OXFORD GENETICS

Dimensionality Reduction of Genetic Data using Contrastive Learning

Filip Thor* and Carl Nettelblad

Division of Scientific Computing, Department of Information Technology, Science for Life Laboratory, Uppsala University, Uppsala, Sweden

*Corresponding author: Department of Information Technology, Uppsala University, 752 37, Uppsala, Sweden. Email: filip.thor@it.uu.se

Abstract

We introduce a framework for using contrastive learning for dimensionality reduction on genetic datasets to create PCA-like population visualizations. Contrastive learning is an example of a self-supervised deep learning method that uses similarities between samples to train the neural network to discriminate between samples. Much of the advances in these types of models have been made for computer vision, but many of the heuristics developed do not translate well from image to genetic data. We define a loss function that outperforms other basic loss functions used in contrastive learning in our experiments, and a data augmentation scheme tailored specifically towards SNP genotype datasets.

In our experiments, our methods outperform PCA in terms of population classification. It is on par with t-SNE, while also providing greater generalization properties to unseen and missing data. A strength of the deep learning framework is the possibility of projecting new samples using a trained model, and the ability to incorporate more domain-specific information in the model. We show examples of population classification on two datasets of dog and human genotypes.

Keywords: Deep learning; Machine learning; PCA; Dimensionality reduction; Population genetics

Introduction

With the advent of next-generation sequencing techniques, the number of genomes sequenced annually is increasing exponentially as the cost continues to drop (Stephens *et al.* 2015). The continued expansion in available data has made analyzing genetic datasets using deep learning (DL) approaches more attractive than before due to their dependence on the availability of large and rich datasets. Neural networks (NN) have been employed with great success in many fields, but have to date only seen limited use and success in genomics.

In this work, we are concerned with dimensionality reduction of genotype data. Dimensionality reduction methods have been broadly used for analyzing population structure by giving a way to visualize similarities and relatedness of populations. Existing tools have helped bridge gaps in historical knowledge and data on, e.g., human migration and admixture events (Lazaridis *et al.* 2014). It has also been used in breeding programs, linking current lines to historical lines and landraces (Pincot *et al.* 2021). We consider single nucleotide polymorphism (SNP) genotype data, and transforming the high-dimensional SNP vectors down to two dimensions yields an understandable visualization of the individuals in a dataset. Classically, this has been done using principal component analysis (PCA), which is a method that transforms the data using orthogonal linear combinations of the inputs such that the variance of the data is maximized in the first dimensions (Pearson 1901). PCA transformations of genetic variant data have been shown to correlate to the geographic origin of samples from natural populations (Novembre *et al.* 2008). PCA is a linear model, and a trained PCA embedding can be used to embed new samples with reference to the training population. This has, for example, been used to project ancient samples onto an embedding trained on modern individuals to investigate the origin of ancient samples (Rodríguez-Varela *et al.* 2023).

Other contemporary methods of dimensionality reduction have become more popular for visualization of population structure.Examples of this are t-distributed stochastic neighbor embedding (t-SNE) (van der Maaten and Hinton 2008) and Uniform Manifold Approximation and Projection (UMAP) (McInnes *et al.* 2018). They are both neighbor-based graph learning algorithms that attempt to find a low-dimensional embedding that has topological structures similar to the high-dimensional vectors and specifically try to preserve local relationships in the data. One weakness of these methods is that projecting new samples onto an embedding is not a straightforward task.

However, recently, the use of UMAP and t-SNE in visualizing population structure in genetics data has sparked controversy and faced community criticism. While they can group samples with similar features they, unlike PCA, do not preserve an interpretable distance between samples in the embedding. This may distort the data, and fail to capture admixture events in a representative way. Also, t-SNE performs well in identifying and displaying local similarities in genetic samples, but often fails to illustrate the global relationships in the data. Like PCA, it considers each marker position in isolation, which means that it can not capture linkage patterns. We want to emphasize that these are different ways of visualizing the data, and the notion of one

method being the best is determined by which characteristics of the data we want the embedding to retain.

Another approach to dimensionality reduction, and the one we take here, is using neural networks. Using NNs in a genomic setting is not unique, and is getting more traction lately, including in applications to analyze population structure.

Most of the existing work on using neural networks to visualize population structure utilizes what are called autoencoders (Ausmees and Nettelblad 2022; Battey *et al.* 2021). These models are trained to encode the input SNPs into a lower dimensional representation, and from this encoding reconstruct the input. We build on the work by Ausmees and Nettelblad (2022), and use convolution layers in the neural networks which in theory enable our models to capture linkage between base pair coordinates.

In this work, we propose the use of contrastive learning. This method also encodes the data into a lower dimensional space but is not trained to reconstruct the input. The model is instead trained using a loss function applied directly to the encoding space, where the loss function considers relations between different samples in their embedding coordinates. Key components here are specialized loss functions, and data augmentation (Chen *et al.* 2020). Augmentation has proven important in many applications of deep learning, including computer vision, and we apply similar techniques in an appropriate way to genomic data. One previous attempt to use contrastive learning on genomic data used a shallow network with two hidden layers and an augmentation scheme that consisted of changing some homozygous variants to heterozygous (Ubbens *et al.* 2022). Inspired by this, we adopt a version of their marker flipping strategy in this study. In addition, we use a more complex NN architecture designed to capture linkage, an augmentation scheme based on missing data, and a new loss function.

When analyzing population structure, it is desirable that an embedding will reflect both global and local structure well. If we can retain much of the vital information in two dimensions, we could assume that a higher dimensional embedding dimension would retain more information. This is the goal for the types of models we develop – we want a foundational model trained to understand the genomic structure of the organism under consideration. With a trained model that can produce a high dimensional embedding, one can employ techniques like fine tuning and transfer learning to tailor the model to the application at hand, whether that be, e.g. population structure analysis, identification of causal variants, or genomic prediction.

We know that the position of variants in the genome is important, and linkage information is not captured in methods like PCA or t-SNE since they consider each variant in isolation. A NN model that incorporates convolutional layers can, in theory, capture linkage patterns and epistatic effects between variants.

Materials and Methods

This section introduces the deep learning setting and the main components of our contrastive learning framework. We also describe the datasets analyzed and the performance metrics used.

Deep Learning

In a general machine learning setting, we consider some dataset $\mathcal{D} = \{\mathcal{X}_i, \mathcal{Y}_i\}_{i=1}^N$ that we want to study, which consists of some samples $\mathcal{X}$, and labels of interest $\mathcal{Y}$. We then want to find the underlying relation that dictates the mapping $f : \mathcal{X} \to \mathcal{Y}$. The approach taken to understand this relation is dependent on the problem setup. We might use *supervised learning*, a case where we assume knowledge of labels $\mathcal{Y}$ on some training set, and want to find a generalizing function $\hat{f}$ that approximates f. An example of this is genomic prediction, where given genotypes $\mathcal{X}$ we want to predict phenotypes $\mathcal{Y}$. Or, we might want to use *unsupervised learning*, where we assume no knowledge of the training labels. An example here would be dimensionality reduction, akin to using PCA to study population structure. Here we only use the genotypes $\mathcal{X}$ and hope that the model can reveal insights into the population structure When labels $\mathcal{Y}$ are available, the level of colocation of samples with identical labels in the resulting embedding can serve as a proxy for embedding quality.

With deep learning, we mean a specific set of machine learning models called deep neural networks. Deep NNs are a family of models with a compositional structure, where information is fed through and transformed in what are called *layers*. They sequentially process the data by taking linear combinations of the output of the previous layer and explicitly introduce nonlinearities through activation functions. This yields a non-linear model with an explicit input-output mapping.

There have been several attempts at creating NN models for visualizing population structure, two examples are popvae (Battey *et al.* 2021) and GCAE (Ausmees and Nettelblad 2022). Both are *autoencoder* architectures. This means that the neural network consists of an encoder network, which is a function $f_E : \mathbb{R}^D \to \mathbb{R}^d$ that takes the full $D-$dimensional input SNP vector, and maps it to a $d-$dimensional space, called the embedding, or latent space, which is used for the population visualization. The embedding is then transformed by a decoder network $f_D : \mathbb{R}^d \to \mathbb{R}^D$, mapping the embedding coordinates back into the $D-$dimensional space. The autoencoder network can be seen as the composition of the two networks $f_{AE} = f_D \circ f_E$ and is trained to reconstruct the input. That is, we use a loss function that captures the discrepancy between the input x and the output $\hat{x} = f_{AE}(x) = f_D \circ f_E(x)$ produced by the autoencoder. One example of a loss function used in this case is the mean squared error between the true genotypes and the reconstruction. The main motivation for why the embedding dimension of an autoencoder would be a good representation of the population structure is that to reconstruct the input accurately, the lower dimensional embedding acts as a bottle-neck, and is required to extract the most distinctive, or representative, information in the data (Goodfellow *et al.* 2016). This is therefore in a way an indirect way of letting the model know what relations to learn. An illustration of the autoencoder concept is shown in Figure 1 a).

With contrastive learning, the architecture is altered slightly. Instead of training the model to reconstruct the input, the model consists only of the encoder network f_E and uses a different loss function that is applied directly to the embedding coordinates $z = f_E(x)$. Figure 1 b) shows an illustration of the concept behind the contrastive learning framework.

Both the autoencoder and contrastive learning are part of a class of models that are referred to as *self-supervised learning*. They utilize completely unannotated data, utilize no information on the data labels we are interested in during training, which are only used in evaluation of the clustering performance. They rather define their own relations or rules from the data that the model is trained to adhere to. We now turn to the details of the contrastive learning framework we developed for this study.

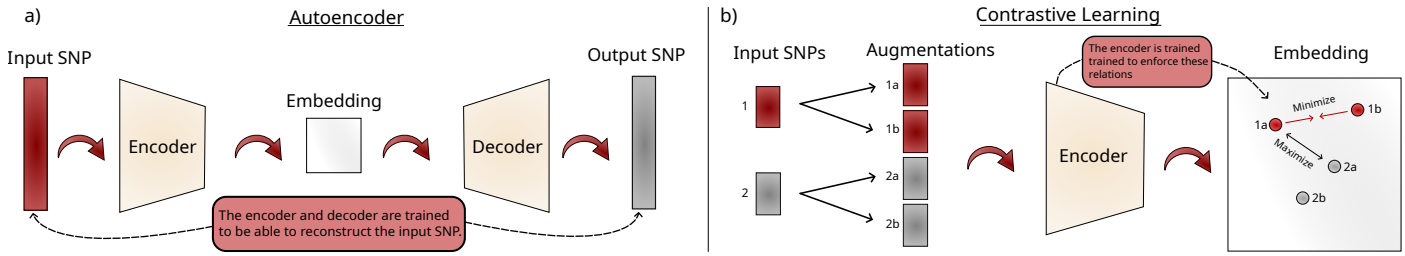

Figure 1 a): Concept of an autoencoder architecture. The NN model is trained to reconstruct the input. The latent space, or embedding, acts as a bottleneck and is used to visualize the population. b): Concept of architecture used for contrastive learning. It utilizes an encoder similar to the autoencoder, but does not attempt to reconstruct the data. The model is trained by comparing embedding coordinates of different samples. In both models, the Embedding coordinates is what is used as the population visualization.

Contrastive Learning

This section introduces the details of the contrastive learning implementation that we propose.

Anchor, negatives, positives In conventional supervised learning, even if samples are considered together in batches, there is no within-batch interaction between samples. In contrastive learning, however, samples and their embeddings are explicitly compared, and the network weights are updated based on the relationships of the embeddings of the samples considered.

The loss functions in contrastive learning are distinct from the commonly used ones in supervised learning in the way that they for each loss contribution consider samples to take on one of three different roles: *anchor*, *positive*, and *negative*. These will be denoted as z, z^+, and z^-, respectively. In some loss functions, we have several negatives per anchor denoted by the set $\{z_i^-\}_{i=1}^{N-1}$. For brevity in the coming equations, we denote this set of embeddings as $Z = \left\{z, z^+, \{z_i^-\}_{i=1}^{N-1}\right\}$ When computing the loss function of a batch, each sample will contribute a term to the total loss where it acts as the anchor. In general, we want the loss functions to train the network to map the anchor closer to the positive than to the negative. The coming sections will describe in detail the loss function we propose, and how the positive and negative samples are defined or chosen.

Loss function One basic contrastive loss function is the Triplet loss function (Schroff *et al.* 2015), which uses only one negative, and has the form

$$\mathcal{L}_{triplet}(z, z^+, z^-) = \max\left(|z - z^+| - |z - z^-| + \alpha, 0\right). \quad (1)$$

That is, if the distance from the anchor to the positive is larger than the distance from the anchor to the negative (with a margin of α), we get a non-zero loss contribution, and the network weights will be updated using the gradient of the loss function in a way that attracts the positive and repel the negative from the anchor. Figure 2 a) shows the distances that are considered when computing the triplet loss. If we get a zero loss, the triplet $\{z, z^+, z^-\}$ are deemed to be in a configuration that is good enough to not be corrected further.

The triplet loss illustrates the concept well but has been improved and generalized by comparing the anchor to multiple negative samples. This is referred to as the *N-pair* loss (Sohn 2016), and is defined in Equation (2) as

$$\mathcal{L}_{N\text{-}pair}(Z) = \log\left(1 + \sum_{i=1}^{N-1} \exp(z^\top z_i^- - z^\top z^+)\right). \quad (2)$$

Here, we have an expression resembling a softmax loss, which gives a larger loss contribution if the distance from the anchor to a negative is small, and weights the contribution more. Another detail here is that instead of using a distance metric like Euclidean distance, the similarity of the samples is measured by their inner product. This inner product is taken in the coordinate system defined by the global origin $(0, 0)$, which is illustrated in Figure 2b).

We saw that using the basic triplet loss as described in Equation (1) would yield embeddings that failed to discern population structure, with limited discriminative power and a blurry visual impression. Also, the more intricate losses, similar to the N-pair loss in Equation (2), seemed to benefit a lot in terms of training stability from using some specific model alterations, for example, forcing the output to have unit length by applying L_2 normalization. This was not feasible in a case where we want to obtain 2-D visualizations. This led us to another, more localized approach we denote the centroid based N-pair loss function, and the use of map projections, described in a later section.

Centroid-based N-pair Loss We propose a centroid-based version of the triplet and N-pair losses, where the contribution of one anchor sample to the total loss is computed in a new reference frame for each anchor-negative pair.

Instead of computing the inner products with respect to the origin, it is performed in a new reference frame which consists of simple translation. For each negative sample, a new centroid C_i is computed, which the vectors will be defined in reference to when computing the inner products. The new vector will simply be shifted by the centroid, e.g., $z_{C_i} = z - C_i$. Since we would assume that in general the positive sample will be embedded closer to the negative, we define the centroid as

$$C_i = \frac{z + 2z_i^- + z^+}{4}. \quad (3)$$

That is, the mean coordinates of the triplet, but with the negatives weighted twice. When distances grow large, the loss function will decrease rather quickly. To control the magnitude, we also scale the vectors by the magnitude of the largest vector, computed as

$$\mu_i = \max\left(||z_{C_i}||_2^2, ||z_{C_i}^+||_2^2, ||z_{C_i}^-||_2^2\right) \quad (4)$$

Figure 2 shows illustrations that highlight the differences between the three approaches to computing the similarity used in triplet, N-pair, and our centroid version, respectively.

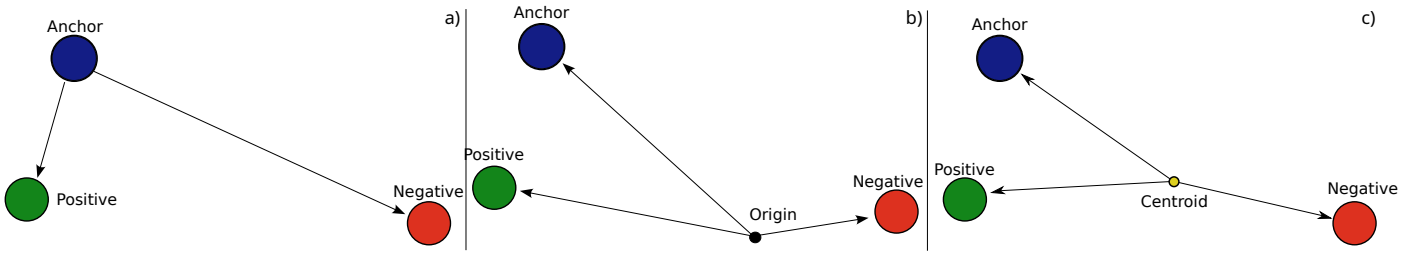

Figure 2 Illustration of how samples are compared in the different contrastive loss functions. a): Triplet loss b): N-pair loss c): Centroid based N-pair loss. A new centroid will be computed and used for each triplet within each batch of samples.

The centroid-based loss function is based on the N-pair loss formulation in Equation (2), with minor changes. If we introduce the following notation, we can make the expression more succinct. Let

$$\tilde{z}_i = \frac{z - C_i}{\sqrt{\mu_i}}, \tag{5}$$

then the centroid-based N-pair loss can be expressed as

$$\mathcal{L}_{centroid}(Z) = \log\left(1 + \sum_{i=1}^{N-1}\left(\frac{\exp\left(\tilde{z}_i^\top \tilde{z}_i^-\right)}{\exp\left(\tilde{z}_i^\top \tilde{z}_i^+\right)} - e^{-2}\right)\right) \tag{6}$$

The final term inside the sum, subtracting by e^{-2} comes from the way the centroids are defined. The "best case" configuration of the samples would be that the anchor and positive sample would get mapped very close compared to the negative. We want this to give a loss contribution of zero. The definition of the centroid in (6) and the scaling in (5) gives that $\tilde{z}_i^\top \tilde{z}_i^- = -1$ and $\tilde{z}_i^\top \tilde{z}_i^+ = 1$, and thus $\frac{\exp(\tilde{z}_i^\top \tilde{z}_i^-)}{\exp(\tilde{z}_i^\top \tilde{z}_i^+)} = \exp(-2)$ in that case. This constant offset from zero is subtracted to keep the loss unbiased with respect to the centroid definition.

The total loss for a batch consists of summing the loss contributions computed as in Equation (6) using each sample in the batch as the anchor once. The coming sections will describe how the positive and negative samples are chosen.

Augmentations as positives In computing the loss function, each sample in a batch will act as the anchor once. For each of these anchors, we need to assign positive and negative samples. We want a positive sample to have a different input signature to the anchor while having the same label. That is, we want the positive to be distinct from the anchor but be but be recognized by the model to be related. An ideal positive sample would be another individual in the same population as the anchor. They will be distinct individuals, but share similar genetic profiles. In a supervised setting this sample could be another sample in the batch which shares the same or has a similar label as the anchor, but even when labels are available, providing them to the model might introduce unwanted bias. A common choice is to generate a new sample to use as the positive through data augmentation (Chen *et al.* 2020). For example, one study used a random flipping of homozygous sites to heterozygous (Ubbens *et al.* 2022).

There is untapped potential regarding the choice of augmentation scheme since it allows us to incorporate domain-specific knowledge. We want to alter the data in a way that should not affect the label information in a meaningful way. The literature is rich on the topic of data augmentation in computer vision for image classification (Shorten and Khoshgoftaar 2019; Cubuk *et al.* 2019). It has been indicated that contrastive learning benefits specifically from strong data augmentation (Chen *et al.* 2020). However, augmentation strategies used in image analysis can not be applied directly to genotype data. There are certain properties that can be exploited when it comes to images that do not translate to vectors of genotypes at pre-determined SNPs. Images are generally rotational and translationally invariant – a slightly shifted and rotated image should still retain its label information. This does not hold for SNP data. Carrying an alternative allele in one SNP is a completely different thing from carrying an alternative allele in the next SNP within the set of positions used. Another common augmentation for images is mirroring it along an axis, which would be unsuitable for the same reason.

We propose an augmentation scheme that is based on missing data. In genotyping, some of the samples will have variants with low confidence and may be marked as missing. One way to deal with this is to impute the data to try to get a more complete dataset. Due to the versatility of the deep learning approach we can utilize the concept of missingness. We build upon work where the input to the network is the full SNP vector with missing values set to -1, and add a mask channel that has the same dimensions as the SNP vector and is 0 everywhere, except at the missing SNP positions where it is set to 1 (Ausmees and Nettelblad 2022). This was introduced to act as a regularization technique, setting a fraction of the data as missing during training, to increase generalization performance.

Here, we use missingness as a form of data augmentation. Given a SNP vector with no missing data, masking parts of the vector should not change the label. The quality of the data will be worse with lower overall information coverage, but behind the masking, it is still the same individual. We want a model that is robust to missing data, and masked versions of the same sample should be recognized as being similar. Depending on the masking rate p_{mask}, different samples with similar genetic makeup will be more likely to have similar augmented versions than samples of other genetic origin. We also adopt the approach taken in (Ubbens *et al.* 2022), where some variants are randomly changed. This is done by increasing or decreasing the allele count by 1 in some positions. That is, each SNP will with a probability p_{flip} be flipped from $0 \rightarrow 1$, $2 \rightarrow 1$ and $1 \rightarrow 0$ or $1 \rightarrow 2$. Finally, we apply a one-hot encoding to the augmented SNP vector before feeding it to the first layer of the NN. An illustration of the data augmentation strategy is shown in Figure 3.

Each sample will be augmented anew each epoch, which means that the network will most likely never see the exact same sample twice during training, since it will be given new augmentations each time. Practically, the augmentation is implemented

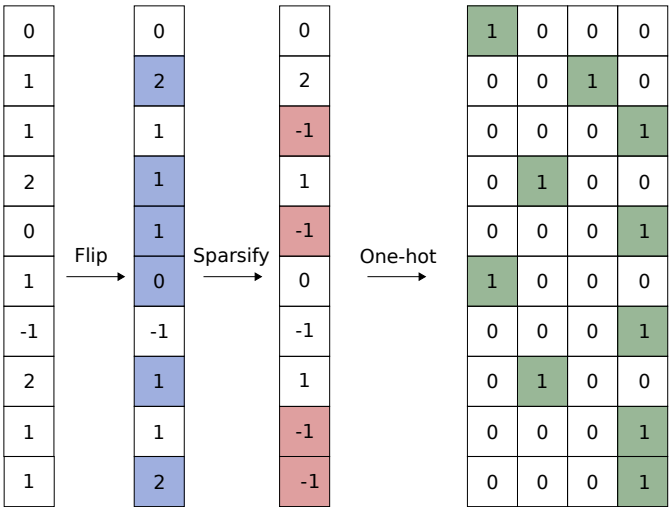

Figure 3 Illustration of the augmentation strategy which consists of first flipping some variants with probability p_{flip}, then masking with probability p_{mask}. Shown is an example of one sample. This procedure is done for all samples and redone every batch.

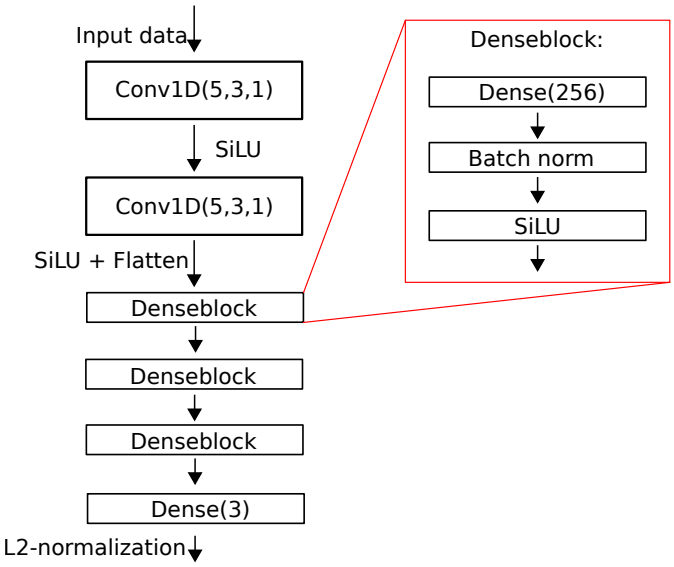

Figure 4 Encoder model architecture used in the experiments, which transforms the data from the high dimensional SNP vector, down to a 3-dimensional embedding on the unit sphere.

as the first layer of the NN model for efficient GPU usage.

Choice of negatives We consider an unsupervised setting, i.e., we assume no knowledge of the true labels of the samples. Ideally, we would choose samples that are of a different class than the anchor but also lie close to it. Since we can not use label information, we can not guarantee that the negatives used are actually of a different class. There are some attempts to remedy this, while a common choice is to pick negatives at random (Robinson *et al.* 2020).

We have investigated several options, and have seen that choosing samples randomly but drawn using weights according to the inverse embedding distance to the anchor has yielded the best results.

Model Architecture

The model used consists of 2 convolutional layers with 5 filters, a kernel size of 3, and a stride of one. Then after flattening the output from the convolution, follows 3 dense layers with 256 units, with batch normalization. The sigmoid linear unit (SiLU, also known as Swish) activation function is used after each weight layer, and is defined as $\sigma(x) = \frac{x}{1+\exp(-x)}$. SiLU is essentially a smoothed version of the commonly used ReLU activation function. The model ends with a dense layer with three units, and a L2-normalization, resulting in samples getting mapped to the unit sphere in 3D. Figure 4 illustrates the model architecture.

In comparison to an embedding mapping to a regular 2D surface, samples will now always have neighbours to compare to in all directions. The risk of a cluster, or a population, being regarded as an outlier group or genetically distinct from other samples is decreased. Visualizing a spherical surface in two dimensions is a well-studied problem, driven by the need to visualize the world map. Such a visualization can not retain all aspects of the 3D map, and some properties will be sacrificed. We use the Equal Earth map projection (Šavrič *et al.* 2019). It is a projection that retains relative areas but does introduce some shape distortions, mainly at the poles. Figure S1 in the Supplementary Materials illustrates the transformation from 3D to 2D. The projection need to made with respect to a coordinate system in which the sphere is defined. The sphere can be rotated arbitrarily before applying the map projection, and how this affects the embedding, and the corresponding classification performance can be seen in Figure S2.

Some datasets might have a population from which others have originated, and thus the population structure could be considered to have an origin, which the model should be able to capture. However, we do not want the model to assume this a-priori.

The motivation for using convolutional layers is to enable the model to catch local patterns over the input. This is in contrast to t-SNE and PCA which consider each SNP location in isolation, and thus lack knowledge about the relative SNP position in the genome, which we know is important information.

More intricate model choices have been explored, including residual and locally connected layers, but with model complexity came increased training difficulties. A more thorough model optimization is left for future study.

For all experiments shown, the model has been trained using the Adam optimizer (Kingma and Ba 2014) with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, with the learning rate decaying by 0.99 every 10 iterations, and trained for 5000 epochs. The masking and flipping rates p_{mask} and p_{flip} are drawn anew for each training iteration, and are unique for each sample. They are both drawn from a uniform distribution between 0 and 0.99, $\mathcal{U}(0.01, 0.99)$, meaning we have a relatively aggressive augmentation strategy. As negatives for each anchor, we use all of the other samples.

Technical Details, Code, and Data Availability

The code written for this project is openly available at https://github.com/filtho/ContrastiveLosses. This work introduces no new data, but focuses on methodological work. The data used in the examples are openly available from the corresponding studies. The software is written in Python, and TensorFlow

(Abadi *et al.* 2015) version 2.12. The PCA and t-SNE embeddings are produced with SciKitLearn version 1.3.1 (Pedregosa *et al.* 2011)

Results

Datasets

We will focus on two different datasets in this study. A dataset consisting of 1,355 samples of 161 different dog breeds, grouped into 23 clades. Each sample has 150,067 SNPs (Parker *et al.* 2017). The only filtering and preprocessing was to remove sex chromosomes, resulting in the samples having 146,330 SNPs. The variants are coded as 0, 1, and 2 for homozygote reference, heterozygote, and homozygote alternate, respectively.

We also consider the Human Origins dataset (Lazaridis *et al.* 2014), and filtered in the way described in (Ausmees and Nettelblad 2022), which consists of removing sex chromosomes, non-informative sites, and applying a MAF filter of 1%. This gives a final dataset with 2,067 samples and 160,858 variants.

We are training the model using augmentations based on missing data, implicitly training the model to be robust to discrepancies or patterns in the dataset related to how the data was acquired. We devised an experiment to investigate this. Say that we have a densely genotyped dataset, and a subset that has been genotyped by more affordable SNP chips with a smaller marker set. Ideally, we want to utilize the many SNPs available for most of the samples, while not capturing patterns associated only with the SNP-chip used.

To simulate this we look at the Human Origins dataset, keep 80% of the samples as-is, and divide the remaining 20% of samples into four groups. These groups each have a unique collection of markers that will represented as missing values for every sample within the same group. In this experiment, 20% of the total marker count is masked, chosen randomly for each group. This means that the four groups will have the same missingness pattern, mimicking a case where the same SNP-chip has been used to genotype samples in the same group. We would want these samples to be mapped according to their genetic makeup, not their missingness pattern. The idea is illustrated in Figure 5. The missing samples were then set to the most common SNP in each position, which means that the samples from the different SNP-chips will have some common structure, not necessarily related to the population structure. This may seem like an odd and naive choice, but it is better than leaving them encoded as a 9, or -1, which are common representations. See Figure S4 in Supplementary materials for a comparison of the t-SNE embeddings where missing values are represented as -1, 9, and using the most common genotype (MCG). The corresponding classification performance is shown in Table S2.

Performance metrics

(McVean 2009).

As discussed above, the goal is for these types of models to understand the genomic structure of the individuals, and produce an informative embedding that can be used as a tool for other analyses. Some studies evaluate embedding performance by measuring the correlation between a distance in genotype space and a distance in embedding space, but we argue that this is not a good proxy for how well the embedding will perform in downstream tasks. This is due to most relationships not necessarily being linear with respect to genotypes, and the highly correlated linkage structure of the genome both of which fails

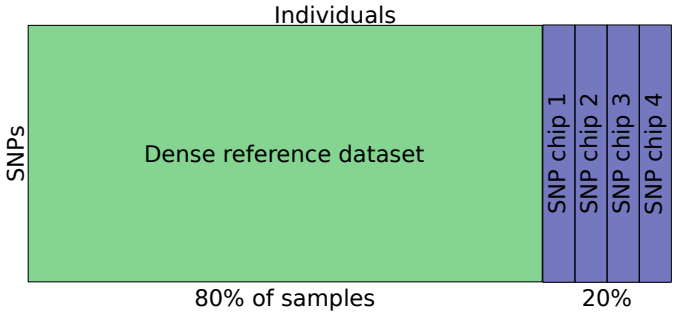

Figure 5 Experimental set-up for the case where we have a dense reference set consisting of 80% unchanged samples, and the remaining 20% are split into four groups. These groups have 20% of their SNPs marked as missing, where the missingness pattern is identical for every sample in the same group. This to mimic samples being genotyped by different SNP chips with lower coverage than the full genotypes.

to be described by simple distance metrics. Also, studies have shown that at least PCA embeddings are sensitive to the sampling distribution of the samples, and the relative distances are affected by the sample sizes of different populations (McVean 2009).

Therefore, the approach we take to measure the performance of the methods is to evaluate the dimensionality reduction based on some downstream task using the embedding coordinates. In this study, we evaluate the embeddings by their population classification performance. Since we would assume that individuals from the same populations would be genetically similar, the performance of a trained population classifier should give an implicit measure of how much of the information has been retained when reducing the dimensionality of the input data.

One such classifier is the k-nearest neighbor (KNN) classifier, classifying a sample as the majority vote of its classes of the k closest samples in the embedding. The predicted class y_i for sample i is computed as

$$y_i = \text{argmax}\Big(\sum_{j \in \mathcal{N}_i^k} \tilde{y}_j\Big), \tag{7}$$

where $\tilde{y}$ is a one-hot encoding of the data label, and $\mathcal{N}_i^k$ is the set of k closest samples in the embedding space. By varying the value of k, we can see how well the methods perform on different scales. When evaluating performance, we report the classification accuracy, that is, the fraction of correctly classified samples using the KNN classifier.

Dog dataset and population classification over different scales

We perform dimensionality reduction using our contrastive learning framework on the dog dataset and compare with other contemporary methods. Figure 6 shows the resulting embeddings using PCA, t-SNE, and contrastive learning using the triplet loss as defined in Equation (1) and the centroid loss as in Equation (6).

Table 1 shows the population classification performance on the dog dataset for the considered methods where PCA and the contrastive learning models have been trained on 80% of the data, and evaluated on the full data set. The values reported under subpopulation and superpopulation accuracy use all samples, and their respective validation accuracies are reported as

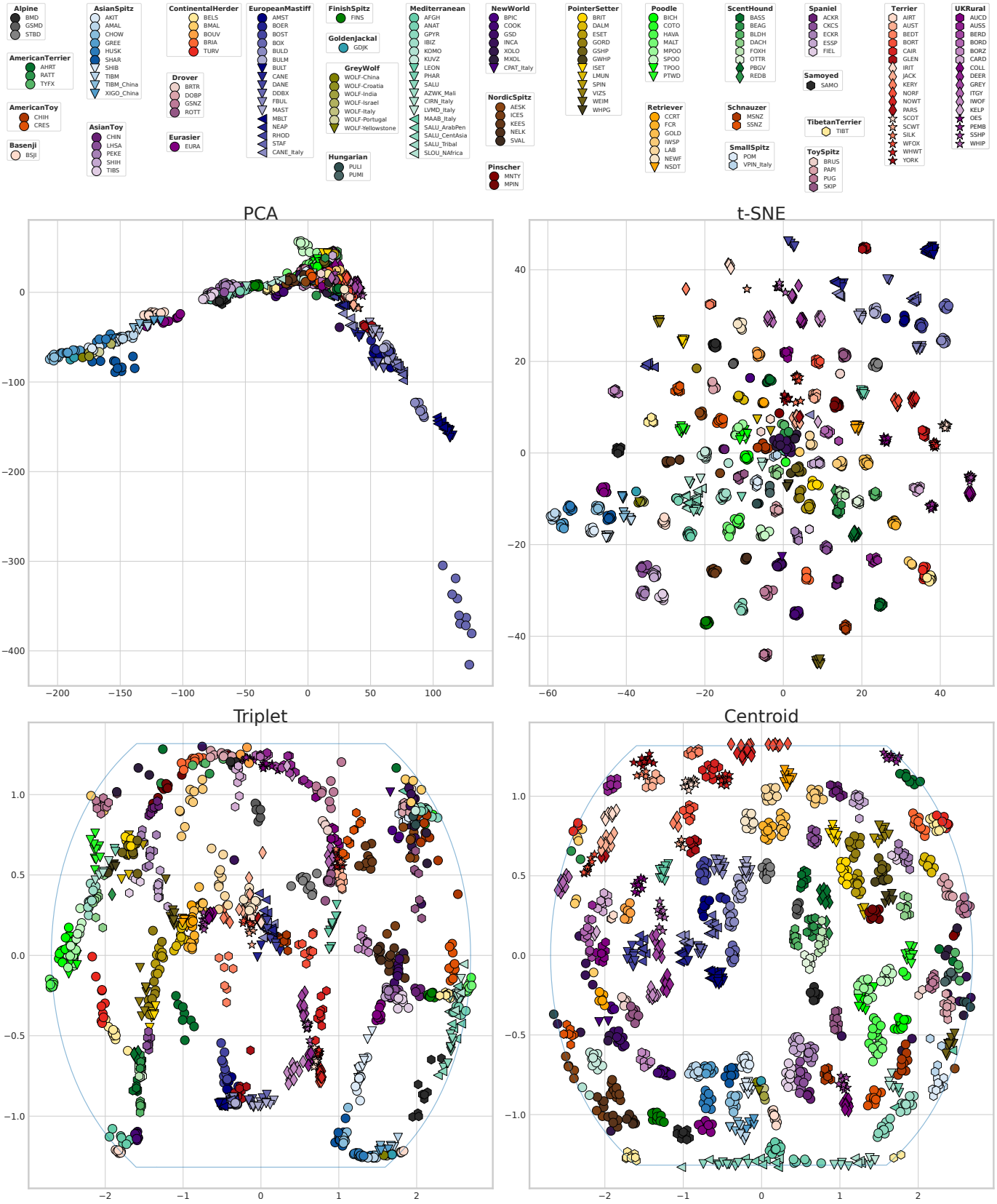

Figure 6 Dimensionality reduction results on the dog dataset using PCA, t-SNE, and contrastive learning using the triplet loss, and our centroid based loss. The legend is shown at the top, where similar colors indicate samples belonging to the same clade. The blue line in the bottom plots indicates the boundary of the map projection of the spherical embedding. Figure S3 in the Supplementary materials shows the resulting embeddings of 3D PCA and t-SNE mapped to 2 dimensions using the Equal Earth projection, and Table S1 shows the classification performance.

well. t-SNE does not have a way to project samples without retraining the model, so t-SNE has seen the validation samples as well, but the KNN model is fitted only with the training samples.

Figure 7 shows how the population classification scores vary depending on how many neighbors the KNN model considers.

Human data visualization and built-in robustness to missing data

Classification accuracies for both cases using a KNN classifier with $k = 3$ is shown in Table 1. Similar to the dog case, we report both the accuracy on the full dataset, and on the validation set when applicable.

Figure 8 compares the embeddings using t-SNE and our method on the masked human data experiment. The top and bottom show the same embeddings, but the top plots are labeled by population, and the bottom plots by the missingness pattern. Here t-SNE produces the embedding by using all samples in training, while our method trains on the reference set, and sees the masked groups only when projecting them, using them as the validation set.

Discussion

Figure 6 shows that both PCA and t-SNE have samples radiating from an origin . This visual cue can enable interpretations that misconstrue the true population. If we have an imbalanced dataset in terms of the number of samples from different populations, samples from less represented populations will tend to be embedded further from the origin than the populations with heavier representation. The coordinates in an embedding will depend on many factors and while sometimes some populations might have diverged at an early point from some others, measuring distances in any embedding can at best serve as a source for such a hypothesis. However, PCA or t-SNE might lead readers to these types of conclusions when some populations are embedded far from the origin, or center of gravity of all samples. Our method enables the embedding to lie more on a continuum – there is no notion of an origin, and each sample will have neighbors in all directions.

For example, in Figure 6 both PCA and t-SNE embeds subpopulations from the European Mastiff clade at the largest distance from the origin. This opens up for invalid interpretations of their status within the study population. They should not be viewed as a peculiar breed of dog, based on this data alone. Our embedding does not have this property, where the European Mastiffs are embedded close together, both within breed and clade, somewhere on the sphere Depending on how the spherical coordinate system is defined, its origin will dictate which samples will lie on the boundary of the embedding after applying the map projection.

Figure 7 shows that, while t-SNE beats our method in peak classification accuracy using a KNN classifier, the contrastive learning model performs better when considering more neighbors in superpopulation classification. This measures how well the methods embed not just the few closest related samples similarly, but how it captures the overall population structure. This indicates the commonly observed good local performance of t-SNE, and its inability to represent the global structure.

For the non-masked cases, t-SNE generally outperforms our method in peak classification accuracy. However, a strength of the NN approach is achieving comparable performance, despite having only been trained on 80% of the samples. Experiments suggest that even when including all samples in training, we still do not surpass t-SNE levels of accuracy. This means that our model can extract sufficient information to perform well on the samples it has not seen in training. This is useful when we have a trained model and want to project new samples onto an existing embedding, which is not a straightforward task for t-SNE.

Using the KNN population label classifier is an attempt at quantifying the performance of the embeddings. This choice is not necessarily the best or only way to evaluate the methods. It is chosen because of the intuition that we would want individuals from the same population to be genetically similar and thus be embedded close to each other. This is however not necessarily true, since it does not consider the relative embedding distances and how it relates to the differences in genetic make-up. There exists no ground truth that we know should hold and can compare to. PCA has often been regarded to have good, linear properties, and the resulting embeddings can reflect the geographical origin for natural populations. Recent studies show that PCA can be biased when visualizing population structure, especially when the dataset has a skewed population label distribution, with some highly represented with many samples, and some less represented with fewer samples (Elhaik 2022).

We observe good generalization properties in two ways. First, in terms of classification accuracy on unseen samples, our models attain higher accuracy on unseen samples than PCA has when applied to the entire dataset entire dataset. Furthermore, the NN model validation accuracy is not much lower than the training accuracy , which suggests that the model does not simply remember the samples it has seen, but can learn more abstract properties mapping similar samples to similar areas on the sphere. The other generalization property is showcased in the experiment on the masked human data. Figure 8 shows that having introduced an artificial but still domain-relevant pattern in the data, t-SNE clusters based on population label but also based on the missingness pattern. This is an undesired artifact in a population structure study. We would want to incorporate as much information available to us as possible, but t-SNE is not able to make proper use of this data. In contrast, our method seems robust to this kind of perturbation in the data and does not visibly cluster on the missingness pattern. Both generalization properties are most likely observed due to the comprehensive data augmentation scheme used in our contrastive learning implementation.

The runtime for the dog dataset on one A100 Nvidia GPU is around 2 seconds per epoch, or around 2.7 hours for the full 5000 epochs, while it takes less than a minute for both PCA and t-SNE. This means that we currently are not competitive when it comes to computational cost. For this reason, we have implemented a distributed framework that can utilize multiple GPUs to process batches in parallel. This means that we can speed up the wall clock time for training significantly given more resources, but this cost is still a drawback of the NN methods.

Having an explicit mapping makes drawing conclusions about how the input affects the output more tractable. An increasing number of applications rely on deep learning (DL) to achieve previously unreachable levels of performance. However, the NN models are considered black boxes, where the user is blind to the inner workings of the machinery behind predictions. In some applications, one may be content with getting good performance, at the cost of not being able to explain why the conclusion was drawn. In other applications, the interpretability of outputs is important, for example, in disease detection

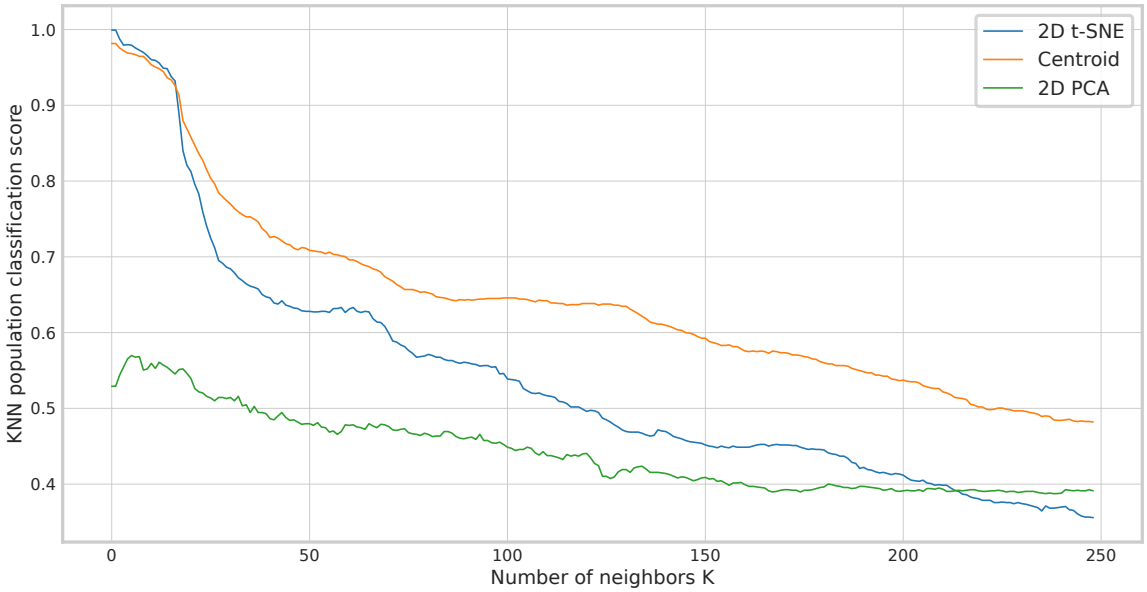

Figure 7 The subpopulation classification score using a KNN classifier on projections of the dog data using PCA, t-SNE, and our contrastive learning implementation. We are varying k on the x-axis, showing the trends in model performance on different scales. The score is computed over all samples.

Table 1 Classification performance of the different methods using a KNN classifier with $k = 3$ on the embedding coordinates. Using a 80/20 train/test split for all methods except t-SNE.

Method	**Dataset**	**Subpop score**	validation	**Superpop score**	validation
PCA	Dog	0.3587	0.3395	0.5557	0.5387
t-SNE	Dog	**0.9491**	**0.9410**	**0.9793**	**0.9815**
Triplet	Dog	0.7328	0.7749	0.8878	0.9041
Centroid	Dog	0.9306	0.9225	0.9720	0.9631
PCA	Human	0.4359	0.4203	0.8771	0.8913
t-SNE	Human	**0.7808**	**0.7609**	**0.9632**	**0.9589**
Centroid	Human	0.5689	0.4638	0.9303	0.9251
PCA	Hu. masked	0.3657	0.0845	0.8157	0.5821
t-SNE	Hu. masked	0.3972	0.2585	0.8244	0.6908
Centroid	Hu. masked	**0.5709**	**0.4614**	**0.9294**	**0.9130**

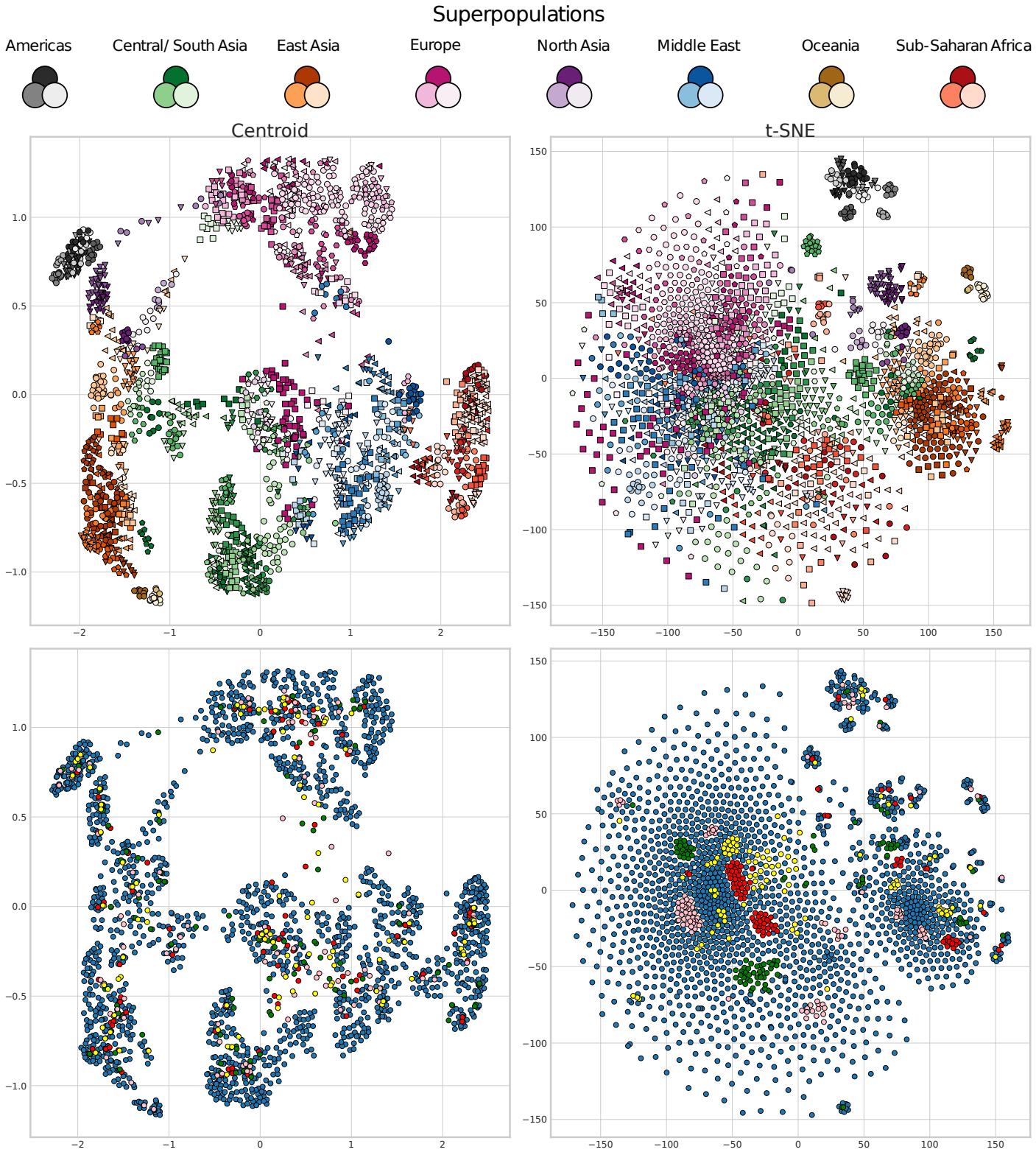

Figure 8 Dimensionality reduction results comparing our method, and t-SNE for the synthetically masked case for the Human Origins dataset. Top figures are labelled by population label, and bottom by the masking group.

and computer-aided diagnosis (Meng *et al.* 2022). Also, in association studies, we want to be able to say something about the feature importance in terms of which variants contribute to desired traits, information that could improve future breeding decisions. Explainable AI (XAI) is an emerging field, where the aim is to make the black boxes more translucent (Xu *et al.* 2019). One example of such an approach is salience maps for image classification models, where the methods can indicate which regions of the image had the most influence on the class prediction (Simonyan *et al.* 2014). There is an interesting parallel that could be explored, where one could use similar methods to explore which variants contribute to phenotype predictions using NNs.

While we are not there yet, we hope that this study is a step towards creating larger neural network models, with more generalized higher dimensional embeddings that can serve as foundational models. These models could be used to build improved analysis models for many different applications – akin to a ResNet (He *et al.* 2016) of genotype data. The creation of a reference model by first training on a large dataset to learn the structure of the genotype data, can with transfer learning techniques bring great benefit when fine-tuned to much smaller study populations, and can constitute a potential remedy for the problems that usually comes with the low sample count in many genomic studies and datasets.

Acknowledgments

Compute resources provided by SNIC through the National Supercomputer Centre (NSC) at Linköping University under Project Berzelius 2022-17, 2022-165, 2023-258, and 2024-121.

Funding

Project funded by Formas, The Swedish government research council for sustainable development, Grant no 2020-00712, Deep Learning for Analyzing Population Structure and Genotype-Phenotype Mapping.

Conflicts of interest

The authors declare no conflicts of interest.

Literature cited

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M *et al*. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ausmees K, Nettelblad C. 2022. A deep learning framework for characterization of genotype data. G3 Genes | Genomes | Genetics. 12:jkac020.

Battey CJ, Coffing GC, Kern AD. 2021. Visualizing population structure with variational autoencoders. G3 Genes | Genomes | Genetics. 11:jkaa036.

Chen T, Kornblith S, Norouzi M, Hinton G. 2020. A simple framework for contrastive learning of visual representations. In: III HD, Singh A, editors, *Proceedings of the 37th International Conference on Machine Learning*. volume 119 of *Proceedings of Machine Learning Research*. pp. 1597–1607. PMLR.

Cubuk ED, Zoph B, Mane D, Vasudevan V, Le QV. 2019. AutoAugment: Learning augmentation strategies from data. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 113–123.

Elhaik E. 2022. Principal component analyses (PCA)-based findings in population genetic studies are highly biased and must be reevaluated. Scientific Reports. 12:14683.

Goodfellow I, Bengio Y, Courville A. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

He K, Zhang X, Ren S, Sun J. 2016. Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778.

Kingma DP, Ba J. 2014. Adam: A method for stochastic optimization. CoRR. abs/1412.6980.

Lazaridis I, Patterson N, Mittnik A, Renaud G, Mallick S, Kirsanow K, Sudmant PH, Schraiber JG, Castellano S, Lipson M *et al*. 2014. Ancient human genomes suggest three ancestral populations for present-day Europeans. Nature. 513:409–413.

McInnes L, Healy J, Saul N, Grossberger L. 2018. Umap: Uniform manifold approximation and projection. The Journal of Open Source Software. 3:861.

McVean G. 2009. A genealogical interpretation of principal components analysis. PLoS Genet. 5:e1000686.

Meng C, Trinh L, Xu N, Enouen J, Liu Y. 2022. Interpretability and fairness evaluation of deep learning models on MIMIC-IV dataset. Scientific Reports. 12:7166.

Novembre J, Johnson T, Bryc K, Kutalik Z, Boyko AR, Auton A, Indap A, King KS, Bergmann S, Nelson MR *et al*. 2008. Genes mirror geography within Europe. Nature. 456:98–101.

Parker HG, Dreger DL, Rimbault M *et al*. 2017. Genomic analyses reveal the influence of geographic origin, migration, and hybridization on modern dog breed development. Cell Reports. 19:697–708.

Pearson K. 1901. On lines and planes of closest fit to systems of points in space. Philosophical Magazine Series 1. 2:559–572.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V *et al*. 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research. 12:2825–2830.

Pincot DDA, Ledda M, Feldmann MJ, Hardigan MA, Poorten TJ, Runcie DE, Heffelfinger C, Dellaporta SL, Cole GS, Knapp SJ. 2021. Social network analysis of the genealogy of strawberry: retracing the wild roots of heirloom and modern cultivars. G3 Genes | Genomes | Genetics. 11:jkab015.

Robinson J, Chuang CY, Sra S, Jegelka S. 2020. Contrastive learning with hard negative samples. arXiv preprint arXiv:2010.04592. 2010:04592.

Rodríguez-Varela R, Moore KH, Ebenesersdóttir SS, Kilinc GM, Kjellström A, Papmehl-Dufay L, Alfsdotter C, Berglund B, Alrawi L, Kashuba N *et al*. 2023. The genetic history of Scandinavia from the Roman Iron Age to the present. Cell. 186:32–46.e19.

Schroff F, Kalenichenko D, Philbin J. 2015. Facenet: A unified embedding for face recognition and clustering. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 815–823.

Shorten C, Khoshgoftaar TM. 2019. A survey on image data augmentation for deep learning. Journal of Big Data. 6:60.

Simonyan K, Vedaldi A, Zisserman A. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. Workshop at International Conference on Learning Representations (ICLR).

Sohn K. 2016. Improved deep metric learning with multi-class n-pair loss objective. Advances in Neural Information Processing Systems. 29.

Stephens ZD, Lee SY, Faghri F, Campbell RH, Zhai C, Efron

MJ, Iyer R, Schatz MC, Sinha S, Robinson GE. 2015. Big data: Astronomical or genomical? PLOS Biology. 13:1–11.

Ubbens J, Feldmann MJ, Stavness I, Sharpe AG. 2022. Quantitative evaluation of nonlinear methods for population structure visualization and inference. G3 Genes|Genomes|Genetics. 12:jkac191.

van der Maaten L, Hinton G. 2008. Visualizing data using t-SNE. Journal of Machine Learning Research. 9:2579–2605.

Xu F, Uszkoreit H, Du Y, Fan W, Zhao D, Zhu J. 2019. Explainable ai: A brief survey on history, research areas, approaches and challenges. In: Tang J, Kan MY, Zhao D, Li S, Zan H, editors, *Natural Language Processing and Chinese Computing*. pp. 563–574. Cham. Springer International Publishing.

Šavrič B, Patterson T, Jenny B. 2019. The Equal Earth map projection. International Journal of Geographical Information Science. 33:454–465.