Plant Phenotype Predictions by Using Deep Neural Networks

Technical Report

Nghi Huynh
ge86puv@mytum.de
TUM Campus Straubing for Biotechnology and Sustainability

January 17, 2024

Abstract

Regarding food security in the face of climate change, this study investigates a promising approach in genetic selection using deep learning as a prediction tool for advancing plant breeding programs. In contrast to resource-intensive and time-consuming traditional breeding methods, genomic selection swiftly and accurately identifies high-quality individuals with desired traits directly from the genotype dataset. The core of the dataset uses a Single Nucleotide Polymorphism (SNP), which is one of the most pivotal genomic resources offering rich information to characterize biological traits. Through a comprehensive comparative analysis of three deep learning models, including Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN), we focus on unveiling their distinct strengths in interpreting complex relationships between genetic markers and phenotypic traits. While acknowledging that the performance results are not good with respect to explained variance, our findings highlight the superior performance of the MLP in phenotype prediction. As an experimental result with the largest independent test dataset, we obtain $70,75\%$ for MLP and 43.52% for CNN, while RNN shows that it is not possible to predict.

Keywords Phenotype Predictions · Machine Learning · Deep neural networks · Bioinformatics

1 Introduction

Plant breeding emerges as a driving force in advancing sustainable agricultural production and effective resource management [Buerstmayr et al., 2022]. The optimization of crop traits, i.e., pest resistance, increased yield, and heightened nutritional value, is crucial for ensuring a stable and abundant food supply [Acquaah, 2009]. However, achieving optimum plant quality has been an ongoing challenge for crop breeders due to the complex interaction between an organism's genetic makeup (genotype) and environmental influences, which are expressed in the phenotype [Grinberg et al., 2022]. Therefore, identifying genotypes with desirable traits is essential to enable breeders to estimate the breeding values of crops and pre-select potential cultivars based on specific trait interests. From genotype to phenotype, the challenge is how we can understand the relation between genotype and environment.

Traditional plant breeding methods involved selecting progeny over multiple generations, a time- and resource-intensive process [Goddard, 2009]. In contrast, genomic selection (GS) stands out as a modern approach, predicting an individual's genetic merit for phenotypic characteristics directly from genotype markers. GS establishes intricate relationships between trait values and genomic information by utilizing whole-genome molecular markers [Goddard, 2009]. This gives a unique advantage, such as the predictive ability of phenotypic trait values before planting, not only enhancing selection quality but also substantially reducing the time required for the breeding cycle [Desta and Ortiz, 2014] [Yu et al., 2016]. However, GS performance is influenced by the amount of available data, as noted in previous studies [Fischer et al., 2014]. Fortunately, recent advancements in next-generation sequencing technologies, such as genotyping by sequencing (GBS) and restriction-site associated DNA sequencing (RAD-seq), have mitigated this challenge [Esposito et al., 2019] [Le Nguyen et al., 2019]. These technologies have provided abundant molecular markers covering the entire genome of crops and have become more accessible due to reduced genome sequencing costs. Single nucleotide polymorphisms (SNPs), for instance, exemplify widely used, cost-effective, and high-density genetic markers [Vignal et al., 2002]. Consequently, an increasing number of modern plant breeding programs are using GS in SNP dataset to guide the development of improved crops. Some noteworthy examples include the selection of superior dairy bulls for breeding [Grinberg et al., 2016] and the development of a new maize variety with enhanced drought resistance [Cooper et al., 2014]. This integration of GS into plant breeding strategies reflects a transformative shift toward more efficient, informed, and impactful crop improvement practices.

Two primary models drive GS, including statistical models and machine learning models. Statistical models, which are often regression-based techniques, assume normally distributed marker effects with equal variance [Meuwissen et al., 2001] [Anilkumar et al., 2022]. These models excel for additive traits based on the linear nature of genotypic markers. However, the statistical models face challenges with high-dimensional genotype data [Crossa et al., 2017] [Meuwissen et al., 2001]. Notably, these challenges manifest in the form of interactions between single nucleotide polymorphisms (SNPs) contributing to complex diseases or traits, as exemplified by the epistasis effects observed in various studies [Wang et al., 2015]. Machine learning (ML), particularly deep learning (DL) models, demonstrate a capacity to capture non-linear relationships between hidden patterns in genotypes, facilitating complex trait prediction [Abdollahi-Arpanahi et al., 2020] [Ma et al., 2014]. Indeed, DL has also been employed in some genotype association research, i.e., to identify SNP interactions [Uppu et al., 2016], as well as classify genomic variants [Liang et al., 2016].

DL employs large neural networks (NNs) with multiple layers to comprehend intricate patterns in data [LeCun et al., 2015]. Recently, DL methods have emerged as powerful tools for unraveling complex relationships in biological data, notably in computational biology fields [Min et al., 2017]. In genomics, genetics, and breeding, DL methods have been introduced to analyze high-dimensional data [Meuwissen et al., 2001], providing a promising alternative to traditional approaches. In the context of phenotype prediction problems, there are several relevant studies applying DL approaches to plant, animal, and human domains, with the comparison of different prediction methods for genotype prediction [John et al., 2022] [Azodi et al., 2019]. These DL-based approaches leverage genotype data, extracting influential features through data preprocessing that reflect phenotype labels. Subsequently, DL algorithms are employed to train prediction models for specific targets, such as phenotype prediction problems. This progress has pushed global research from standard ML methods to novel DL architectures.

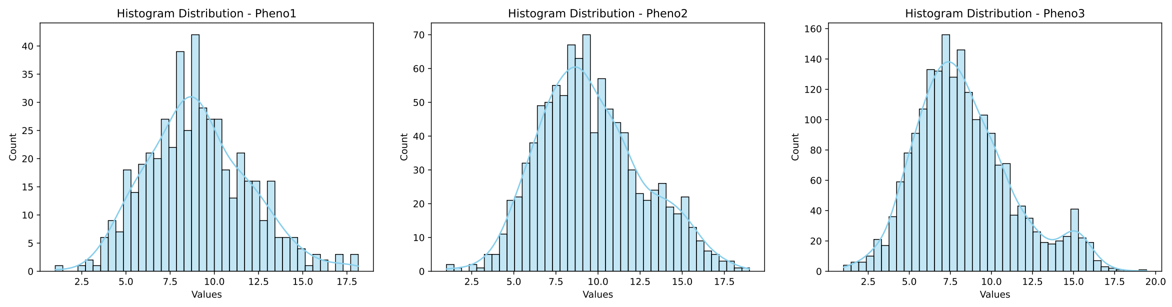
The above is a motivation for this research internship. We focus on analyzing three prominent models of deep NNs, including Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN). Specifically, we aim to implement these three models from scratch to demystify them. This work is undertaken with the goal of providing an objective evaluation of the effectiveness when applying MLP, CNN, RNN to an actual dataset. The outcome is a comparative evaluation of deep NN models in general, shedding light on their strengths as well as potential challenges. Our analysis is conducted from different perspectives on data processing, encoding, and model architecture applied for phenotype predictions.

2 Material and Methods

In the following, we first describe the data used in this research. Second, we outline the architecture overview of different deep NNs trained for phenotype predictions. Finally, we give details on how the evaluation and experiments are set up, including data preprocessing, splitting, training pipeline, hyperparameter optimization, and performance evaluation metrics.

2.1 Dataset

We used a fully imputed homozygous SNP dataset comprising genotype and phenotype matrices, where the genotype matrix stores available SNPs corresponding to each sample, and the phenotype matrix stores sample trait values, which we call labels. Homozygous SNPs indicate that an individual inherits identical alleles from both parents, and only one of these alleles is shown in data. Due to separate collection processes, several samples lack either SNP or trait values, leading to an incomplete match. To address this, we matched the samples based on the sample IDs between phenotype and genotype matrices. The final SNP dataset consists of 10,000 SNPs, which was considered to be the number of input features without using reduction methods. Based on the corresponding phenotypes, the data was categorized into three sub-datasets named pheno1, pheno2, and pheno3, each corresponding to sample sizes of 500, 1000, and 2000, respectively. Histograms illustrating the distribution of phenotype values in these datasets are presented in Figure 1.



(a) Pheno1 Distribution (b) Pheno2 Distribution (c) Pheno3 Distribution

Figure 1: Phenotype histogram distribution on three datasets, where the phenotype values of each indicate a specific trait of plant, such as flowering time or flower color.

2.2 Deep neural network models

2.2.1 Neural networks overview

In terms of DL, a NN refers to a model that can train computing machines to process data in a way that imitates the human brain [Goodfellow et al., 2016]. A general NN model consists of an input layer, an output layer, and many hidden layers in between. Nodes (also called units or neurons) within each layer represent specific values; for example, nodes in the input layer correspond to input values, which can be a single value or a multi-dimensional value, such as vectors, arrays, or the so-called tensors used today.

The hidden layers play a crucial role in processing and transforming input data. To enhance the network's capacity and accuracy, we need to understand NN models that depend on the number of hidden layers and related hyperparameters, e.g., the number of input and output features per layer, activation function, and learning rate. The incorporation of non-linear activation functions serves the primary purpose of introducing non-linearity, enabling the network to effectively learn and model complex relationships within data [Burkov, 2019]. During a training process, the NN model adjusts its internal weights using gradient-based optimization, a method that aims to improve the accuracy of target prediction outputs. This adaptive mechanism allows the network to align its internal parameters with desired outputs, facilitating accurate predictions based on the provided dataset [Goodfellow et al., 2016].

The strength of NN models to capture complex non-linear features makes them particularly suitable for genetic data. The suitability arises from the inherent complexity and non-linear nature characterizing genetic datasets. This research internship focuses on three common deep NN models applied for phenotype prediction, including MLP, CNN, and RNN.

2.2.2 Model overview applied for phenotype prediction

In training prediction models, we employ Adam with exponential decay as an optimizer algorithm. The optimization hyperparameters, such as learning rate and weight decay, were selected during the hyperparameter search, with a focus on reducing the plateau of the validation loss [Kingma and Ba, 2014]. Additionally, for the non-linear activation function, we choose a Rectified Linear Unit (ReLU), Leaky Rectified Linear Unit (LeakyReLU), or a hyperbolic tangent (Tanh). Given the high-dimensional nature of SNP data, which results in a large number of input features, our model incorporates essential regularization techniques, namely dropout and early stopping [Burkov, 2019]. Further details on hyperparameter optimization are presented later in Subsection 2.3.1.

MLP: a classical type of feed-forward NN. Each layer in MLP is characterized by a linear transformation (a fully connected layer denoted by FC) followed by a non-linear activation function, establishing a non-linear mapping between input and output vectors [Goodfellow et al., 2016]. Mathematically, nodes in the input layer represent features (x_1, x_2, ..., x_n), where n is the number of features and $i = 1, 2, ..., n$ denotes the indices. The subsequent hidden layers process the input features through the FC layer involving weighted sums (W_j with j indicates node j) before applying an activation function (a_j). The output computation of each node is expressed by $y_k = a_j(\sum_{i=1}^{n} W_{ij} \cdot x_i + b_j)$. Ultimately, the entire network undergoes training through the adjustment of weights and biases during the back-propagation process [Goodfellow et al., 2016], utilizing optimization algorithms to minimize the error between predicted and actual outputs.

MLP model fine-tuning for prediction: shown in Figure 2 (A), our MLP implementation was structured by a number of hidden layers, each comprising an FC layer followed by an activation function. The output from these blocks further

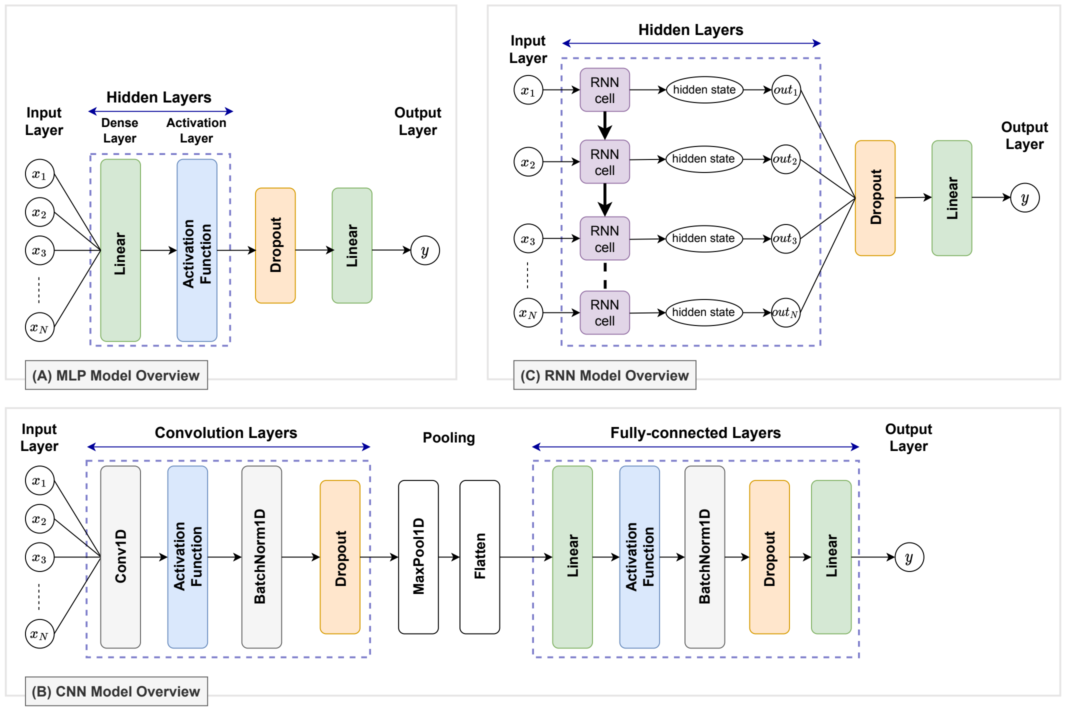


Figure 2: Scheme of three models used for phenotype prediction. These models present the main corresponding layers associated with hyperparameter optimization and the range of values during the tuning process, as shown in Subsection 2.3.1. In detail, Figure (A) is MLP model, (B) CNN Model, and (C) RNN model.

goes to a dropout layer, introducing a regularization mechanism. Finally, the FC output layer is appended to provide the final prediction values.

CNN: a type of NN that is frequently employed for processing structured grid data and tasks, i.e. image recognition. The fundamental building blocks of CNN include the convolutional layer (CL), pooling layer, and FC, followed by non-linear activation functions [LeCun et al., 2010]. Mathematically, CNN functions by utilizing CL that applies filters to the input data to detect specific features. The convolution operation is performed by sliding the filter over the input, computing dot products at each position, and the output of a CL layer is the feature map highlighting detected features. In contrast to an FC layer, where each neuron is connected to every neuron in the previous layer and each connection has a unique weight, a CL layer employs a kernel with trainable parameters that apply uniform weights to different segments of the inputs. This characteristic allows the network to automatically learn spatial hierarchies of features and recognize intricate patterns within image data. The pooling layers are employed to extract significant patterns from the input and diminish the spatial dimensions of convolved features to reduce the number of parameters and computation load [Goodfellow et al., 2016]. Hence, it can also control overfitting in the network.

CNN model fine-tuning for prediction: shown in Figure 2 (B), our CNN implementation was structured by different numbers of convolutional blocks, each composing of a Conv1D layer, an activation function, batch normalization, and dropout layer. The output from these blocks is then directed to a max pooling layer that only captures key features. Subsequently, the flattened layer was fed into an FC layer, which contains an activation function, batch normalization, and dropout layer. Finally, another FC layer is incorporated to yield the network's output, providing the final predictions.

RNN: another type of NN that is different from a traditional feed-forward model because it contains feedback loops. The key idea is that each unit of the recurrent layer has a hidden state, allowing them to retain information from the previous inputs. Basically, RNN operates on sequential data by processing one input at a timestep, updating its hidden state based on the current input and the information stored in the previous hidden states [Burkov, 2019]. Recurrent connection enables the network to capture temporal dependencies within sequential data [Goodfellow et al., 2016].

RNN model fine-tuning for prediction: shown in Figure 2 (C), our RNN implementation was structured into multi-stacks of each RNN cell. Then, the output from these cells is further directed to a dropout layer. Finally, the FC layer is added to deliver the final prediction values.

Table 1: An overview of tuned hyperparameters corresponding to each model. With the common parameters related to an optimizer, the algorithm was chosen Adam, where learning rate and weight decay are configured, as shown in the table. With the corresponding hyperparameters, the values of dropout rate and activation function are set the same across three models, while other values are set up depending on each model's architecture. The optimization algorithm explored different values within the [low, high] range in square brackets to find the optimal configuration. When the step value is not provided, the hyperparameter search considers any possible value from the range.

Object	**Hyperparameter name**	**Values**	**Description**
Optimiser	learning_rate	$[1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}]$	Learning rate
	weight_decay	$[1e^{-6}, 1e^{-2}]$	Weight decay
General model	dropout	$[0.1, 0.5, step = 0.05]$	Dropout rate
	activation	$[\text{LeakyReLU}, \text{ReLU}, \text{Tanh}]$	Activation function
MLP model	n_layers	$[1, 5, step = 1]$	The number of hidden layers
	out_factor	$[0.05, 0.7, step = 0.001]$	The ratio of the number of nodes to the number of input features
	pca	$[0.75, 0.95, step = 0.05]$	The percentage of variance explained in PCA
CNN model	n_layers	$[1, 4, step = 1]$	The number of convolutional layers
	kernel_size	$[2, 8, step = 1]$	The size of convolutional kernel
	stride	$[0.1, 1.0, step = 0.1]$	The ratio of stride to kernel size
	out_factor	$[0.2, 1, step = 0.1]$	The ratio of the number of nodes to the number of input features after convolutional layers
RNN model	n_layers	$[1, 3, step = 1]$	The number of stacks in each cells
	hidden_size	$[32, 64, 128, 512]$	The size of hidden states

2.3 Experimental Setup

As introduced in Subsection 2.3.1, we employ three separated SNP datasets named by their labels, i.e., pheno1, pheno2, and pheno3. The code is implemented in Python, mainly using Pytorch libraries, which serve as the baseline classes and interfaces for constructing DL models. Regarding hyperparameter optimization in training and tuning models, Optuna was used to search for the best hyperparameters [Yang and Shami, 2020].

2.3.1 Hyperparameter Tuning

Hyperparameters act as the rules for a learning algorithm, determining the specific values that a model learns through training. Thus, choosing the approriate hyperparameter values is crucial, especially in DL models, as these values directly influence model performance.

We used Optuna, which is a state-of-the-art Bayesian Optimization (BO) framework, for hyperparameter search [Akiba et al., 2019]. Unlike commonly used decision-theoretic methods like Grid-Search (GS) and Random-Search (RS), which involve exhaustive exploration of the hyperparameter search space and rely on the trial's error for satisfied results, Bayesian Optimization takes a more efficient route. It selects the next hyperparameter values based on previous evaluations, utilizing a surrogate model and an acquisition function. This strategic approach enables Optuna to identify optimal hyperparameters more effectively, changing them in promising directions likely to lead to a global optimum. In comparison to GS and RS, Optuna can avoid unnecessary configurations, making it efficient even in fewer iterations.

Technically, we employ TPESampler as the sampling strategy. The pruning strategy was incorporated with the aim of stopping an optimization process if the loss on a validation set fails to improve for a specific period (further detailed in Subsection 2.3.5). For an overview, Table 1 shows the range of tuned values employed in each model.

2.3.2 Data Preprocessing

In the transformation of the categorical SNP dataset into numerical values, two encoding methods were employed: additive encoding and one-hot encoding. Particularly, genotype matrices in each dataset only contain homozygous alleles. For MLP, we used additive encoding, representing the homozygous major alleles as 0 and the homozygous minor alleles as 2 [Mittag et al., 2015]. For CNN, we used one-hot encoding [John et al., 2022], where each nucleotide (A, C, T, G) is encoded as a binary array. For example, A is denoted by $[1, 0, 0, 0]$, C - $[0, 1, 0, 0]$, T - $[0, 0, 1, 0]$, G –

$[0, 0, 0, 1]$. These encoding methods were appointed to ensure the representation of genetic information suitable to the specific architecture and requirements of each model.

In addition, we also tried to observe the impact of further data normalization and dimension reduction techniques to encode data. For example, y labels are scaled by min-max normalization. With additive encoded data, X features are scaled by standard normalization, and their dimension can be further reduced by principal component analysis (PCA).

2.3.3 Data Splitting

To ensure robust model establishment and evaluation, the initial dataset was split into 90% for a training set and 10% for an independent test set. This division allows training the model on a substantial portion of the data while keeping a separate set for unbiased evaluation. Within training, the Bayesian hyperparameter search - Optuna was employed to tune the best models. We implemented a 5-fold cross-validation strategy by splitting the training set into five subsets, then using the mean performance on each iteration as the objective value for the Optuna tuning. After identifying the optimal hyperparameters, the final model was re-trained on the entire training dataset, and evaluated on the independent test dataset.

2.3.4 Evaluation Metrics

For evaluating both hyperparameter optimization and final model performance on the test dataset, we used two evaluation metrics: mean squared error (MSE) and Explained Variance (ExpVar).

Mean squared error (MSE) was used as a loss function between the target y and prediction $\widehat{y}$ values based on the number of input samples n during training. In hyperparameter tuning, MSE loss was used as the objective value over the cross-validation pipeline with five folds. The average loss value in each fold was considered as intermediate results for potential pruning. MSE lies in the range value $[0, 1]$, with lower values representing a better model performance, calculated by $L(\widehat{y}, y) = \frac{1}{n}(\widehat{y} - y)^2$.

Explained Variance was used to evaluate the overall performance of re-trained models. ExpVar quantifies how well predicted values $\widehat{y}$ account for the variance in the targets y. ExpVar serves as a robust metric for comparing the performance of each model. Represented on a scale from 0 to 1, higher ExpVar values indicate better models that can explain a greater percentage of variance in prediction. ExpVar is calculated by $ExpVar(y, \widehat{y}) = 1 - \frac{Var(y-\widehat{y})}{Var(y)}$.

From a practical perspective, both measures are important. A high MSE loss could indicate suboptimal predictive accuracy for phenotype traits, impacting the model's reliability in real-world applications. Besides, a low ExpVar would result in an insufficient explanation of the variance in predictions, hence misallocating resources to plants with low-quality phenotype traits.

2.3.5 Experimental Summary

We evaluate various configurations for each of the three models across three distinct phenotype datasets (called pheno1, pheno2, and pheno3). MLP is employed with additive encoding, while one-hot encoding is applied for the CNN and RNN models. As mentioned in Subsection 2.3.2, different preprocessing methods are evaluated to check their impacts on prediction models.

Each case of training MLP, CNN, RNN was tuned by 100 trials, utilizing the data splitting strategy mentioned in Subsection 2.3.3. Throughout the searching process, further configurations were set up, such as a maximum of 80 epochs, a batch size of 32 (as recommended by [Millstein, 2020]), a patience value of 20 epochs as a reference value for early stopping or pruning the tuning process. The implementation of a patience value implies that the training process would end early if no improvements on the validation set are observed within consecutive 20 epochs. This value can be set in Optuna with its built-in prune strategy for speeding up the searching process. Principally, if the current performance falls below the 60th percentile of the previous results, a non-promising trial would be terminated.

Finally, the best hyperparameters were utilized to re-train models on the entire dataset divided for training. For evaluation, the independent dataset divided for testing was used. Remarkably, the number of epochs for re-training was determined by an average value of early stopping points corresponding to the best trial with the best hyperparameters. Concretely, this value was recorded in each inner split during hyperparameter search, and the number of epochs with the highest accuracy was recorded. Then, these recorded values are averaged to get the number of epochs for re-train. The performance of each model with independent datasets can ensure a fair evaluation.

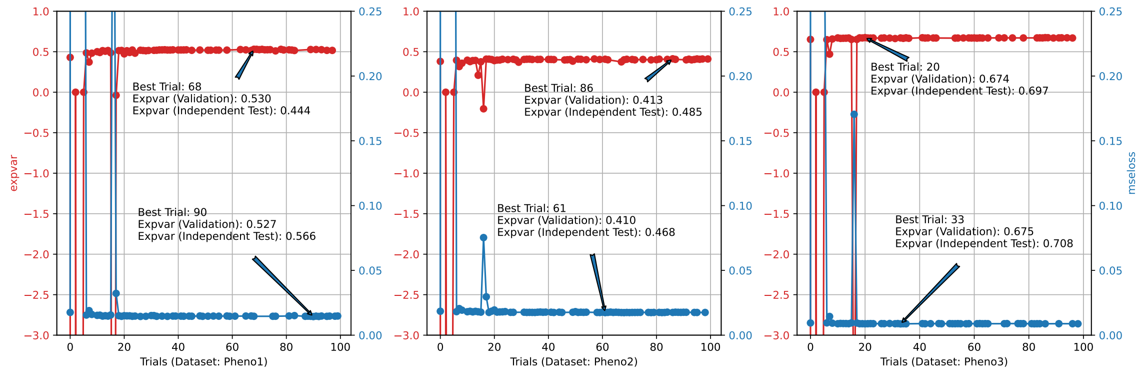


Figure 3: The history data of tuning hyperparameters on three phenotype datasets. The model trained is MLP, where the red line with the left y-axis indicates ExpVar values over each trial, and the blue line with the right y-axis indicates MSE losses over each trial. The number of trials shown on the x-axis is 100.

3 Results and Discussion

In this section, we present the performance results from training and tuning to testing each prediction model. First, the tuned results by using Optuna are analyzed with two objective values. Second, we discuss the impact of using different ways to preprocess data on prediction models, particularly in testing with MLP. Third, we present a comparison of the performance among three models on an independent test dataset, MLP, CNN, and RNN, respectively. The implementation of models, experiments as well as results can be found at `https://github.com/htnghi/intern_pheno_nnets.git`.

3.1 Result Summary

Hyperparameter Search - Objective: By using Optuna for hyperparameter search, we tried to employ two key metrics as the objective values: minimizing MSE loss or maximizing ExpVar. The purpose is to check whether tuning a prediction model has a large difference between these two directions.

As the results of the best trials correspond to each dataset and tuning-objective direction, the texts in Figure 3 annotate the average ExpVar values, where both values on validation and independent test are shown. Generally, the convergence points of best trials and the validated prediction performances do not differ significantly between tuning by MSE loss or ExpVar objective. For example, with dataset pheno2, the best trial of the MSE-loss-tuning direction is 86, and the best trial of the ExpVar-tuning direction is 61, where the validation ExpVar values for these two trials are 0.413 and 0.410; the independent-test ExpVar values are 0.485 and 0.468, respectively.

With dataset pheno3, there is a small difference in evaluation with the independent test between the MSE-loss-tuning direction and ExpVar-tuning direction, but that is trivial (0.697 compared to 0.708). Overall, the test results of the ExpVar-loss-tuning direction are slightly better than those using the MSE-loss objective. However, tuning by ExpVar direction might imply overfitting. Furthermore, the test results are not significantly higher; therefore, the selection of the best-tuned hyperparameters applying for the final models is still based on the MSE-loss-tuning direction. This indicates the balance between fitting to the training data and generalizing to unseen data, positioning MSE loss as a more reliable objective during hyperparameter search.

Impact of Different Data Preprocessing Techniques: To study the impact of different data preprocessing methods, we evaluate their influence based on prediction results. The comparison results are shown in Figure 4.

First, focusing on MLP, we used non-preprocessing and three preprocessing methods, including only y labels scaled by min-max scaler (`y_scaled`), `y_scaled` and X features scaled by standard normalization (`X_norm+y_scaled`), both `X_norm+y_scaled` and X dimension reduction by PCA (`X_norm+PCA+y_scaled`). Figure 4 (MLP) shows the results of MLP on both validation and independent tests. Based on the test results, MLP using only `y_scaled` obtains the best performance, reaching 0.6749 in dataset pheno3.

Second, with CNN and RNN, due to the limitations of the unsuitability of feature input shapes, we exclusively employ `y_scaled` for the y labels compared to non-preprocessing. In the case of CNN, impressive ExpVar results are observed on the validation set, reaching 0.8865 and 0.7433 in both non-preprocessing and `y_scaled`. However, on the independent test set, CNN's performance drops to about half compared to the validation results. Especially without

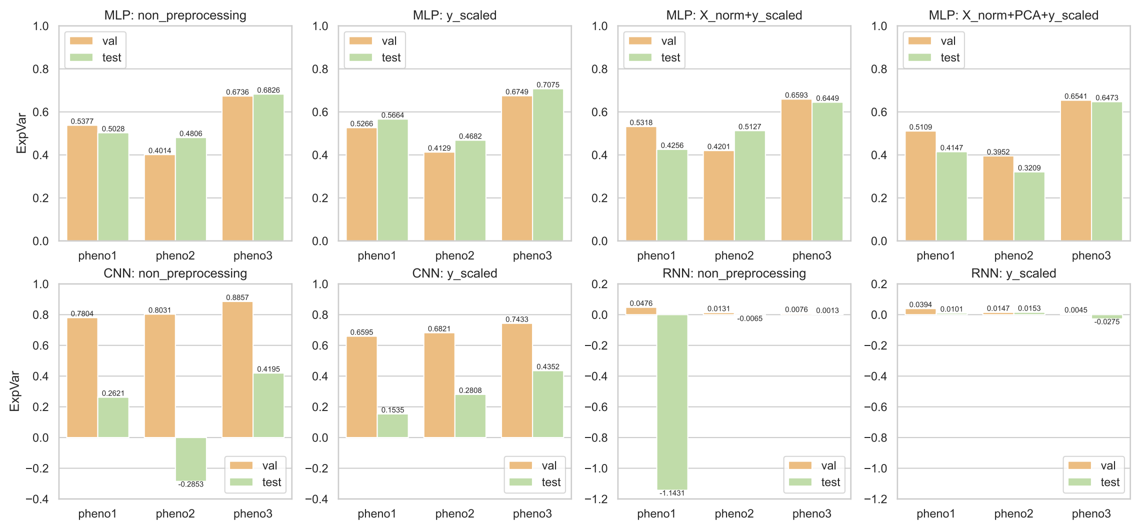


Figure 4: The results provide an overview of the impact of non-preprocessing and various preprocessing methods on three prediction models: MLP, CNN, and RNN. Each prediction model is represented by two bars: an orange bar showing ExpVar values on validation data during training and a green bar showing ExpVar values on independent test data after re-training for the final model.

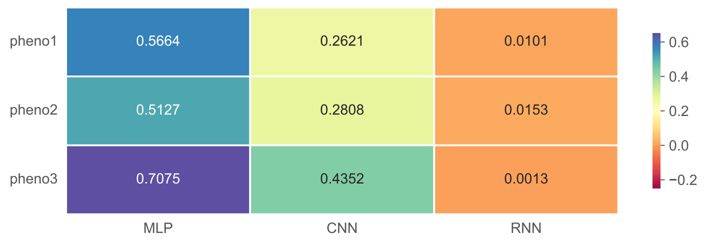


Figure 5: The heatmap of ExpVar results among MLP, CNN, and RNN. In which each corresponds to a best-tuned model and a phenotype dataset.

preprocessing, the test results are negative (-0.2853 for pheno2), which indicates a wrong prediction. Similarly, the validation and test results of RNN are poor, even reaching -1.1431 for pheno1 with non-preprocessed training data. These results underscore the challenges of RNN with the ability to learn and predict phenotypes effectively.

Given the favorable performance observed with `y_scaled`, we opt to implement this method consistently across three models, ensuring a fair comparison between these models.

Prediction Analysis on Independent Test Dataset: In evaluating the performance of all models on independent test datasets, we use ExpVar as the main metric. The number of epochs used to retrain the final models is the average early stopping value obtained from hyperparameter tuning. Figure 5 shows a heatmap where each cell presents the best ExpVar value obtained from the corresponding model and datasets, such as MLP with datasets pheno1, pheno2, pheno3. Overall, the comparative analysis across the three prediction models reveals that MLP consistently outperforms the others, while RNN has the least favorable results.

As we can see in Figure 5, the influence of the number of samples from pheno1 to pheno3 indicates that a larger dataset facilitates the performance of models. MLP and CNN with pheno3 (2000 samples) gain 14.11% and 17.31% better than pheno1 (500 samples). Unexpectedly, the results of pheno2 (1000 samples) are not in a similar trend, which is 5.37% worse than the results of pheno1. This prompts further investigation into the impact of sample size influence. However, we can say that an increasing number of samples, in general, corresponds to better model performance.

Furthermore, a discernible downward trend in performance is observed from MLP to CNN and finally to RNN. Specifically, MLP explains 70.75% of the variance in the dependent variable for pheno3, CNN explains 43.52%, while RNN explains only 0.13%. This emphasizes the important role of model selection in optimizing phenotype prediction.

3.2 Discussion

In summary, all three NNs demonstrated improvement with an increasing sample size. The MLP model consistently outperformed others across all considered phenotypic traits, reaching a maximum of 74.33% explained variance. Surprisingly, the more complex DL models, including CNN and RNN, did not demonstrate a superior ability to capture the dependency between phenotype and genotype compared to simpler models like MLP. There are many possible reasons for our model limitation in phenotype prediction.

From the data aspect, the abundance of SNPs in the dataset introduces considerable noise to the features. This high dimensionality incorporating the relatively low number of samples can pose a challenge in extracting meaningful patterns for NN-based approaches. Besides, the intricate interactions between SNPs can complicate the training and hyperparameter-tuning process, which potentially hinders the models' ability to generalize effectively on the independent set. This is also mentioned in accordance with the related investigations from [Montesinos-López et al., 2018] [Pook et al., 2020]; the observed trait values may be influenced by genetic architecture, or the current volume of available samples for plant phenotype prediction could still be insufficient.

Another limitation of our datasets, which comprise 10,000 features, is the substantial computational demand incurred when running models. Training NNs is computationally exhaustive, and in some cases, it is practically impossible. Therefore, a dimensionality reduction approach, i.e., PCA, is imperative to mitigate the computational burden by reducing the number of SNP features. However, it is important to acknowledge that such a reduction may lead to a loss of information. This aligns with our findings in the MLP model, where using PCA as a data preprocessing technique showed the worst predictive performance.

Furthermore, a notable limitation is observed in the encoding techniques employed. In the case of MLP, the additive encoding technique may inadequately capture intricate relationships within the data, leading to an inherent loss of crucial information. Consequently, the predictive ability of the model for phenotypic traits is worse. Similarly, the use of one-hot encoding in RNN, with results as depicted in Figure 5, proves suboptimal. These limitations underscore the necessity to explore more advanced encoding strategies in future investigations.

Lastly, utilizing the Bayesian Optimization for hyperparameter search, we applied the same number of trials across all prediction models. This approach ensures a fair and balanced exploration of the hyperparameter space for each model. Interestingly, our findings suggest that the less complex models, such as the MLP, outperformed more complex ones like CNN and RNN. Our observation could be explained by the relationship between the model complexity and the efficiency of hyperparameter tuning. In fact, the less complex models may navigate the search more effectively, which achieves the best hyperparameters than the complex models with the same trial setting.

4 Conclusions and Future Work

In this research internship, we explored the application of DL models for genomic studies, specifically focusing on Genomic Selection and testing these models across three distinct phenotype traits. The primary objective was to gain practical insights into the implementation of DL techniques, particularly in the context of working with genomic data.

We developed and evaluated three NN models: MLP, CNN, and RNN. These are integrated with different data preprocessing and encoding methods. MLP, employing min-max normalization and additive encoding, yielded the most favorable results compared to the others. In terms of ExpVar, MLP outperformed the second most effective model (CNN) by 23.19% for pheno2 and 27.23% for pheno3. The results underscore the promising potential of applying DL approaches in GS.

While the current prediction model shows a good result, the practical application in the real world remains challenging due to the observed limitations. Several potential ways to improve prediction performance include:

Firstly, we have focused on three basic NN models; hence, exploring more complex models, such as sequential models inspired by Natural Language Processing (NLP), can potentially understand the intricate patterns within genomic data. Moreover, utilizing different DL architectures may show their suitability for specific genomic prediction tasks.

Secondly, another important aspect for future work is to address the limitation of the small number of samples in our SNP dataset. This limitation restricts our understanding of the dataset's impact on performance. Therefore, to improve our comprehension and the model's learning capabilities, it is essential to explore alternative SNP datasets with more samples and additional phenotypic traits.

Thirdly, exploring alternative encoding methods beyond additive and one-hot encoding could contribute to capturing more complex relationships within the genomic data and be suitable for specific models. Specifically, for RNN, employing techniques such as embedding input features could be particularly relevant and beneficial.

Lastly, real-world phenotypic traits are often influenced by diverse environmental conditions. Relying only on genotype data might not suffice for accurate phenotype prediction. Therefore, future work should consider the incorporation of additional factors, such as environmental parameters. Accounting for these variables, as suggested by [Sandhu et al., 2021], holds promise for improving prediction accuracy, which enhances models' ability to predict complex traits.

In conclusion, our findings open a way for future research, such as encouraging the exploration of advanced encoding methods, the collection of larger datasets, and the optimization of hyperparameters. This ongoing exploration promises to refine our understanding and enhance the predictive capabilities of NN models in the challenging domain of plant phenotype prediction.

References

Hermann Buerstmayr, Maria Fernanda Dreccer, Dragana Miladinović, Lijuan Qiu, Istvan Rajcan, Jochen Reif, Rajeev K Varshney, and Johann Vollmann. Plant breeding for increased sustainability: challenges, opportunities and progress. *Theoretical and Applied Genetics*, 135(11):3679–3683, 2022. doi:https://doi.org/10.1007/s00122-022-04238-1.

George Acquaah. *Principles of plant genetics and breeding*. John Wiley & Sons, 2009.

N.F. Grinberg, O.I. Orhobor, and R.D. King. An evaluation of machine-learning for predicting phenotype: studies in yeast, rice, and wheat. *Machine Learning*, 209(2), 2022. doi:https://doi.org/10.1007/s10994-019-05848-5.

Mike Goddard. Genomic selection: prediction of accuracy and maximization of long-term response. *Genetica*, 136(2): 245–257, 2009. doi:https://doi.org/10.1007/s10709-008-9308-0.

Zeratsion Abera Desta and Rodomiro Ortiz. Genomic selection: genome-wide prediction in plant improvement. *Trends in plant science*, 19(9):592–601, 2014. doi:https://doi.org/10.1016/j.tplants.2014.05.006.

Xiaoqing Yu, Xianran Li, Tingting Guo, Chengsong Zhu, Yuye Wu, Sharon E Mitchell, Kraig L Roozeboom, Donghai Wang, Ming Li Wang, Gary A Pederson, et al. Genomic prediction contributing to a promising global strategy to turbocharge gene banks. *Nature Plants*, 2(10):1–7, 2016. doi:https://doi.org/10.1038/nplants.2016.150.

RA Fischer, Derek Byerlee, and Greg Edmeades. Crop yields and global food security. *ACIAR: Canberra, ACT*, pages 8–11, 2014.

Salvatore Esposito, Domenico Carputo, Teodoro Cardi, and Pasquale Tripodi. Applications and trends of machine learning in genomics and phenomics for next-generation breeding. *Plants*, 9(1):34, 2019. doi:https://doi.org/10.3390/plants9010034.

Khanh Le Nguyen, Alexandre Grondin, Brigitte Courtois, and Pascal Gantet. Next-generation sequencing accelerates crop gene discovery. *Trends in plant science*, 24(3):263–274, 2019. doi:https://doi.org/10.1016/j.tplants.2018.11.008.

Alain Vignal, Denis Milan, Magali SanCristobal, and André Eggen. A review on snp and other types of molecular markers and their use in animal genetics. *Genetics selection evolution*, 34(3):275–305, 2002. doi:https://doi.org/10.1051/gse:2002009.

Nastasiya F Grinberg, Alan Lovatt, Matt Hegarty, Andi Lovatt, Kirsten P Skøt, Rhys Kelly, Tina Blackmore, Danny Thorogood, Ross D King, Ian Armstead, et al. Implementation of genomic prediction in lolium perenne (l.) breeding populations. *Frontiers in plant science*, 7:133, 2016. doi:https://doi.org/10.3389/fpls.2016.00133.

Mark Cooper, Carlos D Messina, Dean Podlich, L Radu Totir, Andrew Baumgarten, Neil J Hausmann, Deanne Wright, and Geoffrey Graham. Predicting the future of plant breeding: complementing empirical evaluation with genetic prediction. *Crop and Pasture Science*, 65(4):311–336, 2014. doi:https://doi.org/10.1071/CP14007.

Theo HE Meuwissen, Ben J Hayes, and ME1461589 Goddard. Prediction of total genetic value using genome-wide dense marker maps. *genetics*, 157(4):1819–1829, 2001. doi:https://doi.org/10.1093/genetics/157.4.1819.

C Anilkumar, NC Sunitha, Harikrishna, Narayana Bhat Devate, and S Ramesh. Advances in integrated genomic selection for rapid genetic gain in crop improvement: a review. *Planta*, 256(5):87, 2022. doi:https://doi.org/10.1007/s00425-022-03996-y.

José Crossa, Paulino Pérez-Rodríguez, Jaime Cuevas, Osval Montesinos-López, Diego Jarquín, Gustavo De Los Campos, Juan Burgueño, Juan M González-Camacho, Sergio Pérez-Elizalde, Yoseph Beyene, et al. Genomic selection in plant breeding: methods, models, and perspectives. *Trends in plant science*, 22(11):961–975, 2017. doi:https://doi.org/10.1016/j.tplants.2017.08.011.

Juexin Wang, Trupti Joshi, Babu Valliyodan, Haiying Shi, Yanchun Liang, Henry T Nguyen, Jing Zhang, and Dong Xu. A bayesian model for detection of high-order interactions among genetic variants in genome-wide association studies. *Bmc Genomics*, 16(1):1–20, 2015. doi:https://doi.org/10.1186/s12864-015-2217-6.

Rostam Abdollahi-Arpanahi, Daniel Gianola, and Francisco Peñagaricano. Deep learning versus parametric and ensemble methods for genomic prediction of complex phenotypes. *Genetics Selection Evolution*, 52:1–15, 2020. doi:https://doi.org/10.1186/s12711-020-00531-z.

Chuang Ma, Mingming Xin, Kenneth A Feldmann, and Xiangfeng Wang. Machine learning–based differential network analysis: a study of stress-responsive transcriptomes in arabidopsis. *The Plant Cell*, 26(2):520–537, 2014. doi:https://doi.org/10.1105/tpc.113.121913.

Suneetha Uppu, Aneesh Krishna, and Raj P Gopalan. A deep learning approach to detect snp interactions. *J. Softw.*, 11 (10):965–975, 2016.

Zhaohui Liang, Jimmy Xiangji Huang, Xing Zeng, and Gang Zhang. Dl-adr: a novel deep learning model for classifying genomic variants into adverse drug reactions. *BMC medical genomics*, 9(2):195–204, 2016. doi:https://doi.org/10.1186/s12920-016-0207-4.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. doi:https://doi.org/10.1038/nature14539.

Seonwoo Min, Byunghan Lee, and Sungroh Yoon. Deep learning in bioinformatics. *Briefings in bioinformatics*, 18(5): 851–869, 2017. doi:https://doi.org/10.1093/bib/bbw068.

Maura John, Florian Haselbeck, Rupashree Dass, Christoph Malisi, Patrizia Ricca, Christian Dreischer, Sebastian J Schultheiss, and Dominik G Grimm. A comparison of classical and machine learning-based phenotype prediction methods on simulated data and three plant species. *Frontiers in Plant Science*, 13:932512, 2022. doi:https://doi.org/10.3389/fpls.2022.932512.

Christina B Azodi, Emily Bolger, Andrew McCarren, Mark Roantree, Gustavo de Los Campos, and Shin-Han Shiu. Benchmarking parametric and machine learning models for genomic prediction of complex traits. *G3: Genes, Genomes, Genetics*, 9(11):3691–3702, 2019. doi:https://doi.org/10.1534/g3.119.400498.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. doi:http://www.deeplearningbook.org.

Andriy Burkov. *The hundred-page machine learning book*, volume 1. Andriy Burkov Quebec City, QC, Canada, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Published as a conference paper at ICLR 2015*, 2014. doi:https://doi.org/10.48550/arXiv.1412.6980.

Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256. IEEE, 2010. doi:https://doi.org/10.1109/ISCAS.2010.5537907.

Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020. doi:https://doi.org/10.1016/j.neucom.2020.07.061.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD'19, pages 2623–2631, 2019. doi:https://doi.org/10.1145/3292500.3330701.

Florian Mittag, Michael Römer, and Andreas Zell. Influence of feature encoding and choice of classifier on disease risk prediction in genome-wide association studies. *PloS one*, 10(8):e0135832, 2015. doi:https://doi.org/10.1371/journal.pone.0135832.

Frank Millstein. *Convolutional neural networks in Python: beginner's guide to convolutional neural networks in Python*. Frank Millstein, 2020.

Osval A Montesinos-López, Abelardo Montesinos-López, José Crossa, Daniel Gianola, Carlos M Hernández-Suárez, and Javier Martín-Vallejo. Multi-trait, multi-environment deep learning modeling for genomic-enabled prediction of plant traits. *G3: Genes, genomes, genetics*, 8(12):3829–3840, 2018. doi:https://doi.org/10.1534/g3.118.200728.

T Pook, J Freudenthal, A Korte, and H Simianer. Using local convolutional neural networks for genomic prediction. front genet 11: 561497, 2020.

Karansher S Sandhu, Dennis N Lozada, Zhiwu Zhang, Michael O Pumphrey, and Arron H Carter. Deep learning for predicting complex traits in spring wheat breeding program. *Frontiers in Plant Science*, 11:613325, 2021. doi:https://doi.org/10.3389/fpls.2020.613325.