Genomic selection with deep neural networks

by

Riley McDowell

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Plant Breeding

Program of Study Committee:
David Grant, Co-Major Professor
William Beavis, Co-Major Professor
Thomas Lübberstedt

Iowa State University

Ames, Iowa

2016

TABLE OF CONTENTS

ABSTRACT

Reduced costs for DNA marker technology has generated a huge amount of molecular data and made it economically feasible to generate dense genome-wide marker maps of lines in a breeding program. Increased data density and volume has driven an exploration of tools and techniques to analyze these data for cultivar improvement. Data science theory and application has experienced a resurgence of research into techniques to detect or "learn" patterns in noisy data in a variety of technical applications. Several variants of machine learning have been proposed for analyzing large DNA marker data sets to aid in phenotype prediction and genomic selection. Here, we present a review of the genomic prediction and machine learning literature. We apply deep learning techniques from machine learning research to six phenotypic prediction tasks using published reference datasets. Because regularization frequently improves neural network prediction accuracy, we included regularization methods in the neural network models. The neural network models are compared to a selection of regularized Bayesian and linear regression techniques commonly employed for phenotypic prediction and genomic selection. On three of the phenotype prediction tasks, regularized neural networks were the most accurate of the models evaluated. Surprisingly, for these data sets the depth of the network architecture did not affect the accuracy of the trained model. We also find that concerns about the computer processing time needed to train neural network models to perform well in genomic prediction tasks may not apply when Graphics Processing Units are used for model training.

CHAPTER 1
GENERAL INTRODUCTION

Introduction

Humans have been breeding plants for food since the dawn of agriculture. This process emerged from a haphazard form of artificial selection where farmers would save seed from the highest yielding, healthiest, and easiest to harvest plants to plant the following year. Over thousands of years, this process produced many of the plants of agricultural value today.

Between 1930 and 1960, academic agricultural and plant breeding communities were setting the stage for agricultural giants such as Norman Borlaug and Henry A. Wallace. These individuals explored options for breeding plants for specific traits and applying rigorous, scientifically-motivated management practices to improve varieties and the total yield of agriculture worldwide.

As an example, in 1930, corn was an open-pollinated crop with an average yield around 20-30 bu/ac. By 1960, field corn had transitioned to become a single-cross hybrid crop producing twice the yield of those only two or three decades earlier. The implementation of agricultural practices that would drive the green revolution had been kicked off, and the stage was set for a rapid improvement in yield and genetic gain in corn as well as many other crops which continued into the 2000s (Evenson and Gollin, 2003).

The green revolution drove crop production yields higher through a combination of improved agricultural practices and varieties that had greater yield potential and agronomic stability. By the year 2000, advances in agronomy were slowing, and molecular breeding technologies were promising the next advancement in crop yield through a focus on a better understanding of crop genetics and genomics. The advent of Bt Corn was the landmark technology delivering on the promise of biotechnology in agriculture. Bt corn was first com-

mercialized in in 1996 and accounted for 63% of U.S. grain corn by 2010 (Fernandez-Cornejo *et al.*, 2012).

Biotechnology in agriculture is not limited only to transgenic traits. The cost of DNA sequencing and genetic marker assays began to fall as the green revolution came to a close. The cost of single nucleotide polymorphism (SNP) assays has dropped so sharply that today it is possible to apply them to many thousands of samples in large breeding programs (Hiremath *et al.*, 2012). In many cases, it is now less costly to genotype a seed than to grow it and observe its phenotype.

Because a whole genome SNP assay can serve as an approximation for the genome sequence, it is possible to use a SNP haplotype to predict the genetic potential of a plant in an affordable way. This process is known as "genome-enabled prediction" hereafter called genomic prediction, and when used to inform breeding program selections is known as genomic selection. Genomic prediction is typically applied early in a breeding program as a way to increase the overall selection pressure in a breeding program and thus increase the rate of genetic gain. A review of the literature and methods to perform this technique is presented in Section 1.3.

The genetic gain in a breeding program is directly related to many factors. A general guideline to expected genetic gain each year on an entry mean basis for an inbred breeding program is presented in Equation 1.1 (Fehr, 1987). The equation is often used as a general guideline to explore whether changes in breeding strategy will have a positive or negative effect on genetic gain.

$$
\begin{aligned}
G_y &= \text{genetic gain per year} \\
k &= \text{selection differential} \\
r &= \text{number of replications} \\
t &= \text{number of environments} \\
n &= \text{plants per plot} \\
\sigma_A^2 &= \text{additive genetic variation} \\
\sigma_u^2 &= \text{within-plot environmental variation} \\
\sigma_{wg}^2 &= \text{within-plot genetic variation} \\
\sigma^2 &= \text{between-plot variation} \\
\sigma_{ge}^2 &= \text{genotype by environment interaction variation} \\
\sigma_g^2 &= \text{genotypic variation} \\
G_y &= \frac{k\sigma_A^2}{y\sqrt{\frac{\frac{\sigma_u^2+\sigma_{wg}^2}{n}+\sigma^2}{rt}+\frac{\sigma_{ge}^2}{t}+\sigma_g^2}}
\end{aligned}
\tag{1.1}
$$

From Equation 1.1, it is clear that improvements in genetic gain per year can be made by increasing the number of replications and/or environments tested, increasing the number of plants within each experimental plot, or increasing overall selection pressure during the breeding program. Genomic selection improves genetic gain by addressing all of these facets simultaneously.

- A breeder may grow many more plants than can be evaluated using field plots in early generations. This allows a program to increase the overall selection differential without reducing the number of new lines at the end of the breeding pipeline.
- Because genomic prediction estimates genetic potential directly from genetic information, genotype by environment interaction is reduced to zero.

- When data collection is properly calibrated and quality controlled typically only one SNP assay is required per sample. This results in near zero measurement error of the genetic information of a plant. This reduces within-plot and between-plot (or more accurately, within genotype and between genotype) variation from all sources to zero.

However, unlike in the genetic gain equation, genomic prediction introduces prediction error to the denominator of the equation. When practicing phenotypic selection, measurement errors directly reduce the rate of genetic gain. When performing genomic prediction, measurement errors affect the training data that the statistical model uses to make predictions, reducing its predictive accuracy. This error can be minimized by taking accurate measurement like with phenotypic selection, but also benefits from exposure to data from related populations or previous testing years. Reducing the total genomic prediction error thus has a direct positive impact on genetic gain. By comparing the magnitude of prediction error with the cost of executing early-generation SNP assays, it is possible to determine if genomic selection is beneficial to a breeding program. This thesis explores the application of regularized neural networks to minimizing genomic selection prediction error.

Thesis Organization

This thesis adheres to the Iowa State Journal Paper format. Chapter 1 begins with an introduction to genomic selection and a review of the genomic selection and prediction literature. Chapter 2 contains an article to be submitted to G3 which evaluates the application of regularized neural network architectures to six genomic prediction problems. Chapter 3 outlines the results of the results presented in Chapter 2 in the context of genomic prediction and data science literature, with a special emphasis on potential future work. The thesis ends with a bibliography section and several appendices describing the location of the raw data, the code required to reproduce the results of the study in Chapter 2, and the code required to generate the text of this thesis itself.

Literature Review

QTL mapping and MAS

The wide availability and reduced cost of molecular marker technology has created opportunities to perform marker assisted selection of genotypes in plant and animal breeding. Quantitative Trait Locus (QTL) mapping techniques have proved useful for identifying markers genetically associated with genes conditioning agronomic phenotypes (Miles and Wayne, 2008). Using identified QTL to support selection has grown in popularity as molecular marker data costs have decreased. Once a sufficient proportion of QTL-associated markers have been identified, the associated markers can be leveraged for making selections on a population. Individuals in a population genotyped for previously identified QTL-associated markers can be selected based on the presence of desired alleles and/or haplotypes in a process known as Marker Assisted Selection (MAS). MAS is usually most effective for traits with a few large effect alleles with high heritability. However, when making selections on traits with many contributing genes with small effect distributed widely across a genome, MAS becomes less effective (Heffner *et al.*, 2009).

In order to apply MAS to traits with diverse genetic architectures, it is advisable to combine MAS on juvenile individuals and subsequent phenotypic selection on adults with favorable MAS scores (Lande and Thompson, 1990). This process, known as two-stage selection, has proven effective for improving the coefficient of selection of breeding programs while avoiding expensive phenotypic trials for individuals with low genetic potential. While inexpensive, two-stage selection only utilizes those markers associated with QTL with significant effects. The present cost of genome wide SNP assays has fallen dramatically enough that today individuals are frequently genotyped for hundreds, thousands, or even tens of thousands of SNPs. Information in SNPs that are not significantly associated with a trait are lost when applying MAS and two-stage selection, but may still have an small effect on

the expressed phenotype and thus could provide improved predictive accuracy of individual's genetic value.

Genomic Prediction

Unlike QTL mapping and associated MAS techniques, genomic prediction methods attempt to predict phenotypes utilizing all available SNP marker data collected from a population, using one of many possible statistical models to predict the marker-trait associations in a data driven way (Meuwissen *et al.*, 2001). The accuracy of genomic prediction relies on an appropriate choice of a statistical model to capture the relationship between the genetic architecture of a trait and the underlying marker calls in a panel of high-density marker data. It is likely that the best statistical model for genomic prediction is dependent on the genetic architecture of the predicted trait (Crossa *et al.*, 2010; Gonzalez-Camacho *et al.*, 2012; Resende *et al.*, 2012; Cleveland *et al.*, 2012; Thavamanikumar *et al.*, 2015). From a mathematical perspective, models incorporating interactions between marker features have the capacity to achieve higher accuracy by capturing non-additive effects. Experimental results support this hypothesis (Gonzalez-Camacho *et al.*, 2012). Alternative prediction methods continue to be an active area of research in plant and animal breeding (de Koning and McIntyre, 2012).

Once an accurate and predictive model of a QTL is discovered and a SNP marker assay has been conducted on an individual, it is trivial to convert the underlying predictions into a selection index. If the predictive model is selected such that it captures only additive effects, the resulting predictions can be considered to be an estimate of the breeding value of the assayed individual. To differentiate breeding values estimated from genomic panels from breeding values calculated via traditional phenotypic measurements and best linear unbiased prediction (BLUP) models, the term genome estimated breeding value or GEBV was coined (Meuwissen *et al.*, 2001).

Choosing a Genomic Prediction Model

Genomic prediction presents a distinct mathematical challenge compared to MAS. When conducting MAS, a large number of individuals n are evaluated at a comparatively smaller number of loci p. In a general sense, this corresponds to solving an overdetermined system of linear equations. The large family of regression techniques that minimize a least-squares loss function are well behaved on overdetermined systems. Genomic prediction is characterized by the opposite scenario where $n < p$. Typically a smaller number of individuals are genotyped at a larger number of marker loci. These problems can be solved using least squares regression, but also require that a regularization penalty is included in the calculations in addition to the least-squares loss function that is used to select between possible solutions to the underdetermined system.

There are many forms of regularization. Perhaps the best known is L2 regularization, which penalizes large regression coefficients in least squares regression problems (Tibshirani, 1996). Because L2 regularization penalizes the squares of the coefficients in the model, solutions that place a small coefficient on many available input features produce a lower value for the model's loss function than a model with equivalent accuracy that uses large coefficients on only a few input features. This results in a trained model that tends to place a small coefficient on all available input features. Ridge regression is an example of an L2 regularized ordinary least squares regression.

Another common form of regularization is L1 regularization (Tibshirani, 1996). L1 regularization penalizes the sum of the regression coefficients in least squares regression problems. This causes solutions that set many input feature coefficients to zero to produce a lower total loss value than alternatives with equivalent prediction accuracy that utilize more input features. As a result, L1 regularization tends to produce solutions that set non-informative feature's coefficients to zero. Least absolute shrinkage and selection operator (LASSO) regression is an L1 regularized ordinary least squares regression. Both L1 and L2 regularization

techniques can be applied to the weights of a neural network's neurons as described in Subsection 1.3.

Different regularization techniques such as L1 and L2 regularization have a relationship to the genetic architecture of the trait they being used to predict. If a trait is associated with many small effect markers, models incorporating L2 regularization are likely to perform better than unregularized models. Classical MAS traits with a small number of large effect markers may be best predicted by algorithms incorporating L1 regularization. Traits falling somewhere in between may do well with models incorporating a combination of L1 and L2 regularization such as elastic net regression (Zou and Hastie, 2005).

A wide variety of regularization techniques exist. Some are broadly used and simple to reason about like L1 and L2 regularization. Others are applicable only to certain classes of mathematical models such as assumed prior distributions in Bayesian regression methods. When choosing a tool for genomic prediction, it is critical to evaluate the available regularization techniques with multiple prediction methods in a data-driven way. These comparisons will ideally identify a single best model with zero or more regularization techniques which can be used to make accurate predictions for the traits of interest.

Data Science

As interest in using genomic prediction as a breeding tool grew, the research in the interdisciplinary field known as data science also increased. Data scientists use machine learning and statistics to make predictions, usually by applying ideas or techniques from a wide variety of domains including mathematics, statistics, and computer science. Often, a data scientist's focus is to create a predictive model that may not be associated with an underlying generative model. Some view this as the distinction between data science and classical statistics (Donoho, 2015; Breiman, 2001). The rapid increase in popularity of data science is associated with better definitions of best practices for predictive modeling across

many disciplines as well as software packages to automate the process of building predictive models from any data source.

Data scientists have used their experience to improve existing statistical models and tackle problems of immense complexity, some of which were previously thought to be unsolvable. Data scientists have applied neural networks to recognize handwritten text or generate transcripts of spoken words from real-time audio recordings (Lang *et al.*, 1990). Data scientists compared the performance of a neural networks model to a traditional logistic regression model used to detect signals indicating epilepsy in electroencephalograph readings and found the network unanimously outperformed the existing model (Subasi and Ercelebi, 2005). A complex neural network architecture was used to play seven different Atari 2600 games using screen pixels as input and training the network to maximize a score metric for the game it was trained on (Mnih *et al.*, 2013). These results demonstrate that machine learning, and neural networks in particular can be employed to solve problems or improve predictive accuracy on large and complex datasets from a wide variety of fields.

These successes have renewed interest in the machine learning field in both private and public sectors. Many universities now offer advanced cross-disciplinary degrees in data science that include statistics, mathematics, and computer science training. Genomic prediction is one of many opportunities for data scientists to apply these tools to plant and animal breeding.

Neural Network Architecture

Neural networks are a type of model frequently employed by data scientists for predictive modeling. Neural networks consist of layers of interconnected neurons which map inputs to one or more outputs. Each neuron in a network can be expressed as a transformation of a weighted sum of n inputs

$$output_{lk} = f_l\left(\sum_{i=1}^{n}(w_{lki} * x_i) + b_{lk}\right) \tag{1.2}$$

where $output_{lk}$ is the output from neuron k in network layer l having activation function f_l, weights w_{lki} and bias b_{lk}.

A neural network thus is a collection of neurons that map a length n input vector $x = (x_1, ..., x_n)$ through a series of j "hidden" layers $(l_1, ..., l_j)$. Each hidden layer consists of a variable number of neurons, each of which apply an associated coefficient, bias, and mathematical transformation to their input and forward the result on to every neuron in the subsequent layer forming an interconnected network (Figure 1.1).

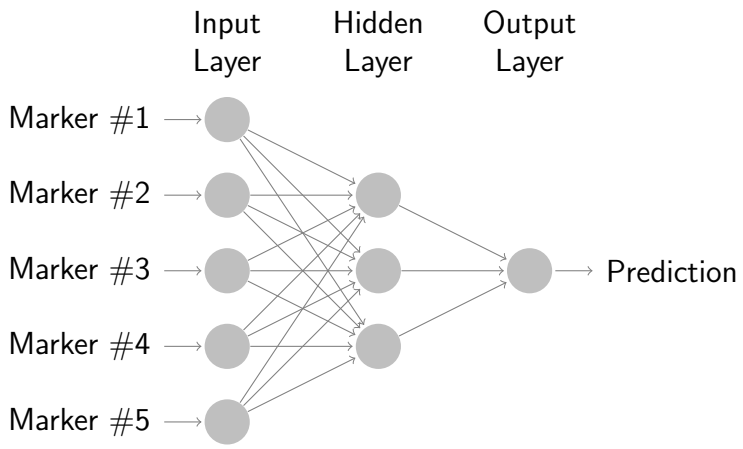

Figure 1.1: A multi-layer perceptron neural network with a single hidden layer. Many input calls are mapped into the hidden layer of neurons. For genomic prediction, the input layer consists of one neuron per marker and the output consists of a single neuron which combines the information from the final hidden layer to predict a phenotype or BLUP. Because this network has only a single hidden layer, it would not be considered a deep network.

Once the network is defined, it must be exposed to input and desired output values, and adjusted to minimize error in output in a process known as training. The weights and biases are often initialized by drawing from a random normal distribution. From this initial state, error in the output of the network is propagated back through the hidden layers, and the weights and biases are updated in the direction that would decrease output error on many randomly drawn subsets of the input data. This turns the network training process into a general function minimization problem where the parameters to the function are the weights and biases of the network neurons and the function to be minimized is the squared differences between the network outputs and the desired true values. The process of propagating output

error back through a neural network is known as backpropagation, and has been used and improved extensively since its original description in the 1980s (Rumelhart *et al.*, 1986). Typically, the training data is split evenly into representative, randomly sampled collections of data called batches. The training algorithm exposes the network to each batch until they are depleted, after which the process is repeated. Each collection of batches is known as an epoch, and training typically involves applying backpropagation for several hundred epochs, or until the network weights and biases reach an equilibrium state that has converged to a globally minimum amount of prediction error.

It is trivial to add additional hidden layers to an existing neural network model (Compare Figure 1.1 to Figure 2.1). Despite the ease of describing their architecture, networks with many hidden layers have been notoriously difficult to train. The amount of error attributed to a neuron by backpropagation decreases in magnitude with each additional layer added to the network. As a result, layers nearest the input train very slowly in a phenomenon known as the vanishing gradient problem (Hochreiter, 1998). Recently, a series of breakthroughs in neural network training along with well-known increases in computer processing speeds have allowed efficient training of deeper networks than were previously possible (Sutskever *et al.*, 2013). A history of the deep learning literature is available in LeCun *et al.* (2015). The increased training efficiency and potential to capture subtle correlation relationships between two or more input features drove a need to differentiate these deeper networks from prior work, resulting in the emergence of the phrase "deep learning" to describe the construction and training of deep neural networks.

Neural Networks for Genomic Prediction

Previous attempts to apply neural networks to genomic prediction have resulted in an overfitting of the network to the training data and raised concerns about the computation time required to fit the model to datasets containing many markers across many genotypes (Heslot *et al.*, 2012; González-Recio *et al.*, 2014). In retrospect, these results are not sur-

prising. Multi-layer feedforward neural networks are capable of approximating functions of arbitrary complexity to arbitrary accuracy if provided enough neurons in even a single hidden layer. This property of neural networks is known as the universal approximation theorem, and can result in overfitting if the weights of the network are not regularized in some way (Kur *et al.*, 1989).

Given the promising results from regularized and Bayesian methods for genomic prediction such as ridge or LASSO regression and the Bayesian family of regressors, it is prudent to evaluate some of the many of neural network training algorithms which incorporate regularization of weights during training. Today, networks based on these and other regularization techniques continue to show success across many domains (Schmidhuber, 2015). Similarly, while neural networks are computationally demanding to train, the training algorithms themselves are often easily expressed with vector and matrix algebra. These algorithms are well suited to execution on Graphics Processing Unit (GPU) hardware, with reports of up to a sixty-fold speedup in training time (Sierra-Canto *et al.*, 2010; Schmidhuber, 2015).

Genomic Selection in a Breeding Program

Genomic selection is practiced by all major plant breeding companies today. Typically, this is accomplished by increasing the number of progeny evaluated early in a breeding program and practicing intense selection based on genomic prediction values. It is now feasible to phenotypically evaluate a randomly selected subset of a cohort of progeny, while genotyping the entire cohort. It is then trivial to build a genomic prediction model from the subset of the progeny with both phenotypic and genotypic data and use the resulting model to make selections on the entire cohort.

One advantage to using genomic prediction methods over MAS is that the patterns in genotypic data that are used for selections naturally regenerate new haplotypes after each recombination event. It has been hypothesized that selecting directly on this information rather than on phenotypic measurements alone may help maintain diversity in a breeding

program (Daetwyler *et al.*, 2007). Other work using either a theoretical high-investment maize breeding program or a low-investment winter wheat breeding program has demonstrated that genetic gain per year could be improved by utilizing genomic selection rather than MAS (Heffner *et al.*, 2010).

Beyond maintaining genetic diversity and increasing genetic gain in a breeding program, genomic selection may also allow breeders to characterize the performance of allele combinations in environments that are critical to a target market but are rarely observed. Heffner *et al.* (2009) suggest that by capturing genotype by environment interaction by modeling genotype performance in severe weather years it may be possible to characterize lines in non-severe years while still enabling selections for traits such as severe weather hardiness or severe drought tolerance.

The adoption of genomic selection and the use of GEBVs in commercial plant breeding has been rapidly increasing as molecular marker technology such as dense marker arrays have become less expensive. Heffner *et al.* (2009) offers the possibility that breeding programs may eventually transition to using genomic selection as a primary selection method in a breeding program with phenotypic evaluation, at least early in a breeding program, used primarily for training statistical models of genotypic performance or updating models to improve predictions on new genotypes or recombinations. These same models could then be used to identify parent candidates without performing expensive field trials. Yield trials would only be strictly needed at the end of a breeding program prior to verify general agronomic performance prior to cultivar release.

The future state described by Heffner *et al.* (2009) where breeding program selections are driven primarily by data from predictive models rather than direct measurements is not unlike the transformation that is currently underway in other industries. Both of these transformations are driven by the growth of data science as a field, though the moniker itself has not been adopted as widely as the techniques it encompasses. Google and Facebook have developed ways to match customers with advertisements for products they are most

likely to purchase. Breeding companies are seeking individuals with the same skills to join their research programs and drive innovation in data analysis and interpretation beyond that which can be achieved by classical statistics alone. As breeding companies apply genomic selection more widely, the choice of mathematical model to use for making selections will become more important. Small percentage improvements in accuracy could generate much larger improvements in genetic gain over the lifetime of a breeding program.

CHAPTER 2
GENOMIC PREDICTION WITH DEEP NEURAL NETWORKS

A paper to be submitted to G3

Riley McDowell and David Grant

Abstract

Reduced costs for DNA marker technology has generated a huge amount of molecular data and made it economically feasible to generate dense genome-wide marker maps of lines in a breeding program. Increased data density and volume has driven an exploration of tools and techniques to analyze these data for cultivar improvement. Research into data science theory and application has experienced a resurgence of research into techniques to detect or "learn" patterns in noisy data in a variety of technical applications. Several variants of machine learning have been proposed for analyzing large DNA marker data sets to aid in phenotype prediction and genomic selection. Here, we apply deep learning techniques from machine learning research to six phenotypic prediction tasks using published reference datasets. Because regularization frequently improves neural network prediction accuracy, we included regularization methods in the neural network models. The neural network models are compared to a selection of Bayesian and linear regression techniques commonly employed for phenotypic prediction and genomic selection. Applying regularization improved neural network accuracy. On three of the phenotype prediction tasks, regularized neural networks were the most accurate of the models evaluated. Surprisingly, for these data sets the depth of the network architecture did not affect the accuracy of the trained model. We also find that concerns about the computer processing time needed to train neural network models

to perform well in genomic prediction tasks may not apply when Graphics Processing Units are used for model training.

Introduction

The wide availability and reduced cost of molecular marker technology has created opportunities to perform marker assisted selection of genotypes in plant and animal breeding. Quantitative Trait Locus (QTL) mapping techniques have proved useful for identifying markers genetically associated with genes conditioning agronomic phenotypes (Miles and Wayne, 2008). Using identified QTL to support selection has grown in popularity as molecular marker data costs have decreased. Once a sufficient proportion of QTL-associated markers have been identified, the associated markers can be leveraged for making selections on a population. Individuals in a population genotyped for previously identified QTL-associated markers can be selected based on the presence of desired alleles and/or haplotypes in a process known as Marker Assisted Selection (MAS). MAS is usually effective on traits with a few large effect alleles with high heritability. However, when making selections on traits with many contributing genes with small effect distributed widely across a genome, MAS becomes less effective (Heffner *et al.*, 2009).

In order to apply MAS to traits with diverse genetic architectures, it is advisable to combine MAS on juvenile individuals and subsequent phenotypic selection on adults with favorable MAS scores (Lande and Thompson, 1990). This process, known as two-stage selection, has proven effective for improving the coefficient of selection of breeding programs while avoiding expensive phenotypic trials for individuals with low genetic potential. While inexpensive, two-stage selection only utilizes those markers associated with QTL with significant effects. The present cost of genome wide SNP assays has fallen dramatically enough that today individuals are frequently genotyped for hundreds, thousands, or even tens of thousands of SNPs. Information in SNPs that are not significantly associated with a trait

are lost when applying MAS and two-stage selection, but may still have an small effect on the expressed phenotype and thus could provide improved predictive accuracy of individual's genetic value.

Unlike QTL mapping and associated MAS techniques, genomic prediction methods attempt to predict phenotypes utilizing all available SNP marker data collected from a population, allowing one of many possible statistical models to predict the marker-trait associations in a data driven way (Meuwissen *et al.*, 2001). The accuracy of genomic prediction relies on an appropriate choice of a statistical model to capture the relationship between the genetic architecture of a trait and the underlying marker calls in a panel of high-density marker data. It is likely that the best statistical model for genomic prediction is dependent on the genetic architecture of the predicted trait (Crossa *et al.*, 2010; Gonzalez-Camacho *et al.*, 2012; Resende *et al.*, 2012; Cleveland *et al.*, 2012; Thavamanikumar *et al.*, 2015). From a mathematical perspective, models incorporating interactions between marker features have the capacity to achieve higher accuracy by capturing non-additive effects. Experimental results support this hypothesis (Gonzalez-Camacho *et al.*, 2012). Alternative prediction methods continue to be an active area of research in plant and animal breeding (de Koning and McIntyre, 2012).

Concurrent with the advent of genomic selection as a practice the popularity of the interdisciplinary field known as data science has increased. Practitioners of data science apply machine learning and statistics to make predictions, usually by applying ideas or techniques from a wide variety of domains including mathematics, physics, and computer science. Often, a data scientist's focus is to create a predictive model than may not be associated with an underlying generative model. This can be viewed as the distinction between data science and classical statistics (Donoho, 2015; Breiman, 2001). The rapid increase in popularity of data science is associated with better definitions of best practices for predictive modeling across many disciplines as well as software packages to automate the process of building predictive models from any data source.

Neural networks are a type of model frequently employed by data scientists for predictive modeling. Neural networks consist of layers of interconnected neurons which map inputs to one or more outputs. Each neuron in a network can be expressed as a transformation of a weighted sum of n inputs

$$output_{lk} = f_l\left(\sum_{i=1}^{n}(w_{lki} * x_i) + b_{lk}\right) \tag{2.1}$$

where $output_{lk}$ is the output from neuron k in network layer l having activation function f_l, weights w_{lki} and bias b_{lk}.

A neural network is thus a collection of neurons that map a length n input vector $x = (x_1, ..., x_n)$ through a series of j "hidden" layers $(l_1, ..., l_j)$. Each hidden layer consists of a variable number of neurons, each of which apply an associated coefficient, bias, and mathematical transformation to their input and forward the result on to every neuron in the subsequent layer forming an interconnected network (Figure 2.1).

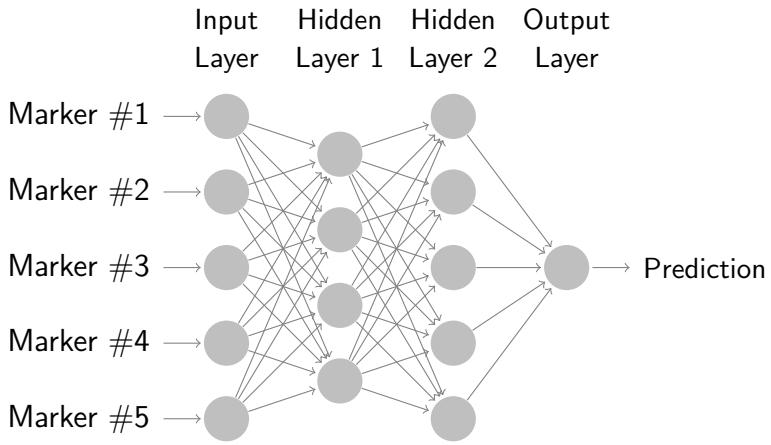

Figure 2.1: A deep feedforward neural network. Many input marker calls are mapped to one or more sequential hidden layers of neurons. For genomic prediction, the input layer often consists of one neuron per marker and the output consists of a single neuron which combines the information from the final hidden layer to predict a phenotype or BLUP. The presence of more than one hidden layer indicates that a network that may be more likely to learn higher order non-linear interactions, and can be called a deep network.

Once the network is defined, it must be exposed to input and desired output values, and adjusted to minimize error in output in a process known as training. The weights and biases

are often initialized by drawing from a random normal distribution. From this initial state, error in the output of the network is propagated back through the hidden layers, and the weights and biases are updated in the direction that would decrease output error on many randomly drawn subsets of the input data. This turns the network training process into a general function minimization problem, where the parameters to the function are the weights and biases of the network neurons and the function to be minimized is the squared differences between the network outputs and the desired true values. The process of propagating output error back through a neural network is known as backpropagation, and has been used and improved extensively since its original description in the 1980s (Rumelhart *et al.*, 1986). Typically, the training data is split evenly into representative, randomly sampled collections of data called batches. The training algorithm exposes the network to each batch until they are depleted, after which the process is repeated. Each collection of batches is known as an epoch, and training typically involves applying backpropagation for several hundred epochs, or until the network weights and biases reach an equilibrium state that has converged to a globally minimum amount of prediction error.

It is trivial to add additional hidden layers to an existing neural network model. Despite the ease of describing their architecture, networks with many hidden layers have been notoriously difficult to train. The amount of error attributed to a neuron by backpropagation decreases in magnitude with each additional layer added to the network. As a result, layers nearest the input train very slowly in a phenomenon known as the vanishing gradient problem (Hochreiter, 1998). Recently, a series of breakthroughs in neural network training along with well-known increases in computer processing speeds have allowed efficient training of deeper networks than were previously possible (Sutskever *et al.*, 2013). A history of the deep learning literature is available in LeCun *et al.* (2015). The increased training efficiency and potential to capture subtle correlation relationships between two or more input features drove a need to differentiate these deeper networks from prior work, resulting in the emer-

gence of the phrase "deep learning" to describe the construction and training of deep neural networks.

Previous attempts to apply neural networks to genomic prediction have resulted in an overfitting of the network to the training data and raised concerns about the computation time required to fit the model to datasets containing many markers across many genotypes (Heslot *et al.*, 2012; González-Recio *et al.*, 2014). In retrospect, these results are not surprising. Multi-layer feedforward neural networks are capable of approximating functions of arbitrary complexity to arbitrary accuracy if provided enough neurons in even a single hidden layer. This property of neural networks is known as the universal approximation theorem, and can result in overfitting if the weights of the network are not regularized in some way (Kur *et al.*, 1989).

Given the promising results from regularized and Bayesian methods for genomic prediction such as ridge or LASSO regression and the Bayesian family of regressors, it is prudent to evaluate some of the many of neural network training algorithms which incorporate regularization of weights during training. Today, networks based on these and other regularization techniques continue to show success across many domains (Schmidhuber, 2015). Similarly, while neural networks are computationally demanding to train, the training algorithms themselves are often easily expressed with vector and matrix algebra. These algorithms are well suited to execution on Graphics Processing Unit (GPU) hardware, with reports of up to a sixty-fold speedup in training time (Sierra-Canto *et al.*, 2010; Schmidhuber, 2015).

To facilitate the evaluation of regularized neural networks, a neural network prediction model was implemented supporting two regularization techniques. The first is a weight decay regularization which penalizes the weights W with very large values similar to ridge regression (Krogh and Hertz, 1992). The second is dropout regularization, in which a portion of neurons and their connections are removed at random during each training epoch, encouraging subsets of the network to learn to recognize input features independently. This allows neurons to adapt and build independently operating units, preventing the neurons from co-adapting

and reduces overfitting (Srivastava *et al.*, 2014). In order to support training deep networks, network weights were initialized in a way causes deep networks to rapidly converge to a global minimum error (Glorot and Bengio, 2010). A modified form of backpropagation designed for deep network training was also selected (Dozat, 2015). Finally, divergence detection and learning rate decay methods were implemented to ensure models converged to a global minimum error.

In this paper, we present the results of using this deep neural network configuration to perform genomic prediction on three benchmark datasets from different species. A high and low heritability trait was selected from each dataset for a total of six genomic prediction tasks. A collection of regression techniques including single hidden layer and deep neural networks were applied to each task. We also quantitatively evaluate the time required to train neural networks using both CPU and GPU based network fitting. Last, we compare the performance of the regularized network implementation with other available genomic prediction models in the collection.

Materials and Methods

Prediction Pipeline

Each set of paired phenotypic and genotypic measurements were divided into five partitions by drawing paired measurements without replacement. Each prediction model was trained on four partitions of the data and used to predict the remaining held-out partition. For statistical models with tunable hyperparameters, a grid-search of the parameter space was conducted and the model predicted out outcome will all sets of parameters in the grid. The prediction accuracy as well as any model parameters were recorded for the partition. The models were then reset and the process was repeated holding out a different partition. This produced five estimates of the prediction accuracy of each model. The datasets

were then shuffled and the pipeline was re-run to produce a total of ten prediction accuracy estimates for each model.

Neural Network Implementation and Training

All neural network models were initialized using Glorot weight initialization (Glorot and Bengio, 2010) and trained for 12,100 epochs of four batches each. Backpropagation was performed using the Nadam backpropagation algorithm with standard parameters (Dozat, 2015). The first 100 epochs were executed on separate networks, and the network with the lower in-sample prediction accuracy was allowed to complete the remaining 12,000 epochs, reducing the incidence of low accuracy due to poor initial network weights. The network learning rate was reduced by a factor of four after 10,100 epochs and reduced by a further factor of four after 11,600 epochs as a simple form of learning rate decay.

The full analysis pipeline, including formatted datasets, quality control processes, implementations of each model, and final hyperparameter settings in file databases are available in versioned files in a public repository on github.com (McDowell, 2016).

Statistical Analysis

Least squares, ridge, LASSO, and elastic net regression were applied as examples of linear regression with and without normalization penalties. A Bayesian ridge regression model was included as an example of a model with an enforced Bayesian prior distribution on coefficient values. These five regression techniques were available in the scikit-learn python package version 0.17.1 (Pedregosa *et al.*, 2011).

A single parameterized neural network with dropout and weight-decay parameters was built using the Keras modular neural network library (Chollet, 2015). The neural network results presented in this manuscript were trained and evaluated using the default Theano backend (Team *et al.*, 2016). Network training was conducted on a combination of a single NVIDIA GTX 680 graphics card with 4GB of RAM and a cluster of NVIDIA GRID K520

graphics cards with 4GB of RAM each. Both types GPUs used the CUDA 7.5 toolkit (Nickolls *et al.*, 2008). A network with no regularization, as well as with either one or both of the weight-decay and dropout regularization parameters supplied was trained on each dataset.

A summary of all regression methods is presented in Table 2.1.

Table 2.1: **Regression Methods**

Method	Abbreviation	Library
Ordinary Least Squares Regression	OLS	scikit-learn
Ridge Regression	RR	scikit-learn
LASSO Regression	LASSO	scikit-learn
Elastic Net Regression	EN	scikit-learn
Bayesian Ridge Regression	BRR	scikit-learn
Unregularized Neural Network	N	Keras (Theano)
Neural Network with Weight Decay	NWD	Keras (Theano)
Neural Network with Dropout	NDO	Keras (Theano)
Neural Network with Weight Decay and Dropout	NWDDO	Keras (Theano)

Each regression technique was trained and evaluated on each phenotypic trait in all ten (5*2) cross-validation partitions of each dataset. The correlation of the predicted and actual phenotypic values was taken for each held-out partition, and the average and standard deviation of the resulting correlation coefficients for each regression technique were reported.

Benchmark Datasets

Arabidopsis, maize, and wheat datasets were collected from the author's web pages or the supplementary information published with their respective papers (Loudet *et al.*, 2002; Crossa *et al.*, 2010; Thavamanikumar *et al.*, 2015). Species, authorship, marker, and sample information is summarized in Table 2.2.

Marker calls were scaled to the range $[0, 1]$ for all SNP information. If more than 20% of marker calls were missing for a sample, the sample was discarded. If fewer than 20% were missing, the average value for that marker was imputed for the missing values with one

Table 2.2: **Benchmark Datasets**

Author	Species	Trait	Markers[1]	Samples[2]	Dependent Variable
Loudet *et al.* 2002	Arabidopsis	Days to flowering w/ short day length	69	415	Measurement
		Low resource dry matter accumulation	69	415	
Crossa *et al.* 2010	Maize	Well-watered grain yield	1,135	264	Measurement
		Female flowering date	1,148	284	
Thavamanikumar *et al.* 2015	Wheat	Time to young microspore	797	324	BLUP
		Spike grain number	797	324	

[1] Markers with greater than 20% missing calls were filtered from analysis.
[2] Samples with missing phenotypic measurements or greater than 50% missing marker calls were filtered from analysis.

exception: if data were published with a marker imputation technique already applied, no further imputation was attempted.

Individuals without phenotypic measurements were discarded from further analysis. A combination of phenotypic measurements and deregressed breeding values were predicted, depending on which measure was provided by the authors.

In addition to their original publications, copies of each of the benchmark datasets as well as modified versions formatted for compatibility with the analyses presented in this paper are available on github (McDowell, 2016). The analysis framework is extensible and can be utilized to evaluate arbitrary datasets and predictive models with minimal coding requirements provided the statistical models are implemented in the python 2.7 programming language.

Results and Discussion

Overall Performance

The unregularized neural network architecture (N) was not a top performer for any of the six datasets. The other unregularized network, ordinary least squares (OLS) was the

worst performer on every dataset. Ridge regression, a model with only L2 normalization, performed best on one dataset. LASSO regression, a form of L1 normalization that favors traits with few markers associated with moderate effects was not best on any dataset. Elastic Net is a regression technique that incorporates L2 and L1 regularization together in a single model and was optimal on one dataset. Bayesian Ridge Regression incorporates L2 regularization, while also fitting regression coefficients to a Gaussian distribution. It did not perform optimally on any problems, but frequently performed above average. For the neural network family of regressors, at least one form of regularization outperformed the unregularized network for every dataset. Networks with both dropout and weight decay were able to achieve best-in-dataset performance on two prediction tasks. These results are summarized in Table 2.3.

Table 2.3: **Model Performance**

		Accuracy								
		OLS	RR	LASSO	EN	BRR	N	NWD	NDO	NWDDO
Species	Trait									
Arabidopsis	Dry Matter	0.36	0.40	0.40	<u>0.42</u>	0.39	0.38	0.35	0.39	0.40
	Flowering	0.80	0.82	0.83	0.82	0.82	0.84	0.83	0.83	<u>0.86</u>
Maize	Flowering	0.22	0.33	0.32	0.33	0.32	0.33	0.34	<u>0.35</u>	0.33
	Grain Yield	0.47	<u>0.59</u>	0.49	0.51	0.57	0.55	0.52	0.55	0.51
Wheat	Spike Grain Number	0.15	0.27	0.33	<u>0.36</u>	0.28	0.27	0.31	0.28	0.33
	Time Young Microspore	0.59	0.61	0.74	0.73	0.64	0.67	0.74	0.68	<u>0.76</u>

Accuracy of the best observed model performance on each dataset rounded to two decimal places. The most accurate model on each dataset is underlined for emphasis.

Contrary to expectation, there was no consistent relationship between the top performing models and the species or trait evaluated, however several models performed notably worse than the others on all prediction tasks. The two unregularized methods, OLS and N, as well as the L1 regularized LASSO were never top performers. The BRR model also was not a top performer, but performed notably worse when predicting highly heritable traits. These results support the hypothesis that regularization is required to achieve optimal per-

formance in neural network models and that the optimal model and regularization strategy is dependent on the specific species, trait, or population studied. Empirical validation of model accuracy is an effective way to select a model and regularization method for genomic prediction.

Dropout regularization tends to produce networks that have reduced co-adaptation of neurons (Srivastava *et al.*, 2014). For genomic prediction tasks, this means that haplotypes specific to only one or two individuals would often be included in training epochs with different sets of neurons enabled due to dropout regularization. Because networks require neurons to be enabled in order to backpropagate error and detect patterns, more common haplotypes with consistent effects that persist across many different dropout iterations would be frequently present when particular sets of neurons are enabled and detect the common haplotype effects instead. The detection of rare haplotypes that are only coincidentally and not causatively associated with trait performance is thus far less likely, making dropout regularization effective.

Weight decay is an application of L2 regularization to neural networks (Krogh and Hertz, 1992). Weight decay penalizes neurons with large weights values, allowing the backpropagation algorithm to reduce total network error by finding solutions with smaller weights across all inputs and interaction terms much like ridge regression. For genomic prediction tasks, this produces models that perform well on traits with many predictive marker calls with small effects.

For traits with a mixture of a few large-effect alleles and many small-effect alleles, some combination of L1 and L2 regularization may be optimal. The elastic net (EN) model incorporates both of these regularization techniques, and is the top performer on two of the six datasets presented.

While it is simple to reason about model selection for hypothetical genetic architectures, the architecture of a trait is rarely known prior to conducting a detailed analysis. To ensure that a highly accurate model is selected without knowledge of genetic architecture, we rec-

ommend evaluating a variety of models and regularization techniques using cross validation to identify a model that performs well for the specific trait(s) of interest to use for making predictions or performing genomic selection. This eliminates the need to understand trait architecture in order to select an appropriate model.

Several models with notably high performance were not available in standard packages for the python programming language at the time of this writing and were not included in this analysis. These included a family of Bayesian regression models known as the "Bayesian Alphabet" (Gianola *et al.*, 2009) and Reproducing Kernel Hilbert Spaces (RKHS) regression (Gianola *et al.*, 2006).

Regularized Network Performance

If a model is overfitted to training data, training data prediction accuracy is greater than the accuracy observed when making predictions on new datasets. Regularization methods generally penalize models for overfitting to data. Because all measurements in this study were collected using five-fold cross validation sampling, models that exhibited overfitting should exhibit lower accuracy.

The decreased performance of the unregularized networks on several datasets in Figure 2.2 suggests that overfitting is sometimes a problem with genomic prediction applications. These results demonstrate that evaluating one or more regularization methods is critical to improve predictive accuracy for genomic prediction.

Deep Network Performance

Contrary to expectation, neural networks with additional hidden layers did not perform significantly better than networks with a single layer (Figure 2.3). One possible explanation for this observation is that by the universal approximation theorem, sufficiently large single layer neural networks can approximate any function, including those that deeper networks can approximate. In this study, single layer networks up to and including those with 2187

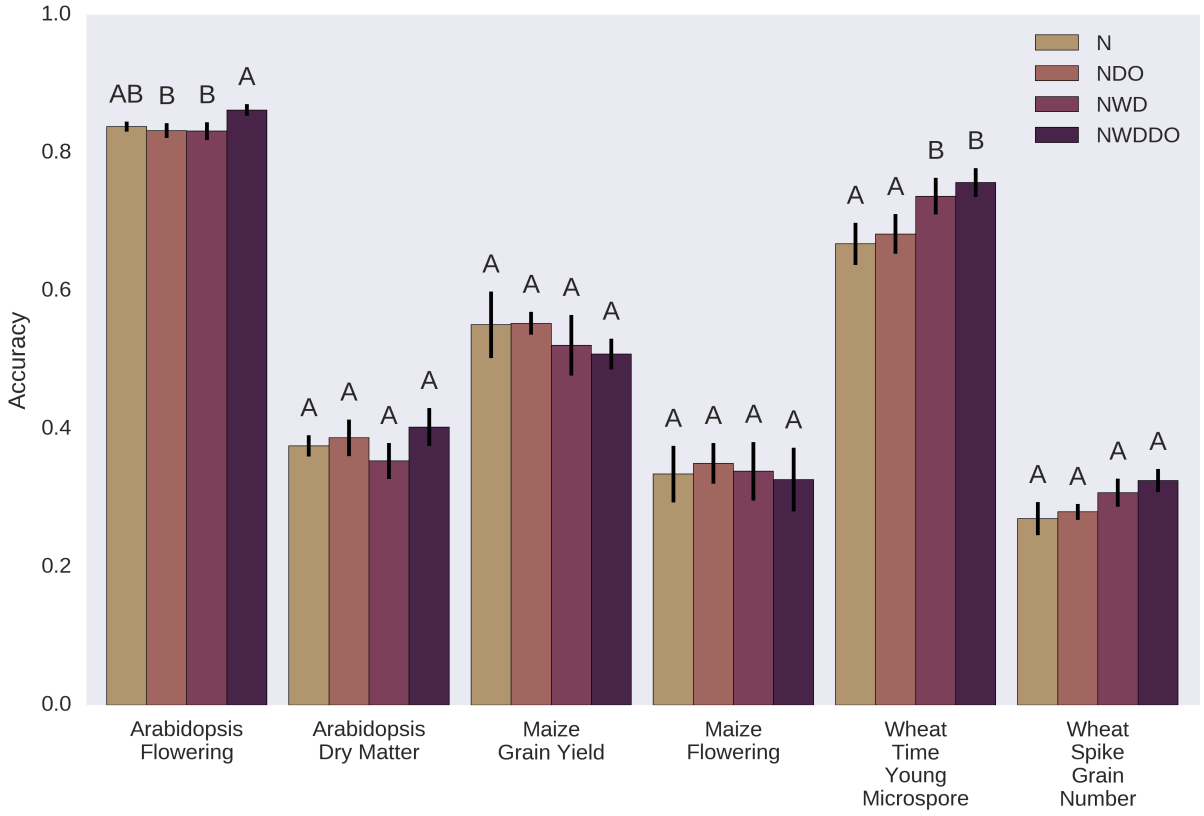

Figure 2.2: Predictive accuracy ($\mu \pm \sigma_{\bar{x}}$) of regularized and non-regularized neural network models for on benchmark datasets. Only the best performing network architecture for each species, trait, and model is included. The accuracy of the best performing model across all folds of data and training cycles were recorded and compared. All pairwise model comparisons within a species and trait were made using a two-sided paired t-test (n=10, paired by training cycle and fold number). The resulting p-values were corrected for multiple comparisons within each species and trait combination using the Holm-Bonferroni method. Columns annotated with the same letter are not significantly different at the $\alpha = 0.05$ level.

(3^7) neurons were evaluated on all datasets. This number is larger than the number of markers in any dataset evaluated and may be enough to encode complex interactions in only a single layer.

Networks with deep architectures have proven effective on a wide variety of problems, but the largest improvements have occurred on problems with high dimensional input data. These include voice recognition tasks where vocal frequencies change over a time dimension or two-dimensional images change over a time dimension such as in video playback. It is possible that the dimensionality of genomic prediction tasks tested are insufficient to benefit from the deep interaction effects that have proven effective in prediction tasks in other domains.

Yet another possibility is that phenotypic measurement error on many datasets is large enough to mask the subtle interaction effects that would otherwise be learned by the network.

GPU and CPU Training Time

Network GPU training time was significantly different from CPU training time in all but one genomic prediction task (Figure 2.4). For small networks trained on smaller datasets such as arabidopsis, CPU training completed significantly faster than GPU training, though only by a small magnitude. For small networks and all other datasets, as well as for all large networks, GPU training time was faster than CPU training time on a per-core basis. These results confirm that the previously observed speedups associated with GPU network training apply to genomic selection applications.

The time required to train a network for genomic prediction is related to the sample size, marker count, and complexity of the network architecture to be trained. The improvement in training time when using GPUs for model training was sub-linear across all three of these measures. This is because the compute capacity of modern GPUs is so large that the processing bottleneck is not the matrix algebra required to perform backpropagation but rather the speed of moving data to and from the memory on the GPU hardware. We

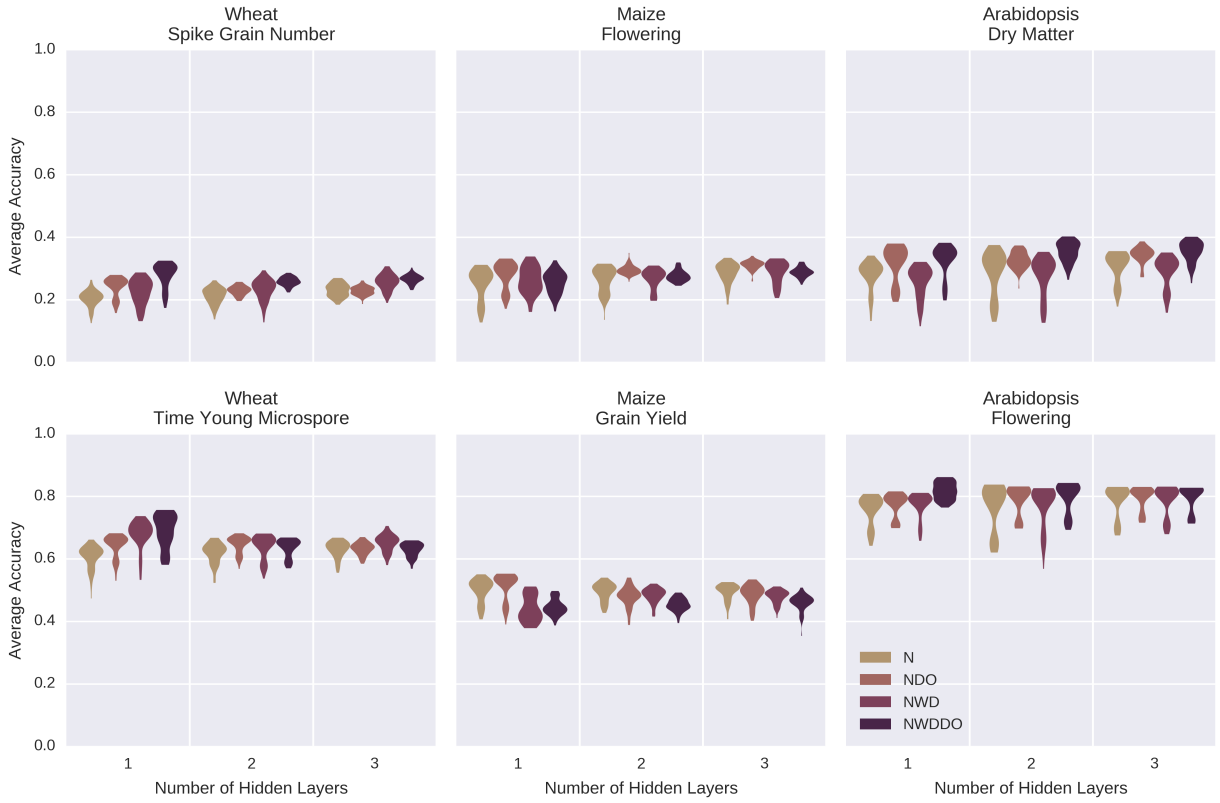

Figure 2.3: Distribution of predictive accuracy by benchmark dataset, network depth, and model. The violin plot width indicates the Kernel Density Estimate (KDE) of all observed accuracies of all models at a given network depth. The sample size of the KDE is 7, 28, 28, and 112 samples for the N, NWD, NDO, and NWDDO models, respectively. The models contributing to each KDE vary across one or more of five weight decay, five dropout, and seven hidden layer architecture parameters and can be understood as the distribution of results across the set of hyper-parameters to all network models with the same depth and regularization type. The KDE bandwidth parameters are set using Scott's normal reference rule. The KDE plots are truncated to the minimum and maximum observed prediction accuracies.

frequently observed CPU utilization of 100% during GPU training, indicating that the GPU was demanding resources at a rate higher than the computer system as a whole could provide them. Consumer grade GPUs such as the NVIDIA Titan X are now available with twice as many CUDA cores as the GTX 680 as well a 50% wider memory bus, making it potentially faster than the model used for this analysis.

It is possible to utilize additional CPU cores to linearly decrease the total time needed to train a network, however it is uncommon to utilize multiple GPUs for training the same network, and this is not supported by most software packages due to limitations in how memory is shared between components in systems with multiple GPUs. For example, a quad-core processor like the Intel i7-4790K CPU can train a single network approximately four times faster than indicated in Figure 2.4 if the entire CPU is assigned to a single network training task. In practice, networks are trained using k-fold cross validation, so the actual time to fit a family of networks and select an optimal architecture would be several times larger than these values. The growth in total time to train when using a CPU suggests that with networks larger than the 128x64 network in Figure 2.4, CPU training time could become prohibitively large even with quad-core processors.

Thus, choosing a network training method depends on the size of the dataset and network to be trained. Low complexity networks train rapidly on CPUs, while larger networks are more efficiently trained using GPUs. Cloud compute infrastructure providers such as Amazon Web Services (AWS) are becoming more popular, and make it much easier to evaluate training options without purchasing expensive computer hardware. AWS provides both CPU and GPU rich machines which can be rented at low cost and are charged by the hour. This makes it feasible to try CPU and GPU training on sample data and select a training platform that is most time or cost effective based on the complexity of the desired network architecture and the quantity of data available for training.

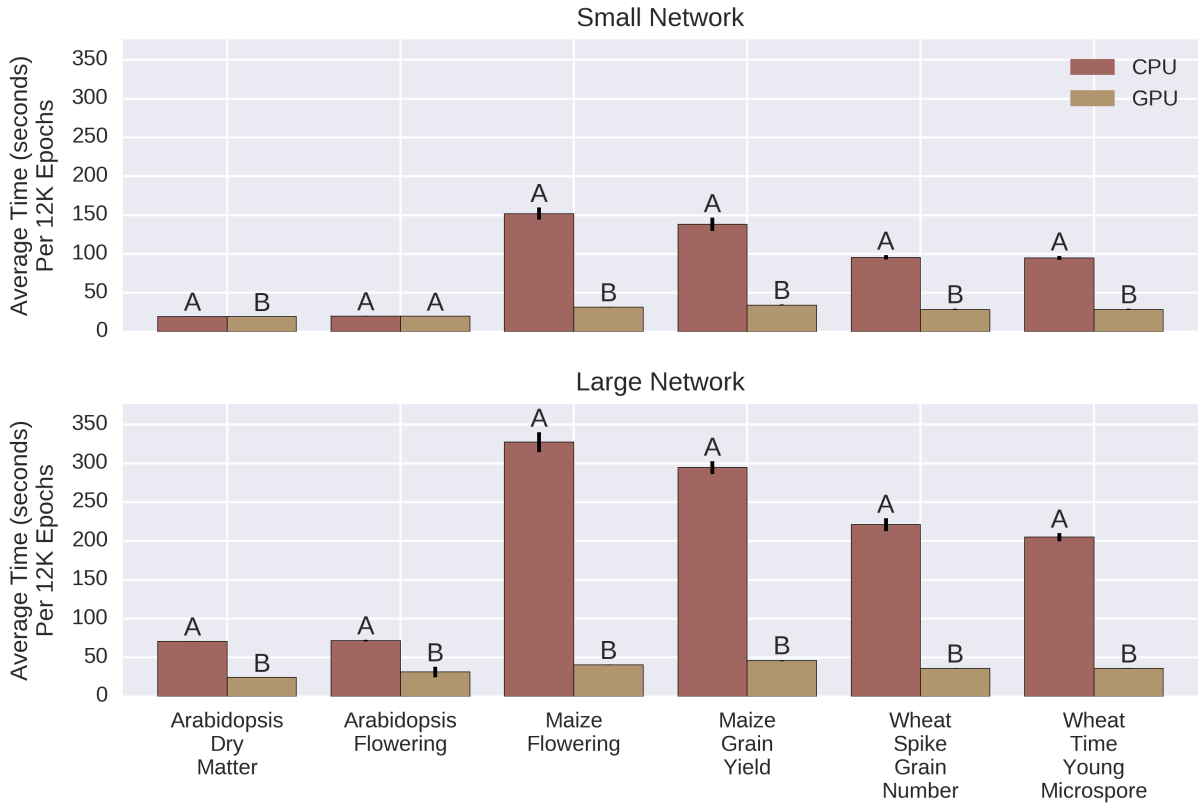

Figure 2.4: Time to train different sized networks on identical datasets using a single dedicated CPU core compared to a single dedicated GPU card. Lower time to train is better. For the small network, an unregularized, single hidden layer of 27 neurons was trained for 12K epochs on each dataset. For the large network, two consecutive hidden layers of size 128 and 64 neurons were trained. Training processes were otherwise equal. This process was repeated ten times per dataset to reduce variation associated with non-deterministic processor scheduling and varying computer system load. The ten CPU and GPU training samples were compared using an independent samples T-test with $n = 30$. CPU-GPU pairs annotated with the same letter are not significantly different at the $\alpha = 0.05$ level.

Overall Conclusion

Breeders are sometimes interested in predicting the additive genetic value of individuals rather than their expressed phenotype. Algorithms such as RKHS regression and neural networks often produce genomic prediction models with high predictive accuracy because they are able to capture interaction effects between genotypes. This is not possible with additive linear models such as ridge regression. As a result, caution must be exercised when interpreting improved prediction accuracy as a better model in the general case. Breeding scenarios where additive genetic value is of primary importance may not benefit from the use of models that capture interactions between genes because these predictions contain non-heritable additive variation. However, when phenotypic predictions are desirable, such as when breeding clonal crop varieties where individual segregates are selected to become commercial varieties, models incorporating interaction effects should perform better than models capturing only additive effects.

When combined with effective regularization techniques, neural networks can function as accurate and effective genomic prediction models with a low risk of overfitting to training data. With the ready availability of GPU computation resources, concerns over total time to train are partially ameliorated. Searches for optimal architectures or regularization techniques can be automated in a time and cost effective manner using cloud compute providers such as AWS.

CHAPTER 3
GENERAL CONCLUSIONS

General Discussion

There are four primary goals of the work presented in this thesis.

- Demonstrate whether neural network models can exhibit competitive accuracy on public datasets.
- Explicitly determine the relationship between model regularization and the genetic architecture of traits.
- Determine the effect of adding additional hidden layers to neural network models for genomic prediction.
- Explore the effect of GPU computing on neural network training time.

To achieve these goals, six public genomic datasets were collected and an experimental framework was created to test several genomic prediction models. A combination of CPU and GPU resources was used to fit nine statistical models including four neural network models to each of the six datasets while collecting information on training time as well as final prediction accuracy.

Contrary to expectation, there was no consistent relationship between the top performing models and the species or trait evaluated, however several models performed notably worse than the others on all prediction tasks (Table 2.3). The unregularized ordinary least squares and the unregularized neural network, as well as the L1 regularized LASSO were never top performers. The Bayesian ridge regression model also was not a top performer, but performed notably worse when predicting highly heritable traits. These results support the

hypothesis that regularization is required to achieve optimal performance in neural network models and that the optimal model and regularization strategy is dependent on the specific species, trait, or population studied. Empirical validation of model accuracy is an effective way to select a model and regularization method for genomic prediction.

Neural networks were top performers on three of the six datasets presented in this study. While this is a small sample size, it is sufficient to demonstrate that neural networks can function as competitive genomic prediction models. Published studies to date have failed to apply regularization to networks, resulting in an underestimation of their predictive ability. A large body of work outside the plant and animal breeding literature demonstrates that networks can rapidly solve complex problems when applied appropriately. Our results provide preliminary evidence that networks can be successfully applied to plant breeding projects, comparison of regularized networks to RKHS and Bayesian Alphabet methods will be required before drawing conclusions about prediction accuracy.

The regularization methods presented in this paper included standard methods such as L1 and L2, as well as Bayesian prior distributions and network dropout. Given the relationship between regularization, genomic architecture, and model performance, it is not surprising that the two unregularized prediction methods, ordinary least squares and a standard MLP neural network, were not top performers on any dataset (Table 2.3). The performance of the neural network models suggests that regularization is critical to achieve optimal performance. This is likely to be true for any problem which is predictive rather than descriptive in nature.

Adding additional hidden layers to the neural network models did not improve network performance on the six genomic prediction tasks presented in this study. This result was unexpected, as similar architectures have resulted in an outstanding gain in accuracy on both new and old prediction problems (Mnih *et al.*, 2013; Subasi and Ercelebi, 2005; Lang *et al.*, 1990). There are several possible explanations for these results. First, the universal approximation theorem predicts that a sufficiently large single layer network can approximate any function, including those that multi-layer networks can approximate. However, improved

predictive accuracy from deep networks on some categories of problems in other domains suggests that there are occasions where, at a minimum, deep networks train more effectively or reach training states that produce better predictive accuracy. It is not clear whether our results imply that genomic prediction is best performed with neural networks with a single hidden layer or whether other neural network types might benefit from additional hidden layers. A second possibility is that the dimensionality of the genomic prediction problems presented in the selected datasets was insufficient to take advantage of the flexible learning process of deep neural networks. Problems where deep networks have made the largest improvements generally involve data that includes an additional dimension such as one-dimensional time series or two-dimensional imagery data.

Aside from linkage, the markers used as input for genomic prediction tasks are mostly independent. It is possible that additional data density such as full sequencing data which has an additional dimensional structure encoded in the order of nucleotides along a strand of DNA, is required to observe the effects of neural networks. A third possibility is that the measurement error on the phenotypes examined in this study was sufficiently large to mask the subtle interaction effects between markers that a deep network could potentially learn. If so, this problem could be resolved by reducing measurement error or increasing sampling sizes in genomic prediction training datasets.

Previous studies have raised concerns about the computational complexity of fitting neural network models (Heslot *et al.*, 2012; González-Recio *et al.*, 2014). Our results show that training neural networks using GPU resources drastically reduces the rate at which training time grows with sample size, marker count, and network complexity (Figure 2.4). The improvement in training time we observed was so large that growth in total time to train was sub-linear across all three of these measures. This is because the compute capacity of modern GPUs is so large that the processing bottleneck is not the matrix algebra required perform backpropagation but rather moving data to and from the memory on the GPU hardware. Hardware manufacturers are aware of this limitation and continue to increase the

speed of the hardware buses that move data between computer hardware components. Our results demonstrate that concerns about computation complexity when fitting networks are no longer valid on small to medium sized datasets such as those presented in this paper. We anticipate that as hardware improves, even the largest of datasets will be able to move rapidly between hardware components, and the model fitting bottleneck will be capacity of the GPU to perform algebraic manipulations, much like CPU hardware is for most model fitting tasks today.

Future Research

Other Models

Future research should directly compare neural network based genomic prediction models with other state-of-the-art models. The RKHS regression family of models has performed well on several prediction tasks (Heslot *et al.*, 2012; Crossa *et al.*, 2010; González-Recio *et al.*, 2014; Gianola *et al.*, 2006). In order to properly evaluate these models, a flexible model fitting and evaluation model architecture such as the one built for this paper will be needed. The Bayesian Alphabet models have also performed notably well on a wide variety of problems (Heslot *et al.*, 2012; Crossa *et al.*, 2010; Thavamanikumar *et al.*, 2015). These also should be included in comparisons with regularized neural networks. Particular focus must be placed on measurable aspects of the genetic architecture of a trait that are correlated with the performance of a particular model. This will enable better modeling decisions during the development of a genomic selection program.

Other Regularizers

This study examined the effects of dropout and L2 normalization on neural networks. It did not consider alternative regularization techniques which exist for MLP neural networks. While LASSO regression utilizes L1 regularization and was evaluated, L1 regularization

on network weights was not evaluated to reduce the total number of network parameters evaluated in this study. Still other regularizers exist such as activity and bias regularization which penalize the total output of all neurons and the bias terms on each neuron, respectively. Both of these forms of regularization can be applied in an L1 and L2 sense. Another series of regularizers are more commonly known as constraints, which constrain network various subsets of network parameters within bounds, within norms, or to positive real numbers. Unlike many of the linear or Bayesian regression techniques evaluated in this and other studies, the family of fully-connected neural networks are amenable to dozens of different regularization techniques for thousands of different network architectures. This poses a combinatorics challenge when attempting to select a single optimal network architecture. However, this also creates opportunities to identify networks that best model the genetic architecture and thus achieve the highest predictive accuracy on a given genomic prediction task.

Additional Input Features

Neural networks are robust to input feature transformations. It is typical to transform inputs to a real number in the range $[0, 1]$ prior to using them as input for a neural network. Molecular marker arrays typically output values that can be linearly scaled within this range, but other values can be easily included in addition to marker calls. Several methods could be used to encode non-marker information to improve predictive accuracy. These include location or population information that can be encoded using binary features to indicate membership in a particular population or that an evaluation was conducted in an environment where a pest was present for example. Non-binary inputs such as weather information on a daily or weekly interval could be included and allowed to interact with marker calls to improve phenotypic estimates. Sophisticated models such as these could be reverse-engineered to estimate optimal genetic combinations for performance in particular environments. Like other regression models, the more features that are included in

the model, the more regularization is needed to prevent overfitting and produce predictive outputs. Many other input features can be conceived to augment the information leveraged by a network when making a prediction. While neural networks pose the aforementioned combinatoric architecture challenge, they are unique in accepting large numbers of inputs from diverse sources of data. Practically any information can be transformed to a $[0, 1]$ scale and leveraged by a network. In this way, networks can form so-called expert systems that perform in ways that are similar to that of a human expert.

Larger Datasets

This study examined both small and medium density marker datasets. Datasets with many thousands of marker calls were not included due to time and budgetary constraints. Future genomic prediction research should include larger datasets with $p \gg n$. Several examples of these datasets are available in the literature (Resende *et al.*, 2012; Cleveland *et al.*, 2012).

Radial Basis Function Networks

This work has focused on deep MLP neural networks with sigmoid activation functions. Architectures such as the Radial Basis Function (RBF) network do not possess the multiple hidden layers of MLP networks. This alternative architecture is not well studied in the machine learning literature, but has shown promising performance on genomic selection problems (Gonzalez-Camacho *et al.*, 2012).

Final Remarks

Additional research is needed to determine the performance of neural networks relative to other state-of-the-art models. This process can be accelerated by leveraging GPU computation to more quickly evaluate large numbers of complex networks. As transistor density continues to increase (Moore, 1965), and GPU computational capacity subsequently in-

creases, a network's computational demands will decrease thus making them more attractive for use in genomic prediction tasks as well as in other domains.

LITERATURE CITED

Breiman, L., 2001 Statistical modeling: The two cultures (with comments and a rejoinder by the author). Stat. Sci. **16**: 199–231.

Chollet, F., 2015 Keras. https://github.com/fchollet/keras.

Cleveland, M. A., J. M. Hickey, and S. Forni, 2012 A common dataset for genomic analysis of livestock populations. G3 **2**: 429–435.

Crossa, J., G. de los Campos, P. Pérez, D. Gianola, J. Burgueño, *et al.*, 2010 Prediction of genetic values of quantitative traits in plant breeding using pedigree and molecular markers. Genetics **186**: 713–724.

Daetwyler, H. D., B. Villanueva, P. Bijma, and J. A. Woolliams, 2007 Inbreeding in genome-wide selection. J. Anim. Breed. Genet. **124**: 369–376.

de Koning, D.-J. and L. McIntyre, 2012 Setting the standard: A special focus on genomic selection in GENETICS and G3. Genetics **190**: 1151–1152.

Donoho, D., 2015 50 years of data science. Tukey Centennial Workshop.

Dozat, T., 2015 Incorporating Nesterov momentum into Adam. Technical report, Standford University.

Evenson, R. E. and D. Gollin, 2003 Assessing the impact of the green revolution, 1960 to 2000. Science **300**: 758–762.

Fehr, W. R., 1987 *Principles of Cultivar Development*, volume 1. Macmillian Publishing Co., Reprinted 1993.

Fernandez-Cornejo, J., S. Wechsler, *et al.*, 2012 Revisiting the impact of Bt corn adoption by US farmers. J. Agr. Resour. Econ. **41**: 377.

Gianola, D., G. de los Campos, W. G. Hill, E. Manfredi, and F. Rohan, 2009 Additive genetic variability and the Bayesian Alphabet. Genetics **183**: 347–363.

Gianola, D., R. L. Fernando, and A. Stella, 2006 Genomic-assisted prediction of genetic value with semiparametric procedures. Genetics **173**: 1761–1776.

Glorot, X. and Y. Bengio, 2010 Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9, pp. 249–256.

Gonzalez-Camacho, J. M., G. de los Campos, P. Pérez, D. Gianola, J. E. Cairns, *et al.*, 2012 Genome-enabled prediction of genetic values using radial basis function neural networks. Theor. Appl. Genet. **125**: 759–771.

González-Recio, O., G. J. M. Rosa, and D. Gianola, 2014 Machine learning methods and predictive ability metrics for genome-wide prediction of complex traits. Livest. Sci. **166**: 217–231.

Heffner, E. L., A. J. Lorenz, J.-L. Jannink, and M. E. Sorrells, 2010 Plant breeding with genomic selection: Gain per unit time and cost. Crop Sci **50**: 1681–1690.

Heffner, E. L., M. E. Sorrells, and J.-L. Jannink, 2009 Genomic selection for crop improvement. Crop Sci. **49**: 1–12.

Heslot, N., H.-P. Yang, M. E. Sorrells, and J.-L. Jannink, 2012 Genomic selection in plant breeding: A comparison of models. Crop Sci. **56**: 146–160.

Hiremath, P. J., A. Kumar, R. V. Penmetsa, A. Farmer, J. A. Schlueter, *et al.*, 2012 Large-scale development of cost-effective SNP marker assays for diversity assessment and genetic mapping in chickpea and comparative mapping in legumes. Plant Biotech J. **10**: 716–732.

Hochreiter, S., 1998 The vanishing gradient problem during learning recurrent neural nets and problem solutions. IJUFKS **6**: 107–116.

Krogh, A. and J. A. Hertz, 1992 A simple weight decay can improve generalization. NIPS **4**: 950–957.

Kur, H., M. Stinchombe, and H. White, 1989 Multilayer feedforward networks are universal approximators. Neural Networks **2**: 359–366.

Lande, R. and R. Thompson, 1990 Efficiency of marker-assisted selection in the improvement of quantitative traits. Genetics **124**: 743–756.

Lang, K. J., A. H. Waibel, and G. E. Hinton, 1990 A time-delay neural network architecture for isolated word recognition. Neural Networks **3**: 23–43.

LeCun, Y., Y. Bengio, and G. Hinton, 2015 Deep learning. Nature **521**: 436–444.

Loudet, O., S. Chaillou, C. Camilleri, D. Bouchez, and F. Daniel-Vedele, 2002 Bay-0 x Shahdara recombinant inbred line population: A powerful tool for the genetic dissection of complex traits in Arabidopsis. Theor. Appl. Genet. **104**: 1173–1184.

McDowell, R., 2016 Genomic neuralnet. https://github.com/rileymcdowell/genomic_neuralnet.

Meuwissen, T. H. E., B. J. Hayes, and M. E. Goddard, 2001 Prediction of total genetic value using genome-wide dense marker maps. Genetics **157**: 1819–1829.

Miles, C. and M. Wayne, 2008 Quantitative trait locus (QTL) analysis. Nature Education **1**: 208.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, *et al.*, 2013 Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 .

Moore, G. E., 1965 Cramming more components onto integrated circuits **38**.

Nickolls, J., I. Buck, M. Garland, and K. Skadron, 2008 Scalable parallel programming with CUDA. Queue **6**: 40–53.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, *et al.*, 2011 Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. **12**: 2825–2830.

Resende, M. F. R., P. Muñoz, M. D. V. Resende, D. J. Garrick, R. L. Fernando, *et al.*, 2012 Accuracy of genomic selection methods in a standard data set of loblolly pine (*Pinus taeda, L.*). Genetics **190**: 1503–1510.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams, 1986 Learning representations by back-propagating errors. Nature **323**: 533–536.

Schmidhuber, J., 2015 Deep learning in neural networks: An overview. Neural Networks **61**: 85–117.

Sierra-Canto, X., F. Madera-Ramirez, and V. Uc-Cetina, 2010 Parallel training of a back-propagation neural network using CUDA. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pp. 307–312, IEEE.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 2014 Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**: 1929–1958.

Subasi, A. and E. Ercelebi, 2005 Classification of EEG signals using neural network and logistic regression. Comput. Meth. Prog. Bio. **78**: 87–99.

Sutskever, I., J. Martens, G. Dahl, and G. Hinton, 2013 On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pp. 1139–1147.

Team, T. T. D., R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, *et al.*, 2016 Theano: A Python framework for fast computation of mathematical expressions. arXiv preprint arXiv:1605.02688 .

Thavamanikumar, S., R. Dolferus, and B. R. Thumma, 2015 Comparison of genomic selection models to predict flowering time and spike grain number in two hexaploid wheat doubled haploid populations. G3 **5**: 1991–1998.

Tibshirani, R., 1996 Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Series B Stat. Methodol. pp. 267–288.

Zou, H. and T. Hastie, 2005 Regularization and variable selection via the elastic net. J. R. Stat. Soc. Series B Stat. Methodol. **67**: 301–320.

APPENDIX A. RAW DATA

Raw data used in this study was collected from the following publications.

- Arabidopsis - Loudet *et al.* (2002)
- Maize - Crossa *et al.* (2010)
- Wheat - Thavamanikumar *et al.* (2015)

The raw data was collected and processed into a consistent format using a series of scripts created to automate data extraction for the analysis presented in this thesis. The raw data, scripts, and their output as used in all analysis in this thesis is available at https://github.com/rileymcdowell/genomic-neuralnet/tree/master/genomic_neuralnet/data.

APPENDIX B. ANALYSIS CODE

The source code used to generate the analysis in this paper is available at https://github.com/rileymcdowell/genomic-neuralnet. The `genomic_neuralnet` directory contains the source code itself. The `genomic_neuralnet/analysis/optimize_all.sh` script fits all available models to all datasets when executed. Command line parameters are available for all python source code when executed with the `--help` command line argument. All genomic prediction models presented in this paper are implemented in the `genomic_neuralnet/methods/` directory.

To facilitate fitting thousands of model variants on each dataset, Amazon Web Services (AWS) resources were utilized for pay by the hour compute capacity. A small compute cluster was created containing a single head cluster node and a variable number of slave nodes depending on the current market price of AWS Elastic Compute Cluster (EC2) instance spot instances at that time. The head cluster node was initialized with the `cloud-init-master.sh` script in the root source code directory. The slave nodes were initialized using either the `cloud-init-slave.sh` or `cloud-init-slave-gpu.sh` scripts depending on the absence or presence of a GPU on the node, respectively. All nodes were running Amazon Linux AMI 2016.03.3 as the base operating system.

APPENDIX C. THESIS CODE

This thesis was generated using the LaTeX document preparation language. All source code files used to generate this paper are available at `https://github.com/rileymcdowell/genomic-neuralnet-paper`. Generating this thesis requires a complete working `texlive` installation. The `genomic-neuralnet` library described in Appendix B. must also be available on the system for the figures and tables to be correctly copied into this thesis.