Simen Nesland

Exploring Graph-Based Deep Learning Models for Genomic Prediction in Wild Populations

Specialization Project in Physics and Mathematics
Supervisor: Stefanie Muff
December 2025

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU

Abstract

Genomic prediction aims to infer genetic components of quantitative traits from dense marker data, but wild populations pose additional challenges due to small sample sizes, environmental heterogeneity, and population structure. In this report, we investigate whether graph neural networks (GNNs) can improve prediction by explicitly incorporating relatedness structure through a graph derived from the genomic relationship matrix (GRM). We study a house sparrow (*Passer domesticus*) meta-population sampled across multiple islands and years, using genome-wide SNP data and three morphological traits: body mass, wing length, and tarsus length.

In the graphs, SNP-based node features represent individuals, while edges are constructed by sparsifying the GRM using thresholding or k-nearest-neighbour schemes. We compare multilayer perceptrons (MLPs), graph convolutional networks (GCNs), and GraphSAGE variants (including transductive and inductive formulations) against a strong classical baseline, the genomic animal model, using its genomic best linear unbiased prediction (GBLUP). Model selection and performance estimation are conducted using nested cross-validation and automated hyperparameter optimization, and the correlation between predictions and phenotypes is used to assess predictive performance.

For prediction within populations, the classical baseline remains difficult to outperform: learned nonlinear models overlap substantially with the genomic animal model across folds and do not show a consistent advantage, with MLPs emerging as the most robust non-linear model and GNNs providing no systematic gains. For prediction across populations, which is known to be more challenging, the relative standing of the models shifts: nonlinear models become more competitive, and GraphSAGE attains the best median performance for tarsus length and wing length. Per-island analyses highlight substantial heterogeneity across the meta-population, with learned models performing best on most islands and the genomic animal model on only one. Finally, the learned graph constructions are typically extremely sparse, suggesting that only very strong genomic similarities are selected as informative neighbourhoods.

Table of Contents

List of Figures

List of Tables

1 Introduction

Living organisms vary in measurable characteristics such as size, growth, and morphology. Many of these characteristics are quantitative traits, meaning they vary continuously in the population rather than falling into a few discrete categories (e.g. body mass or height) (Falconer and Mackay, 1996; Lynch and Walsh, 1998). Quantitative traits typically reflect the combined influence of many genetic loci and environmental factors (Falconer and Mackay, 1996; Lynch and Walsh, 1998; Xu, 2022). Quantitative genetics provides the statistical framework for decomposing phenotypic variation into genetic and environmental components, and formalizing how genetic variation translates into response to selection across generations (Falconer and Mackay, 1996; Lynch and Walsh, 1998; Conner and Hartl, 2004). In its classical formulation, the genetic component may include additive effects (the summed contribution of alleles across loci), dominance, and epistasis (Falconer and Mackay, 1996; Lynch and Walsh, 1998; Conner and Hartl, 2004). The additive component is essential because it equals the expected resemblance between relatives, known as the breeding value (Falconer and Mackay, 1996; Lynch and Walsh, 1998). Key quantities such as heritability (the proportion of phenotypic variance attributable to genetic variance) are therefore not only descriptive parameters, but central targets for statistical inference and prediction (Falconer and Mackay, 1996; Lynch and Walsh, 1998; Jensen, Sæther et al., 2003). A fundamental practical challenge follows, as breeding values and genetic components are latent. Thus, they must be inferred from phenotype data together with information about relatedness or shared genetic background.

Historically, the primary source of information about genetic resemblance was the pedigree, a recorded family tree describing ancestry. Pedigrees enable the quantification of expected relatedness among individuals. This relatedness information is, in turn, used to fit animal models, which are linear mixed models that combine fixed effects (systematic predictors such as sex, age, or cohort) with random effects capturing additive genetic variation structured by relatedness (Henderson, 1975; Kruuk, 2004). Within animal and plant breeding, these models are motivated by the goal of ranking candidates for selection and achieving genetic gain over generations, often through best linear unbiased prediction (BLUP) of breeding values (Henderson, 1975; Desta and Ortiz, 2014; Boichard et al., 2016). The animal-model framework is appealing because it provides a coherent probabilistic model for prediction, and separates genetic from environmental sources of variation in a way that aligns closely with quantitative-genetic theory (Lynch and Walsh, 1998; Kruuk, 2004). However, pedigree-based approaches also have limitations that become increasingly evident outside managed breeding populations. Pedigrees may be incomplete or unavailable, and they encode expected relatedness rather than realized sharing of genomic segments (Lynch and Walsh, 1998; Kruuk, 2004). Moreover, while pedigree models capture aggregate genetic resemblance, they do not directly leverage genomic variation to learn how genetic effects are distributed across the genome. These limitations, together with rapid advances in genotyping technologies, motivated the move from pedigree-informed prediction to genomic prediction.

In genomic prediction, genetic information is represented by genome-wide markers, most commonly single nucleotide polymorphisms (SNPs), which are single-base variants distributed across the genome (Brookes, 1999; Meuwissen et al., 2001; VanRaden, 2008; Meuwissen et al., 2016). From a statistical perspective, each individual is encoded by a high-dimensional marker vector and the prediction task is to learn a mapping from this vector (and other covariates) to the phenotype or breeding value. Two closely related classical paradigms dominate. The first is marker-based regression, in which many marker effects are fitted simultaneously, allowing the polygenic signal to be aggregated even when single-marker effects are too small to estimate reliably (Meuwissen et al., 2001; Gianola, 2013). The second is the genomic animal model, which summarizes marker data through the genomic relationship matrix (GRM), describing realized rather than expected relatedness, and then applies the same mixed-model machinery as the animal model (VanRaden, 2008; Hill and Weir, 2011; Meuwissen et al., 2016). Genomic best linear unbiased predictions (GBLUPs) are obtained in a similar fashion to the original animal model's BLUPs. Both marker-based regression and GRM-based mixed models have proven effective for genomic prediction in a wide range of settings (Meuwissen et al., 2001; VanRaden, 2008; Vlaming and Groenen, 2015; Meuwissen et al., 2016).

Despite their success, classical SNP-based methods face limitations that motivate alternative modelling strategies. A defining feature of marker-based regression is the high-dimensional regime $p \gg n$, which makes regularization, shrinkage, or dimension reduction essential for stable estimation (Fahrmeir, 2013; Vlaming and Groenen, 2015; Hastie et al., 2017). Practical pipelines therefore include SNP filtering, principal component analysis (PCA), or other feature compression to control noise and reduce computational burden (Ødegård et al., 2018). Furthermore, the GRM used in GBLUP is typically dense, so memory requirements and computations scale poorly with sample size (VanRaden, 2008). One might hope to sparsify the GRM by thresholding small entries, but arbitrary sparsification can break positive semidefiniteness and thereby prevent inversion (Banerjee and Roy, 2015). In addition, the classical formulations are effectively linear in the marker representation and are primarily designed to capture additive genetic effects. While additivity often dominates variance components, non-additive mechanisms such as dominance and epistasis can still be biologically relevant and may influence prediction (Mackay, 2014). These issues are particularly salient in wild populations, where goals often differ from those of breeding, and the data-generating process is less controlled. In evolutionary ecology, prediction is used to quantify evolutionary potential, study phenotypic change under natural selection, and understand how genetic variation is maintained under environmental variability (Kruuk, 2004; Jensen, Steinsland et al., 2008; Wilson et al., 2010; Lundregan et al., 2018; Muff et al., 2019). Wild datasets are typically smaller, and traits are measured under substantial environmental heterogeneity across space and time (Kruuk, 2004; Wilson et al., 2010; Husby et al., 2011; Ranke et al., 2021; Ashraf et al., 2022). Non-random survival and reproduction can further complicate inference, and prediction may need to transfer across subpopulations with different genetic backgrounds and environments (Aase et al., 2025).

Motivated by these limitations, machine learning (ML) has been increasingly explored for genomic prediction (Eraslan et al., 2019). Many ML models can be viewed as nonlinear extensions of marker-based regression: SNPs enter as features, and the model assigns an effect to each marker while learning flexible interactions and nonlinear feature combinations (Goodfellow et al., 2016; Hastie et al., 2017). The $p \gg n$ regime does not disappear, but is handled through regularization strategies such as weight decay, dropout, early stopping, and architectural constraints that limit effective model complexity (Goodfellow et al., 2016). Empirical studies suggest that ML can be competitive and sometimes beneficial. Still, gains over strong classical baselines are not consistent and depend on trait architecture, sample size, and population structure (B. Li et al., 2018). Recent experiments have investigated ML pipelines that combine feature compression, reduced SNP sets, and nonlinear predictors, illustrating both potential improvements and practical challenges in the wild-population setting (Singsaas, 2024; Gravdal, 2025; Wold, 2025). Clearly, flexibility alone is not sufficient. In data-limited settings, successful models typically rely on an inductive bias, that is, built-in assumptions or structural constraints that restrict the space of functions a model can learn and thereby favor solutions that generalize (Prince, 2024).

Graph neural networks (GNNs) provide one way to introduce this kind of inductive bias by explicitly encoding relationships between individuals as a graph. GNNs have recently shown promise in other, related fields (Reiser et al., 2022; King and Huang, 2023; X. Li et al., 2023; Qin et al., 2023; Ahmad et al., 2024). In our setting, individuals are nodes and edges encode genetic similarity or relatedness, for example derived from a pedigree or the GRM. Predictions are learned through message passing, where each node updates its representation by aggregating information from a neighbourhood defined by the graph structure (Scarselli et al., 2009; Gilmer et al., 2017; Hamilton, 2022; Prince, 2024). This creates a natural bridge between classical and marker-based approaches. Animal models use the pedigree or GRM to specify how genetic information is shared among relatives via a linear covariance structure. In contrast, marker-based regression and neural networks learn the effects of many markers under regularization. A GRM-based GNN combines these ideas by using relatedness to define which individuals exchange information, while a neural predictor learns a nonlinear mapping from genomic data to the trait. In this formulation, sparsifying the graph is part of the modelling choice rather than a threat to a covariance interpretation. Sparsity controls which neighbours exchange information, enabling scalable neighbourhood-based constructions that would be difficult to justify in a strict covariance-based model. Recent methodological work has begun to develop GNN variants specifically for genomic prediction, with promising results (Kihlman et al., 2024).

This report aims to evaluate whether GRM-derived graph structure can improve the prediction of quantitative traits in a challenging wild-population setting. We study a house sparrow (*Passer domesticus*) metapopulation sampled across multiple subpopulations and years. We construct sparse graphs from genomic relatedness and compare classical mixed-model baselines, marker-based neural networks, and GNN architectures. Model selection and performance estimation are carried out using nested cross-validation with automated hyperparameter optimization (Akiba et al., 2019). We report results for both within-population prediction and across-population transfer, thereby aligning evaluation with the key question of generalization under population structure. All code used for preprocessing, model training, and evaluation is available at https://github.com/simennes/GPWild_GNN.

2 Background

2.1 Fundamentals of quantitative genetics

For a thorough understanding of the methods and results presented in this report, knowledge of some fundamental concepts within genetics is beneficial. In this section, we explore basic topics related to the data and models discussed later in the report.

2.1.1 DNA, genes, and alleles

All hereditary information is stored in deoxyribonucleic acid (DNA), composed of four nucleotides: adenine (A), thymine (T), cytosine (C), and guanine (G). Portions of DNA make up genes. Genes can code for proteins that contribute to the expression of variable traits. An observable trait is called a phenotype and can refer to anything from body mass to blood type. DNA has the shape of a long double helix and is tightly wrapped around proteins to form chromosomes. When a new individual is formed, half of its chromosomes are inherited from the father, and the other half from the mother. This way, all individuals have two versions of all genes, known as alleles (Vologodskij, 2023). In other words, alleles are varieties of a gene, and they could be the same or different. In a situation where an individual has two different alleles, one allele can be more impactful in the expression of a given trait. These alleles are referred to as dominant and are often denoted by upper case letters, as opposed to recessive alleles, which are denoted by lower case letters. This dominance effect was first studied by Gregor Mendel in the 19th century, who conducted common-garden experiments on pea plants. He discovered that the allele for purple flower color was dominant over the allele for white flower color. Thus, a plant with the ”BB” or ”Bb” genotype, possessing at least one of the dominant alleles, would have a purple flower. Only the ”bb” genotype encodes a white flower (Xu, 2022).

2.1.2 Single nucleotide polymorphisms (SNPs)

While alleles represent alternative versions of genes, single nucleotide polymorphisms (SNPs) describe variation at the most fundamental level of the DNA sequence. A SNP occurs when a single nucleotide base differs between individuals. For example, one individual may have an adenine (A) at a given genomic position, while another has a guanine (G). These single-base differences arise from random point mutations that, over evolutionary time, have become stably inherited within populations (Fareed and Afzal, 2013). Such variants occur roughly once every thousand base pairs in the genome, making them the most common form of genetic variation in humans and other species. To distinguish SNPs from rare or spontaneous mutations, the less common variant (the minor allele) must be present in at least 1% of the population (Brookes, 1999). Compared to other types of polymorphisms, such as insertions, deletions, or microsatellites, SNPs are highly abundant and widely distributed across the genome (Schaid et al., 2004).

As with genes, the alternative nucleotides at a SNP locus are also referred to as alleles. Most SNPs are biallelic, meaning that only two allelic variants exist at a given position. Because individuals inherit one set of chromosomes from each parent, a biallelic SNP gives rise to three possible genotypes (Brookes, 1999). For instance, for adenine and guanine, the potential combinations are “AA”, “AG”, and “GG”.These genotypes can be encoded numerically based on the number of copies of a selected reference or minor allele. With guanine as the reference, the example above yields values of 0, 1, and 2, respectively. This additive numerical encoding is particularly useful for statistical and machine learning models, as it provides a straightforward way to incorporate genomic information into analyses.

Although SNPs are dense markers that efficiently capture genetic variation, the majority of the genome is non-coding, meaning that most DNA does not directly contribute to protein production. Consequently, most SNPs do not lie within protein-coding sequences (Brodie et al., 2016). However, this does not make them biologically uninformative. Linkage disequilibrium (LD) describes the non-random association of alleles at different loci, that is, when two alleles are inherited together more often than expected by chance. LD occurs naturally because loci that are physically close on the same chromosome tend to be co-inherited due to limited recombination (Slatkin, 2008). Hence, even if a sequenced SNP is located outside a coding region, it may still be in LD with functional alleles nearby and can therefore serve as an effective marker for studying traits (Brookes, 1999).

2.1.3 Phenotypic variance and heritability

While genetic variation occurs at the molecular level through mechanisms such as mutations and polymorphisms, its consequences are observed at the phenotypic level as variation in measurable traits. In natural populations, phenotypic differences among individuals arise not only from genetic differences but also from environmental influences (Conner and Hartl, 2004). These environmental factors can include variation in climate, access to nutrition, or exposure to pathogens, among others. Mathematically, this relationship can be expressed as $P = G + E$, illustrating that the observed phenotype (P) results from both genetic (G) and environmental (E) components. The genetic component can influence phenotypes in several ways, collectively referred to as gene action. Additive gene action describes the linear contribution of alleles to a trait, where each allele adds a fixed amount to the phenotype regardless of the other alleles present. Dominance occurs when alleles at the same locus interact such that one masks or modifies the effect of another. Epistasis, on the other hand, refers to interactions between alleles at different loci, where the effect of one gene depends on the genotype at another locus (Conner and Hartl, 2004). These forms of gene action together shape how genetic information is translated into phenotypic expression.

Although phenotypes themselves are biologically interesting, statistical models in quantitative genetics aim to explain variation among individuals. Consequently, the phenotypic variance within a population is of primary interest. In parallel with the partitioning of the phenotype into genetic and environmental components, we can express the total variance as $V_P = V_G + V_E$, where V_P is the total phenotypic variance, V_G is the genetic variance, and V_E is the environmental variance (Conner and Hartl, 2004). It is important to note that this formulation ignores gene–environment interactions, the phenomenon that not all genotypes respond equally to environmental variation (Davies, 2012). Keeping this simplification, the genetic variance can be further decomposed as $V_G = V_A + V_D + V_I$, where V_A is the additive genetic variance, V_D the dominance variance, and V_I the epistatic variance (Conner and Hartl, 2004). In breeding and prediction contexts, the additive component corresponds to the breeding value of an individual (Falconer and Mackay, 1996). The proportion of total phenotypic variance attributable to additive genetic variance is known as the narrow-sense heritability, defined as $h^2 = \frac{V_A}{V_P}$. This quantity represents the degree to which genetic differences among individuals explain observed differences in phenotype and determines the potential for a trait to respond to selection (Falconer and Mackay, 1996). In practical terms, a higher heritability implies that phenotypic differences are largely determined by inherited genetic factors rather than environmental noise, and that breeding values can therefore be predicted with greater accuracy. For reference, the broad-sense heritability is defined as $h_B^2 = \frac{V_G}{V_P}$ and thus encompasses the proportion of variance explained by all genetic factors. However, since dominance and interaction effects are not directly heritable, h^2 is often the parameter of interest.

2.2 Genomic prediction and the GRM

2.2.1 The animal model

In selective breeding programs, the primary goal is to identify individuals with the greatest genetic potential for desirable traits. Because the actual genetic value of an individual cannot be observed directly, quantitative genetic methods were developed to estimate the genetic contribution to phenotypic variation, the breeding values. The animal model is a linear mixed model (LMM) used to estimate breeding values based on observed phenotypes and known pedigrees (Kruuk, 2004). In general, LMMs extend ordinary linear models by including both fixed and random effects, thereby accounting for structured sources of variability, such as genetic relatedness or repeated measurements. Fixed effects represent systematic influences shared across individuals, while random effects capture variation associated with specific grouping factors or random processes (Fahrmeir, 2013).

The most basic version of the animal model can be written as

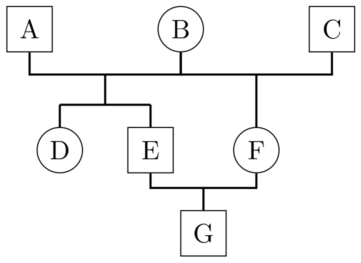
$$y_i = \mu + a_i + \epsilon_i, \quad i = 1, \ldots, n\,,$$

where y_i is the phenotype of individual i, μ represents the population mean (fixed effect), a_i represents the breeding value of individual i (random effect) and ϵ_i is the residual term (random effect). More generally, the animal model takes the form

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon}, \quad \mathbf{u} = \begin{pmatrix} \mathbf{a} \\ \tilde{\mathbf{u}} \end{pmatrix}, \tag{1}$$

where $\mathbf{y}$ is a vector of phenotypes, $\mathbf{b}$ a vector of fixed effects with its corresponding design matrix $\mathbf{X}$, $\mathbf{u}$ a vector of breeding values ($\mathbf{a}$) and other random effects ($\tilde{\mathbf{u}}$), with its design matrix $\mathbf{Z}$. Finally, $\boldsymbol{\epsilon}$ is a vector containing the residuals. In a LMM, the random effects are assumed to follow a distribution. The residual terms are assumed to be independent and identically distributed (iid), following a multivariate normal distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma_\epsilon^2 \mathbf{I})$. Ignoring any other random effects for now, the breeding values are also assumed to follow a normal distribution with zero mean, but with a specific covariance structure $\mathbf{a} \sim \mathcal{N}(0, V_A \mathbf{A})$ (Lynch and Walsh, 1998). We recognize V_A as the additive genetic variance, while $\mathbf{A}$ is, for now, an undefined matrix that describes the covariances among the breeding values.

The study of $\mathbf{A}$, the additive genetic relationship matrix, requires a little more care. Traditionally, it was derived from pedigrees. For any two individuals i and j in a pedigree, one can calculate Θ_{ij}, the coefficient of coancestry. This coefficient is based on the concept of identical by descent (IBD), which refers to alleles that are copies of the same ancestral allele inherited through a common ancestor. The coefficient of coancestry quantifies the probability that an allele randomly drawn from individual j and an allele randomly drawn from individual i are IBD. For parent and offspring, $\Theta_{ij} = 0.25$ (Falconer and Mackay, 1996). The additive genetic relationship is defined as twice the coefficient of coancestry; thus, for parent and offspring, $A_{ij} = 2\Theta_{ij} = 0.5$. The matrix is symmetric since $\Theta_{ij} = \Theta_{ji}$, and assuming no inbreeding, the diagonal elements are all equal to one ($A_{ii} = 1$, $i \in \{1, \ldots, N\}$). In the case of inbreeding, the diagonal element of individual i would be $A_{ii} = 1 + F_i$ where F_i is the coefficient of inbreeding (Wilson et al., 2010). Figure 1 shows an example pedigree and its corresponding additive relationship matrix.



	A	B	C	D	E	F	G
A	1	0	0	0.5	0.5	0	0.25
B	0	1	0	0.5	0.5	0.5	0.5
C	0	0	1	0	0	0.5	0.25
D	0.5	0.5	0	1	0.5	0.25	0.375
E	0.5	0.5	0	0.5	1	0.25	0.5
F	0	0.5	0.5	0.25	0.25	1	0.5
G	0.25	0.5	0.25	0.375	0.5	0.5	1.125

Figure 1: Pedigree (squares = males, circles = females) and corresponding additive relationship matrix **A**. Founders A, B, and C are unrelated. D and E are full siblings from A×B, F is the child of B×C (half-sibling to D and E via B). G is the result of half-sibling mating and thus has an inbreeding coefficient greater than zero.

When estimating the breeding values from the model, the main properties desired are unbiasedness and low variance. Using the formulation from equation (1), we write $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{V}_u)$, where the matrix $\mathbf{V}_u$ represents the covariance structure of all random effects, including the breeding values. Henderson (1975) demonstrated that breeding values are optimally estimated using the best linear unbiased predictor (BLUP). The term "best" refers to achieving the smallest possible prediction error variance among all linear unbiased predictors. The BLUP of $\mathbf{u}$ is obtained by solving the so-called mixed model equations:

$$\begin{pmatrix} \sigma_\epsilon^{-1}\mathbf{X}^T\mathbf{X} & \sigma_\epsilon^{-1}\mathbf{X}^T\mathbf{Z} \\ \sigma_\epsilon^{-1}\mathbf{Z}^T\mathbf{X} & \sigma_\epsilon^{-1}\mathbf{Z}^T\mathbf{Z} + \mathbf{V}_u^{-1} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}} \end{pmatrix} \begin{pmatrix} \sigma_\epsilon^{-1}\mathbf{X}^T\mathbf{y} \\ \sigma_\epsilon^{-1}\mathbf{Z}^T\mathbf{y} \end{pmatrix}.$$

From these, the predictor of the random-effect vector is

$$\hat{\mathbf{u}} = \mathbf{V}_u\mathbf{Z}^T(\mathbf{Z}\mathbf{V}_u\mathbf{Z}^T + \sigma_\epsilon\mathbf{I})^{-1}(\mathbf{y} - \mathbf{X}\hat{\mathbf{b}})\,.$$

The derivation of these results is beyond the scope of this report. However, it is important to note that the animal model yields analytical expressions for the estimators, allowing breeding values to be obtained directly through the mixed model equations.

2.2.2 From pedigrees to genomic data

Pedigree-based approaches estimate genetic relationships from known ancestry and therefore represent expected rather than realized allele sharing. As discussed in Section 2.1.2, single nucleotide polymorphisms (SNPs) provide a dense and informative source of genetic variation across the genome. With the advent of high-throughput genotyping technologies, it has become possible to measure genome-wide similarity directly from these molecular markers, marking a major shift in quantitative genetics (Meuwissen et al., 2016). Traditional pedigree methods assign fixed coefficients of relatedness. For example, full siblings are expected to share 50% of their alleles identical by descent. However, random Mendelian segregation causes variation around this expectation, such that the realized proportion of shared alleles can differ substantially among siblings (Hill and Weir, 2011). SNP-based genotyping enables these differences to be measured directly, allowing estimation of individual-specific relatedness.

To incorporate genomic information into the animal model, the pedigree-based relationship matrix $\mathbf{A}$ must be replaced with a matrix that reflects realized genetic relationships inferred from molecular data. This substitution is not straightforward; SNP genotypes must be transformed into a matrix that has the same scale and interpretation as $\mathbf{A}$. VanRaden (2008) proposed an efficient and widely adopted approach for constructing such a matrix, now known as the genomic relationship matrix (GRM). Let $\mathbf{M}$ be the $n \times p$ matrix of genotypes for n individuals and p loci, where each element $m_{ij} \in \{-1, 0, 1\}$ represents the number of copies of the second allele at locus j (coded as -1 for the first homozygote, 0 for the heterozygote, and 1 for the second homozygote). Let p_j denote the frequency of the second allele at locus j, and define a frequency matrix $\mathbf{P}$ such that column j contains the value $2(p_j - 0.5)$ repeated for all individuals. The centered genotype matrix is then given by

$$\mathbf{C} = \mathbf{M} - \mathbf{P}\,.$$

This centering ensures that each locus has an expected mean of zero across individuals, and makes less common alleles contribute more in the calculation of genomic relationships. After centering, the genomic relationship matrix is then computed as

$$\mathbf{G} = \frac{\mathbf{C}\mathbf{C}^T}{2\sum_{j=1}^{p} p_j(1-p_j)}\,, \tag{2}$$

where the denominator standardizes the matrix so that its scale is equivalent to that of $\mathbf{A}$. In practice, $\mathbf{G}$ replaces $\mathbf{A}$ in the mixed-model covariance structure $\mathrm{Var}(\mathbf{a}) = V_a\mathbf{G}$.

Using the GRM rather than the additive genetic relationship matrix yields the genomic animal model. The predictor for the breeding values in the genomic animal model has precisely the same form as in the original animal model, but is referred to as the genomic best linear unbiased predictor (GBLUP). Thus, the genomic animal model is a natural extension of the original animal model. All the genetic information from the SNP markers enters the model through the covariance structure among individuals, summarised by the GRM.

2.2.3 Marker-based regression

In an alternative but closely related formulation to the genomic animal model, we can regress the marker effects directly on phenotypes, rather than model them through their induced covariance. The theoretical basis for this approach originates from the infinitesimal model, which assumes that quantitative traits arise from the additive contribution of a very large number of loci, each exerting a small effect on the phenotype (Fisher, 1930). Consistent with this view, most complex traits are now known to be highly polygenic, with phenotypic variation explained by the cumulative influence of many loci of small effect (Yang et al., 2010; Choi et al., 2020). This leads to a class of marker-based regression models, in which phenotypes are regressed directly on genotype covariates at each SNP (Meuwissen et al., 2001). Under this formulation, the phenotype vector $\mathbf{y}$ can be expressed as

$$\mathbf{y} = \mu \mathbf{1_n} + \mathbf{Xb} + \tilde{\mathbf{M}}\mathbf{u} + \mathbf{Wd} + \boldsymbol{\epsilon}\,,$$

where μ is the overall mean, $\mathbf{Xb}$ describes fixed effects, and $\mathbf{Wd}$ represents additional random effects. $\tilde{\mathbf{M}}$ is a centered version of the $n \times p$ genotype matrix, and $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \sigma_u^2 \mathbf{I})$ contains the random marker effects. The breeding value for individual i is then given by

$$a_i = [\tilde{\mathbf{M}}\mathbf{u}]_i = \sum_{j=1}^{p} \tilde{m}_{ij} u_j\,,$$

which directly reflects the additive nature of the infinitesimal model, and the definition of breeding value (Lynch and Walsh, 1998). Interestingly, it can be shown that, under certain assumptions, these breeding value estimates are equivalent to the GBLUPs (Meuwissen et al., 2001).

In practice, the number of available SNPs (p) typically far exceeds the number of phenotyped individuals (n), making the estimation of $\mathbf{u}$ ill-posed (Vlaming and Groenen, 2015). When $p \gg n$, the matrix $\mathbf{Z}^T\mathbf{Z}$ becomes singular, and ordinary least squares estimation of marker effects is no longer feasible. A standard solution is to impose a penalty on the size of the regression coefficients, leading to the ridge regression (RR) or ridge regression best linear unbiased prediction (RR-BLUP) model (Hoerl and Kennard, 1970). Ridge regression shrinks the estimated SNP effects toward zero by adding a penalty term proportional to their squared magnitude. This stabilises estimation and avoids overfitting while retaining the additive genetic signal spread across the genome.

Beyond RR-BLUP, a broad class of Bayesian whole-genome regression models, often referred to as the Bayesian alphabet, relaxes the assumption of homogeneous marker effects by assigning flexible prior distributions to SNP effects (Gianola, 2013). Methods such as BayesA, BayesB, BayesC and BayesR allow heterogeneous shrinkage, enabling some markers to retain moderate or large effects while others are strongly shrunk toward zero. This flexibility is advantageous when the genetic architecture departs from the infinitesimal model and traits are influenced by a mixture of small and moderate-effect loci. Even though, in the high-dimensional setting ($p \gg n$), posterior inference on individual SNP effects is heavily driven by prior assumptions rather than the likelihood (Gianola, 2013), many Bayesian alphabet models, particularly BayesR, have shown excellent predictive performance in livestock and plant breeding (Meuwissen et al., 2001; Erbe et al., 2012). Although these methods are not pursued further in this report, they represent influential and competitive alternatives within the broader landscape of genomic prediction models.

2.3 Machine learning approaches

2.3.1 Introduction and motivation

Classical models such as the animal model have provided a robust and interpretable framework for genomic prediction by assuming additive and linear relationships between genotype and phenotype (Meuwissen et al., 2016). However, many biological processes are inherently nonlinear, involving interactions among loci (epistasis), dominance effects (see Section 2.1.3), and complex genotype-environment interactions. Classical genomic prediction models generally ignore these nonlinear effects (Mackay, 2014). In controlled breeding populations and agricultural settings, where environmental conditions are relatively homogeneous and genetic backgrounds are managed, such simplifying assumptions often remain sufficient for accurate prediction (Desta and Ortiz, 2014; Boichard et al., 2016). In contrast, individuals in natural or wild populations experience a far wider range of environmental conditions. These complexities challenge the linear assumptions underlying traditional models and motivate the use of machine learning (ML) methods, which offer more flexible approaches for capturing nonlinear genotype-phenotype relationships. Additionally, many ML methods address the $p \gg n$ problem more naturally, provided they are applied with care. In recent years, ML approaches have gained increasing attention in quantitative genetics (Eraslan et al., 2019).

2.3.2 Machine learning basics

Machine learning has its roots in classical statistical learning, and the terms are often used interchangeably in the literature. The main distinction may lie in their differing emphases on predictive accuracy versus parameter inference, as well as in their origins. Historically, statistical learning focused on interpretable models such as linear and logistic regression, where each estimated parameter could be directly related to an experimental factor. In contrast, modern machine learning has shifted toward high-capacity, so-called black-box models, such as neural networks, that prioritize predictive performance, often at the expense of interpretability. Nevertheless, ongoing developments in explainable artificial intelligence have introduced tools that help interpret the predictions even in complex nonlinear models (Lundberg and Lee, 2017). In this report, we adopt the term machine learning because the primary focus is on neural network–based approaches that originate from computer science, where predictive accuracy is prioritized over direct interpretability.

In a broad sense, methods that learn from data, that is, statistical and machine learning methods, can be classified into supervised and unsupervised approaches (Hastie et al., 2017). In this report, we focus exclusively on supervised learning, where the objective is to learn a mapping from inputs to known outputs. To formalize this, consider a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, consisting of N training samples. Each sample includes a feature vector $\mathbf{x}_i \in \mathcal{X}$, representing the input space, and a corresponding response $y_i \in \mathcal{Y}$, representing the output space. The goal of supervised learning is to find a function $f : \mathcal{X} \to \mathcal{Y}$ from a hypothesis space $\mathcal{H}$, i.e., some space of possible functions, that "best" maps inputs to outputs. The quality of a mapping is quantified by a predefined loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, which measures the discrepancy between the predicted value $f(\mathbf{x})$ and the true value y. We define the risk as the expected loss over the true, unknown data-generating distribution $\mathcal{P}(\mathbf{x}, y)$

$$R(f) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{P}}\big[L(f(\mathbf{x}), y)\big]\,.$$

The ideal function $f^* \in \mathcal{H}$ would be one that minimizes this risk

$$f^* = \underset{f\in\mathcal{H}}{\arg\min}\; R(f)\,.$$

However, since the distribution $\mathcal{P}$ is unknown, the risk $R(f)$ cannot be computed. Instead, the central idea is to approximate the expected risk using the empirical risk, which is the average loss over the finite training data $\mathcal{D}$

$$R_{emp}(f) = \frac{1}{N}\sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i)\,.$$

This approach, known as empirical risk minimization (ERM), seeks to find the hypothesis that best explains the observed data (Vapnik, 1995).

Machine learning models can further be categorized as parametric or non-parametric (James et al., 2021). Parametric models, such as linear regression and neural networks, assume a specific functional form $f(\mathbf{x}; \boldsymbol{\theta})$ governed by a finite set of parameters $\boldsymbol{\theta}$. For these models, the ERM problem reduces to finding the parameter values that minimize the empirical risk

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\arg\min}\; R_{emp}(f(\cdot\,; \boldsymbol{\theta})) = \underset{\boldsymbol{\theta}}{\arg\min}\; \frac{1}{N}\sum_{i=1}^{N} L(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i)\,.$$

A central challenge in machine learning is that the objective is not merely to minimize the empirical risk on the training data, but to learn a function that generalizes well to unseen data drawn from the same underlying distribution. Perfectly minimizing the empirical risk may lead to overfitting, in which the model begins to capture random noise in the training set rather than the underlying signal. Polynomial regression provides a simple demonstration: with $n+1$ datapoints, a polynomial of degree n can interpolate all points exactly, but this does not imply that the underlying data-generating mechanism is itself an n-degree polynomial. This tension between model flexibility and generalization is known as the bias–variance trade-off. Flexible models typically exhibit low bias but high variance, whereas more constrained models show the opposite pattern. As illustrated in

Figure 2, the training error decreases monotonically with increasing polynomial degree, while the test error follows a characteristic U-shape. It achieves the lowest test error when the degree of the fitted polynomial matches the degree of the underlying polynomial.

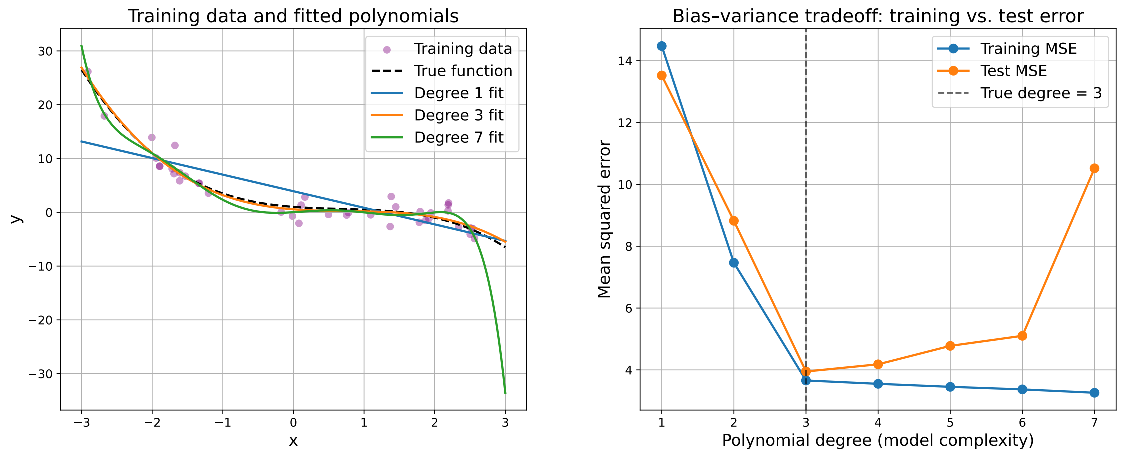


Figure 2: Illustration of model complexity in polynomial regression for a synthetic data-generating process. The underlying true relationship is a cubic function $f(x) = -0.5x^3 + x^2 - x + 1$, and observed data are generated as $y = f(x) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, 2^2)$ represents Gaussian noise. **Left:** Fitted polynomial models of degree 1, 3, and 7 on the training data. **Right:** Mean squared error for both training and test data across polynomial degrees 1–7.

In practice, generalization performance is assessed by partitioning the available dataset $\mathcal{D}$ into disjoint subsets with different purposes. A model is trained on a training set $\mathcal{D}^{(\text{tr})}$, while a separate validation set $\mathcal{D}^{(\text{val})}$ is used to tune hyperparameters and choose among competing models. For example, in the polynomial regression illustration in Figure 2, one could select the polynomial degree by comparing validation errors across degrees and choosing the model with the lowest expected generalization error, typically the degree at the bottom of the U-shaped curve. Finally, a held-out test set $\mathcal{D}^{(\text{te})}$ provides an unbiased estimate of the selected model's predictive performance. When data are limited, setting aside a substantial validation set may not be feasible. In such cases, cross-validation offers an alternative by repeatedly splitting $\mathcal{D}^{(\text{tr})}$ into training and validation folds. In k-fold cross-validation, the data is partitioned into k equally sized folds, and the model is trained k times, each time leaving out one fold for validation. This way, k estimates of the generalization error are obtained. A 5-fold cross validation procedure is illustrated in Figure 3.

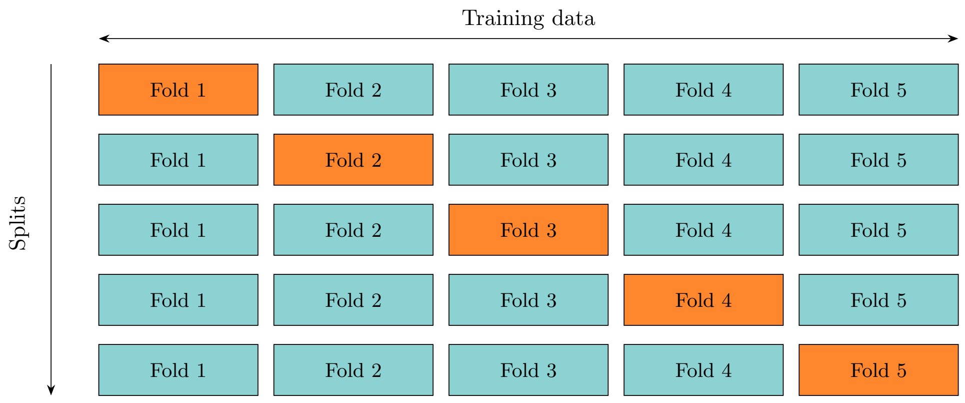


Figure 3: Illustration of cross-validation. In this case, the training data is split into 5 folds.

2.3.3 Neural networks

Neural networks (sometimes also called artificial neural networks to distinguish them from biological systems) are flexible, parametric models built from simple computational units arranged in layers. Their origins trace back to the early work of McCulloch and Pitts (1943), who introduced the first mathematical neuron, and Rosenblatt (1958), who showed that networks of such units could be trained for pattern recognition. Although this early work was conceptually influential, neural networks were not widely adopted for many years. Early models were limited in representational power, and efficient training algorithms for deeper architectures were lacking. Interest was revived in the 1980s with the reintroduction of backpropagation (Rumelhart et al., 1986), but traditional statistical models and kernel methods continued to dominate applied work. Only in the past two decades, driven by large datasets and increased computational power, have neural networks emerged as state-of-the-art tools (Goodfellow et al., 2016).

From the perspective of supervised learning, a neural network is a parametric function $f(\mathbf{x};\boldsymbol{\theta})$ trained to map inputs to outputs by minimizing a loss function over observed data (James et al., 2021). Given training samples $(\mathbf{x}_i, y_i)$, the network produces predictions $\hat{y}_i = f(\mathbf{x}_i;\boldsymbol{\theta})$ during a forward pass. Learning amounts to adjusting the parameters $\boldsymbol{\theta}$ so that these predictions approach the true responses. The structure of a neural network determines how inputs are transformed during this forward pass. The fundamental building block is the perceptron. For an input vector $\mathbf{x} \in \mathbb{R}^N$, the perceptron computes a linear transformation followed by a nonlinear activation function

$$y = \sigma(\mathbf{w}^T\mathbf{x} + b)\,,$$

where $\mathbf{w}$ is a weight vector and b is a bias term. The activation function σ is what introduces nonlinearity into the model. A variety of activation functions have been proposed and evaluated in the literature. Early neural networks commonly employed smooth sigmoidal functions, while modern architectures typically favour the rectified linear unit (ReLU) and its variants due to their favourable optimization properties (Russell and Norvig, 2022). An overview of widely used activation functions and their derivatives is provided in Table 1.

Name	$\boldsymbol{\sigma(z)}$	$\boldsymbol{\sigma'(z)}$
Sigmoid	$\frac{1}{1+e^{-z}}$	$\sigma(z)(1-\sigma(z))$
Tanh	$\tanh(z)$	$1-\tanh^2(z)$
ReLU	$\max(0,z)$	$I(z>0)$
Leaky ReLU	$\max(0.01z, z)$	$\begin{cases}1 & z>0\\ 0.01 & z\le 0\end{cases}$
Softplus	$\log(1+e^z)$	$\sigma(z)$

Table 1: Common activation functions $\sigma(z)$ and their derivatives $\sigma'(z)$.

Arranging many perceptrons in parallel yields a layer that applies the transformation

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})\,,$$

where $\mathbf{W}$ contains the weights of all perceptrons in the layer and $\mathbf{b}$ their biases. Stacking several layers forms a multilayer perceptron (MLP), where each layer transforms the output of the previous one. During the forward pass, information flows from the input toward the output. A simple MLP is illustrated in Figure 4. The expressive power of this architecture is supported by the universal approximation theorem, which states that a feed-forward network with at least one hidden layer and a suitable activation function can approximate any continuous function on a compact set to arbitrary accuracy (Cybenko, 1989; Hornik et al., 1989).

To quantify prediction error, we define a loss function. For regression problems, a common choice is the (half) mean squared error (MSE)

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \ell_i(\boldsymbol{\theta}) = \frac{1}{2N}\sum_{i=1}^{N}\left(f(\mathbf{x}_i;\boldsymbol{\theta}) - y_i\right)^2 .$$

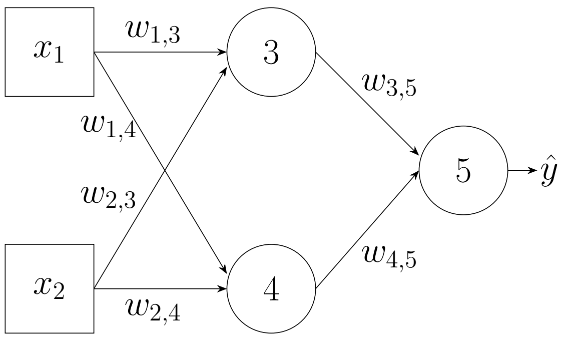


Figure 4: Illustration of a simple multilayer perceptron (MLP) with two input features, a hidden layer of two units, and a single output. Each edge represents a learnable weight, and activation functions are applied elementwise in the hidden and output layers.

For neural networks, iterative gradient-based methods are used to minimize the loss function (Prince, 2024). The gradient of the MSE loss with respect to the parameters is

$$\nabla_{\boldsymbol{\theta}}\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\nabla_{\boldsymbol{\theta}}\ell_i(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial f(\mathbf{x}_i;\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\big(f(\mathbf{x}_i;\boldsymbol{\theta}) - y_i\big)\,,$$

which is then used to update parameters. In practice, one typically samples a batch of training data and uses it to compute the loss and the corresponding gradient. This is called mini-batch gradient descent (MBGD) and is known to enhance computational efficiency and improve generalization (Sra et al., 2012). With mini-batch size B and learning rate η, the parameters at iteration t are updated using the average gradient over a sampled batch $\mathcal{B}_t$

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta\,\frac{1}{B}\sum_{i\in\mathcal{B}_t}\nabla_{\boldsymbol{\theta}}\ell_i(\boldsymbol{\theta}^{(t)})\,.$$

Several refinements of plain MBGD are widely used. Momentum accumulates an exponentially weighted average of past gradients, which dampens oscillations and accelerates convergence (Rumelhart et al., 1986). Adaptive methods such as Adam (short for Adaptive moment estimation) maintain first and second moment estimates $m^{(t)}$ and $v^{(t)}$ and update via

$$\begin{aligned}
m^{(t)} &= \beta_1 m^{(t-1)} + (1-\beta_1)\,g^{(t)}, \\
v^{(t)} &= \beta_2 v^{(t-1)} + (1-\beta_2)\,(g^{(t)})^2, \\
\hat{m}^{(t)} &= \frac{m^{(t)}}{1-\beta_1^t}, \qquad \hat{v}^{(t)} = \frac{v^{(t)}}{1-\beta_2^t}, \\
\boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \eta\,\frac{\hat{m}^{(t)}}{\sqrt{\hat{v}^{(t)}}+\epsilon}\,.
\end{aligned}$$

where $g^{(t)}$ denotes the (mini-batch) gradient and $\beta_1, \beta_2, \epsilon$ are hyperparameters (see Kingma and Ba (2014)). The use of first- and second-moment estimates effectively smooths the optimization trajectory by dampening noisy or sharply varying gradient directions, leading to more stable and directed progress (Figure 5).

The computational challenge in training neural networks lies in the fact that every parameter update requires computing all partial derivatives $\nabla_{\boldsymbol{\theta}}\ell_i$, one for each weight and bias in the model. For networks with many layers and thousands to millions (or even trillions, Wang et al., 2023) of parameters, computing these derivatives independently would be too expensive. The backpropagation algorithm overcomes this computational obstacle by exploiting the network's layered structure. Using the chain rule, the algorithm computes gradients efficiently by propagating error signals backward through the network while reusing intermediate quantities from the forward pass (Rumelhart et al., 1986). This makes backpropagation the key algorithm that enabled modern deep learning.

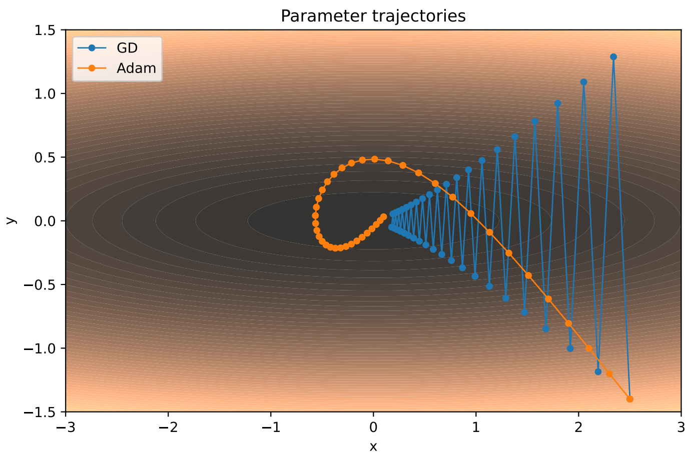


Figure 5: Parameter trajectories for gradient descent (GD) and Adam on a simple anisotropic quadratic loss. Trajectories start at the same initial point and are plotted for 40 iterations. The example (learning rates chosen for illustration: GD $\eta = 0.32$, Adam $\eta = 0.2$) demonstrates that Adam's adaptive moment estimates yield smoother, more directed descent in narrow valleys, whereas plain GD tends to oscillate across the valley.

To see how backpropagation works, consider the MLP in Figure 4. The hidden units and output are

$$\begin{aligned} z_3 &= w_{1,3}x_1 + w_{2,3}x_2 + b_3, & h_3 &= \sigma(z_3), \\ z_4 &= w_{1,4}x_1 + w_{2,4}x_2 + b_4, & h_4 &= \sigma(z_4), \\ z_5 &= w_{3,5}h_3 + w_{4,5}h_4 + b_5, & \hat{y} &= \sigma(z_5), \end{aligned}$$

and let the loss be $\ell = \frac{1}{2}(\hat{y} - y)^2$. For a weight in the output layer, the chain rule gives

$$\begin{aligned} \frac{\partial \ell}{\partial w_{3,5}} &= \frac{\partial \ell}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_5} \cdot \frac{\partial z_5}{\partial w_{3,5}} \\ &= (\hat{y} - y)\, \sigma'(z_5)\, h_3 \,. \end{aligned}$$

For a weight in the hidden layer, the dependency becomes

$$\begin{aligned} \frac{\partial \ell}{\partial w_{1,3}} &= \frac{\partial \ell}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_5} \cdot \frac{\partial z_5}{\partial h_3} \cdot \frac{\partial h_3}{\partial z_3} \cdot \frac{\partial z_3}{\partial w_{1,3}} \\ &= (\hat{y} - y)\, \sigma'(z_5)\, w_{3,5}\, \sigma'(z_3)\, x_1 \,. \end{aligned}$$

For larger networks, it is convenient to move to vector notation. Let L be the index of the output layer, let $z^{(l)}$ and $h^{(l)}$ denote the pre-activation and activation vectors at layer l, respectively, and let $\odot$ denote the Hadamard (elementwise) product. Backpropagation defines the error signals as

$$\begin{aligned} \delta^{(L)} &= \nabla_{\hat{y}} \ell \;\odot\; \sigma'(z^{(L)}) \\ \delta^{(l)} &= \left(W^{(l+1)}\right)^T \delta^{(l+1)} \;\odot\; \sigma'(z^{(l)}) \,. \end{aligned}$$

Once the error signals are known, the gradients of the loss with respect to each weight matrix follow from

$$\nabla_{W^{(l)}} \mathcal{L} = \delta^{(l)}\, (h^{(l-1)})^T \,,$$

which is simply an outer product between the error at layer l and the activations of the preceding layer. These compact matrix equations therefore show why both the forward and backward passes reduce to a sequence of matrix multiplications using the same cached values, making the algorithm extremely efficient on modern hardware (Goodfellow et al., 2016).

The complete training procedure is summarised in Algorithm 1. Training proceeds for several epochs, where an epoch refers to one full pass over the entire training dataset. During each epoch, the model is repeatedly shown mini-batches, the loss is evaluated, gradients are computed by backpropagation, and parameters are updated using an optimisation rule such as stochastic gradient descent or Adam. In practice, several issues can arise. When gradients repeatedly pass through small derivatives, they may vanish, leading to extremely slow learning in early layers. Conversely, if derivatives are large, gradients may explode (Bengio et al., 1994). Techniques such as ReLU activations, careful weight initialization, batch normalization, residual connections, and gradient clipping help mitigate these issues (Prince, 2024). Despite such challenges, neural networks remain powerful tools, and new architectural developments continue to expand their capabilities.

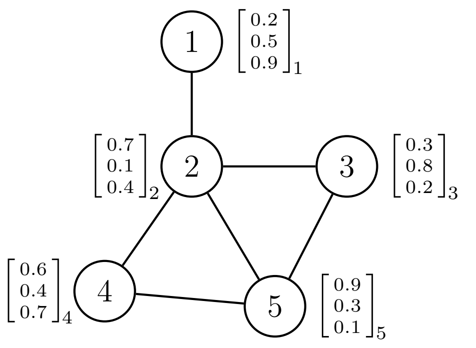
Algorithm 1 Training a neural network over multiple epochs

```
1: Input: training data $\{(\mathbf{x}_i, y_i)\}$, network $f(\cdot;\boldsymbol{\theta})$, learning rate $\eta$, number of epochs $E$
2: for $e = 1$ to $E$ do                                                    (epoch loop)
3:     for each mini-batch $\mathcal{B}$ do
4:         Compute predictions $\hat{y}_i = f(\mathbf{x}_i;\boldsymbol{\theta})$ for all $i \in \mathcal{B}$     (forward pass)
5:         Compute loss $\ell = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} L(\hat{y}_i, y_i)$
6:         Compute gradients $\nabla_{\boldsymbol{\theta}}\ell$ via backpropagation
7:         Update parameters: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}\ell$
8:     end for
9: end for
```

2.3.4 Graphs and their representation

Graphs provide a general mathematical framework for representing systems composed of discrete entities and their relationships. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where each edge encodes a pairwise connection or interaction (Gross et al., 2019). Graphs appear across a wide range of domains: social networks represent individuals linked by friendships or communication, molecules can be viewed as atoms connected by chemical bonds, and transportation networks encode roads between intersections. For our purposes, we note that pedigrees can also be viewed as graphs, with individuals as nodes and family links as edges. Graphs also come in many forms, depending on the type of relationships they encode. They may be directed or undirected, depending on whether connections have an inherent direction. Furthermore, they can be weighted, with edges carrying real-valued weights. Graphs may also include self-loops, allowing a node to be connected to itself. Even data traditionally represented as images can be interpreted as graphs, where each pixel corresponds to a node connected to neighbouring pixels through a fixed grid structure. In this sense, data such as images or time series can be viewed as special cases of graphs, restricted to Euclidean space.

Although graphs are easy to visualize conceptually, representing them in a form suitable for computation requires some form of encoding. The most common representation is the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where each entry A_{uv} denotes the presence of an edge between nodes u and v, and may also encode its strength if the graph is weighted. Directed graphs correspond to asymmetric adjacency matrices, while undirected graphs yield symmetric ones. Nonzero diagonal entries represent self-loops. The degree matrix $\mathbf{D}$ is also useful. It is a diagonal matrix whose entries denote the number of neighbours for the associated node. In addition to the graph topology, each node typically comes with associated attributes. These are collected into a node feature vector $\mathbf{x}_u \in \mathbb{R}^d$ for each node u, where d denotes the dimensionality of the feature space. Examples include physicochemical descriptors for atoms in molecular graphs, coordinates in a road-intersection network, or in our case, SNP data. Stacking all node features yields the feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$. Together, the pair $(\mathbf{A}, \mathbf{X})$ provides a complete description of a feature-augmented graph. Figure 6 illustrates a simple graph, its corresponding adjacency matrix, and the associated node feature matrix. We may also note the similarities between Figure 6 and Figure 1; these similarities will be useful later.



$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} 0.2 & 0.5 & 0.9 \\ 0.7 & 0.1 & 0.4 \\ 0.3 & 0.8 & 0.2 \\ 0.6 & 0.4 & 0.7 \\ 0.9 & 0.3 & 0.1 \end{bmatrix}$$

Figure 6: A graph with five nodes (left), each annotated with a compact 3-dimensional feature vector. The graph is undirected and without self-loops. Right: the adjacency matrix **A** and global feature matrix **X**.

2.3.5 Graph neural networks

Graph neural networks (GNNs) are neural architectures designed to operate directly on graph-structured data. The idea dates back to the work of Scarselli et al. (2009), who first formalized neural networks that iteratively update node representations based on the structure of a graph. Modern GNNs generalize this principle and encompass a broad family of models, including graph convolutional networks, attention-based graph models, and gated graph sequence neural networks. All of these take as input a graph described by an adjacency matrix **A** and a node-feature matrix **X**. GNNs are stacked in layers, just like MLPs. Their goal is to produce useful latent representations (embeddings) of nodes, edges, or entire graphs by exploiting both feature information and the relational structure imposed by the graph (Prince, 2024).

At the core of most GNN architectures is the principle of message passing (Gilmer et al., 2017). Each node u is at every layer l associated with a hidden state (or embedding) $\mathbf{h}_u^{(l)}$, initialized as its feature vector $\mathbf{h}_u^{(0)} = \mathbf{x}_u$. A GNN layer updates this embedding by combining two sources of information: the node's current embedding and the embeddings of its neighbors (its connections in the adjacency matrix). Formally, at layer l, the update takes the form

$$\mathbf{h}_u^{(l+1)} = \textsc{Update}\Big(\mathbf{h}_u^{(l)},\ \textsc{Aggregate}\big(\{\mathbf{h}_v^{(l)} : v \in \mathcal{N}(u)\}\big)\Big),$$

where $\mathcal{N}(u)$ denotes the neighbors of node u. The Aggregate function pools information from a set of neighbors and therefore must be permutation invariant, ensuring that the model does not depend on how nodes are ordered (Zaheer et al., 2018). Typical choices include elementwise mean, sum, or max aggregation, as well as more expressive, learned aggregators. The Update step often consists of an affine transformation followed by a nonlinearity, much like a regular neural network layer. The most basic GNN layer is defined as

$$\mathbf{h}_u^{(l+1)} = \sigma\left(\mathbf{W}_{\text{self}}^{(l)}\mathbf{h}_u^{(l)} + \mathbf{W}_{\text{neigh}}^{(l)} \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(l)} + \mathbf{b}^{(l)}\right), \tag{3}$$

where $\mathbf{W}_{\text{self}}^{(l)}, \mathbf{W}_{\text{neigh}}^{(l)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$ are learnable weight matrices, $\mathbf{b}^{(l)} \in \mathbb{R}^{d^{(l+1)}}$ is a learnable bias vector, σ denotes a non-linear activation function, and $d^{(l)}$ is the dimension of the feature space at layer l (Hamilton, 2022). Importantly, these weight matrices are global parameters, the same $\mathbf{W}_{\text{self}}^{(l)}$ is used for every node's self-embedding, and the same $\mathbf{W}_{\text{neigh}}^{(l)}$ is used whenever a node processes information from its neighbors. For this basic GNN, the Aggregate and Update functions are

$$\textsc{Aggregate}^{(l)}\big(\{\mathbf{h}_v^{(l)} : v \in \mathcal{N}(u)\}\big) = \mathbf{m}_{\mathcal{N}(u)}^{(l)} = \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(l)},$$

$$\textsc{Update}^{(l)}\Big(\mathbf{h}_u^{(l)},\, \mathbf{m}_{\mathcal{N}(u)}^{(l)}\Big) = \sigma\Big(\mathbf{W}_{\text{self}}^{(l)}\mathbf{h}_u^{(l)} + \mathbf{W}_{\text{neigh}}^{(l)}\mathbf{m}_{\mathcal{N}(u)}^{(l)} + \mathbf{b}^{(l)}\Big).$$

We can expand the node-level notation to the graph-level by collecting all node embeddings into a matrix $\mathbf{H}^{(l)} = \left(\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \ldots, \mathbf{h}_{|\mathcal{V}|}^{(l)}\right)^T$, and using the adjacency matrix $\mathbf{A}$. At the graph level, equation (3) becomes

$$\mathbf{H}^{(l+1)} = \sigma\Big(\mathbf{A}\,\mathbf{H}^{(l)}\mathbf{W}_{\text{neigh}}^{(l)} \;+\; \mathbf{H}^{(l)}\mathbf{W}_{\text{self}}^{(l)}\Big)\ , \tag{4}$$

where the bias term is omitted for notational simplicity, and we have again used $\mathbf{W}_{\text{self}}^{(l)}$ and $\mathbf{W}_{\text{neigh}}^{(l)}$ even though they are strictly not the same matrices as in equation (3). A usual simplification is to add self-loops to the adjacency matrix, as described in Section 2.3.4. This way, aggregation is also performed at the node level, so there is no need to define an explicit Update function. In the basic GNN layer from equation (4), this amounts to $\mathbf{W}_{\text{self}}^{(l)}$ and $\mathbf{W}_{\text{neigh}}^{(l)}$ collapsing into one common weight matrix $\mathbf{W}^{(l)}$. The update then takes the form

$$\mathbf{H}^{(l+1)} = \sigma\Big((\mathbf{A}+\mathbf{I})\,\mathbf{H}^{(l)}\mathbf{W}^{(l)}\Big)\ . \tag{5}$$

The embeddings produced by a GNN are passed to a final prediction head, whose design depends on the task at hand. In node-level prediction, the goal is to infer a label or continuous value for each node, for example, predicting atomic charges or reactivity for individual atoms in crystalline frameworks (Reiser et al., 2022), or user preferences in social recommendation networks (X. Li et al., 2023). In a regression setting, the final layer is typically a fully connected layer with a linear activation. For node classification, the final activation is usually a sigmoid (for binary tasks) or a softmax (for multi-class tasks). Edge-level prediction focuses on determining whether an edge exists, with applications such as predicting new social ties or inferring hidden links in communication networks (King and Huang, 2023). Here, the prediction head generally takes the embeddings of two nodes, $\mathbf{h}_u$ and $\mathbf{h}_v$, and combines them using a function such as a dot product, followed by a sigmoid activation. Finally, graph-level prediction assigns a label or value to an entire graph, such as the solubility of a molecule (Ahmad et al., 2024). In this setting, node embeddings are first pooled into a single graph-level representation using, for example, mean, sum, or max pooling. This graph embedding is then passed through a standard feed-forward prediction head. In this report, we focus exclusively on node-level regression.

GNNs offer several practical strengths that have contributed to the architecture's popularity. Firstly, they scale well to large graphs as the operation $(\mathbf{A}+\mathbf{I})\,\mathbf{H}^{(l)}\mathbf{W}^{(l)}$ is a sparse matrix multiplication with complexity $\mathcal{O}(|\mathcal{E}|d^{(l+1)}d^{(l)})$, that is linear in the number of edges (Kipf and Welling, 2017). Secondly, a distinctive feature of the basic GNN introduced is that unlabeled nodes also contribute to the representation of labeled nodes during training. This differs from standard supervised learning, in which training and test data are clearly separated. Therefore, using the forward propagation defined in equation (5) is often referred to as semi-supervised learning. Semi-supervised approaches can enhance accuracy when labeled data is scarce (Chapelle et al., 2006). Finally, the GNN formulation generalizes directly to weighted graphs. Replacing $\mathbf{A}$ with a weighted adjacency matrix allows edges to encode degrees of similarity rather than simple connectivity.

In summary, the role of the graph structure is to influence the nodes' embeddings by allowing information from similar or structurally related neighbours to flow into their representation. After L layers, the resulting embedding $\mathbf{h}_u^{(L)}$ encodes not only the original features of node u but also information from nodes within its L-hop neighbourhood, as displayed in Figure 7. The cross-sample information flow is what distinguishes GNNs from ordinary neural networks, and the intuition connects naturally to genomic prediction. Just as the animal model uses a relationship matrix to guide the estimation of breeding values, a GNN can use a graph constructed from pedigrees or genomic relationships to guide the sharing of information across individuals. By learning embeddings that capture both individual SNP features and patterns of relatedness, GNNs have the potential to enhance prediction accuracy beyond what either source alone could provide.

2.3.6 Graph convolutional networks

The basic GNN layer in equation (5) shows how nodes aggregate information from their neighbours, but it also reveals a practical issue. Simply summing neighbour embeddings can cause nodes with

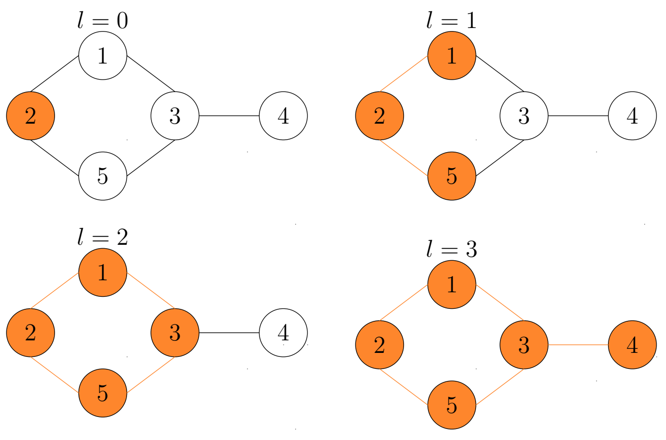


Figure 7: Illustration of information propagation in a graph neural network across layers. Each of the four graphs depict the same graph at different layers l, and how the receptive field of node 2 expands.

many neighbours to dominate the update. If $|\mathcal{N}(u)| \gg |\mathcal{N}(u')|$, then for any reasonable vector norm $\|\cdot\|$ we typically have

$$\left\| \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(l)} \right\| \gg \left\| \sum_{v' \in \mathcal{N}(u')} \mathbf{h}_{v'}^{(l)} \right\|,$$

making the representation of u disproportionately large purely because it has many neighbours. Graph convolutional networks (GCNs) provide a principled solution to this problem by introducing a carefully designed normalization scheme that balances contributions across nodes.

A natural first idea is to average neighbour embeddings rather than sum them. However, Kipf and Welling (2017) argued that an even better alternative is symmetric normalization, which takes into account not only the degree of the central node u but also the degrees of its neighbours. With their suggested normalization, the aggregated message becomes

$$\mathbf{m}_{\mathcal{N}(u)}^{(l)} = \sum_{v \in \mathcal{N}(u)} \frac{\mathbf{h}_v^{(l)}}{\sqrt{|\mathcal{N}(u)|\,|\mathcal{N}(v)|}}. \tag{6}$$

This weighting gives relatively more influence to neighbours with fewer connections (low-degree nodes), while preventing high-degree neighbours from overwhelming the update. Intuitively, if u is connected to nodes v and v' where $|\mathcal{N}(v)| \gg |\mathcal{N}(v')|$, then the link to v' is much more unique and should be weighted higher. To write the update compactly, self-loops are first added so that every node also contributes its own previous embedding. Let $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ denote the adjacency matrix with self-loops, and let $\tilde{\mathbf{D}}$ be the corresponding degree matrix. The GCN update can then be written in matrix form as

$$\mathbf{H}^{(l+1)} = \sigma\Big(\tilde{\mathbf{D}}^{-1/2}\,\tilde{\mathbf{A}}\,\tilde{\mathbf{D}}^{-1/2}\,\mathbf{H}^{(l)}\,\mathbf{W}^{(l)}\Big), \tag{7}$$

where $\mathbf{W}^{(l)}$ is a trainable weight matrix and σ is a nonlinearity such as ReLU. They are called convolutional because the layer update actually approximates a spectral graph convolution. Full details of this construction are beyond the scope of this report, but an introduction is provided in Hamilton (2022).

Since the GCN described is only a slight modification of the more basic GNN introduced in the previous section, it also possesses the main benefits outlined earlier. Despite these strengths, GCNs also have significant limitations. Because the aggregation rule in equation (7) is fixed and uniform across nodes, repeatedly applying the operator $\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ can lead to oversmoothing. Oversmoothing describes the effect that embeddings become increasingly similar as depth increases (Hamilton, 2022). Furthermore, a consequence of the semi-supervised nature of GCNs is that they are fundamentally transductive. The propagation rule depends explicitly on the full adjacency matrix available during training, meaning that the learned model is tied to that specific graph. If new nodes are introduced or edges are modified, the normalization changes, and the learned operator is no longer valid. Where this could be an advantage if the whole graph is known, it also means classical GCNs cannot naturally generate embeddings for unseen nodes without retraining on the new graph.

2.3.7 Inductive learning and GraphSAGE

The transductive limitation of GCNs naturally motivates architectures designed for inductive settings, where new nodes may appear after training or where the test nodes may belong to an entirely different graph, as shown in Figure 8. Although transductive and inductive GNNs can be used for similar downstream tasks, such as node regression or classification, they fundamentally learn different objects. A transductive model effectively learns "given this particular neighbourhood structure and these training nodes, this is how to update their embeddings." In contrast, an inductive model is not tied to any particular set of nodes. Instead of memorizing representations for the training nodes, it learns a function that maps a node's features and local neighbourhood to an embedding. Conceptually, it learns "given any neighbourhood, this is how to combine its information with the node's own features to produce a useful representation." Once such a function has been learned, it can be applied directly to previously unseen individuals or even to entirely new graphs. In a genomic prediction setting, this inductive property can be particularly attractive. A model that learns transferable aggregation rules may generalize more naturally across populations.

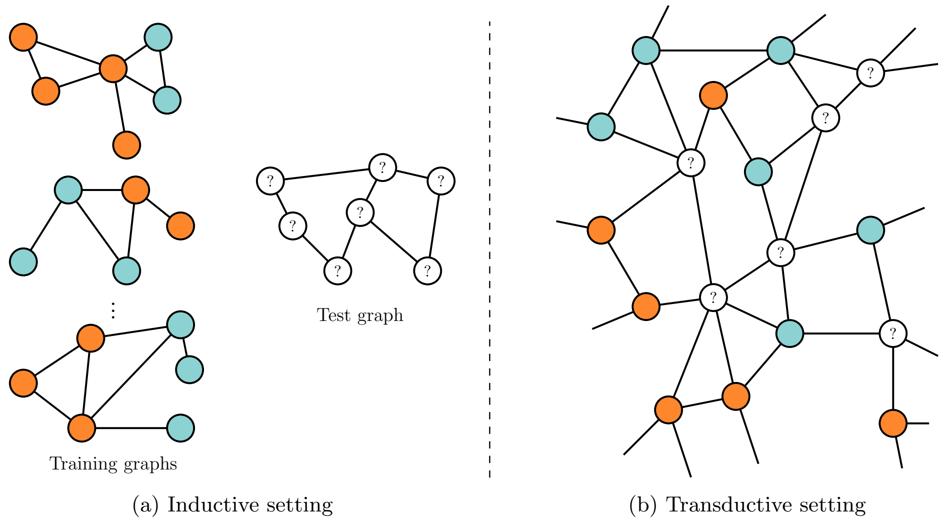


Figure 8: Inductive vs. transductive setting in a node classification problem. Colors represent labeled data while question marks are unlabeled. (a) In the inductive setting, training and test graphs may be separate (shown), or new nodes and edges are added after training. (b) In the transductive setting, learning and prediction occur on a single connected graph.

One popular inductive framework is GraphSAGE (SAmple and aggreGatE), introduced by Hamilton et al. (2018). GraphSAGE constructs node embeddings iteratively over L layers, much like GCNs. However, two modifications distinguish it from GCNs. Firstly, neighbourhoods are not taken in full. For each node u, GraphSAGE samples a fixed-size subset of neighbours. Secondly, neighbourhood information is aggregated using a potentially trainable operator, rather than a propagation rule derived from the normalized adjacency matrix. This general aggregation function can (much like in Section 2.3.5) be written as

$$\mathbf{m}_{\mathcal{N}(u)}^{(l)} = \textsc{Aggregate}_l\left(\{\mathbf{h}_v^{(l)} : v \in \mathcal{N}(u)\}\right), \tag{8}$$

where $\mathcal{N}(u)$ is the sampled neighbourhood of u. Thus, $\mathbf{m}^{(l)}$ describes the "message" node u recives at layer l. With the incoming message $\mathbf{m}^{(l)}$, representations are updated via

$$\mathbf{h}_u^{(l+1)} = \sigma\Big(\mathbf{W}^{(l)}\big[\mathbf{h}_u^{(l)} \,\|\, \mathbf{m}_{\mathcal{N}(u)}^{(l)}\big]\Big) \ , \tag{9}$$

where $[\cdot \,\|\, \cdot]$ denotes concatenation and σ is a nonlinearity. Conceptually, the two steps outlined in equations (8) and (9) yield a learnable extension of the GCN update rule in up to three ways. The first and most obvious way it learns to combine information is by using a learnable aggregation function. Secondly, since the AGGREGATE function does not aggregate a specific set of nodes, but a general set of nodes, $\mathcal{N}(u)$, it generalizes naturally to unseen neighbourhoods. Finally, the information from the node itself is handled differently. Earlier, the adjacency matrix had self-loops. That meant the node itself was included in the aggregation, and the weight matrix $\mathbf{W}^{(l)}$ only learned how to combine already aggregated information. Now, vector concatenation is used, serving as a skip-connection that preserves information about the node itself at each depth. This means the weight matrix now also learns how to combine the node's own information with the aggregated neighbourhood information directly.

Hamilton et al. (2018) propose several choices for the aggregation function. The simplest are the mean and max aggregators, which compute the element-wise average or maximum of neighbour representations, respectively, and include no parameters. More expressive alternatives include an LSTM-based aggregator, inspired by recurrent neural networks, and MLP aggregators, which first pass each neighbour representation through a small MLP and then apply an element-wise pooling operator (e.g., max). For reference, Algorithm 2 reproduces the GraphSAGE forward-pass procedure (slightly adapted from the original paper's notation). This is the algorithm used for both training and inference to generate node embeddings. In practice, GraphSAGE is trained with a loss tailored to the downstream prediction task.

Algorithm 2 GraphSAGE Embedding Generation (Hamilton et al., 2018)

```
1: Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, node features $\{\mathbf{x}_u\}_{u\in\mathcal{V}}$, depth $L$, weight matrices $\{\mathbf{W}^{(l)}\}_{l=1}^{L}$, aggregators $\{\text{AGGREGATE}_l\}_{l=0}^{L-1}$
2: $\mathbf{h}_u^{(0)} \leftarrow \mathbf{x}_u$ for all $u \in \mathcal{V}$
3: for $l = 0, \ldots, L-1$ do
4:     for each node $u \in \mathcal{V}$ do
5:         Sample neighbourhood $\mathcal{N}(u) \subseteq \mathcal{V}$
6:         $\mathbf{m}_{\mathcal{N}(u)}^{(l)} \leftarrow \text{AGGREGATE}_l\big(\{\, \mathbf{h}_v^{(l)} : v \in \mathcal{N}(u) \,\}\big)$
7:         $\mathbf{h}_u^{(l+1)} \leftarrow \sigma\Big(\mathbf{W}^{(l)}\big[\,\mathbf{h}_u^{(l)} \,\|\, \mathbf{m}_{\mathcal{N}(u)}^{(l)}\big]\Big)$
8:     end for
9:     Optionally normalize: $\mathbf{h}_u^{(l+1)} \leftarrow \mathbf{h}_u^{(l+1)}/\|\mathbf{h}_u^{(l+1)}\|_2$
10: end for
11: return $\mathbf{z}_u = \mathbf{h}_u^{(L)}$ for all $u \in \mathcal{V}$
```

2.3.8 Regularization and hyperparameter optimization

As discussed in Section 2.3.2, the primary objective of supervised learning is not merely to minimize the empirical risk on the training data, but to learn a function that generalizes well to unseen data. A model that minimizes the training loss too aggressively risks overfitting, a state where it captures noise rather than the underlying signal. This manifests as the high-variance regime of the bias-variance trade-off (James et al., 2021). To navigate this trade-off, specific constraints and techniques, collectively known as regularization, are employed. Goodfellow et al. (2016) defines regularization as "any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error".

One way to prevent overfitting is to use so-called explicit regularization, which amounts to adding terms to the loss function to constrain the model's complexity (Bühlmann and Geer, 2011). A standard approach is weight decay, or L_2 regularization, which adds a penalty term to the loss function proportional to the squared magnitude of the weights, such that the new loss function becomes

$$\mathcal{L}_{reg}(\theta) = \mathcal{L}(\theta) + \frac{\lambda}{2}\|\mathbf{w}\|^2 .$$

In this modified loss function, λ is a parameter controlling the strength of the regularization. Mathematically, this is equivalent to the ridge penalty discussed in the context of marker-based regression, effectively shrinking weights toward zero to prevent any single feature from exerting a disproportionate influence (James et al., 2021). Beyond parameter shrinkage, there exist many implicit regularization techniques. Dropout is often employed in neural networks, a technique in which individual neurons are randomly "dropped" (set to zero) during training with probability p. This prevents units from co-adapting too strongly and forces the network to learn robust features that are redundant across different pathways (Goodfellow et al., 2016). A different approach used to prevent overfitting in neural networks is early stopping. Since neural networks are iterative learners, the training error typically decreases monotonically, whereas the validation error often exhibits a U-shaped curve (Figure 2). Early stopping exploits this by monitoring performance on a held-out validation set and halting training when the error fails to improve for a specified number of epochs.

The concept of regularization blends naturally into the problem of optimizing hyperparameters. Hyperparameters are parameters that control either the architecture of the model or the optimization process itself. They are not learned by the model from data, but are fixed before training begins. Hyperparameters have been mentioned previously in this report, but without further elaboration. However, all the regularization techniques introduced can in fact be framed as hyperparameter tuning problems. For example, λ in L_2 regularization is a hyperparameter, and adding L_2 regularization amounts to finding the optimal $\lambda \in [0, \infty)$. In a similar fashion, regularization using dropout and early stopping comes down to finding the optimal dropout probability $p \in [0, 1)$ and the optimal number of epochs $E \in \mathbb{N}$.

Another significant hyperparameter is the learning rate η, which determines the step size taken during parameter updates. A value too large can cause the optimization to diverge, while a value too small results in slow convergence or stagnation (Goodfellow et al., 2016). Coupled with the learning rate is the batch size B, which defines the number of samples used to compute the gradient estimate in each iteration. Smaller batch sizes introduce noise into gradient estimation (which can also have a regularization effect, as it may help escape saddle points). In contrast, larger batches yield more stable estimates and enable greater computational efficiency (Goodfellow et al., 2016). Furthermore, the choice of optimizer (MBGD, Adam, etc.) introduces specific hyperparameters, such as momentum coefficients or decay rates for moment estimates (β_1, β_2), which often require tuning to achieve optimal performance. Beyond the training dynamics, the physical structure of the model is defined by architectural hyperparameters. These control the model's capacity, including the number of hidden layers (depth) and the number of units per layer (width). In the specific context of Graph Neural Networks (GNNs), depth is interpreted as the number of layers L, which corresponds to the number of "hops" over which information is aggregated. Consequently, a deeper GNN has a larger receptive field (Hamilton, 2022). However, as discussed in Section 2.3.6, excessive depth can lead to oversmoothing. Finally, when the graph structure is derived from data rather than given through a fixed topology, the graph construction itself yields hyperparameters. These include choosing an upper limit for the number of neighbours, or whether edges should be weighted or binary. These choices effectively change the topology on which message passing occurs and must be optimized alongside the network architecture.

Selecting the optimal hyperparameter combination is a difficult problem, as the relationship between configurations and the validation loss is complex, non-convex, and expensive to evaluate. While simple approaches such as grid search and random search are common, they are often computationally inefficient. Bayesian optimization offers a more principled approach by constructing a probabilistic surrogate model of the objective function. One specific algorithm within this family is the Tree-structured Parzen Estimator (TPE) (Bergstra et al., 2011). Unlike standard Bayesian optimization, which typically models the probability of the observation given the hyperparameters $p(y|\lambda)$ using Gaussian Processes, TPE models the likelihood of the hyperparameters given the score $p(\lambda|y)$. TPE operates by splitting the observations into a "good" group (lower loss) and a "bad" group (higher loss) based on a percentile $\tilde{y}$ of the observed loss values. It then estimates two separate density distributions for the hyperparameters

$$p(\lambda|y) = \begin{cases} l(\lambda) & \text{if } y < \tilde{y} \\ g(\lambda) & \text{if } y \geq \tilde{y} \end{cases} .$$

Here, $l(\lambda)$ represents the density of hyperparameters that yielded promising results, while $g(\lambda)$ represents those that yielded poor results. To propose the next set of hyperparameters to evaluate, TPE seeks to maximize the Expected Improvement (EI), which is proportional to the ratio $l(\lambda)/g(\lambda)$. Intuitively, the algorithm samples hyperparameters that are highly likely under the "good" distribution and unlikely under the "bad" distribution, efficiently guiding the search toward the most promising regions of the hyperparameter space. Further details on TPE are beyond the scope of this report. However, it is important to note that more sophisticated methods than random search exist.

3 Methods

3.1 Data and study system

The data analysed in this report originate from a long-term individual-based study of house sparrows (*Passer domesticus*) inhabiting an insular meta-population along the Helgeland coast in northern Norway. Fieldwork has been conducted annually since 1993 (Jensen, Steinsland et al., 2008; Lundregan et al., 2018), providing a uniquely rich dataset with repeated captures, morphological measurements, and near-complete life histories for thousands of individuals. Each bird is fitted with a numbered metal ring and a set of coloured plastic rings, allowing reliable identification across years. The study system comprises numerous islands that differ in habitat and environmental conditions. In our experiments, 11 different islands will be studied, with the number of individuals ranging from 88 to 1033. The islands can broadly also be divided into inner and outer groups in the system. Inner islands, closer to the mainland, are typically agricultural with access to farms, whereas outer islands are more exposed, lack farms, and impose harsher environmental conditions on the birds (Muff et al., 2019; Ranke et al., 2021). Despite these environmental differences, dispersal occurs between islands, forming a genetically connected meta-population. The island structure is shown in Figure 9.

For genomic data, a 70K SNP array has been used, which contains SNP markers genotyped for 3,379 unique individuals. After quality control for call rate (> 0.95) and minor allele frequency > 0.01, 65,247 markers remain. The markers provide genome-wide coverage with data encoded as 0/1/2 based on minor allele count. Because individuals may be captured multiple times across different years, the full dataset contains more phenotype records than unique genotypes. However, all genotype information is indexed by a unique individual identifier, `ringnr`. Three morphological traits are considered for prediction: body mass, measured to the nearest 0.1 g, tarsus length, measured to the nearest 0.01 mm, and wing length, measured to the nearest millimetre. These traits represent commonly studied morphological measures in ecological and quantitative-genetic analyses. They are also repeatedly measured, meaning that many individuals contribute several trait observations across years. The research group has carried out standard data processing to correct for observer effects and ensure consistency in measurement protocols (Niskanen et al., 2020).

In addition to SNP genotypes and phenotypes, a set of auxiliary variables is available and used as covariates in the prediction models. These include sex, month of capture, age at capture, the island where the individual hatched and where it was currently observed, as well as genomic measures such as the inbreeding coefficient (`FGRM`) and the genetic ancestry proportions `Outer` and `Other`. A summary of these variables is provided in Table 2. These variables will be important in accounting for non-genetic effects.

Variable	**Description**
Sex	Sex of individual (1 = male, 2 = female)
FGRM	Inbreeding coefficient
Month	Month of capture
Age	Age of individual at capture
Outer	Proportion of genetic material from *outer* islands
Other	Proportion of genetic material from *other* islands
Hatch island	Island where the individual was born
Hatch year	Year the individual was born
Island current	Island where the individual was captured
Ringnr	Unique individual ID

Table 2: Description of house sparrow variables used in the report.

Island	Tarsus length	Body mass	Wing length
Nesøy	131	133	132
Myken	89	88	89
Træna	269	272	267
Selvær	222	222	220
Gjerøy	544	543	541
Hestmannøy	1022	1033	1015
Indre Kvarøy	370	371	367
Onøy/Lurøy	267	266	266
Lovund	148	147	148
Sleneset	189	189	186
Aldra	175	173	174

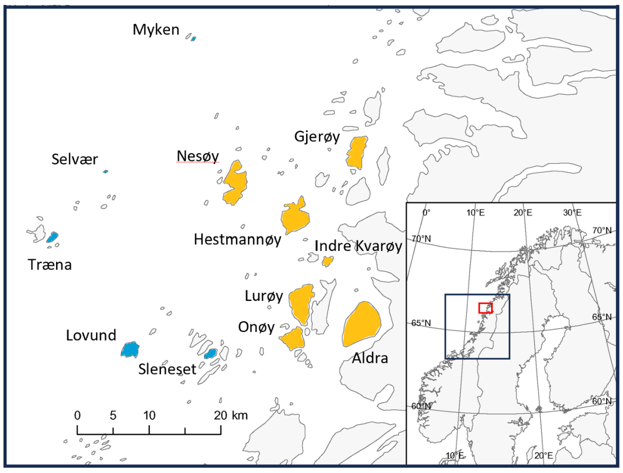


Figure 9: Island structure of the Helgeland house sparrow meta-population. Left: Number of genotyped individuals per island, with corresponding phenotype measurements. Right: Map of the island system, where inner (farm-associated) islands and outer (more exposed) islands are color-coded.

3.2 Animal model baseline

As a baseline for evaluating the machine-learning models, we use the GBLUPs obtained from the genomic animal model. These results are taken from the study of Aase et al. (2025), which uses the same dataset as this report. Consequently, the description below follows their modelling framework.

The model structure corresponds to the standard linear mixed model used for GBLUP, as introduced in Section 2.2.1. In its generic form, the model can be written as

$$\mathbf{y} = \mu \mathbf{1}_n + \mathbf{X}\boldsymbol{\beta} + \mathbf{D}\mathbf{r} + \mathbf{a} + \boldsymbol{\varepsilon}\,, \tag{10}$$

where $\mathbf{y}$ is the vector of phenotypes, $\boldsymbol{\beta}$ contains fixed effects, $\mathbf{u}$ contains random effects, and $\mathbf{X}$ and $\mathbf{Z}$ are their respective design matrices. $\mathbf{a}$ denotes the additive genetic (breeding) value, which is also a random effect, and $\boldsymbol{\varepsilon}$ represents independent residual errors. In the house sparrow analysis, the fixed effects comprised sex (0 = female, 1 = male), month of capture (categorical; January = 1 through November = 11), and age in years (1–11).

The additive genetic effects were assumed to follow $\mathbf{a} \sim \mathcal{N}(\mathbf{0}, V_A\mathbf{G})$, where V_A is the additive genetic variance and $\mathbf{G}$ is the genomic relationship matrix (GRM) computed from SNP data. In addition, the model includes several other random effects that account for known sources of non-genetic variation:

(i) A hatch-year effect, capturing environmental conditions specific to the year an individual was born.

(ii) An island-of-measurement effect, accounting for differences among islands in habitat, food availability, and weather.

(iii) An individual-specific permanent environmental effect, absorbing consistent non-genetic differences among individuals.

(iv) A measurement-session effect, grouping all measurements of an individual taken on the same day and thereby modelling short-term environmental fluctuations shared within a session.

Although Section 2.2.1 derived the GBLUP using Henderson's mixed model equations, an equivalent Bayesian formulation exists. By placing Gaussian priors on the fixed effects and appropriate priors on the variance components, the model becomes a latent Gaussian model (LGM). Integrated

Nested Laplace Approximation (INLA, Rue et al., 2009) provides a computationally efficient alternative to MCMC for approximating the posterior distributions in such models, and was the method used in Aase et al. (2025). In their analysis, fixed effects were assigned independent $\mathcal{N}(0, 10^3)$ priors. At the same time, all variance components received penalized complexity priors calibrated so that each had a 5% prior probability of explaining more than half of the phenotypic variance. The posterior medians of the random genetic effects yield the genomic estimated breeding values (GEBVs), which we adopt here as our baseline for comparison. Details of LGMs and INLA are omitted, as it is sufficient for our purposes to note that this Bayesian formulation provides the GEBVs used throughout this report.

3.3 Machine learning and graph-based models

3.3.1 Phenotype adjustment

The goal of genomic prediction is to predict an individual's genetic merit for a trait. If we were to train a machine-learning model directly on raw phenotypes using genotype data as input, predictive performance would be limited by the fact that not all phenotypic variation is genetically determined (Section 2.1.3). As discussed previously, the narrow-sense heritability h^2 quantifies the proportion of phenotypic variance attributable to additive genetic effects. In the Helgeland house sparrow system, the narrow-sense heritability of body mass, tarsus length, and wing length has been estimated at 0.175, 0.475, and 0.406, respectively (Jensen, Sæther et al., 2003). Even if machine-learning models are, in principle, capable of capturing non-additive effects – thus relating more closely to the broad-sense heritability h_B^2 – these values suggest that a substantial proportion of phenotypic variance arises from environmental or non-genetic factors. Direct prediction of phenotypes may therefore mislead the model to learn noise rather than a meaningful genetic signal.

To mitigate this issue, we adopt a two-step procedure in which we first remove environmental and other non-genetic variation through a linear mixed model (LMM), and then train machine-learning methods on the resulting genetically informed phenotypes. For each trait, we fit the model

$$\mathbf{y} = \mu \mathbf{1}_n + \mathbf{X}\boldsymbol{\beta} + \mathbf{D}\mathbf{r} + \boldsymbol{\gamma} + \boldsymbol{\varepsilon}, \tag{11}$$

where $\boldsymbol{\beta}$ contains fixed effects and $\mathbf{r}$ denotes additional random effects with corresponding design matrices $\mathbf{X}$ and $\mathbf{D}$. We include the same fixed effects as in the animal model, sex, month of capture, and age, along with the individual inbreeding coefficient. The random effects mirror those used in the animal model. The term $\boldsymbol{\gamma}$ represents an individual-level random effect (indexed by ring number) intended to capture the genetic signal relevant for prediction. Importantly, $\boldsymbol{\gamma}$ is not the same as the breeding value vector $\mathbf{a}$ from equation (10). Instead of assuming a structured covariance $\mathbf{G}$, we model $\boldsymbol{\gamma} \sim \mathcal{N}(\mathbf{0}, \sigma_G^2 \mathbf{I})$, so that $\boldsymbol{\gamma}$ absorbs all individual-specific variation not explained by fixed or other random effects. This unstructured genetic effect is used purely as a statistical device for phenotype adjustment.

After fitting the model in equation (11), we extract the posterior means (or BLUPs) $\hat{\gamma}_i$ for each individual and define the adjusted phenotype $y_i^* = \hat{\gamma}_i$. This yields a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i^*)\}_{i=1}^N$, where $\mathbf{x}_i$ denotes the genotype vector for individual i. These adjusted phenotypes serve as targets for the machine-learning models.

3.3.2 Graph construction

Graph neural networks require an adjacency matrix as input, but our dataset consists only of individuals and their SNP vectors, and is therefore not inherently graph-structured. In practice, the adjacency matrix determines which individuals exchange information during message passing. Its construction is therefore a modelling choice that can substantially influence predictive performance. Several matrices can serve as the starting point for defining pairwise similarity. The most natural option in genomic prediction is the genomic relationship matrix (GRM) $\mathbf{G}$, constructed from SNP data as described in Section 2.2.1. The GRM provides estimates of realized genetic relatedness, capturing how similar two individuals are in terms of their genome-wide allele sharing. In this report, we consider only the GRM, but other choices are possible. Examples include pairwise Euclidean distances between genotype vectors (Kihlman et al., 2024), or pedigree-based relationships when pedigree data are available. Regardless of the initial choice, a sparsification step is needed because these matrices are dense and therefore unsuitable for GNNs, whose computational efficiency relies on sparse adjacency. In this report, we adopt two sparsification strategies: thresholding and k-nearest neighbours (k-NN).

In the k-NN method, the GRM is transformed into a cosine-like similarity matrix according to

$$S_{ij}^{\cos} = \frac{G_{ij}}{\sqrt{G_{ii}\,G_{jj}}},$$

where small numerical constants are added to denominators when needed to avoid instability. This normalization rescales pairwise relatedness to the interval $[0, 1]$ and removes individual-specific variation in diagonal entries of G (individuals may have $G_{ii} > 1$ due to inbreeding). A distance matrix is then defined by $\Delta_{ij} = 1 - S_{ij}^{\cos}$. For each individual i, the k smallest distances Δ_{ij} are selected, producing k directed edges. Weighted graphs assign $A_{ij} = S_{ij}^{\cos}$ for these neighbours, whereas unweighted graphs set $A_{ij} = 1$. Because the directed k-NN procedure does not guarantee symmetry, the resulting adjacency is symmetrized either by taking the union of edges or the mutual intersection. Applying the union, an edge is kept if either i chooses j or j chooses i, whereas in the mutual case, an edge is kept only if both directions agree. The union rule produces a denser graph, while the mutual rule yields a sparser graph, emphasising reciprocal similarity. Note that after applying the union or mutual rule, individuals no longer necessarily have exactly k neighbours.

The alternative sparsification approach is thresholding. Given a symmetric similarity matrix S (either the GRM itself or the cosine-normalized GRM), entries below a threshold τ are set to zero, and those above are retained. This method preserves symmetry automatically but does not enforce a fixed number of neighbours. Some individuals may have many connections, others very few. Threshold graphs can therefore highlight only the strongest genetic relationships while allowing the neighbourhood size to vary across individuals. Not requiring all individuals to have the same number of neighbours can be advantageous when the number of closely related individuals varies across the population. For example, if an individual has n closely related individuals, but the k-NN method is applied with $k > n$, the individual is forced to have $k - n$ links that possibly mainly contribute noise. Of course, mutual symmetrization and weighted edges can help mitigate this issue.

After sparsification, additional processing steps may be applied. Edge weights may be dichotomized so that only the presence of a relationship is encoded. Dichotomization discards information that may be useful; however, it may also speed up computations and potentially remove oversensitivity to highly weighted edges. Self-loops can also be added if desired, and normalization can be applied. A potential normalization option is the GCN normalization in equation (6). Applying GCN normalization is, of course, not necessary if a GCN is used, as the normalization will be done in each layer as shown in equation (7). However, if an inductive model is used, GCN normalization could yield similar benefits to those discussed for the GCN in Section 2.3.6. In any case, the multitude of possibilities highlights that graph construction introduces several hyperparameters. The initial similarity measure, whether cosine-like normalization is applied also in the thresholding method, the value of k or τ, whether the graph is weighted or binary, the symmetrization rule, whether self-loops are added, and whether GCN normalization is applied. Each choice affects how information flows between individuals, and there is no universally optimal configuration. In this report, we therefore treat graph construction as part of the model tuning procedure.

3.3.3 GNN architectures

The graph-based models were implemented using `PyTorch Geometric` (PyG). PyG is a `PyTorch`-based library that makes it easy to implement and train graph neural networks for a broad range of applications on structured data. In our implementation, all models take as input the adjusted phenotypic dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i^*)\}_{i=1}^{N}$ together with a sparse adjacency representation of the graph. For GNNs, training proceeds in a node-regression setting. Each node corresponds to an individual with SNP vector $\mathbf{x}_i$, and the model outputs a scalar prediction $\hat{y}_i$. The GCN model is a multilayer stack of `GCNConv` operators, each implementing the normalized message-passing update described in Section 2.3.6. GraphSAGE is implemented using PyG's `SAGEConv` operator, which performs the aggregation-and-update step defined in Section 2.3.7. Note that neighbourhood sampling was not used due to issues with PyG's `NeighborLoader` class, which caused runtimes to surge. In addition, `SAGEConv` does not support weighted edges. We use an L-layer stack with hidden widths $d_1, \ldots, d_L$. The PyG implementation used here applies a ReLU activation. Dropout is used at each layer for regularization, and weight-decay (L_2 normalization) is also employed. After the final layer, a linear output layer maps $\mathbf{h}_i^{(L)}$ to a single prediction for node i. The model therefore consists of L layers followed by a fully connected regression head.

To isolate the contribution of graph structure, we also fit a purely feature-based feed-forward MLP with the same regression head. The MLP processes each $\mathbf{x}_i$ independently, optionally applying batch normalization and dropout. The final linear layer outputs the prediction. Since the MLP receives no graph information, performance differences relative to the GNNs quantify the value added by graph-based context.

3.4 Training and evaluation

3.4.1 Loss functions and metrics

For the MLPs, training is performed using mini-batch stochastic gradient descent with the Adam optimizer. For the GNNs, the full graph is passed through the model at once, such that $|\mathcal{B}| = N_{\text{tr}}$. For a batch $\mathcal{B}$, predictions $\hat{y}_i$ are compared to the adjusted phenotypes y_i^* using either the mean squared error (MSE),

$$\mathcal{L}_{\text{MSE}}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (\, \hat{y}_i - y_i^* \,)^2 \,,$$

or the mean absolute error (MAE),

$$\mathcal{L}_{\text{MAE}}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} |\, \hat{y}_i - y_i^* \,| \,,$$

both of which are included in the hyperparameter search (see Table 3). MSE penalizes large deviations more strongly and often leads to smoother optimization, while MAE is more robust to heavy-tailed noise and outlying individuals (Goodfellow et al., 2016). Allowing the hyperparameter search to choose between the two loss functions enables the model selection procedure to adapt to the noise characteristics of each phenotype.

It is important to note that neither the animal model nor the machine-learning models aim to predict the raw phenotype, and they do not predict the same quantity. The GBLUPs are estimates of additive genetic values. In contrast, the machine-learning models are trained to predict the adjusted phenotypes y_i^* representing the total individual-specific genetic component extracted by the LMM, which may also contain non-additive components. For a fair, common evaluation criterion, the standard in the literature is to assess predictive ability via the Pearson correlation between predictions and the actual phenotypes (averaged across repeated measurements for each individual). Given predicted values $\hat{\mathbf{w}}$, which is either $\mathbf{y}^*$ or $\hat{\mathbf{a}}$, and observed values $\mathbf{y}$, the accuracy metric is

$$\text{Corr}(\hat{\mathbf{w}}, \mathbf{y}) = \frac{\sum_i (\hat{w}_i - \bar{\hat{w}})(y_i - \bar{y})}{\sqrt{\sum_i (\hat{w}_i - \bar{\hat{w}})^2} \sqrt{\sum_i (y_i - \bar{y})^2}} \,.$$

Correlation is a scale-free metric; it does not measure calibration but rather the concordance in ordering between predicted and observed outcomes. Because our interest lies in selecting models that best recover the genetic signal explaining phenotypic differences, correlation is an informative measure for this context.

Category	Parameter	Values / Range (Optuna search)
Model	`hidden_dims`	Categorical: `[[128, 64], [256, 128], [512, 256], [512, 128], [256, 128, 64], [512, 64]]`
Model	`batch_norm`	Categorical: `[true, false]`
Model (GraphSAGE)	`sage_aggr`	Categorical: `["mean", "max"]`
Model (GraphSAGE)	`sage_normalize`	Categorical: `[false, true]`
Graph	`graph_mode`	Categorical: `["knn", "threshold"]`
Graph	`weighted_edges`	Categorical: `[false, true]`
Graph	`self_loops`	Categorical: `[false, true]`
Graph	`grm_norm`	Categorical: `["gcn", "cosine", "none", "cosine_then_gcn"]`
Graph (k-NN)	`knn_k`	Integer, uniform in $[1, 10]$
Graph (k-NN)	`symmetrize_mode`	Categorical: `["mutual", "union"]`
Graph (threshold)	`threshold`	Continuous, uniform in $[0.2, 0.9]$ (GRM edge threshold)
Training	`lr`	Log-uniform in $[10^{-6}, 10^{-3}]$
Training	`weight_decay`	Log-uniform in $[10^{-7}, 10^{-3}]$
Training	`dropout`	Continuous, uniform in $[0.1, 0.6]$
Training	`epochs`	Integer, uniform in $[50, 350]$
Training	`optimizer`	Categorical: `["adam"]`
Training	`loss`	Categorical: `["mse", "mae"]`
Feature selection	`use_snp_selection`	Categorical: `[true, false]`
Feature selection	`num_snps`	Integer, uniform in $[2000, 60000]$ (only if `use_snp_selection = true`)

Table 3: Overview of hyperparameters tuned with Optuna in the nested cross-validation experiments.

3.4.2 Nested cross-validation

The objective of the empirical analysis is to compare several machine-learning models against the GBLUP benchmark obtained from Aase et al. (2025). To ensure statistically meaningful comparisons, two principles are crucial. First, all models must be evaluated under identical data partitions so that differences in accuracy reflect modelling ability rather than differences in the underlying train–test split. Second, all hyperparameters must be tuned without any information from the test set influencing the model. Since each model architecture involves several hyperparameters, ranging from learning rates and network depths to graph-construction choices (see Table 3), a principled tuning procedure is necessary. Nested cross-validation satisfies both requirements by embedding an inner cross-validation loop (used exclusively for hyperparameter tuning) inside an outer cross-validation loop (used for unbiased performance estimation). This prevents any leakage of test information into model selection and yields a realistic measure of generalization error. The procedure is summarized in Algorithm 3. For hyperparameter optimization, we use the `Optuna` package (Akiba et al., 2019), which implements the Tree-structured Parzen Estimator (TPE), a Bayesian optimization strategy described in Section 2.3.8.

As discussed in Section 2.3.6, GCNs rely on a fixed normalized adjacency operator. They are therefore naturally suited to a transductive setting, where the entire graph, including test nodes, is available during training (though test labels are masked). In Algorithm 3, this amounts to constructing a single graph per trial on line 6, including all the data in $\mathcal{D}$. GraphSAGE, by contrast, is explicitly designed for inductive learning, in which the test graph may be entirely different from the one seen during training. In 2.3.7, we argued that generalizing across populations is a key property in genomic prediction. To evaluate GraphSAGE's predictive performance across graphs, we also train inductive models by constructing separate graphs for the training, validation, and test folds inside each trial. In Algorithm 3, this corresponds to building $2k$ graphs at line 6: one graph per outer-fold test set $\mathcal{F}_o$ (k graphs), one graph per inner-fold validation set $\mathcal{F}_{\text{inner}} \setminus v$ ($k-1$ graphs), and one graph for the outer-loop training set $\mathcal{F}_{\text{inner}}$ used on line 18 (1 graph). For completeness, we also evaluate GCNs in the inductive setting and GraphSAGE in the transductive setting, to test the robustness of both architectures across learning paradigms.

In addition to the transductive and inductive settings, the island structure of the study system

Algorithm 3 Nested cross-validation for model selection and evaluation

```
1:  Input: data $\mathcal{D}$, outer folds $\{\mathcal{F}_o\}_{o=1}^{k}$, number of trials $T$
2:  for $o = 1$ to $k$ do                                                  ▷ outer CV
3:      $\mathcal{F}_{\text{inner}} \leftarrow \{\mathcal{F}_j : j \neq o\}$
4:      Initialize best score $\bar{s}^*$ and best hyperparameters $\phi^*$
5:      for $t = 1$ to $T$ do                                    ▷ hyperparameter trial
6:          Sample hyperparameters $\phi_t$ and build graph(s)
7:          Initialize $\mathcal{S}_t$
8:          for each inner fold $v$ in $\mathcal{F}_{\text{inner}}$ do                          ▷ inner CV
9:              Train model: MODEL $\leftarrow$ TRAIN$(\mathcal{F}_{\text{inner}} \setminus v, \phi_t)$
10:             Predict on validation fold: $\hat{\mathbf{y}}_{\text{val}} \leftarrow$ MODEL.PREDICT$(v)$
11:             Append SCORE$(\hat{\mathbf{y}}_{\text{val}}, \mathbf{y}_{\text{val}})$ to $\mathcal{S}_t$
12:         end for
13:         $\bar{s}_t \leftarrow \text{mean}(\mathcal{S}_t)$
14:         if $\bar{s}_t > \bar{s}^*$ then
15:             $\bar{s}^* \leftarrow \bar{s}_t$, $\phi^* \leftarrow \phi_t$
16:         end if
17:     end for
18:     Retrain model on $\mathcal{F}_{\text{inner}}$ using $\phi^*$
19:     Evaluate on $\mathcal{F}_o$ to obtain $s_o$
20: end for
21: Return: outer test scores $\{s_o\}$ and selected hyperparameters $\{\phi_o^*\}$
```

naturally gives rise to two complementary prediction scenarios. Within-population prediction evaluates the ability to predict unobserved individuals drawn from the same set of islands. This setting closely matches the standard genomic-prediction scenario. Across-population prediction evaluates generalization to structurally distinct populations by testing the model on individuals from islands other than those on which it was trained. The across-population prediction is a much more challenging task because island structure induces genetic, environmental, and demographic differences. Individuals across different islands are, in general, less related due to limited migration (Aase et al., 2025). We evaluate these scenarios by modifying the data partition. For the standard within-population procedure, we use $k = 10$ folds matching those of Aase et al. (2025), to compare with the GBLUP. Each fold is randomly sampled from the data and contains individuals from all islands. For the across-population scenario, we use a leave-one-island-out procedure presented in Aase et al. (2025). This amounts to partitioning the dataset into folds based on the islands to which individuals belong. In our case, we have $k = 11$ different islands.

We additionally consider an inductive ensemble variant of GraphSAGE tailored to the across-population scenario. Instead of training on 9 islands and validating on the 10th in the inner loop, we train 10 separate models, each restricted to one island. Each model is evaluated on the other 9 islands, producing a set of 9 correlation scores per trial, whose mean determines the best hyperparameter configuration. After model selection, we obtain 10 island-specific models and use their predictions on the held-out 11th island to form an ensemble prediction by averaging their outputs. This approach exploits the natural subpopulation structure of the dataset. The idea is that each model captures island-specific genetic patterns, and by evaluating them on all other islands, the ensemble favours models that generalize broadly across populations.

3.4.3 SNP subset selection

Training machine–learning models on the complete set of roughly 65,000 SNPs can be computationally demanding and may introduce noise from markers that are largely irrelevant to the adjusted phenotype. To allow the models to focus on the most informative genomic regions, we apply a supervised SNP selection procedure within each training fold. For each SNP, we compute the Pearson correlation between its genotype vector and the adjusted phenotypes y_i^* using only the training data of the current split. The SNPs are then ranked by absolute correlation, and a subset of the top p markers is retained. The value of p is treated as a hyperparameter and selected by Optuna (search range: 2000–60000, see Table 3). This ensures that SNP selection is performed without information leakage from the validation or test sets. The resulting reduced SNP matrix serves as the node feature matrix for all models in that trial.

4 Results

4.1 Analysis of adjusted phenotypes

The distributions of the raw mean phenotypes (y_{mean}) and adjusted phenotypes (y_{adjusted}) are compared across the three traits (Figure 10). Body mass is approximately normally distributed, showing a symmetric, bell-shaped pattern. Wing length, in contrast, exhibits a broader, less clearly unimodal distribution, suggesting more complex underlying variation. Tarsus length is also roughly normal, but with a slight right skew and noticeably lower spread than the other two traits. For all traits, the mean of the adjusted phenotype is effectively zero, as expected from the centering imposed by the mixed model (Table 4). More importantly, the variance is substantially reduced. For body mass, the variance decreases from 4.34 to 1.43, for wing length from 5.13 to 1.27, and for tarsus length from 0.65 to 0.59. This reduction reflects the amount of phenotypic variability explained by the fixed and random effects in the adjustment model. Body mass and wing length seem to be more strongly influenced by non-genetic and temporal factors, as they show the largest proportional reductions in variance. Tarsus length, on the other hand, appears less environmentally labile, as it retains a larger fraction of its original variance.

Despite the overall shrinkage in variability, the adjusted phenotypes remain strongly correlated with the original mean phenotypes (Table 4 and the last row of Figure 10). This indicates that the adjustment primarily removes predictable, systematic variation while preserving the relative ranking of individuals. Consequently, a model that predicts y_{adjusted} well will typically also perform well on the raw phenotype. Wing length is an exception, showing a noticeably weaker correlation with its adjusted version, which suggests that the adjustment removes more trait-relevant signal for this phenotype. A plausible explanation is mild model misspecification in the adjustment step: we assume a single Gaussian error structure (and effectively a single unimodal distribution after accounting for covariates), but, as observed, the empirical wing-length measurements appear less clearly unimodal.

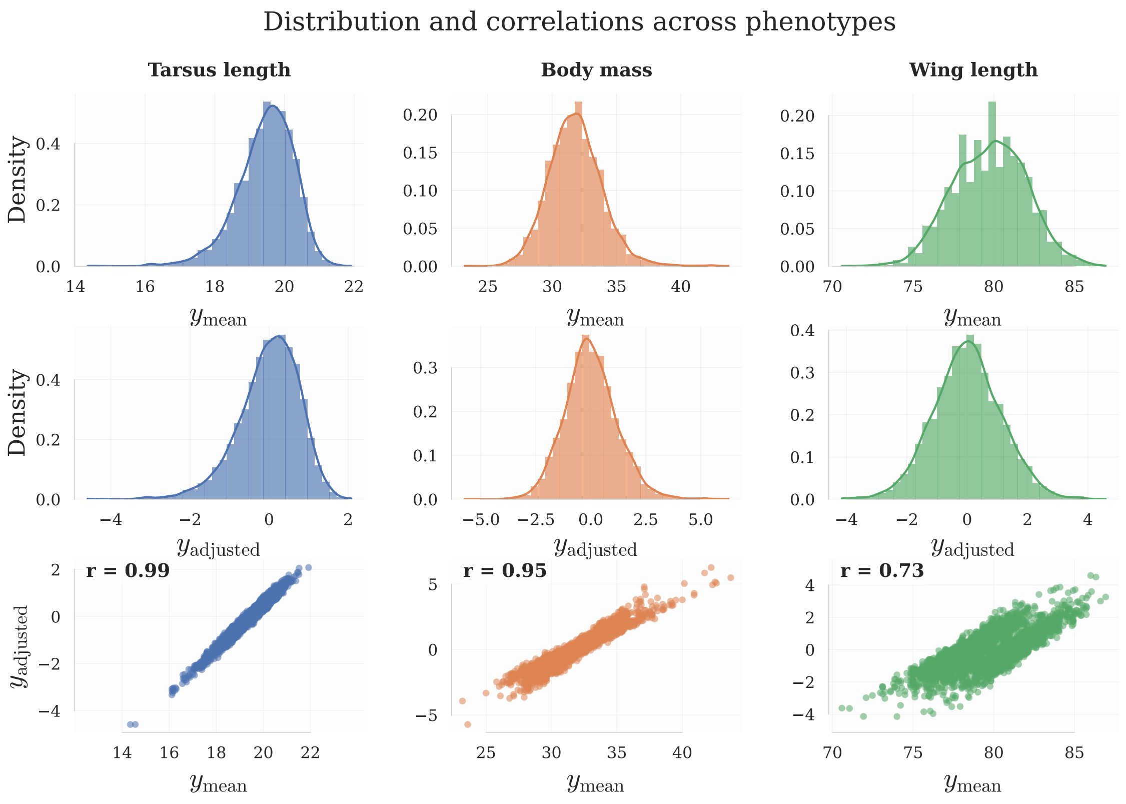


Figure 10: Distribution of mean observed phenotypes and the corresponding adjusted phenotypes for traits body mass, wing length and tarsus length.

Phenotype	Samples	Mean(y_{mean})	Var(y_{mean})	Mean(y_{adj})	Var(y_{adj})	Corr($y_{\text{adj}}, y_{\text{mean}}$)
Body mass	3445	31.80	4.34	0.006	1.43	0.953
Wing length	3384	79.74	5.13	0.005	1.27	0.732
Tarsus length	3400	19.47	0.65	-0.005	0.59	0.986

Table 4: Summary statistics for raw mean phenotypes and adjusted phenotypes.

4.2 Model performance within populations

Across all models, tarsus length is the easiest trait to predict, wing length is intermediate, and body mass is consistently the most challenging (Figure 11). The GBLUP baseline performs strongly across traits and typically achieves the highest correlations, although the machine-learning models often overlap with it on individual folds. The results are relatively stable across folds for tarsus length. In contrast, body mass and wing length show noticeably larger fold-to-fold variability, indicating that performance depends more strongly on which individuals end up in the held-out fold.

Among the machine-learning models, the MLP is a strong non-linear baseline and is generally the most competitive of the learned models in this within-population setting. The GNN models are broadly comparable to the MLP but do not show a systematic advantage. The correlation for the GCN models tends to be slightly lower than GraphSAGE on average. Notably, the GCN distribution for tarsus length is shifted downward relative to the other models. Nevertheless, the spread across folds can be substantial for the learned models, most notably for GraphSAGE on body mass, where the correlation varies by roughly 0.25 between the weakest and strongest outer folds. It is therefore important not to overinterpret minor differences in medians or quartiles. Many of the models are clearly sensitive to the differences in folds, and thus a change in folds might yield different results. In any case, for within-population prediction, the methods seem to perform similarly, and the baseline genomic animal model remains hard to beat. Furthermore, the benefit of explicitly using a graph structure appears limited.

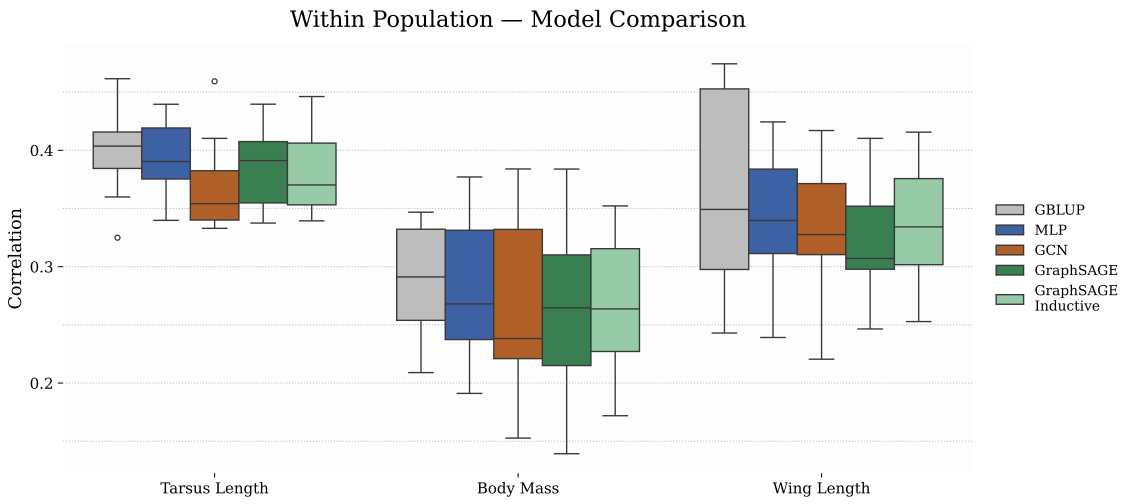


Figure 11: Predictive accuracy within the Helgeland meta-population for the three different traits, measured as Pearson correlation between predicted genetic component and mean phenotype across the $k = 10$ outer folds. The grey boxplots show the genomic animal model baseline (GBLUP) from (Aase et al., 2025), while the coloured boxplots show the machine-learning models trained on adjusted phenotypes.

4.3 Model performance across populations

The results for across-population prediction demonstrate a more challenging scenario than within-population prediction, with performance exhibiting greater fold-to-fold variability (Figure 12).

This variation highlights that certain islands are significantly harder to predict than others. A noteworthy difference from the within-population setting is that the trait-specific difficulty becomes less separated. Although tarsus length remains the most predictable trait overall, the relative gap to body mass and wing length narrows. For example, GBLUP performance for body mass and wing length is more similar across populations than within populations.

In the across-population scenario, the non-linear models are more competitive compared to GBLUP. Across all traits, the best-performing learned models are on par with, or slightly above, the GBLUP baseline on average. In fact, for both tarsus length and wing length, a GraphSAGE model has the best median, while the GraphSAGE Inductive is second best behind MLP Ensemble for body mass. An interesting observation is that the inductive GCN variant performs better than the transductive GCN for body mass, even though the inductive setting uses a different graph for training and testing. Recall that the GCN is inherently a transductive model, as it learns to average specific neighbours. However, the significant fold-to-fold variability again makes it difficult to draw sharp conclusions about a single best model architecture. As an example, the GCN Inductive ranges from a slightly negative correlation to 0.38 at its best. Nonetheless, the results are encouraging in that learned non-linear models match and sometimes outperform the GBLUP benchmark when generalizing across islands.

Per-island prediction accuracy, averaged across the three traits, further clarifies the across-population results and highlights substantial heterogeneity between islands (Figure 13). The dominant pattern is that a non-linear model best predicts most islands, whereas GBLUP performs best only on one of the 11 islands. Nesøy is a particularly clear example: averaged across traits, GBLUP attains a much lower correlation than the learned models (GBLUP around 0.14 versus roughly 0.25–0.26 for the non-linear approaches). More generally, the figure illustrates that island-specific predictability differs strongly across the meta-population, and that the relative advantages of the models depend on which island is held out.

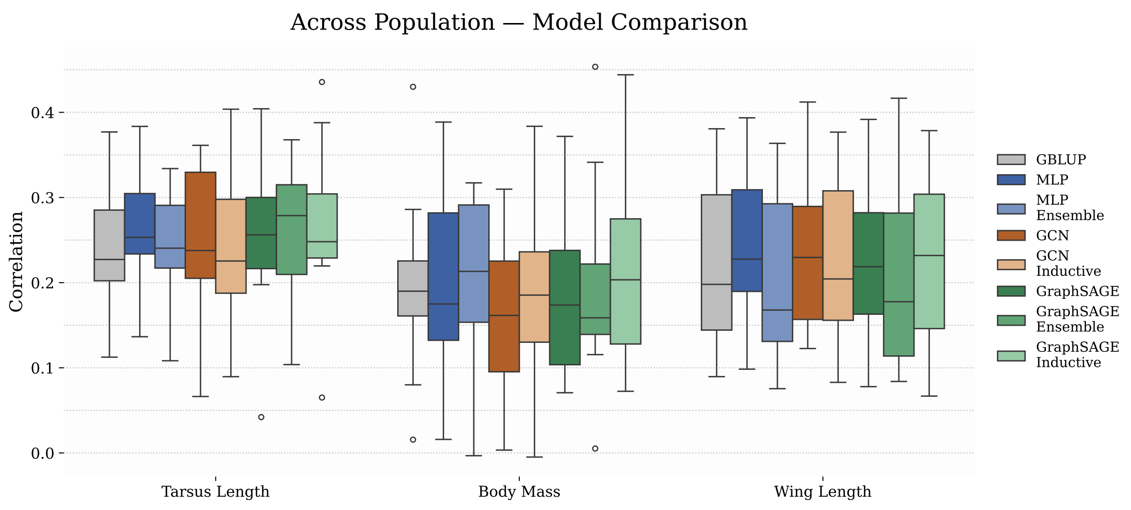


Figure 12: Predictive accuracy in the across-population (leave-one-island-out) setting for the three different traits, measured as Pearson correlation between predicted genetic component and mean phenotype across the $k = 11$ outer folds (one held-out island per fold). The grey boxplots show the GBLUP baseline from (Aase et al., 2025), while the coloured boxplots show MLP, GCN, and GraphSAGE variants.

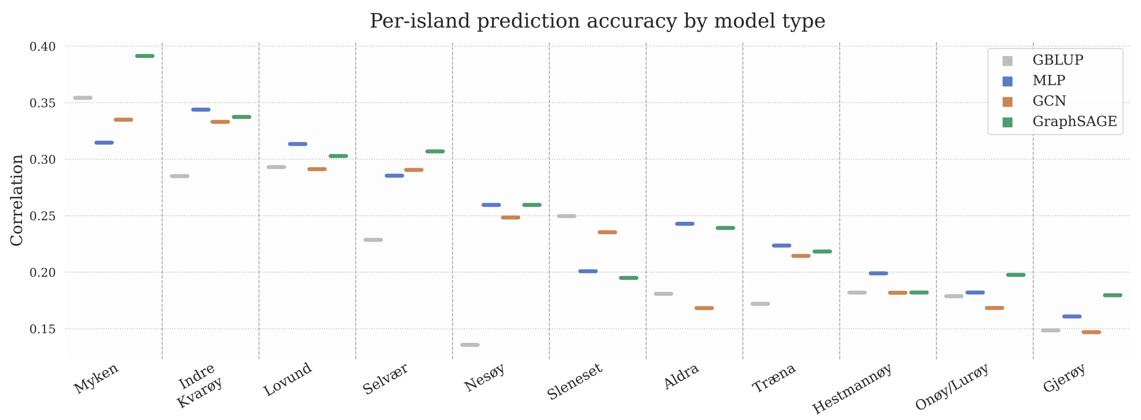


Figure 13: Predictive performance for each island in the Leave-One-Island-Out (LOIO) cross-validation scheme, averaged across all three phenotypes. For each model category (GBLUP, MLP, GCN, GraphSAGE), the specific model architecture variant (e.g., standard, ensemble, or inductive) yielding the highest validation score was selected for each fold. Islands are sorted by the mean correlation across all models.

4.4 Hyperparameter- and graph statistics

Since the defining modelling choice in the GNN experiments is how the GRM is sparsified into a graph, it is informative to summarise which graph construction strategies are preferred and what kinds of graphs they produce. Table 5 summarises how often Optuna selected threshold-based graphs versus k-NN graphs in the final models. Thresholding is preferred in most settings, particularly for the GCN, while GraphSAGE selects k-NN more frequently for across-population prediction. To characterise the graphs produced by the preferred threshold-based constructions, we examine the selected cutoff thresholds, resulting mean node degrees, and the fraction of edges connecting individuals from the same island (Figure 14).

Model	Scenario	Threshold	k-NN
GCN	Within-population	0.900	0.100
GCN	Across-population	0.879	0.121
GraphSAGE	Within-population	0.800	0.200
GraphSAGE	Across-population	0.621	0.379

Table 5: Fraction of nested-CV runs where Optuna selected threshold-based graph construction versus k-NN graph construction.

The selected thresholds imply that the learned graphs are generally sparse, but with systematic differences both between architectures and between evaluation scenarios. Across most runs, the mean degree remains below 1, indicating that many individuals retain no edges and that message passing primarily occurs among a subset of closely connected individuals. This is consistent with the intuition that only strong genomic similarities are informative for defining neighbourhoods in this setting. Recall that for a threshold of 0.5, all retained edges correspond to pairs that are at least as related as parent and offspring. GraphSAGE typically selects less strict thresholds than the GCN, resulting in denser graphs with more retained connections (Figure 14). This pattern is visible both when stratifying by phenotype and when stratifying by scenario.

A second, and particularly informative, trend is that the selected graphs tend to be denser in the across-population (LOIO) setting than in the within-population setting. Across populations, both models more frequently select looser thresholds and correspondingly higher mean degrees, suggesting that retaining additional connectivity becomes more beneficial when the model must generalise to an unseen island. Relatedly, the intra-island edge fraction remains high overall. Still, it decreases in the across-population scenario relative to the within-population scenario, indicating that a larger share of inter-island edges is retained in the across-population scenario. This is not surprising in a system with a strong family structure within islands. Still, it is notable that the hyperparameter optimisation consistently favours graphs with more cross-island links when the prediction target is a held-out island. Finally, GraphSAGE generally retains more inter-island connectivity than the GCN in both scenarios, aligning with its tendency to favour less aggressive sparsification. This suggests that looser thresholds not only produce denser graphs, as expected, but also increase the proportion of edges that cross islands.

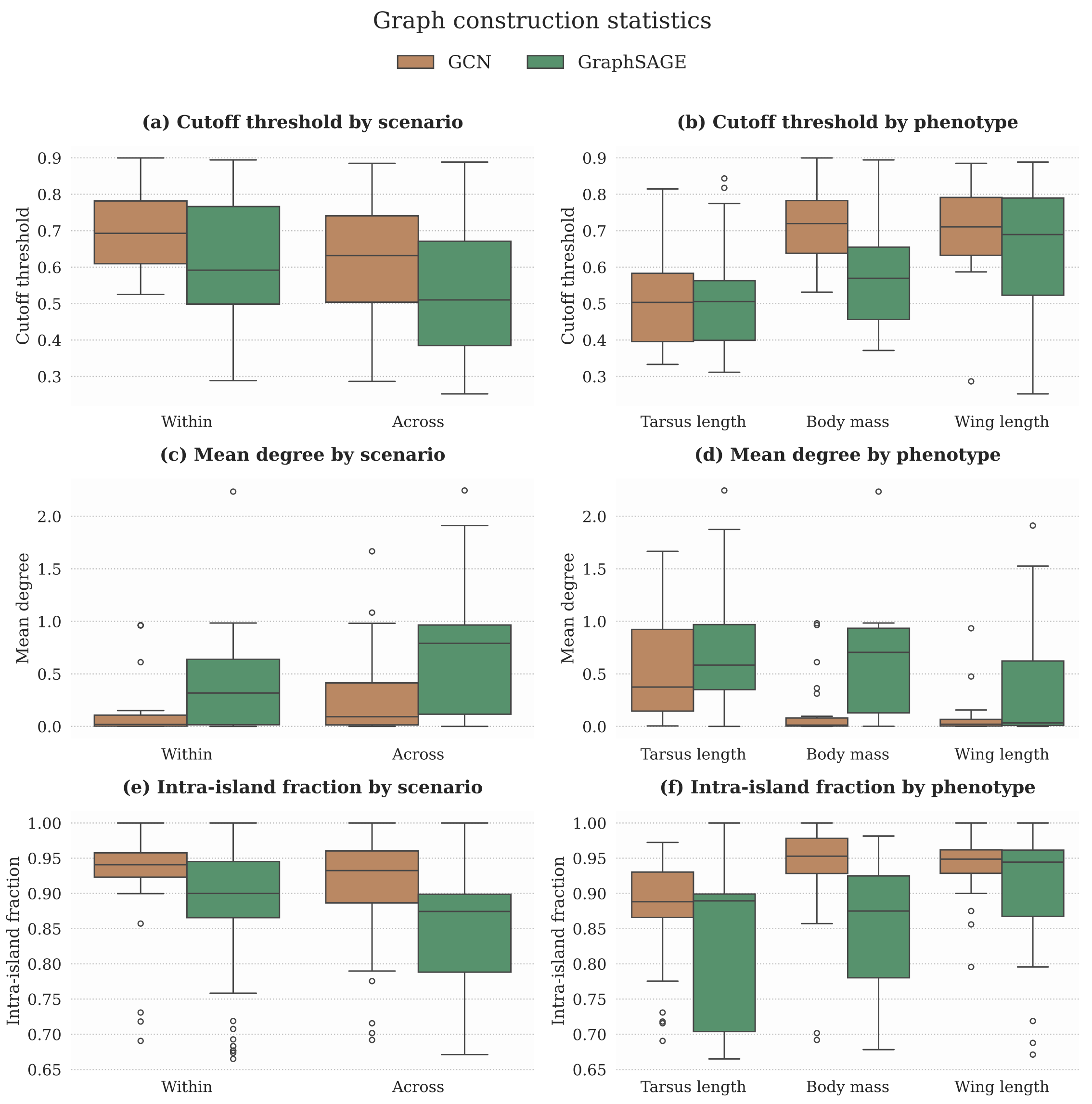


Figure 14: Distribution of graph topology metrics for GCN and GraphSAGE models under threshold-based graph construction. Panels (a) and (c) summarize the selected cutoff thresholds and resulting mean node degree by scenario (within-population vs. across-population), while panels (b) and (d) show the corresponding distributions by phenotype. Panels (e) and (f) report the intra-island edge fraction, i.e., the proportion of edges connecting individuals from the same island. Results are aggregated across all nested-CV runs and outer folds where a cutoff threshold is defined.

An additional modelling choice in the nested-CV experiments is whether to apply SNP pre-selection, where SNPs are ranked by their marginal correlation with the training response and only the top p SNPs are retained. The frequency with which SNP pre-selection is used, as well as the number of SNPs retained when selection is enabled, varies substantially across models and scenarios (Figure 15). We observe that GraphSAGE relies on SNP selection relatively often in both scenarios. In contrast, the GCN uses it mainly in the within-population setting, and the

MLP falls between these two. SNP selection is more frequently chosen in the across-population setting for GraphSAGE and the MLP, suggesting that some feature reduction can be helpful when generalising to a held-out island, whereas the GCN shows the opposite tendency. Furthermore, when selection is enabled, the selected p varies widely across runs rather than concentrating around a single value. Consequently, the medians are very roughly located at 50% of the SNPs (except for MLP within the population), indicating that a substantial amount of shrinkage is often applied when SNP selection is chosen. Finally, the preferred number of retained SNPs depends on the trait. In general, models for wing length seem to use SNP selection more often, and when they do, they also prefer fewer SNPs than the models for tarsus length and body mass. Overall, the results indicate that SNP selection is sometimes beneficial. Still, its usefulness is model-, trait-, and scenario-dependent, and the optimal degree of feature reduction is not stable across folds.

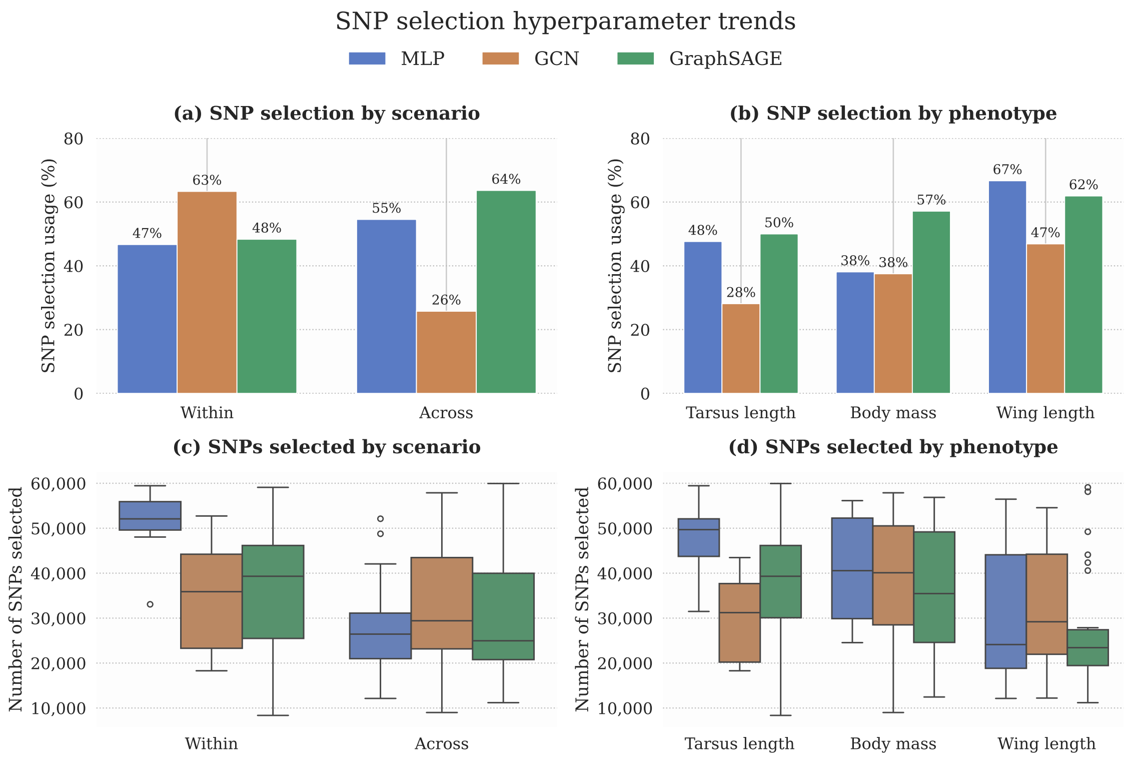


Figure 15: SNP selection hyperparameter trends for the nested-CV experiments. Panels (a) and (b) show the fraction of runs where SNP pre-selection was enabled for each model, stratified by scenario and phenotype. Panels (c) and (d) summarize the distribution of the selected number of SNPs p in the runs where SNP selection was enabled.

5 Discussion

In this report, we investigated whether graph-based neural network models (GNNs) can improve genomic prediction in a wild meta-population, both within and across subpopulations. Concretely, individuals were represented as high-dimensional SNP feature vectors, while edges were derived from the genomic relationship matrix (GRM), enabling message passing between genetically similar individuals. Two GNN architectures, graph convolutional networks (GCNs) and GraphSAGE, were tested, which differ in how message-passing between nodes (individuals) is performed. These architectures were evaluated in both the semi-supervised transductive setting and the supervised inductive setting. They were also evaluated for performance in different scenarios. In the within-population scenario, test individuals were drawn from the same population as the training individuals. In contrast, in the across-population scenario, test individuals belonged to a different population than the training individuals (that is, a different island in our context). We compared the performance of these architectures with predictions from the genomic animal model and standard fully connected neural networks in both scenarios. A nested cross-validation scheme was applied to compare model performance, with extensive hyperparameter optimization.

The within-population results show that the classical baseline, the genomic animal model, remains difficult to outperform. Across all three traits, GBLUP achieved high correlations, while the learned models overlapped substantially with the baseline and did not exhibit a systematic advantage. Among the learned models, the MLP emerged as a robust non-linear baseline, generally matching or exceeding the GNN variants in this setting. The GNNs were broadly comparable to the MLP, but gains were neither consistent nor significant, and the GCN tended to underperform compared to all other models. A key practical observation is the non-negligible fold-to-fold variability for the learned models, especially for body mass, which cautions against overinterpreting minor median differences. A different picture emerged in the across-population scenario. Across-population genomic prediction is known to be more challenging and degrades performance compared to within-population genomic prediction (Aase et al., 2025). The results obtained here showed the same pattern, as the correlations between the predictions and the phenotypes were, in general, lower. Nonetheless, the relative standing of the models shifted: non-linear approaches became more competitive compared to GBLUP, and the best learned models were on par with, or slightly above, the baseline on average. In particular, GraphSAGE achieved the best median performance for tarsus length and wing length, while the best performance for body mass came from the MLP ensemble model.

The motivation for applying GNNs to genomic prediction problems primarily stemmed from two limitations of earlier approaches. First, the genomic animal model and marker-based regression completely disregard any non-linear effects. Epistatic effects (gene-gene interactions) are known to explain some variance in quantitative traits (Mackay, 2014). Boosting algorithms and neural networks that are capable of modelling non-linearities have been experimented with on the Helgeland sparrow data, but with limited success (Singsaas, 2024; Gravdal, 2025; Wold, 2025). Second, the boosting algorithms and neural networks previously applied ignored the correlation structure that makes animal models so successful. GNNs provide a way to unify the strengths of these approaches: they learn a direct non-linear mapping from markers to phenotype and, at the same time, leverage the correlation structure in the GRM by modelling it as a graph. The idea is that the imposed relatedness structure can work as an inductive bias, forcing nodes to consider the inflow of information from neighbours.

Despite the promising ideas GNNs introduce, results for within-population prediction were modest. Looking at GraphSAGE, one plausible explanation is that the main potential advantage – its ability to generalise across graphs – is not strongly expressed in the within-population setting. Under random splits by individual, the training and test folds are drawn from the same underlying population and share the same overall structure in terms of relatedness and allele frequencies. In that case, the inductive bias introduced by explicitly modelling neighbourhood structure may not add much beyond what a flexible non-linear baseline already captures from the SNP features alone. This is consistent with the empirical pattern that GNNs rarely improve systematically over MLPs within a population, and that differences between learned models are often comparable in magnitude to fold-to-fold variability. A related observation is that the transductive GraphSAGE variant does not consistently outperform its inductive counterpart, despite having access to test-node features during training. Part of what the GraphSAGE model learns is how to aggregate neighbourhood information, and it can, in theory, ignore all incoming signal. Thus, allowing test nodes to contribute helpful information during training effectively provides more data to learn from. However, if the graph structure itself offers limited additional signal for this particular within-population task, then adding additional edges to test-nodes may increase the likelihood that noise enters the representations.

For the GCN, the lack of a clear within-population improvement is, in a way, easier to understand. In contrast to GraphSAGE, a GCN enforces neighbourhood aggregation at every layer, which makes it less able to "ignore" unhelpful edges. If many edges reflect only weak or redundant information relative to what is already contained in the node features, then repeated smoothing over the graph can dilute informative individual-level variation, a phenomenon known as oversmoothing (Hamilton, 2022). The graph statistics further suggest that the GCN often selects stringent GRM thresholds for the adjacency matrix, yielding extremely sparse graphs in which most nodes have few or no neighbours. In the limiting case where the adjacency approaches the identity (after self-loops), the GCN update resembles an MLP applied independently to each node (see equation (7)). As a result, one might expect similar performance for the GCN models and the MLPs. In practice, however, the GCN is still constrained by the fixed propagation operator and must simultaneously tune both neural-network hyperparameters and graph-construction hyperparameters. Since all models were allocated the same number of optimisation trials, the graph-based models effectively faced a larger hyperparameter search space under the same budget, which can disadvantage them when the graph does not contribute a substantial, consistent benefit.

The results for GraphSAGE appeared more favourable in the across-population scenario. One plausible explanation is that the across-population scenario aligns more closely with the original motivation for the model. The GraphSAGE model was initially developed to enable inductive learning, allowing it to generalize to different structures. In across-population prediction, the model is tasked with generalizing under a genuine population shift. The inductive bias induced by the GRM graph may serve as a form of regularization, as it provides an explicit mechanism for aggregating data across genetically similar individuals. Recent work has argued that deep neural networks for genomic prediction can fall back on a "shortcut" by basing predictions primarily on overall genetic relatedness rather than learning effects of particular markers, which may limit the gains from flexible nonlinear models (Ubbens et al., 2021). In that light, explicitly supplying relatedness through message passing may offer a more controlled way to use this signal and could (in principle) free the feature encoder to focus more on marker-specific patterns beyond what is already captured by global relatedness. In any case, we do indeed observe denser adjacency matrices in the across-population scenario compared to the within-population scenario (Figure 14).

Feature selection appears to fit naturally into the same explanation: Across-population prediction is known to be harder partly because marker effects and linkage disequilibrium patterns may not transfer perfectly between populations (Aase et al., 2025). Thus, in the across-population scenario, removing weakly informative markers can act as a form of variance reduction, leaving the model to focus on a subset of SNPs with a more stable signal. The observation that SNP pre-selection is used more often in across-population prediction for some models indicates that generalisation across islands benefits from stronger regularisation.

The non-negligible share of inter-island edges is interesting because such links are the only way a GRM-based graph can connect sub-populations and potentially act as a "bridge" between islands. However, the mechanism is not entirely direct. In the inductive setting for across-population prediction, the graph fed to the model when predicting on the test data contains only individuals from the held-out island, so information cannot literally propagate from the training islands to the test embeddings at inference time. Furthermore, the model does not select edges because they are "inter-island"; it only selects a sparsification level, and the retained edges are those with high genomic similarity. With some migration between islands, a subset of individuals will have close relatives elsewhere, so thresholding the GRM naturally retains a few cross-island family links. The higher inter-island edge fraction in across-population prediction may therefore reflect both migration-driven relatedness and generally looser thresholds, rather than an explicit preference for cross-island connections.

Finally, the GCN remains less competitive than GraphSAGE also for across-population prediction. This aligns with the idea that fixed, repeated neighbourhood smoothing can be fragile under population shift and may be more prone to oversmoothing. Again, we see from the graph statistics (Figure 14) that the GCN favours much sparser graphs than the GraphSAGE models, likely because it aims to avoid oversmoothing and acts more like an MLP (as seen within-population). It is worth noting that GraphSAGE models in general tend to outperform GCN models (Hu et al., 2021; Hamilton, 2022)

Several limitations temper the conclusions and motivate future work. First, the graph construction is conceptually "circular" in the sense that both node features and edges are derived from the same SNP data. Unlike molecule graphs or social networks, where edges often provide information not present in node features, the GRM graph primarily injects a structural prior rather than new data. Interestingly, however, this circularity does not preclude benefits; inductive biases can improve generalisation even when they are not adding information, as shown in Kihlman et al. (2024). Of course, gains may be limited unless the graph captures structure that is difficult for an MLP to exploit directly. Second, the SNP pre-selection procedure is deliberately simple and ignores genomic position. In practice, nearby SNPs are often in linkage disequilibrium, meaning they carry highly redundant information about the same underlying haplotypes. Ranking SNPs by marginal correlation can therefore lead to selecting clusters of strongly correlated markers, where many retained SNPs contribute little additional signal beyond the first. This could reduce the effectiveness of feature reduction and make the chosen p harder to interpret biologically. Finally, the significant fold-to-fold variability observed in both within- and across-population scenarios underscores that minor differences between medians are not definitive evidence of superiority for a given model architecture.

These limitations point to several natural extensions. On the modelling side, using weighted edges rather than dichotomized adjacency may preserve more information from the GRM and allow the network to learn how much to trust different relationship strengths. Testing GraphSAGE-like operators that incorporate edge weights is therefore a promising next step. Methodologically, scalable minibatching and neighbourhood sampling would enable training on larger graphs and could make inductive training more faithful to the original GraphSAGE motivation (Hamilton et al., 2018). Additionally, more structured approaches for SNP selection, for example LD-pruning, block-wise selection, or selection procedures that explicitly account for correlation between markers, could yield a smaller and less redundant feature set and potentially change the relative performance of the models. Third, the ensemble strategies explored here suggest another direction: rather than uniformly averaging models, one could weight island-specific predictors by measures of similarity between the training island and the held-out target. Relatedly, training set optimization could be explored. When only limited phenotypic data are available in the focal population, the training set may be augmented with individuals from other populations, but selected deliberately to maximize expected accuracy in the target. Such selection-based augmentation has been proposed as a promising approach for genomic prediction (Rio et al., 2021). The methodology is particularly relevant for wild populations and in conservation contexts where focal populations are small and isolated.

Overall, the results support a nuanced conclusion. For within-population prediction of morphological traits in our study system, the classical genomic animal model remains a strong and competitive baseline, and the benefit of explicitly enforcing a graph structure via message passing appears limited. However, in the more challenging, yet still practically relevant, across-population setting, learned non-linear models, including GNNs, match or sometimes exceed the baseline. Graph-based inductive biases are most useful when the task demands transfer across structured subpopulations. Framing genomic prediction as a graph learning problem therefore remains a promising bridge between quantitative genetics and modern machine learning.

Bibliography

Aase, Kenneth et al. (2025). 'How accurate is genomic prediction across wild populations?' *Evolution.* Ed. by Henrique Teotonio and Jason Wolf.

Ahmad, Waqar et al. (2024). 'SolPredictor: Predicting Solubility with Residual Gated Graph Neural Network'. *International Journal of Molecular Sciences* 25.2, p. 715.

Akiba, Takuya et al. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework.*

Ashraf, Bilal et al. (2022). 'Genomic prediction in the wild: A case study in Soay sheep'. *Molecular Ecology* 31.24, pp. 6541–6555.

Banerjee, Sudipto and Anindya Roy (2015). *Linear Algebra and Matrix Analysis for Statistics.* Chapman & Hall / CRC Texts in Statistical Science. CRC Press.

Bengio, Y., P. Simard and P. Frasconi (1994). 'Learning long-term dependencies with gradient descent is difficult'. *IEEE transactions on neural networks* 5.2, pp. 157–166.

Bergstra, James et al. (2011). 'Algorithms for Hyper-Parameter Optimization'. *Advances in Neural Information Processing Systems.* Vol. 24. Curran Associates, Inc.

Boichard, Didier et al. (2016). 'Genomic selection in domestic animals: Principles, applications and perspectives'. *Comptes Rendus Biologies* 339.7, pp. 274–277.

Brodie, Aharon, Johnathan Roy Azaria and Yanay Ofran (2016). 'How far from the SNP may the causative genes be?' *Nucleic Acids Research* 44.13, pp. 6046–6054.

Brookes, Anthony J. (1999). 'The essence of SNPs'. *Gene* 234.2, pp. 177–186.

Bühlmann, Peter and Sara van de Geer (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications.* Springer series in statistics. Springer.

Chapelle, Olivier, Alexander Zien and Bernhard Schölkopf, eds. (2006). *Semi-Supervised Learning.* Adaptive computation and machine learning series. MIT Press.

Choi, Shing Wan, Timothy Shin Heng Mak and Paul F. O'Reilly (2020). 'A guide to performing Polygenic Risk Score analyses'. *Nature protocols* 15.9, pp. 2759–2772.

Conner, Jeffrey K. and Daniel L. Hartl (2004). *A Primer of Ecological Genetics.* Sinauer Associates.

Cybenko, G. (1989). 'Approximation by superpositions of a sigmoidal function'. *Mathematics of Control, Signals and Systems* 2.4, pp. 303–314.

Davies, Nicholas B. (2012). *An Introduction to Behavioural Ecology.* 1st ed. New York Academy of Sciences Series. John Wiley & Sons, Incorporated.

Desta, Zeratsion Abera and Rodomiro Ortiz (2014). 'Genomic selection: genome-wide prediction in plant improvement'. *Trends in Plant Science* 19.9, pp. 592–601.

Eraslan, Gökcen et al. (2019). 'Deep learning: new computational modelling techniques for genomics'. *Nature Reviews Genetics* 20.7, pp. 389–403.

Erbe, M. et al. (2012). 'Improving accuracy of genomic predictions within and between dairy cattle breeds with imputed high-density single nucleotide polymorphism panels'. *Journal of Dairy Science* 95.7, pp. 4114–4129.

Fahrmeir, L. (2013). *Regression: Models, Methods and Applications.* Springer.

Falconer, Douglas S. and Trudy Mackay (1996). *Introduction to Quantitative Genetics.* 4. ed., [16. print.] Pearson, Prentice Hall.

Fareed, Mohd and Mohammad Afzal (2013). 'Single nucleotide polymorphism in genome-wide association of human population: A tool for broad spectrum service'. *Egyptian Journal of Medical Human Genetics* 14.2, pp. 123–134.

Fisher, Ronald Aylmer (1930). *The Genetical Theory of Natural Selection.* Clarendon Press.

Gianola, Daniel (2013). 'Priors in Whole-Genome Regression: The Bayesian Alphabet Returns'. *Genetics* 194.3, pp. 573–596.

Gilmer, Justin et al. (2017). *Neural Message Passing for Quantum Chemistry.*

Goodfellow, Ian, Yoshua Bengio and Aaron Courville (2016). *Deep Learning.* MIT Press.

Gravdal, Gard Westrum (2025). 'Machine learning approaches to genomic prediction and marker association'. Master thesis. NTNU.

Gross, Jonathan L., Jay Yellen and Mark Anderson (2019). *Graph Theory and Its Applications.* Third edition. Textbooks in mathematics. CRC Press.

Hamilton, William L. (2022). *Graph Representation Learning.* Reprint of original edition of Morgan Claypool 2020. Synthesis lectures on artificial intelligence and machine learning 46. Springer International Publishing.

Hamilton, William L., Rex Ying and Jure Leskovec (2018). *Inductive Representation Learning on Large Graphs.*

Hastie, Trevor, Robert Tibshirani and Jerome H. Friedman (2017). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Second edition. Springer Series in Statistics. Springer.

Henderson, C. R. (1975). 'Best Linear Unbiased Estimation and Prediction under a Selection Model'. *Biometrics* 31.2, p. 423.

Hill, W.G. and B.S. Weir (2011). 'Variation in actual relationship as a consequence of Mendelian sampling and linkage'. *Genetics Research* 93.1, pp. 47–64.

Hoerl, Arthur E. and Robert W. Kennard (1970). 'Ridge Regression: Biased Estimation for Nonorthogonal Problems'. *Technometrics* 12.1, pp. 55–67.

Hornik, Kurt, Maxwell Stinchcombe and Halbert White (1989). 'Multilayer feedforward networks are universal approximators'. *Neural Networks* 2.5, pp. 359–366.

Hu, Weihua et al. (2021). *Open graph benchmark: datasets for machine learning on graphs.*

Husby, Arild, Marcel E. Visser and Loeske E. B. Kruuk (2011). 'Speeding Up Microevolution: The Effects of Increasing Temperature on Selection and Genetic Variance in a Wild Bird Population'. *PLOS Biology* 9.2, e1000585.

James, Gareth et al. (2021). *An Introduction to Statistical Learning: With Applications in R.* Second edition. Springer texts in statistics. Springer.

Jensen, Henrik, B.-E. Sæther et al. (2003). 'Sexual variation in heritability and genetic correlations of morphological traits in house sparrow (*Passer domesticus*)'. *Journal of Evolutionary Biology* 16.6, pp. 1296–1307.

Jensen, Henrik, Ingelin Steinsland et al. (2008). 'Evolutionary Dynamics of a Sexual Ornament in the House Sparrow (passer Domesticus): The Role of Indirect Selection Within and Between Sexes'. *Evolution* 62.6, pp. 1275–1293.

Kihlman, Ragini et al. (2024). 'Sub-sampling graph neural networks for genomic prediction of quantitative phenotypes'. *G3: Genes, Genomes, Genetics* 14.11. Ed. by D-J De Koning.

King, Isaiah J. and H. Howie Huang (2023). 'Euler: Detecting Network Lateral Movement via Scalable Temporal Link Prediction'. *ACM Trans. Priv. Secur.* 26.3, 35:1–35:36.

Kingma, Diederik P. and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization.*

Kipf, Thomas N. and Max Welling (2017). *Semi-Supervised Classification with Graph Convolutional Networks.*

Kruuk, Loeske E. B. (2004). 'Estimating genetic parameters in natural populations using the 'animal model''. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 359.1446, pp. 873–890.

Li, Bo et al. (2018). 'Genomic Prediction of Breeding Values Using a Subset of SNPs Identified by Three Machine Learning Methods'. *Frontiers in Genetics* 9, p. 237.

Li, Xiao et al. (2023). 'A survey of graph neural network based recommendation in social networks'. *Neurocomputing* 549, p. 126441.

Lundberg, Scott and Su-In Lee (2017). *A Unified Approach to Interpreting Model Predictions.*

Lundregan, Sarah L. et al. (2018). 'Inferences of genetic architecture of bill morphology in house sparrow using a high-density SNP array point to a polygenic basis'. *Molecular Ecology* 27.17, pp. 3498–3514.

Lynch, Michael and Bruce Walsh (1998). *Genetics and Analysis of Quantitative Traits.* Sinauer Assoc.

Mackay, Trudy (2014). 'Epistasis and Quantitative Traits: Using Model Organisms to Study Gene-Gene Interactions'. *Nature reviews. Genetics* 15.1, pp. 22–33.

McCulloch, Warren S. and Walter Pitts (1943). 'A logical calculus of the ideas immanent in nervous activity'. *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

Meuwissen, Theo, Ben Hayes and Mike Goddard (2001). 'Prediction of Total Genetic Value Using Genome-Wide Dense Marker Maps'. *Genetics* 157.4, pp. 1819–1829.

— (2016). 'Genomic selection: A paradigm shift in animal breeding'. *Animal Frontiers* 6.1, pp. 6–14.

Muff, Stefanie et al. (2019). 'Animal models with group-specific additive genetic variances: extending genetic group models'. *Genetics Selection Evolution* 51.1, p. 7.

Niskanen, Alina K. et al. (2020). 'Consistent scaling of inbreeding depression in space and time in a house sparrow metapopulation'. *Proceedings of the National Academy of Sciences* 117.25, pp. 14584–14592.

Ødegård, Jørgen et al. (2018). 'Large-scale genomic prediction using singular value decomposition of the genotype matrix'. *Genetics Selection Evolution* 50.1, p. 6.

Prince, Simon J. D. (2024). *Understanding Deep Learning.* The MIT Press.

Qin, Shiyi et al. (2023). 'Capturing molecular interactions in graph neural networks: a case study in multi-component phase equilibrium'. *Digital Discovery* 2.1, pp. 138–151.

Ranke, Peter S. et al. (2021). 'Spatial structure and dispersal dynamics in a house sparrow metapopulation'. *Journal of Animal Ecology* 90.12, pp. 2767–2781.

Reiser, Patrick et al. (2022). 'Graph neural networks for materials science and chemistry'. *Communications Materials* 3.1, p. 93.

Rio, Simon et al. (2021). 'Genomic prediction and training set optimization in a structured Mediterranean oat population'. *Theoretical and Applied Genetics* 134.11, pp. 3595–3609.

Rosenblatt, F. (1958). 'The perceptron: A probabilistic model for information storage and organization in the brain'. *Psychological Review* 65.6, pp. 386–408.

Rue, Håvard, Sara Martino and Nicolas Chopin (2009). 'Approximate Bayesian Inference for Latent Gaussian models by using Integrated Nested Laplace Approximations'. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 71.2, pp. 319–392.

Rumelhart, David E., Geoffrey E. Hinton and Ronald J. Williams (1986). 'Learning representations by back-propagating errors'. *Nature* 323.6088, pp. 533–536.

Russell, Stuart J. and Peter Norvig (2022). *Artificial Intelligence: A Modern Approach.* In collab. with Ming-wei Chang et al. Fourth edition, global edition. Prentice Hall series in artificial intelligence. Pearson.

Scarselli, F. et al. (2009). 'The Graph Neural Network Model'. *IEEE Transactions on Neural Networks* 20.1, pp. 61–80.

Schaid, Daniel J. et al. (2004). 'Comparison of Microsatellites Versus Single-Nucleotide Polymorphisms in a Genome Linkage Screen for Prostate Cancer–Susceptibility Loci'. *The American Journal of Human Genetics* 75.6, pp. 948–965.

Singsaas, Øyvind Hansen (2024). 'Neural networks for genomic prediction'. Master thesis. NTNU.

Slatkin, Montgomery (2008). 'Linkage disequilibrium — understanding the evolutionary past and mapping the medical future'. *Nature Reviews Genetics* 9.6, pp. 477–485.

Sra, Suvrit, Sebastien Nowozin and Stephen J. Wright (2012). *Optimization for Machine Learning.* Neural information processing series. MIT press.

Ubbens, Jordan et al. (2021). 'Deep neural networks for genomic prediction do not estimate marker effects'. *The Plant Genome* 14.3, e20147.

VanRaden, P.M. (2008). 'Efficient Methods to Compute Genomic Predictions'. *Journal of Dairy Science* 91.11, pp. 4414–4423.

Vapnik, Vladimir Naumovich (1995). *The Nature of Statistical Learning Theory.* 1st ed. Springer New York.

Vlaming, Ronald de and Patrick J. F. Groenen (2015). 'The Current and Future Use of Ridge Regression for Prediction in Quantitative Genetics'. *BioMed Research International* 2015, p. 143712.

Vologodskij, Aleksandr Vadimovič (2023). *The Basics of Molecular Biology.* 1st ed. 2023. Springer International Publishing.

Wang, Shawn, Fanelli Alessio and George Hotz (2023). *Commoditizing the Petaflop — with George Hotz of the tiny corp.*

Wilson, Alastair J. et al. (2010). 'An ecologist's guide to the animal model'. *Journal of Animal Ecology* 79.1, pp. 13–26.

Wold, Simen Kristian Lunde (2025). 'Genomic Prediction in Wild Populations: Gradient Boosting with Reduced SNP Sets and Feature Compression'. Master thesis. NTNU.

Xu, Shizhong (2022). *Quantitative Genetics.* Springer International Publishing.

Yang, Jian et al. (2010). 'Common SNPs explain a large proportion of heritability for human height'. *Nature genetics* 42.7, pp. 565–569.

Zaheer, Manzil et al. (2018). *Deep Sets.*

NTNU | Fakultet for informasjonsteknologi og elektroteknikk

Declaration of AI aids and -tools

Have any AI-based aids or tools been used in the creation of this report?

☐ No

☒ Yes

If *yes*: please specify the aid/tool and area of use below.

Text

☒ **Spell checking**. Are parts of the text checked by: *Grammarly, Ginger, Grammarbot, LanguageTool, ProWritingAid, Sapling, Trinka.ai* or similar tools?

☒ **Text generation**. Are parts of the text generated by: *ChatGPT, GrammarlyGO, Copy.AI, WordAi, WriteSonic, Jasper, Simplified, Rytr* or similar tools?

☒ **Writing assistance**. Are one or more of the reports ideas or approach suggested by: *ChatGPT, Google Bard, Bing chat, YouChat* or similar tools?

If *yes*, use of text aids/tools apply to this report - please specify usage here:

ChatGPT was used as used as a tool to reflect on ideas for the project, and discuss problems. It was also used to enhance formulations. Grammarly was used for spell-checking.

Codes and algorithms

☒ **Programming assistance**. Are parts of the codes/algorithms that i) appear directly in the report or ii) have been used to produce results such as figures, tables or numerical values been generated by: *GitHub Copilot, CodeGPT, Google Codey/Studio Bot, Replit Ghostwriter, Amazon CodeWhisperer, GPT Engineer, ChatGPT, Google Bard* or similar tools?

If *yes*, use of programming assistance aid/tools apply to this report - please specify usage here:

ChatGPT and Copilot was used to develop code. They were mainly used to extend already written code, adding more features.
Both were also used extensively to generate code for figures.

Images and figures

☐ **Image generation**. Are one or more of the reports images/figures generated by: *Midjourney, Jasper, WriteSonic, Stability AI, Dall-E* or similar tools?

If *yes,* use of image generator aids/tools apply to this report – please specify usage here:

☐ **Other AI aids or tools**. Have you used other types of AI aids or -tools in the creation of this report? If yes, please specify usage here:

☒ I am familiar with NTNU's regulations: *Submitting a report generated with the assistance of AI tools and claiming the work to be partially or fully my own, is not permitted. I therefore declare that any use of AI aids or tools are explicitly stated i) directly in the report or ii) in this declaration form.*

Simen Nesland / 18.12.2025 / Trondheim

Signature/Date/Place