Universiteit
Leiden
The Netherlands

Improving Genetic Association for Insect Resistance Traits in Plants Using Deep Learning on Video Tracking Data

Jordi Alonso Esteve

Dr. Willem Kruijer, WUR
Dr. Aalt-Jan van Dijk, WUR

Defended on August 22 , 2022

MASTER THESIS

STATISTICS AND DATA SCIENCE
UNIVERSITEIT LEIDEN

Abstract

High-throughput phenotyping platforms provide high-dimensional phenotypical data which, while carrying immense potential, pose a challenge to current genetic mapping methodologies. Handcrafted statistics have been used to summarize this type of data to analyze it through the genetic mapping framework. However, when dealing with highly complex traits, handcrafted summaries of the data may be sub-optimal. Automated approaches have the potential to come with optimal representations of the data towards the genetic mapping endeavour, reducing potential bias and the need for expert human input.

In this thesis, we explore Deep Learning methods to construct useful low-dimensional representations from highly complex raw phenotypic data. In doing so we aim to surpass the manual generation of traits providing a scalable and homogeneous solution to the treatment of complex phenotyping data. We study the usage of supervised and self-supervised Deep Learning approaches for trait generation and explore methodology to deal with inherently multidimensional traits. We focus on Arabidopsis resistance to aphid infestation, using video recordings of the aphid as our raw data.

Our results show that it is possible to generate equal or better representations of the raw data through an automated process using Deep Learning in terms of the heritability of the traits. We obtain the best results through contrastive supervised learning, training the model to separate the traits by genotype.

Optimal trait generation combined with approaches to deal with pleiotropy can lead to better usage of raw high dimensional data and allow to discover new meaningful relationships at a bigger scale than ever.

1 Introduction

Resistance to insect pests is an important target for plant breeders. Specifically, piercing-sucking insects, like aphids, are major vectors of plant viruses causing significant losses in crops. Understanding the functional genomics of plant resistance to this kind of infestation is, consequently, of great interest.

However, how can we study this resistance? Observing insect resistance in the field is difficult and expensive. For this reason, there is a strong interest in high-throughput phenotyping platforms that observe resistance under controlled environments. While, potentially, high-throughput phenotyping platforms are a powerful tool for genetic research, the bridge between the two is sometimes not straightforward. High-dimensional data such as images, video, trajectories, or similar formats are not directly tractable by the statistical methodology that underpins the first steps in genetic research. This is especially difficult when little is known about what the phenotype should be. We want to find which genes drive the variability in our observations but it is unclear which dimensions are relevant.

A common statistical approach for genetic mapping are **Genome Wide Association Studies** (GWAS). It aims to map the observed phenotypes with specific parts of the genotype. GWAS test for tens of thousands of genetic variations (markers) over numerous genomes in order to find the variations that could explain a particular trait. These genetic markers are, usually, Single Nucleotide Polymorphisms (SNPs). The test is typically constructed through linear models. A basic model can be defined as $Y \sim X\beta + e$ where Y is a vector of the observed trait per individual, X the matrix of markers score (frequently binary) composed of one row per individual and possibly tens of thousands of columns, β the marker effect and e the random error term. Random effects to account for genetic relatedness of the individuals or other covariates can be also included.

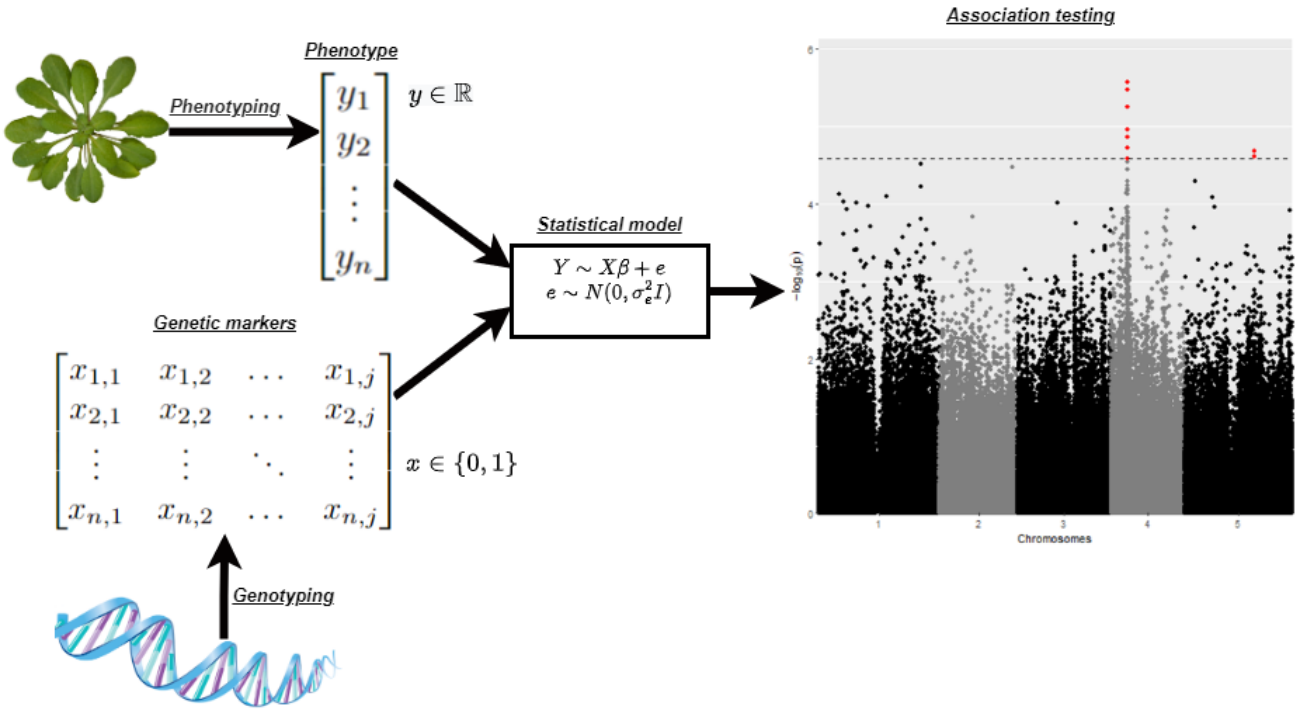


Figure 1: GWAS pipeline. A statistical model is used to assess the relationship between SNPs and the variability in a trait. The P-values derived from the marker effects β are used to test for association. Typically this association is evaluated in a Manhattan plot, displayed on the right. The Manhattan plot displays the $-\log_{10}$(P-value) on the vertical axis and the SNPs on the horizontal axis. SNPs are ordered by their position in the genome so we can observe which segments could be explained by the variability in the trait.

With the fitted model, P-values for each SNP can be obtained, allowing the study of associations throughout the whole genome.

Nevertheless, it is not possible to directly use this framework with the above-mentioned type of high-dimensional data. Some transformation must be applied to extract the most relevant factors of variation in the high-dimensional data. For instance, a black and white image of 128x128 pixels has 128^2 dimensions, however, the interesting variability of the image is likely contained in a much lower dimensional space. Consequently, an intermediate step is needed to transform the raw data into meaningful traits that can be analyzed through a GWAS. This intermediate step can be understood as a part of the phenotyping process (see top left Figure 1).

If the trait to investigate is clear, such as flowering time or size, ad hoc criteria are adequate for the extraction of the summary statistics suitable for GWAS. For instance, it is usual practice to estimate biomass from an image of a plant by reducing the number of vegetative pixels in the image to a single number, or to quantify plant height by counting the contiguous pixels in the vertical axis [13, 2, 9].

Other traits, however, are more complex to analyze. For instance, the texture of a leaf, branching complexity, shape or resistance to an insect. In these cases, multiple proxy measures have to be extracted in order to capture the inherently multidimensional phenotype. Indeed, there is an orientation of what the phenotype should look like but it is not clear which specific ad hoc proxy measure is best. Being that the case, relying on ad hoc statistics can lead to sub-optimal use of the data owing to missing possible interesting traits. Alternatively, automated trait generation from high-dimensional raw data could lead to new discoveries because some traits may be beyond the initial scope of the researchers.

In this thesis, we focus on the treatment of this type of high-dimensional output from a high-throughput phenotyping platform. Specifically, here we study the genes driving arabidopsis resistance to aphids using video recordings. *Arabidopsis thaliana* is a classical model organism which has many advantages such as short generation time, small size, a large number of offspring, a known and relatively small

nuclear genome, and being a deeply studied plant [27]. While it does not have a direct economic impact (it is not a marketable crop), research on *Arabidopsis thaliana* can help improve resistance in economically relevant crops. For instance, it allows the development of methodology that could be used with other species and traits of interest.

We use data obtained through an automated video tracking system [31]. It records the trajectory of the aphids on an arabidopsis leaf [1]. The video recordings are transcribed into a matrix with 3 columns (t, x, y) where t is time and x, y are the Cartesian coordinates representing the position of the insect on the leaf. The recorded behaviour of the insect allows the investigation of differences by genotype and ultimately study which genes of arabidopsis drive insect resistance.

Until now, this data has been studied using ad hoc handcrafted features. Biologists constructed biologically meaningful features by defining some parameters, for instance, the threshold of velocity that determines a specific type of behaviour such as moving or probing. Arguably, when the aphid is halting it means it is probing (trying to penetrate the leaf to feed itself), therefore, lower probing time may indicate arabidopsis resistance to aphid infestation (see Figure 2). The selection of parameters to construct the traits is indeed far from arbitrary and has been successful in finding relevant genes promoting resistance to aphids in arabidopsis [32, 30]. But, it is far from obvious that the traits the biologists extracted are the best possible traits. Current extracted traits have very low heritability.

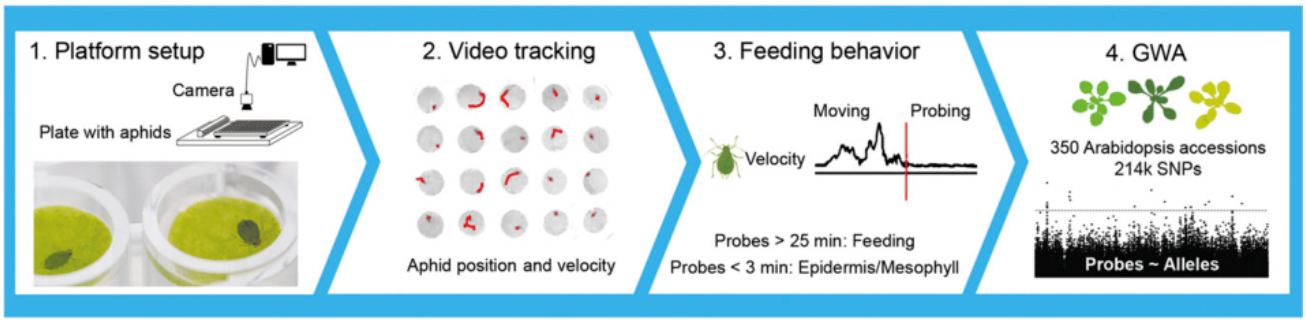


Figure 2: Phenotyping pipeline for handcrafted features. Image extracted from [32].

In this case, while there is an orientation of how the phenotype should look (halting-moving dynamics as a proxy for resistance to aphid), fine-tuning this orientation to concrete parameters is a complex task. Resistance to aphids is likely a multidimensional phenotype and there are hundreds of possible different traits to analyze. Furthermore, what is considered initially relevant may not be the most relevant kind of trait. Automating the phenotyping process allows the discovery of new promising features in a more efficient way. It is well known for fields such as computer vision that learned representations often result in better performance than could be obtained with hand-designed features and require less human input [18]. Here we study the automation of this process using deep learning (see Figure 3).

[1] A movie example of the video tracking can be downloaded here

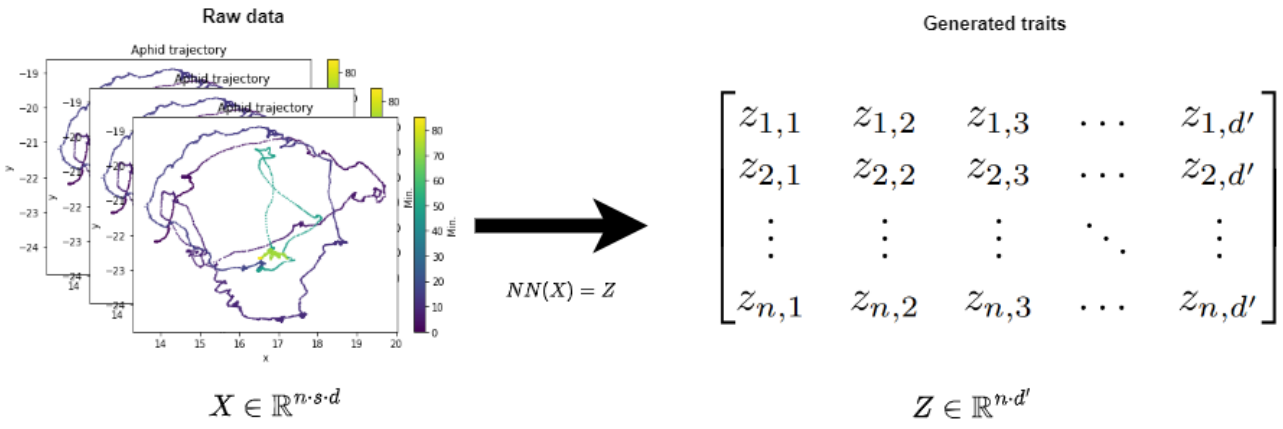


Figure 3: Phenotyping pipeline through deep learning. The original data consists of 2141 observations of 25500 steps and 3 dimensions (Cartesian coordinates and time). A deep neural network (NN) transforms this input and creates a representation with d' dimensions. This low-dimensional representation are the new traits suitable for GWAS. In doing so we expect to automatically extract the most relevant features of the data.

When dealing with inherently multidimensional phenotypes, pleiotropy becomes an important factor. Especially when dealing with traits with opaque interpretation. To deal with this, parallel to GWAS, we also propose a simple reverse mapping (treating marker score as a dependent variable) scheme. We fit a fast classifier on top of the traits and evaluate predictive performance on unseen data.

In summary, the question to address is: Is it possible to obtain better features than the ones extracted by the experts? In order to answer this, we investigate several ways of framing the data, several deep neural network architectures and several ways of understanding the problem with different objective (loss) functions. Specifically, we experiment with several architectures in a supervised learning setting through reverse mapping, contrastive learning and self-supervised learning. This first approach tries to recover the information in the data that could lead to a good predictive performance for a known relevant SNP. We also train an auto-encoder, which assumes that variability is mostly due to genetic information and therefore it tries to recover it in an unsupervised way following the idea that lossy compression can lead to capturing relevant factors of variability. Finally, we use supervised contrastive learning, using the genotype as the class, attempting to explicitly maximize the heritability (amount of variance explained by the genotype in the traits).

We, then, select the traits created by the best models. We use the handcrafted traits to evaluate the relative quality of our automatically generated ones by studying heritability and explained variance by a relevant known SNP. Finally, we explore GWAS and reverse mapping, compared our results with previous findings with this data and propose areas in the genome to investigate.

The present thesis is structured as follows: First, we describe the dataset, the cleaning process and the methodology used to extract the traits. We provide an introduction to the generation and evaluation of the data through hand-designed traits, justify the use of deep learning and detail the experimental pipeline. We discuss the GWAS framework, the implications of population structure, pleiotropy and multiple testing and propose a simple exploratory method for (reverse) genetic mapping. Then, the experimental outcomes are analyzed through the proposed framework. We assess which way of generating traits is better and propose promising areas of the genome either through GWAS or reverse mapping. Finally, we outline relevant contributions made to the topic and acknowledge the limitations of our approach.

The code to reproduce all the experiments is available at `https://github.com/neggor/-improving-genetic-association-using-deep-learning_DEMO`.

2 Materials and methods

2.1 The data

2.1.1 Phenotypic data (trajectories)

The dataset was collected and published by [31, 30, 32] in 2017. A detailed description of the data generation methodology is available in those papers. The data consist of 2141 (19 were lost) 85 minutes of recordings of an aphid on top of a small piece of an Arabidopsis leaf. A step is recorded every 0.2 seconds, which means 25500 steps per recorded arena (well plate). The recording starts immediately after the inoculation of the insects. Each recording is transcribed to Cartesian coordinates by time i.e. a three-dimensional matrix. This raw transcription is the trajectory data we work with.

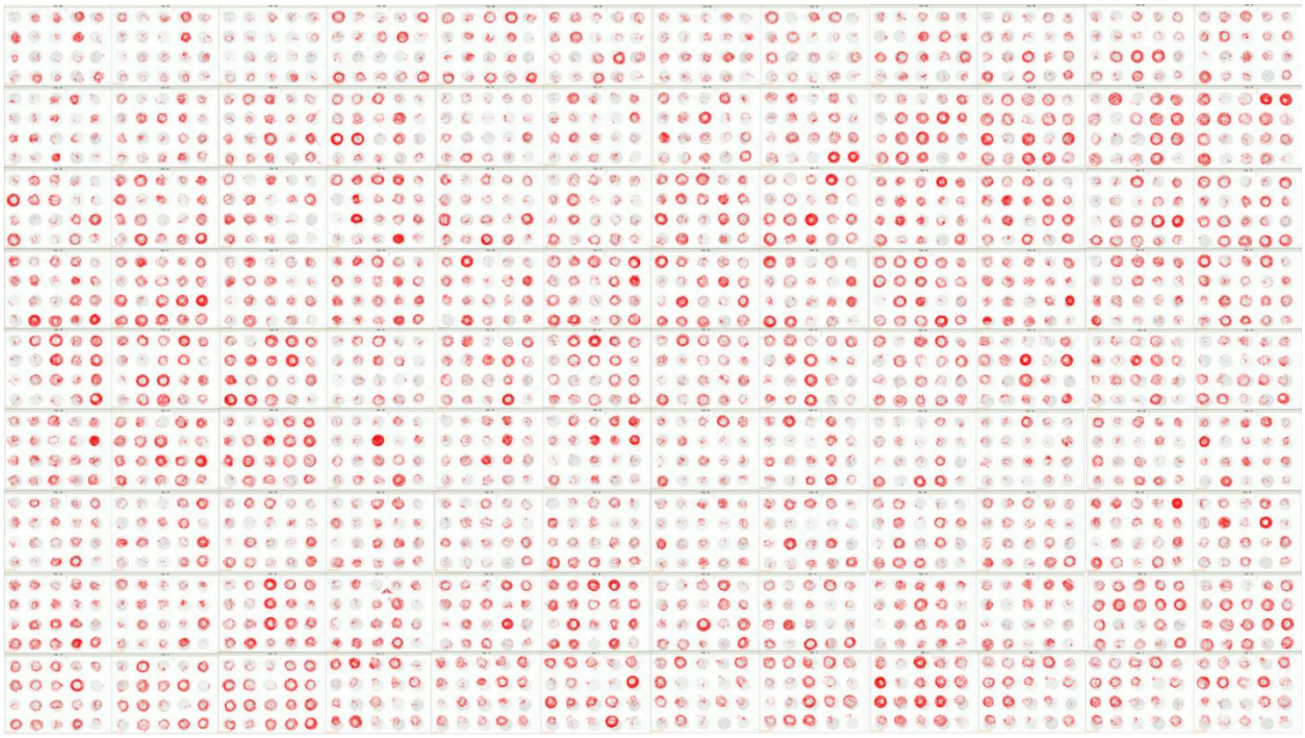


Figure 4: Videotracking traces of aphid on Arabidopsis leaf discs derived from 350 accessions in 5-6 fold replication. Structured in 108 trials of 20 arenas.

Experimental design. Aphid behaviour was tracked for 350 accessions (genotypes) of Arabidopsis. Each genotype had 5 to 6 replicates, this is what we call the **plant** level. In each **trial** 20 aphids were recorded on 20 different accessions simultaneously. The aphids were situated in a well of a 96-well plate containing a leaf disc of 5mm (see Figure 4). Figure 5 shows an example from the dataset.

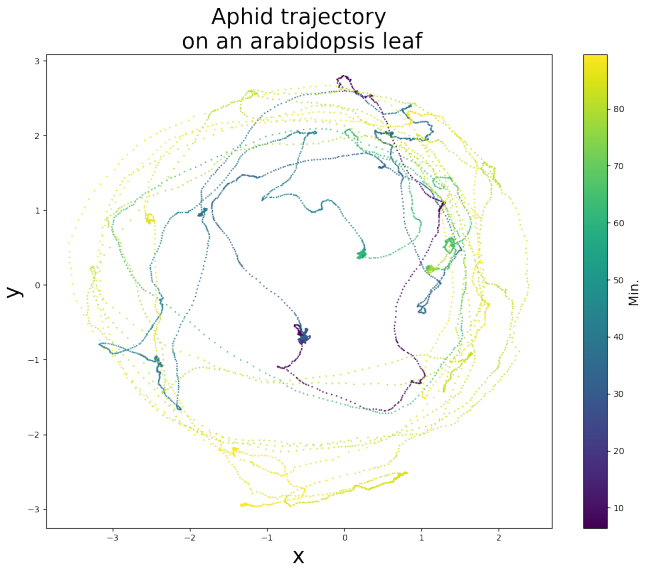


Figure 5: A trajectory example. x and y are a (standardized) measure of the aphid position in the well-plate.

2.1.2 Data wrangling[2]

In order to utilize this data in a deep learning setting it must be free of missing data. To solve this we applied the following procedures:

- Cut all the trajectories up to 25498 steps because there was some variability in the length among them.
- The last part of the trajectory is considered less relevant and missing steps were prominent at the beginning of the trajectory. This is most likely due to difficulties of the software/hardware to locate the Aphid at the beginning of the experiment. In order to solve this, we remove the beginning of the trajectories until there were 5 consecutive steps with available data. Afterwards, we remove 500 steps at the end of all arenas to maintain the length.
- Up to 150 consecutive missed steps, we impute data by distributing the points evenly in the straight line between the last and next available points. Those arenas with more than 150 consecutive steps missing were discarded. Figure 6 illustrates how the missing steps are distributed.

[2]The data wrangling was programmed in Python with extensive use of **Numpy**[19] and **Pandas** [42].

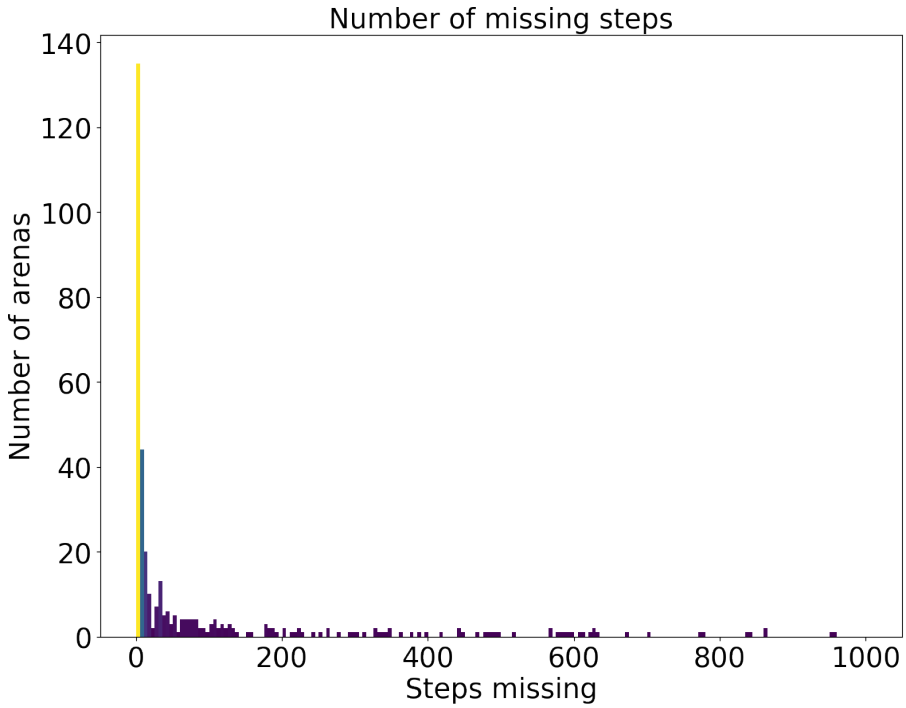


Figure 6: Histogram of the number of steps missing by arena. Most of the arenas have less than 100 steps missing and there are only 57 arenas with more than 1000 steps missing. There are a total of 424 arenas with some missing values. Those are not necessarily consecutively missing.

Outliers are considered the steps with a speed above 2mm per second. Those steps are transformed into missing data and therefore imputed.

2.1.3 Data representation

For animals that freely move on an agar plate, their absolute coordinates are probably meaningless. Therefore, we attempt to extract features that are invariant to translation and rotation. We transform the time series of the coordinates into a multidimensional time series consisting of marginal change in x and y, speed, acceleration, angular speed, travel distance, and moving averages and standard deviations of all of them with a window of 150 steps. We also include the initial dimensions time, x and y. In doing so, we try to achieve position and rotation-invariant trajectory information.

Eventually, we work with a time series of 21 dimensions. All dimensions in each arena are standardized at the arena level. Figure 7 shows how a particular arena looks like.

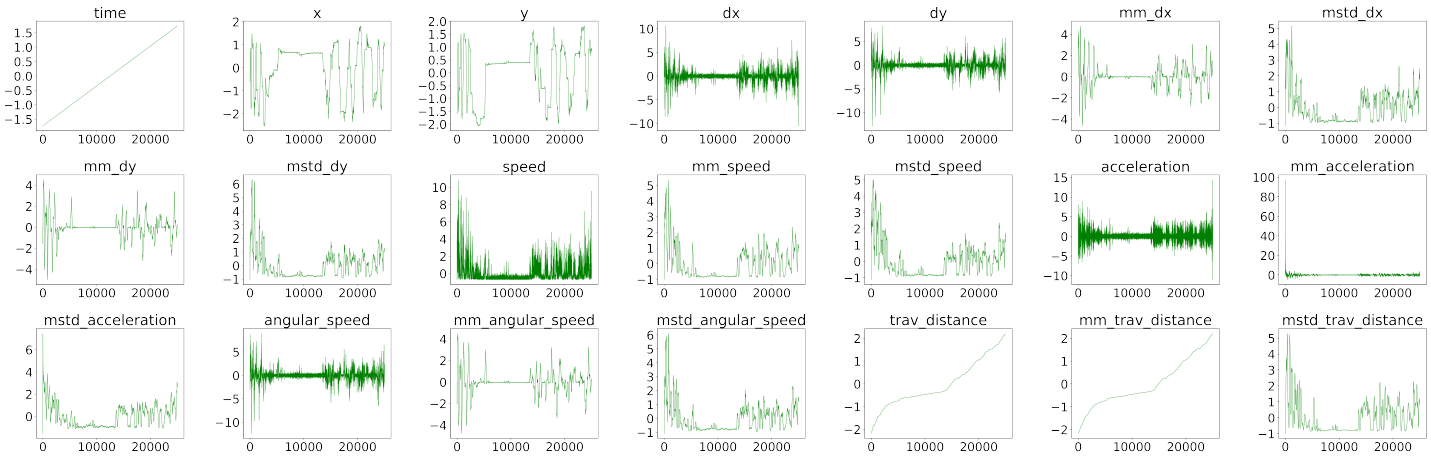


Figure 7: An example of all 21 dimensions that are used as raw input for the time series models. "mm" stands for moving mean, and "mstd" for moving standard deviation.

Additionally, we also frame the data as an image. In this case, the translation invariance relies on the convolution operation of the (2D) Convolutional Neural Network (CNN). Since the convolutional kernel is applied throughout the image, the same patterns output the same disregarding the position in the image. Rotation invariance, however, is not guaranteed, but is likely achieved through the use of several kernels with different initializations. The CNN approach has the potential of capturing aphid behaviour in a local area of the image disregarding the time aspect of the trajectory. Framing trajectory data as an image has been previously investigated, for instance, in boat classification [33].

In this case, only x, y, time and speed are used. The transformation is a simple process:

- We subtract the minimum to x and y making the series have the minimum at 0.
- Divide the chosen resolution by the maximum of x and y in order to obtain a scalar to bound the whole series between 0 and the resolution.
- Multiply x and y by the scalar and then round the result to an integer. This integer is the position of the step in the image.
- The intensity of white in each point is determined by the speed relative to the maximum in each particular arena.

Collisions are possible, however, the higher the resolution the lower the probability of collision. In case of collision, the newest point overrides the older one. The result can be seen in Figure 8. Finally, in order to use them as RGB images, the grayscale is used for red, green and blue.

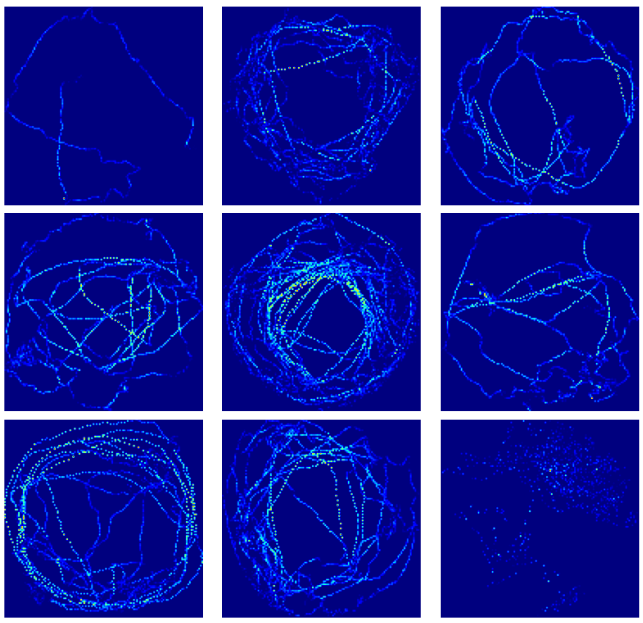


Figure 8: Some examples of trajectories as images. The colour map in this visualization is enhanced to ease visual interpretation. The actually used images are grayscale.

2.1.4 Genetic data

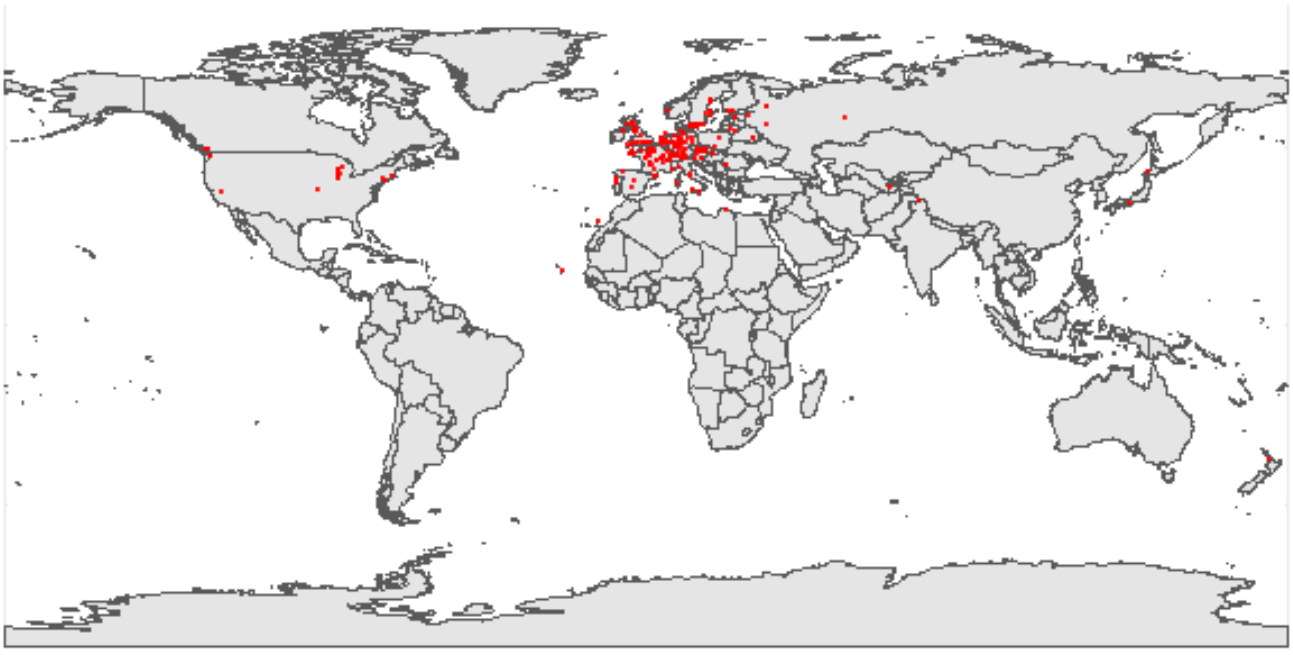


Figure 9: Geographical distribution of our arabidopsis accessions.

We work with $\sim$ 214.000 single nucleotide polymorphisms (SNPs) from 350 different *Arabidopsis thaliana* natural accessions (genetic varieties). Those accessions have been gathered from different natural populations that span various geographic and environmental ranges (Figure 9).

Due to the fact that arabidopsis is primarily a selfing (self-fertilization) species, the majority of the collected plants are from inbred lines that are practically homozygous [1, 47]. Therefore, the allele can be either present (1) or not (0) so the marker score is a binary categorical variable. The allele is constructed with respect to the Columbia (Col) inbred strain which is always 1. The genetic data, therefore, consists of a binary array of $\sim$ 214.000 columns corresponding to the different SNPs with 350

rows corresponding to the different genotypes. When we use a SNP as response variable, we are using a specific column of this matrix as the response. Naturally, each observation is mapped to its genotype value i.e., the 350 rows are used to label each arena given its genotype.

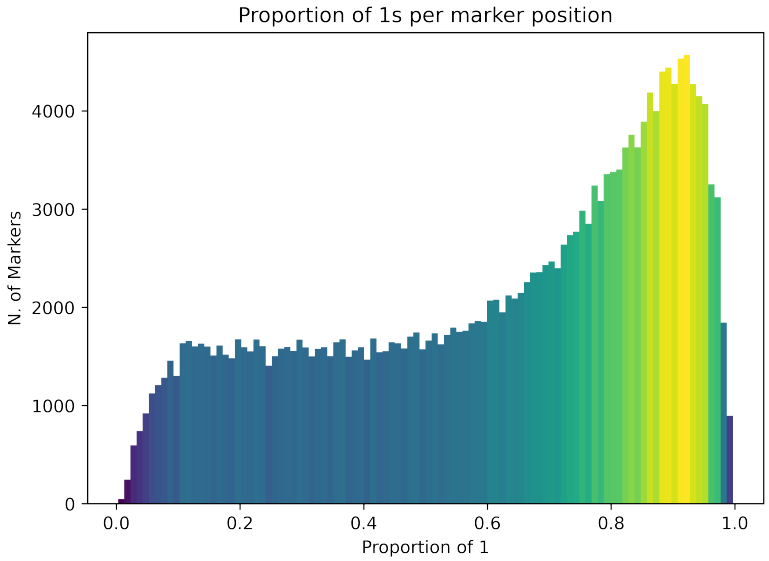


Figure 10: Distribution of marker scores. The marker scores are very imbalanced.

Figure 10 shows the histogram of the marker score, the distribution of this data will have implications for our analysis. Figure 11 illustrates an estimation of the population structure in the genetic data. The broad geographic distribution of the accessions is expected to be reflected in the genetic variability due to adaptation to specific conditions. The exact different systematic relationships between our arabidopsis accessions, however, are unknown. Since in this setting we cannot assume an unstructured population, we need to account for possible relatedness effects to avoid spurious associations in genetic association analysis.

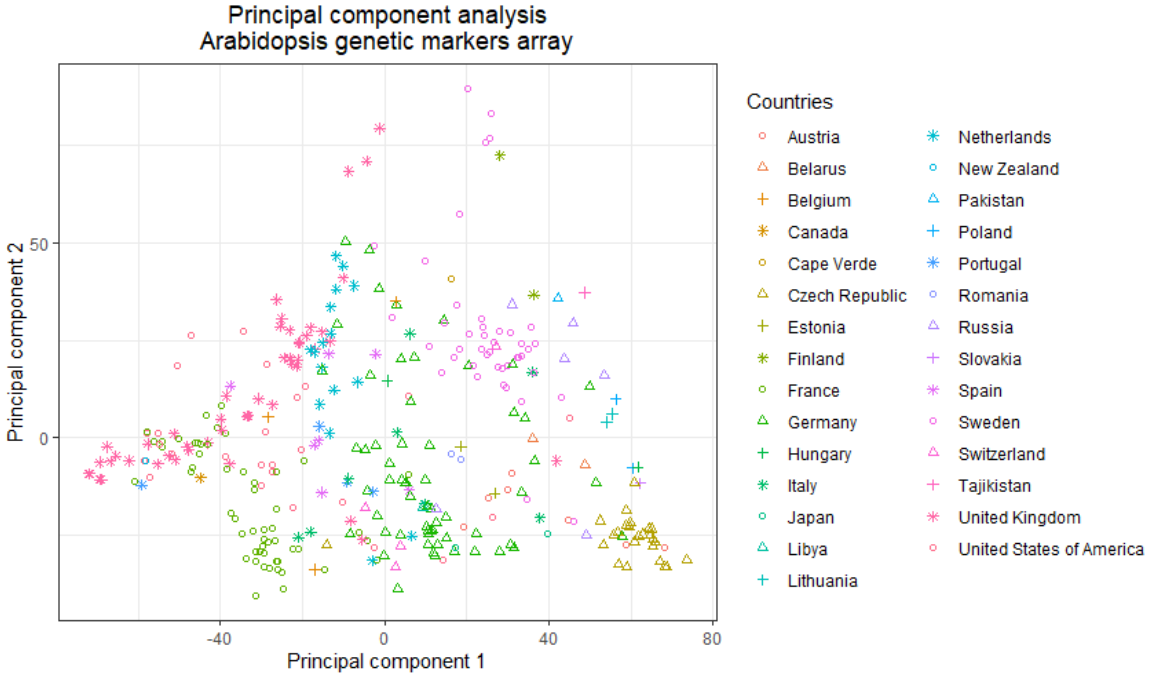


Figure 11: Population structure in the arabidopsis sample, each point represents a different genotype. It shows the first two principal components of the markers matrix.

2.2 Genetic mapping

To study genes driving arabidopsis resistance to aphids we need a framework that allows mapping components of the DNA to behavioural traits.

When a nucleotide (the basic building block of DNA) is susceptible to variation in the population (polymorphism) it is called a Single Nucleotide Polymorphism, these variants are called **alleles** [35]. Our interest, following the central dogma of molecular biology, is in identifying the specific SNPs that are potentially responsible for some interesting traits. Therefore, in order to perform genetic mapping, we need relevant traits and genetic information, called a genetic map.

When our data is susceptible to population stratification and most of the genome is known, Genome-Wide Association Studies are the framework we need. Quantitative trait locus mapping is closely related, however, it is used when the genetic relatedness between genotypes is completely known, typically in an experimental population. Relatedness can be estimated from markers but in that case, it is not completely known. As we mentioned above, our 350 accessions can not be assumed to be unstructured so we need to account for population stratification.

2.2.1 Genome-wide association studies

Here we outline how GWAS are practically implemented in the experiments and discuss the limitations of this approach.

Required data. In order to conduct GWAS, phenotype data and genotype data are needed. The phenotype data can be any measured or discrete trait, in this project it will be the handcrafted features and the artificially generated representation of the trajectories through deep learning. This is an array Y of n observations as rows and d' traits as columns (for instance, array Z in Figure 3). The genotype data are the markers, the specific marker score for each particular SNP. In the case at hand, this is translated into a binary array X with g rows for the different genotypes and q SNPs. Further covariates could be added but we will not be using them.

Model definition. A general definition of a linear mixed model for GWAS is given by [43]:

$$y \sim W\alpha + X\beta + Zg + e$$

$$g \sim N(0, \sigma_A^2 \Psi)$$

$$e \sim N(0, \sigma_e^2 I)$$

where y is a vector of n observations (a column of Y). W is a matrix with covariates and an intercept term, X is, as defined before, the markers matrix. g represents random effects to capture **polygenic** effects of other SNPs owing to population stratification. Z is an incidence matrix assigning observations to its corresponding genotype random effect. e is the residual (non-genetic variance) term. Finally, Ψ is a genetic relationship (kinship) matrix. Now, we need to estimate the parameters associated with intercept and covariates (α), with the SNPs (β), the variance of the genetic random effects (σ_A^2) and σ_e^2. The P-values associated with β are what we are eventually interested in, they are usually displayed in a Manhattan plot.

Dealing with related individuals, polygenic effects. It may be the case that there are related genotypes in our dataset, in our case, related Arabidopsis accessions. Polygenic effects are due to linkage disequilibrium between unlinked loci in a subgroup of related genotypes. As a consequence, some observations will behave similarly in related populations. This may be problematic because it can lead to spurious associations [20]. A solution to the problem is to compute the pairwise relatedness matrix Ψ from high-density markers [26], our X.

Predicted means by genotype. To speed up computations it is sensible to capture the mean by genotype and fit the above model at the mean level. In that case, y is a vector with n means

corresponding to the number of different genotypes. The results are equivalent. Furthermore, it is important to correct for possible effects arising from the experimental design. To do so and to maintain consistency with the EthoAnalysis software pipeline, the means we use are calculated with a linear mixed model that captures random effects at the arena and trial level with genotype as a fixed effect. This is not equivalent to using empirical means. For the data at hand the model looks as follows:

$$y \sim Z\beta + r + t + e$$
$$r \sim N(0, \sigma_r^2)$$
$$t \sim N(0, \sigma_t^2)$$
$$e \sim N(0, \sigma_e^2 I)$$

where Z is the binary array of dummy variables assigning an observation to a particular genotype, β is the vector of coefficients for each genotype, r are random effects at the plant level, t at the trial level and e the error term. Once the model is fitted, the marginal means are extracted to use in our GWAS model. As a result, the means only contain variability explained by the genotype and not noise derived from the experimental design[3].

Computation. We use the R package **statgenGWAS** [44] to perform our GWAS. More information regarding optimization procedures is available in their vignette.

Limitations. The main limitations of this approach are related to multiple testing and the limited capability to study pleiotropic genes. GWASs are limited in their capability to study relationships with several traits at the same time, which may provide new knowledge and more powerful tests. Evaluating multiple univariate GWASs leads to serious problems of family-wise error rate which need to be addressed, for example, by Bonferroni correction.

2.2.2 Reverse genetic mapping

> *"Problems such as stellar recognition or analysis of gene expression data could be high adventure for statisticians. But it requires that they focus on solving the problem instead of asking what data model they can create. The best solution could be an algorithmic model, or maybe a data model, or maybe a combination. But the trick to being a scientist is to be open to using a wide variety of tools"*
>
> — L. BREIMAN (2001)

To overcome the above-mentioned limitations several alternatives have been proposed. Treating the marker score as a response to test for the relationship between multiple phenotypes and a particular SNP with samples of related individuals has been proposed for [15]. As they explain in their paper, "[...] in conventional models, the phenotypic trait is treated as the response and the distribution of the trait values needed to be specified. For example, the normality assumption is often required for a quantitative trait. In our method, the trait is treated as an explanatory variable, which allows us to leave the distribution unspecified." They proposed a quasi-likelihood approach for a logistic regression model and an omnibus test that takes into account population structure. While this approach allows using an arbitrary number of traits with an unspecified distribution, it requires a very specific model to derive a reliable test to assess the relationship between SNP and traits. Hence, more flexible approaches such as K-nearest neighbours, support vector machines or deep learning architectures are not suitable. This makes it difficult to study high-dimensional data such as video, images and similar formats in this

[3]Those models are implemented through the R library **lme4** [4] .

framework. A linear model is not the best approach for this kind of data.

Closely related to this approach are the PheWAS (Phenome Wide Association Studies), also motivated by the limitations of GWAS and thought as a complementary approach. PheWAS also uses the traits as explanatory variables. The main difference concerning the reverse mapping approach explained above is that the latter focuses on the relevance of a particular trait given a SNP while the former focuses on broad relationships through omnibus tests. Usually, the SNPs to investigate are the ones previously derived from GWAS [46], which can detect novel pleiotropic associations. This is coherent with the idea of testing for specific trait relationships and not broad exploratory purposes. Usually, logistic regression is used to test for the relationship and covariates to control for confounding through population structure are included. Those covariates are generally principal components (PC) of the genetic markers [40]. The emphasis of PheWAS on specific traits invalidates this approach for high-dimensional input, on the one hand, including PC to control for population structure invalidates the option of omnibus tests. On the other hand, the same problems with multiple testing arise if we focus on evaluating the relationship for one trait at a time.

For exploratory purposes, we adhere to a simplified version of this reverse mapping framework. This framework is interesting here due to our way of generating the traits. Since what we create is a low-dimensional representation of the input, the important information relies on the relative position of each observation in this low-dimensional space. In order to extract the information contained in this space, we propose to fit a fast classifier on top of the created representation [28]. Similar approaches using factor analysis or logistic regression have been proposed in the multi-trait setting [38, 16]. The main motivation for this kind of approach is, as explained by [38]; "modelling the association between linear combinations of phenotypes and the genotypes at each SNP may uncover genetic association hidden to both single phenotype GWAS and those based on a priori defining a phenotype as a fixed function of traits". While this is important for classical traits, it is even more important for artificially generated traits where the only meaning is a (linear or non-linear) combination of the generated traits. Fitting a fast classifier on top of a low-dimensional representation is a fundamental idea of representation learning (see section 2.4).

In order to explore the whole genome for association and allow flexibility in the models used, we rely solely on predictive performance on unseen data. One problem with this approach is that we cannot account for polygenic effects due to population stratification and this may lead to false positives. Confounding due to experimental design is also possible. However, here we will neglect these problems to superficially explore possible interesting areas in the genome acknowledging that the results may not be reliable.

This strategy can be seen to fit into the ideas of Breiman: "The most obvious way to see how well the model box emulates nature's box is this: put a case x down nature's box getting an output y. Similarly, put the same case x down the model box getting an out-put y'. The closeness of y and y' is a measure of how good the emulation is. For a data model, this translates as: fit the parameters in your model by using the data, then, using the model, predict the data and see how good the prediction is" and a bit after "If the model has too many parameters, then it may overfit the data and give a biased estimate of accuracy. But there are ways to remove the bias. To get a more unbiased estimate of predictive accuracy, cross-validation can be used [...]. If the data set is larger, put aside a test set" [6]. However, the data problem here is more complicated. We need to test associations for thousands of SNPs and the distribution of the marker score in each one is different.

The idea is to train a fast classifier on top of the automatically created and handcrafted features to test for promising SNPs. In what follows, we describe how the genome-wide reverse mapping is implemented:

Model. We use support vector machines (SVM) with a RBG kernel (default) [4]. The model input Z

[4]We use the Python library **scikit-learn** [39] to implement it.

is the matrix of d' traits, the dependent variable is the allele frequency in each SNP i.e. class 1 or 0. The model is fitted for each one of the available SNP.

Training procedure and regularization. The data is split into training and test set. Default regularization parameters are used. Performance is evaluated in the test set.

Metrics. Accuracy is not a suitable performance metric for this task. Trivially over-fitting the training set and predicting just the most common class would render accuracy way above 50% (see Figure 10). In consequence, accuracy is dependent on underlying class frequency.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Matthews correlation coefficient (MCC) is a better metric in this conditions [7]. Its range is from -1 to 1, and it can be understood as the correlation between the predicted and the real class labels.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

The most attractive features of MCC, in this case, are that it equals 0 in the case of trivially over-fitting and the invariance for class swapping. The former is straightforward to see in the definition above, using the convention that $\frac{0}{0} = 0$.

Performance evaluation. We want to test for association between the traits and the SNP based on predictive performance. The null hypothesis, therefore, is that there is no relationship between the two. This could be solved by a simple permutation procedure [17] to construct an empirical approximation to the null distribution of MCC. However, this is not computationally possible, since it would mean performing a permutation test for each SNP. In consequence, for simplicity shake, we explore the relationship based on the MCC value itself.

Indeed, MCC is a better metric than accuracy for this type of scenario, however, in highly imbalanced scenarios, higher MCC values are harder to achieve [49] which necessarily reduces the power of this approach.

With the MCC values per SNP, we can construct a (pseudo) Manhattan plot by changing the negative values to 0. A scheme of this process is depicted in Figure 12.

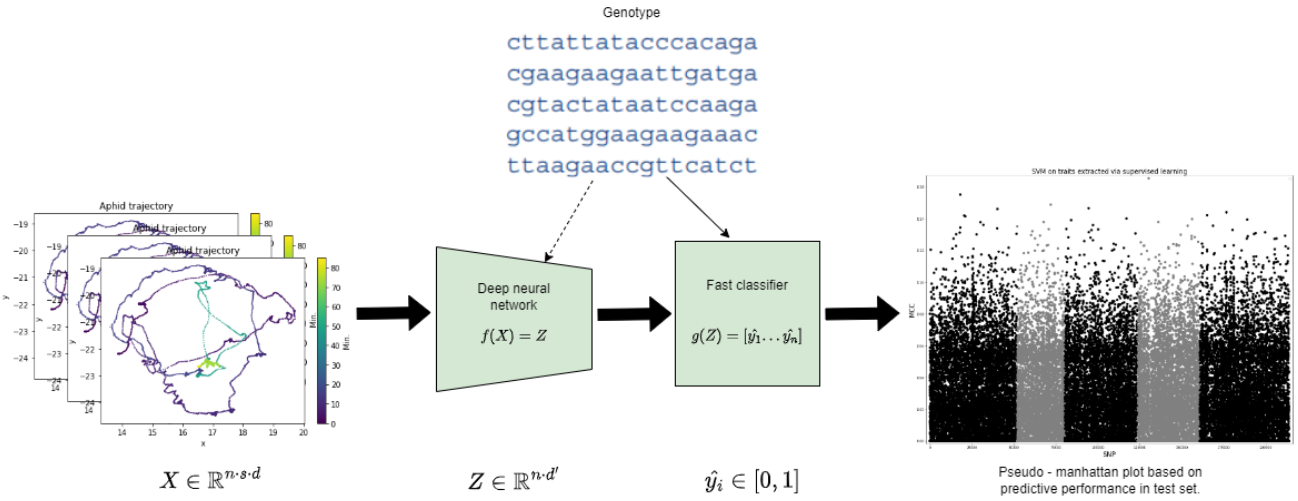


Figure 12: Reverse mapping scheme: We generate new traits with some complex function in a supervised or unsupervised fashion (hence, the dotted line), fit a fast classifier on top of these traits, extract the MCC in the test set and finally we obtain our pseudo-Manhattan plot. While rudimentary, it is interesting to explore if we can confirm previous findings and/or find new SNPs.

2.3 EthoAnalysis

EthoAnalysis is a software designed for the analysis of insect video tracking data provided by tracking software. It is developed by Wageningen University and Research in the TKI project HTP Phenotyping of plant resistance to sucking insects[5]. It is used to extract high-level behaviour statistics such as average velocity, total duration moving, the number of movement events of a certain velocity range or total time pausing. As an end product, it performs GWAS, among other things.

EthoAnalysis generates these statistics through the creation of events. An event is a segment of the trajectory which is labelled as halting, moving, and not detected. Each event has a starting time and a duration, and from these series of events, the various behaviour statistics are extracted. The parameterization defines the criteria to label each event.

2.3.1 Handcrafted feature extraction

Naturally, we will not dive here into how to use the software but we will specify the parameterization used to derive the handcrafted features. The velocity threshold is 0.04 mm/sec, it is used to define what is moving and what is not. The look ahead window is 10 frames (0.2 seconds), which defines an event. Extreme velocity is $>$ 2mm/sec (which filters out events). The maximum duration of short halting events is 180 frames, minimum for long halting events is 1500 frames. For movement, events is 10 and 50 respectively. Velocity is considered slow when it is below 0.2 mm/sec and fast when it is above 0.4 mm/sec.

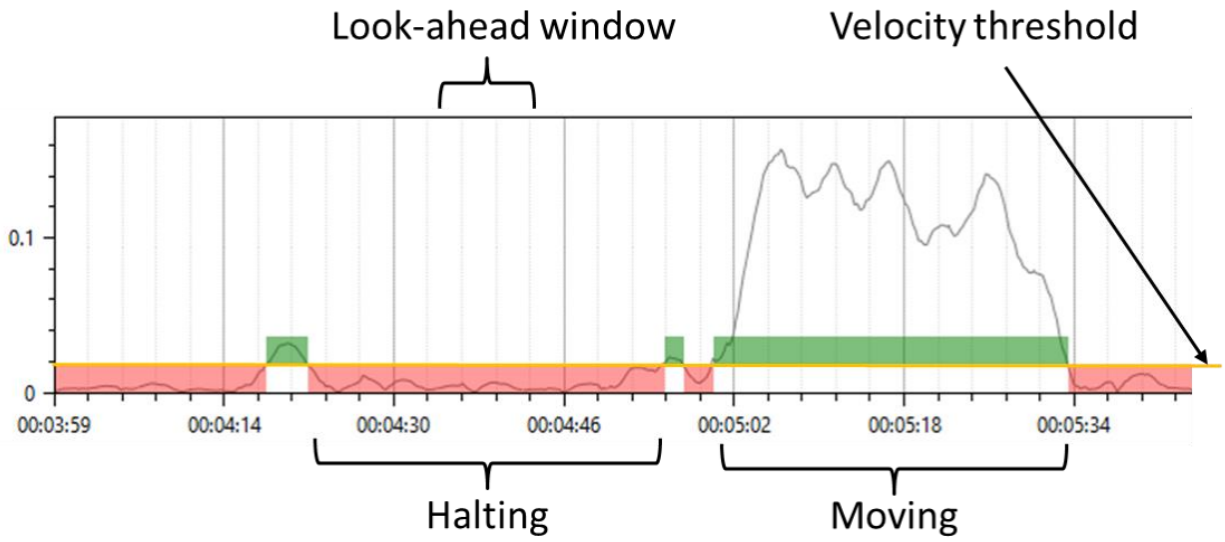


Figure 13: Example of EthoAnalysis parameterization. Image is extracted from EthoAnalysis manual.

There are 29 different traits without missing data in this handcrafted data representation, we will be working just with the traits with no missing data. They are a combination of the specified thresholds to define averages of halting duration, movement distance, movement duration, velocity, halt frequency, ratios between moving and halting and similar measures. This will be further specified in the results.

2.4 Representation learning

As we explained above, the focus of this thesis is on the use of deep learning to extract meaningful traits from the trajectory data. Instead of ad hoc parameters, we construct a function capable of untangling the complex trajectory data into a simpler representation. This means that we recognize important factors of variation while removing the ones that are not relevant to our application. An

[5]Main software development by Maarten Jongsma, Johannes Kruisselbrink, Leo Poleij, Gerrie Wiegers. Acknowledging valuable input of Paul Goedhart, Lia Hemerik, Karen Kloth, Manus Thoen, Hilko van der Voet.

ideal representation, therefore, is one which makes the input data conditionally independent of the response (we recover all the signal available), low-dimensional and disentangled (the components of the representation should be statistically independent) [23]. In consequence, a good representation will make other tasks, such as classification, easier.

This process is known as **representation learning**, a toy example is represented in Figure 14. In this example, the classes are suitable to be linearly discriminated using just ρ once we have transformed the data. We aim for a similar solution here, the created representation should be able to put together similar arenas and far away different ones. Since we assume that non-random variability in the data is due to the plant genome, we hope to obtain a representation in which the genetically explained variability is maximized. However, it is crucial to realize that the "black-box" nature of this family of algorithms may conceal the use of non-genetic variability to improve model performance [41]. Population structure or experimental design could be driving our representation instead of genetic variability. In this context, this is difficult to assess.

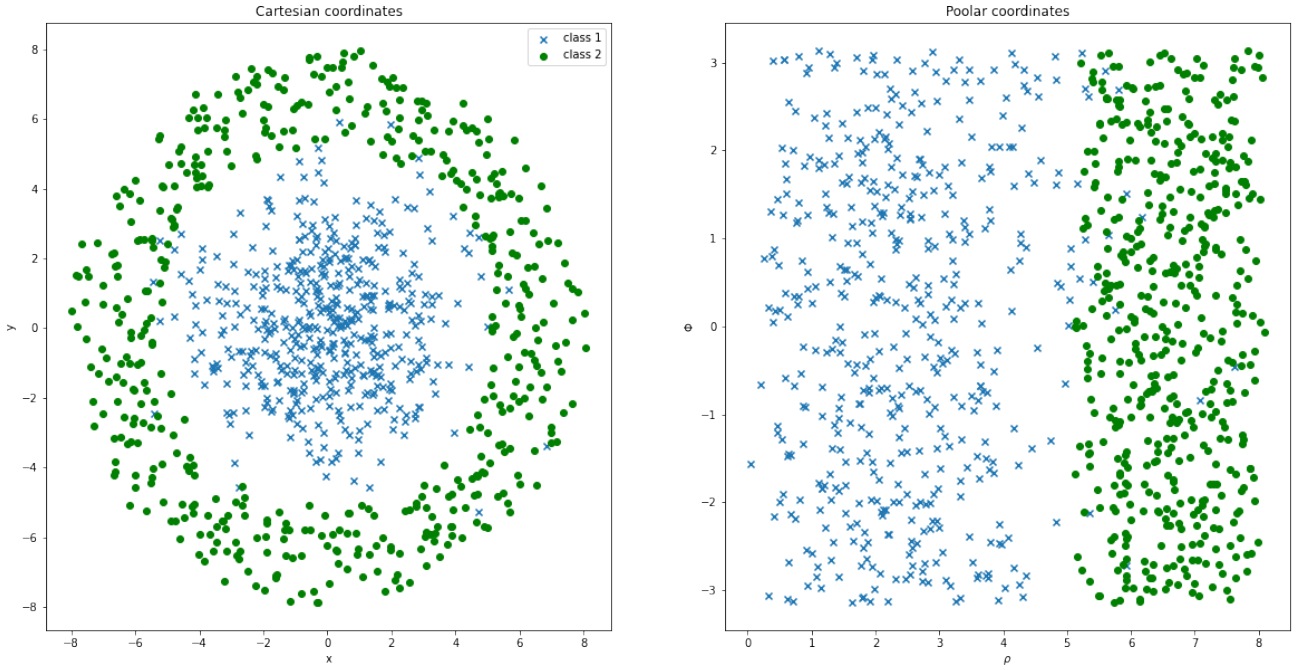


Figure 14: An example of how **representation learning** allows to linearly discriminate between categories, facilitating the classification task. In this case, the "learned" function is just the change from Cartesian coordinates to Polar coordinates.

Deep learning is a powerful framework to accomplish those aims. It is unlikely that the factors determining genetic variability in the trajectories can be extracted through simple (linear) combinations of the data. For instance, average speed, total distance covered or standard deviation of speed are not very useful summaries when it comes to detecting significant differences between genotypes. It is important to notice that most if not all of the features extracted by EthoAnalysis are indeed highly complex (non-linear) transformations due to the discretization of the input.

The capability of deep neural networks to be universal function approximators is, consequently, very useful here. This is achieved by generating a hierarchical representation of the input data with increasingly abstract features. Each consecutive hidden layer in the model can digest the previous representation (by previous layers) to generate an increasingly useful one towards a given goal. This goal is specified by the loss function. A goal could be, recognizing hand-written digits, species of animals in an image or the probability of marker score 1 given an arena. The hierarchical non-linear transformations simplify progressively the task until it can be easily solved, for example, by a linear classifier.

In the task at hand, it is likely that the relationship of optimal traits with the raw input is highly non-linear, that being the case, the use of a deep learning model grants us with enough power to disentangle them.

Now, the objective is to create a low-dimensional representation of d' optimal traits that is suitable to be analyzed in a GWAS. The quality of this representation depends on how much genetically explained variability can we capture. This has to be concretized in a particular model with a specific objective function. In practice, this essentially depends on two choices. First, we have to decide which architecture we will use to create the representation. This is, how the functions we create, the deep learning models, relate to the input. Secondly, and more importantly, we have created a "clue" that helps our model to select the best parameterization to find the best traits. This "clue" is the objective or loss function, we need to make some assumptions to align the minimization of this loss function with our objectives. Indeed, optimal is defined with respect to the loss function, therefore, the usefulness of the outcome depends on how much this function relates to our objective.

In what follows we describe why we select a particular model, which assumptions are made and how it is practically implemented.

2.4.1 Supervised learning

The supervised learning scheme provides a strong indication of which representation is useful. Each observation is assigned a label belonging to a particular class. Under this framework, we want to generate a representation that linearly separates those classes. This is exactly what is represented in Figure 14. This forces our model to select the factors of variability that explain differences between the classes.

Now, to obtain labels we need to go back to [32]. We know that the gene *SLI1* (sieve element-lining chaperone1) is relevant in explaining aphid behaviour, it is 80-kilo bases around the most significant SNP located at position 3338114 in chromosome 3 (in the array, this is column 86001[6]). Knowing that a gene is relevant in explaining aphid behaviour allows us to construct labels that we should be able to explain, i.e. above random predictive performance in unseen data. While there are several ways to construct the labels we simply use the marker score of the most important SNP (86001) as a label for the arenas. Each arena either has the marker score 1 or 0 for SNP 86001.

This frames the problem of obtaining a good representation into a classification problem. We assume that to achieve "good" classification performance in unseen data the model must generate relevant traits. If we can predict correctly the marker score of a SNP that indeed drives resistance to aphids we must have found a data representation that retrieves relevant factors of the data. Or at least, relevant factors with respect to the kind of resistance that this particular gene *SLI1* promotes.

However, this is a huge assumption and it has several drawbacks. Firstly, it is unclear the signal-to-noise ratio for this type of classification task. It is not expected for just one SNP to drive a lot of variability in the data. Secondly, this approach is not suitable when we do not know any gene driving the variability of our data. This is precisely what we ultimately aim to find. Finally, the created representation may be excessively focusing on the variability explained by a particular SNP and we could be missing other relevant behaviour that may be explained by other genes.

Nevertheless, this classification task, ideally, allows us to untangle at least one kind of interesting behaviour. Moreover, the traits extracted do not necessarily have to be correlated with the handcrafted ones which leaves the ground for further investigation and biological interpretation of the traits.

Is worth mentioning that supervised training does not necessarily involve explicitly characterizing the representation. Most of the time, we do not care about what kind of representation our model is creat-

[6]For the shake of simplicity we will refer to the SNPs for its position (column) in the markers array.

ing to classify the data, we just care about classification performance. If the classification performance in unseen data is "good" it means that the model is capable of capturing the relevant factors in the data. Nevertheless, it is important to acknowledge that the power of deep learning resides precisely in the generation of useful representations. The actual classification task is performed usually by a simple linear model.

At first sight, this scheme seems to allow us to perform the genome-wide reverse mapping. Following this idea, we could repeat this classification task for all available SNPs and evaluate the signal retrieved, "good" test performance means that the SNP explains the variability in the data. Naturally, this is computationally infeasible, especially if the hyper-parameters of the model have to be selected each time. But, even more importantly, it is utterly wasteful. Trying to ignore the representation step forces us to re-create a representation for each SNP when it is clear that at least part of the hierarchical representation is shared. For these reasons, we consider that extrapolating the created representation to other SNPs is a better strategy (see Figure 12).

For supervised learning in classification tasks, the loss function is commonly defined as cross-entropy. Given a set of arenas $X \in \mathbb{R}^{n \cdot s \cdot d}$ and marker score for SNP 86001 $Y \in \{0,1\}^{n \cdot 2}$ where each column represents one class, being $Y_n = [1 - Y_{n,1}, Y_{n,1}]$. The loss function per observation is simply defined as:

$$L_n = -\{Y_{n,1} \log(p_{n,1}) + Y_{n,2} \log(p_{n,2})\}$$

Where $p \in (0,1)^{n \cdot 2}$ is defined as $p = NN(X)$ with 2 fully connected nodes D at the last layer with softmax activation function. Being $NN_{-1} \in \mathbb{R}^{n \cdot d'}$ the layer before the last:

$$D_{n,i} = w_{0,i} + w_{1,i} NN_{-1}^{n,1} + w_{2,i} NN_{-1}^{n,2} \cdots + w_{d',i} NN_{-1}^{n,d'} \quad for\ i = 1, 2.$$

$$p_n = [\frac{e^{D_{n,1}}}{e^{D_{n,1}} + e^{D_{n,2}}}, \frac{e^{D_{n,2}}}{e^{D_{n,1}} + e^{D_{n,2}}}]$$

NN_{-1} is our representation, the d' traits. Notice that the representation must tend to be linearly separable to increase predictive performance.

Now let us focus on how $NN(X)$ is defined [7]:

Time series classification with TCN. Here we treat the data as a temporal series and analyse it with a temporal convolutional network (TCN). For the TCN we use the "Inception-Time" architecture proposed in [24] with small adaptations such as dropout layers. This is a translation of the inception module to temporal series, (i.e. different kernel sizes for different 1D convolution layers in parallel plus a pooling layer and a posterior concatenation, batch normalization, and ReLU activation). The architecture consists of n inception modules followed by global average pooling and a fully connected layer with softmax activation. Our traits are the result of the global average pooling, the average of each filter activation in the last convolutional layer gives each dimension. Residual connections are added every 3 inception modules. Each inception module is preceded by a 1D convolution with kernel size 1 called "bottleneck".

[7] All models are implemented in **Tensorflow** [36]

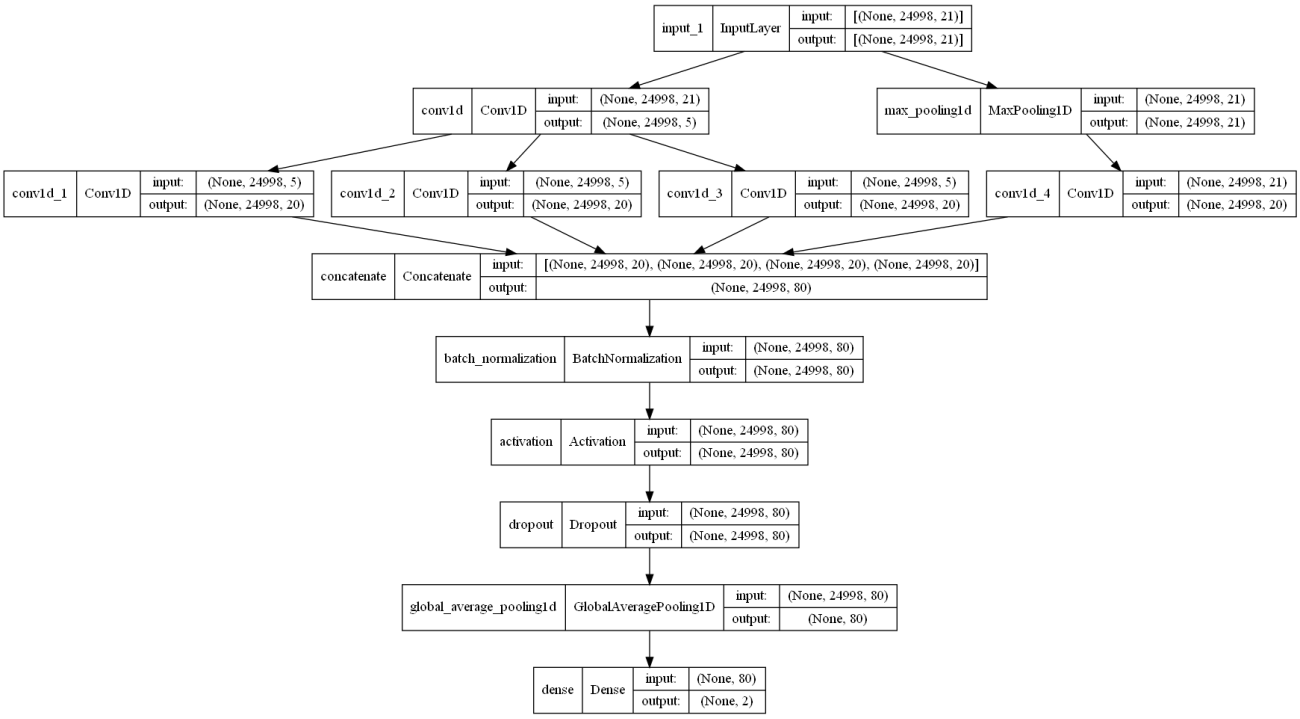


Figure 15: Architecture for Inception Time with depth 1

The hyperparameters defining this architecture, therefore, are the number of inception layers (depth), the number of filters, the kernel size, the probability of dropout, and the hyperparameters for the optimizer (we use ADAM). Figure 15 represents Inception Time with 1 inception module.

Time series classification with Transformers [45]**.** The transformer architecture is based on a multi-head (parallelized) self-attention layer. In essence, this layer performs an outer product between the embedding of parts of the input i.e. (tokenized) words The result is a mask signalling the importance of each input bit with respect to the whole input. This forces the embeddings to increase for the type of input bit that is most relevant given the context of the whole input.

Splitting our trajectory in "events" (following EthoAnalysis) and generating a contextual embedding for each of them using a self-attention mechanism is the aim of using a Transformer architecture in this context. We first use a 1D convolutional layer which downsamples sequence (the with stride size equal to kernel size) into "events". We consider 10 seconds for each "event" and n-dimensional vector representation for each one given by the number of filters in this first layer. This generates the embedding of our events, which ideally carries out the relevant information about the relevance of this behaviour of the aphid in that particular period. With this, we reduce the number of time steps by a factor of 50 and it is possible to use self-attention layers. Then the self-attention mechanism allows the vector representation of the "event" to be constructed taking into account the whole sequence. However, a clear limitation of this approach is the hard boundary between "events". It is unlikely to capture the optimal combination of steps. Higher computational power allowing to reduce the stride, or generating different "events" in parallel, would improve the capabilities of the model.

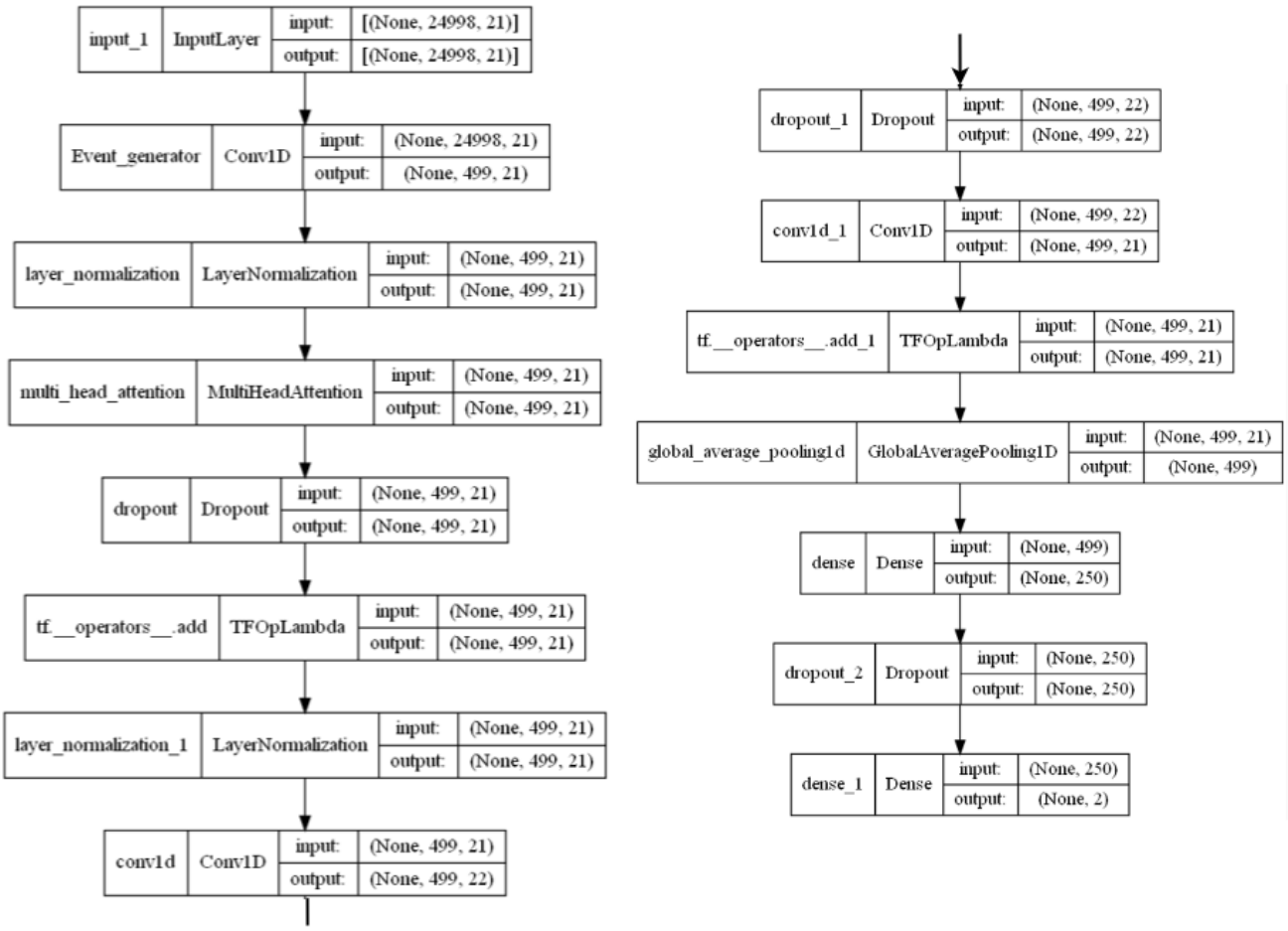


Figure 16: Transformer architecture with depth 1 attention layer. The residual connections are named as "tf.__operators__.add".

This implementation uses residual connections from the generated embeddings to the output of the multi-head self-attention layer and from that to the output of the feed-forward part (layer normalization, convolution, dropout and convolution). The hyperparameters for this architecture are the number of transformer blocks (attention and feed-forward part), probability of dropout and the number of filters in the convolutional layers. The before to last dense layer is fixed to 250 nodes.

Image classification. We transform the trajectory to a grayscale image with a resolution of 128x128 and then transform it to RGB simply by making 3 copies of the array and concatenating them. To process the images we make use of transfer learning, specifically, we use Xception [8] trained in ImageNet. A similar approach for boat trajectory has been used in [33]. The network is already trained to achieve high performance in image classification. We expect to be able to use this training to generate a powerful representation of the data. We use a new top layer to this network for our 2 class classification problem.

2.4.2 Self-supervised learning

> *"The quintessential example of representation learning algorithm is the autoencoder"*
>
> — Goodfellow et al.[18]

Self-supervised (or unsupervised) learning attempts to learn useful representations from unlabeled data. A clear example of unsupervised learning may be principal component analysis (see Figure 24). Probably the most important application of self-supervised learning in deep learning is the autoencoder. An

autoencoder is defined as a neural network that attempts to copy its input to its output, which is a non-trivial task under some constraints.

An autoencoder is composed of two parts, an encoder f and a decoder g. Therefore, we want to minimize the loss given by $L(x, (f(g(x))))$, this loss could be the mean squared error between input and reconstruction. The encoder output is (usually) a lower dimensional representation of the input. This code is used by the decoder to generate a reconstruction of the input. We want this representation to be as close as possible to the original input. However, we impose some constraints in the code such that the model is forced to prioritize the factors of variability that allow for the best reconstruction of the output. We do not want to reconstruct the model perfectly but to understand the main aspects of our data. In general, the basic idea behind the autoencoder scheme is that compression is akin to understanding. Consequently, the code (or representation) of the data is conditional to the input and it is embedded in a space where its variability is relevant to the reconstruction of the output.

In the case at hand, we assume that the main source of non-random variability is the genetic material in each Arabidopsis accession. Ideally, an autoencoder recovering the main aspects of the data should recover dimensions very related to the genetic variability. If the main source of variability in the data is the genetic material, the optimal way of reconstructing the data is using the information contained in the genetic material. However, when the autoencoder is a non-linear function it is not expected to have a linearly disentangled encoding. Being that the case, non-linear mapping methods are more interesting than classic GWAS and we should not expect high heritability values for the generated traits.

β-Variational autoencoder. A very powerful autoencoder can potentially "memorize" the observations and it would not learn anything relevant. Regularized autoencoders allow learning useful representations even when the non-linear encoder could potentially "memorize" the data. Regularization allows us to learn useful complex representations.

Following [18], we can frame the regularization of an autoencoder as approximately training a generative model. This means to approximate the true distribution of our data. For instance, we can train a sparse autoencoder by maximum a posteriori (MAP) estimate of an encoding with a Laplace prior (also known as l^1 penalty). We can go one step beyond and use our encoder to parameterize a mean field approximation to the distribution of our data. Since the decoder is a non-linear function, our mean-field approximation, for instance, a 25-dimensional diagonal gaussian can approximate any distribution [12]. In doing so we are parameterizing the distribution of the relevant factors in our data. In this case, therefore, we do not extract point estimates of our traits but the parameterization of their distribution. Which, being conditional to the input are expected to represent the relevant variability.

The variational autoencoder (VAE) is exactly this, a bayesian variational approximation to the posterior distribution of the data. The main distinction with classical variational Bayes is that the decoder applies a transformation that allows us to create another completely different distribution.

Practically this is achieved through *Bayes by Backpropagation* [5] and implemented using the **Tensorflow probability [11]** module.

Formally, we try to maximize the probability of each observation:

$$P(X) = \int P(X|Z;\theta)P(Z)dZ \ ,$$

where X is our raw data, Z is the encoding or representation and θ is the parameters of the network. Now, we are interested in computing Z, which we hope will contain the main factors driving the data variability. This can be the case when the reconstruction of X from X (through Z) is not a trivial problem. The posterior distribution for Z is:

$$P(Z|X) = \frac{P(X|Z)P(Z)}{P(X)} \ .$$

Since this distribution cannot be computed, in practice, Z is sampled from an arbitrary distribution $Q(Z|X;\theta)$, our encoder parameterises a mean field approximation for $P(Z|X)$, the variational posterior. Naturally, Q must be such that the probability for Z that leads to a good reconstruction is higher than in the prior space $P(Z)$. This generates a tension between fulfilling the likelihood requirements and the prior requirements. Our objective is to minimize the Kullback-Leibler divergence D_{KL} between $Q(Z|X;\theta)$ and $P(Z|X)$:

$$D_{KL}(Q(Z|X;\theta)||P(Z|X)) = \int Q(Z|X;\theta)\log\left(\frac{Q(Z|X;\theta)}{P(Z|X)}\right)dZ\ .$$

This can be shown to be proportional to our new loss $L(Z|X;\theta)$:

$$\begin{aligned}
&\int Q(Z|X;\theta)\log\left(\frac{P(X)Q(Z|X;\theta)}{P(Z|X)P(X)}\right)dZ \\
&= \int Q(Z|X;\theta)\log\left(\frac{P(X)Q(Z|X;\theta)}{P(X|Z)P(Z)}\right)dZ \\
= \int Q(Z|X;\theta)\log(P(X))dZ &+ \int Q(Z|X;\theta)\log(\frac{Q(Z|X;\theta))}{P(Z)}dZ - \int Q(Z|X;\theta)\log(P(X|Z))dZ \\
&= \log(P(X)) + D_{KL}(Q(Z|X;\theta)||P(Z)) - \mathbb{E}_{q(Z|X;\theta)}(\log P(X|Z))\ .
\end{aligned}$$

Since $\log(P(X))$ is constant:

$$D_{KL}(Q(Z|X;\theta)||P(Z|X)) \propto L(Z|X;\theta) := D_{KL}(Q(Z|X;\theta)||P(Z)) - \mathbb{E}_{q(Z|X;\theta)}(\log P(X|Z))\ .$$

Intuitively, the second term is the likelihood and the first term is the prior. In the actual implementation, we use a diagonal 25 dimensional diagonal normal as $Q(Z|X;\theta)$ parameterized by the encoder with a 25-dimensional isomorphic normal prior $P(Z)$. This loss function is practically reduced to:

$$L(Z,\theta|X) := \beta D_{KL}(Q(Z|X;\theta)||P(Z)) - \frac{1}{N}\sum_{i=1}^{N}\big(\sum_{j=1}^{s}\{\hat{Y_j} - Y_j\}^2\big)_i\ .$$

Where Y_j is speed per step j, $\hat{Y_j}$ its reconstruction and N the number of observations. The second term is the average mean squared error For simplicity and computational ease, we only try to reconstruct the speed. We know that speed is the most relevant dimension and the only one used by EthoAnalysis. Focusing on speed allows for recovery the most important behaviour and at the same time makes it easier to understand the quality of the decoder reconstruction. However, we do condition the distribution of Z on all the dimensions in the raw data X. Additionally, we allow for an additional hyperparameter β [22] which allows tuning the relative importance of the likelihood. In this case, $D_{KL}(Q(Z|X;\theta)||P(Z))$ can be calculated exactly or approximated through sampling.

As a final note, it is important to underline that this model can indeed be used as a generative model, since the decoder has learnt to recreate the representations from Z it is possible to explore the latent space sampling from $P(Z)$ and consequently generating new data.

Simple TCN. To implement the β-VAE algorithm we use a simple TCN network (see Figure 17 with a convolutional encoder, which is used to parameterize Q and a deconvolutional decoder to reconstruct the speed dimension. The encoder consists of 5 1D convolutional layers with GELU activation [21] and a final global average pooling layer before a dense layer with 50 nodes used to generate 25 means and 25 standard deviations given X. With this, a TensorFlow probability layer samples Z from $Q(Z|X;\theta)$ and the encoding process is finished. The decoder consists of 3 transpose 1D convolutional layers that upsample the encoding from the original Z to 25000 steps with 64 filters and GELU activation and 2 1D convolutional layers one with GELU activation and 64 filters the last one with linear activation and one filter.

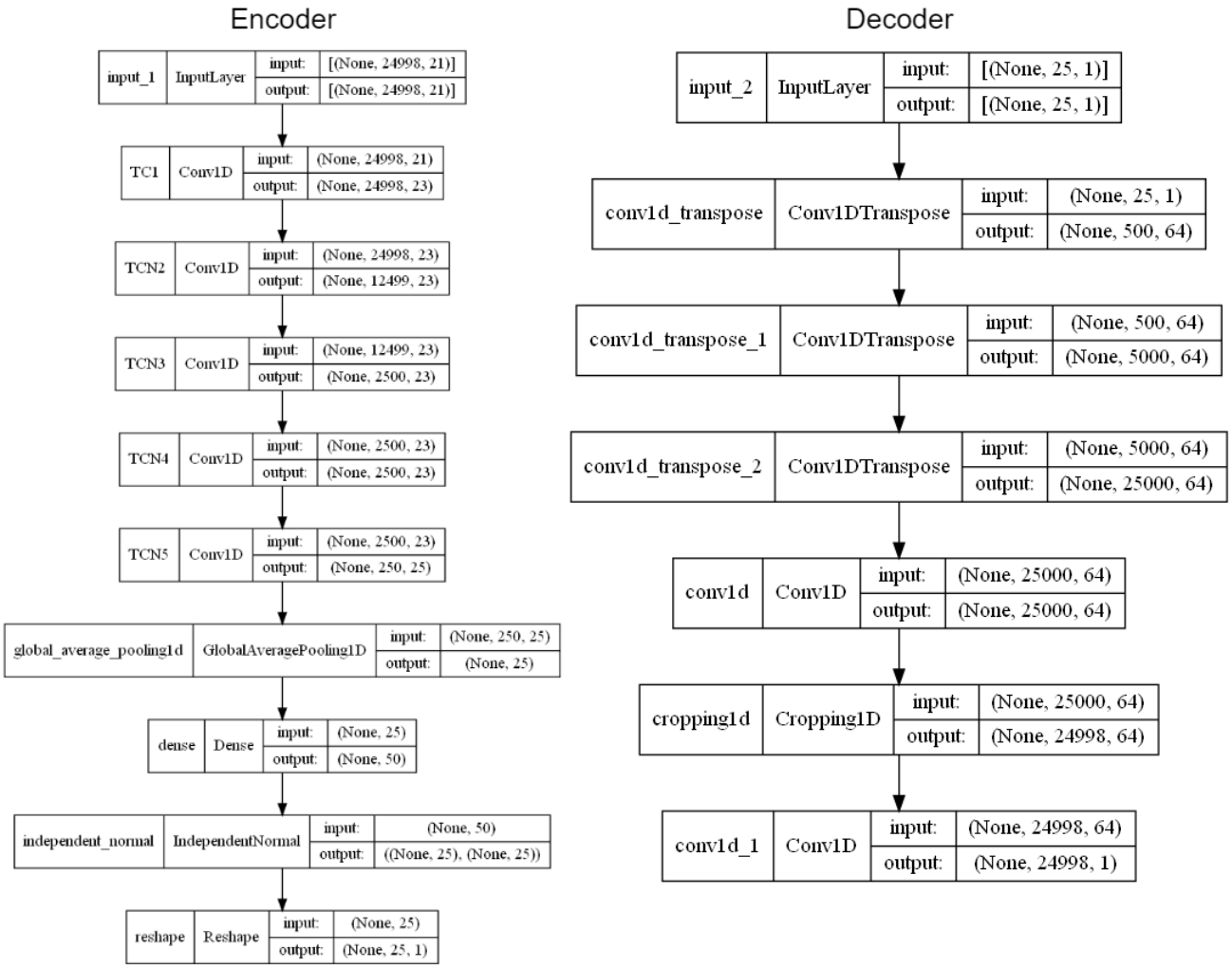


Figure 17: Variational autoencoder architecture. The encoder parameterizes a 25 dimensional normal. Samples from this distribution generate the code for the decoder. The penalty $D_{KL}(Q(Z|X;\theta)||P(Z))$ constrains the parameterized distribution to a neighborhood of the prior.

The hyperparameters to tune in this case are the relative importance of the likelihood β, the number of filters in the encoder and the ADAM optimizer hyperparameters. We explore the effects of increasing β, [22] shows that higher levels of β can lead to more disentangled representations of the data in the case of images.

2.4.3 Supervised *contrastive* learning

We construct a model to explicitly maximize heritability. Contrastive representation learning aims to construct an embedding such that similar observations are close (for instance, euclidean distance) and dissimilar observations are pushed far away. Commonly, contrastive learning is used for unsupervised approaches. These approaches leverage data augmentation to create similar observations. The loss, therefore, takes pairs of embeddings and their labels (similar or dissimilar) and penalizes embeddings that are close to dissimilar observations. We use this approach but, instead of only constructing similar representations through data augmentation, we use the genotype as the label. We closely follow [28], but adapt their approach for the task at hand.

We aim to cluster together observations with the same genotype together to explicitly maximize the heritability of the extracted traits. Once we have done so, we can use these traits in GWAS or reverse-GWAS. The usage of this representation in reverse-GWAS is closely related to the traditional use of contrastive learning as a pre-training strategy. In this case, however, the classes to eventually use for

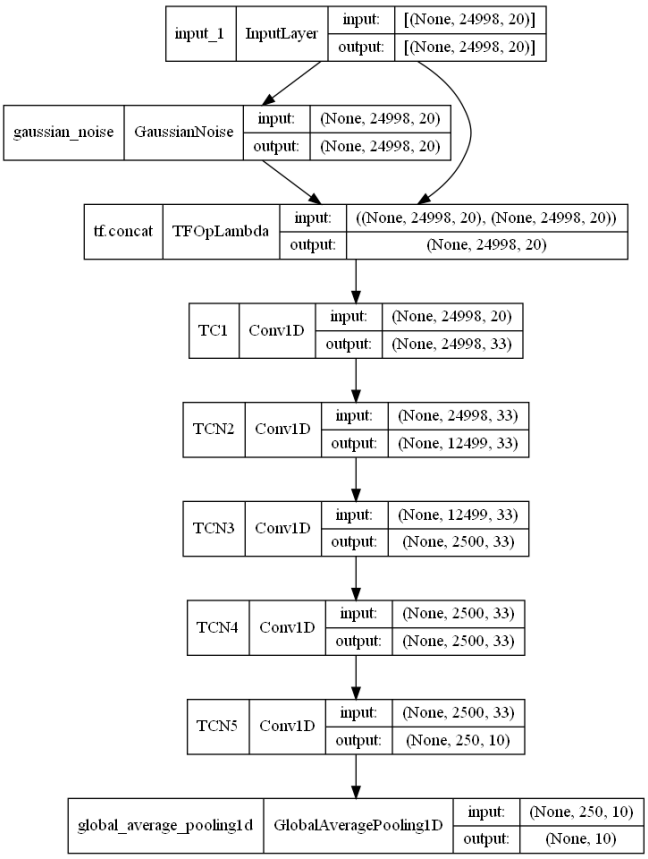


Figure 18: Simple TCN architecture for supervised contrastive learning.

classification (individual marker score) are not the same as the ones used for pre-training (genotype). Nevertheless, we hypostatize that the features detected to maximize heritability should help in predicting the marker score.

The main components of this framework are the data augmentation, the encoder and the loss function:

Data augmentation: It consists of the generation of a similar copy of each observation in our batch. In order to do so, we apply Gaussian noise to the batch and concatenate (over the batch dimension) the noisy batch with the original one, obtaining a "multiviewed batch" of size $2N$ for N being the number of original observations in the batch. This step is especially important in our case, given the relatively high quantity of classes (~ 350) and the limited memory. The probability of having the same class at least 2 times in a batch of 128 observations is less than 30%. Data augmentation allows for generating similar pairs in the same genotype, guaranteeing that there is at least 1 pair with the same class per observation and batch. In consequence, our approach is not far away from self-supervised contrastive learning.

The encoder: Again, we utilize a simple TCN (see Figure 18). The architecture is the same as the one used in the self-supervised approach. The global average pooling at the end of the network generates the vector representation in the desired number of dimensions, given by the number of filters in the last convolutional layer. The encoding is normalized using l^2 normalization, which transforms the representation into a unit hypersphere. The only regularization strategies used are early stopping and the relatively low dimensions of the encoding.

The loss: The loss function is designed to encourage high inner products between representations of the same class and low ones for representations of different classes. This encourages the network to find a representation where the observations in each class have a similar encoding (See figure 19). Naturally, this directly targets the maximization of broad sense heritability. Consequently, the risk of over-fitting

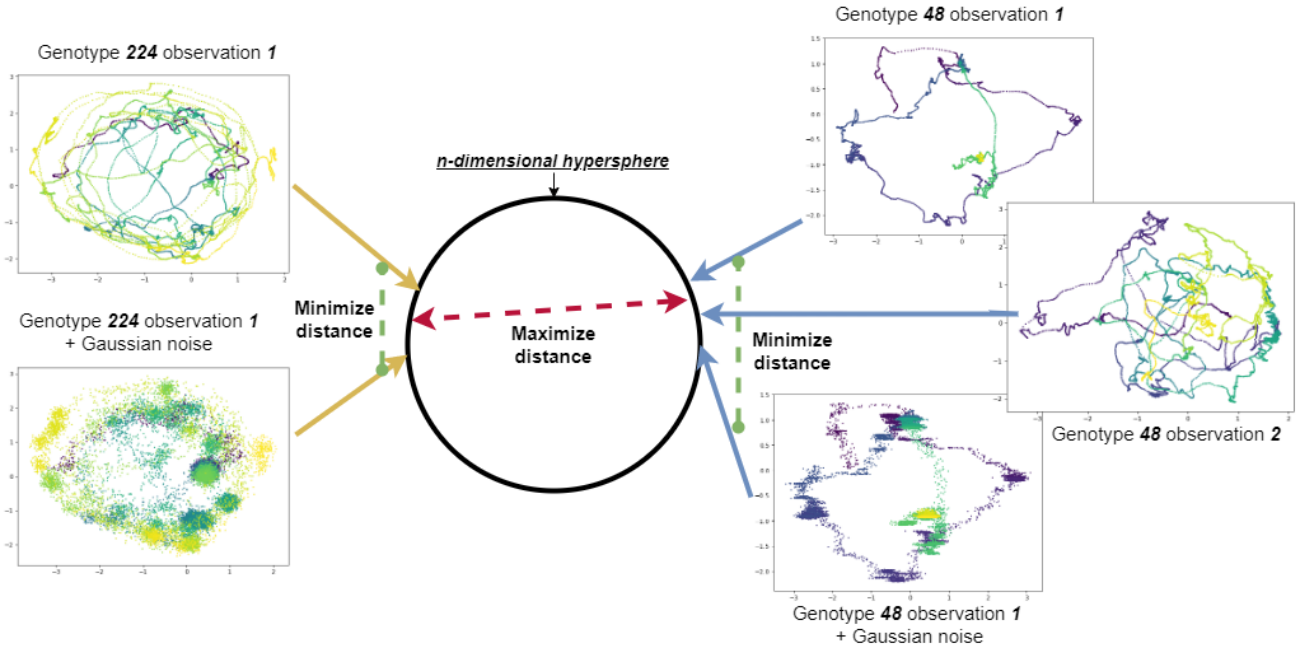


Figure 19: Supervised contrastive learning aims to group observations of the same class allowing a simpler classifier, such as logistic regression, to easily classify the data correctly. The training process uses data augmentation to ensure at least one pair of the same class per batch. Note that, for ease of interpretation, the images represent just the normalized x, y and time dimensions of the actual 20 dimensions used.

must be taken into account. We need good clustering performance in unseen data to consider the generated traits related to the actual genetic variability.

The loss function for this approach proposed by [28] is:

$$Loss = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a / \tau)})$$

Where $i \in I$ are the observations in the "multiviewed batch", $P(i)$ is the set of all pairs with the same class, $j(i)$ are the observations in I with the same genotype as the particular i, $A(i)$ are the ones with different genotype, z is the vector representation or encoding of each observation, and τ constant scalar called "temperature".

The hyperparameters for this approach are the "temperature", the number of filters and the ADAM optimizer hyperparameters. For the sake of simplicity, we maintain the same kernel size and stride size as in the self-supervised approach and fixed the number of dimensions for the representation to 10.

2.4.4 Hyperparameter tuning

The hyperparameter tuning is implemented through "AutoML", specifically the **Keras tuner** software [37]. This framework approaches hyperparameter selection as the optimization of an unknown complex function. This unknown function $f(h)$ can be understood as a "black box" which returns the performance of our neural network given a set of hyperparameters h. Removing the human from the loop avoids the tedious process of manually tuning the hyperparameters and possible human bias.

We use the Bayesian optimization framework with an underlying Gaussian process. Bayesian optimization works by maintaining a probabilistic belief about the unknown function and designing a surrogate function (acquisition function) to determine where to evaluate the function next. This belief is updated through successive iterations, optimally exploring the hyperparameter space. Figure 20 illustrates this process for a two-dimensional hyperparameter space.

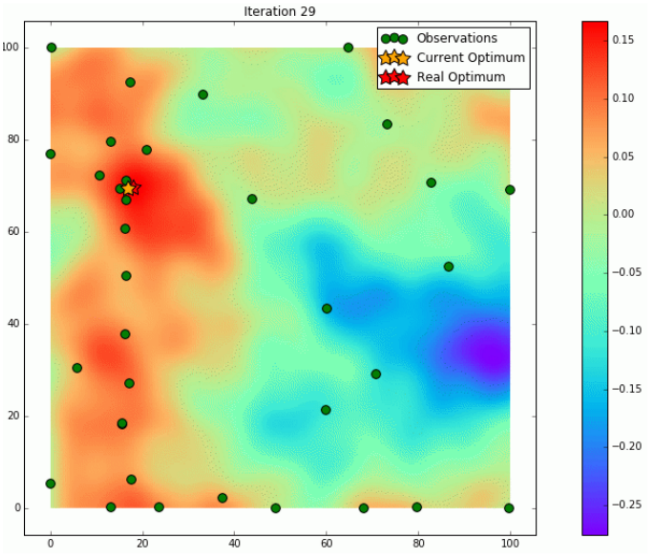


Figure 20: Successive iterations construct a surrogate function that gives a prediction (and confidence bound) of the underlying unknown function in the hyperparameter space. This example depicts an optimization procedure with 2 hyperparameters to tune. The colour corresponds to the approximated performance of the network. Image obtained from: Data Science Stack Exchange (user: stmax), Optimizing parameters for a closed (black-box) system, 2019.

For the supervised models, the objective is as defined minimizing the loss (cross-entropy) in the validation set, the optimization process is set to 50 iterations and 50 epochs. For the self-supervised, the objective is defined as the mean absolute error in the validation set and the exploration consists of 100 models with 250 epochs each. And for supervised contrastive learning, the objective is the contrastive loss in the validation set, the process is set to fit 100 models with 80 epochs each in this case.

To reduce the hyperparameter space, the optimizer is kept fixed and we only use adaptative learning moment estimation (ADAM) [29]. Different data splits to use the same hyperparameters.

2.4.5 Training process and feature extraction

With the optimally selected hyperparameters, training is programmed with early stopping based on the performance in the validation set to avoid over-fitting[8]. We store the number of epochs until early stopping is activated. With this information, we train again the model from scratch using both training and validation data for the registered number of epochs.

The idea underlying this process is to get the optimal amount of regularization through the early stopping procedure and then use it to prevent over-fitting when training without a validation set. This allows us to make more efficient use of the data.

Once the model has gone through the training process, it is time to assess the performance and retrieve the hidden representation, the new traits. To do so, a new model, maintaining the trained parameters, is constructed. The output of this model is the desired layer where the traits are ultimately generated. Then, the whole data is concatenated and passed through this model. This outputs the matrix of our new traits, as seen in Figure 3.

The case of transfer learning with Xception is a bit more complex. The model is pre-trained in ImageNet, therefore we do not use the top of the network. We only use the representation it creates. To train the pre-trained model we need to first freeze (block parameter updates) the imported model while training a top part classifier for our binary classification task. This top part is just a dense layer with

[8]Indeed, "early stopping has the effect of restricting the optimization procedure to a relatively small volume of parameter space in the neighbourhood of the initial parameters" Goodfellow et al.(2016) [18]

two nodes, as in the other models.

The process for self-supervised models is similar, however, since predictive performance is not the aim we do not use a test set. In that case, after fitting the model with the training and validation set we can directly extract the latent dimensions.

2.5 Trait quality assessment

We use two metrics to assess the quality of the generated traits. We use broad-sense heritability (H^2) and explained variance (R^2).

Variance explained is defined as the $1 - \frac{\text{RSS}}{\text{TSS}}$. Where RSS are the residual sum of squares and TSS is the total sum of squares. For a model $Y = \beta X + e$ where Y is the vector trait and X the design matrix with a vector of 1s and (categorical) explanatory variable, this is defined as $R^2 = 1 - \frac{e^t \cdot e}{(Y-\bar{Y})^t \cdot (Y-\bar{Y})}$. In our case, the explanatory variable is a binary vector corresponding to SNP 86001.

Broad-sense heritability is defined by [34] as:

$$H^2 = \frac{\hat{\sigma}^2_G}{\hat{\sigma}^2_G + \hat{\sigma}^2_{\text{Env}}} \text{ , with: } \hat{\sigma}^2_G = \frac{MS(G) - MS(\text{Env})}{r},\ \hat{\sigma}^2_{\text{Env}} = MS(\text{Env}) \text{ .}$$

Which is derived from a simple analysis of variance where $MS(G)$ are the mean sum of squares for genotype and $MS(\text{Env})$ the residual error.

3 Results

3.1 EthoAnalysis Features

Here we use the features extracted from the EthoAnalysis software, the handcrafted traits. Some of these traits can detect the gene around SNP 86001 in a GWAS.

3.1.1 Trait quality

Heritability and explained variance are considerably low. R^2 for SNP 86001 does not achieve more than 1%, H^2 is below 3% . Constructing traits whose variability is well explained by the genotype is indeed difficult in this dataset.

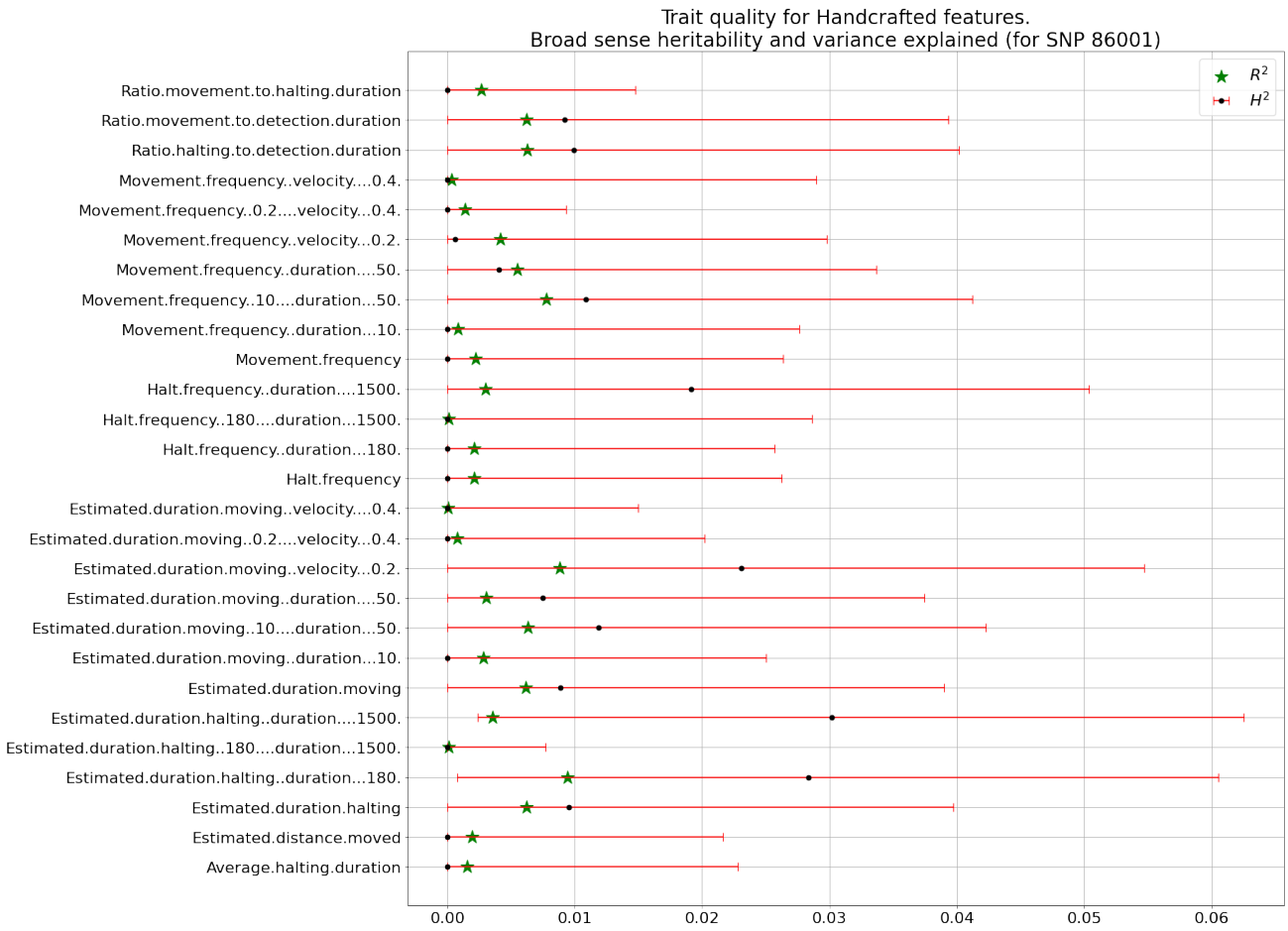
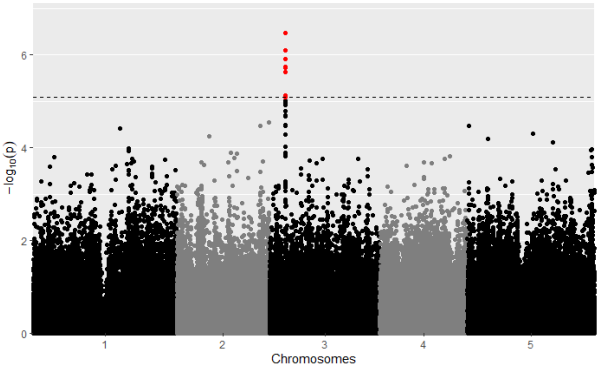


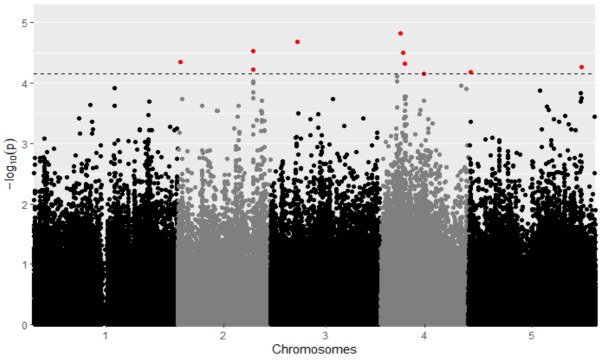
Figure 21: Variance explained for ANOVA using SNP 86001 as a regressor (R^2) and broad-sense heritability (H^2) calculated for each feature. The correlation between the two is quite high, around 70%.

3.1.2 GWAS

Estimated duration halting < 180 has the maximum R^2 0.94% and estimated duration halting between 180 and 1500 has the maximum H^2 3% (see Figure 22). R^2 clearly indicated a trait explained by SNP 86001, as is natural to expect. Heritability, on the other hand, did not display clear SNPs in the area of 86001. However, it displays some interesting SNPs at the beginning of the 4th chromosome.



(a) "Estimated duration halting (duration $<$ 180)"



(b) "Estimated duration halting (180 <= duration $<$ 1500)"

Figure 22: Features with maximum R^2 and H^2 respectively.

3.1.3 Reverse GWAS

Figure 23 displays the performance in the test set of SVM with default hyperparameters on the features extracted by EthoAnalysis. To aid interpretation, values lower than 0 are transformed to 0. There seems to be a clear peak around SNP 132093.

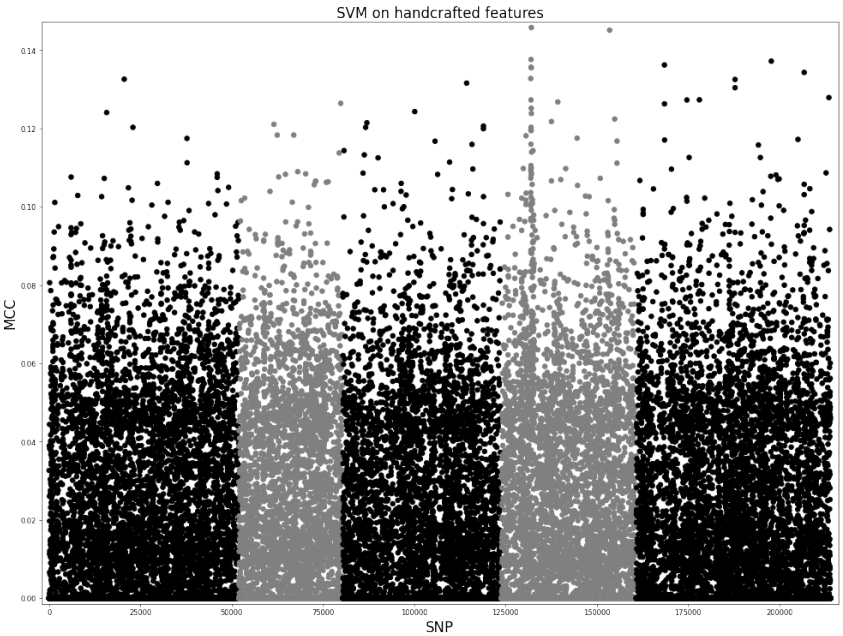


Figure 23: Genome-wide reverse mapping using SVM on all the handcrafted features. There is a clear peak around SNP number 132093 in chromosome 4 with a maximum MCC of 0.1457. The gene associated for this SNP is AT4G07380.

3.2 Supervised learning

We train 4 different architectures. Full Inception Time, small Inception Time (uses just speed and downsampling by a factor of 10), Transformer and Xception. We do so for 3 different types of data splitting strategies. The splitting strategies are random splitting, stratified by genotype and forcing the genotype to be different in all splits. Naturally, we expect the last one to be the most challenging, and it is the one with the worst performance. Stratification for genotype is forced to have fewer observations in the training set since validation and test must also have all the genotypes.

Due to the random nature of neural networks, it is sensible to evaluate performance with different parameter initializations. Therefore, we fit each model 5 times. This is also recommended by the authors of Inception Time [24]. However, instead of constructing an ensemble, we select the best model. We are not as interested in predictive performance as we are in the generated representation.

3.2.1 Selected hyperparameters

As explained before, the hyperparameters are determined algorithmically. We only use an heuristic approach for Xception.

Full Inception Time. The depth is kept at 1, with a dropout rate of 0, learning rate of 0.001 with β_1 of 0.2 and β_2 of 0.999, kernel size of 16, 20 filters and a bottleneck size of 5.

Small Inception Time. The depth is kept at 1, with a dropout rate of 0.15, learning rate of 0.001 with β_1 of 0.2 and β_2 of 0.39, kernel size of 34, 128 filters and a bottleneck size of 5.

It is not surprising to see that the hyperparameter optimization process tend to select rather shallow networks for TCN.

Transformers. Four heads with size 244 and 4 transformer blocks, dropout is 0.15, feed forward layer has 22 units, learning rate of $1.35 \cdot 10^{-6}$ with β_1 of 0.54 and β_2 of 0.62.

Xception. Being an already trained network, the only tuned hyperparameters are optimizer ones. Those are selected by hand. Learning rate of $3 \cdot 10^{-6}$ with β_1 of 0.9 and β_2 of 0.999.

3.2.2 Model performance

Figure 24 depicts the performance in the stratified dataset.

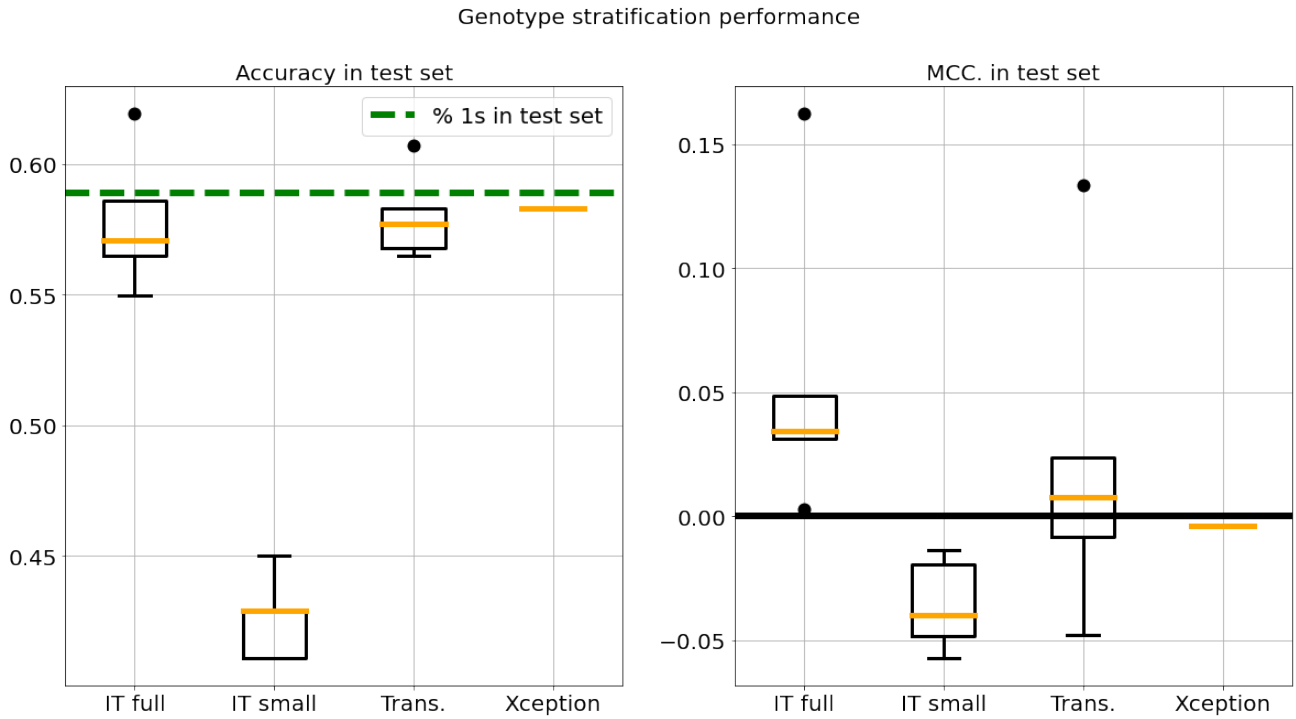


Figure 24: 'IT full' stands for Inception Time using all the features, 'IT small' uses just speed and a down-sampling of 10 steps, 'Trans' stands for transformer. The best results are obtained when the data split is made stratifying by genotype. This means that the same genotype is present in test, validation and train. We will be using the representation obtained with the best model in this type of splitting strategy. It is also the best model overall.

Inception time with all the features seems to perform the best and will be the model from which the features are extracted. It achieves an accuracy of 62%, MCC around 0.16 and AUC-ROC of 0.58. More classification metrics can be seen in table 1.

	precision	**recall**	**f1-score**	**support**
1	0.616162	0.938462	0.743902	195
0	0.647059	0.161765	0.258824	136
macro avg	0.63161	0.550113	0.501363	331
weighted avg	0.628857	0.619335	0.544595	331

Table 1: Classification report for SNP 86001 using the best full Inception Time. It is worth noticing that the recall for the allele frequency 0 is quite low. This means that the network tends to predict 1 most of the time. This is expected due the to the lack of signal and the higher presence of allele frequency 1 (approximately 59%) in train set.

It is worth mentioning that Xception achieves similar performance in the no-stratified dataset. This is surprising due to the almost out-of-the-box usage of the representations created by a network trained

in ImageNet. This indicates that framing this kind of trajectory problem as images may be a good approach, furthermore, leveraging through transfer learning is ideal for settings with not that many observations. Arguably, the lower performance of Xception in the stratified setting is due to the smaller train set.

3.2.3 Trait quality

Both the maximum heritability and variance explained are higher than the maximum in the handcrafted features. The average R^2 over all the generated traits is lower but the average H^2 is higher than the average for the handcrafted ones. However, the differences between the two are really small. The best H^2 is in dimension 57 with 4.4% and the best R^2 is 1.2% for dimension 70.

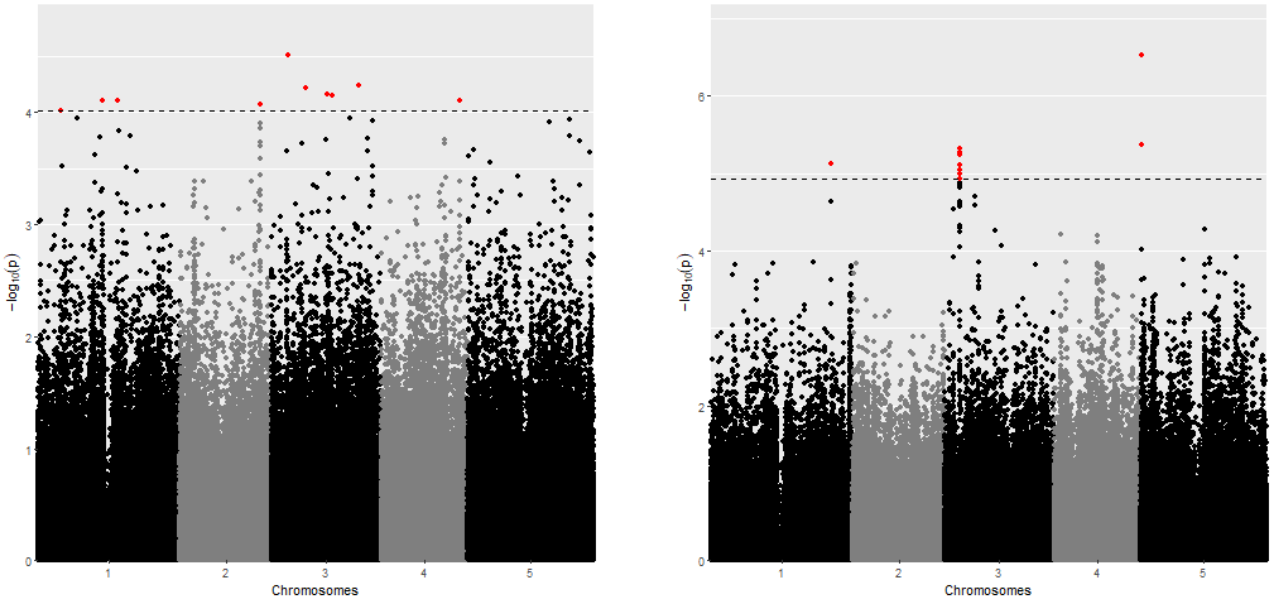
Figure 25: Variance explained for ANOVA using SNP 86001 as a regressor (R^2) and broad-sense heritability (H^2) calculated for each feature. The correlation between the two is around 60%.

The difference between the weights in each node of the last layer indicates the strength of an increase in the predicted probability of marker score 1. This difference is positively correlated with H^2 and R^2

with a correlation coefficient of 0.22 and 0.11 respectively. The dimension with the highest marginal contribution to the increase in the probability of allele frequency 1 is dimension 33.

3.2.4 GWAS

Figure 26 illustrates the GWAS for two of the 80 available dimensions. The dimension 70 which maximizes the R^2 shows a clear peak around the gene related with SNP 86001, being SNP 86001 itself among the 10 more signifficant. Dimension 57 with maximum heritability, however, does not show a really clear pattern.



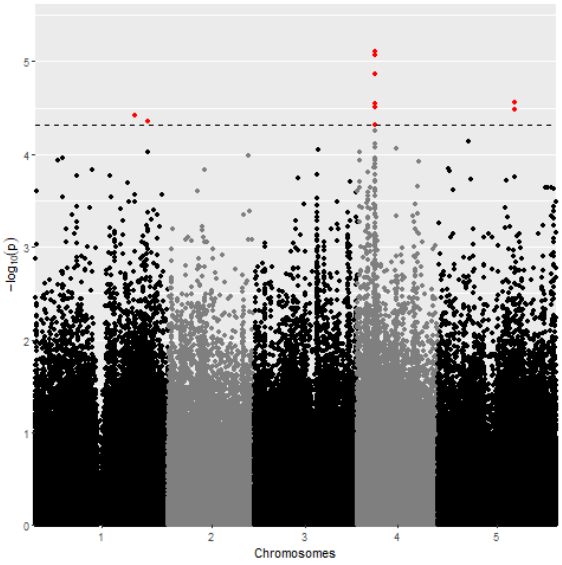
(a) Dimension 57 **(b)** Dimension 70

Figure 26: The best performing dimensions in H^2 and R^2 respectively. As expected, dimension 70 has a clear peak around SNP 86001.

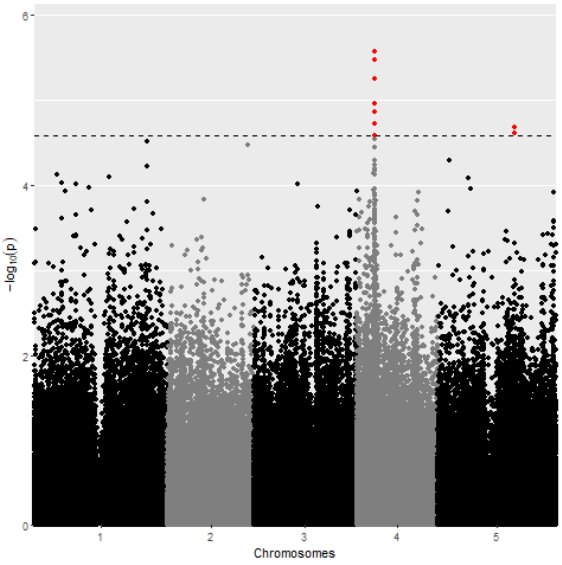
Additionally, there seems to be a peak around SNP 131996, particularly for dimensions 0, 4, 6, 24, 15 and 32 (among others) SNP 131996 is among the 15 most significant SNPs. Interestingly enough, this is the same peak observed in reverse GWAS with the handcrafted features (see Figure 23). In that case, SNP 131996 has the 59th position in test set performance (MCC), and SNP 131993 (considerably close) the 6th position.

Figure 27: Dimensions with clear peaks around SNP 131996.

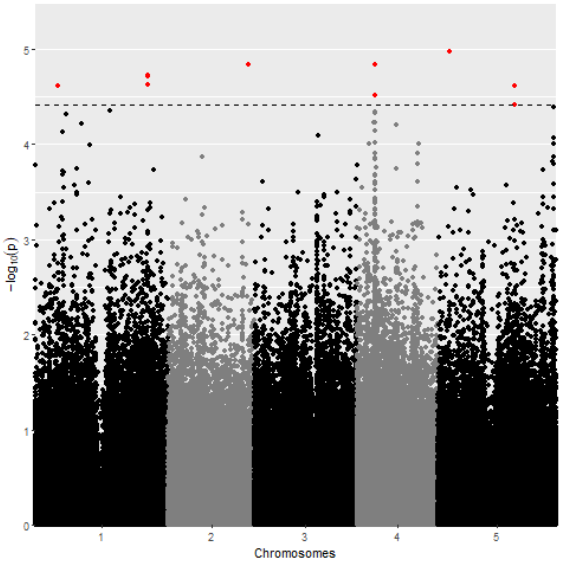
(a) Dimension 0



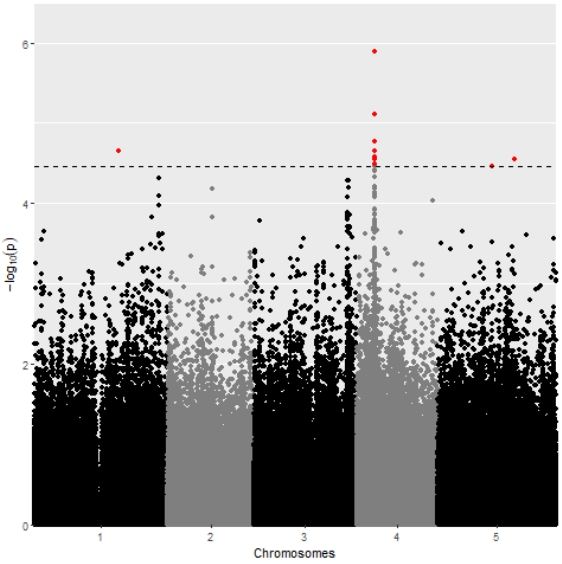
(b) Dimension 4



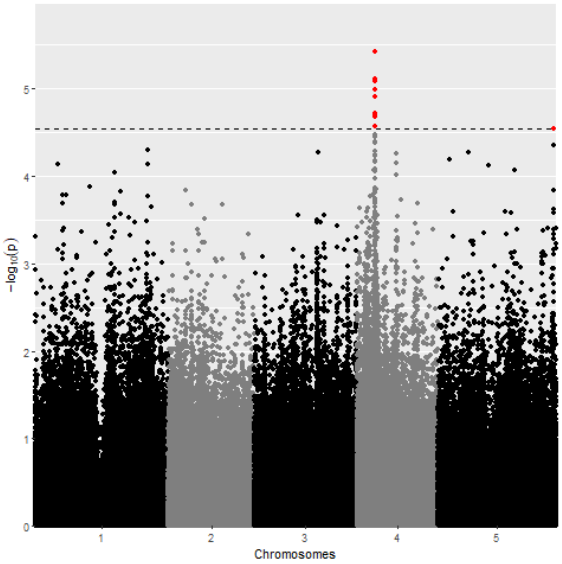
(c) Dimension 6



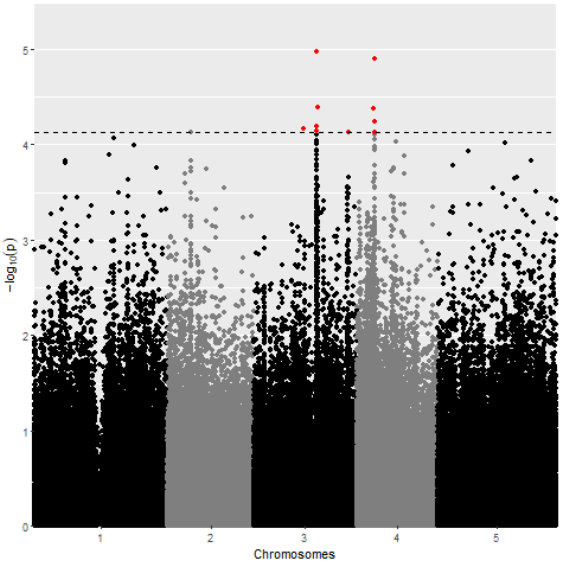
(d) Dimension 15



(e) Dimension 24



(f) Dimension 32



3.2.5 Reverse GWAS

The genome-wide reverse mapping is computed using SVM with default hyper-parameters on the extracted 80 traits. It does not show any particular interesting area in this case.

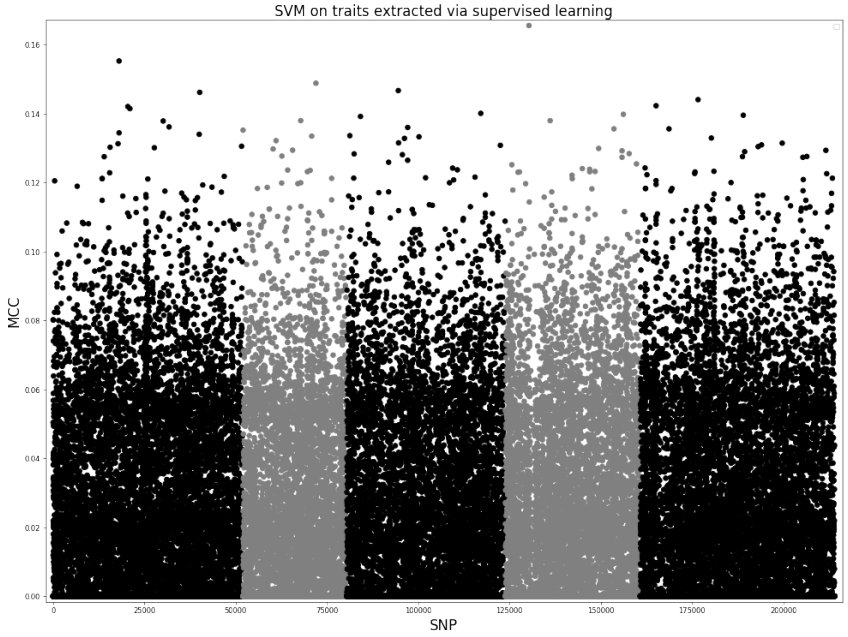


Figure 28: Predictive performance in test set for all SNP using all traits and SVM. There is no clear area with better performance than the rest. The maximum is located at SNP 130338 with an MCC of 0.1655.

3.2.6 Correlations with EthoAnalysis features

There are no relevant correlations with the handcrafted features. This shows that the traits that maximize the predictive performance do not need to be related to the ones extracted using biological reasons. The correlations go from -0.06 to a maximum of 0.08 (Figure 29).

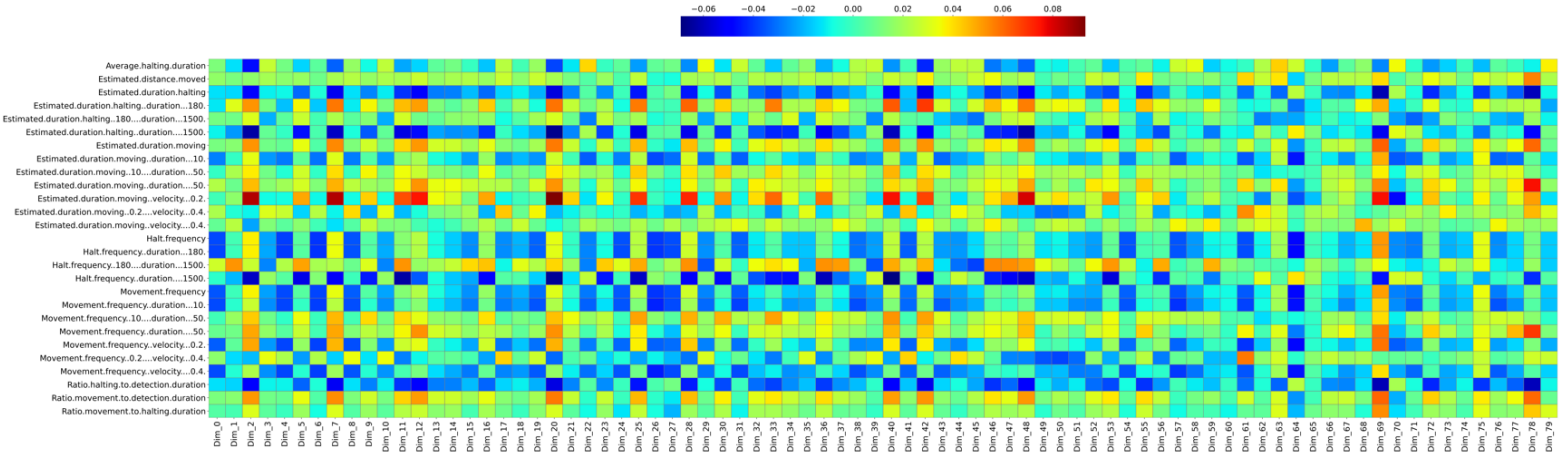


Figure 29: Correlations within dimensions generated through deep supervised learning and EthoAnalysis' extracted features.

3.3 Self-supervised learning

The autoencoder is able to reconstruct in some extent the speed dimension, however, it does not show disentangled representations (evaluated with H^2). Nevertheless, while being an unsupervised approach it does show similar performance as the handcrafted and supervised approach.

3.3.1 Selected hyperparameters

Hyperparameter	Value
β	0.001
Learning rate	0.001
opt. β_1	0.502
opt. β_2	0.999
n. filters	23

Table 2: Hyperparameters selected to construct our self-supervised model. β is the KL divergence importance parameter and β_1, β_2 parameters of the optimizer.

Using higher β such as 2, 6 or 10 did not lead to more disentangled representations, evaluated with H^2, and worsen reconstruction performance.

3.3.2 Model performance

Figure 30 shows how the mean squared error of the reconstruction for validation and training data decrease together. The quality of the reconstruction of the speed dimension by the decoder is considerably below a trivial reconstruction using the mean 0 for all arenas at all steps.

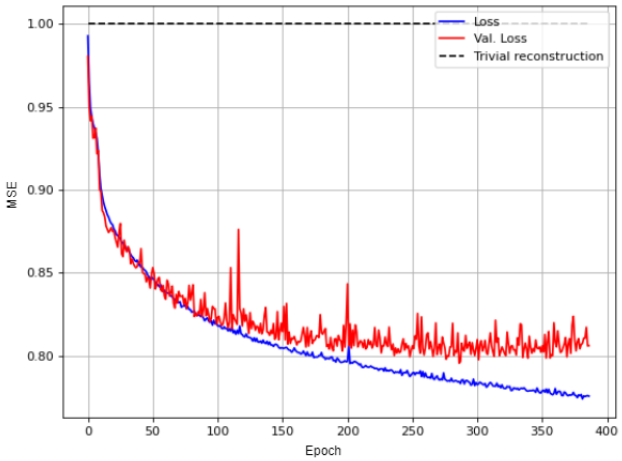


Figure 30: First training with hold-out validation set. Trivial reconstruction is the loss when the output is just the mean speed for all steps. Since the series are standardized this is equal to the standard deviation, which is 1.

We can assess model performance also visually, evaluating how the decoder reconstructs the speed dimension. Figure 31 shows 9 examples where it can be appreciated that, while not recovering all the variability, the decoder can reconstruct some moving average of speed. This indicates that our autoencoder can capture some relevant factors of variability.

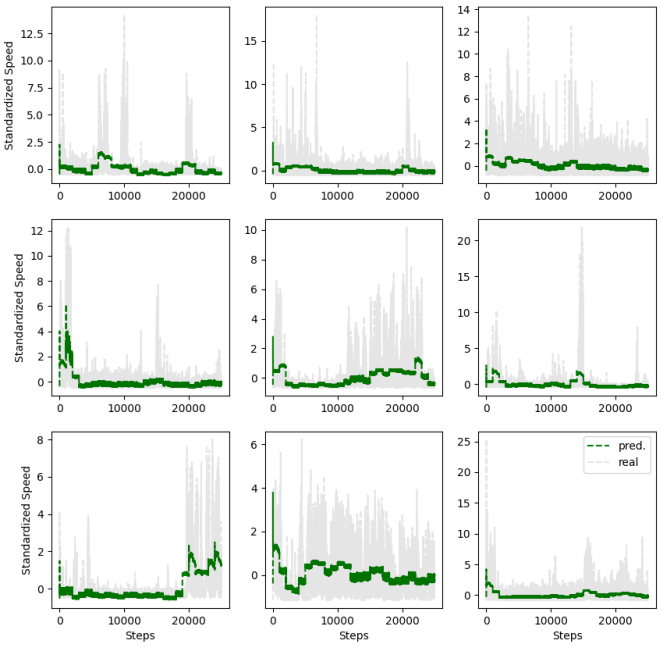


Figure 31: Reconstruction of the speed of some trajectories from a 25 dimensional posterior. The reconstruction tends to an average of speed conditional on the step, however, it is better than an unconditional mean.

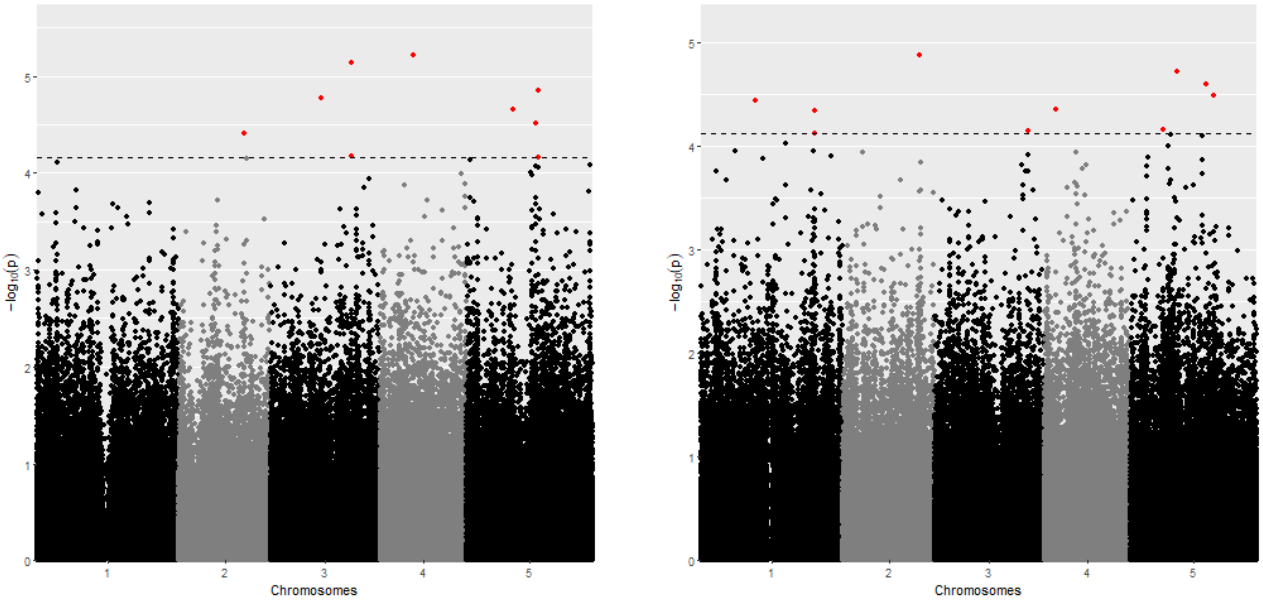
3.3.3 Trait quality

Heritability and variance explained for SNP 86001 are very low, which a maximum for H^2 of 3% for dimension 27 and a maximum for R^2 of 0.4% for dimension 21. This is not very different from the performance obtained in the previous approaches, however, this is a completely unsupervised approach.

Figure 32: Trait performance for the Variational Autoencoder. The first 25 dimensions are the parameterization of the mean , the last 25 the parameterization of standard deviation. The input for the decoder is a 25 dimensional vector per observation.

3.3.4 GWAS

Due to the nature of the autoencoder, the latent representation is not necessarily linearly disentangled. This means that linear approaches such as GWAS are not most likely not suitable for this representation. The relevant information contained in the latent representation is probably contained in particular interactions between the latent dimensions that the highly non-linear decoder can successfully interpret (see Figure 31).



(a) Dimension 21 **(b)** Dimension 27

Figure 33: The best performing dimensions in H^2 and R^2 respectively

In this case, SNP 86001 is not recovered for the dimension with maximum R^2.

3.3.5 Reverse GWAS

In this case, the SVM is fitted on the means and standard deviations that parameterize the diagonal 25-dimensional normal that constitutes the latent representation. Reverse GWAS with a non-linear classifier is more suitable for this type of embedding due to the inherent non-linearity of the latent representation of the VAE. There are some performance peaks, interestingly it seems to be a mild peak around SNP 131996 (the same area in which peaks were found before). Figure 39 shows that the peak, while not very high and very broad is again around the area we have found before.

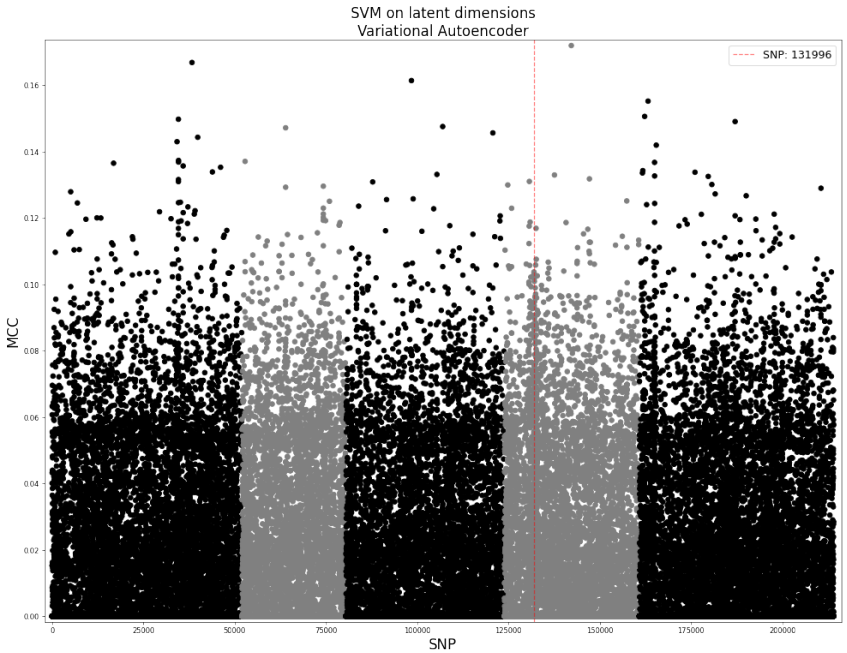


Figure 34: Predictive performance in test set for all SNP using SVM on the parameterization of the latent dimension of the autoencoder.

3.4 Supervised *Contrastive* Learning

This model attempts to separate arenas given a genotype. The loss directly targets the maximization of H^2 by generating dimensions where the arenas with the same genotype tend to cluster together. It results in a much better H^2 than the previous approaches. However, it is important a good control of overfitting. The model must be able to map characteristic aphid behaviours to each genotype, and not just separate the data due to non-genetic variability. Consequently, we leave out a test set in order to evaluate the performance of the model in clustering unseen data. It is important to notice that, differently from classical supervised learning the model does not consider specific genotypes. It only assesses pairs of the same or different genotype, disregarding the actual genotype value.

3.4.1 Selected hyperparameters

Hyperparameter	**Value**
τ	0.01
Learning rate	0.001
opt. β_1	0.524
opt. β_2	0.584
n. filters	33

Table 3: Hyperparameters selected to construct our supervised contrastive model. τ is the "temperature" parameter and β_1, β_2 parameters of the optimizer.

3.4.2 Model performance

Since the loss function is directly targeting heritability, we need to make sure that the model is not trivially over-fitting the data. To guarantee that, we leave a test set and evaluate the performance before and after training. The loss value before training for the training + validation set is 0.951 and 0.90 for the test set. After training it becomes 0.175 and 0.168 respectively. This indicates that the model is, in the best case, indeed able to recognize different aphid behaviours explained by the genotype. In the worst case, this could be due to a confounding effect arising from the experimental design or another similar unknown source.

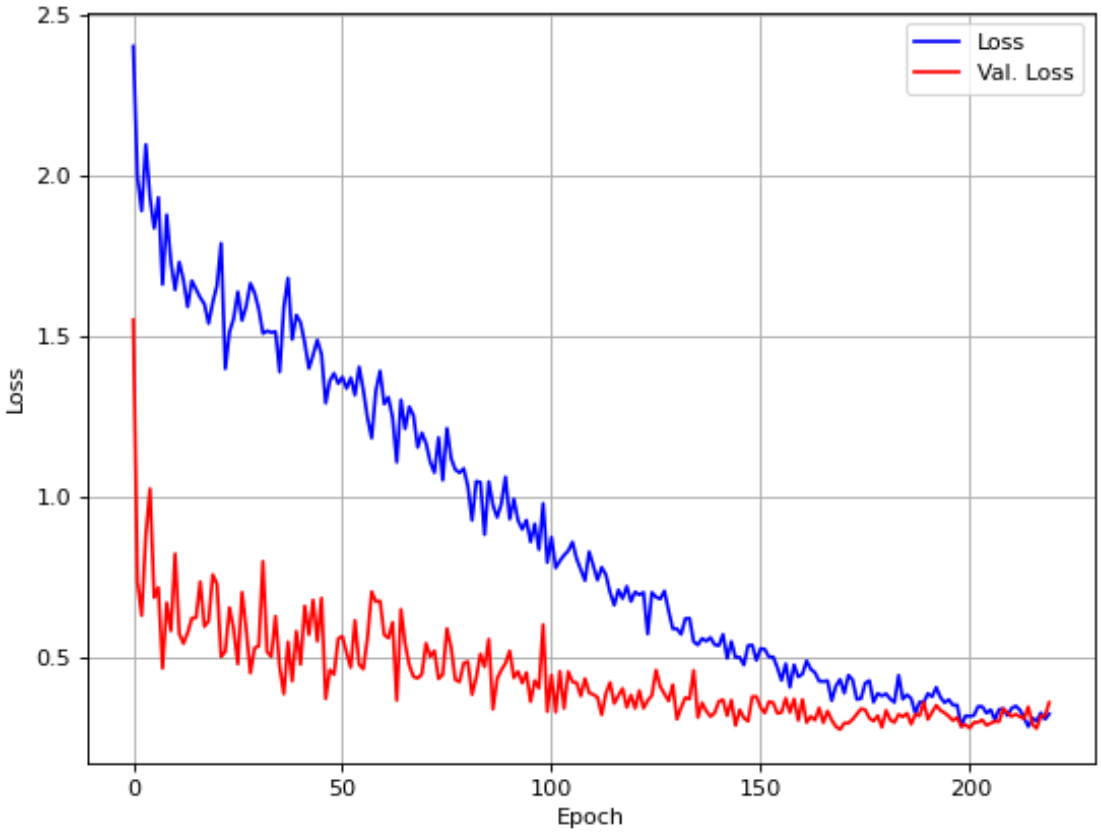


Figure 35: Evolution of validation and train sets during training. Gaussian noise is disabled in evaluation, consequently, the validation loss tends to be lower at the beginning.

Figure 35 is illustrative with respect to the generalization performance of the network. Training loss and validation loss tend to converge. Validation loss is lower because the Gaussian noise layer is disabled outside training, this is the same behaviour that dropout layers have.

We hypostatize that the lower dimensionality of the encoding, the modest number of parameters and the data augmentation contribute to the good generalization performance of the model. However, we leave the further investigation of this behaviour for future research.

3.4.3 Trait quality

Heritability values are much higher with the supervised contrastive learning approach. With a maximum H^2 of 50% for dimension 8 (see Figure 36). While this is much higher than the other approaches, it is not completely surprising. In this case, the genotype information has been directly used to construct the model. Nevertheless, this method outperforms the handcrafted approach and the other automated methods for the extraction of traits with high heritability.

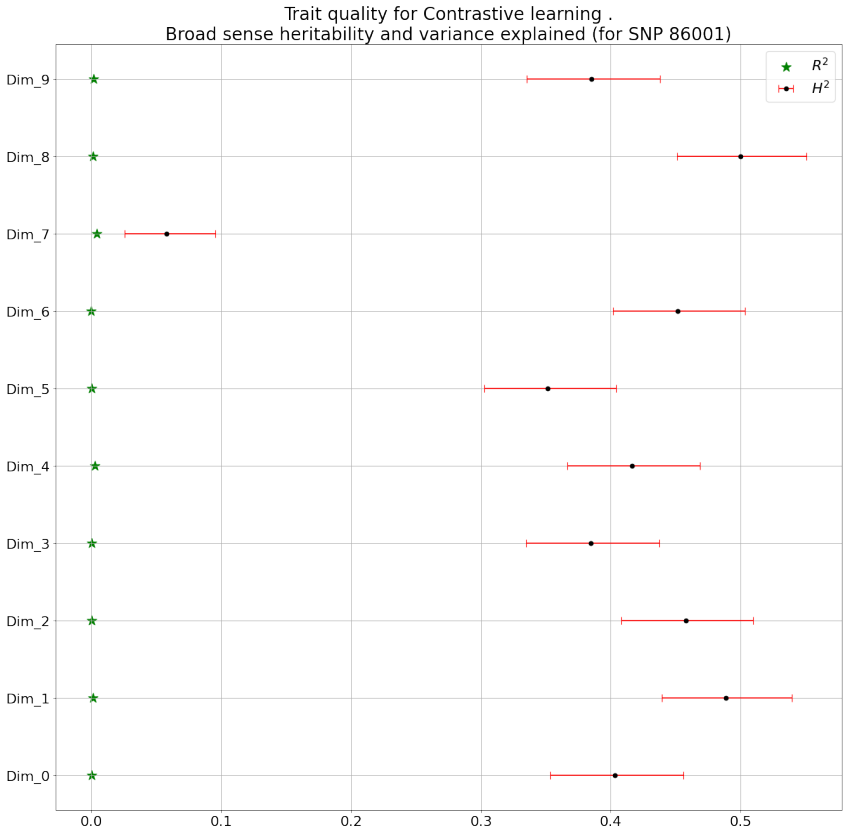


Figure 36: H^2 and R^2 for SNP 86001 in the representation obtained using supervised contrastive learning.

We have observed that increasing the number of dimensions in the encoding diminishes the heritability of each of the dimensions. This indicates that the bottleneck created by a low-dimensional representation can be useful to increase the heritability of the traits and possibly the generalization power of the network.

R^2 for SNP 86001 is low, but not very different from the other approaches, having a maximum of 0.04% for dimension 7.

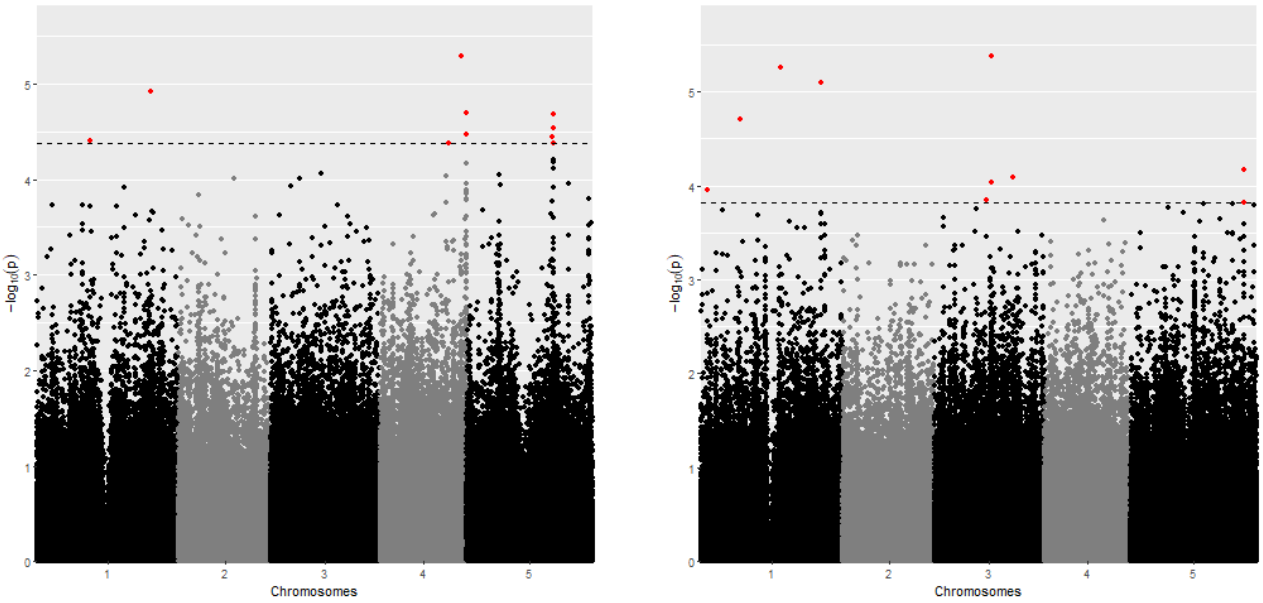
3.4.4 GWAS

Figure 37 shows the dimensions with higher R^2 for SNP 86001 and H^2. SNP 86001 is in position 2277th in significance level for the dimension that maximizes the R^2.

Dimension 6 (see figure 38), has a remarkable high peak for SNP 91737 with the lowest p-value so far.

3.4.5 Reverse GWAS

Reverse GWAS follows directly from contrastive learning as pre-training for the classification task. Overall performance in unseen data is much higher than in previous approaches. The maximum MCC on the test set is for SNP 25290 with 0.35 and an accuracy of 68%. In fact, out of the shelve SVM on the contrastive learning features outperforms our previous approach using supervised learning with an MCC of 0.207 and an accuracy of 63.4%. However, we believe that the increase in general performance may be due, to some extent, to confounding due population structure. While we expect that, if a particular SNP is driving the behaviour of the aphid, the representation tends to cluster together the genotypes with this SNP, this is not directly enforced in the loss function. The effect of relevant SNP could be influencing the predictive performance of other SNPs. Again, we have tried to control the model generalization (to which extent the model learns actual relevant behaviours), however, it is not clear to which extent the model is able to cluster by behaviour and not by genotype. Notwithstanding,



(a) Dimension 8 **(b)** Dimension 7

Figure 37: The best performing dimensions in H^2 and R^2 respectively

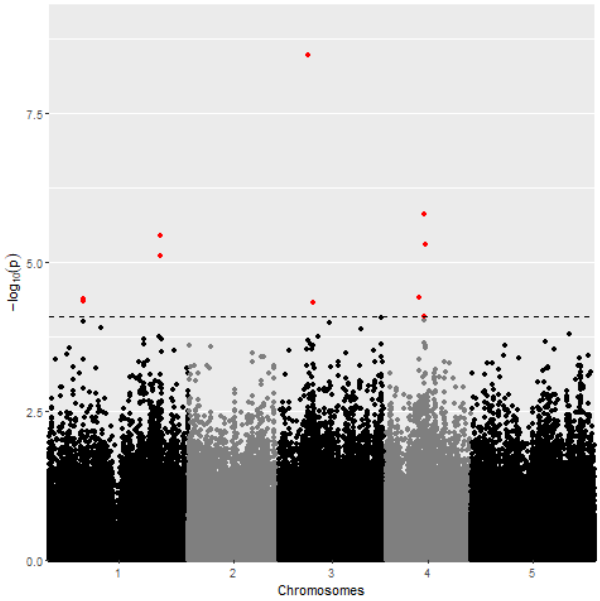


Figure 38: Dimension 6, with a peak at SNP 91737 of $-\log(p) = 8.5$

while acknowledging the potential confounding is fundamental, the increase in performance for a relevant SNP is still remarkable.

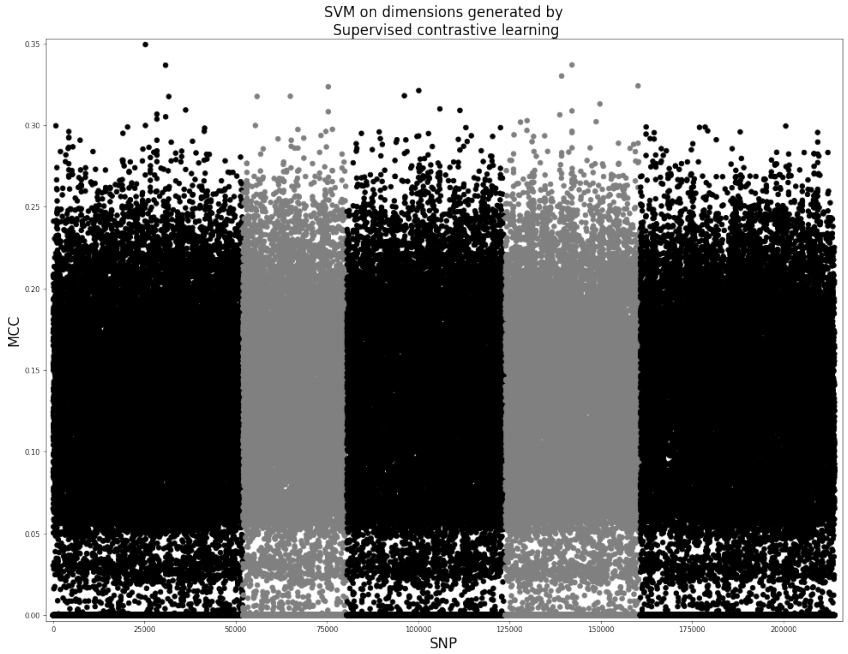


Figure 39: Predictive performance in test set for all SNP using SVM traits generated using supervised contrastive learning.

4 Conclusion

In this thesis, we have investigated the potential of deep learning methods to automatically extract meaningful traits from highly complex phenotypic data. Specifically, we have done so by exploring Arabidopsis resistance to aphids with data extracted from a high-throughput phenotyping platform. Instead of extracting the traits through biologically meaningful statistics, we set a very flexible optimization procedure to come up with the best ones with respect to a particular objective function, expressed in the supervised and self-supervised approaches. We fitted different deep learning models in order to extract meaningful low-dimensional representations of the aphid trajectory and we treated it as our phenotypical data. Our initial hypothesis was that automating the trait generation in the phenotyping process can outperform the quality of the traits extracted by the experts.

Performing GWAS on this generated low-dimensional representation we explored new candidate SNPs driving aphid resistance. Furthermore, we have explored a non-linear multi-trait approach to genetic mapping based on the idea that good predictions in unseen data imply a relevant signal. We used SVM and assess the performance through MCC in unseen data.

We fitted a supervised model assuming that "good" performance predicting correctly the binary marker score for a relevant SNP in explaining aphid behaviour would lead to a good low-dimensional representation of the data. While we are not directly interested in predictive performance, it is worth mentioning that it was remarkably high for the expected signal that a single SNP can have. While the quality of the extracted traits evaluated as R^2 and H^2 is, overall, better than the handcrafted ones, the difference is not significant. GWAS for the dimension with the highest R^2 could recover the SNP 86001 as the top significant SNPs, which is expected. We also found several dimensions with significant SNPs in other areas, around SNP 131996, not related to SNP 86001. Surprisingly, a peak in this area also appeared with the reverse-GWAS in the handcrafted and mildly in the VAE traits. Consequently, we consider that the SNP 131996 and the SNPs surrounding it are good candidates for further investigation. Due to the coincidence and important overlap in completely different approaches, we suspect that this may

indeed lead to a genuine gene driving arabidopsis resistance to aphid infestation.

With the best supervised models, we found that the correlation between the generated traits and the handcrafted ones is really low, with the maximum below 10%. This could mean that there are equally valid traits in explaining the effect of the SNP 86001 that target a different kind of behaviour or that the model is fitting another kind of non-genetical signal. A clear downside of "black-box" like algorithms is the lack of direct interpretation.

Our self-supervised approach using VAE was able to approximate the distribution of the data. The latent dimension lacked high genetic signal evaluated with R^2 (for SNP 86001) and H^2. This is not surprising, on the one hand, the non-linear decoder allows to generate any distribution, which does not necessarily imply a disentangled[9] latent representation. On the other, a self-supervised approach in this context relies on the assumption that the genetic signal is actually the main factor of variability in the data. The performance of self-supervised approaches towards our objective of genetic mapping depends on to which extent this is a reliable assumption.

Finally, the traits generated through contrastive supervised learning had high H^2. With similar performance as previous approaches in R^2 for SNP 86001. We believe that this approach is promising, heritability values of 50% are around 10 times bigger than the values obtained through EthoAnalysis. Furthermore, the lowest p-value in GWAS was found in one of the traits generated by it. One point to underline is the usage of data augmentation, which in this context is probably very useful for increased generalization. More interestingly, however, is the trade-off created between pushing away different genotypes and recognizing similar patterns of behaviour. To which extent the clustering is able to put closer different genotypes with similar behaviours is key to the success of this approach in detecting potential SNPs in GWAS.

In the introduction, we asked the following question: Is it possible to obtain better features than the ones extracted by the experts? While the supervised and self-supervised approaches did not display better performance than the handcrafted traits, contrastive learning did. Not just in terms of H^2 but also in terms of GWAS.

Nonetheless, it is key to acknowledge the important limitations of the approach. The nature of "black-box" algorithms such as deep neural networks makes it a challenge to interpret the results, for instance, to understand why a particular trait is relevant. This can have drastic effects, our models could be using signals that are not related to the task we are interested in. Additionally, the arbitrary number of generated dimensions poses a challenge due to the problems arising from the multi-trait GWAS such as multiple testing, the randomness inherent to the process and selecting the optimal number of dimensions. Reverse-GWAS methods can be useful here, especially when the traits are not disentangled. However, reverse-GWAS approaches also have drawbacks. In our case, we do not take into account the possible confounding effects due to population structure, or possible effects due to experimental design. Furthermore, the use of MCC without more consideration is problematic because with very unbalanced datasets it becomes harder to get higher values.

We believe that interesting future research could be directed to address these issues. Clear directions are: Developing frameworks for model interpretability in automatic trait generation for genetic mapping. Finding statistics to test the hypothesis of relationships between multiple traits and a specific marker with complex models could allow us to exploit entangled representations and pleiotropic effects, increasing the power of our tests. And the study of methodology that can be used to specifically target maximization of heritability in a deep learning framework such as (supervised) contrastive learning.

[9] A disentangled representation can be defined as one where single latent units are sensitive to changes in single generative factors, while being relatively invariant to changes in other factors [22]

5 Discussion

Technological developments in DNA sequencing and high-throughput phenotyping platforms have granted us the potential to study new meaningful patterns at a bigger scale than ever in plant science [10]. On the one hand, the advances in DNA sequencing have made the full genome sequence of many plants available. On the other hand, data extraction from plants in greenhouse or outdoor settings has been made significantly simpler by the availability of inexpensive cameras, data storage, and computing capabilities [13]. Examples of this data include leaf area, root volume, fruit size, chlorophyll content, photosynthetic activity, plant height, biomass, water usage efficiency, yield estimation or, as in this case, insect trajectory.

Nevertheless, this data revolution poses a challenge for traditional methods. A major challenge lies in extracting traits from high-dimensional raw data, such as the one typically obtained from high-throughput phenotyping platforms. However, while the available data continues growing, the frameworks to digest it are increasingly incapable to cope with it, leading to a phenotyping bottleneck. Treatment of phenotypical data "by hand" have deemed sufficient, but its limitation is increasingly noticeable. Manually engineered traits involve highly specialized labour and lead to ad hoc solutions that have limited usage beyond the group working on them [13]. Furthermore, in some complex cases, there is no guarantee that the extracted traits are close to optimal and it is possible that the best traits are beyond the expert's insight. While these handcrafted approaches have had success, for complex traits the limits of these methods have been reached. The dataset studied in this thesis is an example somewhat in the middle. In this case the handcrafted approaches have been successful but it is likely that automated approaches, such as the ones developed here, can lead to new discoveries. Clearly, there is a need for robust tools to extract optimally useful traits from complex data rapidly.

Traditional statistical approaches have also limitations in this new environment. Multi-trait analysis present great challenges in multiple testing and dealing with pleiotropic effects. This is a key issue when dealing with inherently multidimensional traits. Inside the traditional framework, there are several proposals to address these problems, some of them relying on reversing the mathematical relationship in genetic mapping, leading to reverse-GWAS (or reverse-QTL mapping) settings, for example, [15, 14, 38, 16]. However, when dealing with complex and multi-dimensional traits, high-order dependencies between the traits are beyond the scope of the traditional, linear, statistical approaches. For instance, in [38] it is acknowledged that; "There has been much effort to characterise well-defined phenotypes that may correspond more to specific biological function for use in genomewide association studies, but the definition of some phenotypes remains somewhat ad-hoc" however, their proposal is limited to a linear combination of the traits; "[...], we propose that linear combinations of directly measured phenotypes may often capture unmeasured aspects of complex biological networks, such as reaction rates, protein mediators or other uncharacterised or clinically-inferred phenotypes". Again, there is a need for a methodology to deal with the challenges of inherently-multidimensional and potentially non-linearly disentangled traits.

Machine learning approaches are promising for dealing with this quantitative and qualitative revolution in the available data, either for trait extraction or genetic mapping. For instance, in the case of trait generation, there is a broad collection of examples using deep learning for plant phenotyping [25]. As we have seen in this thesis, it is possible to generate better (in terms of H^2) representations using deep learning than using handcrafted methods. In this case, the proposed methodology is broadly usable for any kind of output from high-throughput phenotyping platforms. Nevertheless, expert input is still necessary to evaluate the outcome of the GWAS.

However, machine learning approaches are not free of important drawbacks. As we have seen in this work, machine learning applied to reverse-GWAS, while attractive in overcoming all the above mention challenges has clear problems that need to be addressed. Notwithstanding, we believe these problems do not invalidate the potential of this new type of approach.

Ideally, end-to-end (raw data to proposed SNP) algorithms would solve everything. An arbitrarily complex function (such as a deep neural network) constructed from the raw data to predict each particular SNP score would extract all the signal available in the raw data and this would be reflected in predictive performance in unseen data, leading to a clear landscape of the SNPs for which there is evidence of a relationship. Clearly, we have seen that this is not feasible and it is most likely not optimal. Besides the computational burden of such a task, there is some shared "knowledge" between different SNPs that must be used, therefore training a model each time from scratch is not an option. But there are even bigger problems, as we have seen, evaluation of performance on unseen data is far from trivial when we cannot perform computationally intense approaches such as permutation tests. MCC, while preferable over other metrics [7] is still problematic due to changes in its distribution when the data is highly imbalanced [49] diminishing the "power" of the approach. Furthermore, it is also not straightforward how to incorporate control for confounding due to population stratification or experimental design in reverse-GWAS using (complex) machine learning methods. As has been seen in the case of supervised contrastive learning, the performance of reverse-GWAS increased in most of the SNPs, which surely is related to confounding due to population structure.

It is worth mentioning that current reverse-QTL approaches that rely upon omnibus test cannot incorporate control for population stratification in a simple way either. [14] relies on an ad hoc distribution of the omnibus test that takes into account population structure but [38] does not consider this. PheWas approaches, owing to its focus on single traits do not focus on omnibus testing, control for population structure simply using the PC of the genetic markers matrix [46]. Clearly, a methodology allowing powerful genetic mapping on inherently multidimensional traits would be utterly useful in this context.

Traditional GWAS approaches are useful, however, only to the extent in which it is possible to extract linearly disentangled representations. As we have seen, contrastive supervised learning aiming to cluster the genotypes is a good approach to achieving linearly disentangled traits.

The lack of interpretability is another clear issue with "black-box" like algorithms and is intimately related to confounding issues. Interpretable models are key for the practical usage of this kind of methodology. It is a must to be able to assess possible issues arising from biases in the training data [3]. Furthermore, good performance may mean new biologically meaningful traits but they cannot be understood if the model lacks interpretability. This has been clearly the case in the present thesis.

It is also key to be able to incorporate prior knowledge in machine learning algorithms. This can be in the form of transfer learning (as we have seen using trained Xception) or tailoring the inception bias of the model to focus on what a priory could be biologically relevant. This can be expressed, for instance, in the choice between of 1D or 2D convolutions or self-attention architectures.

Finally, uncertainty estimation. Neural networks give point estimates without consideration of the distribution. Good estimates of confidence (i.e. credible intervals) are another piece required in order to take full advantage of these powerful machine learning methods. Bayesian deep learning is promising in this regard [48].

Challenges in food production due to the population growth, changing climate, and water resources are central issues in our society. Further research effort in the above mentioned direction could help in developing frameworks that allow to digest increasingly big amounts of data rapidly. Then, promoting the discovery of new meaningful patterns can help overcoming the obstacles faced with agricultural production.

References

[1] Carlos Alonso-Blanco and Maarten Koornneef. "Naturally occurring variation in Arabidopsis: an underexploited resource for plant genetics". In: *Trends in plant science* 5.1 (2000), pp. 22–29.

[2] Steven L Anderson et al. "Unoccupied aerial system enabled functional modeling of maize height reveals dynamic expression of loci". In: *Plant Direct* 4.5 (2020), e00223.

[3] Christina B Azodi, Jiliang Tang, and Shin-Han Shiu. "Opening the black box: interpretable machine learning for geneticists". In: *Trends in genetics* 36.6 (2020), pp. 442–455.

[4] Douglas Bates et al. "Fitting Linear Mixed-Effects Models Using lme4". In: *Journal of Statistical Software* 67.1 (2015), pp. 1–48. DOI: 10.18637/jss.v067.i01.

[5] Charles Blundell et al. "Weight uncertainty in neural network". In: *International conference on machine learning*. PMLR. 2015, pp. 1613–1622.

[6] Leo Breiman. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)". In: *Statistical science* 16.3 (2001), pp. 199–231.

[7] Davide Chicco and Giuseppe Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC genomics* 21.1 (2020), pp. 1–13.

[8] François Chollet. "Xception: Deep learning with depthwise separable convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

[9] Nusret Demir et al. "Automated measurement of plant height of wheat genotypes using a DSM derived from UAV imagery". In: *Multidisciplinary Digital Publishing Institute Proceedings* 2.7 (2018), p. 350.

[10] Aalt Dirk Jan van Dijk et al. "Machine learning in plant science and plant breeding". In: *Iscience* 24.1 (2021), p. 101890.

[11] Joshua V Dillon et al. "Tensorflow distributions". In: *arXiv preprint arXiv:1711.10604* (2017).

[12] Carl Doersch. "Tutorial on variational autoencoders". In: *arXiv preprint arXiv:1606.05908* (2016).

[13] Mitchell J Feldmann et al. "Images carried before the fire: The power, promise, and responsibility of latent phenotyping in plants". In: *The Plant Phenome Journal* 4.1 (2021), e20023.

[14] Zeny Feng. "A generalized quasi-likelihood scoring approach for simultaneously testing the genetic association of multiple traits". In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 63.3 (2014), pp. 483–498.

[15] Zeny Feng et al. "Generalized genetic association study with samples of related individuals". In: *The Annals of applied statistics* 5.3 (2011), pp. 2109–2130.

[16] Manuel AR Ferreira and Shaun M Purcell. "A multivariate test of association". In: *Bioinformatics* 25.1 (2009), pp. 132–133.

[17] Polina Golland et al. "Permutation Tests for Classification". In: *Learning Theory*. Ed. by Peter Auer and Ron Meir. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 501–515. ISBN: 978-3-540-31892-7.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[19] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[20] Agnar Helgason et al. "An Icelandic example of the impact of population structure on association studies". In: *Nature genetics* 37.1 (2005), pp. 90–95.

[21] Dan Hendrycks and Kevin Gimpel. "Gaussian error linear units (gelus)". In: *arXiv preprint arXiv:1606.08415* (2016).

[22] Irina Higgins et al. "beta-vae: Learning basic visual concepts with a constrained variational framework". In: (2016).

[23] Jian Huang et al. “Deep Dimension Reduction for Supervised Representation Learning”. In: *arXiv preprint arXiv:2006.05865* (2020).

[24] Hassan Ismail Fawaz et al. “Inceptiontime: Finding alexnet for time series classification”. In: *Data Mining and Knowledge Discovery* 34.6 (2020), pp. 1936–1962.

[25] Yu Jiang and Changying Li. “Convolutional neural networks for image-based high-throughput plant phenotyping: a review”. In: *Plant Phenomics* 2020 (2020).

[26] Hyun Min Kang et al. “Variance component model to account for sample structure in genome-wide association studies”. In: *Nature genetics* 42.4 (2010), pp. 348–354.

[27] Samir Kaul et al. “Analysis of the genome sequence of the flowering plant Arabidopsis thaliana”. In: *nature* 408.6814 (2000), pp. 796–815.

[28] Prannay Khosla et al. “Supervised contrastive learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 18661–18673.

[29] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[30] Karen J Kloth et al. “AtWRKY22 promotes susceptibility to aphids and modulates salicylic acid and jasmonic acid signalling”. In: *Journal of Experimental Botany* 67.11 (2016), pp. 3383–3396.

[31] Karen J Kloth et al. “High-throughput phenotyping of plant resistance to aphids by automated video tracking”. In: *Plant Methods* 11.1 (2015), pp. 1–14.

[32] Karen J Kloth et al. “SIEVE ELEMENT-LINING CHAPERONE1 restricts aphid feeding on Arabidopsis during heat stress”. In: *The Plant Cell* 29.10 (2017), pp. 2450–2464.

[33] Ioannis Kontopoulos, Antonios Makris, and Konstantinos Tserpes. “A deep learning streaming methodology for trajectory classification”. In: *ISPRS International Journal of Geo-Information* 10.4 (2021), p. 250.

[34] Willem Kruijer et al. “Marker-based estimation of heritability in immortal populations”. In: *Genetics* 199.2 (2015), pp. 379–398.

[35] Nan M Laird and Christoph Lange. *The fundamentals of modern statistical genetics*. Springer, 2011.

[36] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[37] Tom O’Malley et al. *Keras Tuner*. `https://github.com/keras-team/keras-tuner`. 2019.

[38] Paul F O’Reilly et al. “MultiPhen: joint model of multiple phenotypes can increase discovery in GWAS”. In: *PloS one* 7.5 (2012), e34861.

[39] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[40] Alkes L Price et al. “Principal components analysis corrects for stratification in genome-wide association studies”. In: *Nature genetics* 38.8 (2006), pp. 904–909.

[41] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “” Why should i trust you?” Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[42] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: `10.5281/zenodo.3509134`. URL: `https://doi.org/10.5281/zenodo.3509134`.

[43] Emil Uffelmann et al. “Genome-wide association studies”. In: *Nature Reviews Methods Primers* 1.1 (2021), pp. 1–21.

[44] Bart-Jan van Rossum and Willem Kruijer. *statgenGWAS: Genome Wide Association Studies*. R package version 1.0.8. 2022. URL: `https://CRAN.R-project.org/package=statgenGWAS`.

[45] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[46] Lijuan Wang et al. “Methodology in phenome-wide association studies: a systematic review”. In: *Journal of medical genetics* 58.11 (2021), pp. 720–728.

[47] Detlef Weigel and Richard Mott. “The 1001 genomes project for Arabidopsis thaliana”. In: *Genome biology* 10.5 (2009), pp. 1–5.

[48] Andrew G Wilson and Pavel Izmailov. “Bayesian deep learning and a probabilistic perspective of generalization”. In: *Advances in neural information processing systems* 33 (2020), pp. 4697–4708.

[49] Qiuming Zhu. “On the performance of Matthews correlation coefficient (MCC) for imbalanced dataset”. In: *Pattern Recognition Letters* 136 (2020), pp. 71–80. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2020.03.030. URL: https://www.sciencedirect.com/science/article/pii/S016786552030115X.