



HAL
open science

Deep learning for population size history inference: Design, comparison and combination with approximate Bayesian computation

Théophile Sanchez, Jean Cury, Guillaume Charpiat, Flora Jay

► To cite this version:

Théophile Sanchez, Jean Cury, Guillaume Charpiat, Flora Jay. Deep learning for population size history inference: Design, comparison and combination with approximate Bayesian computation. *Molecular Ecology Resources*, 2020, 21 (8), pp.2645-2660. <10.1111/1755-0998.13224>. <hal-02942328>

HAL Id: hal-02942328

<https://hal.science/hal-02942328v1>

Submitted on 24 Nov 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

DEEP LEARNING FOR POPULATION SIZE HISTORY INFERENCE: DESIGN, COMPARISON AND COMBINATION WITH APPROXIMATE BAYESIAN COMPUTATION

Théophile Sanchez^{1*}, Jean Cury¹, Guillaume Charpiat¹, Flora Jay^{1*}

1. Laboratoire de Recherche en Informatique, CNRS UMR 8623, Université Paris-Saclay, Inria, Orsay, France

* Correspondence: theophile.sanchez@inria.fr and flora.jay@lri.fr

ABSTRACT

1 For the past decades, simulation-based likelihood-free inference methods have enabled researchers to
2 address numerous population genetics problems. As the richness and amount of simulated and real
3 genetic data keep increasing, the field has a strong opportunity to tackle tasks that current methods
4 hardly solve. However, high data dimensionality forces most methods to summarize large genomic
5 datasets into a relatively small number of handcrafted features (summary statistics). Here we propose
6 an alternative to summary statistics, based on the automatic extraction of relevant information using
7 deep learning techniques. Specifically, we design artificial neural networks (ANNs) that take as input
8 single nucleotide polymorphic sites (SNPs) found in individuals sampled from a single population and
9 infer the past effective population size history. First, we provide guidelines to construct artificial neural
10 networks that comply with the intrinsic properties of SNP data such as invariance to permutation
11 of haplotypes, long scale interactions between SNPs and variable genomic length. Thanks to a
12 Bayesian hyperparameter optimization procedure, we evaluate the performance of multiple networks
13 and compare them to well established methods like Approximate Bayesian Computation (ABC).
14 Even without the expert knowledge of summary statistics, our approach compares fairly well to an
15 ABC approach based on handcrafted features. Furthermore, we show that combining deep learning
16 and ABC can improve performance while taking advantage of both frameworks. Finally, we apply
17 our approach to reconstruct the effective population size history of cattle breed populations.

18 **1 Introduction**

19 In the past years, fields such as computer vision and natural language processing have shown impressive results thanks to
20 the rise of deep learning methods. What makes these methods powerful is not fully understood yet, but one key element
21 is their ability to handle and exploit high dimensional structured data. Therefore, deep learning seems particularly
22 suited to extract relevant information from genomic data. It has indeed been used for many tasks outside population
23 genetics, such as detection of alternative splicing sites, prediction of protein binding sites or other phenotype markers
24 (Alipanahi et al., 2015, Jaganathan et al., 2019, Ma et al., 2018).

25 As genomic data become more and more available, it is now possible to leverage genetic variation within species or
26 populations to investigate complex demographic histories including multiple admixture events, population structure or
27 size fluctuation through time. In fact, initiatives like the 1000 Genomes Project for human populations (Consortium
28 et al., 2010) have been extended for better world coverage and data quality (Bergström et al., 2019, Consortium et al.,
29 2015, Leitsalu et al., 2014, Mallick et al., 2016, Pagani et al., 2016) and opened up to many other species such as *Bos*
30 *taurus* with the 1000 Bull Genomes Project (Daetwyler et al., 2014) or chimpanzees and gorillas with the Great Apes
31 Genome Project (Prado-Martinez et al., 2013). Even for smaller scale studies, researchers often have access to the
32 whole genomes or high-density SNP data of numerous samples. These data collections can only be analyzed with
33 inference methods able to scale to dozens or hundreds of individuals and large numbers of genetic markers.

34 In this study, we propose several deep learning approaches for reconstructing the detailed histories of past effective
35 population sizes from genetic polymorphism within a single population, a task considered difficult for various reasons.
36 First, a present-day population, and even more so a sample of it, is one among many possible outcomes of a stochastic
37 process depending on population sizes, mutations and recombinations. Second, many other factors such as selective
38 pressure, admixture events or population structure also shape the contemporary genetic diversity, which can blur
39 the link between population size history and genetic data. As a result, the accuracy of the reconstruction and its
40 level of resolution depend on the number of individuals available, the quality of the data and the methodology used.
41 Nonetheless, in practice previous methods such as Bayesian skyline plots and their derivatives (Ho and Shapiro, 2011),
42 sequential Markov coalescent (SMC) (PSMC, diCal and their derivatives (Li and Durbin, 2011, Sheehan et al., 2013)),
43 Approximate Bayesian Computation (Boitard et al., 2016b, Navascués et al., 2017) and SFS-based approaches (Bhaskar
44 et al., 2015, Liu and Fu, 2015) have shown great results, supporting archaeological evidence and helping to understand
45 species decline or expansion.

46 The study of genetic variation relies primarily on genotyping and sequencing data of very high dimensionality, which is
47 a major difficulty for most inference methods. Some approaches, such as coalescent-HMMs methods (Spence et al.,
48 2018), enable parameter inference using the full dataset by making simplifying assumptions on the underlying models.
49 A few of them can process unphased data (Terhorst et al., 2017), scale to large sample sizes (Terhorst et al., 2017)
50 or to complex models (Steinrücken et al., 2019). However, no method simultaneously addresses all three. Moreover,
51 handling arbitrarily complex models remains untested (e.g. models with more than three populations) or intractable (e.g.

52 complex spatial models) (Spence et al., 2018). Hence, most frameworks solving complex population genetic tasks do
53 not rely on coalescent-HMMs and reduce the data dimension with a pre-processing step during which the dataset is
54 converted into a smaller set of statistics called summary statistics. These statistics can then be used in likelihood and
55 composite likelihood inference frameworks, when the model or statistics are simple enough, or in simulation-based
56 approaches. Among the latter, the widely used Approximate Bayesian Computation (ABC) framework as well as
57 several machine-learning algorithms, including Support Vector Machines (SVM) and random forests, were able to
58 tackle a variety of tasks such as demographic model selection and parameter inference (Excoffier et al., 2013, Jay et al.,
59 2019), detection of selection (Sugden et al., 2018, Tournéize et al., 2019) and introgression (Schridder et al., 2018). The
60 current trend when addressing complex tasks is to include a large number of summary statistics inspired by population
61 genetic theory in order to minimize the information loss. Summary statistics commonly used are the site frequency
62 spectrum (SFS) and its summaries (e.g. Tajima D), linkage disequilibrium (LD) and statistics based on shared segments
63 that are identical-by-state (IBS) or identical-by-descent (IBD) (Gladstein and Hammer, 2019, Jay et al., 2019, Sheehan
64 and Song, 2016, Smith and Flaxman, 2019). However, they are not guaranteed to be sufficient and the inclusion of
65 numerous statistics can impact the performance of standard ABC, a problem known as curse of dimensionality (Blum,
66 2010). An active research topic in the ABC community is thus the development of methods addressing this curse of
67 dimensionality by (i) selecting the best subset of summary statistics according to some information-based criteria, (ii)
68 integrating machine learning steps into ABC to handle a larger number of summary statistics (e.g. kernel methods,
69 random forests), (iii) constructing summary statistics using linear and non-linear models based on candidate statistics or
70 on the original data when feasible (Aeschbacher et al., 2012, Blum et al., 2013, Fearnhead and Prangle, 2012, Jiang
71 et al., 2017, Nakagome et al., 2013, Raynal et al., 2018).

72 In our study, we use deep learning, a method derived from machine learning. The objective of this method is to design a
73 function, represented by an artificial neural network (ANN), which is a differentiable computational graph organized as
74 a stack of linear and non-linear layers, with a high number of trainable parameters (usually thousands or millions). A
75 network layer takes as input the outputs of the previous layer(s): each node of the layer performs a linear combination
76 of the inputs, followed by a non-linear transformation, and this value is passed to the next layer. Networks vary in their
77 shape (number of layers and nodes) and in the way nodes are connected. For example a Multi Layer Perceptron (MLP)
78 connects all nodes of a layer to all nodes of the following layer (Rumelhart et al., 1986), while a Convolutional Neural
79 Network (CNN) connects only nodes of similar location (LeCun et al., 1995). Despite the differences, any network
80 defines a parameterized function that allows for a complex non-linear mapping from a space to another, and therefore
81 can solve a complex task, when the provided parameters are suitably adjusted. To tune the parameters, the network is
82 trained thanks to a *training set* consisting of examples of (input, desired output) pairs, by optimizing a criterion (*loss*
83 *function*) that expresses how well the network performs on the dataset with its current parameters. For example, for an
84 object recognition task in images, the input is an image, the output is a probability distribution over possible names of
85 objects, and the loss is the distance between the prediction of the ANN and the expected output (a Dirac peak on the
86 name of the object shown by the image). The parameters of the function are tuned to minimize this loss thanks to an

87 optimization algorithm based on gradient descent and backpropagation. This process usually requires a large training
88 dataset, in order for the network to be able to learn and generalize well, that is, to perform well on data never seen so far.
89 Deep learning has only recently been used to tackle population genetics questions. First, multilayer perceptrons (MLPs)
90 were used to process small SNP windows for population assignment (Bridges et al., 2011). Then, the same type
91 of architecture has been used to process large sets of summary statistics for predicting jointly selective sweeps and
92 simple demographic changes (Sheehan and Song, 2016). Villanea and Schraiber (2019) also applied MLP on summary
93 statistics to discriminate between multiple scenarios of archaic introgression and two other studies added an ABC step
94 to address a similar task (Lorente-Galdos et al., 2019, Mondal et al., 2019). A second type of ANN, convolutional
95 neural networks (CNNs), were then applied to summary statistics computed over 5Kb genomic regions in order to
96 predict selective sweeps (Xue et al., 2019). A considerable shift occurred when several studies applied ANNs directly
97 on genomic data instead of using summary statistic. Various CNN architectures processing SNP matrices were proposed
98 to infer recombination rates along the genome (Chan et al., 2018, Flagel et al., 2018), selection (Flagel et al., 2018,
99 Torada et al., 2019), introgression (Flagel et al., 2018) and three-step population size histories (Flagel et al., 2018).
100 The CNN implemented by Chan et al. (2018) and based on Deep Sets (Zaheer et al., 2017) is invariant to haplotype
101 (chromosome) permutation, i.e to the permutation of rows in the SNP matrix, thanks to convolution filters that treat each
102 haplotype in an identical way. The other approaches proposed instead to sort haplotypes by similarity before processing
103 them with filters sensitive to the haplotype order (Flagel et al., 2018, Torada et al., 2019). More recently, Recurrent
104 Neural Networks (RNN) were applied to estimate the recombination rate along the genome (Adrion et al., 2019), and
105 Generative Adversarial Networks (GAN) to learn the distribution of genomic datasets and generate artificial genomes
106 (Yelmen et al., 2019).

107 Among the variety of developed ANN architectures, it is not straightforward to know which one is the most adapted to
108 genomic data for a given population genetic task. In particular, this study aims at reconstructing detailed step-wise
109 effective population size histories with 21 size parameters under an unknown recombination rate, a complex model with
110 a fairly high dimensional parameter space compared to the population genetic tasks previously addressed with ANNs.
111 Hence we propose multiple networks, some of which are new and designed specifically for population genomics, and
112 others which are more basic. We then apply a hyperparameter optimisation procedure (BOHB, Falkner et al. (2018)) to
113 select the best architecture and hyperparameters. We investigate the performance of two MLPs, one using summary
114 statistics and one using SNP data of fixed length. We also compare two novel CNN based architectures, one with
115 mixed convolution filter sizes over multiple individuals and another CNN that is adaptive to the genomic input size
116 and invariant to the permutation of individuals or haplotypes. Both networks incorporate SNP data and their positions
117 (encoded as distances between SNPs), a concept also developed in a different fashion by Flagel et al. (2018). In our last
118 setup, we combine ABC and ANN by using the ANN predictions as summary statistics with the aim to benefit from
119 the advantages of both methods. Because no end-to-end deep learning approach for demographic inference had yet
120 been compared to ABC or other traditional methods, we carefully benchmarked all these networks against variations of
121 PopSizeABC, one of the highly performing methods for step-wise size inference that is based on ABC (Boitard et al.,

122 2016b). We also compare our architecture with CNNs developed for a related demographic task (Fligel et al., 2018).
123 Finally we apply our approach to real genomes in order to reconstruct the size history of three cattle breeds.

124 2 Materials and Methods

125 In this study, we introduce the first deep learning approaches for inferring detailed histories of effective population
126 sizes using genomic data. Based on whole sequences of SNP data of multiple individuals from a single population,
127 we aimed to predict 21 population size parameters, each corresponding to a time step. Our method and the baseline
128 frameworks all relied on large-scale simulated datasets for which the true demographic parameters are known and drawn
129 from prior distributions of population sizes and recombination rates. For each drawn parameter set (i.e. demographic
130 scenario), we simulated 100 independent genomic loci of length 2Mb (i.e. 100 replicates) for 50 haploid individuals
131 using msprime (Kelleher et al., 2016). Using this reference panel, we then trained methods based on ABC, deep learning
132 or a combination of both, to predict the demographic parameters (Figure 1). In this section, we will give an overview of
133 these methods as well as the hyperparameter optimisation procedure.

134 2.1 Simulated data and summary statistics

135 **Neutral simulations.** All methods compared in this study are trained in a supervised fashion, and thus require
136 simulated genetic data from numerous populations under various demographic scenarios. Following Boitard et al.
137 (2016b), we defined 21 time windows that grow exponentially when going further back in time, so that the most ancient
138 size change occurs 130,000 generations ago (Supplementary Text). The time windows are identical for all scenarios.
139 Each demographic scenario is generated by drawing a first population size N_0 between 10 and 100,000 from a uniform
140 distribution which corresponds to the most recent time window. The population sizes of the next time windows follow
141 $N_i = N_{i-1} \times 10^\beta$ for i in $[1, 21]$, with β sampled uniformly between -1 and 1. β is redrawn if it gives a population size
142 out of $]10; 100,000[$. We randomly drew from this prior distribution 50,000 scenarios and simulated 100 independent
143 2Mb-long segments of 50 haploid individuals for each scenario, using the msprime coalescent simulator version 0.6.1
144 (Kelleher et al., 2016). We obtained a total of 5,000,000 SNP matrices X of size $M = 50$ haplotypes $\times S$ SNP sites,
145 each associated with a vector of size S that contains the distances between SNPs (in bp). Ancestral and derived alleles
146 are encoded with 0 and 1. The mutation rate is set to 10^{-8} as in MacLeod et al. (2013). The recombination rate is
147 sampled uniformly between 10^{-9} and 10^{-8} for each scenario to be consistent with the estimations in cattle breeds
148 (Sandor et al., 2012). In order to compare all methods based on the same training panel we set a minimum threshold
149 of 400 SNPs per 2Mb region and designed the networks accordingly. After filtering and splitting (see Supplementary
150 Text) we obtained 1,796,100 SNP matrices for the training set, 50,000 for the validation set, and 76,700 for the test set.
151 Except stated otherwise, methods that are not adaptive to the number of SNPs used only the first 400 SNPs of each SNP
152 matrix. The proportion of these 400 SNPs kept among all SNPs from a simulated matrix is on average 28%. Finally, we
153 make the assumption that the 2Mb-long windows of a scenario are independent, which is true for simulated data but
154 not for real data. Information across windows (100 windows by scenario for simulated data and 1,213 for real data) is

155 combined during the summary statistics computation step for methods using summary statistics or by averaging the
 156 network predictions over all windows for methods using SNP matrices as input. Thus, the spatial information that may
 157 exist across these windows for real data is not conserved.

158 **Simulations with selection.** To investigate the robustness of our approach, an extra set of data was simulated under
 159 demographic changes and selective pressure. We used *msms* (Ewing and Hermisson, 2010) to simulate scenarios
 160 including positive selection with additive fitness using varying values of selection coefficient (s in $2N_e$ units: 100, 200,
 161 400 or 800), selection starting time (T_{sel} : 200, 1000 or 2000 generations ago) and initial frequency of the beneficial
 162 allele (f_0 : 0.1%, 1%, 5%). The SNP under selection was located at the center of the region. The mutation rate was set
 163 to 10^{-8} , the recombination rate to $5 \cdot 10^{-9}$, the number of haplotypes to 50 and the region length to 2Mb. We generated
 164 16×100 replicates for each of the 36 selection parameter combinations (s, T_{sel}, f_0) and 30×100 replicates with no
 165 selection under three demographic scenarios (constant, declining or expanding size) leading to a total of 181,800 SNP
 166 matrices. Inference methods requiring a fixed input size processed the 400 successive central SNPs (ie 200 before and
 167 200 after the SNP under selection).

168 **Summary statistics.** For each group of 100 segments corresponding to one scenario, we computed the site frequency
 169 spectrum and the linkage disequilibrium as a function of the distance between SNPs averaged over 19 distance bins for
 170 a total of 68 summary statistics. Our python script is partly based on the scikit-allel python module (Miles et al., 2019).
 171 These predefined summary statistics constitute the training, validation and test set for all methods based on summary
 172 statistics or on their combination with SNP matrices.

173 2.2 Baselines

174 We compared our approach to five baselines: an ABC approach and a MLP both using linkage disequilibrium and site
 175 frequency spectrum as summary statistics, and another MLP, a *custom* CNN and a CNN from (Flagel et al., 2018), all
 176 using genomic data directly. Each method is evaluated using its prediction error given by the following mean squared
 177 error:

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

178 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
 179 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
 180 neural networks, 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
 181 replicate (independent region) r .

182 **Approximate Bayesian Computation** We compared ABC with the simple rejection procedure (i.e. no correction) and
 183 three correction methods implemented in the R package 'abc' (Csilléry et al., 2012): local linear regression, ridge

184 regression and non-linear regression based on a single-hidden-layer neural network. Settings were set to default except
185 for the tolerance rate set to six possible values (0.05, 0.1, 0.15, 0.2, 0.25 and 0.3). ABC was run on (a) predefined
186 summary statistics, (b) SPIDNA outputs (i.e. automatically computed summary statistics), or (c) a combination of
187 predefined summary statistics and SPIDNA outputs. We used the median of the posterior distribution as the demographic
188 parameter estimate $\hat{\Theta}$.

189 **Multi-Layer-Perceptron Networks** The first MLP is based on summary statistics, has 3 hidden layers, ReLU activation
190 functions and uses batch normalisation. As in Sheehan and Song (2016), the hidden layers have respectively 25, 25, and
191 10 neurons. It takes 34 summary statistics as input. This network and all the following ones output 21 demographic
192 parameters and are trained with a regular L2 loss function and adam optimizer (Kingma and Ba, 2014) unless stated
193 otherwise. This MLP has a total of 2,986 trainable parameters. Our second MLP is based on 'raw' genomic data and
194 takes as input a matrix of 50 haplotypes (rows) for 400 SNPs (columns) and its associated vector of distances between
195 SNPs, both flattened into a single vector. Its hidden layers respectively have 20, 20, and 10 neurons, which gives it
196 408,981 trainable parameters.

197 **Custom CNN** Our convolutional neural network takes as input the same matrix of 400 SNPs and has 2-dimension
198 filters of various shapes. The first layer consists of 5 kernels with rectangular shape (2×2 , 5×4 , 3×8 , 2×10 , 20×1)
199 applied to the SNP matrix X . Each kernel creates 50 filters, which amounts to 250 feature maps after the first layer.
200 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
201 each, making 100 filters in total. The results of the first convolutional layer are then concatenated so that the second
202 convolutional layer will couple information from X and d in a way that emphasizes the original location of the SNPs
203 along the genome. The outputs of this second layer are then combined and go through 5 convolutional layers and 2
204 fully connected layers. Adding convolutional layers one after the other allows our network to combine patterns and
205 reduce the size of the data without adding too many weights to our model. This network has a total of 131,731 trainable
206 parameters.

207 **Flagel network** We reused the code associated with the repository of the first paper using a CNN for demographic
208 inference (Flagel et al., 2018) and adapted it to our dataset and task. We trained the network with the exact same
209 architecture as the one published, except that we changed the last layer to allow the prediction of our 21 population
210 size parameters. We parametrized the network with the set of hyperparameters leading to the best performance in the
211 previous work for two different types of SNP encoding (0/255 or -1/1). It is noteworthy that the actual encoding in their
212 code is 0/-1 and not 0/255, thus we kept the same encoding to be able to compare the performance. The networks were
213 trained with the same procedure of 10 epochs with early stopping in case of no progression of the loss after 3 epochs.
214 The batch size is 200. The input data had 50 haplotypes and either 400 SNPs as processed by our *custom* CNN or we
215 downsampled the data to one every ten SNPs as done in the original work, leading to 1,784 wide input SNP matrices.
216 This size corresponds to the tenth of the biggest SNP matrix in our dataset. Smaller simulations are padded with zeros.
217 All parameters can be found in table S1.

218 **2.3 Sequence Position Informed Deep Neural Architecture**

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

224 **2.3.1 Permutation invariance**

225 One of the SNP matrix properties is its invariance to the permutation of haploid or diploid individuals (rows of the
226 SNP matrix). The same matrix with permuted rows contain the exact same information and should lead to the same
227 predictions. Most summary statistics are already invariant to the haplotype order by definition. On the other hand,
228 typical operations used in ANNs such as rectangular filters and fully connected layers are not invariant, and consequently
229 our baseline ANNs do not respect this data feature. Here we implemented an architecture invariant by design, that stacks
230 functions equivariant and invariant to row permutations (Lucas et al., 2018). In our study, the equivariant function is a
231 convolutional layer with filters of size $1 \times a$, that treats each haplotype (row) independently and computes equivariant
232 features, while the invariant function computes the mean of these features over the row dimension. The invariant
233 function reduces the dimension of the data to one row, which is then concatenated to each equivariant row (Figure 2).
234 Therefore the correlation between rows increases at each layer, which progressively transforms the equivariant input to
235 an invariant output. However, the correlation increase should be moderate and progressive to avoid immediate loss
236 of the information at the haplotype level. To promote this, we perform two independent normalizations, one over the
237 output of the equivariant function and one over the input of the invariant function, and associate a correlation control
238 parameter α that quantifies the contribution of the invariant function to the next layer, thus controlling the speed at
239 which the correlation increases between rows.

240 **2.3.2 Convolution networks to handle data with variable size**

241 A major difficulty that arises with genomic data is that the number of SNP varies from one dataset to another, or
242 from one genomic region to another, due to the stochasticity of biological and demographic processes (and of their
243 corresponding genetic simulations). Therefore, we use convolution layers as they can handle data with variable size
244 while keeping the number of network weights constant. A filter can be repeated more or fewer times to cover the whole
245 input entering each layer, letting the network adapts itself to the data. Consequently, the output size of each convolution
246 layer will vary depending on the input size. This prevents the use of fully connected layers directly after a convolution
247 layer as it is often the case with CNNs. Instead, we use fully-connected layers only after operations independent of the
248 input size and with a fixed output size, namely mean functions over the column and row dimensions (Figure 2).

249 Overall, we designed an architecture accounting for invariance and adaptive specificities by stacking multiple equivariant
250 blocks (Figure 2, label B). An equivariant block consists in one convolution layer with filters of size 1×3 that are

251 equivariant (B1), averages of the convolution outputs across the haplotype axis (B2) and the SNP axis (B3) that are both
252 invariant, a concatenation of the equivariant and invariant features (B4), one max pooling layer that is also adaptive to
253 the number of SNPs (B5) and one fully-connected layer that updates the demographic predictions at each block (B6)
254 via a sum function (C1) (Figure 2).

255 We designed three variations of the SPIDNA permutation-invariant architecture (fully detailed in Supplementary Text).
256 The first one uses batch normalization, after each convolution layer, and therefore takes as input a fixed number of
257 400 SNPs, similarly to two of the baselines. The second one is invariant to the number of SNPs and uses instance
258 normalization, after each convolution layer, to normalize layer inputs per-data instead of per-batch (for the batch
259 normalization). The last variation is also invariant to the number of SNPs, but uses two separate instance normalization
260 steps, as well as the correlation control parameter α . The first variation using batch normalization has 110584 trainable
261 parameters and the other two using instance normalization have 110384.

262 **2.4 Hyperparameter optimization**

263 Compared to other machine learning methods, ANNs have a potentially infinite amount of hyperparameters when
264 including for instance the number of layers, the number of neurons in each of them, the learning rate, weight decay
265 or the batch size. Moreover, a run over a full dataset with enough epochs to reach convergence is time consuming
266 for networks with a complex architecture defined by many learnable parameters. Therefore, the development of deep
267 learning architectures often relies on the experience and intuition of the practitioner in a try-and-repeat process. Grid
268 search and random search are two strategies for exploring the hyperparameter space uniformly. They are commonly
269 used but are limited by the computing resources available. In our study, we used HpBandSter, a package that implements
270 the HyperBand (Li et al., 2016) algorithm to run many hyperparameter trials on a smaller resource budget (i.e. few
271 epochs) and runs the most promising trials on a greater budget. Combined with BOHB (Falkner et al., 2018), a Bayesian
272 optimisation procedure that models the expected improvement of the joint hyperparameters, this method provides more
273 guided and faster search of the hyperparameter space. At each step, BOHB draws a new combination of hyperparameter
274 values to be tested according to the expected improvement and to a predefined prior. Here, we performed a search in
275 a 5-dimensional space defined by uniform priors over the type of architecture (architectures from our baselines and
276 variations of SPIDNA architecture, based on 400 SNPs or the full number of SNPs), the learning rate, the weight decay
277 and the batch size. For SPIDNA architectures that controlled correlation, we added the control parameter α to the
278 Bayesian optimization procedure with a log-uniform prior between 0.5 and 1. The search was performed for 3 budget
279 steps and replicated 5 times, leading to a total of 83 successfully trained networks.

280 As the training time of the MLP using summary statistics was short, we optimized its hyperparameters with a random
281 search by drawing 27 configurations from uniform distributions and trained a network for each configuration during 6
282 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
283 between $5 \cdot 10^{-5}$ and $1 \cdot 10^{-2}$.

284 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
285 independently (4 correction methods \times 3 types of inputs: predefined summary statistics, SPIDNA outputs or both).

286 **2.5 Cattle breed data**

287 We inferred the demographic history of Angus, Fleckvieh and Holstein cattle breeds using the data set of 25 sequenced
288 individuals from the 1,000 genome bull project (Daetwyler et al., 2014) that was analysed by (Boitard et al., 2016b). As
289 the data of real cattle sequence is prone to phasing and sequencing errors, we converted the real data from haplotype
290 to genotype with a minimum allele frequency (maf) of 0.2, as suggested by Boitard et al. (2016b) and applied the
291 same treatment to the simulated training set. We split the real data of each breed into 2Mb and removed segments
292 comprising centromeres leaving 1,213 segments. We obtained a similar number of SNPs for the three breeds: Angus
293 (average: 4,536 SNPs, maximum: 22,391 and minimum: 775), Fleckvieh (average: 4,837 SNPs, maximum: 24,896 and
294 minimum: 896) and Fleckvieh (average: 4,732 SNPs, maximum, 24,098 and minimum: 1,212). Then we trained ABC,
295 SPIDNA and a combination of both with the best hyperparameter configurations on the modified simulated data and
296 performed the inference. The best version of SPIDNA without ABC is non-adaptive and therefore uses 400 SNPs from
297 each segment which represents 10% of the total number of SNPs in the cattle data and 67% for the training dataset.

298 All computational resources used for this study are described in the Supplementary Text.

299 **3 Results**

300 **3.1 Hyperparameter optimization**

301 The configuration with the lowest loss generated by the hyperparameter optimization procedure used 400 SNPs with
302 SPIDNA, 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
303 (Figure S1). Configurations with large batch sizes tended to have lower losses (Figure S1), which is expected as large
304 batches provide a better approximation of the full training set gradient. However, a batch size too close to the training
305 set size can lead to overfitting the training set. Here, we did not observe overfitting for any run when monitoring training
306 and validation losses. The best configurations also tended to have low learning rates and weight decays (Figure S1).
307 These low values slow down the convergence, but usually decrease the final prediction error if the budget (i.e. number
308 of training epochs) is high enough for the network to reach convergence.

309 **3.2 Comparison of the optimized architectures**

310 For each architecture, we selected the best configuration obtained with the hyperparameter optimization procedure
311 and trained it for a greater budget (i.e. 10 epochs), allowing an in-depth comparison. We found no strong decrease of
312 prediction errors after this longer training compared to their counterparts with a 10^7 budget (10^7 training SNP matrix,
313 i.e. 5.57 epochs) (Figures 3 and S1). Prediction errors for the validation set (used in the hyperparameter optimization

314 procedure) and the test set are shown in the table S2. In the following paragraph, each method is designated along its
315 index in the table S2.

316 We first compared the optimized neural networks to optimized ABC approaches based on predefined summary statistics.
317 The prediction errors achieved by ABC using summary statistics ranged from 0.496 (index 0, ABC rejection, i.e.
318 without correction) to 0.364 (ABC neural networks, index 3). The MLP network based on summary statistics performed
319 worse than ABC with correction (0.437, index 4). Moreover, MLP based on raw data performed very poorly (0.675,
320 index 5) and all other networks based on raw data outperformed this MLP. Most of them (all except SPIDNA instance
321 normalization on 400 SNPs, 0.641 and 0.599, index 12 and 14) outperformed the ABC rejection (0.454 and 0.469,
322 index 11 and 15) or led to similar errors (0.489, index 13). The *Flagel* CNNs adapted from Flagel et al. (2018) that
323 were not using dropout had average test losses of 0.541 and 0.444 (index 7 and 8). The two other *Flagel* networks
324 achieved prediction errors similar to SPIDNA (network based on the first 400 SNPs: 0.609, index 9; network based on
325 1784 downsampled SNPs: 0.484, index 10), however they had 8 to 34 times more learnable parameters than SPIDNA.
326 Lastly, we evaluated two methods that combine deep learning and ABC, by considering the features automatically
327 computed by a network as summary statistics for ABC (Jiang et al., 2017). When using only the predictions of SPIDNA
328 as input to ABC with correction (linear regression, ridge regression or neural network), we improved greatly SPIDNA's
329 performance and obtained errors similar to the ABC based on predefined summary statistics (0.369 compared to 0.364,
330 index 21 and 3). When using both SPIDNA predictions and predefined summary statistics as input to the ABC algorithm
331 we decreased further the prediction errors (0.347, index 29).

332 **3.3 Reconstruction of specific demographic histories using SPIDNA and SPIDNA+ABC**

333 We further illustrated the performance of SPIDNA on a subset of demographic scenarios that were previously investigated
334 (Boitard et al., 2016b) (Figure 4). We simulated six scenarios: "Medium", "Large", "Decline", "Expansion", "Bottleneck"
335 and "Zigzag" the same way as the neutral simulations by specifying the demographic parameters instead of drawing
336 them from a prior. The method correctly reconstructed histories of constant size, expansion and decline, as SPIDNA
337 predictions from 100 independent genomic regions (black boxplots) approximately followed the real population size
338 trend and magnitude. The true parameters were always included in the 90% credible intervals (light blue envelopes)
339 predicted by SPIDNA combined with ABC without predefined summary statistics and, for most cases, in the 50%
340 credible intervals (dark blue). Both methods also correctly reconstructed a complex history encompassing an expansion
341 interrupted by a bottleneck and followed by a constant size (see Figure 4 'Bottleneck'). However, they were unable to
342 correctly estimate the parameters of a very complex 'Zigzag' history except for its initial growth period and instead
343 reconstructed a smoother history with values intermediate to the lower and higher population sizes (see Figure 4
344 'Zigzag'). This confirmed the smoothing behavior identified previously for ABC and MSMC on these demographic
345 scenarios (Boitard et al., 2016b). Finally, similarly to ABC on predefined summary statistics (Boitard et al., 2016b),
346 SPIDNA predictions of very recent population sizes were slightly biased toward the center of the prior distribution,
347 however combining SPIDNA with ABC tended to correct this bias in most cases.

348 **3.4 Impact of positive selection on SPIDNA and ABC inference**

349 We investigated the impact of positive selection on SPIDNA and ABC inference for three illustrative demographic
350 cases (scenarios Medium, Decline and Expansion of Figure 4). Because including selection required a change in the
351 genetic simulator (msms instead of msprime), we first ensured that the change of tool to generate the new test dataset
352 had no influence on the prediction accuracy (Figure S2). We then simulated 2Mb regions including a central SNP under
353 positive selection, with varying selection strength, starting time and frequency of the beneficial allele at this time (100
354 regions for each scenario). We chose a conservative approach in which all 100 regions are under selection (worst case
355 scenario). For each scenario we predicted the population size history using SPIDNA (batch normalization) or ABC
356 (with local linear correction) on summary statistics. Both ABC and SPIDNA predictive errors varied with the selection
357 coefficient (Figure S3). On average a moderate selective pressure (100-400) did not decrease the performance (Figure
358 S3 top row). ABC inference for declining population datasets was the only one negatively impacted (increased error
359 for $s=200$ and 400). In fact, in multiple cases increasing s decreased the prediction error mean. Very strong selection
360 ($s = 800$) on the other hand led to an increased prediction error mean in all cases except for the declining histories
361 inferred by SPIDNA. In addition, the 95% quantile and standard deviations of the prediction errors tend to increase with
362 s (Figure S3) indicating that the prediction should be taken more cautiously in the presence of strong positive selection.
363 This variance was systematically smaller for SPIDNA than ABC. In particular, a handful of histories reconstructed with
364 ABC were far off while SPIDNA prediction errors remained comparatively low for all scenarios (Figure S4).

365 **3.5 SPIDNA infers the decline of effective population size of cattle**

366 We inferred the effective population size history of three breeds of cattle (Angus, Fleckvieh and Holstein) based on
367 the same 75 individuals studied by Boitard et al. (2016b) and sampled by the 1,000 Bull Genomes Project (Figure
368 5). The best ABC and SPIDNA configurations both infer a large ancestral effective population size and a decline
369 for the past 70,000 years. However, SPIDNA reports higher recent population sizes (Angus:11,334, Holstein:12,311,
370 Fleckvieh:13,579) than ABC (Angus:344, Holstein:389, Fleckvieh:1,436). Interestingly, SPIDNA infers the same
371 population sizes for all three breeds before 10 thousand years ago. This is in agreement with the estimation of the
372 beginning of the domestication (Zeder, 2008). Posterior point estimates obtained by SPIDNA combined with ABC
373 also indicated a decline after domestication, but with larger population sizes for the last 30,000 years than SPIDNA
374 alone and fairly large credible intervals at recent times (Figure S6). Angus had the largest recent population size and
375 Fleckvieh the smallest in contrary to the two previous methods. Credible intervals of ABC based on SPIDNA outputs
376 overlapped SPIDNA predictions except for the most ancient time window. On the contrary, credible intervals of ABC
377 based on summary statistics overlap SPIDNA predictions except for the most recent time windows (Figure S6). Finally
378 SPIDNA combined with ABC identified an episode of smooth decline and recovery of the population size preceding
379 the domestication (between 400,000 and 30,000 years ago). ABC on summary statistics did not infer this ancient
380 change (this study and Boitard et al. (2016b)), however Boitard et al. (2016a) also estimated that 123,465 years ago the
381 ancestral population size increased from 73,042 to 137,775 using fastsimcoal2 (Excoffier et al., 2013).

382 **4 Discussion**

383 In this paper, we introduced a deep learning approach to infer the detailed size history of a single population directly
384 from genomic data given an unknown recombination rate. This consisted in inferring jointly 21 population size
385 parameters. We not only increased the complexity of the demographic model with respect to previous works such
386 as Fligel et al. (2018), but also compared the performance of our architecture to other methods including ABC, and
387 applied our approach to real data sets. We found that our approach compared competitively with one of the best to date
388 approaches, with the added advantage of not relying on summary statistics. A robustness analysis based on a subset of
389 demographic scenarios also indicated that SPIDNA might be more robust than ABC to the presence of positive selection
390 in the data. Finally, we reconstructed the effective population size fluctuations of three cattle breeds and confirmed that
391 they all had similar sizes when they were part of the same ancestral species *Bos taurus* and underwent a decline likely
392 linked to their domestication, although the estimated strength of this decline depended on the inference method.

393 **4.1 On the practicability and importance of architecture design**

394 When applying deep learning techniques, the design of the neural network architecture is critical, as poor design can lead
395 to a lack of expressive power, information loss, underfitting, overfitting, or unnecessary complications that slow down
396 the training process. The recent history of successes in Computer Vision consists in architecture improvements, leading
397 to performance jumps (e.g. from MLP to LeNet, AlexNet, VGG, Inception and ResNet (He et al., 2016, Krizhevsky
398 et al., 2012, LeCun et al., 1998, Simonyan and Zisserman, 2014, Szegedy et al., 2017)). But these successes have been
399 built incrementally by relatively small changes over the last years, involving a large number of studies, researchers, tasks
400 and tested architectures. Therefore, automating architecture and hyperparameter choice is an important challenge that
401 can yield benefit to smaller fields such as population genetics. In our study, the Bayesian hyperparameter optimisation
402 procedure allowed us to test multiple networks thanks to a better usage of the computational power available by giving
403 more budget to the most promising ANN architectures and hyperparameters. This procedure could be extended to
404 hyperparameters that further describe the architecture of the network such as the number and type of layers, number
405 and type of neurons, the type of non-linearity or the topology. Thanks to this procedure we investigated a series of
406 architectures, starting from the simple multi-layer fully-connected network (MLP) and moving on to more complex
407 architectures, and exhibited the link between design and performance.

408 To interpret the results and compare them, let us first note that in Figure 3, a 0 error means perfect prediction, while an
409 error of 1 means that no information is extracted from the input. Indeed, a function outputting always the same value,
410 for all samples, can at best predict the average target value over the dataset, in which case the error is the standard
411 deviation over the dataset of the value to predict, which is normalized to 1 in our setup.

412 Processing the SNP and distance matrices with a MLP led to high prediction errors, especially for recent population
413 sizes. This is not surprising, since genomic information is encoded as a simple list of values, where the order has no
414 meaning from the MLP point of view, which then cannot exploit information given by the data structure. In summary,

415 an MLP configuration has several drawbacks: (i) the number of network parameters to estimate is high; (ii) the MLP
416 can only retrieve the geometry of the data through training, with no guarantee that it will learn the spatial structure of
417 the genome (i.e. the column order and distance between SNPs) or distinguish from which individual comes each SNP.
418 In spite of all these hindrances, the MLP still performed far better than random guesses or constant prediction (32%
419 better).

420 On the contrary, CNN layers process input elements by groups, allowing close SNPs to be processed together. This
421 feature, combined with the stacking of layers in CNNs, helps the network to construct features dependent on the SNPs
422 proximity. Important summary statistics used in ABC or other inference methods such as linkage disequilibrium can
423 potentially be easily expressed by such CNN. Hence we proposed several novel convolutional architectures, tailored to
424 genetic data. We first developed a *custom* CNN with 2D filters that could have different shapes, i.e. mixed kernel sizes
425 but also non-symmetrical masks. There is indeed no rational behind considering square masks only as is usually done
426 in computer vision to describe pixel neighborhoods, as rows and columns in our case correspond to different entities
427 (individual or phased haplotype versus markers). Using varied mask shapes (e.g., 2×2 , 5×4 or 3×8) helps our *custom*
428 CNN to learn features of various patterns, potentially mimicking different types of summary statistics (“vertical” masks
429 integrate over individuals, enabling the computation of allele frequencies at a SNP, while “horizontal” ones integrate
430 over SNPs, as IBS or IBD sharing tract length does). Such mixed size filters have proven useful in the Computer Vision
431 literature also, under the name of Inception architectures (Szegedy et al., 2017); they allow the extraction of a mixture
432 of different kinds of information from multiple scales within the same layer. The large gap in performance between a
433 simple MLP and this *custom* CNN confirms the importance of such considerations. A natural extension would be to
434 integrate this feature into SPIDNA, our permutation-invariant architecture.

435 **4.2 Novel architectures tailored to genomic data**

436 **4.2.1 Invariance to haplotype permutation**

437 The order in which simulated haplotypes are arranged in a SNP matrix has no meaning. Although the *custom* CNN
438 network above cannot be guaranteed to be exactly invariant to the haplotype order, it can approximately learn this data
439 property. To avoid wasting training time to learn that there is no information in the row order, it has been proposed to
440 systematically sort the haplotypes according to a predefined rule (Flagel et al., 2018, Torada et al., 2019). Because
441 there is no ordering in high dimensional space that is stable with respect to perturbations (Qi et al., 2017), we chose yet
442 another alternative and enforced our network to be permutation-invariant by design. Permutation-invariant networks,
443 or exchangeable networks, were successfully applied in population genetics by Chan et al. (2018) for inferring local
444 recombination, but our architecture is different in that the invariant operations are performed at each block, enabling
445 both individual equivariant features and global invariant features to contribute to the next layer. It has been proven
446 that this type of architecture provides universal approximation of permutation-invariant functions (Lucas et al., 2018,
447 Zaheer et al., 2017). Here we applied the methodology from Lucas et al. (2018) by using the mean as our invariant
448 operation, but we encourage developers to experiment with other invariant functions such as moments of higher

449 order. Among our permutation-invariant architectures, the best one (SPIDNA using batch normalization) had a smaller
450 prediction error than our *custom* CNN. However, it is not clear whether this improvement is directly linked to its built-in
451 permutation-invariance property, or to other differences between the two networks. Controlling the speed to invariance
452 thanks to the parameter α improved the performance of the instance normalization SPIDNA, but not significantly the
453 performance of the instance normalization adaptive SPIDNA (see table S2).

454 **4.2.2 Robustness to the number of individuals**

455 Importantly, SPIDNA adapts to the number of individuals, which is an advantageous property compared to many
456 methods relying on summary statistics. SPIDNA can be trained on data sets having similar or varying sample sizes, and,
457 once trained, it can be directly applied to a dataset of reasonably close sample size, but unobserved during training. We
458 provide an example of robustness in an experiment focusing on a subset of demographic scenarios (decline, growth,
459 medium or large constant size) and a wide range of sample sizes (from 10 to 150, Figure S5). SPIDNA using batch
460 normalization (trained on exactly 50 individuals) did not suffer a strong loss of accuracy when the sample sizes remained
461 in the [45,65] range. Outside of this range, the predictions were inaccurate in two cases: small sample sizes under
462 expanding and constant size scenarios, or large sample sizes under the expansion scenario. This was expected because
463 this specific network was not exposed to diverse sampling sizes during training. Given the observed variations across
464 scenarios and if the sample size is expected to vary substantially from 50, we advise the user to perform a similar
465 experiment based on her/his targeted sample size and a larger number of scenarios drawn from the prior distribution. If
466 needed, the user can then train a new SPIDNA network without any change in its architecture, either on a set containing
467 a wider range of sampling sizes or on a set matching the targeted sample sizes. To fasten the training, this network
468 could be initialized with the weights of the network optimized for the sample size 50, and fine-tuned on the new set.

469 **4.2.3 Automatic adaptation to the number of SNPs**

470 The two networks designed to be adaptive to the number of SNPs have the advantage of being applicable to genetic
471 data of any length, the opposite of networks specific to a particular number of SNPs, which transform the data with
472 padding or compression, or are retrained for different lengths, or take as input portions of larger sequences. Our two
473 SPIDNA adaptive networks show results close to the best of non-adaptive versions, though slightly worse (0.469 versus
474 0.454, see table S2), although the difference disappears when SPIDNA is combined with ABC (0.369 versus 0.372).
475 This small performance drop is likely due to differences in normalization rather than to the adaptive feature. Indeed,
476 the best non-adaptive SPIDNA uses batch normalization while the adaptive versions use instance normalization as
477 there is currently no implementation of batch normalization for batches with inputs of mixed sizes. We think that
478 adaptive architectures could greatly benefit from an optimised implementation of adaptive batch normalization or from
479 an implementation of batches with mixed data sizes. Nonetheless, SPIDNA networks with instance normalization had a
480 much better performance when using all SNPs rather than the 400 first SNPs only, which suggests that adaptability is a
481 useful feature (see table S2).

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

497 **4.3 Advantages and challenges of deep learning**

498 Alongside the ability of deep learning to automatically extract informative features from high dimensional data, artificial
499 neural networks are also very flexible. For instance, they can be used for transfer learning, that is, a network trained for
500 a specific task can be reused for another one by only modifying the last layers (e.g. a network trained for population
501 size history inference could be reused for classification between demographic scenarios) (Pan and Yang, 2009). The
502 new network will benefit from the embedding already learned for the previous task, improving error and learning time.
503 We also highlight that, as for most ABC methods, the parameters are inferred jointly, a major point as the common
504 population genetics model parameters almost never have an independent impact on shaping genetic diversity. We noted
505 that for highly fluctuating population sizes, SPIDNA estimated smooth histories. Smoothing can be seen as a good
506 byproduct and was for example achieved on purpose by SMC++ thanks to a spline regularization scheme (Terhorst
507 et al., 2017). A tentative explanation for SPIDNA’s smoothing effect while no regularizer was used is that it is easy
508 for neural networks to express smoothing filters in their last layer. As, in our task, smoothing is correlated with lower
509 prediction variance, the training of SPIDNA naturally chooses to smooth out its predictions. This could be seen as a
510 tendency to favor low variance in the bias/variance trade-off.

511 **4.3.1 Combining deep learning and Approximate Bayesian Computation to approximate the posterior** 512 **distribution**

513 We found that adding an ABC step to the deep learning approach increased its performance. This ABC step takes as
514 input the demographic parameters predicted by SPIDNA instead of the usual summary statistics. This strategy was

515 proposed by Jiang et al. (2017) who showed that a deep neural network could approximate the parameter posterior
516 means, which are desirable summary statistics for ABC. It was applied under the name of ABC-DL in two population
517 genetics studies for performing model selection, however both papers relied on the joint SFS as predefined candidate
518 summary statistics (Lorente-Galdos et al., 2019, Mondal et al., 2019). Here, we are taking advantage of both the deep
519 architecture to bypass summary statistics and the Bayesian framework to refine the prediction and approximate the
520 posterior distribution. The statistics currently processed by ABC are the average over multiple independent regions of
521 SPIDNA predicted population sizes. A natural future step would be to investigate whether combining differently these
522 regions leads to improved predictions.

523 It not yet clear why this combination decreases the prediction error. Neural networks, such as SPIDNA, learn a very
524 general mapping of the whole input space to the output demographic parameter space. On the other hand, ABC learns a
525 local relationship, the posterior distribution of the demographic parameters, for each targeted/observed example based
526 on its neighbourhood in the input space. Combining ABC with SPIDNA thus adds a local inference step to the general
527 mapping learnt by SPIDNA, and this might help readjust the predictions locally. This is illustrated in Figure 4 where
528 recent population sizes estimated by SPIDNA have a tendency towards the center of the prior while SPIDNA+ABC
529 corrects it. This combination might be modifying the bias/variance trade-off favored by SPIDNA towards higher
530 variance. These hypotheses could be investigated further in future works.

531 This gain however comes with a disadvantage which is the need for ABC to approximate a posterior distribution for each
532 new dataset. This can be fairly time consuming for large panels containing many populations for which demography has
533 to be reconstructed. Contrary to ABC, SPIDNA and other deep learning approaches, once trained, provide immediate
534 predictions. This amortization of the training time is relevant for all studies processing large number of datasets such as
535 meta analyses over populations or species (e.g Roux et al. (2016)) or addressing window-based tasks, such as selection
536 and introgression scans, local ancestry or recombination estimations. In these cases the parameter predictive uncertainty
537 could be estimated by the network (Chan et al., 2018, Lakshminarayanan et al., 2017) rather than through an ABC
538 procedure.

539 Finally, we showed that applying ABC to SPIDNA predictions combined with precomputed summary statistics led to an
540 error 4.7% smaller than the one of a regular ABC and 6.0% smaller than SPIDNA. This indicates that the information
541 retrieved by SPIDNA does not completely overlap the information encoded in the predefined summary statistics but is
542 not completely orthogonal either. The different behaviours of SPIDNA and ABC in terms of robustness to the presence
543 of selection also support this hypothesis. These are the first steps towards understanding and interpreting the artificial
544 neural networks currently used in population genetics, a major challenge that the deep learning field currently faces for
545 many of its applications (Gilpin et al., 2018) and that has not yet been investigated in our community.

546 **4.3.2 Application to real data**

547 Applying a method trained on simulated data to a real dataset can be a difficult task. Here we show that the estimated
548 effective population sizes of the three cattle breeds were qualitatively similar across the different methods used. All of

549 them were able to recover the large ancestral population size shared by the three breeds, followed by its decline after
550 domestication. However, the methods produced size estimates that were quantitatively different, notably in the strength
551 of the decline and the recent population sizes. For quality reasons, inference was done using genotypes pruned of low
552 frequency alleles rather than haplotypes. The architecture and hyperparameters were optimized based on simulated
553 haplotypes, and the network was trained on simulated genotypes. It is possible that an architecture designed with
554 a new hyperparameter optimization procedure calibrated for filtered genotypes would decrease SPIDNA error rate.
555 However, the discrepancy between ABC and SPIDNA reconstructions in the last 10,000 years might also be due to
556 the sensitivity of ANNs to overfitting and to misspecifications in the model generating training data. For example,
557 decrease in performance due to demographic misspecification has already been shown for selection inference based on
558 ANNs (Torada et al., 2019). In our work we investigated whether positive selection on de-novo mutation or standing
559 variation could have such a strong effect on demographic inference and found that SPIDNA was robust to various
560 selective scenarios. In the cattle case, model misspecification arises because cattle breeds are subjected to strong artificial
561 selection pressures based on observed phenotypes, with few males contributing to the next generations, which is an
562 extreme case of selection and a clear violation of the coalescent assumptions underlying our training simulated set. In
563 addition, errors or missing information in real data were not modelled in the training set, a procedure that can improve
564 ABC performance when using multiple summary statistics such as haplotype length statistics (Jay et al., 2019). When
565 comparing performance on training and validation sets, we found that our architectures were not overfitting. Yet it is
566 possible that the features automatically constructed by ANNs are more sensitive to a gap between real and simulated
567 data (e.g. unmodelled errors and artificial selection) than an ABC method based on SFS and LD statistics. Although
568 we checked the robustness of SPIDNA to the simulator tool and to multiple cases of natural selection on haplotype
569 data (Figures S2 and S3), artificial selection based on phenotype and pedigree information is yet another type of model
570 violation. Systematically testing and improving the robustness of ANNs trained on simulations is a great challenge for
571 the coming years.

572 **Conclusion**

573 We addressed a challenging task in population genetics, that is, reconstructing effective population size through
574 time. We showed that this demographic inference could be done for unknown recombination rates. The approach
575 combining SPIDNA and summary statistics has a slight increase in performance compared to the more classical
576 method (ABC based on summary statistics), while the approach based on SPIDNA performs similarly without requiring
577 any expert knowledge regarding the computation of summary statistics. Besides, the combination with an ABC
578 approach (without predefined statistics) allows to obtain posterior distributions. We are confident that a network
579 exchangeable and adaptive to the input size is a promising architecture for future lines of work for other population
580 genetics tasks, as it could prevent premature loss of information and favor learning new features rather than known
581 haplotype invariance. These new features can be seen as automatically learned summary statistics and will be crucial in
582 areas where inference is challenging and for which researchers are always designing novel and hopefully expressive

583 summary statistics (see for example the recent line of research on adaptive introgression (Racimo et al., 2015)). As for
584 now, co-estimating multiple processes remains a hard task, and inference is mostly done under simplifying assumptions,
585 e.g. selection or recombination are inferred under a fixed demographic scenario and step-wise population size is
586 reconstructed for a single panmictic population. The success of ABC and simulation-based methods is partly due
587 to their ability to include complex models via simulations. Here we showed, for the first time, that a well designed
588 artificial neural network is capable of retrieving information about fluctuating effective population size, competes
589 favorably with a commonly used approach, and can also be combined with existing summary statistics if needed.
590 Additionally, recent studies showed that artificial neural networks could detect introgression and selection (Flagel
591 et al., 2018, Torada et al., 2019). For the above reasons, and because extracting information automatically should lead
592 to the identification of features that disentangle processes hardly distinguishable, we are hopeful that future robust
593 networks trained on complex simulations could help solve some of these tasks. Finally, we provided (i) a tool for
594 users wanting to infer population size history of any species that can be applied to phased or unphased genomes
595 (available from https://gitlab.inria.fr/ml_genetics/public/dlpopsiz); (ii) new exchangeable network
596 architectures, some of which have the promising feature of being adaptive to input size ; (iii) guidelines for future
597 developers on building architectures and hyper-optimization to facilitate the development of new artificial neural
598 networks for population genomics.

599 **Acknowledgments**

600 We are grateful to the genotoul bioinformatics platform Toulouse Midi-Pyrenees (Bioinfo Genotoul) and the TAU team
601 for providing computing and storage resources, to the Paris-Saclay Center for Data Science 2.0 (IRS) for funding. JC
602 salary was funded by DIM-IHealth. We are also grateful to Simon Boitard for helpful discussions and providing the
603 cattle dataset. We thank Michael Blum for his comments on a first version of the manuscript and Diviyan Kalainathan
604 for his support with the Titanic platform. We also thank Jeffrey Spence, Lex Flagel and two anonymous reviewers for
605 their comments.

606 **Data Availability Statement**

607 The data that support the findings of this study are openly available in *dlpopsiz* at [https://gitlab.inria.fr/ml_](https://gitlab.inria.fr/ml_genetics/public/dlpopsiz)
608 [genetics/public/dlpopsiz](https://gitlab.inria.fr/ml_genetics/public/dlpopsiz).

609 **References**

- 610 Jeffrey R Adrion, Jared G Galloway, and Andrew D Kern. Inferring the landscape of recombination using recurrent
611 neural networks. *bioRxiv*, page 662247, 2019.
- 612 Simon Aeschbacher, Mark A Beaumont, and Andreas Futschik. A novel approach for choosing summary statistics in
613 approximate bayesian computation. *Genetics*, 192(3):1027–1047, 2012.

614 Babak Alipanahi, Andrew DeLong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of
615 dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831, 2015.

616 Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur.*
617 *Netw.*, 5(2):157–166, March 1994. ISSN 1045-9227. doi: 10.1109/72.279181. URL [https://doi.org/10.1109/](https://doi.org/10.1109/72.279181)
618 [72.279181](https://doi.org/10.1109/72.279181).

619 Anders Bergström, Shane A McCarthy, Ruoyun Hui, Mohamed A Almarri, Qasim Ayub, Petr Danecek, Yuan Chen,
620 Sabine Felkel, Pille Hallast, Jack Kamm, et al. Insights into human genetic variation and population history from 929
621 diverse genomes. *bioRxiv*, page 674986, 2019.

622 Anand Bhaskar, YX Rachel Wang, and Yun S Song. Efficient inference of population size histories and locus-specific
623 mutation rates from large-sample genomic variation data. *Genome research*, 25(2):268–279, 2015.

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

626 Michael GB Blum, Maria Antonieta Nunes, Dennis Prangle, Scott A Sisson, et al. A comparative review of dimension
627 reduction methods in approximate bayesian computation. *Statistical Science*, 28(2):189–208, 2013.

628 Simon Boitard, Mekki Boussaha, Aurélien Capitan, Dominique Rocha, and Bertrand Servin. Uncovering adaptation
629 from sequence data: lessons from genome resequencing of four cattle breeds. *Genetics*, 203(1):433–450, 2016a.

630 Simon Boitard, Willy Rodriguez, Flora Jay, Stefano Mona, and Frédéric Austerlitz. Inferring population size history
631 from large samples of genome-wide molecular data-an approximate bayesian computation approach. *PLoS genetics*,
632 12(3):e1005877, 2016b.

633 Michael Bridges, Elizabeth A Heron, Colm O’Dushlaine, Ricardo Segurado, Derek Morris, Aiden Corvin, Michael Gill,
634 Carlos Pinto, International Schizophrenia Consortium, et al. Genetic classification of populations using supervised
635 learning. *PloS one*, 6(5), 2011.

636 Jeffrey Chan, Valerio Perrone, Jeffrey Spence, Paul Jenkins, Sara Mathieson, and Yun Song. A likelihood-free inference
637 framework for population genetic data using exchangeable neural networks. In *Advances in Neural Information*
638 *Processing Systems*, pages 8594–8605, 2018.

639 1000 Genomes Project Consortium et al. A map of human genome variation from population-scale sequencing. *Nature*,
640 467(7319):1061, 2010.

641 1000 Genomes Project Consortium et al. A global reference for human genetic variation. *Nature*, 526(7571):68, 2015.

642 Katalin Csilléry, Olivier François, and Michael GB Blum. abc: an r package for approximate bayesian computation
643 (abc). *Methods in ecology and evolution*, 3(3):475–479, 2012.

644 Hans D Daetwyler, Aurélien Capitan, Hubert Pausch, Paul Stothard, Rianne Van Binsbergen, Rasmus F Brøndum,
645 Xiaoping Liao, Anis Djari, Sabrina C Rodriguez, Cécile Grohs, et al. Whole-genome sequencing of 234 bulls
646 facilitates mapping of monogenic and complex traits in cattle. *Nature genetics*, 46(8):858, 2014.

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

649 Laurent Excoffier, Isabelle Dupanloup, Emilia Huerta-Sánchez, Vitor C Sousa, and Matthieu Foll. Robust demographic
650 inference from genomic and snp data. *PLoS genetics*, 9(10):e1003905, 2013.

651 Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In
652 Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*,
653 volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446, Stockholmsmässan, Stockholm Sweden,
654 10–15 Jul 2018. PMLR. URL <http://proceedings.mlr.press/v80/falkner18a.html>.

655 Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate bayesian computation: semi-
656 automatic approximate bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Method-*
657 *ology)*, 74(3):419–474, 2012.

658 Lex Flagel, Yaniv Brandvain, and Daniel R Schrider. The unreasonable effectiveness of convolutional neural networks
659 in population genetic inference. *Molecular biology and evolution*, 36(2):220–238, 2018.

660 Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations:
661 An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and*
662 *advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

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

665 Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In
666 *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS’10)*. Society for
667 *Artificial Intelligence and Statistics*, 2010.

668 Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings*
669 *of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

670 Simon YW Ho and Beth Shapiro. Skyline-plot methods for estimating demographic history from nucleotide sequences.
671 *Molecular ecology resources*, 11(3):423–434, 2011.

672 Kishore Jaganathan, Sofia Kyriazopoulou Panagiotopoulou, Jeremy F McRae, Siavash Fazel Darbandi, David Knowles,
673 Yang I Li, Jack A Kosmicki, Juan Arbelaez, Wenwu Cui, Grace B Schwartz, et al. Predicting splicing from primary
674 sequence with deep learning. *Cell*, 176(3):535–548, 2019.

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

677 Bai Jiang, Tung-yu Wu, Charles Zheng, and Wing H Wong. Learning summary statistic for approximate bayesian
678 computation via deep neural network. *Statistica Sinica*, pages 1595–1618, 2017.

679 Jerome Kelleher, Alison M Etheridge, and Gilean McVean. Efficient coalescent simulation and genealogical analysis
680 for large sample sizes. *PLoS Comput Biol*, 12(5):1–22, 05 2016. doi: 10.1371/journal.pcbi.1004842. URL
681 <http://dx.doi.org/10.1371%2Fjournal.pcbi.1004842>.

682 Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

683 Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural
684 networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

685 Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty
686 estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

687 Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain
688 theory and neural networks*, 3361(10):1995, 1995.

689 Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document
690 recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

691 Liis Leitsalu, Toomas Haller, Tõnu Esko, Mari-Liis Tammesoo, Helene Alavere, Harold Snieder, Markus Perola,
692 Pauline C Ng, Reedik Mägi, Lili Milani, et al. Cohort profile: Estonian biobank of the estonian genome center,
693 university of tartu. *International journal of epidemiology*, 44(4):1137–1147, 2014.

694 Heng Li and Richard Durbin. Inference of human population history from individual whole-genome sequences. *Nature*,
695 475(7357):493, 2011.

696 Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel
697 bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*, 2016.

698 Xiaoming Liu and Yun-Xin Fu. Exploring population size changes using snp frequency spectra. *Nature genetics*, 47(5):
699 555, 2015.

700 Belen Lorente-Galdos, Oscar Lao, Gerard Serra-Vidal, Gabriel Santpere, Lukas FK Kuderna, Lara R Arauna, Karima
701 Fadhlouli-Zid, Ville N Pimenoff, Himla Soodyall, Pierre Zalloua, et al. Whole-genome sequence analysis of a pan
702 african set of samples reveals archaic gene flow from an extinct basal population of modern humans into sub-saharan
703 populations. *Genome biology*, 20(1):77, 2019.

704 David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60
705 (2):91–110, 2004.

706 Thomas Lucas, Corentin Tallec, Yann Ollivier, and Jakob Verbeek. Mixed batches and symmetric discriminators for
707 GAN training. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on
708 Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2844–2853, Stockholmssäsan,
709 Stockholm Sweden, 10–15 Jul 2018. PMLR. URL <http://proceedings.mlr.press/v80/lucas18a.html>.

710 Wenlong Ma, Zhixu Qiu, Jie Song, Jiajia Li, Qian Cheng, Jingjing Zhai, and Chuang Ma. A deep convolutional neural
711 network approach for predicting phenotypes from genotypes. *Planta*, 248(5):1307–1318, 2018.

712 Iona M. MacLeod, Denis M. Larkin, Harris A. Lewin, Ben J. Hayes, and Mike E. Goddard. Inferring Demography
713 from Runs of Homozygosity in Whole-Genome Sequence, with Correction for Sequence Errors. *Molecular*
714 *Biology and Evolution*, 30(9):2209–2223, 07 2013. ISSN 0737-4038. doi: 10.1093/molbev/mst125. URL <https://doi.org/10.1093/molbev/mst125>.
715

716 Swapan Mallick, Heng Li, Mark Lipson, Iain Mathieson, Melissa Gymrek, Fernando Racimo, Mengyao Zhao, Niru
717 Chennagiri, Susanne Nordenfelt, Arti Tandon, et al. The simons genome diversity project: 300 genomes from 142
718 diverse populations. *Nature*, 538(7624):201, 2016.

719 Alistair Miles, Peter Ralph, Summer Rae, and Rahul Pisupati. cggh/scikit-allel: v1.2.1, June 2019. URL <https://doi.org/10.5281/zenodo.3238280>.
720

721 Mayukh Mondal, Jaume Bertranpetit, and Oscar Lao. Approximate bayesian computation with deep learning supports
722 a third archaic introgression in asia and oceania. *Nature communications*, 10(1):246, 2019.

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

725 Miguel Navascués, Raphaël Leblois, and Concetta Burgarella. Demographic inference through approximate-bayesian-
726 computation skyline plots. *PeerJ*, 5:e3530, 2017.

727 Luca Pagani, Daniel John Lawson, Evelyn Jagoda, Alexander Mörseburg, Anders Eriksson, Mario Mitt, Florian
728 Clemente, Georgi Hudjashov, Michael DeGiorgio, Lauri Saag, et al. Genomic analyses inform on migration events
729 during the peopling of eurasia. *Nature*, 538(7624):238, 2016.

730 Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*,
731 22(10):1345–1359, 2009.

732 Javier Prado-Martinez, Peter H Sudmant, Jeffrey M Kidd, Heng Li, Joanna L Kelley, Belen Lorente-Galdos, Krishna R
733 Veeramah, August E Woerner, Timothy D O’Connor, Gabriel Santpere, et al. Great ape genetic diversity and
734 population history. *Nature*, 499(7459):471, 2013.

735 Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification
736 and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages
737 652–660, 2017.

738 Fernando Racimo, Sriram Sankararaman, Rasmus Nielsen, and Emilia Huerta-Sánchez. Evidence for archaic adaptive
739 introgression in humans. *Nature Reviews Genetics*, 16(6):359–371, 2015.

740 Louis Raynal, Jean-Michel Marin, Pierre Pudlo, Mathieu Ribatet, Christian P Robert, and Arnaud Estoup. Abc random
741 forests for bayesian parameter inference. *Bioinformatics*, 35(10):1720–1728, 2018.

742 Camille Roux, Christelle Fraise, Jonathan Romiguier, Yoann Anciaux, Nicolas Galtier, and Nicolas Bierne. Shedding
743 light on the grey zone of speciation along a continuum of genomic divergence. *PLoS biology*, 14(12), 2016.

744 David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation.
745 In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing: Explorations in the*
746 *Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA, 1986.

747 Cynthia Sandor, Wanbo Li, Wouter Coppieters, Tom Druet, Carole Charlier, and Michel Georges. Genetic variants in
748 *rec8*, *rnf212*, and *prdm9* influence male recombination in cattle. *PLoS genetics*, 8(7), 2012.

749 Daniel R Schrider, Julien Ayroles, Daniel R Matute, and Andrew D Kern. Supervised machine learning reveals
750 introgressed loci in the genomes of *Drosophila simulans* and *D. sechellia*. *PLoS genetics*, 14(4):e1007341, 2018.

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

753 Sara Sheehan, Kelley Harris, and Yun S Song. Estimating variable effective population sizes from multiple genomes: a
754 sequentially markov conditional sampling distribution approach. *Genetics*, 194(3):647–662, 2013.

755 Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*
756 *preprint arXiv:1409.1556*, 2014.

757 Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *null*, page
758 1470. IEEE, 2003.

759 Chris CR Smith and Samuel M Flaxman. Leveraging whole genome sequencing data for demographic inference with
760 approximate bayesian computation. *Molecular ecology resources*, 2019.

761 Jeffrey P Spence, Matthias Steinrücken, Jonathan Terhorst, and Yun S Song. Inference of population history using
762 coalescent hmms: review and outlook. *Current opinion in genetics & development*, 53:70–76, 2018.

763 Matthias Steinrücken, Jack Kamm, Jeffrey P Spence, and Yun S Song. Inference of complex population histories using
764 whole-genome sequences from multiple populations. *Proceedings of the National Academy of Sciences*, 116(34):
765 17115–17120, 2019.

766 Lauren Alpert Sugden, Elizabeth G Atkinson, Annie P Fischer, Stephen Rong, Brenna M Henn, and Sohini Ramachan-
767 dran. Localization of adaptive variants in human genomes using averaged one-dependence estimation. *Nature*
768 *communications*, 9(1):703, 2018.

769 Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the
770 impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

771 Jonathan Terhorst, John A Kamm, and Yun S Song. Robust and scalable inference of population history from hundreds
772 of unphased whole genomes. *Nature genetics*, 49(2):303, 2017.

773 Luis Torada, Lucrezia Lorenzon, Alice Beddis, Ulas Isildak, Linda Pattini, Sara Mathieson, and Matteo Fumagalli.
774 Imagenet: a convolutional neural network to quantify natural selection from genomic data. *BMC bioinformatics*, 20
775 (9):337, 2019.

776 Rémi Tournebize, Valérie Poncet, Mattias Jakobsson, Yves Vigouroux, and Stéphanie Manel. Mcswan: A joint site
777 frequency spectrum method to detect and date selective sweeps across multiple population genomes. *Molecular*
778 *ecology resources*, 19(1):283–295, 2019.

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

781 Alexander T Xue, Daniel R Schrider, Andrew D Kern, Ag1000G Consortium, et al. Discovery of ongoing selective
782 sweeps within anopheles mosquito populations using deep learning. *bioRxiv*, page 589069, 2019.

783 Burak Yelmen, Aurélien Decelle, Linda Ongaro, Davide Marnetto, Corentin Tallec, Francesco Montinaro, Cyril
784 Furtlehner, Luca Pagani, and Flora Jay. Creating artificial human genomes using generative models. *bioRxiv*, page
785 769091, 2019.

786 Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J
787 Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

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

790 **Figures and Tables**

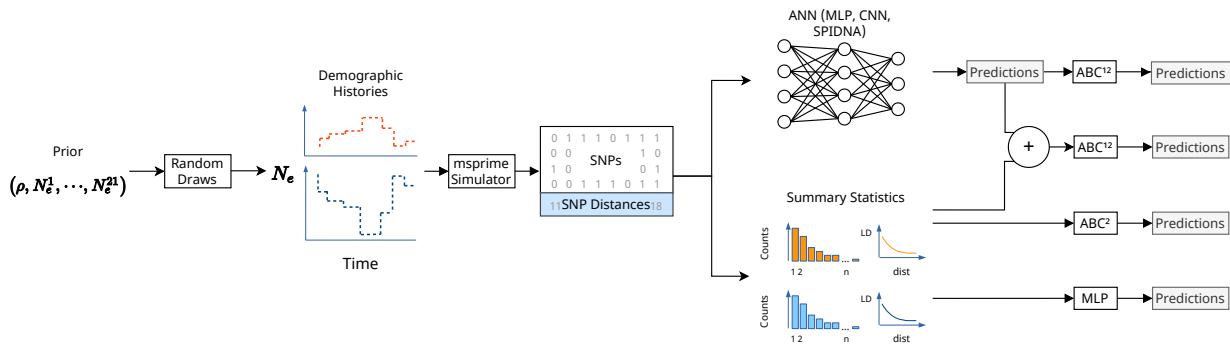


Figure 1: Overview of the methods compared in this study. Demographic histories are drawn from a prior distribution on 21 population sizes N_e^i and one recombination rate ρ , and are used to generate SNP matrices with msprime. Two types of summary statistics are computed from these simulations (SFS and LD). The predictions (outputs) made by different kind of ANNs (MLP, custom CNN and SPIDNA architecture) are compared to an MLP using the summary statistics and to ABC using either the summary statistics, SPIDNA outputs or both. ¹ ANN outputs used are the predictions made by the version of SPIDNA with the lowest prediction error. ² ABC without correction, with linear regression, ridge regression or a single layer neural network are compared.

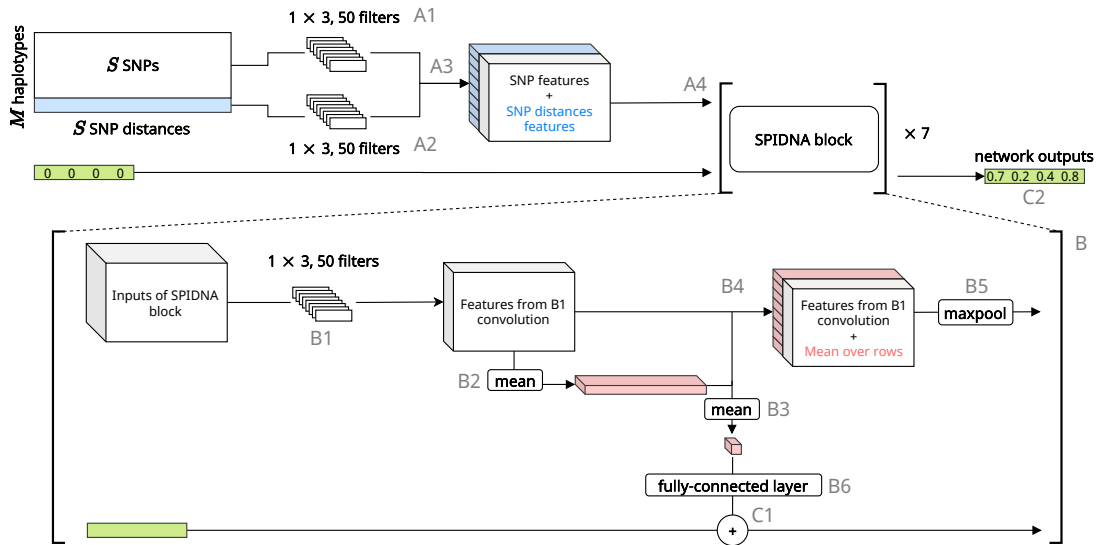


Figure 2: Schematic of SPIDNA architecture. SPIDNA takes as input a SNP matrix associated with its vector of distances between SNPs (in blue). A convolution layer is applied to the SNPs (A1) and another convolution layer is applied to the distances (A2). Results of A2 are repeated to be concatenated with results from A1 (A3). The output is passed to a series of seven SPIDNA blocks (A4). Each SPIDNA block starts with a convolution layer (B1) followed by the mean over rows of the convolution layer result (B2) and the mean over columns of B2 result (B3). The concatenation of B1 and B2 results (B4) is processed by a max pooling layer (B5) and passed to the next SPIDNA block. In parallel, the output of B3 is processed by a fully-connected layer (B6). The prediction vector (in green) is updated at each SPIDNA block with a sum (C1) of its previous value and B6 results. It is finally output by the last block as the predicted demographic parameters (C2).

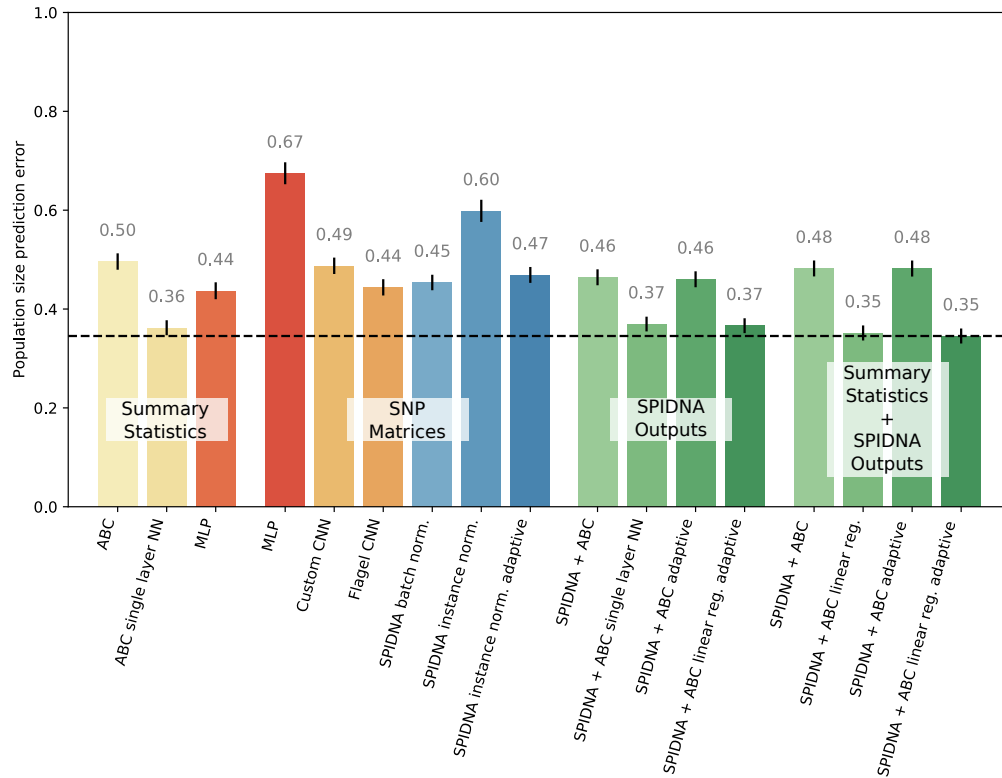


Figure 3: Prediction errors on the test set of the best run of each method after the hyperparameter optimization. The best configurations of each ANN (MLP, *custom* CNN and SPIDNA) have been retrained for 10 epochs. Traditional ABC methods are depicted in yellow, deep MLPs and CNNs in red and orange, SPIDNA ANNs in blue, 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).

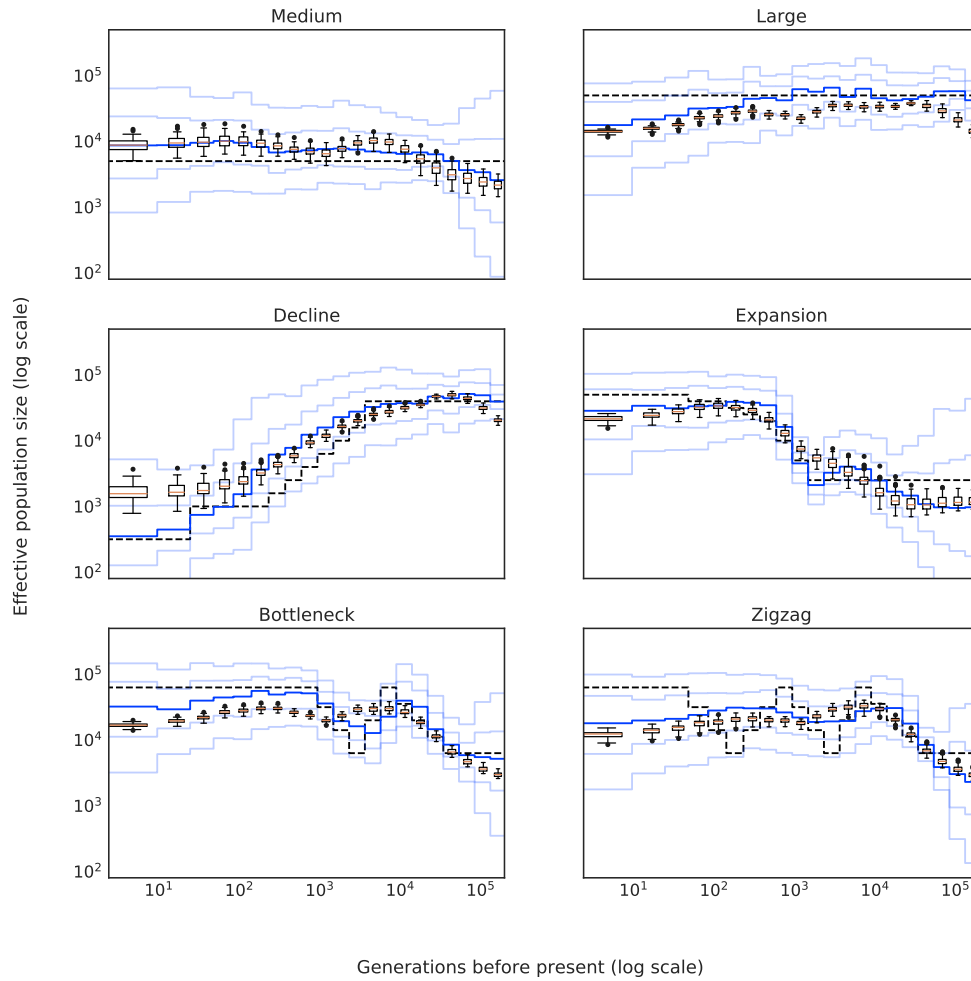


Figure 4: Predictions of SPIDNA and ABC using SPIDNA outputs, for six predefined scenarios (dashed black lines). 100 replicates were simulated for each scenario. Boxplots show the dispersion of SPIDNA predictions (over replicates). For each history inferred by SPIDNA combined with ABC, we display the posterior median (plain blue line), the 50% credible interval (dark blue) and the 90% credible interval (light blue).

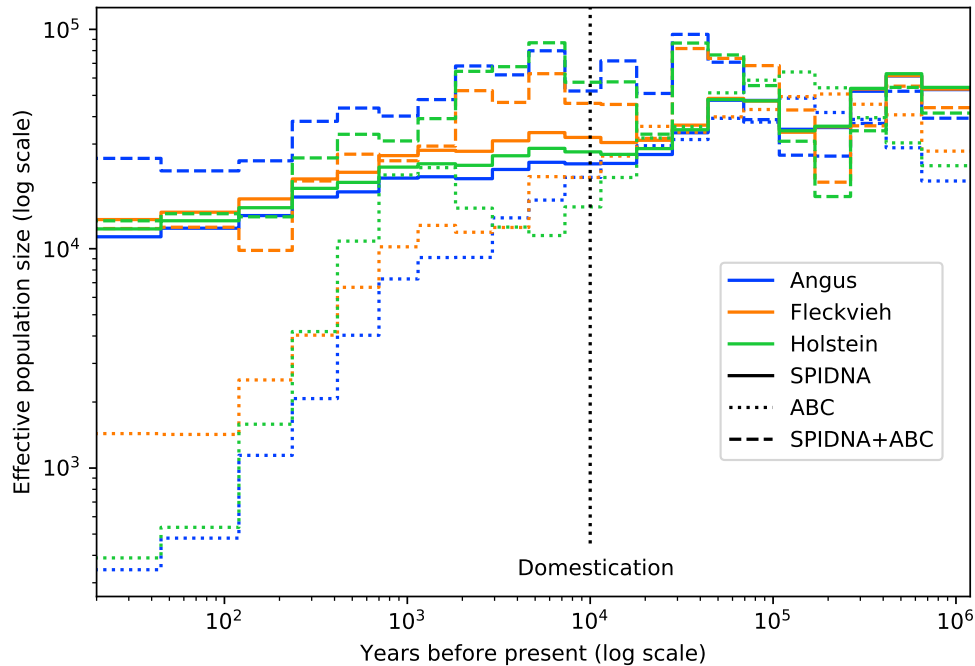


Figure 5: Effective population size of three cattle breeds inferred by ABC (dotted lines), by the best SPIDNA architecture, SPIDNA batch normalization (plain lines), and by ABC based on SPIDNA outputs (dashed lines). Domestication is estimated to have occurred 10,000 years ago (vertical dotted line).

791 **Supplementary Text**

792 **Simulation details**

793 We defined the demographic parameters by following similar rules as Boitard et al. (2016b): $I = 21$ time windows
 794 $[t_i, t_{i+1}]$ 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
 795 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
 796 capture important periods of cattle history. They could be modified to describe more precisely specific parts of the
 797 history by playing with the ratio between the length of recent versus old time windows. By increasing exponentially the
 798 time windows as we go further in the past, we obtain more detailed scenarios for recent times. Generation time for
 799 cattle are assume to be about 5 years.

800 **Dataset filtering and splitting**

801 After simulation, scenarios producing fewer than 400 SNPs in any 2Mb regions were removed. This threshold could
 802 be changed by modifying the networks or simulating longer regions. However, the real cattle dataset has on average
 803 4,357 SNPs across a 2Mb-long region, so these scenarios were far outside the plausible posterior distribution. That

804 reduced the dataset to 18,461 scenarios (i.e. 1,846,100 SNP matrices) out of the 50,000 scenarios simulated with an
805 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.
806 50,000 validation SNP matrices overall) and a training set with the remaining 17,961 scenarios (i.e. 1,796,100 training
807 SNP matrices). In order to check for hyperparameter overfitting, we have also simulated a test set from the same prior
808 distribution. Hence, we randomly drew 2,000 scenarios and kept the 767 scenarios with more than 400 SNPs which
809 gives 76,700 test SNP matrices. Training, validation and test set demographic parameters were all standardized using
810 mean and variance from the training set.

811 **SPIDNA details**

812 Except for the different normalization layers and the correlation control parameter α , the three variations of SPIDNA
813 have the same architecture represented in Figure 2. At each step i of the network, we consider that the data has four
814 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
815 dimension before the first layer), S the column dimension (also the SNP dimension before the first layer) and F the
816 feature dimension (only one feature before the first layer). A first convolution layer of $50 \ 1 \times 3$ filters is applied to
817 the SNP matrix (Main Figure 2, label A1), and another convolution layer of $50 \ 1 \times 3$ filters is applied to the vector of
818 distances between SNPs (A2) and repeated M times. The results of the two convolutions have now the same dimensions
819 and are concatenated along the feature dimension (A3). The resulting tensor is then passed to seven blocks put end to
820 end (A4), each one involving an equivariant function and an invariant function (B). The equivariant function ψ is a
821 convolutional layer of $50 \ 1 \times 3$ filters (B1) 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
822 of the equivariant function is then passed to the invariant function ρ , which is the mean over the dimension M (B2).
823 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
824 as $\phi(X_{i-1})$. Then $\rho(\phi(X_{i-1}))$ and $\phi(X_{i-1})$ are concatenated over the feature dimension (B4). Finally, max-pooling
825 filters of dimension 1×2 are applied, and the result is passed to the next block (B5). In parallel, each block computes the
826 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
827 layer with 21 outputs (B6). The predictions of each block are summed (C).

828 **From batch normalization to instance normalization**

829 Network weight initialization is a difficult task that can lead to vanishing or exploding gradient (i.e. network weight
830 error gradients are too low or high for proper training) when not carefully done and is associated with a poor learning
831 rate (Bengio et al., 1994, Glorot and Bengio, 2010). Most initialization schemes try to force the outputs of each layer to
832 follow some distribution assuming normalized input data. Batch normalization solves this problem by normalizing layer
833 outputs over the whole batch during training and computing a running mean and variance for the evaluation steps. We
834 used this type of normalization for our networks that take as input a fixed number of SNPs. For the networks invariant
835 to the number of SNPs, we could not concatenate all batch data into the same tensor because of their varying sizes.
836 Therefore, we use instance normalization, which computes both mean and variance over the feature dimension.

837 **Computational resources**

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

- 839 • 68 nodes with 2 E5-2670 v2 Intel CPUs (2.50GHz, 20 threads) and 256GB of RAM
- 840 • 48 nodes with 2 E5-2683 v4 Intel CPUs (2.10GHz, 32 threads) and 256GB of RAM.

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

- 843 • 5 nodes with 4 GTX 1080 (12GB of VRAM) GPUs, 2 E5-2650 v4 Intel CPUs (2.20GHz, 24 threads) and
844 252GB of RAM
- 845 • 7 nodes with 4 RTX 2080 (12GB of VRAM) GPUs, 2 Silver 4108 Intel CPUs (1.80GHz, 18 threads) and
846 252GB of RAM
- 847 • 1 node with 4 Tesla P100 (16GB of VRAM) GPUs, 2 E5-2690 v4 Intel CPUs (2.60GHz, 28 threads) and
848 252GB of RAM
- 849 • 1 node with 2 RTX 2080 (8GB of VRAM) GPUs, 2 E5-2650 v4 Intel CPUs (2.20GHz, 24 threads) and 252GB
850 of RAM

851 Both platforms use Slurm as job scheduling system. Batch sizes and deep learning architectures were all designed to fit
852 on less than 12GB of VRAM during training. To train non-adaptive architectures, batches were split between 3 GPUs
853 with at least 12GB of VRAM. Adaptive architectures were trained on one GPU as batch data of varying sizes could not
854 be concatenated in the same tensor. The training of SPIDNA took at most 1h42 per epoch for non-adaptive version and
855 31h31 per epoch for adaptive version. The slow computation time of adaptive SPIDNA is mostly due to data being
856 inputted one by one in the network instead of concatenated in tensors.

857 **Supplementary figures and tables**

● MLP ● Custom CNN ● SPIDNA instance norm. alpha ● SPIDNA instance norm. adaptive
● SPIDNA batch norm. ● SPIDNA instance norm. ● SPIDNA instance norm. alpha adaptive

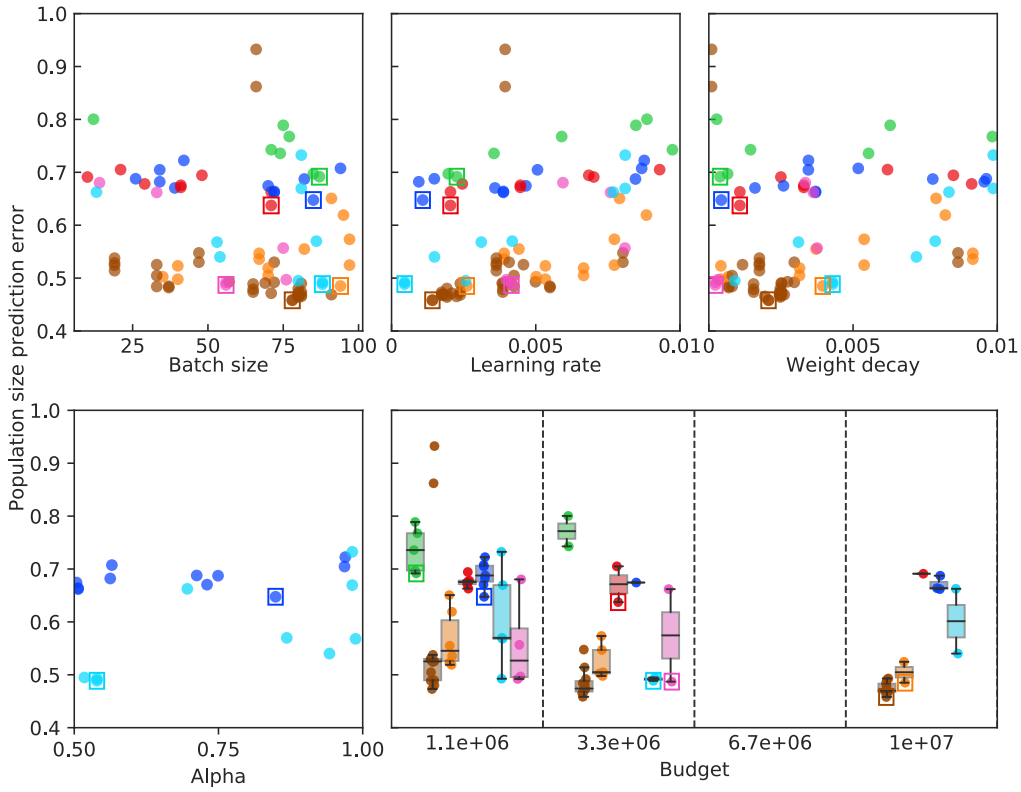


Figure S1: Population size prediction error for each run of the hyperparameter optimization procedure. X-axes indicate the hyperparameter (batch size, learning rate, weight decay and alpha) or budget values, and colors indicate the type of network used for the run (MLP, custom CNN and multiple SPIDNA architectures). For each network the best run is surrounded by a square.

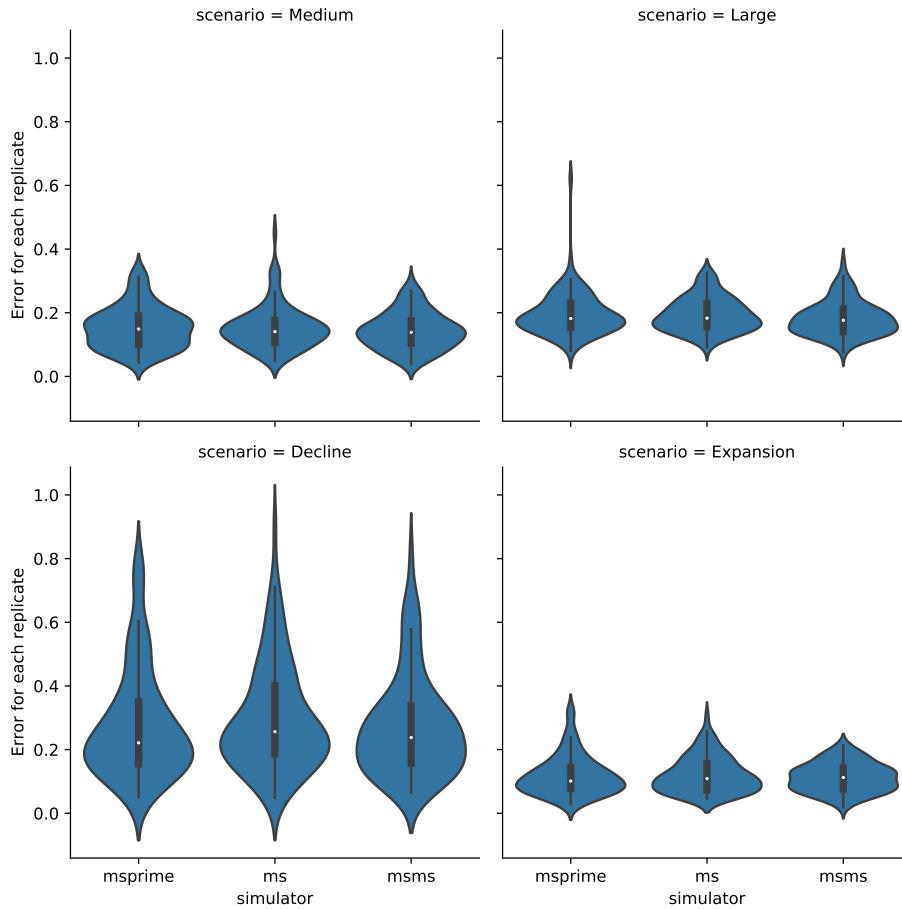


Figure S2: **Robustness to simulator tool.** Distributions of SPIDNA predictive errors per replicate (i.e. per independent genomic region) for four demographic scenarios and three different genetic simulators (msprime, ms, msms). SPIDNA batch norm. network was trained on simulated datasets generated 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.

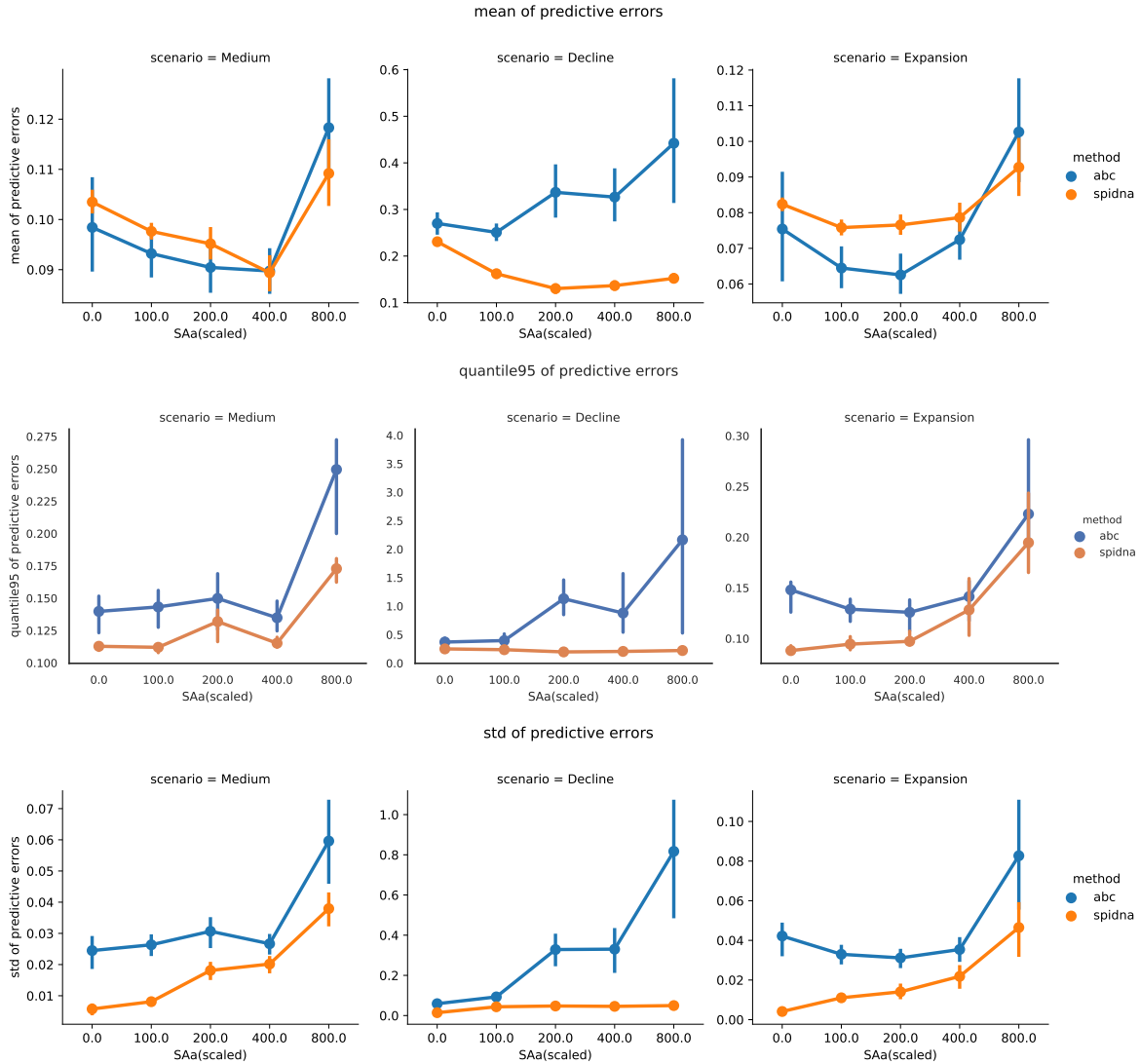


Figure S3: **Robustness to the presence of positive selection.** ABC and SPIDNA (batch norm.) predictive errors computed from 100 2Mb-long regions for three demographic scenarios (Medium constant size, Decline and Expansion) under various selective pressures (with additive fitness effect). The reference/training 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.

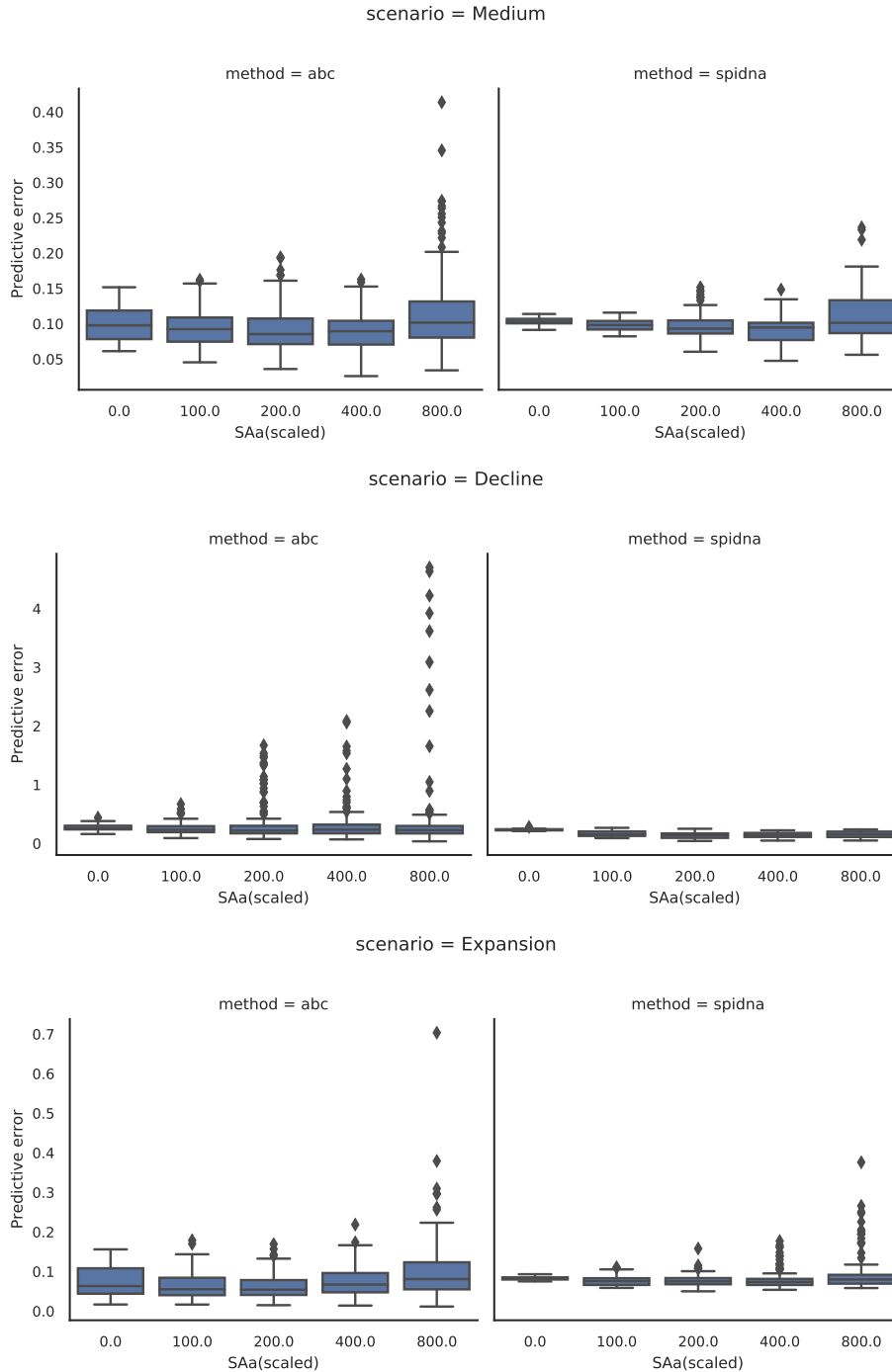


Figure S4: **Robustness to the presence of positive selection.** ABC and SPIDNA (batch norm.) predictive errors computed from 100 2Mb-long regions for three demographic scenarios (Medium constant size, Decline and Expansion) under various selective pressures. The reference/training set is the same as 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).

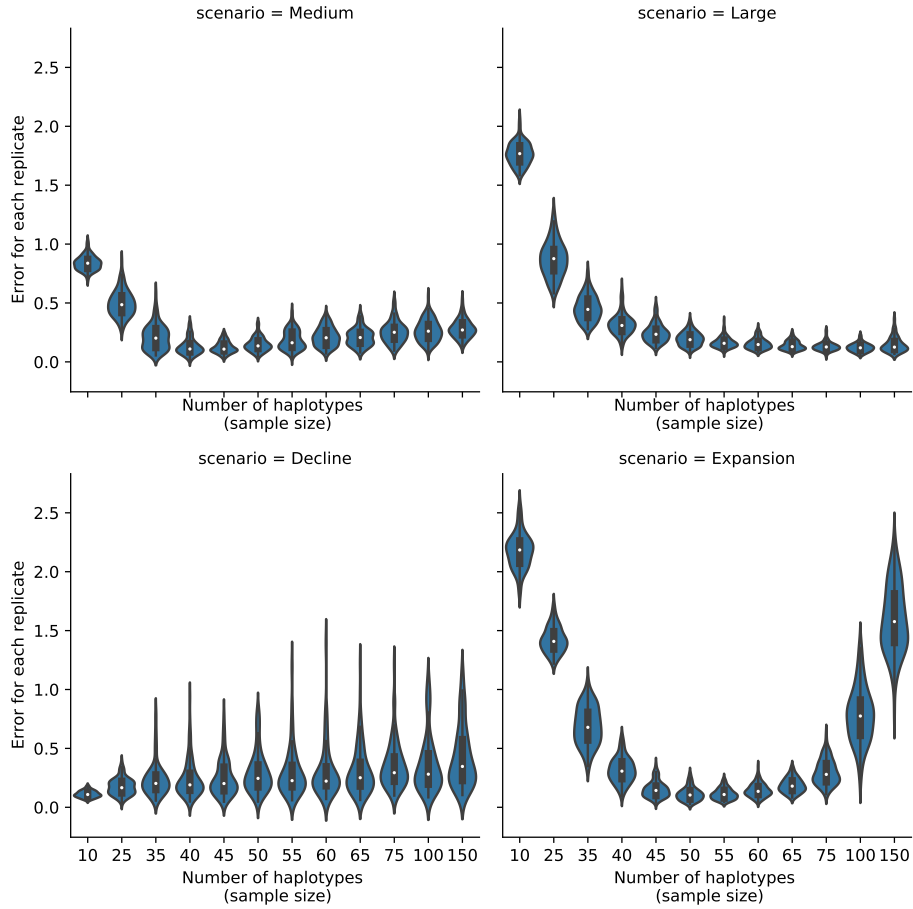


Figure S5: **Robustness to sample size.** Distributions of SPIDNA predictive errors per replicate (i.e., per independent genomic region) for four demographic scenarios and different sampling sizes. SPIDNA (batch norm.) network was trained on simulated datasets containing exactly 50 samples. The test 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.

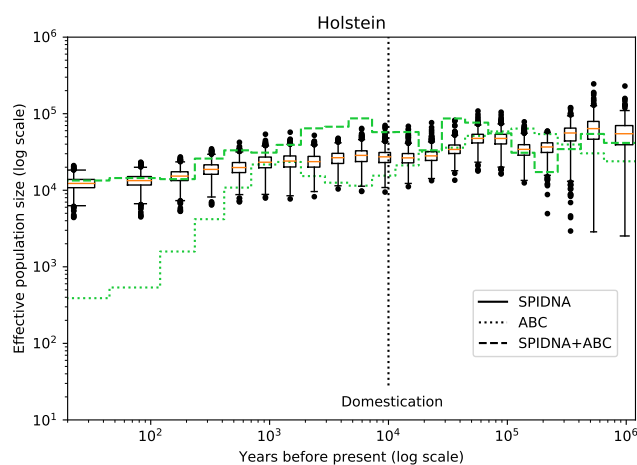
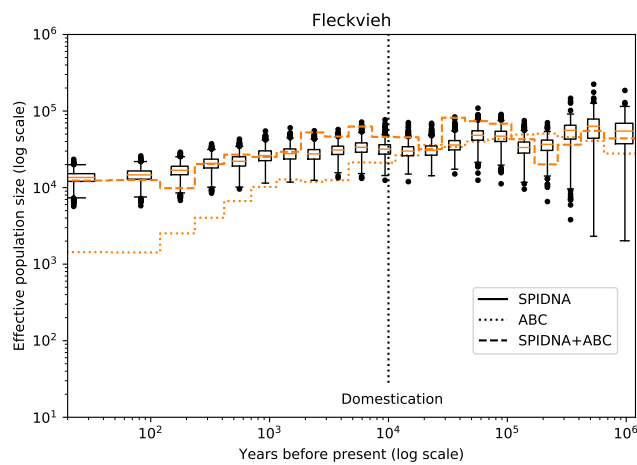
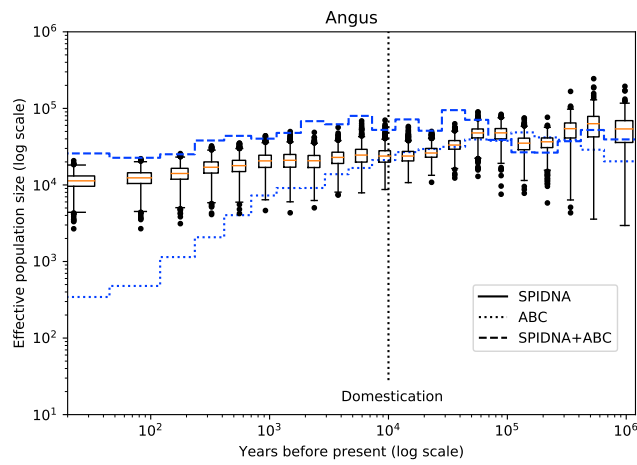


Figure S6: Effective population size of three cattle breeds inferred by the best SPIDNA architecture, SPIDNA batch normalization, by ABC (dotted lines) and by ABC based on SPIDNA outputs (dashed lines). Boxplots show the dispersion of SPIDNA predictions (over replicates). For each history inferred by 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).

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 S1: Parameters used for the *Flagel* CNN

	Method	Adaptive	Summary statistics	ABC correction	Alpha	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

Table S2: Prediction errors of the best configuration of each method after hyperparameter optimization