Haplotype and Population Structure Inference using Neural Networks in Whole-Genome Sequencing Data

Jonas Meisner & Anders Albrechtsen
Department of Biology
University of Copenhagen

December 28, 2020

Abstract

Accurate inference of population structure is important in many studies of population genetics. In this paper we present, `HaploNet`, a novel method for performing dimensionality reduction and clustering in genetic data. The method is based on local clustering of phased haplotypes using neural networks from whole-genome sequencing or genotype data. By utilizing a Gaussian mixture prior in a variational autoencoder framework, we are able to learn a low-dimensional latent space in which we cluster haplotypes along the genome in a highly scalable manner. We demonstrate that we can use encodings of the latent space to infer global population structure using principal component analysis with haplotype information. Additionally, we derive an expectation-maximization algorithm for estimating ancestry proportions based on the haplotype clustering and the neural networks in a likelihood framework. Using different examples of sequencing data, we demonstrate that our approach is better at distinguishing closely related populations than standard principal component analysis and admixture analysis. We show that `HaploNet` performs similarly to `ChromoPainter` for principal component analysis while being much faster and allowing for unsupervised clustering.

1 Introduction

Understanding population structure is a cornerstone in population and evolutionary genetics as it provides insights into demographic events and processes that have affected a population. The most common approaches for inferring population structure from genetic data are using principal component analysis (PCA) [33] and using a clustering algorithm such as `STRUCTURE` [34], or derivations thereof. PCA infers continuous axes of genetic variation that summarize the genetic relationship between samples and the clustering algorithm assigns samples to a predefined number of ancestral sources while allowing for fractional membership. The inferred axes of PCA are very useful to account for population or cryptic structure in association studies or even to simply visualize the genetic data. A limitation of PCA and clustering algorithms are that they usually assume all single-nucleotide polymorphisms (SNPs) to be independent, and they do therefore not benefit from the information of correlated sites or they may be biased thereof in their global estimates [44, 33]. `ChromoPainter` [26] employs the Li and Stephens hidden markov model [27] for haplotype sampling in order to model and utilize correlations between SNPs, by letting samples be a mosaic of each

other's haplotypes. In this way, `ChromoPainter` has become state-of-the-art for inferring fine-scale population structure.

Gaussian mixture models and k-Means are commonly used methods for performing unsupervised clustering. However, these methods suffer from the curse of dimensionality where distances will become almost equidistant between samples in high-dimensional space [50]. A popular approach to overcome the curse of dimensionality is to perform dimensionality reduction, e.g. using PCA, and then perform clustering in the low-dimensional space that still captures most of the variation in the full dataset [13]. Recently, deep autoencoders methods have been very successful for large-scale datasets as they perform dimensionality reduction and clustering either sequentially or jointly to benefit from induced non-linearity and scalability of deep learning architectures using neural networks [48, 47]. Deep autoencoders have also been introduced in generative models, e.g. variational autoencoders, where the unknown data generating distribution is learnt by introducing latent random variables, such that new samples can be generated from this distribution [25, 36].

Most studies in population genetics utilizing neural networks for parameter inference have mainly focused on supervised learning through simulations from demographic models [40, 38, 14, 15, 8]. Here, an overall demography is assumed, based on previous literature, and a lot of different datasets are simulated using small variations in model parameters, e.g. selection coefficient or recombination rate, with evolutionary simulators (e.g. `msprime` [21]). The studies are usually converting a simulated haplotype matrix into a downscaled fixed sized image with rows and/or columns sorted based on some distance measure. The network is then trained on the simulated datasets to learn the specified model parameters with feature boundaries in convolutional layers, and in the end, the model is tested on a real dataset. However, recently more studies have instead focused on deep generative models for data-driven inference or simulation using state-of-the-art unsupervised learning approaches [31, 49, 4, 46].

We here present `HaploNet`, a method for inferring haplotype and population structure using neural networks in an unsupervised approach for phased haplotypes of whole-genome sequencing (WGS) data. We utilize a variational autoencoder (VAE) framework to learn mappings to and from a low-dimensional latent space in which we will perform indirect clustering of haplotypes with a Gaussian mixture prior. Therefore, we do not have to rely on simulated training data from demographic models with a lot of user-specified parameters, but we are able to construct a fully data-driven inference framework. We demonstrate that we can use haplotype encodings from our latent space to infer fine-scale global population structure with PCA, similarly to the state-of-the-art software `ChromoPainter` while being much faster, as we model correlated SNPs in windows using non-linear layers in neural networks. However unlike `ChromoPainter`, we rely on clustering in our inferred latent space instead of pairwise distances between haplotypes. This allows us to compute likelihoods of the observed haplotypes in each window given a latent state, where each state represents an inferred haplotype cluster. From the computed likelihoods, we build a clustering model similar to NGSadmix [42] which we fit using an accelerated expectation-maximization (EM) algorithm to estimate ancestry proportions as well as frequencies of our neural network inferred haplotype clusters. We show that `HaploNet` has much higher resolution than the widely used `ADMIXTURE` software [2]. Additionally, we are also capable of simulating entire new chromosomes based on the inferred population structure and haplotype clusters with the generative properties of a VAE framework. This is done by sampling haplotypes from the most probable sequence of haplotype clusters in a population and the sampled haplotypes will still reflect properties of real samples, such as LD patterns. `HaploNet` is freely available at https://github.com/rosemeis/HaploNet.

2 Material and Methods

The method is based on phased haplotype data from diallelic markers. We define $\mathbf{X}$ as a $2N \times M$ haplotype matrix for a single chromosome, where N is the number of individuals and M is the number of SNPs along the chromosome. The entries of the matrix are encoded as either 0 or 1, referring to the major and minor allele, respectively. For unphased data, we can generalize this formulation to a standard genotype matrix where heterozygous genotypes are encoded as 0.5.

For each chromosome, we divide the sites into windows of a fixed length of L SNPs, which we assume is much smaller than M. The windows are non-overlapping and we will further assume that the parameters estimated in a window are independent from parameters estimated in adjacent windows. The length of the genomic windows can also be defined by a recombination map but we have kept it fixed in this study for the matter of generalizability and ease of application. For each defined window along a chromosome, we will independently train neural networks in a variational autoencoder framework to learn haplotype clusters and encodings using a Gaussian mixture prior. From these trained networks, we will able to calculate distances between pairs of haplotypes using their encodings, and we will additionally be able to calculate a likelihood for an observed haplotype for each haplotype cluster similarly to calculating genotype likelihoods for WGS data.

2.1 Variational autoencoder

An autoencoder is a state-of-the-art approach for performing dimensionality reduction by learning a mapping of the space of the input data, $\mathcal{X}$, to a low-dimensional space $\mathcal{Z}$ and a mapping back to the input data [37, 3]. More formally, we can describe the two mapping functions as $g : \mathcal{X} \mapsto \mathcal{Z}$ and $f : \mathcal{Z} \mapsto \mathcal{X}$, which are commonly called the encoder and the decoder, respectively. Both the encoder and the decoder are parameterized by (deep) neural networks to learn the mapping, as multilayer feed-forward neural networks are universal function approximators [17].

A probabilistic variant of this neural network architecture is used in a variational autoencoder (VAE) to learn the unknown generating process of the input data by introducing latent variables and modelling the joint probability through variational inference. As an optimization method, variational inference is often used to approximate the posterior distribution of a latent variable $\mathbf{z}$, $p(\mathbf{z} \,|\, \mathbf{x})$, by fitting a function that describes a chosen family of distributions, $q_\phi(\mathbf{z} \,|\, \mathbf{x})$. Thus, variational inference turns it into an optimization problem, where the objective is to maximize the evidence lower bound (ELBO) of the marginal log-likelihood of the data using an iterative procedure in contrast to Monte Carlo Markov Chain methods that approximate it by sampling from the posterior distribution [6]. The function approximating the posterior distribution is parameterized with variational parameters ϕ. Kingma and Welling introduced the Stochastic Gradient Variational Bayes (SGVB) estimator [25] of the ELBO for approximate posterior inference in a VAE framework (as well as [36]), where a set of parameters, (θ, ϕ), are optimized with amortized inference using mappings parameterized by neural networks. Here, the marginal log-likelihood, $p_\theta(\mathbf{x})$, is parameterized with parameters θ. This amortization means that the number of parameters does not depend on sample size as in traditional variational inference, but it only depends on the network size [41]. The VAE can be seen as an autoencoder with its latent space being regularized by a chosen prior distribution to make the inferred latent space more interpretable and prevent overfitting. A standard Gaussian prior is the most common choice, however, it is often too simple and a lot of effort has been made to make the approximate posterior richer with normalizing flows and additional stochastic layers [35, 24, 43].

In our proposed method, we will construct a generative model and use a variational autoencoder framework to learn low-dimensional encodings of haplotypes in windows along the genome. However, we also introduce an additional categorical variable y to represent haplotype clusters such that we enforce a Gaussian mixture prior to increase the flexibility of our approximate posterior distribution. In this way, we are able to jointly perform dimensionality reduction and clustering of the haplotypes in a highly scalable approach. In the following model descriptions, we will follow the mathematical notation used in the machine learning literature and Kingma and Welling (2013) [25], where p_θ and q_ϕ are probability functions that define the decoder and encoder part repressively. θ and ϕ are the parameters (biases and weights) in the neural networks. We have a simple visualization of the directed graphical model with our probabilistic encoder and decoder networks in Figure 1. We define the following latent variable model in a genomic window with data $\mathbf{x} \in \{0,1\}^L$, Gaussian latent variable $\mathbf{z} \in \mathbb{R}^D$ and categorical latent variable $y \in \{0,1\}^C$ (one-hot encoded) as follows:

$$p_\theta(\mathbf{x}, \mathbf{z}, y) = p_\theta(\mathbf{x} \,|\, \mathbf{z}) p_\theta(\mathbf{z} \,|\, y) p(y), \tag{1}$$

with generative processes defined as:

$$p(y) = \mathrm{Cat}(y;\, C^{-1}\mathbf{1}), \tag{2}$$

$$p_\theta(\mathbf{z} \,|\, y) = \mathcal{N}(\mathbf{z};\, \mu_\theta(y),\, \sigma^2_\theta(y)\mathbf{1}), \tag{3}$$

$$p_\theta(\mathbf{x} \,|\, \mathbf{z}) = \mathrm{Ber}(\mathbf{x};\, \pi_\theta(\mathbf{z})). \tag{4}$$

Here $\mathbf{z}$ is a D-dimensional vector representing the latent haplotype encoding and C is the number of haplotype clusters, while $\mu_\theta : \{0,1\}^C \mapsto \mathbb{R}^D$, $\sigma^2_\theta : \{0,1\}^C \mapsto \mathbb{R}^D$ and $\pi_\theta : \mathbb{R}^D \mapsto [0,1]^L$ are mapping functions parameterized by neural networks with network parameters θ. In this case, $\mathrm{Ber}(\mathbf{x};\, \pi_\theta(\mathbf{z}))$ is a vectorized notation of Bernoulli distributions and each of the L sites will have an independent probability mass function. We assume that the covariance matrix of the multivariate Gaussian distribution is a diagonal matrix which will promote disentangled factors. We have the following inference (encoder) model:

$$q_\phi(\mathbf{z}, y \,|\, \mathbf{x}) = q_\phi(\mathbf{z} \,|\, \mathbf{x}, y) q_\phi(y \,|\, \mathbf{x}), \tag{5}$$

$$q_\phi(y \,|\, \mathbf{x}) = \mathrm{Cat}(y;\, \pi_\phi(\mathbf{x})), \tag{6}$$

$$q_\phi(\mathbf{z} \,|\, \mathbf{x}, y) = \mathcal{N}(\mathbf{z};\, \mu_\phi(\mathbf{x}, y),\, \sigma^2_\phi(\mathbf{x}, y)\mathbf{1}), \tag{7}$$

where $\mu_\phi : \{0,1\}^{L+C} \mapsto \mathbb{R}^D$, $\sigma^2_\phi : \{0,1\}^{L+C} \mapsto \mathbb{R}^D$ and $\pi_\phi : \{0,1\}^L \mapsto [0,1]^C$ again are mapping functions parameterized by neural networks with network parameters ϕ. Therefore the marginal posterior distribution and marginal approximate posterior distribution of $\mathbf{z}$ will both be a mixture of Gaussians. Thus, $q_\phi(\mathbf{z}, y \,|\, \mathbf{x})$ and $p_\theta(\mathbf{x} \,|\, \mathbf{z})$ will constitute the probabilistic encoder and decoder, respectively, in comparison to the deterministic encoder and decoder of the standard autoencoder.

From the marginal log-likelihood of the data, we derive the following ELBO of our variational autoencoder model for haplotype i,

$$\log p_\theta(\mathbf{x}_i) \geq \mathcal{L}(\phi, \theta; \mathbf{x}_i) = \mathbb{E}_{q_\phi(\mathbf{z}, y \,|\, \mathbf{x}_i)} \left[\log p_\theta(\mathbf{x}_i \,|\, \mathbf{z}) - \log \frac{q_\phi(\mathbf{z} \,|\, \mathbf{x}_i, y)}{p_\theta(\mathbf{z} \,|\, y)} - \log \frac{q_\phi(y \,|\, \mathbf{x}_i)}{p(y)} \right], \tag{8}$$

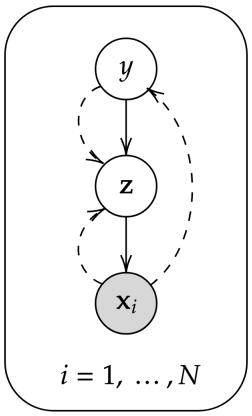

Figure 1: Directed graphical model of `HaploNet`, where the solid edges captures the generative model described in Equation 1, $p_\theta(\mathbf{x}, \mathbf{z}, y)$, with decoder $p_\theta(\mathbf{x} \,|\, \mathbf{z})$. The solid edges are parameterized by neural networks with parameters θ. The dashed edges captures the inference model (encoder) of the variational approximation to the posterior, $q_\phi(\mathbf{z}, y \,|\, \mathbf{x})$. The dashed edges are parameterized by neural networks with parameters ϕ. Color of circular nodes indicate whether the variable is latent or observed.

where the marginal log-likelihood of the full data in a window is given by:

$$\log p_\theta(\mathbf{x}_1, \ldots, \mathbf{x}_{2N}) = \sum_{i=1}^{2N} \log p_\theta(\mathbf{x}_i). \tag{9}$$

The derivation of this ELBO is described in the supplementary material. We immediately see that the first term describes the reconstruction error of mapping from the latent space back to the input space as in an autoencoder framework. The next two terms act as regularization on the learnt latent spaces, where the second term encourages the variational Gaussian distribution to be close to the approximated prior distribution, while the last term encourages anti-clustering behaviour to prevent all haplotypes to cluster in one component. This is a modification of the unsupervised loss of the M2 model [23] as described by Rui Shu[1] where information of the haplotype cluster is also propagated through $p_\theta(\mathbf{z} \,|\, y)$. However, we further approximate the categorical latent variable with samples from a Gumbel-Softmax distribution [19, 28] instead of the categorical distribution. The Gumbel-Softmax distribution is a continuous approximation to the categorical distribution that can be easily reparameterized for differentiable sampling and gradient estimations. In this way, we can avoid an expensive computational step of having to marginalize over the categorical latent variable in the SGVB estimator of the ELBO as is done in the original model. A lot of different interpretations and implementations of the Gaussian Mixture Variational Autoencoder (GMVAE) have been proposed [12, 11, 7, 20], and a similar architecture to ours has been implemented[2].

[1] http://ruishu.io/2016/12/25/gmvae/
[2] https://github.com/jariasf/GMVAE

2.1.1 Reparameterization tricks

End-to-end learning is a desired trait in deep learning models as it enables joint training of model parameters through backpropagation. However with the network architectures of variational autoencoders, the sampling steps of the latent variables are not letting information flow end-to-end as they are non-differentiable. The reparameterization trick is a simple approach to express a random variable as deterministic using the introduction of an auxiliary variable as defined in the SGVB estimator [25]. In this way, the stochastic process of sampling is removed from the computation graph and we can therefore efficiently estimate gradients and perform end-to-end learning of our network parameters, (ϕ, θ), through Monte Carlo samples of the latent variables to approximate the ELBO in Equation 8.

A D-dimensional sample from a multivariate Gaussian distribution, $\mathbf{z}$, can be expressed using the introduction of auxiliary noise variable $\boldsymbol{\epsilon}$, with $\epsilon_d \sim \mathcal{N}(0, 1)$, for $d = 1, \ldots, D$:

$$z_d = \mu_d + \sigma_d \epsilon_d. \tag{10}$$

Similarly, the reparameterization trick for the Gumbel-Softmax distribution is an approach for providing differentiable samples that approximate samples from a categorical distribution. It is an extension of the Gumbel-Max trick [29] for sampling from a categorical distribution, where the softmax function is used as a continuous differentiable approximation to the arg max function. The Gumbel-Softmax distribution was derived simultaneously in two independent studies [19, 28]. A C-dimensional one-hot encoded categorical sample y can be approximated as sample $\tilde{y}$ from the Gumbel-Softmax distribution, and it is defined as follows for its entry c:

$$\tilde{y}_c = \frac{\exp((\log(\pi_c) + g_c)/\tau)}{\sum_{c'=1}^{C} \exp((\log(\pi_{c'}) + g_{c'})/\tau)}. \tag{11}$$

Here $g_c, \ldots, g_C$ are i.i.d. samples from Gumbel$(0, 1)$ (auxiliary variables) and π_c is the probability of haplotype cluster c. The temperature parameter, τ, is governing the smoothness of the distribution such that when τ approaches 0, samples become identical to samples from the categorical distribution. We have provided examples of the Gumbel-Softmax distribution in the supplementary material to provide the reader with a better intuition of it. Note that $\tilde{y} \in [0, 1]^C$ unlike y, and the functions $\mu_\theta, \sigma_\theta, \mu_\phi, \sigma_\phi$ parameterized by the neural networks are therefore redefined as mapping to or from a continuous space in the practical implementation.

2.1.2 Simulation from haplotype clusters

We can use the trained neural networks to generate the most probable haplotypes of the inferred haplotype clusters or to simulate entirely new data from the haplotype clusters using the functions parameterizing the $p_\theta(\mathbf{z} \,|\, y)$ and $p_\theta(\mathbf{x} \,|\, \mathbf{z})$ distributions. The distribution means of $p_\theta(\mathbf{z} \,|\, y)$ are learnt for each haplotype cluster and we use them to estimate the mean haplotype representing each of the C clusters. The mean latent encodings, using only information of the haplotype clusters, $\mathbb{E}[p_\theta(\mathbf{z} \,|\, y)] = \mu_\theta(y)$, can be decoded to generate the most probable haplotypes as follows:

$$\hat{\mathbf{x}}_c = \arg\max_{\mathbf{x}} \, p_\theta(\mathbf{x} \,|\, \mu_\theta(y = c)), \tag{12}$$

where $\hat{\mathbf{x}}_c \in \{0, 1\}^L$ represents the mean haplotype of the c-th haplotype cluster for $c = 1, \ldots, C$ (one-hot encoded). The information learnt by our mappings parameterized using neural networks is

therefore very interpretable. We can also sample from the distributions using the reparameterization trick to generate random haplotype sequences from a haplotype cluster. For example, given the most probable sequence of haplotype clusters a population, we can easily simulate entire chromosomes that will resemble the real data of the population.

2.2 Global population structure inferred using PCA

The inferred local encodings can be used to estimate global population structure by summarizing the information across windows using PCA. There is no obvious way to combine the two haplotypes of a single individual in our setting. However, we propose an approach that computes the covariance between individuals using the mean latent encodings of their haplotypes, $\mathbb{E}[q_\phi(\mathbf{z} \,|\, \mathbf{x}, y)] = \mu_\phi(\mathbf{x}, \pi_\phi(\mathbf{x}))$, where we condition directly on the mean haplotype cluster probabilities, $\mathbb{E}[q_\phi(y \,|\, \mathbf{x})] = \pi_\phi(\mathbf{x})$. For two individuals, we can average the covariance across the four different pairwise combinations of their haplotypes in each window, and then average across all windows for a global estimate. The sample covariance between individual i and j is computed as follows:

$$\mathrm{cov}(i,j) = \frac{1}{4KW} \sum_{w=1}^{W} \sum_{a=1}^{2} \sum_{b=1}^{2} \left(\boldsymbol{\mu}_{i,a}^{(w)} - \hat{\boldsymbol{\mu}}^{(w)} \right)^T \left(\boldsymbol{\mu}_{j,b}^{(w)} - \hat{\boldsymbol{\mu}}^{(w)} \right), \tag{13}$$

where $\boldsymbol{\mu}_{i,a}^{(w)} = \mu_\phi^{(w)}(\mathbf{x}_{i,a}^{(w)}, \pi_\phi(\mathbf{x}_{i,a}^{(w)})) \in \mathbb{R}^D$ represent the mean latent encoding for the a-th haplotype in the w-th window for individual i, and $\hat{\boldsymbol{\mu}}^{(w)} \in \mathbb{R}^D$ represent the average across individuals for the w-th window. We can then finally perform eigendecomposition of the covariance matrix to extract the top eigenvectors that capture population structure as in a standard PCA approach.

2.3 Ancestry proportions and haplotype cluster frequencies

Another often used approach for inferring population structure are by estimating admixture proportions. We propose a model for estimating ancestry proportions and haplotype cluster frequencies assuming K ancestral components based on the model introduced in `NGSadmix` [42], as an extension to the `ADMIXTURE` model [2], where instead of latent states of unobserved genotypes, we have latent states of haplotype clusters. We can exploit and utilize the generative nature of our GMVAE model based on the parameters of the trained neural networks, $\mu_\theta(y)$ and $\pi_\theta(\mathbf{z})$. Define $p(\mathbf{x}_{i,a}^{(w)} \,|\, y = c)$, with y one-hot encoded, as the neural network (NN) likelihood that the data is generated from the c-th haplotype cluster, for the a-th haplotype of individual i in window w. We can then construct the following likelihood model of the data $\mathbf{X}$ given the genome-wide admixture proportions $\mathbf{Q}$ and window-based ancestral haplotype frequencies $\mathbf{F}$:

$$\mathcal{L}(\mathbf{Q}, \mathbf{F}; \mathbf{X}) \propto \prod_{w=1}^{W} \prod_{i=1}^{N} \prod_{a=1}^{2} \sum_{k=1}^{K} \sum_{c=1}^{C} p(\mathbf{x}_{i,a}^{(w)} \,|\, y = c) f_{wkc} q_{ik} \tag{14}$$

with k describing the ancestral state, for $\mathbf{Q} \in [0,1]^{N \times K}$ with constraint $\sum_{k=1}^{K} q_{ik} = 1$ and $\mathbf{F} \in [0,1]^{W \times K \times C}$ with constraint $\sum_{c=1}^{C} f_{wkc} = 1$. The NN likelihoods are calculated as follows using the probability mass function of the Bernoulli distribution and the properties, $\mathbb{E}[p_\theta(\mathbf{z} \,|\, y)] = \mu_\theta(y)$ and $\mathbb{E}[p_\theta(\mathbf{x} \,|\, \mu_\theta(y))] = \pi_\theta(\mu_\theta(y))$:

$$p(\mathbf{x}_{i,a}^{(w)} \mid y = c) \propto \prod_{l=1}^{L} \pi_\theta(\mu_\theta(y = k))^{x_{i,a,l}^{(w)}} (1 - \pi_\theta(\mu_\theta(y = k)))^{1 - x_{i,a,l}^{(w)}}, \tag{15}$$

for $c = 1, \ldots, C$ (one-hot encoded) and $\mathbf{x}_{i,a}^{(w)} \in \{0, 1\}^L$ is the data of the a-th haplotype of individual i in window w. Maximum likelihood estimates of F and Q are obtained using an expectation-maximization (EM) algorithm. The full description of the EM algorithm is detailed in the supplementary material. We use the S3 scheme of the SQUAREM methods [45] for accelerating our EM implementation, such that one large step is taken in parameter space based on the linear combination of two normal steps.

2.4 Implementation

We have implemented `HaploNet` as a `Python` program using the `PyTorch` library (v.1.7) [32], and it is freely available on https://github.com/rosemeis/HaploNet. We have used the `NumPy` [16] and `scikit-allel` [30] libraries for preprocessing the data from Variant Call Format (VCF) into data structures to be used in `HaploNet`. The EM algorithm and other postprocessing scripts for estimating covariance matrices have been written in `Cython` [5] for speed and parallelism.

Our proposed overall network architecture in `HaploNet`, including descriptions of major substructures, are displayed in Figure 2, and we have used fully-connected layers throughout the network. For all inner layers in our neural networks, we are using exponential linear unit (ELU) [10] activations to induce non-linearity into the networks, followed by batch normalization [18], while all outer layers are modelled with linear activations. This means that we are estimating the logits of the probabilities instead the probabilities directly in $\pi_\theta(\mathbf{z})$ and $\pi_\phi(\mathbf{x})$ for computational stability, as well as for $\sigma_\theta^2(y)$ and $\sigma_\phi^2(\mathbf{x}, y)$ that represent $\log \sigma^2$ in inner computations.

We are training our networks with the Adam optimizer [22] with a learning rate of 1.0×10^{-3}, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ with a batch-size of 128. We have used a fixed temperature in the sampling from the Gumbel-Softmax distribution of $\tau = 0.1$ to approximate and encourage a categorical sampling, and we use one Monte Carlo sample of the latent variables to approximate the expectation in equation 8. In practice we have an additional weight term on the categorical loss term in the ELBO, β_{cat}, such that we are able to adjust the anti-clustering effect of the uniform prior $p(y)$, by weighing the impact of this regularization term in the ELBO (see Equation S11). In all analyses, we have used $\beta_{\text{cat}} = 0.1$. We decided to use all data in the training to account for and model all possible haplotypes present in the dataset as we also have no way of validating our training in this unsupervised setup. We are therefore training our models using a fixed number of 200 epochs.

2.5 1000 Genomes Project

We applied `HaploNet` to the phase 3 data of the 1000 Genomes Project (TGP) [1]. The entire dataset consists of 2504 unrelated phased individuals from 26 different populations that are assigned to 5 super populations, which consist of African (AFR), Admixed American (AMR), East Asian (EAS), European (EUR) and South Asian (SAS), and we inferred local and global population structure for each super population.

We chose three smaller subsets of the data with populations of lower genetic differentiation to compare the inferred global population structure against existing methods. For the first subset, we use four European populations (CEU, GBR, IBS, TSI) that span Northern and Southern Europe,

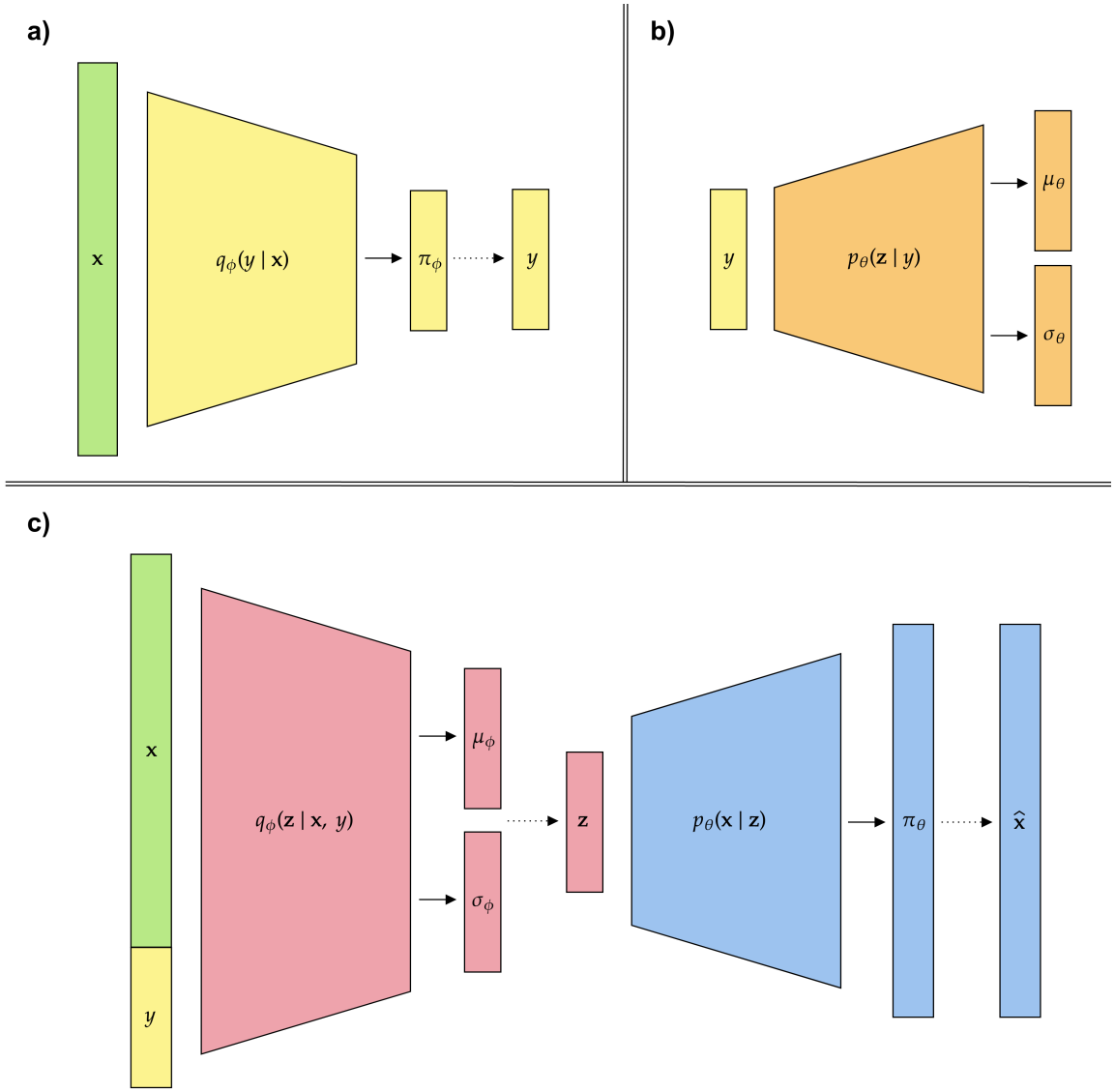

Figure 2: The neural network architecture of `HaploNet` split into three major substructures. Here the solid lines represent the estimation of distribution parameters, while the dashed lines represent sampling of latent variables. ***a)*** Displays the neural network parameterizing the distribution $q_\phi(y \,|\, \mathbf{x})$, for sampling the haplotype cluster, ***b)*** the network parameterizing the regularizing distribution of the sampled encoding, $p_\theta(\mathbf{z} \,|\, y)$, and ***c)*** shows the network parameterizing the distribution $q_\phi(\mathbf{z} \,|\, \mathbf{x}, y)$, for sampling the haplotype encoding, as well as the network decoding the sampled encoding to reconstruct our input. Note that the colors of the network blocks are coherent across substructures such that the sampled y in ***a)*** is used in both ***b)*** and ***c)***.

and in the second subset we use the three Chinese populations (CDX, CHB, CHS), while the third subset consists of five of the African populations (GWD, ESN, MSL, YRI, LWK), excluding ACB and ASW that have European admixture.

For all subsets of the data, we have filtered the variable sites based on minor allele frequency at a standard threshold of 0.05. In `HaploNet`, we used a window size of 1024 SNPs, a latent dimension of 64 for the Gaussian latent variables representing the haplotype encodings and a latent dimension of 16 for the categorical latent variables representing the haplotype clusters. We also tried to use different window-sizes (512, 2048, 5120) for the European super population (EUR) to investigate the effect and robustness of our method, as well as only using SNPs that overlap with the available high density genotype chip data to also explore `HaploNet`'s capabilities for genotype chip datasets. A more detailed description of the network architecture used in each scenario can be found in the supplementary material.

2.5.1 Computational comparisons

All models of `HaploNet` presented in this study have been trained on a machine with a NVIDIA GeForce RTX 2060 Super GPU (8GB VRAM), using CUDA v.11.0 and cuDNN v.8.0, and an Intel Core i5-9600 CPU (3.1 GHz). All other analyses have been performed on a cluster with an Intel Xeon E5-2699 CPU (2.2 GHz). The runtimes listed are therefore related to the system on which they have been run, unless otherwise specified.

In the smaller subsets of the 1000 Genomes Project dataset, we have used `ChromoPainter` (v.4.1.0) [26] to estimate the shared genome chunks between individuals in an unsupervised manner such that no population information is given and all individuals can be used as donors of each other. We are using their linked model that utilizes pre-estimated recombination rates from genetic maps of the human chromosomes to model the correlation between SNPs using default parameters. We used their own R library for performing PCA on the estimated chunk-counts matrix, and we have used `PLINK` (v.1.9) [9] for performing standard PCA on unphased genotypes. We heuristically evaluate the population structure inferred in the different methods by clustering accuracy in principal component space using Gaussian mixture models (GMMs) of the `mclust` R library [39] based on population labels. We have only used the principal components that capture population structure. We further compare the performance of our EM algorithm for estimating ancestry proportions on our NN likelihoods to the widely used `ADMIXTURE` (v.1.3) [2] software across all scenarios. `ADMIXTURE` uses unphased genotypes.

3 Results

3.1 Subsets of the 1000 Genomes Project

We applied `HaploNet` to three subsets of the phased whole-genome sequencing data of the 1000 Genomes Project in order to identify fine-scale population structure using haplotype information, and we compare our method to standard PCA, PCA from `ChromoPainter` and admixture proportions estimated from `ADMIXTURE`. We have summarized the runtimes for `HaploNet` and `ChromoPainter` in each of the three scenarios in Table 1.

We first analyzed a subset of the European populations (CEU, GBR, IBS, TSI) excluding FIN to capture fine-scale structure patterns. After filtering, the dataset consists of 404 individuals and 6.0 million SNPs. We have displayed the inferred population structure in Figure 3 and S2. The

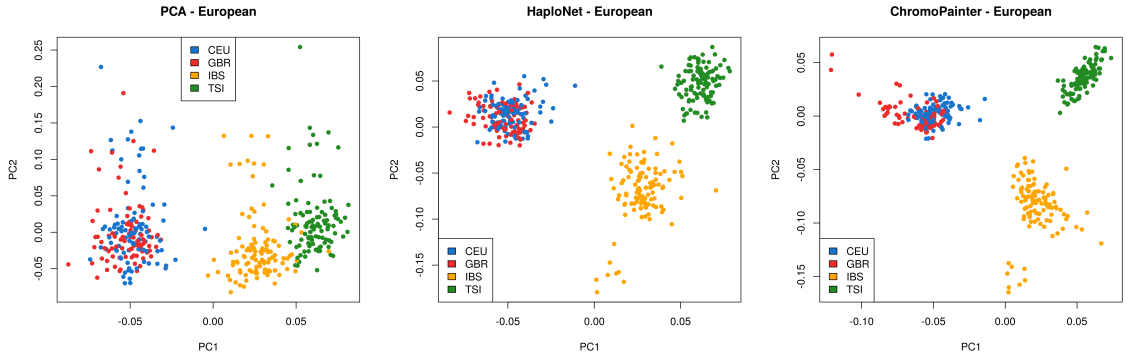

Figure 3: Inferred global population structure using standard PCA, `HaploNet` and `ChromoPainter`, respectively, in a subset (CEU, GBR, IBS, TSI) of the European populations of the 1000 Genomes Project.

benefit of using haplotype information is immediately clear, as `HaploNet` and `ChromoPainter` are able to separate the Southern European populations on the second principal component, which is not possible in standard PCA. In this scenario, `ChromoPainter` had a runtime of 2.7 days for the longest chromosome (chromosome 2), while it took 1.0 hours of training in `HaploNet` on the GPU setup. When looking at the estimated ancestry proportions, it is also clear that `HaploNet` has estimated more distinct components between Northern and Southern Europe in comparison to `ADMIXTURE`, assuming $K = 2$ (Figure S2).

We also analyzed the Chinese populations (CHB, CHS and CDX) that consist of 301 individuals and 5.5 million common SNPs. The results are visualized in Figure 4 and S3, where we see that `HaploNet` is very good at capturing fine-scale structure at both PCA and ancestry proportions level. We again capture similar information in comparison to `ChromoPainter`, where both methods reduce the within population variance, except for a few outliers, however, standard PCA also captures similar overall population structure. The runtime was 1.3 days for `ChromoPainter`, while it was 0.7 hours for training `HaploNet` on chromosome 2. However in terms of estimating ancestry proportions, we are able to separate the sources of ancestry in the two Han Chinese populations (CHB and CHS) to a degree which is not possible in `ADMIXTURE` for $K = 3$. In the PCAs, a CHS individual clusters in the center of the CHB population, which is also perfectly captured by `HaploNet` while not being captured by `ADMIXTURE` due to noise. We did observe convergence issues for `ADMIXTURE` in this scenario for $K = 3$.

In the last of the smaller subsets, we analyzed the African populations excluding ACB and ASW that have European admixture, such that the dataset consists of 504 individuals and 8.5 million SNPs. The PCA plots are displayed in Figure S4 and the estimated ancestry proportions in Figure 5. It appears, as populations are more genetically differentiated, that the three PCA methods are performing similarly and there are no immediate advantages of using `HaploNet` and `ChromoPainter` for PCA. However, `ChromoPainter` seems to have less variance in its populations leading to tighter clustering but also less separation between the two Nigerian populations, ESN and YRI. On chromosome 2, it took 6.1 days to run `ChromoPainter` on this African subset of the 1000 Genomes Project data, while it only took 1.8 hours to train `HaploNet`. For ancestry proportions, we see a striking difference between `HaploNet` and `ADMIXTURE`, assuming $K = 5$, where `ADMIXTURE`

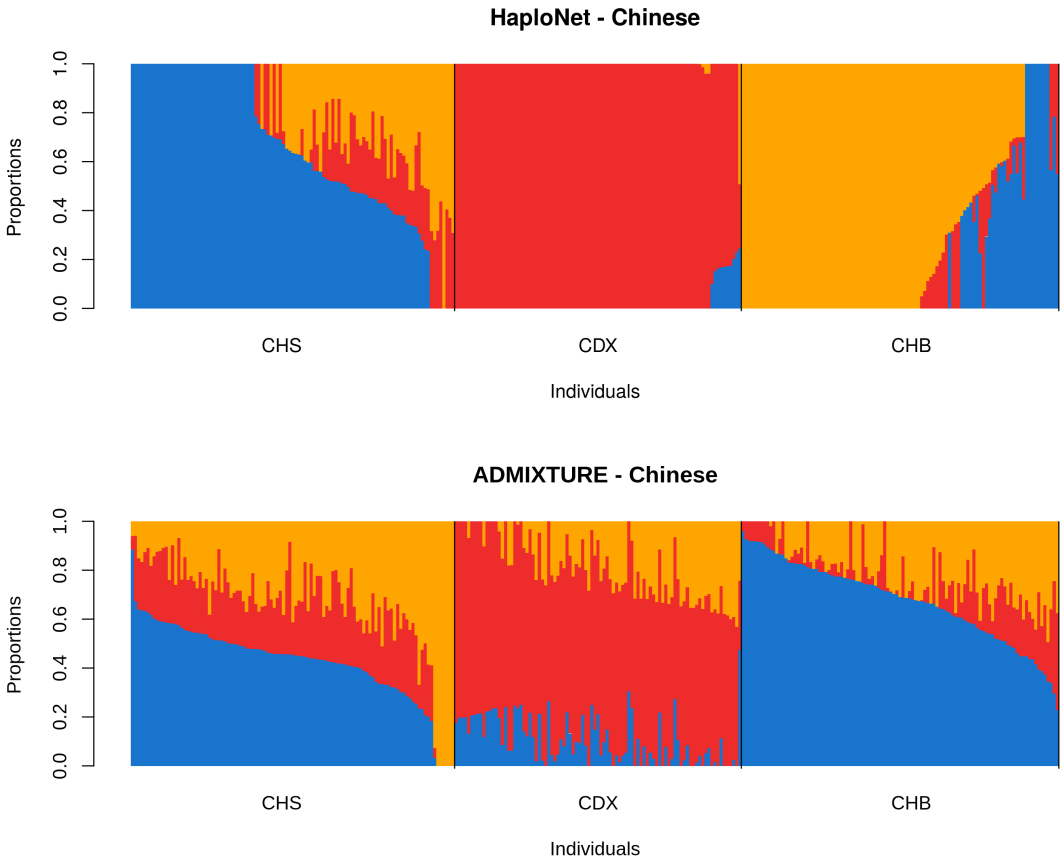

Figure 4: Estimated ancestry proportions of the three Chinese populations in the 1000 Genomes Project based on `HaploNet` and `ADMIXTURE`, respectively, for $K = 3$.

is not able to utilize the fifth ancestry component for capturing structure at all. `HaploNet` is able to split up the ESN and YRI, while overall also estimating more distinct components of the populations. We did observe convergence issues for `ADMIXTURE` in this scenario for $K = 5$.

We evaluated the inferred global population structure by the three approaches using a Gaussian mixture model (`mclust`) to cluster the populations and the results are described in Table S1. We see that the two methods based on haplotype information are much better at distinguishing the European populations (here assuming CEU and GBR to be a single population), which would also be expected based on the PCA plots. For the other two scenarios, we do not observe any major differences between the methods.

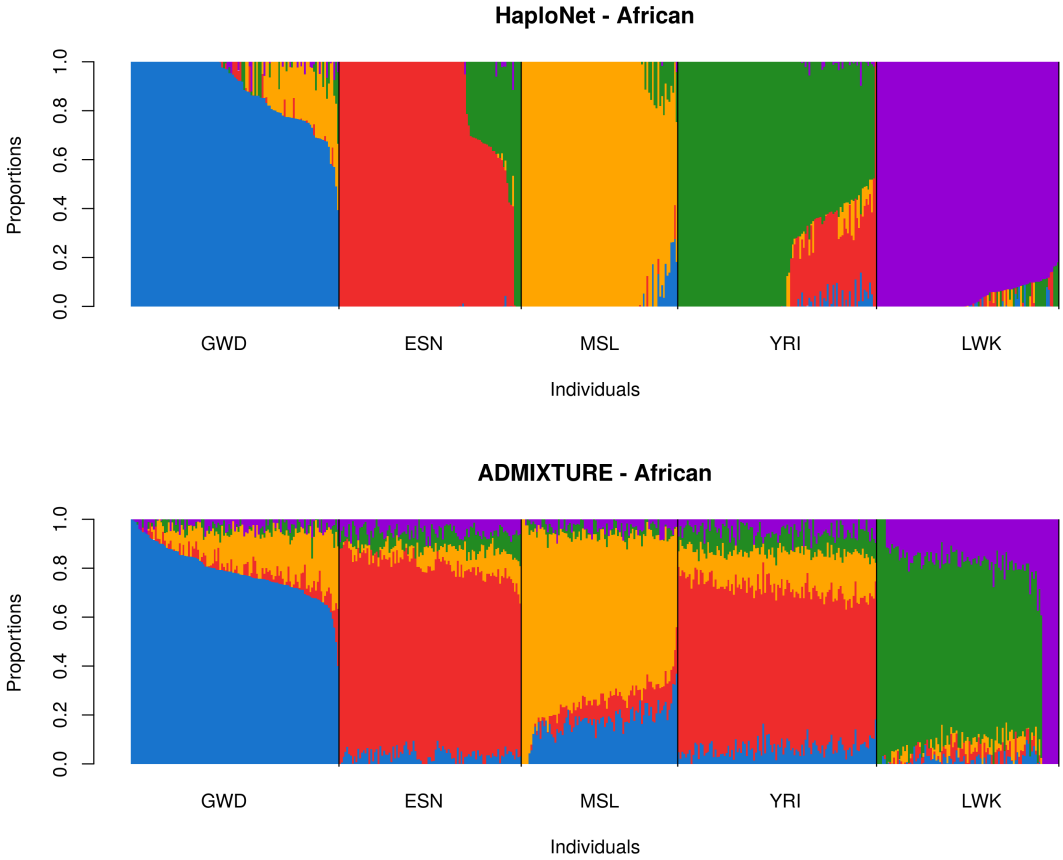

Figure 5: Estimated ancestry proportions of the five African populations in the 1000 Genomes Project without European admixture based on `HaploNet` and `ADMIXTURE`, respectively, for $K = 5$.

3.2 Super populations

We also applied `HaploNet` to all super populations in the 1000 Genomes Project and inferred global population structure and estimated ancestry proportions. The results are visualized for AFR, AMR, EAS, EUR and SAS in Figure S5, S6, S7, S8 and S9, respectively. We observe a general pattern when performing PCA in the different super populations, where `HaploNet` is inferring similar global population structure to standard PCA, though with tighter population clustering. However for EAS and EUR, `HaploNet` is also capturing structure on the third principal component, which is not captured using standard PCA. When estimating ancestry proportions, we see a general pattern of `HaploNet` capturing much more fine-scale structure by improving the signal-to-noise ratio in comparison to `ADMIXTURE`, as was also seen for the smaller subsets. For the EAS super population, the admixed Japanese (JPT) individual is also perfectly captured by `HaploNet`.

We filtered out variable sites not overlapping with the high density genotype chip of the Omni

	HaploNet (GPU)	HaploNet (CPU)	ChromoPainter
African	1.8 hours	4.0 hours	146.0 hours
Chinese	0.7 hours	1.5 hours	30.3 hours
European	1.0 hours	2.4 hours	64.0 hours

Table 1: Runtimes for chromosome 2 in the three smaller subsets of the 1000 Genomes Project using `HaploNet` and `ChromoPainter`. `HaploNet` has been trained on a GPU machine as well as on the same cluster that `ChromoPainter` has been tested on using CPU with 64 threads.

platform [1] in European super population (EUR) to test how well `HaploNet` would perform in this scenario. After filtering, the dataset contained of 1.3 million SNPs. The results are visualized in Figure S12, where `HaploNet` has similar performance in comparison to when it is used on the full dataset, however, it has a slight loss of resolution for estimating ancestry proportions.

3.3 Changing window size

To test the robustness of `HaploNet` in terms of model hyperparameters, we applied it to the European super population with varying window sizes and evaluated its performance of inferring global population structure based on both PCA and ancestry proportions. The results are displayed in Figure S10 and S11. For the PCA plots that are inferred using haplotype encodings, we observe that `HaploNet` is not able to capture the structure on the third principal component for a small window size of 512, however, it is capable of capturing the structure for all larger window sizes. Small differences between the larger window sizes could be due to the stochastic nature of our model framework. The ancestry proportions are estimated based on our generated neural network likelihoods, where we see an opposite pattern with smaller window sizes showing fine-scale resolution and the usage of larger window sizes results in coarser ancestry proportions estimations.

3.4 Reconstruction and simulation

With the generative properties of the `HaploNet` framework, we are able to visualize and evaluate what our neural networks are learning. In Figure 6 and S13, we show two arbitrary chosen segments on chromosome 2 trained on the European super population (EUR). In both segments we see that `HaploNet` is fully capable of reconstructing the haplotype matrix from the haplotype encodings with accuracies of 99.7% and 99.2% for predicting the correct alleles of the haplotypes in the two segments, respectively. We also display the most probable haplotypes of the inferred haplotype clusters, as described in section 2.1.2, to gain insight into the haplotype clustering process in `HaploNet`. It has used 7 and 6 haplotype clusters to model the haplotype encodings in the two different segments, respectively.

We are able to generate new haplotypes by sampling from the learnt generative probability distributions. For each of the five European populations, we used the haplotype encoding of the most probable haplotype cluster in the population of each window to sample the full genome of 50 new haplotypes. As shown in Figure S14, the generated haplotypes demonstrate the same structure patterns as seen in the real data, Figure S8.

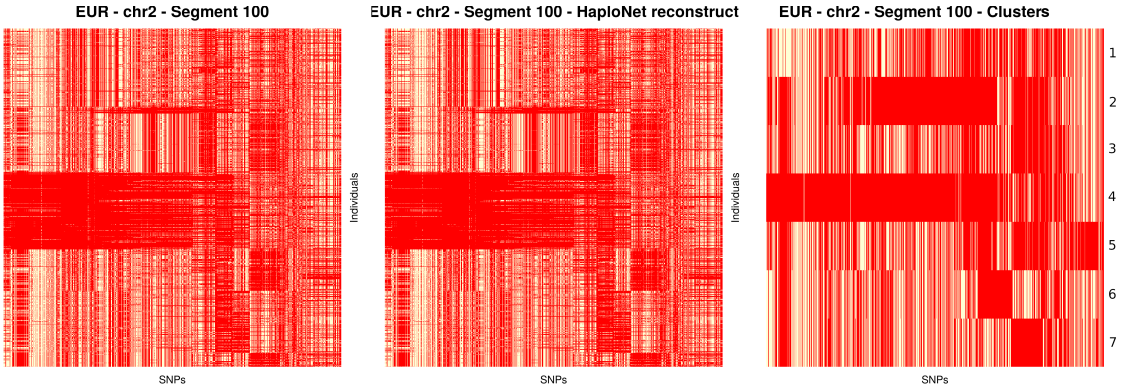

Figure 6: The 100th window in chromosome 2 for the European super population (EUR). Left plot shows the true haplotype matrix with haplotypes ordered by their inferred haplotype cluster in this window by `HaploNet`. Middle plot shows the reconstruction of the haplotype matrix by `HaploNet`, while the last plot shows the reconstructions from the mean haplotype encoding of each haplotype cluster.

4 Discussion

We have presented our new framework, `HaploNet`, which performs dimensionality reduction and clustering on haplotypes using neural networks. We explored its capability to infer population structure based on local parameters, which are estimated in windows along the genome, as well as its generative properties for reconstruction and simulation. We show the benefits of merging machine learning with traditional statistical frameworks, as we develop a novel method for estimating ancestry proportions from neural networks in a likelihood model. We tested `HaploNet` on data from the 1000 Genomes Project and compared its results to commonly used software for inferring population structure using PCA or ancestry proportions. We demonstrate that `HaploNet` is capable of utilizing haplotype information for inferring fine-scale population structure similarly to `ChromoPainter` while being much faster. Besides performing PCA, we also demonstrate that we are capable of estimating ancestry proportions to a degree which is not possible for standard software, such as `ADMIXTURE`, where we utilize neural network architecture to incorporate haplotype information. This architecture further allows us to explore and understand the processes for encoding and clustering the haplotypes through the learnt mappings.

The number of clusters, C, is usually a non-trivial hyperparameter to set in a Gaussian mixture model or in a genetic clustering setting. Multiple runs of varying C are usually performed and evaluated based on some criteria. In our study, we saw that `HaploNet` seemed capable of inferring the number of haplotype clusters to use by setting C to a high fixed number. It would then only use a subset of the haplotype clusters to model the haplotype encodings, which has also been observed in a different application of the GMVAE model [7]. This is especially beneficial for our approach, as it would not be feasible to verify or validate C along the genome of varying window sizes, though future work into this behaviour would be of high interest for the GMVAE model.

A limitation of our model, as for all deep learning models, is that only relatively small genetic datasets are usually available to researchers, which introduces problems with training convergence and overfitting. We have therefore kept the number of parameters in our model low, while using

an autoencoder architecture that naturally regularizes its reconstruction performance. Another advantage of having a small model configuration is observed with the low training times on both GPU and CPU setups that broadens the application opportunities of `HaploNet`. However, the difference between GPU and CPU will be larger when running chromosomes in parallel. For all analyses in this study, we have been using the entire data for training our neural networks to model all available haplotypes, whereas with larger datasets, one would usually partition the data in three; one for training, one for tuning hyperparameters of the model and one for ensuring that the trained model generalizes to new data. We have also limited ourselves to using fixed window lengths across the chromosomes for the matter of simplicity and ease of use, where we instead could have included external information from genetic maps to define windows of variable size. We show that `HaploNet` is somewhat robust to changes in window size when inferring global population structure, either using PCA or ancestry proportions, however we find a trade-off in resolution between the two measures as a function of the window size that is subject for future research. We further show that we are still able to capture much of the fine-scale structure when only evaluating variable sites available on a common genotype chip that allows for broader applications of our method.

Our model also serves as a proof-of-concept and an exploration for how non-linear neural networks and specialized architectures can be utilized to learn haplotype clusters and encodings across a full genome in a very scalable procedure. We hypothesize that as the number of large-scale genetic datasets are growing, we will see the increasing importance of deep learning in population genetics, as deeper models can be trained and more bespoke architectures can be developed. As also shown in our study, we can even use learnt encodings or estimated parameters together with standard statistical frameworks to further improve our understanding of genetic variation. Future developments of our framework would be to use the haplotype clusters and encodings in sequential models, to e.g. infer local ancestry in a semi-supervised approach, as well as investigate its potential properties for imputation based on reconstruction.

5 Data availability statement

The 1000 Genomes Project phase 3 data used in this study is publically available at `https://www.internationalgenome.org/category/phase-3/`.

6 Acknowledgements

The study was supported by the Lundbeck foundation (R215-2015-4174).

References

[1] 1000 Genomes Project Consortium et al. A global reference for human genetic variation. *Nature*, 526(7571):68–74, 2015.

[2] D. H. Alexander, J. Novembre, and K. Lange. Fast model-based estimation of ancestry in unrelated individuals. *Genome research*, 19(9):1655–1664, 2009.

[3] P. Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49, 2012.

[4] C. Battey, G. C. Coffing, and A. D. Kern. Visualizing population structure with variational autoencoders. *BioRxiv*, 2020.

[5] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

[6] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[7] Y. Bozkurt Varolgunes, T. Bereau, and J. F. Rudzinski. Interpretable embeddings from molecular simulations using gaussian mixture variational autoencoders. *arXiv e-prints*, pages arXiv–1912, 2019.

[8] J. Chan, V. Perrone, J. Spence, P. Jenkins, S. Mathieson, and Y. Song. A likelihood-free inference framework for population genetic data using exchangeable neural networks. In *Advances in neural information processing systems*, pages 8594–8605, 2018.

[9] C. C. Chang, C. C. Chow, L. C. Tellier, S. Vattikuti, S. M. Purcell, and J. J. Lee. Second-generation plink: rising to the challenge of larger and richer datasets. *Gigascience*, 4(1):s13742–015, 2015.

[10] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[11] M. Collier and H. Urdiales. Scalable deep unsupervised clustering with concrete gmvaes. *arXiv preprint arXiv:1909.08994*, 2019.

[12] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

[13] C. Ding and X. He. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*, page 29, 2004.

[14] L. Flagel, Y. Brandvain, and D. R. Schrider. The unreasonable effectiveness of convolutional neural networks in population genetic inference. *Molecular biology and evolution*, 36(2):220–238, 2019.

[15] G. R. Gower, P. I. Picazo, M. Fumagalli, and F. Racimo. Detecting adaptive introgression in human evolution using convolutional neural networks. *bioRxiv*, 2020.

[16] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[17] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, 3(5):551–560, 1990.

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[19] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[20] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

[21] J. Kelleher and K. Lohse. Coalescent simulation with msprime. *Statistical Population Genomics*, page 191, 2020.

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

[23] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[24] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[25] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[26] D. J. Lawson, G. Hellenthal, S. Myers, and D. Falush. Inference of population structure using dense haplotype data. *PLoS Genet*, 8(1):e1002453, 2012.

[27] N. Li and M. Stephens. Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. *Genetics*, 165(4):2213–2233, 2003.

[28] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[29] C. J. Maddison, D. Tarlow, and T. Minka. A* sampling. In *Advances in Neural Information Processing Systems*, pages 3086–3094, 2014.

[30] A. Miles and N. Harding. scikit-allel: A python package for exploring and analysing genetic variation data, 2017.

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

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[33] N. Patterson, A. L. Price, and D. Reich. Population structure and eigenanalysis. *PLoS genet*, 2(12):e190, 2006.

[34] J. K. Pritchard, M. Stephens, and P. Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155(2):945–959, 2000.

[35] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[36] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[38] D. R. Schrider and A. D. Kern. Supervised machine learning for population genetics: a new paradigm. *Trends in Genetics*, 34(4):301–312, 2018.

[39] L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. mclust 5: clustering, classification and density estimation using gaussian finite mixture models. *The R journal*, 8(1):289, 2016.

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

[41] R. Shu, H. H. Bui, S. Zhao, M. J. Kochenderfer, and S. Ermon. Amortized inference regularization. *Advances in Neural Information Processing Systems*, 31:4393–4402, 2018.

[42] L. Skotte, T. S. Korneliussen, and A. Albrechtsen. Estimating individual admixture proportions from next generation sequencing data. *Genetics*, 195(3):693–702, 2013.

[43] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pages 3738–3746, 2016.

[44] H. Tang, J. Peng, P. Wang, and N. J. Risch. Estimation of individual admixture: analytical and study design considerations. *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society*, 28(4):289–301, 2005.

[45] R. Varadhan and C. Roland. Simple and globally convergent methods for accelerating the convergence of any em algorithm. *Scandinavian Journal of Statistics*, 35(2):335–353, 2008.

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

[47] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[48] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pages 3861–3870. PMLR, 2017.

[49] B. Yelmen, A. Decelle, L. Ongaro, D. Marnetto, F. Montinaro, C. Furtlehner, L. Pagani, and F. Jay. Creating artificial human genomes using generative models. 2019.

[50] A. Zimek, E. Schubert, and H.-P. Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5):363–387, 2012.

Supplementary Material

Derivation of the evidence lower bound (ELBO)

We here derive the ELBO from the marginal log-likelihood of a haplotype, $\mathbf{x}$, using Jensen's inequality and the following definitions, $p_\theta(\mathbf{x}, \mathbf{z}, y) = p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z} \mid y) p(y)$, $q_\phi(\mathbf{z}, y \mid \mathbf{x}) = q_\phi(\mathbf{z} \mid \mathbf{x}, y) q_\phi(y \mid \mathbf{x})$. We follow the mathematical used in the machine learning literature and Kingma and Welling [25].

$$\log p_\theta(\mathbf{x}) = \log \sum_y \int_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}, y)\, d\mathbf{z} \tag{S1}$$

$$= \log \sum_y \int_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}, y) \frac{q_\phi(\mathbf{z}, y \mid \mathbf{x})}{q_\phi(\mathbf{z}, y \mid \mathbf{x})}\, d\mathbf{z} \tag{S2}$$

$$= \log \mathbb{E}_{q_\phi(\mathbf{z}, y \mid \mathbf{x})} \left[\frac{p_\theta(\mathbf{x}, \mathbf{z}, y)}{q_\phi(\mathbf{z}, y \mid \mathbf{x})} \right] \tag{S3}$$

$$\geq \mathbb{E}_{q_\phi(\mathbf{z}, y \mid \mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z}, y)}{q_\phi(\mathbf{z}, y \mid \mathbf{x})} \right] \tag{S4}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}, y \mid \mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z} \mid y) p(y)}{q_\phi(\mathbf{z} \mid \mathbf{x}, y) q_\phi(y \mid \mathbf{x})} \right] \tag{S5}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}, y \mid \mathbf{x})} \left[\log p_\theta(\mathbf{x} \mid \mathbf{z}) - \log \frac{q_\phi(\mathbf{z} \mid \mathbf{x}, y)}{p_\theta(\mathbf{z} \mid y)} - \log \frac{q_\phi(y \mid \mathbf{x})}{p(y)} \right]. \tag{S6}$$

We can rewrite the last term in the expectation using $p(y = c) = C^{-1}$, for $c = 1, \ldots, C$:

$$\mathbb{E}_{q_\phi(\mathbf{z}, y \mid \mathbf{x})} \left[\log \frac{q_\phi(y \mid \mathbf{x})}{p(y)} \right] = \mathbb{E}_{q_\phi(y \mid \mathbf{x})} \mathbb{E}_{q_\phi(\mathbf{z} \mid \mathbf{x}, y)} \left[\log \frac{q_\phi(y \mid \mathbf{x})}{p(y)} \right] \tag{S7}$$

$$= \mathbb{E}_{q_\phi(y \mid \mathbf{x})} \left[\log \frac{q_\phi(y \mid \mathbf{x})}{p(y)} \right] \tag{S8}$$

$$= \mathbb{E}_{q_\phi(y \mid \mathbf{x})} \left[\log q_\phi(y \mid \mathbf{x}) \right] - \mathbb{E}_{q_\phi(y \mid \mathbf{x})} \left[\log p(y) \right] \tag{S9}$$

$$= -H(q_\phi(y \mid \mathbf{x})) + \log C, \tag{S10}$$

where $H(X)$ is the entropy of a given distribution. In that way, both the second and third term will act as regularization on the reconstruction term through the latent space distributed by $q_\phi(\mathbf{z}, y \mid \mathbf{x})$.

ELBO with β_{cat} weight term

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}, y \mid \mathbf{x})} \left[\log p_\theta(\mathbf{x} \mid \mathbf{z}) - \log \frac{q_\phi(\mathbf{z} \mid \mathbf{x}, y)}{p_\theta(\mathbf{z} \mid y)} \right] - \beta_{\text{cat}} \left(-H(q_\phi(y \mid \mathbf{x})) + \log C \right) \tag{S11}$$

Derivation of EM algorithm

Here we derive the EM algorithm for estimating ancestry proportions $\mathbf{Q} \in [0,1]^{N \times K}$ and window-based haplotype frequencies $\mathbf{F} \in [0,1]^{W \times K \times C}$, and we will use the following notation:

W number of windows

N number of individuals

C number of haplotype states

K number of ancestral states

$\mathbf{x}_{i,a}^{(w)}$ observed data of the a-th haplotype of individual i in window w

$\psi^{(n)}$ auxiliary variable in iteration n

$q_{ik}^{(n)}$ ancestry proportion of state k for individual i in iteration n

$f_{wck}^{(n)}$ ancestral frequency of haplotype state c in ancestral state k in window w in iteration n

We define the likelihood function:

$$\mathcal{L}(\mathbf{Q}, \mathbf{F}; \mathbf{X}) \propto \prod_{w=1}^{W} \prod_{i=1}^{N} \prod_{a=1}^{2} p(\mathbf{x}_{i,a}^{(w)} \mid \psi) \tag{S12}$$

$$p(\mathbf{x}_{i,a}^{(w)} \mid \psi) = \sum_{k=1}^{K} \sum_{c=1}^{C} p(\mathbf{x}_{i,a}^{(w)} \mid \psi, k, y = c) p(y = c \mid \psi, c) p(k \mid \psi) \tag{S13}$$

$$= \sum_{k=1}^{K} \sum_{c=1}^{C} p(\mathbf{x}_{i,a}^{(w)} \mid y = c) f_{wkc} q_{ik}. \tag{S14}$$

The EM update for q_{ik} is defined as follows for iteration $n+1$:

$$q_{ik}^{(n+1)} = \frac{\sum_{w=1}^{W} \sum_{a=1}^{2} p(k \mid \mathbf{x}_{i,a}^{(w)}, \psi^{(n)})}{\sum_{k=1}^{K} \sum_{w=1}^{W} \sum_{a=1}^{2} p(k \mid \mathbf{x}_{i,a}^{(w)}, \psi^{(n)})} \tag{S15}$$

$$= \frac{\sum_{w=1}^{W} \sum_{a=1}^{2} p(k \mid \mathbf{x}_{i,a}^{(w)}, \psi^{(n)})}{2W}, \tag{S16}$$

and the update for f_{wkc}:

$$f_{wkc}^{(n+1)} = \frac{\sum_{i=1}^{N} \sum_{a=1}^{2} p(k, y = c \mid \mathbf{x}_{i,a}^{(w)}, \psi^{(n)})}{\sum_{c'=1}^{C} \sum_{i=1}^{N} \sum_{a=1}^{2} p(k, c = k' \mid \mathbf{x}_{i,a}^{(w)}, \psi^{(n)})}, \tag{S17}$$

where

$$p(k \,|\, \mathbf{x}_{i,a}^{(w)}, \psi^{(n)}) = \sum_{c=1}^{C} p(k, y = c \,|\, \mathbf{x}_{i,a}^{(w)}, \psi^{(n)}), \tag{S18}$$

and

$$p(k, y = c \,|\, \mathbf{x}_{i,a}^{(w)}, \psi^{(n)}) = \frac{p(\mathbf{x}_{i,a}^{(w)} \,|\, k, y = c, \psi^{(n)}) p(y = c, \mathbf{x}_{i,a}^{(w)} \,|\, \psi^{(n)})}{\sum_{k'=1}^{K} \sum_{c'=1}^{C} p(\mathbf{x}_{i,a}^{(w)} \,|\, k', y = c', \psi^{(n)}) p(y = c', \mathbf{x}_{i,a}^{(w)} \,|\, \psi^{(n)})} \tag{S19}$$

$$= \frac{p(\mathbf{x}_{i,a}^{(w)} \,|\, y = c) p(y = c \,|\, k, \psi^{(n)}) p(k \,|\, \psi^{(n)})}{\sum_{k'=1}^{K} \sum_{c'=1}^{C} p(\mathbf{x}_{i,a}^{(w)} \,|\, y = c') p(y = c' \,|\, k', \psi^{(n)}) p(k' \,|\, \psi^{(n)})} \tag{S20}$$

$$= \frac{p(\mathbf{x}_{i,a}^{(w)} \,|\, y = c) f_{wkc}^{(n)} q_{ik}^{(n)}}{\sum_{k'=1}^{K} \sum_{c'=1}^{C} p(\mathbf{x}_{i,a}^{(w)} \,|\, y = c') f_{wk'c'}^{(n)} q_{ik'}^{(n)}}. \tag{S21}$$

Network architectures

In all models, the architectures for the networks of GMVAE were defined as follows:

- $\pi_\phi(\mathbf{x})$: L-H-ELU-BN-C
- $\mu_\phi(\mathbf{x}, y)$, $\sigma^2_\phi(\mathbf{x}, y)$: $(L + C)$-H-ELU-BN-D
- $\mu_\theta(y)$, $\sigma^2_\theta(y)$: C-D
- $\pi_\theta(\mathbf{z})$: D-H-ELU-BN-L

with a hidden layer of 256 units ($H = 256$), ELU being an ELU activation layer, and BN a batch normalization layer. For a window size of $L = 1024$, the number of model parameters is 8.5×10^5.

AFR - including smaller subset (GWD, ESN, MSL, YRI, LWK)

- Window size: $L = 1024$
- Dimensions: $D = 64$, $C = 16$
- Batch size: 128
- Epochs: 200

AMR

- Window size: $L = 1024$
- Dimensions: $D = 64$, $C = 16$
- Batch size: 128
- Epochs: 200

EAS - including smaller subset (CDX, CHB, CHS)

- Window size: $L = 1024$
- Dimensions: $D = 64$, $C = 16$
- Batch size: 128
- Epochs: 200

EUR - including smaller subset (CEU, GBR, IBS, TSI)[3]

- Window size: $L = \{512, 1024, 2048, 5120\}$
- Dimensions: $D = 64$, $C = 16$
- Batch size: 128
- Epochs: 200

SAS

- Window size: $L = 1024$
- Dimensions: $D = 64$, $C = 16$
- Batch size: 128
- Epochs: 200

[3] $L = 256, H = 128, D = 32$ were used for only evaluating sites of the high density genotype chip.

Supplementary figures and tables

Gumbel-Softmax sampling

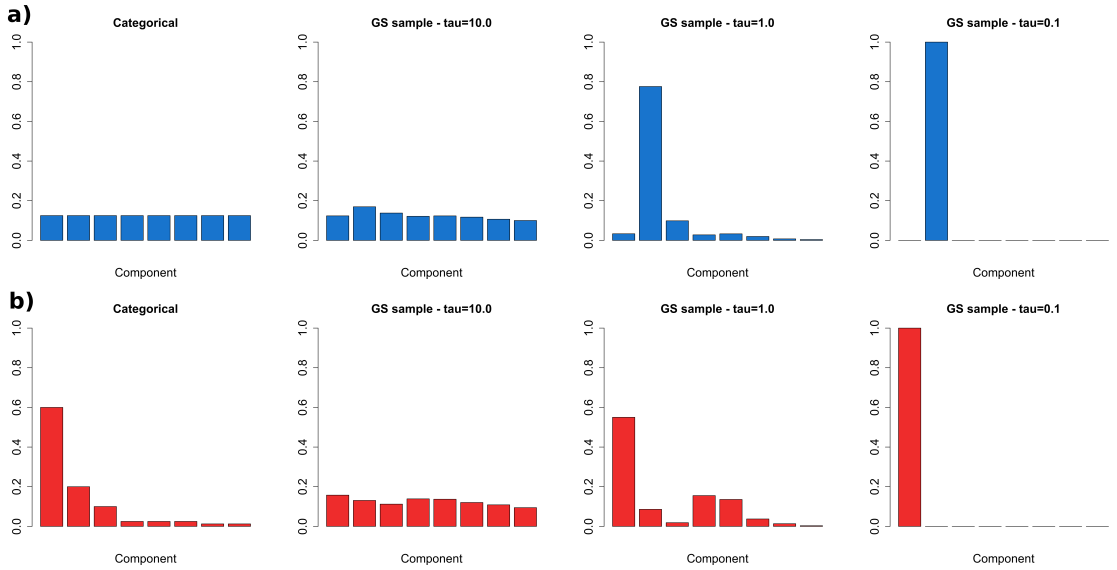

Figure S1: Two examples of approximate sampling from different categorical distributions using the Gumbel-Softmax distribution. First column displays the two categorical distributions with the two examples distinguishable by color, and the next three columns display random samplings from the distributions. The random sampling is performed using the reparameterization trick as described in Equation 11 with $\tau = 10, 1, 0.1$. ***a)*** A discrete uniform distribution, ***b)*** a categorical distribution with decreasing probabilities. Here it can be seen that as τ approaches 0, the sample from the Gumbel-Softmax distribution will converge to a categorical sample, while as τ approaches ∞, it will converge towards a uniform sample.

Smaller subsets

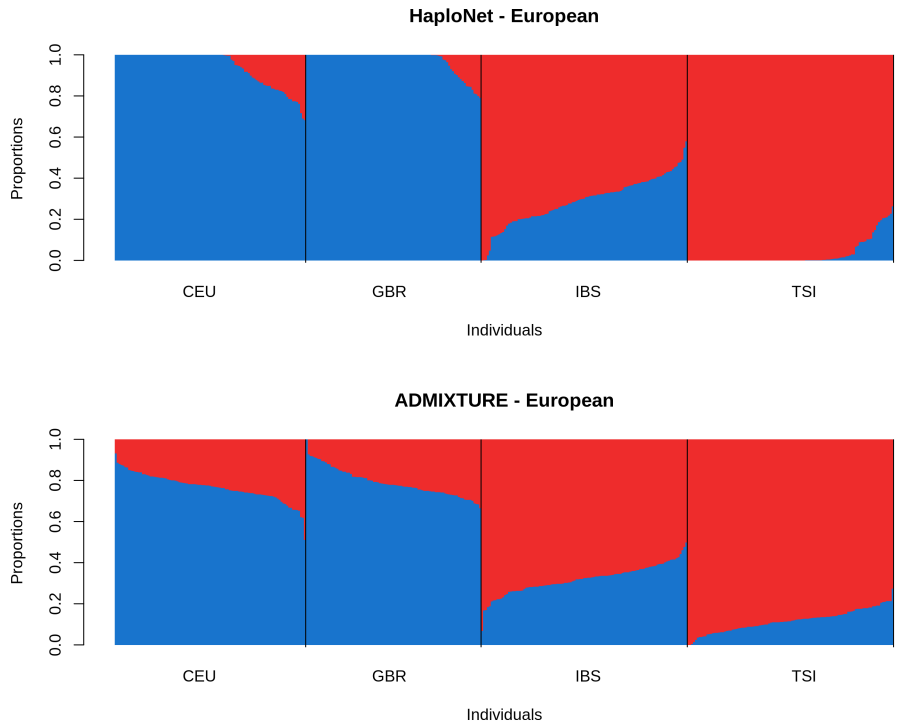

Figure S2: Estimated ancestry proportions of the European populations excluding FIN in the 1000 Genomes Project based on `HaploNet` and `ADMIXTURE`, respectively, for $K = 2$.

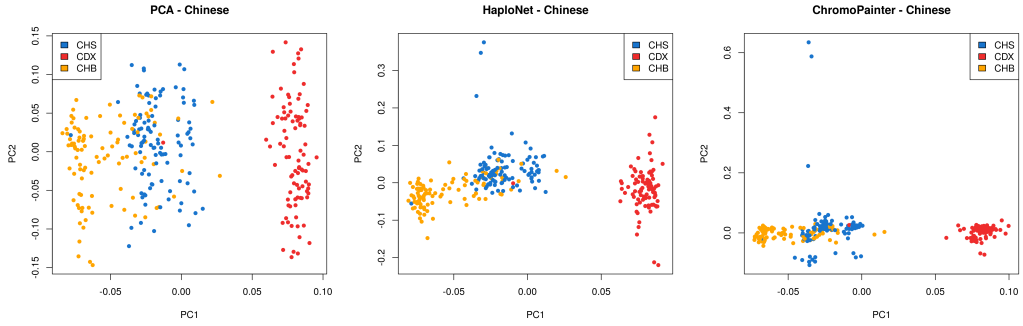

Figure S3: Inferred population structure based on PCA in the Chinese populations (CDX, CHB, CHS) of the 1000 Genomes Project. The PCA plots show the top two inferred principal components using standard PCA, `HaploNet` and `ChromoPainter`, respectively.

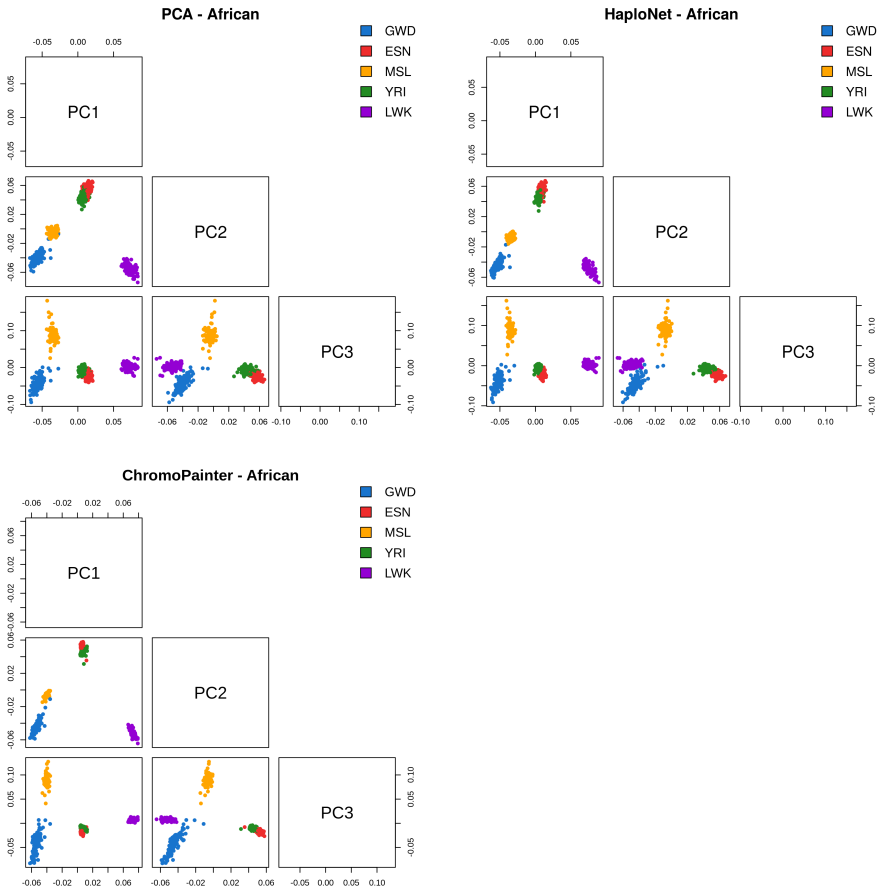

Figure S4: Global population structure of the African populations without European admixture in the 1000 Genomes Project. Top left figure displays the pairwise plots of the top 3 inferred principal components using standard PCA, the top right figure displays the pairwise plots of the principal components inferred by `HaploNet` and the bottom figure shows the pairwise plots of the principal components inferred by `ChromoPainter`.

	PCA	**HaploNet**	**ChromoPainter**
African (4 clusters)	0.784 (3)	0.784 (3)	0.784 (3)
Chinese (3 clusters)	0.837 (1)	0.841 (1)	0.837 (1)
European (3 clusters)	0.592 (1)	1.000 (2)	1.000 (2)

Table S1: Clustering accuracy in inferred PC-space based on population labels using GMMs in `mclust` for standard PCA, `HaploNet` and `ChromoPainter`. The number of PCs used for the clustering for each method is specified in parentheses, where only PCs capturing global structure are used.

Super populations

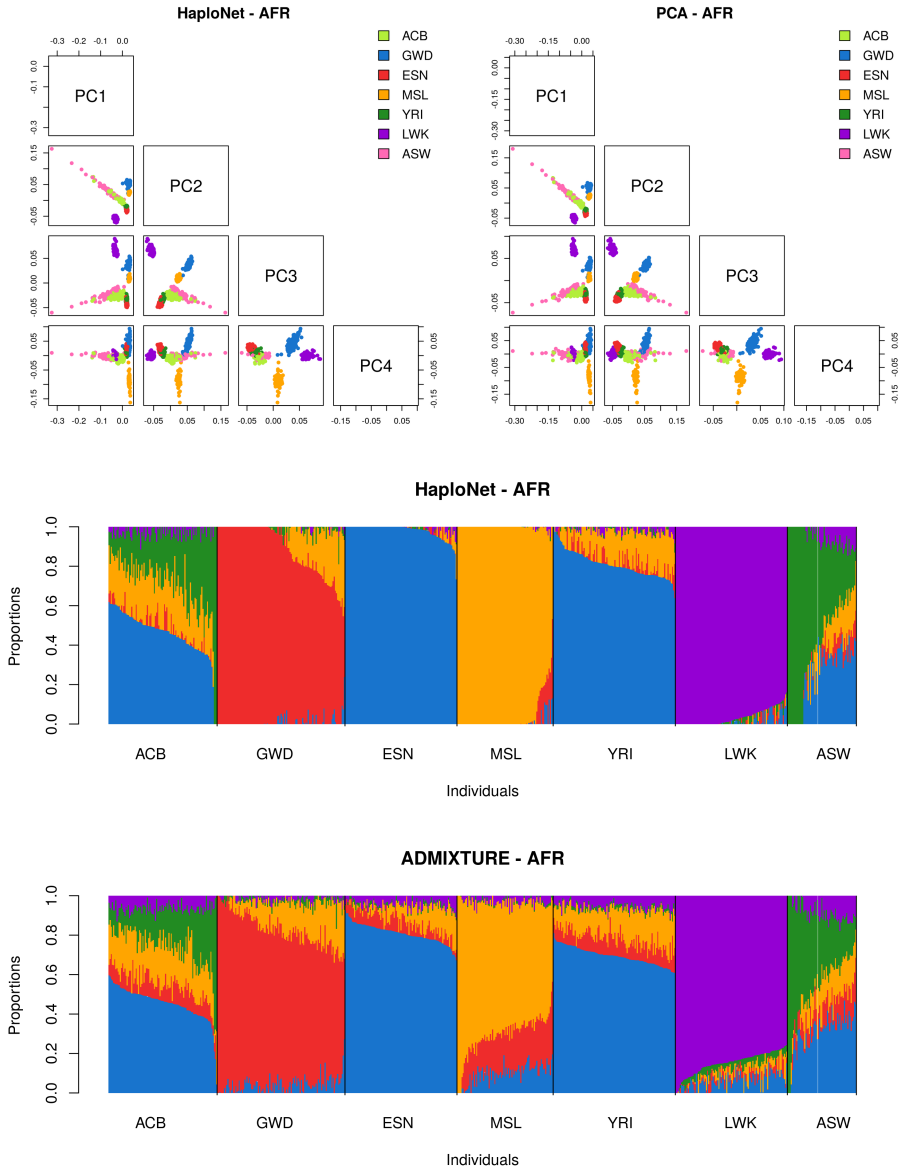

Figure S5: Global population structure inferred for the African (AFR) super population in the 1000 Genomes Project. Top left figure displays the pairwise plots of the top four inferred principal components from `HaploNet`, while the top right figure displays the results of standard PCA. The two bottom figures shows the estimated ancestry proportions of `HaploNet` and `ADMIXTURE` for $K = 5$, respectively.

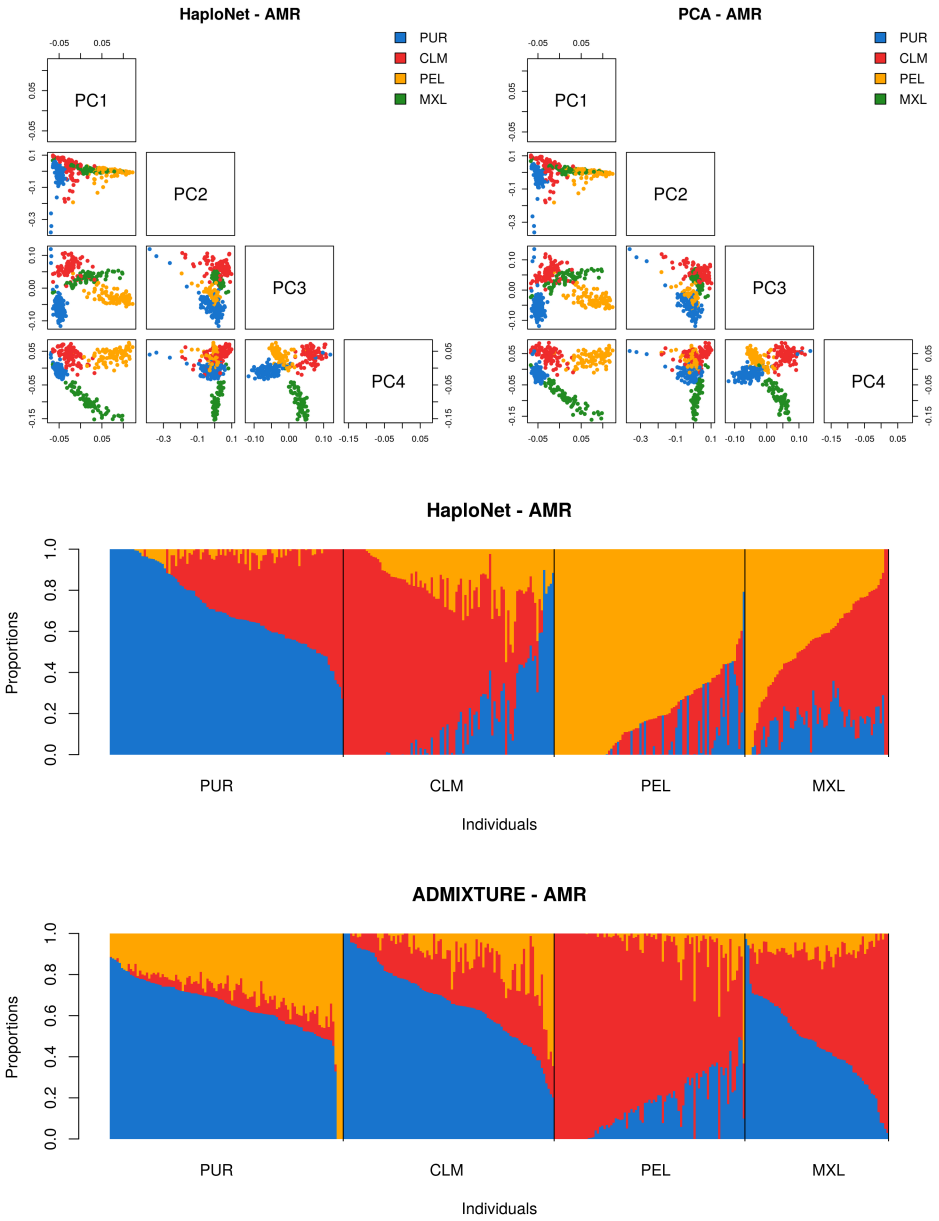

Figure S6: Global population structure inferred for the American (AMR) super population in the 1000 Genomes Project. Top left figure displays the pairwise plots of the top four inferred principal components from `HaploNet`, while the top right figure displays the results of standard PCA. The two bottom figures shows the estimated ancestry proportions of `HaploNet` and `ADMIXTURE` for $K = 3$, respectively.

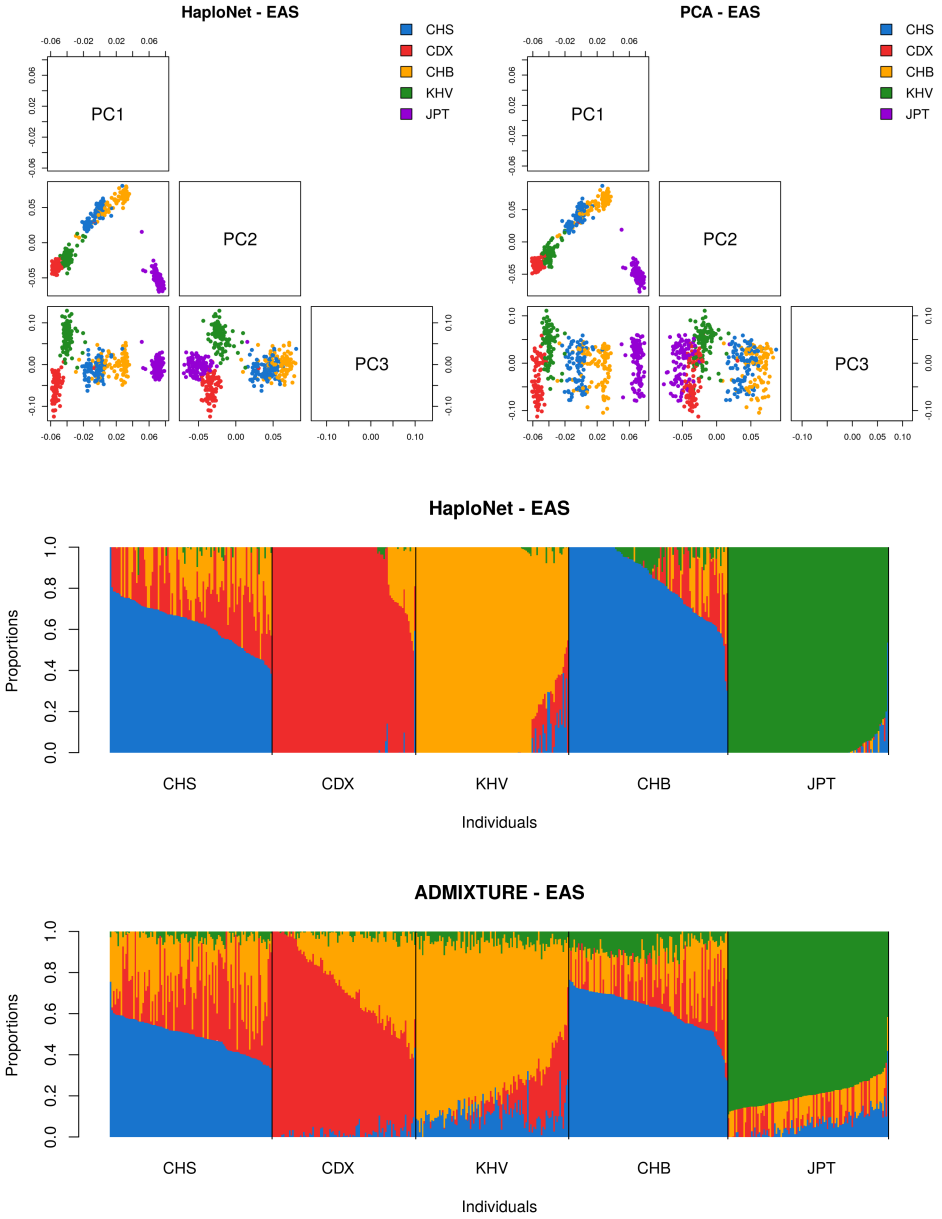

Figure S7: Global population structure inferred for the East Asian (EAS) super population in the 1000 Genomes Project. Top left figure displays the pairwise plots of the top three inferred principal components from `HaploNet`, while the top right figure displays the results of standard PCA. The two bottom figures shows the estimated ancestry proportions of `HaploNet` and `ADMIXTURE` for $K = 4$, respectively.

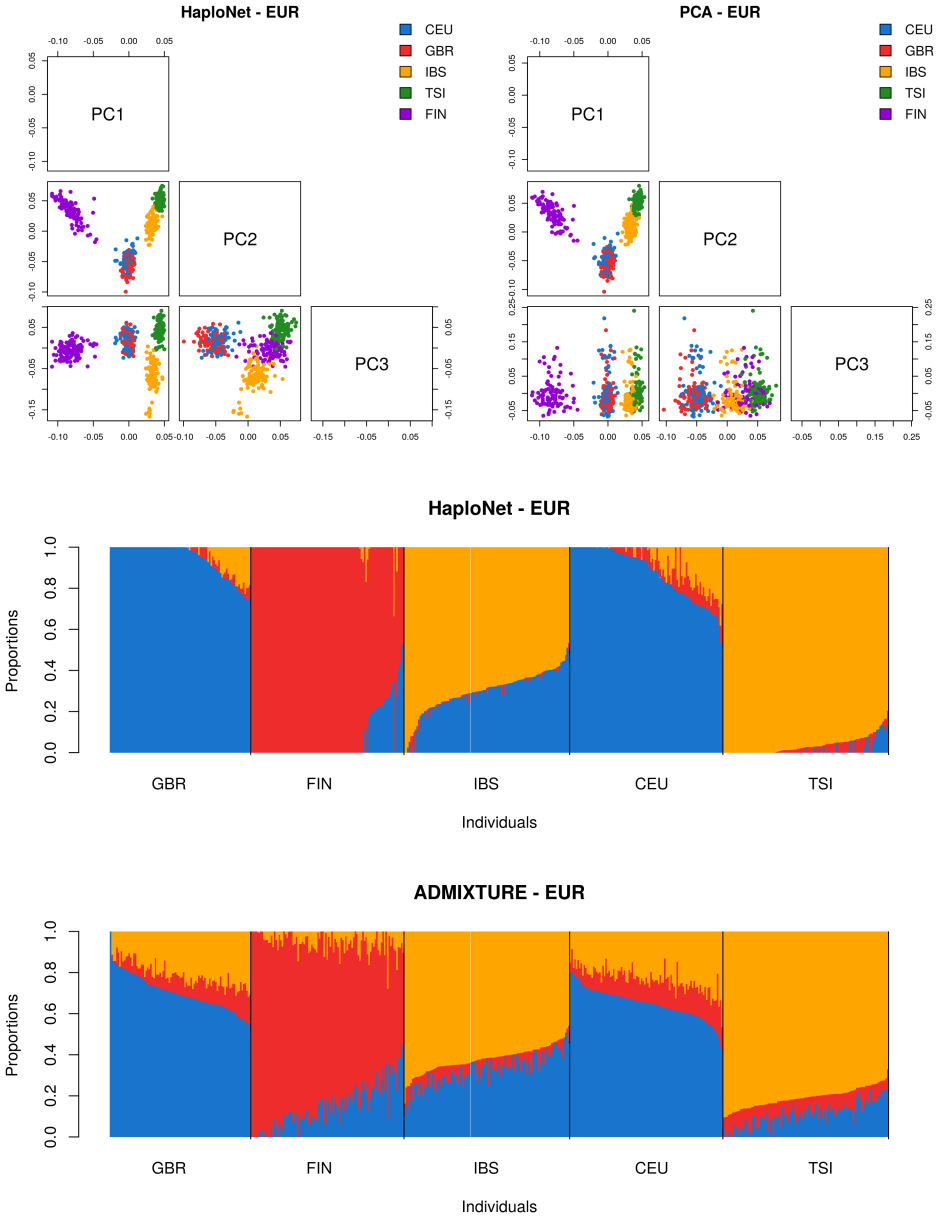

Figure S8: Global population structure inferred for the European (EUR) super population in the 1000 Genomes Project. Top left figure displays the pairwise plots of the top three inferred principal components from `HaploNet`, while the top right figure displays the results of standard PCA. The two bottom figures shows the estimated ancestry proportions of `HaploNet` and `ADMIXTURE` for $K = 3$, respectively.

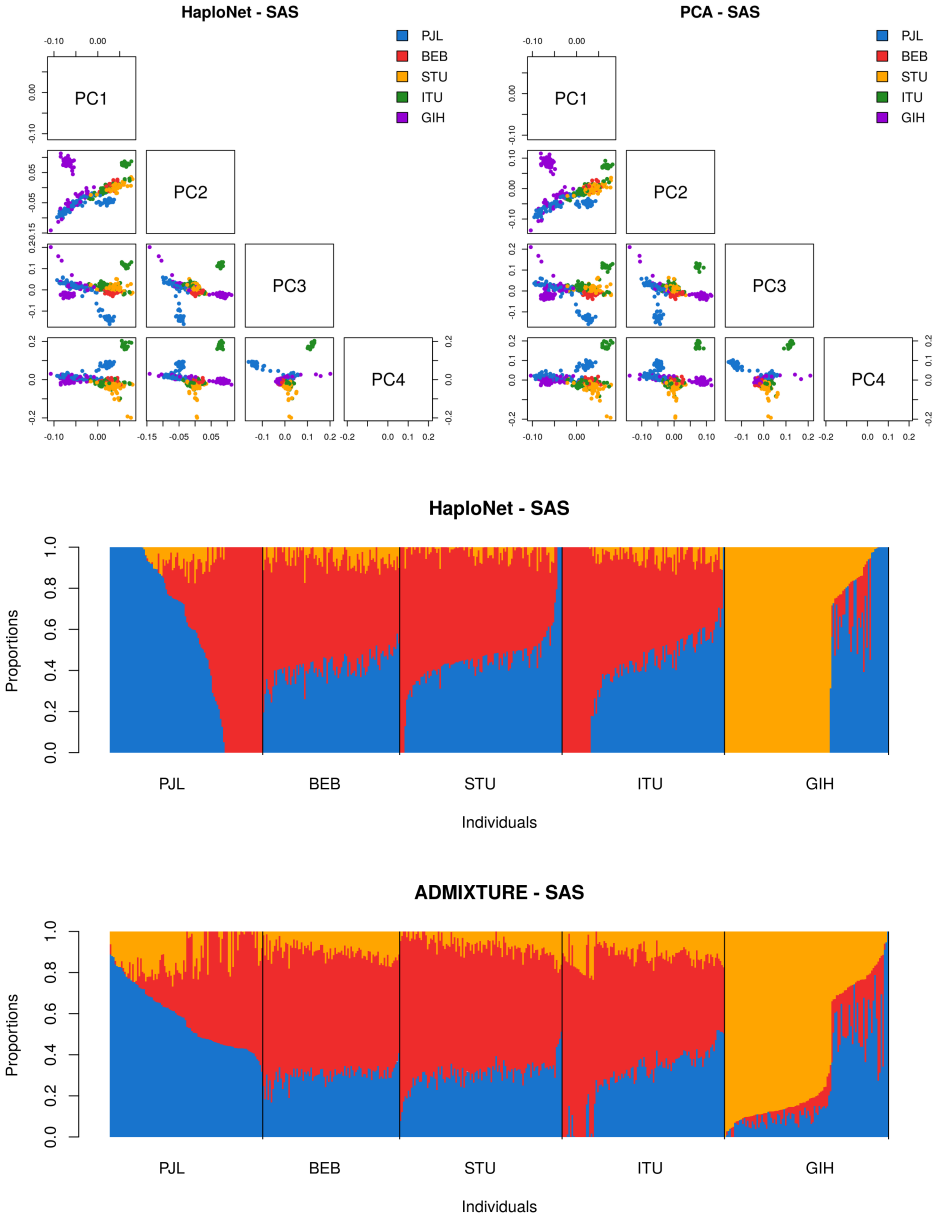

Figure S9: Global population structure inferred for the South Asian (SAS) super population in the 1000 Genomes Project. Top left figure displays the pairwise plots of the top four inferred principal components from `HaploNet`, while the top right figure displays the results of standard PCA. The two bottom figures shows the estimated ancestry proportions of `HaploNet` and `ADMIXTURE` for $K = 3$, respectively.

Window size in `HaploNet`

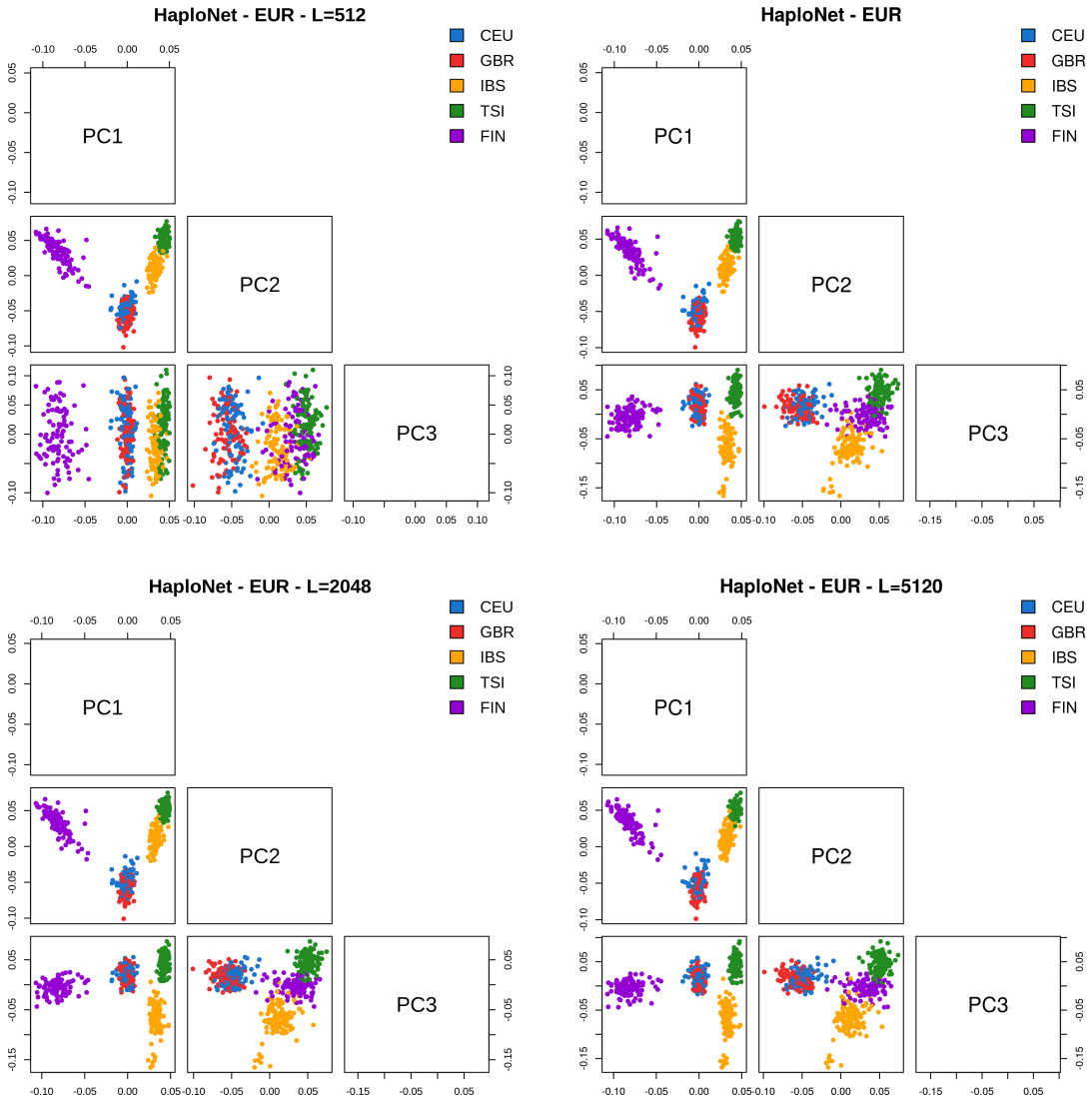

Figure S10: Inferred global population structure using PCA by `HaploNet` with different window sizes (512, 1024, 2048, 5120) in the European super population (EUR).

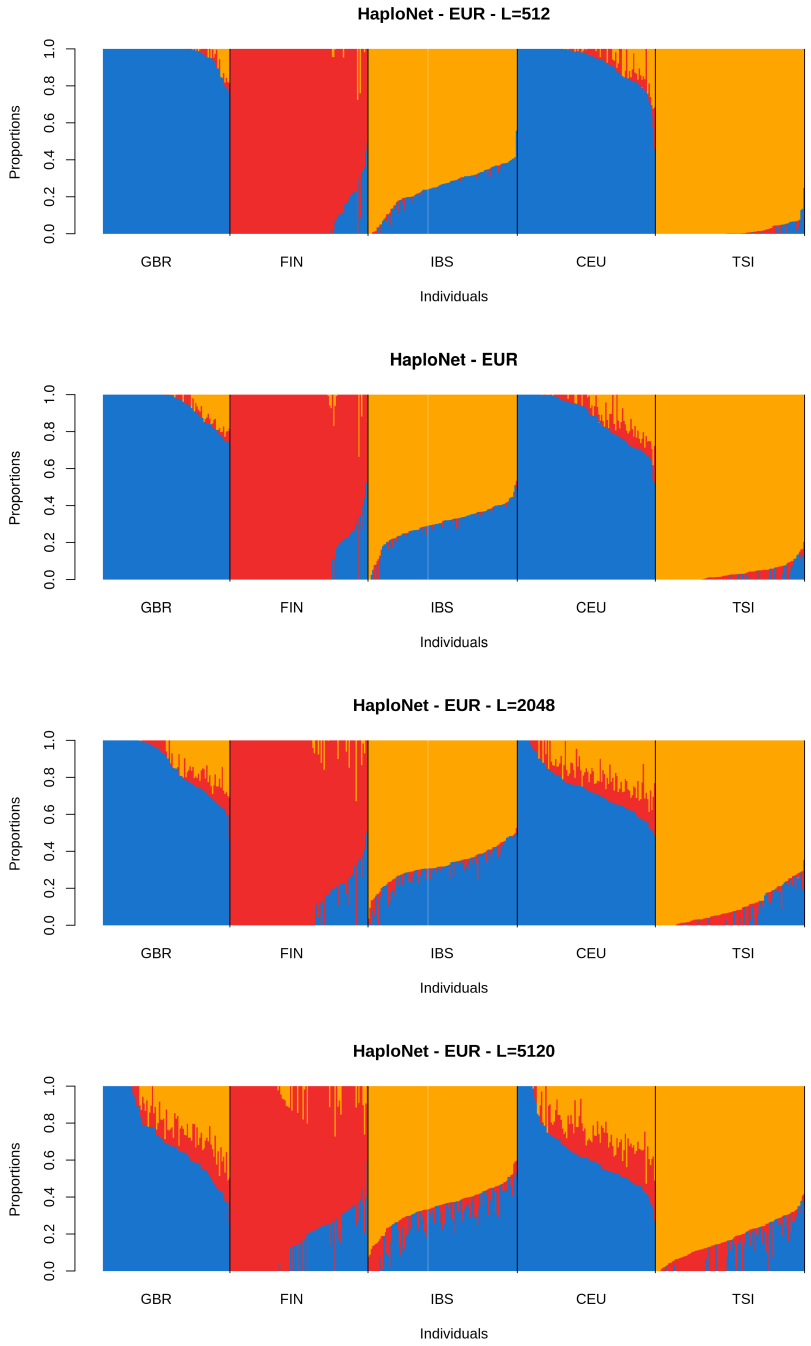

Figure S11: Estimated ancestry proportions by `HaploNet` using different window sizes (512, 1024, 2048, 5120) in the European super population (EUR) for $C = 3$.

1000 Genomes Project Omni-chip sites

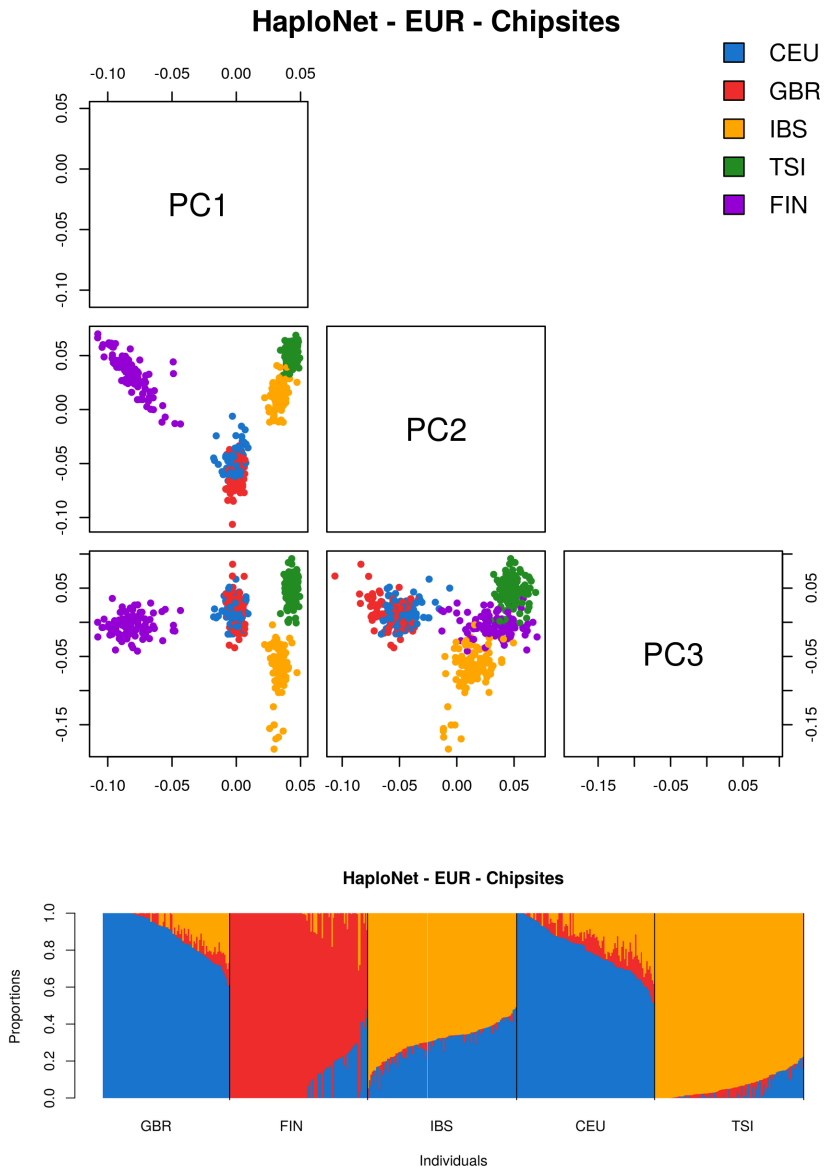

Figure S12: Inferred population structure in the European super population (EUR) using only sites of the high density genotype chip of the Omni platform in the 1000 Genomes Project. First row shows pairwise plots of the three top PCs inferred using `HaploNet`, while the second row shows the estimated ancestry proportions by `HaploNet` for $K = 3$.

Reconstruction and simulation

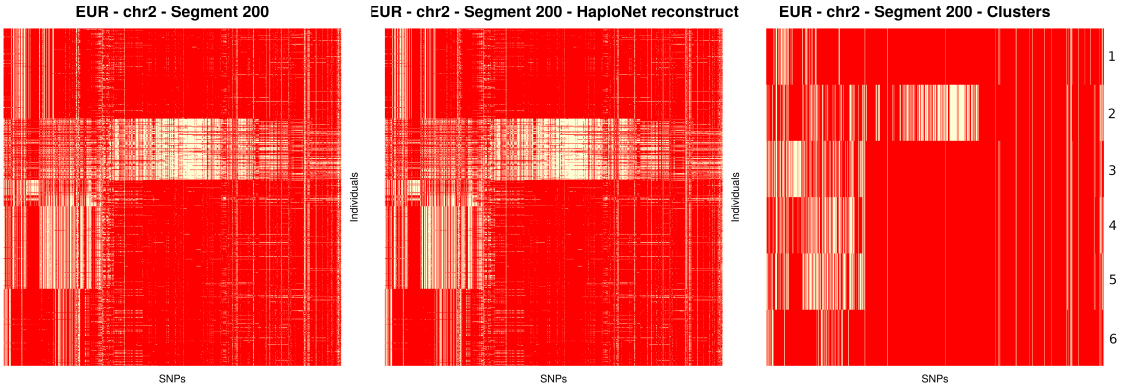

Figure S13: The 200th window in chromosome 2 for the European super population (EUR). Left plot shows the true haplotype matrix with haplotypes ordered by their inferred haplotype cluster in this window by `HaploNet`. Middle plot shows the reconstruction of the haplotype matrix by `HaploNet`, while the last plot shows the reconstructions from the mean haplotype encoding of each haplotype cluster.

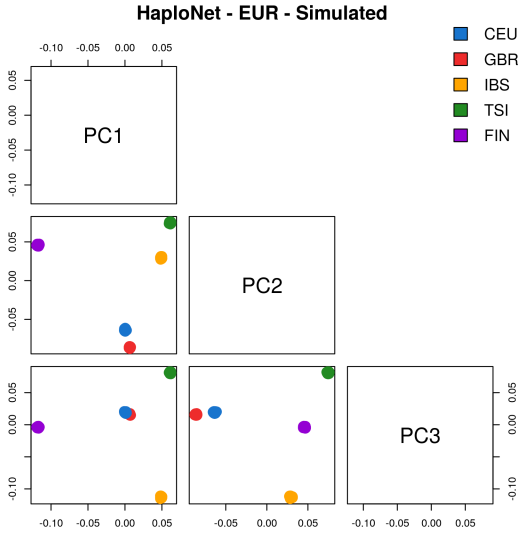

Figure S14: Inferred population structure of genomes generated from sequences of haplotype clusters. 50 haplotypes have been generated from each population in the European super population (EUR).