UNIVERSIDADE D
COIMBRA

Elmer Jorge Inácio Carlos

Generative models for synthesis of artificial human genomes using GWAS summary statistics

Dissertation in the context of the Master's in Computational Biology, advised by Joel Perdiz Arrais and presented to the Faculty of Sciences and Technology / Department of Life Sciences.

July 2023

Faculty of Sciences and Technology

Department of Life Sciences

Generative models for synthesis of artificial human genomes using GWAS summary statistics

Elmer Jorge Inácio Carlos

Dissertation in the context of the Master's in Computational Biology, advised by Professor Joel Perdiz Arrais and presented to the Faculty of Sciences and Technology / Department of Life Sciences.

July 2023

1 2 9 0

UNIVERSIDADE D
COIMBRA

This work was developed in collaboration with:

Center for Informatics and Systems of the University of Coimbra

Contributions and Funding

I, Elmer Carlos, declare to have fully written and perform the research behind this thesis. In addition, I declare that this thesis has not been previously submitted, as a whole or in part, to obtain another graduation level.

“42.”

Resumo

A geração de genótipos é uma tarefa importante na pesquisa genética, oferecendo oportunidades para aumento de dados e preservação da privacidade em Estudos de associação do genoma completo (GWAS). Neste trabalho, propusemos um framework de Rede Generativa Adversaria de Wasserstein com Penalidade de Gradiente (WGAN-GP) para sintetizar dados de genótipos realistas. Nossa abordagem consistiu em duas etapas separadas: um processo de pré-treino auto-supervisionado e o processo de treinamento do WGAN-GP. Ao longo do processo de pré-treino, o gerador atua como um modelo semelhante a um decodificador, mapeando as estatísticas genéticas de baixa dimensão para os genótipos originais. Esse processo serve como uma forte inicialização para o subsequente processo de treinamento do WGAN-GP, onde o objetivo é fazer o gerador aprender a sintetizar genótipos diversos e realistas. Os resultados de nosso framework proposto, auxiliado pelas informações codificadas nas estatísticas genéticas de resumo, demonstram resultados favoráveis, destacando valores promissores de estatísticas genéticas e utilidade dos dados sintéticos. No entanto, as diversas discrepâncias observadas nos gráficos de Análise de Componentes Principais (PCA) e a divergência na validação apontam para várias áreas de melhoria futura. Planeamos melhorar ainda mais a eficácia e aplicabilidade da geração de genótipos sintéticos do modelo, explorando a geração multimodal e o melhoramento do desempenho do pré-treinamento. Com essa abordagem baseada em aprendizagem profunda, expandimos os limites da geração de genótipos sintéticos e impulsionamos o progresso da pesquisa genética.

Palavras-Chave

Aprendizagem profunda, Modelos generativos, Redes Generativas Adversarias, Estudo de associação do genoma completo, Coorte caso-controlo

Abstract

Genotype generation is an useful complement for genetic research, which offers opportunities for data augmentation and privacy preservation of Genome-Wide Association Studies (GWAS). With this study, we proposed a Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) framework for synthesizing realistic genotype data. Our approach consisted of two separate steps: a self-supervised pretraining process and the WGAN-GP training process. Throughout the pretraining process the generator acts like a decoder-like model, mapping the lower dimensional summary statistics to the original genotype data. This process serves as a strong initialization for the subsequent WGAN-GP training process, where the goal is to make the generator learn to synthetize diverse and realistic genotypes. The results of our proposed framework aided with the information encoded in the summary statistics file demonstrate favourable outcomes, highlighting promising genetic statistics values and utility of the synthetic data. Nevertheless, the various discrepancies observed with the principal component analysis (PCA) plots and divergence in validation hints towards multiple future areas of improvement. We aim to further advance the model's synthetics genome generation efficacy and applicability, by exploring multi-modal generation and fine-tuning pretraining. With this Deep learning based approach, we push the boundaries of synthetic genotype generation and foster the progress of genetic research.

Keywords

Deep Learning, Generative Models, Generative Adversarial Networks, Genome-wide Association Studies, Case-Control Cohorts, Genotype-Informed Synthesis,

Contents

Abbreviations

Adam Adaptive Moment Estimation

AI Artificial Intelligence

ANN Artificial Neural Network

BBB Blood-Brain Barrier

CNN Convolution Neural Network

DL Deep Learning

FCNN Fully Connected Neural Network

GRU Gated recurrent units

GWAS Genome Wide Association Study

KNN K-Nearest Neighbors

LSTM Long Short-Term Memory

ML Machine Learning

MSE Mean Squared Error

NLP Natural Language Process

RF Random Forest

ReLU Rectified Linear Unit

RL Reinforcement Learning

SGD Stochastic Gradient Descent

SVM Support Vector Machine

SVR Support Vector Regression

tanh Hyperbolic Tangent

HMM Hidden Markov Model

GMM Gaussian Mixture Models

List of Tables

List of Figures

1

Introduction

Biological systems are complex and connect different biological entities within and across different scales. These biological scales are typically described as atomic/molecular, subcellular, cellular, organism, population, communities, and ecosystems. In each scale, researchers usually focus on the interaction of their units [1]. These systems are difficult to understand and have millions of interactions that occur throughout the lifespan of all their 'members', but by increasing our knowledge on how each scale works, we can better integrate biological data across different scales. Resulting in a more nuanced and complete view of biological systems, and consequently a departure from this view of compartmentalized scales.

To have an idea of how these systems work, one needs to first understand what the building blocks of biology are. On the subcellular and cellular scale, the building blocks consist of the four main classes of biological macromolecules: proteins (polymers of amino acids), carbohydrates (polymers of sugars), lipids (polymers of lipid monomers) and nucleic acids (DNA and RNA; polymers of nucleotides).

All of these play important roles on the physiologic functions necessary for the life of an organism. For this work, a special focus will inside on nucleic acids. These biomolecules, comprised of deoxyribonucleic acid (DNA) and ribonucleic acid (RNA), are the biological macromolecules that are considered the molecular repositories of genetic information. The structure of proteins and every biomolecule and cellular component is programmed into the sequence of nucleotides (the monomers of these macromolecules). Throughout the lifespan of a biological entity, this information is accessed, interpreted, and changed by a range of factors. These processes provide a biological entity with the ability to store and transmit genetic information from one generation to the next.

The genome is the complete set of genetic information of an organism or cell. This

provides all the instructions an organism needs to function throughout its life [2]. Single or double-stranded sequences of nucleic acids store this information in a linear or circular conformation. Methods to determine this sequence from the DNA molecule, *i.e.* genome sequencing methods, have been a matter of interest for life science researchers for decades. As, this sequence can show us how all the intricacies of biological systems are coded with the four nucleotides present in the molecule.

In 1953, Watson and Crick, supported by unpublished research of Wilkins and Franklin, published the papers revealing the double helix structure of the DNA molecule to the world and the possible genetics implications of such structure [3, 4]. In their work, they postulated that the *sequence of bases on a single chain does not appear restricted in any way* [3]. This affirmation highlights the possible role that the DNA sequence has on storing the genetic information of organisms, and further show the importance of figuring out this exact sequence.

In the following decades, a multitude of researchers, such as Allan Maxan, Walter Gilbert, and Frederick Sanger, developed new DNA sequencing methods that continued to reduce the time needed to sequence the genome of a biological entity. This era is called the First Generation of sequencing methods. Their collective work was fundamental to the appearance of the next-generation sequencing (NGS) methods, marked by the introduction of the pyrosequencing technique. The key difference between the methods of the first generation and the methods of the next generation is the parallelization of the reactions needed to obtain the sequence, this significantly decreases the amount of time needed and allows automatizing the process. These features allowed companies like Roche and Illumina/Solexa to implement this technology in an automated system and put it on the market.

Recently, even quicker methods started to appear leading to the current generation of sequencing methods, the Next-Next Generation or Third Generation (TGS). What differentiates the TGS from the NGS methods, besides the single-molecule sequencing method used, is the possibility to sequence much longer reads, which helps to overcome technical and computational limitations of NGS-based assemblies and again a significant increase in speed, which translates to an even shorter amount of time needed to obtain a complete sequenced genome.

These developments and advances in sequencing and mapping methods, coupled with their ever-increasing speed and cost-effectiveness, have significantly increased the amount of genomic data that life sciences research fields can use for a multi-

tude of purposes, such as analysing the effect of drugs on an individual's genomic expression or looking for certain mutations associated with a disease, all to further our understanding of biological systems. However, the exponential growth of the amount of data available raises the need to work on four key phases when dealing with data: Data Acquisition, Data Storage and Management, Data Distribution, and Data Analysis [5]. While data acquisition and storage are factors that already have putative solutions with favourable results, data distribution, and data analysis are two phases that are what one might call "a bit problematic" when we are dealing with genomic data.

Regarding distribution, while some types of genomic data that are obtained from sequencing and following downstream analysis are safe to store and distribute when dealing with human cohorts, ethical, and privacy concerns are something researchers must consider, especially, in the case of raw DNA sequences. It has been shown that it is possible to identify a particular person in a cohort even with personal information obfuscated [6, 7], in addition to that, there has been an increase in legislation regarding the privacy of huge amounts of data generated every day, not only in genomics, but other fields *e.g.* social media.

Regarding data analysis, with the zettabytes of genomic data being generated each year, it is an impossible task to manually analyse and extract useful information from all of it. Therefore, researchers have been looking for novel ways to analyse and process this huge quantity of available data. Machine learning has been a solution that rapidly spread and is now ubiquitous in genomics, mainly because of its ability to manage large datasets and make accurate predictions. Also, with recent developments in deep learning, there has been a huge output of research regarding models that can be used to anonymize data by creating an artificial representation of that same data.

By exploring the anonymization capabilities of deep learning models, researchers would have a set of tools that would allow them to publicly store and distribute human cohorts' datasets while preserving the patient's right to privacy. This also allows a more complete set of data to be available for other researchers to fully explore the information encoded in these data sets.

1.1 Motivation

GWAS are studies where the researchers have to deal with the distribution and analysis problem previously presented. These problems lead to low availability of data to other researchers to use on their independent analyses and can contribute to the replication crisis. While there has been an effort to deal with these problems by using summarized representations of the original GWAS data, being able to generate new datasets with similar characteristics to the original GWAS data, according to the metrics from its summarized files would be beneficial to the landscape. That way one could use the summarized file to obtain an artificial approximation of the original GWAS data, this would solve the problem of privacy and analysis, while also contributing to data storage and management.

1.2 Objectives

In this work, we propose a deep learning-based framework to augment genomic datasets based on their calculated summary statistics. The main objective of our framework is to generate synthetic data that exhibits the same properties as the original dataset and therefore overcome the limitation of lack of available genomic data in biological contexts. To reach this goal, a review of the generative modelling landscape was performed, where previous works pertaining to generation of synthetic genomic data are presented to identify architectures commonly used in the field. Then, the development of a Wasserstein Generative Adversarial Network model to generate synthetic genomic data was done with a subsequent validation of the results obtained, comparing a subset of the original data with a synthetic sample generated with the associated summary statistics.

1.3 Document Structure

This document is structured in the following way. Chapter 2 is to provide the context needed for one's ability to comprehend this work, comprised of an overview of subjects focusing on GWAS and Machine Learning methods. Chapter 3 is to provide a detailed overview of the methodology, datasets, and architectures used for the work presented in this document. Chapter 4 presents and discusses the results obtained from the methodologies described in the previous chapter. Lastly, Chapter 5 is to showcase the conclusions and further work regarding this subject.

2

Context

2.1 Genomics

Genomics is a sub-field of computational biology that concerns itself with studying the ever-increasing amount of data generated by sequencing methods to get a better insight into the structure, function, evolution, mapping, and editing of genomes. While genetics refers to the study of individuals and/or groups of genes, genomics aims to have a complete characterization of the full set of an organism's genes, *i.e*, its genome. For that, researchers use various techniques such as genome assembling, genome annotation, and variant calling, that can extract the genetic information contained within the organism's DNA.

Through these techniques and subsequent downstream analysis, researchers can find and investigate possible relationships between genes and certain traits, gain a better understanding of mechanisms that underlie complex and/or inherited diseases, and allow for more personalized medical support as it made possible to develop tailored treatments based on an individual's unique genetic makeup. With all this information at the researchers' hands, genomics has the potential to aid the development of various sectors like healthcare, agriculture, ecology.

2.2 Genome Assembly

The process of assembling a genome starts with using the DNA reads obtained with sequencing methods. For DNA, the length of these reads can vary between a few hundred base pairs to around thirty thousand base pairs in the function of the generation of the sequencing method used. Given that they frequently come from various sections of the genome, these reads are next examined and aligned to identify any regions that coincide. Researchers can put the DNA fragments back together to

form contigs, which are larger DNA lengths created by finding overlapping sequences.

However, putting together a full genome might be difficult because of repetitive areas or gaps in the sequencing data. Advanced algorithms and computational techniques are used to resolve ambiguities, fix errors, and scaffold the contigs using extra information like mate-pair reads or optical mapping to manage these complications.

The final output of the genome assembly process is a high-quality, ordered representation of the organism's genome, providing a valuable resource for understanding its genetic makeup, evolutionary history, and functional elements.

2.3 Variant Calling

Variant calling is a process in genomics used for identifying and cataloguing the differences between the observed sequencing reads and a reference genome, meaning that this process identifies the genetic variations of an individual or a certain population of interest. These variants can be expressed in several ways, single-nucleotide polymorphisms (SNPs), indels (insertions or deletions), and even other larger changes in the DNA sequence such as CNV (copy number variants) or Structural variations. This is one of the processes that are crucial for attaining a better understanding of a plethora of characteristics pertaining to genetic diversity, evolutionary patterns, disease susceptibility and personalized medicine.

The standard form for this kind of data is the Variant Call Format, or VCF file. This file is used to store the meta-information, header and data lines containing the genome information. It can also contain the genotype information of the samples for each position.

```
##fileformat=VCFv4.2
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6,species="Homo sapiens">
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
```

```
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT NA00001 NA00002 NA00003
20 14370 . G A 29 PASS NS=3;DP=14;AF=0.5;DB;H2 GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1:43:5:.,.
20 17330 . T A 3 q10 NS=3;DP=11;AF=0.017 GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3 0/0:41:3
20 1110696 . A G,T 67 PASS NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2 2/2:35:4
20 1230237 . T . 47 PASS NS=3;DP=13;AA=T GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2
20 1234567 . GTC G,GTCT 50 PASS NS=3;DP=9;AA=G GT:GQ:DP 0/1:35:4 0/2:17:2 1/1:40:3
```

As presented on this example, the VCF file has lines starting with "##", preceding its meta-information, here, information pertaining to the INFO, FILTER, and FORMAT entries used in the body of the file are described. After the meta-information lines, comes the header line, this line specifies the order for the following data lines. This line has a mandatory fixed number of eight fields:

- #CHROM – chromosome: An identifier from the reference genome or an angle-bracketed ID String ("¡ID¿") pointing to a contig in the assembly file.
- POS – position: The reference position, i.e. the start coordinate of the variant.
- ID – identifier: Semicolon-separated list of unique identifiers where available. If this is a dbSNP variant, it is encouraged to use the rs number(s).
- REF – reference base(s): The reference allele is the allele found in the reference genome. It is not necessarily the major allele. Each base must be one of A, C, G, T, or N. Multiple bases are permitted.
- ALT – alternate base(s): The alternative allele is the allele found in the sample of interest. Comma-separated list of alternate nonreference alleles. Options are strings made up of the bases A, C, G, T, N, or *.
- QUAL - quality: Phred-scaled quality score for the assertion made in ALT.
- FILTER – filter status: PASS if this position has passed all filters, FAIL if it has not, it may also present custom flags, depending on the quality control and filtering tools used on the data.
- INFO – additional information: Allow us to provide further information on the variants. INFO fields are encoded as a semicolon-separated series of short keys with optional values defined in the metainformation portion of the file.

If there is genotype information present, two additional columns can appear:

- FORMAT – Information about the following columns specifying the data types and order, using keywords reserved to this field, like:

  - GT – genotype: encoded as allele values separated by either of / or |. The allele values are 0 for the reference allele (what is in the REF field), 1 for the first allele listed in ALT, 2 for the second allele listed in ALT and so on. For diploid calls, examples could be 0/1, 1 | 0, or 1/2, etc.
  - GQ – genotype quality: encoded as a phred quality $-10 \log_{10}$p(genotype call is wrong, conditioned on the site's being variant)
  - DP – read depth at this position for this sample.
  - HQ - haplotype qualities: two comma-separated phred qualities.
- NA00001 – individual identifier: Show the data in the format specified by the FORMAT column

Following the meta-information and header lines, the data lines contain all the data described and formatted by the previous lines.

2.4 GWAS

Genome-wide association studies, or GWAS, aim to identify associations of genotypes with phenotypes by comparing the differences in allelic frequency of genetic variants between individuals with similar ancestry but different phenotypes. The genetic variants that the majority of GWAS report are in the form of single-nucleotide polymorphisms (SNPs). Single-nucleotide polymorphisms (SNPs) are base-pair (bp) variations at certain locations in the genome. GWAS reports groups of correlated SNPs that show a statistically significant association with a certain trait of interest [8]. Results from these kinds of studies have led to the discovery of many variants that underlie complex genetic disorders and/or traits and the subsequent publishing of the data obtained allows the creation of databases of human genetic variations for further analysis, raising the possibility of finding even more links between genetic variants and unexplored traits.

However, besides the already mentioned problems regarding storage and management, sharing GWAS data raises certain privacy and ethical problems. For example, in the European Union, the increase of privacy protection legislation such as the General Data Protection Regulation (GDPR) requires the de-identification and consent of the patients, which complicates sharing and publication of this kind of data. On the other hand, GWAS summary statistics (files typically including in-

formation such as odds ratio (OR)/effect size (beta), standard error (SE), p-values, and case/control sample sizes for each SNP being analysed) are becoming increasingly available thanks to endeavours like the GWAS Catalog, GWAS Atlas, and OpenGWAS databases.

2.5 GWAS Summary Statistics

Summary statistics are the aggregate p-values and association data for every variant analysed in a GWAS. The information contained in these summary statistics represents a potential to extend the already significant power of GWAS and improve disease understanding. Recently, a number of techniques have been developed to harness the data present in these GWAS summary statistics to further understand mechanisms of complex human diseases, the identification of drug targets and postulate eventual disease risk [9–11]. However, one of the challenges associated with sharing this data is that there is no standard format for this type of file, resulting in a lack of guidelines for the generation and sharing of these files.

The GWAS Summary Statistics File (GWAS-SSF) format is a collaborative effort to standardize the format in which the summary statistics of a GWAS are reported. The specification of this format is detailed in [12]

2.6 Machine Learning

As it was previously referred, the sheer amount of data being generated around the world is around 120 zettabytes. This makes it impossible to manually explore and extract meaningful information efficiently, however this kind of problem is a prime opportunity to use machine learning algorithms.

Machine learning seeks to automatically learn meaningful relationships and patterns from examples and observation [13]. During the last decades, the field of ML has brought forth a variety of remarkable advancements in sophisticated learning algorithms and efficient pre-processing techniques.

2.6.1 Types of Machine Learning

When dealing with a machine learning related problem or task, the problem and the type of data associated usually dictates the type of algorithms that it will be

used to solve it. These algorithms are mainly divided into four categories: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [14].

Supervised Learning

In the supervised learning category, the algorithms need to have labelled data, that is, raw data that has been assigned at least one label to add context or meaning to it. This label is what the various supervised learning algorithms use to learn the patterns associated to a certain label. The use of supervised learning algorithms is for two different tasks associated with the type of label that the dataset has: classification or regression

For example, if we consider a dataset of emails in which each email is either assigned the label "spam" or "not spam". To be able to filter these spam emails, we need to have a model that is capable to classify the emails as spam or not. To train this model we need to feed it a portion of this email dataset, usually called the training set, for it to learn to associate patterns in the emails to the assigned label.

Unsupervised Learning

In contrast to the algorithms in the supervised learning category, unsupervised learning algorithms do not need to have labelled data, instead algorithms from this category learn hidden patterns in the data that, at plain sight, might not be evident for us. The use of unsupervised learning algorithms are typically associated with grouping similar examples together, i.e. clustering, dimensionality reduction, and density estimation.

Semi-supervised Learning

The semi-supervised learning algorithms are a kind of algorithms that can be understood as the middle point between the supervised and unsupervised learning algorithms category. The usage of this type of algorithms usually entails a problem or task where the amount of labelled samples in the dataset is low and consequentially the number of unlabelled samples is high. These algorithms are able to make use of this additional unlabelled data to better capture the shape of the underlying data distribution and generalize better to new samples.

Reinforcement Learning

Algorithms in the reinforcement learning category, like the unsupervised learning ones, do not need to have labelled data to be able to learn, this means that the learner has no knowledge of which actions to take until it is given a situation. These actions taken by the learner may affect situations and their actions in the future. This class of algorithms learn by interacting with its surrounding environment via these actions, balancing exploration (trying new strategies) and exploitation (making use of known successful techniques).

2.6.2 Classification & Regression

As it was previously said, in the supervised learning category, we can further separate the algorithms between classifiers and regressors according to the type of label associated with the dataset of interest which then dictates the type of task the algorithm will have to perform.

Classification

When the class labels of the dataset of interest are categorical values, the type of algorithm used for that scenario is a classifier. These class labels are discrete and unordered values that can be understood as groups to which the assigned data points belong. The goal here is to be able to predict the categorical class label of a new sample or data point based on the past observations used for training the algorithm.

Regression

Like the classifiers in the previous section, a regressor is an algorithm that belongs to the supervised learning category of algorithms, but there is a key difference between these algorithms *i.e.*, the regressors and the classifiers. While the classifiers need a discrete/categorical class label to be able to learn from and later be able to make predictions based on the patterns it observed, when dealing with regressors, instead of a discrete class label it needs a continuous value to be able to learn from and make predictions based on the patterns it was able to "catch".

One important note is that these categories are not mutually exclusive, meaning that some can be considered classifiers and regressors. In cases like this, it is the task and/or the objective of that task that dictates if the algorithms are going to be used as a classifier or regressor.

2.7 Deep Learning

Up until this point several notions regarding important paradigms encompassed in the field of Machine Learning have been briefly explained, but due to significant advances and the uprising of tools that integrate this kind of algorithms in our everyday life, in the form of facial recognition technology, self-driving cars, and others, a certain subcategory of machine learning algorithms has been the centre of attention for the general population and also from researchers from various areas like mathematics, physics, chemistry, and biology [15–19]. This subcategory is called Deep Learning.

Deep learning, as a subcategory of machine learning algorithms, take inspiration from the human brain and the method this organ uses to transmit information through the 80 billion of brain cells, the neurons [5, 20]. Differently to traditional machine learning, deep learning does not need to focus on feature engineering, but directly extracts features due to the different abstractions of the input data and the incremental complexity of the representations in each layer [21]. That is, deep learning cuts down the work of designing feature extractors for each problem [22]. Deep learning has been widely used in many fields, such as object detection, face recognition, image segmentation, machine translation, and text classification [23].

2.8 Notorious Models

Linear Regression

One of the most basic but at the same time most important algorithms in machine learning is the Linear Regression [24, 25]. Like the name suggests, this algorithm belongs to the regressor subcategory of the supervised family of the machine learning algorithms. This means that it is usually used to predict values within a continuous range, (e.g. the price of a certain house) based on the patterns observed from the data that it was trained with.

Logistic Regression

Similar to linear regression, logistic regression (Logit) [26] is also used to estimate the relationship between a dependent variable and one or more independent variables, but it is used to make a prediction about a categorical variable versus a continuous one. Making it a supervised machine learning commonly used for classification tasks.

Decision Trees

Decision trees [27] are algorithms that can be used for classification and regression tasks. It is a flowchart-like structure that uses yes or no questions to arrive at a final decision.

The algorithm starts from the root node of the tree and will recursively ask questions about the features in the data. The algorithm then follows the branch that leads to the answer to that question. This process continues until the algorithms reaches a leaf node, which represents a final decision.

Decision trees can be used for a variety of tasks. Their popularity is justified for being easy to understand, robust to noise, and scalable. Yet, they are prone to overfitting and are not as accurate as the following machine learning algorithms.

Random Forest

Random Forest (RF) [27] is another algorithm that is popular, as it can be used for classification and regression problems in machine learning. This algorithm uses ensemble learning to build the model. Ensemble learning consists of a technique that combines multiple models into one with the goal of improving the overall predictive performance. In the case of the random forest algorithms, the individual models are the decision tree algorithms explored in the previous section.

The strength of RFs lies in its ability to reduce the overfitting that an individual decision tree is prone to. By using random subsets of data and features, it generates diversity among the trees, which helps to capture different aspects of the underlying patterns of the data, ultimately improving generalization.

Beyond their robustness to overfitting this algorithm, it is also easy to interpret and understand. However, they can be computationally expensive to train and are sensitive to hyperparameters, like for example the number of estimators pertaining to the number of individual decision trees in the forest and maximum depth pertaining to the maximum depth of each individual decision tree.

Support Vector Machine

Support Vector Machines (SVMs) [28] are another class of machine learning algorithms that can be used for classification and regression. They work by finding the hyperplane that best separates different classes of data, with the hyperplane being

the line or plane that splits the data into different regions. Each separated region is a representation of a different class.

These algorithms try to maximize the margin, *i.e.* the distance between the hyperplane and the closest data points of each class. A large margin translates to a higher confidence the SVM has in its classification.

SVMs can be used as a classifier for problems including spam filtering, image and text classification, while also being a popular for regression problems, like predicting stock prices. They are a powerful machine learning algorithm known for being robust and scalable, but they can be computationally expensive and not as interpretable as some other machine learning algorithms.

Multilayer Perceptron

The multilayer perceptron (MLP), feedforward neural networks (FFNN) or deep feedforward network (DFFN) is the quintessential deep learning model [29] as it forms the basis of all neural networks and have greatly improved the performance of machine learning when applied to classification and regression problems. A typical MLP is a fully connected network consisting of an input layer, one or more hidden layers, and an output layer, represented in the figure 2.1. These layers operate on the outputs of its preceding layer and all of them, except the input layer, uses a nonlinear activation function.

To learn, an MLP uses the backpropagation algorithm. This iterative process adjusts the weight values of the model. The MLP is sensitive to the scale of the features and to achieve satisfactory results, a variety of hyperparameters have to be tuned, such as the number of hidden layers, the number of nodes (or neuron) of each layer and the number of iterations (epochs) of the training process.

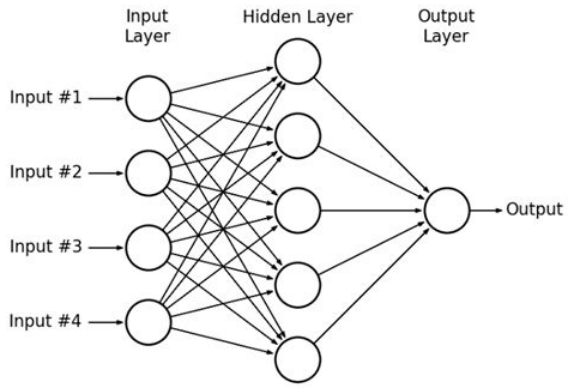

Figure 2.1: Representation of a Multilayer Perceptron (MLP)

Convolutional Neural Network

Convolutional Neural Networks (CNNs) [30], are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels.

CNNs are comprised of three types of layers. These are convolutional layers, pooling layers and fully-connected layers. When these layers are stacked, a CNN architecture has been formed. A simplified CNN architecture for MNIST classification is illustrated in the figure 2.2

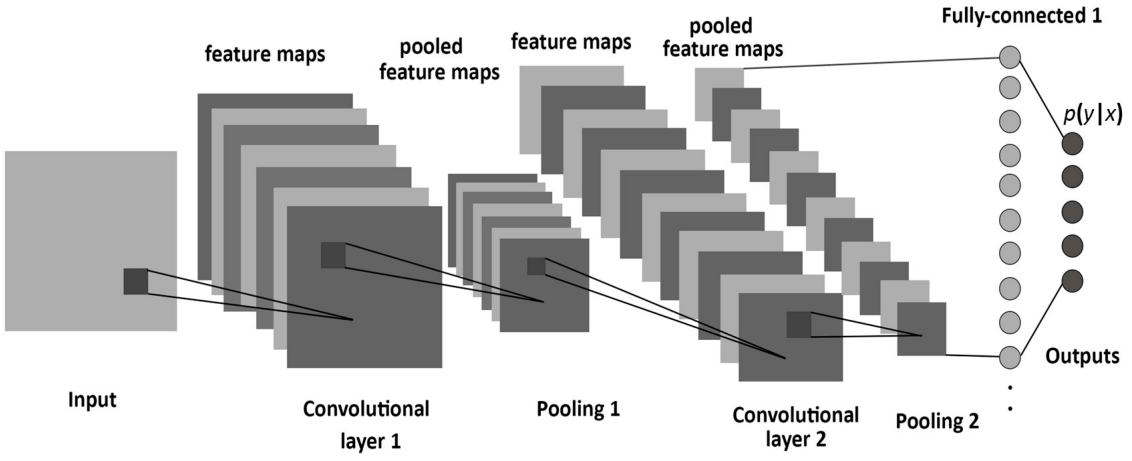

Figure 2.2: Diagram of a Convolutional Neural Network

To understand how CNNs are able to successfully process data in a grid-like topology, the convolution, and pooling are operations that one needs to know.

Convolution

Convolution is an operation commonly used in signal processing and deep learning. This operation is crucial to extract meaningful patterns and features from the input data. In the context of image processing, convolutions involve sliding a small filter (also referred to as a kernel) over an input image, performing element-wise multiplication between the filter and the regions of the image, and summing them, resulting in a new feature map. The use of convolutions allows the detection of edges, textures, and other patterns which captures complex features. Convolutional Neural Networks (CNNs) leverage this operation to automatically learn and discover meaningful features from raw images.

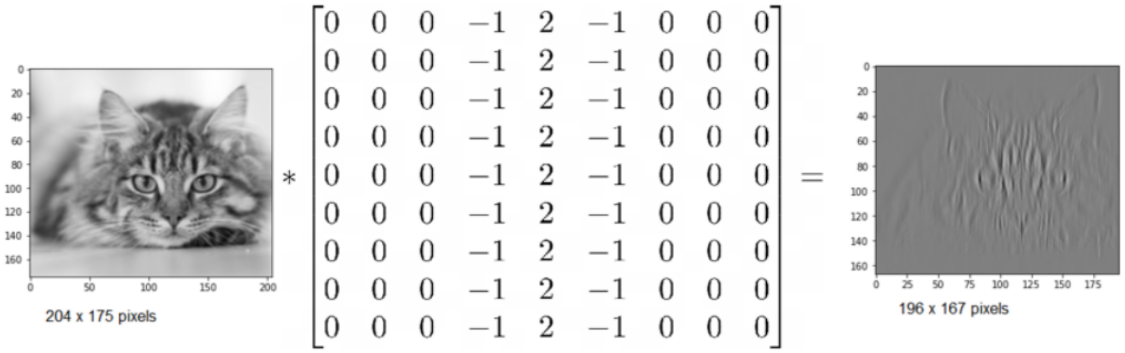

Figure 2.3: Representation of the Convolution operation

Pooling

A limitation of the feature map output of convolutional layers is that they record the precise position of features in the input. This means that small movements in the position of the feature in the input image will result in a different feature map. This can happen with re-cropping, rotation, shifting, and other minor changes to the input image.

A common approach to address this problem is to perform some kind of down sampling, that is, the creation of a lower resolution version of an input signal that still contains the large or important structural elements, without the fine detail that may not be as useful to the task. A robust and common approach for down sampling is to use the pooling operation.

The pooling operation shown in the figure 2.4 is specified, rather than learned. Two common functions used in the pooling operation are:

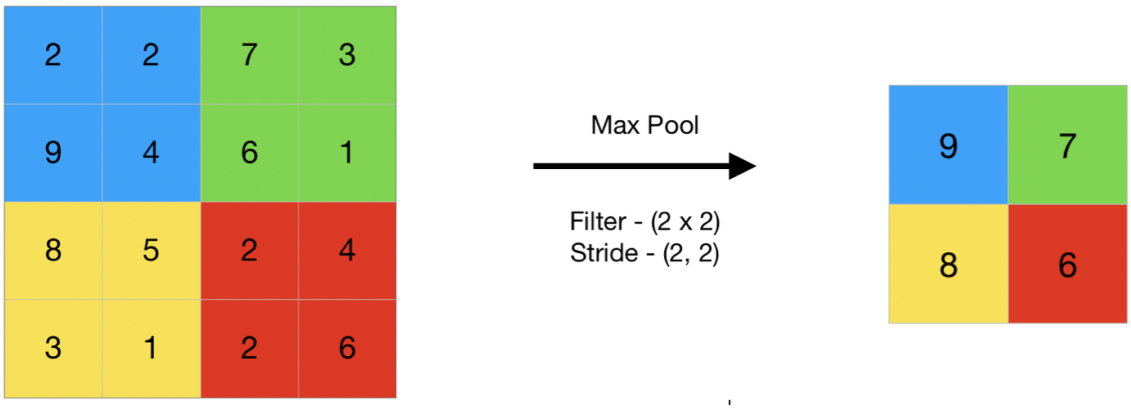

Figure 2.4: Representation of the Pooling operation, specifically the max polling operation

- **Average Pooling**: Calculates the average of each patch of the feature map.
- **Maximum Pooling**: Calculates the maximum value of each patch of the feature map.

Recurrent Neural Network

Recurrent Neural Networks (RNNs) [31] are a family of neural networks for processing sequential data. A recurrent neural network is a neural network that is specialized for processing a sequence of values $x^{(1)}, \ldots, x^{n}$. Just as convolutional networks can readily scale to images with large width and height, recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialization [29]. The basic structure of a recurrent neural network is shown in the figure 2.5.

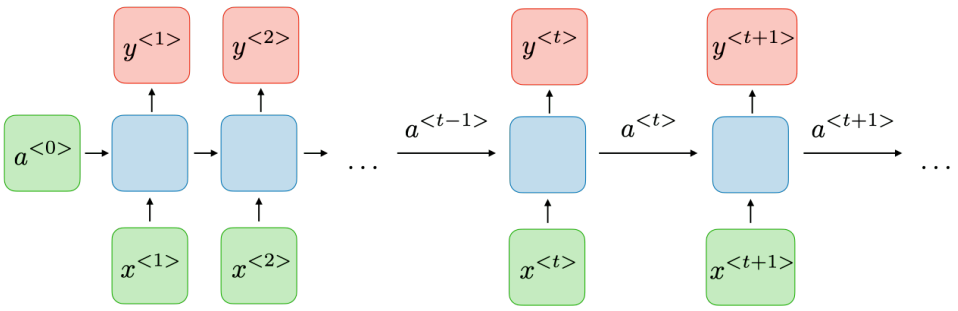

Figure 2.5: An RNN and the corresponding computational graph unfolded in time

Variational Autoencoders

Variational Autoencoders (VAEs) [32] are a family of generative models, but first to understand how the variational autoencoder architecture works, having basic notions about the architecture of a vanilla autoencoder can be useful.

An autoencoder is a neural network that is trained to attempt to copy its input to its output. Its basic structure is defined in the figure 2.6. This architecture has a hidden layer h that encodes a latent representation of the received input. This architecture can be described as two parts, an encoder $h = f(x)$ and a decoder $x' = g(h)$

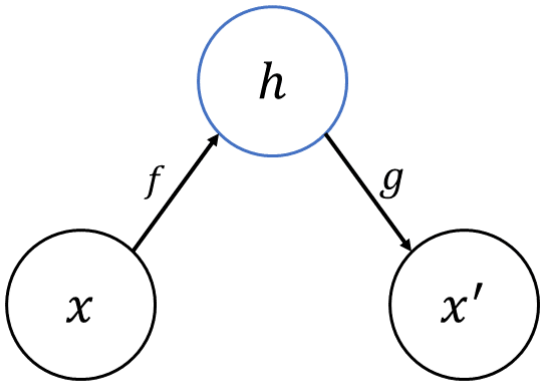

Figure 2.6: Structure of an autoencoder

Traditionally, autoencoders were used for dimensionality reduction or feature learning. Recently, theoretical connections between autoencoders and latent variable models have brought autoencoders to the forefront of generative modelling, with the appearance of the VAEs, represented in the image 2.7.

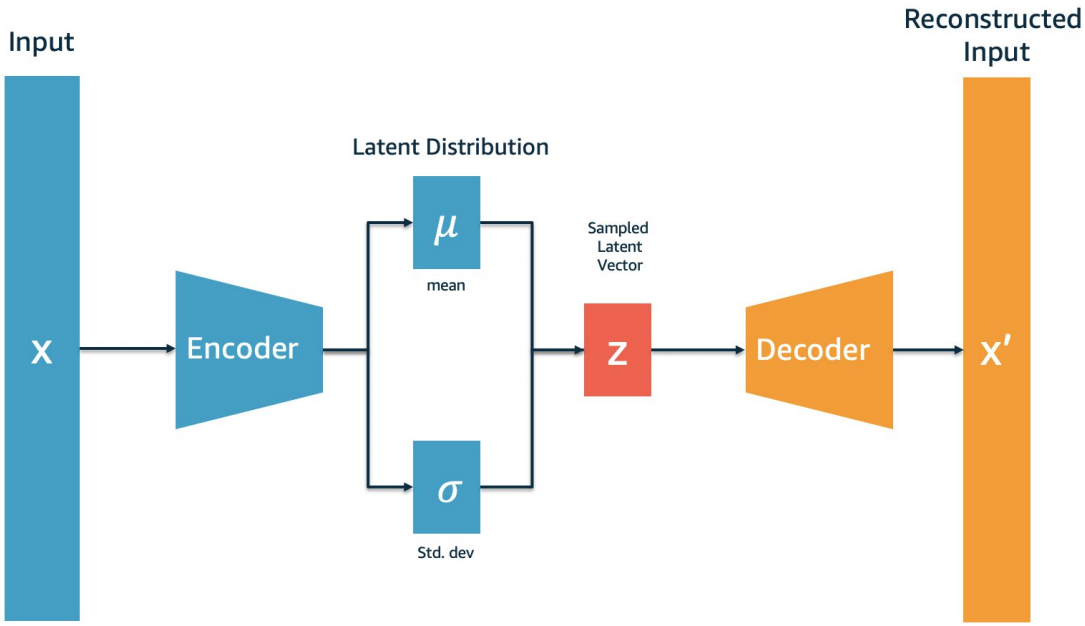

Figure 2.7: Diagram of a Variational Autoencoder (VAE)

For the VAEs, instead of directly generating a latent representation directly from the input, the encoder outputs a mean μ and a standard deviation σ which then both are sampled from a Gaussian distribution with mean μ and standard deviation σ. This key difference makes their latent spaces to be continuous, allowing easy random sampling and interpolation.

Generative Adversarial Networks

Generative Adversarial Networks (GANs) [29] are a class of deep learning models introduced by Ian Goodfellow and his colleagues, that can generate new, previously unseen examples of data. They have been used for a wide variety of tasks. The basic idea behind GANs is to train two neural networks, a generator, and a discriminator, in an adversarial manner, similar to a zero-sum game, where the loss of a player is the gain of the other [33].

The generator produces new examples of data, and the discriminator tries to determine whether each example is real or fake. Through this process, the generator learns to produce more realistic examples, and the discriminator learns to better distinguish between real and fake examples.

The architecture of a GAN typically consists of a generator network and a discrim-

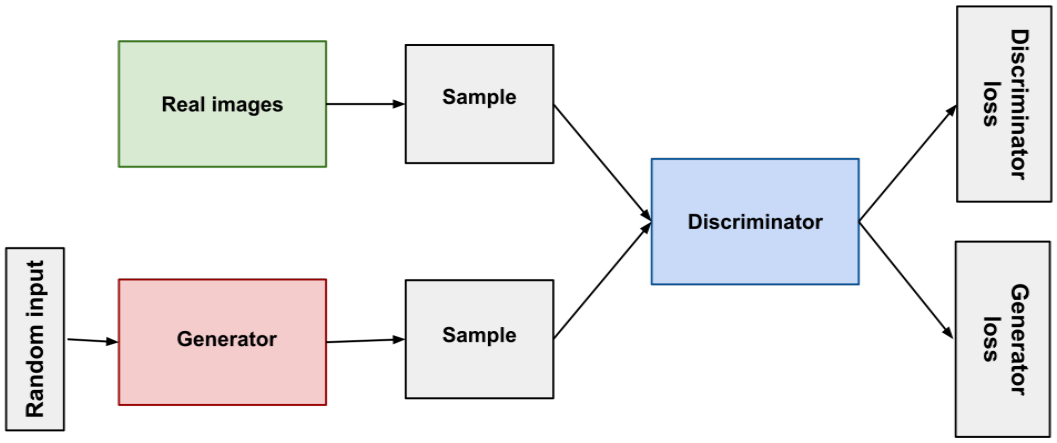

Figure 2.8: Diagram of a Generative Adversarial Network (GAN)

inator network. The generator network maps a random noise vector to an example of data, while the discriminator network takes an example of data and outputs a probability of whether it is real or fake.

3

State of the Art

Generative modelling has been a compelling paradigm in the field of ML. This form of unsupervised learning revolves around the use of algorithms with the capability to learn the underlying statistical patterns of a dataset and with this learned knowledge the algorithms can subsequently generate new data samples based on the distribution of the original data [34].

For instance, in an image classification scenario, an image is fed to a model and this model outputs the category of the image. This process is flipped in generative modelling. Instead of feeding an image to the model, it is fed a description and the model outputs an image corresponding to the description provided. As the overall field of machine learning continues to evolve, generative modelling has shown to be a powerful tool because of its several applications, such as the generation of realistic images. An example of the generative capabilities of this kind of modelling can be found on https://www.thispersondoesnotexist.com and https://www.whichfaceisreal.com/index.php [35,36]. These two website display high quality synthetic face images generated by a GAN.

The use of generative modelling in machine learning has been prevalent since the 1950s, with the development and application of algorithms like Hidden Markov Models (HMMs) [37] and Gaussian Mixture Models (GMMs). Models like this were usually tasked with the generation of synthetic sequential data such as text and time series [38–40]. However, the recent increase of interest in this category of machine learning algorithms can be associated with the ever-increasing quantity of freely available datasets as well as the appearance and subsequent development of deep learning algorithms.

The use of neural networks in the context of generative modelling has its roots in the 1980s. In that era, the goal was simply to be able to do supervised learning and

reinforcement learning while using less labelled data [34, 41]. This way researchers did not have to spend time and resources to collect labelled data or label themselves the unlabelled data that they had but at the same time could extract the underlying statistical information from this unlabelled data using the unsupervised generative models.

More recently the landscape pertaining to generative modelling has been the centre of attention of a large portion of the research around machine learning, with several kinds of architectures being developed and used for a myriad of purposes such as image synthesis, super-resolution, text-to-image and image-to-image conversion, in painting, attribute manipulation, pose estimation; video: synthesis and retargeting; audio: speech and music synthesis; text: summarization and translation; reinforcement learning; computer graphics: rendering, texture generation, character movement, liquid simulation; medical: drug synthesis, modality conversion; and out-of-distribution detection [34].

One family of generative models of interest are the Generative Adversarial Networks (GANs). Since their introduction to the machine learning landscape in 2014 by Goodfellow et al. [42], this architecture has achieved remarkable success in the computer vision field such as generating synthetic images [42], image style transfer, image inpainting, and image super-resolution.

However, there are intrinsic issues associated with this architecture. It is notorious for being prone to mode collapse, being slow and unstable to converge, and vanishing gradients [43, 44]. For that reason, in the following years modifications of the original architecture and/or the method of training that try to remedy these issues, were proposed. The implementation of the Wasserstein GAN (WGAN) [45] and Wasserstein GAN with gradient penalty (WGAN-GP) [46], are two examples that yield good results compared to their standard counterpart.

The Wasserstein Generative Adversarial Network (WGAN) training process like the standard GAN consist of a zero-sum game between two adversarial networks where one player's loss is the gain of another [33]. The key difference between the GAN and WGAN training procedure lies in the loss function used. The WGAN uses the Wasserstein-1 metric (W_1) also called Earth-Movers Distance (EMD), because it can be understood as the minimum work needed to transform one distribution to another, where work is defined as the product of mass of the distribution that has to be moved and the distance to be moved. This metric can be expressed in the

following manner:

$$W_1(P,Q) = \inf_{\pi \in \Pi(P,Q)} \sum_{x,y \in X} \pi(x,y) \cdot d(x,y)$$

where:

- $W_1(P,Q)$ represents the Wasserstein-1 distance between distributions P and Q.
- $\Pi(P,Q)$ is the set of all joint probability distributions $\pi(x,y)$ with marginals P and Q.
- $d(x,y)$ is the distance between elements x and y in the metric space (X, d).

The reason that the change of loss function was proposed is that the goal of the GAN training process is to minimize the Jensen-Shanon divergence between the distribution of the original data and the distribution of the generated synthetic data, but a problem arises during this process. As the critic gets better, the Jensen-Shanon divergence locally saturates, making the gradients converge to zero and vanishing. In contrast, such phenomenon doesn't happen when using the Wasserstein-1 metric. Another benefit of the Wasserstein-1 metric is that is continuous and almost differentiable everywhere [45], which allows the model to be trained to optimality. The Wasserstein distance is also considered a meaningful metric, *i.e*, as it converges to 0 as the distributions get close to each other and diverges as they get farther away.

Following the proposal of the WGAN, another proposition of the which expands the WGAN was made. While the original WGAN [45] improves training stability, there still are cases where it generates poor samples or fails to converge. The issues with WGAN arises mainly because of the weight clipping method used to enforce Lipschitz continuity on the critic. This new proposed WGAN, called Wasserstein GAN with gradient penalty (WGAN-GP), replaces weight clipping with a constraint on the gradient norm of the critic to enforce Lipschitz continuity [46]. This allows for more stable training of the network than the WGAN with weight clipping and requires very little hyperparameter tuning.

The use of deep generative models like the previously explored GAN, VAEs and Large Language Models, has proven to be useful not only to computer vision and natural language processing but also pertaining to biological and biomedical research [47–53] and also for a particular case, the generation of artificial genomes [54,

55].

The potential of generative modelling has been underexplored in genetics, which is surprising due to the difficulties of researchers in these areas to obtain and analyse genomics data due to its reduced access to many databases due to concerns about violations of individual privacy. With that in mind, Yelmen et al. [54, 55] proposed the use of generative models like the GAN, WGAN, and RBM for the generation of artificial human genomes. This approach proved to be able to learn the complex distribution of real genomic datasets and also be able to generate high quality artificial genomes that replicate genomic characteristics such as allele frequencies, linkage disequilibrium and population structure. These results using generative models, that are able to capture the underlying statistical characteristics of genomic data and generate samples with those same characteristics, only reinforce the potential of this subsection of machine learning has for the field of genomics.

4

Materials & Methods

4.1 Data description

The dbGaP dataset phs000473.v2.p2 was investigated and analysed for this work. It comprises 12,380 samples from a Swedish case-control Whole-Exome Sequencing project, including 6,245 controls, 4,969 SCZ patients, and 1,166 Bipolar disorder cases.

The cases of schizophrenia were identified using the Swedish Hospital Discharge Register. Such cases must have had at least two hospitalizations with a discharge diagnosis of schizophrenia to be included in this dataset. Regarding the controls, they were selected at random from population registries. The single exclusion criterion was hospitalization for SCZ. The participants in all samples were at least 18 years old, with both parents born in Scandinavia.

The original data had previously been filtered for a variant Phred-score quality (QUAL) greater than thirty percent. Variant sites having a mean read depth (DP) of less than eight or a genotype call rate of less than ninety percent were also filtered from our dataset.

The Variant Quality Scores Recalibration steps on the Genome Analysis Toolkit (Version 3.8) Best Practices Workflow were followed. Multi-allelic variants were separated and filtered according to the resultant genotype call rate.

For our work, we worked with the genotype call encoded in an additive format, meaning that `0` refers to a homozygous to the reference allele, `1` refers to a heterozygous with one reference allele and one alternative allele, and `2` refers to an homozygous alternative allele.

4.2 Genotype-phenotype association

For the genotypes of cases and controls, a chi-square test was run on all variations in a 3x3 contingency table, with the exclusion of bipolar samples. Sexual chromosomal indels and variations were also screened. The SNPs were annotated using the most recent Annovar version for the hg19 genome assembly. To include a larger and more representative number of variations in the subsequent analysis, the p-values were not adjusted for multiple comparisons. The total dataset had 18,970 variations derived from 9,160 genes.

4.3 Summary Statistics Calculation

The calculation of the summary statistic pertaining to the dataset phs000473.v2.p2. was performed with the plink2 program [56] with the resulting file was outputted in the GWAS-SSF format [12].

4.4 Pretrained Generator

Pretraining is a technique used to initialize a model's parameters before fine-tuning it to the target task. The goal of pretraining is to provide the model with a useful initial set of weights, which can speed up the convergence and improve performance on the target task.

With that in mind, in this work the process of pretraining was used in a self-supervised learning manner, where the generator is treated as a decoder-like model that maps the lower dimensional data of the summary statistics to the original data. This was done to prevent dealing with common problems of WGANs like mode collapse, slow training and stability issues, since the pretrained generator of the WGAN has a better starting point and also a good internal representation of the data ultimately leading to a faster convergence

The generator was designed with two portions. One portion f, that processes the summary statistics input $SumStats$:

$$\begin{cases} f_1 = GELU(FC_{4\to16})(SumStats) \\ f_2 = GELU(FC_{16\to32})(f_1) \\ f_3 = GELU(FC_{32\to32})(f_2) \\ ProcSumStats = GELU(FC_{32\to64})(f_3) \end{cases}$$

and the second portion g, that concatenates the processed summary statistics to a random vector z sampled from a Gaussian distribution $\mathcal{N}(0,1)$:

$$\begin{cases} z \sim \mathcal{N}(0,1) \\ in = ProcSumStats \oplus z \\ g_1 = GELU(FC_{100\to128})(in) \\ g_2 = GELU(FC_{128\to256})(g_1) \\ g_3 = GELU(FC_{256\to512})(g_2) \\ g_4 = GELU(FC_{512\to1024})(g_3) \\ y_{synth} = \tanh(FC_{1024\to5000})(g_4) \end{cases}$$

The pretraining process was performed using the parameters showcased in table 4.1, using the Binary Cross Entropy (BCE) loss function between the real genotype y_{real} and the synthetic sample y_{synth}. The BCE function is described in the following manner:

$$\text{BCE}(y_{real}, p(y_{real_i})) = -\frac{1}{N}\sum_{i=1}^{N} y_{real_i} \times \log(p(y_{real_i})) + (1 - y_{real_i}) \times \log(1 - p(y_{real_i}))$$

where:

N is the number of samples in the dataset,

y_{real_i} is the true binary label of the ith sample,

$p(y_{real_i})$ is the predicted probability of the ith sample belonging to the positive class.

Table 4.1: Hyperparameters used for the pretraining of the WGAN-GP generator

Parameter	Value
epochs	200
Generator Learning rate	1×10^{-4}
Optimizer	Adam

4.5 WGAN implementation

This architecture was implemented and trained using python-3.10 & pythorch-2.01 [57, 58]. The implementation of this work's WGAN is a Wasserstein GAN with gradient penalty (WGAN-GP) that consists of a critic which estimates the Wasserstein metric between real and generated data distributions, and a generator which generates new genomic data from a concatenation of summary-statistics file and a random vector with values sampled from a Gaussian distribution. The generator of the WGAN-GP follow the same structure of the pretrained generator previously described. The Discriminator d (usually called the Critic while using the WGAN architecture), has the following design:

$$\begin{cases} in = y_{real} \mid y_{synth} \\ d_1 = GELU(FC_{5000 \to 1024})(y) \\ d_2 = GELU(FC_{1024 \to 512})(d_1) \\ d_3 = GELU(FC_{512 \to 256})(d_2) \\ d_4 = GELU(FC_{256 \to 128})(d_3) \\ out = FC_{128 \to 1}(d_4) \end{cases}$$

A relevant characteristic to note is that the last layer of the Critic of the WGAN doesn't have an activation function like, for example, the commonly used sigmoid function. This is done to be able to calculate the Wasserstein-1 Distance between the synthetic data generated and the real data. The training process for the WGAN-GP was performed with the parameters showcased in the table 4.2

Table 4.2: Hyperparameters used for the training of the WGAN-GP model

Parameter	Value
epochs	200
Generator Learning rate	1×10^{-4}
Critic Learning rate	4×10^{-4}
Optimizer	Adam
beta_1	0.555
beta_2	0.9
lambda gp*	10

*value used on the gradient

4.6 Evaluation

To evaluate the performance of our generative model, we calculated a group of metrics pertaining to basic statistics regarding genomics data and compare them to the correspondent original data values. Also, as a complement we proceeded to simulate a scenario where there is a need to classify the diagnosis of a patient, to evaluate the utility of the synthetic data compared to its original counterpart.

The statistics that will be used as comparison between the real and the synthetic genomic data are:

Allelic Frequencies - the relative frequencies of an allele in a population, expressed as a fraction or percentage. if $f(0)$, $f(1)$, and $f(2)$ are the frequencies of the three genotypes of a particular variant. The frequency p and q correspond to the alternative allele and reference allele frequency, respectively. These frequencies are calculated in the following manner:

$$p = f(1) + 2 \times f(2)$$

$$q = f(1) + 2 \times f(0)$$

Minor Allele Frequency - is the frequency of the second most common allele in a given population. This value range from 0 to 0.5. With the allele frequencies p and q of a specific variant var the calculation of the MAF is:

$$MAF_{var} = min(p_{var}, q_{var})$$

Fixation Index - is the measure of population differentiation due to genetic structure. It ranges from 0 (i.e. no genetic differentiation) to 1 (complete genetic differentiation). To calculate it we considered the control and diagnosed case samples as two populations with their allele frequencies being p_{case}, q_{case}, $p_{control}$ and $q_{control}$. Now this statistic has the following formula:

$$H_S = \frac{Hs_{case} + Hs_{control}}{2}$$
$$H_T = 2 \times p_{total} \times q_{total}$$
$$F_{ST} = \frac{H_T - H_S}{H_T}$$

where:

$$H_S = \text{expected heterozygosity} = 2pq$$
$$H_T = \text{average expected heterozygosity}$$

5

Results

5.1 Model Training

As it was mentioned, in our work the generator was pretrained with the goal of having a better starting point and an encoded representation of the data to achieve a faster convergence for the subsequent training process of the WGAN.

During the self-supervised pretraining phase (figure 5.1), the generator showed to converge. As the generator was treated as a decoder-like model that maps the lower dimensional summary statistics to the original data, it demonstrated consistent progress in reconstructing the original data accurately. Here, the summary statistic acted as informative intermediaries, allowing the generator to learn how to decode the compressed information present in the summary statistics file. Analysing the figure, the downward trend signifies a consistent reduction in the discrepancy between the generated synthetic data and the original data. This convergence provided a strong starting point for the generator of the WGAN-GP training phase.

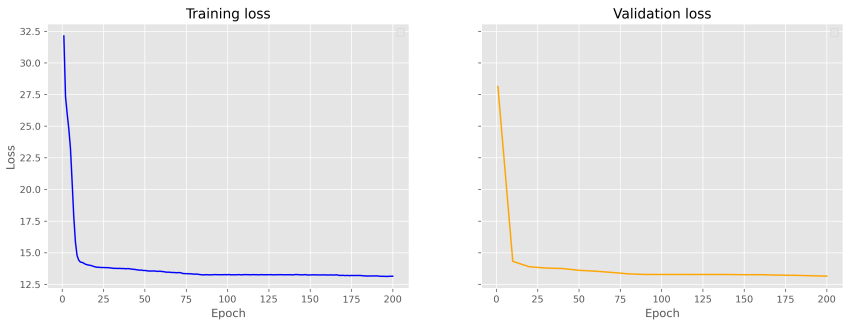

Figure 5.1: Evolution of the pretraining process of the generator

Throughout the WGAN-GP training phase, the generator showed promising hints of convergence with a diverging trend present in the validation values at the later

epochs. The loss curves displayed an evident initial reduction, demonstrating the generator's ability to generate realistic synthetic data. However, as training progressed, the validation value showed hints that it was starting to diverge from the training one.

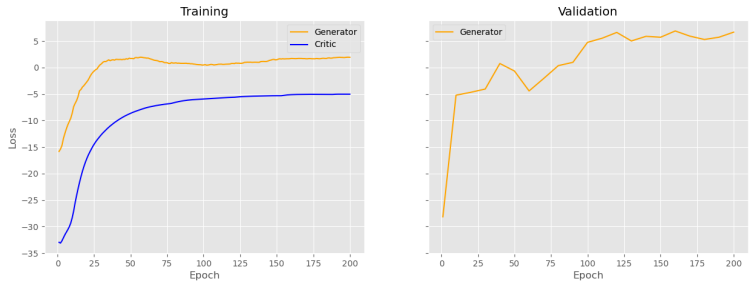

Figure 5.2: Evolution of the Wasserstein loss during the training process of the WGAN. On the left we have the training loss values of the Generator and the Critic. On the right we have the validation loss value of the Generator

5.2 Principal Component Analysis

Results of a Principal Component Analysis (PCA) can provide insight into the performance of the model and its ability to capture the overall structure of the data. By reducing the dimensionality of the data and projecting onto the principal components (PCs), we obtained a visual representation of the data's variability. The PCA plots in the figures 5.3 & 5.4 suggest that our model has learned some prominent features and structures from the original data.

Visually, the plots reveal that the synthetic data tends to condense around specific regions, suggesting that the model has successfully learned and generated instances that share similar patterns. Even though the clustering might hint that the synthetic data is focused on specific spots, it is important to consider the context and distribution of the original data. This could mean that the model is prioritizing regions of high density or that are essential to capture the overall characteristics of the data.

Coupled with the focus on specific spots, the marginal plots for PC1 and PC2 in the figure 5.3 provide further insight into the model's ability to capture the structure of the data. Inspecting the marginal plots, which provide a one-dimensional view of the data distribution along the two principal component, it is evident that the synthetic data aligns with the original data along PC2. However, when looking at

PC1, the PCA plot reveals deviations between the synthetic and original data. This deviation, supported by the calculation of the Wasserstein distance between the real data and fake data first component ($W_1 = 7.98$), suggests that our model may be struggling to accurately capture the underlying distribution of PC1 values.

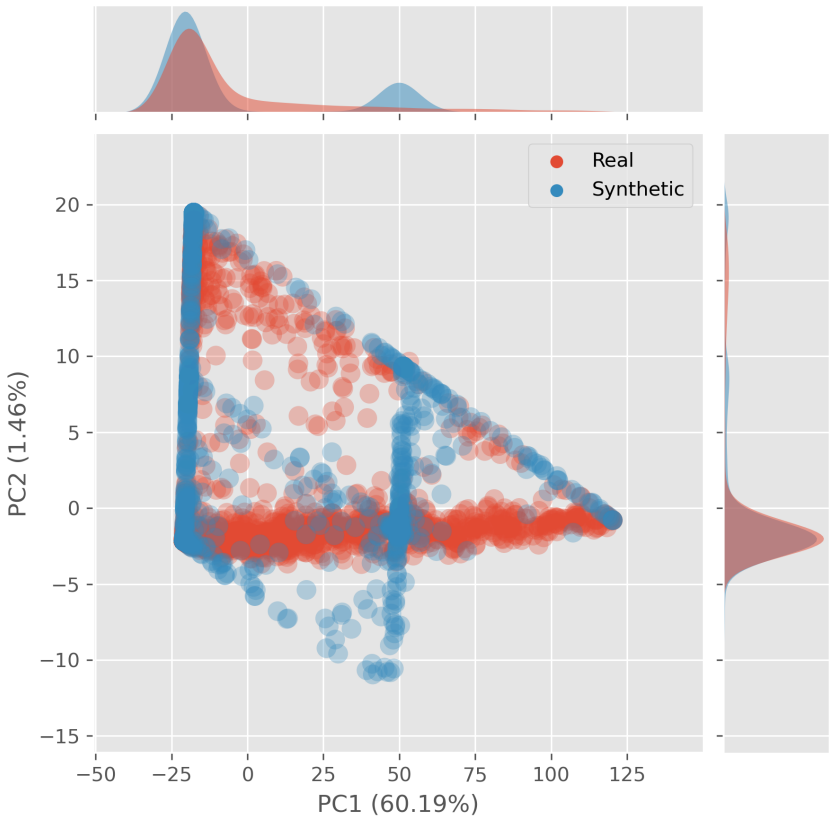

Figure 5.3: The first two principal components resulted from a Principal component analysis of the real and synthetic data

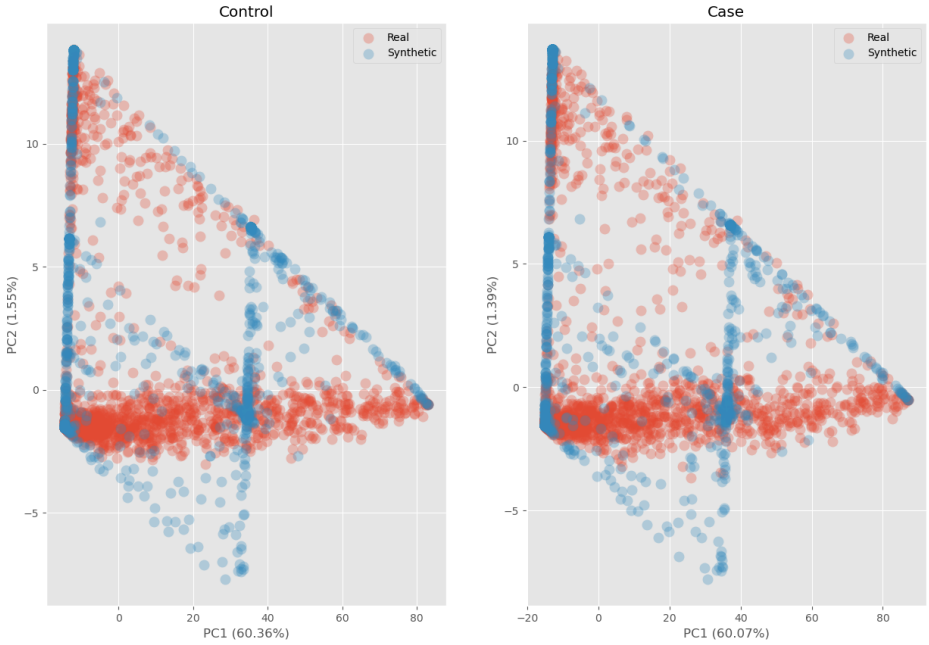

Figure 5.4: The first two principal components resulted from Principal component analysis of the real and synthetic data, separated by the diagnosis of the sample

5.3 Alternative Allele Counts

Showcased in the figures 5.5 & 5.6 is the distribution of the alternative allele counts present in the samples. The goal of calculating this value was to ascertain if the model was able to deviate from its initial predictions of total homozygosity to the reference allele i.e. outputting all `0`'s. These results show that the distribution of the synthetic data exhibits some deviation from the distribution for the original data. However, these discrepancies are not substantial, given that the Wassertein distance calculated between the two distributions was relatively low ($W_1 = 0.0229$). These results suggest that the model is successful in approximating the underlying distribution of the real data to a certain extent.

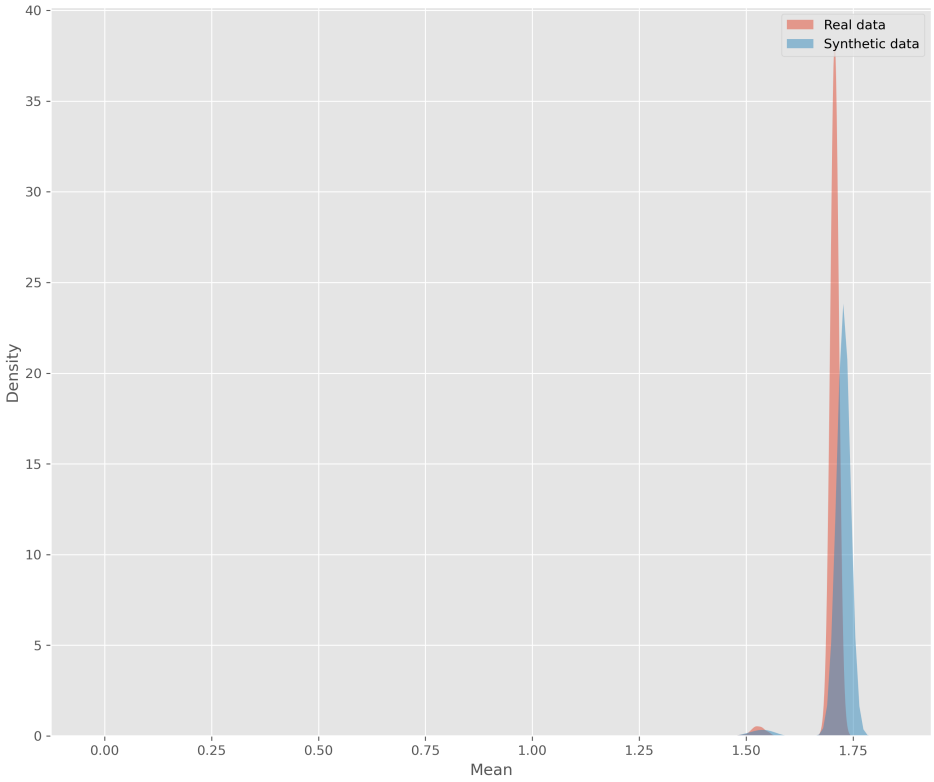

Figure 5.5: Alternative allele counts of the 5000 samples for the real and synthetic data

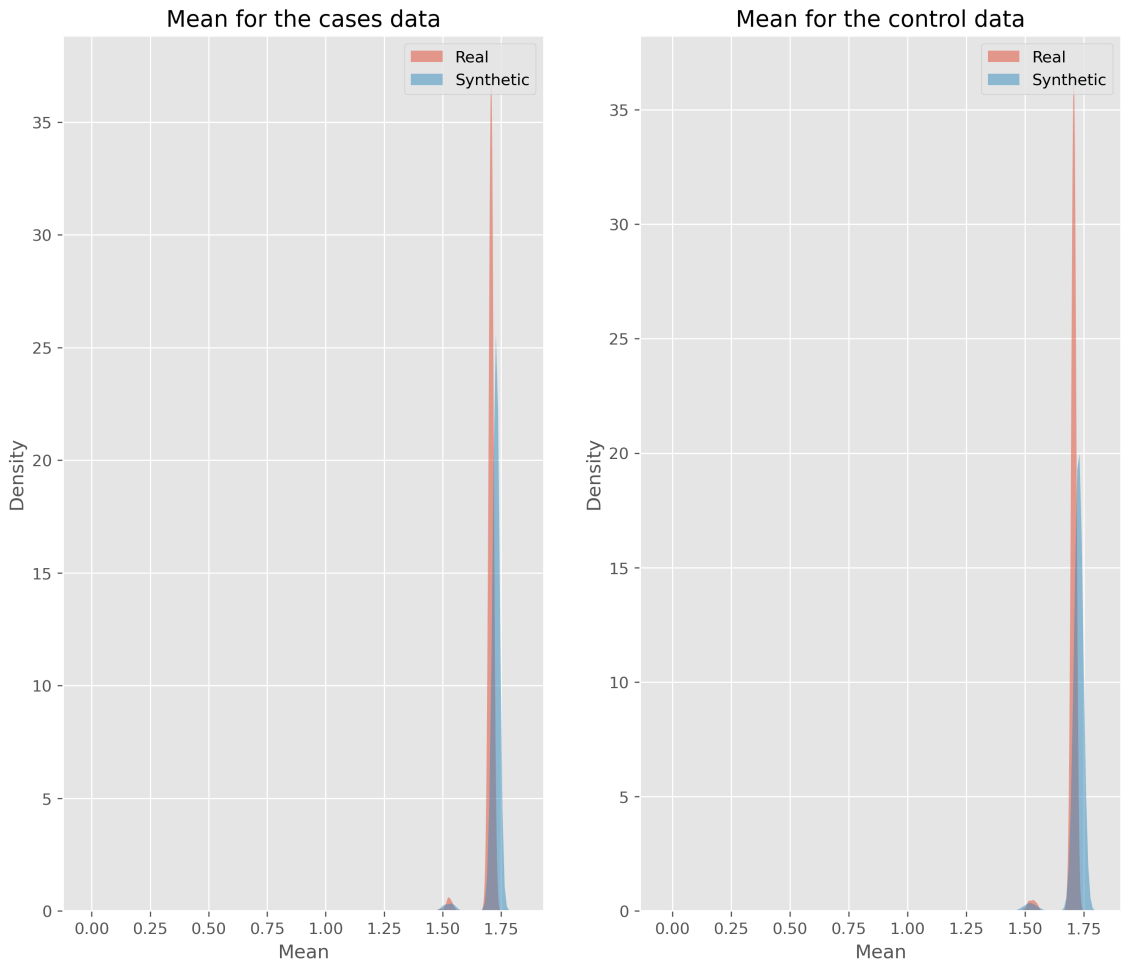

Figure 5.6: Alternative allele counts of the 5000 samples for the real and synthetic data, separated by the diagnosis of the sample

5.4 Allelic Frequencies

The allelic frequencies can provide insight into the ability of our model to capture the genetic characteristics of the original dataset. The distributions of the calculated allelic frequencies are displayed in the figures 5.7 for the whole dataset, 5.8 for the control samples, and 5.9 for the SCZ samples.

Upon analysing the allelic frequency distribution in the figures, it is evident that there are deviations between the real and the synthetic data. These discrepancies might suggest that the generative model might not be able to fully replicate the complex genetic patterns present in the original data. However, it is relevant to note that while there are deviations present, the Wasserstein distances calculated for both the p (alternative allele) and q (reference allele) between the real and the

synthetic data are relatively low, with $W_1 = 0.05613$ and $W_1 = 0.05612$, respectively. The low Wasserstein distances suggest that the allelic frequency distributions of the synthetic data are close to their real counterpart.

Overall, these results hint that our model was able to capture a considerable portion of the genetic information present in the original data, by approximating the allelic frequency of the real data to a considerable degree.

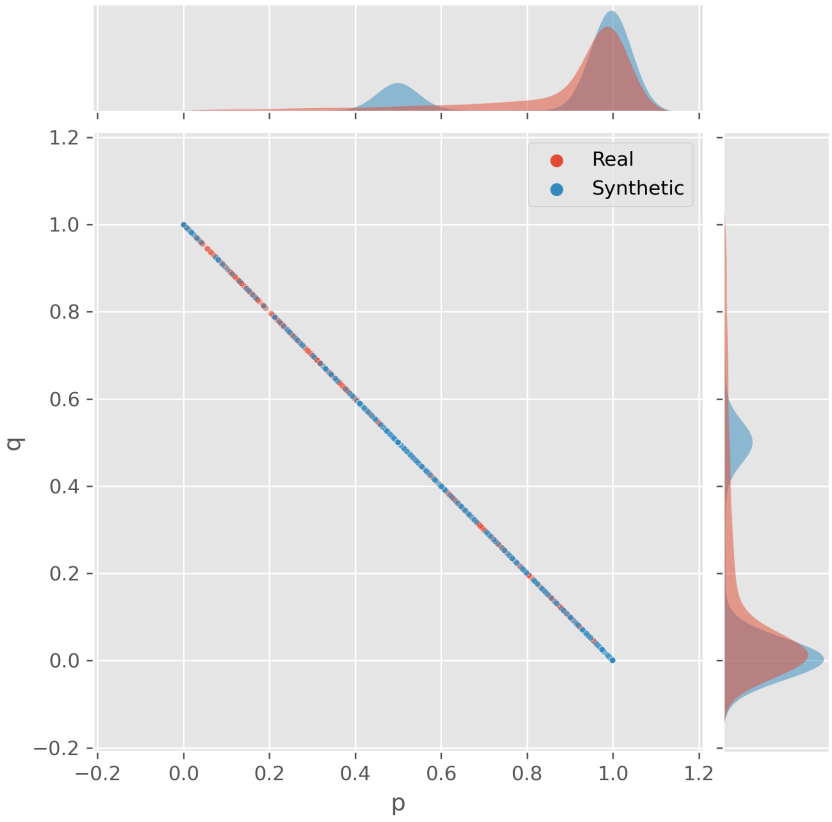

Figure 5.7: Allelic frequency distributions of the real and the synthetic data

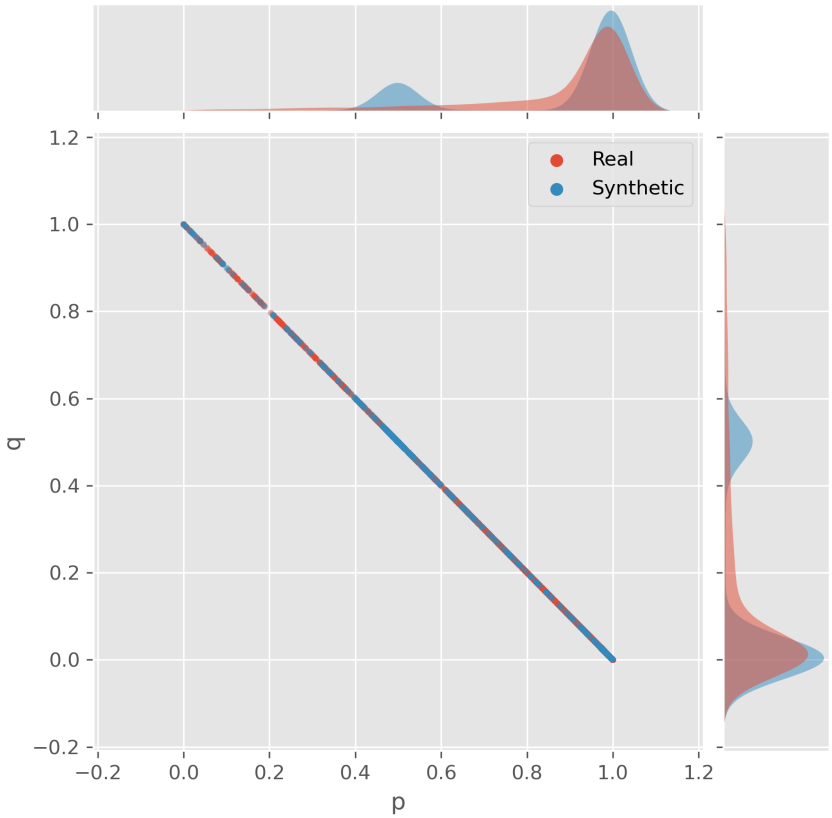

Figure 5.8: Allelic frequency distributions of the real and synthetic data from the control samples

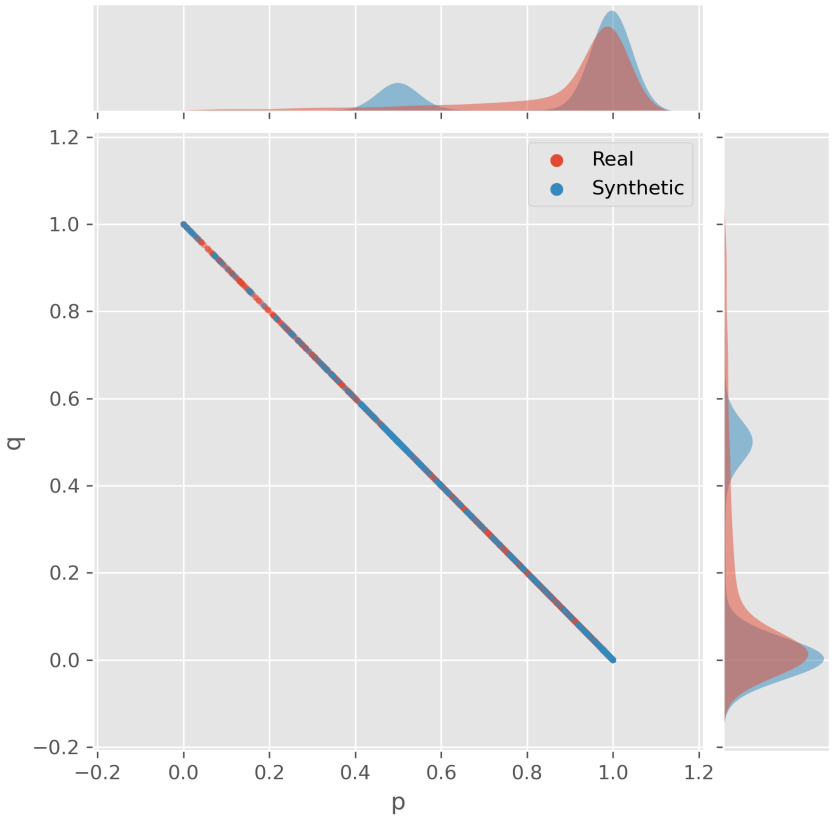

Figure 5.9: Allelic frequency distribution of the real and synthetic data from the samples with a schizophrenia diagnosis

5.5 Minor Allele Frequency

The minor allele frequency (MAF) is another genetic feature that can provide useful insight about the performance of our model capability of capturing genetic information from the real data. MAF is defined as the frequency of the less common allele at a specific locus. This feature is crucial to understand genetic diversity and population genetics. After calculating the allelic frequencies, we proceeded to calculate the MAFs, with the results displayed in the figure 5.10.

Again, upon analysing the resulting figures, there is a certain amount of deviation present between the real and the synthetic data. This further suggests the challenges of the model to fully replicate the intricate genetic patterns and patterns encoded in the original data. Nevertheless, this discrepancy is not substantial with the Wasserstein distance between the real and the fake MAF values being relatively low, at ($W_1 = 0.0528$).

These results of the MAFs comparison, coupled with the low Wasserstein distance between the real and synthetic data, may be signs that the generative model was capable of capturing the genetic diversity of the original dataset.

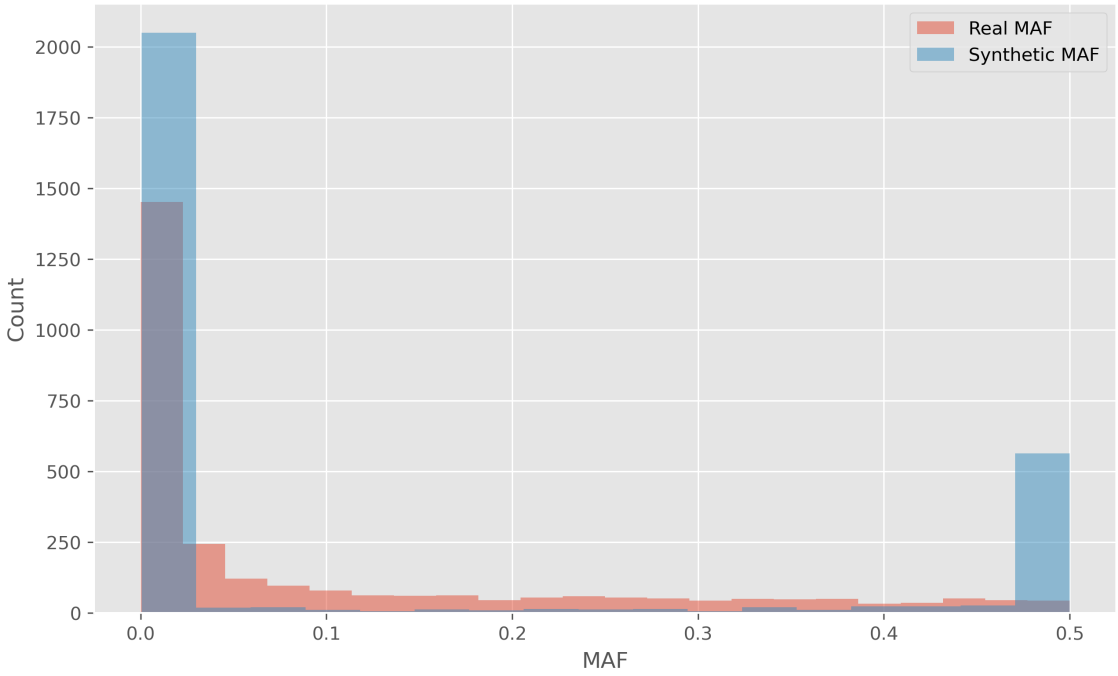

Figure 5.10: Minor allele frequency distribution of the real and synthetic data

5.6 Fixation Index

Using the Fixation Index (F_{ST}) we can obtain even more insight into our model's ability to capture genetic information. F_{ST} is a measure used in population genetics to quantify the degree of genetic differentiation between populations, meaning that values closer to zero show a strong indication that the populations of interest belong to the same population. Here with the F_{ST}, we measure if our model is capable of generating synthetic data close enough to the original data, to the point that the F_{ST} hints that these two belong to the same population.

It is apparent in the figure **??**, that the difference between the FST values of the real and the synthetic data is not very evident. These small difference supports the notion that the model was able to replicate the genetic variation and population structure encoded in the original data. A difference of this magnitude indicates that our model is able to capture a considerable amount of the genetic differentiation and population structure encoded in the original data.

Overall, the results of the FST comparison between the real and synthetic data highlight the model's success in approximating the genetic differentiation and population structure present in the original dataset.

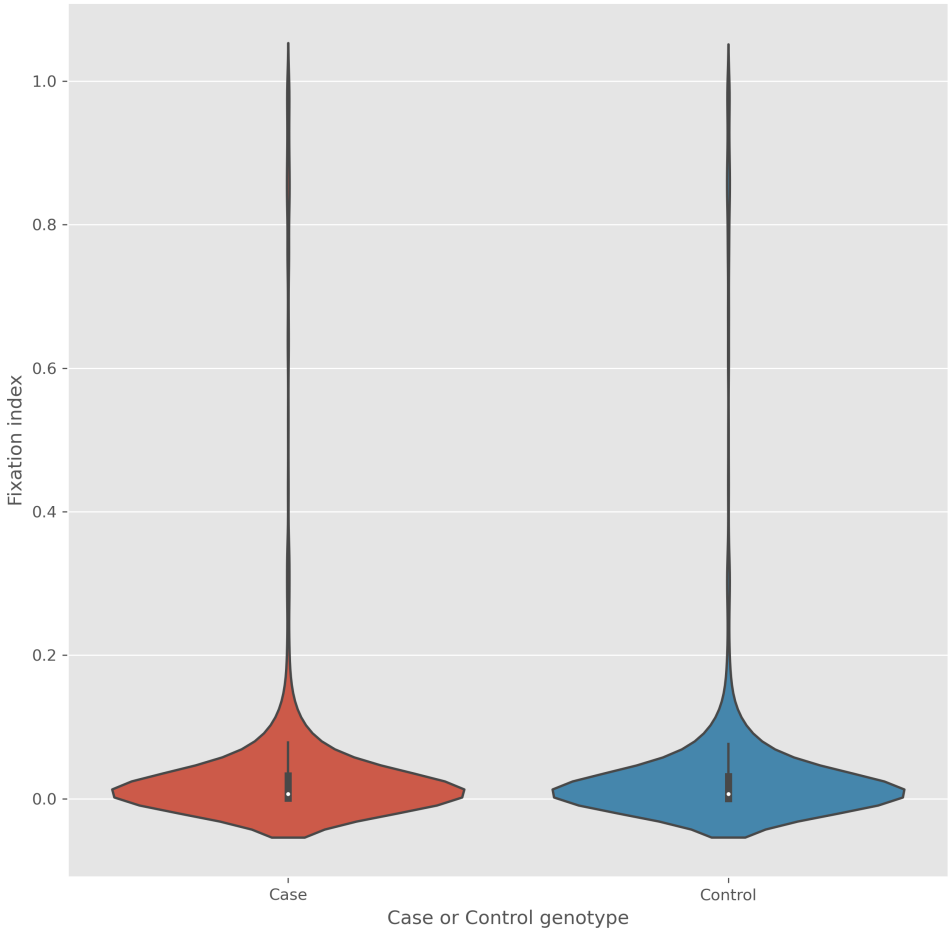

Figure 5.11: Fixation index values for the real and synthetic data

5.7 Data Utility

In the previous sections of this work, the results pertaining to the genetic characteristics of the real and synthetic data were showcased, now we'll compare the utility of the real data and the generated data. For that purpose, we're going to try to predict if a specific sample has the SCZ diagnosis or not. Here, the goal was to

assess our model's ability to produce synthetic data that could mimic the real data while also keeping its predictive characteristics. The results of this experiment and the models used are in the tables 5.1

The models used, and their performance, are showcased in the tables 5.1 and 5.2. We could observe that the results are not the best. However, the fact that they are comparable suggests that the synthetic data has in it encoded some predictive characteristics of the original data.

Table 5.1: Models performance with the real data

	Random Forest	kNN	SVM	Logit
Accuracy	0.582000	0.500000	0.624000	0.605000
Precision	0.569536	0.524731	0.624788	0.613074
Recall	0.822180	0.466539	0.703633	0.663480
F1	0.672926	0.493927	0.661871	0.637282

Table 5.2: Models performance with the synthetic data

	Random Forest	kNN	SVM	Logit
Accuracy	0.553000	0.532000	0.543000	0.502000
Precision	0.553672	0.549020	0.543536	0.523277
Recall	0.749522	0.588910	0.787763	0.537285
F1	0.636881	0.568266	0.643247	0.530189

6

Discussion

In this work, the WGAN generative model was implemented for the generation of artificial genomes based on their common summary statistics.

The training process was divided in two portions. In the first, the generator was pretrained in a self-supervised manner with the aid of summary statistic and in the second the weights of the pretrained generator were transferred to the wgan generator and then the conventual WGAN training process took place. During the self-supervised portion, the generator exhibited a noteworthy ability to converge. This decoder-like model, was able to successfully map the lower-dimensional summary statistics data to the original data and demonstrated consistent progress while doing so. Such convergence was vital for providing the subsequent WGAN-GP training phase a strong starting point, which enables the generator to build upon the learned genetic knowledge encoded in the summary statistics and further refine its performance.

During the WGAN-GP training process, the initial convergence, made evident by the loss curves, showed that the generator was able to generate high quality synthetic data. However, as the training process continued the validation values began to fluctuate and ultimately showing signs of divergence at the later epochs. This observed divergence suggests a certain inability of the model in capturing the full complexity of the data. Even though, our models showed promising results, the divergence at the end of the training process showed that further fine-tuning might be beneficial to fully harness its capabilities

With the PCA, we were able to see that the generated synthetic data when compared with the original data, had a tendency to cluster in the certain regions. This could mean that these clusters present in the synthetic data are the evidence of important regions that are needed to be able to capture the overall structure of the real data.

Another possibility is that these regions have a high density of data points and that made the model start to prioritize the generation of data with those characteristics resulting in the clusters we observed. The marginal plots, show that this cluster of data mainly occurs along the first principal component (PC1) in contrast to what occurs along the second principal component that even though there is some deviation between the real and the fake distribution it is not to the degree observed in PC1. The concentration of data on PC1 and the calculation of the Wasserstein distance between the real and the synthetic data distribution that resulted in a value of $W_1 = 7.98$ make it evident that the divergence between these distributions are considerably high. This hints that the model was not able to capture the information regarding variation encoded in the original data. However, it is important to note that PCA is a linear dimensionality reduction method. There's the possibility that the model might be prioritizing the non-linear characteristics of the data, then these observed divergences would be expected. However, this does not mean that the model is not able to capture important information during the training process. In the future, to see if the model is able to capture linear and non-linear characteristics of the original data, the usage of PCA coupled with a non-linear dimensionality reduction method, for example, the T-distributed stochastic neighbour embedding (t-SNE).

Regarding the statistics pertaining to genetics, like the alternative allele counts, allelic frequency, minor allele frequency and fixation index, we could see that visually there is always deviation between the distributions of these statistics for the original and the synthetic data. This deviation show that the model is facing challenges in perfectly replicating the information pertaining to these statistics encoded in the original data. Nevertheless, the Wasserstein distance shows that these differences evident in the plots are relatively low across the board, with the values of $W_1 = 0.0229$ for the alternative allele counts, $W_1 = 0.05613$ and $W_1 = 0.05612$ for the allelic frequencies of the reference allele q and the alternative allele p between the real and the synthetic data, and $W_1 = 0.0528$ for the minor allele frequencies. Even though these differences are visually apparent, genetic data is complex and influenced by factors such as population demographic, genetic drift, and selective pressure. Our model was able to generate, with some success, these statistics without explicitly supplying this information, made evident by the low Wasserstein distance across the board. This shows that in spite of showing some differences between their original counterpart, there's a potential of using synthetic data from generative

models like ours for genetics studies simulations and downstream analysis.

This contrast between the results of the PCA and the results of the genetic statistic raise intriguing questions about our models' ability to generate high quality synthetic data. While the model showed that it was as able to replicate the genetic characteristics by encoded in the original data, but at the same time encountered challenge in accurately reproducing the more complex and high-dimensional structure of the data.

Lastly, the predictive characteristics of the data were compared, by simulating a scenario where the goal was to be able to predict if the patient was a control or diagnosed with SCZ. For this portion of the work the models used showed for the original data and the generated synthetic data similar performance for all the metrics considered. This suggests that the predictive characteristics of the original data were retained by the model. However, to verify if the generated data could be used for similar scenarios, a more thorough analysis and more complex predictive models would be needed. That way would be able to have a deeper understanding of how these models understand the patterns hidden in the data.

7

Conclusion

Our proposed framework for genotype generation with the aid of their associated summary statistics consisted of two steps: self-supervised pretraining and WGAN-GP training. The results obtained from this proposed approach have been promising, with the achievements observed through the comparison of the genetic statistics and the utility of the generated synthetic data with its real data counterpart.

The self-supervised pretraining upon the summary statistics effectively provided the WGAN-GP a strong starting point, aiding in the accurate decodification of this lower-dimensional data format to reconstruct the original data. This results in the generation of realistic synthetic genotypes.

Nevertheless, there are some faults. We observed certain areas that require further refinement. The discrepancy made evident by the principal component analysis (PCA) indicate that the model representation might not entirely align with the original data distribution. Also, during the training process of the WGAN-GP, we were able to detect hints of divergence in the validation loss at the later epochs, suggesting that our model faces challenges regarding generalization and stability.

7.1 Future Work

In future work, we plan to extend the application of this framework to additional complex diseases other than schizophrenia. By incorporating diverse datasets representing various diseases, our goal would be to assess our WGAN-GP based framework ability to generate synthetic genotypes with characteristics of different genetic conditions.

Additionally, we will focus on enhancing the fine-tuning process to achieve better results regarding the framework's ability to generate synthetic data that matches

real-world complex datasets. This requires the investigation of regularization techniques and optimization strategies with the goal of improving the overall quality and diversity of generated genotypes.

Addressing these future directions, our research aims to aid the field of generative modelling in the context of bioinformatics, specifically synthetic genotype generation.

Bibliography

[1] C. St. Mary, T. H. Q. Powell, J. S. Kominoski, and E. Weinert, "Rescaling biology: Increasing integration across biological scales and subdisciplines to enhance understanding and prediction," vol. 61, no. 6, pp. 2031–2037. [Online]. Available: https://academic.oup.com/icb/article/61/6/2031/6362607

[2] A. M. Giani, G. R. Gallo, L. Gianfranceschi, and G. Formenti, "Long walk to genomics: History and current approaches to genome sequencing and assembly," vol. 18, pp. 9–19. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2001037019303277

[3] J. D. Watson and F. H. C. Crick, "Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid," vol. 171, no. 4356, pp. 737–738. [Online]. Available: https://www.nature.com/articles/171737a0

[4] ——, "Genetical implications of the structure of deoxyribonucleic acid," vol. 171, no. 4361, pp. 964–967. [Online]. Available: https://www.nature.com/articles/171964b0

[5] P. Larrañaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armañanzas, G. Santafé, A. Pérez, and V. Robles, "Machine learning in bioinformatics," vol. 7, no. 1, pp. 86–112. [Online]. Available: https://academic.oup.com/bib/article/7/1/86/264025

[6] R. Wang, Y. Li, X. Wang, H. Tang, and X. Zhou, "Learning your identity and disease from research papers: Information leaks in genome wide association study."

[7] B. Malin, "Re-identification of familial database records," vol. 2006, pp. 524–8.

[8] E. Uffelmann, Q. Q. Huang, N. S. Munung, J. de Vries, Y. Okada, A. R. Martin, H. C. Martin, T. Lappalainen, and D. Posthuma,

"Genome-wide association studies," vol. 1, no. 1, p. 59. [Online]. Available: https://www.nature.com/articles/s43586-021-00056-9

[9] T. Cui, K. E. Mekkaoui, A. Havulinna, P. Marttinen, and S. Kaski, "Improving neural networks for genotype-phenotype prediction using published summary statistics." [Online]. Available: http://biorxiv.org/lookup/doi/10.1101/2021.11.09.467937

[10] W. Chen, Y. Wu, Z. Zheng, T. Qi, P. M. Visscher, Z. Zhu, and J. Yang, "Improved analyses of GWAS summary statistics by reducing data heterogeneity and errors," vol. 12, no. 1, p. 7117. [Online]. Available: https://www.nature.com/articles/s41467-021-27438-7

[11] T. W. Willis and C. Wallace, "Accurate detection of shared genetic architecture from GWAS summary statistics in the small-sample context." [Online]. Available: http://biorxiv.org/lookup/doi/10.1101/2022.10.13.512103

[12] J. Hayhurst, A. Buniello, L. Harris, A. Mosaku, C. Chang, C. R. Gignoux, K. Hatzikotoulas, M. A. Karim, S. A. Lambert, M. Lyon, A. McMahon, Y. Okada, N. Pirastu, N. W. Rayner, J. Schwartzentruber, R. Vaughan, S. Verma, S. P. Wilder, F. Cunningham, L. Hindorff, K. Wiley, H. Parkinson, and I. Barroso, "A community driven GWAS summary statistics standard." [Online]. Available: http://biorxiv.org/lookup/doi/10.1101/2022.07.15.500230

[13] D. Vanhonacker, M. Verdonck, and H. Nogueira Carvalho, "Impact of closed-loop technology, machine learning, and artificial intelligence on patient safety and the future of anesthesia," vol. 12, no. 4, pp. 451–460. [Online]. Available: https://link.springer.com/10.1007/s40140-022-00539-9

[14] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," vol. 2, no. 3, p. 160. [Online]. Available: https://link.springer.com/10.1007/s42979-021-00592-x

[15] S. Nosratabadi, A. Mosavi, P. Duan, P. Ghamisi, F. Filip, S. Band, U. Reuter, J. Gama, and A. Gandomi, "Data science in economics: Comprehensive review of advanced machine learning and deep learning methods," vol. 8, no. 10, p. 1799. [Online]. Available: https://www.mdpi.com/2227-7390/8/10/1799

[16] K. Choudhary, B. DeCost, C. Chen, A. Jain, F. Tavazza, R. Cohn, C. W. Park, A. Choudhary, A. Agrawal, S. J. L. Billinge, E. Holm, S. P. Ong, and C. Wolverton, "Recent advances and applications of deep learning

methods in materials science," vol. 8, no. 1, p. 59. [Online]. Available: https://www.nature.com/articles/s41524-022-00734-6

[17] G. B. Goh, N. O. Hodas, and A. Vishnu, "Deep learning for computational chemistry," vol. 38, no. 16, pp. 1291–1307. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/jcc.24764

[18] C. Ma, H. H. Zhang, and X. Wang, "Machine learning for big data analytics in plants," vol. 19, no. 12, pp. 798–808. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1360138514002192

[19] N. Sapoval, A. Aghazadeh, M. G. Nute, D. A. Antunes, A. Balaji, R. Baraniuk, C. J. Barberan, R. Dannenfelser, C. Dun, M. Edrisi, R. A. L. Elworth, B. Kille, A. Kyrillidis, L. Nakhleh, C. R. Wolfe, Z. Yan, V. Yao, and T. J. Treangen, "Current progress and open challenges for applying deep learning across the biosciences," vol. 13, no. 1, p. 1728. [Online]. Available: https://www.nature.com/articles/s41467-022-29268-7

[20] C. S. Von Bartheld, J. Bahney, and S. Herculano-Houzel, "The search for true numbers of neurons and glial cells in the human brain: A review of 150 years of cell counting: Quantifying neurons and glia in human brain," vol. 524, no. 18, pp. 3865–3895. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/cne.24040

[21] F. Chollet, *Deep learning with Python*, second edition ed. Manning Publications, OCLC: on1289290141.

[22] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A comprehensive survey of loss functions in machine learning," vol. 9, no. 2, pp. 187–212. [Online]. Available: https://link.springer.com/10.1007/s40745-020-00253-5

[23] M. Raghu and E. Schmidt, "A survey of deep learning for scientific discovery." [Online]. Available: http://arxiv.org/abs/2003.11755

[24] R. Johnston, "Linear regression model," in *Encyclopedia of Quality of Life and Well-Being Research*, A. C. Michalos, Ed. Springer Netherlands, pp. 3621–3627. [Online]. Available: http://link.springer.com/10.1007/978-94-007-0753-5_1659

[25] K. Kasmaoui, "Linear regression," in *Global Encyclopedia of Public Administration, Public Policy, and Governance*, A. Farazmand, Ed.

Springer International Publishing, pp. 1–11. [Online]. Available: http://link.springer.com/10.1007/978-3-319-31816-5_478-1

[26] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*, third edition ed., ser. Wiley series in probability and statistics. Wiley, no. 398.

[27] L. Breiman, "Random forests," vol. 45, no. 1, pp. 5–32. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[28] M. M. Adankon and M. Cheriet, "Support vector machine," in *Encyclopedia of Biometrics*, S. Z. Li and A. Jain, Eds. Springer US, pp. 1303–1308. [Online]. Available: http://link.springer.com/10.1007/978-0-387-73003-5_299

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press.

[30] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," vol. 1, no. 4, pp. 541–551.

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," vol. 323, no. 6088, pp. 533–536. [Online]. Available: https://www.nature.com/articles/323533a0

[32] D. P. Kingma and M. Welling, "Auto-encoding variational bayes." [Online]. Available: http://arxiv.org/abs/1312.6114

[33] F. Farnia and A. Ozdaglar, "GANs may have no nash equilibria." [Online]. Available: http://arxiv.org/abs/2002.09124

[34] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep generative modelling: A comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models," vol. 44, no. 11, pp. 7327–7347. [Online]. Available: http://arxiv.org/abs/2103.04922

[35] J. Vincent, "ThisPersonDoesNotExist.com uses AI to generate endless fake faces," publication Title: The Verge. [Online]. Available: https://www.theverge.com/tldr/2019/2/15/18226005/ai-generated-fake-people-portraits-thispersondoesnotexist-stylegan

[36] C. Bergstrom and J. West, "Which face is real?" publication Title: Which Face Is Real? [Online]. Available: https://www.whichfaceisreal.com/index.php

[37] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," vol. 37, no. 6, pp. 1554–1563, publisher: Institute of Mathematical Statistics. [Online]. Available: http://www.jstor.org/stable/2238772

[38] Y. Zhongliang, J. Shuyu, H. Yongfeng, Z. Yujin, and L. Hui, "Automatically generate steganographic text based on markov model and huffman coding."

[39] G. Szymanski and Z. Ciota, "Hidden markov models suitable for text generation."

[40] E. Eirola and A. Lendasse, "Gaussian mixture models for time series modelling, forecasting, and interpolation," in *Advances in Intelligent Data Analysis XII*, A. Tucker, F. Höppner, A. Siebes, and S. Swift, Eds. Springer Berlin Heidelberg, vol. 8207, pp. 162–173, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-642-41398-8_15

[41] A. Lamb, "A brief introduction to generative models." [Online]. Available: http://arxiv.org/abs/2103.00265

[42] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks." [Online]. Available: http://arxiv.org/abs/1406.2661

[43] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs." [Online]. Available: http://arxiv.org/abs/1606.03498

[44] J. Langr and V. Bok, *GANs in action: deep learning with generative adversarial networks*. Manning Publications, OCLC: on1050335878.

[45] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN." [Online]. Available: http://arxiv.org/abs/1701.07875

[46] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein GANs." [Online]. Available: http://arxiv.org/abs/1704.00028

[47] C. J. Battey, G. C. Coffing, and A. D. Kern, "Visualizing population structure with variational autoencoders," vol. 11, no. 1, p. jkaa036.

[Online]. Available: https://academic.oup.com/g3journal/article/doi/10.1093/g3journal/jkaa036/6105578

[48] K. T. Ahmed, J. Sun, S. Cheng, J. Yong, and W. Zhang, "Multi-omics data integration by generative adversarial network," vol. 38, no. 1, pp. 179–186. [Online]. Available: https://academic.oup.com/bioinformatics/article/38/1/179/6355579

[49] R. Luo, L. Sun, Y. Xia, T. Qin, S. Zhang, H. Poon, and T.-Y. Liu, "BioGPT: Generative pre-trained transformer for biomedical text generation and mining," vol. 23, no. 6, p. bbac409. [Online]. Available: http://arxiv.org/abs/2210.10341

[50] Q. Liu, H. Lv, and R. Jiang, "hicGAN infers super resolution hi-c data with generative adversarial networks," vol. 35, no. 14, pp. i99–i107. [Online]. Available: https://academic.oup.com/bioinformatics/article/35/14/i99/5529246

[51] K. Shimagaki and M. Weigt, "Selection of sequence motifs and generative hopfield-potts models for protein families," vol. 100, no. 3, p. 032128. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.100.032128

[52] N. Killoran, L. J. Lee, A. Delong, D. Duvenaud, and B. J. Frey, "Generating and designing DNA with deep generative models." [Online]. Available: http://arxiv.org/abs/1712.06148

[53] A. Madani, B. Krause, E. R. Greene, S. Subramanian, B. P. Mohr, J. M. Holton, J. L. Olmos, C. Xiong, Z. Z. Sun, R. Socher, J. S. Fraser, and N. Naik, "Large language models generate functional protein sequences across diverse families." [Online]. Available: https://doi.org/10.1038/s41587-022-01618-2

[54] B. Yelmen, A. Decelle, L. Ongaro, D. Marnetto, C. Tallec, F. Montinaro, C. Furtlehner, L. Pagani, and F. Jay, "Creating artificial human genomes using generative neural networks," vol. 17, no. 2, p. e1009303. [Online]. Available: https://dx.plos.org/10.1371/journal.pgen.1009303

[55] B. Yelmen, A. Decelle, L. L. Boulos, A. Szatkownik, C. Furtlehner, G. Charpiat, and F. Jay, "Deep convolutional and conditional neural networks for large-scale genomic data generation." [Online]. Available: http://biorxiv.org/lookup/doi/10.1101/2023.03.07.530442

[56] C. C. Chang, C. C. Chow, L. C. Tellier, S. Vattikuti, S. M. Purcell, and J. J. Lee, "Second-generation PLINK: rising to the challenge of larger and richer datasets," vol. 4, no. 1, p. 7. [Online]. Available: https://academic.oup.com/gigascience/article-lookup/doi/10.1186/s13742-015-0047-8

[57] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. CreateSpace.

[58] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf