TÉCNICO LISBOA

Deep Learning Model using trained SNP vector representations for High Order Epistasis Detection

Henrique Braz Vieira de Sousa

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisors: Prof. Aleksandar Ilic
Prof. Sergio Santander-Jiménez

Examination Committee

Chairperson: Prof. Paolo Romano
Supervisor: Prof. Aleksandar Ilic
Member of the Committee: Maurício Breternitz

November 2022

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

Acknowledgments

I would like to thank my supervisors Prof. Aleksandar Ilic and Prof. Sergio Santander-Jiménez for giving me the opportunity to explore such a complex and interesting topic during my masters dissertation and for the insightful advice and guidance for the development of the thesis. I would like to thank my parents for the support they have given me throughout my studying academic years without which none would have been possible.

Abstract

Genome-wide association studies (GWAS) investigate the relationship between genotype and phenotype where one of the most used biomarkers are Single Nucleotide Polymorphisms (SNP). When conducting this analysis, epistasis, the phenomenon where the effect of SNPs on the phenotype is conditioned on the presence of other SNPs, must be considered when dealing with complex diseases. Employing exhaustive statistical testing when considering SNP-SNP interaction effects regarding the phenotype results in a high computational burden because of the number of possible combinations, especially for higher order interactions. Therefore, researchers proposed a wide range of methods based on machine learning, heuristic or stochastic techniques as well as filtering approaches to tackle this problem. This thesis proposes a deep learning neural network using an embedding layer to represent the SNP information as vectors and an “attention layer” which is responsible for producing a single vector which acts as a SNP importance filter. The “attention layer” information is leveraged to use the proposed model as a filtering method with enough precision to reduce the initial SNPs number to a much smaller set on which exhaustive testing of combinations can be applied. The proposed model is compared with multiple state of the art methods using simulated datasets with 2-way, 3-way and 4-way epistasis patterns where it was able to outperform the majority of other methods for most of the simulated datasets and proven to be a promising non exhaustive method when considering the detection of higher order interactions with and without marginal effects.

Keywords

Single nucleotide polymorphisms; Epistasis Detection; Deep learning; Higher order interactions

Resumo

Em estudos de associação de genoma completo (GWAS), que investigam a relação entre genótipo e fenótipo, Polimorfismos de Nucleotídeo Único (SNP) são um dos biomarcadores mais utilizados. Neste tipo de análise, a epistasia, fenómeno em que o efeito dos SNPs no fenótipo é condicionado pela presença de outros SNPs, deve ser considerado em doenças complexas. Detetar efeitos da interação SNP-SNP em relação ao fenótipo através da aplicação de testes estatísticos de forma exaustiva tem alta carga computacional devido ao número de combinações possíveis, especialmente para interações de ordem superior. Portanto, os pesquisadores propuseram uma ampla variedade de métodos baseados em "machine learning", técnicas heurísticas ou estocásticas, bem como métodos de filtragem. Esta tese propõe uma rede neural de aprendizagem profunda que usa uma "embedding layer" para representar as informações dos SNPs como vetores e uma "layer de atenção" responsável por produzir um vetor que atua como um filtro da importância dos SNPs. A informação da "camada de atenção" é usada pelo modelo proposto para filtrar e reduzir com precisão suficiente o número inicial de SNPs para um conjunto muito menor no qual testes exaustivos de combinações podem ser aplicados. O modelo proposto é comparado com vários métodos de última geração usando dados simulados com padrões de epistasia formados por 2, 3 e 4 SNPs, onde superou os resultados da maioria dos outros métodos na maioria dos dados simulados e provou ser um método promissor não exaustivo na detecção de interações de ordem superior com e sem efeitos marginais.

Palavras Chave

Polimorfismos de Nucleotídeo Unico; Detecção de Epistasia; Aprendizagem Profunda; Interações de Ordem Superior

Contents

List of Figures

List of Tables

Acronyms

SNP Single Nucleotide Polymorphism

MAF Minor Allele Frequency

MLP Multi-Layer Perceptron

CNN Convolutional Neural Network

DNN Deep Learning Neural Network

RNN Recurrent Neural Network

ACO Ant Colony Optimization

GA Genetic Algorithm

RF Random Forest

1

Introduction

Contents

1.1 Motivation

The time and cost of sequencing genomes (the process of determining the DNA sequence of an organism) since 2006 was reduced by a factor of 1 million in less than 10 years [1]. This led to an increase in the availability and quantity of genome data and alongside it came the interest in genome-wide association studies, with a focus on understanding the relationship between genome and certain phenotypic traits (a trait that is manifested, e.g., color of the eyes, a disease). When conducting research to determine which part of the genome data is responsible for such traits, single nucleotide polymorphisms (SNPs) are commonly processed in this kind of biological analyses. SNPs represent a mutation of a gene in a single nucleotide that is present in at least 1% of the population. The standard research to reveal associations between SNPs and diseases consisted in assessing the effect of each individual biomarker (SNP) on the disease through statistical tests [2]. This standard approach suffers from a problem known as "missing heritability" [3] [4], where complex diseases cannot be explained solely by the single SNPs individual effects.

Researchers [5] have pointed out that the "missing heritability" comes from the fact that the standard approach is ignoring SNPs interactive effects, where the effect of a SNP on a disease is conditioned on the presence of another SNP. This process is known as epistasis. The presence of epistasis has been observed on complex diseases such as asthma [6] or diabetes [7]. It is therefore important to develop methods able to detect these interactive effects when trying to understand the impact SNPs might have on a disease.

The most accurate approaches consist of using exhaustive statistical testing of all possible SNP-SNP interaction effects, such as MDR [8] and BOOST [9]. Exhaustive statistical testing does come with the drawback of being too computationally heavy because of the number of possible SNPs combinations, and either can only test for lower interaction order of SNPs or can test higher order interactions (more than 3) but with a limited number of SNPs considered.

Machine learning based methods on the other hand are quite promising since they can use heuristics and stochastic searches thus reducing the computational load of the testing. Some of these methods involve using the Random Forest algorithm (such as EpiForest [10] , Random Jungle [11] and SNPInterforest [12]) or Bayesian networks (e.g., BEAM [13] and SMMB [14]). Ant colony optimization has also been applied to epistasis detection on AntEpiSeeker [15] and MACOED [16], as well as evolutionary algorithms CES [17].

A more recent type of machine learning algorithms, Deep Learning Neural Networks (DNNs), have shown great performance on genotype-phenotype prediction problems. Research works proposed in [18], [19], [20], [21], [22], [23] have used DNNs to model the genotype-phenotype relationships, although the work specific to SNP data still lacks the development of the identification of epistatic interactions and were mainly focused on the performance of classification tasks. When focusing on epistasis detection,

in [21] the deep learning model was still used in an exhaustive way which is not efficient for higher order interactions or a big number of SNPs. Beyond the high prediction accuracy, DNNs are able to learn the important features and model non-linear feature interactions without any prior assumptions making them promising on detecting epistatic interactions.

DNN models are complex and as such is hard to understand what they have learned during the training for a classification task. After the DNN models are fully trained, although the models might have great prediction, it is not obvious what SNP interactions the model has learned. To estimate the importance/effects of the SNPs regarding the prediction of the phenotype we must rely on deep learning explainability methods. One of the initial proposed methods in the field is the saliency map [24] which measures how much each input contributes to the prediction by computing the partial derivative of the output (e.g., phenotype) with respect to the inputs (e.g., SNPs). The information given by the saliency map can be used to filter the initial huge number of SNPs to a much smaller set which can then be exhaustively tested for interactive effects.

1.2 Objectives

The objective of the proposed solution is the implementation of a deep neural network capable of epistasis detection as an alternative to exhaustive statistical testing and other state of the art machine learning methods. The whole process should aim to:

- Propose a neural network model capable of modelling epistasis;
- Present how, through architectural adjustments, the importance of SNPs for a certain disease can be extracted from a neural network model including SNPs with only epistatic effects;
- Leverage the learned information from the model to identify epistasis;
- The proposed solution should be an efficient filtering method when compared to exhaustive statistical testing for epistasis detection, meaning the overall number of combinations tested is reduced;
- Compare the performance of our solution to other state of the art methods in a varied set of simulated datasets.

1.3 Main contributions

In this thesis we propose a deep learning model capable of learning epistasis patterns, while also allowing to interpret what the model has learned by creating an architecture that allows for the extraction of the SNPs correlation with the disease. The architecture of the model contains a vector which is responsible to act as a feature selector indicating the importance of each SNP, which values are learned during model training. The goal of interpreting our model is not to explain how exactly the classification

mechanisms are working but rather the implementation of a feature attribution method to measure the causal associations between a disease and the SNPs. The importance scores of the SNPs learned by the vector represent the correlation to the phenotype the model learned from all the samples making our solution to not require any perturbations, inferences or the use of partial derivative methods to calculate the average importance of the features over the samples.

The model is effectively used as a filtering method capable of reducing the initial huge number of SNPs to a much conciser set upon which an exhaustive statistical testing is applied to identify significative epistatic interactions. The architecture of the model is different from a standard neural network to maintain the number of parameters low when using an embedding layer which projects each SNP value into a vector. The idea was based in the type of input mixing present in Transformers although still different, to adapt to the type of data concerning the topic of epistasis detection. The current non exhaustive state of the art methods have some problems in their approach for detecting higher order interactions (3 or more) when the individual SNPs and lower order combinations of SNPs contained in the higher order interaction do not present a significative correlation with the phenotype. The proposed model has the capability of modelling the correlation of all the possible combinations of SNPs with the phenotype, of any order, at the same time which makes it a more promising approach.

The model was able to outperform multiple state of the art methods in epistasis detection in the majority of a varied set of simulated datasets with 2-way, 3-way and 4-way SNP interactions. We also concluded that the sample size of the datasets used has a big impact on the performance of the proposed model, meaning the increase in samples size increases the performance of the model which was expected based on the available literature on deep learning models. This conclusion and the promising results demonstrated in this thesis, suggest that deep learning models should become even more promising for epistasis detection over the next years since the availability and amount of data will continue to increase.

1.4 Outline

Chapter 2 - Background: will start by introducing the concept of epistasis. Then, the type of simulated datasets used to test epistasis detection and their parameters is described. An overview of the state of the art methods and their limitations for Epistasis Detection is provided. Finally, we explore different deep learning models characteristics and methods which allow an interpretation of what the models learn.

Chapter 3 - Proposed solution: starts by giving a description of the proposed solution model architecture and how such model makes phenotype predictions based on the genotype information. Then, the rationale behind the architecture of the model and how such architecture allows for the detection

of causal associations between the SNPs and a disease is explained. Finally we explore how such information is leveraged for epistasis detection.

Chapter 4 - Experimental Results: begins by introducing how the testing of the model is done including how the model is run and what metrics are used to measure and compare its performance to other methods. Then, a description of the parameters and how they were selected for optimization of the proposed model is given. The specific details of each dataset used for the experimental results are also provided. The experimental results of the proposed solution are analyzed and compared to other state of the art algorithms on simulated datasets. The model detection power of epistasis is both tested in the presence or absence of marginal effects on datasets with 2nd, 3rd and 4th order interactions and different levels of heritability and minor allele frequencies (datasets parameters).

Chapter 5 - Conclusion and Future Work: finishes the thesis with a summary of the major conclusions and suggestions for continuation and improvement of the work.

2

Background

Contents

This chapter provides the necessary background to understand what are single nucleotide polymorphisms (SNPs) and the concept of epistasis, which is essential to understand the main goals of the thesis. Then, a description of how the SNPs are represented in datasets is provided, as well as the parameters and details of simulated datasets for different epistasis models. Further, the exhaustive statistical methods for epistasis detection and their limitations are discussed as well as alternative methods which include multiple state of the art machine learning algorithms with stochastic and heuristic search for epistasis. We will also mention the main type of neural networks architectures and their applications for the problem of epistasis detection. Finally, we address approaches which allow to gain understanding of the inner workings of these models (neural networks) regarding feature importance in classification tasks. Explainable modelling being one such approach which concept was used to extract SNPs correlation with disease in this thesis proposed solution.

2.1 Single nucleotide polymorphisms and Epistasis

Genes are composed by sequences of nucleotides (DNA). A single nucleotide polymorphism (SNP) is the mutation of a nucleotide in a gene which is present in at least 1% of the population, represented in Figure 2.1. This SNP can affect the manifestation of a phenotype (i.e., a trait), leading for example to an increased risk of having a particular disease which is why they are a commonly used biomarker in Genome-wide association studies (GWAS).

These mutations produce variants called alleles. Since humans are diploid organisms, we have two sets of chromosomes which are paired and each chromosome in a pair contains the same genes with the same or different alleles. The pairing of the alleles of both chromosomes can result in a homozygous combination, if both alleles are the same, or a heterozygous combination if the alleles are different. These combinations of the alleles are called the genotype of an individual for that SNP and are sometimes represented as “AA”,”Aa” and “aa” where “A” is the dominant allele and “a” the recessive allele.

With this genotype information, statistical tests can be used to which measure how the change in the alleles correlates with the manifestation of a disease to assess the importance of each individual SNP. Which is important to acquire a better understanding of our biological systems and potentially improve medicine (e.g., patient diagnosis). Although this type of testing might work fine for Mendelian diseases (diseases primarily resulting due to alterations in a single gene) but for complex diseases such as type 2 diabetes [7] or asthma it provides a poor performance leading to the problem known as “missing heritability” [5]. One reason being that such tests assume the impact the SNPs have on the disease are independent and do not measure the interactive effects between the SNPs. This concept of interactive effects is called epistasis.

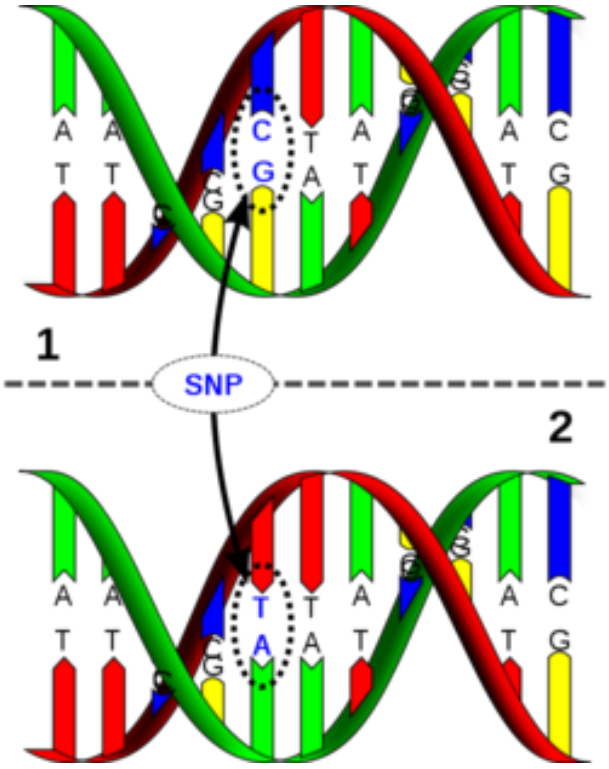


Figure 2.1: Single nucleotide polymorphism [25]

The original definition of biological epistasis was established in 1909 by Bateson [26], as the effect of an allele at a certain locus being masked by the effect of another allele at another locus. More recent definitions already include the fact that an allele effect can also be enhanced by the effect of another allele [27]. In more general terms, epistasis is the process through which the effect of an allele of a gene on the phenotype is regulated by the presence of other alleles.

Statistical epistasis can be defined as the deviation from the addition of the individual effects of alleles compared to the actual total contribution they have to the phenotype (i.e., when including interactive effects), this is the definition proposed by Fisher (1918) [28]. The deviation indicates the presence of non-additive interactive effects of different alleles.

This concept is present in the XOR function represented in Table 2.1. Considering X0 and X1 as two SNPs and the value of X2 as indicating the presence of a disease when it takes the value of 1 and the absence of disease when it takes the value of 0. Each SNP has two possible alleles represented by the values 0 or 1. By solely considering the X0 information represented in Table 2.2, the disease manifests itself with a 50% chance for both alleles 0 and 1 therefore it has zero correlation with disease. But if the combination of both SNPs alleles is taken into account, the presence or absence of disease can be perfectly predicted by using an XOR function to model the SNPs interactive effects.

As another example consider a pair of SNPs with epistatic/interaction effects and a dataset with 1800

Table 2.1: XOR function

X0	X1	X2
0	0	0
0	1	1
1	0	1
1	1	0

Table 2.2: X0 only function

X0	X2
0	0
0	1
1	1
1	0

samples, where 900 samples are cases (individuals with a disease) and 900 are controls (individuals without the disease). Each sample has the genotype of the individual regarding the two SNPs. The first SNP genotype is represented as AA, Aa or aa and the second SNP genotype as BB, Bb or bb. Figure 2.2 (A) shows the frequency of the genotypes of each individual SNP for cases and controls. Neither SNP shows any marginal effects (independent effects) since the distribution of the cases and controls is the same for each genotype. Figure 2.2 (B) is a contingency table showing the distribution of cases and controls where the combination of the two SNPs is taken into consideration and the epistatic effects between both SNPs observable since the distribution of the phenotype (disease) is not the same for every combination.

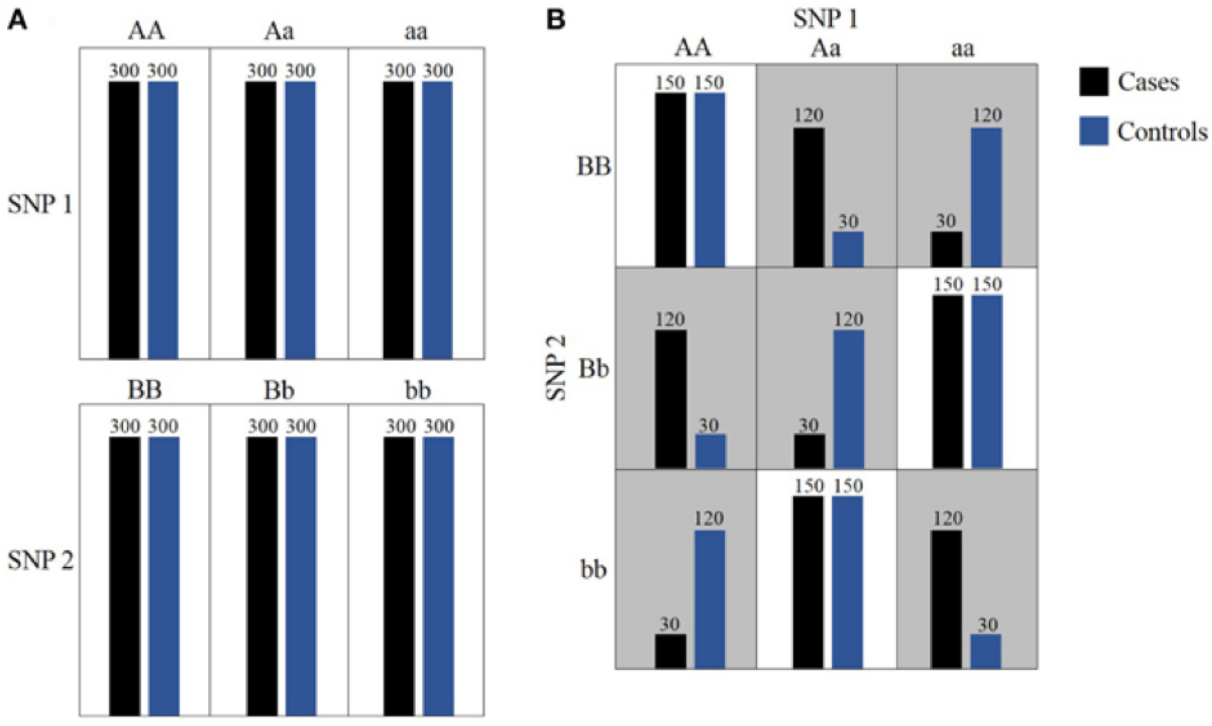


Figure 2.2: Statistical Epistasis illustrated through a contingency table [29]

Methods capable of detecting such interactive effects from the combinations of SNPs are required to model complex diseases.

2.2 Datasets structure

2.2.1 SNPs genotype as data

In a dataset each sample represents the SNPs genotype of an individual. Each SNP value is coded as 0,1 and 2 referring to the ”AA”,”Aa” and ”aa” possible allele combinations. The last value of each sample takes the value of 0 or 1 to indicate the absence (control) or manifestation (case) of the disease respectively. Table 2.3 is an example of a dataset with 3 samples and 5 SNPs.

Table 2.3: SNP dataset

	SNP 1	SNP 2	SNP 3	SNP 4	SNP 5	Class (control/case)
Sample 1	0	2	1	0	2	1
Sample 2	0	1	2	1	0	0
Sample 3	0	2	1	1	2	1

2.2.2 Penetrance function

To generate a simulated dataset a penetrance function is used which models the SNPs interaction and their effects. A penetrance function contains the probability of expressing the phenotype (disease) under study given each possible combination of the SNPs alleles. In Table 2.4 is an example of a penetrance function for a pair of SNPs where for example the combination of alleles ”AA” and ”BB” have a probability of disease of 9.8%. The same could be used to model the XOR function depicted in Table 2.5.

Table 2.4: Penetrance table

Penetrance function	AA	Aa	aa
BB	0.0980	0.0980	0.0980
Bb	0.0980	0.2989	0.5222
bb	0.0980	0.5222	0.9121

Table 2.5: Penetrance table for XOR function

Penetrance function	0(X0)	1(X0)
0(X1)	0	1
1(X1)	1	0

Some models for designing penetrance functions very commonly used to produce simulated datasets in the field of epistasis detection which were proposed by Marchini [30] are the multiplicative and threshold interaction models which representation is available in table 2.6. The α and β being the 2 values which control the value of the probability of disease, for example, Table 2.4 follows a multiplicative model were α takes the value of 0.0980 and β the value of approximately 1.432.

Table 2.6: Multiplicative and Threshold interaction models

Multiplicative Model	BB	Bb	bb
AA	α	α	α
Aa	α	$\alpha(1+\beta^2)$	$\alpha(1+\beta^3)$
aa	α	$\alpha(1+\beta^3)$	$\alpha(1+\beta^4)$
Threshold Model	BB	Bb	bb
AA	α	α	α
Aa	α	$\alpha(1+\beta)$	$\alpha(1+\beta)$
aa	α	$\alpha(1+\beta)$	$\alpha(1+\beta)$

2.2.3 Distinguishing marginal from epistatic effects

There are two types of datasets that need to be acknowledged, marginal and non-marginal. In a marginal dataset where two SNPs share the epistatic interaction, each one of them has a correlation with the disease independently of the epistatic effects. The independent probability of disease of each SNP genotype, represented on the "Marginal Penetrance" row and column of Table 2.7, are changing as the values of the genotype also change which means that each SNP has a correlation with the disease on its own. In a non marginal model all genotypes of the SNP have the same probability of disease, depicted in Table 2.8, and therefore a correlation of zero with the disease on its own. Both examples of epistasis represented in Figure 2.2 and Table 2.1 have no marginal effects.

Table 2.7: Marginal Effects

Marginal model		SNP 1			Marginal penetrance
		AA	Aa	aa	
SNP 2	BB	0.098	0.098	0.098	0.098
	Bb	0.098	0.299	0.522	0.118
	bb	0.098	0.522	0.912	0.14
Marginal penetrance		0.098	0.118	0.14	

Table 2.8: Non marginal effects

Non marginal model		SNP 1			Marginal penetrance
		AA	Aa	aa	
SNP 2	BB	0.486	0.96	0.538	0.64
	Bb	0.947	0.004	0.811	0.64
	bb	0.64	0.607	0.909	0.64
Marginal penetrance		0.64	0.64	0.64	

2.2.4 Dataset parameters

Minor allele frequency

The minor allele frequency (MAF) as the name indicates is the frequency of the minor allele of a SNP. Using the MAF we can calculate the theoretical percentage of each SNP allele combination in the dataset using the Hardy-Weinberg Equilibrium [31], or in other words, the intersection of the probabilities of the alleles (probabilities multiplication). The probability of the major allele is simply $1-p(a)$ where $p(a)$ is equal to the MAF. So for an MAF of 0.2 the probability of genotype "AA" is equal to $(1-0.2)*(1-0.2)=0.64$ and genotype "aa" is equal to $0.2*0.2=0.04$. Table 2.9 shows the distribution of a SNP genotype when the minor allele "a" has a MAF of 0.2 and Table 2.10 a MAF of 0.4.

Table 2.9: Percentage of each SNP genotype with an MAF of 0.2

MAF = 0.2	A (p=0.8)	a (p=0.2)
A (p=0.8)	AA (p=0.64)	Aa (p=0.16)
a (p=0.2)	aA (p=0.16)	aa (p=0.04)

In the datasets we will be working with on Section 4.4 the MAF parameter is the same for the SNPs present in the epistatic interaction which is simply a common practice. Having now the probability of each allele combination with an MAF of 0.2 represented in Table 2.9 for a SNP, we can calculate the probability of each genotype combination of a pair of SNPs by calculating the probabilities intersection again depicted in Table 2.11. For example, since AA and BB would both have a probability of 0.64, then $P(AA\&BB)=P(AA)*P(BB)=0.4096$.

Heritability and disease prevalence

There are two other important parameters for the dataset: the disease prevalence and heritability. The genetic heritability h^2 describes how much of the variation of an observable trait (phenotype) in a population can be attributed to genetic variation, in our case the SNPs genotype [32]. Heritability can be calculated using equation Equation (2.1) [33] where $P(D|G_i)$ is the probability of the phenotype being manifested when genotype G_i is present and $P(D)$ is the disease prevalence (the general probability of the phenotype being manifested in the population) Equation (2.2).

$$h^2=\frac{\sum_i(P(D|G_i)-P(D))^2P(G_i)}{P(D)(1-P(D))} \tag{2.1}$$

$$P(D)=\sum_i P(D|G_i)P(G_i) \tag{2.2}$$

Table 2.10: Percentage of each SNP genotype with an MAF of 0.4

MAF = 0.4	A (p=0.6)	a (p=0.4))
A (p=0.6)	AA (p=0.36)	Aa (p=0.24)
a (p=0.4)	aA (p=0.24)	aa (p=0.16)

Table 2.11: Probability of each genotype combination between a pair of SNPs with an MAF of 0.2

Probability of each SNPs combination	AA	Aa	aa
BB	0.4096	0.2048	0.0256
Bb	0.2048	0.1024	0.0128
bb	0.0256	0.0128	0.0016

The heritability parameter is directly correlated with how much of a significative effect the SNPs have on the disease probability.

2.3 State of the art methods for epistasis detection

The previous section described the concept of epistasis and how SNP interactive effects can be important to understand the causal relationship between genotype and phenotype. It also described how genotype information is represented into datasets. The next subsections will describe state of the art methods which are used to detect epistasis using those datasets.

2.3.1 Exhaustive statistical methods

The most accurate way to search for interactions between SNPs is by searching exhaustively using statistical tests, enumerating all possible combinations of SNPs and testing their interaction score. The problem with exhaustive methods is that testing for all combinations involving more than 2 or 3 SNPs at a time (i.e., higher-order interactions), has very poor scalability due to high computational burden since the number of tests increases exponentially with the order of the interactions (e.g., the number of all possible combinations of 500 SNPs in sets of 3 is equal to $2.07 * 10^7$). Adaption of these methods with certain techniques can improve their performance.

Traditional statistical methods

The most traditional statistical methods are parametric regression models such as logistic regression models which are a widely used method for exhaustive search of interactions. The PLINK software [34] is an example of a method that has implemented logistic regression models to detect epistasis. Logistic regression models can be represented as expressed in Equation (2.3), where x_1 and x_2 are the genotype values of two SNPs and β_1, β_2 are the coefficients which are adjusted iteratively by the model so that the whole equation predicts the likelihood of disease for each sample in the dataset as best possible. Equation (2.4) has an extra parameter $\beta_3 x_1 x_2$ to model the interactive effects of both SNPs while Equation (2.3) is only capable of adding the SNPs independent effects.

$$Log(\frac{p}{1-p}) = \alpha + \beta_1 x_1 + \beta_2 x_2, \tag{2.3}$$

$$Log(\frac{p}{1-p}) = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2, \tag{2.4}$$

These kind of methods struggle in high dimensional data and suffer as well from the curse of dimensionality (i.e., cases where the number of samples is small compared to the number of SNPs lead to poor performance). Therefore logistic regression is not suitable to manage genome-wide datasets.

Despite their shortcomings, traditional parametric regression methods serve as a foundation in the field.

BOOST

Boolean operation-based testing and screening (BOOST) [9] performs exhaustive testing of all possible pairwise SNP interactions. The program detects the presence of epistasis by measuring the deviation between the predicted likelihood of disease for models which only contain marginal effects and models which include interactive effects, for example the difference between Equation (2.3) and Equation (2.4) for a pair of SNPs. This deviation represents the definition of statistical epistasis.

After detecting promising SNP interactions based on the deviation between the models BOOST has a second stage where it measures the significance of such interactions using a chi-square test. An important shortcoming is that BOOST is restricted to analyzing pairwise interactions and no higher order interactions.

MDR

Multifactor Dimensionality Reduction (MDR) [8] is a nonparametric model (i.e, makes no assumptions about the genetic model) method that exhaustively searches for gene-gene or epistasis interactions. MDR uses the contingency tables from SNPs genotypes combinations and the ratio of cases to controls to either classify a combination as high risk coded as 1 or low risk coded as 0 according to a threshold. To understand this concept consider the XOR function and its contingency table in Table 2.12. After the MDR transformation, the information of SNP X0 and X1 would be reduced to variable Z depicted in Table 2.13 according to the high risk and low risk classification.

MDR tests the accuracy of a model using this new variable, in the case of Table 2.13 the Z variable achieves 100% accuracy perfectly predicting the samples. The accuracy level is used a proxy to indicate

Contingency table	0	1
0	$\frac{0}{1}$ =Low Risk	$\frac{1}{0}$ =High Risk
1	$\frac{1}{0}$ = High Risk	$\frac{0}{1}$ = Low Risk

Table 2.12: XOR function contingency table

Z	X2
0	0
1	1
1	1
0	0

Table 2.13: MDR X0 X1 transformed

the correlation of the SNPs with the disease. The main goal of MDR is to reduce the data dimensionality, and so it can be combined with different classification methods [17], as a result of these many different versions of MDR extensions have been proposed. Even though MDR is not restricted to pairwise interaction testing [35], for higher order interactions it struggles with more than several hundred SNPs.

2.3.2 Machine learning and artificial intelligence methods

Machine learning and combinatorial optimization offer different and promising approaches to epistasis detection, where non parametric models and heuristics avoid the high computational burden and the restriction to low order interactions. In machine learning the SNPs can be used as the feature inputs and the phenotype (e.g., disease) the class of a supervised classification problem. Machine learning methods are able to discover which SNPs are correlated with the phenotype and can be combined with a statistical testing of interactions. The flexibility in combining machine learning methods with other methods results in various algorithms with a multi-stage approach for example using Random Forests [10] and Bayesian networks [14]. Combinatorial optimization algorithms like Ant Colony Optimization [16] and evolutionary algorithms [17] will also be introduced.

Filtering methods

Filtering methods are used to reduce the initial number of SNPs to a smaller set on which the application of exhaustive testing is now feasible. Filtering methods which rely on the marginal effects of the SNPs to consider them important [30] defeat the purpose of the search for epistasis since SNPs without marginal effects can have strong epistatic effects. RelieF [36] for example is a filtering method which does not rely on marginal effects. There are also data integration filtering methods which rely on a priori knowledge. These methods rely on the use of databases like IntAct [37], BioGRID [38], STRING [39]

or ChEMBL [40], with information regarding protein-protein interaction or pathways, and although these methods avoid the lack of interpretation of the results given by a data mining procedure, they suffer from our lack of knowledge of epistatic effects in complex biological systems. These methods need to be combined with an additional stage responsible for finding significative epistatic interactions in the filtered subset of SNPs.

Decision Trees and Random Forests

A decision tree classifies the samples of a dataset by continuously splitting them into two subsets at each node. At each node the variable (e.g., a SNP) that best discriminates the samples according to the classification label (disease status) is selected. By constructing the whole tree, the algorithm finishes with a set of selected SNPs which were useful in the classification of the phenotype of the samples, either because of strong marginal or epistatic effects. The epistatic effect can be expressed through the tree model because the selected SNPs are conditioned on the SNPs already selected in the upper nodes of the decision tree. An important shortcoming of the tree construction is that the variable selected in the root node is not conditioned on anything, therefore it is selected taking only into account its marginal effect. Another problem is a single decision tree considering all possible variables when selecting a variable to split a node would not be feasible because of the number of SNPs. On other hand, considering only a subset of variables would result in a highly biased model, since it is working right from the start under the assumption that only a few SNPs contribute to the prediction of the phenotype.

Random Forest is an ensemble method which avoids bias by building multiple decision trees, the trees are trained on different data samples and only consider a subset of the variables at each node promoting diversity in the trees and reducing the computational load of each model. When the random forest is finished, the classification estimates of all trees are considered to classify each sample using majority voting. The construction of multiple models using subsets of the data and considering subsets of the variables at each node makes random forest suitable to handle genome-wide data. Epiforest [10] uses a random forest and the SNP markers as features to discriminate cases against controls. After the random forest is constructed, Epiforest tries to find a subset of SNPs that minimize the classification error. In the second stage this subset goes through an exhaustive statistical SNP-SNP interaction testing.

Bayesian networks

In a Bayesian network the variables are represented as nodes in a direct acyclic graph where the edges connecting those nodes map the correlation of the variables with the probability of disease and

the conditional effects between SNPs.

BEAM (Bayesian Epistasis Association Mapping) [13] is a program based on Bayesian networks used often as a reference for performance comparisons. BEAM uses a Markov Chain Monte Carlo algorithm (MCMC) where various combinations of SNPs taking specific values are explored and is able to test the association of each SNP to the phenotype iteratively.

An alternative to building a Bayesian network, since it is a NP hard problem, is to find a Markov blanket, this being the minimal set of SNPs that truly matter and make the prediction of the phenotype value conditionally independent of all other SNPs. Along the past decade various algorithms to build a Markov blanket were proposed and applied to epistasis detection, for example DASSO-MB [41]. One problem with the learning of the Markov blanket is that the first SNP added to the Markov blanket only takes into consideration SNPs with a strong marginal effect, and that first SNP selected conditions the rest of the SNPs added to the Markov blanket. Since most traditional Markov blanket methods are having trouble detecting SNPs with little or no marginal effect, Stochastic Multiple Markov Blanket (SMMB) [14] utilizes a stochastic exploration to learn multiple Markov blankets and just like the idea of a random forest it combines multiple models to form a consensus to address this problem.

Ant Colony Optimization

Ant colony optimization was inspired by how ants use pheromones to communicate with each other about the quality of a path to get food. For example, a shorter path to food will get an increased amount of pheromones, and so in this way, when one ant learns about a better path this knowledge is transmitted to all the other ants by the pheromone levels. Ant colony optimization uses artificial ants to explore, where each ant is responsible for evaluating the quality of a group of SNPs according to some metric. Depending on the result of that assessment, the SNPs with better quality will get a higher pheromone level, and through iterations all the ants will end up being guided by the pheromone levels to converge on a subset of SNPs with great quality. The selection of the subset of SNPs in each iteration is done based on the pheromone levels of the SNPs but can also be random, it is a balance between exploration and exploitation, so that the algorithm does not converge too quickly to a local optimum.

MACOED (Multi-objective ant colony optimization for epistasis detection) [16] uses ant colony optimization and it is a multi-objective heuristic search since it uses two different models to assess the quality of the SNPs. In MACOED, unlike the traditional ant colony optimization where the best solutions are lost between iterations, the best solutions are saved and compared with the solutions of the next iteration, this feature allows for a faster convergence and keeps the algorithm from losing good solutions. As most methods explained so far, MACOED uses a two stage approach, in the first stage a subset of SNPs is found by the ant colony optimization and in the second stage an exhaustive testing for epistatic

interactions is performed on that subset.

Genetic Algorithms

A Genetic Algorithm (GA) is a metaheuristic method that mimics evolution to find optimal solutions to a problem, for example a group of SNPs. In a GA applied to epistasis detection there is a population of solutions, also known as individuals, each individual of the population is represented as a group of SNPs and the fitness of each individual in the population is assessed according to some metric, for example testing the subset of SNPs for association with the phenotype using a statistical test. Then the population goes through selective pressure, a procedure in which some promising individuals (e.g., combination of SNPs) are selected based on the fitness scores, to generate an individual of the next generation offspring. When generating the offspring, two parents are normally selected through tournament selection and used to generate a new individual via crossover, which means the new combination of SNPs inherits combined parts from the group of SNPs of their parents. A mutation can also occur which results in an individual changing some part of its components.

An example of an application of GAs in epistasis detection is Epi-GTBN [42]. Epi-GTBN uses the GA as a heuristic search strategy for a Bayesian Network. The major advantages of a GA are the rapid global search and avoidance of falling into local optimum. It is scalable and easy to integrate with other algorithms.

2.4 Deep Learning Models

When developing AI, the early approaches attempted to explicitly program all the required knowledge to solve a task. Machine learning algorithms provided a good alternative to this otherwise very demanding job. Although they can learn from the data to extract patterns to solve for example a classification task, they are still relying on the quality and relevance of the preprocessed features contained in the raw data.

Deep Learning models can themselves construct relevant features of higher complexity by transforming the initial features through successive operations yielding a higher prediction power, especially in problems where the class of a sample cannot be well discriminated solely relying on the weighted sum of the input's features (additive effects only) since the solving of the problem requires complex non-linear combinations of the simpler features. The big difference in Deep Learning is the capacity for the discovery and construction of these more effective features directly from the data instead of having to rely on hand designed features. The models are able to create hierarchical representations of data with increasing levels of abstraction.

For example, when comparing to the machine learning approaches for epistasis detection described

so far, deep learning models are able to avoid at least three problems. The first being the reliance on a stochastic search for a group of SNPs with epistatic patterns, in deep learning (as explained in Section 2.4.1) the transformation of the initial features through non linear combinations (which are learned during the training of the model using gradients) allows the model to take in all the SNPs at once and discern exactly what are the SNPs which have a correlation with the disease, which avoids missing the SNPs which share an epistatic pattern by chance. Secondly, because of this feature, the problem of having to rely on a marginal effect SNP in order to start forming a correlated group of SNPs (as mentioned in decision trees and bayesian networks) is also completely avoided. The third problem is the apriori parameters that machine learning methods require in order to be efficient which includes defining the interaction order of the SNPs combination the algorithm is looking for, in deep learning models the interaction order is not defined therefore the model has the potential to learn an epistasis pattern including as many SNPs as necessary to predict a disease. These features make deep learning models a superior tool for epistasis detection which explains the choice for the approach used in our proposed solution.

Given their power of learning complex patterns from the data deep learning models have been successfully applied to a multitude of different fields including computer vision [43], speech recognition [44], natural language processing [45], machine translation [46] and surpassing the best human players performance on complex board games [47]. Many researchers believe domains such as bioinformatics can benefit tremendously from the application of deep learning models. The recent survey [48] provides a review of deep learning in bioinformatics and mentions applications in medical image analysis, omics and biomedical signal processing.

In this section we will provide an overview of the main architectures used for Deep Learning models starting with the standard multi-layer perceptron as well as mentioning how such models are trained. Then we will move into two other popular architectures these being Convolutional Neural Networks and Recurrent Neural Networks. We will also mention some of the existing applications to genotype-phenotype classifications tasks, epistasis detection and discovery of important features in DNA sequences for gene regulatory mechanisms.

2.4.1 Multi-layer perceptron

Neural networks architecture [49], is supposed to represent an abstraction of how the human brain works, more specifically how information is transmitted through the neurons. Neural networks are very expressive and flexible in the way they abstract the inputs into outputs. Figure 2.3 represents the architecture of a simple neural network formed by 3 layers. The input layer is responsible for introducing the values of the data the neural network is supposed to learn patterns from. Following the input layer, neural networks can have as many hidden layers as desired or required to model the data. Finally,

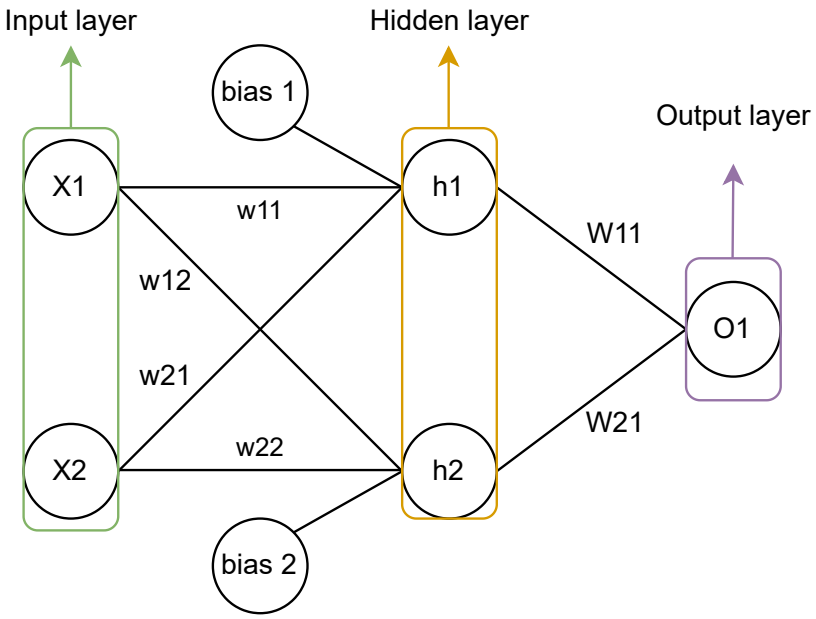


Figure 2.3: Neural network architecture

the neural network ends with the output layer which is responsible for providing a classification to the samples of the dataset.

Each layer is constituted by nodes and each node carries a value. The input layer has as many nodes as the number of features of the samples, in the hidden layers the number can vary and is chosen in the implementation of the model and for the output layer it depends on the number of possible categories a sample can be classified as. In the case of a binary classification (e.g., absence or presence of a disease), the output layer can have a single node whose value indicates the probability of disease. To compute the value of $Node_{l,j}$ we use Equation (2.5), where j is the number of the node in layer l and where i the number of the node in layer $l-1$ (i.e., the previous layer) while w are the weights associated with each node to node connection between the layers represented by the edges in Figure 2.3. For example, for node $h1$ it would be represented as $h1 = X1 * w11 + X2 * w21 + bias\ 1$. The input values are propagated forward through all the layers while being transformed by the weights and biases until they reach the output layer which will determine the neural network prediction of the sample class. One important detail is the combination of values that reaches each node can be further transformed using a non linear activation function which is required to model non linear interactions between inputs.

$$Node_{l,j} = \sum^{i} Node_{l-1,i} * w_{l-1,i,j} + bias_j \tag{2.5}$$

A neural network adjusts the values of the weights and biases iteratively while going through the samples of a dataset, this whole process being called Backpropagation. For each sample of the dataset

with inputs X_i and class of the sample Y_i a loss function is used, for example the mean squared error (MSE) represented in Equation (2.6) to obtain the error of the neural network. The larger the distance between the desired output Y_i and the predicted output Y_i' the larger the error. Then the gradient of the loss function with respect to each weight represented as $\frac{\partial E}{\partial w}$ is computed which indicates the direction in which the value of the weight should be adjusted in order to increase the loss function. Therefore, to minimize the error each weight is adjusted in the opposite direction using Equation (2.7), where w_{t+1} and w_t are the values of weight after and before the adjustment respectively.

$$MSE = \frac{1}{n}\sum (Y_i - Y_i')^2 \tag{2.6}$$

$$w_{t+1} = w_t - learning\_rate * \frac{\partial E}{\partial w} \tag{2.7}$$

2.4.2 Convolutional neural networks

There is a specific type of neural networks where the motivation is to enable machines to be able to extract concepts from images. The combination of Computer Vision with Deep Learning led to the development of the Convolutional Neural Network (CNN), which has been used in a multitude of tasks such as Image & Video recognition [50], Image Analysis Classification [51], Recommendation Systems [52], Natural Language Processing [53], etc.

A CNN can capture the spatial patterns in an image by using filters. We can think of the filter as the weights of a neural network layer by which the input values are multiplied to produce some output, but in this case, instead of multiplying all the inputs by the weights, the filter is made of a specific size and so will be applied at a local part of the image or data. The filters/kernels are eventually applied to the whole image as show on Figure 2.4 by shifting the position of the filter through the whole data (how much the filter shifts at each step is called stride), in this case we have 3 different filters one for each RGB channel of the image.

The intuition behind the filters is that they adjust their weights to detect important local patterns in the data and output a new transformed data. The values of the output from the filters are used to represent the presence of for example edges in an image and in the higher layers the combination of such edges reveal the presence of important concepts such as for example the ears of a cat. A CNN ends up in a fully connected network, a standard neural network, which takes as input the output of the convolutional layers (e.g., ears of a cat) and classifies the image (e.g., cat).

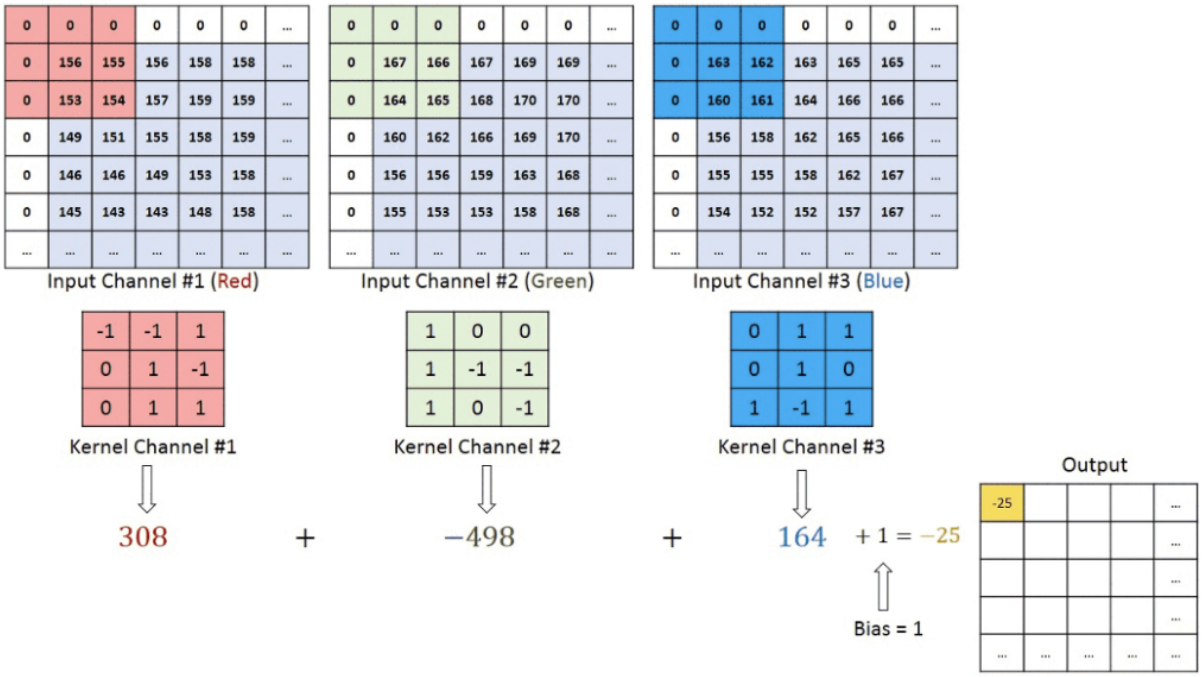


Figure 2.4: Convolutional layer filtering [54]

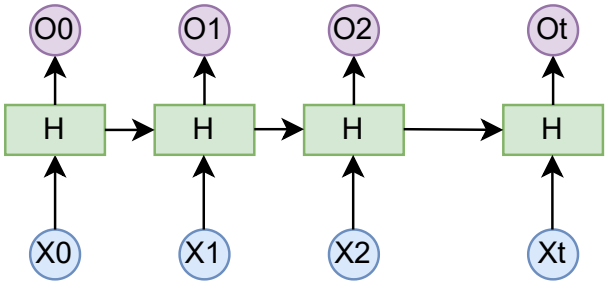


Figure 2.5: RNN structure

2.4.3 Recurrent Neural Networks

Recurrent Neural Networks [55] were built to deal with taking a sequence of inputs and also output a sequence of outputs where the relationships between inputs and previous outputs information must be taken into account for the rest of the outputs. RNNs sequentially "slide along" the input, doing some version of the same computation step at each position (weight sharing). They merge the information from the input at the current position plus the hidden state from the last position. The hidden state from the last input contains the relevant information of all past inputs. The way the RNN operates resembles reading. The RNN processes the first input, saves its relevant information, and passes on this information when reading the next inputs, which is extremely important when the meaning of the inputs is related to the context of the whole sequence.

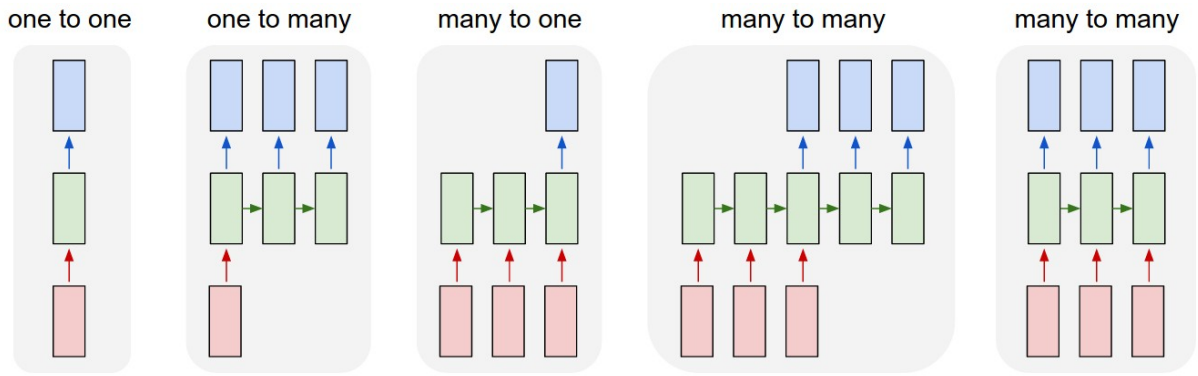


Figure 2.6: RNN architectures [56]

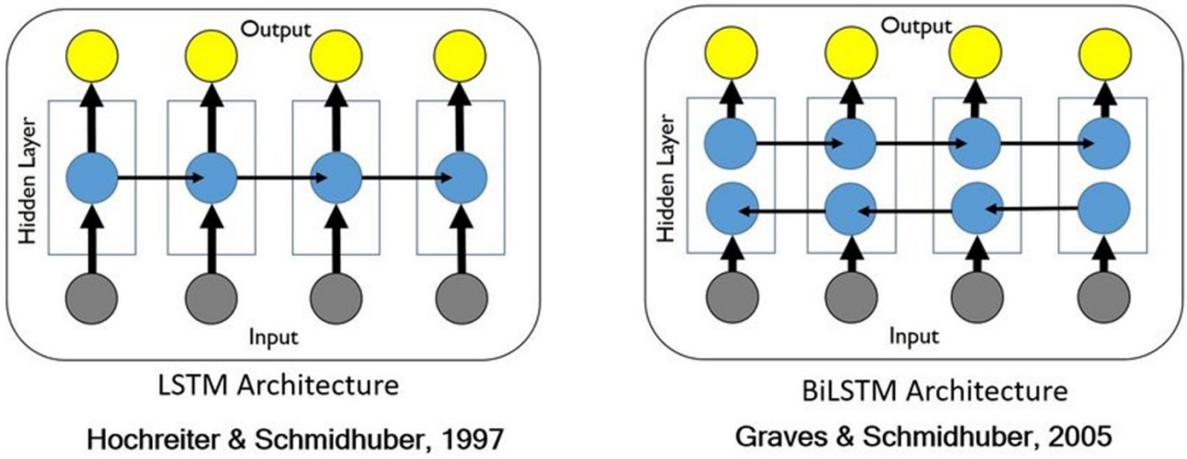


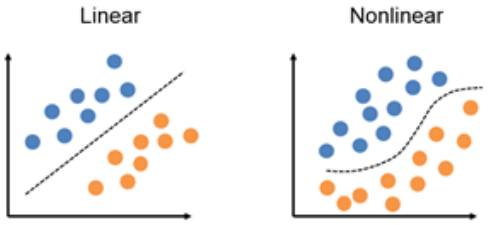
Figure 2.7: BiLSTM Architecture [57]

Examples of applications include using the “one to many” for labeling an image with a sentence or using “many to many” to handle a sequence of images (e.g., input is video generating multiple text outputs) represented in Figure 2.6. When involving images, the input is given to the RNN already transformed by a CNN. Furthermore, the recurrent connections increase the network depth while keeping the number of parameters low since the weights are shared between the inputs. One of the observed problems of RNNs is that the final input will have the context of all past inputs, but the first input does not have the context of the next inputs therefore BiLSTMs were developed which consists of stacking two RNN layers on top of each other as depicted on Figure 2.7.

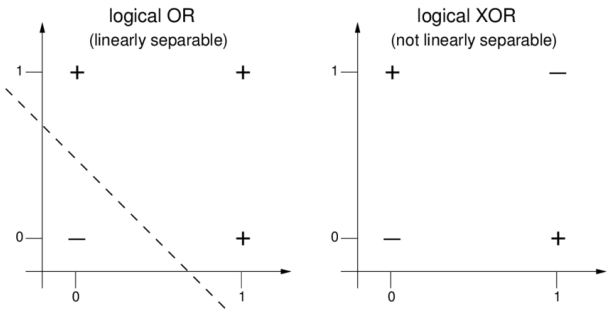
2.4.4 Activation functions

The use of Activation functions is essential for the neural networks mentioned so far. An activation function transforms the value of a neuron before the value is passed to the next layer and important to

be used in final output node of the model to convert values into probabilities for binary or categorical classification. Non-linear activation functions are necessary to produce a non-linear decision boundary via non-linear combinations of the weights and inputs. In Figure 2.8(a), the instances on the first image can be discriminated using a linear decision boundary while to perfectly discriminate the inputs of the second image requires non linearity in the decision boundary. A similar example is given in Figure 2.8(b) where the OR function can be solved with a linear model while the XOR function requires a non linear combination of the two inputs.



(a) Decision boundaries



Input $_1$	Input $_2$	Output
1	1	1
1	0	1
0	1	1
0	0	0

Input $_1$	Input $_2$	Output
1	1	0
1	0	1
0	1	1
0	0	0

(b) OR vs XOR

Figure 2.8: Example of problems not solvable without non-linearity in the model

Some examples of commonly known activation functions are depicted in Figure 2.9.

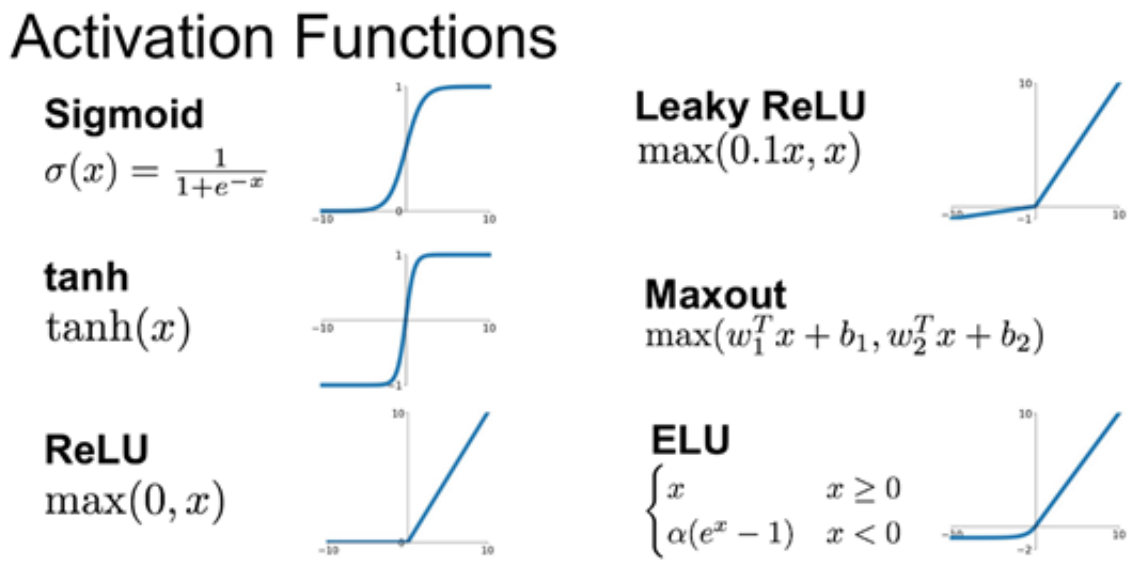


Figure 2.9: Activation functions

2.4.5 Application of Deep Learning Models in genomics

The work proposed in [21] explores the use of MLPs to perform an exhaustive search for SNPs associated with a disease where the results indicate a significant rise in prediction accuracy over other conventional algorithms such as random forest, logistic regression, naïve bayes and gradient boosting machines. The MLP is trained for the classification of samples of a dataset using a multifactor combination of the genotype of SNPs. Combinations with an order of up to 10 SNPs were tested and the classification results (prediction accuracy, classification error) used to search for high risk interacting SNPs associated to a disease. The research work in [23] trained a MLP to discover associations between SNPs and polygenic obesity. The model was trained with 4 different sets of SNPs obtained by filtering the initial number of SNPs based on their individual correlation with the disease using a p-value of 10^{-5} (5 SNPs), 10^{-4} (32 SNPs), 10^{-3} (248 SNPs) and 10^{-2}(2465 SNPs). The model obtained better results as the number of SNPs considered increased meaning the single SNP analysis fails to evaluate the contribution of significant SNPs which was able to be captured by the deep learning model.

The works [22] and [18] both used a CNN and a perturbation-based approach to identify predictive cis-regulatory patterns in DNA sequences. The work [18] proposes an improvement to the efficiency of the perturbation based approach used in [22] and although the work is done with DNA sequence data the same approach can be used for SNPs in epistasis detection. In the work [20] a CNN is used to predict phenotypes from SNPs. A saliency map deep learning visualization approach was used to select significant associated biomarkers from the trained model. The biomarkers indicated by the saliency map were further verified as significant based on gene annotations, literature and GWAS traditional statistical

methods. Both the perturbation-based and saliency map approaches will be discussed in Section 2.5.

In [21] a MLP is used to search exhaustively for groups of SNPs associated with a disease, testing a specific group of SNPs at a time and evaluating the performance of the model. This method is low in terms of efficiency since it is still an exhaustive approach. Modeling the whole SNPs data at the same time and combining it with model interpretability approaches, which will be discussed in Section 2.5, is a far more efficient approach, such as the one used in [20]. Most works are solely focused on comparing predictive performance while this type of approach for actual epistasis detection still seems rather unexplored.

2.5 Deep Learning Models Interpretability

There is a varied number of approaches [58] and different purposes for introducing or adding interpretability to a machine learning model. This section focuses on deep learning models interpretability and the goal of discovering potentially causal associations. The DNNs have a high prediction accuracy and the capability of learning important features and model non-linear interactions without any prior assumptions. However, using DNNs merely as a good prediction model is not enough since the goal of epistasis detection is the identification of SNPs which are significantly correlated with a disease through interactive effects. A major disadvantage of using deep learning models is that insights about the data and the task the machine solves is hidden in a complex model.

The need for explainability arises when there is a discrepancy between the explicit objectives of a deep learning model (e.g., classification task) and the real goal such as epistasis detection. In the case of epistasis detection, interpretability means the discovery of causal associations between the variables and outputs of the trained deep learning model. One simple way to introduce interpretability would be to restrict the choice of the type of model to families considered inherently interpretable such as linear models, decision trees, generalized additive models, etc. These model families are in general relatively simple and thus inadequate for capturing the complexity of some real-world problems. A common conception, the explainability vs performance tradeoff, is that the more performant a model is the less interpretable it will be and vice versa. Yet several researchers have shown that very high-performance complex models can be made interpretable, the work [59] provides a literature review on those methods. Two main methods can be used to gain insights of the causal associations:

- **Explainable modelling:** Development of a model that is explainable by design;
- **Post-modelling Explainability:** Understanding of a "non-directly interpretable" and complex model after the model has been trained.

2.5.1 Explainable modelling

For explainable modelling the main problem when developing such a model is to construct it in a way that makes it simple enough to guarantee interpretability and at the same time properly fit the underlying data.

"Self-Explaining Neural Network" (SENN) [60] is an example of how through architectural adjustments a complex neural network model can be made inherently interpretable sharing similar principles with the proposed solution in this thesis. SENN is a deep learning model whose architecture resembles a linear model while still making use of complex neural networks. Linear models are arguably considered interpretable when the number of features is kept small and easy to interpret. The following equation describes a linear model $f(x) = \sum_i^n \theta_i x_i + \theta_0$ where inputs x_i represent the features of a model and θ_i the coefficients which are interpreted as the importance/relevance score of such feature to the predicted value.

To achieve these properties and resemblance with a linear model the architecture is separated into three neural networks represented in Figure 2.10. These neural networks are referred to as concept encoder, relevance parametrizer, and aggregator. The encoder neural network is responsible for learning higher order features from the pixels from an image (e.g., ears, eyes). These learned features will be the x_i used in the linear model. The relevance parametrizer is a neural network that from the input image learns how much importance should be attributed to each higher order feature, representing the θ_i (coefficients) of the linear model. Finally, a final neural network receives the aggregation of the features (e.g., ears plus eyes), each scaled by their respective relevance score and is responsible for doing the classification of the model (e.g., cat).

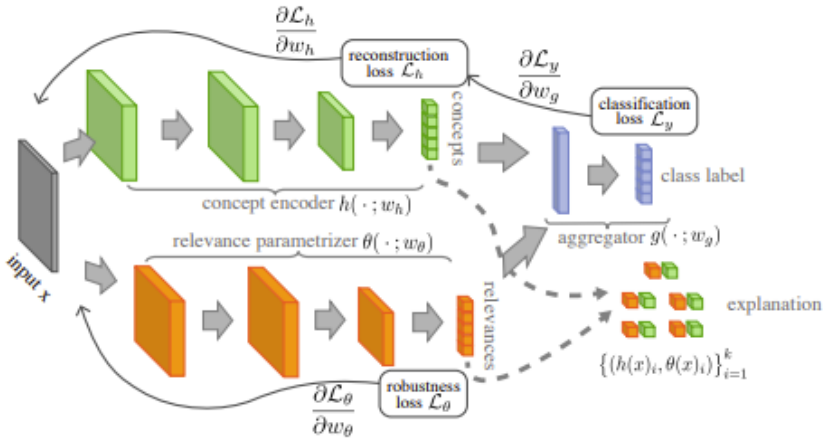


Figure 2.10: SENN model architecture [60]

In this case the strength of each feature causal association with the output corresponds to the respective relevance score learned by the neural network.

2.5.2 Post-modelling explainability

Given the requirement for high performing models that tend to be less explainable in nature given the complex architecture, researchers have given most of their attention to post-modelling approaches on models that have already been developed and trained.

The explanations are commonly constructed by estimating the importance of each raw input feature in relation to the output prediction, but this is not always the case. Sometimes using the raw input features for the basis of the explanation is not computationally feasible or even helpful because of the lack of interpretability such as in images. Instead, we might want to base our explanation on higher order features captured by convolutional layers. In genomics the target of the explanations is the importance of each SNP or local DNA sequence patterns for the cause of a disease or impact in regulatory mechanisms.

Perturbation-based methods

The idea of perturbation-based methods is to generate the importance score of input features by introducing perturbations to the values of such features and observing the impact it has on the output prediction of the model. This kind of method can be applied to any black-box model therefore being a model-agnostic approach.

Local Interpretable Model-agnostic Explanations (LIME) [61] is an example of a perturbation-based method that learns a simpler interpretable linear model that faithfully describes the overall behaviour of the complex model at least locally. LIME generates multiple samples similar to a original sample selected and obtains the output predictions of the model for those generated samples. Then a linear model is learned using the samples and each respective output. The coefficients of the linear model will give us the importance scores of the features. Another popular perturbation-based method is ”SHapley Additive exPlanations” (SHAP) [62]. SHAP takes from game theory the concept of Shapley Values, which in short, represent the average impact that each feature has on the output of the model. The impact of each feature is taken by measuring the difference in the output value of the model when the feature is present versus when it is not. This impact is measured for different subsets of features where the values taken by these are sampled from the available dataset. This guarantees the method is testing values that are reasonably expected and accounting for possible nonlinear interactions, meaning, the impact of the feature being measured can be dependent on the values of other features. SHAP comes with many global interpretation methods based on aggregations of Shapley values. LIME and SHAP are both good methods for explaining models. In theory, SHAP is the better approach as it provides mathematical guarantees for the accuracy and consistency of explanations. One advantage of perturbation methods is not being limited to a specific model, but they are relatively computationally expensive because of the

number of perturbations that need to be used to get a fair explanation of the model.

Backward propagation methods

The backward propagation mechanism is commonly used to generate post-hoc explanations for deep network models. To obtain the importance score of features the method works from the output layer of the model and goes back through the layers estimating how much each neuron is contributing to each previous layer, the process is repeated until the features input layer is reached. The estimation of the contribution of each neuron to the next layer can be done using gradients or by decomposing the activation values of the layers.

Some examples include layer-wise Relevance Propagation (LRP) [63], DeepLIFT [64], SmoothGrad [65], Integrated Gradients (IG) and Deep Taylor [66]. These methods are more computationally efficient when compared to perturbation-based methods since a few only a few inferences of the model are required. However, several researchers have pointed out that these methods can be unreliable. For example, Ghorbani et al. (2019) [67] showed that introducing small (adversarial) perturbations to an image, which still lead to the same prediction, can lead to very different pixels being highlighted as explanations.

Application of model interpretability methods to genomics

The work [22] has implemented a perturbation-based approach to measure the importance of non-coding variants in the DNA sequence in affecting different regulatory mechanisms after the convolutional neural network has been trained. The approach consisted in mutating each nucleotide and measuring the log_2 fold change of the probability of the class of the sample given by the model between the reference and mutated sequence. This method fails to consider the possible effects that are manifested based on the interaction with mutations on other nucleotides and is computationally inefficient because of the number of forward propagations needed to evaluate the effect of feature perturbations. The problem increases exponentially if it tried to include the effect of higher order interactions because of the number of all possible perturbations needed to get a fair evaluation.

The work [18] uses a CNN for the same classification tasks as [22] but proposes a new interpretability method to increase the efficiency and allow to test epistatic effects. Their method is a mix between a perturbation and a backpropagation method which is called a path attribution method. A path attribution method compares the difference between features gradients of a sample to a reference sample. In [18] the “Feature Interaction Score” was a metric responsible for measuring epistatic interactions between pairs of nucleotides. Given a DNA sequence the gradients of the features is extracted from the trained model. Then one of the nucleotides is mutated and the new gradients of all other nucleotides computed

again. The difference in the gradients between the reference sequence and the mutated sequence are the “Feature Interaction Scores”, meaning, the pairwise epistatic effects between the nucleotides and the one that was mutated. All the pairwise interaction scores of a nucleotide with all other nucleotides is measured using only two backpropagations, one for the gradients in the reference DNA sequence and one for the mutated sequence.

The work [20] used a backpropagation approach after their CNN model had been trained. The absolute gradient of each SNP is summed and averaged across all samples to estimate the SNP importance for the prediction of the phenotype of soybean. This method is capable of including the effect of epistasis of any order since the gradient includes the combination of features throughout the layers of the model.

2.6 Attention Mechanism

The attention mechanism which application was experimented in work [68] is introduced to give a basis for the rationale behind the attention vector of the proposed solution in Chapter 3 responsible for filtering the importance of the features. These importance values are the main goal of this thesis since the problem at hand is not solely obtaining a good classification accuracy but as well be able to detect which SNPs share epistatic effects. First, this section will present the initial attention mechanisms introduced to enhance the performance of RNNs and then also mention a type of neural networks which heavily rely on attention mechanisms called Transformers.

2.6.1 Attention mechanism in RNNs

Instead of encoding all the inputs relationships and compressing that information in each single hidden state as explained in Section 2.4.3, we can use an attention mechanism that allows the deep learning model to be able to access multiple hidden states at the same time, instead of relying only on the information contained in the last hidden state. In the case of a translation this would allow the model to prevent sequence overload (too much information contained in a single hidden state about all inputs) by giving the model a way to focus on the likeliest words at each step where for each output the model can directly access specific inputs with an attention mechanism.

For the prediction of the second word, the RNN decoder hidden state D1 receives as input the last output ”Je” plus the context vector C1 as depicted in Figure 2.11. The context vector contains each input hidden state scaled by their respective importance (A0 and A1) for the prediction. To know the values of A0 and A1 an alignment score between the decoder hidden state for the last prediction, represented as D0, and the hidden states of each input, represented as H0 and H1, is computed as depicted in Figure 2.12.

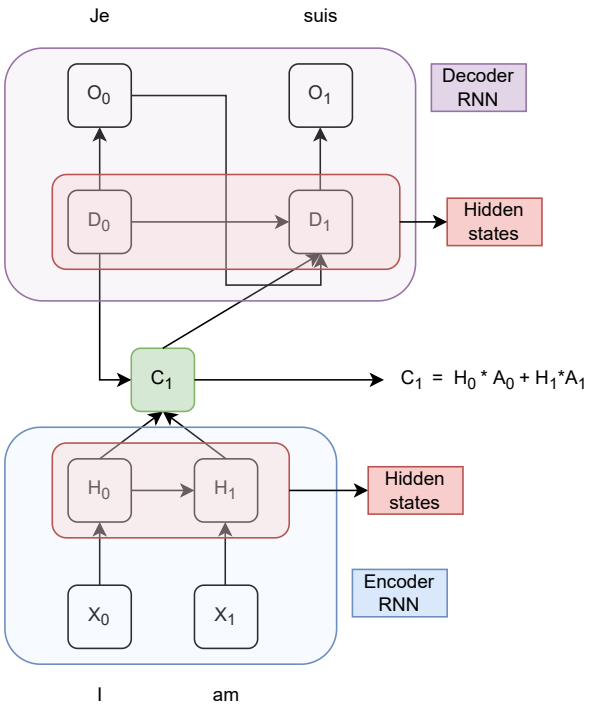


Figure 2.11: Attention mechanism

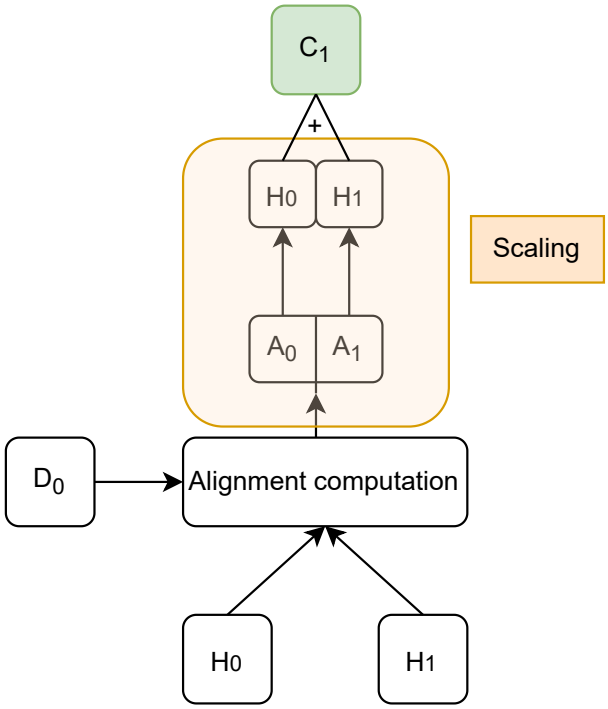


Figure 2.12: Context vector

In this way the RNN can use an attention mechanism to not have to rely on a single hidden state to contain all the information for the different outputs and instead is able to get information from all relevant inputs.

2.6.2 Self-attention and Transformers

Transformers [69] were introduced in 2017 by a team at Google Brain and have become the model of choice for NLP problems replacing RNNs. The full architecture of the model is represented in Figure 2.13, where the inputs pass to an embedding layer, represented by the "Input Embedding" block in the figure, which is responsible for representing each input as a vector. Then, the inputs (e.g., words) represented as vectors go through attention layers represented by the "Multi-Head Attention" blocks in the figure which learns the input-input interactions using the self-attention mechanism. This is computed in a single step instead of a sequential type of architecture like the RNN, this avoids having to store this information in the hidden states of an RNN. The rest of the complex architecture used for translation problems applies multiple attention layers and standard feed forward neural layers to select which groups of words in a text are relevant for the translation of each specific word. For the sake of simplicity and relevance for this thesis we are more concerned in exploring how the attention mechanism is used in the model to learn input-input relationships.

As demonstrated in Figure 2.14(a), which is a simplification of the Transformer model, the attention mechanism present in the Self-Attention layer is basically producing 4 outputs $X1'$,$X2'$,$X3'$ and $X4'$ which are augmented versions of the initial inputs with the input-input interactions information already embedded. For example, output X1' is the result of the sum of the inputs X1, X2, X3 and X4 each scaled by their respective importance where the importance of each input is dependent on how much interaction each input shares with input X1. The rest of the section gives the intuition behind summing inputs which share interactions as well as how the model computes the strength of such interactions.

The context of a sentence is extremely important to understand the meaning of a word therefore a word is dependent on the rest of the words in the sentence just as the effect of a SNP is dependent on its epistatic interaction with the rest of the SNPs. The attention mechanism is responsible for weighing the importance of neighboring inputs to enhance the meaning of the input of interest.

The words in the Transformer are represented as vectors which do not just contain arbitrary values, but instead, words of similar meaning or in this case semantic meaning tend to cluster up as vectors. Taking the sentence represented in Figure 2.15 we might want to "push" the vector representing the word "bank" closer to words like "river" for the model to understand that "bank" in this case is not referring to a financial institution but the side bank of a river. The "importance scores" are calculated using dot products and the dot product of vectors which are closer together obtain a higher score as represented in Figure 2.16, since words which share semantic meaning tend do cluster up as vectors, the dot product

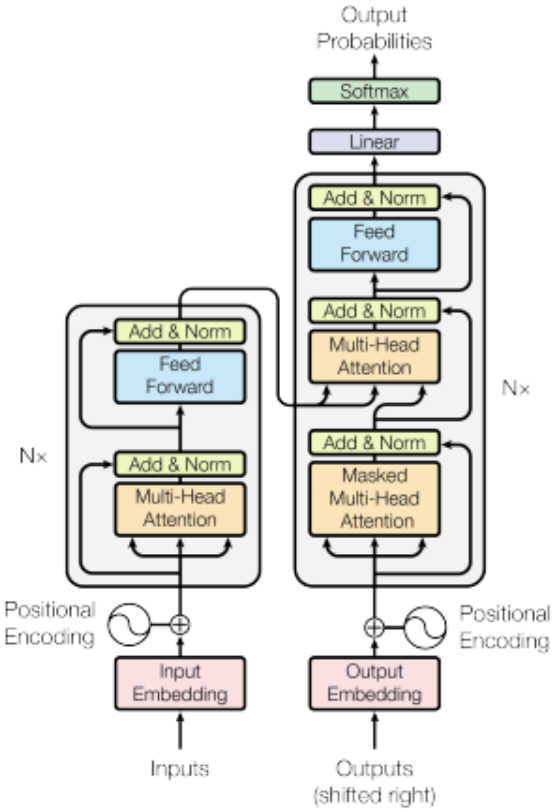
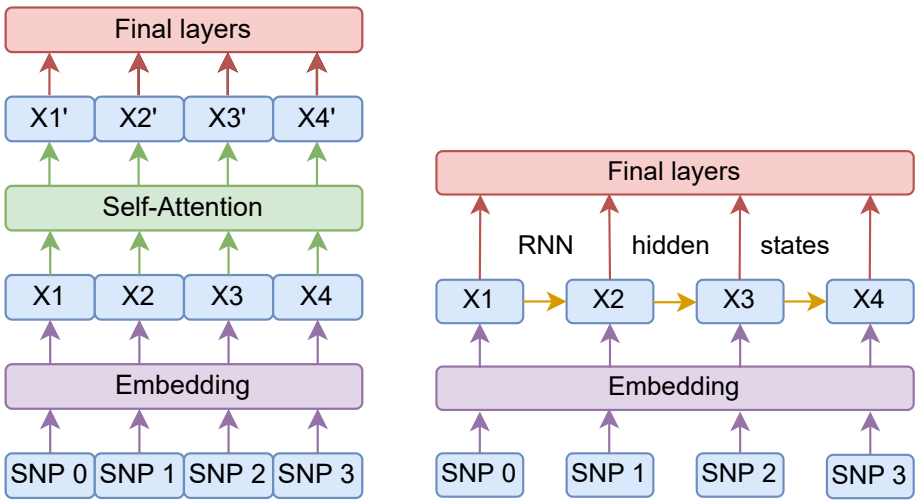


Figure 2.13: Transformer model architecture [69]



(a) Transformer Self-Attention **(b)** RNN hidden states

Figure 2.14: Input-input relationships

will be higher among those words.

The word vector bank after the shift in values when combined with the word river will allow the model to include the interactions and context needed for that word in the augmented input itself. In the attention layer all the interactions between inputs are modeled in a single step between all inputs at all ranges (distances in the input sequence) without the need for sequential steps like the RNN.

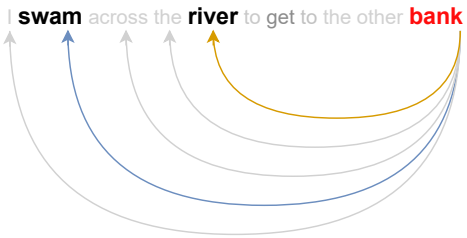


Figure 2.15: Word context

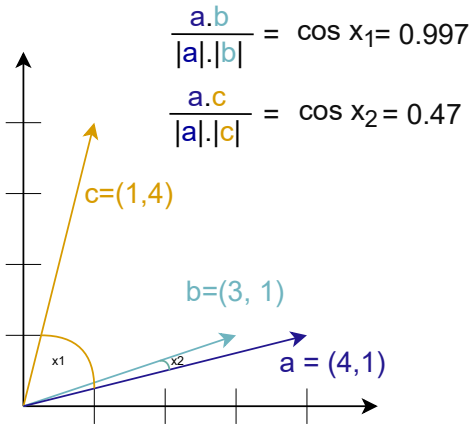


Figure 2.16: Dot product

The interaction scores between the inputs (e.g., SNPs or words) as represented in Figure 2.17 could be used to visualize epistatic interactions.

2.6.3 Application of the attention mechanism for feature selection in genotype-phenotype classification

In the case of problems which require multiple outputs such as text translation, the alignment score mechanism of the attention layers in the RNN and Transformers are useful since the importance of the input features is dynamically adjusted based on the output required. The input-input attention mechanism of the Transformers is also very useful for text problems since the input features (e.g., words) change from sample to sample (e.g., different sentences) therefore the interaction between input 1 and input 2 will always be changing.

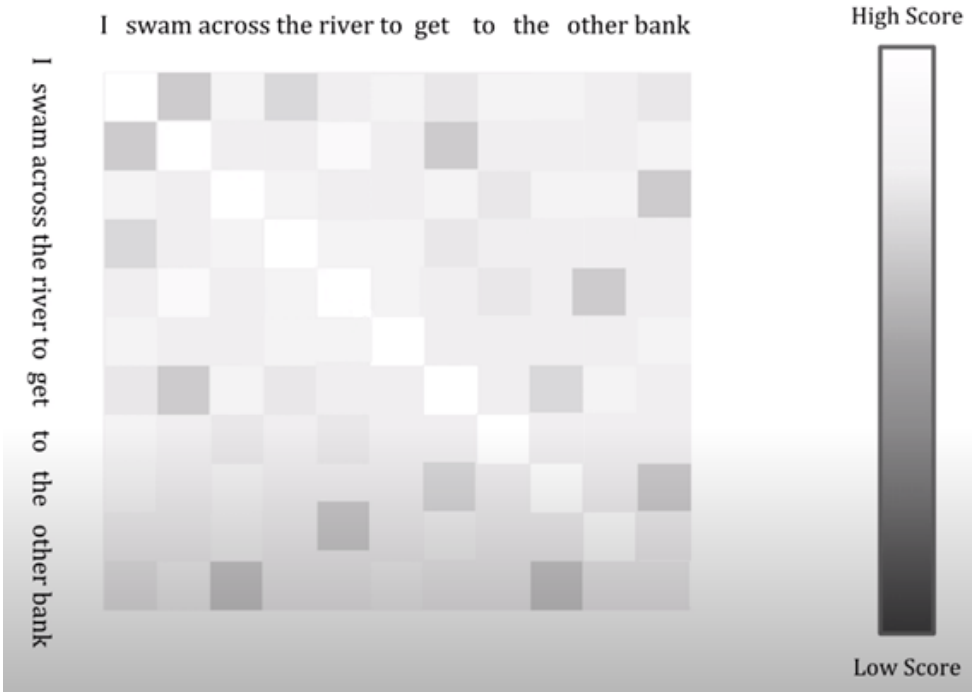


Figure 2.17: Word interaction strength

In the problem of epistasis detection there are no multiple outputs (i.e., the classification only concerns a single class being the probability of disease) and so the context for selecting the inputs based on the type of output is constant for all samples. Besides that, the inputs also remain constant meaning input 1 will always refer to values of the same SNP in the samples of the dataset. Because of these reasons a standard neural network can be responsible for acting as the attention layer just as demonstrated by the "relevance parametrizer" in SENN [60] explored in Section 2.5.1. The attention layer and the concept of the augmented vectors including all the necessary information for the model understanding of interactions will be leveraged for epistasis detection and explored in Chapter 3.

2.7 Summary

This chapter was important to point out the current machine learning methods which avoid an exhaustive approach and that Deep Learning Models (DNNs) have been proven to surpass the performance of these machine learning methods in genotype-phenotype classification tasks. Then, the chapter mentioned that the goal of epistasis detection can be achieved by using model interpretability approaches for Deep Learning Models such as perturbation or backpropagation methods and explainable modelling. Perturbation methods analysis of feature importance were described as computationally expensive to

conduct. Backpropagation methods being the more efficient alternative but not without the problem of still requiring an average of the effects of features over all samples to get a global, instead of a local explanation since the gradients can shift from sample to sample. Also, the explanations given by gradients are not perfectly accurate and in some cases prove themselves to be unreliable. Examples of both methods application in the genomics field were described. Finally, an introduction into attention mechanisms which can act as feature selectors, as well as how the interactions between features can be modeled by joining features together as vectors start to paint a picture of how explainable modelling could be a novel approach to epistasis detection with a deep learning model which will be explored in Chapter 3.

3

Proposed Solution

Contents

In this chapter a deep learning model for epistasis detection is proposed. The neural network architecture allows for a novel approach to detect SNP importance after the model has been trained on genotype-phenotype data. A specific layer is introduced for producing a vector that acts as a feature selector. This vector allows the whole architecture to resemble a linear model, while still making use of nonlinear combinations of features which are required to guarantee the necessary complexity to model difficult epistasis patterns of any order. The vector has as many values as the number of SNPs contained in the dataset and each one of these values correspond to the importance given by the model to each SNP for the classification of the dataset samples. Therefore, these importance scores correspond to the strength of the causal associations between SNPs and the disease. This information is leveraged to filter and obtain a small set of important SNPs upon which an exhaustive statistical test can be efficiently applied to detect significative epistatic interactions.

This chapter starts by providing a detailed description of how the SNPs information is converted into a prediction of phenotype. Then, it will explore the rationale behind the attention vector to estimate the SNP importance and finally present the whole process of extracting this information and detecting epistasis.

3.1 Model architecture overview

The proposed model in the scope of this MSc Thesis is a neural network which receives as input the genotype information to be trained on and predicts the probability of disease based on this information. The architecture of the model is depicted in Figure 3.1 where at the bottom of the figure are represented the SNPs values of a sample. Before passing these values into the input of the model they are preprocessed and transformed into unique natural numbers represented by the Tokenization layer in the figure. These numbers are then used as input to the Embedding layer, this being the actual first layer of the model and represented as orange in the figure. The embedding layer is responsible for transforming each natural number representing a SNP value into a unique vector.

Then, each SNP vector is scaled by their importance, which is represented by the green nodes in the right side of the figure (A_o, A_1, ...) these being the output of a dense layer. The importance value representing how much correlation there is between the SNP and the disease including not only the independent but also interactive effects. After being scaled, the SNP vectors are summed elementwise into a single vector, represented by the ”Elementwise sum of SNPs vectors” layer in the figure. Finally, this single vector is used as input for the final dense layers of the model where the values are transformed until reaching the Output node of the neural network represented by the yellow node at the top.

The next subsections will describe the whole process mentioned above in detail, as well the reasons for the structure of the attention layer.

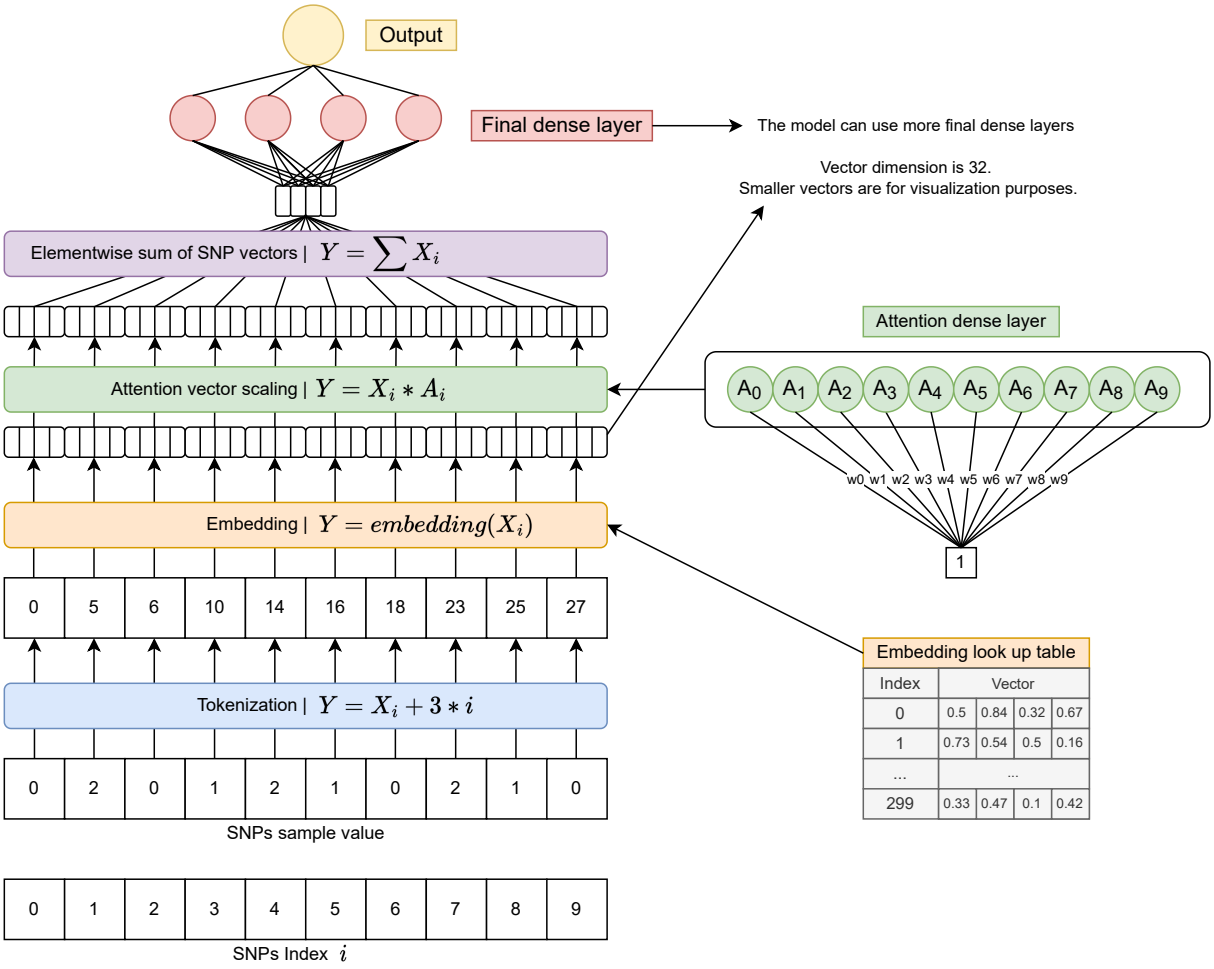


Figure 3.1: Proposed solution architecture

3.1.1 Input and the embedding layer

Categorical variables such as the variants of a SNP “AA, Aa and aa” must first be turned into a numerical variable to be used by the neural network. One example would be encoding the 3 different variants with the values of 0, 1 and 2. The problem in this approach is assuming the variants to be an ordinal variable which is not reflective of the SNP genotype values, for example value 2 does not mean an increase in magnitude over value 1. Since genotype values are a nominal variable, One Hot Encoding is a common approach. One Hot Encoding involves transforming each SNP genotype into a sparse vector with size 3 filled with zeros where the value 1 indicates the genotype matches that variant represented in Table 3.1.

The proposed model instead of using these sparse vectors from One Hot Encoding, uses an embedding layer responsible for optimizing the N-dimensional space representation (i.e., a vector) of the values

SNP variant	Vector		
AA	1	0	0
Aa	0	1	0
aa	0	0	1

Table 3.1: One Hot Encoding vectors for each SNP genotype

of the SNPs making it easier for the next layers to learn the decision boundary between the absence and manifestation of disease. The reason behind this approach is that the values of the vectors used to represent the SNP variants can be learned and adjusted during training the same way the weights are with backpropagation. As such, SNPs variants vectors which are correlated with the probability of disease can be transformed and differentiated from the redundant SNPs in the spatial representation just as the words of similar and different semantic meaning are for models solving NLP problems explored in Section 2.6.2.

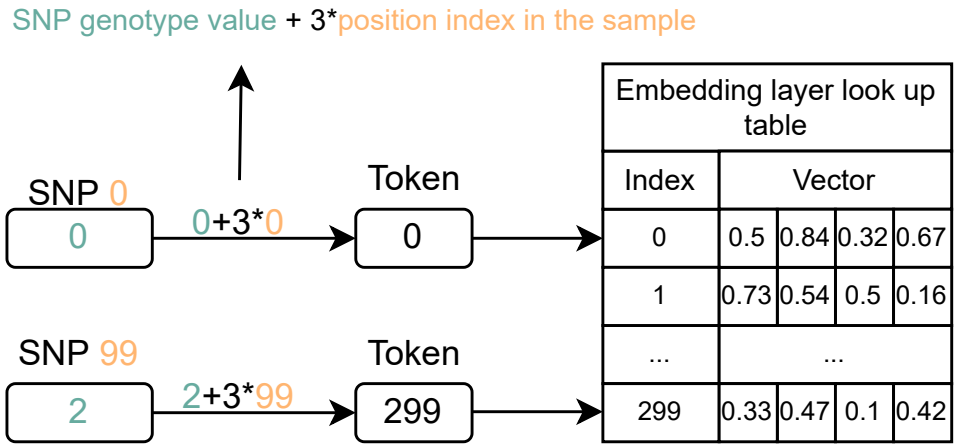


Figure 3.2: SNP as vectors

Transforming the SNP values into vectors with a dimension of 32 aims at providing an increase in the performance of the model. The embedding layer is a look up table where each unique token (i.e., a natural number) is attributed initially a random vector which values are then adjusted during training. Each SNP value X_i (highlighted as green in Figure 3.2), where i represents the index of the SNP in the sample sequence (highlighted as orange in Figure 3.2), needs to first be converted into a token Y. This is achieved by using equation $Y = X_i + 3 * i$ which guarantees that each of the 3 possible variants of a SNP gets a unique token. Given a token, the embedding layer retrieves a vector.

3.1.2 Attention layer

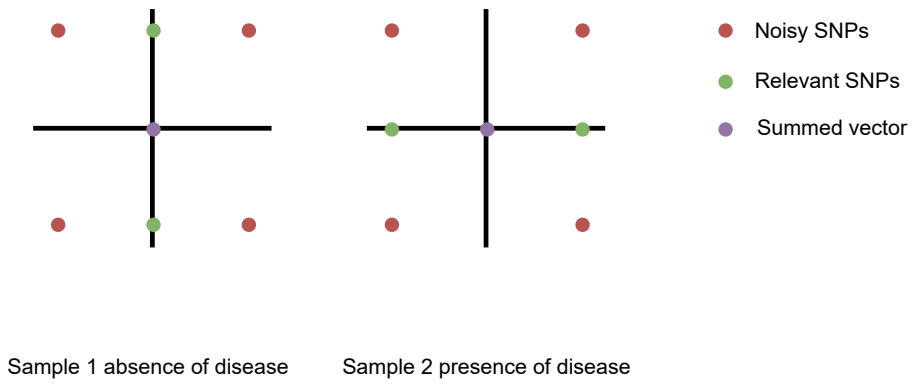
After transforming each SNP variant into a vector, which is the output from the orange embedding layer in Figure 3.1, comes the attention layer of the model. The attention layer has one weight for each SNP represented by the green circles in Figure 3.1. These weights (A_0, A_1 ...) are the output neurons of a dense layer which input is not connected to the rest of the neural network model, the reason for this detail will be explored in Section 3.1.3. Each SNP vector is afterwards scaled by their respective weight and because each weight is learned during the model training, it means the higher the value of the weight, the higher the importance of the SNP for the classification of the samples. After scaling each one of the SNP vectors by their respective importance (multiplication with the attention values), all the SNPs are summed into a single vector.

This single vector contains the combination of the values of all the SNPs, so any interaction between SNPs is accounted for by mixing their values. The better the model is at giving importance to the right SNPs, the better the quality of the vector (i.e., less noisy information). This vector can be viewed as an ”augmented” input from the Transformer model. In a translation problem multiple different outputs (i.e., translated words) require information from different ”augmented” inputs (i.e., different groups words from a text). In the case of epistasis detection the output of the proposed model always remains being the probability of disease in question, therefore a single ”augmented” input is used. Finally, the vector is fed into a final dense layer or layers responsible for doing the classification with a sigmoid activation function in the final node which outputs a value of 0 or 1 which respectively represents a control or case of the disease.

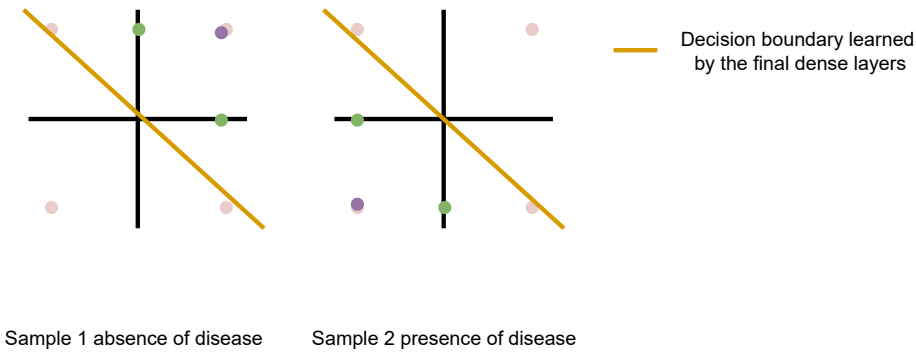
An attempt at providing a more intuitive visual representation is represented in Figure 3.3 where the SNPs vectors are represented in a 2-dimensional space for visualization purposes. The green points represent the interactive SNPs correlated with the disease while the red points represent the noisy SNPs of a dataset. The purple point represents the elementwise sum of all SNP vectors that will be passed on to the final layers responsible for giving the classification of the sample as explained before which in this example is the sum of all SNPs points.

The importance of each SNP will be represented by the opacity of the points represented in the figure. After the model has been trained the noisy SNPs are attributed a lower importance and the relevant SNPs a higher importance as shown by the difference in opacity of the points between Figure 3.3(a) and Figure 3.3(b). As a consequence, the purple point will be mostly composed by the green points information which is represented by the shift of the purple point between Figure 3.3(a) and Figure 3.3(b).

At the same time during training, the embedding layer has also adjusted the representation of the SNPs which is represented by the shift in the coordinates of the relevant SNPs between Figure 3.3(a) and Figure 3.3(b). The shift in the importance of the SNPs and the shift in the representation of the SNPs vectors allows the model to use the new purple point representation to distinguish between cases



(a) SNP vectors before training



(b) SNP vectors after training

Figure 3.3: Visual representation of the SNPs vectors weighted sum

and control samples. This discrimination is done by the orange decision boundary represented in Figure 3.3(b) which is created by the final dense layers of the proposed model.

Assuming the model is trying to learn the epistatic interaction depicted in Table 3.2 between SNP 1 and SNP 10 and that all the other SNPs do not have a correlation with the disease, Figure 3.4 shows a perfectly learned attention vector where none of the noisy SNPs are attributed any importance. Since all the noisy SNPs are given no importance, the model can be represented as in Figure 3.5 where the model takes as input 4 different genotype combinations of the important SNPs. The model can now classify each sample as a case or control according to the 4 different possible values of the summed SNPs vectors.

SNP1	SNP10	Case/Control(1/0)
0	0	0
0	1	1
1	0	1
1	1	0

Table 3.2: Simple epistatic interaction example using an XOR function

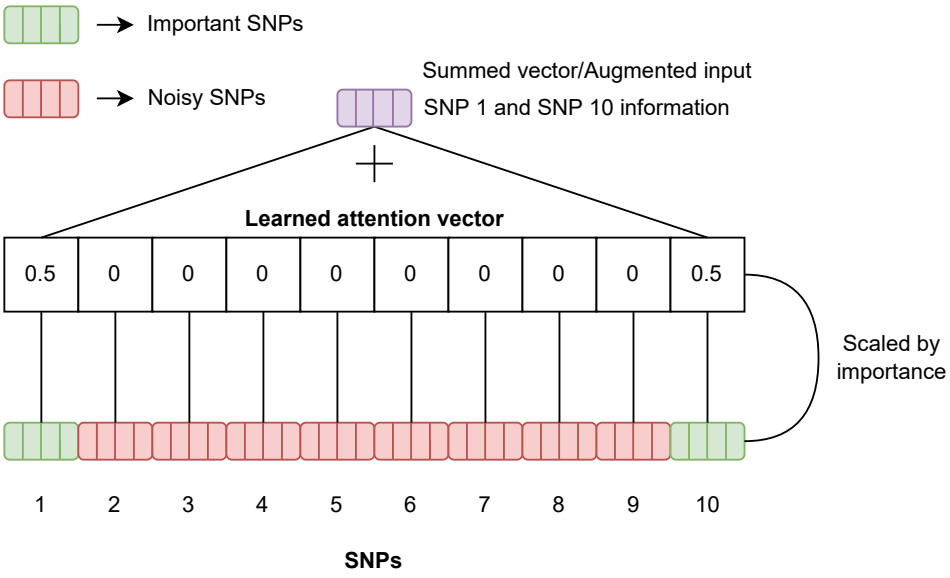


Figure 3.4: Perfect attention vector

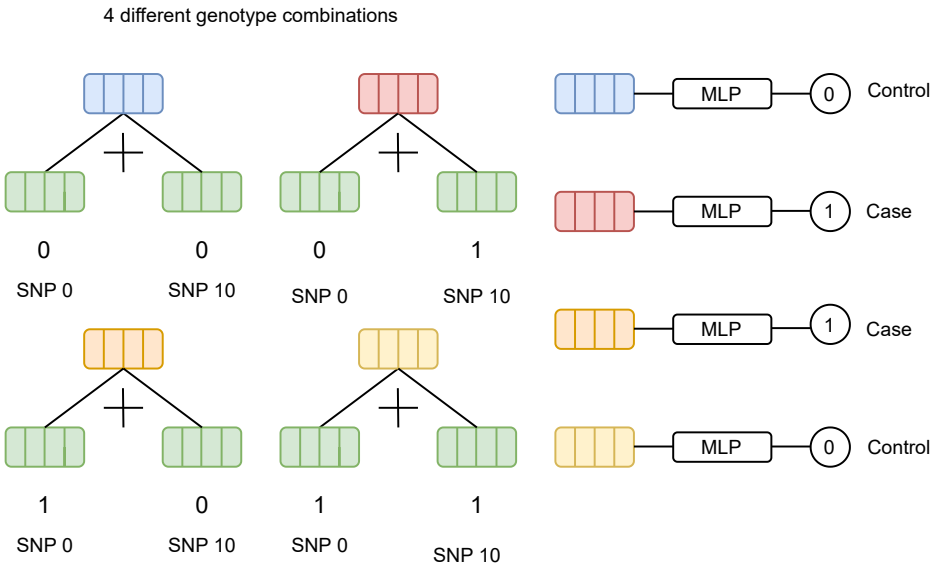


Figure 3.5: 4 different genotype combinations of SNPs classified by the model

3.1.3 Model interpretability through the attention layer

The modelling power of standard dense layers mixing the values of the SNPs is the same as the architecture of the model described so far except for the improvement in using the embedding layer which is the reason for the rest of the architecture as explained below.

The concept of summing the features together while being scaled by the weights is already established since it is present in every node of every standard neural network model. The important property of this model architecture is the summing of the SNP vectors idea which came from the Transformers model and is important to maintain a low number of parameters. For example, applying a standard dense layer with 50 neurons to an input with 500 SNPs vectors with size 32, which is equivalent to 16000 values (500*32), would result in 800000 different weights (16000*50) just in this first layer. By using this architecture a single vector with 32 values with all the required information is propagated to the next layers instead.

This is why the weights/attention values are learned in a dense layer where the input is separated from the original neural network as represented in Figure 3.1 for coding reasons. The dense layer is used to provide a single weight for each vector while the rest of the calculations are produced from custom layers which have the simple task of multiplying (i.e., scaling the SNPs) and doing an element-wise addition of vectors. Just summing the vectors together without the attention values scaling would not allow for the distinction between valuable SNP vectors and noisy SNP vectors and the filtering of noise. By consequence the presence of these attention values allows for a different type of approach for model interpretability.

The problem is dense layers are not directly interpretable because the importance of each feature is scattered along the multiple neurons of the model. As explained in Section 2.5 in order to gather the importance of each feature in neural network perturbation methods are one of the alternatives but are not efficient. To measure the importance of the SNPs it would require changing each individual SNP into its other possible variants, one at a time to measure the impact in the output of the model for the sample in question. For 100 SNPs it would require 200 tests since each SNP has 2 other possible values (2*100) besides the one already present in the sample. And this procedure only measures marginal effects. For pair-wise interactions and high order interactions it would get exponentially worse. To measure the impact of pairs of SNPs it requires 39600 tests for a single sample since we have 4950 pair-wise combinations and each pair can take 8 different other possible genotype combinations (4950*8). This type of testing would make the deep learning model approach become less efficient than an exhaustive statistical testing approach.

Backpropagation methods are a better approach because of their efficiency. Any epistasis patterns effects modeled by the layers will be recognized in the gradient computation. Still, the epistatic and the independent effects are not present in a single sample or SNP combination. Therefore, the gradients

provide a local explanation and for a fair estimation of the SNPs importance the average of the absolute of the gradient over all samples is required.

The approach used in this thesis is explainable modelling. The architecture presented allows to gather the importance of each SNP without making use of either perturbation or backpropagation methods because the attention vector in the model makes the model represented in Figure 3.1 resemble a linear model represented in Figure 3.6. The model using 3 SNPs for example can be represented by the equation $Y = f(y)$ where $y = x1 * w1 + x2 * w2 + x3 * w3$. Function f is representing the transformations applied by the final dense layers of the model.

To measure how much the change in each SNP value is responsible for the error of the output of the model we can calculate the gradients of the error function with respect to each SNP represented in Equation (3.1), like in a backpropagation method . Since the model resembles a linear model and all the SNP values are funneled into a single vector the terms $\frac{\partial E}{\partial Y}$ and $\frac{\partial Y}{\partial y}$ are the same for all SNPs and can be ignored when comparing the importance of SNPs. Leaving the last term $\frac{\partial y}{\partial x_i}$ to represent the importance of the SNP which is just the weight associated with each SNP vector by solving Equation (3.2).

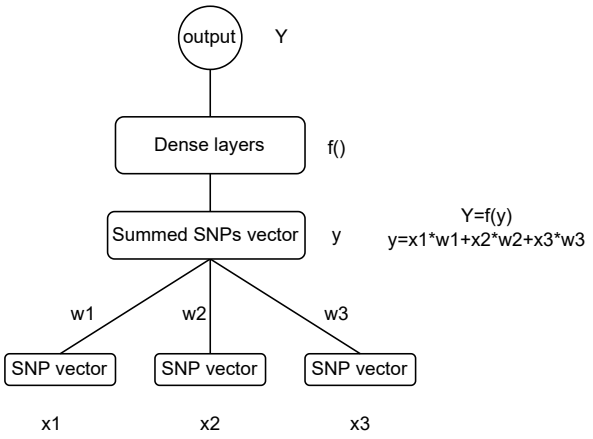


Figure 3.6: The architecture representation as a linear model

$$\frac{\partial E}{\partial x_i} = \frac{\partial E}{\partial Y} * \frac{\partial Y}{\partial y} * \frac{\partial y}{\partial x_i} \tag{3.1}$$

$$\frac{\partial y}{\partial x_i} = w_i \tag{3.2}$$

Since the values of the weights were trained to represent how correlated the change in the SNP value is with the predictability of disease over all the samples of the dataset, it represents the global effect of each SNP. The weights values are also independent of the samples values after training, therefore the importance of each SNP using the proposed model is given by the values of the weights in the attention layer without having to do any further testing. This feature is useful since the goal is to compare the

overall SNP importance and not to provide an explanation of the classification of each specific sample. When using gradient backpropagation methods for the same purpose, to obtain a fair estimation of the global importance of each SNP it requires the computation of the gradients for each sample and then an average of those results since the gradients change depending on the activation values of the layers which are dependent on the samples values.

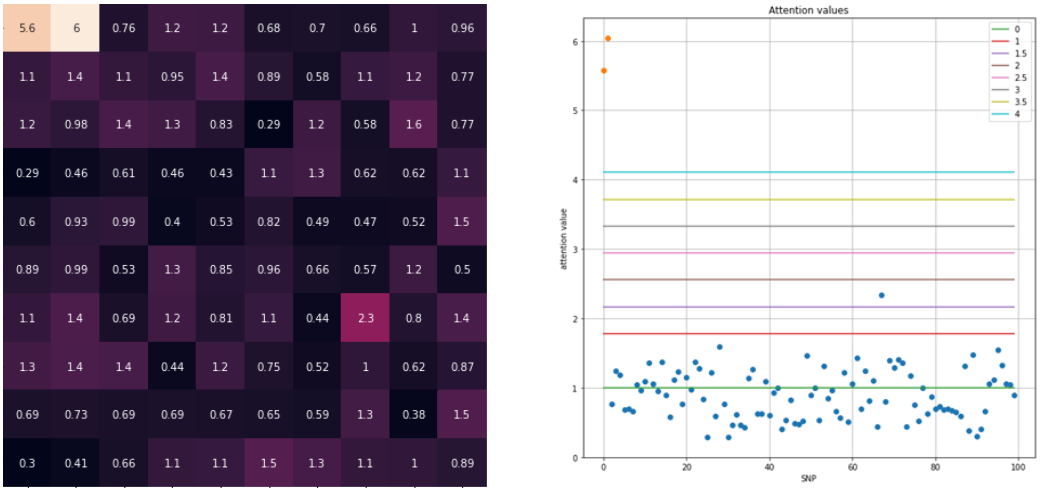
3.2 Filtering stage for Epistasis detection

3.2.1 SNPs filtering overview

In order to filter the SNPs we use standard deviations from the mean of the attention values as a threshold to decide which SNPs are considered significant enough to have a strong correlation with the disease. In Figures 3.7(a) and 3.7(b), the squares of the first image represent the 100 attention values for each of the 100 SNPs in the samples of a dataset after the model has been trained while the second image represents the same values in a scatter plot. The dataset contained a significative epistatic interaction between SNP 0 and SNP 1 represented by the first two values of the attention vector of Figure 3.7(a) highlighted by a lighter color based on the higher values of attention. The same two attention values are represented by the orange points in Figure 3.7(b). The colored lines in Figure 3.7(b) representing the standard deviations from the mean.

After the SNPs are selected based on a standard deviation threshold, all possible combinations of those SNPs based on the orders of interactions we desired to search for go through statistical tests as represented in Figure 3.8.

In Figure 3.8 are represented 10 attention values learned during the training of the proposed model and represent the importance attributed to 10 SNPs. The "SNP attention value $> std_{dev} * x + mean$ " purple block represents the standard deviations above the mean threshold filtering applied to the attention values which is chosen by the value x in the formula. Then, the filtered SNPs represented in the next green block are combined into all the possible combinations up to a desired order, in this case 3, represented in the next yellow block of the figure. Finally, the last blue block in the figure represents the testing of the combinations of SNPs which consist of a standard independence chi square test which measures the correlation with the disease followed by a conditional independence chi square test which distinguishes SNPs with epistatic effects from SNPs with only marginal effects. The next subsection describes the details of the testing.



(a) Attention Vector **(b)** Scatter Plot of the Attention Values

Figure 3.7: Attention Values Standard Deviation Thresholds

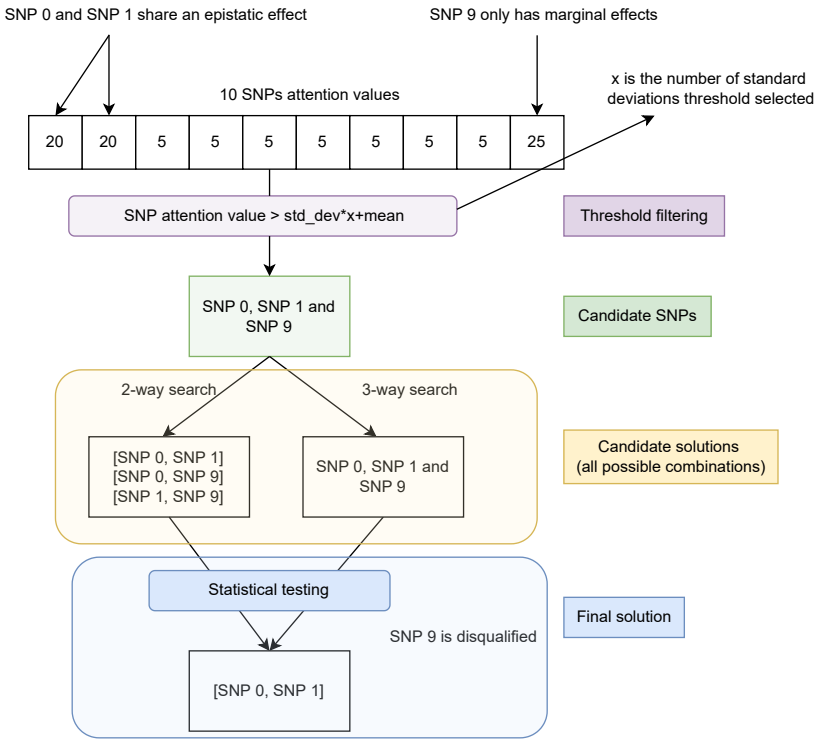


Figure 3.8: Detection of epistatic interactions

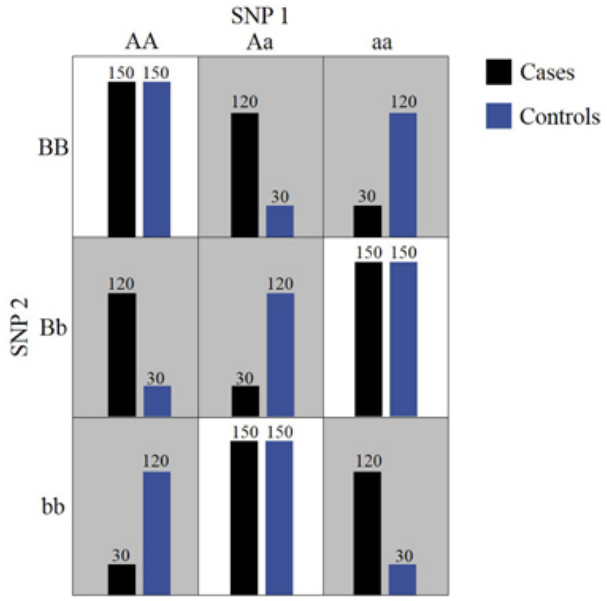


Figure 3.9: SNPs contigency table

3.2.2 Statistical testing

After the threshold filtering the possible combinations of SNPs go through two statistical tests, the chi-square test, also commonly represented as the χ^2 test, and the conditional χ^2 test. Both tests are important for precision in the discovery of epistasis which reasons will be explored in this section.

χ^2 test and G-test

The χ^2 test is common a statistical test used in epistasis detection to check if the proposed SNP or SNP combinations are of significance.

The χ^2 test is used to determine whether there is a statistically significant difference between the expected frequencies and the observed frequencies of the SNPs genotype contingency table such as the one represented in Figure 3.9, where the expected frequencies are the frequencies of a null hypothesis. The null hypothesis is the case where the SNPs have no correlation with the disease therefore for a combination of SNPs it would mean a 50% probability of disease for all genotype combinations (assuming the general probability of disease is 50% as well). The bigger the difference between the null hypothesis and the observed frequencies, meaning the further away from a 50% frequency of disease on the genotype combinations, the stronger are the effects of SNPs on the probability of disease. Depending on the difference measured it can be deemed significative or not significative based on a specified threshold using p-values.

The χ^2 test formula is represented by the following equation $\chi^2 = \sum_{i=1}^{I} \sum_{j=1}^{J} \frac{(O_{ij}-E_{ij})^2}{E_{ij}}$, where

O_{ij} corresponds to n_{ij}, the number of observations where the SNPs take the i^{th} genotype and the j^{th} phenotype. Taking n_i as the number of observations with the i^{th} genotype and n_j as the number of observations with the j^{th} phenotype, the expected frequency is calculated using $E_{ij} = \frac{n_i * n_j}{n}$.

Another alternative is a G-test which is increasingly being used to substitute the chi-squared test and works in a very similar way which formula is represented in Equation (3.3) [70].

$$G = 2 \sum_{i=1}^{I} \sum_{j=1}^{J} O_{ij} * ln \frac{O_{ij}}{E_{ij}} \tag{3.3}$$

χ^2 Conditional Independence Test

The χ^2 test measures the correlation of combinations of SNPs with the disease but is unable to distinguish marginal from non marginal effects. It is possible that a SNP pair is strongly correlated with the disease because of the marginal effect of one or both of the SNPs.

In order to evaluate the strength of interactive effects in a combination of SNPs we can rely on a χ^2 conditional independence test. It measures the difference between the observed frequencies of cases and controls of a combination of SNPs genotype compared to the values of another combination of SNPs or single SNP genotype (which in a standard test would be the expected frequencies of a null hypothesis).

The difference between both tests is represented in Figure 3.10 where SNP 0 presents marginal effects while SNP 1 has no effect on the probability of disease which is why no change on the frequency of disease occurs when changing from genotype BB to Bb or bb. The χ^2 test would compare the observed frequencies (top left) and the expected frequencies (top right) under a null hypothesis. The difference between the frequencies classifies the pair of SNPs as significant. The conditional test however would consider the third contingency table on the bottom which are the frequencies of disease based solely on the genotype of SNP 0 (AA,Aa and aa) repeated in 3 rows for a visual comparison. There is no difference between the frequencies of disease based on the combination of both SNPs (top left table) and based solely on SNP 0 which means SNP 1 does not share an epistatic interaction with SNP 0.

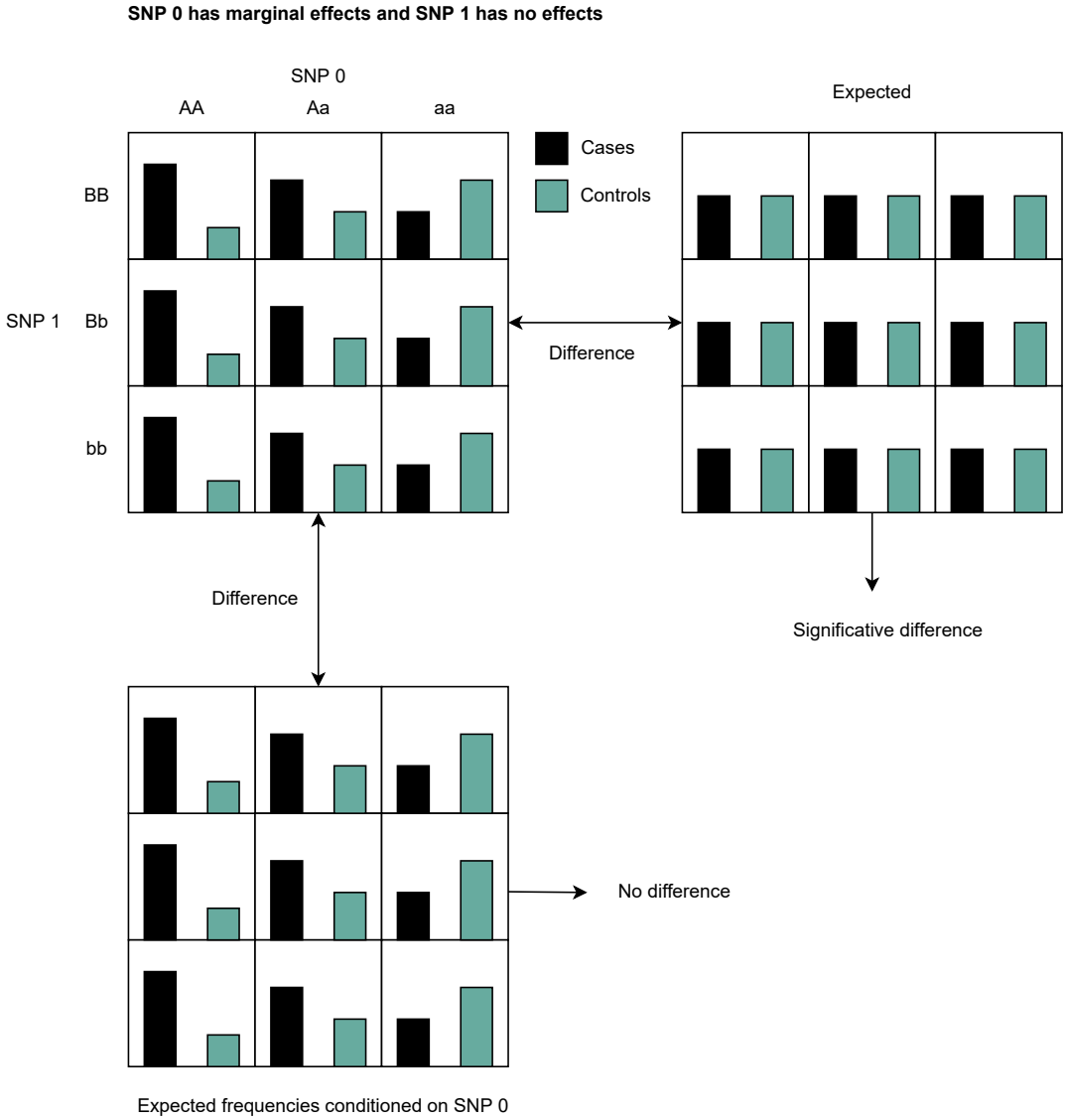


Figure 3.10: Standard vs conditional expected frequencies

Table 3.3 and Table 3.4 show the p-value of both the standard χ^2 test and the conditional independence χ^2 test on a pair of an epistatic SNP with a random SNP (SNP 0 and SNP 98) and the epistatic SNP pair (SNP 0 and SNP 1) respectively. Even though both pairs present significant p-values with the standard test, the conditional test allows us to reject the first pair as sharing an epistatic interaction.

Table 3.3: Pair wtih a single strong marginal effect SNP example from a marginal dataset

Epistatic SNP with a noisy SNP	SNP 0	SNP 98	SNP 0, SNP 98
Standard test P-value	1.84e-16	0.044	1.87e-13
Conditional independence P-value	2.44e-13	0.56	0.56(maximum)

Table 3.4: Epistatic SNP pair example from a marginal dataset

Epistatic SNP pair	SNP 0	SNP 1	SNP 0, SNP 1
Standard test P-value	1.84e-16	2.17e-10	3.18e-22
Conditional independence P-value	1.86e-14	8.53e-09	8.53e-09(maximum)

The conditional independence p-values of each SNP represent the p-value of that SNP conditioned on the other SNP of the pair, where the lower the p-value the more significant is the correlation. In Table 3.3 SNP 0 shows as contributing strongly to the correlation with the disease when conditioned on SNP 98. SNP 98 on the other hand did not contribute significantly to the correlation with the disease when conditioned on SNP 0 since it has a very poor p-value of 0.56. The exact opposite effect is observed in Table 3.4 where each SNP contributes strongly to the correlation with the disease while being conditioned on the other SNP of the pair. In the epistatic pair both SNPs get an increase in the correlation with the disease when combined together observed in the difference between each individual SNP χ^2 test p-value and the p-value of the the combination (3.18-e22).

In our filtering stage we use the highest p-value or in the other words the least significant p-value of the conditional test to check if both SNPs are contributing significantly to form an epistatic interaction. This is fundamental to ensure the model is able to have a good precision when distinguishing independent from interactive effects.

3.3 Summary

The proposed solution is a deep learning model used to learn the classification of samples as a case or control from SNP data. The model uses an embedding layer to represent the SNPs values as vectors and is capable of learning the correlation between SNP epistatic effects and the disease. After model training, a single vector with the importance values of the SNPs for predicting the probability of disease in question is obtained which are weights trained by the model. These attention values are an approximation of the same concept as gradients for model interpretability and represent how relevant the information of each SNP is for the classification of all samples of a dataset. The proposed model

does not require any testing with different perturbations (requirement for perturbation methods) or backpropagation computation over different samples (required for backpropagation methods) therefore being a more efficient and direct way to interpret feature importance. All details were covered in Section 3.1.

Section 3.2 focused on how the epistasis detection process is conducted after model training, describing the details of the process and the reason for necessary statistical tests which are able to detect significative epistatic interactions present in the SNPs filtered by the proposed model based on the learned importance values.

4

Experimental Results

Contents

This chapter starts by explaining the need for different weights initializations which is important to know how the model will be run during the testing on simulated datasets. Then, a description of the metrics used to compare versions of the model during development is given, as well as the parameters of the best version of the model which were used for comparing the model against other state of the art methods in Section 4.4. The metrics that are used to measure algorithms performance in the field of epistasis detection and which will be used in Section 4.4 are also described. Section 4.4 is the final section which contains the comparison of the model on a varied set of simulated datasets against other state of the art methods including datasets with and without marginal effects, 2nd, 3rd and 4th order epistasis interactions and low heritability (strength of the effects of the SNPs on the disease mentioned in Section 2.2.4).

4.1 Seed initialization

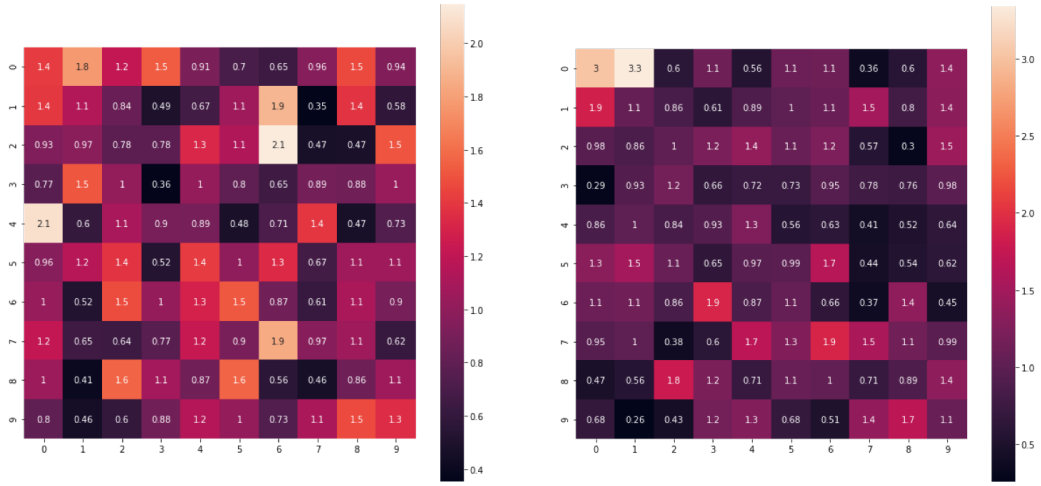
Depending on the initialization values of the weights of the neural network given by the random number generator or also referred to as ”seed”, the model can be unable to learn the weights which capture the epistatic pattern. This is because the adjustment of the weights is sensitive to the initialized values which can make the model get ”stuck” and not achieve the optimal convergence of the gradient to optimize the weights in the correct manner.

In Figure 4.1 each image has 100 squares representing the attention values of each of the 100 SNPs the model was trained with. The epistatic pair is represented by the first two attention values of the first row, representing SNP 0 and SNP 1. The different initialization of weights resulted in the model missing the epistatic pair interaction in Figure 4.1(a) and being able to detect it in Figure 4.1(b) which is observed the by the difference in the values of SNP 0 and SNP 1 between both images where in Figure 4.1(a) the top two attention values correspond to the epistatic pair.

Because of this phenomenon, each dataset is run with different initializations of the model until a significative epistatic interaction of the desired order is discovered or a defined maximum number of tries is reached, although any other stopping method could have been used depending on the objective of the testing.

4.2 Performance metrics of methods for epistasis detection

The detection power of the algorithm is measured by recall, meaning the percentage of datasets in which the epistatic interaction was correctly identified. Another metric is precision which measures the percentage of combinations of SNPs proposed as solutions by the algorithm which are the desired solution, an algorithm that only proposes the desired solution would achieve a precision of 100%. The



(a) Trained attention vector

(b) Trained attention vector with a different initialization

Figure 4.1: Effect of different initializations

calculation of these metrics relies on some terms which are often used to evaluate the performance of machine learning methods in classification tasks but can also be used to understand the performance of methods in finding epistatic interactions. These terms being true positives, false positives, true negatives and false negatives.

True positives (TP): Number of solutions detected by the method which are epistatic interactions **(found interactions)**.

False positives(FP): Number of solutions detected by the method which are **not** epistatic interactions **(false interactions detected)**.

True negatives(TN): Number of solutions **not** detected by the method which are **not** epistatic interactions **(correctly dismissed interactions)**.

False negatives(FN): Number of solutions **not** detected by the method which are epistatic interactions **(missed interactions)**.

Using the terms defined before, precision and recall are obtained from eq. (4.1) and eq. (4.2) respectively. Another metric called the F-score combines both precision and recall represented in eq. (4.3).

$$Precision = \frac{TP}{FP + TP} \tag{4.1}$$

$$Recall = \frac{TP}{FN + TP} \tag{4.2}$$

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.3}$$

Recall or F-score are the standard metrics that will be used to compare our proposed solution against other state of the art methods in Section 4.4.

4.3 Model hyperparameter experimentation

An overview of the range of parameters tested during the optimization of the proposed model is displayed in table 4.1. The first row of the table refers to the activation function which transforms the attention values of the SNPs in the dense layer responsible for learning them, which will be referred as attention layer throughout the rest of this section. The second row refers to the activation functions used in the dense layers that receive as input the summation of the SNPs vectors which neurons are represented by the red nodes in Figure 3.1 which are referred throughout this section as "final dense layers". The third row refers to the number of these final dense layers used in the proposed model architecture. The 4th and 5th rows concern the use of batch normalization applied to the vector which results from the summation of all SNPs vectors and to the output values of the final dense layers respectively. The last two rows are the learning rate and batch size which are two of the main hyperparameters of deep learning models which affect the training process. The rest of this section will go over the details of these hyperparameters and how the optimization of the model was conducted.

Table 4.1: Hyperparameters

Hyperparameter	**Range**
Attention layer activation function	[Softmax, ReLU, LeakyReLU, Linear, Mish]
Final dense layers activation function	[ReLU, LeakyReLU]
Nº of final dense layers	[1,2]
Batch normalization on summation vector	[On, Off]
Batch normalization on final dense layers	[On, Off]
Learning rate	[0.01,0.001]
Batch size	[1,10,16,32,64]

The different parameters were evaluated in 6 different settings of simulated datasets represented in table 4.2. The settings have different degrees of difficulty in the detection of the epistatic pair which is controlled by the heritability parameter of the datasets. Each setting was tested using 100 simulated datasets without marginal effects and a pair-wise epistasis interaction. The different settings had different degrees of difficulty in the detection of the epistatic pair which is controlled by the heritability parameter of the datasets.

The most common approach for optimizing a neural network is performing an exhaustive search of the possible combinations of different values of the hyperparameters which is referred to as a grid-search optimization. Given the number of different possible combinations in the proposed model we instead

Table 4.2: Non marginal effects datasets from MACOED

DNME Model 1	h2=0.4,MAF=0.2			DNME Model 21	h2=0.4,MAF=0.4		
	AA	Aa	AA		AA	Aa	aa
BB	0.4865	0.9601	0.5377	BB	0.0774	0.6563	0.8804
Bb	0.9473	0.0042	0.8113	Bb	0.8917	0.2350	0.3120
bb	0.6401	0.6065	0.9089	bb	0.1742	0.8417	0.1064
DNME Model 26	h2=0.2,MAF=0.4			DNME Model 6	h2=0.2,MAF=0.2		
	AA	Aa	AA		AA	Aa	aa
BB	0.3563	0.8907	0.8085	BB	0.4278	0.7569	0.8117
Bb	0.9547	0.5078	0.6110	Bb	0.7884	0.1316	0.0439
Bb	0.6167	0.7549	0.6304	bb	0.5594	0.5485	0.3730
DNME Model 11	h2=0.05,MAF=0.2			DNME Model 20	h2=0.01,MAF=0.2		
	AA	Aa	AA		AA	Aa	aa
BB	0.4916	0.6637	0.4807	BB	0.1391	0.1882	0.2214
Bb	0.6419	0.3302	0.7457	Bb	0.1901	0.1114	0.0198
bb	0.6555	0.3960	0	Bb	0.2056	0.0514	0.2530

performed an iterative optimization, meaning we started first by selecting the optimal learning rate and batch size and then using these selected hyperparameters as a baseline before proceeding to optimize the rest of the parameters. This approach was also done since the problem of epistasis detection is different from a simple classification task. In the case of a classification task, the model is evaluated according to the prediction error of the model in a dataset. In the case of an epistasis detection problem the model is not evaluated based on the prediction performance and instead based on the features the model considered important for the classification of the samples. These being the weights attributed to the SNPs. In this case we evaluate the model based on the recall of epistatic interactions, the precision of model depending on the number of standard deviations threshold and also the average number of different initializations of weights required to achieve such performance. For this process a more manual and monitored analysis was required instead of an automated optimization search.

4.3.1 Batch size and learning rate

The range of the batch size and learning rate values selected for testing was done based on the most common values used for neural networks according to the literature [71], [72], [73].

The Batch size is the number of samples from the training dataset used by the model to estimate the error before doing a weight update.

The 3 main common configurations are:

• Batch Gradient Descent: Batch size is set to the total number of examples in the training dataset;

• Stochastic Gradient Descent: Batch size is set to one;

• Minibatch Gradient Descent: Batch size is set to more than one and less than the total number of examples in the training dataset.

The effects of hyperparameters differs from dataset to dataset and model to model, therefore there is no optimal number. However, the batch size impacts how fast and stable the model learning is [54] [55] and should be optimized by the research given the case in which the model will be used. Using a large batch size can lead to a faster training time because of GPU speed up from parallelism but it is known that larger batch sizes lead to a poor generalization. When using smaller batch sizes the neural network model goes through a sort of “tug-and-pull” dynamic by constantly adjusting the weights in favor and against the specific details or ”noise” contained inside each small sample size of the whole dataset which avoids overfitting. This is one of the reasons vaguely attributed to explain the better generalization of the model learning when lowering the batch size. It is also generally accepted that stochastic gradient descent performs poorly therefore the optimal number is somewhere between 1 and the entire training dataset. The overfitting and poor generalization meaning the model relies too much on some niche specific characteristics of the training dataset to boost its accuracy, this reliance on such characteristics ends up making the model perform badly on the test dataset where such patterns are no longer reliable to make good classifications. In the case of epistasis detection this would mean the model would be capable of considering noisy SNPs important because in the samples provided by the dataset these noisy SNPs are able to create a correlation with the disease. These spurious correlations disappearing if the number of samples was increased since the distribution of SNPs values would be better represented.

A batch size of 1 would make the training of the proposed model be too slow while using a higher batch size made the training more time efficient. We experimented with a batch size of 1,10, 16, 32 and 64. Although a batch size of 10 is not commonly used, it provided a better recall and consistent results than all the other values which is why it was selected as the optimal value.

The learning rate controls how strong are the updates applied to the weights of the neural network. With a learning rate of 0.0001 the weights are not updated in a fast enough manner to allow the model to efficiently converge and learn. While a higher learning rate than 0.01 makes the model present too much of a divergent behavior, meaning the model would not consistently be able to converge and learn the epistatic interactions due to the updates of the model weights being too drastic.

Given the reasons mentioned above we tested the proposed model with a batch size of 10 and a learning rate of 0.01 and 0.001. We also tested the version of the model with no final hidden dense layers, meaning that after obtaining the elementwise sum of SNP vectors, there will be a single dense layer connected to the output node function. During this testing, the single hidden final dense layer, when present in the architecture, used a Relu activation function and the attention layer a softmax activation function. The results are available in the appendix at table Table A.1. The results for each dataset setting are represented according to the standard deviation threshold selected, where ”Std dev 2” implies the filtering stage only selected SNPs with the attention values 2 standard deviations above the mean. The

”Standard chi square filtering” and ”Including the conditional testing” columns represent the results when applying only the standard chi square test or when also using the conditional test respectively, which details were explained in Section 3.2. For each standard deviation threshold the results have 2 columns which represent the precision and recall of the model respectively. The most important results being the precision and recall of the proposed model when using a standard deviation of 2 and conditional testing since it is the setting which guarantees the model is being as precise as possible, these specific results being represented in Table 4.3 from which was concluded that a final dense layer between the output layer and the summation of the SNPs vectors is necessary to achieve a good detection power and overall the learning rate of 0.01 provided an increase in recall over the alternative learning rate of 0.001.

Table 4.3: Batch size and learning rate optimization results

Model	Batch	Layer	Learning rate	Including conditional testing, Std dev 2	
				Precision	Recall
1	Batch size 10	True	0.001	1	0.84
			0.01	1	0.82
		False	0.001	1	0.01
			0.01	0	0
21	Batch size 10	True	0.001	1	0.57
			0.01	1	0.73
		False	0.001	0	0
			0.01	0	0
26	Batch size 10	True	0.001	1	0.38
			0.01	0.98	0.51
		False	0.001	0	0
			0.01	0	0
6	Batch size 10	True	0.001	1	0.32
			0.01	1	0.62
		False	0.001	0	0
			0.01	0	0
11	Batch size 10	True	0.001	0	0
			0.01	0.9	0.09
		False	0.001	0	0
			0.01	0	0

4.3.2 Attention layer, final dense layers number and activation functions

Having already established the batch size of 10 and a learning rate of 0.01 as the optimal parameters as well as the use of final dense layers as necessary, we then proceeded to experiment with a different number of final dense layers and different activation functions in the attention layer. In this second phase of the proposed model optimization the final dense layers activation function was changed from Relu to LeakyRelu given that the latter is an improved version of the Relu which solves the vanishing gradient problem and resulted in a better detection power of the proposed model.

The results from Table 4.4 compared to the results from Table 4.3 show how changing the activation function of the attention layer from a softmax to a LeakyRelu results in a big improvement for the detection power of the proposed model in Model 1, Model 21, Model 26 and Model 6 datasets, only not

achieving 100% recall in model 6.

Table 4.4: Attention layer with a LeakyRelu activation results

Parameters = Learning rate 0.01, Batch size 10, Single final dense layer, Attention layer LeakyReLU activation					
Models	Iterations	Avg runs per dataset	Std dev 2 Recall	Std dev 2 Avg Candidates	Total Runs Allowed
Model 1	20	1.66	1	2	30
Model 21	20	2.05	0.97	2.22	30
	50	2.29	1	2.29	30
Model 26	20	4.23	0.89	2.75	30
	50	2.33	0.98	2.64	30
	100	2.0	1.0	2.44	30
Model 6	20	3.25	0.81	2.975	30
	50	2.48	0.96	3.05	30
	100	2.5	0.99	2.87	30

The proposed model was also tested with using 2 final dense layers in combination with a LeakyRelu, Relu and Mish activation function in the attention layer in Model 6 and Model 26. From these results it was concluded two final dense layers provided an increase in detection power and that the LeakyRelu function not only achieved 100% recall in all dataset models, but it also kept the average number of initializations required to achieve those levels of recall lower than the Mish and Relu activation functions shown by the difference in the values of the “Avg runs per dataset” column in Table 4.5 which contains the results from Model 6 datasets.

4.3.3 Batch normalization

The final parameters optimization was done on Model 11 and Model 20 having a lower heritability which makes the detection of the epistatic interactions more difficult. To improve the detection power of the proposed model we experimented the use of batch normalization on the summation of the SNPs vector, on the final dense layers and on both at the same time.

Batch normalization is used to stabilize the activation values outputs of a layer by normalizing them

Table 4.5: 2 final dense layers and different attention layer activation functions results

Parameters = Learning rate 0.01, Batch size 10, Two single final mlp layer with Mish activation in attention layer				
Models	Iterations	Avg runs per dataset	Std dev 2 Recall	Std dev 2 Avg Candidates
Model 6	50	**3.55**	1.00	3.29
Parameters = Learning rate 0.01, Batch size 10, Two single final mlp layer with Relu activation in attention layer				
Models	Iterations	Avg runs per dataset	Std dev 2 Recall	Std dev 2 Avg Candidates
Model 6	50	**4.5**	0.93	3.28
Parameters = Learning rate 0.01, Batch size 10, Two final mlp layer with LeakyReLU activation in attention layer				
Models	Iterations	Avg runs per dataset	Std dev 2 Recall	Std dev 2 Avg Candidates
Model 6	50	**2.4**	1.00	3.07

taking into account the distribution of values of a batch of samples. In this way the shift in values between different samples is stabilized into a distribution which makes the model converge faster and makes it less sensitive to different weight initializations.

The batch normalization was not used in the attention layer since the importance of the SNPs are not using the samples values as input as explained in section 3. We also included in this experiment the use of a linear activation function in the attention layer, meaning the values are not transformed, which is something not commonly used for dense layers in standard neural networks but since the attention layer is not dependent on input values, and rather the weights are the outputs of the layer themselves, a linear function should work as well.

With the use of batch normalization on both the summation of the SNPs vector and final dense layers and the removal of the LeakyRelu activation function in the attention layer, the proposed model had an increase from 28% to 98% recall in Model 11 datasets shown in Table 4.6 while having the highest recall of 92% when limited to a total of 30 different weight initializations. The final conclusion was made based on the best performance of the proposed model on the Model 20 datasets which was achieved with those same parameters. Based on the results the best version of the model parameters is represented in Table 4.7. This version of model was also tested on the rest of the datasets from Model 1, Model 21, Model 26 and Model 6 to guarantee the improvement in performance was consistent in all scenarios. The final dense layers each used a total of 32 neurons which is a commonly used size for dense layers and the SNP vectors created using the embedding layer also had a dimension of 32 values which makes the elementwise sum of them also have 32 neurons which is the input for the final dense layers .

Table 4.6: Batch normalization and attention layer with a linear activation results

Dataset setting	Model 11		Model 20	
Parameters	Std dev 2 Recall	Total Runs Allowed	Std dev 2 Recall	Total Runs Allowed
No norm	0.28	30	-	-
LeakyReLU, Norm on SNP vector	0.87	30	-	-
	0.99	100	0.65	100
LeakyReLU, Norm on SNP vector and final dense layers	0.79	30	0.86	30
	0.98	100	0.98	100
Linear activation, Norm on SNP vector	0.87	30	-	-
	0.98	100	0.72	100
Linear activation, Norm on SNP vector and final dense layers	0.92	30	0.87	30
	0.98	100	**0.99**	100

4.3.4 Proposed model training details and experimental setup

The model was coded in Python and built using the Tensorflow and Keras libraries which is one of the most common ways of implementing deep learning models.

The model is trained using 50 iterations with the Adam optimizer and reports SNPs which achieve an attention value above 2 standard deviations from the mean to guarantee the findings from the model

Table 4.7: Best Hyperparameters

Best model hyperparameters	
Attention layer activation function	Linear
Final dense layers activation function	LeakyReLU
Nº of final dense layers	2
Batch normalization on summation vector	On
Batch normalization on final dense layers	On
Learning rate	0.01
Batch size	10

are significative. Thus resulted in the model outputting less than 3% of the total number of SNPs most of the time which is important to reduce the amount of exhaustive testing performed in the filtering stage. Which can be observed by the average candidates proposed by the model for all the tested datasets which is made available in the appendix in Table A.2. The table also includes the average number of weights initializations used for each type of dataset, which when high means the model is having a hard time to converge and has become more sensitive to the initialization which occurs when the difficulty given by the parameters of the datasets also increases. All the details of these datasets and analysis of the results will be given in the next section.

The model is run with different initializations until a significative epistatic interaction of the desired order is found using the statistical testing of the filtering stage or until it reaches a maximum of 100 initializations which was chosen empirically to impose a necessary limit. For the filtering stage, the p-values of the standard χ^2 test and conditional test used as thresholds are 2e-5 and 1e-4 respectively. These values were obtained after testing multiple simulated datasets and observing the corresponding p-value to determine the p-value threshold which distinguishes true epistatic interactions from noisy SNPs.

4.4 Experimental evaluation

To evaluate and compare the proposed model to state of the art methods we rely on the results reported in the research works [74] and [16]. Besides the results from the state-of-the-art methods reported in the above mentioned works we also provide the results from a deep learning model from [75] to compare against the proposed model, which will be referred as DL2 throughout this section.

The deep learning model DL2 layers details are represented in Table 4.8. The model used the "SGD" optimizer for training with a momentum of 0.5, a batch size of 32, a learning rate of 0.01 and 200 epochs/iterations.

Table 4.8: Deep learning model layers for model DL2

Layers	Neurons	Activation	Dropout rate	Weight regularization
Dense layer 1	64	Relu	0.1	l1 and l2 norm with 1e-5
Dense layer 2	32	LeakyRelu	0.1	l1 and l2 norm with 1e-5

The implementation of the deep learning model from the research work [75] from 2020 used the information from the state of the art MLP configurations for epistasis detection from the resarch works [76] and [77]. The researchers optimized the selection of the hyperparameters considering the following values: algorithm = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam'], batch size = [32, 64, 128, 256], epochs = [50, 100, 200, 500, 1000], learning rate = [0.001, 0.01, 0.1, 0.2, 0.3], weight decay = [0.00001, 0.0001, 0.001, 0.01], dropout rate = [0.1, 0.2, 0.3, 0.4], units = [8, 16, 32, 64, 128], and layers = [1, 2, 3]. The configuration with the highest prediction accuracy was selected resulting in the following parameters: optimization algorithm = 'SGD', batch size = 32, epochs = 200, learning rate = 0.01, weight decay = 0.00001, dropout rate = [0.1, 0.1], units = [64, 32] and hidden layers = 2. The nonlinear activation function for the first hidden layer was the rectifier linear unit ("ReLU") and for the second hidden layer it was "softReLU" as reported in the paper. We changed the second hidden layer activation function for a "LeakyRelu" since it is more commonly used and verified that it increased the performance of the model in all the simulated datasets it was experimented on. The momentum hyperparameter was considered as 0.5. This model will also be run using different initializations of weights and also following the exact same filtering stage methods to do a fair comparison to the proposed model. In this case, the importance of the SNPs is calculated based on the average of the absolute gradients of the SNPs over all samples.

4.4.1 Marginal datasets of 3rd and 4th order interactions

For testing the performance of the proposed model for the detection of epistasis in marginal effect datasets we relied on the datasets produced from penetrance tables of third and fourth-order interactions, with MAF values of 0.10, 0.25 and 0.40 and heritability of 0.10, 0.25, 0.50 and 0.80 which were downloaded from the repository made available by the authors of [74]. Each dataset has a total of 500 SNPs and 2000 samples with a single, 3rd or 4th order, epistatic interaction. A total of 55 settings were tested which details are made available in the appendix of thesis in Figure A.1. The model is evaluated based on the recall metric as this was the metric used to evaluate the state-of-the-art methods in [74].

Before proceeding to the analysis of the results, a brief description of the type of methods reported in [74] will be given below which is important to understand the conclusions made in Section 4.4.4. The 28 methods are grouped into 6 groups, these being Exhaustive, Filtering, Depth First, Swarm Intelligent, Genetic Algorithms (GA) and Random-search-based methods (RS). MDR [8] and MPI3SNP [78] are exhaustive methods which apply an exhaustive statistical testing of all possible combinations of SNPs. The filtering methods discard a large number of SNPs or combinations of SNPs to reduce the computational burden. The simplest approach, used in EpiMiner [79] and Mendel [80], is to filter the individual SNPs of the dataset before exploring any type of combinations and interactions. HiSeeker [81] and MECPM [82] enumerate all possible combinations of 2 SNPs and filter a group of candidates, upon which, additional

SNPs are iteratively added to find higher order interactions. DCHE [83], EDCF [84] and SingleMI [85] use clustering techniques to filter combinations of SNPs starting from pairwise combinations up to a selected order. In short, besides EpiMiner and Mendel, the filtering methods create all the pair-wise SNPs combinations, then filter them according to a chosen evaluation metric which measures the correlation of the combination with the disease and then extend the possible order of interaction by iteratively testing the inclusion of additional SNPs into the promising combinations.

Depth-first search methods such as FDHE-IW [86], LRMW [87], BADTrees [88], StepPLR [89] and SNPRuler [90] iteratively add SNPs to an initially empty set while maximizing some measurement until convergence is detected. This process is repeated until a maximum number of combinations is reached or no further significative combinations are found. For example, FDHE-IW starts with an empty set and iteratively adds the SNP that maximizes the Symmetrical Uncertainty until a maximum number of SNPs is reached while LRMW, BADTrees use decision trees to model the candidate interactions.

Inside the Swarm Intelligent approaches, Ant colony optimization is the most explored metaheuristic in epistasis detection, and as already mentioned, it iteratively explores different combinations of SNPs and evaluates such combinations according to the association with the phenotype. In each iteration the SNPs which form the new set of combinations are selected according to a probability function based on the evaluations performed in the previous iterations. MACOED [16], IACO [91], epiACO [92] and HiSeeker [81] (HiSeeker is a filtering method which can be coupled with an ACO algorithm) all use this approach only with different ways to measure the association with phenotype while AntMiner [93] and EACO [94] innovate by incorporating a heuristic into the probability function. CINOEDV [95] implements the Particle Swarm Optimization (PSO) algorithm and NHSA-DHSC [96] method implements the Niche Harmony Search Algorithm. The details of how Genetic Algorithms work are in section Section 2.3.2.

Finally, random-search-based algorithms such as SNPHarvester [97], BEAM3 [98] and BHIT [99] stochastically searchs possible combinations for several iterations, evaluating and saving each solution. SNPHarvester explores multiple combinations by doing local searchs at random points of the combination space. Then follows a swapping technique, testing for all SNPs if any replacement in the combination can improve the 2 association value until no more replacements can be made. BEAM3 and BHIT explore the search space using Monte Carlo Markov Chain (MCMC) sampling, after a number of iterations are completed, the algorithm ends, and the best model is returned.

Table 4.9 contains the recall from 28 state of the art methods as reported in [74], the recall of the proposed model as well as the recall from the deep learning model DL2 on datasets containing marginal effects following either an additive or threshold interaction model. The results of the datasets were grouped based on the type of interaction, additive or threshold, and order since presenting the result for each dataset for each method would be impractical.

For the additive, 3rd order interactions there is a total of 12 settings of datasets and each result

Table 4.9: Marginal datasets results

Recall from the marginal datasets		Additive		Threshold	
Type	Algorithm	3rd Order	4th Order	3rd Order	4th Order
Exhaustive	MDR	0.98	-	0.73	-
	MPI3SNP	1	1	1	0.82
Filtering	EpiMiner	0.85	-	0.61	-
	Mendel	0	0.12	0.13	0.07
	HiSeeker-E	0	-	1	-
	HiSeeker-ACO	0	-	1	-
	MECPM	0.87	0.76	1	0.96
	DCHE	0.47	0.38	0.99	0.79
	EDCF	1	1	0.83	0.78
	SingleMI	1	1	1	0.86
	LAMPLINK	0.87	0.51	0	0
Depth First	FDHE-IW	1	1	1	0.96
	LRMW	0.33	0.13	0.57	0.26
	BADTrees	0.99	1	1	0.96
	StepPLR	0.03	0	0.99	0.88
	SNPRuler	0	0	0	0
Swarm Intelligent	MACOED	0.34	0.01	0	0
	IACO	1	0.99	0.84	0.5
	epiACO	0.86	0.20	0.81	0.36
	AntMiner	0	0	0.03	0.03
	EACO	0.68	0.22	0.95	0.68
	CINOEDV	0.04	-	0.05	-
	NHSA-DHSC	0.92	0.22	0.89	0.05
GA	GALE	0	0	0	0
	ATHENA	0.13	0.01	0.02	0
RS	SNPHarvester	1	1	0.99	0.86
	BEAM3	0.76	0.69	0.96	0.87
	BHIT	0	0	0	0
Deep Learning	Proposed model	**1**	**1**	**1**	**0.99**
	DL2	1	1	1	0.97

represents an average from such 12 settings. The recall of each individual setting was obtained from the recall of 100 datasets. For such datasets only 8 methods were capable of achieving 100% recall, 3 out of those being the proposed model, the alternative deep learning model DL2 and the exhaustive method M13SNP. Excluding the deep learning models, only 5 out of the 26 non exhaustive methods were capable of achieving a perfect performance while 7 methods achieved a recall of zero. 11 of the methods were capable of achieving a recall above 90% and 15 above 80%.

For the threshold, 3rd order interactions a total of 12 settings of datasets were tested as well, where 9 methods were able to achieve 100%, 15 above 90%, 19 above 80% and 5 achieved a recall of zero. Some of the methods are restricted for 3rd orders interactions, their results on datasets of 4th order interactions are filled with the symbol "-" in the table of results since they are not capable of being tested for such scenarios, which includes 5 of the methods, these being HiSeeker in both its variations, MDR, EpiMiner and CINOEDV. The increase to 4th order interactions should make the methods perform worse based on the increased difficulty of detecting all 4 SNPs in the combination. In this case, for the additive datasets, the methods which achieved 100% recall in the additive 3rd order interactions remain with the same recall or with a small decrease to 99%. Out of the 7 methods which achieved a recall of zero for the 3rd order additive interactions, excluding the 2 methods from HiSeeker since they are not testable in

this scenario, all remain with a recall of zero apart from Mendel surprisingly getting an increase in recall to 12%. The remaining 13 testable methods suffered a loss in performance, MACOED and ATHENA getting a 1% recall and NHSA-DHSC dropping from 92% to 22% recall.

Finally, for the 4th order threshold interactions no method was capable of achieving a 100% recall, the closest being the proposed model at 99% followed by the other deep learning model DL2 at 97%. Only 5 methods were capable of obtaining a recall above 90% in this scenario. The methods capable of maintaining an average recall above 90% for all dataset settings were the two deep learning models and the BADTrees and FDHE-IW methods, while maintaning it above 85% is also accomplished by SNPHarvester and SingleMI.

Based on the results we conclude that the proposed model as well as the DL2 model surpassed the performance of all methods in these marginal effect datasets with 3rd and 4th order interactions, the proposed model having the best performance. Although the proposed model does not implement a stochastic search which could make it miss an interaction, the randomness nature of the initialization of the weights values made it not able to find a good initialization after reaching the maximum number of tries for some of the datasets in the 4th order threshold interactions datasets dropping the average recall to 99% in that scenario.

4.4.2 2nd order interaction in the absence of marginal effects

The next testing of the proposed model was done with datasets from [16] where they used an ant colony optimization algorithm with a filtering stage. The filtering stage used a standard χ^2 test which filters combinations based on a p-value of 2e-5. We also adopted this value in our filtering stage to only select significative pair of SNPs proposed by the model and to do a fair comparison.

From the paper MACOED [16] there are a total of 40 different settings for non marginal datasets which can be organized into 8 different types. The 8 different types come from the 8 possible combinations of the range of values for heritability and MAF these being 0.4, 0.2, 0.05, 0.01 and 0.4, 0.2 respectively. We selected 6 settings each belonging to a different type of datasets with the details available in Table 4.2. The first 4, Model 1, Model 6, Model 21 and Model 26 are the easier settings of datasets out the 6 selected for epistasis detection, having a higher heritability value of 0.4 and 0.2 with the MAF values of 0.2 and 0.4. Model 11 was arbitrarily selected since it is the first setting of datasets with a heritability of 0.05 and because we want to guarantee that our testing has all heritabilities values represented. Model 20 having a heritability of 0.01 was specifically picked for being the ”hardest” for epistasis detection based on the results of state of the art methods tested in [16]. As the heritability decreases, the harder it is for the model to detect the epistatic interaction since its strength and effect on the disease is much lower.

Each dataset setting is tested using 100 simulated datasets, where each dataset contains a single

pair-wise interaction, meaning that for each setting the models can detect up to a maximum of 100 interactions, except for Model 20 datasets the reason being mentioned below. To showcase the maximum detection power the model can achieve, we make available for each dataset setting the total number of epistatic SNP pairs that achieve a p-value below 2e-5 represented in Table 4.10, all other pairs are impossible to be detected since they will be filtered out according to such threshold. The generation of the simulated datasets with low heritability can fail to produce significative pairs according to the chi square test due to the sample size, this problem is mentioned and explored further in Section 4.4.3. The average heritability was also measured and represented in Table 4.10 since there is some difference between the theoretical parameter introduced in the simulator and the real values obtained in the generated dataset. All the datasets were downloaded from the repository made available by the authors of [16]. Each dataset has a total of 1600 samples and 100 SNPs. The results are available on Table 4.11 where the averaged recall, precision and F-score of 100 datasets of each dataset setting are represented by the respective 3 columns of values. Method 1 through 5 denote Stage I of MACOED (same as MACOED without the filtering stage), MACOED, AntEpiSeeker [15], BEAM [13] and BOOST respectively which were the state of the art methods tested in the paper. MACOED and AntEpiSeeker both using ant colony optimization algorithms, BEAM being a Bayesian network-based algorithm using Markov Chain Monte Carlo to search for interactions which would classify him as a random-search-based algorithm and BOOST being an exhaustive approach [9].

Table 4.10: MACOED non marginal datasets significative pairs

Model	Heritability	Number of significative pairs (2e-5)
1	0.43	100
21	0.4	100
26	0.25	100
6	0.2	100
11	0.053	100
20	0.016	13

Table 4.11: MACOED non marginal datasets

	DNME Model 1			DNME Model 6			DNME Model 21			DNME Model 26			DNME Model 11			DNME Model 20		
Method	h2=0.4,MAF=0.2			h2=0.2,MAF=0.2			h2=0.4,MAF=0.4			h2=0.2,MAF=0.4			h2=0.05,MAF=0.2			h2=0.01,MAF=0.2		
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.95	0.98	0.47	0.03	0.05
2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.12	0.75	0.21
3	0.94	0.94	0.94	0.93	0.93	0.93	0.88	0.97	0.92	0.94	0.96	0.95	1.00	0.96	0.98	0.10	0.71	0.18
4	0.05	0.24	0.08	0.03	0.19	0.05	0.01	0.14	0.02	0.00	0.00	0.00	0.05	0.36	0.09	0.00	0.00	0.00
5	1.00	0.67	0.80	1.00	0.60	0.75	1.00	0.63	0.77	1.00	0.60	0.75	1.00	0.65	0.78	0.41	0.41	0.41
Proposed model	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.98**	**1.00**	**00.99**	**0.09**	**1.00**	**0.16**
DL2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.61	1.00	0.76	0.02	1.00	0.04

For the first 4 dataset settings (Model 1, Model 6, Model 21 and Model 26) MACOED, the proposed model and DL2 maintain a recall and precision of 100% while AntEpiSeeker was capable of maintaining it above 88%, BEAM being the only one with a poor performance. The method BOOST was capable of detecting all interactions as expected since it is an exhaustive method but has a lower precision since it uses a less stringent chi square test with only 4 degrees of freedom which makes the method detect

false interactions, compared to the 8 degrees of freedom used in MACOED and in the proposed model filtering stage.

For the Model 11 datasets MACOED maintains a perfect recall and precision while the proposed model loses 2% recall. The reason for this, as already previously mentioned, has to do with the initialization of the weight values and as the difficulty of the dataset increases the more sensitive the model becomes, not being able to converge even though the difficulty of these 2 datasets is similar to rest of the 98 datasets. It should be mentioned that if instead we decided to always use the same initialization of values after verifying its improved performance the results would not be valid. This is because the weights would create a bias to always work on datasets where the 2 first SNPs of the input sample share an epistatic interaction. Still, the proposed model tests the same initialization of values for all datasets which means there is not a specific initialization that works well in all the datasets of a specific dataset setting, since the difference in sample values also affects the training. DL2 had a decrease in recall compared to the previous dataset settings only achieving a recall of 61%.

In the results of Model 20 it is a completely different scenario. All methods suffer a big loss of recall since the dataset has an extremely low heritability of only 1% which is directly correlated with the strength of the epistatic interaction, but also because as shown before in Table 4.10 only 13 out of the 100 datasets contain a significative interaction according to the p-value threshold of 2e-5. BOOST is able to achieve a recall of 41% for the same reason it did not achieve perfect precision in the other dataset settings, meaning that the lesser stringent chi square test allowed 41 out of the 100 interactions to be considered significative enough.

Based on the results from table 4.11 it is concluded that the proposed model can detect the epistatic interaction without having to rely on any marginal effects at least when the interactions are significative enough and continues to outperform the DL2 model. The testing of Model 20 is repeated in the next section where all datasets contain significative interactions.

4.4.3 Sample size effects on non marginal effect datasets

2nd order non marginal effect epistasis datasets from MACOED

The reason datasets from the Model 20 only had 13 out of 100 significative interactions is because the chi square test is affected by the number of samples. This detail is even more relevant in the context of model 20 datasets since most of the correlation of the SNPs with the disease is attributed to the combinations of the genotype with the presence of minor alleles which represent a very small portion of the dataset as represented in Table 4.12.

We generated datasets using the same Model 20 dataset settings but with an increase of the sample size to 6400 (3200 cases, 3200 controls). The model was able to achieve 99% recall in these datasets

Table 4.12: Probability of each genotype combination between a pair of SNPs with an MAF of 0.2

Probability of each SNPs combination	AA	Aa	aa
BB	0.4096	0.2048	0.0256
Bb	0.2048	0.1024	0.0128
bb	0.0256	0.0128	0.0016

compared to the 9% recall from before, even though the epistatic interaction pattern represented in the data is the same. The recall being higher is expected because the interactions are now able to pass the p-value threshold of 2e-5.

We further explore the effect of the number of samples on the detection power of the model on datasets where the problem of the interactions being significative according to a certain threshold is not relevant anymore. Model 11 from MACOED already had all datasets with significative pairs and a recall of 98% for the proposed model but we still tested the impact of increasing the number of samples. This led to the increase of the recall to 100%, but it also accelerated the process of finding the epistatic interactions tremendously. With a sample size of 1600 to achieve 98% recall the average number of different initializations needed per dataset to find the epistatic interaction was 11, while with 6400 samples a mere 1.8 was needed. The results are represented in table 4.13 where the three columns for each dataset setting represent the recall, precision and F-measure/F-score of the proposed model respectively. This confirms the already well-known fact that deep learning models training benefits from an increase in the number of samples.

Table 4.13: 4x increase in sample size impact

	DNME Model 11 6400 samples			DNME Model 20 6400 samples		
Method	h2=0.05, MAF=0.2			h2=0.01, MAF=0.2		
Proposed model	1	1	1	0.99	1	1

3rd order interactions without marginal effects

For testing the performance of the proposed model in 3rd and 4th order interactions without the presence of marginal effects and comparing it to state of the art methods, we relied once again on datasets used and results reported in [74], which details are available in the appendix at Figure A.1(c). A total of 7 dataset settings with 3rd order interactions and 4 dataset settings with 4th order interactions were tested, the datasets remain with 500 SNPs and 2000 samples. The results represented in Table 4.14 were grouped based on the order of interactions where each dataset setting was tested using the average recall from 100 datasets. All of the non exhaustive methods including the proposed model obtained a poor performance, the highest being DCHE and SNPRuler with 24% and 13% recall respectively.

Table 4.14: Non marginal datasets reults

Recall from the marginal datasets		Non marginal	
Type	Algorithm	3rd Order	4th Order
Exhaustive	MDR	1	-
	MPI3SNP	1	1
Filtering	EpiMiner	0	-
	Mendel	0	0
	HiSeeker-E	0.01	-
	HiSeeker-ACO	0.01	-
	MECPM	0	0
	DCHE	0.24	0
	EDCF	0.08	0
	SingleMI	0	0
	LAMPLINK	0	0
Depth First	FDHE-IW	0	0
	LRMW	0	0
	BADTrees	0	0
	StepPLR	0	0
	SNPRuler	0.13	0
Swarm Intelligent	MACOED	0	0
	IACO	0	0
	epiACO	0.01	0
	AntMiner	0	0
	EACO	0	0
	CINOEDV	0.02	-
	NHSA-DHSC	0.02	0
GA	GALE	0	0
	ATHENA	0	0
RS	SNPHarvester	0	0
	BEAM3	0	0
	BHIT	0	0
Deep Learning	Proposed model	**0**	**0**

Table 4.15: Top performing methods from the 3rd and 4th order marginal effects interactions datasets

Minimum recall maintained across all settings	Methods
90%	FDHE-IW, BADTrees, Proposed model, DL2
85%	SNPHarvester, SingleMI
80%	MPI3SNP
75%	MECPM, EDCF

Given the results obtained from the increase in samples on the datasets from MACOED, 2 of highest heritability dataset settings which contain 3rd order non marginal effect interactions were selected from the total 7 to repeat the same experiment. In this case the datasets were generated with 8000 controls and 8000 cases with the number of SNPs reduced to 100 instead of the 500 in order to check if the proposed model is able to detect the interactions in such conditions since a lower number of features reduces noise and reduces the difficulty for epistasis detection. We also include the recall from 2 of the best performing methods in the marginal datasets according to Table 4.9 which are also represented in Table 4.15, these being FDHE-IW and SNPHarvester.

The results are represented in Table 4.16. As can be observed, the proposed model was able to achieve a great performance of 99% recall and 90% recall in the datasets with a heritability of 0.8 and 0.5 respectively. The other state of the art methods tested failed to achieve any recall at all. The deep learning model DL2 shows also a recall of zero which makes us suggest that the representation of the

Table 4.16: 3rd order non marginal datasets

Method	Dataset	Samples	SNPs	Recall
Proposed model	3rd Order, MAF=0.4, h2 =0.8	16000	100	0.99
Proposed model	3rd Order, MAF=0.4, h2 =0.5	16000	100	0.90
DL2	3rd Order, MAF=0.4, h2 =0.8	16000	100	0
DL2	3rd Order, MAF=0.4, h2 =0.5	16000	100	0
SNP Harvester	3rd Order, MAF=0.4, h2 =0.8	16000	100	0
SNP Harvester	3rd Order, MAF=0.4, h2 =0.5	16000	100	0
FDHE-IW	3rd Order, MAF=0.4, h2 =0.8	16000	100	0
FDHE-IW	3rd Order, MAF=0.4, h2 =0.5	16000	100	0

SNPs as vectors and the weighted sum of such vectors according to the proposed model architecture is the reason for the difference in performance since besides that detail both deep learning models are very similar using 2 dense layers with Relu or LeakyRelu activations. We conclude that the increase of the number of samples makes more evident the presence of the epistatic interaction for the proposed model. We can also conclude, at least based on the results shown, that the proposed model was the best overall non exhaustive method in terms of detection power when considering both the performance in the marginal datasets in which the next best 5 methods were DL2, BADTrees, FDHE-IW, SNPHarvester and SingleMI and also the fact that DL2, SNPHarvester and FDHE-IW only achieved a recall of zero while the proposed model was capable of achieving 99% recall in the same 3rd non marginal effects interactions datasets. And although not all the other state of the art methods were able to be tested with the increase in sample size, the next section will pinpoint the reason for their lack of performance in the non marginal effects datasets based on how the algorithms search for interactions and give the theoretical rationale behind the superior performance of deep learning models for epistasis detection.

One additional experiment was also performed to verify if the model would be capable of detecting the interactions while maintaining the total number of 500 SNPs. The experiment was done on the non marginal effect datasets of 3rd order interactions with an MAF of 0.4 and h^2 of 0.8 with 100 thousand samples per dataset. The model was able to achieve a perfect recall but required an average of 30 plus different initializations which confirms as expected that the increase in the number of features also greatly affects the detection power of the proposed model.

4.4.4 State of the art methods problem in the absence of marginal effects

The filtering methods EpiMiner and Mendel, as mentioned before, filter the individual SNPs before any type of combinations. In the case of non marginal effects datasets, the SNPs which share an epistatic interaction do not contain any independent effects, therefore this type of method will have a poor performance since the correlation can only be observed when the SNPs are combined . The rest of the filtering methods mentioned start with an initial exhaustive stage, where all possible pair-wise combinations are filtered based on their measured association with the phenotype and then follows an iterative

testing where additional SNPs are included in the filtered combinations. The depth-first search methods start instead with an empty set, and then do the iterative testing of adding SNPs while maximizing some metric which measures the correlation with the phenotype, repeating this process to form and explore various combinations.

Swarm intelligent based methods start from a set of random combinations which are evaluated according to the correlation with the phenotype, and then iteratively form new combinations. SNPs which were present in combinations associated with the phenotype have a higher chance of being selected to form these new combinations. Finally, random-search-algorithms stochastically search for random combinations of SNPs, for example using Monte Carlo Markov Chain (MCMC) sampling while SNPHarvester after exploring random combinations then swaps SNPs in these combinations to measure if it results in an increase in the association with the phenotype.

Regardless of their differences or specific details it seems that the reason these methods fail in non marginal effect datasets of higher order is the same. The reason being, that for example, in the case of a 3rd non marginal effect interaction if the SNPs only share a significative interaction when all 3 SNPs are considered in a combination there is no way to guide the search.

In depth-first search algorithms, starting from an empty set, they suffer the same problem as decision trees which is that the first SNPs of the set selected for maximizing some evaluation metrics require marginal effects, otherwise, no correlation is possibly measured. Even in the case of the filtering methods where the first exhaustive step is already considering all possible 2-way interactions, if the combinations of 2 of the SNPs contained in the 3rd order epistatic interaction are tested and they do not share a significative pair-wise interaction, the methods will fail since again no correlation can be measured and the filtering process will not be effective. Swarm intelligence based methods suffer the same problem since they cannot effectively apply the heuristic search for guiding the formation of new combinations since the only possible combination which has correlation with the disease is the solution itself which renders the method as good as a random search.

The statements made above do not mean that all these state of the art methods cannot be effective in finding non marginal effects interactions, but rather point out that the methods are only useful when the epistatic interaction of higher order already includes significative lower order epistatic interactions with the same SNPs (nested epistasis), which the non exhaustive search methods will be able to detect and such information be used to help guide the search for the higher order interaction.

The distinction between a pure (non marginal effects) strict and nested epistasis is represented in Figure 4.2 where the white circles represent SNPs without marginal effects while the darkened circles represent SNPs with marginal effects and SNPs connected by edges share epistatic effects. Non marginal effects datasets only contain cases of pure epistasis while the distinction between strict or nested epistasis is due to the absence or presence of lower order interactions inside the higher order

interaction. It is also possible to have nested epistasis but the interactive effects of the lower order interactions not be significative enough depending on the specified threshold which would result in the methods performing as in the case of a strict epistasis.

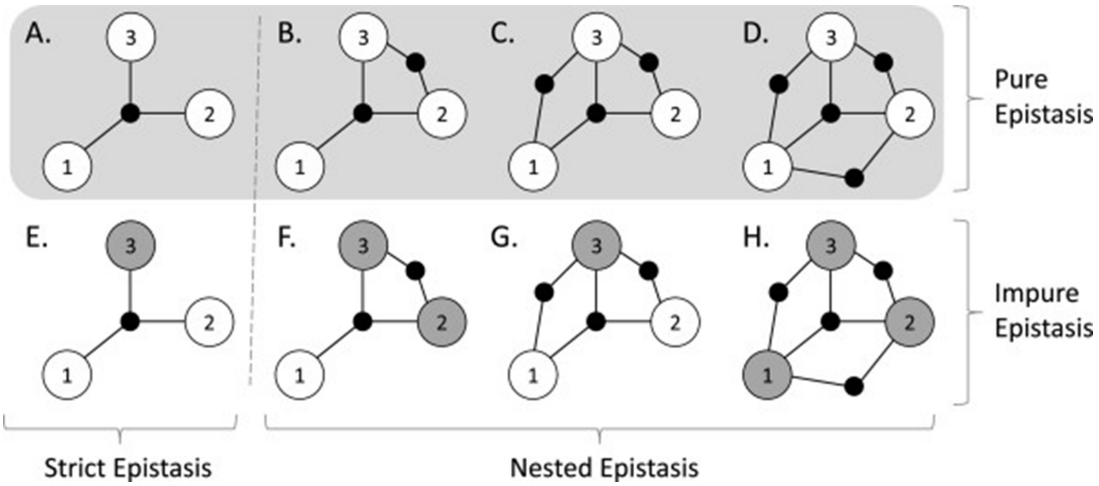


Figure 4.2: Different types of epistasis [32]

In short, the only way to detect a strict pure interaction is to measure the correlation with the phenotype including all the relevant SNPs which is only accomplished by using an exhaustive search or deep learning models which try to model the correlation with the phenotype of all the possible combinations of SNPs of any order at the same time. For this reason, the proposed model has an advantage in the case of pure strict epistasis and also in the case of nested epistasis where the lower order interactions have weak effects.

4.5 Summary

The proposed model tries to address the problem mentioned above which was experimentally observed by achieving 99% recall while 2 of the top performing methods in the marginal effect datasets achieved a recall of 0% in the same non marginal effect datasets. The proposed model also achieved the best performance in marginal effects datasets. Also, based on the results, it achieved a better epistasis detection power than the alternative deep learning model DL2 which suggests the architecture of the proposed model provides a better performance than the standard neural networks architecture. However, the proposed model still achieved a poor performance based on the non marginal effect results from the datasets with 2000 samples and 500 SNPs, one of its major flaws being the sensitivity to the initialization values of the weights and the need for an increased number of samples to detect the interactions. Another problem of the proposed model is that the number of samples necessary for the detection of epistasis also increases as the number of SNPs considered increases which is an already established

fact that it affects the performance of machine learning methods include deep learning models.

5

Conclusion

Contents

5.1 Conclusions

The analysis of the relationship between phenotype (e.g., diseases) and genotype requires the inclusion of the effect that a SNP only presents when another SNP is also present in the individual genotype, this phenomenon being called Epistasis. To detect these effects, the approach used must measure the association of the SNPs combinations with the phenotype. The problem of epistasis detection is difficult because of the number of genes combinations that need to be tested when considering interactions of higher order, which makes the use of exhaustive methods undesirable. The alternative state of the art non exhaustive approaches make use of stochastic, heuristics and machine learning methods for the detection of epistasis. We point out that to find the higher order interactions, the state of the art methods normally detect the independent effects of the SNPs, or when not possible, at least the presence of pair-wise interactions which are associated with the phenotype to use as a starting point to form the higher order combination. Therefore, if the SNPs included in the higher order interaction do not present significative lower order interactions there is no way for the methods to use heuristics and guide the search in a non-exhaustive way.

The main goal of this thesis was developing an alternative to the current state of the art non exhaustive methods for higher order epistasis detection using a deep learning neural network (DNN) model. DNNs are able to model the combinations of all SNPs of any given order at the same time which addresses the problem mentioned above. The proposed model innovates the approach of applying DNNs to epistasis detection by making use of an embedding layer to represent each SNP value as a vector and an “attention layer” to learn the importance of each SNP based on its association with the phenotype.

The model also keeps the number of parameters low, even though each SNP value is now projected into a vector with 32 values. This is due to all the SNP vectors being summed elementwise together into a single vector while simply applying standard dense layers to all the SNP vectors would result in a very high number of weights. This single vector is used as input to 2 dense layers which use this information discriminate between cases and controls,

Before summing the SNP vectors, each is scaled by the importance given to it by the attention layer which allows a filtering of the relevant SNPs and noisy SNPs information, the more relevant SNPs getting a much higher importance. These SNP importance values learned by the attention layer of the proposed model are used to filter the initially large number of SNPs to a much smaller set, which based on the results was maintained below 3% of the total number of SNPs most of the time. Upon this filtered set exhaustive statistical testing is applied to find significative epistatic interactions. By using the attention layer importance scores as a feature attribution method, the method does not require any additional computation after training to calculate the SNPs importance.

The proposed model was able to surpass the performance of all the non exhaustive state of the art methods, and the alternative deep learning model that was also experimented, on the marginal datasets

tested. The experimental results also showed the proposed model was capable of achieving 99% recall in non marginal effect datasets of 3rd order interactions while the alternative deep learning model and two of the top performing state of the art non exhaustive methods in marginal datasets achieved a recall of 0%.

To conclude, the proposed model is a promising approach which tries to address some the problems in the current state of the art and presents a novel architecture which was able to outperform the state of the art standard neural network used for SNPs epistasis detection. The fact that genotype information availability and accessibility are increasing, and that most of the major flaws of the proposed approach can be ameliorated by the increase in sample size, suggests that deep learning models will become an even more promising approach for epistasis detection.

5.2 System Limitations and Future Work

Of the possible future research works is to couple the proposed model with another model, where, the proposal could be used as an initial filtering stage for example in an exhaustive method such as MPI3SNP which would probably be the easiest first step to implement and improve our solution.

As of now we limited the use of the model as a filtering method based on the importance scores of the SNPs, but a model could be developed which does not only provide us with the importance of each SNP but as well the exact interactions that are modeled inside the neural network similar to the concept of the Transformer input-input relationships.

Another improvement would be to restrict the use of the filtering stage which conducts exhaustive statistical on the candidates SNPs provided by the attention vector only when the weights/attention values show the model was able to converge. This convergence can be noticed by the deviation in the attention values, meaning, the presence of high and low attention values instead of all attention values remaining in the same range which indicates the model was unable to learn.

Bibliography

[1] "The genomic data challenges of the future," https://medicalfuturist.com/the-genomic-data-challenges-of-the-future/.

[2] J. N. Hirschhorn *et al.*, "Genomewide association studies–illuminating biologic pathways," *New England Journal of Medicine*, vol. 360, no. 17, p. 1699, 2009.

[3] T. A. Manolio, F. S. Collins, N. J. Cox, D. B. Goldstein, L. A. Hindorff, D. J. Hunter, M. I. McCarthy, E. M. Ramos, L. R. Cardon, A. Chakravarti *et al.*, "Finding the missing heritability of complex diseases," *Nature*, vol. 461, no. 7265, pp. 747–753, 2009.

[4] B. Maher, "The case of the missing heritability: when scientists opened up the human genome, they expected to find the genetic components of common traits and diseases. but they were nowhere to be seen. brendan maher shines a light on six places where the missing loot could be stashed away," *Nature*, vol. 456, no. 7218, pp. 18–22, 2008.

[5] E. E. Eichler, J. Flint, G. Gibson, A. Kong, S. M. Leal, J. H. Moore, and J. H. Nadeau, "Missing heritability and strategies for finding the underlying causes of complex disease," *Nature Reviews Genetics*, vol. 11, no. 6, pp. 446–450, 2010.

[6] T. D. Howard, G. H. Koppelman, J. Xu, S. L. Zheng, D. S. Postma, D. A. Meyers, and E. R. Bleecker, "Gene-gene interaction in asthma: Il4ra and il13 in a dutch population with asthma," *The American Journal of Human Genetics*, vol. 70, no. 1, pp. 230–236, 2002.

[7] Y. Cho, M. Ritchie, J. Moore, J. Park, K.-U. Lee, H. Shin, H. Lee, and K. Park, "Multifactor-dimensionality reduction shows a two-locus interaction associated with type 2 diabetes mellitus," *Diabetologia*, vol. 47, no. 3, pp. 549–554, 2004.

[8] M. D. Ritchie, L. W. Hahn, N. Roodi, L. R. Bailey, W. D. Dupont, F. F. Parl, and J. H. Moore, "Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer," *The American Journal of Human Genetics*, vol. 69, no. 1, pp. 138–147, 2001.

[9] X. Wan, C. Yang, Q. Yang, H. Xue, X. Fan, N. L. Tang, and W. Yu, "Boost: A fast approach to detecting gene-gene interactions in genome-wide case-control studies," *The American Journal of Human Genetics*, vol. 87, no. 3, pp. 325–340, 2010.

[10] R. Jiang, W. Tang, X. Wu, and W. Fu, "A random forest approach to the detection of epistatic interactions in case-control studies," *BMC bioinformatics*, vol. 10, no. 1, pp. 1–12, 2009.

[11] D. F. Schwarz, I. R. König, and A. Ziegler, "On safari to random jungle: a fast implementation of random forests for high-dimensional data," *Bioinformatics*, vol. 26, no. 14, pp. 1752–1758, 2010.

[12] C. Wei, D. J. Schaid, and Q. Lu, "Trees assembling mann-whitney approach for detecting genome-wide joint association among low-marginal-effect loci," *Genetic epidemiology*, vol. 37, no. 1, pp. 84–91, 2013.

[13] Y. Zhang and J. S. Liu, "Bayesian inference of epistatic interactions in case-control studies," *Nature genetics*, vol. 39, no. 9, pp. 1167–1173, 2007.

[14] C. Niel, C. Sinoquet, C. Dina, and G. Rocheleau, "Smmb: a stochastic markov blanket framework strategy for epistasis detection in gwas," *Bioinformatics*, vol. 34, no. 16, pp. 2773–2780, 2018.

[15] Y. Wang, X. Liu, K. Robbins, and R. Rekaya, "Antepiseeker: detecting epistatic interactions for case-control studies using a two-stage ant colony optimization algorithm," *BMC research notes*, vol. 3, no. 1, pp. 1–8, 2010.

[16] P.-J. Jing and H.-B. Shen, "Macoed: a multi-objective ant colony optimization algorithm for snp epistasis detection in genome-wide association studies," *Bioinformatics*, vol. 31, no. 5, pp. 634–641, 2015.

[17] J. H. Moore and D. P. Hill, "Epistasis analysis using artificial intelligence," in *Epistasis*. Springer, 2015, pp. 327–346.

[18] P. Greenside, T. Shimko, P. Fordyce, and A. Kundaje, "Discovering epistatic feature interactions from neural network models of regulatory dna sequences," *Bioinformatics*, vol. 34, no. 17, pp. i629–i637, 2018.

[19] H. Wang, T. Yue, J. Yang, W. Wu, and E. P. Xing, "Deep mixed model for marginal epistasis detection and population stratification correction in genome-wide association studies," *BMC bioinformatics*, vol. 20, no. 23, pp. 1–11, 2019.

[20] Y. Liu, D. Wang, F. He, J. Wang, T. Joshi, and D. Xu, "Phenotype prediction and genome-wide association study using deep convolutional neural network of soybean," *Frontiers in genetics*, p. 1091, 2019.

[21] S. Uppu, A. Krishna, and R. P. Gopalan, "A deep learning approach to detect snp interactions." *J. Softw.*, vol. 11, no. 10, pp. 965–975, 2016.

[22] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning–based sequence model," *Nature methods*, vol. 12, no. 10, pp. 931–934, 2015.

[23] C. A. C. Montaez, P. Fergus, A. C. Montaez, A. Hussain, D. Al-Jumeily, and C. Chalmers, "Deep learning classification of polygenic obesity using genome wide association study snps," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[24] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[25] "Single-nucleotide polymorphism," https://isogg.org/wiki/Single-nucleotide_polymorphism.

[26] W. Bateson and G. Mendel, *Mendel's principles of heredity*. Courier Corporation, 2013.

[27] J. Siemiatycki and D. C. Thomas, "Biological models and statistical interactions: an example from multistage carcinogenesis," *International journal of epidemiology*, vol. 10, no. 4, pp. 383–387, 1981.

[28] R. A. Fisher, "Xv.—the correlation between relatives on the supposition of mendelian inheritance." *Earth and Environmental Science Transactions of the Royal Society of Edinburgh*, vol. 52, no. 2, pp. 399–433, 1919.

[29] C. Niel, C. Sinoquet, C. Dina, and G. Rocheleau, "A survey about methods dedicated to epistasis detection," *Frontiers in genetics*, vol. 6, p. 285, 2015.

[30] J. Marchini, P. Donnelly, and L. R. Cardon, "Genome-wide strategies for detecting multiple loci that influence complex diseases," *Nature genetics*, vol. 37, no. 4, pp. 413–417, 2005.

[31] "Hardy–weinberg principle," https://en.wikipedia.org/wiki/Hardy%E2%80%93Weinberg_principle.

[32] R. J. Urbanowicz, J. Kiralis, N. A. Sinnott-Armstrong, T. Heberling, J. M. Fisher, and J. H. Moore, "Gametes: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures," *BioData mining*, vol. 5, no. 1, pp. 1–14, 2012.

[33] C. Ponte-Fernández, J. González-Domínguez, A. Carvajal-Rodríguez, and M. J. Martín, "Toxo: a library for calculating penetrance tables of high-order epistasis models," *BMC bioinformatics*, vol. 21, no. 1, pp. 1–9, 2020.

[34] S. Purcell, B. Neale, K. Todd-Brown, L. Thomas, M. A. Ferreira, D. Bender, J. Maller, P. Sklar, P. I. De Bakker, M. J. Daly *et al.*, "Plink: a tool set for whole-genome association and population-based linkage analyses," *The American journal of human genetics*, vol. 81, no. 3, pp. 559–575, 2007.

[35] J. H. Moore, J. C. Gilbert, C.-T. Tsai, F.-T. Chiang, T. Holden, N. Barney, and B. C. White, "A flexible computational framework for detecting, characterizing, and interpreting statistical patterns of epistasis in genetic studies of human disease susceptibility," *Journal of theoretical biology*, vol. 241, no. 2, pp. 252–261, 2006.

[36] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine learning*, vol. 53, no. 1, pp. 23–69, 2003.

[37] S. Kerrien, B. Aranda, L. Breuza, A. Bridge, F. Broackes-Carter, C. Chen, M. Duesbury, M. Dumousseau, M. Feuermann, U. Hinz, C. Jandrasits, R. C. Jimenez, J. Khadake, U. Mahadevan, P. Masson, I. Pedruzzi, E. Pfeiffenberger, P. Porras, A. Raghunath, B. Roechert, S. Orchard, and H. Hermjakob, "The IntAct molecular interaction database in 2012," *Nucleic Acids Research*, vol. 40, no. D1, pp. D841–D846, 11 2011. [Online]. Available: https://doi.org/10.1093/nar/gkr1088

[38] A. Chatr-aryamontri, B.-J. Breitkreutz, R. Oughtred, L. Boucher, S. Heinicke, D. Chen, C. Stark, A. Breitkreutz, N. Kolas, L. O'Donnell, T. Reguly, J. Nixon, L. Ramage, A. Winter, A. Sellam, C. Chang, J. Hirschman, C. Theesfeld, J. Rust, M. S. Livstone, K. Dolinski, and M. Tyers, "The BioGRID interaction database: 2015 update," *Nucleic Acids Research*, vol. 43, no. D1, pp. D470–D478, 11 2014. [Online]. Available: https://doi.org/10.1093/nar/gku1204

[39] A. Franceschini, D. Szklarczyk, S. Frankild, M. Kuhn, M. Simonovic, A. Roth, J. Lin, P. Minguez, P. Bork, C. von Mering, and L. J. Jensen, "STRING v9.1: protein-protein interaction networks, with increased coverage and integration," *Nucleic Acids Research*, vol. 41, no. D1, pp. D808–D815, 11 2012. [Online]. Available: https://doi.org/10.1093/nar/gks1094

[40] E. L. Willighagen, A. Waagmeester, O. Spjuth, P. Ansell, A. J. Williams, V. Tkachenko, J. Hastings, B. Chen, and D. J. Wild, "The chembl database as linked open data," *Journal of cheminformatics*, vol. 5, no. 1, pp. 1–12, 2013.

[41] B. Han, M. Park, and X.-w. Chen, "A markov blanket-based method for detecting causal snps in gwas," in *BMC bioinformatics*, vol. 11, no. 3. Springer, 2010, pp. 1–8.

[42] Y. Guo, Z. Zhong, C. Yang, J. Hu, Y. Jiang, Z. Liang, H. Gao, and J. Liu, "Epi-gtbn: an approach of epistasis mining based on genetic tabu algorithm and bayesian network," *BMC bioinformatics*, vol. 20, no. 1, pp. 1–18, 2019.

[43] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[44] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[45] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. ARTICLE, pp. 2493–2537, 2011.

[46] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[47] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[48] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," *Briefings in Bioinformatics*, vol. 18, no. 5, pp. 851–869, 07 2016. [Online]. Available: https://doi.org/10.1093/bib/bbw068

[49] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[50] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[51] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[52] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," *Advances in neural information processing systems*, vol. 26, 2013.

[53] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 160–167. [Online]. Available: https://doi.org/10.1145/1390156.1390177

[54] S. Saha, "A comprehensive guide to convolutional neural networks — the eli5 way."

[55] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[56] A. Karpathy, “The unreasonable effectiveness of recurrent neural networks.”

[57] A. T. Mohan and D. V. Gaitonde, “A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks,” *arXiv preprint arXiv:1804.09269*, 2018.

[58] C. Molnar, *Interpretable Machine Learning*, 2nd ed., 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book

[59] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, “Explainable ai: A review of machine learning interpretability methods,” *Entropy*, vol. 23, no. 1, p. 18, 2020.

[60] D. Alvarez Melis and T. Jaakkola, “Towards robust interpretability with self-explaining neural networks,” *Advances in neural information processing systems*, vol. 31, 2018.

[61] M. T. Ribeiro, S. Singh, and C. Guestrin, “”why should i trust you?”: Explaining the predictions of any classifier,” 2016. [Online]. Available: https://arxiv.org/abs/1602.04938

[62] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[63] A. Binder, G. Montavon, S. Bach, K.-R. Müller, and W. Samek, “Layer-wise relevance propagation for neural networks with local renormalization layers,” 2016. [Online]. Available: https://arxiv.org/abs/1604.00825

[64] A. Shrikumar, P. Greenside, and A. Kundaje, “Learning important features through propagating activation differences,” 2017. [Online]. Available: https://arxiv.org/abs/1704.02685

[65] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, “Smoothgrad: removing noise by adding noise,” *arXiv preprint arXiv:1706.03825*, 2017.

[66] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, “Explaining nonlinear classification decisions with deep taylor decomposition,” *Pattern recognition*, vol. 65, pp. 211–222, 2017.

[67] A. Ghorbani, A. Abid, and J. Zou, “Interpretation of neural networks is fragile,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 3681–3688.

[68] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

[69] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[70] J. H. McDonald, “G-test of goodness-of-fit,” *Handbook of biological statistics*, vol. 487, pp. 53–58, 2014.

[71] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.

[72] D. Masters and C. Luschi, “Revisiting small batch training for deep neural networks,” *CoRR*, vol. abs/1804.07612, 2018. [Online]. Available: http://arxiv.org/abs/1804.07612

[73] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” *CoRR*, vol. abs/1206.5533, 2012. [Online]. Available: http://arxiv.org/abs/1206.5533

[74] C. Ponte-Fernández, J. González-Domínguez, A. Carvajal-Rodríguez, and M. J. Martín, “Evaluation of existing methods for high-order epistasis detection,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 19, no. 2, pp. 912–926, 2022.

[75] R. Abdollahi-Arpanahi, D. Gianola, and F. Peñagaricano, “Deep learning versus parametric and ensemble methods for genomic prediction of complex phenotypes,” *Genetics Selection Evolution*, vol. 52, no. 1, pp. 1–15, 2020.

[76] P. Waldmann, “Approximate bayesian neural networks in genomic prediction,” *Genetics Selection Evolution*, vol. 50, no. 1, pp. 1–9, 2018.

[77] M. Pérez-Enciso and L. M. Zingaretti, “A guide on deep learning for complex trait genomic prediction,” *Genes*, vol. 10, no. 7, p. 553, 2019.

[78] C. Ponte-Fernández, J. González-Domínguez, and M. J. Martín, “Fast search of third-order epistatic interactions on cpu and gpu clusters,” *The International Journal of High Performance Computing Applications*, vol. 34, no. 1, pp. 20–29, 2020.

[79] J. Shang, J. Zhang, Y. Sun, and Y. Zhang, “Epiminer: a three-stage co-information based method for detecting and visualizing epistatic interactions,” *Digital Signal Processing*, vol. 24, pp. 1–13, 2014.

[80] T. T. Wu, Y. F. Chen, T. Hastie, E. Sobel, and K. Lange, “Genome-wide association analysis by lasso penalized logistic regression,” *Bioinformatics*, vol. 25, no. 6, pp. 714–721, 2009.

[81] J. Liu, G. Yu, Y. Jiang, and J. Wang, “Hiseeker: detecting high-order snp interactions based on pairwise snp combinations,” *Genes*, vol. 8, no. 6, p. 153, 2017.

[82] D. J. Miller, Y. Zhang, G. Yu, Y. Liu, L. Chen, C. D. Langefeld, D. Herrington, and Y. Wang, “An algorithm for learning maximum entropy probability models of disease risk that efficiently searches

and sparingly encodes multilocus genomic interactions," *Bioinformatics*, vol. 25, no. 19, pp. 2478–2485, 2009.

[83] X. Guo, Y. Meng, N. Yu, and Y. Pan, "Cloud computing for detecting high-order genome-wide epistatic interaction via dynamic clustering," *BMC bioinformatics*, vol. 15, no. 1, pp. 1–16, 2014.

[84] M. Xie, J. Li, and T. Jiang, "Detecting genome-wide epistases based on the clustering of relatively frequent items," *Bioinformatics*, vol. 28, no. 1, pp. 5–12, 2012.

[85] D. Jünger, C. Hundt, J. G. Domínguez, and B. Schmidt, "Speed and accuracy improvement of higher-order epistasis detection on cuda-enabled gpus," *Cluster Computing*, vol. 20, no. 3, pp. 1899–1908, 2017.

[86] S. Tuo, "Fdhe-iw: A fast approach for detecting high-order epistasis in genome-wide case-control studies," *Genes*, vol. 9, no. 9, p. 435, 2018.

[87] C. Wei and Q. Lu, "Gwggi: software for genome-wide gene-gene interaction analysis," *BMC genetics*, vol. 15, no. 1, pp. 1–6, 2014.

[88] R. T. Guy, P. Santago, and C. D. Langefeld, "Bootstrap aggregating of alternating decision trees to detect sets of snp s that associate with disease," *Genetic epidemiology*, vol. 36, no. 2, pp. 99–106, 2012.

[89] M. Y. Park and T. Hastie, "Penalized logistic regression for detecting gene interactions," *Biostatistics*, vol. 9, no. 1, pp. 30–50, 2008.

[90] X. Wan, C. Yang, Q. Yang, H. Xue, N. L. Tang, and W. Yu, "Predictive rule inference for epistatic interaction detection in genome-wide association studies," *Bioinformatics*, vol. 26, no. 1, pp. 30–37, 2010.

[91] Y. Sun, J. Shang, J. Liu, and S. Li, "An improved ant colony optimization algorithm for the detection of snp-snp interactions," in *International Conference on Intelligent Computing*. Springer, 2016, pp. 21–32.

[92] Y. Sun, J. Shang, J.-X. Liu, S. Li, and C.-H. Zheng, "epiaco-a method for identifying epistasis based on ant colony optimization algorithm," *BioData mining*, vol. 10, no. 1, pp. 1–17, 2017.

[93] J. Shang, J. Zhang, X. Lei, Y. Zhang, and B. Chen, "Incorporating heuristic information into ant colony optimization for epistasis detection," *Genes & Genomics*, vol. 34, no. 3, pp. 321–327, 2012.

[94] Y. Sun, X. Wang, J. Shang, J.-X. Liu, C.-H. Zheng, and X. Lei, "Introducing heuristic information into ant colony optimization algorithm for identifying epistasis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 17, no. 4, pp. 1253–1261, 2018.

[95] J. Shang, Y. Sun, J.-X. Liu, J. Xia, J. Zhang, and C.-H. Zheng, "Cinoedv: A co-information based method for detecting and visualizing n-order epistatic interactions," *BMC bioinformatics*, vol. 17, no. 1, pp. 1–15, 2016.

[96] S. Tuo, J. Zhang, X. Yuan, Z. He, Y. Liu, and Z. Liu, "Niche harmony search algorithm for detecting complex disease associated high-order snp combinations," *Scientific reports*, vol. 7, no. 1, pp. 1–18, 2017.

[97] C. Yang, Z. He, X. Wan, Q. Yang, H. Xue, and W. Yu, "Snpharvester: a filtering-based approach for detecting epistatic interactions in genome-wide association studies," *Bioinformatics*, vol. 25, no. 4, pp. 504–511, 2009.

[98] Y. Zhang, "A novel bayesian graphical model for genome-wide multi-snp association mapping," *Genetic epidemiology*, vol. 36, no. 1, pp. 36–47, 2012.

[99] J. Wang, T. Joshi, B. Valliyodan, H. Shi, Y. Liang, H. T. Nguyen, J. Zhang, and D. Xu, "A bayesian model for detection of high-order interactions among genetic variants in genome-wide association studies," *Bmc Genomics*, vol. 16, no. 1, pp. 1–20, 2015.

Appendix

Table A.1: Batch and learning rate optimization

Model	Batch	Layer	Learning rate	Standard chi square filtering						Including the conditional testing					
				Std dev 2		Std dev 1.5		Std dev 1		Std dev 2		Std dev 1.5		Std dev 1	
1	Batch size 10	True	0.001	0.99	0.84	0.98	0.86	0.96	0.88	1	0.84	1	0.86	1.0	0.88
			0.01	0.99	0.82	0.99	0.82	0.99	0.83	1	0.82	1	0.82	1.0	0.83
		False	0.001	0.5	0.01	0.25	0.01	0.33	0.02	1	0.01	1	0.01	1.0	0.02
			0.01	0	0	0.33	0.01	0.38	0.03	0	0	1	0.01	1.0	0.03
21	Batch size 10	True	0.001	0.98	0.57	0.95	0.59	0.94	0.63	1	0.57	0.97	0.59	0.95	0.63
			0.01	0.99	0.73	0.97	0.73	0.96	0.73	1	0.73	0.99	0.73	0.97	0.73
		False	0.001	0	0	0	0	0.6	0.03	0	0	0	0	0.75	0.03
			0.01	0	0	0.5	0.02	0.71	0.05	0	0	0.66	0.02	0.83	0.05
26	Batch size 10	True	0.001	0.97	0.38	0.93	0.41	0.92	0.47	1	0.38	0.98	0.41	0.96	0.47
			0.01	0.96	0.51	0.94	0.58	0.94	0.61	0.98	0.51	0.97	0.58	0.97	0.61
		False	0.001	0	0	0.25	0.01	0.5	0.04	0	0	0.5	0.01	0.67	0.04
			0.01	0	0	0.57	0.04	0.5	0.05	0	0	0.8	0.04	0.63	0.05
6	Batch size 10	True	0.001	0.94	0.32	0.91	0.44	0.93	0.5	1	0.32	0.98	0.44	0.98	0.5
			0.01	0.95	0.62	0.94	0.62	0.93	0.66	1	0.62	0.98	0.62	0.97	0.66
		False	0.001	0	0	0	0			0	0	0	0	0.5	0.02
			0.01	0	0	0.25	0.01	0.38	0.03	0	0	1	0.01	0.6	0.03
11	Batch size 10	True	0.001	0	0	0.42	0.03	0.64	0.09	0	0	0.5	0.03	0.75	0.09
			0.01	0.8	0.09	0.75	0.12	0.82	0.19	0.9	0.09	0.85	0.12	0.9	0.19
		False	0.001	0	0	0	0	0.16	0.01	0	0	0	0	0.25	0.01
			0.01	0	0	0.16	0.01	0.38	0.03	0	0	0.25	0.01	0.5	0.03

Order	MAF	P(D)	h^2
3	0.10	0.000012	0.10
3	0.10	0.000004	0.25
3	0.10	0.000002	0.50
3	0.10	0.000001	0.80
3	0.25	0.005370	0.10
3	0.25	0.001153	0.25
3	0.25	0.000504	0.50
3	0.25	0.000306	0.80
3	0.40	0.254558	0.10
3	0.40	0.022186	0.25
3	0.40	0.008545	0.50
3	0.40	0.005091	0.80
4	0.25	0.000234	0.10
4	0.25	0.000068	0.25
4	0.25	0.000031	0.50
4	0.25	0.000019	0.80
4	0.40	0.036282	0.10
4	0.40	0.003383	0.25
4	0.40	0.001374	0.50
4	0.40	0.000822	0.80

(a) Additive relationship

Order	MAF	P(D)	h^2
3	0.10	0.064602	0.10
3	0.10	0.025561	0.25
3	0.10	0.013270	0.50
3	0.10	0.008417	0.80
3	0.25	0.477516	0.10
3	0.25	0.267707	0.25
3	0.25	0.154539	0.50
3	0.25	0.102529	0.80
3	0.40	0.780354	0.10
3	0.40	0.586967	0.25
3	0.40	0.415395	0.50
3	0.40	0.307526	0.80
4	0.10	0.012563	0.10
4	0.10	0.005140	0.25
4	0.10	0.002590	0.50
4	0.10	0.001623	0.80
4	0.25	0.275518	0.10
4	0.25	0.132034	0.25
4	0.25	0.070683	0.50
4	0.25	0.041819	0.80
4	0.40	0.668428	0.10
4	0.40	0.446405	0.25
4	0.40	0.287337	0.50
4	0.40	0.201273	0.80

(b) Threshold relationship

Order	MAF	P(D)	h^2
3	0.25	0.5860	0.10
3	0.25	0.4923	0.25
3	0.25	0.4223	0.50
3	0.40	0.5163	0.10
3	0.40	0.5644	0.25
3	0.40	0.5019	0.50
3	0.40	0.4970	0.80
4	0.25	0.4201	0.10
4	0.25	0.5910	0.25
4	0.40	0.4720	0.25
4	0.40	0.4356	0.10

(c) Relationship with NME

Figure A.1: Interaction Orders, Minor Allele Frequencies , Prevalence Values and Heritability Values of the Penetrance Tables used in "Evaluation of Existing Methods for High-Order Epistasis Detection"

Model 3	P(D)=0.1, h2=0.02, MAF=0.05								
	CC			Cc			cc		
	BB	Bb	bb	BB	Bb	bb	BB	Bb	bb
AA	0.0942	0.0942	0.0942	0.0942	0.0942	0.6535	0.0942	0.6535	0.0942
Aa	0.0942	0.0942	0.6535	0.0942	0.1469	0.0942	0.6535	0.0942	0.0942
aa	0.0942	0.6535	0.0942	0.6535	0.0942	0.0942	0.0942	0.0942	0.0942

Model 3	P(D)=0.1, h2=0.02, MAF=0.1								
	CC			Cc			cc		
	BB	Bb	bb	BB	Bb	bb	BB	Bb	bb
AA	0.099	0.099	0.099	0.099	0.099	0.549	0.099	0.549	0.099
Aa	0.099	0.099	0.549	0.099	0.1436	0.099	0.549	0.099	0.099
aa	0.099	0.549	0.099	0.549	0.099	0.099	0.099	0.099	0.099

Model 3	P(D)=0.1, h2=0.02, MAF=0.2								
	CC			Cc			cc		
	BB	Bb	bb	BB	Bb	bb	BB	Bb	bb
AA	0.0916	00916	0.0916	0.0916	0.0916	0.3141	0.0916	0.3141	0.0916
Aa	0.0916	0. 0916	0.3141	0.0916	0.1014	0.0916	0.3141	0.0916	0.0916
aa	0.0916	0.3141	0.0916	0.3141	0.0916	0.0916	0.0916	0.0916	0.0916

Model 3	P(D)=0.1, h2=0.02, MAF=0.5								
	CC			Cc			cc		
	BB	Bb	bb	BB	Bb	bb	BB	Bb	bb
AA	0.0842	0.0842	0.0842	0.0842	0.0842	0.213	0.0842	0.213	0.0842
Aa	0.0842	0.0842	0.213	0.0842	0.095	0.0842	0.213	0.0842	0.0842
aa	0.0842	0.213	0.0842	0.213	0.0842	0.0842	0.0842	0.0842	0.0842

Figure A.2: 3rd order datasets with no marginal effects from SMMB

Table A.2: Tested datasets performance metrics

Dataset	Parameters		Avg_runs_per_dataset	Avg_candidates	Recall
MACOED 2nd order no marginal effects	Model 1,21,26,6		1 (approximation)	2 (approximation)	1
	Model 11		11.01	3.12	0.98
	Model 20 6400 samples		11.36	3.98	0.99
3rd order Additive	MAF 0.1	h2=0.1	1	13.49	1
		h2=0.25	1	12.64	1
		h2=0.5	1	12.94	1
		h2=0.8	1	7.99	1
	MAF 0.25	h2=0.1	1	8.26	1
		h2=0.25	1	6.22	1
		h2=0.5	1	6.59	1
		h2=0.8	1	7.46	1
	MAF 0.4	h2=0.1	1	12.33	1
		h2=0.25	1	7.61	1
		h2=0.5	1	6.69	1
		h2=0.8	1	7.16	1
3rd order Threshold	MAF 0.1	h2=0.1	4.8	top 5% SNPs to accelerate search	1
		h2=0.25	1.02	11.62	1
		h2=0.5	1	9.79	1
		h2=0.8	1	8.34	1
	MAF 0.25	h2=0.1	1.59	top 5% SNPs to accelerate search	1
		h2=0.25	1	10.92	1
		h2=0.5	1	8.4	1
		h2=0.8	1	5.6	1
	MAF 0.4	h2=0.1	1	10.29	1
		h2=0.25	1	8.3	1
		h2=0.5	1	6.59	1
		h2=0.8	1	4.6	1
4rd order Additive	MAF 0.25	h2=0.1	1	7.66	1
		h2=0.25	1	7.21	1
		h2=0.5	1	7.47	1
		h2=0.8	1	8.41	1
	MAF 0.4	h2=0.1	1	10.66	1
		h2=0.25	1	7.56	1
		h2=0.5	1	8.03	1
		h2=0.8	1	8.25	1
4rd order Threshold	MAF 0.1	h2=0.1	13.35	top 5% SNPs to accelerate search	0.99
		h2=0.25	1.24	13.29	1
		h2=0.5	1	11.1	1
		h2=0.8	1	9.64	1
	MAF 0.25	h2=0.1	19.06	top 5% SNPs to accelerate search	0.83
		h2=0.25	1.06	13.33	1
		h2=0.5	1	9.95	1
		h2=0.8	1	6.72	1
	MAF 0.4	h2=0.1	1.36	13.65	1
		h2=0.25	1	10.73	1
		h2=0.5	1	7.73	1
		h2=0.8	1	5.68	1