UPPSALA UNIVERSITET

UPTEC X21 020

Examensarbete 30 hp
Juni 2021

Simulating Artificial Recombination for a Deep Convolutional Autoencoder

Fredrik Levin

Abstract

Simulating Artificial Recombination for a Deep Convolutional Autoencoder

Fredrik Levin

Population structure is an important field of study due to its importance in finding underlying genetics of various diseases.This is why this thesis has looked at a newly presented deep convolutional autoencoder that has been showing promising results when compared to the state-of-the-art method for quantifying genetic similarities within population structure. The main focus was to introduce data augmentation in the form of artificial diploid recombination to this autoencoder in an attempt to increase performance and robustness of the network structure.

The training data for the network consist of arrays containing information about single-nucleotide polymorphisms present in an individual. Each instance of augmented data was simulated by randomising cuts based on the distance between the polymorphisms, and then creating a new array by alternating between the arrays of two randomised original data instances. Several networks were then trained using this data augmentation. The performance of the trained networks was compared to networks trained on only original data using several metrics. Both groups of networks had similar performance for most metrics. The main difference was that networks trained on only original data had a low genotype concordance on simulated data. This indicates an underlying risk using the original networks, which can be overcome by introducing the artificial recombination.

Handledare: Carl Nettelblad
Ämnesgranskare: Miguel Navascués
Examinator: Pascal Milesi
ISSN: 1401-2138, UPTEC X21 020

Simulering av artificiella barn för maskininlärning

Populärvetenskaplig sammanfattning

Att det finns genetiska skillnader mellan befolkningar från olika delar av världen är nog inget som överraskar. Det som många kanske inte vet är att dessa skillnader kan vara väldigt viktiga för att hitta genetiska kopplingar till svåra sjukdomar. Om man inte vet vad som är gemensamt eller skiljer sig mellan befolkningsgrupper finns risken att man kopplar något av det till en sjukdom. Det här är en av anledningarna till att forskning inom befolkningsstruktur är oerhört viktigt!

Att arbeta med genetiska data är dock ingen enkel uppgift. Genomet hos en individ innehåller nämligen kring sex miljarder positioner. Om man tänker sig dessa som bokstäver i ett text-dokument så blir det kring en miljon sidor. Detta gör att man helst vill minska mängden data man arbetar med, samtidigt som man inte vill förlora någon information. Det är helt enkelt enklare att lägga ett pussel med 100 bitar än ett med 1000 bitar, men man vill inte att halva bilden ska försvinna på köpet. Därför har jag i detta projekt arbetat med en maskininlärningsmodell som först komprimerar alla data, för att sedan försöka återställa dessa data igen. Man kan jämföra det med att man har ett utskrivet papper från det här text-dokumentet som man knycklar ihop för att det ska ta mindre plats, och sen vill släta ut så att det är läsbart igen.

Målet med projektet var att förbättra modellen genom att skapa nya, artificiella data så att den har mer material att träna på. Detta ville jag även göra på ett sätt som artificiellt efterliknar den biologiska processen för när barn blir till, så att nya data efterliknar äkta data från människor. Genom att slumpmässigt välja två individer och kopiera delar av deras genetiska material skapade jag en ny individ. Man kan likna det vid att jag kopierar de tre första raderna från det ena dokumentet till ett nytt dokument, för att sedan kopiera de fem nästa raderna från det andra dokumentet till det nya. Det här repeterade jag sedan med olika antal rader tills jag hade ett helt nytt dokument. I princip har jag då skapat en ny individ, precis som när ett barn får sitt genetiska material från sina föräldrar.

Med de flesta mått jag använde för att utvärdera modellen påverkades den inte så mycket av att lägga till mina nya data. Däremot visade det sig att grundmodellen hade svårt att hantera data från människor den inte hade sett förut. Detta blev bättre efter att jag hade introducerat data från lika många artificiella människor som det fanns människor från början. Slutsatsen från projektet är alltså att introducerandet av nya, artificiella data gör modellen bättre på att hantera data från individer modellen inte tränats på, men med nackdelen att den behöver arbeta längre för att hantera alla data. Eftersom det är väldigt viktigt att modellen fungerar för alla, även om den inte har sett den specifika människan, var det här projektet ett viktigt tillägg till modellen.

Table of Contents

Abbreviations

cM	Centimorgans
DNA	Deoxyribonucleic acid
GCAE	Genotype Convolutional Autoencoder
GPU	Graphics processing unit
NN	Neural network
PCA	Principal component analysis
SNP	Single-nucleotide polymorphisms

1 Introduction

The genetic diversity in humans is dependent on our geographical ancestry. How much two lineages differ genetically is influenced by their geographical distance and history. Studying this population structure becomes especially important when performing large association studies, aiming to understand the genetics behind various diseases. There are many markers that are shared within populations. Without understanding the genetic similarities and differences between populations there is a large risk of false positives and negatives (Marchini *et al.* 2004, Goldstein & Chikhi 2002). Therefore, population structure is an important subject in order to map the genetics behind diseases.

Machine learning is becoming increasingly popular within many scientific fields. These machine learning models often outperform the state-of-the-art methods due to their ability to detect intricate patterns. This project will be focusing on a neural network model that has been shown to improve upon established methods for analysing population structure.

2 Background

Population structure is something that has been studied using multiple different techniques. One group of techniques that is starting to show up more within the field is machine learning, and more specifically neural networks. For these techniques it is important to have enough data in order to get a good and generalisable result. It is not always possible or preferable to collect new data, either due to time, money, or other limitations. Therefore, alternative solutions are necessary for achieving the best possible results.

The genetic data used for studying population structure contain a lot of information. This is why one widely used technique for studying this is the dimensionality reduction method principal component analysis (PCA) (Patterson *et al.* 2006). PCA is used to reduce the dimensions of the genetic data, allowing for both an easy visualisation using a 2D plot and quantification of the genetic similarities. The Nettelblad group at the Division of Scientific Computing at Uppsala University has recently presented a deep learning network denoted Genotype Convolutional Autoencoder (GCAE) that has shown promising results on improving the dimensionality reduction of single-nucleotide polymorphism (SNP) data compared to PCA (Ausmees & Nettelblad 2020). By utilising the non-linearity of both the data and the network it improves the performance of dimensionality reduction compared to the linear method PCA. This improved method has been shown to be extremely useful for quantifying genetic similarities.

Neural networks such as GCAE generally perform better the more training data that are available. When working with SNP data, the amount of data is limited by population sizes. In this thesis I will aim to solve this by introducing data augmentation in the form of artificial recombination. This will in theory allow for unlimited training data through an otherwise naturally occurring process. The goal of this thesis is to improve the performance of GCAE on unseen data by finding a way of simulating artificial recombination of the training data. Simulating new data through this process allows the network to train on more data than are otherwise available.

2.1 Recombination

Our DNA consists of a long sequence of nucleotides made up by the four nucleobases: adenine, cytosine, guanine, and thymine. The order of these makes up our individual genomes, which carries our genetic information that determines what proteins are expressed or not. Some mutations can change the ordering of the genome, but the most common mutation type is one that only changes a single nucleotide. This kind of mutation is called a single-nucleotide polymorphism (SNP) (Campbell N. et al., 2015, p462).

The human genome is arranged in 46 chromosomes, which partially consist of packed DNA. Humans are also what is called a diploid organism. This means that we have two sets of chromosomes, meaning that the 46 chromosomes form 23 pairs. These pairs of chromosomes contain the same genes, but they could have different variants of the gene. In order to create gametes, i.e. sperm and egg cells, necessary for reproduction the specific type of cell division called meiosis occurs. During meiosis, the pairs of chromosomes are separated such that there are only one of each chromosome, resulting in the gametes being haploid. During this process, the chromosomes exchange genetic material (Figure 1), which is what is called meiotic recombination. Another commonly used word for this is chromosomal crossover. This occurs in both parents of an offspring, meaning that a new diploid individual is created by a combination of two haploid cells that each have their own possible recombination.

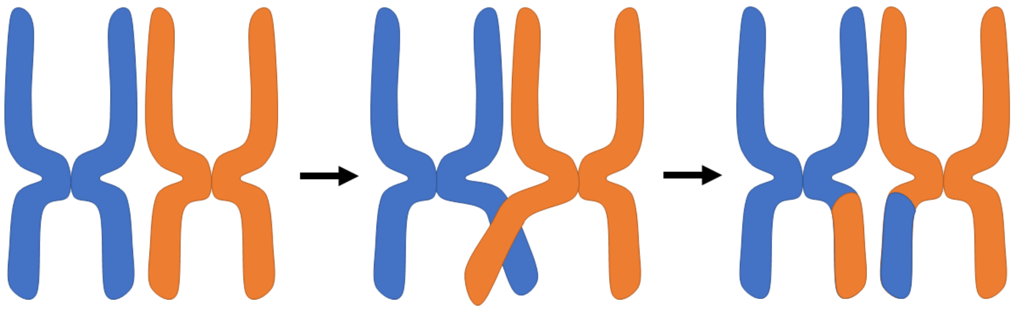

Figure 1: A visualisation of meiotic recombination. The two chromosomes exchange genetic material.

The ability to handle recombination could be important for the robustness of dimensionality reduction methods used on genetic data, since it is a process that results in a higher genetic diversity. Recombination can occur both in mitosis and in meiosis, but the most commonly talked about recombination in eukaryotes is the meiotic recombination (Andersen & Sekelsky 2010). There are an average of one to three crossover events per chromosome pair during human meiosis (Campbell N. et al., 2015, p320).

2.1.1 Centimorgans

Centimorgan (cM) is a unit that is used for measuring genetic linkage. One cM is defined as the distance between two positions on the chromosome for which an average of 0.01 crossover events are expected in one generation (Lodish H. et al., 2016, p233). It is not directly linked to physical distance due to the differing recombination rates of different parts of the genome, but 1 cM can be roughly translated to 1 million base pairs in humans. Since an even number of crossover events on the same location results in the exact same genome, only odd numbers of crossover events result in genetic recombination. Therefore, the Haldane function (Equation 1) has been introduced in order to correct for this and calculate the probability of genetic recombination between two loci (Rédei 2008).

Equation 1: The Haldane function. r is the recombination rate; d is the distance in Morgans and e is the base of the natural logarithm. $r = \frac{1}{2}(1 - e^{-2d})$

2.2 Dimensionality Reduction

Reducing the dimensions of data used for machine learning is widely practiced. Dimensionality reduction is important since a machine learning model will be increasingly complex the more features used to train it. Simple models are preferred due to having a lower risk of becoming overfitted, being easier to interpret, and being more computationally efficient. Models trained on a high-dimensional dataset with many features will also be more prone to overfitting. Another important use of dimensionality reduction is to create a low-dimensional and clear representation for visualising the data.

The basic idea behind dimensionality reduction is to lower the dimensionality of the data in a way such that the data still retain the important information. There are multiple ways to reduce the number of features, and these are classified as linear and non-linear. Linear methods handle linear data well by trying to find a representation of the data in a hyperplane of lower dimensionality (van der Maaten *et al.* 2009). For non-linear data it is better to use a model which can learn non-linear patterns. GCAE is an example of a non-linear deep learning model that has been implemented on genomic data.

2.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is one of the more known and commonly used linear dimensionality reduction methods (van der Maaten *et al.* 2009). PCA computes the eigenvectors to the covariance matrix of the dataset. The eigenvectors represent the direction in which the data has the most variation, and they are sorted from most variance to least. PCA then projects the data on the directions with the highest variation. By doing this PCA captures the variation in the data while reducing the dimensions. When plotting it is usually enough to chose either two or three dimensions, but when using PCA for other purposes it is recommended to determine the number of dimensions with a significant variation (Wold S *et al.* 1987).

2.3 Neural Networks

Machine learning is the study of finding computer algorithms that learn from data or experience. It originated in the pursuit of artificial intelligence and the field has been growing quickly in the last years. A neural network (NN) is a common method of machine learning that is inspired by the neurons in the brain. It is considered a feature learning method. This means that it identifies features from the data on its own, unlike many other methods where pre-defined features are needed (Goodfellow I *et al*. 2016). NN consists of an input layer, an output layer, and one or multiple hidden layers between these. A NN is considered a deep NN if it has multiple hidden layers. Each layer has a number of nodes that is connected to nodes in adjacent layers. Signals are passed from nodes in the layer before a node, through weighted connections, into the node. Based on the input an activation function decides the output of the node. If the layer is convolutional, the input will only come from a subset of the nodes in the previous layer, and if it is a fully connected layer it will come from all nodes.

2.3.1 Data Augmentation

When training a NN, the goal is to minimise a loss function. This is done by running training data through the network and tweaking the weights of the connections in order to optimise this function. The more training data available the better this optimisation will be. With a small dataset there is a risk of adapting too much to the training data. This is what is called overfitting and means that the network has specialised on the training dataset. It will not perform as well on a new, unseen dataset. To counteract overfitting, it is common to introduce data augmentation.

Data augmentation means to introduce more unseen data by modifying the existing data slightly. This is commonly used within the field of image analysis to, among other things, improve existing image classification networks (Mikołajczyk & Grochowski 2018, Perez & Wang 2017). One reason for it being so common is that it is particularly easy to create new data by e.g. rotating or resizing the images.

2.3.2 Autoencoders

An autoencoder is a model design consisting of a bottleneck. The data are encoded in the first parts of the network, and then decoded after the bottleneck (Figure 2). The most common version of an autoencoder uses NN as encoder and decoder (Bank *et al.* 2021). In the middle layers of an autoencoder the data are a low-dimensional representation of the original state. When decoding the compressed data, the network tries to replicate the original input. Due to the goal being to reconstruct the data from the compressed state, the compressed data needs to include the essential parts of the original data. Therefore, it contains information about the original data while being easier to store and handle, which is what makes that a particularly useful dataset for analyses. (Baldi 2012, Bank *et al.* 2021)

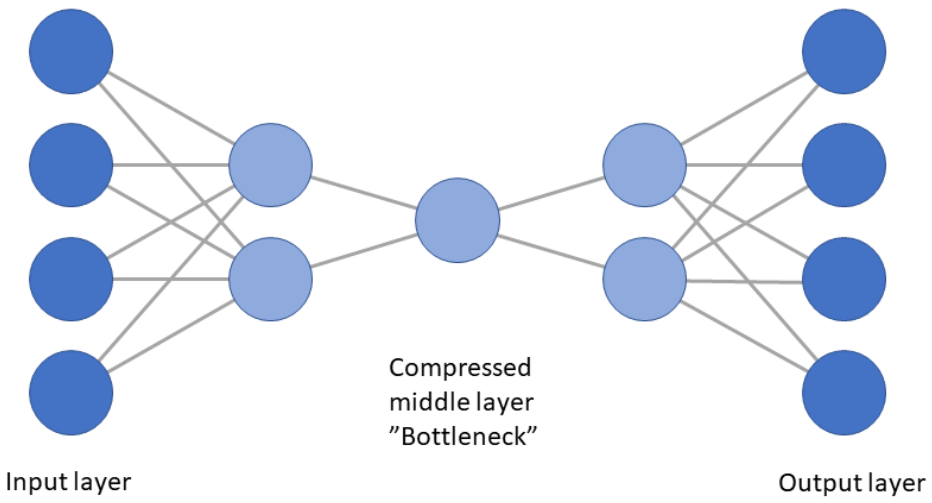

Figure 2: A general example of the structure of an autoencoder. The input from the input layer is compressed in the bottleneck, and then retracted after the bottleneck.

2.3.3 GCAE

The network presented by the Nettelblad group at the Division of Scientific Computing at Uppsala University is denoted Genotype Convolutional Autoencoder (GCAE). GCAE is a deep convolution autoencoder network. This means that it is an autoencoder with multiple hidden layers. The encoding part of GCAE alternates between convolutional layers and pooling layers, while the decoding part has a similar structure using upsampling instead of pooling. In the centre part of the model there is a series of fully connected layers (Ausmees & Nettelblad 2020). Convolutional networks are popular in image analysis, but GCAE applies similar techniques to SNP data. It uses a sequence of SNPs and tries to recreate the data in the same way an image is recreated.

GCAE also includes a representation of missing data in the input, which is something not present in PCA. This makes GCAE especially robust compared to PCA when working with low-coverage samples.

3 Material and Methods

This project was based on the network model created by Ausmees & Nettelblad (2020) which can be found on GitHub: https://github.com/kausmees/GenoCAE. The implementations can be found on a fork to the GitHub repository at https://github.com/Runpunchwin/GenoCAE.

The code is written and tested using Python 3.7 and the analyses were run on the Alvis cluster using Python 3.6. The Alvis cluster is a resource hosted by the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE). The analyses were run using the prebuilt Nvidia container image tensorflow_20.09-tf2-py3, which is running Python 3.6.

3.1 Data

The data used in this study is the same dataset used in the original study by Ausmees & Nettelblad (2020). It is designed for population genetic studies. Both the dataset and the grouping of populations and superpopulations originates from Lazaridis et al. (2016). The dataset is available for download from https://reich.hms.harvard.edu/datasets. The data consists of information about 160 858 SNPs for 2067 individuals from 166 populations. These are also categorized into eight superpopulations. The data are unphased and diploid.

The format used is EIGENSTRAT, which is a collection of files with the file extensions .eigenstratgeno, .fam and .snp (David Reich Lab., 2021). In the main file the columns consist of individuals and the rows of SNPs. The value in each position represents the number of copies that is not the reference allele, where a nine indicates a missing value. Information about the populations is stored in the .fam file, and information about the SNPs is stored in the .snp file.

3.2 Implementation

Indices of markers located on each chromosome were computed from information in the .snp file. These were used to define each chromosome. To simulate one instance of recombined data, two genomes were randomly selected from the training set. For each chromosome, the number of cuts was randomly chosen to be between zero and four to get close to the average amount of crossover events per chromosome. The cuts were then randomised along the length of the chromosome based on distance in base pairs, using a uniform distribution. An alternative way of randomising the cuts were tried in a few networks. Here the randomisation was performed based on the distance measured in centimorgans (cM).

Finally, the new instance of data was built by alternating between the two original genomes. Between the start of the chromosome and the first cut, the same positions of the first genome were copied. Between the first cut and the second cut, the second genome were copied, and vice versa. This means that we here simulate an artificial and diploid recombination. It is assumed that the cut is done on the same position on both haploid strands. This full process was repeated within each batch until it had enough simulated data. For each batch, an equal amount original data and simulated data were used in training.

The recommended hyperparameters from Ausmees & Nettelblad (2020) were used for training the network. The only two deviations were that the batch size and the learning rate was decreased due to memory issues. For this project, a batch size of 30 and a learning rate of 0.005 were used. Training was done using one Nvidia Tesla T4 graphics processing unit (GPU) on the Alvis cluster. The training process was timed to allow for comparison between the different methods. After training, the compressed representation of the data points was plotted in order to visualise the dimensionality reduction. The density of simulated individuals was visualised in the same plot using a heatmap ranging from transparent to black.

3.3 Evaluation

GCAE has two main metrics implemented to evaluate the performance of the network. These are the F1-score and the convex hull error (Ausmees & Nettelblad 2020), which was applied to the compressed data points. The F1-score measures the harmonic mean of precision and recall (Sasaki 2007). Precision is the fraction of all data points classified to a group that are relevant, and recall is how good the model is at finding data points belonging to the group. This was applied to data points belonging to each population and an average was calculated over all populations. It was also applied to each superpopulation, where an average was calculated over all superpopulations. The hull error is measured by defining the convex hull of all points belonging to a population. This is done by finding the smallest convex shape that includes all points of the population. To get the error one calculates the fraction of samples from other populations within the hull. This is then averaged over all populations. Both metrics were calculated for the original network and the network trained on simulated data, such that they could be compared.

Another metric built into the GCAE network is genotype concordance. Genotype concordance is the amount of similarity between the model output and the model input. It ranges from zero to one, where one says that the genomes are identical. A baseline is also calculated, showing the genotype concordance given by assuming the genotype based on what is most frequently occurring for each marker. Genotype concordance was calculated and plotted for the original dataset and the simulated dataset separately.

4 Results

When the compressed data points are visualised in a 2D-environment they all share a common legend describing which populations and superpopulations they belong to (Figure 3). In the early stages during training the simulated data points span the gap between the original data points when plotted in their compressed state (Figure 4).

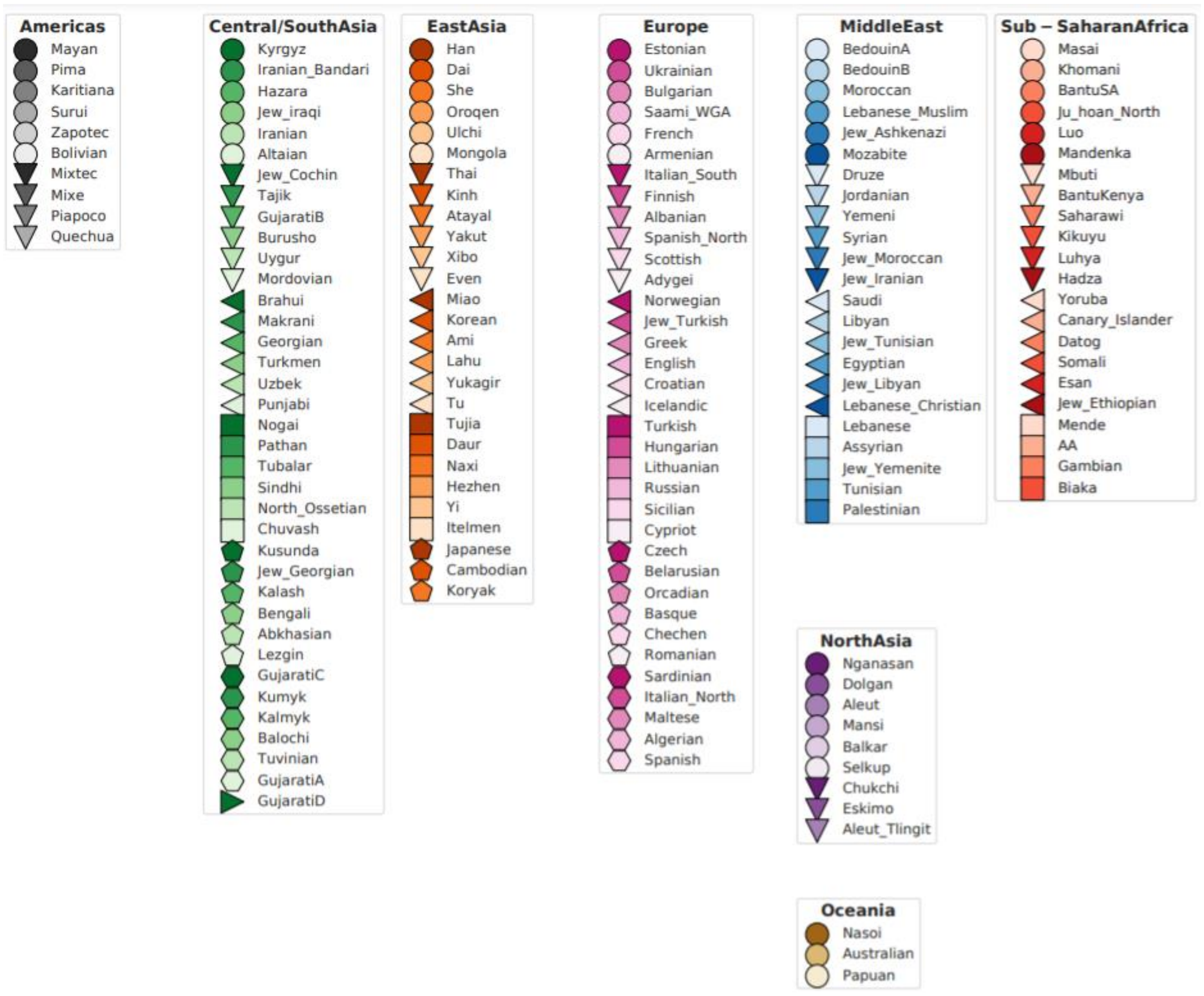

Figure 3: The legend used when visualising the compressed data points in a 2D-environment. It contains eight superpopulations, of which each has several populations.

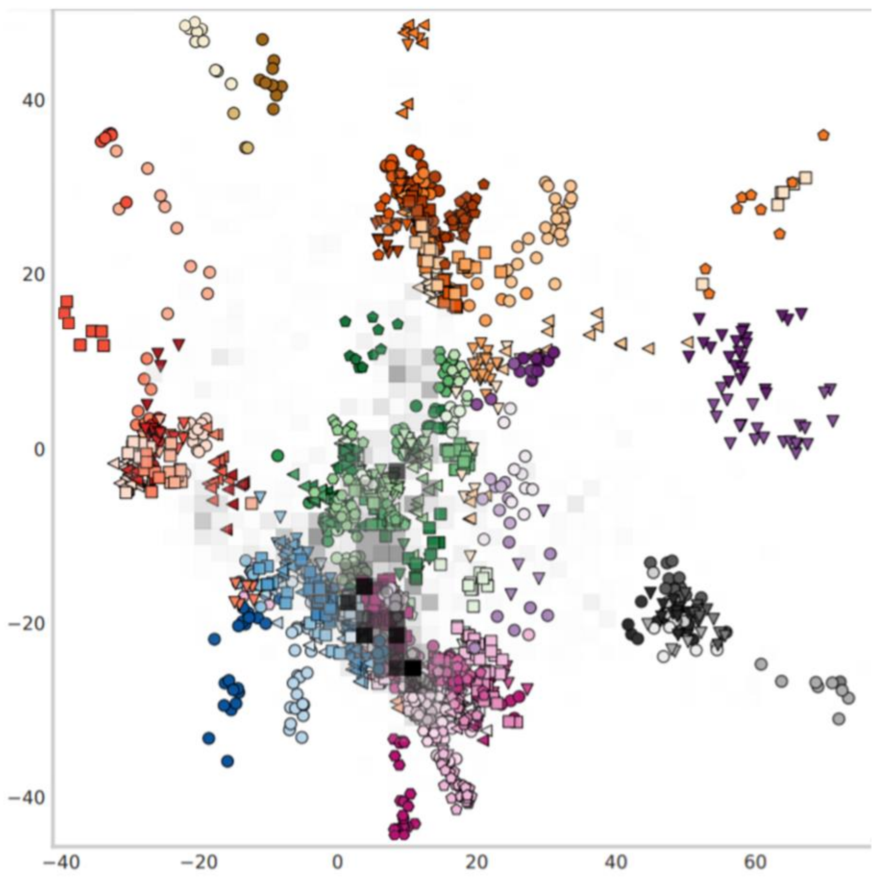

Figure 4: Visualisation of the compressed data points 1/8 into the training process when training a network using both simulated data and the original data. The simulated data are shown as a heatmap ranging from transparent to black and is shown to span the gaps between the original data.

Several networks using equal amounts of simulated data and original data were trained in order to see a general trend. To have something to compare the results to, networks without recombined data were trained due to the variables differing from the original article by Ausmees & Nettelblad (2020). The compressed data points from the bottleneck in the network were used for both visualisation and evaluation. For visualisation, the data points were simply plotted in a 2D environment (Figure 5). Training times for the original network and the network using simulated data was compared (Table 1), showing that the original network takes approximately 1.5 minutes shorter per epoch to train.

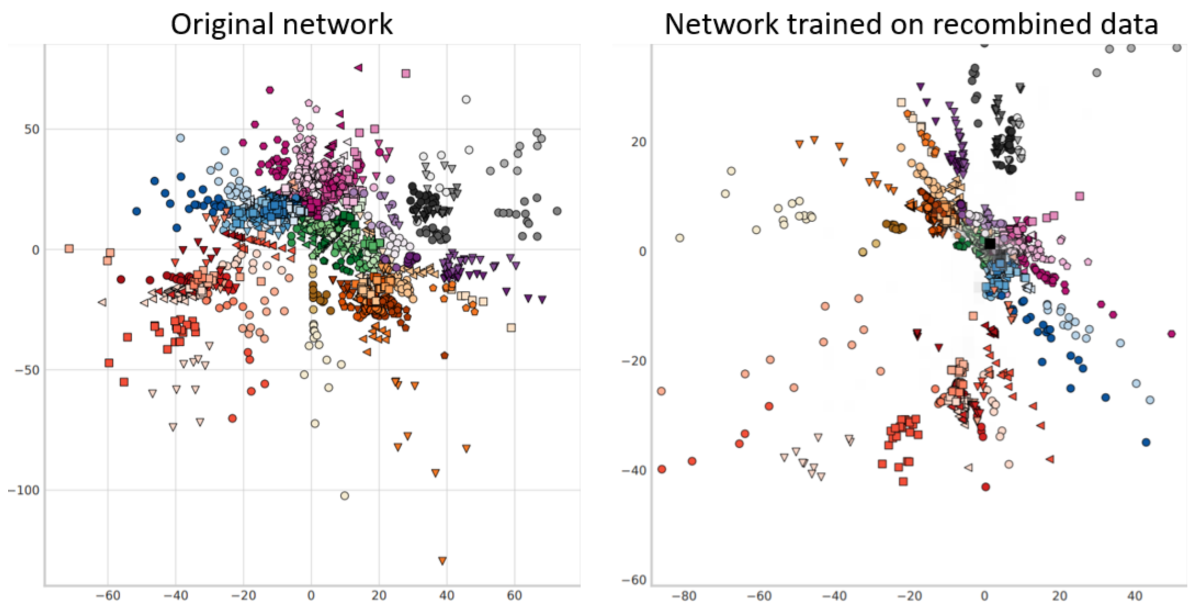

Figure 5: The compressed data points from epoch 800 plotted in a 2D-environment. To the left is a network trained only using the original data, and to the right a network trained using both the original data and simulated data.

Table 1: Comparison of the training times for one epoch when training the different networks.

Network	Average training time for one epoch in seconds
Original data	303
Original data + recombined data (using distance in base pairs)	390

The genotype concordance per epoch was calculated on a network trained on solely the original data and on a network trained on a combination of original and simulated data. Both these values were plotted against the baseline genotype concordance (Figure 6). On the network trained on original data there is a clear difference in genotype concordance between the original and the simulated data. The original data has a concordance greater than the baseline concordance, while the simulated data has a decreasing concordance below the baseline. For the network trained on simulated data both genotype concordances are above the baseline. Concordance for the original data is slightly higher than for the simulated data.

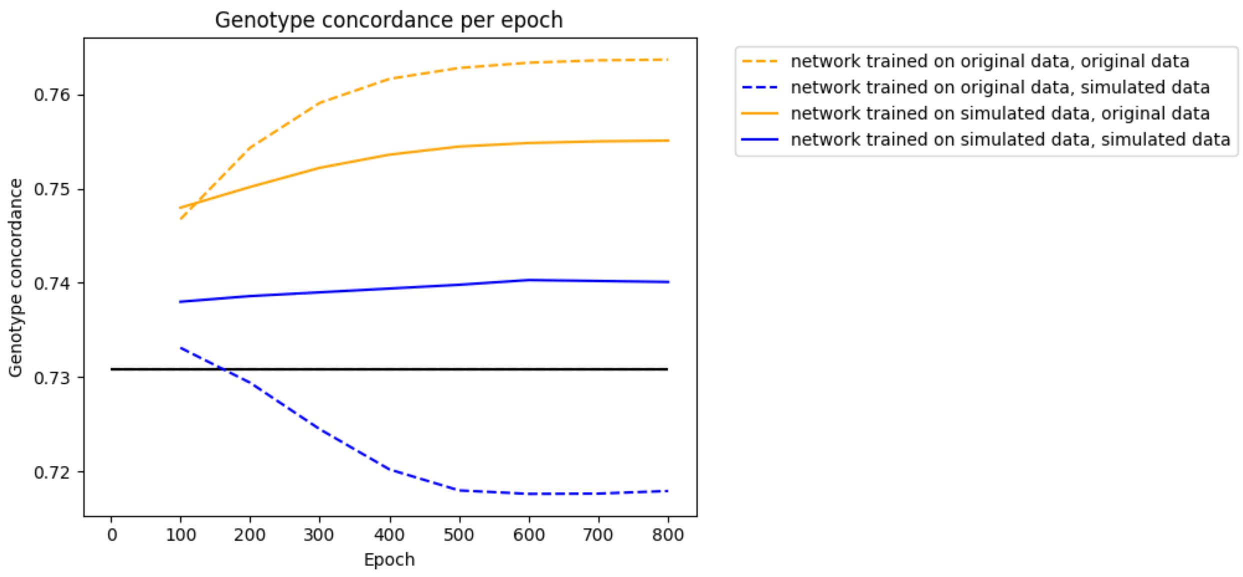

Figure 6: Visualisation of the genotype concordance of the two networks. The orange lines represent the original data, the blue lines simulated data and the black line the baseline value. The dashed lines represent the network trained on only the original data and the solid lines represent the network trained on both the original data and simulated data.

The convex hull error reaches a stable level for both the networks (Figure 7). The network trained on both recombined data and the original data stabilizes after approximately 200 more epochs than the network that only used the original data. When comparing the best level of convex hull error between multiple runs of the two networks the one trained on the original data has an average value of 0.254, while the one trained on recombined data has an average value of 0.287 (Table 2). For the F1-score, both networks have a slight gradient upwards which indicates that they might continue to improve if trained further (Figure 8). The average of the optimal F1-score over multiple runs for the original network is 0.826, while the same value for multiple networks trained on both recombined data and original data is 0.805.

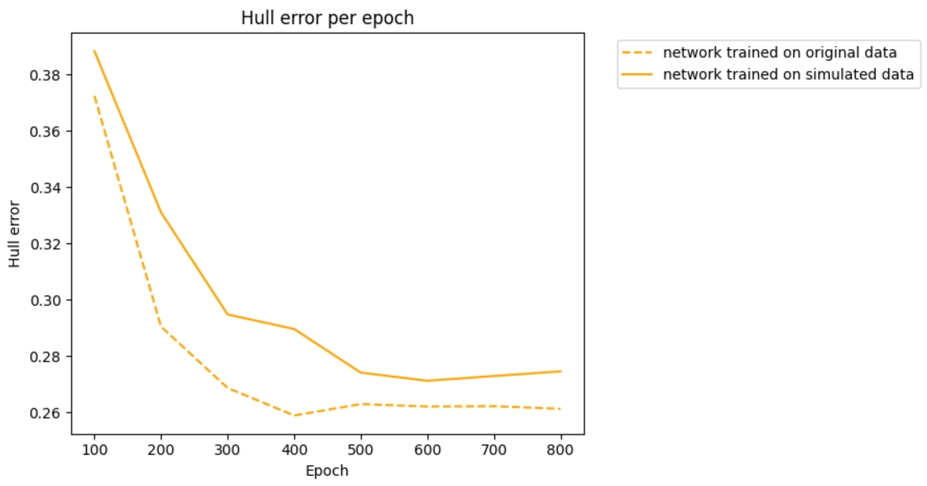

Figure 7: A comparison of the convex hull error of the two networks. The dashed line represents the network trained on only the original data and the solid line represents the network trained on both the original data and simulated data.

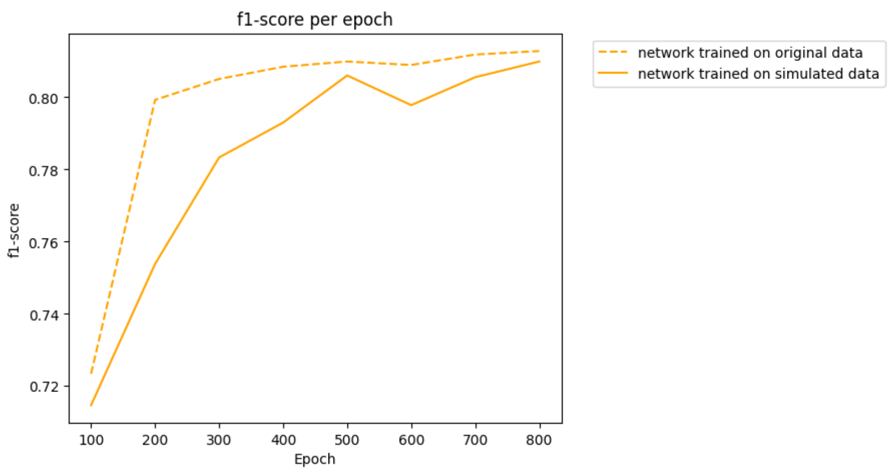

Figure 8: A comparison of the F1-score of the two networks. The dashed line represents the network trained on only the original data and the solid line represents the network trained on both the original data and simulated data.

The networks trained on simulated data based on cM were compared to the networks trained on simulated data based on base pairs, which were the standard in most tests. Both networks using simulated data have similar evaluation metrics when looking at the average over multiple runs and does not differ significantly (Table 2).

Table 2: Mean value (μ) and standard deviation (σ) of evaluation metrics for the different networks. For the convex hull error and the F1-score for populations the best epochs were used. Each value is an average of three to four networks in order to counteract randomness.

Network	Convex hull error (best value)		F1-score (best value)		F1-score – superpopulations		Genotype concordance (original data)	
	μ	σ	μ	σ	μ	σ	μ	σ
Original data	0.254	0.006	0.826	0.009	0.818	0.011	0.761	0.005
Original data + recombined data (using distance in base pairs)	0.287	0.016	0.805	0.005	0.800	0.004	0.754	0.001
Original data + recombined data (using distance in cM)	0.284	0.014	0.808	0.007	0.798	0.004	0.754	0.002

5 Discussion

The goal of this project has been to introduce data augmentation by simulating recombination to create new and unseen data. Since it was preferable that this process was done during training in an efficient way, implementing it within the TensorFlow code was preferred above using an external software. In order to simulate recombined data as in nature one need phased genomes. GCAE was originally trained using a dataset containing non-phased genomes, and the same dataset was used for this project to keep it connected. Simulating recombination as described in the methods is therefore not the exact natural way, but it is an artificial way of doing it. Since the simulation makes use of a distance measure representing genetic linkage, we still preserve linkage disequilibrium in some sense. Markers located close to each other are less likely to be separated. Since the simulated data points are a combination of existing data points through artificial recombination, it is expected that the simulated individuals are spanning the gap between the original data points. This is also shown to be the case when looking at the earlier epochs in the training of the networks. In later epochs the data points clusters together more, making this effect less visible. This indicates that the simulated data are looking as expected.

The graphical plotting of the network trained on only the original data and of the network trained on a combination of simulated data and original data are fairly similar. There is nothing from this that makes any alternative a clearly better option for separating the different clusters. The one using simulated data has slightly tighter groups, while the original network is more spread out. Spreading the data out is good when it separates the different populations, but gets an opposite effect when populations start overlapping. Therefore, we mainly look at the evaluation metrics when comparing the results.

Assessing the averages of the evaluation metrics F1-score and convex hull error we see a small difference between the networks. Even though the values are based on an average they could differ slightly when run multiple times. When looking at the convex hull error, both networks seem to reach an equilibrium. The original network reaches it in about 400 epochs, compared to 500-600 epochs for the network using simulated data. While this is a distinct difference, it is also expected. The original network has half the amount of training data to work through, meaning that it will finish earlier. Comparing the best value of the two networks we see that the original network has an average convex hull error that is approximately 0.03 lower than the other network. This is a minor difference, but it also indicates that the network's performance does not increase when introducing simulated data. For the F1-score for populations the equilibrium is not as obvious, but it reaches a somewhat stable level. The averages of the F1-scores here differ by approximately 0.02. There is a small advantage to the original network, which again indicates that the performance does not increase when adding the simulated data. The averages of F1-scores for the superpopulations show a remarkably similar pattern.

When considering the genotype concordance of the different networks, it is clear that the original network performs significantly worse when trying to recreate the simulated data. It even performs worse than the baseline concordance. This indicates that even though the simulated data did not improve the performance of GCAE, there might be some underlying risks with using the original model as it is. Since the original network does not handle the simulated data particularly well, it might have tendencies to be overfitted towards the original data and not optimal for all unseen data. When simulated data are included in the training the genotype concordance of simulated data becomes significantly better. This indicates that the network becomes more robust on simulated unseen data. Since new data are simulated for calculating the genotype concordance, the network should not be overfitted on specific data. There could be some overfitting towards the data generator. This is however less of an issue than being overfitted on a specific dataset. The data generator is also made to emulate recombination, which is a realistic process of introducing variation. If the network becomes more robust on unseen data, it seems like an inclusion that is valuable even though it increases the training time significantly.

Most of the work has been centred around the distance measured in base pairs. This might at first glance seem weird when there are centimorgans, which are a measurement dedicated to genetic linkage. However, the cutting based on base pairs already made the simulated data span the gaps between the original data points. Since the individual data points chosen for recombination are chosen from all populations, they often differ drastically. This will affect the results more than the size of the chunks merged into the new data point. Using recombination distances in cM, the new data would still span the gaps, just in a slightly different way.

When using cM to estimate the number of crossover events, it is recommended to use the Haldane mapping function to correct for only an odd number of crossover events being detectable. This function is not linear by itself, but for small distances it can be approximated as linear on a local scale. Since this project only considers the distance of adjacent markers, and the markers are spread across the chromosomes with small distances between each other, it was reasonable to use the cM measure as it is. The main use of this was also to see whether the use of cM would result in a better network. If other methods of simulating recombination are used, then the Haldane function might become more important. When comparing the network trained on simulated data using base pairs with the network trained on simulated data using cM there are tiny differences in the averages of the metrics, indicating that the two methods have similar performance. When using cM one get marginally better values for the convex hull error and the F1-score for populations. However, the F1-score for superpopulations is marginally better for the other network. These differences are probably due to the randomness in the methods.

5.1 Continuing the work

Even though the results of this study point towards simulating recombination not improving the result in a visible way, there are alternatives that could affect the results. Since the results are as similar as they are, a small change could make all the difference. There are several main steps I would consider if continuing to look at this question. To change the way the cutting is randomised, to change the way the data points are randomised, or to compare genotype concordance for a validation set.

For the cutting the next step would be to introduce the Haldane function into the process. By calculating the actual probability of a crossover event happening instead of using cMs, you could assess each gap individually. By assessing each individual gap, the need for using an estimation of the amounts of cuts gets removed. It would require more calculations, resulting in a slower program. However, it might result in an even more realistic representation of recombination that could improve the performance further.

In this report, the individual data points have been randomised from the full dataset. By instead randomising another data point within the same population as the first one chosen, the simulated data would populate the populations instead of spanning between them. This could make it so that the network performs better by introducing more unseen data in each population. One problem with this is that it also potentially could overexpress certain patterns available within specific populations in the existing data. However, the network would be less likely to become overfitted on existing individuals, and therefore might perform better on unseen data.

Another possible step is to use the validation dataset when comparing genotype concordance. The validation set is a part of the original dataset not used in training the network. This has not been used specifically to compare metrics earlier due to the evaluation metrics being calculated on parameters not used in training. By utilising the validation set it becomes possible to compare the genotype concordance of the different networks on real, unseen data. This would show whether the networks trained using recombined data are better at reconstructing real data as well as simulated data.

6 Conclusion

When looking at graphical plotting, convex hull error and F1-scores, performance does not seem to improve when simulating recombination in the data used for training GCAE. It does not make it much worse, but it takes more time to train the network. However, the low genotype concordance of simulated data on a network trained on original data indicates that there might be underlying risks with using the original network. When the simulated training data was used, it performed better on data simulated with the same data generator. Given that the simulation of new data tries to emulate natural recombination and simulates new data when calculating the genotype concordance, this indicates that introducing simulated recombination will improve the performance of the network on unseen data.

7 Acknowledgements

The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE) partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

I would like to thank my supervisor, Carl Nettelblad, for continuous advice and support throughout the process, as well as Kristiina Ausmees for providing me with the necessary resources related to GCAE. I also want to thank my subject reader, Miguel Navascués, for critically evaluating my work and making sure I kept everything well structured. I would like to acknowledge the course administrator, Lena Henriksson, for making sure all of this was possible. I want to thank my opponents, Karl Svärd and Mei Wu, for their input on the report and presentation. Finally, I would like to thank my family and friends for continuous support through everything!

References

Andersen SL, Sekelsky J. 2010. Meiotic versus Mitotic Recombination: Two Different Routes for Double-Strand Break Repair. BioEssays: news and reviews in molecular, cellular and developmental biology 32.

Ausmees K, Nettelblad C. 2020. A deep learning framework for characterization of genotype data. bioRxiv 2020.09.30.320994.

Baldi P. 2012. Autoencoders, Unsupervised Learning, and Deep Architectures. Proceedings of ICML Workshop on Unsupervised and Transfer Learning. JMLR Workshop and Conference Proceedings,

Bank D, Koenigstein N, Giryes R. 2021. Autoencoders. arXiv:2003.05991 [cs, stat]

Campbell N, Reece J, Urry L, Cain M, Wasserman S, Minorsky P, Jackson R. 2015. Biology, A Global Approach. Tenth edition. Pearson Education Limited. Printed and bound by Courier/Kendallville in the United States of America.

Goldstein DB, Chikhi L. 2002. HUMAN MIGRATIONS AND POPULATION STRUCTURE: What We Know and Why it Matters. Annual Review of Genomics and Human Genetics 3.

Goodfellow I, Bengio Y, Courville A. 2016. Deep Learning. MIT Press.

Input File Formats and Conversion Program | David Reich Lab. WWW document: https://reich.hms.harvard.edu/software/InputFileFormats. Accessed 18 May 2021.

Lazaridis I, Nadel D, Rollefson G, Merrett DC, Rohland N, et al. 2016. Genomic insights into the origin of farming in the ancient Near East. Nature 536.

Lodish H, Berk A, Kaiser C, Krieger M, Bretscher A, Ploegh H, Amon A, Martin K. 2016. Molecular Cell Biology. Eighth edition. Parker KA. Printed by W.H. Freeman and Company.

Marchini J, Cardon LR, Phillips MS, Donnelly P. 2004. The effects of human population structure on large genetic association studies. Nature Genetics 36.

Mikołajczyk A, Grochowski M. 2018. Data augmentation for improving deep learning in image classification problem. 2018 International Interdisciplinary PhD Workshop (IIPhDW).

Patterson N, Price AL, Reich D. 2006. Population Structure and Eigenanalysis. PLOS Genetics 2: e190.

Perez L, Wang J. 2017. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. arXiv:1712.04621 [cs]

Rédei GP (ed). 2008. Haldane’s Mapping Function. Encyclopedia of Genetics, Genomics, Proteomics and Informatics, pp. 836–836. Springer Netherlands, Dordrecht.

Sasaki Y. 2007. The truth of the F-measure. Teach Tutor Mater

Van der Maaten L, Postma E, van den Herik J. 2009. Dimensionality reduction: a comparative review. J Mach Learn Res.

Wold S, Esbensen K, Geladi P. 1987. Principal component analysis. Chemometrics and Intelligent Laboratory Systems 2.