Deep learning approaches for yield prediction and crop disease recognition

by

Luning Bi

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Industrial Engineering

Program of Study Committee:
Guiping Hu, Major Professor
Ali Jannesari
Qing Li
Daren S Mueller
Hantang Qin

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2022

DEDICATION

I would like to dedicate this dissertation to my wife, Zhuqing Liu, without whose support I would not have been able to complete this work.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

ACKNOWLEDGMENTS

I would like to express my thanks to those who helped me with various aspects of conducting research and writing this dissertation. First and foremost, thanks to Dr. Guiping Hu for her guidance, patience and support throughout this research and editing this dissertation. Her insights and words of encouragement have inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Ali Jannesari, Dr. Qing Li Dr. Daren S Mueller and Dr. Hantang Qin. Many thanks to the faculty and staff from the IMSE Department and my friends for making my time at Iowa State University a wonderful experience. Lastly, I would like to thank Dr. Fei Tao and Dr. Ying Zuo for their teachings and guidance during my undergraduate study.

ABSTRACT

The increase of the world population has brought significant challenges to the agriculture production system. Although mechanization has been realized in agriculture, many tasks (e.g., breeding, field inspection) are still labor-intensive and time-consuming. Therefore an automatic and intelligent solution is needed for the advancement of agricultural production. During this process, the biggest challenge is how to teach computers to understand the concepts in the real world. For example, an experienced expert can easily determine whether a plant is diseased or healthy. However, this may be challenging for the computer. Thus, the motivation of this dissertation study is to tackle these challenges in precision agriculture. This dissertation consists of four papers that propose different deep learning methods for the most challenging problems in agriculture.

In the first paper, a genetic algorithm (GA)-assisted deep neural network was built for yield prediction using genetic information and environmental factors. In the global search phase, the GA was introduced to help determine the best initial weights of the neural network. In the local phase, random perturbation was used to avoid the local optimum. By using the proposed method, the root mean square error can be reduced by up to 10%.

In the second paper, we proposed a generative adversarial network (GAN)-based approach to generate additional images for the classification of plant species and diseases using limited data. CNN was used as the basic network to classify species and diseases. GAN and label smoothing regularization (LSR) were combined to generate additional training images. Regular data augmentation techniques were also used to expand the dataset. The results showed that compared with using the real dataset only, the proposed method can improve the prediction accuracy by 6%.

In the third paper, the potential of using satellite imagery for plant disease detection was explored. A gated recurrent units (GRU)-based model was presented for early detection of

soybean sudden death syndrome (SDS) through time-series satellite imagery. The results showed that, compared to XGBoost and fully connected deep neural network (FCDNN), the GRU-based can improve the overall prediction accuracy by 7%. In addition, the proposed method can also be adapted to predict the future development of SDS.

In the fourth paper, a transformer-based approach was proposed for soybean yield prediction using time-series camera images and seed treatments information. First, a vision transformer (ViT) base model was designed to extract features from the images. Then another transformer-based model was established to predict the yield using the time-series features. A case study was been conducted using a data set that was collected during the 2020 soybean-growing seasons in Canada. The experiment results show that compared to non-time series prediction and other baseline models, the proposed approach can reduce the mean squared error by 25%-40%.

In conclusion, this dissertation aims to apply different state-of-art deep learning methods in agriculture. The study covers different topics, which range from yield prediction, species classification, to plant disease classification and prediction. At the model level, the application of linear models, tree-based methods, fully connected neural networks, convolutional neural networks, time-series models and transformers to different tasks have been investigated. In terms of the learning type, both unsupervised learning and supervised learning have been utilized. The experimental results have shown that appropriate deep learning methods can achieve better performance than traditional methods on specific tasks. Based on our work, more applications of deep learning techniques can be developed in the future.

CHAPTER 1. GENERAL INTRODUCTION

In the past decade, deep learning has become increasingly popular across many fields of study, such as speech recognition, face recognition, and self-driving cars. Under this context, this dissertation aims to explore the potential applications of deep learning to various problems in agriculture.

1.1 Challenges in Agriculture

The world population is expected to grow from 7.2 billion to 9.6 billion in 2100. This imposes rising demand in agriculture production. To alleviate this challenge, using different techniques to manage crop efficiently is necessary. Deep learning methods, which can help the computer to build complex concepts, becomes a potential solution. This dissertation aims to apply deep learning methods in agriculture settings. In this dissertation, three types of problems in agriculture, i.e., yield estimation, species classification and plant disease recognition, are considered.

- Yield Prediction. The world population continues to increase which imposes rising demand in agriculture production. How to improve crop breeding to feed the growing population is a significant challenge. Predictive modeling on crop phenotype can speed up the process and make it resource efficient. There are two common ways for yield prediction. One is to predict the crop yield using genetic (G) information and environment (E) information, which is also known as G by E prediction. The biggest challenge of G by E prediction is that the interactions among genes and environment factors are too complicated to formulate. Another way is to monitor the grow process of the crop through the time-series images of leaves. However, the images, which consist of thousands of pixels, have a high requirement for the efficiency and effectiveness of the model.

- Species Recognition. Automatic species recognition can help reduce the classification time and human factors. There are two types of species recognition. One type of study is to classify different crop plants. Grinblat et al. proposed a deep convolutional neural network for plant identification using vein morphological patterns [1]. He pointed out that it was not necessary to build a feature extraction method for this task. Wu et al. proposed a approach based on artificial neural network for the automated leaf recognition.The prediction accuracy of classifying 32 kinds of plants was greater than 90% [2]. Another study was the classification between crop plants and weed. Pantazi et al. used a variant of artificial neural network (ANN) to identity the weed in a field using unmanned aircraft system (UAS) multispectral imagery [3]. Ahmed et al. used texture features and SVM to classify the weed image, which achieved 98.5% prediction accuracy [4].

- Plant Disease Recognition. Plant diseases can have a significant impact on production and profits [5]. Early detection and timely management of plant diseases are essential to reducing yield loss. Traditional manual inspection is often time-consuming, laborious and biased. Some widely used methods include thermography [6, 7, 8], fluorescence measurements [9, 10, 11] and hyperspectral techniques [12, 13, 14]. Recently, automated imaging techniques have been successfully applied to the detection of plant diseases. Sensing instruments can record radiation in various parts of the electromagnetic spectrum, ultraviolet, visible, near-infrared (NIR) and thermal infrared, to name a few [15]. Healthy and diseased plant canopies absorb and reflect incident sunlight differently due to changes in leaf and canopy morphology and chemical constituents [16, 17]. These changes can alter the optical spectra, such as a decrease in canopy reflectance in the near-infrared band and an increase of reflectance in the red band [16].

1.2 Deep Learning in Agriculture

There are two main reasons behind the increasing popularity and applications of deep learning in various scientific disciplines. The first is the increase of computing resources. The

development of GPU accelerates the computing of matrix calculation and further analysis. The second is the development of data storing and management techniques makes it possible to collect, store, and manage data of increasing sizes. In this dissertation, deep learning is used for three types of tasks, i.e., multivariate forecasting, image classification, and time-series prediction.

- Multi-variable forecasting. Multi-variable forecasting is a useful tool to model the relationship between the input (e.g., seed type, temperature, sunlight) and the output (e.g., yield, healthiness). Many of the existing studies are based on linear models [18]. For example, Singh et al. predicted the crop yield by using piecewise linear regression model. The predicted values were very close to the observed values [19]. However, for some problems which have a large number of input variables as well as some complicated interactions among these variables, more high-level should be created manually for the linear regression models, resulting in the decay of model performance. For example, in G by E prediction, there are thousands of genes. Most interactions among genes are unknown. The common way to find the interactions are combine two or more genes as one input variable. The factorial combinations will add a lot bias to the regression models. Deep neural networks, which introduce the non-linear properties and high-level features to the model, provide another option. Ramos et al. constructed a machine vision system (MVS) to count the number of fruits on a coffee branch, which showed a correlation higher than 0.9 at early states of crop development [20] . Kaul et al. used ANN models to predict Maryland corn and soybean yield. The experiments showed that the prediction accuracy of artificial neural networks (ANNs) is higher than that of regression models [21].

- Image classification. Image classification based on leaf images is of vital importance in determining the plant status. The difference between image classification and multi-variable forecasting is that a large part of image information are redundant. Therefore, how to extract useful information from the image and improve the processing speed is the standard for the model. Convolutional neural network (CNN) is one of the most popular methods. Different from full connected deep neural network (FCDNN), CNNs have two special types

of layer. The convolutional layers extract features from the input images. The pooling layers reduce the dimensionality of the features. Dhakate et al. used a CNN for the recognition of pomegranate plant diseases and achieved 90% overall accuracy [22]. Ferentinos developed CNN models to classify the healthy and diseased plants. The success rate reached 99.53% [23].

- Time-series prediction. Since the grow process of crop usually last several months, the information collected during different stages could be helpful for the prediction of final status. To solve the multivariate time-series prediction problem, Zhang et al. proposed a recurrent neural network (RNN) [24]. However, RNN is faced with gradient vanishing/exploding problems [25]. As an improved version of RNN, long short term memory model (LSTM) is used for its successful application to natural language modeling [26]. Compared with RNN, LSTM has more gates that can control the reset of the memory and the update of the hidden states. Turkoglu et al. [27] proposed an LSTM-based CNN for the detection of apple diseases and pests, which scored 96.1%. Namin et al. [28] utilized a CNN-LSTM framework for plant classification of various genotypes as well as the prediction of plant growth and achieved an accuracy of 93%. Although LSTM has alleviated the gradient vanishing/exploding problem of RNNs, the training speed of LSTM is much slower due to the increased number of parameters. To solve this issue, Chung, et al. [29] introduced the gated recurrent unit (GRU) in 2014. Since GRU only has two gates (i.e., reset gate and update gate) and uses the hidden state to transfer information, its training speed is much faster. Jin et al. [30] used a deep neural network which combined CNN and GRU to classify wheat hyperspectral pixels and obtained an accuracy of 0.743.

In this dissertation, different types of deep learning models are proposed to solve the problems in agriculture.

1.3 Dissertation

This dissertation includes four papers. The first paper proposes a deep neural network based method to predict crop yields using genetic and environment information. The second paper presents an approach which combines convolutional neural network and general adversarial network (GAN) to classify crop species and diseases simultaneously. In the third paper, gated recurrent units (GRU) is used for soybean sudden death syndrome prediction with time-series satellite imagery. The last paper introduces a transformer-based approach for yield estimation using time-series images and seed treatment information.

The organization of this dissertation is as follows. Chapter 1 begins with a general introduction of the application methods in agriculture. The motivation and the contributions of this dissertation are also elaborated. Chapter 2 is an article published in the Soft Computing. Chapter 3 is an article published in Frontiers in Plant Science. Chapter 4 is an article published in Remote Sensing. Chapter 5 is an article to be submitted to Frontiers in Plant Science. Chapter 6 concludes with the results and future work.

References

[1] Guillermo L Grinblat, Lucas C Uzal, Mónica G Larese, and Pablo M Granitto. Deep learning for plant identification using vein morphological patterns. *Computers and Electronics in Agriculture*, 127:418–424, 2016.

[2] Stephen Gang Wu, Forrest Sheng Bao, Eric You Xu, Yu-Xuan Wang, Yi-Fan Chang, and Qiao-Liang Xiang. A leaf recognition algorithm for plant classification using probabilistic neural network. In *2007 IEEE international symposium on signal processing and information technology*, pages 11–16. IEEE, 2007.

[3] Xanthoula Eirini Pantazi, Alexandra A Tamouridou, TK Alexandridis, Anastasia L Lagopodi, Javid Kashefi, and Dimitrios Moshou. Evaluation of hierarchical self-organising maps for weed mapping using uas multispectral imagery. *Computers and Electronics in Agriculture*, 139:224–230, 2017.

[4] Faisal Ahmed, Md Hasanul Kabir, Shayla Bhuyan, Hossain Bari, and Emam Hossain. Automated weed classification with local pattern-based texture descriptors. *Int. Arab J. Inf. Technol.*, 11(1):87–94, 2014.

[5] Anne-Katrin Mahlein, Erich-Christian Oerke, Ulrike Steiner, and Heinz-Wilhelm Dehne. Recent advances in sensing plant diseases for precision crop protection. *European Journal of Plant Pathology*, 133(1):197–209, 2012.

[6] Laury Chaerle and Dominique Van Der Straeten. Imaging techniques and the early detection of plant stress. *Trends in plant science*, 5(11):495–501, 2000.

[7] N Mastrodimos, D Lentzou, Ch Templalexis, DI Tsitsigiannis, and G Xanthopoulos. Development of thermography methodology for early diagnosis of fungal infection in table grapes: The case of aspergillus carbonarius. *Computers and Electronics in Agriculture*, 165:104972, 2019.

[8] Yuxuan Wang, Shamaila Zia-Khan, Sebastian Owusu-Adu, Thomas Miedaner, and Joachim Müller. Early detection of zymoseptoria tritici in winter wheat by infrared thermography. *Agriculture*, 9(7):139, 2019.

[9] Kathrin Bürling, Mauricio Hunsche, and Georg Noga. Use of blue–green and chlorophyll fluorescence measurements for differentiation between nitrogen deficiency and pathogen infection in winter wheat. *Journal of plant physiology*, 168(14):1641–1648, 2011.

[10] Anne-Katrin Mahlein, Elias Alisaac, Ali Al Masri, Jan Behmann, Heinz-Wilhelm Dehne, and Erich-Christian Oerke. Comparison and combination of thermal, fluorescence, and hyperspectral imaging for monitoring fusarium head blight of wheat on spikelet scale. *Sensors*, 19(10):2281, 2019.

[11] María Luisa Pérez-Bueno, Mónica Pineda, and Matilde Barón. Phenotyping plant responses to biotic stress by chlorophyll fluorescence imaging. *Frontiers in plant science*, 10:1135, 2019.

[12] Koushik Nagasubramanian, Sarah Jones, Soumik Sarkar, Asheesh K Singh, Arti Singh, and Baskar Ganapathysubramanian. Hyperspectral band selection using genetic algorithm and support vector machines for early identification of charcoal rot disease in soybean stems. *Plant methods*, 14(1):1–13, 2018.

[13] Koushik Nagasubramanian, Sarah Jones, Asheesh K Singh, Soumik Sarkar, Arti Singh, and Baskar Ganapathysubramanian. Plant disease identification using explainable 3d deep learning on hyperspectral images. *Plant methods*, 15(1):1–10, 2019.

[14] Alina Förster, Jens Behley, Jan Behmann, and Ribana Roscher. Hyperspectral plant disease forecasting using generative adversarial networks. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 1793–1796. IEEE, 2019.

[15] H Nilsson. Remote sensing and image analysis in plant pathology. *Annual review of phytopathology*, 33(1):489–528, 1995.

[16] Reyer Zwiggelaar. A review of spectral properties of plants and their potential use for crop/weed discrimination in row-crops. *Crop protection*, 17(3):189–206, 1998.

[17] Stephane Jacquemoud and Susan L Ustin. Leaf optical properties: A state of the art. In *8th International Symposium of Physical Measurements & Signatures in Remote Sensing*, pages 223–332. CNES Aussois France, 2001.

[18] David L Des Marais, Kyle M Hernandez, and Thomas E Juenger. Genotype-by-environment interaction and plasticity: exploring genomic responses of plants to the abiotic environment. *Annual Review of Ecology, Evolution, and Systematics*, 44:5–29, 2013.

[19] Ramesh P Singh, Anup Krishna Prasad, Vinod Tare, and Menas Kafatos. Crop yield prediction using piecewise linear regression with a break point and weather and agricultural parameters, April 20 2010. US Patent 7,702,597.

[20] PJ Ramos, Flavio Augusto Prieto, EC Montoya, and Carlos Eugenio Oliveros. Automatic fruit count on coffee branches using computer vision. *Computers and Electronics in Agriculture*, 137:9–22, 2017.

[21] Monisha Kaul, Robert L Hill, and Charles Walthall. Artificial neural networks for corn and soybean yield prediction. *Agricultural Systems*, 85(1):1–18, 2005.

[22] Mrunmayee Dhakate and AB Ingole. Diagnosis of pomegranate plant diseases using neural network. In *2015 fifth national conference on computer vision, pattern recognition, image processing and graphics (NCVPRIPG)*, pages 1–4. IEEE, 2015.

[23] Konstantinos P Ferentinos. Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145:311–318, 2018.

[24] Yunong Zhang, Danchi Jiang, and Jun Wang. A recurrent neural network for solving sylvester equation with time-varying coefficients. *IEEE Transactions on Neural Networks*, 13(5):1053–1063, 2002.

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[26] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[27] Muammer Turkoglu, Davut Hanbay, and Abdulkadir Sengur. Multi-model lstm-based convolutional neural networks for detection of apple diseases and pests. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–11, 2019.

[28] Sarah Taghavi Namin, Mohammad Esmaeilzadeh, Mohammad Najafi, Tim B Brown, and Justin O Borevitz. Deep phenotyping: deep learning for temporal phenotype/genotype classification. *Plant methods*, 14(1):1–14, 2018.

[29] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[30] Xiu Jin, Lu Jie, Shuai Wang, Hai Jun Qi, and Shao Wen Li. Classifying wheat hyperspectral pixels of healthy heads and fusarium head blight disease using a deep neural network in the wild field. *Remote Sensing*, 10(3):395, 2018.

CHAPTER 2. A GENETIC ALGORITHM ASSISTED DEEP LEARNING APPROACH FOR CROP YIELD PREDICTION

Luning Bi and Guiping Hu

Department of Industrial and Manufacturing Systems Engineering, Iowa State University

Modified from a manuscript published in *Soft Computing*

Abstract

The world population continues to increase which imposes rising demand in agriculture production. How to improve crop breeding to feed the growing population is a significant challenge. The traditional crop breeding is resource-intensive and time-consuming. Predictive modeling on crop yield can speed up the breeding process and make it resource-efficient. In this paper, a genetic algorithm (GA)-assisted deep learning solution method is proposed for the crop yield prediction. The proposed method consists of two phases, i.e., the global search phase and the local search phase. In the global search phase, GA is used to search for the best initial weights of the neural network. In the local search phase, random perturbation is added to avoid the local optimum and vanishing gradient problems. A case study of crop yield prediction is conducted to compare the proposed method and other gradient-based methods. The results show that the proposed method outperforms the gradient-based methods in terms of convergence speed and prediction accuracy.

Keywords: crop yield prediction, gradient descent, genetic algorithm, neural network

2.1 Introduction

The world population is expected to grow from 7.2 billion to between 9.6 billion by 2050 and 12.3 billion by 2100 [1]. This imposes significant challenges for agriculture production due to the

increasing demand and limited arable land. One way to alleviate this problem is to improve crop production with efficient crop breeding. Traditional breeding is phenotype based, which requires selection of the individual after phenotyping. The development of genomic and analytic technologies has reduced the genotyping and sequencing cost significantly. Now there are over 7.4 million plant accessions in gene banks around the world. To effectively leverage these resources and shorten the breeding time, analytic models are essential to predict the phenotype of the crops. This would significantly reduce the phenotyping cost and improve the efficiency of crop breeding.

Crop yield is jointly determined by the genotype and environment of the plants through complex biological and physiologic relationships. Understanding these relationships remains a significant challenge. Many of the existing studies are based on linear models [2]. However, the limitations of the linear models include the interactive effects of genotype and environment factors cannot be modeled and analyzed effectively. The phenotypes expressed by a genotype under varied environmental conditions can be very different. In other words, the resulting phenotypic variation based on the same genotype can be environment dependent [3]. Then, linear mixed model based on maximum likelihood was developed which can detect the correlation between genotype and environment [4]. However, this method has limitations on the size of dataset and the prediction accuracy due to the sensitivity of the types of input parameters. The complexity of the G x E problem and the limitation of the existing prediction model serve as the major motivation for this study to design mathematical models and the solution method that can overcome these shortcomings. In this paper, we proposed a neural network method for the crop yield prediction. Instead of analyzing the G x E interaction, we predict the crop yield using genotype and environment information through neural networks.

Neural Networks are function approximators which have achieved state-of-the-art accuracy in many applications, such as self-driving cars, game-playing and face identification [5]. They especially excel at learning from large datasets with labeled samples [6]. The most popular methods to train the deep artificial neural networks (DNNs) are gradient-based learning algorithms which is also called first-order methods. The reason are to their ease of

implementation, memory efficiency (typically requiring storage on the order of the parameter dimension), and convergence guarantees [7]. There are many mature gradient-based algorithms, such as stochastic gradient descent (SGD) [8], and improved versions, e.g., Adadelta [9] and Adam [10].

Despite its popularity as a universal function approximator and easy implementation, the gradient-based algorithms are faced with inherent drawback of getting trapped in local minima and slow convergence due to random initialization of synaptic weights and biases prior to training a neural network [11]. With every re-run of neural network during training phase, the gradient-based algorithms train the neural network based on different initial weights leading to different prediction performance and convergence speed [12]. The other problem is the vanishing/exploding gradient problem caused by the long training process. Many techniques have been designed to reduce the impact of the problem, such as variations on weight initialization strategy, alternative network structures and gradient descent schemes [13, 14, 15]. The use of gradient descent schemes is the most common change since it can keep the gradient in a reasonable range to avoid gradient vanishing/exploding. The commonly adopted gradient descent schemes include rectified linear unit (ReLU) activation function [16], gradient clipping [17] and L2 normalization [18]. However, the drawback is that these methods require more computation resources. In order to minimize the probability of inconsistency and solve the vanishing/exploding gradient problem, it is necessary to develop an effective methodology to improve its prediction accuracy and convergence to global optima. This is major motivation for the algorithm design in this study.

Neuroevolution is a machine learning method that uses evolutionary algorithms to optimize neural networks [19]. One important feature of neuroevolution algorithms is that they can adapt to a dynamic environment by their population-based search strategy [20]. At each iteration, evolution strategies will generate many children solutions, i.e., different neural networks. By comparing the fitness of generated solutions, the best neural network will be selected. Genetic algorithm (GA) is one of the popular evolutionary algorithms due to its ease implementation and

good convergence [21]. GAs can effectively address the limitations of the traditional gradient-based method through two improvements. First, GA can help avoid the local optimum problem by generating a population which consists of multiple solutions [22]. Second, GA can be used as a zeroth-order optimization method using approximate gradient. However, these methods exhibit poor convergence properties when the parameter dimension is large [23].

The combination of evolutionary algorithms and neural network has been tried since the 1960s [24]. However, most of them were designed for shallow neural networks which have only one or several hidden layers. For a given degree of function approximation, the number of neurons needed by shallow networks is exponentially larger than that of deep neural networks [25]. Recently some studies have proposed some neuroevolutionary methods for different deep neural networks, e.g., multi-layered neural network [26], deep reinforcement learning [27]. Although these neuroevolution methods have been extended to different structures of deep neural networks, the drawback of current methods includes two aspects. First, each generated solution means a new neural network. To evaluate the fitness of the generated solution, we need to train it until it converges so that we can determine which solution is better. It requires many computing resources to evaluate every generated solution. Second, although evolution strategies can help jump out the local minimum, it is not always as efficient as the gradient method due to the randomness of evolution strategies.

To overcome the aforementioned problems, this paper proposes a new framework that employs GA and gradient decent to update the parameters of neural networks. The contribution of this paper includes the following aspects.

- To improve the initialization of neural networks, we proposed a GA assisted method for deep neural networks. In the global search phase, we generate multiple sets of parameters for the neural network and use GA to search the solution space.
- To address the problem of local optimum in neuroevolution, our proposed method combines the gradient method and the GA method in the local search phase. At first, the GA

algorithm will evolve a low-dimensional subspace, i.e., the nodes in one layer, by adding a random perturbation. Then the weights will be updated by gradient decent method.

- The challenge of crop yield prediction from Syngenta is used as case study. The results show that it outperforms the gradient decent method about 10% in terms of prediction accuracy. The proposed method outperforms the gradient-based methods in terms of convergence speed and prediction accuracy. The proposed method is also compared with other methods that address the gradient vanishing problem. The results show that the root mean square error of the proposed method is 3%-5% lower than that of others.

The remainder of this paper is organized as follows. In Section 2.2, the problem description of crop yield prediction is illustrated. In Section 2.3, the application of the neural network to plant traits (e.g., yield) prediction is introduced and the drawbacks of the gradient-based methods are discussed. The proposed GA-assisted approach is presented in Section 2.4. Section 2.5 includes the numerical experiments and analysis of the results. Finally, conclusions are drawn in Section 2.6.

2.2 Crop Yield Prediction Using Genotype and Environment information

The objective of this study is to predict the phenotype of crop, i.e., yield, with the genotype and environment information.

Genotype (G) refers to genetic makeup of an individual which is the nucleotide sequences of DNA (a gene or genes) that are transmitted from parents to offspring [28]. A gene is defined as a sequence of DNA or RNA that is the basic physical and functional unit of heredity [29]. Genotype includes one gene locus (AA, Aa, aa), two gene loci (AABB, AaBB, AAbb, etc.), or multiple genes (aabbcc, aabbccddee, AABbCCDDEEFF, etc.). For example, Figure 2.1 shows genomic information of 40 genotypes which consists of 100 gene loci.

Environment can be defined as the circumstances surrounding an organism or a group of organisms [28]. Phenotype (P) refers to an individual's discernible traits, such as yield, height and stalk number. Phenotype is the expression of a genotype in an environment (E). Phenotypes are

Figure 2.1 Genomic information of 40 genotypes (each color represents one gene type).

determined by genotype and environment together. However, there are some interactions between genotype and environment. Figure 2.2 is used to illustrate the relationship among phenotype, genotype and environment from three points. (1) The plants of identical genotypes (e.g., Genotype A) growing in different environments may show different phenotypes. (2) When the environment is fixed, the plants of different genotypes (e.g., the left endpoints of the three lines shows that) also show different phenotypes. (3) Different genotypes react differently when environment factors change. For example, when the soil moisture increases in a specific range, plants of Genotype C will have higher yield while plants of Genotype B will have lower yield.

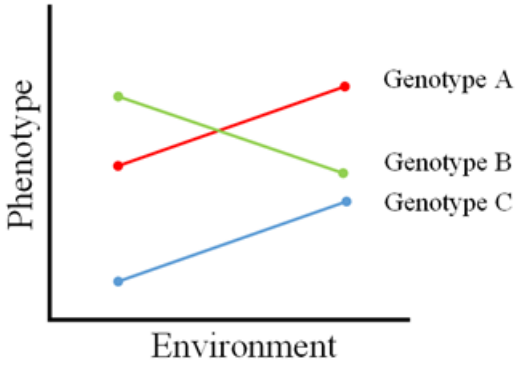


Figure 2.2 Genotype and environment interaction [28].

This paper introduced a neural network-based approach to predict the crop yield. The advantage of the neural networks is that they can fit the complex non-linear relationship without analyzing the G by E interaction specifically.

2.3 Deep Neural Network and the Drawback of the Gradient Decent Method

Traditional methods used to predict crop yield is linear model. However, the prediction of additive models not satisfying. In recent years, the linear mixed model and regression approaches become very popular [30, 31, 32],. They are more advanced and have achieved better prediction accuracy. The problem is that their performance is not satisfying when dealing with large dataset which consists of millions of variables and samples. In this case, the complexity of the problem increases significantly. Neural network due to its ability to map complex non-linear and unknown relationships is a preferred choice among researchers for modeling this kind of problems [33, 34]. Deep neural networks (DNNs) are multi-layer feed-forward neural networks which are usually trained using gradient-based methods [35]. The gradient-based algorithm is a local search algorithm which employs gradient descent to iteratively update the weights and biases of the neural network, minimizing the loss function commonly measured in terms of a squared error between the actual results and the output of the neural network [12].

As shown in Figure 2.3, a DNN consists of input layer, hidden layers and output layer. Each layer consists of multiple nodes. A node multiplies input from the data with a set of coefficients. Then, these products are summed. So far, this model is still linear. However, some problems cannot be solved by a linear model. To solve this problem, a node's so-called activation function is introduced. The widely used activation functions includes Sigmoid function, Hyperbolic Tangent (Tanh) function and Rectified Linear Unit (ReLU) function. The Sigmoid function has been successfully applied in prediction problems as well as classification problems because it is bonded, differentiable and monotonic [36]. The Tanh function is a transformed version of the Sigmoid function, which has larger gradients. It performs better than the Sigmoid function in the training of multi-layer neural networks [37]. The ReLU function was proposed by Nair and Hinton

in 2010 [38]. Since it preserves the properties of linear models, the ReLU function reduces the computing complexity of gradient-descent methods [39]. However, all these functions sometimes suffer from gradient vanishing problem.

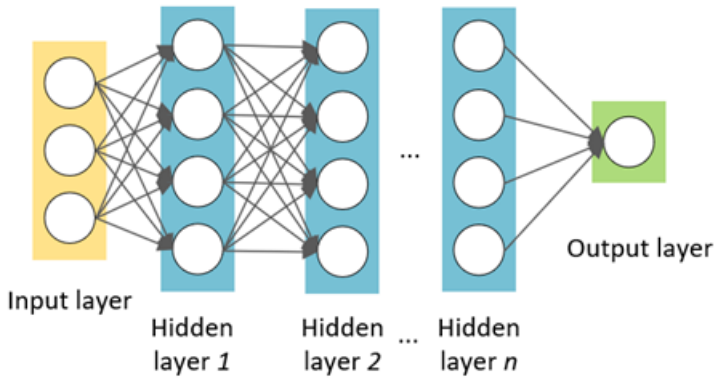


Figure 2.3 Structure of deep neural network.

Given an input vector of dimension d, we consider a neural network with L layers of neurons for prediction. We denote by M_l the number of neurons in the l-th layer (note that $M0 = d$). We denote the neural activation function by σ. Let denote the weight matrix connecting the and l-th layer and bl denote the bias vector for neurons in the l-th layer. Let W_{L+1} and R denote the weight vector and bias scalar in the output layer, respectively. Therefore, the output of the network R can be expressed by

$$f(x; W) = W_{L+1}^T \sigma \left(W_L^T \sigma \left(\ldots \sigma \left(W_1^T x + b_1 \right) + b_{L-1} \right) + b_L \right) + b_{L+1} \tag{2.1}$$

A standard method to update the W in Eq. (2.1) is gradient decent [40]. It can be represented by the following update rule:

$$W^{(t)} = W^{(t-1)} - \eta_t \nabla f \left(x, W^{(t-1)} \right) \tag{2.2}$$

However, this update rule come along with a problem, i.e., gradient vanishing or gradient exploding problem [41]. Take b_1 as an example. The update rule is shown in Eq.(2.3). Generally, the initial value of w is lower than 1. If the value of σ' is less than 1, the output tends to be very

small, which means that the update of the parameter in the first layer is very small. On the other hand, if $\sigma' > 1$, the term will be very large. The gradient vanishing or gradient exploding problem is essentially caused by the chain multiplication.

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_{L+1}} \sigma'(z_{L+1})\, w_{L+1} \sigma'(z_L)\, w_L \ldots \sigma'(z_2)\, w_2 \sigma'(z_1) \tag{2.3}$$

Techniques have been proposed to address this issue, such as Leaky ReLU, gradient clipping and L2 normalization. As shown in Eq. (2.4), the gradient of the LeakyReLU activation function is 1 when the unit is active; otherwise, it is a small, positive value. Gradient clipping is to assign a fixed value to the gradient when the gradient is too large or too small. L2 normalization is to normalize the gradient so that the sum of squares will always be up to a fixed value. In this paper, we proposed a GA-assisted method to update the parameters of the neural network.

$$f(x) = \left\{ \begin{array}{ll} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{array} \right\} \tag{2.4}$$

2.4 Proposed GA-Assisted Neuroevolution Approach

To overcome the drawback of gradient decent method motioned above, we proposed a novel zeroth-order method which is named as GA-assisted neuroevolution approach. The advantages of the proposed approach include the ability to avoid the local optimum and gradient vanishing problem, the ease of the computation, the applicability to deep neural networks, and the ability to handle large dataset.

2.4.1 Genetic algorithm

The idea of genetic algorithms is based on one nature mechanism, namely evolution [42, 43]. The workflow of genetic algorithm is shown in Figure 2.4. There are three main operators:

Selection: During each successive generation, two individuals of the existing population are selected to generate a new individual. The selection process is usually based on the fitness of

individuals. A typical selection method is the roulette in which the individuals of better fitness are preferred. Because there is good reason that the child solution which combines the component of good parent solutions is more likely to be good. However, this method might be time-consuming when the population is large. There are also some methods that randomly select the parent solutions. In these methods, we do not need to calculate the fitness of each solution.

Crossover: After the selection of the parent solutions, the crossover operator is executed. Crossover is also called recombination. It is used to combine the advantageous genes of the parent solution to generate a new individual solution. If the generated solution is better than the parent solutions, replace the old ones with the new one. The key point of the crossover operator is how to detect the advantageous genes of the parent solutions, especially for the large-scale optimization problem. One solution to this is to shrink of the exchange scope of the parent solutions.

Mutation: In genetic algorithms, mutation is a genetic operator used to maintain genetic diversity. In the process of selection and crossover, there is no new gene introduced. On the contrary, some genes might be lost in the process. The consequence of this is that all the individuals in the population will be identical and the algorithms will converge early. To avoid this case, mutation is used to alter one or more genes randomly. Since mutation operator is a random search strategy, mutation may change the previous solution largely. Therefore, the mutation operator is usually executed according to a small threshold probability.

GA has been successfully applied to the artificial neural networks. However, different from the shallow neural networks, deep neural networks have millions of weights to be evolved. In other words, it is a large-scale optimization problem [44]. In a large-scale problem, the search space will expand exponentially due to the increase of variables [45]. The performance of general evolutionary algorithms will deteriorate rapidly due to the "curse of dimensionality" [23, 46]. A popular solution is the divide-and-conquer strategy. In this paper, a GA-based approach which integrates global search strategy and local search strategy is proposed.

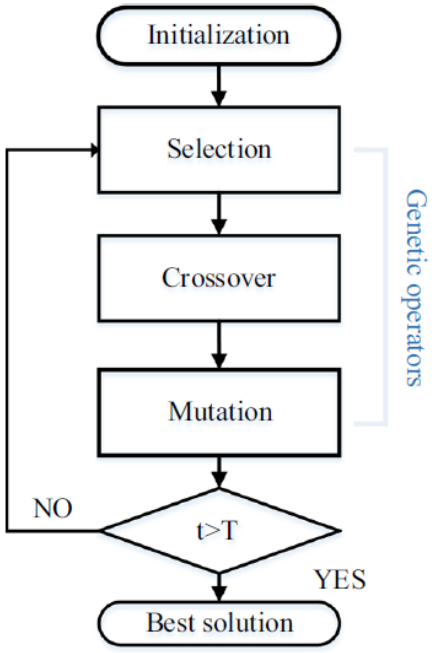


Figure 2.4 Workflow of genetic algorithm.

2.4.2 GA-assisted Neuroevolution Approach

The workflow of the approach is shown in Figure 2.5. There are two phases in the proposed approach, i.e., global search and local search. In the initialization, we randomly generate multiple candidate matrices for each weight and bias. The goal of global search is to search for the best combination. The best individual in the global search phase is used as the base for further update in local search.

Global search phase: The parameter initialization of neural networks has great influence on the algorithm performance. In order to avoid this impact, a novel global search strategy is designed to ensure the quality of the initial point. Instead of generating only one group of parameters, it randomly generates n groups of parameters for each layer. Therefore, there are n candidate weights and biases for each layer. The total number of possible solutions will be nL. Then, the global search strategy is used to search for the best combinations in this solution space.

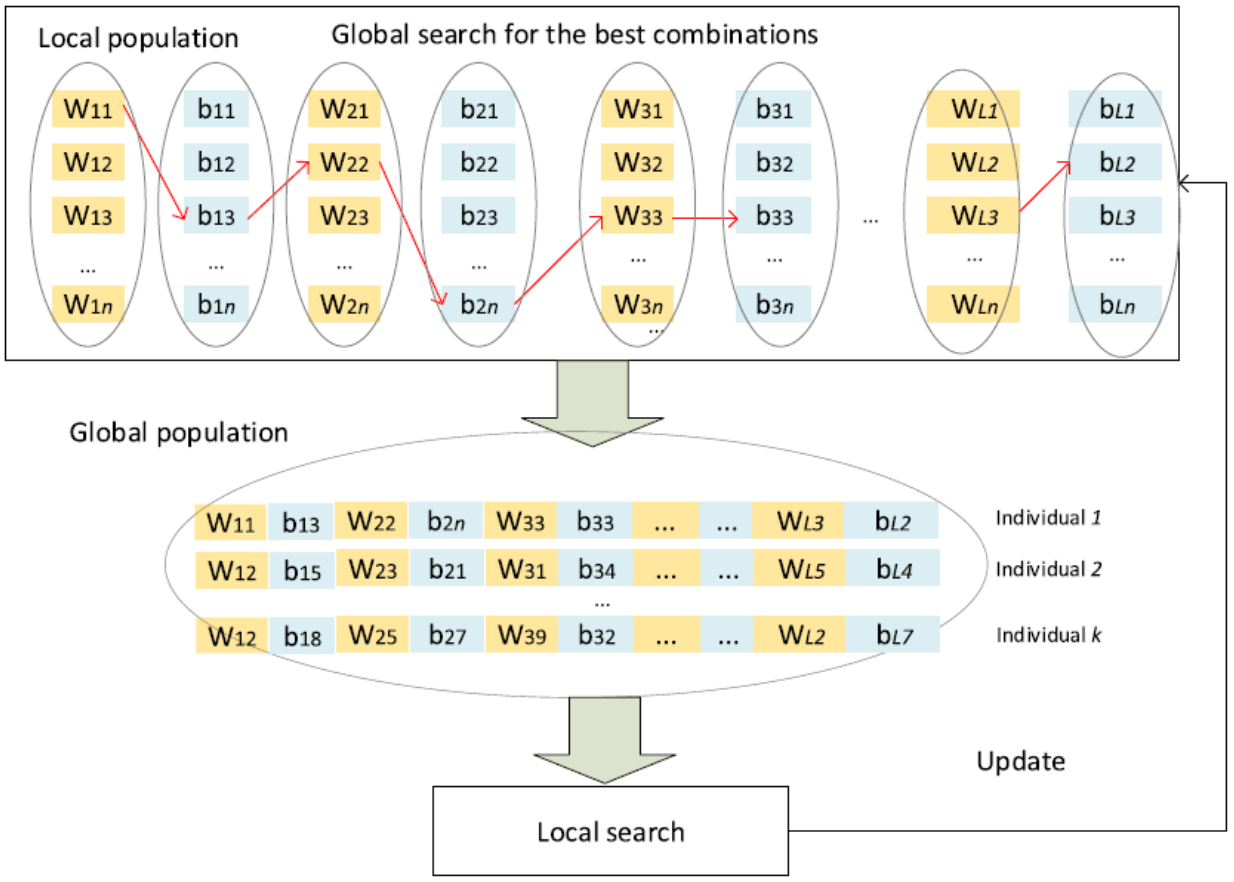


Figure 2.5 Workflow of GA-assisted deep learning approach.

- Initialization. The global population is randomly generated. An individual can be represented by the number of selected weight or bias matrices. For example, can be represented as (1, 3, 2, n, 3, . . .).

- Evolution of elites. The elites are the best m individuals. The fitness of an elite is good enough that it is not easy to improve it by selection and crossover operators. Therefore, for each elite, two positions are selected randomly. Then the selected positions will be mutated, i.e., replaced by a random selected solution in the candidate pool. For example, as shown in Figure 2.6, the W_{22} and b_{33} of the elite are replaced by the W_{25} and b_{37}, respectively.

- Evolution of worst m individuals. The strategy to improve bad individuals is to combine them with the elites. Since it is safe to make big changes on the current individual, the current individual is used as the father solution. Randomly selected one of the elites as the mother solution. Then the two-point crossover operator is executed by randomly selecting two cutting points and exchange the segment between two cutting points. For example, as shown in Figure 2.7, the child solution inherits the W_{11}, b_{13}, W_{22}, WL_3 and bL_2 from Parent 1 solution and other parts from Parent 2 solution. There are also some other versions of crossover operators, e.g. the single-point crossover and the uniform crossover. The single-point crossover operator is to select one point and then recombine two individuals. Assuming the number of genes is N, the total number of possible combinations is (N-1) since there are (N-1) possible cutting points. The uniform crossover is to choose each bit with equal possibility. For uniform crossover, there are 2N possible combinations because each gene has two states, i.e., selected or not selected. However, the uniform crossover can't guarantee the exploration ability in 2N space when N is large. Thus, the two-point crossover, which is able to generate C(N+1, 2) combinations, can achieve better balance between the exploration ability and the exploitation ability. The two-point crossover also requires less time than the uniform crossover operator in terms of computing efficiency.

- Evolution of other individuals. The current individual is selected as Parent 1 solution. One elite is randomly selected as Parent 2 solution. Then randomly select a starting point and exchange the next S parameters. For example, as shown in Figure 2.8, a segment consisting of 3 matrices is crossovered.

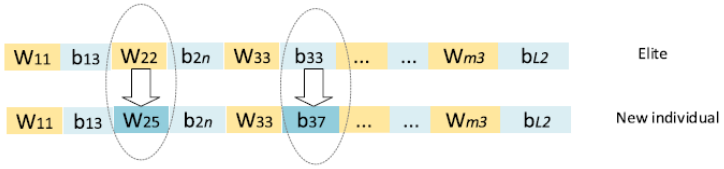


Figure 2.6 Evolution strategy of elites.

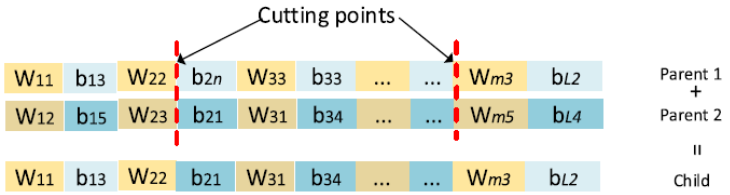


Figure 2.7 Evolution strategy of worst individuals.

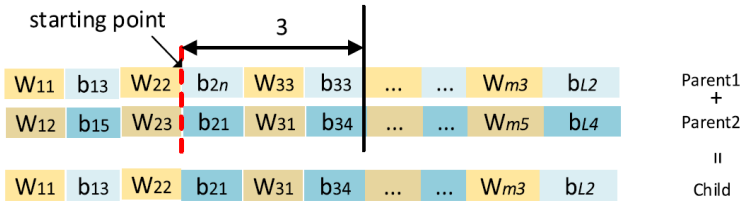


Figure 2.8 Evolution strategy of other individuals.

Local search phase: In addition to computing the gradient, we add a perturbation to the parameters to help the algorithm get out of local optimum [47, 19, 48]. At each iteration, only one weight or bias matrix is selected to add a randomly generated perturbation. Each weight or bias matrix is trained batch by batch. We also introduced the drop out strategy to avoid over-fitting. It is realized by the mutation operation which randomly select multiple values of the matrix and set it as zero. If the fitness of the new individual is better, replace the old individual with the new one and break the loop.

Besides, a Simulated Annealing algorithm based strategy has been designed to assist the algorithm to jump out the local optimal [49]. The simulated annealing algorithm is inspired by the processes which occur during the cooling of physical systems and is a probabilistic technique for approximating the global optimum. If a liquid is cooled slowly, the atoms anneal to increase the size of its crystals. The slow cooling process can be interpreted as a slow decrease in the probability of accepting worse solutions. The algorithm will start with an initial, often randomly generated solution X and evolve it according to some evolution strategies, to propose an updated solution X'. If X' has better fitness, the algorithm replaces X with X'. However, if X' is worse

than X, it will be accepted with a calculated probability. Thus, in the local search phase, even if the fitness of the generated individual is worse, it is still possible to be accepted.

By combining the strength of the gradient decent and random search, the convergence and efficiency of the proposed method can be guaranteed. In summary, the proposed method is a two-phase algorithm. The global search strategies are utilized to increase the exploration ability while the local search strategies are introduced to increase the exploitation ability. The local search can also help avoid the vanishing gradient and local optimum by adding a random Gaussian perturbation. To increase the robustness and the efficiency of the algorithm, the gradient decent method is used to update the weights of the neural networks.

2.5 Case study

To validate the proposed solution technique, numerical experiments have been conducted. A dataset from Syngenta Crop Challenge is used as a case study. The target is to predict the crop yield using the genetic variables and environment variables. The codes are implemented in Python. All the experiments are carried out in a computer with 3.1 GHz CPU and NVIDIA GTX1080 GPU.

2.5.1 Data

The Syngenta Crop Challenge dataset consists of 148,452 samples which were collected from 2009-2016. The dataset consists of two types of data, i.e., genetic data and environment data. The genetic data is represented by three discrete values, i.e., “-1” , “0” and “+1”. The environment data includes soil data and weather data, which is represented by continuous values. The total number of variables is 13,550. The target of the project is to predict the difference between yield and the benchmark result.

Before using the dataset to train the neural network, we should deal with the missing values in genetic data. We dropped the variables whose data is missing for more than 30% sample. Next, we deleted the samples that have more than 30% missing values. For the other missing

values, we adopted the imputation method, i.e., filling the value with mean values. Since the number of variables is very large, a linear regression model is used to select most relevant variables. As a result, one hundred and fifty-four variables have been selected. We used 20% of the samples as the testing dataset.

2.5.2 Experimental parameters and Result Analysis

The neural network we used in the experiment consists of 20 layers. The first hidden layer contains 128 neurons. Each of other hidden layers contains 64 neurons. The activation function is Tanh function.

In the first experiment, the neural network is trained by using the package of three gradient-based methods, i.e., gradient descent, Adam and Adagrad, in the TensorFlow framework. The learning rate is 0.003. The batch size is set as 800. The number of iterations is 1000. As shown in Fig. 9, before 80 iterations, the descent speed of the three methods is very fast. The RMSE is decreased by about 20%, from 20.1 to 15.6. After 80 iterations, the speed slow down and stop updating. In terms of the quality of initial points, Adagrad is 17.74 which is better than 20.14 of gradient descent and 19.93 of Adam. However, they all converge at about 15.6.

In the second experiment, the neural network is train by using the proposed method. The number of individuals in the local population is set as 10. The number of individuals in the global population is 120. The number of iterations in the global search phase is set as 120 iterations. It should be noted that it does not make any sense by comparing the iterations in the two experiments since they are programmed in different ways.

The running process is shown in Figure 2.9. Compared with the gradient descent methods, the quality of the initial point of the proposed method is much better. The reason is that, in the global search phase, the RMSE is decrease by about 11%, from 18.163 to 16.250. When the decreasing speed slows down, the local search strategy is performed. In the initial phase of local search, the RMSE is decreased dramatically, from 16.250 to 15.27. Then it decreases in a gentle way until it converges at 14.874.

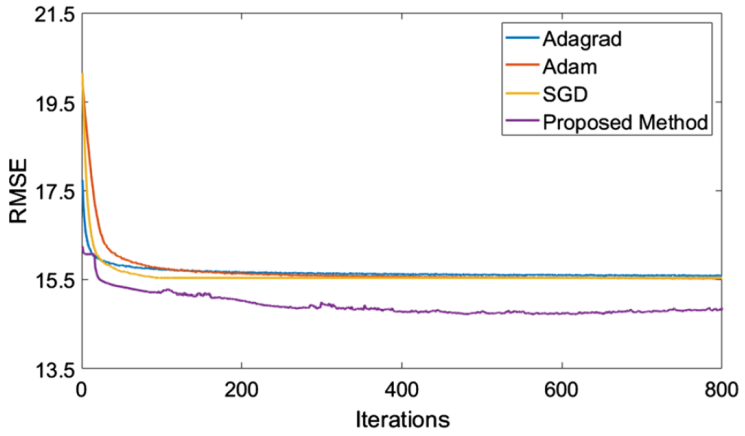


Figure 2.9 Training process of gradient-based methods.

The above experiment has been repeated for five times. The result comparison is shown in Table 2.1. Compared with the traditional gradient decent method, our proposed method improves the training RMSE and testing RMSE by 4% and 5%, respectively. In terms of algorithm stability, the training standard deviation (SD) of the proposed is better than that of Adam and Adagrad. The testing SD of the proposed is comparable that of other methods.

Table 2.1 Comparison results among SGD, Adagrad, Adam and the proposed method.

	Training RMSE	Training SD	Testing RMSE	Testing SD
SGD	15.571	0.073	15.422	0.123
Adagrad	15.620	0.114	15.587	0.074
Adam	15.608	0.116	15.619	0.094
Proposed method	14.944	0.093	14.957	0.094

SD: standard deviation

The proposed method has been compared with other techniques that aim to improve the quality of gradients, i.e., Leaky ReLU, gradient clipping and L2 normalization. The parameter of Leaky ReLU is set to 0.01, meaning the gradient is equal to 0.002 when the unit is not activated. The gradient clipping value is specified as 0.5, meaning that if a gradient value was less than - 0.5, it is set to 0.5 and if it is more than 0.5, then it will be set to 0.5. The parameter of L2 normalization is set to 0.9, meaning that the sum of squares of the gradients is up to 0.9. Since

SGD performs better than Adagrad and Adam according to Table 1, it is used as the baseline optimizers. The experiment was repeated for five times. As shown in Table 2.2, both of the training RMSE and the testing RMSE of the proposed method are 3%–5% lower than those of other three methods. It proves that the combination of GA and the gradient descent method can help improve the algorithm performance. Besides, the training SD and the testing SD of the proposed method are also the lowest, which validates the algorithm stability.

Table 2.2 Comparison results among Leaky ReLU, Gradient Clipping, L2 normalization and the proposed method (SD: standard deviation).

	Training RMSE	Training SD	Testing RMSE	Testing SD
SGD + LeakyReLU	15.367	0.117	15.422	0.114
SGD + Gradient Clipping	15.644	0.138	15.664	0.117
SGD + L2 Normalization	15.748	0.100	15.740	0.128
Proposed method	14.944	0.093	14.957	0.094

2.6 Conclusion

Improving the crop performance through crop breeding is one important path to alleviate the potential problems to feeding the growing population. Efficient and effective crop breeding relies on accurate crop yield prediction, which is one of the most important crop phenotypes. However, the crop phenotype is jointly determined by the genotype and environment factors though complicated interactive relationships. We proposed a GA assisted deep learning method to predict the crop yield. Different from the first-order method, i.e., gradient decent method, the proposed method combines the zeroth-order method and the gradient method to improve the efficiency and robustness. Besides, it also can avoid the gradient degradation problem in training deep neural networks. We also add the batch strategy to the algorithm so that it can deal with large datasets. In the case study of Syngenta crop challenge, the experiment results show that the proposed method can reduce the RMSE by about 10%.

On the basis of this paper, plant traits prediction using neuroevolution deep learning methods can be further investigated. However, current approach is designed only for fully connected deep neural network. Future work will include the investigation of the effect of the perturbation on the evolution of the neural network. We are going to apply the proposed method to different deep neural network, e.g. convolutional neural network.

References

[1] Patrick Gerland, Adrian E Raftery, Hana Ševčíková, Nan Li, Danan Gu, Thomas Spoorenberg, Leontine Alkema, Bailey K Fosdick, Jennifer Chunn, Nevena Lalic, et al. World population stabilization unlikely this century. *Science*, 346(6206):234–237, 2014.

[2] David L Des Marais, Kyle M Hernandez, and Thomas E Juenger. Genotype-by-environment interaction and plasticity: exploring genomic responses of plants to the abiotic environment. *Annual Review of Ecology, Evolution, and Systematics*, 44:5–29, 2013.

[3] RE Comstock and R_ H Moll. Genotype environment interactions. statistical genetics and plant breeding. Technical report, 1963.

[4] James B Holland. Estimating genotypic correlations and their standard errors using multivariate restricted maximum likelihood estimation with sas proc mixed. 2006.

[5] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

[6] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30078, 2020.

[7] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[8] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[9] Mahesh Chandra Mukkamala and Matthias Hein. Variants of rmsprop and adagrad with logarithmic regret bounds. In *International Conference on Machine Learning*, pages 2545–2553. PMLR, 2017.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Fabian Ruehle. Evolving neural networks with genetic algorithms to study the string landscape. *Journal of High Energy Physics*, 2017(8):1–20, 2017.

[12] Vinay Chandwani, Vinay Agrawal, and Ravindra Nagar. Modeling slump of ready mix concrete using genetic algorithms assisted training of artificial neural networks. *Expert Systems with Applications*, 42(2):885–893, 2015.

[13] Bekhzod Olimov, Sanjar Karshiev, Eungyeong Jang, Sadia Din, Anand Paul, and Jeonghong Kim. Weight initialization based-rectified linear unit activation function to improve the performance of a convolutional neural network model. *Concurrency and Computation: Practice and Experience*, page e6143, 2020.

[14] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[15] Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? *arXiv preprint arXiv:1801.03744*, 2018.

[16] Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

[17] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*, 2019.

[18] Dan Wang, Sung-Kwun Oh, and Eun-Hu Kim. Design of space search-optimized polynomial neural networks with the aid of ranking selection and l2-norm regularization. *Journal of Electrical Engineering and Technology*, 13(4):1724–1731, 2018.

[19] Joel Lehman, Jay Chen, Jeff Clune, and Kenneth O Stanley. Safe mutations for deep and recurrent neural networks through output gradients. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 117–124, 2018.

[20] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[21] Darrell Whitley, Timothy Starkweather, and Christopher Bogart. Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel computing*, 14(3):347–361, 1990.

[22] Abdellah Salhi, Ghazwan Alsoufi, and Xinan Yang. An evolutionary approach to a combined mixed integer programming model of seaside operations as arise in container ports. *Annals of operations research*, 272(1-2):69–98, 2019.

[23] Hiroaki Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex systems*, 4:461–476, 1990.

[24] Hans J Bremermann et al. Optimization through evolution and recombination. *Self-organizing systems*, 93:106, 1962.

[25] Chitta Baral, Olac Fuentes, and Vladik Kreinovich. Why deep neural networks: a possible theoretical explanation. In *Constraint programming and decision making: theory and applications*, pages 1–5. Springer, 2018.

[26] Filipe Assunção, Nuno Lourenço, Penousal Machado, and Bernardete Ribeiro. Fast denser: Efficient deep neuroevolution. In *European Conference on Genetic Programming*, pages 197–212. Springer, 2019.

[27] Sebastian Risi and Kenneth O Stanley. Deep neuroevolution of recurrent and discrete world models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 456–462, 2019.

[28] Manjit S Kang. Using genotype-by-environment interaction for crop cultivar development. *Advances in agronomy*, 62:199–252, 1997.

[29] Adam S Davis, Jason D Hill, Craig A Chase, Ann M Johanns, and Matt Liebman. Increasing cropping system diversity balances productivity, profitability and environmental health. 2012.

[30] LR Schaeffer and J Jamrozik. Random regression models: a longitudinal perspective, 2008.

[31] John R Stinchcombe, Mark Kirkpatrick, Function valued Traits Working Group, et al. Genetics and evolution of function-valued traits: understanding environmentally responsive phenotypes. *Trends in Ecology & Evolution*, 27(11):637–647, 2012.

[32] Matthew R Robinson and Andrew P Beckerman. Quantifying multivariate plasticity: genetic variation in resource acquisition drives plasticity in resource allocation to components of life history. *Ecology letters*, 16(3):281–290, 2013.

[33] Martin Schwardt and Kathrin Fischer. Combined location-routing problems—a neural network approach. *Annals of Operations Research*, 167(1):253–269, 2009.

[34] Zeineb Affes and Rania Hentati-Kaffel. Forecast bankruptcy using a blend of clustering and mars model: case of us banks. *Annals of Operations Research*, 281(1):27–64, 2019.

[35] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[36] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks*, pages 195–201. Springer, 1995.

[37] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.

[38] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[39] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8609–8613. IEEE, 2013.

[40] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

[41] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[42] Lawrence Davis. Handbook of genetic algorithms. 1991.

[43] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[44] Yi Mei, Xiaodong Li, and Xin Yao. Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation*, 18(3):435–449, 2013.

[45] Firoozeh Kaveh, Reza Tavakkoli-Moghaddam, Chefi Triki, Yaser Rahimi, and Amin Jamili. A new bi-objective model of the urban public transportation hub network design under uncertainty. *Annals of Operations Research*, 296(1):131–162, 2021.

[46] Helenice de Oliveira Florentino, Chandra Irawan, Dylan F Jones, Daniela Renata Cantane, Jonis Jecks Nervis, et al. A multiple objective methodology for sugarcane harvest management with varying maturation periods. *Annals of Operations Research*, 267(1):153–177, 2018.

[47] Cédric Colas, Vashisht Madhavan, Joost Huizinga, and Jeff Clune. Scaling map-elites to deep neuroevolution. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 67–75, 2020.

[48] Andri Ashfahani, Mahardhika Pratama, Edwin Lughofer, and Yew-Soon Ong. Devdan: Deep evolving denoising autoencoder. *Neurocomputing*, 390:297–314, 2020.

[49] Dimitris Bertsimas and John Tsitsiklis. Simulated annealing. *Statistical science*, 8(1):10–15, 1993.

CHAPTER 3. IMPROVING IMAGE-BASED PLANT DISEASE CLASSIFICATION WITH GENERATIVE ADVERSARIAL NETWORK UNDER LIMITED TRAINING SET

Luning Bi and Guiping Hu

Department of Industrial and Manufacturing Systems Engineering, Iowa State University

Modified from a manuscript published in *Frontiers in Plant Science*

Abstract

Traditionally, plant disease recognition has mainly been done visually by human. It is often biased, time-consuming, and laborious. Machine learning methods based on plant leave images have been proposed to improve the disease recognition process. Convolutional neural networks (CNNs) have been adopted and proven to be very effective. Despite the good classification accuracy achieved by CNNs, the issue of limited training data remains. In most cases, the training dataset is often small due to significant effort in data collection and annotation. In this case, CNN methods tend to have the overfitting problem. In this paper, Wasserstein generative adversarial network with gradient penalty (WGAN-GP) is combined with label smoothing regularization (LSR) to improve the prediction accuracy and address the overfitting problem under limited training data. Experiments show that the proposed WGAN-GP enhanced classification method can improve the overall classification accuracy of plant diseases by 24.4% as compared to 20.2% using classic data augmentation and 22% using synthetic samples without LSR.

Keywords: plant disease, classification, regularization, convolutional neural network, generative adversarial network

3.1 Introduction

With the increasing global population, the demand for agriculture production is rising. Plant diseases cause substantial management issues and economic losses in the agricultural industry [1]. It has been reported that at least 10% of global food production is lost due to plant disease [2]. The situation is becoming increasingly complicated because climate change alters the rates of pathogen development and diseases are transferred from one region to another more easily due to the global transportation network expansion [3]. Therefore, early detection, timely mitigation, and disease management are essential for agriculture production [4].

Traditionally, plant disease inspection and classification have been carried out through optical observation of the symptoms on plant leaves by human with some training or experience. Plant disease recognition has known to be time-consuming and error-prone. Due to the large number of cultivated plants and their complex physiological symptoms, even experts with rich experience often fail to diagnose specific diseases and consequently lead to mistaken disease treatments and management [5].

Many methods have been developed to assist disease recognition and management. Lab-based techniques have been developed and established in the past decades. The commonly used techniques for plant disease recognition include enzyme-linked immunosorbent assay (ELISA), polymerase chain reaction (PCR), immunoflourescence (IF), flow cytometry, fluorescence in situ hybridization (FISH), and DNA microarrays [6]. However, these techniques require an elaborate procedure and consumable reagents. Meantime, image-based machine learning methods for plant disease recognition, which identify plant diseases by training computers with labeled plant images, have become popular. The advantages of image recognition include: (1) the ability to deal with a large number of input parameters, i.e., image pixels, (2) the minimization of human errors, and (3) the simplified process [7].

The key to improving the plant disease recognition accuracy is to extract the right features of the surface of plant leaves [8, 9]. The emergence of deep learning techniques has led to improved performance. Although deep learning based models take a long time to train, its testing time is

fast because all information from the training dataset has been integrated into the neural network [10]. For the agricultural applications, convolutional neural networks (CNN) have been used for image recognition [11]. Dhakate et al. used a convolutional neural network for the recognition of pomegranate plant diseases and achieved 90% overall accuracy [12]. Ghazi et al. proposed a hybrid method of GoogLeNet, AlexNet, and VGGNet to classify 91,758 labeled images of different plant organs. Their combined system achieved an overall accuracy of 80% [13]. Ferentinos developed CNN models to classify the healthy and diseased plants using 87,848 images. The success rate was significantly high which can reach 99.53% [5]. Ma et al. proposed a deep CNN to recognize four cucumber diseases. The model was trained using 14,208 images and achieved an accuracy of 93.4% [14]. With the high classification accuracy, it can be concluded that CNNs on leave images are highly suitable for plant disease recognition [15].

It should be noted that the high prediction accuracy is predicated on that thousands of labeled images were used to train CNNs. A major problem often facing the automatic identification of plant diseases with CNNs is the lack of labeled images capable of representing the wide variety of conditions and symptom characteristics found in practice [16]. Experimental results indicate that while the technical constraints linked to automatic plant disease classification have been largely overcome, the use of limited image datasets for training brings many undesirable consequences that still prevent the effective dissemination of this type of technology [17]. Real datasets often do not have enough samples for deep neural networks to properly learn the classes and the annotation errors, which may damage the learning process [4]. If the model learns to assign a full probability to the ground truth label for each training example, it is not guaranteed to generalize because the model becomes too confident about its predictions [18]. It should be noted that although it is relatively cheap to collect images, using additional unlabeled data is non-trivial to avoid model overfitting. This serves as the major motivation for this study on developing a new method that can address the plant disease classification with limited labeled training images.

Data augmentation using synthetic images is the most common method used in training CNN with small amounts of data [19]. Hu et al. synthesized face images by compositing the

automatically detected face parts from two existing subjects in the training set. Their method improved over the state-of-the-art method with a 7% margin [20]. Guo et al. merged the training set with another dataset from the same domain and obtained a performance improvement of 2% [21]. Papon et al. proposed a rendering pipeline that generates realistic cluttered room scenes for the classification of furniture classes. Compared to using standard CNN, the proposed method improved the classification accuracy by up to 2% [22]. These methods generate synthetic images by extracting and recombining of local regions of different real images.

In this study, we designed a generative adversarial network (GAN) to generate completely new synthetic images to enhance the training set. GAN was designed based on game theory to generate additional samples with the same statistics as the training set. Compared with the methods in the existing literature, GAN is capable to generate full synthetic images that can increase the diversity of the dataset. Therefore, it has become an increasingly popular tool to address the limited dataset issue [23]. Nazki et al. proposed Activation Reconstruction (AR) – GAN to generate synthetic samples of high perceptual quality to reduce the partiality introduced by class imbalance [24]. Compared with Nazki's work which considered 9 classes of images with about 300 images in each category, our work has considered a more stringent situation of limited dataset which includes 38 classes with 10-28 images in each category. Therefore, one of the key objectives of this study is to reduce overfitting of the model. Label smoothing regularization (LSR) is introduced in this paper. In addition to maximizing the predicted probability of the truth-ground class, LSR also maximizes the predicted probability of the non-truth ground classes [18]. Similarly, Xie et al. proposed a method named DisturbLabel which prevents the overfitting problem by adding label noises to the CNN [25]. Pereyra et al. found out that label smoothing can improve the performance of the models on benchmarks without changing other parameters [26]. In our paper, Wasserstein generative adversarial network with gradient penalty (WGAN-GP) is combined with LSR to generate images that can enlarge the training dataset and regularize the CNN model simultaneously.

The main contributions of this study lie in two dimensions:

- To improve the generalization of the proposed method, multiple diseases and multiple plant types have been considered in this paper. The majority of the existing studies focused on a single type of disease or only one plant type. In reality, there may exist multiple diseases for one plant type. However, in reality, it is often necessary to detect the multiple diseases of multiple plant types. Therefore, it would be preferable to design recognition methods with the capability to address the multi-disease and multi-plant type situation.

- To address the issue of limited training set, an approach that combines classical data augmentation and synthetic augmentation is proposed. LSR has also been employed to increase the generalization ability of the model. Four experiments have been conducted to validate the effectiveness of each component in the proposed framework. The results show that compared to the classic data augmentation methods, the proposed method can improve the total accuracy by 4.2%.

The rest of this paper is organized as follows. Section 3.1 introduces the motivation of this paper. Section 3.2 explains the structure of the proposed regularized GAN-based approach. Section 3.3 includes a case study, the experiment results and comparisons. Finally, the paper concludes with the summary, findings, and future research directions in Section 3.4.

3.2 Materials and Methods

Image-based plant disease recognition techniques have been developed with the reduced cost for image collection and the increased computational resources. However, in many situations for plant disease, there is not enough well-labeled data due to the high cost of data annotation. Under these circumstances, the machine learning models are prone to overfitting and fail to make accurate classifications for new observations. This study aims to achieve high plant disease classification accuracy with limited training dataset.

3.2.1 Framework of the Proposed Method

To improve the prediction accuracy of CNN in the classification of plant diseases using a limited training dataset, three techniques have been designed and implemented in this study, i.e., data augmentation, WGAN-GP, and LSR. The framework of the proposed method is shown in Figure 3.1. The first step is to train the WGAN-GP with LSR using real images. The trained WGAN-GP is then used to generate additional labeled images. The synthetic images will be mixed with real images and then augmented through classic data augmentation methods. Finally, the combined dataset will be used to train the CNN. In the following few sections, we will discuss each of the components in detail.

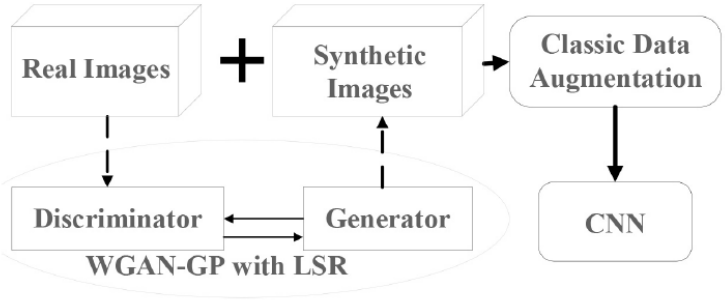


Figure 3.1 Framework of the proposed method.

3.2.2 Convolutional Neural Networks (CNN)

CNN is used as the supporting framework of our method. CNN is a class of deep, feed-forward artificial neural networks. It was adopted widely for its fast deployment and high performance on image classification tasks. CNNs are usually composed of convolutional layers, pooling layers, batch normalization layers and fully connected layers. The convolutional layers extract features from the input images whose dimensionality is then reduced by the pooling layers. Batch normalization is a technique used to normalize the previous layer by subtracting the batch mean and dividing by the batch standard deviation, which can increase the stability and improve the computation speed of the neural networks. The fully connected layers are placed near

the output of the model. They act as classifiers to learn the non-linear combination of the high-level features and to make numerical predictions. Detailed descriptions on each type of function can be accessed from Gu et al. [27].

It should be noted that CNN requires a large training dataset, which is typically not the case for plant disease recognition. With the number of model parameters is greater than the number of data samples, a small training dataset will lead to the overfitting problem, which results from a model that responds too closely to a training dataset and fails to fit additional data or predict future observations reliably. One of the commonly adopted methods to address this problem is data augmentation.

3.2.3 Data Augmentation

Data augmentation is a method to increase the number of labeled images. The classic data augmentation methods include vertical flipping, horizontal flipping, 90° counterclockwise rotation, 180° rotation, 90° clockwise rotation, random brightness decrease, random brightness increase, contrast enhancement, contrast reduction and sharpness enhancement. Figure 3.2 lists the examples of original image (Figure 3.2(a)), rotation (Figure 3.2(b)), brightness increase (Figure 3.2(c)), and contrast increase (Figure 3.2(d)).

(a) Original (b) Rotation (c) Brightness (d) Contrast

Figure 3.2 Augmentation methods.

Although data augmentation techniques decrease the impact of the limited training dataset problem, they cannot reproduce most of the practical diversity. This is also the reason why the generative adversarial network has been incorporated in this study.

3.2.4 Wasserstein Generative Adversarial Network (WGAN)

Unlike regular data augmentation methods, GAN is able to generate new images for training, which increases the diversity of data. GANs were firstly introduced by Ian Goodfellow et al. [23]. The generative adversarial networks (GANs) consist of two sub-networks: a generator and a discriminator. The generator captures the training data distribution while the discriminator estimates the probability that an image came from the training data rather than the generator.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}\,(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{3.1}$$

Where D represents the discriminator network, G is the generator network, z is a noise vector drawn from a distribution $p_{\text{Noise}\,(z)}$, x is a real image drawn from the original dataset $p_{\text{data}}\,(x)$.

The idea behind Eq. (3.1) is that it increases the ability of the generator to fool the discriminator which is trained to distinguish synthetic images from real images. The training process of the original GAN is shown in Figure 3.3. The specific steps are as follows.

Step 1 Initialize the parameters of the generator and the discriminator.

Step 2 Sample a batch of noise samples for the generator. Usually, uniform distribution or Gaussian distribution is used.

Step 3 Use the generator to transform the noise samples and predefined labels into images that are labeled as fake.

Step 4 The real images are labeled as true. Then the real images and the synthetic images are mixed and used as the input of the discriminator.

Step 5 Train the discriminator to improve the ability to classify the synthetic images and the real images.

Step 6 Train the generator to generate more images that will be discriminated as true by the generator.

Step 7 Repeat Step 2 - Step 6 until the termination condition is satisfied.

Many variants of GAN have been proposed in the past several years. Mirza et al. proposed the conditional GAN, which can provide better representations for multimodal data generation

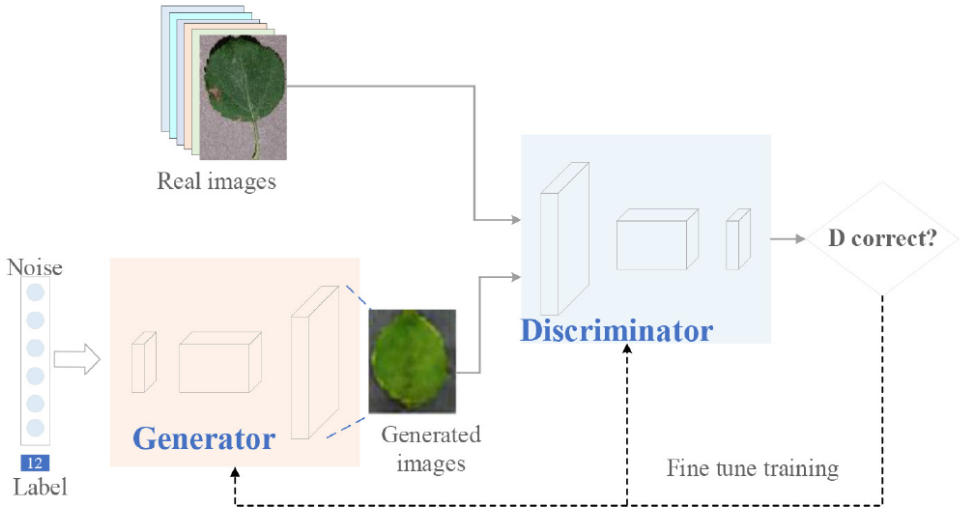


Figure 3.3 Training process of the original GAN.

[28]. Radford et al. proposed the deep convolutional GAN (DCGAN), which allows training a pair of deep convolutional generator and discriminator networks [29]. Arjovsky et al. proposed the Wasserstein GAN (WGAN) which uses Wasserstein distance to provide gradients that are useful for updating the generator [30]. Even though the WGAN performs more stable in the training process, it sometimes fails to converge due to the use of weight clipping. Therefore, Gulrajani et al. proposed an improved version of WGAN in which the weight clipping is replaced by the gradient penalty [31].

As shown in Figure 3.4, the major differences between the implementation of WGAN-GP and the original GAN include two aspects. The first is that the WGAN-GP uses the Wasserstein loss function with gradient penalty. Compared with the Jensen–Shannon (JS) and Kullback–Leibler (KL) divergence used in the DCGAN, Wasserstein distance can measure the distance between the distribution of real images and fake images, which can help improve the convergence of the network. The second is that in the WGAN-GP, the real and fake images are labeled as 1 and -1, while in the DCGAN, they are labeled as 1 and 0. This encourages the discriminator (critic) to output scores that are different for real and fake images.

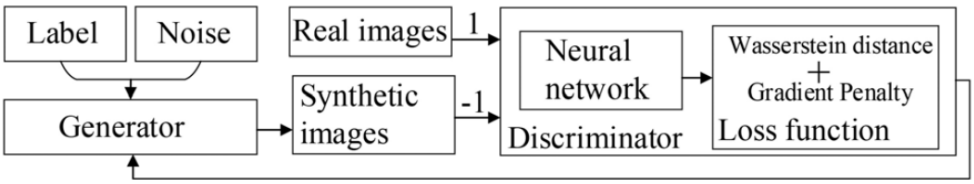


Figure 3.4 Training process of the WGAN-GP. The real images are labeled as "1". The synthetic images are labeled as "-1". The Wasserstein distance and gradient penalty are used in the loss function.

3.2.5 WGAN–GP with Label Smoothing Regularization (WGAN-GP-LSR)

In this paper, we made two changes to the WGAN-GP. The first is that we combined the conditional GAN and the WGAN-GP so that the generator can generate images of specific labels. For the generator, the input is a noise vector and a predefined label. Firstly, the label will be represented following the one-hot encoding method. Then the label will be converted to a vector that has the same size as the noise vector by multiplying a matrix. In practice, we used the built-in embedding function of Keras in which each input integer label is used as the index to access a table that contains all possible vectors. The final input vector is obtained by conducting an elementwise multiply operation between the noise vector and the label vector. The generator is basically a neural network that outputs matrices of the image size with one matrix representing one image. For the discriminator, the output includes the class labels and the validity labels. The second is that LSR is used to modify the loss function of GAN. Compared with L1 and L2 regularization methods which change the weights, LSR directly influences the output of the network through the loss function. At the same time, LSR can increase the robustness of GAN and help avoid model collapse.

In the training of GAN, the most widely used loss function for multiclass classification tasks is the cross-entropy loss as Eq. (3.2),

$$L = -\sum_{i=1}^{N} \log(p(i))q(i) \tag{3.2}$$

where i is the index of the disease type, N is the total number of disease types, $p(i)$ is the predicted probability of the image belonging to class i, $q(i)$ equals to 1 if the label of the image is i; otherwise, $q(i)$ equals to 0.

The minimization of the cross-entropy loss is achieved when the predicted probability of ground-truth classes is maximum. However, if the model assigns full probabilities to ground-truth labels, it is likely to be overfitted. In other words, it will be very easy for CNN to determine the truth-ground classes of the images. It means that the improvement brought by generating additional images for training will be limited. Thus, the regularization is introduced. Regularization is a technique that makes the model less confident such that the model generalizes better.

The LSR method is used in this paper. The objective function of GAN is as Eq. (3.3) [18],

$$L_{LSR} = -(1-\varepsilon)\log(p(y)) - \frac{\varepsilon}{N}\sum_{i=1}^{N}\log(p(i)) \tag{3.3}$$

where ε is a hyperparameter between 0 and 1, i is the index of the disease type, N is the total number of disease types, $p(i)$ is the predicted probability of the image belonging to non-truth ground class i, $p(y)$ is the predicted probability of the image belonging to truth-ground class y.

If ε is equal to 0, Eq. (3.3) is the same as Eq. (3.2) since the second term in Eq. (3.3) becomes 0. The objective is to maximize the predicted probability of the truth-ground class. If ε is equal to 1, the first term equals to 0. The objective is to maximize the summation of the predicted probability of the other non-truth ground classes. Therefore, in addition to maximizing the predicted probability of the truth-ground class, the LSR function also maximizes the predicted probability of the other non-truth ground classes. In the training process of the generator, the synthetic images will learn the same distribution of the probability. In other words, each generated image contains the features of all disease types, which can improve the generalization ability of the model. In practice, a generated image will be assigned with the label of the largest predicted possibility.

3.3 Case Study

To validate the effectiveness of the proposed method, a case study on plant disease classification has been conducted. The dataset contains images of different plant diseases from multiple species. Four experiments were conducted to compare the results. In Experiment I, the CNN was trained without data augmentation. In Experiment II, the CNN was trained with classic data augmentation methods. In Experiment III, the CNN was trained with classic augmentation methods and WGAN-GP. In Experiment IV, the CNN was trained with classic data augmentation methods and WGAN-GP-LSR.

3.3.1 Data Source and Performance Measure

The dataset used in this paper is from www.plantvillage.org. The original dataset contains 43,843 labeled images. To imitate the limited dataset problem, we randomly selected 873 images (i.e., 1.9% of all available images) as the training dataset. For each category, there are 10-28 images for training. We also randomly selected 4,384 images (i.e., 10% of all available images) as the testing dataset. This step was completed by using the train test split function from sklearn package.

As shown in Table 3.1, the images include 14 crop species: Apple, Blueberry, Cherry, Corn, Grape, Orange, Peach, Bell Pepper, Potato, Raspberry, Soybean, Squash, Strawberry, and Tomato. It contains images of 17 fungal diseases, 4 bacterial diseases, 2 mold (oomycete) diseases, 2 viral diseases, and 1 disease caused by a mite. Twelve crop species also have images of healthy leaves that are not visibly affected by a disease [32]. The total number of classes is 38 which includes 12 groups of healthy leaves and 26 groups of diseased leaves.

Four measurements have been used as the performance indicators in this study, i.e., overall accuracy, precision, recall, and F_1 score. The recall, precision and F_1 score can be calculated as in Eq. (3.4)- Eq. (3.6).

$$\text{Recall}_{\ i} = \frac{M_{ii}}{\sum_j M_{ij}} \tag{3.4}$$

$$\text{Precision}_{\ i} = \frac{M_{ii}}{\sum_j M_{ji}} \tag{3.5}$$

$$\text{F}_1 \text{ score}_{\ i} = \frac{2 \times \text{ Recall}_{\ i} \times \text{ Precision}_{\ i}}{\text{Recall}_{\ i} + \text{ Precision}_{\ i}} \tag{3.6}$$

Where M_{ij} is the number of images belonging to the ith category that are predicted to be in the jth category, $\sum_j M_{ij}$ is the number of samples belonging to the ith category, $\text{Recall}_{\ i}$ is the ratio of samples belonging to the ith category that are correctly classified, $\text{Precision}_{\ i}$ is the ratio of samples predicted to be in the ith category that are correctly classified.

Table 3.1 Dataset for classification of plant disease.

Specie	**Class**	$\mathbf{N_1}$	$\mathbf{N_2}$	**Specie**	**Class**	$\mathbf{N_1}$	$\mathbf{N_2}$
	1. Botryospaeria obtuse	10	46	Potato	14. Alternaria solani	16	81
Apple	2. Venturia inaequalis	13	58		15. Phytophthora Infestans	24	58
	3. Gymnosporangium juniperi-virginianae	15	30		H. Healthy	22	15
	A. Healthy	20	162	Squash	16. Erysiphe cichoracearum	28	168
Blueberry	B. Healthy	22	117	Strawberry	17. Diplocarpon earlianum	25	65
Cherry	4. Podosphaera spp.	19	98		I. Healthy	28	40
	C. Healthy	14	76	Raspberry	J. Healthy	19	51
	5. Cercospora zeae-maydis	27	32	Soybean	K. Healthy	28	378
Corn	6. Puccinia sorghi	25	90	Tomato	18. Xanthomonas campestris pv. vesicatoria	27	163
	7. Exserohilum turcicum	24	69		19. Alternaria solani	25	93
	D. Healthy	27	85		20. Phytophthora Infestans	28	142
	8. Guignardia bidwellii	28	94		21. Fulvia fulva	24	70
Grape	9. Phaeomoniella spp.	21	117		22. Septoria lycopersici	28	136
	10. Pseudocercospora vitis	26	90		23. Tetranychus urticae	27	149
	E. Healthy	26	31		24. Corynespora cassiicola	21	121
Orange	11. Candidatus Liberibacter	28	467		25. Mosaic Virus	20	416
Peach	12. Xanthomonas campestris	27	187		26. Yellow leaf curl virus	25	26
	F. Healthy	24	26		L. Healthy	24	136
Pepper	13. Xanthomonas campestris	16	96				
	G. Healthy	22	105				

N_1 represents the number of training images. N_2 represents the number of testing images. The healthy classes are numbered from A to L. The diseased classes are numbered from 1 to 26.

3.3.2 Parameters of Neural Networks

The architectures of the generator and the discriminator are shown in Table 3.2. For the generator, we established a network with a 1000-dimensional vector input. The inputs consist of

two parts, i.e., noise and label. The noise is a vector of 1000 randomly generated variables. The label is converted to a vector of size using the built-in embedding function in Keras. In the function, each integer label is used as the index to access a table that contains all possible vectors. Then the input can be obtained by conducting element-wise multiplication on the two 1000-dimensional vectors. A dense layer is then used to covert the input vector to a vector of size 128×16×16. Through three convolutional layers, the output is an image of dimension 128×128×3. For the discriminator, all input images have been resized to 128×128×3. The real images are assigned with label 1 while the synthetic images are assigned with label -1. There are two output layers. One output layer has one neuron telling whether the input image is real or fake. The other output layer has 38 neurons representing the 38 classes of leaves. The optimizer is RMSprop with the learning rate 0.00005. The objective functions of the discriminator include Wasserstein loss function, gradient penalty function, and cross-entropy function as Eq. (3). We have conducted numerical experiments and analyses to tune the parameter in Eq. (3). The results showed that the quality of the synthetic images of WGAN-GP with LSR was better when was between 0.20 and 0.25. Therefore, the is set as 0.22 in this analysis.

Table 3.2 Architectures of the generator and the discriminator.

Generator		**Discriminator**	
Type	**Output Size**	**Type**	**Output Size**
Dense	$8 \times 8 \times 128$	Conv3-16(stride size = 2)	$64 \times 64 \times 16$
Up sampling	$16 \times 16 \times 128$	Conv3-32(stride size = 2)	$32 \times 32 \times 32$
Conv3-128	$16 \times 16 \times 128$	Zero padding	$33 \times 33 \times 32$
Up sampling	$32 \times 32 \times 128$	Conv3-64(stride size = 2)	$17 \times 17 \times 64$
Conv3-64	$32 \times 32 \times 64$	Conv3-128	$17 \times 17 \times 128$
Up sampling	$64 \times 64 \times 64$	Dense	1
Conv3-32	$64 \times 64 \times 32$	Dense	38
Up sampling	$128 \times 128 \times 32$		
Conv3-3	$128 \times 128 \times 3$		

The convolutional layer parameters are denoted as "Conv (kernel size) - (number of channels)." Each convolutional layer is attached with a batch normalization layer and an activation layer (Leaky ReLU).

As shown in Table 3.3, the CNN used to classify the images is the VGG16 with updated 128×128×3 input [33]. The input layer is based on image RGB color space with a size of 128×128×3. The output layer has 38 neurons representing the 38 classes of leaves. The optimizer is RMSprop. The learning rate is 0.0001. The batch size is 100. All the above networks were built using the Keras framework [34].

Table 3.3 Architectures of the CNN.

Type		**Output Size**	**Type**		**Output Size**
Block 1	Input Layer	$128 \times 128 \times 3$	Block 4	Conv3-512	$16 \times 16 \times 512$
	Conv3-64	$128 \times 128 \times 64$		Conv3-512	$16 \times 16 \times 512$
	Conv3-64	$128 \times 128 \times 64$		Conv3-512	$16 \times 16 \times 512$
	MaxPooling	$64 \times 64 \times 128$		MaxPooling	$8 \times 8 \times 512$
Block 2	Conv3-128	$64 \times 64 \times 64$	Block 5	Conv3-512	$8 \times 8 \times 512$
	Conv3-128	$64 \times 64 \times 128$		Conv3-512	$8 \times 8 \times 512$
	MaxPooling	$32 \times 32 \times 128$		Conv3-512	$8 \times 8 \times 512$
Block 3	Conv3-256	$32 \times 32 \times 128$		MaxPooling	$4 \times 4 \times 512$
	Conv3-256	$32 \times 32 \times 128$		AverPooling	$1 \times 1 \times 512$
	Conv3-256	$32 \times 32 \times 128$		Dense	512
	MaxPooling	$16 \times 16 \times 256$		Dense	38

3.3.3 Experiment Design

To validate the proposed CNN framework, a comparative experiment using 90% of the original dataset (i.e., 39459 images) as train set and 10% (i.e., 4384 images) as the test set. The training accuracy achieved 99.9% while the testing accuracy achieved 99.8%. The results are comparable to the results obtained by Mohanty et al. [35]. It means that this framework can achieve a high prediction accuracy if there are enough data samples. Therefore, the proposed CNN framework can be used as the baseline model for this study. The influence of the CNN framework on the model performance can be ruled out.

Four numerical experiments have been designed, which used 873 training images and 4,384 testing images to keep consistency in the number of testing images. In Experiment I, the CNN is trained using the real dataset without any data augmentation. In Experiment II, the CNN is

trained using real images with classic data augmentation methods. The classic augmentation methods include 360 rotation range, 0.3 width shift range, 0.3 height shift range, 0.3 zoom range, horizontal flip, and vertical flip. In Experiment III, the CNN is trained using the classic augmented data and the synthetic images generated by WGAN-GP without LSR. In each epoch, we use the trained generator to generate 30 new synthetic images for each category. In Experiment IV, the CNN is trained using the dataset generated by the proposed method. The training process is the same as that of the third experiment. It should be noted that, in Experiment III and IV, WGAN-GP is trained using the classic augmented data and then be used to generate synthetic images.

The number of images used for training in each epoch is shown in Table 3.4. In Experiment I, the 873 images used in each epoch are the same. In Experiment II, III and IV, the classic augmented images and synthetic images used in each epoch are new images that are generated randomly by the classical data augmentation methods and WGAN-GP, respectively. This paper implements the classic augmentation by using the ImageDataGenerator function from Keras package which replaces the original batch with the new, randomly transformed batch. Therefore, in Experiment II, III and IV, the number of original images used in each epoch is 0. The generator ran in parallel to the model for improved efficiency. For instance, this allows us to do real-time data augmentation on images on CPU in parallel to training our model on GPU.

Table 3.4 Number of images used for training in each epoch.

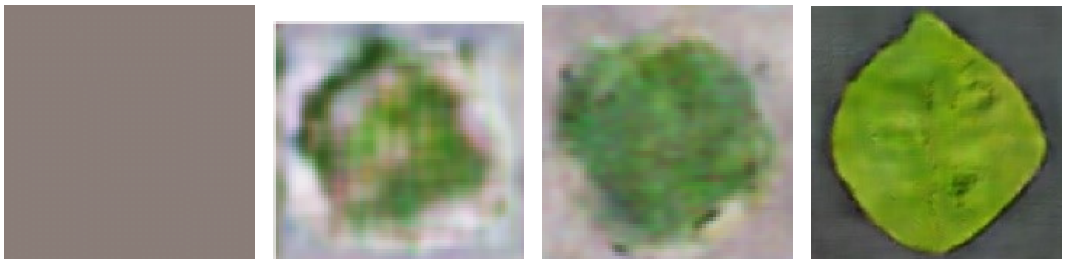
Methods	**# of original images**	**# of classic augmented images**	**# of synthetic images**
Experiment I	873	0	0
Experiment II	0	873	0
Experiment III	0	873	30*38
Experiment IV	0	873	30*38

To eliminate the influence of training time, the models are trained until the curve of training accuracy converges. This means the model performance cannot be improved by increasing the

training time. Therefore, the number of epochs is set as 700. All experiments including the comparative experiment used the same testing dataset.

3.3.4 Results and Comparisons

The most important process is the training of the GAN. The training effectiveness of WGAN-GP-LSR can be illustrated by Figure 3.5. At the beginning, the output of the generator is just white noise. After 12,000 iterations, the outline of the leaf can be identified visually. At the 22,000th iteration, the shape of the leaf is much clearer. Figure 3.6 is the train loss curve of WGAN-GP-LSR. It can be seen that after 20,000, the Wasserstein distance, which is used to measure the distance between generated images and real images, converges. Figure 3.7(a) shows the real images drawn from 38 categories while Figure 3.7(b) shows the 38 samples generated by the regularized GAN. Each sample belongs to one unique class.



(a) 0 (b) 2,000 (c) 12,000 (d) 22,000

Figure 3.5 Synthetic images in different training stages of WGAN-GP-LSR (# of iterations).

It can be found that the synthetic images look different from the original ones. There are two reasons for this. The first reason is that the synthetic images also contain information from other classes because of LSR. For example, for a classification problem of five classes, the ideal output of discriminator for a sample of class 1 should be [1,0, 0, 0, 0]. However, to increase the generalization ability of the model, the ideal output is expected to be [0.6, 0.1, 0.1, 0.1, 0.1]. This means the generated images also have small probabilities to be classified as other non-ground-truth classes. The second reason is that the WGAN-GP cannot generate perfect

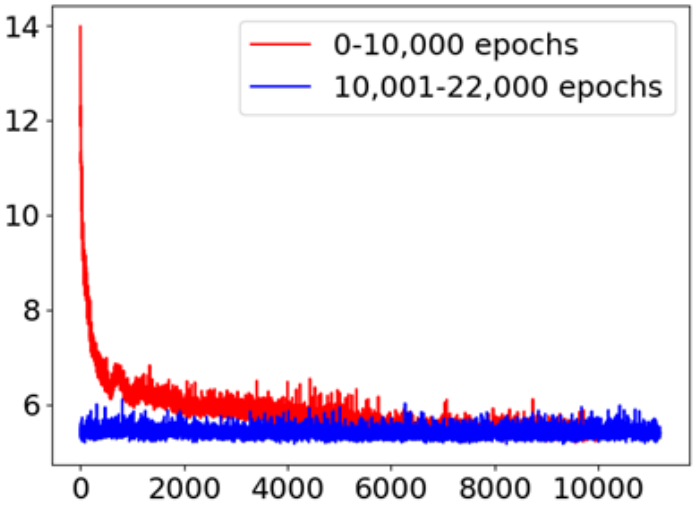
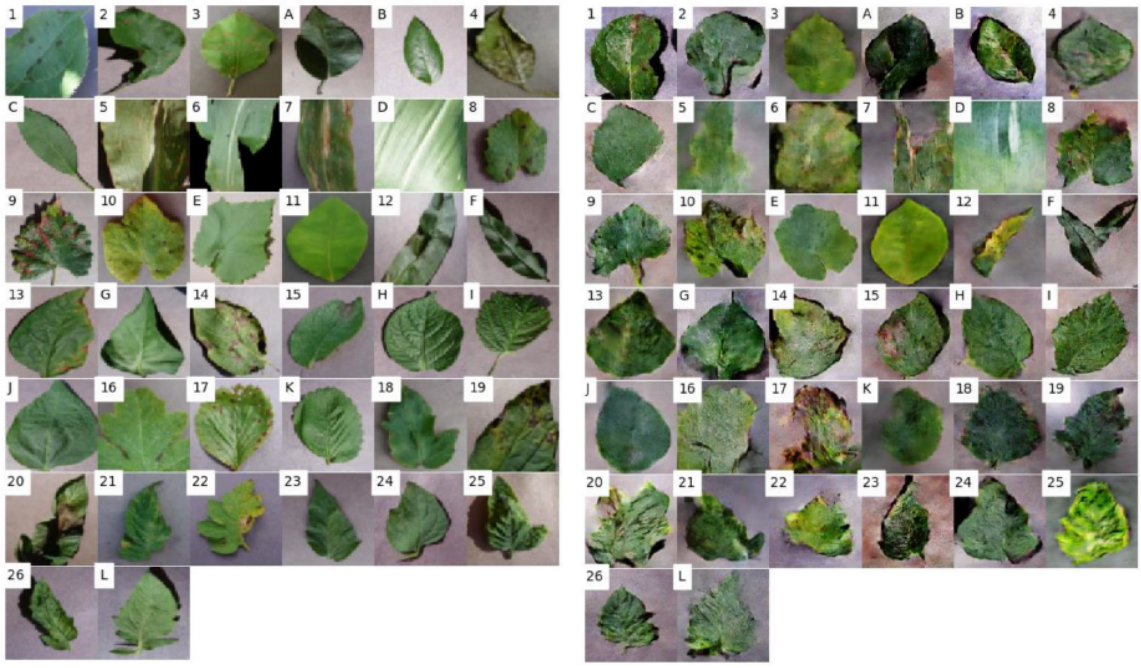


Figure 3.6 Train loss of WGAN-GP-LSR.

images that restore all details of real images due to the limited training set. The discriminator of WGAN only focuses on some specific regions (e.g., leaf shape, yellow spot, hole) that it can extract features from. Therefore, some information, such as background color and contrast degree, may be lost. However, the neural network can extract the right features to make predictions. The trained generator is used to generate additional images. Those images are mixed with real images and used as the input of the CNN.

The results of the four experiments are shown in Figure 3.8. From Figure 3.8(a), it can be found that after about 60 epochs, the training accuracy in Experiment I is close to 1 while the test accuracy is only about 60%. This is an indicator that the model is overfitted. It can be seen from Figure 3.8(b) that after using the classic data augmentation methods, the test accuracy in Experiment II is about 80%, which is 20% higher than that in Experiment I. Figure 3.8(c) shows the results of training CNN with classic data augmentation methods and synthetic data augmentation. After introducing the WGAN-GP, the test accuracy is improved by 1.9%. It proves that the synthetic images can increase the diversity of the dataset and improve the prediction accuracy. Since there are more training images, the curve of test accuracy is more



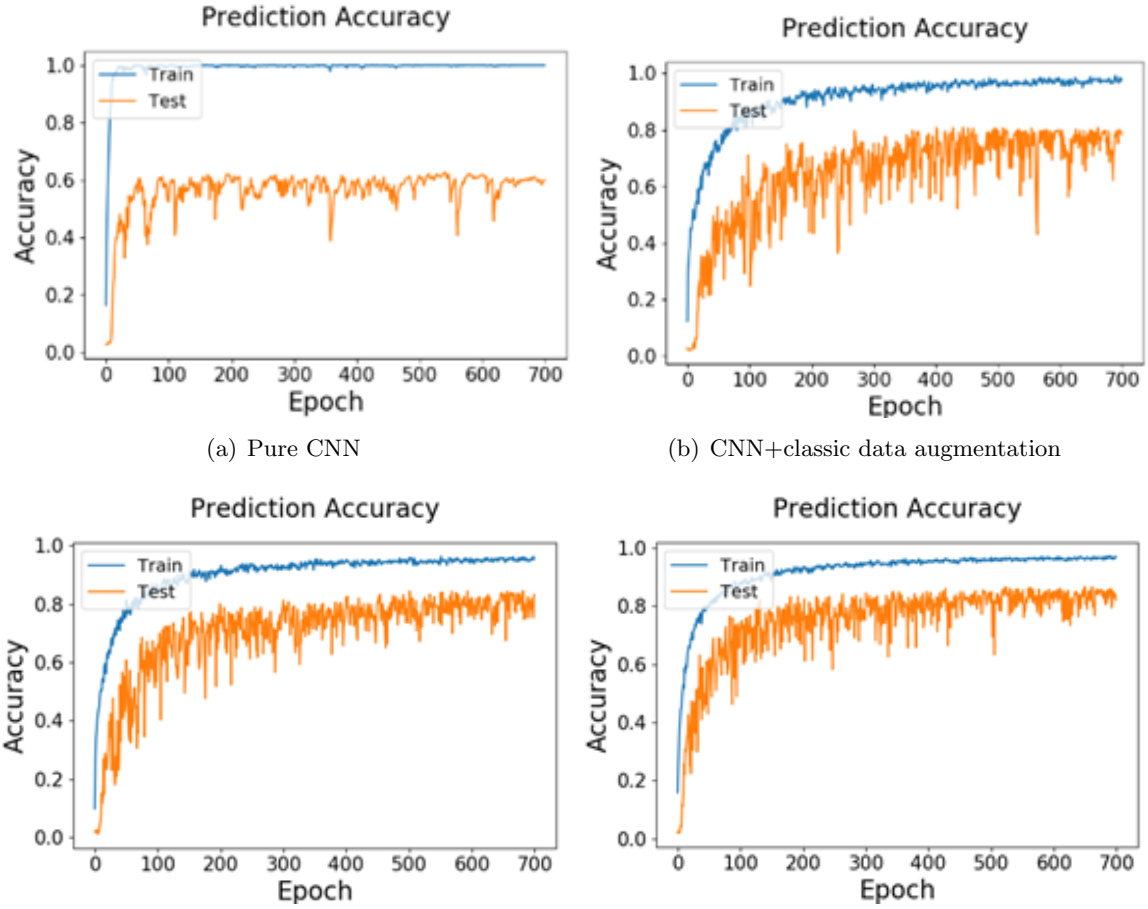
(a) Original images (b) Samples generated by the WGAN-GP-LSR

Figure 3.7 Original images and generated image samples. The images at the same location belong to the same class. The healthy classes are numbered from A to L. The diseased classes are numbered from 1–26.

stable than that in Experiment I and Experiment II. The results of Experiment IV is shown in Figure 3.8(d). Compared to using WGAN-GP without LSR, the proposed method can improve the test accuracy by 2.1%, which validates the effectiveness of LSR.

Table 3.5 lists the training accuracy and test accuracy of the above four experiments. Compared to using CNN only, the proposed method improves the test accuracy by 21.6%. Compared to using CNN with classic data augmentation methods, the proposed method can improve the test accuracy by 4.2%. Compared to using CNN with classic data augmentation method and WGAN-GP, the proposed method can improve the test accuracy by 2.3%.

Table 3.6 includes the recall, precision, and F_1 scores of 26 diseases. The top-5 F_1 scores achieved by the proposed method are 0.91 on disease type 9 (Grape Phaeomoniella Spp.), 0.98 on disease type 11 (Orange Candidatus Liberibacter), 0.91 on disease type 14 (Potato Alternaria Solani), 0.91 on disease type 16 (Squash Erysiphe Cichoracearum) and 0.98 on disease type 25



(a) Pure CNN

(b) CNN+classic data augmentation

(c) CNN+ data augmentation +WGAN-GP,

(d) CNN+ data augmentation+WGAN-GP-LSR

Figure 3.8 Results of the four numerical experiments.

(Tomato Mosaic Virus). Compared to using the CNN only, the advantages of the proposed method are dominant in terms of F_1 score in almost all classes (i.e., 24 out of 26). For example, the proposed method improves F_1 scores by 0.38 on disease type 8 (Grape Guignardia Bidwellii), 0.57 on disease type 15 (Potato Phytophthora Infestans) and 0.38 on disease type 21 (Tomato Fulvia Fulva). The proposed method outperforms the CNN with classic data augmentation on most of the disease classes (i.e., 23 out of 26). Compared to using WGAN-GP without LSR, the proposed method performs much better on disease type 4 (Cherry Podosphaera Spp.) and disease type 14 (Potato Alternaria Solani). The average F_1 score of the proposed method (i.e., 0.77) is

Table 3.5 Comparisons among four methods.

Methods	**Training accuracy**	**Test accuracy**
Pure CNN	100%	60.40%
CNN + classic data augmentation	90.08%	80.57%
CNN + classic data augmentation + WGAN-GP	98.23%	82.41%
Proposed method (CNN + classic data augmentation + WGAN-GP-LSR)	97.84%	84.78%

higher than that of the CNN with classic data augmentation method (i.e., 0.71) and that of using WGAN-GP without LSR (i.e., 0.75).

When comparing the recall and the precision of each disease type, specific patterns of the models can be observed. For example, the difference between the recall and the precision of the disease type 10 (Grape Pseudocercospora Vitis) is significantly different for all four models. The recall is $0.51 \sim 0.6$ while the precision is $0.84 \sim 0.98$. This means only a small number of images that have type 10 disease are classified as disease type 10. However, most of the images predicted that are classified to be type 10 are correctly labeled. The model might be confused between disease type 10 and other diseases, so it set a high standard for the classification of type 10. Therefore, the prediction of disease type 10 is highly reliable but the sensitivity of the model is low since the false negative predictions are high.

Since the objective of the training process is to improve the total prediction accuracy over all disease classes, it is not guaranteed that the proposed method will outperform other models in all categories. For example, the F_1 score of disease type 3 (Apple Gymnosporangium juniperi-virginianae) in Experiment IV is much lower than that of other diseases. The reason is that the disease is more likely to be predicted as corn fungus diseases by the model. The comparison between the recall and the precision of each disease type can help to gain additional insights into the models and make the right decision according to different situations.

Table 3.7 lists the recall, precision and F_1 scores of 12 healthy groups. The average F_1 scores in the four experiments are 0.46, 0.76, 0.78 and 0.81, separately. However, all of the four models

Table 3.6 Recall, precision and F_1 scores of 26 diseases (R: Recall; P: Precision; F: F_1 score).

No.	**Experiment I**			**Experiment II**			**Experiment III**			**Experiment IV**		
	R	**P**	**F**	**R**	**P**	**F**	**R**	**P**	**F**	**R**	**P**	**F**
1	0.17	0.89	0.29	0.48	0.54	0.51	0.33	0.88	0.48	0.46	0.60	0.52
2	0.59	0.61	0.60	0.86	0.77	0.81	0.66	0.84	0.74	0.93	0.72	0.81
3	0.37	0.41	0.39	0.50	0.88	0.64	0.70	0.75	0.72	0.30	0.56	0.39
4	0.41	0.85	0.55	0.88	0.43	0.58	0.43	0.93	0.59	0.71	0.96	0.82
5	0.50	0.42	0.46	0.69	0.56	0.62	0.47	0.68	0.56	0.56	0.72	0.63
6	0.53	0.80	0.64	0.89	0.90	0.89	0.96	0.73	0.83	0.92	0.86	0.89
7	0.87	0.65	0.75	0.78	0.76	0.77	0.61	0.91	0.73	0.81	0.79	0.80
8	0.28	0.40	0.33	0.94	0.45	0.61	0.69	0.87	0.77	0.74	0.69	0.71
9	0.92	0.55	0.69	0.67	0.85	0.75	0.91	0.91	0.91	0.91	0.91	0.91
10	0.51	0.84	0.63	0.62	0.95	0.75	0.58	0.91	0.71	0.60	0.98	0.74
11	0.96	0.66	0.78	0.93	0.98	0.95	0.96	0.95	0.96	0.99	0.97	0.98
12	0.66	0.93	0.77	0.95	0.64	0.77	0.90	0.90	0.90	0.95	0.80	0.87
13	0.73	0.43	0.54	0.85	0.55	0.67	0.81	0.71	0.76	0.93	0.53	0.68
14	0.52	0.75	0.61	0.84	0.65	0.74	0.35	0.93	0.50	0.89	0.94	0.91
15	0.12	0.78	0.21	0.79	0.54	0.64	0.72	0.95	0.82	0.69	0.89	0.78
16	0.57	0.98	0.72	0.94	0.88	0.91	0.94	0.82	0.88	0.86	0.97	0.91
17	0.88	0.78	0.83	0.71	0.63	0.67	0.86	0.84	0.85	0.89	0.75	0.82
18	0.28	0.62	0.38	0.62	0.86	0.72	0.84	0.79	0.81	0.66	0.98	0.79
19	0.22	0.65	0.32	0.53	0.52	0.52	0.76	0.57	0.65	0.62	0.66	0.64
20	0.43	0.59	0.50	0.62	0.73	0.67	0.67	0.77	0.72	0.70	0.75	0.73
21	0.29	0.54	0.37	0.54	0.78	0.64	0.81	0.92	0.86	0.81	0.70	0.75
22	0.65	0.57	0.61	0.52	0.89	0.66	0.94	0.52	0.67	0.76	0.70	0.73
23	0.54	0.67	0.60	0.63	0.82	0.71	0.58	0.97	0.73	0.72	0.92	0.81
24	0.60	0.45	0.52	0.31	0.73	0.44	0.93	0.53	0.67	0.81	0.67	0.73
25	0.95	0.74	0.83	0.89	0.98	0.94	0.97	0.89	0.92	0.99	0.97	0.98
26	0.88	0.62	0.73	0.92	1.00	0.96	0.85	0.92	0.88	0.92	0.80	0.86

do not perform well for the classification of potato healthy leaves. Since there are only 15 testing images in this group, the reason might be that the distribution of the training set is not close to that of the testing set. Except for this, the F_1 scores of most groups in Experiment II, III and IV are greater than 0.75.

Table 3.7 Recall, precision and F_1 scores of 12 healthy groups (R: Recall; P: Precision; F: F_1 score).

Specie	**Experiment I**			**Experiment II**			**Experiment III**			**Experiment IV**		
	R	**P**	**F**	**R**	**P**	**F**	**R**	**P**	**F**	**R**	**P**	**F**
Apple	0.37	0.7	0.49	0.93	1.00	0.9	0.81	0.92	0.86	0.89	0.94	0.91
Blueberry	0.44	0.7	0.55	0.91	1.00	0.95	0.76	0.97	0.85	0.84	0.8	0.82
Cherry	0.74	0.35	0.48	0.88	0.97	0.92	0.97	0.71	0.82	0.92	0.84	0.88
Corn	0.67	0.72	0.69	1.00	0.97	0.98	0.99	0.89	0.94	0.91	1.00	0.95
Grape	0.65	0.50	0.57	0.74	0.93	0.82	0.81	0.62	0.70	0.77	0.71	0.74
Peach	0.42	0.55	0.48	0.96	0.71	0.82	0.88	0.82	0.85	0.65	0.94	0.77
Pepper	0.73	0.17	0.28	0.79	0.83	0.81	0.86	0.68	0.76	0.94	0.8	0.86
Potato	0.13	0.22	0.16	0.13	0.67	0.22	0.40	1.00	0.57	0.53	0.47	0.50
Raspberry	0.37	0.76	0.50	0.90	0.68	0.77	0.27	1.00	0.43	0.8	0.85	0.82
Soybean	0.44	0.77	0.56	0.98	0.99	0.98	0.94	0.92	0.93	0.92	0.86	0.89
Strawberry	0.15	0.38	0.22	0.53	0.78	0.63	0.80	0.71	0.75	0.6	0.8	0.69
Tomato	0.54	0.70	0.61	0.85	0.92	0.88	0.86	0.97	0.91	0.86	0.97	0.91

3.4 Conclusion

Plant disease recognition plays an important role in disease detection, mitigation, and management. Even though some deep learning methods have achieved good results in plant disease classification, the problem of the limited dataset is overlooked. In practice, it is time-consuming to collect and annotate data. The performance of CNN will drop dramatically if there is not enough training data. Therefore, a method for plant disease recognition under the limited training dataset is necessary.

In this paper, a CNN has been built for plant disease recognition, which can recognize multiple species and diseases. To address the overfitting problem caused by the limited training

dataset, a GAN-based approach is proposed. The LSR method is also employed, which works by adding a regularization term to the loss function.

The experiments show that the proposed method can improve the prediction accuracy by 4.2% than the CNN with the classic data augmentation method. Compared with using the CNN only, the proposed method can improve the prediction accuracy by 24.4%. Compared with using the WGAN-GP without LSR, the proposed method can improve the prediction accuracy by 2.3%. Based on our work, plant disease classification can be conducted under the limited training dataset, which will bring benefits to the rapid diagnosis of plant diseases.

It should be noted that this proposed plant disease classification method is subject to a few limitations which suggest future research directions. First, significant computational resources are needed to train the GAN and generate new labeled images for training. This problem can be addressed using pre-trained models. Next, the proposed method still needs enough images to train the GAN. If the size of dataset is very small, it is not able to extract enough information to generate new labeled images. One potential solution to this is to introduce transfer learning techniques. Last, in this paper, we only used one CNN framework. In future, we will try different CNN frameworks and investigate the relationship between the size of the real image dataset and the effectiveness of the proposed method.

References

[1] SS Abu-Naser, KA Kashkash, and M Fayyad. Developing an expert system for plant disease diagnosis. *Journal of Artificial Intelligence*, 3(4):269–276, 2010.

[2] Richard N Strange and Peter R Scott. Plant disease: a threat to global food security. *Annu. Rev. Phytopathol.*, 43:83–116, 2005.

[3] Srdjan Sladojevic, Marko Arsenovic, Andras Anderla, Dubravko Culibrk, and Darko Stefanovic. Deep neural networks based recognition of plant diseases by leaf image classification. *Computational intelligence and neuroscience*, 2016, 2016.

[4] Jayme GA Barbedo. Factors influencing the use of deep learning for plant disease recognition. *Biosystems engineering*, 172:84–91, 2018.

[5] Konstantinos P Ferentinos. Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145:311–318, 2018.

[6] Sindhuja Sankaran, Ashish Mishra, Reza Ehsani, and Cristina Davis. A review of advanced techniques for detecting plant diseases. *Computers and Electronics in Agriculture*, 72(1):1–13, 2010.

[7] Jayamala K Patil and Raj Kumar. Advances in image processing for detection of plant diseases. *Journal of Advanced Bioinformatics Applications and Research*, 2(2):135–141, 2011.

[8] YG Naresh and HS Nagendraswamy. Classification of medicinal plants: an approach using modified lbp with symbolic representation. *Neurocomputing*, 173:1789–1797, 2016.

[9] Shanwen Zhang and Zhen Wang. Cucumber disease recognition based on global-local singular value decomposition. *Neurocomputing*, 205:341–348, 2016.

[10] Andreas Kamilaris and Francesc X Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90, 2018.

[11] Yang Lu, Shujuan Yi, Nianyin Zeng, Yurong Liu, and Yong Zhang. Identification of rice diseases using deep convolutional neural networks. *Neurocomputing*, 267:378–384, 2017.

[12] Mrunmayee Dhakate and AB Ingole. Diagnosis of pomegranate plant diseases using neural network. In *2015 fifth national conference on computer vision, pattern recognition, image processing and graphics (NCVPRIPG)*, pages 1–4. IEEE, 2015.

[13] Mostafa Mehdipour Ghazi, Berrin Yanikoglu, and Erchan Aptoula. Plant identification using deep neural networks via optimization of transfer learning parameters. *Neurocomputing*, 235:228–235, 2017.

[14] Juncheng Ma, Keming Du, Feixiang Zheng, Lingxian Zhang, Zhihong Gong, and Zhongfu Sun. A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network. *Computers and electronics in agriculture*, 154:18–24, 2018.

[15] Guillermo L Grinblat, Lucas C Uzal, Mónica G Larese, and Pablo M Granitto. Deep learning for plant identification using vein morphological patterns. *Computers and Electronics in Agriculture*, 127:418–424, 2016.

[16] Jayme Garcia Arnal Barbedo. Plant disease identification from individual lesions and spots using deep learning. *Biosystems Engineering*, 180:96–107, 2019.

[17] Chitta Baral, Olac Fuentes, and Vladik Kreinovich. Why deep neural networks: a possible theoretical explanation. In *Constraint programming and decision making: theory and applications*, pages 1–5. Springer, 2018.

[18] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[19] Žiga Emeršič, Dejan Štepec, Vitomir Štruc, and Peter Peer. Training convolutional neural networks with limited training data for ear recognition in the wild. *arXiv preprint arXiv:1711.09952*, 2017.

[20] Guosheng Hu, Xiaojiang Peng, Yongxin Yang, Timothy M Hospedales, and Jakob Verbeek. Frankenstein: Learning deep face representations using small data. *IEEE Transactions on Image Processing*, 27(1):293–303, 2017.

[21] Jian Guo and Stephen Gould. Deep cnn ensemble with data augmentation for object detection. *arXiv preprint arXiv:1506.07224*, 2015.

[22] Jeremie Papon and Markus Schoeler. Semantic pose using deep networks trained on synthetic rgb-d. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 774–782, 2015.

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[24] Haseeb Nazki, Sook Yoon, Alvaro Fuentes, and Dong Sun Park. Unsupervised image translation using adversarial networks for improved plant disease recognition. *Computers and Electronics in Agriculture*, 168:105117, 2020.

[25] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4753–4762, 2016.

[26] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[27] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[28] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[29] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[30] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[31] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[32] David Hughes, Marcel Salathé, et al. An open access repository of images on plant health to enable the development of mobile disease diagnostics. *arXiv preprint arXiv:1511.08060*, 2015.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] François Chollet et al. Keras, 2015.

[35] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

CHAPTER 4. A GATED RECURRENT UNITS (GRU)-BASED MODEL FOR EARLY DETECTION OF SOYBEAN SUDDEN DEATH SYNDROME THROUGH TIME-SERIES SATELLITE IMAGERY

Luning Bi[1], Guiping Hu[1], Muhammad Mohsin Raza[2], Yuba Kandel[2],
Leonor Leandro[2], and Daren Mueller[2]
[1]Department of Industrial and Manufacturing Systems Engineering, Iowa State University
[2]Department of Plant Pathology and Microbiology, Iowa State University
Modified from a manuscript published in *Remote Sensing*

Abstract

In general, early detection and timely management of plant diseases are essential for reducing yield loss. Traditional manual inspection of fields is often time-consuming and laborious. Automated imaging techniques have recently been successfully applied to detect plant diseases. However, these methods mostly focus on the current state of the crop. This paper proposes a Gated Recurrent Units (GRU) based model to predict soybean sudden death syndrome (SDS) disease development. To detect SDS at a quadrat level, the proposed method uses satellite imageries collected from PlanetScope as the training set. The pixel image data includes the spectral bands of red, green, blue and near-infrared (NIR). Data collected during the 2016 and 2017 soybean growing seasons were analyzed. Instead of using individual static imagery, the GRU-based model converts the original imagery into time-series data. SDS predictions were made on different data scenarios and the results were compared with fully connected deep neural network (FCDNN) and XGBoost methods. The overall test accuracy of classifying healthy and diseased quadrates in all methods was above 76%. The test accuracy of FCDNN and XGBoost were 76.3%-85.5% and 80.6%-89.2%, separately, while the test accuracy of the GRU-based model

was 82.5%-90.4%. The calculation results show that the proposed method can improve the detection accuracy by up to 7% with time-series imagery. Thus, the proposed method has the potential to predict SDS at a future time when enough number of historical imageries are available.

Keywords: soybean disease, sudden death syndrome, gated recurrent unit, remote sensing, satellite imagery, disease detection

4.1 Introduction

Soybean (Glycine max L. Merrill) diseases can have a significant impact on production and profits [1]. During the years 2015 to 2019, soybean diseases are responsible for losses of around 8.99% of the production potential in the U.S., which equates to an average of $ 3.8 billion annually [2]. Sudden death syndrome (SDS) is one of the most damaging soybean diseases found throughout most of the soybean production area in the United States. SDS is caused by a soilborne fungus Fusarium virguliforme (Fv) that causes root rot and foliar symptoms that typically become visible during reproductive stages [3]. Visual assessment of SDS requires intensive crop scouting that is time-consuming and labor-intensive. Therefore, an automated method for the detection of SDS is necessary.

Timely detection and management of plant diseases are essential for reducing yield loss. Remote sensing technology has proven to be a practical, noninvasive tool for monitoring crop growth and assessing plant health [4, 5]. Sensing instruments can record radiation in various parts of the electromagnetic spectrum, ultraviolet, visible, near-infrared (NIR) and thermal infrared, to name a few [6]. Healthy and diseased plant canopies absorb and reflect incident sunlight differently due to changes in leaf and canopy morphology and chemical constituents [7, 8]. These changes can alter the optical spectra, such as a decrease in canopy reflectance in the near-infrared band and an increase of reflectance in the red band [7]. Some widely used methods include thermography [9, 10, 11], fluorescence measurements [12, 13, 14] and hyperspectral techniques [15, 16, 17].

In the past few years, several studies have been conducted for the detection of plant diseases [18, 19, 20]. For example, Durmus et al. used RGB cameras for disease detection on tomato leaves [18]. In another study, Gold et al. used lab-based spectroscopy to detect and differentiate late and early blight diseases on potato leaves [19]. Specific to SDS, Bajwa, et al. [21] and Herrmann, et al. [3] used handheld and tractor-mounted sensors, respectively, for SDS detection. These successful applications help ease the demand for expert resources and reduce the human errors in the visual assessment and eventually management of plant diseases.

However, most of these methods are near-sensing techniques that focus on individual plants. In practice, it is more reasonable and efficient to diagnose plant diseases like SDS from the quadrat level. There are several options for collecting imagery for the detection of plant diseases at the quadrat level, including unmanned aerial vehicle (UAV) [22, 23], tractor-mounted tools [20], and satellite imagery [24]. Satellite imagery is typically less expensive, covers wide ground swath, and can provide temporal flexibility because the fields can be continuously monitored non-destructively. The frequency of image collection is determined by the number of satellites that pass that field [25]. Although the imagery from UAVs and tractor-mounted tools are often higher resolution at the quadrat level, these tools are expensive, cumbersome for farmers to operate and maintain in a commercial system and often lack spatial information. Satellite imagery, on the other hand, contains spatial information, comes preprocessed and continuously improving in resolution.

In addition to quality data, efficient and accurate analysis of the sensor data is essential for accurate detection of plant diseases. For the analysis of visible sensing information several machine learning methods have been used. Common methods that have been used in plant disease diagnosis using imageries are convolutional neural networks (CNNs) and random forest. Different from full connected neural networks, CNNs have two special layer types. The convolutional layers extract features from the input imageries. The pooling layers reduce the dimensionality of the features. Dhakate et al. used a CNN for the recognition of pomegranate diseases with 90% overall accuracy [26]. Ferentinos developed CNN models to classify the healthy and diseased plants with

99.5% success [27]. Polders et al. used fully convolutional neural networks to detect potato virus in seed potatoes [28]. However, in our case, the resolution of satellite imagery was 3 m x 3 m, which means the satellite imagery of each generalized quadrat only contained several pixels. CNN or other methods are less suitable for this task. How to deal with low pixel-level remote sensing data is important for improving classification accuracy. Random forest was also used for disease detection and classification [29]. Samajpati et al. used a random forest classifier to classify different apple fruit diseases [30]. Chaudhary et al. proposed an improved random forest method, which achieved 97.8% for multi-class groundnut disease dataset [29]. Although random forest is popular for its easy implementation and high efficiency on large datasets, its performance is influenced by the hyperparameter choices, such as random seeds and the number of variables [31].

Most of the current plant disease identification methods use field imageries at a single time point to identify the contemporary status of the disease. The use of temporal image sequences can help improve detection accuracy. For example, a recurrent neural network (RNN) was designed for solving the multivariate time-series prediction problem [32]. However, RNN is faced with gradient vanishing/exploding problems. As an improved version of RNN, long short term memory model (LSTM) is used for its successful application to natural language modeling [33]. Compared with RNN, LSTM has more gates that can control the reset of the memory and the update of the hidden states. Turkoglu et al. proposed an LSTM-based CNN for the detection of apple diseases and pests, which scored 96.1% [34]. Namin et al. utilized a CNN-LSTM framework for plant classification of various genotypes as well as the prediction of plant growth and achieved an accuracy of 93% [35].Although LSTM has alleviated the gradient vanishing/exploding problem of RNNs, the training speed of LSTM is much slower due to the increased number of parameters. To solve this issue, Chung, et al. introduced the gated recurrent unit (GRU) in 2014 [36]. Since GRU only has two gates (i.e., reset gate and update gate) and uses the hidden state to transfer information, its training speed is much faster. Jin et al. used a deep neural network which combined CNN and GRU to classify wheat hyperspectral pixels and obtained an accuracy of 0.743 [37].

The objective of this paper is to detect SDS in soybean quadrats using 3 m resolution satellite imagery. GRU-based model was compared in different scenarios (i.e., percentage of diseased quadrats) to the most popular neural network and tree-based methods, namely, fully connected deep neural network (FCDNN) and XGBoost.

This paper is organized as follows. Section 4.2 introduces the dataset and methods used. Section 4.3 includes a case study, the calculation results, and comparisons of the three methods. The paper concludes with the summary, findings, and future research directions in Section 4.4, Section 4.5 and Section 4.6.

4.2 Materials and Methods

Data were collected in 2016 and 2017 from an ongoing soybean field experiment located at the Iowa State University Marsden Farm, in the Boone County, Iowa (Figure 4.1) [38, 39, 40]. This study site was chosen because soybean plots in this experiment have consistently displayed a range of SDS levels since 2010 [41]. In the trial site, there were three cropping systems, 2-year, 3-year and 4-year crop rotations, which is represented by using one-hot encoding, i.e., "100", "010" and "001". In the 2-year cropping rotation, corn and soybean were planted in rotation with synthetic fertilizers at conventional rate. In the 3-year cropping rotation, corn, soybean, oat and red clover were planted in rotation with composted cattle manure and synthetic fertilizers at reduced rates. In the 4-year cropping rotation, corn, soybean, oat and alfalfa, and alfalfa were planted in rotation with composted cattle manure and synthetic fertilizers at reduced rates. More information about the experiment can be found in these studies [38, 39, 40]. In each year, 240 soybean quadrats (3 m wide $\times$ 1.5 m long) were marked to collect disease data.

4.2.1 Data Processing

Figure 4.2 illustrates our method for data collection, data processing and analysis for the detection of SDS in this study. This methodology has been explained in detail.

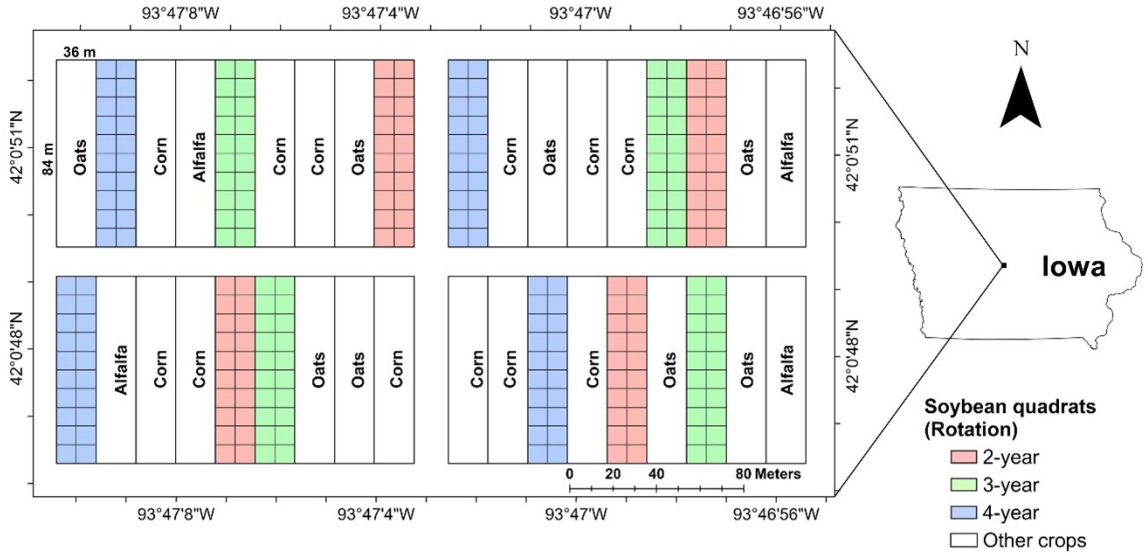


Figure 4.1 Experimental layout of the Marsden Farm located in Boone County, Iowa. The experiment was designed using a randomized complete block design with four blocks and each block has nine main plots. Each soybean plot was divided into 20 quadrats (8 m × 9 m, shown as square grids).

Satellite images were collected on 5 July 2016, 9 July 2016, 20 July 2016, 5 August 2016, 21 August 2016, 31 August 2016, 5 July 2017, 9 July 2017, 20 July 2017, 2 August 2017, 18 August 2017 and 23 August 2017. In addition to spectral information, the dataset also included ground-based crop rotation information that is an explanatory variable. The data source of satellite imagery is PlanetScope (https://www.planet.com/) satellite operated by Planet Labs (San Francisco, CA), a private imaging company. PlanetScope satellite imagery comes with four bands, including red (590–670 nm), green (500–590 nm), blue (455–515 nm) and NIR (780–860 nm). Soybean quadrats were generalized to large quadrats (8.6 m W × 9.1 m L) for data extraction from images and subsequent data analysis. As such, the imagery of each quadrat has 6–9 pixels.

The number of total plants and plants showing foliar symptoms of SDS was counted in each quadrat to calculate the disease incidence on a 0 to 100% scale, based on the percentage of diseased plants. The distribution of SDS incidence is shown in Figure 4.3. It can be observed that

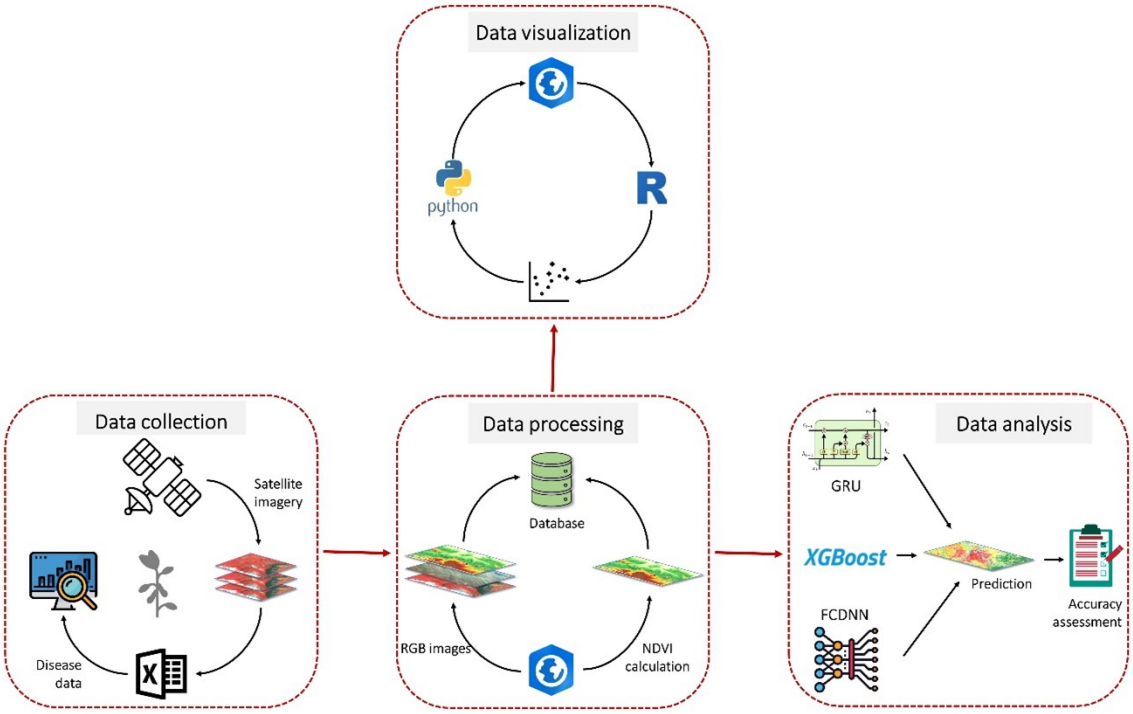


Figure 4.2 Flow diagram of the data collection, processing and analysis we employed in this study for sudden death syndrome (SDS) detection. We divided our methodology into four major steps including data collection, data processing, data visualization and analysis.

in 2016, SDS incidence in more than half of the 240 quadrats was less than 5% and the SDS incidence of the rest quadrats ranged from 5% to 100%. In 2017, the SDS incidence of most quadrats was below 5%. In this paper, if a quadrat had an SDS incidence above 5%, it was considered as diseased (positive); otherwise, it was considered as healthy (negative). The inspection dates used for the analysis were 27 July 2016, 5 August 2016, 22 August 2016 and 29 August 2016. Since the visual disease scores on 27 July 2016 were all zeros, all the imagery collected before that date was labeled as healthy. The human visual ratings recorded on 5 August 2016 were mapped to the imagery collected on the same date. The visual score on 22 August 2016 was mapped to the imagery collected on 21 August 2016, while the SDS rating recorded on 31 August 2016 was mapped to the imagery collected on 29 August 2016 Similarly, in 2017, the

human visual ratings recorded on 17 August 2017 were mapped to the imagery collected on 18 Aug and the human visual ratings recorded on 24 August 2017 were mapped to the imagery collected on 23 August 2017. There was only one diseased quadrat on 5 August 2016. In 2016, all quadrats were healthy before 5 August 2016. In 2017, all quadrats were healthy before 17 August 2017. The aim was to determine whether the quadrat had SDS or not.

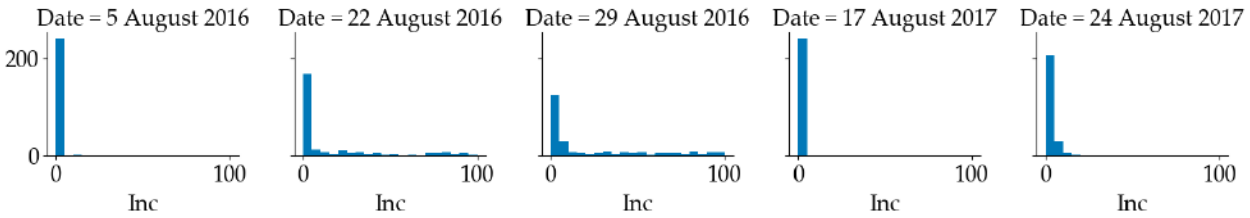


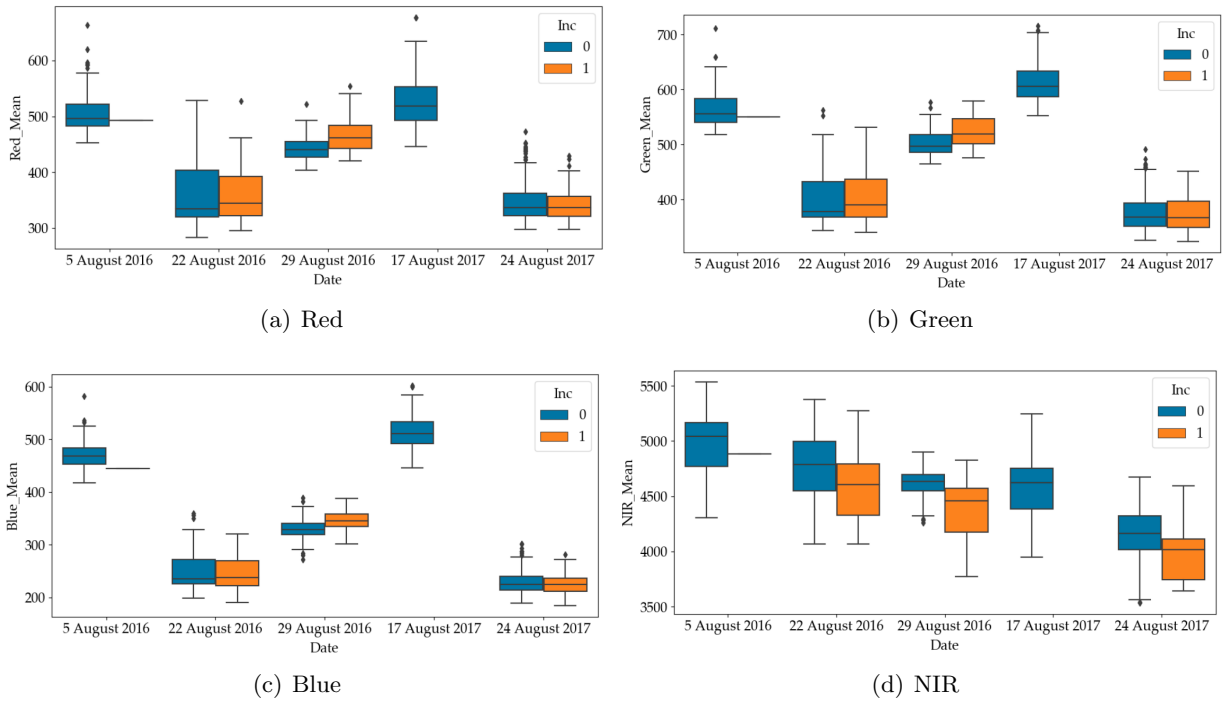
Figure 4.3 Distribution of sudden death syndrome (SDS) incidence in soybean quadrats (Inc: Incidence) at Marsden Farm.

The mean and variance of the RGB, normalized difference vegetation index (NDVI) and NIR information of each quadrat were used as predictors for two reasons. First, the number of pixels varied from quadrat to quadrat. Second, the SDS rating represented the entire quadrat. Some information could be lost if each pixel was used separately. The crop rotation information was used as a categorical variable. Therefore, there were a total of 12 variables for each quadrat, which included the year, crop rotation, the mean values, and the variance values of RGB and NIR bands and NDVI. The NDVI is calculated as Eq. (4.1).

$$\text{NDVI} = \frac{\text{NIR} - \text{Red}}{\text{NIR} + \text{Red}} \tag{4.1}$$

The box plots of RGB and NIR of diseased quadrats and healthy quadrats are shown in Figure 4.4. It can be noticed that the RGB values of diseased quadrats were less than that of healthy quadrats while the NIR values of diseased quadrats were greater than that of healthy quadrats. This indicates that SDS does influence light emissions of leaves. So one of the objectives of this paper is to use these predictors to detect the SDS infected quadrats. To avoid mutual interference between data samples collected in a quadrat at different time points, the

dataset was divided into training and testing dataset according to quadrat number. For the experiment in each year, we randomly selected 200 quadrats (83% of the dataset) as the training dataset and 40 quadrats (17% of the dataset) as the testing dataset of the total 240 quadrats.



(a) Red (b) Green

(c) Blue (d) NIR

Figure 4.4 Differences in red, green, blue and near-infrared (NIR) values between healthy and diseased quadrats (Inc=0: Healthy; Inc=1: Diseased).

4.2.2 Measurements

Three indices including overall accuracy, precision and recall, as calculated in Eq. (4.2)- Eq. (4.4) were measured to evaluate the performance of models. The definitions of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) are shown in Table 4.1.

$$\text{OverallAccuray} = \frac{\text{TP} + \text{TN}}{\text{Total}} \tag{4.2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.3}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.4}$$

Table 4.1 Definitions of true positive (TP), false positive (FP), true negative (TN) and false negative (FN).

Predicted \ **Actual**	**Positive**	**Negative**
Positive	TP	FP
Negative	FN	TN

4.2.3 Methods

Three methods were investigated in this paper. The first is the GRU-based method. The second is XGBoost, which is the representative of tree-based methods. The third is FCDNN which is the most widely used deep learning method. As for comparisons, both XGBoost and FCDNN used individual imageries while the GRU-based method used time-series imageries.

4.2.3.1 Gated Recurrent Units (GRU) based method

Most of the existing methods use one individual spectral measurement to predict the corresponding SDS [3]. Nevertheless, the multiple imageries in different time of the same quadrat may help improve the prediction accuracy of the model. Thus, a method that can handle time-series imagery is needed. Recurrent neural networks (RNNs) are suitable for non-linear time series processing [42]. As shown in Figure 4.5, the RNN consists of an input layer x, a hidden layer h and an output layer y. When dealing with time-series data, the RNN can be unfolded as the right part. The output and hidden layers can be calculated according to Eq. (4.5) and Eq. (4.6), respectively.

$$y_t = g\left(s_t * w_{hy}\right) \tag{4.5}$$

$$s_t = f\left(x_t * w_{sx} + s_{t-1} * w_{sS}\right) \tag{4.6}$$

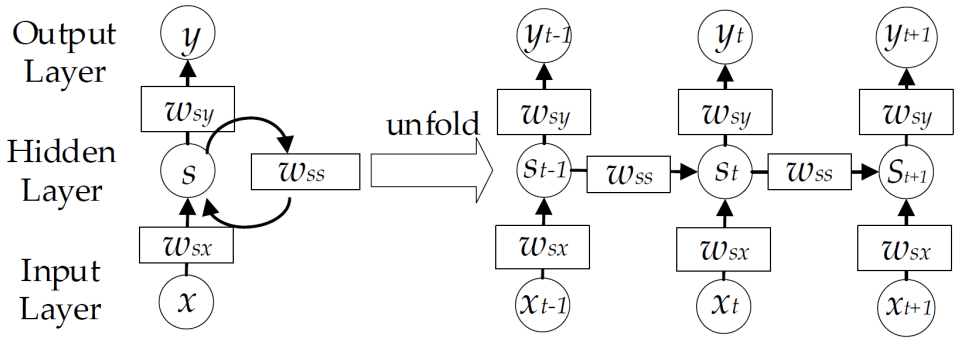


Figure 4.5 Structure of the recurrent neural network. At each time step, the network uses the output and internal state from the previous time step as the input of the current time step.

Despite its popularity as a universal function approximator and easy implementation, the RNN method is faced with the gradient vanishing/exploding problem. In the training process of RNNs, gradients are calculated from the output layer to the first layer of RNN. If the gradients are smaller than 1, the gradients of the first several layers will become small through many multiplications. On the contrary, the gradients will become very large if the gradients are larger than 1. Therefore, it sometimes causes the gradients to become almost zero or very large when it reaches to the first layers of RNNs. Consequently, the weights of first layers will not get updated in the training process. Therefore, simple RNNs may not be suitable for some complex problems.

In this paper, a GRU-based method is proposed to deal with the multivariate time-series imagery data that will solve the vanishing gradient problem of a standard RNN. As shown in Figure 4.6, based on the previous output h_{t-1} and the current input x_t, reset gate is used to determine which part of information should be reset as calculated in Eq. (4.7) while update gate is used to update the output of the GRU ht as calculated in Eq. (4.8). The candidate hidden layer is calculated according to Eq. (4.9). The current output can be obtained according to Eq.

(4.10). The gates, namely, z_t and r_t, and parameters, namely, W_z, W_r and W, of the GRU were updated in the training process

$$z_t = \sigma\left(W_z \bullet [h_{t-1}, x_t]\right) \tag{4.7}$$

$$r_t = \sigma\left(W_r \bullet [h_{t-1}, x_t]\right) \tag{4.8}$$

$$h'_t = \tan h\left(W \bullet [r_t * h_{t-1}, x_t]\right) \tag{4.9}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * h'_t \tag{4.10}$$

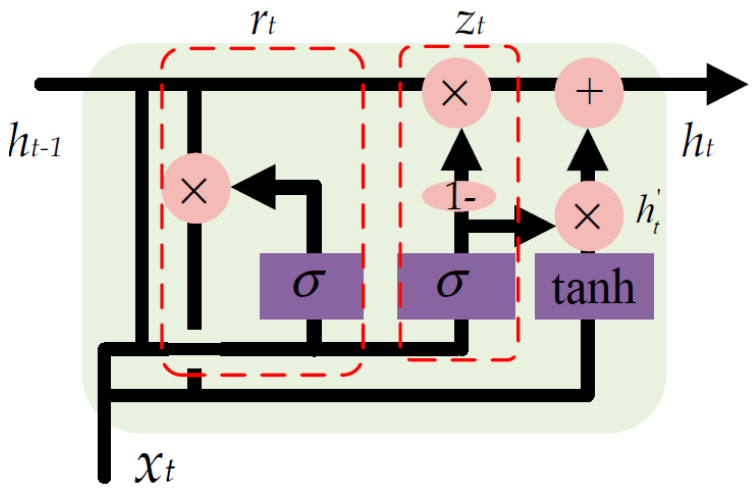


Figure 4.6 Structure of the recurrent neural network. At each time step, the network uses the output and internal state from the previous time step as the input of the current time step.

4.2.3.2 FCDNN

FCDNN is a class of methods that use multiple layers to extract information from the input data [43]. The basic layers are a fully connected layer and an activation layer. The fully

connected layer consists of multiple neurons. Each neuron in a fully connected layer connects to all neurons in the next layer. The output of a fully connected layer is calculated as Eq. (4.11).

$$y = W * x + b \tag{4.11}$$

The fully connected layer can only deal with a linear problem. To add the non-linear characteristic to the model, the concept of activation layers was introduced. Some widely used activation functions include sigmoid function, hyperbolic tangent function (Tanh) and Rectified Linear Unit (ReLU) function. In this paper, ReLU is used as Eq. (4.12).

$$y = max(0, x) \tag{4.12}$$

The loss function is used to measure the performance of models. In this paper, binary cross-entropy loss function is used, which can be calculated as Eq. (4.13),

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log\left(p\left(y_i\right)\right) + \left(1 - y_i\right)\log\left(1 - p\left(y_i\right)\right)\right) \tag{4.13}$$

where N is the total number of the samples, i is the index of the sample, (y_i is the label of the ith sample and $p(yi)$ is the predicted probability of the sample belonging to the class.

The most common training method is stochastic gradient descent which calculates the gradients and then updates the weights and biases iteratively. The final goal of training is to minimize the loss function. FCDNN can deal with large datasets and executes feature engineering without explicit programming.

4.2.3.3 XGBoost

XGBoost method is a popular tree-based method for classification tasks [44]. A decision tree consists of three parts: branches, internal nodes, and leaf nodes. The internal nodes are a set of conditions that can divide the samples into different classes. The branches represent the outcome of internal nodes. The leaf nodes represent the label of the class. A decision-tree can break down a complex classification problem into a set of simpler decisions at each stage [45].

One of the most famous tree-based methods is random forest. Random forest is an ensemble method that grows many trees and output the class based on the results of each individual tree. Given a training set X with labels Y, it randomly select samples with replacement of the training set and train the trees, b1, b2, b3, ..., bB. After training, for sample x', the prediction can be made by averaging the predictions of each tree or using the majority voting method. Since the bootstrap sampling method can reduce the variance of the model without increasing the bias, the model can have better generalization performance. Since each tree is trained separately, it can be implemented in parallel which can save a lot of time. Another merit of random forest is that it can deal with high-dimensional dataset and identify the importance of each variable. Different from a random forest algorithm, which generates many trees at the same time, XGBoost is trained in an additive manner. In each iteration of XGBoost training, a new decision tree that improves the model performance the most will be added to the model. The prediction of a sample can be calculated by summarizing the scores of each tree. XGBoost also includes a regularization term in the loss function which can further improve the generalization ability of the model.

4.3 Results

To validate the proposed technique, numerical calculations have been conducted. The codes were implemented in Python.

4.3.1 Model Parameters

respectively. After each fully connected layer, a ReLU layer is attached. The binary cross-entropy function is used as the loss function of the model. The optimizer is stochastic gradient descent. To avoid overfitting, the dropout technique was also used for each fully connected layer. The learning rate is set as 0.008. The batch size is 100. The maximum number of iterations is 300.

For XGBoost, the max number of leaves is set as 40. The max depth is set as 10. The objective is the binary cross-entropy function. The maximum number of iterations is 300.

For the proposed model, two GRU layers were stacked. The first GRU layer coverts each input sequence to a sequence of 50 units. Then the second GRU layer converts the sequences of 50 units to sequences of 25 units. Then a fully connected layer is used to transform the 25 variables to one output. The learning rate is set as 0.012. The batch size is 100. The maximum number of iterations is 300.

4.3.2 Calculations in Different Scenarios

To determine the effectiveness of the GRU-based method in different scenarios (i.e., the percentage of diseased quadrats), four calculations have been conducted. Since both the dataset of 2016 and 2017 had six pairs of imagery and human visual ratings, the dates of each year were numbered from 1 to 6. All the soybean plants at time points 1 to 3 in 2016 and time points 1 to 4 in 2017 were healthy; therefore, to reduce the influence of imbalanced data, we targeted to predict SDS at time points 3 to 6 in 2016 and 5 to 6 in 2017.

The specific settings of each calculation are shown in Table 4.2. In calculation I, the target was to predict SDS at time points 3 to 6 in 2016 and time points 5 and 6 in 2017. In the training of the GRU-based model, the imagery collected at time points 1 and 2 were added to construct the time-series imagery samples. The sequence length was set as 3. Thus, four sequences of satellite imagery (i.e., 1-2-3, 2-3-4, 3-4-5 and 4-5-6) can be generated for each quadrat in the experiment of 2016 and two sequences of satellite imagery (i.e., 3-4-5 and 4-5-6) can be generated for each quadrat in the experiment of 2017. In the training of XGBoost and FCDNN, only labeled satellite imagery at time points 3, 4, 5 and 6 were used to train the model. For the three methods, since there are two hundred quadrates for training and 40 quadrats for testing in each year, the number of training samples was 1200 (200×6) and the number of testing samples was 240 (40×6). In calculation II, the target was SDS at the time points 4, 5 and 6 in 2016 and time points 5 and 6 in 2017. In the training of the GRU-based model, the sequence length was set as 4. Thus, three sequences of satellite imagery (i.e., 1-2-3-4, 2-3-4-5 and 3-4-5-6) can be generated for each quadrat in the experiment of 2016 and two sequences of satellite imagery (i.e.,

Table 4.2 Settings of different calculations (N1: Number of training samples; P1: Percentage of diseased samples in the train set; N2: Number of testing samples P2: Percentage of diseased samples in the test set).

No.	Methods	Input (Target)	N1	P1	N2	P2
I	GRU	2016: 1-2-3 (3), 2-3-4 (4), 3-4-5 (5), 4-5-6 (6) 2017: 3-4-5 (5), 4-5-6 (6)	1200	14.92%	240	19.58%
	XGBoost	2016: 3 (3), 4 (4), 5 (5), 6 (6); 2017: 5 (5), 6 (6)				
	FCDNN	2016: 3 (3), 4 (4), 5 (5), 6 (6); 2017: 5 (5), 6 (6)				
II	GRU	2016: 1-2-3-4 (4), 2-3-4-5 (5), 3-4-5-6 (6) 2017: 2-3-4-5 (5), 3-4-5-6 (6)	1000	17.90%	200	23.50%
	XGBoost	2016: 4 (4), 5 (5), 6 (6); 2017: 5 (5), 6 (6)				
	FCDNN	2016: 4 (4), 5 (5), 6 (6); 2017: 5 (5), 6 (6)				
III	GRU	2016: 1-2-3-4-5 (5), 2-3-4-5-6 (6) 2017: 1-2-3-4-5 (5), 2-3-4-5-6 (6)	800	22.25%	160	29.38%
	XGBoost	2016: 5 (5), 6 (6); 2017: 5 (5), 6 (6)				
	FCDNN	2016: 5 (5), 6 (6); 2017: 5 (5), 6 (6)				
IV	GRU	2016: 1-2-3-4-5-6 (6) 2017: 1-2-3-4-5-6 (6)	400	30%	80	40.00%
	XGBoost	2016: 6 (6); 2017: 6 (6)				
	FCDNN	2016: 6 (6); 2017: 6 (6)				

2-3-4-5 and 3-4-5-6) can be generated for each quadrat in the experiment of 2017. The number of training samples was 1000 (200 × 5) and the number of testing samples was 200 (40 × 5).

In calculation III, the target was SDS at the time points 5 and 6 in 2016 and time points 5 and 6 in 2017. In the training of the GRU-based model, the sequence length was set as 5. Thus, two sequences of satellite imagery (i.e., 1-2-3-4-5 and 2-3-4-5-6) can be generated for each quadrat in the experiment of 2016 and two sequences of satellite imagery (i.e., 1-2-3-4-5 and 2-3-4-5-6) can be generated for each quadrat in the experiment of 2017. The number of training samples was 800 (200 × 4) and the number of testing samples was 160 (40 × 4).

In calculation IV, the target was SDS at the time point 6 in 2016 and time point 6 in 2017. In the training of the GRU-based model, the sequence length was set as 6. Thus, one sequence of satellite imagery (i.e., 1-2-3-4-5-6) can be generated for each quadrat in the experiment of 2016 and one sequence of satellite imagery (i.e., 1-2-3-4-5-6) can be generated for each quadrat in the experiment of 2017. The number of training samples was 400 (200 × 2) and the number of testing samples was 80 (40 × 2).

The results are shown in Table 4.3.

Table 4.3 Comparisons among the three methods in different calculations.

No.	Methods	Training Accuracy	Test Accuracy	Test Precision	Test Recall
I	GRU	0.909	0.904	0.800	0.681
	XGBoost	0.947	0.892	0.800	0.596
	FCDNN	0.903	0.829	0.800	0.170
II	GRU	0.885	0.860	0.890	0.532
	XGBoost	0.928	0.840	0.742	0.489
	FCDNN	0.929	0.855	0.855	0.638
III	GRU	0.865	0.856	0.800	0.681
	XGBoost	0.931	0.838	0.784	0.617
	FCDNN	0.905	0.806	0.643	0.766
IV	GRU	0.820	0.825	0.950	0.594
	XGBoost	0.865	0.813	0.905	0.594
	FCDNN	0.853	0.763	0.697	0.719

In calculation I, the three methods had the same test precision. However, the test recall of GRU was 9% and 51% greater than that of XGBoost and FCDNN, respectively. The reason why the test recall of FCDNN is much less is that the model was overfitted and most of the samples were predicted as healthy. In calculation II, the test accuracy of GRU was 2% greater than that of XGBoost and 0.5% greater than that of FCDNN owing to the improved test precision. It means that most of the positive predictions were accurate. In Calculation III, the test accuracy of GRU was 1.8% and 5% greater than that of XGBoost and FCDNN, respectively. In Calculation IV, the test accuracy of GRU was 1.2% and 6% greater than that of XGBoost and FCDNN, respectively. It can be observed that the training accuracy of GRU was less in the four calculations; however, the test accuracy of GRU is the highest among the three methods in the four calculations. It proves the good generalization performance of the GRU-based method. The test precision of GRU was highest in the four calculations, which was about 80%-95%. In terms of test recall, FCDNN outperformed the other two methods in Calculation II, III and IV. However, its test accuracy was less because the proportion of positives that are correctly identified was less.

The confusion matrices were shown in Figure 4.7. In calculation I, although GRU has the lowest number of true predictions for healthy quadrats, it has more true predictions for diseased quadrats than the other two methods. For FCDNN, the opposite is the case since 230 quadrats were classified as healthy and only 10 quadrats were classified as diseased. The reason is that FCDNN is more likely to predict samples as healthy. In calculation II, GRU made more true predictions for healthy quadrats. In calculation III, Both GRU and XGBoost classified 105 healthy quadrats correctly; however, GRU has more true predictions for diseased quadrats than XGBoost. In calculation IV, the performance of GRU was almost similar to that of XGBoost. The difference was only one healthy quadrat. FCDNN did not perform well in the classification of healthy quadrats. In conclusion, the calculation results show that the prediction accuracy can be improved by using the sequence-based model, i.e., GRU with the time-series imagery.

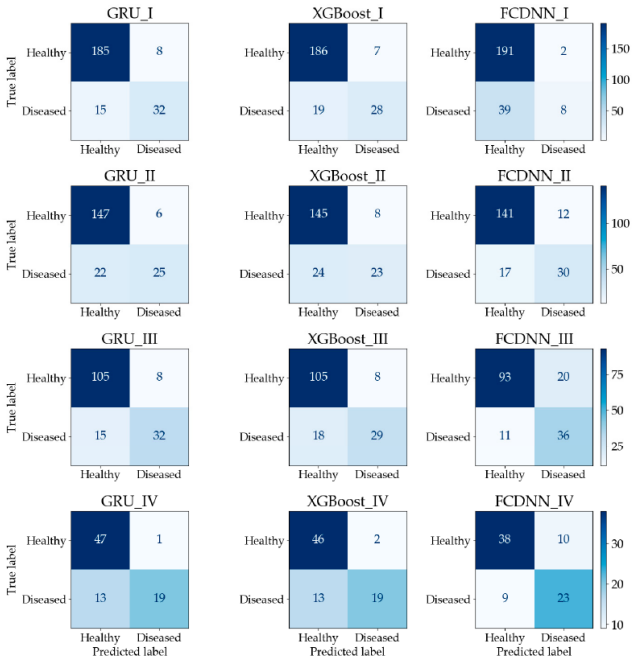


Figure 4.7 Confusion matrix of the testing dataset (each figure is named as method name _ calculation number).

4.3.3 Data Imbalance

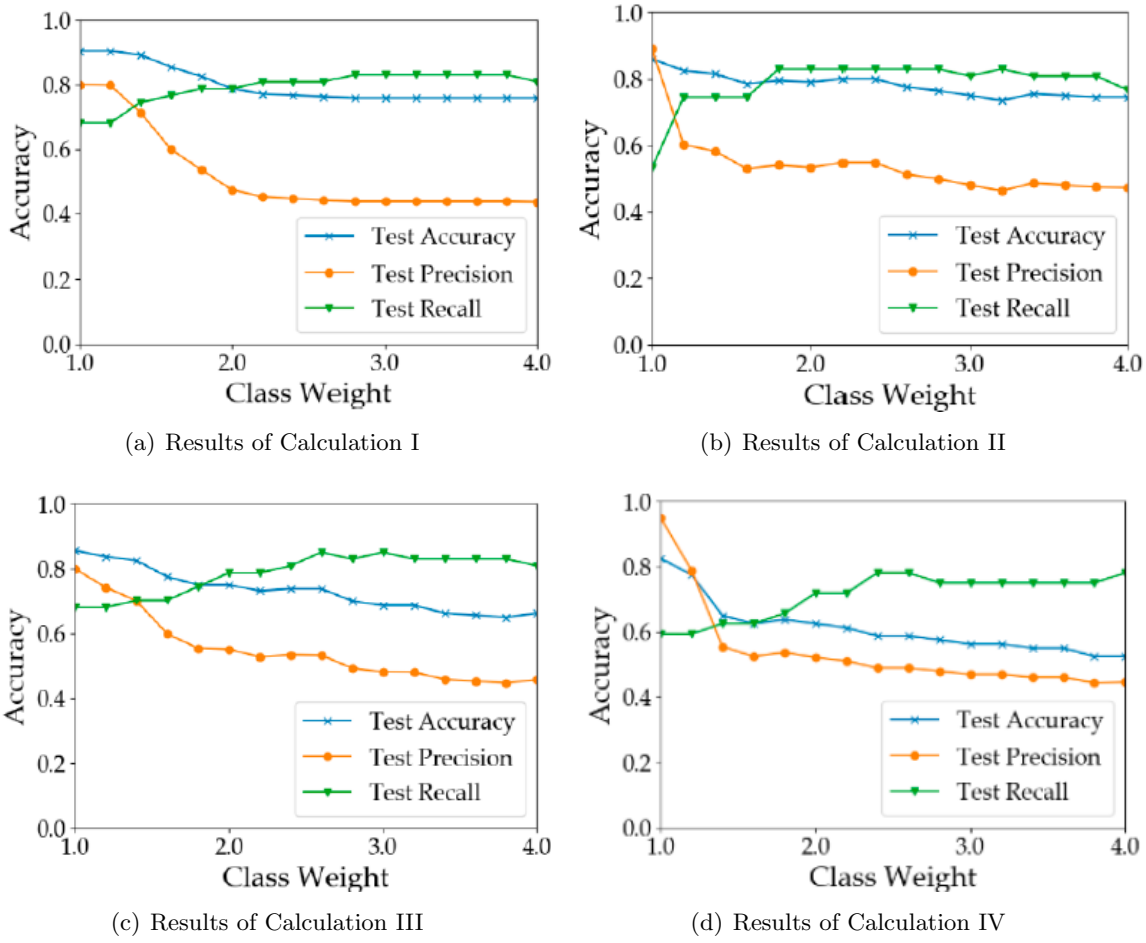
The number of healthy samples was much larger than that of diseased samples (Table 4.2). This is a typical example of data imbalance issue. One possible consequence is that the learning models will be guided by global performance, while the minority samples may be treated as noises [46]. For example, for a dataset consisting of 90 negative examples and 10 positive samples, a classifier that predicts all samples to be positive can achieve an overall accuracy of 90%. However, the classifier does not have the ability to predict positive examples. In our case, the training process of the models were guided by the cross-entropy loss function. To reduce the loss, models were more likely to predict the samples as healthy since the healthy samples were in majority. This pattern can be observed from the results shown in Table 3. In most of the situations, the test precision was 10-20% greater than the test recall. There were two common ways to address the issue of data imbalance. One way is to eliminate the data imbalance by using sampling methods. Over-sampling methods create more new minority classes while under-sampling

methods discard some samples in the majority class. Another way is to increase the costs for the misclassification of minority class samples. Since the number of samples is limited, the second way was adopted to test the model performance by assigning different weights to the minority class samples. The results were shown in Figure 4.8. In calculation I (Figure 4.8(a)), when the wight of the 'disease' class was increased from 1 to 1.2, the result did not change. When the class weight was between 1.2 and 2.8, with the increase of class weight, the recall of diseased samples increased from 0.681 to 0.830. However, the overall test accuracy and the test precision dropped 15% and 35%, separately. In calculation II (Figure 4.8(b)), when the weight was 1.8, the test recall achieved 0.830. After that, the model performance deteriorated with the increase of the class weight. In calculation III (Figure 4.8(c)), the best test recall was achieved when the class weight was increased to 2.6. The overall test accuracy and the test precision dropped 20% and 34%, separately. In calculation IV (Figure 4.8(d)), the test recall can be improved to 0.8. In summary, the increase of minority class weight can help improve the classification accuracy of diseased samples. However, the improvement was at the cost of overall accuracy and precision. Therefore, the value of the class weight should be adjusted according to the practical need. If the objective is to detect as many diseased quadrats as possible, a larger value of the class weight should be used; otherwise, a smaller value should be used.

4.3.4 Forecast of the SDS

Since the GRU-based method is a time sequence prediction model, it should forecast the SDS occurrence in future. To measure the prediction performance, calculations of four different scenarios have been conducted, as shown in Table 4.4.

The first method was named as GRU_Current, which was the same as the proposed method. The second was named as GRU_Next, which uses the time-series imagery to predict SDS at the next time point. The results have been compared with the results in Table 4.3. Take calculation A as an example; the target was SDS at time points 3, 4, 5 and 6 in 2016 and time points 5 and 6 in 2017. The sequence length of GRU_Next was 2. The sequences were 1-2, 2-3, 3-4 and 4-5 in



(a) Results of Calculation I

(b) Results of Calculation II

(c) Results of Calculation III

(d) Results of Calculation IV

Figure 4.8 Test accuracy, precision and recall using different weights for the minority class.

2016 and 3-4 and 4-5 in 2017. The sequence length of GRU_Current was 3. The sequences were 1-2-3, 2-3-4, 3-4-5 and 4-5-6 in 2016 and 3-4-5 and 4-5-6 in 2017. The two methods have the same target, i.e., SDS at the time points 3, 4, 5 and 6 in 2016 and at time points 5 and 6 in 2017.

The results are shown in Table 4.5. In Calculation A, B and C, the test accuracy of GRU_Current was 1%-3% greater than that of GRU_Next. In terms of test precision and test recall, GRU_Current also outperformed GRU_Next. However, with the increase of the sequence length, the gap between the test accuracy of the two methods becomes smaller. In Calculation D, although the train accuracies were slightly different, the test accuracy, recall and precision of the two methods were the same. It indicates that the proposed model can be used to predict the SDS

Table 4.4 Input imagery sequence and target dates of the two methods. The values in parenthesis indicate target dates.

No.	Input (target) of GRU_Current	Input (target) of GRU_Next
A	2016: 1-2-3 (3), 2-3-4 (4), 3-4-5 (5), 4-5-6 (6) 2017: 3-4-5 (5), 4-5-6 (6)	2016: 1-2 (3), 2-3 (4), 3-4 (5), 4-5 (6) 2017: 3-4 (5), 4-5 (6)
B	2016: 1-2-3-4 (4), 2-3-4-5 (5), 3-4-5-6 (6) 2017: 2-3-4-5 (5), 3-4-5-6 (6)	2016: 1-2-3 (4), 2-3-4 (5), 3-4-5 (6) 2017: 2-3-4 (5), 3-4-5 (6)
C	2016: 1-2-3-4-5 (5), 2-3-4-5-6 (6) 2017: 1-2-3-4-5 (5), 2-3-4-5-6 (6)	2016: 1-2-3-4 (5), 2-3-4-5 (6) 2017: 1-2-3-4 (5), 2-3-4-5 (6)
D	2016: 1-2-3-4-5-6 (6) 2017: 1-2-3-4-5-6 (6)	2016: 1-2-3-4-5 (6) 2017: 1-2-3-4-5 (6)

in the next time point when enough number of historical imageries are available, which will bring benefits to the prediction of the future development of the plant diseases.

Table 4.5 Comparisons between two methods.

Methods	No.	Train Accuracy	Test Accuracy	Test Precision	Test Recall
GRU_Current	A	0.909	0.904	0.800	0.681
GRU_Next	A	0.885	0.875	0.718	0.596
GRU_Current	B	0.885	0.860	0.890	0.532
GRU_Next	B	0.871	0.830	0.760	0.404
GRU_Current	C	0.865	0.856	0.800	0.681
GRU_Next	C	0.864	0.844	0.789	0.638
GRU_Current	D	0.820	0.825	0.950	0.594
GRU_Next	D	0.830	0.825	0.950	0.594

4.4 Discussion

Remote sensing using satellite imagery can be a potentially powerful tool to detect plant diseases at a quadrat or field level. Our results prove that the stress-triggered changes in the pattern of light emission due to soybean SDS can be detected through high-resolution satellite imagery and the classification accuracy of diseased and healthy quadrats can be further improved

by incorporating time-series prediction. Our proposed method has manifested its ability to improve SDS detection accuracy by incorporating time-series information.

A growing interest has been observed recently in the early detection of plant diseases.Researchers have used different remote sensing tools for early detection and monitoring of plant diseases at different spatial levels, such as leaf scale [47, 48, 49, 50], plant canopy scale [3, 51], plot-scale based on aerial imagery [52, 53] and field-wide-scale based on satellite imagery [54, 55, 56, 57]. Although many studies obtained remotely sensed data at different time points, none of them incorporated time-series information in the analytical models.

Instead of only using individual static imagery, we proposed a model that treats the satellite imagery, captured at different time points, as a time series for the detection of SDS. We compared the SDS prediction results from our proposed GRU-based method with XGBoost and the FCDNN, both non-sequence-based methods. Although the test accuracies of all three methods were above 76%, accuracy improved by up to 7% after incorporating time-series prediction. This substantial improvement in accuracy reveals that the GRU-based method uses the characteristic information of spectral bands, ground-based crop rotation and time series in an optimal way. The main advantage of the GRU-based method goes back to its workability of learning from history data. In the learning process, GRU can determine the influence of images at different time points on the decision-making of the current status through the reset gate and the update gate. Information from past images can help the model to eliminate the effect of some noise, such as weather conditions.

In all calculation scenarios, the GRU-based method outperformed XGBoost and the FCDNN in SDS prediction accuracy. Although XGBoost and the FCDNN are very powerful methods, they do not incorporate time-based information. In our data, we found that reflectance values in the RGB spectrum were lower for healthy quadrats than diseased quadrats, while this was the opposite in the NIR spectrum. Moreover, this pattern became clearer near the end of the cropping season, which indicates that incorporating time-based information for satellite images can add information to the data analysis.

Besides accuracy, the GRU-based method also achieved greater precision (80–95%) for SDS detection in all calculation scenarios, as compared to the XGBoost and FCDNN methods. This means that soybean quadrats predicted either as diseased or healthy were assigned correctly 80% to 95% of the time. However, in terms of recall, the FCDNN performed better than the GRU-based method and XGBoost in three out of four calculation scenarios. This means that the FCDNN predicted diseased soybean quadrats more accurately than the other two methods. However, its test accuracy was lower because this method was not correctly predicting the healthy soybean quadrats. The reason is that FCDNN was more likely to predict a sample as diseased, so the precision was sacrificed.

In the end, we made predictions of SDS at future time points (GRU_Next) based on previous time-based imagery and we compared these results with the predictions made at time points included in the training (GRU_Current). SDS prediction accuracy, precision and recall were greater in GRU_Current models in small sequence scenarios. It is possible that this improved precision is because the models in the GRU_Current scenarios can extract information from the imagery collected at the current time point. On the other hand, in GRU_Next scenarios, we were predicting SDS using the history data only. Moreover, in the last sequence scenario, accuracy, recall and precision of both GRU_Next and GRU_Current models became equal, which indicates that the GRU-based method can predict SDS in soybean quadrats with high accuracy when enough historical images are available.

A recent study conducted at the same site detected SDS through satellite images with above 75% accuracy using the random forest algorithm. Contrary to our study, they used static images for data analysis and predicted SDS on the 30% test subset from the same satellite images [54]. In comparison,we proposed a method that can predict SDS with high accuracy at future time points, based on time-series satellite imagery.

4.5 Limitations and Future Work

It should be noted that this proposed plant disease recognition using a sequence-based model with remote sensing is subject to a few limitations which suggest future research directions.

In the data collection, the number of imageries in our case study is limited because the PlanetScope satellite can only take pictures of the locations every one or two weeks. More frequent observations of SDS can help the model capture the subtle changes along time. Some studies installed the camera system around the canopies or plots so that they can monitor the real-time status of the plants. However, since it requires a high intensity of cameras, the scalability of this method is limited. Another issue of satellite imagery is that the collection frequency of the imagery may not correspond to the development of diseases. As shown in the previous section, SDS development in June and July was slow while it was much faster at the end of August. So, ideally, it would have been better to collect imageries more frequently in August. Besides, the locations of the quadrats of 2016 and 2017 were different. So the time sequences of 2016 and 2017 were constructed separately. In the future, the experimental fields should be evaluated consistently so that longer time sequences can be used to further improve the prediction accuracy.

In terms of data preprocessing, this paper only used the mean values and variance values of the pixels due to the resolution limitations. The importance of different pixels may be different due to disease intensities. One alternative way is to construct a small-scale CNN to get features of each image and then feed the extracted features to the GRU. Additionally, multi-band imagery can also be used to improve the accuracy.

In some calculations, the percentage of diseased samples was very low, meaning that the data was not balanced. In this paper, this issue was addressed by assigning weights to the diseased samples in the calculation of loss of function. There are also some other methods. For example, the oversampling method can generate new diseased samples by recombining the pixels of the diseased samples. In the future, we will compare the effectiveness of different methods.

Lastly, in this study, SDS is labeled as diseased (denoted as “1”) and not diseased (denoted as “0”) using 5% as a threshold value which constitutes a classification problem. One concern is the

influence of the threshold value on the model performance. For example, in our case, quadrats with 5.1% and 99.1% incidence were categorized as diseased samples, which may influence the prediction precision of the model. Therefore, the classification accuracy of the model will be tested using different threshold values in the future. We can also convert the classification problem to the prediction problem of SDS continuous incidence. These are reserved for future research.

4.6 Conclusions

The development of sensing techniques has brought significant improvements to plant disease detection, treatment and management. However, there is a lack of research on detecting SDS using pixel-level satellite imagery at the quadrat level. The major challenge is how to use the limited information, i.e., only a few pixels for each quadrat, to improve SDS prediction accuracy. Some traditional methods for the analysis of near sensing data are based on CNNs, which can efficiently extract the most important features from thousands of RGB values and other information. In contrast, for the analysis of low pixel-level satellite imagery, additional information is required.

In this paper, a GRU-based model is proposed to predict SDS by incorporating the temporality into the satellite imagery. Instead of using individual static imagery, time-series imagery is used to train the model. Different test case scenarios have been created for the comparisons between the proposed method and other non-sequence based methods. The results show that, compared to XGBoost and FCDNN, the GRU-based can improve the overall prediction accuracy by 7%. In addition, the proposed method can also be adapted to predict future development of SDS.

References

[1] Tom W Allen, Carl A Bradley, Adam J Sisson, Emmanuel Byamukama, Martin I Chilvers, Cliff M Coker, Alyssa A Collins, John P Damicone, Anne E Dorrance, Nicholas S Dufault, et al. Soybean yield loss estimates due to diseases in the united states and ontario, canada, from 2010 to 2014. *Plant Health Progress*, 18(1):19–27, 2017.

[2] Crop Protection Network. Estimates of corn and soybean yield losses due to disease: An online tool, 2020.

[3] Ittai Herrmann, Steven K Vosberg, Prabu Ravindran, Aditya Singh, Hao-Xun Chang, Martin I Chilvers, Shawn P Conley, and Philip A Townsend. Leaf and canopy level detection of fusarium virguliforme (sudden death syndrome) in soybean. *Remote Sensing*, 10(3):426, 2018.

[4] Craig VM Barton. Advances in remote sensing of plant stress. *Plant and Soil*, 354(1):41–44, 2012.

[5] Anne-Katrin Mahlein, Erich-Christian Oerke, Ulrike Steiner, and Heinz-Wilhelm Dehne. Recent advances in sensing plant diseases for precision crop protection. *European Journal of Plant Pathology*, 133(1):197–209, 2012.

[6] H Nilsson. Remote sensing and image analysis in plant pathology. *Annual review of phytopathology*, 33(1):489–528, 1995.

[7] Reyer Zwiggelaar. A review of spectral properties of plants and their potential use for crop/weed discrimination in row-crops. *Crop protection*, 17(3):189–206, 1998.

[8] Stephane Jacquemoud and Susan L Ustin. Leaf optical properties: A state of the art. In *8th International Symposium of Physical Measurements & Signatures in Remote Sensing*, pages 223–332. CNES Aussois France, 2001.

[9] Laury Chaerle and Dominique Van Der Straeten. Imaging techniques and the early detection of plant stress. *Trends in plant science*, 5(11):495–501, 2000.

[10] N Mastrodimos, D Lentzou, Ch Templalexis, DI Tsitsigiannis, and G Xanthopoulos. Development of thermography methodology for early diagnosis of fungal infection in table grapes: The case of aspergillus carbonarius. *Computers and Electronics in Agriculture*, 165:104972, 2019.

[11] Yuxuan Wang, Shamaila Zia-Khan, Sebastian Owusu-Adu, Thomas Miedaner, and Joachim Müller. Early detection of zymoseptoria tritici in winter wheat by infrared thermography. *Agriculture*, 9(7):139, 2019.

[12] Kathrin Bürling, Mauricio Hunsche, and Georg Noga. Use of blue–green and chlorophyll fluorescence measurements for differentiation between nitrogen deficiency and pathogen infection in winter wheat. *Journal of plant physiology*, 168(14):1641–1648, 2011.

[13] Anne-Katrin Mahlein. Plant disease detection by imaging sensors–parallels and specific demands for precision agriculture and plant phenotyping. *Plant disease*, 100(2):241–251, 2016.

[14] María Luisa Pérez-Bueno, Mónica Pineda, and Matilde Barón. Phenotyping plant responses to biotic stress by chlorophyll fluorescence imaging. *Frontiers in plant science*, 10:1135, 2019.

[15] Koushik Nagasubramanian, Sarah Jones, Soumik Sarkar, Asheesh K Singh, Arti Singh, and Baskar Ganapathysubramanian. Hyperspectral band selection using genetic algorithm and support vector machines for early identification of charcoal rot disease in soybean stems. *Plant methods*, 14(1):1–13, 2018.

[16] Koushik Nagasubramanian, Sarah Jones, Asheesh K Singh, Soumik Sarkar, Arti Singh, and Baskar Ganapathysubramanian. Plant disease identification using explainable 3d deep learning on hyperspectral images. *Plant methods*, 15(1):1–10, 2019.

[17] Alina Förster, Jens Behley, Jan Behmann, and Ribana Roscher. Hyperspectral plant disease forecasting using generative adversarial networks. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 1793–1796. IEEE, 2019.

[18] Halil Durmuş, Ece Olcay Güneş, and Mürvet Kırcı. Disease detection on the leaves of the tomato plants by using deep learning. In *2017 6th International Conference on Agro-Geoinformatics*, pages 1–5. IEEE, 2017.

[19] Kaitlin M Gold, Philip A Townsend, Adam Chlus, Ittai Herrmann, John J Couture, Eric R Larson, and Amanda J Gevens. Hyperspectral measurements enable pre-symptomatic detection and differentiation of contrasting physiological effects of late blight and early blight in potato. *Remote Sensing*, 12(2):286, 2020.

[20] Wei Liu, Xueren Cao, Jieru Fan, Zhenhua Wang, Zhengyuan Yan, Yong Luo, Jonathan S West, Xiangming Xu, and Yilin Zhou. Detecting wheat powdery mildew and predicting grain yield using unmanned aerial photography. *Plant disease*, 102(10):1981–1988, 2018.

[21] Sreekala G Bajwa, John C Rupe, and Johnny Mason. Soybean disease monitoring with leaf reflectance. *Remote Sensing*, 9(2):127, 2017.

[22] Nicholle M Hatton. *Use of small unmanned aerial system for validation of sudden death syndrome in soybean through multispectral and thermal remote sensing.* PhD thesis, 2018.

[23] Lin Yuan, Ruiliang Pu, Jingcheng Zhang, Jihua Wang, and Hao Yang. Using high spatial resolution satellite imagery for mapping powdery mildew at a regional scale. *Precision Agriculture*, 17(3):332–348, 2016.

[24] Rakesh Roshan Satapathy. Remote sensing in plant disease management. *J. Pharmacogn. Phytochem*, 9:1813–1820, 2020.

[25] Alan Mc Gibney, Martin Klepal, and Dirk Pesch. Agent-based optimization for large scale wlan design. *IEEE Transactions on Evolutionary Computation*, 15(4):470–486, 2011.

[26] Mrunmayee Dhakate and AB Ingole. Diagnosis of pomegranate plant diseases using neural network. In *2015 fifth national conference on computer vision, pattern recognition, image processing and graphics (NCVPRIPG)*, pages 1–4. IEEE, 2015.

[27] Konstantinos P Ferentinos. Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145:311–318, 2018.

[28] Gerrit Polder, Pieter M Blok, Hendrik AC de Villiers, Jan M van der Wolf, and Jan Kamp. Potato virus y detection in seed potatoes using deep learning on hyperspectral images. *Frontiers in plant science*, 10:209, 2019.

[29] Archana Chaudhary, Savita Kolhe, and Raj Kamal. An improved random forest classifier for multi-class classification. *Information Processing in Agriculture*, 3(4):215–222, 2016.

[30] Bhavini J Samajpati and Sheshang D Degadwala. Hybrid approach for apple fruit diseases detection and classification using random forest classifier. In *2016 International conference on communication and signal processing (ICCSP)*, pages 1015–1019. IEEE, 2016.

[31] Philipp Probst, Marvin N Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1301, 2019.

[32] Yunong Zhang, Danchi Jiang, and Jun Wang. A recurrent neural network for solving sylvester equation with time-varying coefficients. *IEEE Transactions on Neural Networks*, 13(5):1053–1063, 2002.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[34] Muammer Turkoglu, Davut Hanbay, and Abdulkadir Sengur. Multi-model lstm-based convolutional neural networks for detection of apple diseases and pests. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–11, 2019.

[35] Sarah Taghavi Namin, Mohammad Esmaeilzadeh, Mohammad Najafi, Tim B Brown, and Justin O Borevitz. Deep phenotyping: deep learning for temporal phenotype/genotype classification. *Plant methods*, 14(1):1–14, 2018.

[36] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[37] Xiu Jin, Lu Jie, Shuai Wang, Hai Jun Qi, and Shao Wen Li. Classifying wheat hyperspectral pixels of healthy heads and fusarium head blight disease using a deep neural network in the wild field. *Remote Sensing*, 10(3):395, 2018.

[38] Matthew Z Liebman, Lance R Gibson, David N Sundberg, Andrew Howard Heggenstaller, Paula R Westerman, Craig Chase, Robert G Hartzler, Fabián D Menalled, Adam S Davis, and Philip M Dixon. Agronomic and economic performance characteristics of conventional and low-external-input cropping systems in the central corn belt. *Agronomy Journal*, 100(3):600, 2008.

[39] Adam S Davis, Jason D Hill, Craig A Chase, Ann M Johanns, and Matt Liebman. Increasing cropping system diversity balances productivity, profitability and environmental health. 2012.

[40] Robin Gómez, Matt Liebman, David N Sundberg, and Craig A Chase. Comparison of crop management strategies involving crop genotype and weed management practices in conventional and more diverse cropping systems. *Renewable agriculture and food systems*, 28(3):220–233, 2013.

[41] Leonor FS Leandro, Alison E Robertson, Daren S Mueller, and Xiao-Bing Yang. Climatic and environmental trends observed during epidemic and non-epidemic years of soybean sudden death syndrome in iowa. *Plant health progress*, 14(1):18, 2013.

[42] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[43] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[44] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[45] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[46] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.

[47] T Rumpf, A-K Mahlein, U Steiner, E-C Oerke, H-W Dehne, and L Plümer. Early detection and classification of plant diseases with support vector machines based on hyperspectral reflectance. *Computers and electronics in agriculture*, 74(1):91–99, 2010.

[48] Heba Al-Hiary, Sulieman Bani-Ahmad, M Reyalat, Malik Braik, and Zainab Alrahamneh. Fast and accurate detection and classification of plant diseases. *International Journal of Computer Applications*, 17(1):31–38, 2011.

[49] E Bauriegel, A Giebel, M Geyer, U Schmidt, and WB Herppich. Early detection of fusarium infection in wheat using hyper-spectral imaging. *Computers and Electronics in Agriculture*, 75(2):304–312, 2011.

[50] Jinzhu Lu, Reza Ehsani, Yeyin Shi, Ana Isabel de Castro, and Shuang Wang. Detection of multi-tomato leaf diseases (late blight, target and bacterial spots) in different stages by using a spectral-based sensor. *Scientific reports*, 8(1):1–11, 2018.

[51] Cedric Bravo, Dimitrios Moshou, Jonathan West, Alastair McCartney, and Herman Ramon. Early disease detection in wheat fields using spectral reflectance. *Biosystems Engineering*, 84(2):137–145, 2003.

[52] Nicholle Hatton, Ajay Sharda, William Schapaugh, Deon Van der Merwe, et al. Remote thermal infrared imaging for rapid screening of sudden death syndrome in soybean. In *2018 ASABE Annual International Meeting*, page 1. American Society of Agricultural and Biological Engineers, 2018.

[53] Hai Pham, Yixiang Lim, Alessandro Gardi, Roberto Sabatini, and Eddie Pang. A novel bistatic lidar system for early-detection of plant diseases from unmanned aircraft. In *Proceedings of the 31th Congress of the International Council of the Aeronautical Sciences (ICAS 2018), Belo Horizonte, Brazil*, pages 9–14, 2018.

[54] Muhammad M Raza, Chris Harding, Matt Liebman, and Leonor F Leandro. Exploring the potential of high-resolution satellite imagery for the detection of soybean sudden death syndrome. *Remote Sensing*, 12(7):1213, 2020.

[55] Lin Yuan, Jingcheng Zhang, Yeyin Shi, Chenwei Nie, Liguang Wei, and Jihua Wang. Damage mapping of powdery mildew in winter wheat with high-resolution satellite image. *Remote sensing*, 6(5):3611–3623, 2014.

[56] Qiong Zheng, Wenjiang Huang, Ximin Cui, Yue Shi, and Linyi Liu. New spectral index for detecting wheat yellow rust using sentinel-2 multispectral imagery. *Sensors*, 18(3):868, 2018.

[57] S Yang, X Li, C Chen, P Kyveryga, and XB Yang. Assessing field-specific risk of soybean sudden death syndrome using satellite imagery in iowa. *Phytopathology*, 106(8):842–853, 2016.

CHAPTER 5. A TRANSFORMER-BASED APPROACH FOR EARLY PREDICTION OF SOYBEAN YIELD USING TIME-SERIES IMAGES

Luning Bi[1], Owen Wally[2], Guiping Hu[1], Albert U. Tenuta[2],

Yuba R. Kandel[3], and Daren S Mueller[3]

[1] Department of Industrial and Manufacturing Systems Engineering, Iowa State University

[2]Ontario Ministry of Agriculture

[3]Department of Plant Pathology and Microbiology, Iowa State University

Modified from a manuscript to be submitted to *Frontiers in Plant Science*

Abstract

Crop yield prediction which provides critical information for management decision-making is of significant importance in precision agriculture. Traditional manual inspection and calculation are often laborious and time-consuming. For yield prediction using high-resolution images, existing methods, e.g., convolutional neural network, are hard to model long range multi-level dependencies across image regions. This paper proposes a transformer-based approach for yield prediction using early-stage images and seed information. First, each original image is segmented into plant and soil categories. Two vision transformer (ViT) modules are designed to extract features from each category. Then a transformer module is established to deal with the time-series features. Finally, the image features and seed features are combined to estimate the yield. A case study has been conducted using a dataset that was collected during the 2020 soybean-growing seasons in Canada fields. Compared with other baseline models, the proposed method can reduce the prediction error by over 40%.

Keywords: transformer, image recognition, time-series prediction, soybean yield prediction, deep learning

5.1 Introduction

The increasing world population imposes significant challenges for agriculture production due to the increasing food demand combined with limited arable land. Accurate yield prediction can help seed companies breed for better cultivars and guide farmers to make informed management and financial decisions. However, crop yield prediction is exceptionally challenging due to several complex factors, e.g. seed type, seed treatment, soil, temperature, etc. Thus, an analytical model that can predict crop yield accurately is essential.

Machine learning methods have been designed for crop monitoring and yield prediction. Various models have been proposed for crop yield prediction. For example, Kaul et al. developed an artificial neural network model that used field-specific rainfall data and soil rating to predict soybean yield [1]. Khaki et al. proposed a deep neural network approach for soybean yield prediction using genetic information and environmental information [2]. Compared to yield prediction using meteorological driven variables (e.g., temperature, sunlight, and precipitation), using the sensing images can capture more information about the plant growing status. For example, Rembold et al. used low-resolution satellite imagery for yield prediction [3]; Nevavuori et al. presented a convolutional neural network (CNN) for crop yield prediction based on NDVI and RGB data acquired from unmanned aerial vehicles (UAVs) [4]; and Pantazi et al. built a hybrid model to associate the high-resolution soil sensing data with wheat yield [5]. However, even with advanced remote sensing techniques producing high resolution images, yield prediction using the remote sensing images still suffers from information loss and variable weather conditions.

Compared to hyperspectral images, camera images of the canopy can capture more information since they have higher resolution, i.e., more pixels. However, in terms of forecasting future yield, the information extracted from only one timestamp is typically insufficient. For instance, a single image of a field is very likely to be influenced by the light condition, soil status or plant growth stage at the time of that particular image.

These undetermined factors and noise can confuse models in the training stage, resulting in the deterioration of generalization ability. The incorporation of time-series prediction is necessary for yield prediction to eliminate the influence of these noises.

There are two challenges for yield prediction using time-series images, i.e., image processing and time-series prediction. Existing studies usually use the convolutional neural network with long short term memory model (CNN-LSTM) framework for feature extraction of time-series images. For example, Sun et al. combined the CNN and LSTM to predict soybean yield using in-season and out-season image data collected from Google Earth [6]. Newton et al. used 16-day remote sensing images (30m by 30m) to predict potato yield [7]. Sharifi et al. applied different machine learning approaches to the barley yield prediction using the time-series NDVI and environmental information [8]. However, this framework has some drawbacks.

For image classification/recognition, although the CNNs have outstanding performance on many tasks [9, 10, 11], the CNNs have some redundancy issues in both computation and representations since each pixel bears varying importance for the target task. Recently, the transformer module has been considered as an alternative architecture and has achieved competitive performance on many computer vision tasks [12]. Vision transformer (ViT) is a transformer-based method that is designed for image classification [13]. In ViT, an image is split into fixed-size patches. Each patch is then linearly embedded, position embeddings are added, and the resulting sequence of vectors is fed to a standard transformer encoder. Compared to CNN, ViT has a better global understanding of the images.

Regarding the time-series prediction, LSTMs have been employed to model time series in different tasks [14, 15, 16]. In a LSTM, the hidden state is updated with every new input token to remember the entire sequence it has seen. Theoretically, this structure can propagate over infinitely long sequences. However, in practice, due to the vanishing gradient problem, the LSTM will eventually forget earlier tokens [17]. Another drawback of the LSTM is that it can only be implemented sequentially due to its structure. In comparison, transformers retain direct

connections to all previous timestamps, allowing information to propagate over much longer sequences and be processed in parallel.

To solve the aforementioned challenges, a transformer-based method is used to predict soybean yield using time-series images and seed treatment information. The contribution of our work includes the following aspects:

- A method consisting of two ViT modules and one transformer is proposed for the feature extraction of time-series images. Instead of using the original images directly, the proposed method process the plant part and soil part of the image separately to reduce the computation complexity and improve the interpretability of the model.
- A wide-deep structure is adopted to better combine the seed features with the image features.
- Different baseline models were compared to validate the effectiveness of the proposed approach. The experiments show that the proposed method can significantly improve yield prediction accuracy.

The rest of the paper is organized as follows. Section 5.2 introduces the dataset, data processing steps and this study's workflow. Section 5.3 explains the structure of the proposed model as well as the details of each module. Section 5.4 compares the performance of four baseline models. Section 5.5 discusses the results of the methods. Section 5.6 covers the main conclusions for this study.

5.2 Materials and Methods

5.2.1 Data collection

This study used a dataset collected from three soybean fields in Ontario, Canada in 2020. There are 450 plots in total. The data includes two types of input information. The first is the time-series images. The second part is the seed treatment information used in each plot. For each

plot, there are three images, as shown in Figure 5.1, collected in three dates, on June 14, 2020, on July 13, 2020 and on August 20, 2020.

Figure 5.1 An example image of a plot

The distribution of the yield of plots is shown in Figure 5.2. The distribution is a little right-skewed. Most plots have a yield between 3500 kg/ha and 5000 kg/ha. Thus, the objective of this paper is to predict the yield using the time-series images and seed treatment information.

5.2.2 Image segmentation

In the data processing, each image is segmented into two parts, i.e., plant segmentation and soil segmentation, as shown in Figure 5.3. This is for two reasons. First, the information extracted from plant itself with the soil can be decoupled. Each module only needs to calculate the same type of information, i.e., either plant or soil part, which will reduce the redundant computation. The interaction between plant and environment is calculated afterward. Second, it can help reduce the influence of the diagonal camera angles. The segmentation can directly tell the model the distance between two adjacent rows of plants. Thus the model can distinguish the plants at the near-end from the plants at the far end.

5.2.3 Workflow of soybean yield estimation

As shown in Fig. 5.4, the workflow can be divided into three steps: data collection, data processing, and prediction. In the data collection, a sensing system is built to take the images of a field at a certain frequency. The images along the soybean growth stage and the checked yield are

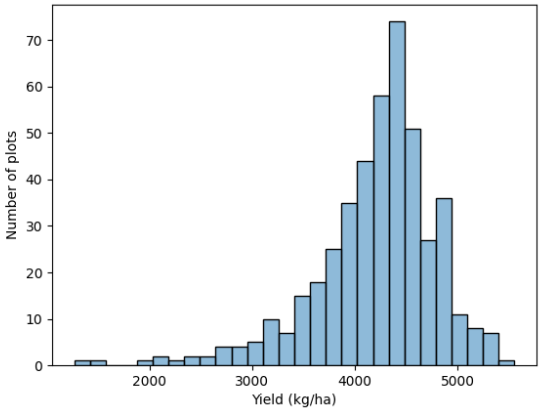
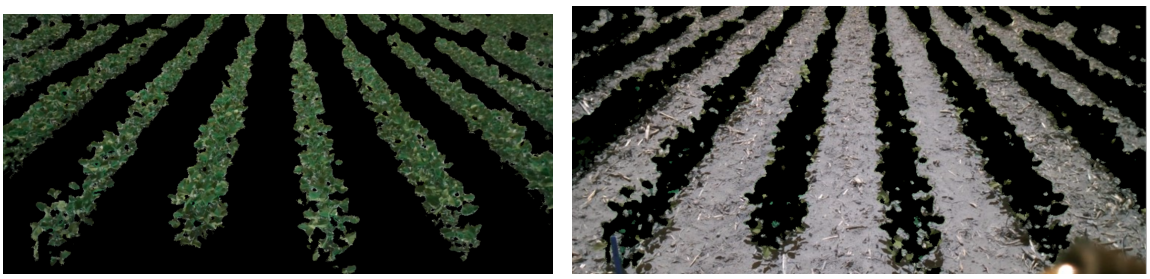


Figure 5.2 Distribution of the soybean yield.



(a) Segmentation of plant part

(b) Segmentation of soil part

Figure 5.3 Image segmentation. **(a)** Segmentation of plant part. **(b)** This is the caption for Segmentation of soil part.

stored in the database. In data processing, some statistical analysis and image segmentation are conducted to prepare for the following analysis. Finally, various prediction models are designed to predict soybean yield. The models will be evaluated by some feasible metrics so that they can be further optimized accordingly.

The prediction is the most challenging component. The solution needs to answer three questions. How to efficiently extract features from a single image? How to detect the hidden pattern in the time-series images? How to combine different sources of information, i.e., images and seed information? This serves as the motivation of this paper.

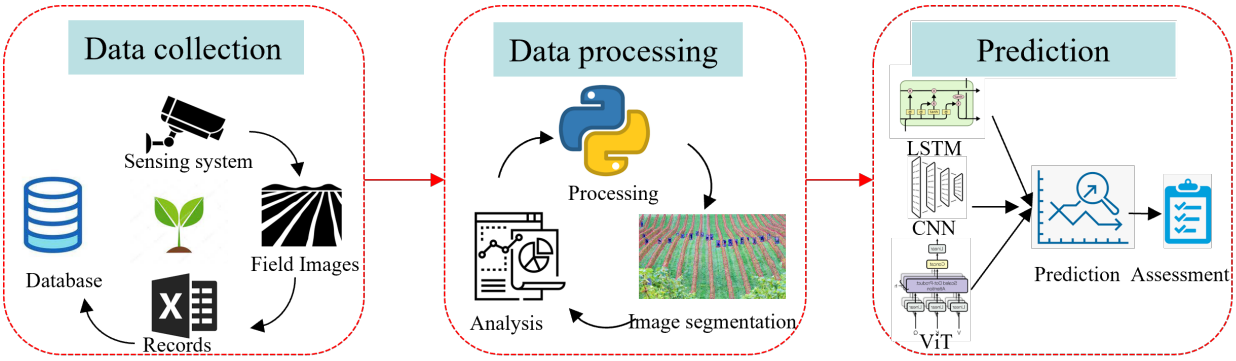


Figure 5.4 Flow diagram of the data collection, processing and prediction we employed in this study for yield prediction.

5.3 Proposed Model

To address the aforementioned challenges, a wide-deep method based on the attention mechanism is proposed. In this section, we will focus on the prediction part of the workflow, as shown in Fig. 5.4, especially the design logic and module about feature extraction of the images and seed information.

5.3.1 A wide-deep framework

As introduced in Sec. 5.2, this study considers two types of inputs: time-series images and seed information. Thus, different modules should be applied due to the heterogeneity of the inputs. The time-series images have a large number of pixels. The model should be capable of extracting the most important interactions between pixels effectively. Thus, a high-level feature representation of the images is needed. In contrast, the seed treatment information only contains one categorical variable in this study. It is not necessary to apply a complex or extremely deep neural network. Therefore, a wide-deep framework is proposed as shown in Fig. 5.5.

The left tower of the proposed framework is composed of two ViT modules and one transformer module. Two ViT modules are used to extract features from the plant and soil, separately. The outputs from the two ViTs are combined using a dot product operator. Then the

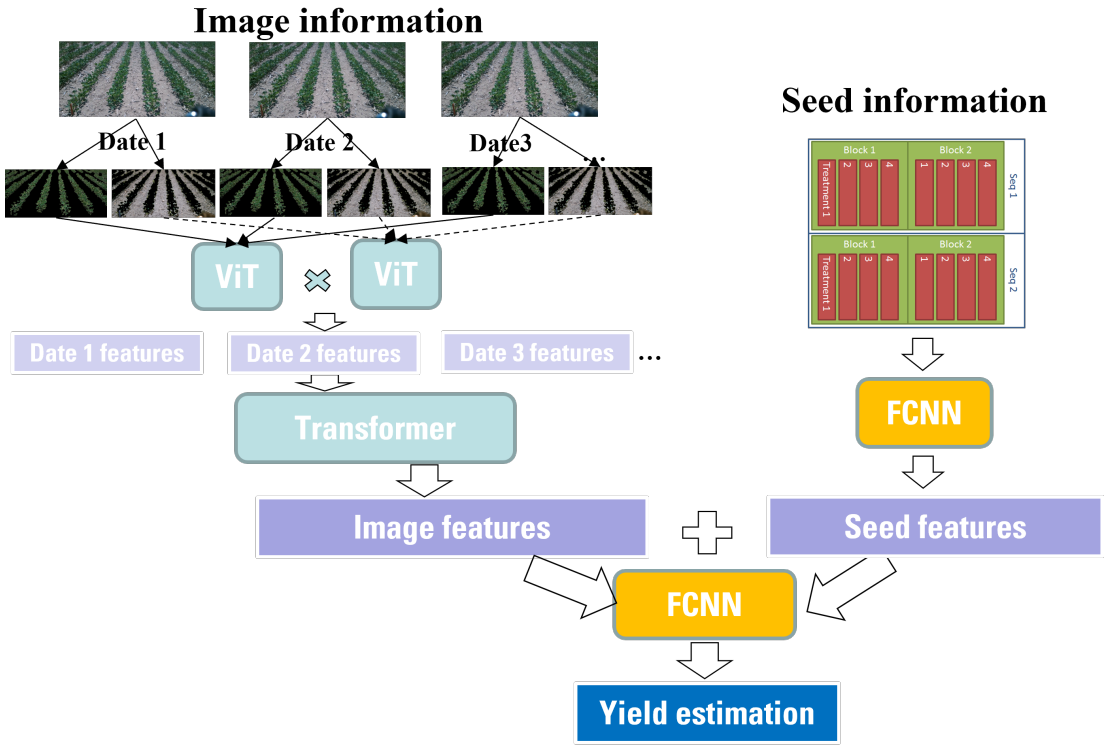


Figure 5.5 A wide-deep framework for yield prediction

transformer is leveraged to deal with the time-series features. The right tower is just a fully connected neural network. The seed treatment is one-hot encoded. Then the neural network is used to further extract information from the one-hot encoding. Finally, the wide component (i.e., seed features) and deep component (i.e., image features) are combined using one common FCDNN for joint training according to Eq. 5.1.

$$\varphi = \Phi((f_{plant} \cdot f_{soil}) + f_{seed}) \tag{5.1}$$

Where f_{soil} is the feature obtained from the soil segmentation of an image, f_{plant} is the feature extracted from the plant segmentation, f_{seed} is the feature extract from the seed treatment, φ represent the predicted yield and Φ denotes a one-layer fully connected neural network (FCDNN).

It should be noted that the image features and seed features are combined and then jointly trained. This is different from the ensemble train. In an ensemble model, individual models or

weak estimators are trained separately without any interaction during the training process. Then their outputs are combined only at the final step (i.e., prediction) by majority voting or averaging. In contrast, the wide-deep framework will jointly train all parameters simultaneously by taking both the image features and treatment features as well as the weights of their sum into account. The training of the deep-wide model is done by backpropagating the gradients from the output to both the wide and deep part of the model simultaneously using stochastic gradient descent (SGD) or other optimizers such as Adam and Adagrad. By leveraging this deep-wide framework, the training time or inference time can be significantly reduced due to fewer parameters in the wide part.

In the following sections, we will explain the details of the attention mechanism, transformer and ViT.

5.3.2 Attention mechanism

Attention is a technique proposed to mimic cognitive attention [18]. The effect enhances some parts of the input data while diminishing other parts as the network should focus more on the small but important parts of the data.

As shown in Eq. 5.2, for each input in a given vector $a_1, a_2, a_3...$, three matrices, i.e. query W_q, key W_k and value W_v, are employed to generate three representation vector i.e., Q, K and V, by multiplication. Q represents the query to match other inputs. K is the key to be matched by others. V represents the information to be extracted. Then the attention score between two inputs can be calculated by Eq. 5.3 to obtain the attention coefficients.

$$Q = aW_q, K = aW_k, V = aW_v \tag{5.2}$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5.3}$$

Where d_k is the dimension of the keys and queries which is used to scale the dot product of Q and K

Specifically, we repeat the attention for h times and concatenate the learned embeddings as the final representation of the inputs:

$$\text{MultiHead}(Q, K, V) = \text{Concat}\left(\text{head}_1, \ldots, \text{ head }_{\text{h}}\right) W^O \tag{5.4}$$

Where $\text{ head }_{\text{i}} = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$

The attention mechanism is the backbone of transformer and ViT.

5.3.3 Vision transformer for image feature extraction

Self-Attention is capable of understanding the connection between inputs. However, it is challenging to apply it between the pixels of an image. For instance, if the size of the input image is 300x300, a self-attention layer has 90K combinations to calculate. In fact, a lot of the calculation are redundant because only part of the connections between two pixels are meaningful. To overcome this problem, ViT is proposed by segmenting images into small patches (like 16x16) [13]. A patch is the basic unit of an image instead of a pixel to efficiently tease out patterns.

In ViT, an image $x \in \mathbb{R}^{H \cdot W \cdot C}$ is reshaped into N patches $x_p \in \mathbb{R}^{N \cdot P^2 \cdot C}$, where (H, W) is the resolution of the original image, C is the number of channels, P^2 is the resolution of each patch. In addition to patches, ViT also use a learnable embedding $\mathbf{E}_{pos}$ for each patch to represent the relative position. Thus, the patch embeddings can be represented as in Eq. 5.5.

$$\mathbf{z}_0 = \left[\mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}\right] + \mathbf{E}_{\text{pos}}, \quad \mathbf{E} \in \mathbb{R}^{\left(P^2 \cdot C\right) \times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{5.5}$$

Assuming that there are L layers in the ViT, then in each layer, multi-head attention and MLP is applied to the input of each layer as shown in Eq. 5.6 and Eq. 5.7 . The calculation of multi-head attention is explained in Eq. 5.4 .

$$\mathbf{z}'_\ell = \text{MultiHead}\left(\text{LN}\left(\mathbf{z}_{\ell-1}\right)\right) + \mathbf{z}_{\ell-1}, \ell = 1 \ldots L \tag{5.6}$$

$$\mathbf{z}_\ell = \text{MLP}\left(\text{LN}\left(\mathbf{z}'_\ell\right)\right) + \mathbf{z}'_\ell, \ell = 1 \ldots L \tag{5.7}$$

Where LN is the Layernorm operator [19]. LN is applied before every block, and residual connections after every block.

The last step is to output the image features as calculated using 5.8

$$\mathbf{y} = \mathrm{LN}\left(\mathbf{z}_L^0\right) \tag{5.8}$$

5.3.4 Transformer for time-series prediction

For time-series prediction, RNN or LSTM are usually the first ones to consider. However, this type of models is hard to parallel because the models process the input of each timestamp in sequence order. Then, some studies adopted CNN to realize parallelization of the feature extraction. Nevertheless, CNN can only consider the input in a limited range. For long-term dependency modeling, CNN needs to increase the number of filters and the number of layers. Therefore, transformers based on the self-attention mechanism are applied for time-series prediction. It computes the relation between two timestamps in a bi-directional manner, which means it can be implemented in parallel.

The basic structure of a transformer used for sequence-to-sequence tasks includes encoder and decoder parts [20]. Nevertheless, in this study, the task is to transform a sequence to some features. Thus, only the encoder part is used for the transformer. The encoder of the transformer is composed of an input layer, a positional encoding layer, and a stack of multi-head attention layers. The input layer maps the input time-series data to a vector through a fully-connected network. Positional encoding with sine and cosine functions is used to encode sequential information in the time series data by element-wise addition of the input vector with a positional encoding vector, which is the same as Eq. 5.5. Each multi-head layer is to calculate the attention coefficients between the image features of every two timestamps. Finally, there is an output layer that maps the output of the last multi-head attention layer to image features.

5.4 Results

To validate the effectiveness of the proposed method, we compared it with other baseline models.

5.4.1 Baseline models

The three most commonly used models are implemented as the baseline models, i.e., CNN-LR, CNN-LSTM and ViT-T. The treatment of seed information is the same for all baseline models and the proposed method.

5.4.1.1 CNN-LR

CNN is a class of deep, feed-forward artificial neural networks. It was adopted widely for its fast deployment and high performance on image classification tasks. CNNs are usually composed of convolutional layers, pooling layers, batch normalization layers and fully connected layers. The convolutional layers extract features from the input images whose dimensionality is then reduced by the pooling layers. Batch normalization is a technique used to normalize the previous layer by subtracting the batch mean and dividing by the batch standard deviation, which can increase the stability and improve the computation speed of the neural networks. The fully connected layers are placed near the output of the model. They act as classifiers to learn the non-linear combination of the high-level features and to make numerical predictions. Detailed descriptions on each type of function can be accessed from Gu et al. [21].

In CNN-LR, firstly, a CNN is built to extract features from a single image. Then the obtained features from time-series images are concatenated with seed features and then used as the input of a linear regression model. Since the linear regression model cannot detect the dependency in a time series, CNN-LR is used to show the influence of time-series features.

5.4.1.2 CNN-LSTM

Despite its popularity as a universal function approximator and easy implementation, RNN is faced with the gradient vanishing/exploding problem. In the training process of RNNs, gradients are calculated from the output layer to the first layer of the RNN. If the gradients are smaller than 1, the gradients of the first several layers will become small through many multiplications. On the contrary, they will become very large if the gradients are larger than 1. Therefore, it sometimes causes the gradients to be almost zero or very large when it reaches the first layers of RNNs. Consequently, the weights of the first layers will not get updated in the training process. Therefore, simple RNNs may not be suitable for very long time series. LSTM solves this issue by introducing the concept of gates. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. At each timestamp, the cell adjust its state value according to the current input and memory of previous steps. And the three gates regulate the flow of information into and out of the cell. Therefore, LSTM can extract features from long time series. Detailed explanations and calculations of each function can be accessed from Hochreiter et al. [22].

In CNN-LSTM, the first step is to extract features from a single image. Then the extracted features of images taken at different timestamps are treated as a time series. LSTM is employed to deal with the time-series features. The output obtained by LSTM is combined with seed features to get the yield prediction through a fully connected neural network.

5.4.1.3 ViT-T

Different from the proposed method, in ViT-T, the image is not segmented into soil and plant parts. Thus only one ViT module is utilized to read images. Then the time-series image features are used as the input of the transformer. The yield prediction is made based on the output of the transformer and the seed features.

5.4.2 Experiment settings

In CNN-LR, the CNN module uses the VGG-16 architecture which consists of 13 convolutional layers and 3 fully connected layers. The linear regression module is applied with L2 norm regularization. In CNN-LSTM, the CNN module is the same as that in CNN-LR. The LSTM module have two bi-directional LSTM layers. For the ViT module in ViT-T, there are two multi-head attention layers and each layer has 3 heads. The transformer module in ViT also has 3 multi-head attention layers and each layer has 5 heads. The difference between ViT-T and the proposed method is that two ViT modules are used in the proposed method to process the plant part and soil part of the image separately. All models use the Adam optimizer with 0.001 as learning rate. 344 plots are used as the train set. 38 plots are used as the validation set. 68 plots are used as the test set.

Three metrics are used to assess the model performance, i.e., root mean squared error (RMSE), R squared value, and mean absolute error percentage (MAPE). The calculations are as in Eq. 5.9, Eq. 5.10 and Eq. 5.11.

$$RMSE = \sqrt{\frac{1}{n}\Sigma(y - \widehat{y})^2} \tag{5.9}$$

$$R^2 = 1 - \frac{RSS}{TSS} \tag{5.10}$$

$$MAPE = \frac{1}{n}\Sigma(|\frac{y - \widehat{y}}{y}|) \tag{5.11}$$

Where n is the number of samples, y is the ground-truth yield, $\widehat{y}$ is the predicted yield, RSS is the sum of squares of residuals, and TSS represents the total sum of squares.

5.4.3 Comparisons with baseline models

The predicted values of models and the ground truth for the test set are plotted in Fig. 5.6.

The performance of CNN-LSTM and the proposed method are compared. It can be seen that for most plots, the predicted values of the proposed method are closer to the ground truth. It can

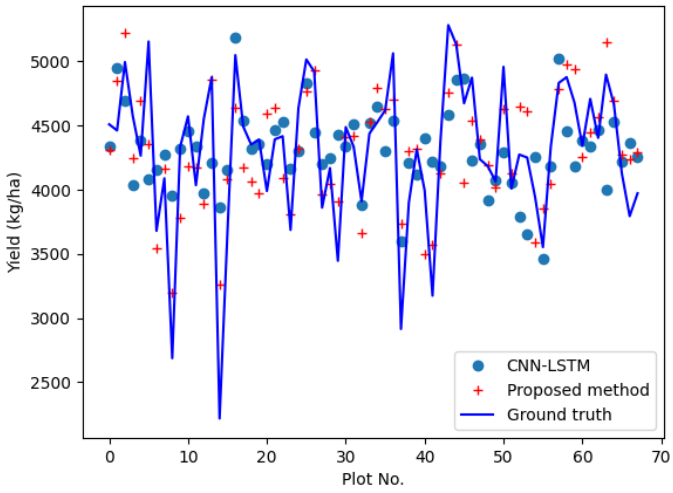


Figure 5.6 Predicted values and the ground truth for the test set.

also be observed that the models are conservative in making the prediction. For instance, the ground truth values of Plot 8 and Plot 14 are between 2200 kg/ha and 2500 kg/ha. However, the predictions of the proposed model and CNN-LSTM are above 3140 kg/ha. The proposed method has the best performance on these two plots. The predicted values of CNN-LSTM is between 3900 kg/ha and 4500 kg/ha while the predicted values of the proposed method have more diversity. It shows that the proposed method can perform better in some extreme cases.

The test RMSE, R squared and MAPE are shown in Table 5.1. If the mean value of each seed treatment group is used as the estimate, the test RMSE, R squared and MAPE are 570.596, 0.010, and 12.412%. The R squared is 0.010, meaning that only using the seed treatment is just a little bit better than using the mean values of all train plots. The introduction of CNN can improve the RMSE, R squared, MAPE by 11.7%, 0.19 and 2.8%, separately. Compared to CNN-LR, CNN-LSTM performs better: 6.2% reduction in RMSE, 0.09 increase in R squared and 0.5% in MAPE. This validates that the hidden pattern in time series features can help improve the prediction accuracy. ViT-T is an updated version based on the CNN-LSTM structure. The multi-head self-attention mechanism improve RMSE by 8.9%, R squared by 0.1 and MAPE by

0.3%, separately. The proposed method, which has two ViT modules and one transformer, significantly reduces RMSE by 21.7%, increases R square by 0.2 and reduces MAPE by 1.6%.

Table 5.1 Comparisons between baseline models and the proposed method. Average: using the mean values of each seed treatment group as the estimate.

Methods	Test RMSE	Test R squared	Test MAPE (%)
Average	570.569	0.010	12.412
CNN-LR	510.959	0.205	9.648
CNN-LSTM	481.191	0.295	9.176
ViT-T	445.619	0.395	8.811
Proposed method	**368.620**	**0.608**	**7.296**

5.5 Discussion

Crop yield prediction help farmers estimate yield before a field is harvested. Additionally, it can also serve as an essential tool for the decision-makers to make plans regarding food security. However, many factors both genetic and environmental factors, before and during the season, make it challenging to obtain an accurate prediction.

Yield prediction using images recently became a popular topic due to two reasons. The first reason is that images can store all the phenotype information of the plant as well as some environmental information (i.e., soil color, light condition, etc.). The second reason is that the development of deep learning techniques in computer vision has facilitated information extraction from plant-level or field-level images. Different from the research using satellite [3, 23] or UAV [24, 25] images, this study used high-resolution camera images of field level. This will help to improve the prediction accuracy since more pixels represent more information about the plant.

Instead of using individual static imagery, the proposed framework leverages the time-series images for yield prediction. The time-series images can monitor the plant status of plants at different time points and eliminate the influence of noise on the model performance. This has been supported by many researches [26, 27, 28]. In our case study, the single image method, i.e.,

CNN-LR, is compared with the time-series image method, i.e., CNN-LSTM. The results show that time-series images can help improve test RMSE by 6.2%, R squared by 0.9%, and MAPE by 0.5%. Since each plot only has about three images, the improvement could be more significant if additional images were provided. Besides, the traditional CNN-LSTM framework [6, 29] is upgraded to the ViT-T framework by introducing the attention mechanism. CNNs are efficient in image information extraction compared to fully connected neural networks due to shared kernel weights. However, CNNs only aggregates the global information in high-level layers. ViTs incorporate more global information than CNNs at lower layers, leading to quantitatively different image features. In terms of time-series prediction, although LSTM can capture the long-term dependencies of the time series, it get inputs in sequence and cannot be implemented in parallel. Thus, ViT-T is better in the global understanding of images, computation efficiency and parallel implementation. In our case, the images were taken from one side of the plot. The information density of the image in the far end and the near end are different. Since ViT sigments images into small patches, it can assign different weights according to the region/patch and achieve better granularity. The comparison results show improvements of 8.9% in test RMSE, 0.1 in R squared and 0.3% in MAPE.

Another contribution of our work is that the proposed method segmented the image into the plant part and the soil part. By using two ViT modules, the plant status and the environmental influence can be modeled separately. Then the two parts are multiplied to obtain soybean yield. Compared to the one-ViT version, i.e., ViT-T, the proposed method significantly reduces RMSE by 21.7%, increases R square by 0.2 and reduces MAPE by 1.6%. In this study, seed treatment is only used as supplementary information for yield prediction. The result of the average method indicates that the importance of seed treatment in the model is limited because the test R squared is only 0.01. However, the wide-deep framework can be used to include more categories of input information, e.g., genetic information, in the future.

5.6 Conclusions

Yield prediction can provide more guidelines for farmers to decide on the management plan. The development of deep learning techniques has facilitated the application of sensing techniques in precision agriculture through satellite imagery, UAV imagery or camera imagery. In this study, in order to catch more global interactions between image patches and timestamps, a transformer based method is proposed to extract image information and time-series changes of soybean status. Besides, the original images are segmented into the plant part and soil parts. A wide-deep structure is adopted to incorporate other information, i.e., seed treatments, into prediction. Compared to other baseline models, the proposed model can reduce the RMSE by up to 35%.

However, there also exist some limitations in this study. First, the influence of the length of the time series on the final prediction accuracy has not been investigated due to the limited data. Although it is reasonable to believe that more images for training will help improve the model performance, the redundant information can also impact the model generalization ability or increase the need for computation resources. Thus, how to search for the balance point of the trade-off between these factors is meaningful. Second, only seed treatment information is considered. However, the proposed framework can incorporate more input, such as genetic information and soil characteristics. Last, in this study, the attention score of images and time series are calculated separately. We may consider the attention score between image patches in different timestamps. In the future, in addition to addressing the aforementioned limitations, we are going to design a more efficient multi-modal method for yield prediction using sensing imagery and environment information, such as soil PH value and moisture.

References

[1] Monisha Kaul, Robert L Hill, and Charles Walthall. Artificial neural networks for corn and soybean yield prediction. *Agricultural Systems*, 85(1):1–18, 2005.

[2] Saeed Khaki and Lizhi Wang. Crop yield prediction using deep neural networks. *Frontiers in plant science*, 10, 2019.

[3] Felix Rembold, Clement Atzberger, Igor Savin, and Oscar Rojas. Using low resolution satellite imagery for yield prediction and yield anomaly detection. *Remote Sensing*, 5(4):1704–1733, 2013.

[4] Petteri Nevavuori, Nathaniel Narra, and Tarmo Lipping. Crop yield prediction with deep convolutional neural networks. *Computers and electronics in agriculture*, 163:104859, 2019.

[5] Xanthoula Eirini Pantazi, Dimitrios Moshou, Thomas Alexandridis, Rebecca L Whetton, and Abdul Mounem Mouazen. Wheat yield prediction using machine learning and advanced sensing techniques. *Computers and electronics in agriculture*, 121:57–65, 2016.

[6] Jie Sun, Liping Di, Ziheng Sun, Yonglin Shen, and Zulong Lai. County-level soybean yield prediction using deep cnn-lstm model. *Sensors*, 19(20):4363, 2019.

[7] Imran Hossain Newton, AF M Tariqul Islam, AKM Saiful Islam, GM Tarekul Islam, Anika Tahsin, and Sadmina Razzaque. Yield prediction model for potato using landsat time series images driven vegetation indices. *Remote Sensing in Earth Systems Sciences*, 1(1):29–38, 2018.

[8] Alireza Sharifi. Yield prediction with machine learning algorithms and satellite images. *Journal of the Science of Food and Agriculture*, 101(3):891–896, 2021.

[9] Juncheng Ma, Keming Du, Feixiang Zheng, Lingxian Zhang, Zhihong Gong, and Zhongfu Sun. A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network. *Computers and electronics in agriculture*, 154:18–24, 2018.

[10] Konstantinos P Ferentinos. Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145:311–318, 2018.

[11] Xiu Jin, Lu Jie, Shuai Wang, Hai Jun Qi, and Shao Wen Li. Classifying wheat hyperspectral pixels of healthy heads and fusarium head blight disease using a deep neural network in the wild field. *Remote Sensing*, 10(3):395, 2018.

[12] Yutong Xie, Jianpeng Zhang, Chunhua Shen, and Yong Xia. Cotr: Efficiently bridging cnn and transformer for 3d medical image segmentation. *arXiv preprint arXiv:2103.03024*, 2021.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[14] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[15] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[16] Zheng Zhao, Weihai Chen, Xingming Wu, Peter CY Chen, and Jingmeng Liu. Lstm network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2):68–75, 2017.

[17] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, 32:5243–5253, 2019.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[19] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*, 2019.

[20] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*, 2020.

[21] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[23] Raí A Schwalbert, Telmo Amado, Geomar Corassa, Luan Pierre Pott, PV Vara Prasad, and Ignacio A Ciampitti. Satellite-based soybean yield forecast: Integrating machine learning and weather data for improving crop yield prediction in southern brazil. *Agricultural and Forest Meteorology*, 284:107886, 2020.

[24] Muhammad Adeel Hassan, Mengjiao Yang, Awais Rasheed, Guijun Yang, Matthew Reynolds, Xianchun Xia, Yonggui Xiao, and Zhonghu He. A rapid monitoring of ndvi across the wheat growth cycle for grain yield prediction using a multi-spectral uav platform. *Plant science*, 282:95–103, 2019.

[25] X Zhou, HB Zheng, XQ Xu, JY He, XK Ge, X Yao, T Cheng, Y Zhu, WX Cao, and YC Tian. Predicting grain yield in rice using multi-temporal vegetation indices from uav-based multispectral and digital imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130:246–255, 2017.

[26] JGPW Clevers. A simplified approach for yield prediction of sugar beet based on optical remote sensing data. *Remote sensing of Environment*, 61(2):221–228, 1997.

[27] Hossein Aghighi, Mohsen Azadbakht, Davoud Ashourloo, Hamid Salehi Shahrabi, and Soheil Radiom. Machine learning regression techniques for the silage maize yield prediction using time-series images of landsat 8 oli. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(12):4563–4577, 2018.

[28] Sebastian Varela, Taylor Pederson, Carl J Bernacchi, and Andrew DB Leakey. Understanding growth dynamics and yield prediction of sorghum using high temporal resolution uav imagery time series and machine learning. *Remote Sensing*, 13(9):1763, 2021.

[29] Lobna Nassar, Ifeanyi Emmanuel Okwuchi, Muhammad Saad, Fakhri Karray, Kumaraswamy Ponnambalam, and Prarabdha Agrawal. Prediction of strawberry yield and farm price utilizing deep learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

CHAPTER 6. GENERAL CONCLUSION

Precision agriculture is a management strategy that utilizes various sources of information to support management decisions based on estimated variability, productivity and quality of agricultural production. However, there are two aspects of the challenges. One challenge is the diversity of the task types. Popular topics in agriculture include yield prediction, species recognition, plant disease classification and etc. Some of the tasks, e.g., yield prediction, are regression problems and others, e.g. plant disease classification, are multi-classification problems. The other challenge is the heterogeneity of the data. For instance, in G by E problem, the genetic information is encoded in discrete integers while the environmental features are real values. Similarly, in soybean yield prediction problem, the input includes field images and treatment information. Besides, some data are temporal, while others are discrete values. Therefore, this dissertation aims to address the challenges by leveraging deep learning methods for multi-variable forecasting, image classification, and time-series prediction.

The first paper proposed a genetic algorithm (GA) assisted deep learning method to predict the crop yield using genetic information and the environment information. Since linear regression based models are not able to formulate the complex biological and physiologic relationships among genes and environment factors due to the dependency, the deep neural network was used. However, after feature filtering, there are still hundreds of input variables. It means the neural network is extremely large, leading to the slow convergence speed and the local optimum issue. Thus the proposed algorithm was designed to overcome two drawbacks in neural networks, i.e. the influence of initialization on the convergence of the algorithm and the gradient vanishing/exploding problem, by combining the zeroth-order method and the gradient method. The main contribution of this paper is that a two-stage GA-assisted method for deep neural networks. In the global search phase, multiple sets of parameters were generated. Then GA is

used to search the solution space. In the local search phase, the GA algorithm will evolve a low-dimensional subspace, i.e., the nodes in one layer. At each iteration, evolution strategies will generate many children solutions, i.e., different neural networks. The local search adapts to a dynamic environment by the population-based search strategy which can help avoid the vanishing gradient and local optimum. Then the weights will be updated by the gradient decent method. By combining the strength of the gradient decent and evolutionary search, the convergence and efficiency of the proposed method can be guaranteed.

The second paper focused on the overfitting problem in plant disease classification using limited data. Convolutional neural network (CNN) is a class of deep, feed-forward artificial neural networks for image classification. It was adopted widely for its fast deployment and high performance on image classification tasks. However, CNN requires a large training dataset, which is typically not the case for plant disease recognition. When the number of parameters of the neural network is much greater than the number of data samples, a small training dataset will lead to the overfitting problem. One of the commonly adopted methods to address this problem is data augmentation. To improve the prediction accuracy of CNN in the classification of plant diseases using a limited training dataset, three techniques have been designed and implemented in this study, i.e., data augmentation, Wasserstein generative adversarial network with gradient penalty (WGAN-GP), and label smoothing regularization (LSR). The first step is to train the WGAN-GP with LSR using real images. The trained WGAN-GP is then used to generate additional labeled images. The synthetic images will be mixed with real images and then augmented through classic data augmentation methods. Finally, the combined dataset will be used to train the CNN. Experiments showed that the proposed method can improve the overall classification accuracy of plant diseases is 2%-4% than other augmentation methods. The main contributions of this study lie in two dimensions. Firstly, the majority of the existing studies focused on a single type of disease or only one plant type. This method was designed with the capability to address the multi-disease and multi-plant type situation. Because there may exist multiple diseases for one plant type in the actual situation. Secondly, to address the issue of

limited training set, a WGAN-GP was combined with LSR to generate images that can enlarge the training dataset and regularize the CNN model simultaneously.

The third paper developed a gated recurrent unit (GRU) based method to detect soybean sudden death syndrome (SDS) disease development using satellite imagery. It was found out that reflectance values in the RGB spectrum were lower for healthy quadrats than diseased quadrats, while this was the opposite in the (near-infrared) NIR spectrum. This pattern became clearer near the end of the cropping season indicating that time-series satellite images may help improve the prediction accuracy. However, the challenge was that the resolution of satellite imagery was 3 m x 3 m, which means the satellite imagery of each generalized quadrat only contained several pixels. To deal with low pixel-level remote sensing data, three models, i.e., GRU, XGBoost, and the fully connected deep neural network (FCDNN) were tested in calculation scenarios simulating different stages of the plant growing season. Predictions of SDS at future time points were made based on previous time-based imagery and then were compared with the predictions made at time points included in the training. In all calculation scenarios, the GRU-based method outperformed XGBoost and the FCDNN in SDS prediction accuracy. The results also indicated that the GRU-based method could predict SDS in soybean quadrats with high accuracy when enough historical images were available. This study has two contributions. The first contribution is that the proposed method used satellite imageries to detect SDS at a quadrat level. The second contribution is that the GRU-based model used temporal image sequences for prediction instead of using field imageries at a single time point to identify the contemporary status of the disease.

The last paper proposed a transformer-based approach for yield prediction using early-stage images and seed information. First, inspired from G by E problem in the first paper, each original image is segmented into plant and soil categories. Two vision transformer (ViT) modules are designed to extract features from each category. Then a transformer module is established to deal with the time-series features. Finally, the image features and seed features are combined to estimate the yield. Thus, the information extracted from plant itself with the soil can be decoupled which will reduce the redundant computation. The interaction between plant and

environment is calculated by joint training. The reason why ViT was used instead of CNN is that self-Attention mechanism of ViT is capable of understanding the connection between inputs. In ViT, an image is segmented into small patches (e.g., 16x16) and then transformed into the query, key and value matrices for feature extraction. A patch is the basic unit of an image instead of a pixel to efficiently tease out patterns. The proposed method was tested on a dataset collected during the 2020 soybean-growing seasons in Canada fields. Compared with other baseline models, the proposed method can reduce the prediction error by over 40%. There are three contributions of this work. Firstly, a method consisting of two ViT modules and one transformer is proposed for the feature extraction of time-series images. Instead of using the original images directly, the proposed method process the plant part and soil part of the image separately to reduce the computation complexity and improve the interpretability of the model. Secondly, a wide-deep structure is adopted to better combine the seed features with the image features. Lastly, different baseline models were compared to validate the effectiveness of the proposed approach. The experiments show that the proposed method can significantly improve yield prediction accuracy.

However, there also exist some limitations in this dissertation which suggest future research directions. Firstly, most of the studies are based on deep learning methods because neural networks can learn by discovering intricate structures in the data they experience. However, it is sort of a 'black-box' algorithm. At each layer of neural networks, the output of the last layer will be combined though linear multiplication (e.g., weight matrix) and non-linear activation functions (e.g. sigmoid function). It is hard to explain how the input determines the predictions specifically. Even though real-world data encompasses high-dimensional inputs with multiple interactions, it is common to possess prior domain knowledge about the monotonic trend (nonincreasing / non-decreasing) between a subset of input features and the output, giving rise to partial monotonicity. Thus, in the future, we can enhance the interpretability by incorporating monotonicity. Secondly, some of the studies in this dissertation are subject to the limited data, especially in temporal images. Theoretically, more frequent observations of the plant growing stage can help the model capture the subtle changes along time. However, the redundant

information can also impact the model generalization ability or increase the need for computation resources. Thus, it is meaningful to search for the balance point of the trade-off between these factors. Thirdly, the studies in this dissertation were conducted separately. Some obtained results in a study could be Incorporated in another study. One potential solution to this is to introduce transfer learning techniques which store knowledge gained while solving one problem and applying it to a different but related problem. Finally, although this dissertation covers various deep learning architectures for different tasks, it is necessary to build a more general framework for deep learning in agriculture. This general framework should be able to process and combine different sources of input (e.g., genetic variables, environmental variables, treatment variables, and non-temporal or temporal images) as well as generating the output for different types of tasks, i.e., forecasting or binary/multiple class classification. This will provide more reliable and accurate results.