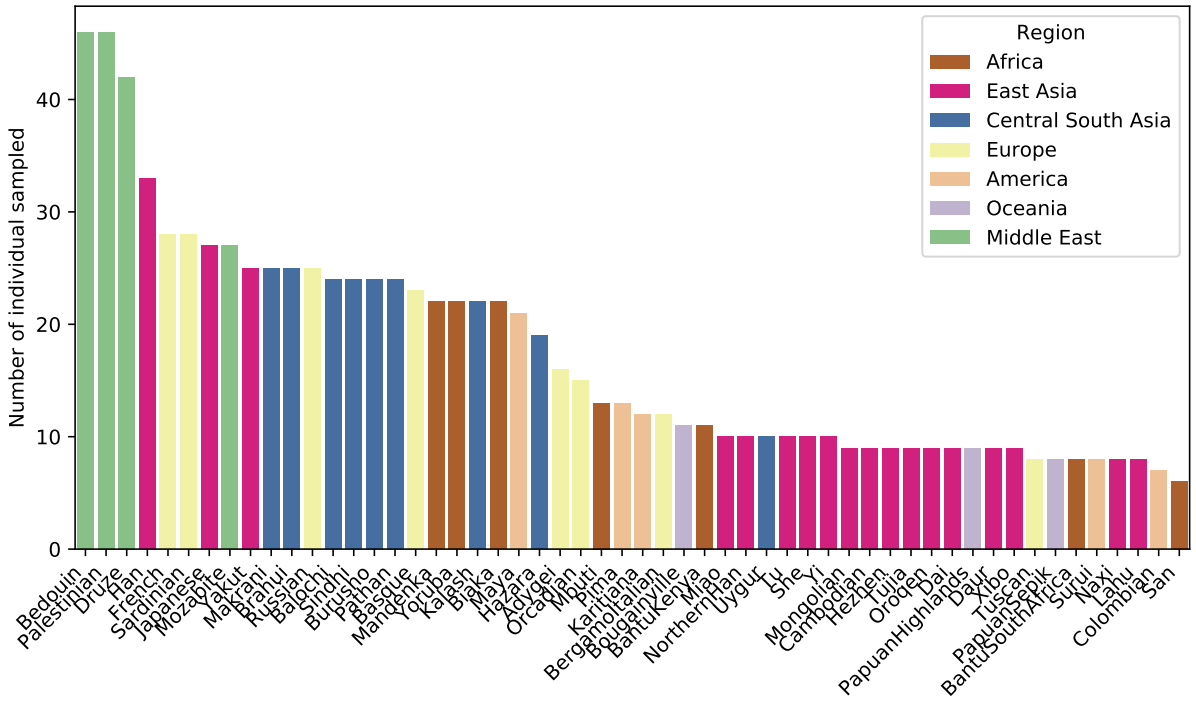

Figure 3.1: **Number of sample peer population in the HGDP dataset (Bergström et al., 2020).**

For the purpose of this thesis, this dataset has been processed as follows: after removing telomeres and centromeres, autosomes have been split into 2Mb segments and then polyallelic sites have been removed and SNPs have been encoded with zeros for ancestral and ones for derived by comparing either to the Ensembl database or to chimpanzee reference genome or encoded as minor and major alleles for the few SNPs for which ancestral information was not available. Figure 3.2 shows the great variability of the number of individual sampled for each population in the HGDP dataset. Therefore, we developed the MixAttSPIDNA (Section 3.4.2) architecture while keeping in mind that it should be able to handle this feature of the data). For that, we compared different mini-batch formats on the simulated cattle dataset (Section 4.2.4) and retained the best for the inference on the real HGDP dataset (Section 4.3.2).

Simulations

Simulations designed to encompass the HGDP dataset have been performed using msprime. The 21 time windows follow the same formula as the cattle dataset with the first window representing the time before 1 million years ago. 100 replicates of 2 Mbp-long regions are generated for each of 30,000 scenarios. The mutation rate is set

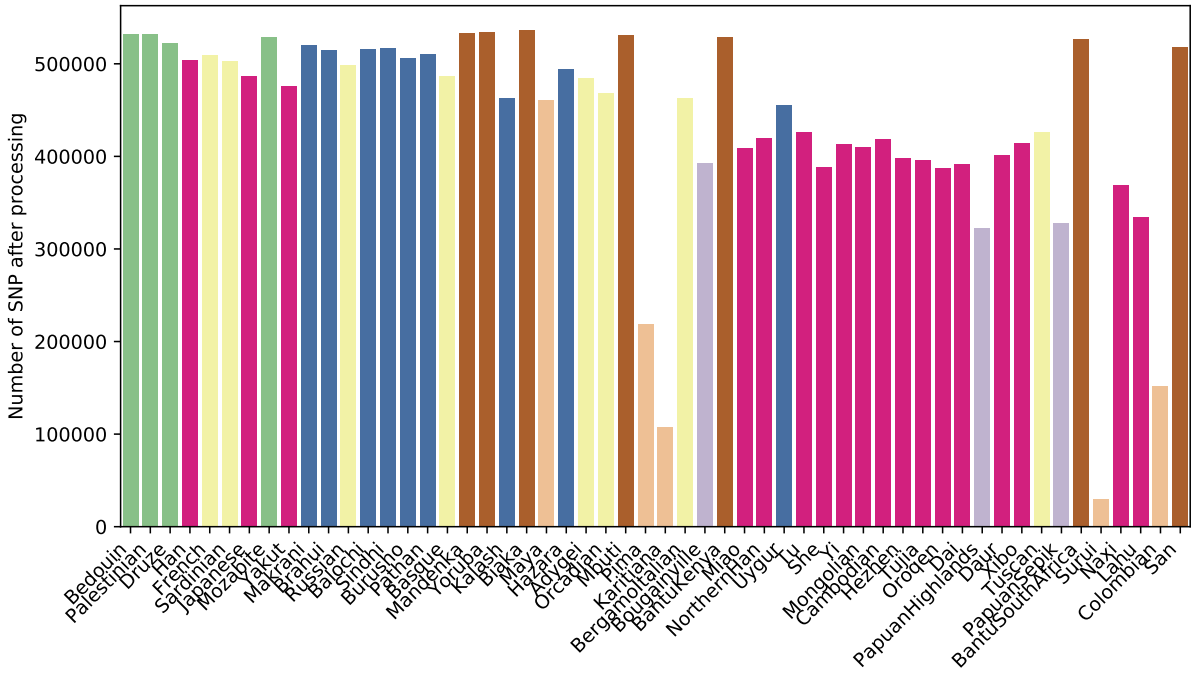

Figure 3.2: **Number of SNP per population after removal of telomeres and centromeres the HGDP dataset (Bergström et al., 2020).**

to 0.5×10^{-9} mutation per base and year, and a generation is 29 years. The recombination in centimorgan per base ρ is randomly drawn from a kernel Gaussian distribution fitted over the distribution from the recombination map of the 1000 genomes project (Consortium et al., 2015) (see Figure 3.3).

We choose this recombination rate prior over a constant value or a simpler uniform distribution to take into account more accurately this confounding factor. It is a good compromise with the most realistic strategy that would be to simulate the complete genome alignments with corresponding recombination rates along each segment. We preferred to simulate only 100 replicates of 2Mb segments by scenario in order to generate a wider range of demographic scenarios, as we focus primarily on the inference of population size histories.

The number of haplotypes sampled is uniformly drawn between 10 and 100. The effective population size for the most ancient of the 21 time steps is drawn uniformly on log_{10} scale between 100 and 1,000,000. Then, for each time window t, a growth rate $g^{(t)}$ is drawn with the following the formula:

$$g^{(t)} = \left(y^{(t)}/10\right)^{(1-\sqrt{\beta})\gamma} \tag{3.1}$$

with $\beta \sim U(0,1)$, $\gamma \sim B(-1,1)$ and $y^{(t)}$ the length of time window t. The growth rate is redrawn if the resulting effective population size fall outside the $]100; 1,000,000[$ range. We choose this distribution to have most of its mass around a growth rate of 1 while allowing some rare extreme values depending on the duration of the time window has shown by Figure 3.4. The idea behind this choice of prior is to have realistic population size changes that depend on the time elapsed (contrary to the prior proposed in Boitard et al. (2016b) and Sanchez et al. (2021b) that was already reducing the space

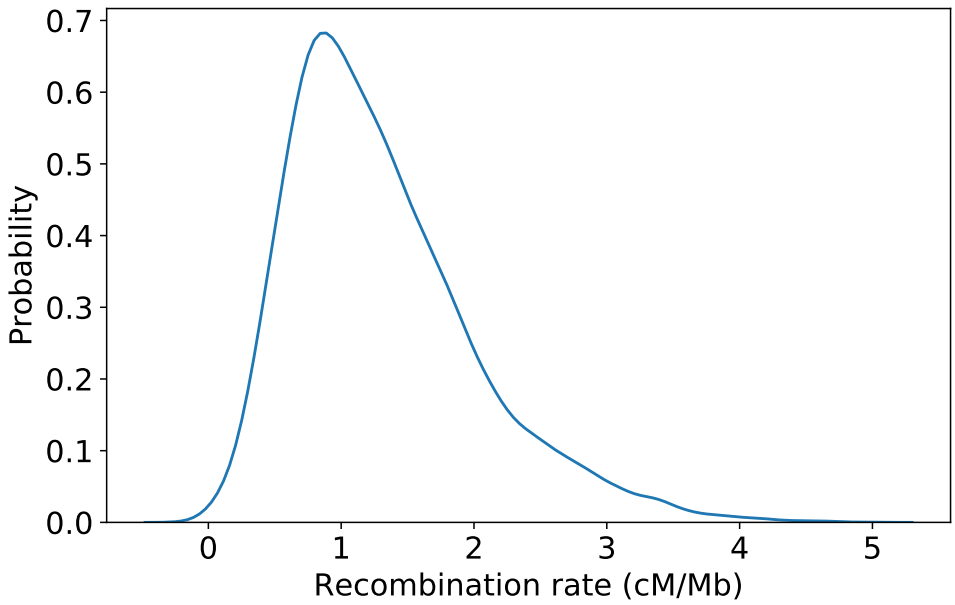

Figure 3.3: **Distribution of the Human recombination rate from which ρ is drawn.** The recombination rate from the 1000 genomes project (Consortium et al., 2015) is averaged over 2Mb windows after masking centromeres and telomeres and fitted with a kernel Gaussian distribution.

of plausible histories by preventing extreme jumps (increases or decreases) in a single step, however was not taking step duration into account), while allowing for some extreme growth rates that cannot be greater than $\frac{y^{(1)}}{10}$.

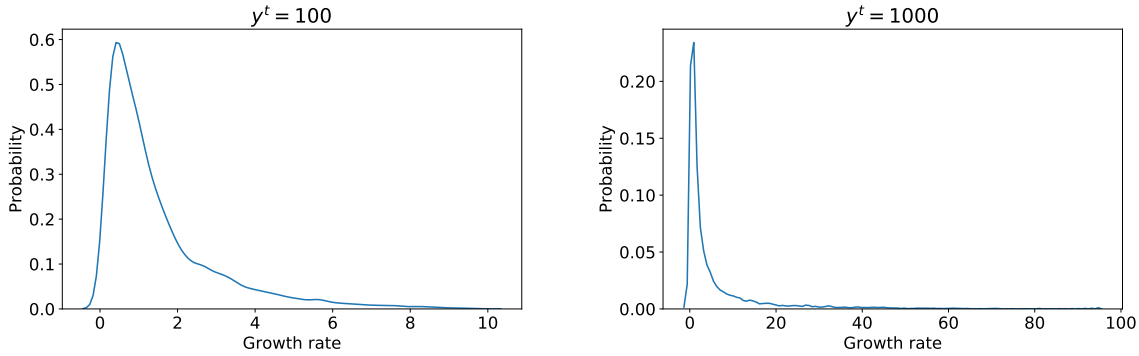

Figure 3.4: **Distribution of the growth rate for time windows of 100 years (left) and 1000 years (right) in HGDP simulations.** These distributions follow equation 3.1.

After removing scenarios that contain at least one replicate with fewer than 400 SNPs, the 21,044 scenarios are separated between a training set with 20,044 scenarios (i.e., 2,004,400 training SNP matrices) and a validation set with 1,000 scenarios (i.e., 100,000 validation SNP matrices). Independently, we simulated a test set including 1,499 scenarios (i.e., 149,900 validation SNP matrices) after preprocessing.

3.2 Baselines

The choice of the methods that have been included in the baselines is motivated either because the method has shown great results in previous studies: ABC, *Flagel* network and the MLP using summary statistics, or because they are ANNs with simple architectures that can be later compared to the more complex SPIDNA architectures: MLP and *custom* CNN with SNP matrices as inputs. It is noteworthy that most deep learning methods cited in chapter 2 for demographic inference were published during the course of this thesis, except for Sheehan and Song (2016). Moreover, we started this thesis by developing our *custom* CNN on the simpler task of inferring five demographic parameters representing the demographic scenario of a population that undergo a bottleneck: three changes in population size and two dates of decline and expansion (the results will not be shown here). We later complexified the task by switching to a demographic model with 21 parameters representing population size changes at fixed dates in order to apply our method to the cattle and HGDP datasets.

3.2.1 Approximate Bayesian computation (ABC)

Tested ABC algorithms included the simple rejection procedure (i.e., no correction) or one of the three correction methods implemented in the R package abc (Csilléry et al., 2012): local linear regression, ridge regression and non-linear regression based on a single-hidden-layer neural network. Hyperparameters were set to default except for the tolerance rate set to six possible values (0.05, 0.1, 0.15, 0.2, 0.25 and 0.3). ABC was ran on (a) predefined summary statistics, (b) SPIDNA outputs (i.e., automatically computed summary statistics), or (c) a combination of predefined summary statistics and SPIDNA outputs. The median of the posterior distribution was used as the demographic parameter estimate $\hat{\Theta}$.

3.2.2 Multi-layer perceptron (MLP)

The first MLP is based on summary statistics, has 3 hidden layers, ReLU activation functions and uses batch normalization. As in Sheehan and Song (2016), the hidden layers have respectively 25, 25, and 10 neurons. It takes 34 summary statistics as input. This network and all the following ones output 21 demographic parameters and are trained with a regular L2 loss function and adam optimizer (Kingma and Ba, 2014) unless stated otherwise. This MLP has a total of 2,986 trainable parameters. The second MLP is based on "raw" genomic data and takes as input a matrix of 50 haplotypes (rows) for 400 SNPs (columns) and its associated vector of distances between SNPs, both flattened into a single vector. Its hidden layers respectively have 20, 20, and 10 neurons, which gives it 408,981 trainable parameters.

3.2.3 *Custom* convolutional neural network (*custom* CNN)

Prior to developing the SPIDNA architecture, we started with a more classical convolutional neural network that we later included in our comparison baseline. CNN layers

process input elements by groups, allowing close SNPs to be processed together. This feature, combined with the stacking of layers in CNNs, helps the network to construct features dependent on the SNPs proximity. Important summary statistics used in ABC or other inference methods such as linkage disequilibrium can potentially be easily expressed by such CNN. Hence, our *custom* CNN has 2D filters that could have different shapes, i.e., mixed kernel sizes but also non-symmetrical masks. There is indeed no rationale behind considering square masks only as is usually done in computer vision to describe pixel neighbourhoods, as rows and columns in our case correspond to different entities (individual or phased haplotype versus markers). Using varied mask shapes helps our *custom* CNN to learn features of various patterns, potentially mimicking different types of summary statistics ("vertical" masks integrate over individuals, enabling the computation of allele frequencies at a SNP, while "horizontal" ones integrate over SNPs, as IBS or IBD sharing tract length does).

The *custom* CNN takes as input the same matrix of 400 SNPs and has 2-dimension filters of various shapes. The first layer consists of 5 kernels with rectangular shape (2×2, 5×4, 3×8, 2×10, 20×1) applied to the SNP matrix X. Each kernel creates 50 filters, which amounts to 250 feature maps after the first layer. The SNP distance vector d is treated by the 5 associated kernel shapes (1×2, 1×4, 1×8, 1×10, 1×1) with 20 filters each, making 100 filters in total. The results of the first convolutional layer are then concatenated so that the second convolutional layer will couple information from X and d in a way that emphasizes the original location of the SNPs along the genome. The outputs of this second layer are then combined and go through 5 convolutional layers and 2 fully connected layers. Adding convolutional layers one after the other allows our network to combine patterns and reduce the size of the data without adding too many weights to our model. This network has a total of 131,731 trainable parameters.

3.2.4 *Flagel* network

We reused the code associated with the repository of the first paper using a CNN for demographic inference (Flagel et al., 2018) and adapted to the dataset and task. The network was trained with the exact same architecture as the one published (Figure 3.5 from the original paper shows a schematic of the architecture), except that the last layer was changed to allow the prediction of our 21 population size parameters. The network was parametrized with the set of hyperparameters leading to the best performance in the previous work for two different types of SNP encoding (0/255 or -1/1). It is noteworthy that the actual encoding in their code is 0/-1 and not 0/255, thus the same encoding was kept to be able to compare the performance. The networks were trained with the same procedure of 10 epochs with early stopping in case of no progression of the loss after 3 epochs. The batch size is 200. The input data had 50 haplotypes. Its number of SNPs is either 400 (as processed by the *custom* CNN) or it is downsampled to one every ten SNPs (as done in the original work), leading to 1,784 wide input SNP matrices. This size corresponds to the tenth of the biggest SNP matrix in our dataset. Smaller simulations are padded with zeros. All parameters can be found in Table 3.1.

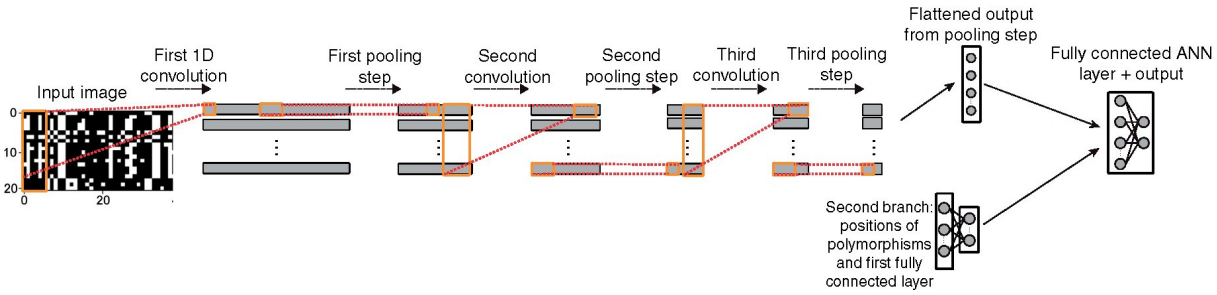

Figure 3.5: **Schematic of *Flagel* network.** Figure from the original paper (Flagel et al., 2018). Input images are SNP matrices that undergo a series of convolution and pooling layers. The filters of the first convolution cover all haplotypes, then the filters of each layer cover all features computed by the previous layer. In parallel, a fully-connected layer process the SNP positions. Finally, its result is concatenated to the convolution layer output and fed to another fully-connected layer in order to output the predictions of the network.

Input dimension	SNP encoding	Convolution type	Kernel size	Pooling size	Log-scaled output?	Sort chromosomes?	Use dropout?
50×400	0/-1	1D	2	2	Yes	Yes	Yes
50×1784	0/-1	1D	2	2	Yes	Yes	Yes
50×400	-1/1	1D	2	2	Yes	Yes	No
50×1784	-1/1	1D	2	2	Yes	Yes	No

Table 3.1: **Parameters used for the *Flagel* CNN.**

3.3 Sequence position informed deep learning architecture (SPIDNA)

The Sequence Position Informed Deep Neural Architecture (SPIDNA) is designed to comply to the principal features of SNP data: data heterogeneity (data includes genetic markers and their positions encoded as distances between SNPs), haplotype permutation invariance, long range dependencies between SNPs and variable number of SNPs. Similarly to the *custom* CNN, SPIDNA takes as input a matrix describing haploid individuals as rows and SNP as columns, with an additional row for the SNP distances.

3.3.1 Permutation invariance

One of the SNP matrix properties is its invariance to the permutation of haploid or diploid individuals (rows of the SNP matrix), meaning that the same matrix with permuted rows contains the exact same information and should lead to the same predictions. Most summary statistics are already invariant to the haplotype order by definition. On the other hand, typical operations used in ANNs such as rectangular filters and fully connected layers are not invariant, and consequently the baseline ANNs do not respect this data feature, but can still approximately learn this data property. To avoid wasting training time to learn that there is no information in the row order, it has been proposed to systematically sort the haplotypes according to a predefined

rule (Flagel et al., 2018; Torada et al., 2019). However, because there is no ordering in high dimensional space that is stable with respect to perturbations (Qi et al., 2017), we chose yet another alternative and enforced our network to be permutation-invariant by design. Permutation-invariant networks, or exchangeable networks, were successfully applied in population genetics by Chan et al. (2018) for inferring local recombination, but our architecture is different in that the invariant operations are performed at each block (there is only one invariant layer in Chan et al. (2018)), enabling both individual equivariant features and global invariant features to contribute to the next layer. Figure 3.6 shows the difference between equivariant and invariant functions. It has been proven that this type of architecture provides universal approximation of permutation-invariant functions (Lucas et al., 2018; Zaheer et al., 2017). Here we applied the methodology from Lucas et al. (2018) by using the mean as our invariant operation for our SPIDNA and MixAttSPIDNA architectures. However, the mean over the haplotype dimension has limited expressivity. To circumvent this, we first tried to add a higher moment statistics (the variance) alongside the mean, but in practice, this did not improve the predictions. We have finally chosen to develop a more flexible invariant function based on attention mechanisms that we called attention hub (see Section 3.4.1 for more details), and added it alongside the mean in our MixAttSPIDNA architecture.

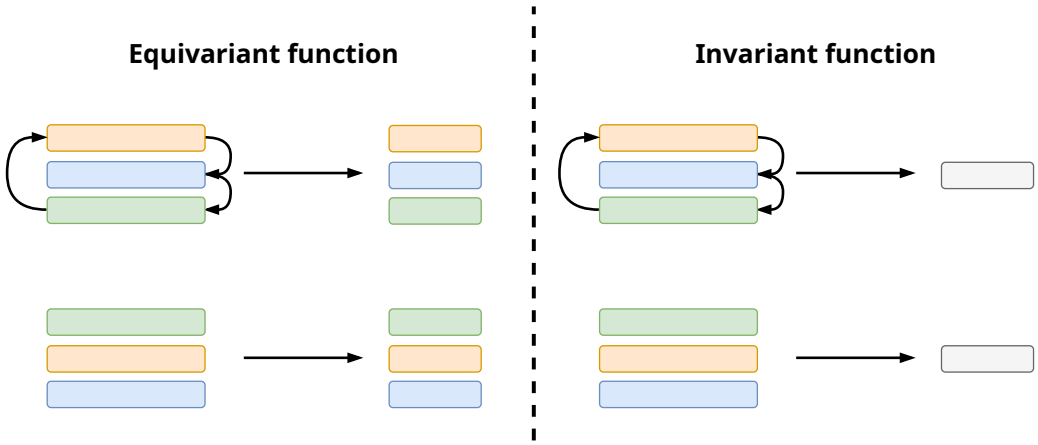

Figure 3.6: **Schematic of equivariance (left) and invariance (right).** Here, the function on the left is equivariant to rows' permutation by permuting the outputs accordingly to the input permutation. The function on the right is invariant to rows' permutations because it produces the same output for any permutation.

In the SPIDNA architecture, the equivariant function is a convolutional layer with filters of size $1 \times a$, that treats each haplotype (row) independently and computes equivariant features, while the invariant function computes the mean of these features over the row dimension. The invariant function reduces the dimension of the data to one row, which is then concatenated to each equivariant row (Figure 3.7). Therefore, the correlation between rows increases at each layer, which progressively transforms the equivariant input to an invariant output. However, the correlation increase should be moderate and progressive to avoid immediate loss of the information at the haplotype level. To promote this, two independent normalizations were performed, one over the output of the equivariant function and one over the input of the invariant function. A correlation control parameter α that quantifies the contribution of the invariant func-

tion to the next layer is added to control the speed at which the correlation increases between rows. We studied the internal variance of SPIDNA in Section 4.2.1 in order to assess the effect of α and better understand how this issue is handled during the optimization.

3.3.2 Adaptability to varying size

A major difficulty that arises with genomic data is that the number of SNP varies from one dataset to another, or from one genomic region to another, due to the stochasticity of biological and demographic processes (and of their corresponding genetic simulations). Therefore, we use convolution layers as they can handle data with variable size while keeping the number of network weights constant. A filter can be repeated more or fewer times to cover the whole input entering each layer, letting the network adapts itself to the data. Consequently, the output size of each convolution layer will vary depending on the input size. This prevents the use of fully connected layers directly after a convolution layer, as it is often the case with CNNs. Instead, we use fully-connected layers only after operations independent of the input size and with a fixed output size, namely mean functions over the column and row dimensions (Figure 3.7).

Our adaptive architecture provides an alternative to data compression based on computer vision algorithms: since compression is not optimized for the task of interest, it could induce information loss by reducing data prematurely. Note indeed that the success of deep learning in computer vision lies precisely in the replacement of ad-hoc data descriptors and processing pipelines (e.g., SIFT features to describe image key points (Lowe, 2004), and the "bag of visual words" pipeline (Sivic and Zisserman, 2003) to build an exploitable representation of them through clustering and histograms) by ones that can be optimized. It is also an alternative to padding, a technique that consists in completing the SNP and distance matrices at the edges so that they all match the biggest simulated SNP matrix; it is left to the neural network to guess where the real genetic data stops and where padding starts. As such, it may make the task more difficult, given that the SNP matrix size is highly variable between different demographic histories and some examples would contain more padding values than actual genetic information. RNN are also a natural alternative to process sequence of variable size, though they induce an unequal contribution of SNPs to the final result, depending on their ordering along the sequence. Indeed, as the information from the previous elements of the sequence is stored in the internal state of the RNN, earlier parts of the sequence can be more easily forgotten. Nonetheless, they were very recently proven to be useful to predict local recombination rate along the genome (Adrion et al., 2019) and future works should investigate whether this scales up to global characteristics and to a different task.

We designed an architecture accounting for invariance and adaptive specificities by stacking multiple equivariant blocks (Figure 3.7, label B). An equivariant block consists in one convolution layer with filters of size 1×3 that are equivariant (C3), averages of the convolution outputs across the haplotype axis (M1) and the SNP axis (M2) that are both invariant, a concatenation of the equivariant and invariant features (I3), one max pooling layer that is also adaptive to the number of SNPs (M3) and one fully-connected layer that updates the demographic predictions at each block (F1) via a sum function

(O1) (Figure 3.7).

We designed three variations of the SPIDNA permutation-invariant architecture. Layer normalization techniques help the optimization of a neural network in practice. The most common one is batch-normalization, which we applied to SPIDNA, obtaining thus a first variation of this architecture. However, batch-norm requires all input samples to have the same size, and therefore this first variation takes as input a fixed number of 400 SNPs, similarly to two of the baselines. Instance normalization (see Figure 2.3) is another normalization technique, that does not require fixed size inputs and normalize layer inputs per-data instead of per-batch (for the batch normalization). We apply it and thus obtain a second variation of SPIDNA, which is invariant to the number of SNPs. As mentioned at the end of Section 3.3.1, we also consider a variation of SPIDNA using two instance normalizations and an additional parameter α, in order to control the speed at which the network becomes invariant (when going through the layers). This network and the influence of α are studied in Section 4.2.1. The first variation using batch normalization has 110584 trainable parameters, and the other two using instance normalization have 110384.

Except for the different normalization layers and the correlation control parameter α, the three variations of SPIDNA have the same architecture represented in Figure 3.7. At each step i of the network, we consider that the data has four dimensions $B_i \times M_i \times S_i \times F_i$, B being the batch dimension, M the row dimension (also the haplotype/genotype dimension before the first layer), S the column dimension (also the SNP dimension before the first layer) and F the feature dimension (only one feature before the first layer). A first convolution layer of 50 1×3 filters is applied to the SNP matrix (Figure 3.7, label C1), and another convolution layer of 50 1×3 filters is applied to the vector of distances between SNPs (C2) and repeated M times. The results of the two convolutions have now the same dimensions and are concatenated along the feature dimension (I1). The resulting tensor is then passed to seven blocks put end to end (I2), each one involving an equivariant function and an invariant function (B). The equivariant function ψ is a convolutional layer of 50 1×3 filters (C3) that outputs a tensor of size $B_{i-1} \times M_{i-1} \times (S_{i-1} - 2) \times F_{i-1}/2$. The result of the equivariant function is then passed to the invariant function ρ, which is the mean over the dimension M (M1). Thus $\rho(\phi(X_{i-1}))$ has size $B_{i-1} \times (S_{i-1} - 2) \times F_{i-1}/2$, which is repeated M times to maintain the same dimension as $\phi(X_{i-1})$. Then $\rho(\phi(X_{i-1}))$ and $\phi(X_{i-1})$ are concatenated over the feature dimension (I3). Finally, max-pooling filters of dimension 1×2 are applied, and the result is passed to the next block (M3). In parallel, each block computes the average over the column dimension S of the 21 first features of $\rho(\phi(X_{i-1}))$ that are then passed to a fully-connected layer with 21 outputs (F1). The predictions of each block are summed (O1).

3.3.3 SPIDNA combined with ABC

ABC have already shown great results for demographic inference and has the advantage of predicting estimated posterior instead of simple point estimates. Therefore, we designed two setups to leverage the advantages of both deep learning and ABC. In the first setup, we combine ABC to our SPIDNA architecture by using the predictions made by the SPIDNA version with batch normalization already trained. This strategy

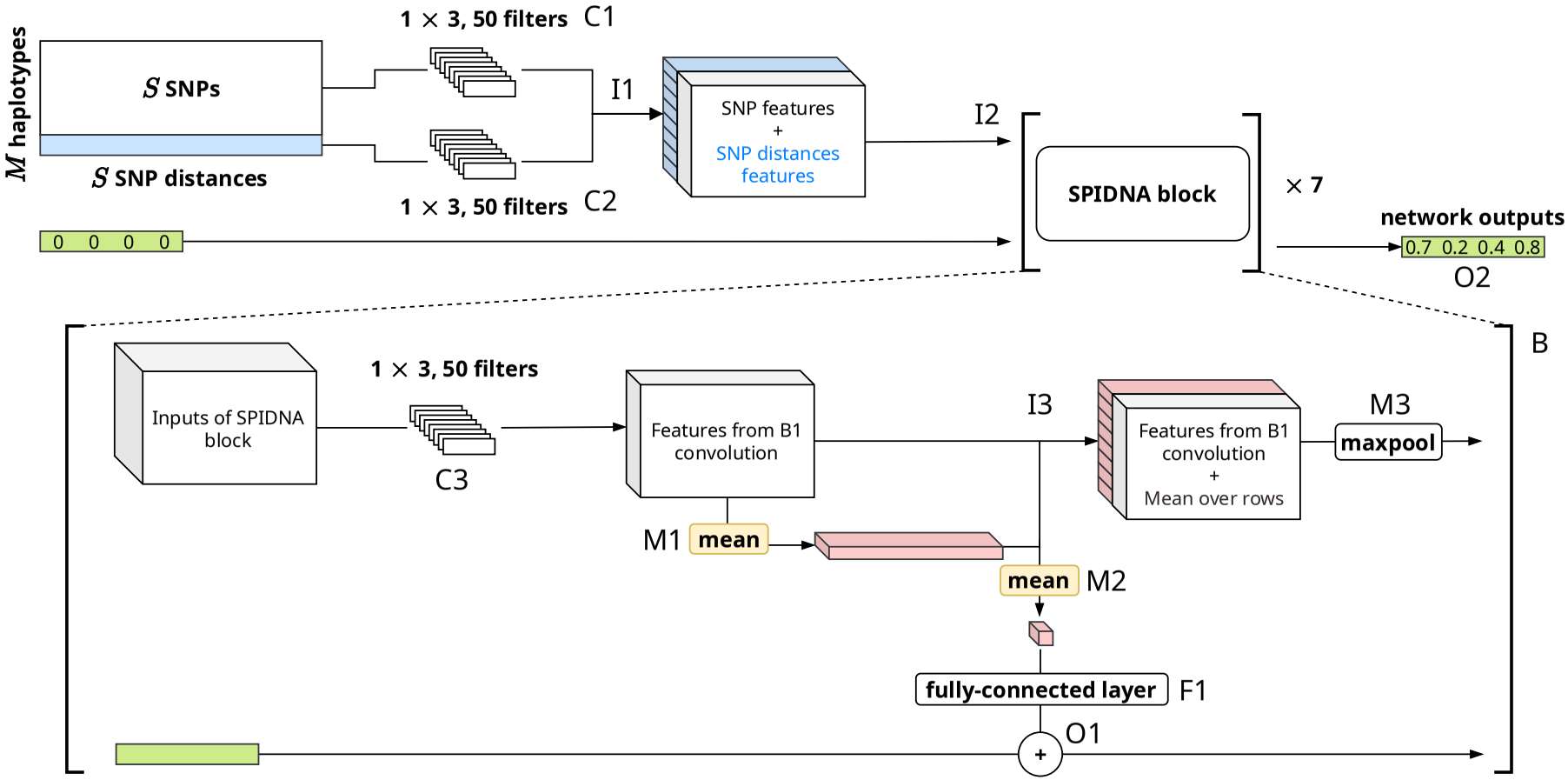

Figure 3.7: **Schematic of SPIDNA architecture.** SPIDNA takes as input a SNP matrix associated with its vector of distances between SNPs (in blue). A convolution layer is applied to the SNPs (C1) and another convolution layer is applied to the distances (C2). Results of C2 are repeated to be concatenated with results from C1 (I1). The output is passed to a series of seven SPIDNA blocs (I2). Each SPIDNA block starts with a convolution layer (C3) followed by the mean over rows of the convolution layer result (M1) and the mean over columns of M1 result (M2). The concatenation of C3 and M1 results (I3) is processed by a max pooling layer (M3) and passed to the next SPIDNA block. In parallel, the output of M2 is processed by a fully-connected layer (F1). The prediction vector (in green) is updated at each SPIDNA block with a sum (O1) of its previous value and F1 results. It is finally output by the last block as the predicted demographic parameters (O2).

was proposed by Jiang et al. (2017) who showed that a deep neural network could approximate the parameter posterior means, which are desirable summary statistics for ABC. It was applied under the name of ABC-DL in two population genetics studies for performing model selection, however both papers relied on the joint SFS as predefined candidate summary statistics (Lorente-Galdos et al., 2019; Mondal et al., 2019). Here, we are taking advantage of both the deep architecture to bypass summary statistics and the Bayesian framework to refine the prediction and approximate the posterior distribution. The statistics currently processed by ABC are the average over multiple independent regions of SPIDNA predicted population sizes. In the second setup, we added the summary statistics, that we previously computed for the ABC without deep learning, alongside to the inputs. For both setups, we applied the same hyperparameter optimization that we applied to the ABC without deep learning, i.e., we tested local linear regression, ridge regression and non-linear regression based on a single-hidden-layer neural network as correction step and six possible tolerance rates (0.05, 0.1, 0.15, 0.2, 0.25 and 0.3).

3.4 Mixed attention SPIDNA (MixAttSPIDNA)

The work presented in this section is a collaboration with Pierre Jobic.

We developed the MixAttSPIDNA architecture with the intention of increasing the expressivity of the original SPIDNA while retaining its invariance feature. One bottleneck of SPIDNA comes from its equivariant part, which is only a simple mean over the haplotype dimension of the data flowing through the network. The mean is not a very rich descriptor of a distribution of features over individuals. We thus are interested in considering additional statistical descriptors, to better extract information about the distribution of individuals. Indeed, while Lucas et al. (2018) proves that the mean is informative enough, in the sense that all permutation-invariant functions can be expressed with the Deep Sets architecture making use only of the mean, provided there are many layers enough, the number of required layers is not given. One can hope that with richer statistical descriptors in each layer, the network will not need to be as deep. Therefore, a first approach has been to also compute variance additionally to the mean, but it showed similar or worst result compared to the original SPIDNA architecture. In order to address this issue, an attention mechanism called attention hub was added in parallel to the mean. This strategy greatly improves the overall expressivity of the network while adding learnable parameters (410,786 in MixAttSPIDNA compared to 114,847 in SPIDNA) that are included in the optimization by gradient descent procedure. These parameters are reused, similarly to convolution layers, in a way that allows the network to compute complex operations.

The attention mechanism introduced here is a variation of the original self-attention mechanism from Vaswani et al. (2017); that relies on a new component called hub. In the original self-attention, an affinity is computed between every pair of input elements, which leads to a n^2 term in the overall complexity (where n is the number of individuals). Here instead, the affinity is computed between each input element and a predefined number of hubs, to reduce the complexity (now linear in n). Intuitively, these affinity values will tell how to map the values computed from each input element to the hubs.

The hubs are then mixed together and mapped back to the input space thanks to another set of affinity values. In the context of a network that has SNP matrices as input, each hub is a combination of the haplotypes and expresses a specific, learnable statistic over them. The attention mechanism allows expressing statistics over individuals that share a particular trait only.

We later improved MixAttSPIDNA by adding a similar attention mechanism to combine the outputs from the different replicates of one scenario. The predictions from the different replicates have been previously combined simply by performing a mean, which does not take into account that different replicates can contain more or less information about the demography. This final attention mechanism allows the network to express how confident it is in the prediction yielded by each replicate, in order to better combine them. This last iteration of MixAttSPIDNA being more difficult to train, a pretraining scheme was also introduced to improve convergence.

3.4.1 Attention hub

To express richer statistics, the attention hub mechanism is added to each SPIDNA block alongside the computation of the mean over the haplotypes' dimension. First, a set of keys K and values V are computed with two fully-connected layers over the features dimension of the input tensor. Then a third set of affinity values A^{in} between each element of the sequence (here the elements are a set of features corresponding to a haplotype and a SNP) and each hub is computed using a fully-connected layer with K as input. The outputs A^{in} of this fully-connected layer replace the product between Q and K from the original attention mechanism (described in Section 2.1.3). These affinity values are passed through a softmax function and multiplied to the set of value V to create the hubs H^1. This way, each hub selects individuals, according to their descriptors K, and mix their values V, in a weighted sum depending on K. In our architectures, we set up the fully-connected layer dimensions so that operation yields 10 hubs, with 50 features each and as many "pseudo-SNPs" (elements of the SNP dimension) as the inputs of the attention mechanism. Each hub processes its input data internally (through 2 fully-connected layers), independently of other hubs. Hubs are passed to another fully-connected layer (H^2) in order to match their dimensions with the output dimensions (O), in the case where the number of features computed for K and V is different from the number of features required for the outputs. The hubs then dispatch the information they computed to each individual. For this, another attention mechanism allows each individual to choose which hubs it would like to listen to (by expressing weights for each hub). Therefore, another set of affinity values A^{out} between hub and the original input data is computed with a fully-connected layer over the inputs X and then passed through a softmax. Finally, the hubs are mapped to the output space by multiplying them with this second set of affinities A^{out} and sent to the next MixAttSPIDNA block. The affinities A^{out} determine the contribution of each hub to the output. Figure 3.8 shows an overview of the attention hub mechanism.

We now consider the same dimension notation than Section 3.3.2 with B the batch dimension, M the row dimension (also the haplotype/genotype dimension before the first layer), S the column dimension (also the SNP dimension before the first layer) and F the feature dimension. The attention hub takes as input a tensor data of $B \times M \times S \times F$

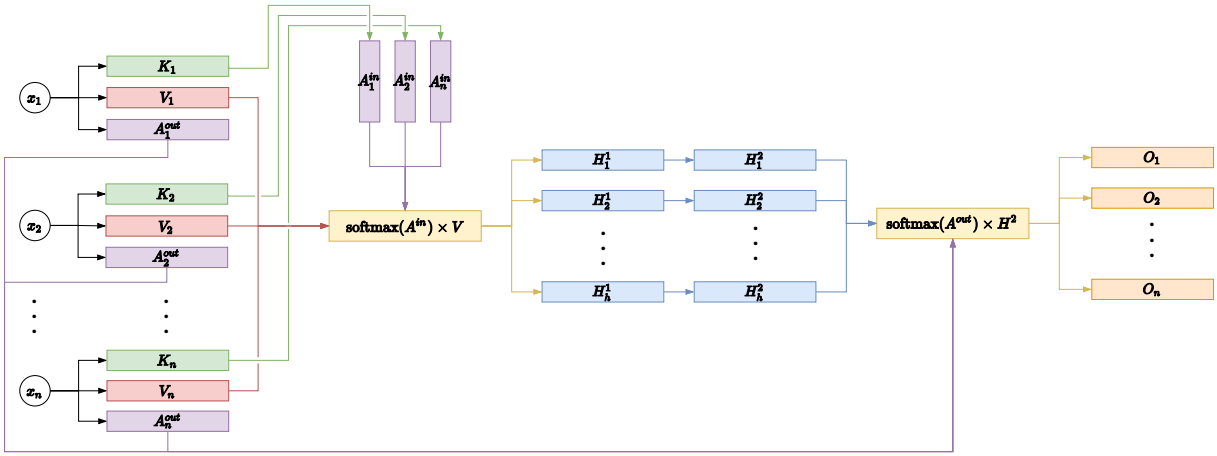

Figure 3.8: **Schematic of attention hub.** Two fully-connected layers compute keys K and values V from the attention hub inputs. A third fully-connected layer computes the affinity values A^{in} from the keys K. The product between the softmax of A^{in} and V gives the hub values H^1 (softmax$(A^{in}) \times V$). A fourth fully-connected layer computes H^2 from H^1. The last fully-connected layer computes the affinity values A^{out} used by the second attention dot product (softmax$(A^{out}) \times H^2$) to output O.

and starts by swapping M and S. In practice, we choose our fully-connected layers so that values V have dimensions of $B \times S \times M \times F_1 = 50$, keys K have dimensions of $B \times S \times M \times F_2 = 50$ and affinity values A^{in} have dimensions of $B \times S \times M \times N_{hubs} = 10$. After permutation of M and N_{hubs} in A^{in}, the matrix multiplication between A^{in} and V gives a hub tensor H^1 of $B \times S \times N_{hubs} = 10 \times F_2 = 50$, transformed by a fully-connected layer into H^2 of $B \times S \times N_{hubs} = 10 \times F_{out} = 50$ dimensions. A^{out} are computed in parallel by a fully-connected layer with the tensor data of the attention hub as input and has dimensions of $B \times S \times M \times N_{hubs} = 10$. The matrix multiplication between A^{out} and H^2 leads to the output tensor O of dimension $B \times S \times M \times F_{out} = 50$. Finally, M and S are swapped back so that the output dimensions correspond to the input ones.

3.4.2 MixAttSPIDNA architecture

The MixAttSPIDNA architecture has been build upon the SPIDNA architecture from Section 3.3. It takes the same data format as input, has the same first layers and also uses a series of blocks that updates the outputs (demographic parameters to be estimated) before passing the data to the next block. The main differences happen inside the block (now called MixAttSPIDNA block) depicted in Figure 3.9. Thanks to the addition of an attention hub mechanism, the features that are passed to the next block can now be more complex. These features of the mean M1 and attention hub A1 are mixed thanks to a fully-connected layer F3. Then, they are concatenated (I3) with the features from the convolution C3 and a max pooling layer is applied before being passed to the next block.

In parallel, the first attention hub mechanism A1 also contributes to the overall inferred values by the network. Before being projected back to the output dimensions, the first hubs are duplicated and sent to a series of two other hub attention mechanisms (A2 and A3 in Figure 3.9) with attention computed over the hubs. Finally, pre-

dictions from A3 are combined to the predictions from the mean over the haplotype (M1) and SNP (M2) dimensions thanks to a fully-connected layer (F2). The part of the network that outputs predictions based on the mean over the haplotype and SNP dimensions (F2) is the same has the original SPIDNA architecture, but its fully-connected layer maps all the features to the 21 outputs (previously only the first 21 features were mapped to the 21 outputs) as it has shown to be a simple improvement of the SPIDNA architecture during the development of MixAttSPIDNA.

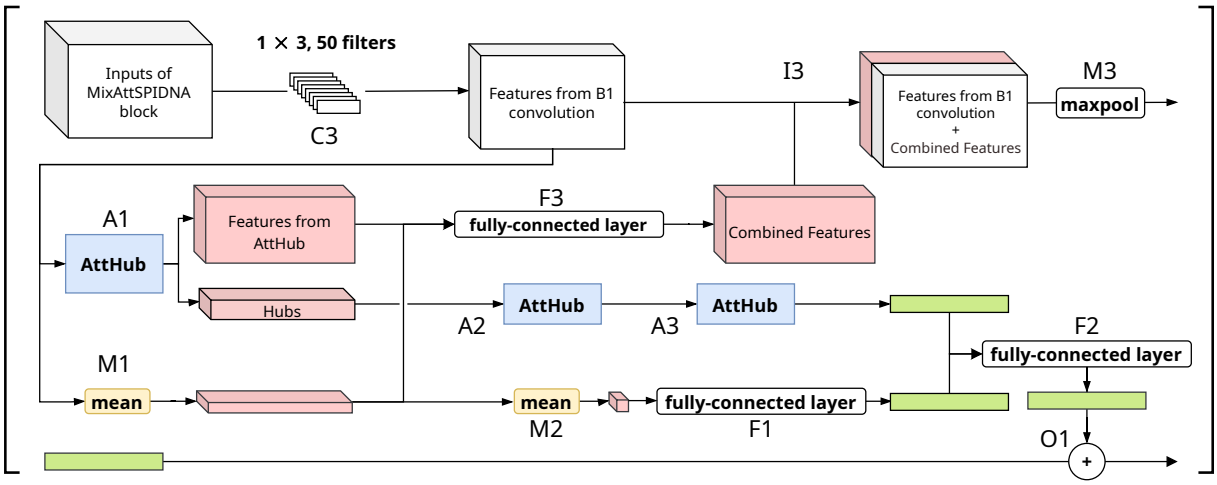

Figure 3.9: **Schematic of MixAttSPIDNA architecture block.** Each MixAttSPIDNA block starts with a convolution layer (C3) followed by the mean over rows of the convolution layer result (M1) and the mean over columns of M1 result (M2). The output of M2 is processed by a fully-connected layer (F1) to constitute the first part of the block output. In parallel, an attention hub (A1) process C3 output. The hub features from A1 are then process by two attention hubs (A2 and A3) to constitute the second part of the block output. The fully-connected layer F2 combine the two block output parts, and the result is added (O1) to the network output (in green) to be passed to the next block. The next block also takes as input the combination of C3 output and outputs of M1 and A1 with a fully-connected layer F3. Outputs of C3 and F3 are combined (I3) and passed to the next block after passing through a max pooling layer M3.

3.4.3 Inference by scenario

In order to combine the predictions performed over each replicate (representing one of the simulated or observed genomic regions) of one scenario (one specific population size history) in a more complex way than the mean previously used, another attention mechanism has been added to MixAttSPIDNA. In this version called MixAttSPIDNA with attention on scenario, each batch contains all replicates of a single scenario and nothing more. The features computed in the second layer of hubs of each block (A2 in Figure 3.9) are averaged on the hub dimension and fed to a fully-connected layer followed by a softmax. The attention vector obtained determines the contribution of each replicate to the final prediction by weighting the outputs (computed by F2 in Figure 3.9). This step finally leaves one prediction by block in the network that are combined thanks to a last fully connected layer. The idea behind this strategy is to let the network give different

weights to the different replicates for predicting the demographic parameters of a scenario. Therefore, the network could potentially learn to give more importance to the most informative replicates. Moreover, even though we have not tested this, including noise (e.g., selection or sequencing error) in some replicates during training could help the network learning to discard uninformative replicates that may be present in real datasets.

3.5 Training and hyperparameter optimization

Most methods presented in this thesis are based on deep learning and thus require to define a loss function for their training. We chose to minimize the mean squared error (MSE) over the demographic parameters inferred. We also used this metric to guide the design of the ANNs, optimize the hyperparameters and compare the methods in the next chapter (Chapter 4). Let us remind the different population size inference methods studied during this thesis. The baselines include:

- An approximate Bayesian computation (ABC) using summary statistics
- A multi-layer perceptron (MLP) using summary statistics
- A MLP using flatten SNP matrices
- A *custom* convolutional neural network (*custom* CNN) using SNP matrices
- The *Flagel* network from Flagel et al. (2018) using SNP matrices

We compare these baselines to the following variations of our SPIDNA and MixAttSPIDNA architectures:

- SPIDNA with batch normalization using SNP matrices
- SPIDNA with instance normalization using SNP matrices
- SPIDNA with instance normalization using SNP matrices with different number of SNPs
- ABC using SPIDNA predictions
- ABC using SPIDNA predictions and summary statistics
- MixAttSPIDNA using SNP matrices
- MixAttSPIDNA using SNP matrices with an attention mechanism on scenario predictions
- MixAttSPIDNA using SNP matrices with an attention mechanism on scenario predictions with an unfreezing learning strategy

Flagel network and the ANNs that we developed use the MSE over the training set as the loss metric for their backpropagation algorithms. The MSE is also computed over the validation set to guide the development of our architectures and to perform automatic hyperparameter optimization. ABC and MLP hyperparameters have been optimized with random or grid searches. In order to include more hyperparameters and cope with the longer training time of the SPIDNA architecture, we used a more advanced hyperparameter optimization procedure called HyperBandSter (Falkner et al., 2018b; Li et al., 2016). However, we did not yet reuse this procedure for the MixAttSPIDNA architecture because we preferred to focus first on substantial changes in the architecture structure (which is not easily amenable to hyper optimization) and because these newly designed architectures already outperformed previous methods (as we will see in the next chapter). We also tried different learning rate strategies for the MixAttSPIDNA architecture with attention mechanism on scenario predictions because we trained it in two steps, first to infer demographic parameters by replicate, and second to infer them per scenario (i.e., exploiting all replicates of a scenario).

3.5.1 Mean squared error (MSE)

Before computing the prediction error, the demographic parameters $\Theta^\star$ are standardized with the following formula:

$$\Theta = \frac{ln(\Theta^\star) - \mu_{train}}{\sigma_{train}} \tag{3.2}$$

where μ_{train} and σ_{train} are the mean and standard deviation of $ln(\Theta^\star)$ over the training set.

Then, each method is evaluated using its prediction error given by the following mean squared error:

$$\frac{1}{I \times J} \sum_{i,j}^{I,J} \left(\hat{\Theta}_j^i - \Theta_j^i \right)^2 \tag{3.3}$$

where Θ_j^i and $\hat{\Theta}_j^i$ are respectively the true and predicted standardized population size for the time window i and scenario j, $I = 21$ is the number of time windows and J the number of scenarios in the set. For inference based on raw data and neural networks that perform one prediction by replicates, the prediction $\hat{\Theta}_j^i$ is given by the average of the population sizes $(\hat{\Theta}_{jr}^i)_{r=1,\dots,nrep}$ estimated for each replicate (independent region) r. The two version of MixAttSPIDNA that perform prediction by scenarios apply an attention mechanism over the predictions of all scenario replicates to obtain the prediction $\hat{\Theta}_j^i$.

3.5.2 Automated hyperparameter optimisation

Compared to other machine learning methods, ANNs have a potentially infinite amount of hyperparameters when including for instance the number of layers, the number of neurons in each of them, the learning rate, weight decay or the batch size. Moreover, a run over a full dataset with enough epochs to reach convergence is time-consuming for

networks with a complex architecture defined by many learnable parameters. Therefore, the development of deep learning architectures often relies on the experience and intuition of the practitioner in a try-and-repeat process. Grid search and random search are two strategies for exploring the hyperparameter space uniformly. They are commonly used but are limited by the computing resources available. In our study, we used HpBandSter, a package that implements the HyperBand (Li et al., 2016) algorithm to run many hyperparameter trials on a smaller resource budget (i.e., few epochs) and runs the most promising trials on a greater budget. Combined with BOHB (Falkner et al., 2018b), a Bayesian optimisation procedure that models the expected improvement of the joint hyperparameters, this method provides more guided and faster search of the hyperparameter space. At each step, BOHB draws a new combination of hyperparameter values to be tested according to the expected improvement and to a predefined prior. Here, we performed a search in a 5-dimensional space defined by uniform priors over the type of architecture (architectures from our baselines and variations of SPIDNA architecture, based on 400 SNPs or the full number of SNPs), the learning rate, the weight decay and the batch size. For SPIDNA architectures that controlled correlation, we added the control parameter α to the Bayesian optimization procedure with a log-uniform prior between 0.5 and 1. The search was performed for 3 budget steps and replicated 5 times, leading to a total of 83 successfully trained networks. The results of this procedure are shown in Section 4.1.1.

As the training time of the MLP using summary statistics was short because of its small input size and number of parameters, we optimized its hyperparameters with a random search by drawing 27 configurations from uniform distributions and trained a network for each configuration during 6 epochs. The batch size was drawn between 10 and 100, learning rate between $5 \cdot 10^{-5}$ and $1 \cdot 10^{-2}$ and weight decay between $5 \cdot 10^{-5}$ and $1 \cdot 10^{-2}$.

For ABC, the tolerance rates ranged from 0.05 to 0.3 by step of size 0.05 and were optimized for 12 ABC algorithms independently (4 correction methods $\times$ 3 types of inputs: predefined summary statistics, SPIDNA outputs or both).

3.5.3 Learning rate strategies of MixAttSPIDNA

The work presented in this section is a collaboration with Pierre Jobic.

We tested numerous versions of MixAttSPIDNA during its development, but we will only focus on three versions: MixAttSPIDNA, MixAttSPIDNA with attention on scenario and MixAttSPIDNA with attention on scenario unfreezing. They all include a learning rate decay strategy that divide by two their learning rate after five epochs to improve the final prediction error. When trained on the cattle dataset, the three versions of MixAttSPIDNA use the same batch format as SPIDNA with batch normalization (batches of SNP matrices with 50 haplotypes and 400 SNPs). However, because the HGDP dataset has a number of haplotypes that varies for each scenario to mimic the different sample sizes of the real HGDP populations. We tested different strategies of mini-batch formatting on the cattle dataset (see Section 4.2.4) and used the best for the HGDP dataset. MixAttSPIDNA with attention on scenario and MixAttSPIDNA with attention on scenario unfreezing are trained in two steps. First they are trained to predict demographic parameters replicate-wise like MixAttSPIDNA for ten epochs. Then, they are trained to pre-

dict demographic parameters scenario-wise, either by only optimizing the weights and biases used for scenario predictions (MixAttSPIDNA with attention on scenario) or by giving a higher learning rate to the weights and biases used for scenario (MixAttSPIDNA with attention on scenario unfreezing). This scheme is meant to stabilize the learning when the network task moves from predicting replicate-wise to scenario-wise. In experiments not shown in this manuscript, training scenario-wise MixAttSPIDNA without pretraining on replicates have shown unstable behaviours with error increases on the training set. MixAttSPIDNA has 410,786 learnable parameters, MixAttSPIDNA with attention on scenario and MixAttSPIDNA with attention on scenario unfreezing have 414,076 learnable parameters.

3.6 Interpreting deep neural networks with Canonical Correlation Analysis (CCA)

Whether neural networks are artificial or biological and despite being very different, they are often seen as a blackbox difficult to interpret because they perform an important amount of complex operations on the input data which could have more or less importance for the final prediction. The interpretability of neural networks is a very active research area as it helps to better understand how they work and possibly to improve how they should be build. In most fields of deep learning applications, it is also crucial to provide human understandable explanations for any prediction made by the network. Medicine diagnosis made by ANNs are a famous example where the physician needs to understand the inner reasoning of the ANN to avoid any mistakes that could have been easily avoided by human. Nonetheless, neuroscientists have been tackling this goal from a theoretical and an experimental angle since the discovery of brain cells, but are still far from a complete theory.

Understanding ANNs should be in principle much simpler, as they are less complex than their biological counterparts and operate in the controlled environment of computers. Some of the approaches that have been recently developed generate a more interpretable model from the ANNs such as a decision tree with semantic annotations (Zhang et al., 2019). Others aim at generating saliency maps or mask over the input data to highlight its most important features with variants of backpropagation and gradient analysis (Chattopadhay et al., 2018; Selvaraju et al., 2019). These last methods are mostly used for image data but can also be used to highlight binding motifs (Shrikumar et al., 2017), identify variants associated to a particular trait (Sharma et al., 2020) or show alleles under selection in DNA sequences.

ABC methods for population size inference are based on a set of handcrafted summary statistics computed from the SNP data, and the method developed here seeks to find if a network computes the same summary statistics internally. This should help to understand what are the most relevant summary statistics for predictions, but also which ones are not computed by the network and could be added as input alongside the raw genomic data in order to improve the predictions. Searching for such correlations between the ANN activations and the summary statistics is a difficult task as the network potentially computes combinations of summary statistics in a complex non-linear fashion and the computation of a summary statistics can be performed by the combi-

nation of multiple parts of the network. Some methods applied to computer vision already seek for high correlations between sets of activations and a single summary statistic (often called concept), but they do not seek for correlations with a combination of multiple summary statistics (Graziani et al., 2018; Kim et al., 2018).

The method presented here is based on canonical correlation analysis (CCA) and inspired by SVCCA, a method designed to understand ANN training dynamics by comparing the activations of a network at different training stages (Raghu et al., 2017). Here, the CCA is not performed between two sets of activations, but between a set of activations and a set of summary statistics. Activations are grouped by layer and each layer is compared to a set of 279 summary statistics suggested by Jay et al. (2019), including bins of site frequency spectrum (SFS), identity by state (IBS), linkage disequilibrium (LD) and other such as expected heterozygosity, Tajima's D and nucleotide diversity π. Similarly to Raghu et al. (2017), the method first uses singular value decomposition (SVD) with a variance conserved threshold of 0.99 to reduce the dimensionality of the activations when greater than 50 in a layer. This helps to reduce noise and spurious correlations in the next step of the analysis. SVD is preferred over other dimensionality reduction methods because it does not standardize the data, and thus performs well on sparse data such as neuron activations in ANNs with ReLU activation function.

Then, the sets of reduced activations and summary statistics are analyzed with CCA. The two sets after standardization are denoted by $X_a \in \mathbb{R}^{n\times p}$ and $X_b \in \mathbb{R}^{n\times q}$ with n the number of samples and q and p the numbers of variables observed in each set. CCA searches for linear combinations of the features (columns) of X_a and X_b that have maximal correlation with each other. That is, CCA searches for two vectors $\vec{w}_a \in \mathbb{R}^p$ and $\vec{w}_b \in \mathbb{R}^q$ (the canonical weight vectors) such that the resulting linear combinations $\vec{z}_a = X_a\vec{w}_a$ and $\vec{z}_b = X_b\vec{w}_b$ (that are 2 real data-sample-dependent values) are the most correlated (across samples).

The algorithm seeks $\vec{w}_a$ and $\vec{w}_b$ that minimize the enclosing angle between $\vec{z}_a$ and $\vec{z}_b$, $\theta \in [0, \frac{\pi}{2}]$, with the constraint that $\vec{z}_a$ and $\vec{z}_b$ are unit norm vectors. The cosine of this angle (also referred as the canonical correlation) is given by the formula:

$$\cos(\vec{z}_a, \vec{z}_b) = \frac{\langle \vec{z}_a, \vec{z}_b \rangle}{\|\vec{z}_a\| \|\vec{z}_b\|} = \langle \vec{z}_a, \vec{z}_b \rangle \tag{3.4}$$

The first pair of $\vec{z}_a^1$ and $\vec{z}_b^1$ corresponds to the smallest angle θ_1 given by:

$$\cos\theta_1 = \max_{\vec{z}_a, \vec{z}_b} \langle \vec{z}_a^1, \vec{z}_b^1 \rangle, \quad \|\vec{z}_a^1\|_2 = 1 \quad \text{and} \quad \|\vec{z}_b^1\|_2 = 1 \tag{3.5}$$

Then the next enclosing angle θ_r is found in the orthogonal complements of $\vec{z}_a^{r-1}$ and $\vec{z}_b^{r-1}$. The pair $\vec{z}_a^r$, $\vec{z}_b^r$ is defined by :

$$\cos\theta_r = \max_{\vec{z}_a, \vec{z}_b} \langle \vec{z}_a^r, \vec{z}_b^r \rangle, \quad \|\vec{z}_a^r\|_2 = 1 \quad \text{and} \quad \|\vec{z}_b^r\|_2 = 1 \tag{3.6}$$

with the constraints:

$$\langle \vec{z}_a^r, \vec{z}_a^j \rangle = 0, \quad \langle \vec{z}_b^r, \vec{z}_b^j \rangle = 0, \quad \forall j \neq r \tag{3.7}$$

There exist several methods to compute the canonical variates, with some of them extending CCA to non-linear correlation (Akaho, 2006; Andrew et al., 2013). Here, the

CCA implemented in scikit-learn (Pedregosa et al., 2011; Wegelin, 2000) was used to seek for linear relationships between the two sets in order to interpret the correlations more easily. Finally, we compared a variable from a set to all variables from the other one, by averaging the correlation between the canonical variates pondered by the weights of the variable. These weights can be interpreted as the contribution of the variable to each canonical variates.

This method was applied as a proof-of-concept to the *custom* CNN. Activations were measured after training for the validation set in the five last layers of the CNN (the computation time being too long for the first two layers because of their large number of activations) and 50 canonical variates were computed for each layer.

3.7 `dnadna`: a python package for deep learning applied to population genetics

Alongside the development of new deep learning architectures for demography inference in population genetics, we released a python software called `dnadna`, supported by a pre-print (Sanchez et al., 2021a). It is task-agnostic and aims at facilitating the development, reproducibility, dissemination, and reusability of neural networks designed for genetic polymorphism data.

`dnadna` defines multiple user-friendly workflows. First, users can implement new architectures and tasks, while benefiting from `dnadna` input/output and other utility functions, training procedure and test environment, which not only saves time but also decreases the probability of bugs. Second, implemented networks can be re-optimized based on user-specified training sets and/or tasks. Finally, users can apply pretrained networks in order to predict evolutionary history from alternative real or simulated genetic datasets, without the need of extensive knowledge in deep learning. Thanks to `dnadna`, newly implemented architectures and pretrained networks are easily shareable with the community for further benchmarking or applications.

`dnadna` comes with a peer-reviewed exchangeable neural network allowing demographic inference from SNP data, that can be used directly or retrained to solve other tasks. Toy networks are also available to ease the exploration of the software, and we expect that the range of available architectures will keep expanding thanks to contributions from the community.

Availability: `dnadna` repository is available at `https://gitlab.com/mlgenetics/dnadna` and its associated documentation at `https://mlgenetics.gitlab.io/dnadna/`.

3.8 Chapter conclusion

This chapter gave a description of all the materials and methods that have been used or developed through this thesis. We built two tools with the intention of helping the population genetic community to develop new deep learning architectures. The first one uses the CCA to better understand which kind of features a network learns to compute on genomic data. The second one is a python package designed to facilitate the development, usage, and distribution of ANNs processing SNP data. The main focus of

this thesis is the development of deep learning methods for population size inference. To this end, we designed priors for our demographic scenario simulations to be as close as possible to the information known about the real cattle and human datasets that we studied, and generated simulations accordingly. Our strategy for developing the two main architectures (SPIDNA and MixAttSPIDNA) was to take into account as much as possible the characteristics of population size inference and could be adapted to other population genetic tasks. We created and implemented architectures that are invariant to haplotype permutations and changes in SNP number. Then, we designed attention hubs in order to make the MixAttSPIDNA network more expressive while keeping the computational time reasonable and without breaking the previously mentioned characteristics of the network. The attention mechanism also allowed to combine predictions from different segment alignments of the same sample of individuals in a way that allows the network to give more or less importance to each segment in the final prediction. The comparison baselines include a MLP and an ABC that use summary statistics as input, the *Flagel* network, and ANNs that are less adapted to the input data (MLP and *custom* CNN). By comparing them to SPIDNA and MixAttSPIDNA, we will see in the next chapter whether the architecture choices made translate into better predictions.

Chapter 4
Inferring demography from genomic data

Contents

We evaluate and compare the methods described in the previous chapter thanks to the real and the simulated datasets. This chapter presents and analyses the results to assess which choices in the architecture design improve the predictions. The second section presents the results of several procedures aimed at better understanding the ANN inner workings and how they behave under changes in test or real data that were not present in the training set. Finally, the last section includes SPIDNA's and MixAttSPIDNA's predictions for the two real datasets, in order to show their usage in practice.

4.1 Study of ANN performances on simulated data

We simulated two main datasets with priors designed to fit the real data from the cattle and HGDP datasets. We used these datasets to train and compare the baseline methods to the different versions of SPIDNA and MixAttSPIDNA architectures by evaluating

their mean squared errors (MSE) on validation and test sets. We also compared their predictions for specific simulated demographic scenarios. The MSE has been used to guide the design of the multiple versions of these two architectures by iteratively introducing new mechanisms and evaluating their impact on this metric.

4.1.1 SPIDNA hyperparameter optimization

As explained in Section 3.5.2 of the previous chapter, finding the best hyperparameters of an ANN mainly relies on the practitioner intuition because of the infinite amount of configurations possible and the training time of most architectures. However, we can still apply an automatic optimization to some hyperparameters in order to improve predictions. Therefore, we used the HpBandSter procedure on baseline ANNs and SPIDNA architectures. The best configuration in terms of loss corresponds to the SPIDNA architecture processing 400 SNPs with batch normalization, a weight decay of $2.069 \cdot 10^{-2}$, a learning rate of $1.416 \cdot 10^{-2}$ and a batch size of 78 (Figure 4.1). Configurations with large batch sizes tended to yield lower losses (Figure 4.1), which is expected, as large batches provide a better approximation of the full training set gradient. However, a batch size too close to the training set size can lead to overfitting the training set. Here, we did not observe overfitting for any run when monitoring training and validation losses. This is probably due to the large size of our simulated datasets, as training often converges even before seeing the full dataset. The best configurations also tended to have low learning rates and weight decays (Figure 4.1). These low values slow down the convergence, but usually decrease the final prediction error if the budget (i.e., number of training epochs) is high enough for the network to reach convergence.

The Bayesian hyperparameter optimization procedure allowed to test multiple networks (MLP, *custom* CNN with heterogeneous filter sizes, SPIDNA with different normalization schemes, adaptive or not to SNP number) thanks to a better usage of the computational power available by giving more budget to the most promising ANN architectures and hyperparameters. Note that it would be possible to extend this procedure to hyperparameters that further describe the architecture of the network, such as the number and type of layers, number and type of neurons, the type of non-linearity or the topology. However, even if this optimization of hyperparameters is faster than simpler methods such as random or grid search, it still requires an important amount of computational power, and many GPUs to perform each run in parallel. Another caveat is that these runs have been performed on budgets that represent few epochs, and it is possible that a longer training would reveal another best set of hyperparameters. Nonetheless, the barplots in Figure 4.1 show very similar scores regardless of the extended number of epochs for the best architecture (SPIDNA with batch normalization). For all the reasons above, we did not rely on hyperparameter optimization for the development of MixAttSPIDNA, but rather focused on more radical changes related to the architecture. Indeed, those changes led to more substantial differences in scores than fine-tuning hyperparameters, such as the batch size, the learning rate or the weight decay, since the latter led to moderate differences in score.

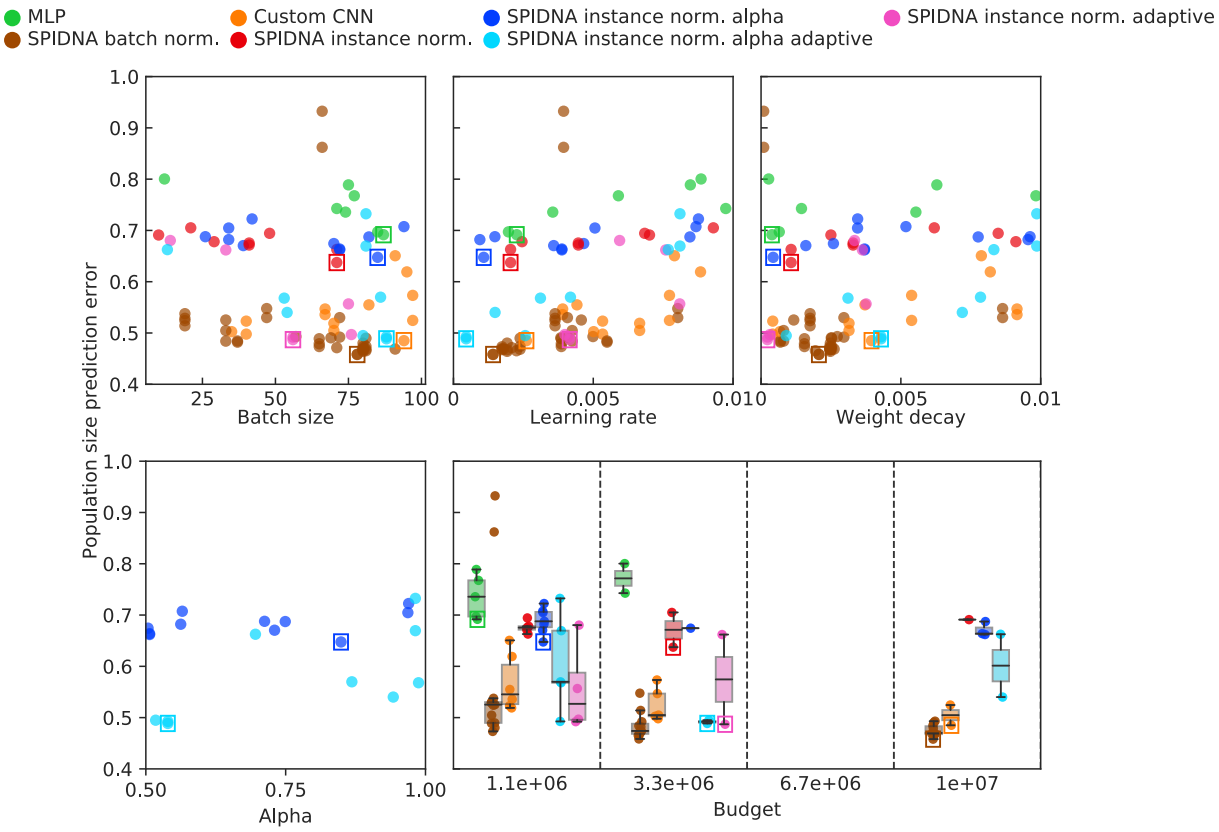

Figure 4.1: **Population size prediction error for each run of the hyperparameter optimization procedure.** X-axes indicate the hyperparameter (batch size, learning rate, weight decay and alpha) or budget values, and colors indicate the type of network used for the run (MLP, *custom* CNN and multiple SPIDNA architectures). For each network the best run is surrounded by a square.

4.1.2 Results on predefined scenarios

The performances of SPIDNA, SPIDNA combined with ABC and MixAttSPIDNA are illustrated on a subset of cattle demographic scenarios (Figure 4.2) that were previously investigated by Boitard et al. (2016b) (see Figure A.1). Six scenarios were simulated: "Medium", "Large", "Decline", "Expansion", "Bottleneck" and "Zigzag" by specifying the demographic parameters instead of drawing them from the prior of the training set. SPIDNA correctly reconstructed histories of constant size, expansion and decline, as SPIDNA predictions from 100 independent genomic regions (Figure 4.3) approximately followed the real population size trend and magnitude. In Figure 4.4, the true parameters were always included in the 90% credible intervals (light green envelopes) predicted by SPIDNA combined with ABC without predefined summary statistics and, in most cases, in the 50% credible intervals (dark green). Similarly to the inferred values by ABC in Boitard et al. (2016b), the credible interval increases during the most ancient times for all scenarios except "Large" and "Decline". These time steps are older than the TMRCA (see Figure A.1 for TMRCA estimations) which can explain this increase, as most information in the sample is lost beyond this point. SPIDNA and SPIDNA combined with ABC correctly reconstructed a complex history consisting in an expansion interrupted by a bottleneck and followed by a constant size (see Figure 4.2 "Bottleneck"), but SPIDNA predicted an earlier and weaker bottleneck than the true scenario. However, both methods were unable to correctly estimate the parameters of a very complex "Zigzag" history, except for its initial growth period, and instead reconstructed a smoother history with values intermediate within the range of the lower and higher population sizes (see Figure 4.2 "Zigzag"). This confirmed the smoothing behavior identified previously for ABC and MSMC on these demographic scenarios (Boitard et al., 2016b) (Figure A.1). Similarly to ABC on predefined summary statistics (Boitard et al., 2016b), SPIDNA predictions of very recent population sizes were slightly biased toward the center of the prior distribution, however combining SPIDNA with ABC tended to correct this bias in most cases. MixAttSPIDNA performed better than SPIDNA by including the true effective size in almost all boxplot of each scenario, as shown by Figure 4.5. However, similarly to SPIDNA, it failed at precisely predicting the most recent population sizes for some scenarios. On the opposite of our two other methods, MixAttSPIDNA was able to reconstruct the most ancient bottleneck of the "Zigzag" scenario (Figure 4.5).

The experiments shown in this section have been conducted to have an insight on the predictions made by each architecture. Although the prediction error on a large dataset, which will be presented in the next section (Section 4.1.3), can be a good metric to compare them, the simulations used to compute it might include many unrealistic scenarios on which the architecture performs well. This might lower the overall prediction error without improving the predictions on realistic simulated scenarios and thus, on real scenarios. These experiments are also important to assess whether the predictions are capable to capture the scenario dynamics, by reconstructing the expansion, decline, bottleneck and stable phases. This last point is crucial because most studies of population size evolutions try to link ecological or historical events to changes in the population size dynamic, rather than predicting very precisely the effective population size. Here, the MixAttSPIDNA architecture, which is the best of the three architectures compared in terms of prediction error (see next section), is also the only architecture that has been able to identify the most ancient bottleneck in the "Zigzag" scenario.

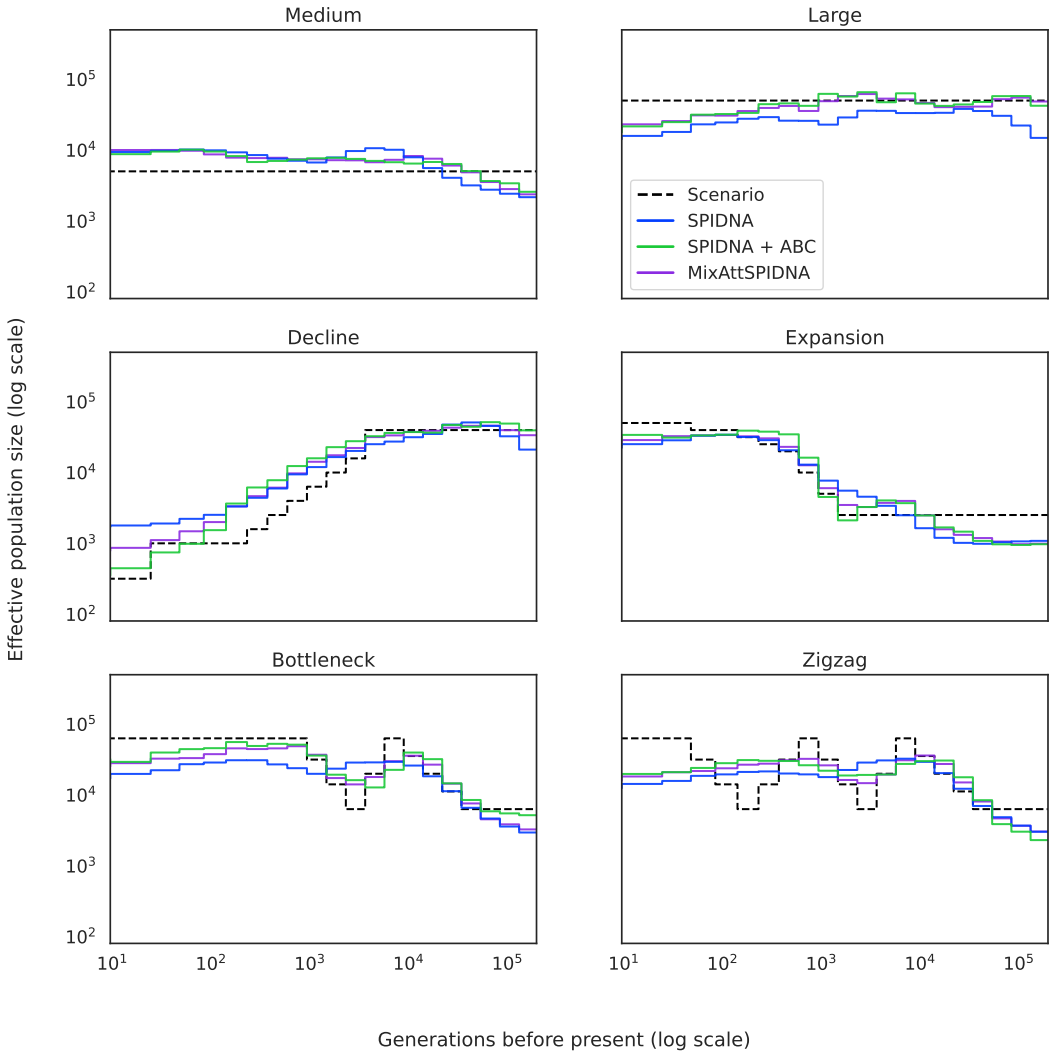

Figure 4.2: **Predictions of SPIDNA, ABC using SPIDNA outputs and MixAttSPIDNA, all trained on the simulated cattle dataset for six predefined scenarios (dashed black lines).** 100 replicates were simulated for each scenario and predictions were averaged.

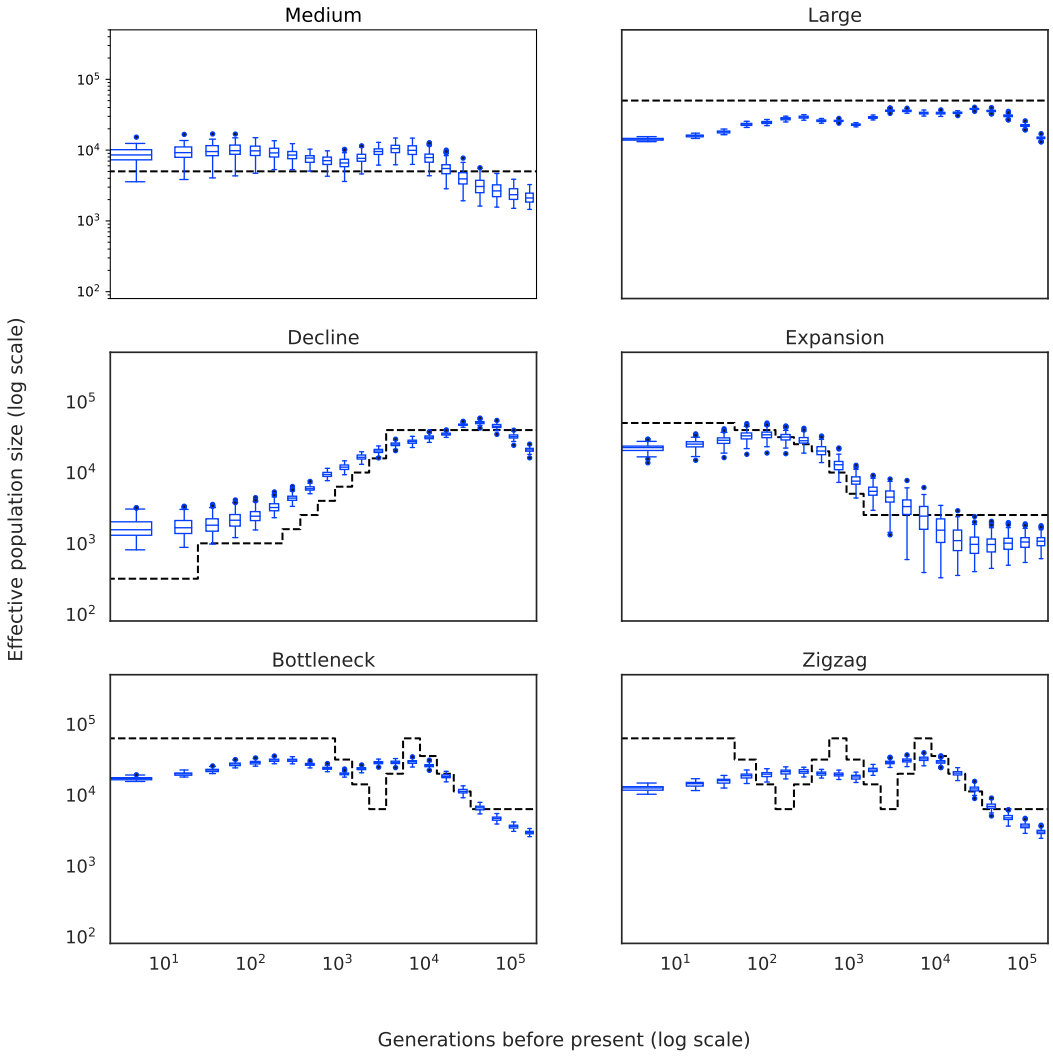

Figure 4.3: **Predictions of SPIDNA trained on the simulated cattle dataset for six predefined scenarios (dashed black lines).** 100 replicates were simulated for each scenario. Boxplots show the dispersion of SPIDNA predictions over replicates.

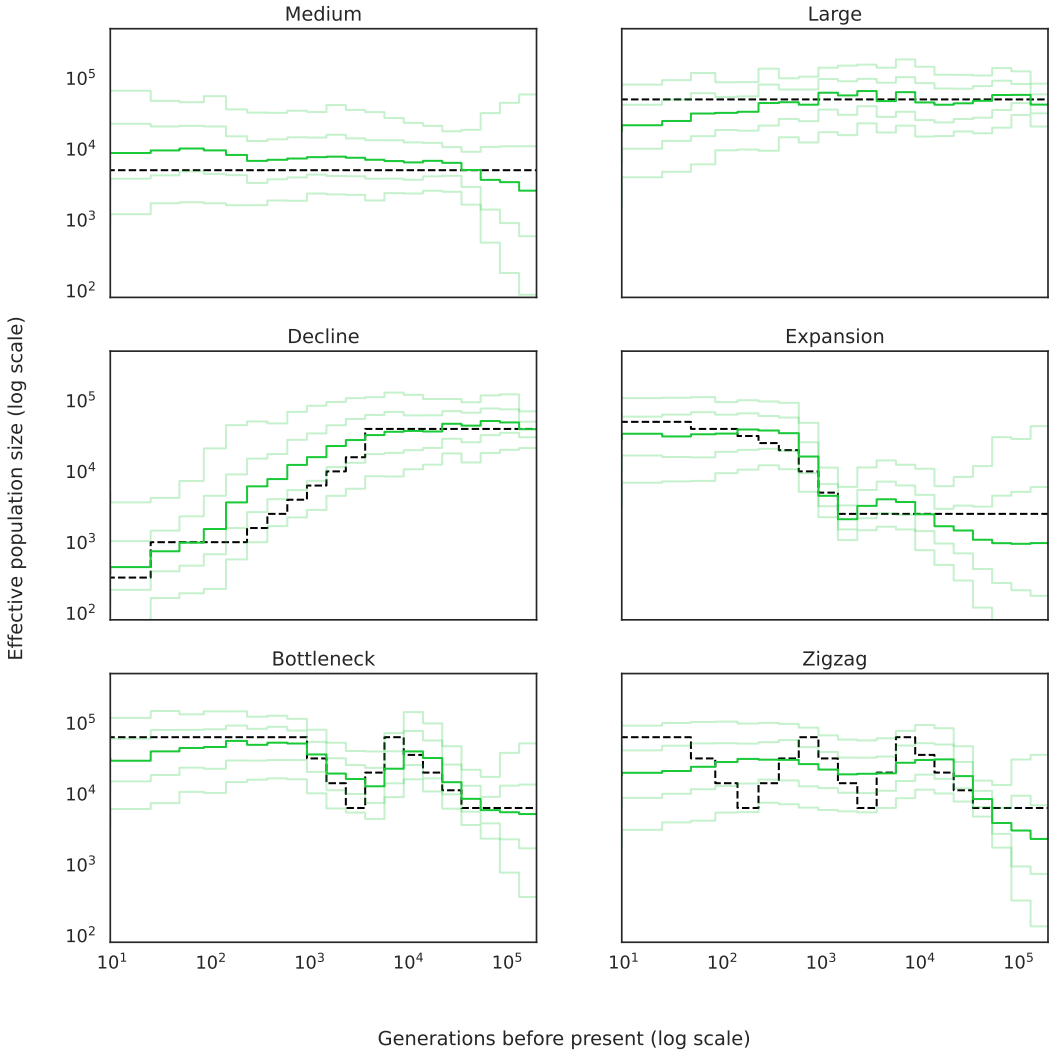

Figure 4.4: **Predictions of ABC using SPIDNA outputs trained on the simulated cattle dataset for six predefined scenarios (dashed black lines).** 100 replicates were simulated for each scenario. For each history inferred by SPIDNA combined with ABC, the posterior median is display with plain green line, the 50% credible interval (dark green) and the 90% credible interval (light green).

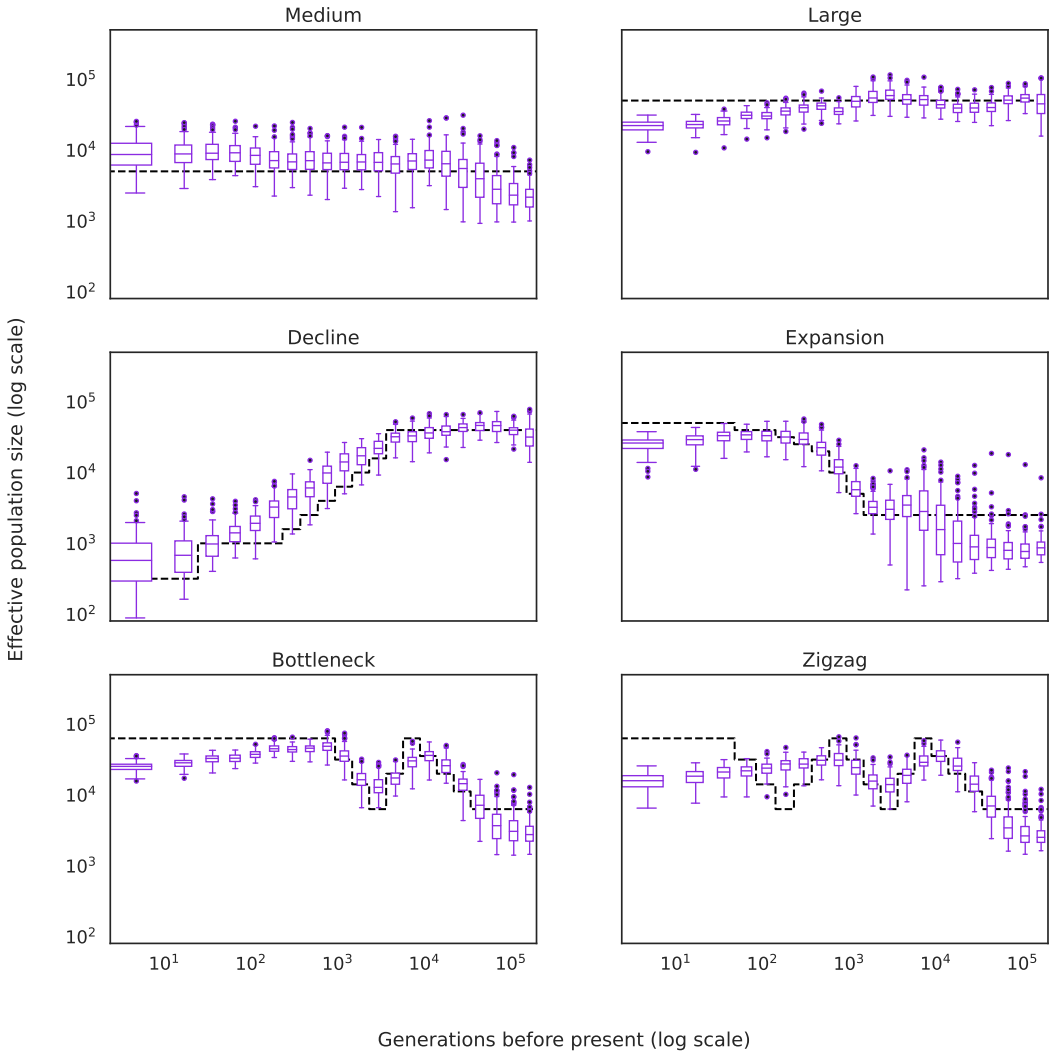

Figure 4.5: **Predictions of MixAttSPIDNA trained on the simulated cattle dataset, for six predefined scenarios (dashed black lines).** 100 replicates were simulated for each scenario. Boxplots show the dispersion of SPIDNA predictions over replicates.

4.1.3 Prediction error on the whole set of simulated datasets

The first part of this section presents the results obtained on the simulated cattle dataset with all the methods from the baselines, all versions of SPIDNA and MixAttSPIDNA. Second, our best version of SPIDNA and all versions of MixAttSPIDNA are trained and compared on the simulated HGDP dataset.

To interpret the results and compare them, let us first note that in Figure 4.6, Tables 4.1 and 4.2, a 0 error means perfect prediction, while an error of 1 means that no information is extracted from the input. Indeed, a function outputting always the same value, for all samples, can at best predict the average target value over the dataset, in which case the mean squared error (also referred to as the prediction error) is the standard deviation over the dataset of the value to predict, which is normalized to 1 in our setup, as explained in Section 3.5.1.

Simulated cattle dataset

The hyperparameter optimization procedure of ANNs considered a certain number of architectures, for each of which we selected and trained the best version with a greater budget (i.e., during 10 epochs), allowing for an in-depth comparison to the baseline methods. This longer training did not yield any substantial decrease in prediction error compared to their counterparts with a smaller 10^7 budget (10^7 training SNP matrices, i.e., 5.57 epochs) (Figures 4.1 and 4.6). Prediction errors for the validation set (used in the hyperparameter optimization procedure) and the test set (never used before) are shown in Table 4.1. In the following text, each method is designated along its index in Table 4.1.

Summary statistics based methods The prediction errors achieved by ABC using summary statistics ranged from 0.496 (index 0, ABC rejection, i.e., without correction) to 0.364 (ABC neural networks, index 3). The gap of performance between the ABC without and with correction shows that the data points selected by the rejection are structured in the demographic parameters space. The small difference between linear corrections (linear and ridge, 0.369 and 0.376, indexes 1 and 2) and the non-linear one (0.364, ABC neural networks, index 3) seems to indicate that the relation between data points (summary statistics) and demographic parameters is mostly linear in the neighborhood of each validation or test example.

The MLP based on summary statistics performed worse than ABC with correction (0.437, index 4). It is slightly overfitting on the validation set (0.399 validation error versus 0.437 test error, index 4), which may be due to the hyperparameter optimization procedure, but we did not investigate this phenomenon further.

Overall, we obtained good predictions with summary statistics based methods, especially for the ABCs including a correction step. Therefore, the summary statistics chosen seemed to preserve a great part of the relevant information from the dataset. Compared to ANNs, ABC methods have the advantage of constructing an approximate posterior of the demographic parameters that allows to construct credible intervals, as in Figure 4.4. However, ABC can be time-consuming when inferring the demographic parameters of numerous populations at once, because it requires to repeat the rejec-

tion algorithm (i.e., compares the observed data to all the data from the training set) for each population. Note that, for the exact same reason, ABC is also expansive to benchmark on large validation and test sets. On the other side, ANNs have only one long training phase, while the inference phase is fast and scales easily when applied to many populations. Relying on summary statistics also has some disadvantages: (i) the loss of information is possibly impeding further improvement of the prediction error, despite improvements in the inference method itself; (ii) it is difficult to know which summary statistics should be added to the dataset in order to improve inference (a solution to this issue is to design them automatically, as we do with SPIDNA combined to ABC); (iii) for the most recent biological questions and data types, expert summary statistics might not yet exist; (iv) the computation of certain types of summary statistics on all the different datasets can be particularly time-consuming (for instance, the linkage disequilibrium is in practice computed on a subsample of SNPs because measuring the pairwise dependence between SNPs at different distances is computationally too expensive on large dataset).

Baselines ANNs The MLP based on raw data performed very poorly (0.675, index 5) and all other networks based on raw data outperformed this MLP. This is not surprising, since genomic information is encoded as a simple list of values, where the order has no meaning from the MLP point of view, which then cannot exploit information given by the data structure. This MLP configuration has two major drawbacks: (i) the number of network parameters to estimate is high; (ii) the MLP can only retrieve the geometry of the data through training, with no guarantee that it will learn the spatial structure of the genome (i.e., the column order and distance between SNPs) or distinguish from which individual comes each SNP. In spite of all these hindrances, the MLP still performed far better than random guesses or constant prediction (32% better).

The *custom* CNN obtained results similar results as the ABC without correction (0.487, index 6) and that, without the disadvantages of summary statistics enumerated previously. The mixed size filters have proven useful in the Computer Vision literature, under the name of Inception architectures (Szegedy et al., 2017); they allow the extraction of a mixture of different kinds of information from multiple scales within the same layer. The large gap in performance between a simple MLP and this *custom* CNN confirms the importance of such considerations. A natural extension would be to integrate this feature into SPIDNA, our permutation-invariant architecture.

The *Flagel* CNNs adapted from Flagel et al. (2018), that were not using dropout, had average test losses of 0.541 when based on the first 400 SNPs (index 7) and 0.444 when based on 1784 downsampled SNPs (index 8). The two using dropout achieved prediction errors of 0.609 (with 400 SNPs, index 9) and 0.484 (with downsampling, index 10). We can see that, for this network, downsampling the number of SNPs lead to better predictions than using the 400 first SNPs of each matrix. Similarly to the *custom* CNN, this network is another example of the usefulness of convolution layers (also used in SPIDNA and MixAttSPIDNA), and more generally, why tailoring the network to the data characteristics (taking into account the spatial dependency between SNPs) can improve predictions compared to MLPs. The best configuration of the *Flagel* CNN (0.444, index 8) performed similarly to the best SPIDNA version without ABC (0.454, index 11), however *Flagel* CNNs have 8 to 34 times more learnable parameters than SPIDNA, and thus,

are longer to train.

SPIDNA Most of the SPIDNA architectures (all except SPIDNAs with instance normalization and 400 SNPs, 0.641 and 0.599, index 12 and 14) outperformed the ABC rejection (0.454 and 0.469, index 11 and 15) or led to similar errors (0.489, index 13). However, they did not outperform the ABC with correction (0.369, 0.376 and 0.364, indexes 1, 2 and 3). This could be explained by SPIDNA not being able to leverage the local linear relationships in each data point neighborhood found by the ABC, which motivates furthermore the combination of both methods.

Among our permutation-invariant architectures, the best one (SPIDNA using batch normalization, (0.454, index 11 in Table 4.1) had a smaller prediction error than our *custom* CNN (0.487, index 6). However, it is not clear whether this improvement is directly linked to its built-in permutation-invariance property, or to other differences between the two networks.

The best non-adaptive SPIDNA (using 400 SNPs of each matrix) uses batch normalization while the adaptive versions (using all the SNPs of each matrix) use instance normalization, as there is currently no implementation of batch normalization for batches with inputs of mixed sizes (see Section 2.1.4 for more information about why our adaptive networks do not use batch normalization). The Non-adaptive SPIDNA architecture using batch normalization (0.454, index 11) achieved better results than its adaptive counterpart versions that use instance normalization (0.489 and 0.469, indexes 13 and 15). To understand if this loss of performance can be attributed to the addition of the adaptive feature or to the difference of normalization layer, we also trained two versions that have the same configuration as the adaptive ones, but using only 400 SNPs. These two non-adaptive versions with instance normalization have a much higher test error (0.641 and 0.599, indexes 12 and 14), suggesting that our networks using instance normalization tend to underfit, and that the adaptive feature of these networks seems to partially compensate for this effect. Therefore, adaptive architectures could greatly benefit from an optimized implementation of adaptive batch normalization or from an implementation of batches with mixed data sizes.

Controlling the speed to invariance thanks to the parameter α improved the performance of the instance normalization SPIDNA (0.641 versus 0.599, indexes 12 and 14), but less significantly the performance of the instance normalization adaptive SPIDNA (0.489 versus 0.460, indexes 13 and 15). The impact of the parameter α is studied in more details in Section 4.2.1.

SPIDNA combined with ABC We evaluated two types of inputs for ABC: (i) SPIDNA outputs only (inspired by Jiang et al. (2017)) or (ii) both SPIDNA outputs and summary statistics. When using only the predictions of SPIDNA as input to ABC with correction (linear regression, ridge regression or neural network), we improved greatly SPIDNA's performance and obtained errors similar to the ABC based on predefined summary statistics (0.369 compared to 0.364, index 21 and 3). When using both SPIDNA predictions and predefined summary statistics as input to the ABC algorithm, we decreased further the prediction errors (0.347, index 29).

It is not yet entirely clear why this combination of ABC and deep learning (without using summary statistics) decreases the prediction error. Neural networks, such as

SPIDNA, learn a very general mapping of the whole input space to the output demographic parameter space. On the other hand, ABC learns a local relationship, the posterior distribution of the demographic parameters, for each observed example based on its neighborhood in the input space. Combining ABC with SPIDNA thus adds a local inference step to the general mapping learned by SPIDNA, and this might help readjust the predictions locally. This is illustrated in Figure 4.3 where recent population sizes estimated by SPIDNA have a tendency towards the center of the prior, while SPIDNA+ABC corrects it (Figure 4.4). This combination might be modifying the bias/variance trade-off favored by SPIDNA towards higher variance.

MixAttSPIDNA The best results were obtained with the three versions of MixAttSPIDNA (indexes 32, 33 and 34), making them the first purely "end-to-end without ABC step" approaches to outperform ABC on summary statistics. The first one that includes solely an attention mechanism on lines of tensor matrices reaches an error of 0.320 (index 32). Additionally, including an attention mechanism on the predictions of all replicates for each scenario (0.287, indexes 33) greatly improved the prediction error (0.287). During this run, the architecture is first trained to perform predictions replicate wise, then its weights are frozen, while a group of layers allowing for predictions scenario wise is added and trained. Finally, if instead of freezing the weights we kept training them together with the added layers, the predictions improved further (0.251, index 34). In this last configuration, MixAttSPIDNA is first trained to perform replicate-wise predictions, then, instead of being completely frozen, the learned weights are assigned a lower learning rate and are further optimized during the scenario-wise training. These three architectures are the first that do not rely on any summary statistic or ABC step to have significantly outperformed all the other methods presented here.

Simulated HGDP dataset

Now, we investigate the performance of a subset of architectures on our second training set, the HGDP simulated dataset (Section 3.1). Precisely, using this dataset, we train the best SPIDNA architecture (without ABC nor summary statistics, index 11 in Table 4.1), and all MixAttSPIDNA version (indexes 32, 33 and 34 in Table 4.1) to solve the same task (reconstructing population sizes through time). We present the results in Table 4.2.

It is important to note that summary statistics cannot be easily compared directly across scenarios of this dataset. Indeed, each example has a varying number of haplotypes with the purpose of representing real data heterogeneity (where a different population might have a different number of sequenced individuals). This heterogeneity impacts most summary statistics, and one would need to redesign them in order to be insensitive to the sampling size. Previous works have instead decided to simulate datasets with a large number of individuals, recompute summary statistics on subsets corresponding to each smaller sampling size, and use different reference tables for each prediction (Boitard et al., 2016b). Although this solves the issue, it is not the most efficient fashion, particularly for machine learning methods that would need to be retrained on each reference table (i.e., for each sampling size, i.e., almost each population), hence losing the advantage of a single training and fast predictions.

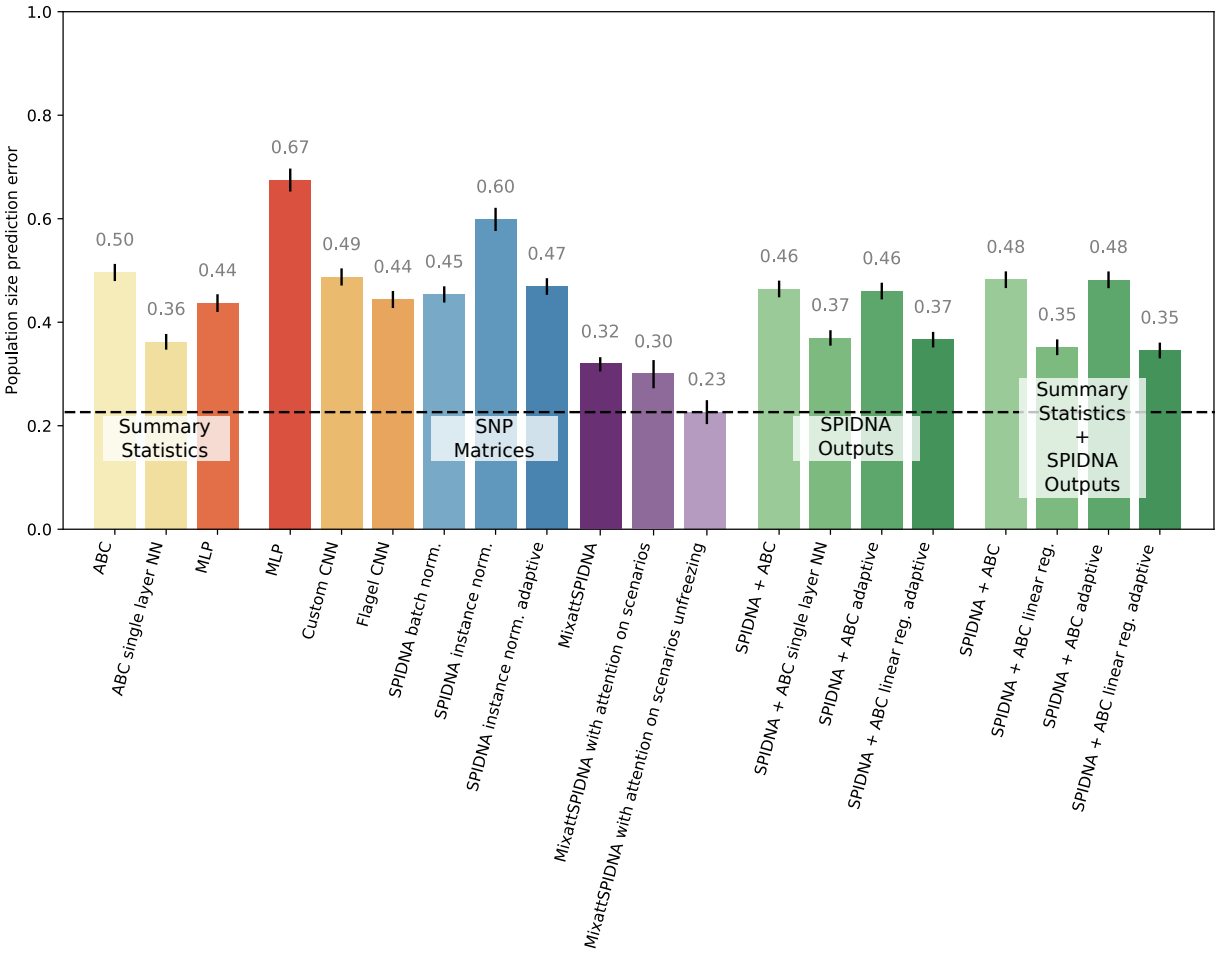

Figure 4.6: **Prediction errors on the test set of the best run of each method after the hyperparameter optimization.** The best configurations of each ANN (MLP, *custom* CNN, SPIDNA and MixAttSPIDNA) have been retrained for 10 epochs. Traditional ABC methods are depicted in yellow, deep MLPs and CNNs in red and orange, SPIDNA ANNs in blue, MixAttSPIDNA ANNs in purple and combinations of ANNs and ABC in green. Methods are grouped into 4 families: "Summary statistics" (processed by ABC or ANN), "SNP matrices" (processed by ANN), "SPIDNA outputs" (processed by ABC, no summary statistic used), "Summary statistics and SPIDNA outputs" (processed by ABC). Vertical black lines on top of each bar represent the 95% confidence interval of prediction errors. Horizontal dashed line indicate the lowest error obtained (adaptive SPIDNA + ABC with local linear regression using summary statistics and SPIDNA outputs).

	Method	Adaptive	Summary statistics	ABC correction	α value	Validation error	Test error
0	ABC	No	Yes	No	No	0.490	0.496
1	ABC	No	Yes	Linear reg.	No	0.357	0.369
2	ABC	No	Yes	Ridge reg.	No	0.363	0.376
3	ABC	No	Yes	Single layer NN	No	0.352	0.364
4	MLP	No	Yes	No	No	0.399	0.437
5	MLP	No	No	No	No	0.690	0.675
6	*Custom* CNN	No	No	No	No	0.485	0.487
7	*Flagel* CNN 0/-1 encoding	No	No	No	No	0.537	0.541
8	*Flagel* CNN 0/-1 encoding	Downsampling	No	No	No	0.437	0.444
9	*Flagel* CNN -1/1 encoding	No	No	No	No	0.610	0.609
10	*Flagel* CNN -1/1 encoding	Downsampling	No	No	No	0.482	0.484
11	SPIDNA	No	No	No	No	0.453	0.454
12	SPIDNA	No	No	No	No	0.637	0.641
13	SPIDNA	Yes	No	No	No	0.487	0.489
14	SPIDNA	No	No	No	0.849	0.592	0.599
15	SPIDNA	Yes	No	No	0.539	0.466	0.469
16	ABC + SPIDNA	No	No	No	No	0.462	0.462
17	ABC + SPIDNA	No	No	Linear reg.	No	0.364	0.377
18	ABC + SPIDNA	No	No	Ridge reg.	No	0.371	0.380
19	ABC + SPIDNA	No	No	Single layer NN	No	0.363	0.372
20	ABC + SPIDNA	Yes	No	No	0.539	0.458	0.460
21	ABC + SPIDNA	Yes	No	Linear reg.	0.539	0.363	0.369
22	ABC + SPIDNA	Yes	No	Ridge reg.	0.539	0.382	0.391
23	ABC + SPIDNA	Yes	No	Single layer NN	0.539	0.374	0.384
24	ABC + SPIDNA	No	Yes	No	No	0.476	0.478
25	ABC + SPIDNA	No	Yes	Linear reg.	No	0.339	0.353
26	ABC + SPIDNA	No	Yes	Ridge reg.	No	0.341	0.357
27	ABC + SPIDNA	No	Yes	Single layer NN	No	0.345	0.361
28	ABC + SPIDNA	Yes	Yes	No	0.539	0.474	0.478
29	ABC + SPIDNA	Yes	Yes	Linear reg.	0.539	0.335	0.347
30	ABC + SPIDNA	Yes	Yes	Ridge reg.	0.539	0.339	0.354
31	ABC + SPIDNA	Yes	Yes	Single layer NN	0.539	0.347	0.365
32	MixAttSPIDNA	No	No	No	No	0.314	0.320
33	MixAttSPIDNA with attention on scenarios	No	No	No	No	0.288	0.287
34	**MixAttSPIDNA with attention on scenarios unfreezing**	**No**	**No**	**No**	**No**	**0.234**	**0.251**

Table 4.1: **Prediction errors of the best configuration of each method on the simulated cattle dataset.** Section 3.3.2 explains the adaptive characteristic of SPIDNA to the number of SNP indicated in column "Adaptive". Section 4.2.1 explains the utility of the parameter α in SPIDNA indicated in column "α value".

For these reasons, we built architectures flexible to the input size and investigated in Section 4.2.4 the best procedure for training them on data heterogeneous in sample size. Thanks to this experiment, we chose to format the batches by subsampling the number of haplotypes to 50 when greater than this value in the unprocessed SNP matrix (they are between 10 and 100 haplotypes for each matrix in the simulated HGDP dataset). On the other hand, matrices with fewer than 50 haplotypes are padded with the value 255 to reach 50 rows.

Overall, the rank between architectures is the same as in Table 4.1. However, the errors are greater on the simulated HGDP dataset than on the previously simulated cattle dataset, probably because the new priors are more complicated and include a varying number of haplotypes in each scenario. This explains why the gap in terms of prediction error between SPIDNA on the simulated cattle dataset (0.454, index 11 in Table 4.1) and HGDP dataset is important (0.560, index 0 in Table 4.2). Similarly, the two versions of MixAttSPIDNA without weight unfreezing mechanism have a greater error on the simulated HGDP dataset (0.484 and 0.392 in Table 4.2, versus 0.320 and 0.287 in Table 4.2). However, this gap of performance is much lower for MixAttSPIDNA with attention mechanism on scenario and weight unfreezing (0.288, index 3 in Table 4.2 versus 0.251, index 34 in Table 4.1). This observation shows that the weight unfreezing mechanism seems to play an important role to overcome the difficulties of the HGDP dataset, which are the wider priors and the variable number of haplotypes. We thus later applied this trained network to the real HGDP dataset in Section 4.3.2.

	Method	Scenario prediction aggregation	Training procedure	Validation error	Test error
0	SPIDNA	Mean	Normal	0.554	0.560
1	MixAttSPIDNA	Mean	Normal	0.480	0.484
2	MixAttSPIDNA	Attention mechanism	Normal	0.407	0.392
3	**MixAttSPIDNA**	**Attention mechanism**	**Unfreezing**	**0.298**	**0.288**

Table 4.2: **Prediction errors on the simulated HGDP dataset.** The SPIDNA version included here is based the same as index 11 in Table 4.1. Section 3.4.3 details the differences between prediction aggregators indicated in the column “scenario prediction aggregation”. Section 3.5.3 explains the differences between the training procedures indicated in the column “Training procedure”.

4.2 Insight into the inner workings and robustness of ANNs

Although ANNs performed very well on the simulated data, several experiments have been conducted to understand how they behave during training, which features they compute from the data and whether they are robust to perturbations of the input data. These experiments are important in the context of simulation-based inference because of the discrepancy between artificial and real data. Indeed, real data are certainly more noisy and often do not respect the evolutionary model assumptions, which may greatly bias the predicted demography. Because real labeled datasets are not available for our task, we are compelled to test for these biases with simulated data. The first experiment

shown aims at understanding the behaviour of the internal variance of data flowing through SPIDNA and the impact of the parameter α added to control this variance. The second experiment evaluates the robustness of ABC and SPIDNA after training to changes in the population genetic simulator, as well as the addition of selection and variations of the number of haplotypes in the SNP matrices. The third experiment is a proof of concept using Canonical correlation analysis (CCA) to find correlations between the features computed by an ANN and summary statistics computed from the dataset. Finally, we compare different ways to format the batch for MixAttSPIDNA and evaluates them on the simulated cattle dataset in order to choose which one is the most robust to variations of the number of haplotypes. The best procedure will be later used for prediction on the real HGDP dataset.

4.2.1 Internal variance of SPIDNA

As explained in Section 3.3.1, each block of SPIDNA combines invariant and equivariant features for the network to be invariant to row permutations in the input matrix (see step I3 in Figure 3.7). In more details, the invariant features are the mean (over the haplotype dimension) of the equivariant ones obtained after the convolution layer. In order to combine both types of features after each block, the invariant ones are copied multiple times (i.e., once for each haplotype) to match the size of the equivariant tensor. Therefore, the network becomes gradually more invariant, rather than just equivariant. Actually, a decrease in the activation variance is expected in most networks, where the complex high dimensional features computed from the inputs are step by step combined into simpler features as data reach the output. However, in SPIDNA case, the duplication of invariant features at each block accentuates this phenomenon.

To control the speed at which the network becomes invariant with depth, i.e., to control the proportion of equivariant and invariant features, we added a parameter α that multiplies equivariant features by α and invariant features by $1 - \alpha$. We tested adding this parameter to two versions of SPIDNA with instance normalization: the first takes as input the first 400 SNPs of each data matrix and the second uses all the SNPs (their number varies between matrices). An α greater than 0.5 gives more weights to equivariant features and, thus, increase the overall variance of the network activities across the haplotypes.

In practice, figures 4.7 at training step 0 (the darkest blue lines) and 4.8 show that the variance with $\alpha > 0.5$ is indeed greater when measured before the convolution layer of each block, since more weight is given to the equivariant (unique) features than to the averaged and copied invariant features. There is a clear trend of variance decreasing more smoothly with larger alphas before convolution and, to a lesser extent, after convolution. However, after training this trend remains only before convolution (as it is expected by construction of the tensor fed to the convolution) and disappears after each convolution (Figure 4.9). But the training curves in Figure 4.10 show similar to worse results for high values of α in the first training steps (90,000 training steps represent less than one epoch). We also tested different values of α on longer runs during the hyperparameters' optimization procedure (Figure 4.1) and there seems to be no effect of α for SPIDNA with a number of SNPs fixed to 400 (dark blue) and debatably a correlation between α and the network loss for SPIDNA with variable number of SNPs

(light blue). Figure 4.7 shows that the variance quickly converges to a stable regime which does not depend on α after few training steps. Figure 4.9 shows similar variance patterns when measured after the convolution layer.

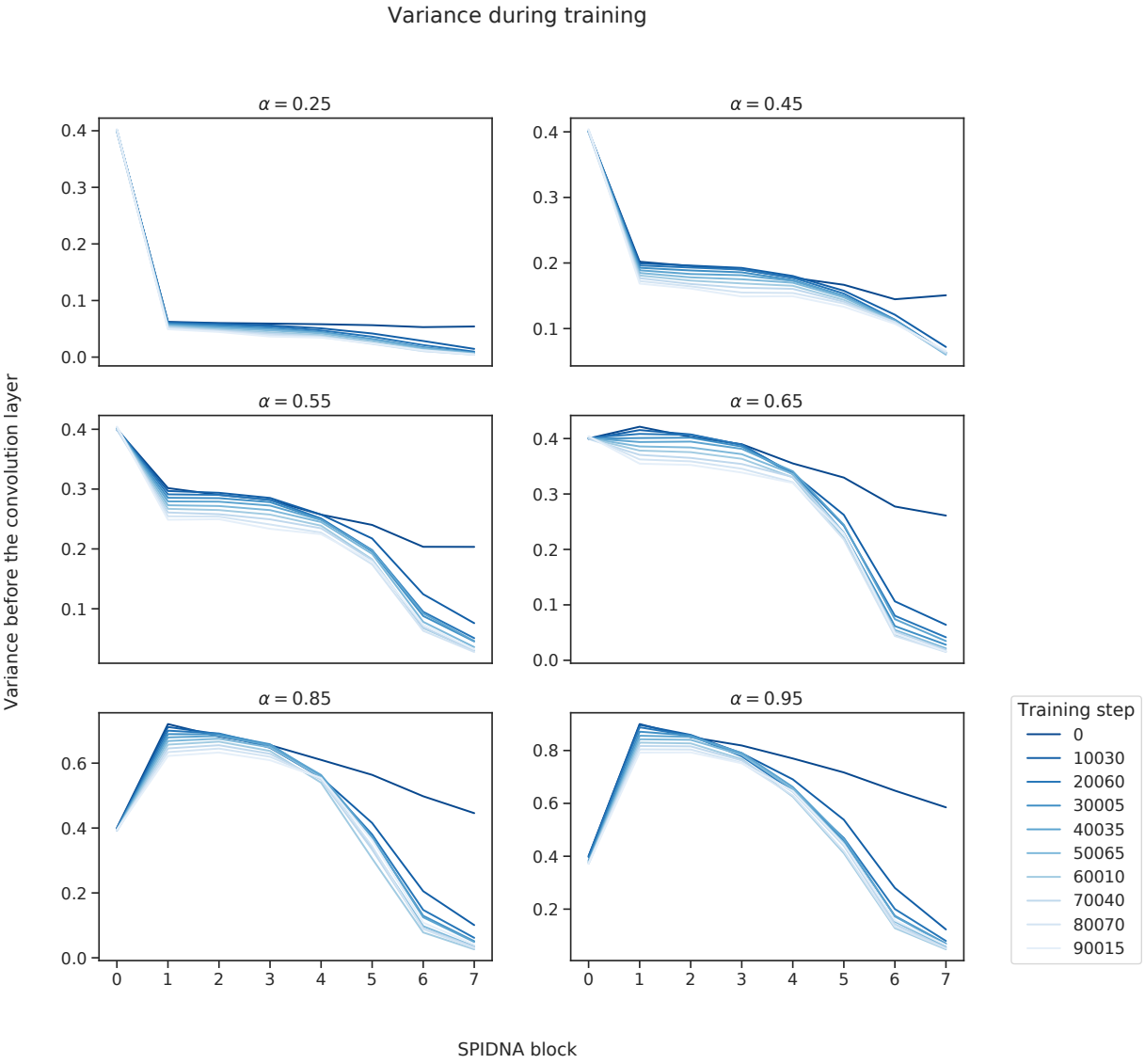

Figure 4.7: **Evolution of the data tensor variance during training at each SPIDNA block before the convolution layer for different values of α.** The x-axis indicates the depth of the convolution layer. The network is composed of seven blocks, with the first convolution layer denoted as block 0 on the x-axis. One training step corresponds to one SNP matrix sampled from the training set.

These experiments show that despite that parameter α has indeed an effect on the data variance, it has little effect on the network performances even with extreme values. One explanation could be that the optimization is able to quickly set up the weights to compensate for any variance profile to the required one for prediction. Although this parameter does not seem to improve the predictions or the optimization, it should be considered for more extreme architectures where the optimization does not converge easily. α could also be improved by replacing it by a function depending on the current

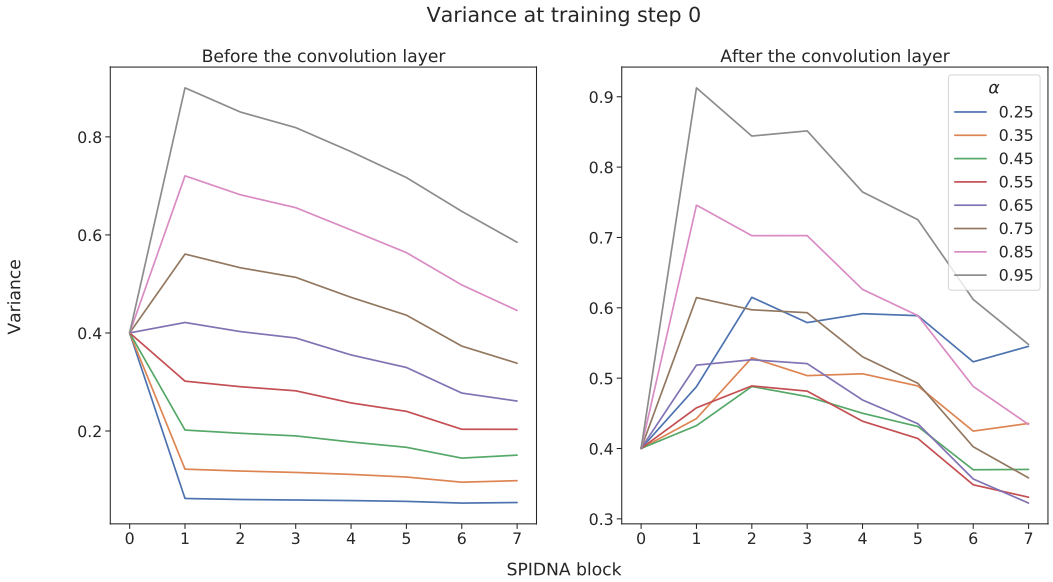

Figure 4.8: **Variance of the data tensor before and after the convolution layer at each SPIDNA block at the beginning of training for different values of α.**

Figure 4.9: **Variance of the data tensor before and after the convolution layer at each SPIDNA block after 90015 training steps for different values of α.** One training step corresponds to one SNP matrix sampled from the training set.

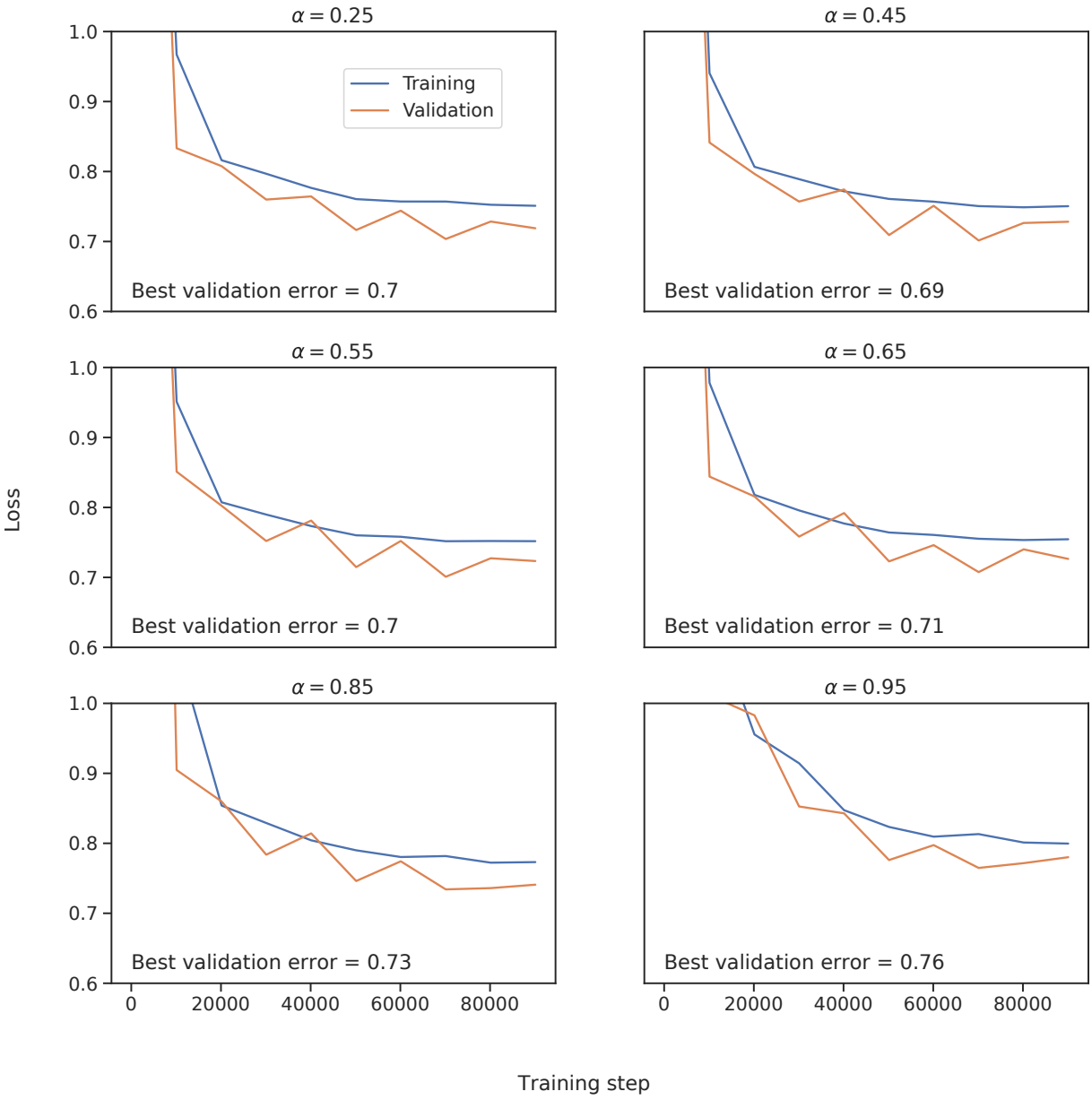

Figure 4.10: **Evolution of the SPIDNA network losses during training for SPIDNA trained from scratch with different values of α.** Validation loss is computed after averaging the predictions of SPIDNA for all replicates of each scenario. Training loss is the average of the MSE of all replicates. One training step corresponds to one SNP matrix sampled from the training set.

depth in the network to more precisely define the variance pattern.

4.2.2 Impact of positive selection on SPIDNA and ABC inference

The impact of positive selection, that is when a novel mutation is beneficial for the individuals carrying it and thus increases in frequency together with its surrounding neutral mutations (physically linked to the beneficial one), on SPIDNA and ABC inference was investigated for three illustrative demographic cases (scenarios "Medium", "Decline" and "Expansion" of Figure 4.2). Because including selection required a change in the genetic simulator (msms instead of msprime), we first ensured that the change of tool to generate the new test dataset had no influence on the prediction accuracy (Figure 4.11). We then simulated 2Mb regions including a central SNP under positive selection, with varying selection strength, starting time and frequency of the beneficial allele at this time (100 regions for each scenario). We chose a conservative approach in which all 100 regions are under selection (worst case scenario). For each scenario, we predicted the population size history using SPIDNA (batch normalization) or ABC (with local linear correction) on summary statistics. Both ABC and SPIDNA predictive errors varied with the selection coefficient (Figure 4.12). On average a moderate selective pressure (100-400) did not decrease the performance (Figure 4.12 top row). ABC inference for declining population datasets was the only one negatively impacted (increased error for s=200 and 400). In fact, in multiple cases, increasing s decreased the prediction error mean. Very strong selection ($s = 800$) on the other hand led to an increased prediction error mean in all cases except for the declining histories inferred by SPIDNA. In addition, the 95% quantile and standard deviations of the prediction errors tend to increase with s (Figure 4.12) indicating that the prediction should be taken more cautiously in the presence of strong positive selection. This variance was systematically smaller for SPIDNA than ABC. In particular, a handful of histories reconstructed with ABC were far off, while SPIDNA prediction errors remained comparatively low for all scenarios (Figure 4.13).

Robustness to the number of individuals

Importantly, SPIDNA adapts to the number of individuals, which is an advantageous property compared to many methods relying on summary statistics. SPIDNA can be trained on data sets having similar or varying sample sizes, and, once trained, it can be directly applied to a dataset of reasonably close sample size, but unobserved during training. We provide an example of robustness in an experiment focusing on a subset of demographic scenarios (Medium, Large constant size, Decline and Expansion) and a wide range of sample sizes (from 10 to 150, Figure 4.14). SPIDNA using batch normalization (trained on exactly 50 haplotypes) did not suffer a strong loss of accuracy when the sample sizes remained in the [45,65] range. Outside of this range, the predictions were inaccurate in two cases: small sample sizes under expanding and constant size scenarios, or large sample sizes under the expansion scenario. This was expected because this specific network was not exposed to diverse sampling sizes during training. Given the observed variations across scenarios and if the sample size is expected to vary substantially from 50, we would advise the user to perform a similar experiment based on her/his targeted sample size and a larger number of scenarios drawn from

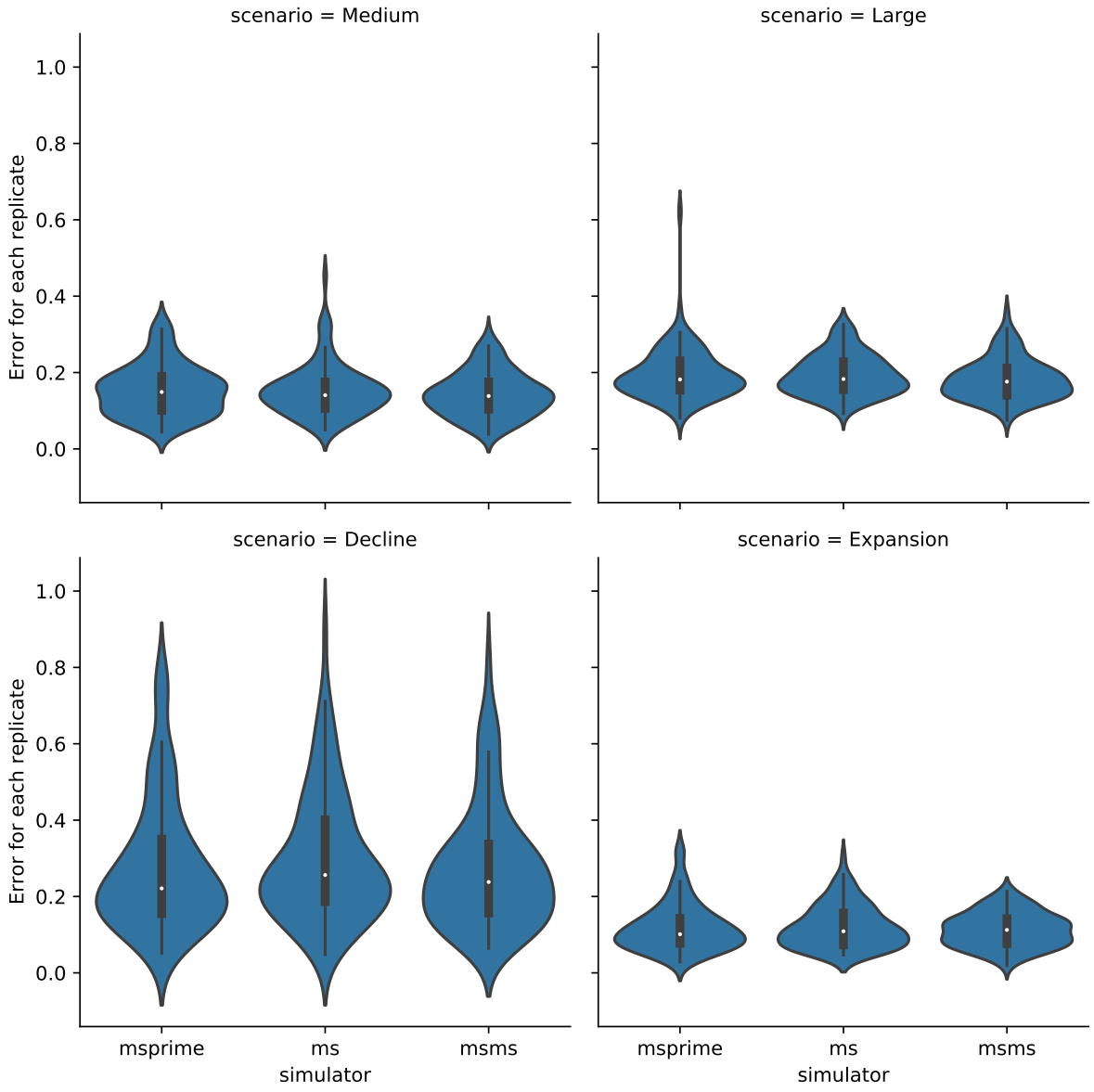

Figure 4.11: **Robustness of SPIDNA to simulator tool.** Distributions of SPIDNA predictive errors per replicate (i.e, per independent genomic region) for four demographic scenarios and three different genetic simulators (msprime, ms, msms). SPIDNA batch norm. network was trained on simulated datasets generated with msprime. The test datasets were generated by different simulators, based on the same demographic parameters and under neutrality. X-axes: simulator for the test set ; Y-axes: predictive error for each region/replicate (i.e., for each matrix of size 50 samples $\times 400$ SNPs) averaged over the 21 time steps. Each violin describes 100 replicates.

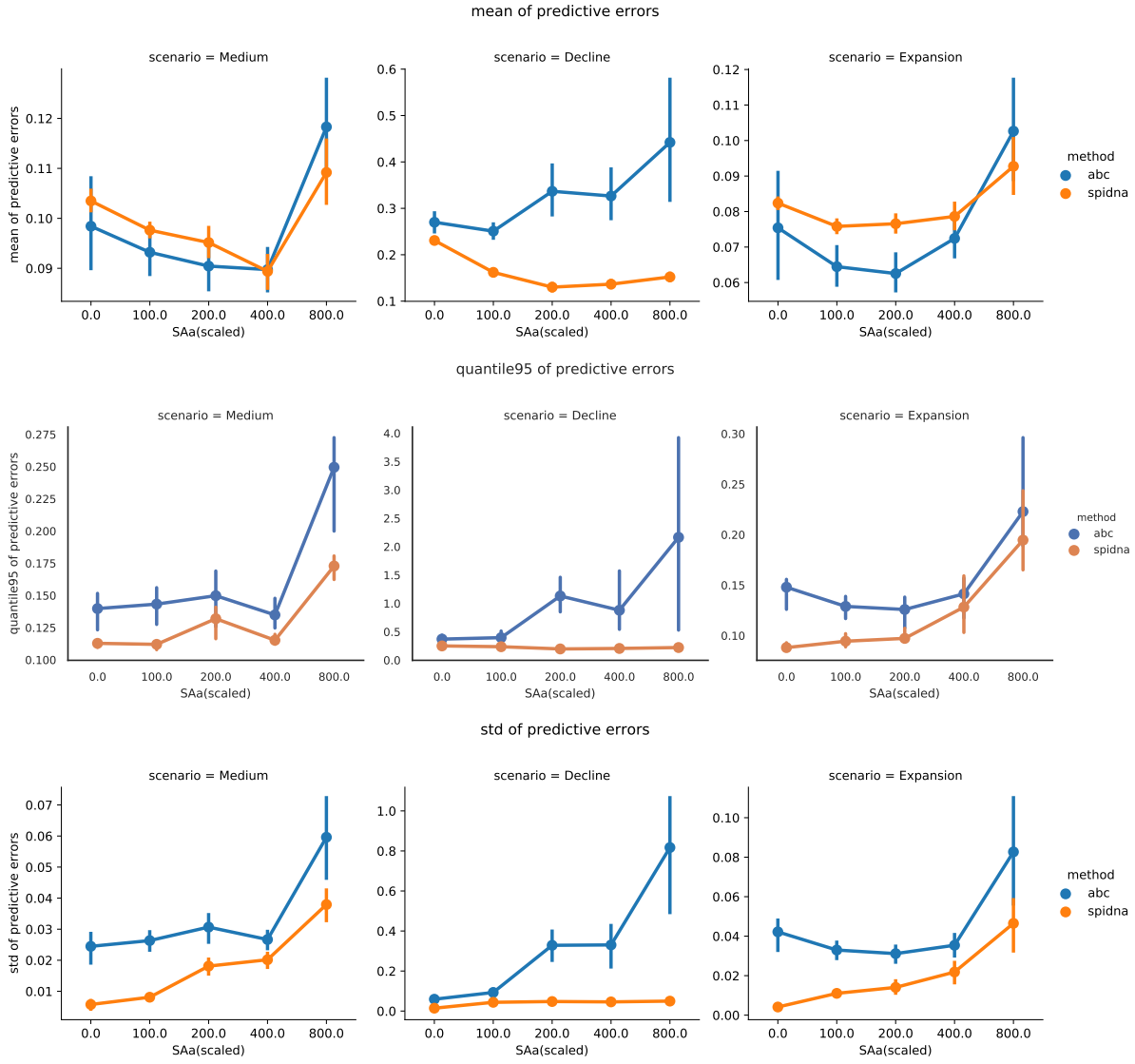

Figure 4.12: **Robustness of SPIDNA to the presence of positive selection.** ABC and SPIDNA (batch norm.) predictive errors computed from 100 2Mb-long regions for three demographic scenarios ("Medium" constant size, "Decline" and "Expansion") under various selective pressures (with additive fitness effect). The reference/training set is the same as the one used throughout the paper (neutral simulations generated with msprime from a prior distribution on recombination rate and population sizes). The test datasets were simulated using msms with multiple values of selection strength, starting time and initial frequency of the beneficial allele. X-axes: Selection coefficient SAa. Y-axes: Mean (top), 95% quantile (middle row) and variance (bottom row) estimators of the predictive error (across 30 test sets for SAa=0 and 144 test sets for any other SAa value). Vertical bars correspond to 95% confidence intervals computed via bootstrap.

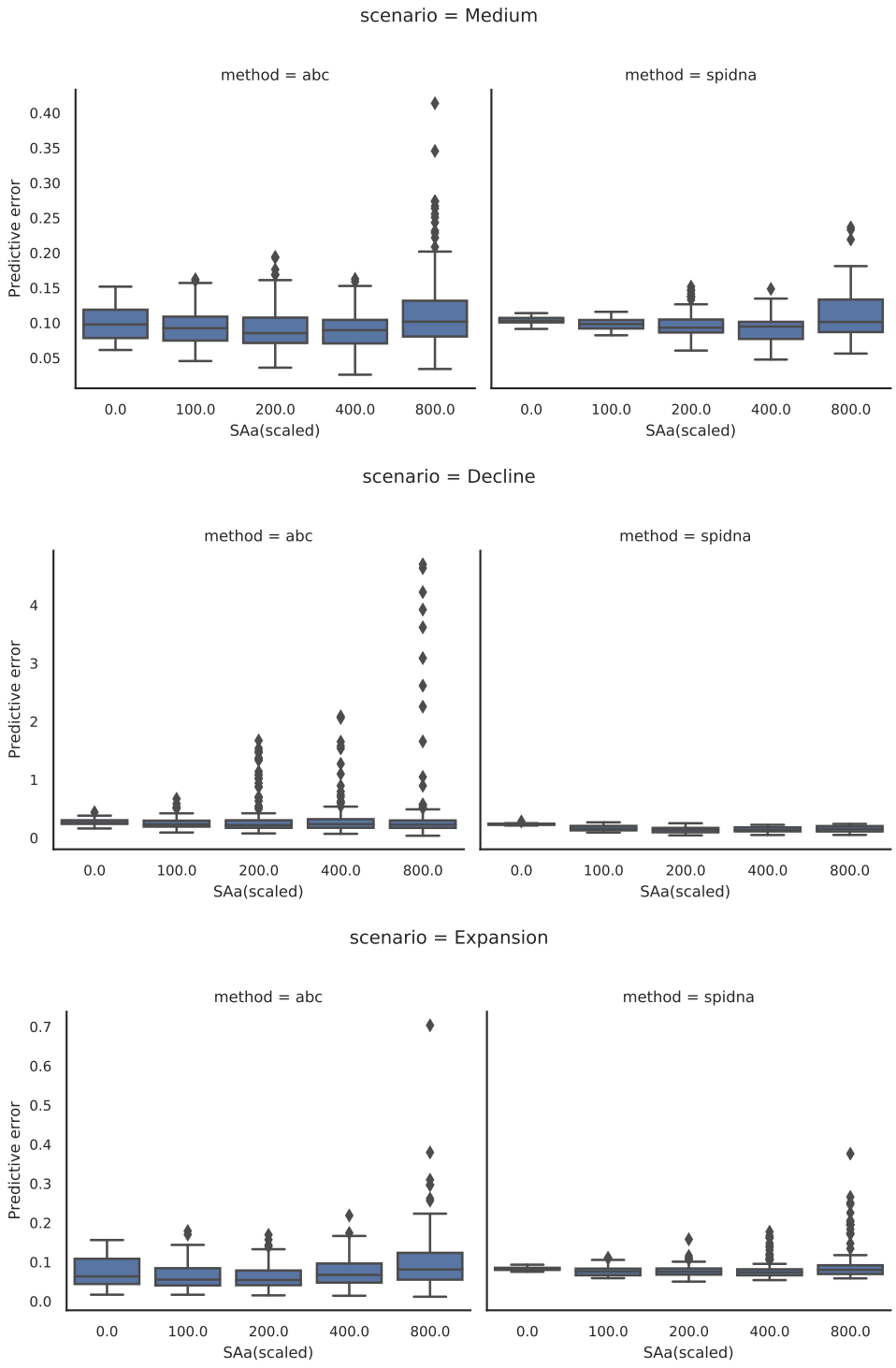

Figure 4.13: **Robustness of SPIDNA to the presence of positive selection.** ABC and SPIDNA (batch norm.) predictive errors computed from 100 2Mb-long regions for three demographic scenarios ("Medium" constant size, "Decline" and "Expansion") under various selective pressures. The reference/training set is the same as the one used throughout the paper (neutral simulations generated with msprime from a prior distribution on recombination rate and population sizes). The test datasets were simulated using msms with multiple values of selection strength, starting time and initial frequency of the beneficial allele. X-axes: Selection coefficient SAa. Y-axes: Distribution of predictive errors (across 30 test sets for SAa $= 0$ and 144 test sets for any other SAa value).

the prior distribution. If needed, the user can then train a new SPIDNA network without any change in its architecture, either on a set containing a wider range of sampling sizes or on a set matching the targeted sample sizes. To fasten the training, this network could be initialized with the weights of the network optimized for the sample size 50, and fine-tuned on the new set.

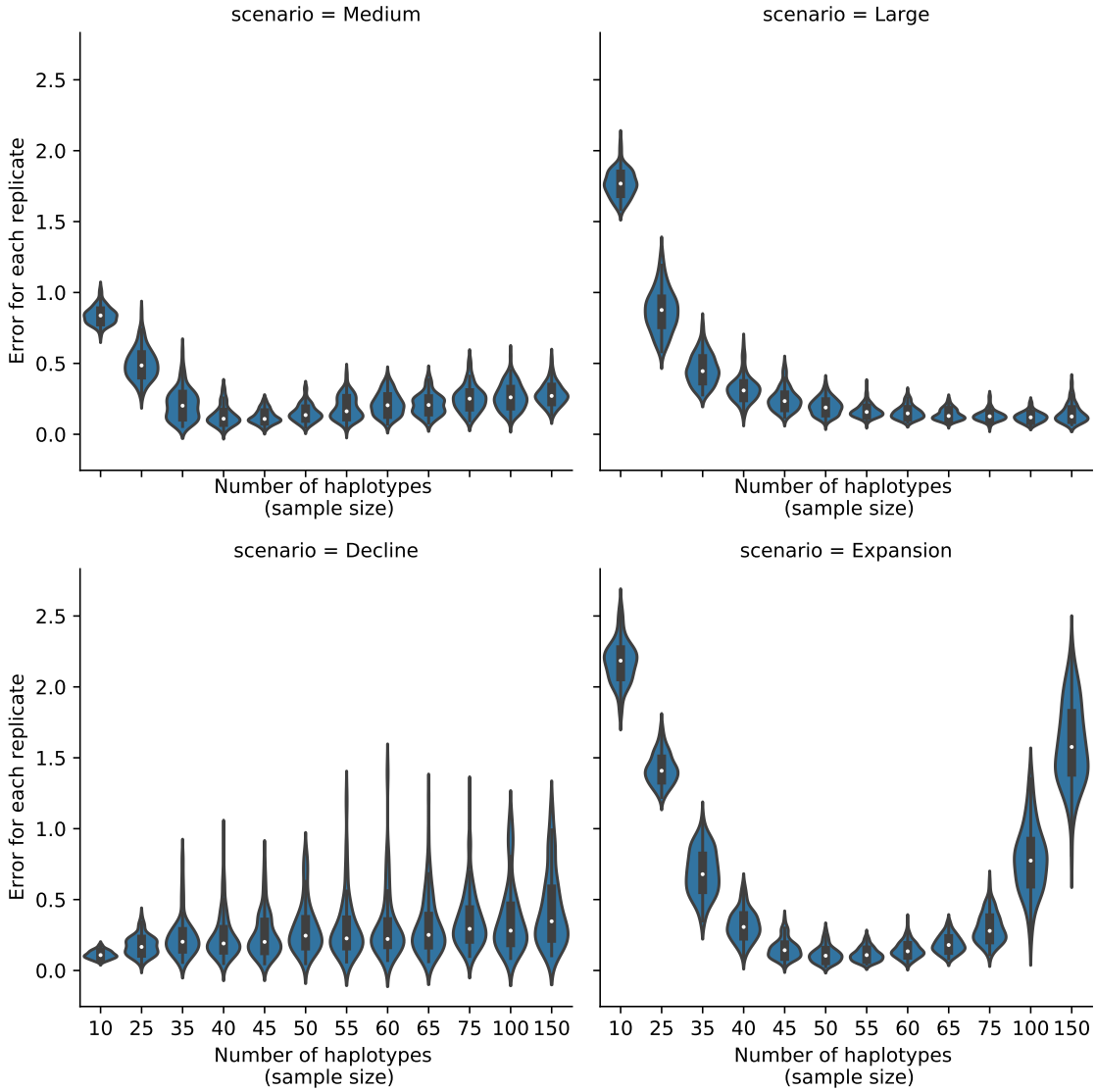

Figure 4.14: **Robustness of SPIDNA to sample size.** Distributions of SPIDNA predictive errors per replicate (i.e, per independent genomic region) for four demographic scenarios and different sampling sizes. SPIDNA (batch norm.) network was trained on simulated datasets containing exactly 50 samples. The test datasets were simulated with msprime based on the same four demographic parameter sets but with different sample sizes (ranging from 10 to 150 haplotypes). X-axes: sample size M of the targeted region ; Y-axes: predictive error for each replicate (i.e., for each matrix of size M samples $\times 400$ SNPs) averaged over the 21 time steps. Each violin describes 100 replicates.

4.2.3 Interpreting the *custom* CNN with canonical correlation analysis (CCA)

We apply the method based on CCA described in Section 3.6 to the *custom* CNN architecture trained on the simulated cattle dataset in order to understand if ANNs compute features similar to the summary statistics that are often used in population genetics. As explained in the previous chapter, this method seeks for correlation between sets of network activations and summary statistics. After training the network, we start by reducing the dimensionality of the activations by applying a singular value decomposition (SVD). Then, CCA seeks for the projection that maximizes the correlation between the two sets. In this section, we analyse the results obtained.

Figure 4.15 shows the distributions of the summary statistic weights in the CCA, pondered by the correlation between the canonical variates associated, for each layer of the network. If this quantity is equal to 1 for a summary statistic, it would mean that it exists one canonical variate explaining all the correlation between the sets of activation and summary statistics, and this variate would only depend on one summary statistic. Figure 4.16 is similar, but distinguish each summary statistic by type. These two figures show an overall decrease in correlation between summary statistics and activations when moving close to the output of the network. This observation could have two meanings, either the correlation decreases because the network computes features that are more and more complex combinations of summary statistics (and therefore less correlated to individual summary statistics), or this is an artefact from the method itself.

Let us remind that an SVD with a variance conserved threshold of 0.99 was applied on all layers studied except for the last that have already only 50 activations. By setting a threshold on the variance conserved, the activations of the first layers are projected in a space with much more than 50 dimensions (dimensions of each layer after SVD: 950, 907, 730, 393, 163 and 50) and therefore, the CCA has more chances to find spurious correlation with the first layers than the last, explaining the decreases in correlation. In Figures 4.17 and 4.18, the SVD has been set up to project the activations of each layer to a 50-dimensional space, which resulted in an inverse dynamic with an increase in correlation in the last layers of the network and particularly with the identity by state summary statistics. This discrepancy between the results obtained with the two different settings of the SVD tends to confirm that this method is not robust enough and could be improved. For instance, the contribution of each activation to the final prediction is not taken into account, and doing otherwise should reduce the risk of interpreting spurious correlations. This amelioration and others will be discussed in more details in the perspectives.

The last layer of the *custom* CNN contains 50 activations and thus did not require to be projected with an SVD. Therefore, the contributions of the activations and summary statistics to the first canonical variates (i.e., those who explain most of the correlation between the two sets) can be directly analysed with the correlation circle in Figure 4.19. First, the vector of the nucleotide diversity π (PI in Figure 4.19) and expected heterozygosity (HET in Figure 4.19) are confounded, which is expected because they are in reality the exact same measure in our data. These two summary statistics mostly contribute to the second canonical component which is also very related to activations 7 and 20,

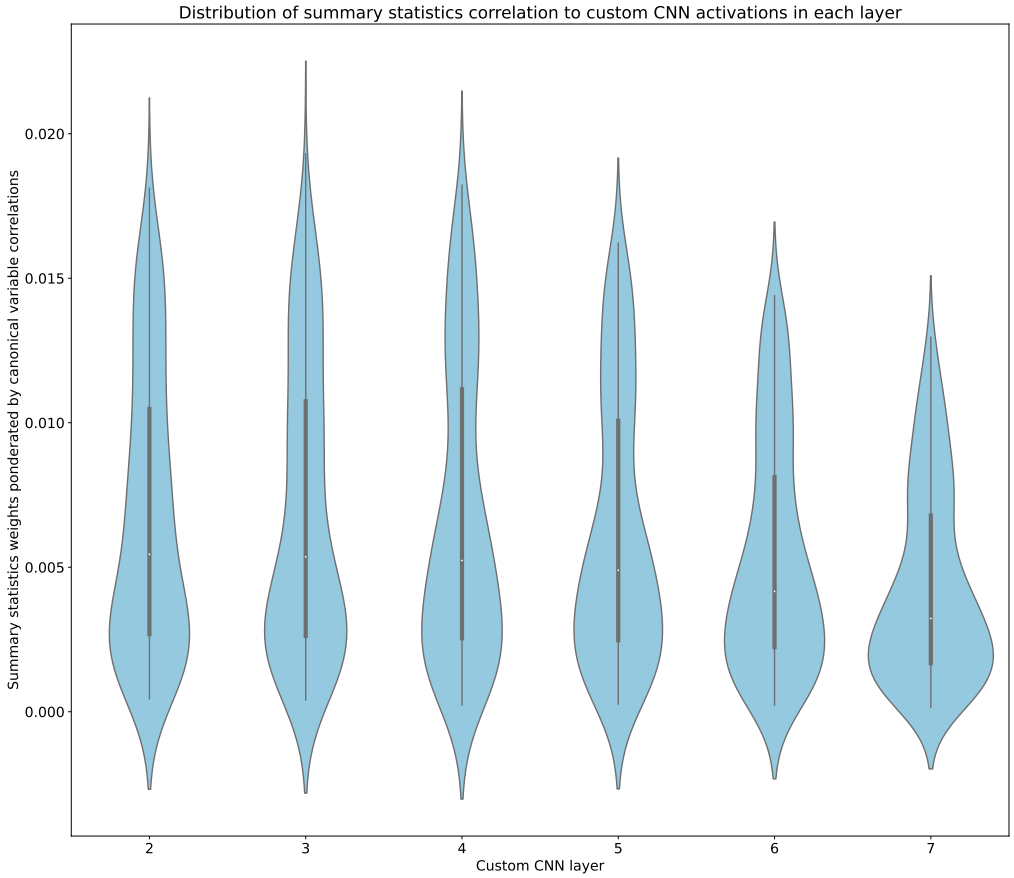

Figure 4.15: **Distribution of summary statistic CCA weights pondered by the canonical variates to *custom* CNN activations in each layer.**

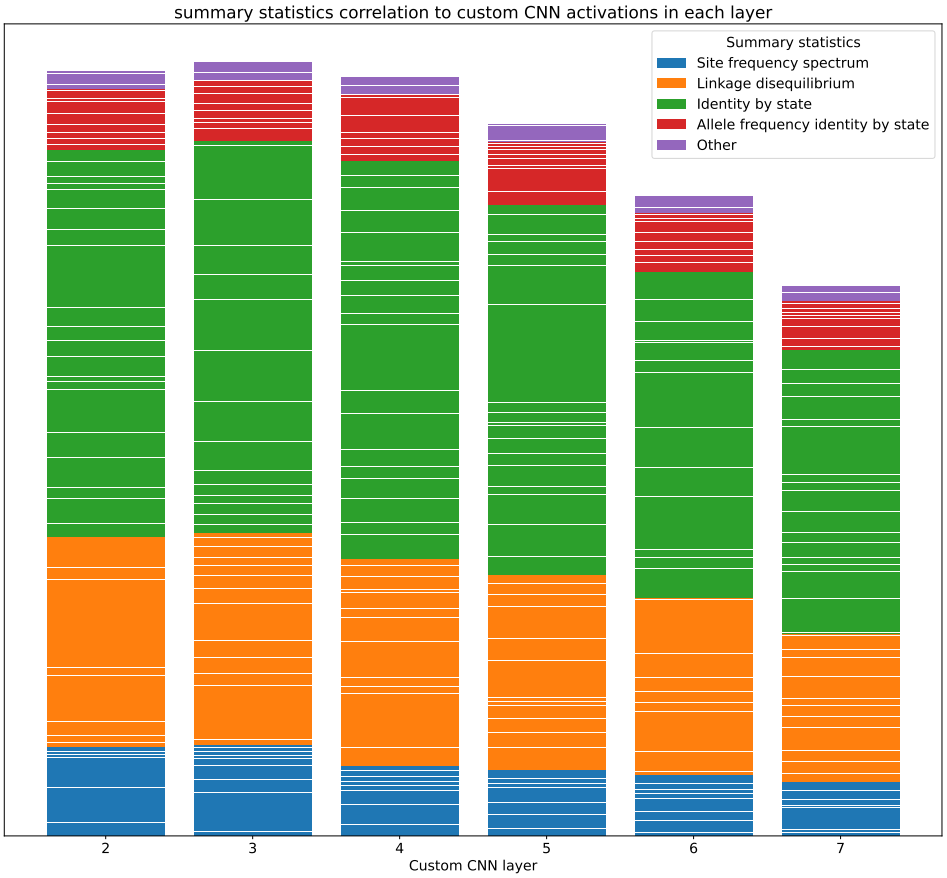

Figure 4.16: **Summary statistic CCA weights pondered by the canonical variates to *custom* CNN activations in each layer.** The height of a bar correspond to the CCA weights pondered by the canonical variate of a summary statistic, and its colour on which family of summary statistics it is part of.

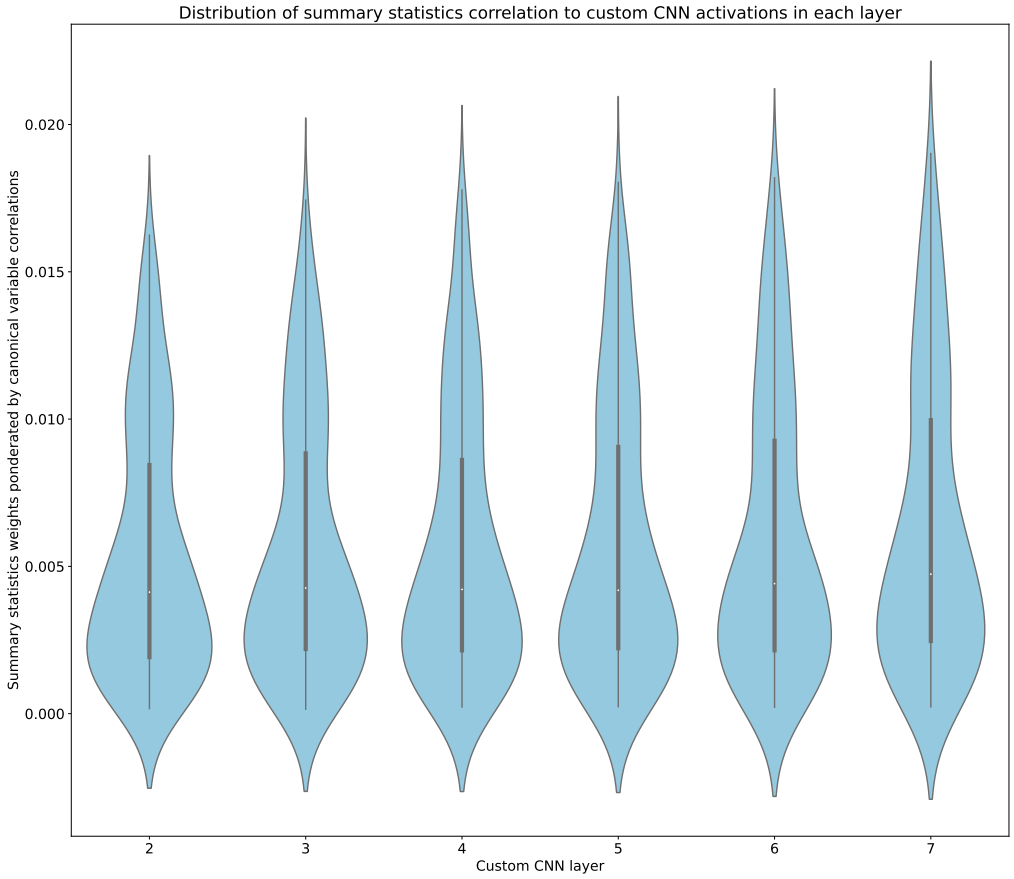

Figure 4.17: **Distribution of summary statistic CCA weights pondered by the canonical variates to *custom* CNN activations, with activations reduced to a 50-dimensional space in each layer.**

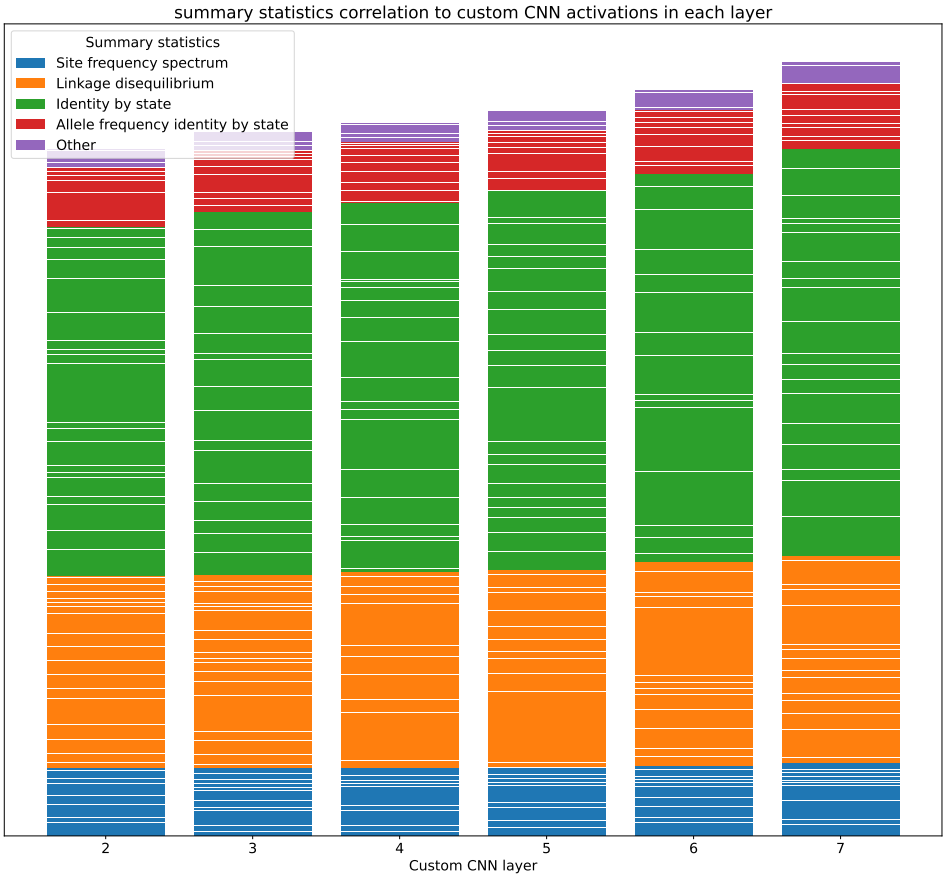

Figure 4.18: **Summary statistic CCA weights pondered by the canonical variates to *custom* CNN activations, with activations reduced to a 50-dimensional space in each layer.** The height of a bar correspond to the CCA weights pondered by the canonical variate of a summary statistic, and its colour on which family of summary statistics it is part of.

therefore these activations seem to capture a data feature that is very close to these two summary statistics. Similarly, the first bin of the site frequency spectrum (SFS_1, the number of singletons in the data), contributes mainly to the first component. Many activations also contribute to this component (mainly activations 4, 17, 21, 31 and 48), showing that these activations compute features that are partially related to the number of singletons. If these activations contribute a lot to the final prediction, this would mean that the network heavily relies on features similar to the expected nucleotide diversity π and number of singletons. This result is interesting because the nucleotide diversity π is strongly linked to the effective size of a population with a constant size and increases with large population. Moreover, singletons are considered particularly useful to distinguish expansions.

4.2.4 Comparison of MixAttSPIDNA batch formats

SPIDNA and MixAttSPIDNA are designed to take a variable number of haplotypes without modification of the SNPs matrices, thanks to the use of layers that easily adapt to variations in the input sizes. More specifically, the layers that we use (the convolutional and hub attention layers) treat each element of the haplotype dimension in the same manner. Therefore, adding an element (a haplotype) to the input only requires repeating the same processing that is already applied to each other element (e.g., the filter convolutions are repeated to match the number of element of the haplotype dimension in the case of convolutional layers). Furthermore, the mean steps of the networks operate the same regardless of the number of haplotype, leading to an output of the same size when the number of haplotype varies.

However, in practice, SNP matrices with different sizes cannot be collated in the same batch (the same tensor). Therefore, training a network on these matrices would require to input them one by one and performing the backpropagation once they each have been processed by the network. This considerably reduces the parallelization of ANNs, and thus, slows down the training of the network. This is why it is preferable to collate SNP matrices in the same batch by either padding or by collating matrices of the same sizes together.

This last experiment compares four different batch formats that can be used to collate SNP matrices. To this end, between 5 and 50 haplotypes are drawn for each replicate of the simulated cattle dataset, and SNP sites that are not polymorphic (variable) in the subset of selected haplotypes are removed. We then trained MixAttSPIDNA by using the four batch formats described in Figure 4.20. After training, we evaluated the four trained architectures on the validation dataset with all matrices subsampled to between 5 and 50 haplotypes.

From Figure 4.20, the best method consists in making batches with SNP matrices that have a different number of haplotypes (heterogeneous batches) by padding every SNP matrices with 255 to match 50 haplotypes (blue line). We choose the value 255 because the SNP in the matrices are encoded as unsigned integer, and it is the most distinguishable value from 0 and 1 for this type of data. With this method, all the SNP matrices reach the dimension 50×400, i.e., 50 “haplotypes” (some will be completely filled with value 255) and 400 SNPs. It shows a very constant error for any number of haplotypes with a slight increase for the smallest number, which is expected due to lack

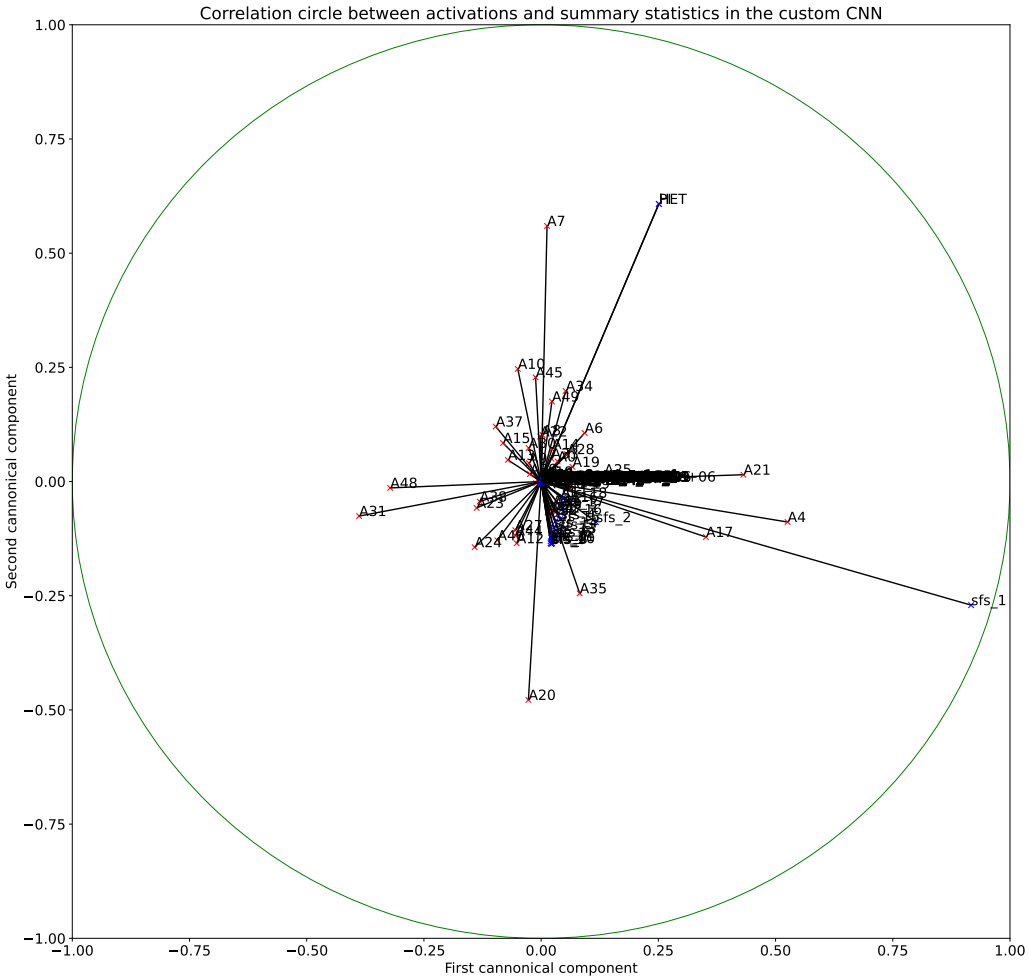

Figure 4.19: **Correlation circle showing the contributions of each summary statistics and *custom* CNN activations to the first two canonical components.** The *custom* CNN have been trained on the simulated cattle dataset and the canonical components have been obtained by performing a CCA between the network activations in its last layer and 279 summary statistics.

of information when the number of haplotypes is very low. Thanks to this experiment conducted on the simulated cattle dataset, we chose the batch format later used for the simulated HGDP dataset, where matrices have a different number of haplotypes to match the different number of individuals in the real HDGP dataset.

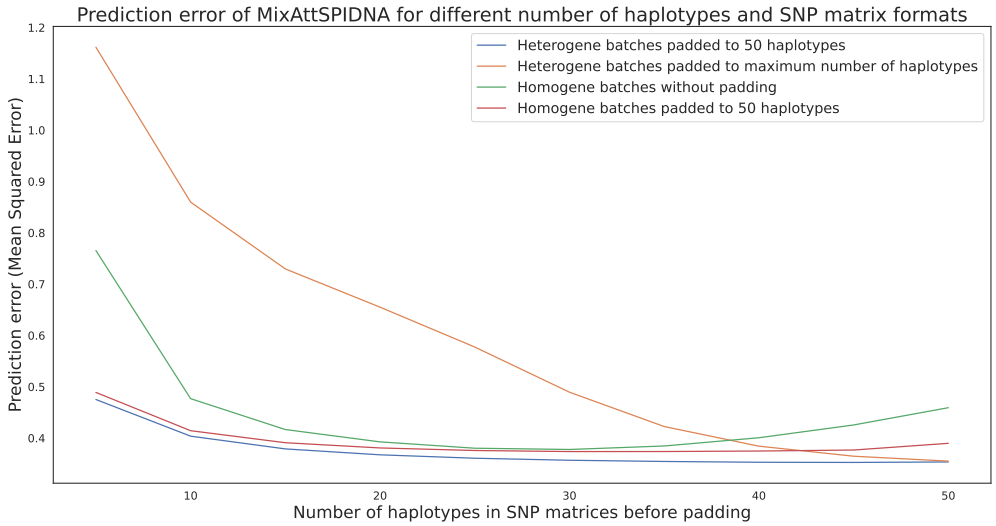

Figure 4.20: **Comparison of MixAttSPIDNA prediction error for different number of haplotypes and SNP matrix format.** Each MixAttSPIDNA architecture is trained on the simulated cattle training dataset, with between 5 and 50 haplotypes randomly subsampled for each SNP matrix. Blue and orange lines represents MixAttSPIDNA trained with batches composed of SNP matrices with different number of haplotypes. SNP matrices in batches are either padded with 255 to reach 50 "haplotypes" (blue line) or padded to fit the SNP matrix with the highest number of haplotypes in the batch (orange line). Green and red lines represents MixAttSPIDNA trained with batches composed of SNP matrices with the same number of haplotypes. SNP matrices in batches are either padded with 255 to reach 50 "haplotypes" (red line) or not padded (green line). Prediction errors are computed over the cattle validation dataset, with all matrices subsampled to the number of haplotypes indicated on the x-axis.

4.3 Population size histories inferred by ANNs on real data

Although it is not possible to have a precise error metric over the real datasets because the true population size histories are not known, the archaeological and historical insights about these populations can hint the accuracy of the predictions. Moreover, performing predictions over real population not only allows to evaluate each inference methods, but also allows having an insight about the rightfulness of the overall model including the data collect and processing, and the simulation model. Indeed, in the case where the inference method perform well on simulated data but not on real data, one could suspect that other parts of the model like for instance the simulator, the priors

are not tailored enough to the real evolutionary process. This chapter will also discuss the results obtained with each method and pinpoints their advantages and features.

4.3.1 Cattle

We inferred the effective population size history of three breeds of cattle (Angus, Fleckvieh and Holstein) based on the same 75 individuals studied by Boitard et al. (2016b) and sampled by the 1,000 Bull Genomes Project (Figure 4.21). The best ABC and SPIDNA configurations both infer a large ancestral effective population size and a decline for the past 70,000 years. However, SPIDNA reports higher recent population sizes (Angus:11,334, Holstein:12,311, Fleckvieh:13,579) than ABC (Angus:344, Holstein:389, Fleckvieh:1,436). Interestingly, SPIDNA infers the same population sizes for all three breeds before 10 thousand years ago. This is in agreement with the estimation of the beginning of the domestication (Zeder, 2008). Posterior point estimates obtained by SPIDNA combined with ABC also indicated a decline after domestication, but with larger population sizes for the last 30,000 years than SPIDNA alone and fairly large credible intervals at recent times (Figure 4.22). Angus had the largest recent population size and Fleckvieh the smallest, in contrary to the two previous methods. Credible intervals of ABC based on SPIDNA outputs overlapped SPIDNA predictions except for the most ancient time window. On the contrary, credible intervals of ABC based on summary statistics overlap SPIDNA predictions except for the most recent time windows (Figure 4.22). Finally, SPIDNA combined with ABC identified an episode of smooth decline and recovery of the population size preceding the domestication (between 400,000 and 30,000 years ago). ABC on summary statistics did not infer this ancient change (this study and Boitard et al. (2016b)), however Boitard et al. (2016a) also estimated that 123,465 years ago the ancestral population size increased from 73,042 to 137,775 using fastsimcoal2 (Excoffier et al., 2013).

4.3.2 HGDP

The experiments conducted in the previous sections of this chapter helped us to select which architecture and configuration should be applied to the HGDP real genomes. The results on the cattle test set and HGDP test set presented in Figure 4.6, Tables 4.1 and 4.2 show that all versions of MixAttSPIDNA have outperformed the other architectures in terms of prediction error. We selected the best architecture configuration and batch format, trained it on simulated data to finally infer the effective population size history of the 54 HGDP populations from the 929 whole genome sequences. The results presented in Figures 4.23 and 4.24 can be interpreted in the light of our knowledge about human population history to assess the accuracy of the values inferred. Figure 4.23 shows that the inferred histories gradually diverge when moving forward to the present time. We expected this behavior as populations share a common ancestry before the “out of Africa” dispersal event.

Overall, most African populations (in blue) have a higher effective size during the last 400,000 years. This is not expected for the most ancient time prior to the “out of Africa” dispersal event, and could be an effect of non-panmictic events that are not modeled, such as structure in the African populations. Starting around 50,000 years ago, African

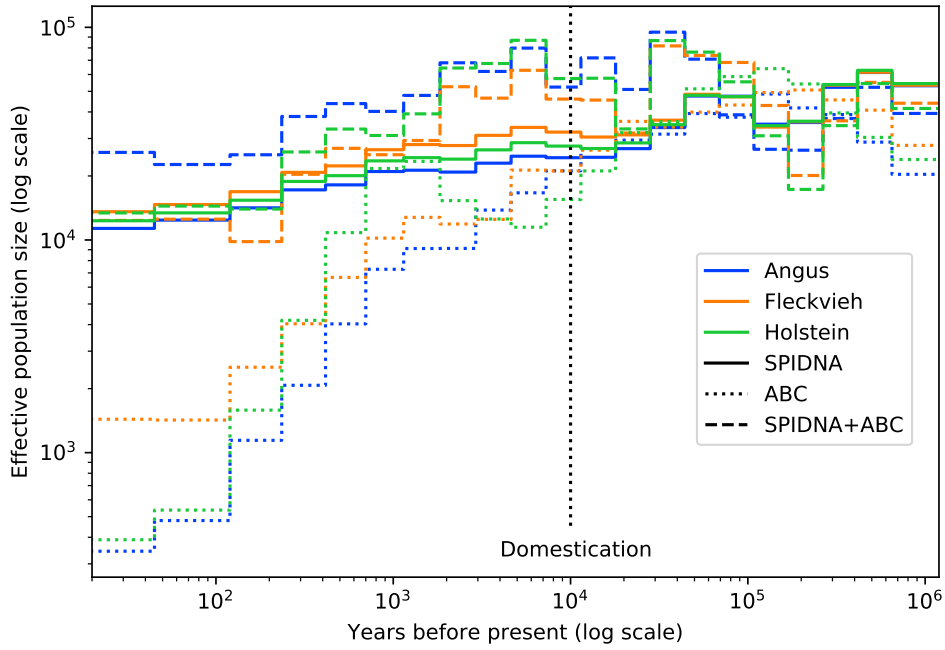

Figure 4.21: **Effective population size of three cattle breeds inferred by ABC (dotted lines), by the best SPIDNA architecture, SPIDNA batch normalization (plain lines), and by ABC based on SPIDNA outputs (dashed lines).** Domestication is estimated to have occurred 10,000 years ago (vertical dotted line).

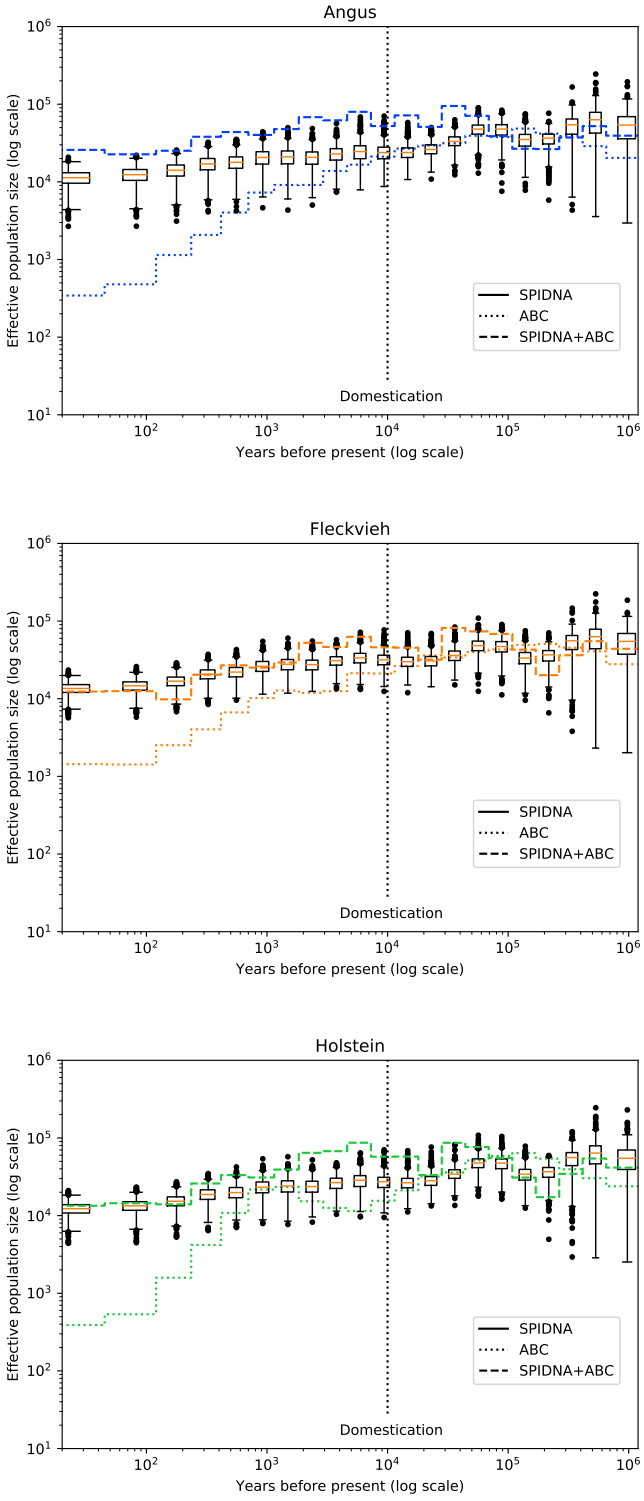

Figure 4.22: **Effective population size of three cattle breeds inferred by the best SPIDNA architecture, SPIDNA batch normalization, by ABC (dotted lines) and by ABC based on SPIDNA outputs (dashed lines).** Boxplots show the dispersion of SPIDNA predictions (over replicates). For each history inferred by ABC and by SPIDNA combined with ABC, we display the posterior median (dotted and dashed lines) and the 95% credible interval. Domestication is estimated to have occurred 10,000 years ago (vertical dotted line).

populations go through an expansion, while most non-African populations remain stable or undergo a bottleneck shortly after. This observation corroborates the "out of Africa" dispersal dated at around 50,000-100,000 years ago (Harris and Nielsen, 2013; Nielsen et al., 2017), where ancestors of non-African populations are expected to have a lower size than the ones of African populations at the time of their migration out of Africa and in the following generations. Despite this, the bottleneck in non-African populations is not systematically as strong as what was inferred in the literature. Ideally, MixAttSPIDNA should have inferred a recent expansion for most populations during the recent time steps, but instead, it often infers an expansion just after 10,000 years (which could correspond to the Neolithic expansion), followed by a decline. Some possible explanations for not observing this would be that the recent human expansion falls outside the priors or that our method has difficulties to reconstruct recent events, similarly to SMC based methods.

From Figure 4.24, we can observe that most populations from the same region show similar trends. For instance, MixAttSPIDNA infers a similar population size histories for Middle Eastern populations, with two bottlenecks, one around 150,000 years ago and another one 10,000 years ago that is followed by an expansion. All European populations follow a similar pattern, except for the Tuscan population, that does not have the signal of the second bottleneck and expansion. Following a different history, American populations go under a decline after a short expansion around 40,000 years ago. Unlike the other populations from the same region, the Maya population has a second expansion phase starting 4,000 years ago, which corroborates the strong expansion found by Bergström et al. (2020).

Although we identified some well known trends such as the "out of Africa" dispersal and common origins of human population in our results, further investigations are required to explain the variations between regions and understand why some populations display a different pattern from others within the same region. It would be particularly interesting to find if these observations overlap with cultural, archaeological and historical evidences, for example the presence or absence of expansions could be linked to population lifestyles, as previous papers have detected differences in the demographic histories of nomadic and sedentary populations (Aimé et al., 2013). Furthermore, it seems to be no correlation between the inferred pattern and the number of individuals sampled in each population, however, other sources of bias could be investigated, such as the effective coverage (the sequencing aimed for a $35\times$ coverage but after sequencing, each sample can have a slightly different coverage) or non-random sampling of individuals that could misrepresent the real population.

4.4 Chapter conclusion

This chapter presented the results obtained for our different architectures, as well as a robustness study and experiences designed to better understand the inner workings of the ANNs developed here. During the first part of this thesis, we developed the SPIDNA architecture that was able to match the results obtained by ABC, an already well established method for demographic inference, without relying on summary statistics. Then we further improved the error by using SPIDNA's predictions as input for an ABC. The

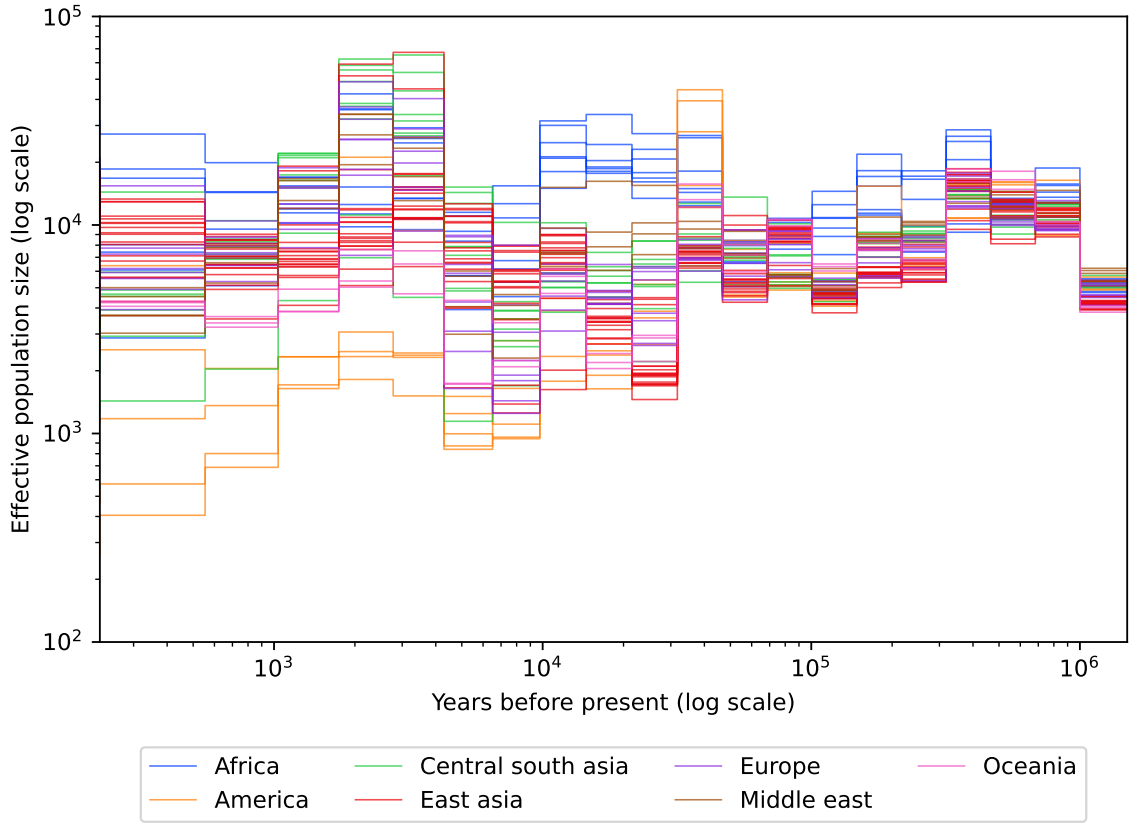

Figure 4.23: **Effective population sizes inferred by MixAttSPIDNA for the human populations from the HGDP dataset.** Effective sizes are inferred by the MixAttSPIDNA version with the lowest prediction error on the simulated dataset, i.e., with attention mechanism on scenario, batches with padding and the weight unfreezing mechanism.

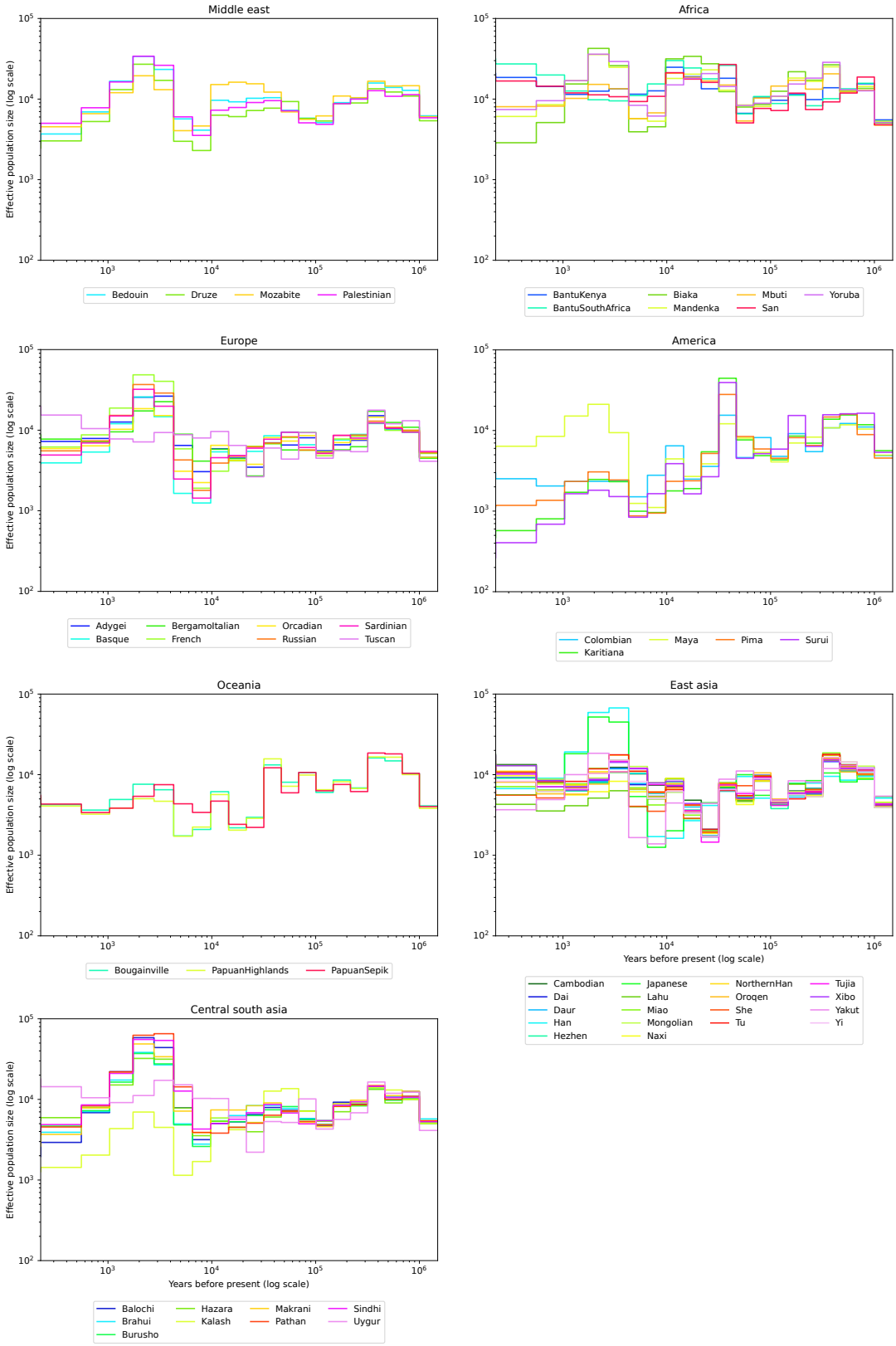

Figure 4.24: **Effective population sizes inferred by MixAttSPIDNA for the human populations from the HGDP dataset separated by region.** Effective sizes are inferred by the MixAttSPIDNA version with the lowest prediction error on the simulated dataset, i.e., with attention mechanism on scenario, batches with padding and the weight unfreezing mechanism.

best error, not including the use of MixAttSPIDNA, was obtained by adding summary statistics alongside the predictions of SPIDNA before an ABC step. When applied to the real cattle data, our methods inferred a decline after domestication, which was expected knowing cattle history. However, the decline observed is less strong than what we obtained with ABC, and it is still unclear what are the most likely scenarios (strong or soft decline) for these cattle breeds. SPIDNA showed to be robust to the introduction of positive selection, differences of simulator tools and variations of the number of haplotypes in the matrices on predefined scenarios. These experiments demonstrated the robustness of SPIDNA to small perturbations for which it has not been trained for, and seems to indicate that it could better handle the reality gap between simulated and real data at least for these specific cases. By comparing the different versions of SPIDNA, we were able to understand the usefulness of the mechanisms introduced. The permutation invariant design of this architecture showed better results compared to the *custom* CNN and MLP despite having a similar number of learnable parameters. We think that this difference in performance comes from the fact that the *custom* CNN and MLP are less adapted to the data characteristics, which shows the soundness of developing architectures tailored to the data. The SPIDNA versions that are able to adapt to any number of SNP in the matrices did not outperform the versions relying on the first 400 SNPs of each matrix. This loss of accuracy seems to be attributed to the change of normalization layer from a batch normalization to an instance normalization, rather than to the introduction of all SNP from each data. Moreover, the parameter α designed to control the contribution of invariant and equivariant features in the network does not have a significant importance for the prediction error because the network automatically converges to a variance profile after few training iterations.

We tested the CCA based method described in the previous chapter on the *custom* CNN as a proof of concept to compare summary statistics to network activations. Although this method is promising, it still needs some refinements to handle the high number of activations and to take into account their importance for the final inferred values. Indeed, the results were not robust to changes in the SVD step used to reduce the activation space. However, we still found some strong correlations between activations in the last layer (where the activation space is already low dimensional and does not require reduction) and summary statistics such as the first bin of the SFS (number of singletons) and the nucleotide diversity π.

During the second part of this thesis, we developed another deep learning architecture based on SPIDNA. Called MixAttSPIDNA, this architecture is intended to predict the population size history of human populations from the HGDP dataset. In order to improve the predictive power of this architecture, we added a more general invariant function in the network than the mean used by SPIDNA. We also introduced an attention mechanism to combine the inferred values over multiple replicates in a manner that assigns automatically a different weight to each replicate. Moreover, we performed an experiment aimed at finding the best way to collate SNP matrices with different number of haplotypes in the same batch tensor. Alongside the weight unfreezing mechanism used to train this version of MixAttSPIDNA, it achieved the best results on both cattle and HGDP datasets without using summary statistics nor ABC. Finally, the results on the real populations of the HGDP dataset showed that MixAttSPIDNA was able to recover some well studied parts of the human history, as the recent common origin of

populations, their divergence through time and the first out of Africa dispersal event. A more extensive study of these results would help to understand if the precise trends displayed by each population have a sense from a cultural, archaeological and historical viewpoint.

Chapter 5
Conclusion

Contents

Demographic inference is a challenging task in population genetics. The inference methods developed in this field are the product of decades of research in sequencing technology, computation technology, statistical analysis and evolution theory, and rely on a wide variety of inference frameworks. As seen in the first chapter, most of them are either tied to an evolutionary model or to the computation of summary statistics that may lose information from the original data. However, the many successes of deep learning show that it is possible to build powerful inference methods without integrating much domain-based knowledge. Therefore, we developed a deep learning framework with the idealized goal to provide a trained network that would infer demography for any species.

When we started the works presented in this thesis, few publications were included deep artificial neural networks to tackle problems in population genetics, and only the work from Sheehan and Song (2016) aimed explicitly toward the inference of population demography. This pioneering work introduced a multilayer perceptron using summary statistics to infer the presence of selection and a demographic model with three piece-wise constant effective population sizes. Inspired by this, we have driven our research towards a more challenging problem by increasing the number of demographic parameters (effective population sizes) to 21 and by bypassing summary statistics. Sequencing data is still preprocessed (by us or others) for the classical alignment, discarding untrusted reads, genotype calling, phasing, filtering out low-quality regions, etc.; however, once those pre-steps are done, relevant information should be preserved by the data encoding (SNP and position matrices) and the automatic construction of features. Moreover, we wanted to develop a flexible approach that requires few expert knowledge in population genetics (e.g., summary statistics or inference method based directly on evolutionary models), as such method would be easier to adapt to changes in the evolutionary or the demographic models, and could potentially be transferred to challenging questions in population genetics concerning processes leaving complex

patterns in the data, that can not be fully addressed with the summary statistic approach. Moreover, following this logic could also extend the application field of this approach to problems based on genomic data that are outside the scope of population genetics.

We first tested the MLP that, despite having many learnable parameters and complete access to the genomic data, did not yield good prediction error compared to methods relying on summary statistics. At that time, we observed that the success of deep learning in most fields can be explained partially by the fact that researchers have developed architectures tailored to the data characteristics when the variables are not independent. For instance, one major contribution to the field of image processing was the introduction of convolution layers that take into account the spatial dependence between the pixels of an image. Similarly, recurrent neural networks and the long short-term memory architectures are an essential milestone for the field of natural language processing, and more generally, for processing time series because of their abilities to handle sequences of infinite length. Following the same path, we identified the characteristics of SNP data that a network could handle by design and picked from the deep learning literature or designed ourselves mechanisms taking into account these characteristics.

We developed a first convolutional neural network (*custom* CNN) to take into account the spatial dependencies between SNPs. We later introduced the SPIDNA architecture that also uses convolution filters, but remove any steps that could not cope with the variations of the number of haplotypes and SNPs in the data. Moreover, we designed SPIDNA to be invariant to permutations of haplotypes, a second propriety of our data, by combining invariant and equivariant operations. This architecture gave predictions competitive with the best baseline, ABC with summary statistics, but only after adding an ABC step and using only 400 SNPs. From these results, we concluded that taking into account the permutation invariance propriety of the data helps the prediction. However, although taking into account the varying number of SNPs is a promising lead, our architecture still needs some refinement before effectively surpassing their counterpart trained on a fixed number of SNPs. Our last architecture, MixAttSPIDNA, was designed with the intention to further improve the predictions by removing two limitations of SPIDNA, thanks to a new type of permutation invariant attention mechanism, that we called hub attention. We added this mechanism alongside the original invariant layer of SPIDNA with a simple mean operation in order to give the network more freedom in the invariant computations that could be learned during training. We also used this mechanism to replace the mean operation performed by SPIDNA over the predictions made for each replicate (i.e., 2 Mbp-long regions) of a scenario. This provided an alternative (and more accurate) solution for combining information coming from multiple regions of large genomes. Both of these improvements led to a substantial reduction of the prediction error, beating all the other approaches studied, and in a statistical framework that only relies on deep learning.

Although the design of new deep learning architectures for genomic data represents the main contribution of this thesis, we also explored other aspects of the inference framework. Firstly, we used our implementation experience to develop a package aimed at facilitating the development and usage of deep learning in the population genetic community. Secondly, we proposed a method based on canonical correlation

analysis, which still requires further development but could help understand if neural networks compute features similar to summary statistics. Finally, we trained all architectures with the final goal of inferring the population size history of populations from two real datasets of *Bos taurus* (cattle) and *Homo sapiens*. This required careful design of the priors of our simulated datasets to minimize the reality gap.

5.1 Research perspectives

In this section, we will discuss some leads to improve the methods presented in this thesis, along new methods that have a great potential for demographic inference. These perspectives could further improve the demographic parameter values inferred, but also offer better interpretability and better handling of the data features.

5.1.1 Improving deep learning architectures

Deep learning profits from an important community outside of population genetics, constantly offers new tools that could benefit demographic inference. Therefore, they are numerous ways to build upon our architectures by integrating new mechanisms developed by this community.

We identified that the higher prediction error of SPIDNA adaptive to a variable number of SNPs, compared to the non-adaptive one, is probably a consequence of replacing the batch normalization layers with instance normalization layers. We used a different kind of normalization layer because data could not be collated in the same batch after the dimension sizes no longer match. As this is only a technical limitation and batch normalization could be in theory for heterogeneous batches, we think that this version of SPIDNA could greatly benefit from an implementation of the batch normalization adapted to this type of batch.

We saw during this thesis that the development of new architectures is a tedious task that relies primarily on trial and error, as the architecture often needs to be evaluated each time a new mechanism is introduced. It also requires a good understanding of the data properties and domain of application. Although we used an hyperoptimization procedure, this approach cannot be used as a replacement for the numerous design choices we made during the development of our ANNs because it was not enough computationally efficient. Consequently, a new field called automated deep learning has recently emerged with the goal of automating the search for new architectures. We think that using such workflow could significantly improve the optimization of architecture's hyperparameters that for now relies mainly on human interactions.

Although increasing the number of features computed by SPIDNA did not yield to substantial improvement of the prediction error, very large networks such as GPT-3 (Brown et al., 2020) have shown unprecedented results. Moreover, after training, these networks can be transferred to smaller ones by a process called knowledge distillation, removing the burden of requiring numerous GPUs and of storing a large network for the users during inference. Therefore, it would be interesting to increase the expressive capability of our networks by increasing drastically its number of weights and layers.

However, SPIDNA and MixAttSPIDNA architectures are already fairly large and are distributed on three GPUs for mini-batches of size around one hundred. Thus, increasing their number of weights would require a lot of engineering work to parallelize it over many more GPUs, by not only splitting the mini-batches over GPUs, but also the architecture itself.

5.1.2 Solving inverse problem

Demography inference falls in the category of inverse problems because it seeks to find the causal explanations (a demographic scenario and an evolutionary model) of the observed genomic data. In this thesis, we used the evolutionary model as a simulator to generate labeled genomic data under specific demographic scenarios and ANNs are then trained to reverse the process by mapping genomic data to demographic scenarios. However, other methods that are invertible by design have been recently developed in order to inverse a simulator by mimicking it. This section will describe a promising type of architecture, called invertible network, that has not yet been applied to demographic inference.

The intuition of invertible networks in the context of inverse problems is to train the network to map the hidden variable x to the observed data y (Ardizzone et al., 2018), where x would be the demographic parameter values and y the genomic data or some summary statistics. Once the forward mapping from x to y is learned by using data generated by an evolutionary simulator, the inverse mapping from y to x is obtained for free because of the invertible property of the network. Combined to normalizing flows (Tabak and Vanden-Eijnden, 2010), a generative model capable of learning non-linear transformation between a complex data distribution and a simple prior distribution, this process allows approximating the posterior $p(x \mid y)$ instead of having point estimations. However, the network must have a tractable Jacobian and triangular in order to be invertible and being capable of mapping one distribution to another. To this extent, the NICE architecture (Dinh et al., 2014) uses coupling layers that have been later complexified in the Real NVP architecture (Dinh et al., 2016). These layers split the input data into two; one part of the input is treated by an arbitrarily complex function that can be any type of neural network. Then the result is combined with the other part of the input data with an easily invertible function, which is simply a sum in the original NICE architecture (Dinh et al., 2014). Multiple coupling layers are stacked to compose the overall architecture. In the context of inference, an additional latent output variable z is added alongside y to counteract the information loss in the forward process, as explained in Ardizzone et al. (2018). Nonetheless, in its current configuration, this type of network require the inputs and outputs to have the same dimension and would need further development to map genomic data to demographic scenarios. This approach is still very promising because the network is directly maximizing the likelihood and learns the real posterior distribution, contrarily to variational autoencoders (VAEs) and generative adversarial networks (GANs).

5.1.3 Evolutionary and demographic models

The predictions made by our deep learning framework not only depend on the network architecture, but also on the training data generated.

Unlike methods that derive a likelihood from an evolutionary model, our deep learning approach uses it as a simulator. Hence, it could be easily swapped for another one that has fewer assumptions about the real evolutionary process, such as a non Wright-Fisher model, with the advantages presented in Section 1.4. A different evolutionary model could also introduce nuisance parameters such as variations of the mutation rate, recombination rate, and the presence of natural selection in order to simulate more realistic data. Furthermore, the simulator could also simulate data at different generations to leverage the increasing availability of ancient DNA sequences. Note that such improvements would not translate into better prediction errors on simulated datasets (except maybe when incorporating ancient DNA data), but could greatly improve the differences between the effective sizes inferred and the real census sizes by minimizing the reality gap between simulations and the real sequences.

Although our demographic model is fairly complex in terms of the number of parameters to infer, it does not include other demographic factors other than population sizes. Therefore, another main source of improvement would be to have parameters describing population structure, admixture, and migration events.

To conclude, the work conducted during this thesis opens the path for a new inference approach in population genetics that offers great perspectives of improvement. We hope that this work will help to find new discoveries regarding the past of populations and inspire the community to develop new deep learning architectures.

Chapter A

Appendix

A.1 Computational resources

Simulations have been performed on the genotoul bioinformatics platform with the following hardware:

- 68 nodes with 2 E5-2670 v2 Intel CPUs (2.50GHz, 20 threads) and 256GB of RAM
- 48 nodes with 2 E5-2683 v4 Intel CPUs (2.10GHz, 32 threads) and 256GB of RAM.

All summary statistics, trainings and predictions were computed on the TAU's Titanic platform with the following hardware:

- 5 nodes with 4 GTX 1080 (12GB of VRAM) GPUs, 2 E5-2650 v4 Intel CPUs (2.20GHz, 24 threads) and 252GB of RAM
- 7 nodes with 4 RTX 2080 (12GB of VRAM) GPUs, 2 Silver 4108 Intel CPUs (1.80GHz, 18 threads) and 252GB of RAM
- 1 node with 4 Tesla P100 (16GB of VRAM) GPUs, 2 E5-2690 v4 Intel CPUs (2.60GHz, 28 threads) and 252GB of RAM
- 1 node with 2 RTX 2080 (8GB of VRAM) GPUs, 2 E5-2650 v4 Intel CPUs (2.20GHz, 24 threads) and 252GB of RAM

Both platforms use Slurm as job scheduling system. Batch sizes and deep learning architectures were all designed to fit on less than 12GB of VRAM during training. To train non-adaptive architectures, batches were split between 3 GPUs with at least 12GB of VRAM. Adaptive architectures were trained on one GPU as batch data of varying sizes could not be concatenated in the same tensor. The training of SPIDNA took at most 1h42 per epoch for non-adaptive version and 31h31 per epoch for adaptive version. The slow computation time of adaptive SPIDNA is mostly due to data being inputted one by one in the network instead of concatenated in tensors.

A.2 ABC predictions from Boitard et al. (2016b)

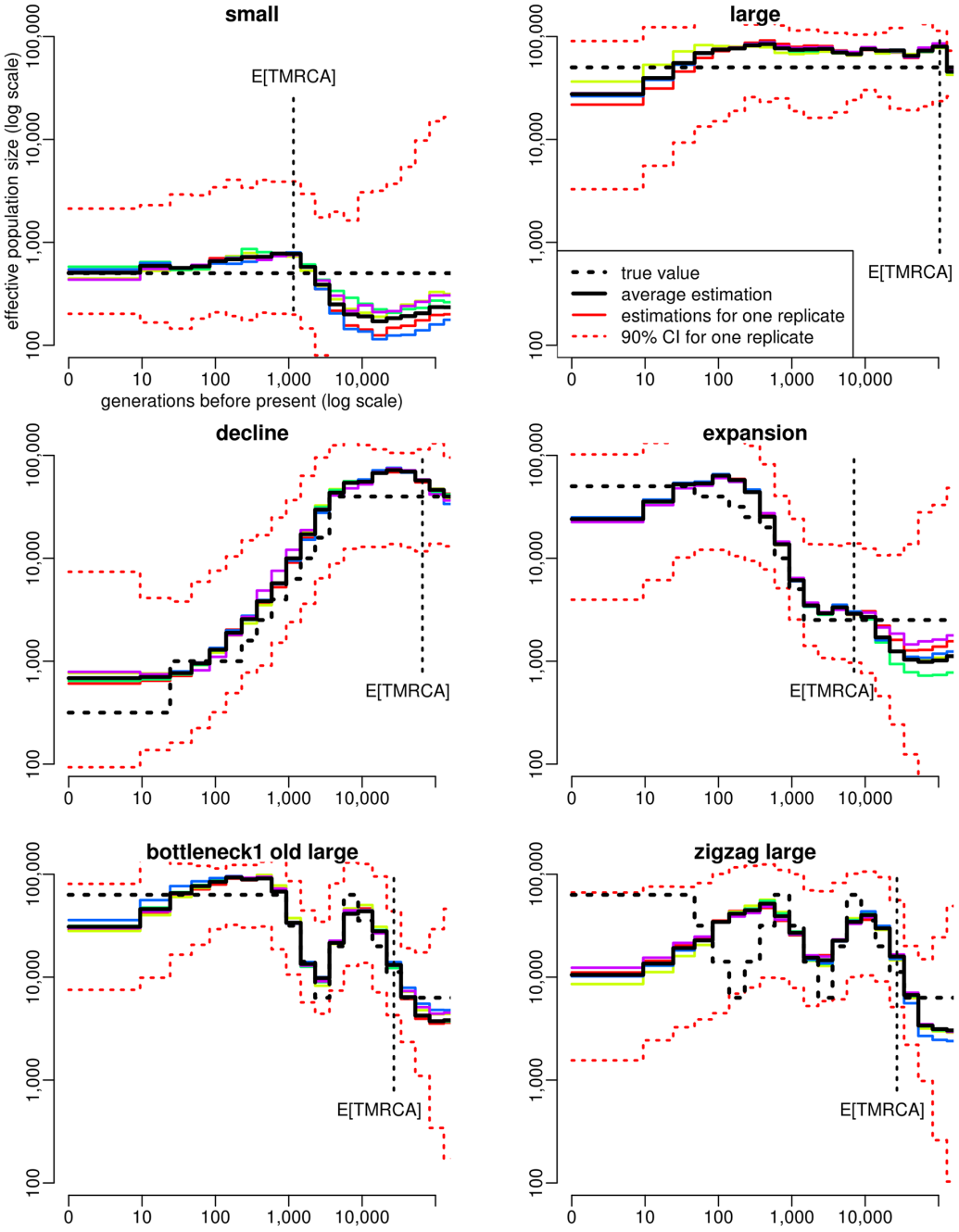

Figure A.1: **Estimation of population size history using ABC in six different simulated scenarios from Boitard et al. (2016b).** From Boitard et al. (2016b): *a small constant population size (N = 500, top left), a large constant population size (N = 50,000, top right), a decline scenario mimicking the population size history in Holstein cattle (middle left), an expansion scenario mimicking the population size history in CEU human (middle right), a scenario with one expansion followed by one bottleneck (bottom left) and a zigzag scenario similar to that used in Schiffels and Durbin (2014) (bottom right), with one expansion followed by two bottlenecks. For each scenario, the true population size history is shown by the dotted black line, the average estimated history over 20 PODs is shown by the solid black line, the estimated histories for five random PODs are shown by solid colored lines, and the 90% credible interval for one of these PODs is shown by the dotted red lines. The expected time to the most recent common ancestor (TMRCA) of the sample, E[TMRCA], is indicated by the vertical dotted black line. Summary statistics considered in the ABC analysis were (i) the AFS and (ii) the average zygotic LD for several distance bins. These statistics were computed from n = 25 diploid individuals, using all SNPs for AFS statistics and SNPs with a MAF above 20% for LD statistics. The posterior distribution of each parameter was obtained by neural network regression (Blum and François, 2010), with a tolerance rate of 0.005. Population size point estimates were obtained from the median of the posterior distribution.*

A.3 Synthèse en français

Depuis les premières découvertes des principes régissant l'évolution des génomes, les généticiens n'ont eu cesse de proposer des modèles de plus en plus réalistes afin de décrire l'évolution des variations génétiques au sein d'une population. Ces modèles de d'évolution ont permis de mettre évidence que les génomes d'une population dépendent non seulement de paramètres comme le taux de mutation, de recombinaison ou encore de la sélection naturelle, mais aussi de son histoire démographique passée. Parallèlement, le développement de techniques de séquençage de plus en plus performantes a permis de constituer des bases de données pouvant regrouper jusqu'à plusieurs centaines de génomes d'une même population. Il est donc maintenant possible de retracer l'histoire de ces populations à l'aide de méthodes statistiques capables d'exploiter l'information laissée par les événements démographiques dans les génomes. Cette thèse présente de nouvelles méthodes basées sur l'apprentissage statistique profond (*deep learning*) pour l'inférence des paramètres démographiques d'une population et, plus particulièrement, sa taille efficace (c'est-à-dire du nombre d'individus sous les conditions d'un modèle d'évolution) dans le passé.

Les deux principaux réseaux de neurones présentés dans cette thèse sont inspirés d'architectures ayant fait leurs preuves dans d'autres domaines traitant des données en grande dimension, mais ils intègrent aussi des stratégies permettant de prendre en compte des caractéristiques propres aux données génomiques. La première, SPIDNA, utilise une organisation des neurones en couche de convolution issue du traitement d'image. Ces couches de convolution génèrent des filtres capables de détecter des motifs de mutations particuliers dans les séquences. La seconde architecture, MixAttSPIDNA est une amélioration de la première incluant un mécanisme d'attention inspiré par les réseaux traitant le langage naturel. Ce mécanisme permet de calculer des statistiques globales sur les différentes séquences d'un échantillon, tout en étant

invariant à leurs différentes permutations possibles. Ces deux architectures comportent aussi un enchainement de fonction invariante et équivariante afin d'assurer l'équivariance globale. De plus, leur empilement de couches assure aux réseaux la possibilité de détecter des dépendances entre des SNPs éloignés dans les séquences.

Afin d'être entrainés, ces réseaux ont besoin d'une grande quantité de données reliant des scénarios démographiques à des échantillons de séquences de populations. Cependant, une telle base de données n'existe pas pour des populations réelles, car leurs histoires démographiques précises sont très difficiles à caractériser autrement que par l'étude de leurs génomes. Il a donc fallu utiliser un simulateur basé sur un modèle d'évolution afin de constituer une base d'entrainement de taille conséquente. Pour chaque simulation, un scénario démographique est tiré au hasard d'un apriori sur les différents paramètres et est ensuite donné au simulateur afin qu'il génère un échantillon de séquences possibles. L'objectif de nos réseaux de neurone est donc de réaliser le processus inverse, et de retrouver les valeurs des paramètres démographiques à partir des séquences de l'échantillon. Une fois entrainé, le réseau peut être utilisé sur un échantillon de séquences d'une population réelle afin d'inférer sa démographie passée.

L'entrainement de ces réseaux vise à minimiser l'erreur quadratique moyenne de la taille efficace inférée au cours du temps. Cette métrique permet de comparer les performances des réseaux entrent elles, mais aussi de les comparer à d'autres méthodes plus couramment utilisées en génétique des populations comme l'*approximate Bayesian computation* (ABC). La comparaison présentée dans cette thèse montre qu'un réseau de neurones est capable d'obtenir des performances similaires ou meilleures que l'ABC, sans passer par une étape de réduction des données génétiques de grande dimension en des statistiques expertes. Plus précisément, l'architecture SPIDNA utilisant des données brutes donne des performances égales à l'ABC sur statistiques résumées lorsqu'elle est combinée à cette dernière. Quant à MixAttSPIDNA, elle montre des performances bien supérieures à toutes les autres méthodes testées et repose uniquement sur son réseau de neurones.

Cette thèse présente aussi d'autres expériences visant à mieux comprendre le fonctionnement des réseaux de neurones en comparant les activations des neurones aux statistiques classiquement utilisées en génétique des populations. D'autres expériences ont aussi permis de vérifier la robustesse des prédictions faites par les réseaux à certaines des perturbations pour lesquels elles n'ont pas été entrainées, comme la présence de sélection positive. Enfin, les méthodes développées ont été testées dans la pratique en inférant la démographie passée de populations réelles de *Bos taurus* et d'*Homo sapiens*. Les prédictions faites sur les populations de ces deux espèces ont ensuite été comparées à nos connaissances de leurs démographies passées afin d'avoir une deuxième source d'évaluation de nos méthodes.

Ainsi, la principale contribution de cette thèse est d'apporter de nouvelles méthodes profitant des avantages de l'apprentissage statistique profond que sont sa capacité à traiter la grande dimension des données génomiques sans les résumer par des statistiques expertes, sa flexibilité vis-à-vis des paramètres démographiques à prédire et ses performances comparables à l'état de l'art, tout cela en étant indépendant du modèle d'évolution employé. Pour leurs nombreux avantages, ces méthodes sont amenées à être de plus en plus utilisées en génétique des populations et plus généralement pour résoudre les taches d'inférences basées sur des données génomiques.

Bibliography

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

J. R. Adrion, J. G. Galloway, and A. D. Kern. Inferring the landscape of recombination using recurrent neural networks. *bioRxiv*, page 662247, 2019.

S. Aeschbacher, M. A. Beaumont, and A. Futschik. A novel approach for choosing summary statistics in approximate bayesian computation. *Genetics*, 192(3):1027–1047, 2012.

C. Aimé, G. Laval, E. Patin, P. Verdu, L. Ségurel, R. Chaix, T. Hegay, L. Quintana-Murci, E. Heyer, and F. Austerlitz. Human genetic data reveal contrasting demographic patterns between sedentary and nomadic populations that predate the emergence of farming. *Molecular biology and evolution*, 30(12):2629–2644, 2013.

S. Akaho. A kernel method for canonical correlation analysis. *arXiv preprint cs/0609071*, 2006.

B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8): 831–838, 2015.

G. Andrew, R. Arora, J. Bilmes, and K. Livescu. Deep canonical correlation analysis. In *International conference on machine learning*, pages 1247–1255. PMLR, 2013.

L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730*, 2018.

D. Balduzzi, M. Frean, L. Leary, J. Lewis, K. W.-D. Ma, and B. McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In *International Conference on Machine Learning*, pages 342–350. PMLR, 2017.

C. Battey, G. C. Coffing, and A. D. Kern. Visualizing population structure with variational autoencoders. *G3*, 11(1):1–11, 2021.

D. R. Bentley, S. Balasubramanian, H. P. Swerdlow, G. P. Smith, J. Milton, C. G. Brown, K. P. Hall, D. J. Evers, C. L. Barnes, H. R. Bignell, et al. Accurate whole human genome sequencing using reversible terminator chemistry. *nature*, 456(7218):53–59, 2008.

A. Bergström, S. A. McCarthy, R. Hui, M. A. Almarri, Q. Ayub, P. Danecek, Y. Chen, S. Felkel, P. Hallast, J. Kamm, et al. Insights into human genetic variation and population history from 929 diverse genomes. *Science*, 367(6484), 2020.

M. G. Blum. Approximate bayesian computation: a nonparametric perspective. *Journal of the American Statistical Association*, 105(491):1178–1187, 2010.

M. G. Blum and O. François. Non-linear regression models for approximate bayesian computation. *Statistics and computing*, 20(1):63–73, 2010.

M. G. Blum, M. A. Nunes, D. Prangle, S. A. Sisson, et al. A comparative review of dimension reduction methods in approximate bayesian computation. *Statistical Science*, 28 (2):189–208, 2013.

S. Boitard, M. Boussaha, A. Capitan, D. Rocha, and B. Servin. Uncovering adaptation from sequence data: lessons from genome resequencing of four cattle breeds. *Genetics*, 203(1):433–450, 2016a.

S. Boitard, W. Rodriguez, F. Jay, S. Mona, and F. Austerlitz. Inferring population size history from large samples of genome-wide molecular data-an approximate bayesian computation approach. *PLoS genetics*, 12(3):e1005877, 2016b.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

M. Bridges, E. A. Heron, C. O'Dushlaine, R. Segurado, D. Morris, A. Corvin, M. Gill, C. Pinto, I. S. Consortium, et al. Genetic classification of populations using supervised learning. *PloS one*, 6(5):e14802, 2011.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

S. R. Browning and B. L. Browning. Accurate non-parametric estimation of recent effective population size from segments of identity by descent. *The American Journal of Human Genetics*, 97(3):404–418, 2015.

K. E. Burger, P. Pfaffelhuber, and F. Baumdicker. Neural networks for self-adjusting mutation rate estimation when the recombination rate is unknown. *bioRxiv*, 2021.

J. Chan, V. Perrone, J. Spence, P. Jenkins, S. Mathieson, and Y. Song. A likelihood-free inference framework for population genetic data using exchangeable neural networks. In *Advances in Neural Information Processing Systems*, pages 8594–8605, 2018.

A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. Generative pre-training from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020.

F. Collins, E. Lander, J. Rogers, R. Waterston, and I. Conso. Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–945, 2004.

. G. P. Consortium et al. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061, 2010.

. G. P. Consortium et al. A global reference for human genetic variation. *Nature*, 526 (7571):68, 2015.

B. H. Consortium, R. A. Gibbs, J. F. Taylor, C. P. Van Tassell, W. Barendse, K. A. Eversole, C. A. Gill, R. D. Green, D. L. Hamernik, S. M. Kappes, et al. Genome-wide survey of snp variation uncovers the genetic structure of cattle breeds. *Science*, 324(5926):528–532, 2009.

K. Cranmer, J. Brehmer, and G. Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

K. Csilléry, O. François, and M. G. Blum. abc: an r package for approximate bayesian computation (abc). *Methods in ecology and evolution*, 3(3):475–479, 2012.

J. Cury, B. C. Haller, G. Achaz, and F. Jay. Simulation of bacterial populations with slim. *bioRxiv*, 2021. doi: 10.1101/2020.09.28.316869. URL https://www.biorxiv.org/content/early/2021/03/03/2020.09.28.316869.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

H. D. Daetwyler, A. Capitan, H. Pausch, P. Stothard, R. Van Binsbergen, R. F. Brøndum, X. Liao, A. Djari, S. C. Rodriguez, C. Grohs, et al. Whole-genome sequencing of 234 bulls facilitates mapping of monogenic and complex traits in cattle. *Nature genetics*, 46(8):858, 2014.

W. Deelder, E. D. Benavente, J. Phelan, E. Manko, S. Campino, L. Palla, and T. G. Clark. Using deep learning to identify recent positive selection in malaria parasite sequence data. *Malaria journal*, 20(1):1–9, 2021.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

G. Ewing and J. Hermisson. Msms: a coalescent simulation program including recombination, demographic structure and selection at a single locus. *Bioinformatics*, 26(16): 2064–2065, 2010.

L. Excoffier, I. Dupanloup, E. Huerta-Sánchez, V. C. Sousa, and M. Foll. Robust demographic inference from genomic and snp data. *PLoS genetics*, 9(10):e1003905, 2013.

L. Excoffier, N. Marchi, D. A. Marques, R. Matthey-Doret, A. Gouy, and V. C. Sousa. fastsimcoal2: demographic inference under complex evolutionary scenarios. *Bioinformatics*, 37(24):4882–4885, 2021.

A. N. Fadja, F. Riguzzi, G. Bertorelle, and E. Trucchi. Identification of natural selection in genomic data with deep convolutional neural network. 2021.

S. Falkner, A. Klein, and F. Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1437–1446. PMLR, 2018a.

S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018b. PMLR. URL http://proceedings.mlr.press/v80/falkner18a.html.

P. Fearnhead and D. Prangle. Constructing summary statistics for approximate bayesian computation: semi-automatic approximate bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):419–474, 2012.

R. A. Fisher. *The genetical theory of natural selection*. 1958.

L. Flagel, Y. Brandvain, and D. R. Schrider. The unreasonable effectiveness of convolutional neural networks in population genetic inference. *Molecular biology and evolution*, 36(2):220–238, 2018.

N. R. Garud, P. W. Messer, E. O. Buzbas, and D. A. Petrov. Recent selective sweeps in north american drosophila melanogaster show signatures of soft sweeps. *PLoS genetics*, 11(2):e1005004, 2015.

L. M. Gattepaille, M. Jakobsson, and M. G. Blum. Inferring population size changes with sequence and snp data: lessons from human bottlenecks. *Heredity*, 110(5):409–419, 2013.

M. Geiger, A. Jacot, S. Spigler, F. Gabriel, L. Sagun, S. d' Ascoli, G. Biroli, C. Hongler, and M. Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, Feb

2020. ISSN 1742-5468. doi: 10.1088/1742-5468/ab633c. URL http://dx.doi.org/10.1088/1742-5468/ab633c.

A. L. Gladstein and M. F. Hammer. Substructured population growth in the ashkenazi jews inferred with approximate bayesian computation. *Molecular biology and evolution*, 36(6):1162–1171, 2019.

B. H. Good, M. J. McDonald, J. E. Barrick, R. E. Lenski, and M. M. Desai. The dynamics of molecular evolution over 60,000 generations. *Nature*, 551(7678):45–50, 2017.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

G. Gower, P. I. Picazo, M. Fumagalli, and F. Racimo. Detecting adaptive introgression in human evolution using convolutional neural networks. *Elife*, 10:e64669, 2021.

S. Gravel. Population genetics models of local ancestry. *Genetics*, 191(2):607–619, 2012.

M. Graziani, V. Andrearczyk, and H. Müller. Regression concept vectors for bidirectional explanations in histopathology. In *Understanding and Interpreting Machine Learning in Medical Image Computing Applications*, pages 124–132. Springer, 2018.

R. N. Gutenkunst, R. D. Hernandez, S. H. Williamson, and C. D. Bustamante. Inferring the joint demographic history of multiple populations from multidimensional snp frequency data. *PLoS genetics*, 5(10):e1000695, 2009.

B. C. Haller and P. W. Messer. Slim 3: Forward genetic simulations beyond the wright–fisher model. *Molecular biology and evolution*, 36(3):632–637, 2019.

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

K. Harris and R. Nielsen. Inferring demographic history from a spectrum of shared haplotype lengths. *PLoS genetics*, 9(6):e1003521, 2013.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

H. A. Hejase, Z. Mo, L. Campagna, and A. Siepel. Sia: Selection inference using the ancestral recombination graph. *bioRxiv*, 2021.

R. D. Hernandez. A flexible forward simulator for populations subject to selection and demography. *Bioinformatics*, 24(23):2786–2787, 2008.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

R. R. Hudson. Properties of a neutral allele model with intragenic recombination. *Theoretical population biology*, 23(2):183–201, 1983.

R. R. Hudson. ms a program for generating samples under neutral models. 2004.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

U. Isildak, A. Stella, and M. Fumagalli. Distinguishing between recent balancing selection and incomplete sweep using deep neural networks. *Molecular Ecology Resources*, 2021.

K. Jaganathan, S. K. Panagiotopoulou, J. F. McRae, S. F. Darbandi, D. Knowles, Y. I. Li, J. A. Kosmicki, J. Arbelaez, W. Cui, G. B. Schwartz, et al. Predicting splicing from primary sequence with deep learning. *Cell*, 176(3):535–548, 2019.

F. Jay, S. Boitard, and F. Austerlitz. An abc method for whole-genome sequence data: inferring paleolithic and neolithic human expansions. *Molecular biology and evolution*, 36(7):1565–1579, 2019.

B. Jiang, T.-y. Wu, C. Zheng, and W. H. Wong. Learning summary statistic for approximate bayesian computation via deep neural network. *Statistica Sinica*, pages 1595–1618, 2017.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, pages 1–11, 2021.

J. Kelleher, A. M. Etheridge, and G. McVean. Efficient coalescent simulation and genealogical analysis for large sample sizes. *PLoS computational biology*, 12(5): e1004842, 2016.

E. Kerdoncuff, A. Lambert, and G. Achaz. Testing for population decline using maximal linkage disequilibrium blocks. *Theoretical population biology*, 134:171–181, 2020.

A. D. Kern and D. R. Schrider. diplos/hic: an updated approach to classifying selective sweeps. *G3: Genes, Genomes, Genetics*, 8(6):1959–1970, 2018.

B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

M. Kimura. Diffusion models in population genetics. *Journal of Applied Probability*, 1(2): 177–232, 1964.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

J. F. Kingman. On the genealogy of large populations. *Journal of applied probability*, 19 (A):27–43, 1982.

P. Kirschner, M. F. Perez, E. Záveská, I. Sanmartín, L. Marquer, B. C. Schlick-Steiner, N. Alvarez, F. M. Steiner, and P. Schönswetter. Congruent evolutionary responses of european steppe biota to late quaternary climate change. *Nature Communications*, 13 (1):1–11, 2022.

G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *Proceedings of the 31st international conference on neural information processing systems*, pages 972–981, 2017.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

L. Leitsalu, T. Haller, T. Esko, M.-L. Tammesoo, H. Alavere, H. Snieder, M. Perola, P. C. Ng, R. Mägi, L. Milani, et al. Cohort profile: Estonian biobank of the estonian genome center, university of tartu. *International journal of epidemiology*, 44(4):1137–1147, 2014.

L. Leitsalu, T. Haller, T. Esko, M.-L. Tammesoo, H. Alavere, H. Snieder, M. Perola, P. C. Ng, R. Mägi, L. Milani, et al. Cohort profile: Estonian biobank of the estonian genome center, university of tartu. *International journal of epidemiology*, 44(4):1137–1147, 2015.

H. Li and R. Durbin. Inference of human population history from individual whole-genome sequences. *Nature*, 475(7357):493–496, 2011.

H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017a.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*, 2016.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017b.

M. Liang and R. Nielsen. The lengths of admixture tracts. *Genetics*, 197(3):953–967, 2014.

H. Lin and S. Jegelka. Resnet with one-neuron hidden layers is a universal approximator. *arXiv preprint arXiv:1806.10909*, 2018.

X. Liu and Y.-X. Fu. Exploring population size changes using snp frequency spectra. *Nature genetics*, 47(5):555–559, 2015.

B. Lorente-Galdos, O. Lao, G. Serra-Vidal, G. Santpere, L. F. Kuderna, L. R. Arauna, K. Fadhlaoui-Zid, V. N. Pimenoff, H. Soodyall, P. Zalloua, et al. Whole-genome sequence analysis of a pan african set of samples reveals archaic gene flow from an extinct basal population of modern humans into sub-saharan populations. *Genome biology*, 20(1):77, 2019.

E. D. Lorenzen, D. Nogués-Bravo, L. Orlando, J. Weinstock, J. Binladen, K. A. Marske, A. Ugan, M. K. Borregaard, M. T. P. Gilbert, R. Nielsen, et al. Species-specific responses of late quaternary megafauna to climate and humans. *Nature*, 479(7373):359–364, 2011.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

T. Lucas, C. Tallec, Y. Ollivier, and J. Verbeek. Mixed batches and symmetric discriminators for GAN training. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2844–2853, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/lucas18a.html.

W. Ma, Z. Qiu, J. Song, J. Li, Q. Cheng, J. Zhai, and C. Ma. A deep convolutional neural network approach for predicting phenotypes from genotypes. *Planta*, 248(5):1307–1318, 2018.

I. M. MacLeod, D. M. Larkin, H. A. Lewin, B. J. Hayes, and M. E. Goddard. Inferring Demography from Runs of Homozygosity in Whole-Genome Sequence, with Correction for Sequence Errors. *Molecular Biology and Evolution*, 30(9):2209–2223, 07 2013. ISSN 0737-4038. doi: $10.1093/\mathrm{molbev}/\mathrm{mst125}$. URL https://doi.org/10.1093/molbev/mst125.

S. Mallick, H. Li, M. Lipson, I. Mathieson, M. Gymrek, F. Racimo, M. Zhao, N. Chennagiri, S. Nordenfelt, A. Tandon, et al. The simons genome diversity project: 300 genomes from 142 diverse populations. *Nature*, 538(7624):201, 2016.

P. Marjoram and J. D. Wall. Fast" coalescent" simulation. *BMC genetics*, 7(1):1–9, 2006.

O. Mazet, W. Rodríguez, S. Grusea, S. Boitard, and L. Chikhi. On the importance of being structured: instantaneous coalescence rates and human evolution—lessons for ancestral population size inference? *Heredity*, 116(4):362–371, 2016.

G. A. McVean and N. J. Cardin. Approximating the coalescent with recombination. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1459):1387–1393, 2005.

A. Miles, P. Ralph, S. Rae, and R. Pisupati. cggh/scikit-allel: v1.2.1, June 2019. URL https://doi.org/10.5281/zenodo.3238280.

M. Mondal, J. Bertranpetit, and O. Lao. Approximate bayesian computation with deep learning supports a third archaic introgression in asia and oceania. *Nature communications*, 10(1):246, 2019.

D. M. Montserrat, C. Bustamante, and A. Ioannidis. Class-conditional vae-gan for local-ancestry simulation. *arXiv preprint arXiv:1911.13220*, 2019.

P. A. P. Moran. Random processes in genetics. In *Mathematical proceedings of the cambridge philosophical society*, volume 54, pages 60–71. Cambridge University Press, 1958.

S. Nakagome, K. Fukumizu, and S. Mano. Kernel approximate bayesian computation in population genetic inferences. *Statistical applications in genetics and molecular biology*, 12(6):667–678, 2013.

R. Nielsen, J. M. Akey, M. Jakobsson, J. K. Pritchard, S. Tishkoff, and E. Willerslev. Tracing the peopling of the world through genomics. *Nature*, 541(7637):302–310, 2017.

M. Nordborg. Coalescent theory. *Handbook of statistical genetics*, 2001.

L. Pagani, D. J. Lawson, E. Jagoda, A. Mörseburg, A. Eriksson, M. Mitt, F. Clemente, G. Hudjashov, M. DeGiorgio, L. Saag, et al. Genomic analyses inform on migration events during the peopling of eurasia. *Nature*, 538(7624):238, 2016.

P. F. Palamara, T. Lencz, A. Darvasi, and I. Pe'er. Length distributions of identity by descent reveal fine-scale demographic history. *The American Journal of Human Genetics*, 91(5):809–822, 2012.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

J. Prado-Martinez, P. H. Sudmant, J. M. Kidd, H. Li, J. L. Kelley, B. Lorente-Galdos, K. R. Veeramah, A. E. Woerner, T. D. O'Connor, G. Santpere, et al. Great ape genetic diversity and population history. *Nature*, 499(7459):471, 2013.

J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman. Population growth of human y chromosomes: a study of y chromosome microsatellites. *Molecular biology and evolution*, 16(12):1791–1798, 1999.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Point-net: Deep learning on point sets for 3d classification and segmentation. corr abs/1612.00593 (2016). *arXiv preprint arXiv:1612.00593*, 2016.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *arXiv preprint arXiv:1706.05806*, 2017.

A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021.

L. Raynal, J.-M. Marin, P. Pudlo, M. Ribatet, C. P. Robert, and A. Estoup. Abc random forests for bayesian parameter inference. *Bioinformatics*, 35(10):1720–1728, 2018.

L. Raynal, J.-M. Marin, P. Pudlo, M. Ribatet, C. P. Robert, and A. Estoup. Abc random forests for bayesian parameter inference. *Bioinformatics*, 35(10):1720–1728, 2019.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

T. Sanchez, E. M. Bray, P. Jobic, J. Guez, G. Charpiat, J. Cury, and F. Jay. dnadna: Deep neural architectures for dna-a deep learning framework for population genetic inference. 2021a.

T. Sanchez, J. Cury, G. Charpiat, and F. Jay. Deep learning for population size history inference: Design, comparison and combination with approximate bayesian computation. *Molecular Ecology Resources*, 21(8):2645–2660, 2021b.

C. Sandor, W. Li, W. Coppieters, T. Druet, C. Charlier, and M. Georges. Genetic variants in rec8, rnf212, and prdm9 influence male recombination in cattle. *PLoS genetics*, 8 (7), 2012.

S. Schiffels and R. Durbin. Inferring human population size and separation history from multiple genome sequences. *Nature genetics*, 46(8):919–925, 2014.

F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, Oct 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. URL http://dx.doi.org/10.1007/s11263-019-01228-7.

D. Sharma, A. Durand, M.-A. Legault, L.-P. L. Perreault, A. Lemaçon, M.-P. Dubé, and J. Pineau. Deep interpretability for gwas. *arXiv preprint arXiv:2007.01516*, 2020.

S. Sheehan and Y. S. Song. Deep learning for population genetic inference. *PLoS computational biology*, 12(3):e1004845, 2016.

A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.

J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *null*, page 1470. IEEE, 2003.

C. C. Smith and S. M. Flaxman. Leveraging whole genome sequencing data for demographic inference with approximate bayesian computation. *Molecular ecology resources*, 2019.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

M. Spellings. Agglomerative attention. *arXiv preprint arXiv:1907.06607*, 2019.

Z. Su, J. Marchini, and P. Donnelly. Hapgen2: simulation of multiple disease snps. *Bioinformatics*, 27(16):2304–2305, 2011.

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

E. G. Tabak and E. Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

F. Tajima. Statistical method for testing the neutral mutation hypothesis by dna polymorphism. *Genetics*, 123(3):585–595, 1989.

S. Tavaré, D. J. Balding, R. C. Griffiths, and P. Donnelly. Inferring coalescence times from dna sequence data. *Genetics*, 145(2):505–518, 1997.

Y. W. Teh and G. E. Hinton. Rate-coded restricted boltzmann machines for face recognition. *Advances in neural information processing systems*, pages 908–914, 2001.

J. Terhorst, J. A. Kamm, and Y. S. Song. Robust and scalable inference of population history from hundreds of unphased whole genomes. *Nature genetics*, 49(2):303–309, 2017.

L. Torada, L. Lorenzon, A. Beddis, U. Isildak, L. Pattini, S. Mathieson, and M. Fumagalli. Imagene: a convolutional neural network to quantify natural selection from genomic data. *BMC bioinformatics*, 20(9):337, 2019.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

P. Verdu, T. J. Pemberton, R. Laurent, B. M. Kemp, A. Gonzalez-Oliver, C. Gorodezky, C. E. Hughes, M. R. Shattuck, B. Petzelt, J. Mitchell, et al. Patterns of admixture and population structure in native populations of northwest north america. *PLoS genetics*, 10(8):e1004530, 2014.

F. A. Villanea and J. G. Schraiber. Multiple episodes of interbreeding between neanderthal and modern humans. *Nature ecology & evolution*, 3(1):39–44, 2019.

Z. Wang, J. Wang, M. Kourakos, N. Hoang, H. H. Lee, I. Mathieson, and S. Mathieson. Automatic inference of demographic parameters using generative adversarial networks. *bioRxiv*, 2020.

J. A. Wegelin. A survey of partial least squares (pls) methods, with emphasis on the two-block case. 2000.

W. Weinberg. Über vererbungsgesetze beim menschen. *Zeitschrift für induktive Abstammungs-und Vererbungslehre*, 1(1):440–460, 1908.

C. Wiuf and J. Hein. Recombination as a point process along sequences. *Theoretical population biology*, 55(3):248–259, 1999.

S. Wright. Evolution in mendelian populations. *Genetics*, 16(2):97, 1931.

Y. Wu and K. He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

A. T. Xue, D. R. Schrider, A. D. Kern, A. Consortium, et al. Discovery of ongoing selective sweeps within anopheles mosquito populations using deep learning. *bioRxiv*, page 589069, 2019.

B. Yelmen, A. Decelle, L. Ongaro, D. Marnetto, C. Tallec, F. Montinaro, C. Furtlehner, L. Pagani, and F. Jay. Creating artificial human genomes using generative models. *bioRxiv*, page 769091, 2019.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

M. A. Zeder. Domestication and early agriculture in the mediterranean basin: Origins, diffusion, and impact. *Proceedings of the national Academy of Sciences*, 105(33):11597–11604, 2008.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization, 2017.

Q. Zhang, Y. Yang, H. Ma, and Y. N. Wu. Interpreting cnns via decision trees. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6261–6270, 2019.

J. Zhou and O. G. Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931–934, 2015.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Reconstruire notre passé : apprentissage statistique profond pour la génétique des populations.
Mots clés: Génétique des populations, Apprentissage statistique profond, Inférence démographique, Réseaux de neurones artificiels, Données de taille variable

Résumé: Avec l'explosion des technologies de séquençage, de plus en plus de données génomiques sont disponibles, ouvrant la voie à une connaissance approfondie des forces évolutives en œuvre et en particulier de l'histoire démographique des populations. Toutefois, extraire l'information intéressante de ces données massives de manière efficace reste un problème ouvert. Compte tenu de leurs récents succès en apprentissage statistique, les réseaux de neurones artificiels sont un candidat sérieux pour mener à bien une telle analyse. Ces méthodes ont l'avantage de pouvoir traiter des données ayant une grande dimension, de s'adapter à la plupart des problèmes et d'être facilement mis à l'échelle des moyens de calcul disponibles. Cependant, leur performance dépend fortement de leur architecture qui requiert d'être en adéquation avec les propriétés des données afin d'en tirer le maximum d'information. Dans ce cadre, cette thèse présente de nouvelles approches basées sur l'apprentissage statistique profond, ainsi que les principes permettant de concevoir des architectures adaptées aux caractéristiques des données génomiques. L'utilisation de couches de convolution et de mécanismes d'attention permet aux réseaux présentés d'être invariants aux permutations des haplotypes échantillonnés et de s'adapter à des données de dimensions différentes (nombre d'haplotypes et de sites polymorphes). Les expériences conduites sur des données simulées démontrent l'efficacité de ces approches en les comparant à des architectures de réseaux plus classiques, ainsi qu'à des méthodes issues de l'état de l'art. De plus, la possibilité d'assembler les réseaux de neurones à certaines méthodes déjà éprouvées en génétique des populations, comme l'*approximate Bayesian computation*, permet d'améliorer les résultats et de combiner leurs avantages. La praticabilité des réseaux de neurones pour l'inférence démographique est testée grâce à leur application à des séquences génomiques complètes provenant de populations réelles de *Bos taurus* et d'*Homo sapiens*. Enfin, les scénarios obtenus sont comparés aux connaissances actuelles de l'histoire démographique de ces populations.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Reconstructing our past: deep learning for population genetics

Keywords: Population genetics, Deep learning, Demographic inference, Artificial neural networks, Data of variable size

Abstract: Constant improvement of DNA sequencing technology that produces large quantities of genetic data should greatly enhance our knowledge of evolution, particularly demographic history. However, the best way to extract information from this large-scale data is still an open problem. Neural networks are a strong candidate to attain this goal, considering their recent success in machine learning. These methods have the advantages of handling high-dimensional data, adapting to most applications and scaling efficiently to available computing resources. However, their performance dependents on their architecture, which should match the data properties to extract the maximum information. In this context, this thesis presents new approaches based on deep learning, as well as the principles for designing architectures adapted to the characteristics of genomic data. The use of convolution layers and attention mechanisms allows the presented networks to be invariant to the sampled haplotypes' permutations and to adapt to data of different dimensions (number of haplotypes and polymorphism sites). Experiments conducted on simulated data demonstrate the efficiency of these approaches by comparing them to more classical network architectures, as well as to state-of-the-art methods. Moreover, coupling neural networks with some methods already proven in population genetics, such as the *approximate Bayesian computation*, improves the results and combines their advantages. The practicality of neural networks for demographic inference is tested on whole genome sequence data from real populations of *Bos taurus* and *Homo sapiens*. Finally, the scenarios obtained are compared with current knowledge of the demographic history of these populations.