TU WIEN Informatics

Identification of pests and diseases in Agriculture - conventional models vs Artificial Intelligence

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

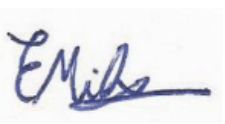
Enri Miho, BSc
Registration Number 00929003

to the Faculty of Informatics

at the TU Wien

Advisor: O.Univ.-Prof. Dipl.-Ing. Dr.techn. A Min Tjoa
Assistance: Dipl.-Ing. Mag. Dr.techn. Thomas Neubauer

Vienna, 23rd November, 2022



Enri Miho | A Min Tjoa

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Enri Miho, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 23. November 2022

Enri Miho

Acknowledgements

I would like to thank everyone who is working in the fields of machine learning and especially plant disease identification. Writing this thesis would not have been possible without the necessary data, made available in the context of numerous articles.
A special thanks goes to my supervisors O.Univ.Prof. Dipl.-Ing. Dr.techn. A Min Tjoa and Dipl.-Ing. Mag. Dr.techn. Thomas Neubauer for giving me the opportunity to work on this topic. I am particularly grateful to Thomas Neubauer for his valuable input on the topic.
I dedicate this thesis to the memory of my uncle, Vasil, who will always be an inspiration to me. Finally, I want to express my gratitude to my family for their endless support. They, and also my friends kept me motivated to work hard. My heartfelt thanks.

Kurzfassung

Der Pflanzenbau hat einen wichtigen Einfluss auf die Weltwirtschaft. In den letzten Jahren hat eine effiziente Produktion zunehmend an Bedeutung gewonnen. Dies hat vor allem mit der steigenden Weltbevölkerung zu tun, dass Probleme wie Unterernährung verursachen könnte. Krankheiten in der Landwirtschaft sind oft für Ernteverluste verantwortlich. Deswegen sind Modelle entwickelt worden, die zur Identifikation von Krankheiten und Schädlingen dienen. Diese Modelle basieren in den meisten Fällen auf der Verwendung von Bildverarbeitung in Kombination mit Wetterdaten. In der Zwischenzeit hat Künstliche Intelligenz an Popularität gewonnen, insbesondere weil es die Hardware von heute erlaubt. Dies ermöglicht neue Wege, um die Modelle zu verbessern.

Ziel dieser Masterarbeit ist es, konventionelle Modelle, die zur Identifikation von Krankheiten und Schädlingen dienen, mit Modellen zu vergleichen, die auf Künstlicher Intelligenz basieren.

Zuerst werden existierende herkömmliche Modelle untersucht. Danach werden neue Modelle definiert, die auf Künstlicher Intelligenz basieren. Dabei ist wichtig zu beachten, dass diese die gleichen Daten wie die herkömmlichen Modelle verwenden, um einen Vergleich zu ermöglichen. Die Modelle werden mit einem Prototyp evaluiert, welcher im Rahmen dieser Arbeit entwickelt wird. Der Prototyp ermöglicht nicht nur die Identifikation von Krankheiten. Es wird auch Wert auf die Erweiterbarkeit gelegt, sodass neue Modelle einfach hinzugefügt werden können.

Anschließend werden die Vergleiche zwischen den Modellen durchgeführt. Es wird gezeigt, wie Künstliche Intelligenz die Identifikation von Krankheiten und Schädlingen in der Landwirtschaft verbessern kann und was für Vorteile sie mit sich bringt. Das gilt nicht nur für die Genauigkeit der Identifikation, sondern es wird auch gezeigt, dass Prozesse automatisiert werden können, um schließlich eine Anwendung in der Praxis zu ermöglichen.

Abstract

Crop production in agriculture has an important impact on the global economy. In recent years, effective production has played a key role, considering that the world population is continuously growing, increasing the risk of malnutrition. Plant diseases are one of the biggest threats, being often responsible for crop losses. Traditional plant and pest disease identification models have been developed, which rely on computer vision algorithms and weather data. On the other hand, Artificial Intelligence has gained popularity in recent years, also due to better hardware capabilities, offering new ways to improve these models.

This thesis aims at comparing conventional plant disease identification models with Artificial Intelligence models.

First, traditional plant disease identification models are identified and explored. Next, Artificial Intelligence models are defined. These models use the same data as the traditional ones, making a comparison possible. In order to evaluate the models, a tool is implemented. It allows applying different models to identify and monitor plant diseases. It is built with extensibility in mind, allowing implementing and integrating new models. Finally, conventional and artificial intelligence models are compared. It is shown how Artificial Intelligence can improve plant disease identification and what benefits it brings to it, not only in terms of disease identification accuracy, but also allowing for an automated solution that can be used in practice.

Contents

CHAPTER 1

Introduction

1.1 Motivation

According to a study, "severe acute malnutrition contributes to 1 million deaths among children annually"[TGL+13]. Others state that "the world will need 70 to 100% more food by 2050"[GBC+10]. Taking this into account, the efficiency in food production becomes an important factor for the future, and it has to improve. Crops or plants that are cultivated in agriculture represent an important basis for food for daily use. These food crops include popular grains, such as rice and corn, and also vegetables and fruits that are harvested for human consumption.

One of the major challenges during cultivation that cause a decrease in productivity are plant diseases. These are mostly caused by pests or other environmental factors. A pest can be any kind of harmful insect. Identifying and eliminating these diseases and pests from plants is the key to an efficient food productivity.
In order to deal with plant diseases, farmers use traditional means, such as chemicals and pesticides. This approach is not selective, i.e. also the healthy plants will unnecessarily be affected by the usage of these pesticides. Taking and analyzing each plant one by one takes for the farmers too much time and does not scale well. Also, identifying diseases requires a good lab infrastructure, which is often unavailable. This is one of the main reasons why farmers apply pesticides to the entire crop field. There is, however, no guarantee that this method will always succeed. Sometimes, the only solution is to remove an infected plant from the field, in order to prevent the disease from spreading further across the whole field.

Because of this, methods that perform better are imperative. Agriculture is already undergoing a digital revolution. Terms such as the "Fourth Agricultural Revolution" and "Agriculture 4.0" have been coined to refer to the impact of technology in agriculture[FNJ19]. This includes the identification of diseases and pests by using technology. When it comes

to identifying a disease, the main challenge is to find methods that will have a high identification accuracy.
Using sensors, cameras and other technologies can speed up the process of identification. This way, the farmer can quickly find out which plants are infected. They spend less time searching for problematic plants and can effectively deal with them, helping the productivity to increase. The application of pesticides is more controlled, since less material will be wasted. This reduces the impact on the environment. Sensors can detect problems with plants immediately, which is important to prevent diseases from developing further. Also, the costs for dealing with the diseased plants will drop. Only an initial investment is required to integrate all the required technology into the farm.

1.2 Problem statement

The application of technology in agriculture brings a lot of benefits to the farmers, such as reduced costs and time. Models that deal with the problem of plant disease identification have been published. They are mostly based on *Artificial Intelligence* (AI)[Gre14], but there are other conventional techniques, such as *Image Processing*[NS20]. Most of these models expect high-resolution images of plants as an input for further analysis. Mostly, these images are captured using normal cameras, but sometimes, more sophisticated cameras are used. Statistical analysis, often based on environmental data, is applied for forecasting diseases.

It is important to choose a model that proves to be effective. Given some information about a plant, the model should be able to detect whether a plant has a disease or not, and also give information about the type of disease. Applying AI to solve this problem is currently actively being researched. New models have to be proposed from time to time.

In order to propose the right model, it is necessary to compare it with other models. Conventional approaches have their limitations. A drawback is that they cannot be always used as part of a real system that the farmers could use. Some manual work is needed to identify a disease. This is where AI comes into play. It can improve these conventional models and allows for an automated solution. It does not only help to solve the problem of the identification itself, but also gives insight into the features that characterize a disease.

1.3 Aim of the work

Traditional ways to analyze plants for infections are not very effective when it comes to time and costs. The aim of this work is to provide a better, quicker, effective and more intelligent way to obtain disease information about plants. Two problems are treated:

- forecasting whether a plant will have a certain disease or not

- after the disease has occurred, distinguishing healthy plants from infected ones, usually done by analyzing images

In order to solve these problems, AI models are defined. An important prerequisite for these models to work is the data, which is needed for training. Such data can be found on the internet[HS15][WZL+19]. The images have been captured using IoT-devices. Sometimes, however, training data is not available. In this case, the data set has to be manually created. In order to achieve this, different techniques need to be investigated and applied.

Another important part of this work is comparing the AI models with other (conventional) models. The goal is to identify these conventional models. A prototype is developed in a selected common programming language. It is used to evaluate the AI models. It is also extendable, i.e. new models can be added in the future.

Finally, the conventional models are compared with the AI models. The best AI models are selected for comparison. Key factors, when it comes to comparing the models, are the time that is needed to classify the data and the accuracy, i.e. how good the model is able to correctly identify a plant disease. Moreover, it is discussed how to fine-tune these models, so that they can give better results, and also what can be expected in the future.

The target group are farm owners who want to monitor their plants, but also people who want to research on applying AI for plant disease identification. It is expected that the presented models also have real use in agriculture. They can be implemented and installed on several IoT devices or drones, in order to monitor the crops. Together with other monitoring devices, they form a farm management system, which helps people managing their farms.

1.4 Methodological approach

This section presents the methodological approach applied to this work, in order to solve the described problem. It gives a short overview about how the research is carried out. Figure 1.1 depicts the methodology, which consists of the following steps:

1. **Literature review**
   Background information is required and must be collected first. This includes a general understanding on typical algorithms that are used in AI. Getting familiar with the types of diseases and pests for a plant is also important.
   Also, information about the state-of-art plant disease identification models is needed before proceeding further. This should help to get an idea on how to define models

2. **Model types**
   Next, research about model types is carried out. They group models together that solve similar problems. For instance, some models use leaf images to identify a disease, while others can use weather data to forecast a disease

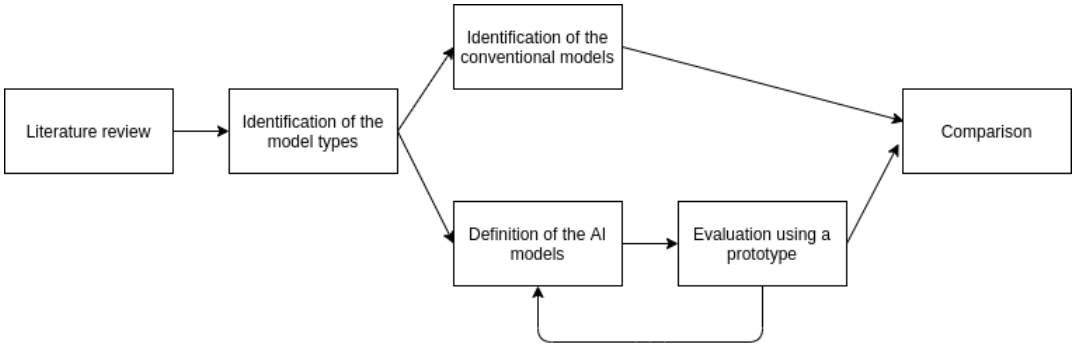


Figure 1.1: The methodological approach.

3. **Conventional models**
   For each model type, research about conventional models is conducted. It is important to keep in mind that the selected models are in a way comparable with AI models

4. **Definition of AI models**
   Then, for each model type, plant disease identification models are defined. These are researched during the literature review

5. **Evaluation**
   A prototype is implemented. It is used for model evaluation purposes. In order to evaluate a model, it is important to have a data set, with enough training and test data. If the evaluation results are not satisfactory, then the AI models are redefined

6. **Comparison**
   Finally, for each model type, the best AI models are selected and compared against conventional models. Several aspects of the models are compared, including both qualitative and quantitative ones. The results should also give insight into what features make a healthy plant different from an infected one

1.5 Structure

This section gives a brief description about the structure of this work. It mainly reflects the steps of the methodological approach. The current chapter gives an overview about this thesis, including the motivation behind it. It defines the problem and the methodology used to deal with the problem.

Chapter 2 provides the required fundamentals needed in this work. It defines what plant diseases are and gives some information about the integration of technology into agriculture and its advantages. It also gives an introduction into the field of AI.

In Chapter 3, similar work is researched, which is mainly based on the fundamentals provided in Chapter 2.

Chapter 4 identifies model types and conventional models that are used to identify diseases in plants. These models are used later for comparison purposes.

In Chapter 5, the AI models are defined. This and the upcoming chapters represent a central part of this work.

A prototype is implemented and used to evaluate the AI models. This is described in Chapter 6.

In Chapter 7, the conventional models from Chapter 4 are compared with the best AI models from Chapter 5.

Chapter 8 finalizes this thesis and gives a short summary about the results. Future research possibilities are also discussed.

CHAPTER 2

Background

This chapter gives information about the fundamentals of this work. Since it deals with the identification of plant diseases, a general overview about plants and diseases is required and provided in Section 2.1. To make the identification work, algorithms from the field of study of AI are used. It is important to know how these algorithms work and how they can be optimized. Section 2.2 lists and describes the most important ones from machine learning (a sub-field of AI). It is expected that the reader is familiar with state-of-the-art programming languages, which are required to develop the models and the prototype.

2.1 Plant diseases and Smart Farming

A *plant* is a living organism that grows in the earth and has roots and leaves. A *food crop* is a plant that can be grown and harvested. The term plant is used interchangeably with the word crop. Important plants include groins, vegetables, fruits, and more. A *plant disease* is something that will prevent the vital functions of a plant. *Pathogens* are sometimes created by pests such as insects. *Plant pathology* is the field of study that deals with plant diseases that are caused by pathogens. Plant diseases are often triggered by a combination of

- environmental factors such as temperature, humidity, wind and light
- pathogens
- the susceptible host

This concept is known as the *plant disease triangle*[Isl18]. If one of the three factors is missing, then the disease will not occur.

Figure 2.1 shows an image of a grape leaf, infected with the black rot disease. While in its early stages the disease cannot be seen yet, after a certain amount of time, it starts to become visible to the naked eye. When this is the case, it is possible to distinguish healthy from unhealthy plants, due to color changes and other recurring patterns. Sometimes, the morphology of a plant is affected as well. For some diseases, a temperature change can be noticed in the leaves.

Figure 2.1: Black rot on a grape leaf[HS15].

Smart farming (sometimes also known as *precision agriculture* or *precision farming*) refers to the integration of technology into agriculture, with the goal to facilitate and increase the productivity of crops, decrease the costs, and also reduce the environmental impact for the farmers[VNF+20]. Different hardware implementations exist, such as:

- sensors that are used for monitoring temperature and humidity of a plant, and also the water temperature, which is used to feed the plant. Some sensors do not require making physical contact with the plant. This way of working is known as *proximity sensing*[Joh07]
- cameras and satellites, making *remote sensing*[FRJ86] possible. Remote sensing is the acquisition of information about plants/fields, without making physical contact, from long distances, allowing to cover large areas of interest. It has many applications, such as locating floods or fires, but can also identify fields that are affected by a certain disease. A drawback of remote sensing is that it is a very expensive technology
- other common technologies such as GPS for determining the exact location of the plants
- robotics, performing some of the work that the farmers do. This includes watering the fields, spraying pesticides, and more

These components are equipped with network modules, i.e. they are interconnected. The acquired data is fed into a unit that is responsible for analyzing it and can give detailed

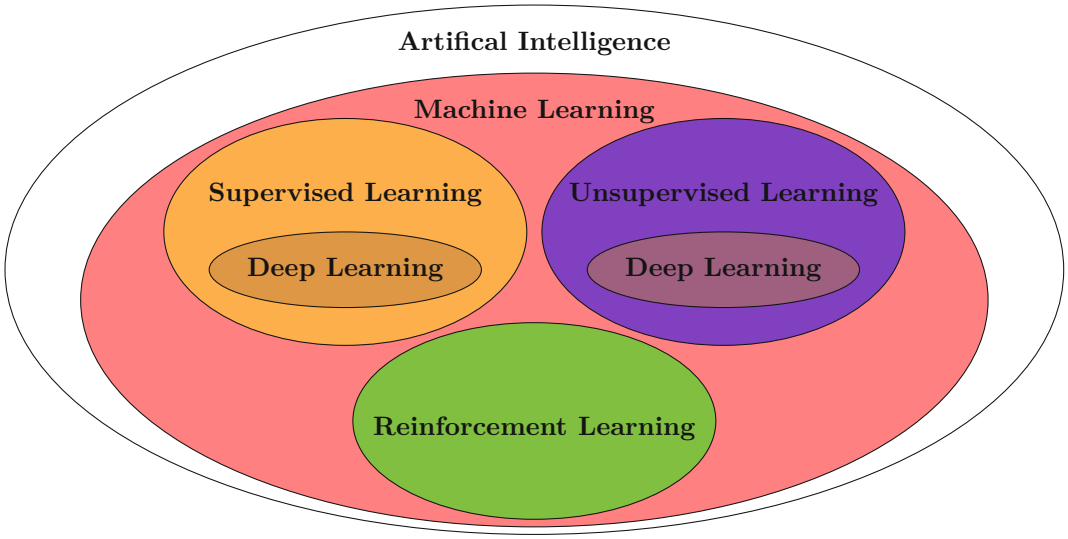


Figure 2.2: Artificial Intelligence and its subsets.

information about the plants. For instance, it can help farmers to use the right amount of water and other materials. By using AI, these decisions can be automatically made, without the intervention from the growers. Another use case is identifying a disease of interest.

2.2 Plant disease identification AI algorithms

AI refers to the ability of computers to perform tasks that require human intelligence. One important subset of AI is *machine learning*[Sam59]. When not using machine learning, specific instructions have to be provided to solve a problem. In contrast, in machine learning, solving a problem is learned by training a model with solutions to this problem. This is, in many cases, easier than implementing an explicit complex algorithm. The availability of the data (needed to train a model) is imperative. In some situations, this data can also be generated, leading to an improvement of the accuracy of the model.

Machine learning is an old concept, but has only started to gain popularity in recent years. Its algorithms require a lot of computation power, not available in the past. The most important fields of study of machine learning are *supervised learning* and *unsupervised learning*[SA13]. One widely used model is *deep learning*[LBH15]. This learning model is based on artificial neural networks, which try to simulate the way the human brain works. Depending on the way it is used, deep learning can be supervised or unsupervised. Figure 2.2 depicts the hierarchy of AI and its subsets. Machine learning represents a very large subset of AI, which is why they are often considered the same. Another subset of machine learning is *reinforcement learning*[KLM96]. In this area, software agents perform actions and depending on whether they are successful or not, they get rewarded

or penalized. The goal is to maximize the reward. However, reinforcement learning is not very suitable for identifying plant diseases.

2.2.1 Supervised Learning

In supervised learning, a set of *input data* (training data) is given, for which it is known what the *output data* should be. This data is used to train a model, which can later calculate output data for some given *new* input. The output is known as the *label* or *class*, which is why the terms labeled or classified data are often used. Each training *element*, *sample* or *example* of the input data consists of one or more *features*, and the goal is to find a connection between these features and the labeled data. Unlabeled data is known as *test data* and is used to test how well the model performs.

More formally, consider $N = ((X_1, y_1), ..., (X_n, y_n))$ to be the ordered set(because of the indices)[1] of all n labeled training examples, where X_i represents the feature vector of the training example i and y_i the output for that vector. In supervised learning, a *hypothesis* function $h(X)$ must be constructed. It computes for each training example a close value to the labeled output, i.e. the error or difference between the computed output and the expected labeled output is as small as possible. This function is used to compute or predict the output for some new unlabeled data.

Finding the right function is a minimization problem, which depends on some parameters. The success of the algorithm depends on these parameters too. Literature that describes how to find and optimize these parameters exists[SCZZ19]. Determining $h(X)$ is beyond the scope of this thesis. However, a short intuition is given. The accuracy of $h(X)$ is measured using a *cost function* $J(\Theta)$, where Θ represents the *parameter vector*, which has the same number of elements as the feature vector X. These parameters are also known as the *weights*. The cost function quantifies the error of $h(X)$, i.e. by how much the results produced by the estimated function $h(X)$ differ from the desired output. For instance, for linear regression, a machine learning algorithm from statistics, the mean squared error is used as a cost function, which represents the average difference between the results produced by $h(X)$ and the expected outputs. The goal is to minimize the error computed by the cost function. This is nothing but finding the minima of the cost function. One popular optimization algorithm that achieves this is the *stochastic gradient descent*[BB07] algorithm, which tries to find the minima of the cost function in an iterative way. Each parameter θ_i is randomly initialized at the beginning. With every iteration step, each parameter θ_i is updated using $\theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\Theta)$, where α is the *learning rate* and Θ the parameter vector from the previous iteration.

There are two types of supervised learning problems: *classification* and *regression* problems[Alp14]. In classification, the class for some input data is computed. This class is estimated within a discrete set of values. A typical classification problem would be classifying handwritten digits, given as images, into ten classes, from zero to nine. In regression, the value that is computed for an input is continuous. Predicting house prices,

[1] An ordered set is considered as a vector.

depending on some features of a house, such as the number of rooms and the location, would be a typical example.

One of the most popular regression methods is *linear regression*[Fre05]. The goal is to find a linear function for the input data that computes the desired output. The vector of the input data consists of only one feature. Figure 2.3 shows an example. The straight line fits to the data points, minimizing the distance to all points as good as possible.

More generally, if the input data has more than one feature, then the method is known as *multivariate linear regression*[Fre05]. The function is given as $h(X) = X \odot \Theta$, where $\odot$ represents the sum over the element-wise multiplication for two vectors. The goal is to compute the parameter vector Θ.

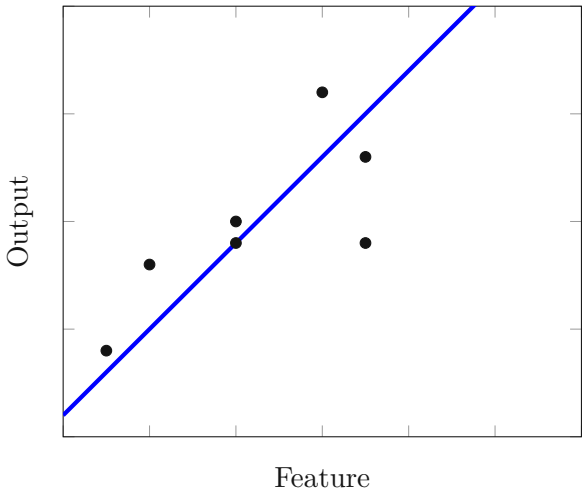


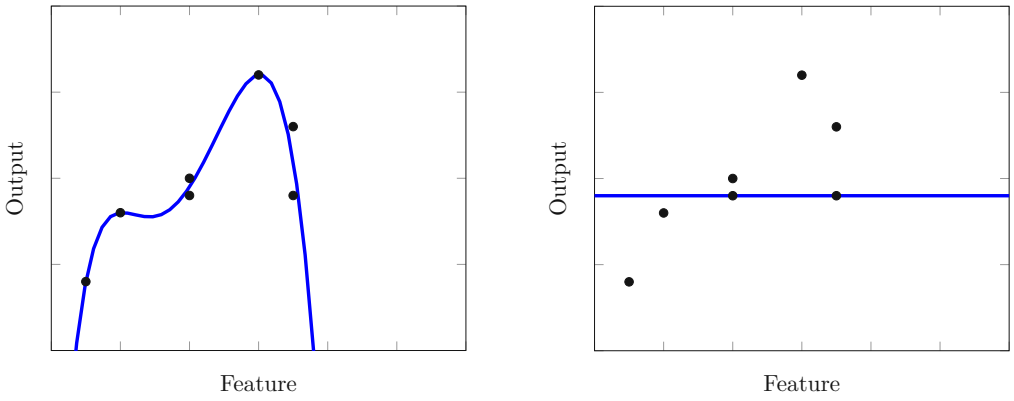
Figure 2.3: Linear regression for a data set with one feature.

Sometimes, it is not possible to find a linear function that will fit to the data well. To create a non-linear function, features are converted to polynomial ones and added together: $h(X) = \Theta_0 + \Theta_1 \odot X + \Theta_2 \odot X^2 + \cdots + \Theta_k \odot X^k$, where k is the polynomial degree.

However, fitting to the data too perfectly might cause this function to not generalize to new data. This problem is known as overfitting. A function that does not fit the training data well is said to underfit the data. Refer to Figure 2.4 for a visual description. In order to prevent overfitting, regularization can be used. By adjusting the importance of some parameters, it reduces the impact of some features, which might be responsible for overfitting.

Logistic regression[Cra02] is a classification method. Since linear regression computes continuous values, it cannot be used for classification purposes. These values have to be mapped to a discrete set of values. In order to achieve this, the logistic function[2]

[2]The logistic sigmoid function is given as $f(x) = \frac{1}{1+e^{-x}}$ and maps numbers to an interval $[0, 1]$.



(a) Overfitting (b) Underfitting

Figure 2.4: Overfitting and underfitting.

is used: $h(X) = \frac{1}{1+e^{-\Theta \odot X}}$, where $\Theta \odot X$ is the value computed by the (multivariate) linear regression function. The computed output indicates the probability that the input will belong to one class or another. The default probabilistic threshold is 0.5. However, sometimes, changing this parameter might improve the algorithm. This is the way this method works for binary classification, i.e. the number of classes is two.

For the case of k classes, where $k > 2$, for each class, a binary classification problem is defined, where the two classes are:

- the class itself
- the union of the other $k - 1$ classes

For each classification problem, the probability is computed that a certain input belongs to a class. From the computed probabilities, the class with the highest wins. This method is also known as the *One-vs-all method*[Bis06].

In order to prevent overfitting, but also underfitting, logistic regression uses regularization. One popular technique is *L2 regularization* or *ridge regression*[HK00]. The term $\frac{\lambda}{2n}\sum_{i=1}^{m}\theta_i^2$ is added to the cost function $J(\Theta)$, which has to be minimized using an optimization algorithm, such as stochastic gradient descent. The hyperparameter λ is known as the *regularization parameter*. A large value of λ penalizes more the m parameters of the vector Θ and will lead to underfitting, while a small value might lead to overfitting.

In this thesis, the function $lr(T, Y, o)$ computes the hypothesis function $h(X)$ for logistic regression and has the following parameters:

- $T = (X_1, \cdots, X_n)$ is the vector of n training examples, where each example X is a feature vector

- $Y = (y_1, \cdots, y_n)$ is the vector of the corresponding classes
- o is the optimization algorithm. Gradient descent is mentioned as the default method to minimize the cost function. However, there do exist other methods that can achieve the same, but in a more optimized way:
  - *L-BFGS*[ZBLN97], which is a quasi-Newton method that is used to find the minima of a function. L stands for limited, meaning that this method is able to work well with limited memory
  - *Stochastic Average Gradient (SAG)*[SRB13] and *SAGA*[DBLJ14], which work well with large data sets

  A more detailed description of these optimization algorithms is beyond the scope of this work. They converge much faster than stochastic gradient descent and work well with numerous features. o can be respectively one of `"lbfgs"`, `"sag"` or `"saga"`

The function uses L2 regularization with $\lambda = 1$. The logistic regression model[3] from Scikit-learn[PVG+11] provides a reference implementation. As a cost function, the negative log-likelihood[4] is used. At the beginning, the weights of the vector Θ are randomly initialized.

Support-vector machines[CV95] are popular models used for classification problems, but also for regression. For classification, the objective is to classify the data, using an n-dimensional hyperplane, where n represents the number of features of the data. In simple terms, this hyperplane must separate the data the best way possible, i.e. the width of the hyperplane must be maximized. This width is also known as the *margin*. Maximizing the margin helps to prevent overfitting. In case the data is linearly separable, the two ends of hyperplane are given as follows:

- $\Theta \odot X - b = 1$, i.e. values greater or equal to 1 belong to the class with label 1
- $\Theta \odot X - b = -1$, i.e. values less or equal to -1 belong to the class with label -1

The original labels need to be mapped to labels 1 and -1. If there are more than two classes, then the One-vs-all method can be used. The goal is to find values for Θ and b such that the margin, i.e. the distance between these two hyperplanes, is maximized. Explaining how to do this is beyond the scope of this work. However, there are important hyperparameters:

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html`

- the regularization parameter C controls how much misclassified data should be allowed. The smaller C, the larger the margin will be, allowing more data to be misclassified

- sometimes, the data cannot be separated using linear functions. This is where *kernels*[VTS04] come into play. A kernel applies a transformation to the data, allowing this data to be separable. The parameter γ decides how much influence the data has on the margin. A small γ will have an impact on the curvature of the hyperplane

The function $svm(T, Y, k)$ computes the hypothesis function $h(X)$ for support-vector machines and has the following parameters:

- $T = (X_1, \cdots, X_n)$ is the vector of n training examples, where each example X is a feature vector

- $Y = (y_1, \cdots, y_n)$ is the vector of the corresponding classes

- k is the kernel. Three kernels can be configured:

  - the *linear kernel*, as the name suggests, works well, if the data is linearly separable. It is one of the most common kernels and can perform well, if the number of features is high. The regularization parameter C has the value 1.0
  - the *polynomial kernel*, which is used for non-linear data. It tries to add polynomial features, usually by combining the features together. Increasing the polynomial degree might help to make the data separable (since it might not be linearly separable as it is in its original form). C has the value 1.0 and γ the default value from the implementation of Scikit-learn[PVG+11]
  - the *radial basis function (RBF) kernel*, which also adds new features by combining the existing ones, but the new features it introduces are different. It can be more time-consuming, but is often more accurate. C has the value 1.0 and γ the default value from the implementation of Scikit-learn[PVG+11]

  k can be respectively one of `"linear"`, `"poly"`, or `"rbf"`

The hinge loss[5] is minimized using stochastic gradient descent. The support-vector machine model[6] from Scikit-learn[PVG+11] provides a reference implementation.

[5] https://en.wikipedia.org/wiki/Hinge_loss

[6] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

2.2.2 Unsupervised Learning

Unsupervised learning works without knowing the belonging class of a training example, the data is said to be unlabeled. The goal is to find some structure in the data, without knowing anything about it in advance. A typical example of an unsupervised learning problem would be grouping related news stories together or email type classification, based on the content(spam/social/promotions/etc.). There are two important groups of unsupervised learning approaches:

- clustering, which groups similar data together. One popular clustering algorithm is k-means. This algorithm groups the input data into k clusters, usually by calculating the distances between the data points
- anomaly detection is a technique that identifies rare data elements in a data set. It can be used for data pre-processing purposes, regarding the filtering of abnormal data. It is typically used to detect frauds or anomalies in manufacturing. Furthermore, it tries to determine high and low probability features in a data set

2.2.3 Deep Learning

Deep learning is a method that is based on *artificial neural networks*[Neg01]. It can be used to solve supervised and unsupervised learning problems. Neural networks have become a state-of-art technique for many problems. Using linear regression or support-vector machines to learn from data, which has many features, might be computationally expensive, due to the polynomial terms.

Artificial neural networks try to simulate the way neurons in the human brain work. The brain contains lots of interconnected neurons. The neuron's input wires are called dendrites and the output wires are called axons. A neuron will receive signals through its dendrites, process these in some fashion, and use the axons to send signals to other neurons. An artificial neural network consists of many connected neurons.

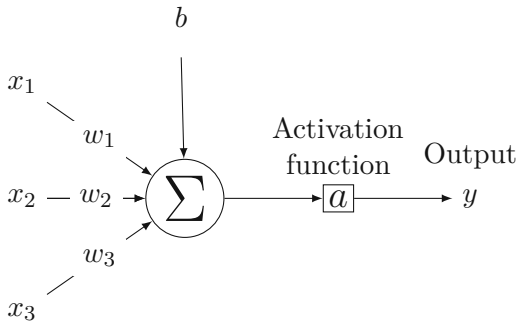


Figure 2.5: An artificial neuron.

Figure 2.5 depicts an artificial neuron. It consists of a number of inputs (dendrites), does some computation on these inputs and finally produces an output (axon). The function

that performs the computation is known as the *activation function*. The weighted sum over all the inputs is computed, and the result is fed into the activation function a. The logistic sigmoid function and the rectifier[7] are popular activation functions. To shift the activation function, a *bias term* is often added to the weighted input sum. The calculation a neuron does is given formally as $y = a(\sum_{i=1}^{n} w_i x_i + b) = a(\Theta \odot X + b)$, where w_i is the weight(parameter) for input x_i and b the bias term.

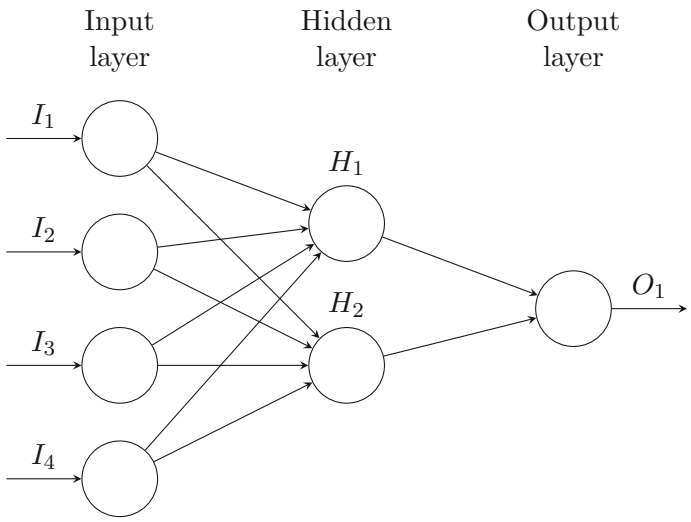


Figure 2.6: An artificial neural network with three layers.

Figure 2.6 shows an example of an artificial neural network with three layers. An artificial neural network consists of an input layer, zero or more hidden layers and an output layer. Each layer contains a number of neurons. Each neuron of a layer l is connected in some way to neurons of the next layer $l + 1$. An output y produced by a neuron at layer l serves as an input for a neuron at layer $l + 1$. Similar to other machine learning models, the goal is to find the parameters that fit to the data the best. The parameters in this case are the weights. For each training example, the final output produced by the neural network has to be as close as possible to the expected output (in case of supervised learning). This is done by adjusting the weights. The process of calculating the final output (starting off from the input values) for a neural network is known as *forward propagation*.

More formally, a layer function maps an output vector from a previous layer (or inputs) to another output vector, to be used by the next layer.

A *fully-connected layer* is defined as $fc(I, in, out, a)$, where:

- I is a vector with in elements, each from $\mathbb{R}$, i.e. there are in neurons from the previous layer

[7]The rectifier or ReLU (Rectified Linear Unit) computes the maximum positive value for an input and is given as $f(x) = max(0, x)$.

- *out* is the number of produced outputs, i.e. the number of neurons for the layer. Each neuron receives *in* inputs and the respective *in* weights must be learned
- a is the activation function for each neuron

In simple terms, each neuron from the previous layer is connected to each neuron of this layer. The output of the function is a vector with *out* elements, each from $\mathbb{R}$. For a reference implementation, see the linear transformation[8] from PyTorch[PGM+19].

In order to train a neural network with a hypothesis function h, for each training example X, the following steps are performed:

1. by using forward propagation, the prediction $h(X)$ is computed
2. a cost function $J(\Theta)$ is selected
3. for each parameter θ of the cost function, the partial derivatives(or gradients) $\frac{\partial}{\partial\theta}J(\Theta)$ are computed. This is known as *backward propagation*
4. using an optimization algorithm, such as stochastic gradient descent, $J(\Theta)$ is minimized, with the help of the computed gradients. This step will update the weights in Θ for h

Convolutional neural networks[LBBH98] have proven to be very effective in image analysis and classification. Using traditional neural networks, where each layer is fully-connected, is very expensive from a computational point of view. Also, in image classification, dependencies between pixels are important, in order to detect recurring patterns. Using a traditional neural network, this information is lost. The same holds for other learning algorithms, such as logistic regression and support-vector machines. A typical convolutional neural network consists of the following layers:

- *convolution layers* apply *filters*(also known as *kernels*) of the same size, by sliding them through the entire image. The *stride* defines by how many pixels the kernel should be moved. *Feature maps* are produced, which indicate where in the image the filter matches the most. A filter is nothing but a set of weights, which have to be learned. The function $conv2d(I, in, out, kernel, stride, a)$ is responsible for computing the convolution operation where:
  - I must be a 3-dimensional vector with shape[9] (l, h, w) (there are inputs from $(l * h * w)$ neurons), where l is the number of layers or channels of the image(in case of RGB $l = 3$), h the height and w the width. This vector might be the initial input of the neural network, or it might be an output produced by another (hidden) layer

[8] `https://pytorch.org/docs/stable/generated/torch.nn.Linear.html`

[9] The shape of a vector is the number of elements for each dimension. The shape is represented as a vector itself. The lowest dimension contains elements from $\mathbb{R}$.

- *in* must match l
- *out* is the number of (filtered) images to be created
- *kernel* is the size of the kernel that should slide through the image, represented as (k_h, k_w), with height k_h and width k_w
- *stride* is the stride
- a is the activation function

In order to create a filtered image, for each image layer and kernel position, the convolution[10] operation is applied regularly. After this is done, the filtered image F is created. The height of the image will be $h_f = (h - k_h)/stride + 1$ and the width $w_f = (w - k_w)/stride + 1$, i.e. a 2-dimensional vector with shape (h_f, w_f) is produced.

Formally, let m and n be the position of the kernel in the image I. For each position, the convolution is computed by a neuron as

$$F_{m,n} = a(\sum_{d=1}^{l} \sum_{i=m}^{m+k_h} \sum_{j=n}^{n+k_w} w_{d,i,j} I_{d,i,j} + b)$$

This process is repeated *out* times, i.e. the resulting vector will have the shape (out, h_f, w_f) and the number of neurons will be $out * h_f * w_f$. For a reference vector implementation, see the convolution operation[11] from PyTorch[PGM+19]

- *max pooling layers* reduce the resolution of an image by sliding patches over the image and applying max pooling[12]. Visually speaking, this allows preserving the spatial invariance, i.e. forms will still be visible as before, they might just be slightly deviated and with a worse quality. The function $maxpool2d(I, kernel, stride)$ reduces the size of the images, but does not change the number of them. The size of the new filtered images is computed the same way as in $conv2d$. For each position of the kernel K, max pooling takes the highest value of K and sets $F_{m,n}$. This process is repeated for each input vector

  Technically, the max pooling layer does not use the concept of neurons, there are no weights to learn. However, it is treated as a layer in literature. A reference implementation[13] is provided by PyTorch[PGM+19]

- fully-connected layers

These layers can be combined in different orders and occurrences, in order to improve the algorithm.

[10] Given an image, the convolution operation is the sum over all weighted pixel values.
[11] `https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html`
[12] Max pooling takes the highest pixel value from an image.
[13] `https://pytorch.org/docs/stable/generated/torch.nn.MaxPool2d.html`

CHAPTER 3

Related Work

This work aims at identifying conventional plant disease identification models, to be compared with AI models. In this chapter, related work in the research area of this thesis is presented. Contributions and trends are discussed. This chapter is divided into four parts:

- Section 3.1 describes the plant diseases that can be predicted/identified by the models
- Section 3.2 discusses the contributions of this work to the literature. This includes the research fields that it touches and what value this work adds to them. Plant disease identification is a research field that is still growing and has gained attention in recent years
- Section 3.3 gives an overview about the existing research in the area of plant disease identification. State-of-the-art techniques are shortly listed
- Section 3.4 presents similar work that solves a similar problem to the one defined in Section 1.2. This section tries to motivate why comparing AI models with conventional models should be researched more

3.1 Plant diseases

Plant diseases that are taken into consideration are divided into three categories, depending on their causes:

- **Category of plant diseases that are favored by cold weather.** Two are the diseases of interest:

  - late blight of tomato, which is caused by the pathogen Phytophthora infestans, is favored by cool and wet weather. Initially, infected leaves become irregular in shape and lesions are identified as spots, colored from light to dark brown. The entire tomato fruit may be attacked later too
  - stripe rust of wheat is one of the major wheat rust diseases, which develops with low temperatures in late winter. Caused by the fungus Puccinia striiformis, this disease is characterized by yellow streaks on the leaves. The fungus grows in the plant and produces spores, causing greater loss of water in the plants and making them collapse

- **Category of plant diseases that are favored by moderate to hot weather.** Three diseases are identified:
  - northern leaf blight of corn is a disease that can develop with moderate temperatures and some wetness. The pathogen is the fungus Exserohilum turcicum and symptoms are canoe-shaped lesions in brown color. The diseased areas allow the fungus to grow and develop further, causing serious harm to the plant
  - Fusarium wilt is caused by the fungus Fusarium oxysporum and development is favored by hot and dry weather. It may cause different types of tea and cucumber diseases, and others. This disease restricts water flow in the plant, making the leaves to wilt and turn yellow. Due to lack of water, the yellow areas may have a higher temperature. Deficiency of water may cause serious damage to the plants
  - sheath blight of rice is a disease, which occurs in areas with high temperatures, around 30 degrees. It is a fungal disease, caused by Rhizoctonia solani. Lesions have the form of an ellipsis and are in gray color

- **Category of plant diseases that are caused by pests.** Chlorosis occurs due to lack of chlorophyll, which is needed to perform the photosynthesis. This deficiency of chlorophyll can be caused by missing amounts of iron in the plant. The chlorosis virus can be carried also by pests, such as whiteflies, and transmitted into the plants. This disease may affect different types of plants. Typically, the affected leaves become yellow over time and the entire plant may die

3.2 Contributions

Section 1.1 motivates why identifying and detecting infected plants is of great importance in agriculture, from a cost and productivity perspective. Detecting a disease early can have an economic impact. In recent years, identifying plant diseases is receiving a lot of attention and is actively being researched. Mostly, algorithms from the field of AI are used. The mathematical approaches behind these algorithms have existed for a long time, but only during the last years the computational power has increased up to a point that it is possible to run these approaches and make practical use of them.

In the context of this work, different AI methods are implemented for plant disease identification. In Chapter 4, an overview is given about what kind of conventional models can be used to solve this problem. It is shown in the next chapters how AI can be used to improve these conventional models. This is done by comparing these models with each other, quantitatively, but also qualitatively. The defined models that are based on machine learning and convolutional neural networks represent a good contribution to the (still evolving) research field of plant disease identification.

Plant disease identification is a special area of interest in precision agriculture (farming), which deals with the integration of technology into the farming processes. The methods in this thesis can be combined and become part of these processes, contributing to an improved farming management system, allowing to monitor the development of the plants.

Convolutional neural networks are very useful when it comes to the extraction of features from an image. In the concrete case, these features can represent characteristics of a disease, such as color, holes and other patterns. It can help to discover *new* features that characterize a disease. Consequently, it is of particular interest for people who are diagnosing diseases manually in a laboratory. This is an important contribution to the research field of plant pathology, which is the scientific study of the development of plant diseases.

Finally, it is worth mentioning that the presented approaches in this work are not only limited to researching purposes. They can be used also in practice. This depends, however, on the available data, since it is required for training purposes. If this data is not available, it has to be acquired in some fashion. The approaches could be implemented as a mobile application or as part of a broader monitoring system for large fields of plants. Robots could make use of these techniques, in order to selectively treat or eliminate infected plants.

3.3 Current trends

Common ways of identifying plant diseases include serological and microbiological techniques, or just visual diagnostics. Since these methods are time-consuming and require experienced people, better approaches are imperative. Most of the current efforts for detecting plant diseases have been made using algorithms from AI. The approaches use convolutional neural networks and the algorithms are trained with images of leaves.

Selvaraj et al. [SVR+19] developed a banana disease detection system, in order to support farmers. Oppenheim et al. [OSET18] similarly did this for tomato diseases, by analyzing the leaves. Others did the same for various plant diseases. Venkataramanan et al. [VA19] gave a general overview about different convolutional neural networks techniques and identified their advantages and disadvantages. Another popular method are support-vector machines. Iniyan et al. [IJM+20] used support-vector machines, but also artificial neural networks for the detection of common crop diseases, such as blight

and rust. Jasim et al. [JAT17] compared the performance of support-vector machines on detecting infected leaves. Prathusha et al. [PMS20] used k-nearest neighbors algorithm and suggested that it is one of the most successful algorithms. Sharma et al. [SMS20] used Bayes classifiers to identify rice diseases.
Also, unsupervised learning has been used to identify plant diseases. Sankaran et al. [SVN20] used k-means clustering to detect infected leaves. Türkoğlu et al. [tH19] made use of the presence of pests in plants, besides analyzing the plants. Kaur et al. [KPG18] gave information about state-of-the-art disease identification techniques, which are trained with leaf images.

Applying machine learning is not only limited to using digital images (from a close range) as an input. Rocha et al. [RAPSN20] used satellite images to train different machine learning models, in order to detect coffee crop diseases. Nagasubramanian et al. [NJS+19] used hyperspectral images. These are useful for situations where a disease is not visible yet to the naked eye.
Another technique is the identification using thermal images, where pixels in a certain color indicate the temperature of some part of a leaf. Sometimes, there exists a correlation between the temperature in regions of a leaf and the presence of a pathology. Oerke et al. [OSDL06] used regression models for this problem. Thermal imaging still has the potential to score good results in detecting some diseases.

There do exist other identification ways, when it comes to forecasting a disease. One methodology made use of weather data. Shah et al. [SPDWM19] used regression models based on series of weather data over time to predict plant disease epidemics. Khattab et al. [KHI+19] defined a monitoring system for forecasting diseases. A weather station equipped with different sensors such as temperature and humidity sensors was used to collect data over time and to feed a prediction algorithm. Several attempts have been made to use fluorescence spectroscopy to analyze the reflectance of the plants, since it can be related to the presence of an infection. Saleem et al. [SAAB20] used laser induced fluorescence spectroscopy, in order to predict diseases in grapefruit plants.
Mahlein at al. [Mah15] discussed some future techniques regarding plant disease detection, not necessarily based on AI.

3.4 Similar work

From the previous section, it is noticeable that the majority of the approaches is focusing on the identification of diseases. It is also of particular interest to compare approaches with each other, for instance in terms of accuracy, but not only. Comparative work is also present in this field of research. Mostly, AI models are compared with each other. Academic writing that compare these models with others that are *not* based on AI (also known as conventional models in the context of this work), rarely exist. First, comparative research gives details about what the advantages and disadvantages of a model are and also, whether to choose a model over another one or not. Concretely, it shows the benefits of the application of AI techniques. Second, nearly all the models

are using close-range leaf images as an input. More models need to be explored and compared. Third, most models are focused too much on identifying a disease *after* it has appeared, i.e. it is visible to the human eye. Forecasting a disease can have an important impact on productivity and economy. The approaches need to be built with practical use in mind as well. For instance, it should be possible to use them in a real farm.

Nevertheless, comparing AI models with each other is also valuable for research, since it describes methods on how to compare models. Sandhu et al. [SK19] discussed and compared plant disease detection techniques, with close-range leaf images as input data. Pattnaik et al. [PP20] also reviewed advanced machine learning techniques for the identification of various plant diseases. Barbedo [Bar20] discussed different machine learning approaches using proximal images and how these approaches can be improved. Ngugi et al. [NAAZ20] discussed recent advancements in image processing methods.

CHAPTER 4

Conventional Models

This chapter describes the conventional models that are compared with AI models. First, the concept of a model type is defined, with respect to plant disease identification. Models give a concrete implementation for these model types. Then, the conventional model types are described, which are nothing but a subset of general model types. Conventional model types do not use methods from AI. For each conventional model type, models that implement them are shortly listed and summarized in Table 4.1. Finally, some other model types are described, with the goal to extend this work further in the future.

4.1 Definitions

A *plant disease model type* is an abstract function which, given some input, returns information about a plant. It may predict or identify the presence of a disease. Alternatively, it may also return information about the *severity* of the disease. It gives a high-level description for a plant disease. This function is abstract, it describes high-level details about the inputs and outputs, i.e. (informal) constraints that have to be fulfilled. A disease model type may have one or more *disease models*. Each model defines or implements a concrete function for the model type. This function complies in some way with the required high-level rules that are set by the model type.
One important limitation of the models that are implemented in the context of this work is the fact that, in general, they deal with only *one* disease (mostly for one type of plant) at a time, i.e. they are not able to identify or predict multiple diseases, given some plant information. Models that can detect multiple diseases are therefore not discussed further. However, it should be possible to combine these disease-specific models so that they form a broader disease model. There are challenges regarding these kinds of models, which have to be addressed. Certain diseases are visually similar and easy to mix up. Also, many diseases are caused by similar environmental changes, for example, when it comes to predicting them.

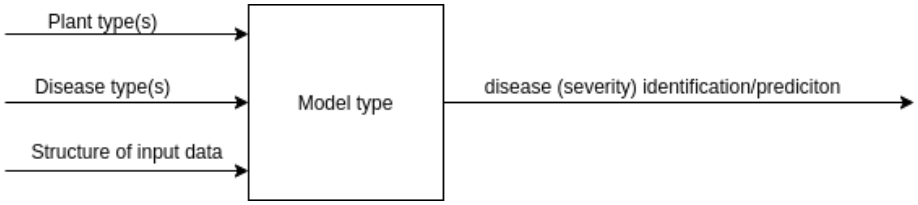


Figure 4.1: A plant disease model type is an abstract function that identifies/predicts a plant disease (severity), given some input data.

To give the reader a better intuition regarding disease model types, the ideas are depicted in Figure 4.1. A disease model type has the following components:

- the type of plant(s) that should be checked against a disease
- the disease(s) of interest
- the input data, which has to be acquired in some fashion. Acquiring this data is not always easy and often represents a major challenge. The structure of the input data is a constraint that distinguishes one model type from another

The input data can be obtained in different ways and can have one of the following types:

- weather data about the location of the plant
- thermographic (thermal) images of plant leaves
- images of pests on leaves, from a close range
- images of leaves, also from a close range
- satellite images of fields

This list is not exhaustive. However, it represents the most important input types. The first three types can be used to predict a disease, the others are used for identification purposes, i.e. after the disease is present and visible to the naked eye. Therefore, depending on the inputs, the model type produces two kinds of outputs, answering one of the two questions:

- *is* a certain disease present or not, and if possible, with what severity?
- *will* a certain disease be present or not, and if possible, with what severity?

The input and output data distinguishes a model type from another. Two categories of model types do exist:

- model types that can identify a disease
- model types that can predict a disease

A *conventional disease model type* is also a disease model type, but it has some limitations, i.e. it is a subset of general model types. The defined models for conventional model types are limited to not using methods from AI. This is to clearly distinguish them from AI models, which will be used later for comparison purposes. Since these conventional disease model types are always used in the context of plant diseases, the shorter term *conventional model type* is used. The terms model and model type are sometimes interchangeably used. It should be clear from the context which one is meant.

There are other traditional ways to identify plant diseases. One way is visual diagnosis, which requires human experts, who manually try to identify pathogens or disease symptoms. Other methods include microbiological, molecular and serological techniques. These traditional methods are not discussed further, nor they are used for comparison purposes. First, they are time-consuming and not very efficient. In order to be more efficient, there are specific guidelines that can be followed, but it still remains a major problem. Second, they are not technology-based, i.e. it is difficult to compare them directly with methods from AI, from a qualitative and quantitative perspective.

4.2 Conventional model types

In this section, model types are described, for each type of input data.

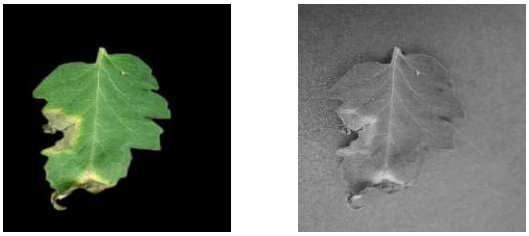
4.2.1 Conventional model type based on images from a close range

One of the most common model types uses images as the input. Models should have the following characteristics:

- as an input, images show leaves of plants from a close range, but in some situations also the whole plant
- as an output, they should identify whether a disease is present or not

The images are acquired from a close range, therefore they can show very detailed information. Data sets for this purpose, such as the PlantVillage data set[HS15], are available. The acquisition of the images can be done in different ways. A common method is to manually capture the images, using a digital camera. Since this is a very time-consuming method, the process can be automated by using robots that regularly capture the plants. Usually, images come in RGB format.

Conventional models usually use computer vision algorithms to identify a disease. They analyze visual aspects (also known as features) of the leaves or plants. First, a feature of interest is defined, which distinguishes healthy leaves or plants from infected ones.



(a) (b)

Figure 4.2: Segmented (a) and grayscale (b) images of late blight in tomato leaves, from the PlantVillage data set[HS15].

Depending on the plant disease, different features might be selected. Common symptoms of a disease include change in color and form of a leaf. The algorithm is then implemented around these features.

In order to make the models produce better results, sometimes it is necessary to do some pre-processing on the input data. Since the background of the image is of low interest and can also cause problems detecting a disease, it is often removed. This technique is often known as segmentation. It is a widely used method from computer vision[1], where only objects of interest are extracted from an image. Other challenges regarding the detection of diseases from images are different light conditions and angles of the images that were taken. Sometimes, it can be an advantage for some diseases to analyze not the colored version, but the grayscale[2] version of an image.

4.2.2 Conventional model type based on the presence of pests

Another model type deals with the detection of pests on plant leaves. In agriculture, a pest can be any harmful insect. Known pests are aphids that attack the leaves and live on plant sap, which is a fluid transported in the cells of a plant. After they occupy a plant, they start to reproduce and their numbers quickly increase. They can cause serious problems and also spread fungal diseases. Therefore, it is interesting to detect the presence of these pests, since it might be linked to a certain disease and can prevent the spreading of it.

Whiteflies are another type of insect that feeds on plant leaves. Besides damaging the plant by feeding on them, like aphids, they also do transmit plant diseases. They are also very small and sometimes difficult to detect. They are difficult to control, since they can learn to be resistant against pesticides. Therefore, it is important to detect them in their early stages, before they start to reproduce.

Similar to the previous model type:

[1] Computer vision deals with the extraction of information from images.

[2] The pixels of a grayscale image indicate the amount of light, usually in a range from 0 to 255. In contrast to RGB, a grayscale image has only one channel.

- images of leaves are used as an input, from a close range. On the leaves, pests should be visible
- as an output, this model type should detect the presence of pests. Their presence might be an indicator for an infection in the future

Ideally, images are automatically captured through some device, rather than doing it manually. The quality of the images has to be relatively high, since these pests are of a very small size and not always easy to see. In many situations, the human eye might not be enough to deal with this problem. Using an algorithm to automatically detect these pests can be an advantage.

Computer vision techniques are also used to detect pests. Most of them try to count the number of aphids or whiteflies on a leaf.

4.2.3 Conventional model type based on thermal images

Some plant diseases cause a change of the temperature in a leaf. Concretely, the infected areas may show a slightly higher temperature than the other healthy parts. This phenomenon can be observed in some diseases. Detecting these temperature changes in their early stages means also forecasting the occurrence of a possible disease. While a human expert cannot do this process manually, there are other ways that help measuring the temperatures. The most common one uses thermographic cameras (sometimes also known as infrared cameras). They create a thermal image of the objects they capture, by using infrared radiation. The created image shows the temperature distribution of the captured object. Areas with a high temperature are usually depicted in white color, a bit less warm temperatures are shown as red and yellow, and low temperatures are indicated in black and blue color. Figure 4.3 shows an image of an apple leaf, infected with the scab disease. Ideally, these cameras can be used to automatically monitor the plants for temperature changes, in order to immediately report if there is a potential risk for a disease.

Models that implement this model type have the following requirements:

- they should use thermal images as an input, from a close range
- they should return as an output, whether there are areas with high temperatures in a leaf or not. A high temperature might indicate that a disease is in its initial stages of development

This kind of model type allows predicting or detecting a disease in its earliest stages, when it is still barely visible to the naked eye. Since a thermographic image shows a clear distinction of the areas that are infected, due to the difference in color, it can be also analyzed manually. Consequently, it is also easy to apply computer vision techniques to identify these problematic areas in the leaf. The more complicated part is the actual

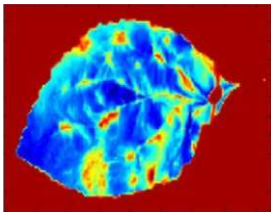


Figure 4.3: Thermographic image of apple scab in a leaf. The red spots indicate infected areas with a higher temperature[GVG+17].

acquisition of the data. External environmental factors, such as sunlight, may cause problems regarding the temperature distribution in the areas of a leaf and the cameras may deliver a wrong image.

4.2.4 Conventional model type using remote sensing

Monitoring plants for diseases is also possible from longer distances. Conventional model types that use remote sensing should fulfill the following requirements:

- as an input, images of fields are used, from a long distance
- as an output, it should be identified whether a disease is present or not. This can be done for a plant, a group of plants or an entire field, depending on the quality of the image

Acquiring the images can be done in different ways. Typically, satellites are used to capture images, on a regular basis. Also, unmanned aircraft vehicles, such as drones, can be used. Planes equipped with cameras are also often used.

Analyzing these images does not only help to identify problematic fields regarding a specific disease, but it can also help to check the severity of that disease, since the images cover a lot of information. It also gives the possibility to identify spots of diseases, in early stages.
Data that is captured using these devices has also other applications, such as monitoring road infrastructure, and more. These re-usages of the data represent an important advantage, especially from a cost perspective. A disadvantage is that the images are of a relatively large size, requiring a lot of computation power.

Images are represented in different ways. Spectral images, i.e. images the human eye can observe, are used. Hyperspectral imaging is used more often, especially for monitoring plant diseases. It allows seeing things, not visible to the human eye, such as chemical composition of the fields. The spatial resolution, which refers to the smallest object size that can be detected on the ground, can be very small, providing very detailed information. Another important aspect is the temporal resolution, which indicates how

often images are acquired. When it comes to checking for a disease development, the frequency of the acquisition of the images being high can be a benefit.

The data acquisition is the most complicated part. Although such data exists, it is not always accessible. Sometimes, it is not possible to access the data, simply because it is intended for specific purposes only or because of other reasons. Usually, commercial satellites will offer the best images. Once the data is available, using it to detect diseases should not be very complicated in most of the situations. It can also be analyzed manually.

Not many implementations exist for this kind of model type. At the time of this writing, no AI models are identified, which should motivate for more research in this area.

4.2.5 Conventional model type based on environmental data

Many plant diseases are influenced by wetness and low/high temperatures. Because of this, many of them are predicted using weather forecasts. As a consequence, this has led to the creation of plant disease forecasting systems. They do not only forecast a disease, but they also help the growers to make economic decisions, in order to treat the infected plants in the right way. The concept behind these systems is the *plant disease triangle*[Isl18]. It defines that a disease will occur only if the following three factors interact in some way:

- the susceptible host, which usually represents the plant that may have some vulnerability regarding some disease
- the pathogen, which needs a vulnerable plant to attack and favorable environmental conditions
- the environmental conditions

Although the pathogen may exist, if the environmental conditions are not favorable, the plant may still not get infected. The same holds also for the other case, i.e. environmental conditions favor the development of a disease, but the pathogen is missing.

Forecasting systems do already exist in practice. For instance, the New York State Integrated Pest Management implemented a system[3], that growers can use to make predictions for their plants.

The part of these systems that is responsible for forecasting can be seen as a model. Models that make predictions for a disease are the most common known and have existed for a long time. The algorithms of these models have been obtained from research and many trials over time. Models should:

[3] https://newa.cornell.edu

- use as an input environmental data, mostly, this data contains series of weather events. Typical data would be temperature, humidity, rainfall data, leaf wetness, and more
- predict whether there might be a risk for a disease or not

For most of the plants, such conventional models do exist. Since the goal is to compare these models with AI models, it is important to be able to run the models based on the same data. Unfortunately, the majority of the models in the literature do not provide the data they use for verification.

4.2.6 Other conventional model types

A few other model types are shortly described. Because of the lack of the necessary input data that is needed to implement models for these model types, they are not used for further comparison. However, they can serve as a reference, in order to extend this work in the future, with more model types and models.

Model types that use morphological data over time. This model type analyzes the plant morphology, in order to identify a disease. Plant morphology studies the physical form of plants. Some diseases cause a change to the form, for example they might cause an enlargement of parts of the plant. Data about the plant should be, ideally, three-dimensional and acquired on a regular basis. This way, the data can be analyzed, in order to understand the morphological development over time.

Model types that are based on fluorescence imaging. Similar to model types that use thermographic images, also these models use images as their input data. The images show the fluorescence across a leaf. Concretely, they can show how much light the chlorophyll absorbs. This can be related to the presence of some diseases.

Model types that are based on hyperspectral imaging. Hyperspectral images show the reflectance of images, from a wider spectral range than the human eye can see. It creates spatial images, which can help with the disease identification. Even though hyperspectral imaging was already mentioned as part of conventional models that were based on remote sensing, also close-range images can be used for analysis.

Model types that use spectroscopy. Models that use spectroscopy analyze data about the wavelength of electromagnetic radiation. Spectroscopy can measure the spectrum of light that is reflected or absorbed by a plant. It can be used to analyze this reflectance or absorption for different wavelengths (spectra). The reflectance and absorption can be important indicators for a disease. For instance, it is known that healthy plants absorb more infrared light than infected ones.

4.3 Conventional models

This section describes conventional models that are used later for comparison purposes.

4.3.1 Conventional models for diseases caused by cold weather

Tomato late blight image-processing model
Pamplona et al.[PCB20] developed an algorithm, which was able to detect black Sigatoka and yellow Sigatoka diseases in banana leaves, and also Phytophthora infestans in tomato leaves, which causes late blight. In order to segment infected areas on the plants, with images from the PlantVillage data set[HS15] as an input, the Gaussian filter[4] was applied. After the background noise was removed, the leaf images were transformed from RGB to YIQ color model. The I channel was used for thresholding[5], in order to create a binary image. The binary image was used as a mask to remove the healthy parts from the leaf. The resulting image was transformed to HSV color model and the Gaussian filter was applied to the H channel. Using thresholding, a mask was created and used to produce the final image, with the healthy areas removed. In order to verify the results of the algorithm, diseased areas from 100 images were manually selected and used as a reference for comparison.

Wheat stripe rust weather model
Although many conventional models that are based on weather data do not publish the data, Jarroudi et al.[JLK+20] published the data they used for testing their approach. Weather data was used to predict wheat stripe rust infection on different fields in Morocco. The data consisted of weekly rainfall, temperature and humidity values. In order to verify and compare the results the model produced, the incidence and severity of the disease were manually assessed over time.

4.3.2 Conventional models for diseases caused by moderate to hot weather

Color features plant disease identification model
El Sghair et al.[ESJT17] researched into possible ways to detect infected areas in plants, using color features. RGB and other color models were tested. The median filter[6] was applied for image smoothing. Out of the researched color models, the HSI color model resulted as the most successful one. In order to produce a binary image, the H channel was taken as an input for a thresholding method, called Kapur's thresholding. Only a few images were used for testing and no plant or disease was specified. Later, for comparison, this model is tested for corn northern leaf blight detection.

Image-processing plant disease severity model
Camargo et al.[CS09] used image-processing to detect the severity of a disease. The area of a leaf that was infected, was calculated in percentage. The Gaussian filter and some other morphology operations, such as erosion[7] and dilation[8], were applied to the images.

[4]The Gaussian filter, also known as Gaussian blur, is used for blurring an image.
[5]Thresholding is used for segmentation, where pixels with a value greater or less than the threshold value are ignored. The result is a binary image.
[6]The median filter removes noise from an image.
[7]Erosion removes pixels from the boundaries of an object.
[8]Dilation adds pixels to the boundaries of an object.

The RGB images were converted to other color models such as HSV, and also to some others, which worked well with varying lighting conditions. The best color model was selected as a basis for segmentation. The segmented image was a binary image, where white pixels represented the diseased area and black pixels the healthy parts. An optimal threshold was identified, which could differentiate the background from the target object, i.e. the diseased spots. The segmented images were compared with reference images. The tests were run on 20 images, each containing different plants such as banana, cotton and corn, with different diseases. One of them was northern leaf blight of corn.

Infrared image-processing plant disease identification model
Yang et al.[YYW+19] implemented an approach to detect diseases in tea leaves, using thermographic images as an input. Segmentation was applied, and some color transformations were performed. The images were converted from RGB to HSV. Thresholding was applied to the H component and the image was converted to a grayscale one. After removing the noise (using median filtering), a binary image was created as a result. This image was used to count the diseased areas. It was verified that the red spots matched with the infected areas of the leaves. This was achieved by checking manually, using a microscope. Also, the size of the affected area and the severity of the disease could be calculated. The greater the size, the higher the severity of the disease.
As the authors also claimed, this technique should also work for other types of plant diseases, such as Fusarium wilt of cucumber. The problem is broken down into detecting areas with different colors in an image, under the assumption, that the images were captured under ideal conditions, without any influence from the environment.

Rice sheath blight multispectral imaging model
Zhang et al.[ZZZ+18] researched into rice sheath blight identification methods, using long-range images. The images, which contained rice cultivars, were acquired using a quadcopter, which was equipped with both, digital and hyperspectral cameras. Using the captured images, the severity of the disease was analyzed. Using the normal digital images, color transformations were applied, in order to qualitatively detect the diseased areas. However, this method didn't perform well, regarding the assessment of the severity of the disease. To quantify the severity, the vegetation indices from the hyperspectral images were calculated. The vegetation indices indicate whether the crop fields contain green vegetation or not. Results were compared with ground-truth data. The image data was made public by the authors.

4.3.3 Conventional models for diseases caused by pests

Image-processing aphids detection model
Maharlooei et al.[MSB+17] developed an approach to count aphids on soybean leaves. Segmentation was used, in order to extract the leaf from the background and create a binary image. A similar step was repeated to remove the leaf, in order to extract the aphids from it. Color transformations were applied as well. As a last step, the number of aphids was counted. Each white dot in the binary image represented an aphid. However,

Figure 4.4: Rice field with sheath blight[ZZZ+18]. Infected tissue has yellow to brown color.

no data with aphids was made available to implement an AI model. Therefore, this conventional model is skipped from further comparison.

Image-processing whiteflies detection model
Computer vision can be used to identify whiteflies on leaves. Bodhe et al.[BM13] proposed an approach for segmenting whiteflies. HSV, YIQ and other color model transformations were applied. For each color channel, the entropy was calculated. The channel with the highest entropy was used for thresholding.

Figure 4.5: Whiteflies occupying a leaf[BM13].

Image-processing whiteflies counting model
Barbedo et al.[Bar13] proposed a method to count whiteflies on soybean leaves. The approach was tested with images in RGB color format. Different color transformations and morphological operations were applied. Attempts were made to detect whiteflies in different stages of their life cycle.

Image-processing pest detection model
Miranda et al.[MGI14] proposed a general way to detect insects, not necessarily on leaves. RGB images were transformed into grayscale images as a first transformation step. Pixels were compared with the background of the image. If these pixels had a different color from the background, then they were considered as pixels of some part of the insect.

Model	Model type	Authors	Disease	Cause	Methods
Color features plant disease identification model	Close-range images	El Sghair et al.[ESJT17]	Corn northern leaf blight	Hot weather	Segmentation, color transformations, Kapur's thresholding
Tomato late blight image-processing model	Close-range images	Pamplona et al.[PCB20]	Tomato late blight	Cold weather	Color transformations, thresholding, filtering
Image-processing plant disease severity model	Close-range images	Camargo et al.[CS09]	Corn northern leaf blight	Hot weather	Filtering, morphology operations, color transformations, thresholding
Image-processing whiteflies detection model	Close-range pest images	Bodhe et al.[BM13]	Chrolosis	Pest	Color transformations, entropy based thresholding
Image-processing whiteflies counting model	Close-range pest images	Barbedo et al.[Bar13]	Chrolosis	Pest	Color transformations, thresholding
Image-processing pest detection model	Close-range pest images	Miranda et al.[MGI14]	Chlorosis	Pest	Color transformations, thresholding
Infrared image-processing plant disease identification model	Thermal images	Yang et al.[YYW+19]	Cucumber fusarium wilt	Hot weather	Color transformations, thresholding
Rice sheath blight multispectral imaging model	Remote sensing	Zhang et al.[ZZZ+18]	Rice sheath blight	Hot weather	Color transformations, vegetation index calculations
Wheat stripe rust weather model	Environmental data	Jarroudi et al.[JLK+20]	Wheat stripe rust	Cold weather	Formula based on temperature, humidity and precipitation

Table 4.1: Summary of the conventional models. The disease column indicates the disease of interest during comparison with the AI models.

CHAPTER 5

AI Models

This chapter defines AI models for plant disease identification. Models are implemented for each model type and disease category. The discussed algorithms in Chapter 2 serve as a basis for the definition of these models. Since the data is labeled and the classes are discrete, supervised learning classification algorithms are taken into consideration. The process of defining an AI model always goes through the following stages:

1. **Data acquisition.** Often, the type of data that is used to train a model is the first step. There might be many sources for the data. In this work, the data is mainly coming from normal and thermographic cameras, from close and long ranges. Weather data from weather stations is another kind of data source. The behavior and success of a model will depend on the quality of the data it is trained with, i.e. the definition of such a model is a data-driven process

2. **Data pre-processing.** Since it is often necessary to clean/pre-process the data, visualization tools need to be used, in order to give a visual description of the data. This can be helpful, when it comes to detecting anomalies, missing data or data imbalance. There may be also other aspects, such as the size of the data, as the performance of many algorithms depends on it, and many others, depending on the problem that needs to be solved

3. **Data augmentation.** Sometimes, it might be necessary to generate more data. Section 5.1 describes the techniques that are used to augment existing data

4. **Training and evaluation data.** In general, the data is split into training and evaluation data. Chapter 6 explains this in a more detailed way. Usually, 80% of the data is used for training and the rest for evaluation

5. **Choosing the model.** Models[1] are chosen depending on the problem and different criteria. Depending on whether the data is labeled or not, supervised or unsupervised machine learning models may be used. However, in general, a systematic way to define a model for a problem does not exist. Choosing the right model is often a combination of experience, some intuition and experimentation

6. **Training.** During the training phase, the model's parameters are adjusted, in order to fit the model to the data as close as possible. This process may take some time, depending on the number of features and type of the model

7. **Evaluation.** After the model is built, it can be evaluated. Different metrics are used for evaluation. See Chapter 6 for the evaluation strategy, and also for the evaluation results

8. **Parameter adjustments.** It might be necessary to go back to step 5 and adjust some parameters, in order to obtain better results. This is done until the model performs as expected

The models are built by using the functions defined in Chapter 2. All the models can be found in the supplementary materials. Experiments in this chapter are conducted using the Python[2] programming language. It is the most popular language that machine learning developers use, and it comes with many popular libraries that offer the possibility to quickly apply different algorithms. It offers cross-platform support and a good performance, since it does not require a virtual machine to run, like some other languages do. The runtime of the models is a key metric for evaluation. Table 5.2 gives a summary of the defined AI models, including their (adjustable) parameters.

5.1 Data augmentation techniques

Data plays a crucial role in a machine learning model. Unfortunately, it is not always available in large amounts or difficult to collect, making the machine learning models perform not well and also prone to overfitting and underfitting. Because of this, the data set has to be constructed in some way. There are a couple of ways to achieve this. In machine learning, *data augmentation* refers to a data analysis technique, which increases a small data set, by making some modifications to the original data.

5.1.1 Augmentation using image operations

In image classification problems, modifications may include changing the color, rotating, flipping of the original images, and more. These operations are described next.

Rotation. This operation applies clockwise rotation to the image. The rotation angle is a randomly chosen value between 0 and 360 degrees. The images are not resized, i.e. the

[1] The term *model* often refers to machine learning algorithms.
[2] `https://www.python.org/`

original size is kept. This means that, depending on certain angles, such as 45 degrees, the image might appear smaller and that there will be some space outside the boundaries of the image. This space is filled using the background color of the original image, or at least with a color close to it, in order to keep the consistency.

Horizontal flip. As the name suggests, this operation mirrors the image in the horizontal direction.

Homography. Also known as projective transformation[3], the goal of this operation is to show how the viewed object changes when the angle of the observer changes. This should mimic the capturing of images from different angles. Describing how this method works is out of the scope of this work. However, it is important to know that the transformation is based on the homogeneous transformation matrix, which in this case is a $3x3$ matrix. Depending on the values of the matrix, the transformations may be rotation, translation, scaling, skewing or a combination of these. For the concrete case, the matrix looks as follows:

$$\begin{pmatrix} 1 & -0.5 & 30 \\ random([0.1, 0.7]) & random([0.4, 1]) & 0 \\ 0.0015 & 0.0015 & 1 \end{pmatrix}$$

As a result, the generated images will also look warped and distorted.

Noise. This operation adds noise to the image. Speckle[4] is used as the interference technique, which reduces the quality of the image. The motivation behind this operation is the fact that some cameras may produce low quality images.

5.1.2 Augmentation using GAN

Applying image operations works well only with image data sets. Due to the nature of these transformations, the relation between the pixels is still preserved. Therefore, this will not lead to a change in the classification label. In numerical data sets, however, a label shift may happen, i.e. causing generating samples that belong to the wrong class. Therefore, other approaches are needed.

AI is not only used for classification purposes, it can also be used to generate data. A machine learning model, based on artificial neural networks, can learn from some data samples and generate similar data. Goodfellow et al.[GPAM+14] designed a framework, called *Generative Adversarial Networks* (GAN), which follows this principle. The reader should refer to this paper, in order to understand the way GANs work. Nevertheless, the intuition behind it is shortly described next.

GANs have many applications, one of which is face generation. They are mainly focused on generating image data, but are not limited to it. An image is nothing but a n-dimensional vector, or, if flattened, it can be expressed as a vector. Numerical data, such

[3] https://en.wikipedia.org/wiki/Homography
[4] https://en.wikipedia.org/wiki/Speckle_(interference)

as weather data, can be represented similarly, making GAN suitable for generating such data as well.

A GAN consists of two neural networks, which compete against each other, one is called the *generator* and the other one the *discriminator*. The generator tries to generate fake data, which is verified by the discriminator. The generator is initialized with random data, which resembles the real data more and more, every time the discriminator is able to classify this data as fake. This process continues until the discriminator is not able to detect anymore that this data is fake.

In more technical terms, the goal of the generator is to increase the error rate of the discriminator. The generator and the discriminator are trained simultaneously. The discriminator is continuously trained with real data, while the generator is always fed with fake data and tries to learn how to map this fake data to real data. This mapped data is sent to the discriminator for verification. This process of training and verification continues until the generator learns how to map randomly generated fake data to real data. A GAN does not need labels to generate the data, i.e. it is unsupervised in this regard.

5.2 Notations

In order to build the models, some functions are needed, i.e. some notation needs to be defined:

- $flatten(I)$ flattens an n-dimensional vector I. The result is a 1-dimensional vector
- given a vector N, the function $first(N, n)$ returns the first n elements
- given a vector T and a vector Y, both with same dimensions and number of elements, the function $split(T, Y, fraction)$, where $fraction$ needs to be between 0 and 1, randomly splits T and Y into four vectors and returns (T_t, Y_t, T_e, Y_e), where
  - T_t contains a $1 - fraction$ fraction of T, and Y_t the respective elements from Y
  - T_e contains the rest, i.e. the remaining $fraction$ fraction of T. Y_e are the respective elements from Y
- for a neural network, the cross-entropy loss function is defined as $cel(h, X, Y) = -\sum_{i=1}^{n} y_i \cdot ln(\frac{e^{a_i(X)}}{\sum_{j=1}^{n} e^{a_j(X)}})$, where:
  - X is the vector of the training example
  - Y is the label, expressed in one-hot encoding[5], with n elements
  - n is the number of neurons of the last layer of the network

[5] https://en.wikipedia.org/wiki/One-hot

  - a_i is the activation function of the neuron i from the last layer of the network, i.e. the function that produces the output value
  - h is the hypothesis function, which is a vector $(a_1, \ldots, a_n)$

- the binary cross-entropy loss function is defined as $bce(h, X, Y) = \frac{-\sum_{i=1}^{n}(y_i \cdot ln(h(X_i)) + (1-y_i) \cdot ln(1-h(X_i)))}{n}$, where:
  - X is the vector of the training examples, with n elements
  - Y is the vector of the respective labels, with n elements
  - h is the hypothesis function

- $backprop(loss)$ computes derivatives (or gradients) for each parameter(weight) of the $loss$ function
- $sgd(h, gradients, lr)$ is the stochastic gradient descent algorithm, which minimizes the loss function by adjusting the weights, with a given learning rate lr. It returns a new hypothesis function h with the updated weights
- $adam(h, gradients, lr)$ is the *Adam algorithm*[KB14], with a given learning rate lr. It returns a new hypothesis function h with the updated weights
- $relu$ is the rectifier activation function
- sig is the sigmoid activation function
- $lrelu$ is the leaky rectifier activation function, given as

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise.} \end{cases}$$

- the activation function $identity(e)$ returns e
- $zeros(n)$ returns a vector V with n elements, where each element is 0
- $ones(n)$ returns a vector V with n elements, where each element is 1

5.3 AI models for diseases that are caused by cold weather

5.3.1 Tomato late blight AI models

These models are able to detect tomato late blight disease. They are built with comparison in mind: the tomato late blight image processing model analyzes tomato late blight disease, which is why this disease is selected. The models can be built as follows:

1. **Data.** Images, which are in RGB format and captured by normal cameras, come from the tomato late blight data set, available in the supplementary materials. The data consists of two classes:

   - the class of images of healthy tomato leaves
   - the class of images of tomato leaves that are infected with late blight disease

   Each image is a 3-dimensional vector with the shape $(3, 256, 256)$. Pixel values are in the range $[0, 255]$. Some images are segmented, some healthy leaf images are incorrectly labeled as diseased

2. **Data pre-processing and augmentation.** Each class contains at least 600 images, making an augmentation unnecessary. As there is enough data for each class, the problem of unbalanced data does not occur. For the convolutional neural network model (see below), the pixel values are normalized with a mean 0.5 and standard deviation 0.5, for each of the three color channels

3. **Constructing the tomato late blight logistic regression model.** Since the images are labeled, the application of a supervised machine learning algorithm, such as logistic regression, is taken into consideration.

   Logistic regression expects a (1-dimensional) feature vector, each image needs to be flattened:

   $$X = flatten(I)$$

   For each image, the class is defined:

   $$y = c(I))$$

   where $c(I)$ is a function that returns 0 if the image I shows a healthy tomato leaf, and 1 otherwise. Consider T the vector of all images and Y the vector of their corresponding classes.

   80% of the data is used for building the model, the rest is used for evaluation:

   $$(T_t, Y_t, T_e, Y_e) = split(first(T, 600), first(Y, 600), 0.2)$$

   Finally, the model is built:

   $$h = lr(T_t, Y_t, o)$$

   where o can be respectively one of `"lbfgs"`, `"sag"` or `"saga"`. Chapter 2 describes the logistic regression function. The hypothesis function $h(I)$ predicts the probability that the flattened image I shows an infected leaf, with a threshold of 0.5. In the evaluation phase, this probabilistic threshold can be changed

4. **Constructing the tomato late blight support-vector machine model.** Support-vector machines also expect a feature vector, each image needs to be flattened:

$$X = flatten(I)$$

Each image is classified the same way as for the logistic regression model:

$$y = c(I))$$

T is the vector of all flattened images and Y their labels. The model is trained with 80% of the data:

$$(T_t, Y_t, T_e, Y_e) = split(first(T, 600), first(Y, 600), 0.2)$$

The model is trained using the support-vector machine function, which is defined in Chapter 2:

$$h = svm(T_t, Y_t, k)$$

where k can be respectively one of the kernels `"linear"`, `"poly"`, or `"rbf"`.

The hypothesis function $h(I)$ determines the probability that a given flattened image I shows an infected tomato leaf, with a threshold of 0.5

5. **Constructing the tomato late blight convolutional neural network model.** The normalized images are classified:

$$y = c(I)$$

where $c(I)$ is a function that returns $(1, 0)$ if the image I shows a healthy tomato leaf, and $(0, 1)$ otherwise. Consider T the vector of all images and Y their corresponding classes.

80% of the data is used for building the model, the rest is used for evaluation:

$$(T_t, Y_t, T_e, Y_e) = split(first(T, 600), first(Y, 600), 0.2)$$

The architecture of the convolutional neural network is shown in Figure 5.1. The first hidden layer applies the convolution operation to the RGB image, with a kernel of size 10 and stride of size 4, creating 64 filters, which have to be learned. This layer uses the rectifier as the activation function. The learned filters are used to detect different features of the images. The filters are resized and made smaller using a max pooling layer, with kernel size of 3 and stride of 2 pixels, and the rectifier activation function is used once more. The output serves as an input for the next fully-connected layer, consisting of 128 neurons. This layer also uses the

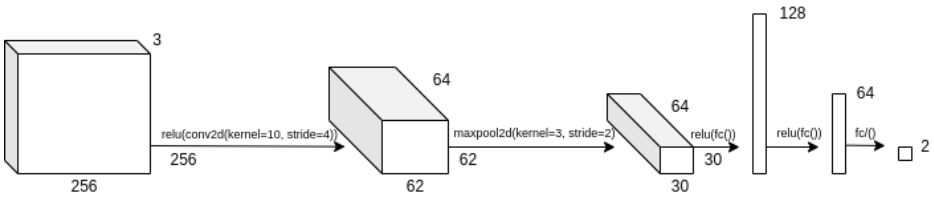


Figure 5.1: Tomato late blight convolutional neural network model.

rectifier activation function and sends the data to the next fully-connected layer, which has 64 neurons. These send their results to the final layer, which has 2 neurons, each firing for each of the classes.

In a more formal notation, the network is described as follows:

$l_1 = conv2d(I = I, in = 3, out = 64, kernel = (10, 10), stride = 4, a = relu)$
$l_2 = maxpool2d(I = l_1, kernel = (3, 3), stride = 2)$
$l_3 = fc(I = flatten(l_2), in = 57600, out = 128, a = relu)$
$l_4 = fc(I = l_3, in = 128, out = 64, a = relu)$
$l_5 = fc(I = l_4, in = 64, out = 2, a = identity)$

The layer functions are described in Chapter 2. Let $h(I)$ be the hypothesis function that applies the steps above, and Y the labeled output for I. For every normalized training image I and label Y, the cross-entropy loss function is used. As a minimization algorithm, the stochastic gradient descent is used, with a learning rate 0.001. These training steps are formally described as follows:

$lossFunc = cel(h, I, Y)$
$gradients = backprop(lossFunc)$
$h = sgd(h, gradients, 0.001)$

The training process is repeated 5 times[6]. The function $h(I)$ returns a vector (p_h, p_d), where p_h is the probability that the normalized image I is healthy, and p_d the probability that it is infected with tomato late blight

5.3.2 Wheat stripe rust logistic regression model

Most of the traditional plant disease models rely on environmental data. The wheat stripe rust weather model used weather data to predict infections in Morocco. The assessments about the incidence of this disease in different sites across Morocco and on different days in 2018 and 2019 were made available by the authors. However, the weather data, which was computed using a regional atmosphere model and then used to make the predictions, was only published graphically. This data included weekly frequencies of different levels of rainfall, temperature and relative humidity. Regarding the assessments, same as for other models, the wheat crops are assumed to be equally susceptible to wheat stripe rust. Also, it is assumed that there exists some presence of a pathogen, which favors

[6]The *epoch* indicates how many times the training process is repeated.

the development of the disease. This allows focusing only on the environment part of the plant disease triangle. The same assumptions were also made by the authors of the conventional model.

A wheat stripe rust identification logistic regression model is defined as follows:

1. **Data.** For the data set, the necessary weather data needs to be collected, for each site. The frequencies of specific temperatures, relative humidity and rainfall are manually extracted from the plots that were published by the authors. Some sites are very close to each other, which is why the weather data for them is nearly the same or identical.

   One sample is created for the same site, date and species. Since there can be more than one assessment, the average incidence of wheat stripe rust is taken. Weather data from the week of the assessment is used for each sample.

Feature	**Description**
Site	The site where the assessment is made
Date	The date when the assessment is made
Species	The wheat species, can be bread or durum
R_1	Weekly frequency of no rain
R_2	Weekly frequency of rain >0mm and <=1mm
R_3	Weekly frequency of rain >1mm and <=5mm
R_4	Weekly frequency of rain >5mm
H_1	Weekly frequency of humidity <=60%
H_2	Weekly frequency of humidity >60% and <=75%
H_3	Weekly frequency of humidity >75% and <=85%
H_4	Weekly frequency of humidity >85% and <=90%
H_5	Weekly frequency of humidity >90%
T_1	Weekly frequency of temperature <0
T_2	Weekly frequency of temperature >0 and <=4
T_3	Weekly frequency of temperature >4 and <=8
T_4	Weekly frequency of temperature >8 and <=12
T_5	Weekly frequency of temperature >12 and <=16
T_6	Weekly frequency of temperature >16 and <=20
T_7	Weekly frequency of temperature >20
Average incidence (I)	The average incidence for a specific site, date and species

Table 5.1: Description of the features of the constructed data set.

The constructed wheat weather data set can be found in the supplementary materials and the features are described in Table 5.1. The ranges of the frequencies have been proposed by the authors

2. **Data pre-processing and augmentation.** Since machine learning algorithms require numerical data to work with, the non-numerical features are mapped as follows:

   - the 9 sites are mapped to numerical values from 0 to 8, in alphabetical order
   - bread wheat species is mapped to 0 and durum wheat species is mapped to 1

   In the assessments, the average incidence represents the percentage of infected plants on a field. For each assessment, these values are made more discrete and mapped to the following classes:

   - an incidence less than 20 is mapped to risk level 0 (low)
   - otherwise, the incidence is mapped to risk level 1 (high)

   As it can be seen in the constructed data set, the durum species has, in general, a lower incidence than the bread species. Bread wheat infections are favored with a humidity greater than 90% for up to 15% of the time and low rainfall, during the week. Temperatures should be between 8 and 16 for 10% to 25% of the time. In Bouderbala, the high humidity should be present for a greater amount of time. Because of this, the disease depends not only on the species, but also on the specific site.

   The constructed data set includes, however, only 46 samples, which is not enough to build a proper model, as it would be prone to overfitting. In order to overcome this problem, data needs to be enhanced. Applying image transformations is not possible, which is why a GAN is defined to generate data.

   Some data pre-processing is required, in order to facilitate the work of the neural networks. Therefore, each feature is normalized. That is, each feature X is transformed by using the following formula:

   $$Z = \frac{X - min(X)}{max(X) - min(X)}$$

   The values will always be in the interval $[0, 1]$. Sometimes, $min(X)$ and $max(X)$ may match. This can happen, when all the classified samples have the same assessment date. In this case, in order to prevent a division by zero, Z is set to 0.5. The assessment date is ignored from the features, as it has no relevance.

   The GAN learns from each site to generate data. It does this separately for each class, i.e. the GAN is run, in order to learn to generate new data for each site and class.

   The generator is defined as follows, where the number of features for X is 18:

   $$g(N) = fc(I = N, in = 18, out = 18, a = sig)$$

   The neural network of the discriminator $d(X)$ has the following configuration:

$$l_1 = fc(I = X, in = 18, out = 128, a = lrelu)$$
$$l_2 = fc(I = l_1, in = 128, out = 1, a = sig)$$

Data from the same site and class is grouped together into batches and used to run the GAN, i.e. the batch has a shape $(k, 18)$, where k is the number of training examples for the batch. With every batch B_{real} the process is as follows:

A batch B_{noise}, with the same shape as B_{real} is generated, where each feature vector contains features with random values between 0 and 8.

For each feature vector N from B_{noise}, the generator generates fake data G_{fake}:

$$G_{\text{fake}} = g(N)$$

$h(G_{\text{fake}})$ returns the probability that the provided fake data is real. Let D_{fake} be the batched result from the discriminator, i.e. a vector with k elements.

Next, the generator is trained:

$g_{\text{loss}} = bce(g, D_{\text{fake}}, ones(k))$
$gradients = backprop(g_{\text{loss}})$
$g = adam(g, gradients, 0.001)$

The loss of the generator uses a vector of ones as the predictions. The loss is high, if the discriminator detects that the data is fake.

Next, the discriminator is trained. For each feature vector X from B_{real}, the discriminator is called with X. Let D_{real} be the batched result. The loss function is calculated as a combination between the discriminator's loss on real and fake data:

$d_{\text{realLoss}} = bce(d, D_{\text{real}}, ones(k))$
$d_{\text{fakeLoss}} = bce(d, D_{\text{fake}}, zeros(k))$
$d_{\text{loss}}(f, X, Y) = (d_{\text{realLoss}}(f, X, Y) + d_{\text{fakeLoss}}(f, X, Y)) \cdot 0.5$

Finally, the gradients are calculated, and the weights are updated using the Adam optimizer:

$gradients = backprop(d_{\text{loss}})$
$d = adam(d, gradients, 0.001)$

This process is repeated 150 times for B_{real}. The built generator generates 10 samples using $g(N)$, where N is a feature vector that contains features with values between 0 and 8.

Overall, 160 samples are generated. Since the features were initially normalized, each feature X is transformed back.

A big advantage of a GAN is that once the generator has been created, it is then able to generate data in a very fast way. One major shortcoming is that there are no metrics to evaluate how good a GAN works. For instance, in case of generating faces, it is easy to manually verify the generated faces. Interpreting generated numerical data, however, is much more difficult. Another common problem with a

GAN is the lack of stability. Sometimes, the generator will get stuck, i.e. it will produce the same data all the time, after a certain number of epochs. This requires some manual verification, in order to know when to stop the generator. Running the GAN again might produce better results. The GAN might oversample some samples. This is the case for situations, where only one assessment is available for the same site, date and species

3. **Constructing the model.** Let $c(X)$ be the function that classifies each training example from the augmented data set. It returns 0 if the weather data is not expected to be favorable for stripe rust, and 1 otherwise. Let T be the vector of all training examples and Y the vector of the classes. 80% of the augmented data is used for training the model:

$$(T_t, Y_t, T_e, Y_e) = split(T, Y, 0.2)$$

The model is trained using:

$$h = lr(T_t, Y_t, o)$$

where o is one of the optimization algorithms from Chapter 2.

Given some data sample X with weather observations, which has the same structure as previously described, the hypothesis function $h(X)$ predicts if the weather data in X is favorable for development of stripe rust, with a threshold of 0.5.

The quality of this model is very dependent on the generated data set, i.e. it is a more theoretical model than the others

5.4 AI models for diseases that are caused by hot weather

5.4.1 Corn northern leaf blight AI models

These models are able to identify corn northern leaf blight disease. They are built with comparison in mind: the color features plant disease identification model does not specify any disease concretely, therefore, it is selected for comparison with this model. The image-processing plant disease severity model deals with identification of corn blight diseases. They have similar visual symptoms to northern leaf blight. The models are trained as follows:

1. **Data.** The models use RGB images from the corn northern leaf data set, available in the supplementary materials. It consists of two classes:
   - the class of images of healthy corn leaves
   - the class of images of corn leaves that are infected with northern leaf blight disease

Each image is a 3-dimensional vector with the shape $(3, 256, 256)$, with pixel values in the range $[0, 255]$. Some images are segmented, some healthy leave images are incorrectly labeled as diseased

2. **Data pre-processing and augmentation.** Each class contains at least 600 images, which is enough data for training and testing. For the convolutional neural network model, the pixels are normalized with a mean 0.5 and standard deviation 0.5, for each of the three color channels

3. **Constructing the corn northern leaf blight logistic regression model.** Logistic regression requires every image to be flattened:

$$X = flatten(I)$$

Each image is classified:

$$y = c(I))$$

where $c(I)$ is a function that returns 0 if the image I shows a healthy corn leaf, and 1 otherwise. Consider T the vector of all images and Y the vector of their corresponding classes. 80% of the data is used for building the model:

$$(T_t, Y_t, T_e, Y_e) = split(first(T, 600), first(Y, 600), 0.2)$$

The model is trained using the logistic regression function from Chapter 2:

$$h = lr(T_t, Y_t, o)$$

where o can be one of the three optimization algorithms. The hypothesis function $h(I)$ determines the probability that a flattened image I shows an infected corn leaf, with a threshold of 0.5. This threshold is configurable for evaluation purposes

4. **Constructing the corn northern leaf blight support-vector machine model.** Support-vector machines also expect a feature vector, each image is flattened and classified the same way as for the logistic regression model:

$$\begin{aligned} X &= flatten(I) \\ y &= c(I)) \end{aligned}$$

Let T be the vector of the flattened images and Y the classes. The model is trained with 80% of the data, the rest is used for evaluation:

$$(T_t, Y_t, T_e, Y_e) = split(first(T, 600), first(Y, 600), 0.2)$$

The model is trained using:

$$h = svm(T_t, Y_t, k)$$

where k can be respectively one of the kernels from Chapter 2.

The hypothesis function $h(I)$ determines the probability that a given flattened image I shows an infected corn leaf, with a threshold of 0.5

5. **Constructing the corn northern leaf blight convolutional neural network model.** The normalized images are classified:

$$y = c(I)$$

where $c(I)$ is a function that returns $(1, 0)$ if the image I shows a healthy corn leaf, and $(0, 1)$ otherwise. Consider T the vector of all images and Y their corresponding classes. 80% of the data is used for building the model:

$$(T_t, Y_t, T_e, Y_e) = split(first(T, 600), first(Y, 600), 0.2)$$

The architecture of the convolutional neural network is shown in Figure 5.2.

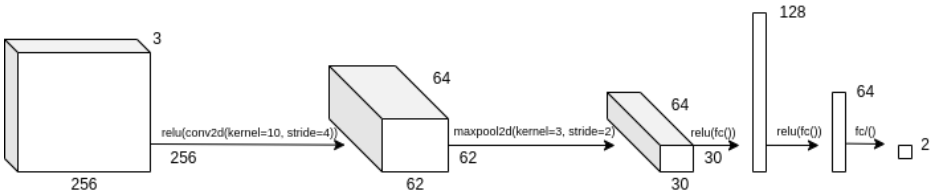


Figure 5.2: Corn northern leaf blight convolutional neural network model.

The network is described as follows:

$l_1 = conv2d(I = I, in = 3, out = 64, kernel = (10, 10), stride = 4, a = relu)$
$l_2 = maxpool2d(I = l_1, kernel = (3, 3), stride = 2)$
$l_3 = fc(I = flatten(l_2), in = 57600, out = 128, a = relu)$
$l_4 = fc(I = l_3, in = 128, out = 64, a = relu)$
$l_5 = fc(I = l_4, in = 64, out = 2, a = identity)$

The hypothesis function $h(I)$ applies the steps above, and Y is the expected class for I. The training process is carried out using cross-entropy loss, minimized through stochastic gradient descent:

$lossFunc = cel(h, I, Y)$
$gradients = backprop(lossFunc)$
$h = sgd(h, gradients, 0.001)$

The training is repeated 5 times. $h(I)$ returns a vector (p_h, p_d), where p_h is the probability that the normalized image I shows a healthy corn leaf, and p_d is the probability that is infected

5.4.2 Cucumber Fusarium wilt AI models

The models use images that show the distribution of temperatures of the captured object. Unfortunately, at the time of this writing, no large data sets showing thermal images of healthy and diseased leaves do exist. Also, collecting images manually, is not possible.

However, Wang at al.[WLD+12] created thermographic visualizations of cucumber leaves, infected with the soil-borne pathogen Fusarium oxysporum f. sp cucumerinum. This fungal pathogen is responsible for Fusarium vascular wilt on cucumber leaves. Infected leaves tend to have more water loss than healthy ones, which causes the temperature of the leaves to be higher than usual. Therefore, it is possible to analyze the leaves by taking the thermal images of them.

The authors captured thermal images of inoculated and non-inoculated leaves on 6 days (days 4, 5, 7, 9, 10 and 11 after the inoculation). As their images show, the temperature of an infected leaf is already high in the initial stages of the disease, even though the leaf still looks healthy to the naked eye. This could be important, when it comes to dealing with diseases and should, at the same time, motivate to do more research in this direction.

The AI models detect cucumber Fusarium wilt disease. The infrared image-processing plant disease identification model can detect any disease using infrared images and is used for comparison. The AI models are trained as follows:

1. **Data.** Since the authors didn't publish the visualization in the format of a data set, six healthy and six diseased RGB images, with the shape $(1, 124, 124)$ and pixel values in the range $[0, 255]$, are extracted manually, by using a tool. These images can be found in the supplementary materials

2. **Data pre-processing and augmentation.** Since there are only six images per class, data augmentation techniques have to be used. Image data augmentation techniques are applied in the following order:

   - homography is applied with a probability of 60%
   - clockwise rotation is applied with a random angle between 0 and 360 degrees and with a probability of 90%
   - images are horizontally flipped with a probability of 20%

   This process is repeated for each image 100 times, i.e. there are overall 106 samples per class. When performing rotation or homography, the background outside the boundaries of the images has to be filled. For this, the background color of the original image is used, and erosion is used to eliminate possible noise. For the convolutional neural network model (see below), the pixels are normalized with a mean 0.5 and standard deviation 0.5, for each of the three color channels

3. **Constructing the cucumber Fusarium wilt logistic regression model.** Based on the augmented data set, a logistic regression model can be trained. Since it expects a feature vector, each image needs to be flattened. Also, for each image, the class is defined:

$$X = flatten(I)$$
$$y = c(I))$$

where $c(I)$ is a function that returns 0 if the image I shows a healthy leaf, and 1 otherwise. Consider T the vector of all images and Y the vector of their corresponding classes. 80% of the data is used for training the model:

$$(T_t, Y_t, T_e, Y_e) = split(T, Y, 0.2)$$

Finally, the model is built, by using one of the three cost functions o from Chapter 2:

$$h = lr(T_t, Y_t, o)$$

The function $h(I)$ determines the probability that a given flattened image I shows an infected leaf, i.e. there are areas with high temperatures. As a probabilistic threshold, 0.5 is used, which can be changed in the evaluation phase

4. **Constructing the cucumber Fusarium wilt convolutional neural network model.** The images are labeled as follows:

$$y = c(I))$$

where $c(I)$ is a function that returns $(1, 0)$ if the image I shows a healthy leaf, and $(0, 1)$ otherwise. Consider T the vector of all images and Y the vector of their classes. 80% of the data is used for building the model, the rest is used for evaluation:

$$(T_t, Y_t, T_e, Y_e) = split(T, Y, 0.2)$$

The architecture of the convolutional neural network is shown in Figure 5.3. Fewer filters are needed, in order to achieve satisfying results. Fewer features have to be learned, that is, the color is the most important one, which distinguishes the classes from each other. The morphology of the images plays a less important role in this model.

The network is configured as follows:

$l_1 = conv2d(I = I, in = 3, out = 32, kernel = (4, 4), stride = 2, a = relu)$
$l_2 = maxpool2d(I = l_1, kernel = (3, 3), stride = 2)$

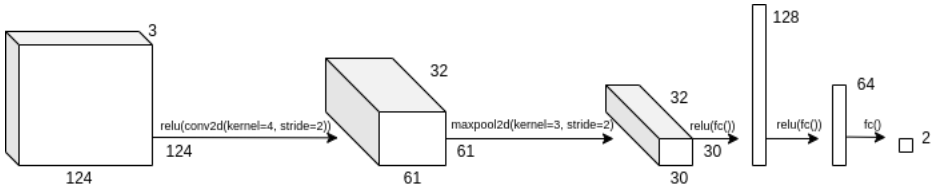


Figure 5.3: Cucumber Fusarium wilt convolutional neural network model.

$l_3 = fc(I = flatten(l_2), in = 28800, out = 128, a = relu)$
$l_4 = fc(I = l_3, in = 128, out = 64, a = relu)$
$l_5 = fc(I = l_4, in = 64, out = 2, a = identity)$

$h(I)$ is the hypothesis function that applies these steps, and Y the labeled output for I. For every normalized training image I and label Y, the cross-entropy loss function is minimized using stochastic gradient descent:

$lossFunc = cel(h, I, Y)$
$gradients = backprop(lossFunc)$
$h = sgd(h, gradients, 0.001)$

The training process is repeated 5 times. The function $h(I)$ returns a vector (p_h, p_d), where p_h is the probability that the normalized image I is healthy, and p_d the probability that it is diseased.

It is expected that the convolutional neural network will perform the best. The main challenge for this model type is the actual capturing of the images, for which special cameras are required

5.4.3 Rice sheath blight convolutional neural network model

Identifying plant diseases using remote sensing represents a challenging task. This is because of the longer distance to objects, which affects the quality of images. However, cameras have improved a lot in recent years and can acquire high-resolution images, even from longer distances. It should be possible that the images show even the morphological details of the plants. While this may still be difficult to achieve using satellites, drones can acquire images from a much closer range.

Zhang et al.[ZZZ+18] monitored a rice field for the sheath blight disease. By using a drone, which was equipped with multispectral and digital cameras, they captured images of 67 rice cultivars. This was done on 2 different days (producing 2 images of the fields): after the acquisition was finished for the first day, one end section of the field was inoculated with a sheath blight pathogen. After this change took effect, the cultivars started to change the color from green to yellow and brown, and the cultivars were captured again. This color change was used to identify the sheath blight severity in the cultivars, the change of the color model from RGB to HSL generated better results, regarding identification. For each cultivar, the severity of sheath blight was assessed with values from 0 to 9. It could also be observed how the disease started to spread across

the plants. Diseased cultivars were usually next or close to each other. Therefore, it is important to detect the disease as early as possible, in order to prevent the spreading.

Using this data, the AI model is described as follows:

1. **Data.** For the data set, the two images of the rice fields are used. The first image has a higher quality than the second one. However, for both it is difficult to see the morphology of the cultivars, even though the images are acquired from a close range (altitude of 27 meters). In order to use the data, some pre-processing is required. As a first step, with a tool, for each image, each cultivar is manually extracted (resized to 256x256 pixels, with a black background, i.e. segmented with a clean background), resulting in 134 images. In the future, this extraction process can be automated, by giving a drone fixed coordinates, for each cultivar.

   The previous models solve binary classification problems. In this model, each image is classified into the following three classes:

   - low risk of sheath blight, with a severity in the interval [0, 4), containing 26 samples
   - medium risk of sheath blight, with a severity in the interval [4, 7), containing 48 samples
   - high risk of sheath blight, with a severity in the interval [7, 9], containing 60 samples

   Each image is a vector with the shape $(1, 256, 256)$. Because there are not enough assessments that can be used for training, using one class per severity is not possible. The constructed data set is available in the supplementary materials

2. **Data pre-processing and augmentation.** Since the size of the data set is small, data augmentation is needed. Augmentation techniques are applied in the following order:

   - clockwise rotation is applied with a random angle between 0 and 360 degrees with a probability of 90%
   - images are horizontally flipped with a probability of 20%
   - noise is added to the images with a probability of 30%

   It can be seen that the classes from the original data set are imbalanced. Therefore, the augmentation is performed for as long as 600 samples are reached per class. Satellites and drones will acquire images from above with a precise fixed configured angle, that is, the images will always look flat. Because of this, they will not be skewed or have different angles, which is why the homography operation is skipped, since it would have generated unreal images. It is also worth noting that images from different classes are very similar to each other. As a pre-processing step, the images are transformed into the HSL color space, since it is reported by the authors

that this transformation increases the accuracy of the disease severity detection. Pixels are normalized the same way as for the other convolutional neural networks, by using mean and standard deviation of 0.5

3. **Constructing the rice sheath blight convolutional neural network model.** Since the data is labeled, supervised learning algorithms are used. The images are flattened and classified:

$$y = c(I)$$

where $c(I)$ is a function that returns $(1, 0, 0)$, $(0, 1, 0)$ or $(0, 0, 1)$, depending on the risk level of the disease. Consider T the vector of all images and Y their corresponding classes. 80% of the data is used for building the model, the rest is used for evaluation:

$$(T_t, Y_t, T_e, Y_e) = split(T, Y, 0.2)$$

Figure 5.4 shows the convolutional neural network. Given some image of a rice cultivar, this model assigns the image to one of the three classes. Compared to the previous convolutional neural networks, this one has some additional convolution and max pooling layers. It is very difficult to distinguish healthy from diseased cultivars, because of the similarities in colors, which is why more features need to be learned. This is expressed with the higher number of layers.

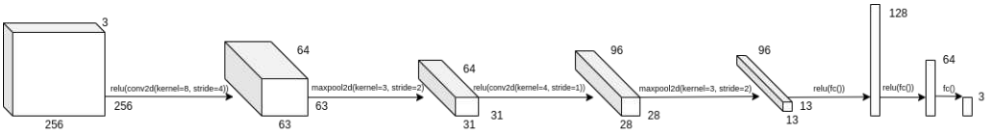


Figure 5.4: The rice sheath blight convolutional neural network model.

The model is described with the following steps:

$l_1 = conv2d(I = I, in = 3, out = 64, kernel = (8, 8), stride = 4, a = relu)$
$l_2 = maxpool2d(I = l_1, kernel = (3, 3), stride = 2)$
$l_3 = conv2d(I = l_2, in = 64, out = 96, kernel = (4, 4), stride = 1, a = relu)$
$l_4 = maxpool2d(I = l_3, kernel = (3, 3), stride = 2)$
$l_5 = fc(I = flatten(l_4), in = 16224, out = 128, a = relu)$
$l_6 = fc(I = l_5, in = 128, out = 64, a = relu)$
$l_7 = fc(I = l_6, in = 64, out = 2, a = identity)$

This network is trained using stochastic gradient descent, which minimizes a cross-entropy loss function:

$lossFunc = cel(h, I, Y)$
$gradients = backprop(lossFunc)$
$h = sgd(h, gradients, 0.001)$

The training is repeated 50 times. Due to the high similarity between the images from different classes, it is not expected that this model will be as good as the others. The function $h(I)$ returns a vector (p_l, p_m, p_h) for a pre-processed cultivar image I, where

- p_l is the probability that the rice cultivar has a low risk of sheath blight
- p_m is the probability that the rice cultivar has a medium risk of sheath blight
- p_h is the probability that the rice cultivar has a high risk of sheath blight

5.5 AI models for diseases that are caused by pests

This model uses images from a close range to detect the presence of whiteflies. Images are captured *before* a disease has occurred, i.e. diseases are not visible to the naked eye. Images always show healthy leaves, and some of them may contain whiteflies. The presence of whiteflies might be an indicator for a disease, which might develop in the future. This model tries to detect the presence of these insects, as they represent a potential threat to the plants. One typical disease that is caused by whiteflies is chlorosis, which is a yellowing of the leaves, because of the lack of chlorophyll.

A model that detects whiteflies i.e. predicts chlorosis disease is described as follows:

1. **Data.** At the time of this writing, no data sets containing images of whiteflies are found. Data with images of whiteflies on leaves is collected manually from different sources, because of its availability on the internet. Some images may also show whitefly eggs on leaves, for example, during a whitefly infestation. The images have different dimensions. As a first step, images with whiteflies on leaves are manually pre-processed. That is, noise and other irrelevant background data are removed, by using a modeling application. Then, they are resized to 256x256 pixels (the aspect ratio is ignored in this case, as it does not have much negative impact on the images, in terms of distortions), and a black background is added. This step is necessary, since the data must have the same structure, in order to be comparable. Images without whiteflies, showing just healthy leaves, also have the same dimensions and are randomly chosen from the PlantVillage data set[HS15] (segmented images are used, in order to have the same black background). There are 178 samples for each class, i.e. there are two classes:

   - the class of images of healthy leaves (from any plant)
   - the class of images of leaves that are occupied by at least one whitefly

   Each RGB image is a vector with the shape $(1, 256, 256)$. The constructed data set can be found in the supplementary materials. There are some challenges regarding this manually created data set. Images are not very similar to each other: all the images contain leaves, however, of different types. Regarding the images that

contain whiteflies, since they are coming from different sources, they have different dimensions. Whiteflies appear in some images bigger and in some others smaller. Also, the colors of the whiteflies appear in different shades of white, and some images are more saturated than others

2. **Data pre-processing and augmentation.** In order to obtain more realistic results (preventing overfitting), more images are needed. Augmentation techniques are applied in the following order:

   - clockwise rotation is always applied, with a random angle between 0 and 360 degrees
   - images are horizontally flipped, with a probability of 20%
   - noise is added to the images, with a probability of 30%

   This process is repeated for each image three times, i.e. there are overall 712 samples per class. Additionally, a pre-processing step is applied, that is, images are converted from RGB to HSV color space. The hue component makes sure that lighting variations are eliminated. This is an important improvement, since parts with a high intensity of light can be confused with whiteflies. Also, pixels are normalized for each color channel, with mean 0.5 and standard deviation 0.5

3. **Constructing the chlorosis convolutional neural network model.** Since the data is labeled, supervised learning algorithms are applied. The model should predict if there is a potential risk for a disease, due to the presence of the whiteflies. A convolutional neural network is selected for implementing this model. Other algorithms, such as logistic regression (being a linear model) and support-vector machines (being a very complex model from a computational point of view) are not expected to perform better, which is why they are skipped.

   The images are classified:

   $$y = c(I)$$

   where $c(I)$ is a function that returns $(1, 0)$ if the image I shows a healthy leaf, and $(0, 1)$ otherwise. Consider T the vector of all images and Y their corresponding classes. 80% of the data is used for building the model, the rest is used for evaluation:

   $$(T_t, Y_t, T_e, Y_e) = split(T, Y, 0.2)$$

   The convolutional neural network is depicted in Figure 5.5. It is very similar to the other networks, but with a reduced kernel size in the convolution layer. The features that need to be learned, in this case the whiteflies, are small.

   This is described with the following steps:

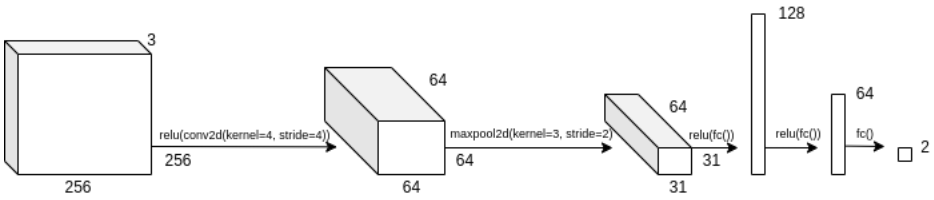


Figure 5.5: The chlorosis convolutional neural network model.

$l_1 = conv2d(I = I, in = 3, out = 64, kernel = (4, 4), stride = 4, a = relu)$
$l_2 = maxpool2d(I = l_1, kernel = (3, 3), stride = 2)$
$l_3 = fc(I = flatten(l_2), in = 61504, out = 128, a = relu)$
$l_4 = fc(I = l_3, in = 128, out = 64, a = relu)$
$l_5 = fc(I = l_4, in = 64, out = 2, a = identity)$

The hypothesis function $h(I)$ applies these steps, and Y is the expected class for I. The training is done using cross-entropy loss and stochastic gradient descent:

$lossFunc = cel(h, I, Y)$
$gradients = backprop(lossFunc)$
$h = sgd(h, gradients, 0.001)$

The training is repeated 8 times. $h(I)$ returns a vector (p_h, p_d), where p_h is the probability that the (pre-processed) image I shows a healthy leaf, and p_d is the probability that the leaf contains at least one whitefly

Model	AI model type	Cause	Parameters
Tomato late blight logistic regression model	Close-range images	Cold weather	Optimizer and probabilistic threshold
Tomato late blight support-vector machine model	Close-range images	Cold weather	Kernel
Tomato late blight convolutional neural network model	Close-range images	Cold weather	N/A
Corn northern leaf blight logistic regression model	Close-range images	Hot weather	Optimizer and probabilistic threshold
Corn northern leaf blight support-vector machine model	Close-range images	Hot weather	Kernel
Corn northern leaf blight convolutional neural network model	Close-range images	Hot weather	N/A
Chlorosis convolutional neural network model	Close-range pest images	Pest	N/A
Cucumber Fusarium wilt logistic regression model	Thermal images	Hot weather	Optimizer and probabilistic threshold
Cucumber Fusarium wilt convolutional neural network model	Thermal images	Hot weather	N/A
Rice sheath blight convolutional neural network model	Remote sensing	Hot weather	N/A
Wheat stripe rust logistic regression model	Environmental data	Cold weather	Optimizer

Table 5.2: Summary of AI models.

CHAPTER 6

Evaluation

In order to prove the usefulness of the presented models, they are evaluated using a prototype. The prototype can be trained to evaluate a specific model. This should not only answer whether a model is performing well or not, but also if it can be used in practice. It is also important to analyze whether the model is underfitting or overfitting, which are two common problems in machine learning. The evaluation is performed according to some metrics, which are defined in Section 6.1. Such metrics include the accuracy from the confusion matrix, precision, recall, and more. The evaluation strategy is described as well. Section 6.2 describes the prototype, which is not only able to evaluate a model, but also offers a graphical user interface and the possibility to be integrated with other tools, through an API. In Section 6.3, the results of the evaluation for each model are presented.

6.1 Metrics

Evaluating a model using the right metrics is a crucial part. Analyzing only one metric is often not enough. When performing classification, there are four different outcomes that a model produces for each class C:

- the true positives (TP) are the outcomes that were correctly predicted for C
- the true negatives (TN) are the outcomes that were correctly predicted for all other classes, i.e. all classes except C
- the false positives (FP) are the outcomes that were incorrectly predicted as class C. Another class would have been correct
- the false negatives (FN) are the outcomes that were incorrectly predicted as another class, i.e. any class except C. Class C would have been correct

These outcomes are part of many evaluation metrics. They are summarized as a **confusion matrix**[Ste97]. In this matrix, each column represents the number of samples for a *predicted* class and each row the number of samples for a *real* class.

Next, some other metrics are described.

Time. This includes the time that is needed to build a model. It represents the required time to learn from the data, in order to update the parameters and create a model, which is then able to predict the class for a sample. It includes the time needed for evaluation as well, which is usually shorter than the build time.

Model size. Depending on the parameters that models need to learn, they may have different sizes. Size is an important metric, since the models will need to be loaded later, in order to use them for classification purposes.

Accuracy. This metric is calculated by using the confusion matrix M and is given as: $\frac{M_{diagsum}}{M_{sum}}$, where $M_{diagsum}$ represents the sum across the diagonal of M and M_{sum} the sum of all values of M. It indicates the percentage of correctly predicted samples. This metric is not very good in case of class imbalance.

Precision. Indicates the correctly predicted samples for a class, normalized by the samples *predicted* as correct and is given as: $\frac{TP}{TP+FP}$. The overall precision is calculated as the average mean across all class precisions.

Recall. Indicates the correctly predicted samples for a class, normalized by the *actual* correct samples, and is given as: $\frac{TP}{TP+FN}$. The overall recall is calculated as the average mean across all class recalls.

F-score. Looking at precision and recall in an isolated way might still not be very useful. If every sample is predicted as positive, recall will be 1 and precision very small. Similarly, if a model predicts only real positive samples as positive, precision is high, but recall can still be small. In order to overcome this trade-off, using the F-score (sometimes also known as F1) is a better metric, as it summarizes both of these metrics. It is calculated as $\frac{2 \cdot precision \cdot recall}{precision+recall}$. A high F-score means precision and recall are also high.

ROC curve. The *receiver operating characteristic curve*[Swe86] is a commonly used graphical metric. Machine learning classification models map probabilities to classes. In binary classification, a threshold, typically 0.5, is used to predict one class or another. Finding the right threshold depends on the machine learning problem. The y-axis is the recall and x-axis are the false positive rates (FPR), given as $\frac{FP}{FP+TN}$. The goal is to find the threshold for which the recall is maximized and the false positive rate is minimized. This can be generalized for multiclass classification problems as well. This metric is used for all classification algorithms except for neural networks, such a metric does not exist for them.

Precision-recall curve. Similar to the ROC curve, the precision-recall curve is another way to visually determine the classification threshold. The y-axis represents the precision and the x-axis the recall, which both have to be maximized. This metric is used for all classification algorithms except for neural networks.

Feature map. This is a qualitative metric. In case of convolutional neural networks, the learned filters can be used to visually and manually analyze how the model is performing. Feature maps are created by applying these filters to images. Feature maps for the filters of the first convolution layers are created, since they are manually interpretable. In the next convolution layers, even though the filters are very useful for the algorithm, they cannot be visually interpreted anymore, which is why they are not used for this kind of qualitative evaluation.

Other metrics. Other metrics might be used. For instance, manually analyzing the parameters that a model has learned, as they give information about the importance of certain features.

Evaluation strategy. The models have to be trained and also tested. Therefore, the data set is split into the training data and test data. Usually, the test data represents 20% of the data set. This should avoid overfitting, but there is still a risk that the parameters of the model are manually changed (in order to achieve better results) and this model would depend on the training data. To overcome this problem, *cross-validation* is performed. In k-fold cross-validation, the model is trained using the $k-1$ folds, and the fold that is left is used for validation (and evaluated using the metrics that are discussed before). This process is repeated k times, where a typical number for k is five. Only as a final step, the model is tested with the remaining test data.

6.2 Prototype

The AI models from Chapter 5 are implemented and evaluated using a prototype. The *Plant Disease Identification System* (PDIS) is a system that assists people who work in agriculture. It can examine plants, regarding different pests or diseases. It can also be extended with new models. Figure 6.1 depicts a high-level view of PDIS. It is composed of different modules.

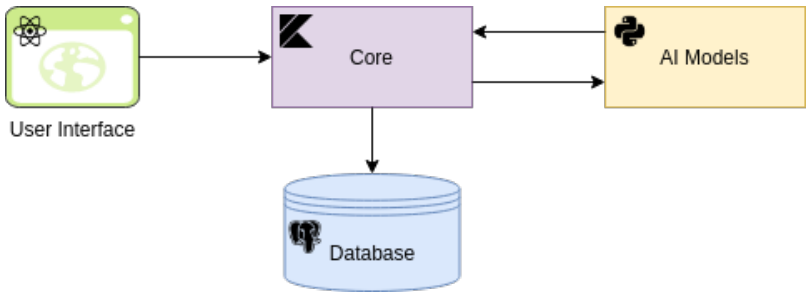


Figure 6.1: High-level description of PDIS.

Models module. This module contains all the implementations for the defined AI models, written in the Python language, which is the most popular language for machine learning. The convolutional neural networks are built around the PyTorch[PGM+19]

library, the other algorithms are implemented using Scikit-learn[PVG+11]. Both libraries are actively being improved and have a lot of support.

When the module is invoked, a configuration file (specified in JSON format) is initially loaded. Listing A.1 shows the JSON schema[1], which is used to validate the JSON configuration file. This file defines the model types (Listing A.2) and the models (Listing A.3) that are exposed by the module.

The model settings (Listing A.4) are defined as well. These settings can also be changed from the user interface - the user interface elements are built dynamically, depending on the type of setting. The settings may contain hyperparameters that are used by the model, the data set to use, and more. The descriptions of the respective schemas can be found in the supplementary materials.

The `implementation` attribute of a model points to the function that builds and evaluates the model. It should receive a list of settings (Listing A.4) and return a pair, which contains:

- the build and evaluation result, which is a JSON, with the schema defined in Listing A.5
- the actual built model, which should be serializable, in order to be loaded later again for classification purposes. The model must implement a function called `predict_file`, which takes as an argument a file and returns a prediction result, which is a JSON (Listing A.6). The `predict_file` function is invoked when classifying a file object

After loading the configuration, the module starts an HTTP server. The server exposes the functionality for building and evaluating a model. Also, it can predict file objects using that model. A file object is a sample, which must have the same structure as the samples of the data set that is used for training. Table A.1 shows a description of all functionalities, exposed through an API.

Core and database. Written in Kotlin[2], the core module is responsible for managing the built models. The models (including the results) are persisted using a Postgres[3] database instance. The core's exposed functions can be used by any client, such as the user interface module. Initially, the core module loads the configuration file from the models module and then starts an HTTP server. Table A.2 lists and describes all the endpoints that this HTTP server makes available.

User interface. The user interface for PDIS is written in React[4], a front end library that allows building user interfaces. Regarding the language, TypeScript[5] is used, which

[1] `https://json-schema.org`
[2] `https://kotlinlang.org`
[3] `https://www.postgresql.org`
[4] `https://reactjs.org`
[5] `https://www.typescriptlang.org`

is a superset of JavaScript, bringing a lot of benefits, such as static typing, type safety, immutability, null safety, and more. These features help to prevent runtime errors, which are very common in dynamic programming languages, such as JavaScript. The user interface makes use of all the functionalities exposed by the core:

- it is able to (dynamically) render information about the models types, models and their settings, as they are described in the configuration file. It allows changing these settings too
- it can build new models and evaluate them
- it can use them to predict or identify a disease

Implementing models. In order to implement a model, build it and use it to classify a file object, the following steps need to be performed (illustrated through an example):

1. In this example, as described in Listing A.8, a model is implemented in a file called `logistic_regresion.py`, inside the directory `close_range_images`. A new function `build_logistic_regression_model` is implemented, which can be used to build the model. The logistic regression model classifies images of plants as `tomato_healthy` or `tomato_late_blight`
2. A configuration file like in Listing A.9 is provided, which configures one model type and one model for it. The model points to the builder, configures the data to be used and the solver, which is rendered as a drop-down in the user interface. Optionally, cross-validation can be performed. The model and core modules are then started
3. A new model build is started. The server dispatches a new model build job and returns the build with ID 1 in `RUNNING` state, as described in Listing A.10
4. Later, when the build is successfully finished, the state is set to `FINISHED_SUCCESS` and the models-module sends the results and the model file back to the core module. Fetching the finished model build shows also the results (Listing A.11), without the cross-validation report
5. Finally, the model is used to classify an image, which shows a diseased leaf (Listing A.12)

6.3 Results

This section presents the results of the application of the described AI models to their respective data sets. All experiments are conducted using the following hardware:

- Operating System: Ubuntu 21.04 (64 bit)

- Storage: SSD
- RAM: 8 GB, with 8 GB swap space enabled
- CPU: Intel® Core™ i7-6500U CPU @ 2.50GHz × 4
- GPU: Intel Corporation Skylake GT2 [HD Graphics 520]

All the models are evaluated using Y_e and T_e, obtained from the *split* operation in the build phase from Chapter 5. The evaluation results are summarized in Table 6.35.

6.3.1 Evaluation results for AI models for diseases that are caused by cold weather

Tomato late blight logistic regression model

Tables 6.1, 6.2 and 6.3 show the evaluation results for this model. The best results are achieved by using the L-BFGS optimization algorithm. An accuracy of 90% is achieved, according to the confusion matrix.

Actual/Prediction	Healthy	Late blight
Healthy	107	11
Late blight	13	109

Table 6.1: Confusion matrix for tomato late blight logistic regression model.

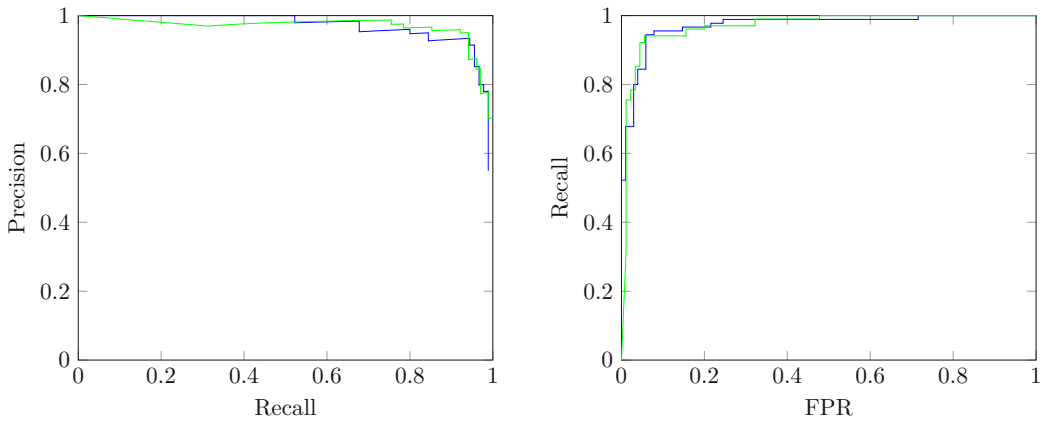
Class	**Precision**	**Recall**	**F-score**
Healthy	0.91	0.91	0.9
Late blight	0.89	0.89	0.9

Table 6.2: Class specific metrics for tomato late blight logistic regression model.

Metric	**Value**
Accuracy	0.9
Build time	98 seconds
Evaluation time	5 seconds
Model size	3.1 MB

Table 6.3: Results for tomato late blight logistic regression model.

Figure 6.2 shows the curves, which reflect the high precision, recall and F-score. With the help of these curves, a probability threshold equal to 0.9 is manually selected for predicting the late blight disease, i.e. a leaf is classified as diseased only with high certainty, requiring a high precision. Without adjusting this threshold, accuracy, precision and recall are smaller. Similar results are also obtained during cross-validation, i.e. no overfitting is present.



(a) Precision-recall curve

(b) ROC curve

Figure 6.2: Curves for tomato late blight logistic regression model, blue stands for the healthy class and green for the late blight class.

Building the model takes longer than evaluation, since $256 * 256$ features are used as an input, being a considerably high number. The model size of 3.1 MB makes this model small, allowing it to be loaded quickly for classification. Diseased images, that are misclassified as healthy, have the following characteristics:

- some images are segmented, which is causing problems for the algorithm. These images are labeled as diseased, however, they appear to be completely healthy
- the rest of the images appear under high intensity of light, therefore, also the shadow of a leaf can be seen

All the healthy images that are classified as diseased are very similar to each other, light is causing problems as well.

Tomato late blight support-vector machine model

Tables 6.4, 6.5 and 6.6 show the results after applying the support-vector machine model, with the linear kernel. The other kernels achieve similar results. Adjusting the probability threshold is not needed, using the default one (0.5) achieves the best results. Therefore, the curves, which look very similar to the ones of the logistic regression model, are not shown. Build time and size of the model are high, since complex functions need to be learned.

The healthy images that are wrongly identified with late blight are the same as the ones that the logistic regression model incorrectly identifies. Images with late blight classified as healthy do not follow a clear pattern, some have more light, some less. Overall, this model achieves good results, with the cost of high building, evaluation and also loading

times. Similar results are achieved during all folds of cross-validation, no overfitting can be detected.

Metric	Value
Accuracy	0.91
Build time	12 minutes
Evaluation time	45 seconds
Model size	258 MB

Table 6.4: Results for tomato late blight support-vector machine model.

Actual/Prediction	Healthy	Late blight
Healthy	106	9
Late blight	13	112

Table 6.5: Confusion matrix for tomato late blight support-vector machine model.

Class	Precision	Recall	F-score
Healthy	0.92	0.92	0.91
Late blight	0.9	0.9	0.91

Table 6.6: Class specific metrics for tomato late blight support-vector machine model.

Tomato late blight convolutional neural network model

From the defined models for tomato late blight disease, the best one is the convolutional neural network model. As Tables 6.7, 6.8 and 6.9 show, the accuracy and the other scores are very high. Building a model takes about 4 minutes, and evaluation is fast. Also, the model size allows a fast loading of the model for later usage. All the diseased images are classified as such. Only three healthy images are classified as diseased. For this model, the curves cannot be calculated, as it returns no probabilities as its final output, but a vector.

Metric	Value
Accuracy	0.99
Build time	4 minutes
Evaluation time	3 seconds
Model size	27.4 MB

Table 6.7: Results for tomato late blight convolutional neural network model.

Figure 6.3 shows an infected leaf, that is correctly classified. This is one of the leaves that the other models are not able to classify as diseased. The darker spots on the feature map are the interesting parts that the neural network uses in the next layers. In some

Actual/Prediction	Healthy	Late blight
Healthy	110	3
Late blight	0	127

Table 6.8: Confusion matrix for tomato late blight convolutional neural network model.

Class	**Precision**	**Recall**	**F-score**
Healthy	0.97	0.97	0.99
Late blight	1.0	1.0	1.0

Table 6.9: Class specific metrics for tomato late blight convolutional neural network model.

images, the dark spots match with the late blight spots. Other filters concentrate on the background of the image and on the edges. This is because, besides changing color, the leaf also changes the shape during infection.

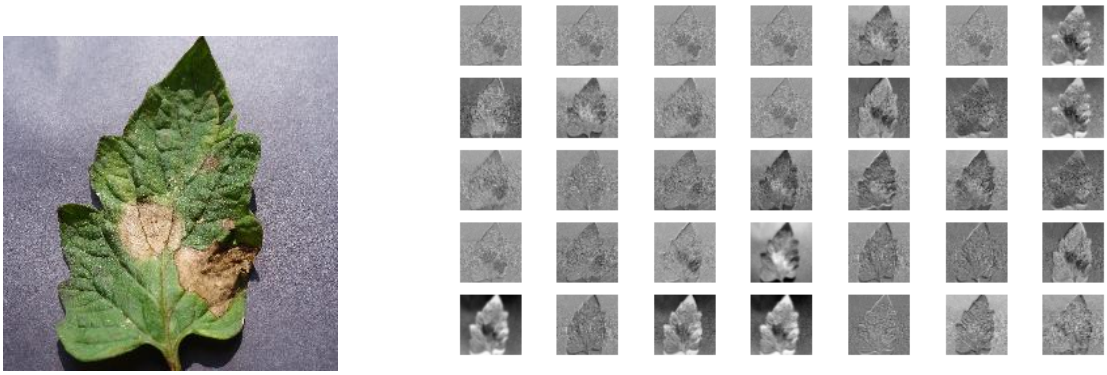


Figure 6.3: Feature map of a tomato leaf infected with late blight.

Wheat stripe rust logistic regression model

Table 6.10 shows the feature specific *odds*[Spe14] for the low risk class (0). They represent the influence of a feature on the classification result. It can be seen that a low risk is predicted with high frequencies of a humidity greater than 90% and dominant frequencies of high rainfall. Otherwise, a high risk is predicted. These properties comply with the interpretation of the data set from Chapter 5. The species has an important impact as well. Durum species are, in this data set, much less susceptible to wheat stripe rust.

Tables 6.11, 6.12 and 6.13 list the evaluation results for this model. Since only a few features need to be learned, the model is very small, with a fast build. Looking at the confusion matrix, it can be seen that only for two samples, a high risk was incorrectly predicted. The values are very favorable for development of wheat stripe rust, however, there are no incidents reported. As the authors state, this could be also because of a

Feature	Odd
Site	1.13241162
Species	3.18307169
R_1	0.64860079
R_2	1.14355611
R_3	1.23635266
R_4	1.0
H_1	0.22596029
H_2	1.33353062
H_3	1.92664038
H_4	0.76745411
H_5	3.27317224
T_1	0.9999897
T_2	0.5411552
T_3	1.49680053
T_4	1.91722448
T_5	1.25100998
T_6	1.11536433
T_7	2.53703226

Table 6.10: The odds for the low risk class of the logistic regression model.

selective application of pesticides, preventing the development of the disease. Most of the high-risk samples that are incorrectly classified as low risk have an incidence less than 50, i.e. it is close to the incidence for low risk samples. This represents a limitation of this model, which cannot distinguish between levels of disease, but is rather discrete. Therefore, this should be improved in the future, as more data becomes available.

Metric	Value
Accuracy	0.86
Build time	4 seconds
Evaluation time	1 second
Model size	1.1 KB

Table 6.11: Results for wheat stripe rust logistic regression model.

Actual/Prediction	Stripe rust low risk	Stripe rust high risk
Stripe rust low risk	27	2
Stripe rust high risk	9	40

Table 6.12: Confusion matrix for wheat stripe rust logistic regression model.

Class	Precision	Recall	F-score
Stripe rust low risk	0.93	0.93	0.93
Stripe rust high risk	0.82	0.82	0.82

Table 6.13: Class specific metrics for wheat stripe rust logistic regression model.

6.3.2 Evaluation results for AI models for diseases that are caused by hot weather

Corn northern leaf blight AI models

Tables 6.14, 6.15 and 6.16 show the evaluation results for the corn northern leaf blight logistic regression model.

Metric	Value
Accuracy	0.97
Build time	85 seconds
Evaluation time	2 seconds
Model size	3.1 MB

Table 6.14: Results for corn northern leaf blight logistic regression model.

Actual/Prediction	Healthy	Northern leaf blight
Healthy	94	4
Northern leaf blight	3	91

Table 6.15: Confusion matrix for corn northern leaf blight logistic regression model.

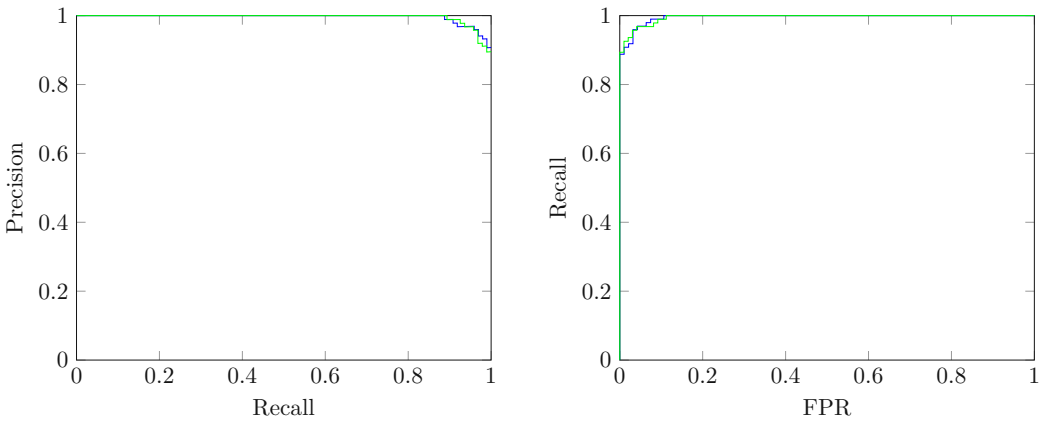
Class	Precision	Recall	F-score
Healthy	0.96	0.96	0.96
Northern leaf blight	0.97	0.97	0.97

Table 6.16: Class specific metrics for corn northern leaf blight logistic regression model.

The best results are achieved by using the L-BFGS solver. The accuracy is very high. Figure 6.4 shows the curves, which suggest that 0.85 is the best probabilistic threshold to use. Very good results are obtained during cross-validation as well. Short building times and model size of 3.1 MB make this model useful.

The support-vector machine (with the linear kernel) and the convolutional neural network models perform better. The convolutional neural network obtains the best results in terms of accuracy and building performance. See the respective tables for the evaluation results.

In general, identifying corn northern leaf blight is not very difficult. This is because of the nature of the disease, northern leaf blight is spread across all the leaf. Another reason



(a) Precision-recall curve (b) ROC curve

Figure 6.4: Curves for corn northern leaf blight logistic regression model, blue stands for the healthy class and green for the northern leaf blight class.

Metric	**Value**
Accuracy	0.99
Build time	8 minutes
Evaluation time	25 seconds
Model size	85 MB

Table 6.17: Results for corn northern leaf blight support-vector machine model.

Actual/Prediction	Healthy	Northern leaf blight
Healthy	127	1
Northern leaf blight	1	111

Table 6.18: Confusion matrix for corn northern leaf blight support-vector machine model.

Class	**Precision**	**Recall**	**F-score**
Healthy	0.99	0.99	0.99
Northern leaf blight	0.99	0.99	0.99

Table 6.19: Class specific metrics for corn northern leaf blight support-vector machine model.

are the types of images from the data set. The corn images show *only* the leaf, i.e. there is no background noise. Also, the actual shape of the leaves doesn't play an important role. For corn leaves, fewer features have to be learned.

Metric	Value
Accuracy	0.99
Build time	4 minutes
Evaluation time	3 seconds
Model size	27.4 MB

Table 6.20: Results for corn northern leaf blight convolutional neural network model.

Actual/Prediction	Healthy	Northern leaf blight
Healthy	115	1
Northern leaf blight	0	127

Table 6.21: Confusion matrix for corn northern leaf blight convolutional neural network model.

Class	Precision	Recall	F-score
Healthy	0.99	0.99	0.99
Northern leaf blight	1.0	1.0	1.0

Table 6.22: Class specific metrics for corn northern leaf blight convolutional neural network model.

Cucumber Fusarium wilt logistic regression model

Tables 6.23, 6.24 and 6.25 show the evaluation results for the logistic regression model, with the L-BFGS solver.

Metric	Value
Accuracy	0.96
Build time	1 minute
Evaluation time	1 second
Model size	710 KB

Table 6.23: Results for cucumber Fusarium wilt logistic regression model.

Actual/Prediction	Healthy	Diseased
Healthy	97	3
Diseased	4	90

Table 6.24: Confusion matrix for cucumber Fusarium wilt logistic regression model.

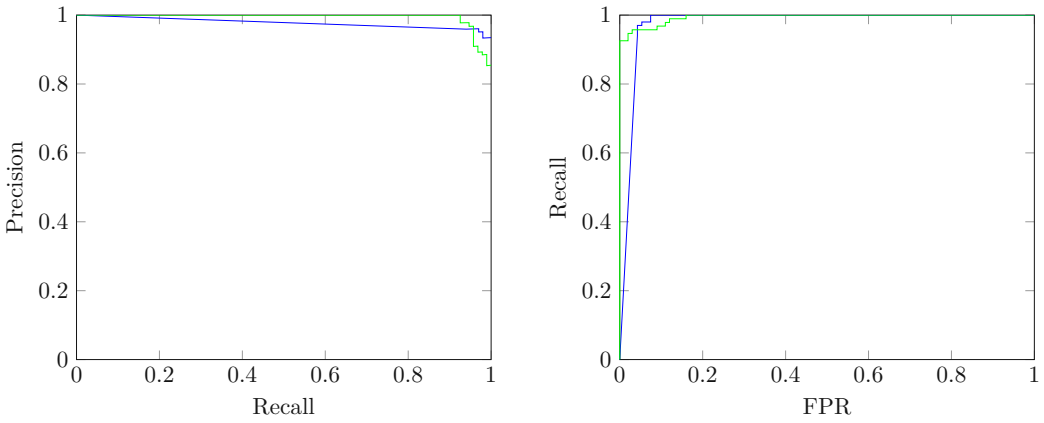
Build and evaluation times are short. The model size is small and the scores very high. Similar results are achieved during the cross-validation runs. The precision-recall curves, but also the ROC curves from Figure 6.5, show that this model is performing good. There are just a few samples that are misclassified:

Class	Precision	Recall	F-score
Healthy	0.97	0.97	0.97
Diseased	0.96	0.96	0.96

Table 6.25: Class specific metrics for cucumber Fusarium wilt logistic regression model.

- images of diseased leaves from the first day are very similar to the healthy leaves. The only difference is the temperature of the veins, which is slightly higher for the diseased leaves, making them appear in yellow color. These diseased images are incorrectly classified as healthy
- similarly, healthy images are confused with diseased images from the first day

It is worth mentioning that, even manually, it is not easy to distinguish between healthy and diseased images for day one.



(a) Precision-recall curve (b) ROC curve

Figure 6.5: Curves for the cucumber Fusarium wilt logistic regression model, blue stands for the healthy class and green for the diseased class.

Cucumber Fusarium wilt convolutional neural network model

Tables 6.26, 6.27 and 6.28 show the evaluation results for the convolutional neural network model. It is performing slightly better than the logistic regression model, with very good scores and optimal build time and size. The only confused diseased sample is one from day one, which is incorrectly classified as healthy.

Figure 6.6 shows some generated filters. The image represents an infected cucumber leaf, for which the infection has just started. It can be seen how the filters are concentrating on the infected parts, which in this case are only the veins of the leaf. Other filters concentrate, as usual, on the background of the leaf.

Metric	Value
Accuracy	0.99
Build Time	4 minutes
Evaluation Time	1 second
Model size	13.6 MB

Table 6.26: Results for cucumber Fusarium wilt convolutional neural network model.

Actual/Prediction	Healthy	Diseased
Healthy	117	0
Diseased	1	125

Table 6.27: Confusion matrix for cucumber Fusarium wilt convolutional neural network model.

Class	Precision	Recall	F-score
Healthy	1.0	1.0	1.0
Diseased	0.99	0.99	1.0

Table 6.28: Class specific metrics for cucumber Fusarium wilt convolutional neural network model.

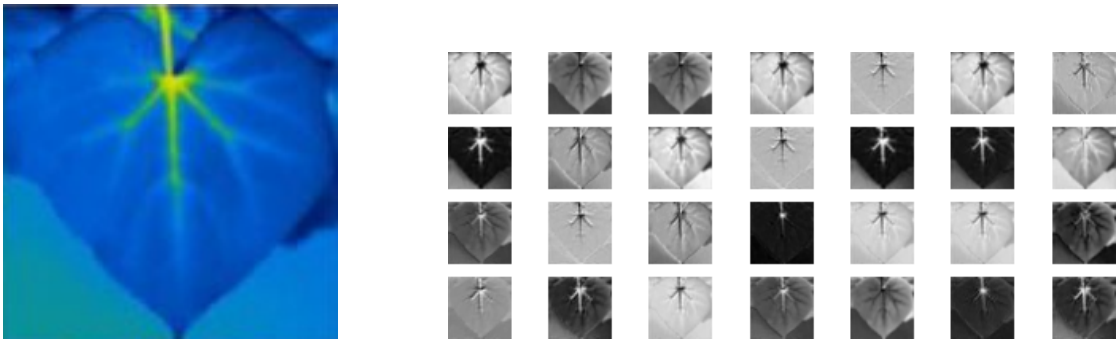


Figure 6.6: Feature map for an infected cucumber leaf in its early infection stage.

Rice sheath blight convolutional neural network model

Tables 6.29, 6.30 and 6.31 show the evaluation results for this model.

A major noticeable drawback is the time needed to build this model, which is one hour. This happens because the number of training epochs is large. Taking into account the high similarity between images from different classes, an accuracy, precision, recall and F-score around 70% is still a good score. Looking at the confusion matrix, it can be seen that the model is able to distinguish between low risk and the other two classes. This happens because the difference in color between low risk images and the others is more obvious than it is between medium and high-risk classes. This is the case for most of the

Metric	Value
Accuracy	0.73
Build time	1 hour
Evaluation time	3 seconds
Model size	8.1 MB

Table 6.29: Results for rice sheath blight convolutional neural network model.

Actual/Prediction	Sheath blight low	Sheath blight medium	Sheath blight high
Sheath blight low	91	16	12
Sheath blight medium	12	92	21
Sheath blight high	13	25	83

Table 6.30: Confusion matrix for rice sheath blight convolutional neural network model.

Class	Precision	Recall	F-score
Sheath blight low	0.76	0.76	0.77
Sheath blight medium	0.74	0.74	0.74
Sheath blight high	0.69	0.69	0.7

Table 6.31: Class specific metrics for rice sheath blight convolutional neural network model.

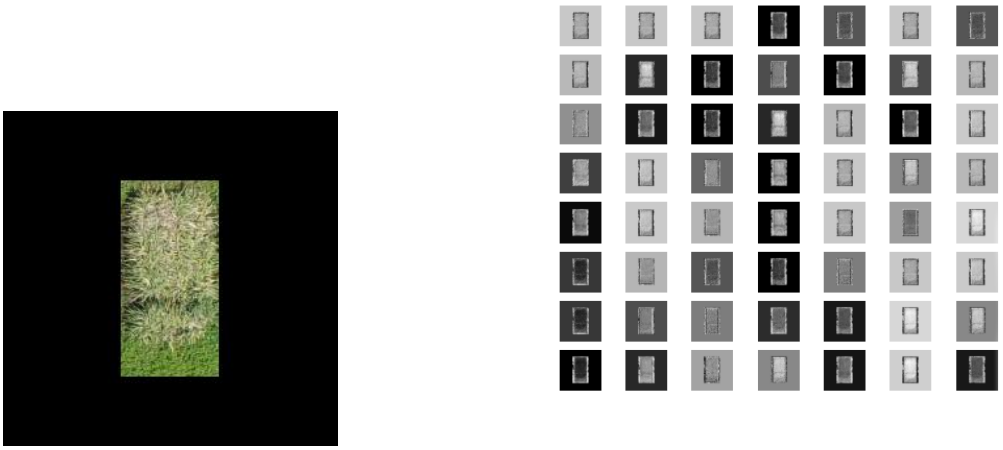


Figure 6.7: Feature map for a rice canopy, infected with sheath blight.

images. However, in some cases, a color change doesn't necessarily mean the presence of the disease, making the color not the ideal feature to solve this problem. Besides the color, the morphology of the cultivars could have been an important feature. With a better quality, it would have been possible to detect problems, such as collapsed plants.

Closer images were captured from an altitude of 5.5 meters, but only 4 cultivars were made available by the authors, not being enough to construct a proper data set, with a high diversity.

Figure 6.7 shows the feature map for the filters (kernels) from the first convolutional layer, for a high-risk rice sheath blight image. Filters are focusing on some areas of the canopy, which cannot be seen normally. By using these filters, the model is able to correctly classify this image.

6.3.3 Evaluation results for AI models for diseases that are caused by pests

Tables 6.32, 6.33 and 6.34 show the evaluation results for the chlorosis convolutional neural network model.

Metric	**Value**
Accuracy	0.91
Build time	7 minutes
Evaluation time	3 seconds
Model size	29.2 MB

Table 6.32: Results for the chlorosis convolutional neural network model.

Actual/Prediction	Healthy	Whitefly
Healthy	134	15
Whitefly	11	125

Table 6.33: Confusion matrix for the chlorosis convolutional neural network model.

Class	**Precision**	**Recall**	**F-score**
Healthy	0.9	0.9	0.91
Whitefly	0.92	0.92	0.91

Table 6.34: Class specific metrics for the chlorosis convolutional neural network model.

During cross-validation, the accuracy, precision, recall and F-score are always around 0.9. This is an acceptable result, considering the high diversity of the data set. Rotation is one of the augmentation operations that are performed, and the results show that the model is also invariant against it. The building takes time, because of the number of epochs. The model size is suitable for loading. The evaluation itself is very fast. Images of healthy leaves that are confused as images with whiteflies have the following characteristics:

- some leaves have small white spots. These are detected as whiteflies
- the white veins of the leaves are causing problems as well

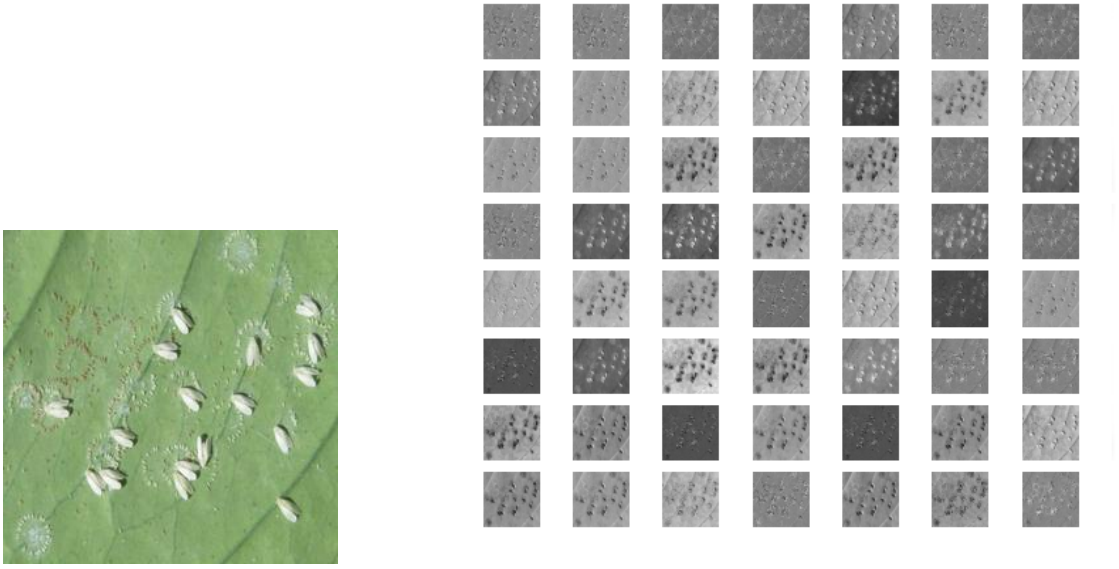


Figure 6.8: Feature map for a leaf with whiteflies.

- light, reflecting on some leaves, is very similar to some whitefly infestation images. Even manually, it is difficult to distinguish them at first glance

Images with whiteflies classified as healthy have the following characteristics:

- for some images, even manually, it is difficult to see that they contain whiteflies
- some images have very few whiteflies
- the rest of the images are confused due to the diversity of the data set. The healthy images are more similar to each other than the ones that contain whiteflies

In general, if an image is confused, so are its augmented versions.

Figure 6.8 shows the feature map, generated after classifying a test image. It can be seen that some filters are concentrating not only on the whiteflies, but also on their eggs, which are more difficult to see, because of their small size. Results show that this is the case for most of the whiteflies, i.e. only a few are missed. Other filters are focused on the background.

Model	Build time	Eval. time	Size	Acc.	Precision	Recall	F-score
Tomato late blight logistic regression model	01:28	00:05	3.1	0.9	Healthy: 0.91 Diseased: 0.89	Healthy: 0.91 Diseased: 0.89	Healthy: 0.9 Diseased: 0.9
Tomato late blight support-vector machine model	12:00	00:45	258	0.91	Healthy: 0.92 Diseased: 0.9	Healthy: 0.92 Diseased: 0.9	Healthy: 0.91 Diseased: 0.91
Tomato late blight convolutional neural network model	04:00	00:03	27.4	0.99	Healthy: 0.97 Diseased: 1.0	Healthy: 0.97 Diseased: 1.0	Healthy: 0.99 Diseased: 1.0
Corn northern leaf blight logistic regression model	01:25	00:02	3.1	0.97	Healthy: 0.96 Diseased: 0.97	Healthy: 0.96 Diseased: 0.97	Healthy: 0.96 Diseased: 0.97
Corn northern leaf blight support-vector machine model	08:00	00:25	85	0.99	Healthy: 0.99 Diseased: 0.99	Healthy: 0.99 Diseased: 0.99	Healthy: 0.99 Diseased: 0.99
Corn northern leaf blight convolutional neural network model	04:00	00:03	27.4	0.99	Healthy: 0.99 Diseased: 1.0	Healthy: 0.99 Diseased: 1.0	Healthy: 0.99 Diseased: 1.0
Chlorosis convolutional neural network model	07:00	00:03	29.2	0.91	Healthy: 0.9 Diseased: 0.92	Healthy: 0.9 Diseased: 0.92	Healthy: 0.91 Diseased: 0.91
Cucumber Fusarium wilt logistic regression model	01:00	00:01	0.7	0.96	Healthy: 0.97 Diseased: 0.96	Healthy: 0.97 Diseased: 0.96	Healthy: 0.97 Diseased: 0.96
Cucumber Fusarium wilt convolutional neural network model	04:00	00:01	13.6	0.99	Healthy: 1.0 Diseased: 0.99	Healthy: 1.0 Diseased: 0.99	Healthy: 1.0 Diseased: 1.0
Rice sheath blight convolutional neural network model	60:00	00:03	8.1	0.73	Low risk: 0.76 Med risk: 0.74 High risk: 0.69	Low risk: 0.76 Med risk: 0.74 High risk: 0.69	Low risk: 0.77 Med risk: 0.74 High risk: 0.7
Wheat stripe rust logistic regression model	00:04	00:01	0.01	0.86	Low risk: 0.93 High risk: 0.82	Low risk: 0.93 High risk: 0.82	Low risk: 0.93 High risk: 0.82

Table 6.35: Summary of evaluation results. Times are in minutes, sizes in MB.

CHAPTER 7

Discussion

In this chapter, AI models are compared with conventional models. For each model type and disease, the best AI model is selected and compared with the respective conventional model(s), also of the same model type, identifying the same disease. The comparison results are summarized in Table 7.1.

7.1 Comparison of models for diseases that are caused by cold weather

Tomato late blight convolutional neural network model vs image-processing model

The convolutional neural network model is selected for comparison. According to the metrics from the previous chapter, this model has the highest scores in terms of accuracy, precision and recall. It has an optimal size, allowing it to be loaded fast, in order to perform the classification. Through the feature maps, this model does (visually) provide details about the learned features.

The tomato late blight image-processing model is tested with the tomato late blight data set, available in the supplementary materials. It is able to detect, in most of the cases, the correct spots where the leaves are infected. However, besides detecting these spots, it is also wrongly indicating other regions as diseased. Since the leaves change their shape (that is, they fold up) during the disease, in some images, shadows appear over the leaves. These shadows are wrongly detected as infected spots, since their regions have a different color than the leaf. For the convolutional neural network model, this problem does not occur. Since it is trained enough, its filters are focusing also on the (changed) shape of the leaves.

The image-processing model is also having troubles with high light intensities on a leaf and is identifying these spots as diseased. While this happened only a few times for

the convolutional neural network model, this problem occurs much more often with the image-processing one. Another disadvantage for the image-processing model is that it is very sensitive to background noise. Sometimes, it ends up segmenting the entire leaf, instead of the diseased spots on it. This is for healthy tomato leaves, the case. According to the feature maps, the convolutional neural network model creates filters that focus on the background too, preventing this problem from happening.

The convolutional neural network is more focused on identifying whether a disease is present or not, not offering a possibility to see *where* the disease occurs. However, feature maps provide a way to visualize the diseased spots. Because of this and its better practical application, it can be concluded that it performs better than the image-processing model.

Wheat stripe rust logistic regression model vs weather model

Both models are tested with the wheat weather data set, available in the supplementary materials. The wheat stripe rust logistic regression model can only distinguish between low and high risk for wheat stripe rust, while the weather model is able to assess more levels of the disease severity, from 0 to 100. The logistic regression model is more discrete because more data is needed, this would allow assessing more levels too.

As the evaluation results show, the logistic regression model is able to learn that low rainfall and a low frequency of humidity greater than 90% during a week are suitable for the development of the disease. This is nothing but what the authors of the weather model report in their work. Both models are light from a computational point of view, i.e. no special hardware capabilities are required. The accuracy is for both models around 90%. The results from the models depend on the site and wheat species. Therefore, a general model for identifying wheat stripe rust does not exist and the location and species need to be taken into consideration too, when performing the calculations. The two models find that durum wheat is more resistant against wheat stripe rust disease than the bread species.

Both models, however, have their limitations. They assume that all the assessed plants have the same disease severity, which is normally not the case in practice. Also, there are other factors that might have an impact, such as whether pesticides are applied or not. These need to be integrated into the process of building a plant disease identification model as well. For both models, their robustness depends on the available data. For the logistic regression model, training data is essential to prevent problems such as overfitting. The weather model needs more data, in order to verify the correctness. The authors state that more assessments are needed so that the model can generalize better.

The logistic regression model is able to perform at least as good as the weather model. With enough data, it can be extended to identify different levels of the disease. An advantage of the logistic regression model is, as it is typical in machine learning, that there is no need to describe how to classify the data. Due to the nature of machine learning algorithms, they are able to learn patterns, as more data and features come in,

allowing them to improve over time. The authors of the weather model needed to look into historic weather data to find a relationship between the weather variables and the disease, which is a much more time-consuming task. For both models, it is still difficult to use them in a real farm, because of the lack of data. They both opt for automated plant disease identification and serve as a good starting point for further development and research.

7.2 Comparison of models for diseases that are caused by hot weather

From the three corn northern leaf blight AI models, the convolutional neural network model delivers the best evaluation results. It is compared with two conventional models. In the supplementary materials, the corn northern leaf blight data set can be found, which is used for testing the models.

Corn northern leaf blight convolutional neural network model vs image-processing plant disease severity model

The image-processing plant disease severity model is able to identify parts that are infected with northern leaf blight. Segmentation makes this model robust to background noise. However, sometimes, the edges of the leaf are not removed. Not all healthy pixels are removed, some can still be seen. This model is, in some situations, sensitive to shadows and high light intensities. Sometimes, the tail of a leaf and its veins are not removed, due to the color difference. This is the case also for the few healthy images that the convolutional neural network model incorrectly classifies as diseased.

The operations that the image-processing plant disease severity model performs on an image are less complex than what the convolutional neural network model does, which makes the process of identification quicker.

The convolutional neural network model uses more resources than traditional computer vision algorithms do. However, in recent years, the processing power has increased and does not represent a big challenge anymore, as it might have been in the past.

The conventional model does not require a training phase and consumes fewer resources. Sometimes, it wrongly detects and classifies some parts of the leaves as diseased. Because of this, it is not very suitable for automated disease identification, which is its biggest disadvantage, especially for practical application in agriculture. However, it is useful when analyzing images manually.

Corn northern leaf blight convolutional neural network model vs color features plant disease identification model

The color features plant disease identification model is another conventional approach. The healthy corn images from the data set are shot under high light intensities. These

images contain some white spots. These spots are incorrectly segmented as diseased parts by this model. The veins, having a different (whiter) color than the rest of the leaf, are causing problems. The model deals better with shadows, these are successfully eliminated. Overall, the corn northern leaf blight convolutional neural network model is able to correctly classify nearly all images, regardless if they show healthy or diseased leaves, independent of the conditions on which they were captured. Therefore, it is a better choice to (automatically) identify northern leaf blight of corn.

Combining AI models with conventional models might lead to an improvement. The computer vision based approaches can be used to pre-process images for the AI models. This is a very common data preparation step.

Cucumber Fusarium wilt convolutional neural network model vs infrared image-processing plant disease identification model

Both models use the cucumber thermal leaf images data set, which can be found in the supplementary materials. The cucumber Fusarium wilt convolutional neural network model is selected for comparison, because its evaluation results are the best. It can be trained in a short amount of time, and classification is fast. Although the infrared image-processing plant disease identification model focuses on detecting diseases for tea leaves, the authors also state that their approach can be used for other plant types as well.

The goal of the conventional model is to remove healthy parts from the leaf image, leaving out only the diseased ones. It is correctly detecting the red spots for the diseased leaves. Since this model applies image-processing techniques, it is rotation angle-invariant. The convolutional neural network model has this property too.

The conventional model does not require any training or complex algorithms to classify images, making it faster and less CPU-intensive. A shortcoming of this model is that it focuses only on the red color to detect an infected leaf. It is not able to detect infected leaves in their earlier stages of the disease. That is, the model is removing yellow parts from the leaf, although they represent diseased parts. The convolutional neural network model does better in this regard.

The conventional model is very sensitive to background noise. If the leaves are completely healthy and a background with a different color is present, then, in some situations, the background is removed, but not the leaf. As it is the case for other conventional models that are based on image-processing, also this one does not completely remove the healthy areas, even though it is applying operations such as erosion, making it difficult for automated use.

Because of this, the convolutional neural network model is the better choice to be used for disease identification. Because this model focuses on color features, fewer filters are needed to be learned, making it less CPU-intensive than other AI models. However, this comes at a cost. While the actual process of identification is easier than it is for

the other models, the environmental dependence is higher. Images need to be taken in ideal conditions, which is often difficult, due to the temperature of the environment and sunlight. This can lead to incorrect infrared images.

Not only the weather conditions can have an impact, but also objects being close to the plants. For example, because of shadows and reflectance coming from these objects, the cameras might deliver different results. For a perfect infrared image, a flat leaf morphology is the best, making the angle from which the images are taken also an important choice. Only if all these conditions are met, then this model will have a practical application. Regarding the cameras, fortunately, they have become affordable in recent years. This type of model cannot be used for all the diseases, since not for all plants the temperature will change when there is a lack of water.

Rice sheath blight convolutional neural network model vs multispectral imaging model

The models are compared using the rice remote sensing data set, available in the supplementary materials. It is worth mentioning that the rice sheath blight multispectral imaging model can distinguish between nine disease severities, while the convolutional neural network model can do it only for three levels. The reason for that is that there is not enough data, which is needed for training. When enough training data for each class is available, then this model can be extended to differentiate between nine classes. This also holds for the conventional model, i.e. it will be able to deliver more realistic results, the more data is available. This data has to be acquired under ideal weather conditions, otherwise both of these models will not output useful results.

The main disadvantage for the convolutional neural network model is the time needed for training. This is because of the high number of training epochs. This problem does not exist for the conventional model, which is only based on calculations on the fly, using formulas. A long calculation time shouldn't be a big problem for practical application, since this process can happen in the background from time to time, not affecting the usage at all. It will be possible to reduce this number only when the quality of the images will be higher, allowing to observe more details of the canopies, such as their morphology. For instance, an indicator to the severity of the disease could be the number of collapsed plants. While the authors of the conventional model state that using color features is only possible to qualitatively assess the severity of a disease, this is not the case for the convolutional neural network model. As the evaluation results show, this model is still able to quantify three levels of rice sheath blight. However, it is worth mentioning that a color change of the tissue from green to yellow or brown does not necessarily mean that the tissue is infected. This remains an important disadvantage for the convolutional neural network model, if based only on color features.

The authors of the conventional model also report that using NDVI (image) data, better results can be obtained. Their model can quantify disease levels with an accuracy of 63%. Considering that the AI model uses the normal RGB images and can distinguish

between three levels of the disease with an accuracy of at least 70%, it offers a good alternative to the conventional one. Both models can be used for automated disease monitoring. One drawback of the conventional model is that for each canopy, a sampling area needs to be selected, and this area can affect the accuracy of detecting sheath blight. The convolutional neural network model is able to automatically learn which area of the canopy to concentrate on. By using NDVI images, it is easier to visually immediately see the severities of the disease. This is not possible with model the convolutional neural network model, the feature maps will only be helpful in some situations. Unfortunately, the NDVI data is not made available by the authors, which is why it can't be used to develop an AI model. This should serve as a reference for further research in this direction, since it is expected that an AI model would outperform the conventional model, if the needed training data would be available.

7.3 Comparison of models for diseases that are caused by pests

For whitefly identification, the chlorosis convolutional neural network is compared with the conventional models. As the evaluation results show, it has a good size, so that it can be loaded fast for classifying images. Although some conventional models are focusing more on counting whiteflies, they can be used for comparison as well. That is, if they count at least one whitefly, then there exists a potential risk for a disease, such as chlorosis. If zero whiteflies are counted, then the leaf is considered as healthy. For all models, the whiteflies data set from the supplementary materials is used.

Chlorosis convolutional neural network model vs image-processing detection model

Since the image-processing whiteflies detection model checks for white colors, it is image rotation invariant. In many situations, whiteflies are correctly detected. However, different intensities of light and shadows are causing problems and are therefore not removed. The same holds for veins and white spots (which are not whiteflies) on healthy leaves. Also, whiteflies being very close to each other are detected as a single one. This is the case, especially for images showing whitefly infestation. Small eggs are also difficult to detect. As it can be seen from the feature map, the convolutional neural network model is better in this regard. The conventional model has problems with the background of the leaves too. That is, due to the difference between background color and leaf color, a healthy leaf is detected as a whitefly. The model works better without the presence of background noise.

Adding noise (small dots) across the image is one of the augmentation operations that increases the image entropy. The color model that is used for the transformation is chosen based on this entropy, in order to detect transitions of colors, i.e. transitions from the leaf color to the whitefly color and vice-versa. This should only happen for the

whitefly, but because of the color variations, it does also happen for the added noise. The convolutional neural network model, if trained enough, is very robust and resistant to this noise.

Another disadvantage is the test data that the conventional model uses, which consists only of a few samples. Because of this, this model does not tend to generalize very well. In general, it is difficult to use this conventional model for automated whitefly identification, making it not very usable in a real system to assist farmers, which is its major drawback. This conventional model should have a different focus - it should serve more for researching purposes.

Chlorosis convolutional neural network model vs image-processing counting model

The image-processing whiteflies counting model was tested with over 400 images when it was implemented by the authors. All images were manually segmented as a first pre-processing step, not taking into account the morphology of the leaf, which should make things easier, but can represent a disadvantage in practice. Several color transformations were proposed and one was selected, since it was reported that it performs the best for whiteflies, in all their life cycle stages. The authors reported a few limitations for this model. First, the model was tested always under perfect light conditions. Second, the model was tested only with whiteflies on soybean leaves. According to the results, the convolutional neural network model does not have these limitations.

As part of the comparison, the model was also tested with the constructed whiteflies data set. In some small images, which are heavily occupied by whiteflies, this model does not detect any of them, due to them being too close to each other. In some other images, veins and stems of the leaves are not removed and detected as whiteflies. High light intensities parts are successfully removed. However, this does also include the whiteflies on these parts. The same behaviour is observed with low light conditions, in this case, no whiteflies are detected. In general, the bigger the whiteflies on the image, the better this model performs. Exceptions include situations, where the whitefly is very big, occupying more than 50% of the image, then it is also not detected. Almost for every image, some whiteflies are missed.

The convolutional neural network model, being also invariant to the whitefly size, does not have these problems, as the results show. From the conventional models that were analyzed, this one obtained the best results.

Chlorosis convolutional neural network model vs image-processing pest detection model

The image-processing pest detection model is more focused on the general detection of pests and insects on leaves, which makes it applicable for detecting whiteflies too. This model was trained only with a few images, i.e. it does not generalize very well. Like

the other models, it has problems with high and low light intensities, leaf veins, and stems. Whitefly eggs are difficult to detect. It performs better on bigger whiteflies. For healthy images, like the image-processing whiteflies counting model, it does not detect any whiteflies, if there is an external disturbance, such as light. This model produces similar results to the image-processing whiteflies counting model. The convolutional neural network model is a better choice, in terms of accuracy and practical application in agriculture.

Models that are focused on counting the whiteflies are more suitable for integration into an automated system for classification. However, they are too sensitive against varying light intensities and other external factors. They do not require training data like the AI models do, which makes them very light from a computational point of view. However, a disadvantage is the test data they use. Either it consists only of a few samples, or images were taken in too perfect conditions. They are too dependent on the data. Therefore, it is expected that the convolutional neural network model will perform better.

For identification, the feature maps show how the whiteflies are identified. The accuracy is high, making the AI model, if trained with enough data, the better choice for automated whitefly identification. Training and building the convolutional neural network model takes some time, and this is a drawback. However, building newer models is a process that can happen in the background from time to time, while the current one can still be used, not affecting or blocking the usage.

Models compared	Data	Conventional model	AI model
Tomato late blight convolutional neural network model vs image-processing model	Tomato late blight	- identifies shadows as diseased - sensitive to light and noise - locates disease - not suitable for practical use	- shadows and noise ignored - feature maps show diseased areas - computationally more expensive - practical automated application
Wheat stripe rust logistic regression model vs weather model	Wheat weather	- can assess 100 disease levels - predicts disease using formula - all plants equally susceptible - application in a real farm difficult because of lack of data	- can assess low or high risk - learns the formula, builds fast - ignores external factors - application in a real farm difficult because of lack of data
Corn northern leaf blight convolutional neural network model vs image-processing plant disease severity model	Corn northern leaf blight	- edges, veins and tails of leaves detected as infected - sensitive to shadows and light - useful for analyzing images manually	- sometimes, veins and tails interpreted as diseased spots - computationally more expensive - suitable for practical automated application
Corn northern leaf blight convolutional neural network model vs color features plant disease identification model	Corn northern leaf blight	- shadows ignored, but light causes problems - useful for analyzing images manually	- requires some computation power, but nearly all images correctly classified - suitable for practical application
Cucumber Fusarium wilt convolutional neural network model vs infrared image-processing plant disease identification model	Cucumber thermal leaf images	- not able to detect early stages of the disease - sometimes, problems with background noise - not suitable for practical use	- requires more resources, but identifies disease - suitable for practical application only if external factors are considered too
Rice sheath blight convolutional neural network model vs multispectral imaging model	Rice remote sensing	- detects 9 levels of disease - requires manual sampling of area of interest - more data available for testing - can be used in agriculture	- detects 3 levels of disease - requires computation time - color change does not always indicate an infection - can be used in agriculture if more data is available
Chlorosis convolutional neural network model vs image-processing detection model	Whiteflies	- light, shadows, leaf veins, noise confused with whiteflies - does not detect leaves with high whitefly infestation - not suitable for practical use	- very resistant to noise - classifies nearly all images correctly - suitable for practical application
Chlorosis convolutional neural network model vs image-processing counting model	Whiteflies	- assumes ideal conditions - tested with soybean leaves - detects veins as whiteflies - issues with eggs and infestation - not suitable for practical use	- resistant to light and takes morphology of leaves into account - tested with different leaf-types - invariant against whitefly size - suitable for practical application
Chlorosis convolutional neural network model vs image-processing pest detection model	Whiteflies	- does not generalize well - problems with varying light intensities and leaf veins - does not detect whitefly eggs - experimental, not suitable for practical automated application	- tested with a data set that has a high diversity - detects whitefly eggs - suitable for practical automated application

Table 7.1: Summary of comparison results. The AI models require more computation time, but generalize better and can be used in agriculture.

CHAPTER 8

Conclusion

8.1 Summary

This thesis presents models that are able to identify pests and diseases in plants, based on algorithms from Artificial Intelligence (AI). A tool is implemented, which allows:

- integrating models that are developed in the Python programming language
- customizing these models through a user interface, which is able to generate views that can manipulate model properties
- building and evaluating models
- using models to predict/identify pests/diseases in plants
- integration with other tools, through an API
- growers to monitor their agricultural fields

The implemented models should also inspire for more research in this direction. The first step would be the acquisition of more data, allowing not only to build new models, but also verifying that they work as intended, i.e. they are not biased.

Additionally, the AI models are compared with conventional models, which are traditional models that do not use AI techniques. Models are categorized into model types, which serve to group similar models together. Some model types may use weather data, others use image data. For five of the identified model types, AI models are defined, and conventional models are identified as well. Comparison results show that the AI models are able to outperform the conventional models, in most of the situations. The availability of data is imperative to achieve this. Challenges include lack and diversity of the data. Data augmentation techniques are described and applied.

The models are evaluated according to different metrics, which include quantitative features, such as accuracy and the time needed to build a model, but also qualitative features, like the possibility to integrate the models into a real-life scenario. In image classification tasks, convolutional neural networks usually perform the best, but other machine learning algorithms are able to deliver acceptable results too. Conventional models that use images for disease identification are based on computer vision algorithms and have many limitations. Different light conditions, shadows and other external factors cause problems and make these models often return incorrect results. Machine learning is able to overcome these problems by learning from these special cases. Also, conventional models use only a few images for testing, i.e. they don't generalize very well. Models focus too often on the color, in order to identify a disease, which doesn't necessarily indicate the presence of a disease. Models should also verify the shape of the plants. Some diseases are easier to detect, because of their simpler patterns.
Conventional models that use weather data for disease identification are defined by *manually* deriving a formula from historical data, while the AI models are able to learn from the data automatically. AI models are more suitable for integration into real applications.

Chapter 1 gives an overview about the thesis and the motivation behind it. It defines the problem and the methodology as well. Chapter 2 provides the required knowledge. In Chapter 3, similar work is researched. Chapter 4 identifies model types and conventional plant/pest disease identification models. In Chapter 5, AI models are defined. A prototype is implemented and used to evaluate the AI models, described in Chapter 6. In Chapter 7, conventional and AI models are compared. This chapter concludes this work and discusses its limitations, but also the future.

8.2 Limitations

Both, the models and the tool have their limitations. Work must be carried out to improve them. The following shortcomings arise:

- all the models distinguish between healthy and diseased plants. However, they only check for *one* particular disease. They should be extended to identify different diseases, not only one. This limitation is related to the unavailability of the data
- more data sets are needed. Currently, only the PlantVillage data set[HS15] looks to satisfy requirements, regarding size and diversity. Data sets, which consist only of a few samples, are more theoretic. Therefore, plant disease data sets with thermal images, pest images, remote sensing images and weather data are needed
- all plants are considered *equally* susceptible to a disease. For all data sets, external factors need to be considered as well, for instance, whether pesticides were applied or not. They might have an impact, especially on the models that predict diseases. This is the case for models that use weather data as an input

- the quality of the images needs to be higher. This is the case for satellite images. The more detailed they are, the more features will be learned. AI techniques could be applied to improve the quality of the images too

- the performance of the models that use images as an input depends on the size of the images. Resizing images, or applying other transformations, might make the models run faster. Especially, support-vector machine models have a large size, which has to be reduced

- model types are described in an informal way, not constraining the implemented models

- the usability of PDIS can be improved. Especially, showing better error messages is important, since they can give information about failing model builds. Also, the way data set files are passed around should be improved. Currently, it is expected that the data sets are available on the server side. Instead, users should be able to send data from their machines, through the browser. Also, model builds cannot be stopped

8.3 Future work

This thesis provides a good basis for future work:

- only supervised learning models are used. Unsupervised learning algorithms should be applied as well for plant disease identification

- models that are able to assess the severity of a specific disease should be developed

- different models can be combined. For instance, a composite model can be built, which uses a combination of weather data and close range images to identify diseases

- in Chapter 4, some model types are briefly mentioned. These include model types that use morphological data of plants over time, fluorescence imaging, hyperspectral imaging, and spectroscopy. For all these model types, models should be implemented too. The first step would be acquiring the necessary data

- the models should be able to identify *where* the disease occurs

- a monitoring tool should be developed, which focuses more on the automation part of plant disease identification. For monitoring, drones or cameras at fixed positions can be used. These will continuously interact with PDIS. They will use it to predict/identify diseases, and if it is the case, the tool should warn the farmers about this event. Depending on the event, pesticides could be automatically applied too

Protecting the global food supply is not an easy task. Preventing plant disease outbreaks is important for an efficient production. In order to achieve this, the process of disease identification should be automated in the future. This will be possible by applying identification and monitoring tools that are able to work in an independent way.

APPENDIX A

Code listings and tables

This appendix contains code listings that are used throughout this thesis. Tables that describe the endpoints are included as well.

A.1 Description of data types

This section describes the data types that are used by the modules of PDIS.

```
{
  "$id": "https://com.enrimiho.pdis/config_schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "description": "Description of the PDIS config",
  "type": "object",
  "properties": {
    "modelTypes": { "type": "array", "items": { "$ref": "file:model_type_schema.json" }
        },
    "models": { "type": "array", "items": { "$ref": "file:model_schema.json" } }
  }
}
```

Listing A.1: config_schema.json

```
{
  "$id": "https://com.enrimiho.pdis/model_type_schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "description": "Description of the model type",
  "type": "object",
  "required": [ "id", "name", "description" ],
  "properties": {
    "id": { "type": "number", "description": "Must be unique" },
    "name": { "type": "string" },
    "description": { "type": "string" }
  }
}
```

Listing A.2: model_type_schema.json

```
{
  "$id": "https://com.enrimiho.pdis/model_schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
```

```
  "description": "Description of the model",
  "type": "object",
  "required": [ "id", "modelTypeId", "name", "implementation", "plant", "disease", "
      settings"],
  "properties": {
    "id": { "type": "number", "description": "Must be unique, also across all model
        types" },
    "modelTypeId": { "type": "number", "description": "Must refer to an existing model
        type" },
    "name": { "type": "string" },
    "implementation": { "type": "string", "description": "Path to model builder and
        evaluator" },
    "plant": { "type": "string" },
    "disease": { "type": "string" },
    "settings": { "type": "array", "items": { "$ref": "file:settings_schema.json" } }
  }
}
```

Listing A.3: model_schema.json

```
{
  "$id": "https://com.enrimiho.pdis/settings_schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "description": "Allows to configure settings for a model and also how the user
      interface should render them",
  "type": "object",
  "required": ["id", "type", "title", "value"],
  "properties": {
    "id": { "type": "number", "description": "Must be unique in a model or list" },
    "type": {
      "type": "string",
      "enum": [ "INPUT", "OPTIONS", "OPTION" ],
      "description": "INPUT should render as a text field, OPTIONS as a drop down (with
          possible values defined in options attribute), OPTION as a checkbox "
    },
    "name": { "type": "string" },
    "value": {"type": ["number", "string", "boolean"], "description": "Represents the
        value of a setting"},
    "options": {
      "type": "array",
      "items": { "type": "string" },
      "description": "The possible values for the OPTIONS setting type"
    }
  }
}
```

Listing A.4: settings_schema.json

```
{
  "$id": "https://com.enrimiho.pdis/build_result_schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "description": "A representation of a build and evaluation result",
  "type": "object",
  "required": ["buildTime", "evaluationTime", "classificationReport", "
      crossValidationReports", "labels"],
  "properties": {
    "buildTime": { "type": "number" },
    "evaluationTime": { "type": "number" },
    "labels": { "type": "array", "items": { "type": "string" }
    },
    "classificationReport": { "$ref": "#/$defs/classificationReport" },
    "crossValidationReports": {"type": "array", "items": { "$ref": "#/$defs/
        crossValidationReport" }}
  },
  "$defs": {
      "classificationReport": {
        "required": ["accuracy", "classReports", "confusionMatrix"],
        "properties": {
          "accuracy": { "type": "number" },
          "confusionMatrix": {"type": "array", "items": {"type": "array", "items": {"
              type": "number"}  } },
```

```
          "classReports": { "type": "array", "items": { "$ref": "#/$defs/classReport" }
              }
        }
      },
      "classReport": {
        "required": ["labelId", "precision", "recall", "f1"],
        "properties": {
          "labelId": {"type": "number"},
          "precision": {"type": "number"},
          "recall": {"type": "number"},
          "f1": {"type": "number"},
          "precisionRecallCurve": { "$ref": "#/$defs/precisionRecallCurve" },
          "rocCurve": { "$ref": "#/$defs/rocRecallCurve" }
        }
      },
      "crossValidationReport": {
        "required": ["accuracy", "confusionMatrix", "classReports"],
        "properties": {
          "accuracy": { "type": "number" },
          "confusionMatrix": {"type": "array", "items": {"type": "array", "items": {"
              type": "number"}  } },
          "classReports": { "type": "array", "items": { "$ref": "#/$defs/classReport" }
              }
        }
      },
      "precisionRecallCurve": {
        "required": ["precisions", "recalls", "thresholds"],
        "properties": {
          "precisions": {"type": "array", "items": {"type": "number"}},
          "recalls": {"type": "array", "items": {"type": "number"}},
          "thresholds": {"type": "array", "items": {"type": "number"}}
        }
      },
      "rocRecallCurve": {
        "required": ["fpr", "tpr", "thresholds"],
        "properties": {
          "fpr": {"type": "array", "items": {"type": "number"}},
          "tpr": {"type": "array", "items": {"type": "number"}},
          "thresholds": {"type": "array", "items": {"type": "number"}}
        }
      }
  }
}
```

Listing A.5: build_result_schema.json

```
{
  "$id": "https://com.enrimiho.pdis/prediction.schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "description": "A representation of a classification prediction",
  "type": "object",
  "required": ["prediction", "labels"],
  "properties": {
    "prediction": { "type": "number" },
    "labels": { "type": "array", "items": { "type": "string" },
    "featureMap": {"type": "string"}
    }
  }
}
```

Listing A.6: prediction_result_schema.json

```
{
  "$id": "https://com.enrimiho.pdis/model_build_schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "description": "A representation of model build",
  "type": "object",
  "required": ["id", "model", "buildTimestamp", "state"],
  "properties": {
    "id": { "type": "number" },
    "buildTimestamp": { "type": "number" },
```

```
    "state": { "type": "string", "enum": ["FINISHED_SUCCESS", "FINISHED_FAILED", "
        RUNNING"] },
    "model": { "$ref": "file:model_schema.json" },
    "result": { "$ref": "file:build_result_schema.json" }
  }
}
```

Listing A.7: model_build_schema.json

A.2 Code listings for implementing a new model

```
class LogisticRegressionModel:
    ...

    def fit(self, x_train, y_train):
        # fit model to the data

    def predict_file(self, file):
        # returns prediction

def build_logistic_regression_model(settings):
    # load data set from the settings
    data = load_data_set(settings)

    solver = settings[2]['value']
    model = LogisticRegressionModel(solver=solver, ...)

    run_cross_validation = settings[3]['value']
    cross_validation_reports = []
    # optionally perform cross-validation
    if run_cross_validation:
        cross_validation_reports = cross_validation(model, data.training, ...)

    # build model
    model.fit(data.training)

    # evaluation
    report = to_classification_report(model, data.test, ...)

    result = {
        "buildTime": ...,
        "evaluationTime": ...,
        "classificationReport": report,
        'crossValidationReports': cross_validation_reports,
        'labels': ["tomato_healthy", "tomato_late_blight"],
    }
    return result, model
```

Listing A.8: logistic_regression.py

```
{"modelTypes": [
    { "id": 1,
      "name": "Close-range images model type",
      "description": "Description of the model type"
    }
  ],
  "models": [
    { "id": 1,
      "modelTypeId": 1,
      "name": "Tomato late blight logistic regression model",
      "implementation": "close_range_images.logistic_regression.
          build_logistic_regression_model",
      "plant": "Tomato",
      "disease": "Late Blight",
      "settings": [
        {"id": 1, "type": "INPUT", "title": "Healthy Leaves Images Location", "value":
            "/media/user/Storage/ML/PlantVillage-Dataset/raw/color/Tomato___healthy"},
```

```
        {"id": 2, "type": "INPUT", "title": "Diseased Leaves Images Location", "value":
             "/media/user/Storage/ML/PlantVillage-Dataset/raw/color/
            Tomato___Late_blight"},
        {"id": 3, "type": "OPTIONS", "title": "Solver", "value": "lbfgs", "options": ["
            lbfgs", "sag", "saga"]},
        {"id": 4, "type": "OPTION", "title": "Run Cross-validation", "value": false }
      ]
    }
  ]
}
```

Listing A.9: example_config.json

```
POST http://localhost:8000/api/models/builds

BODY
{
  "id": 1,
  "modelTypeId": 1,
  "name": "Tomato late blight logistic regression model",
  "settings": [
    {
      "id": 1,
      "title": "Healthy Leaves Images Location",
      "type": "INPUT",
      "value": "/media/user/Storage/ML/PlantVillage-Dataset/raw/color/Tomato___healthy"
    },
    {
      "id": 2,
      "title": "Diseased Leaves Images Location",
      "type": "INPUT",
      "value": "/media/user/Storage/ML/PlantVillage-Dataset/raw/color/
          Tomato___Late_blight"
    },
    {
      "id": 3,
      "options": [
        "lbfgs",
        "sag",
        "saga"
      ],
      "title": "Solver",
      "type": "OPTIONS",
      "value": "lbfgs"
    },
    {
      "id": 4,
      "title": "Run Cross-validation",
      "type": "OPTION",
      "value": false
    }
  ]
}

RESPONSE
{
  "id": 1,
  "model": {
    "id": 1,
    "modelTypeId": 1,
    "name": "Tomato late blight logistic regression model",
    "settings": [
      {
        "id": 1,
        "title": "Healthy Leaves Images Location",
        "type": "INPUT",
        "value": "/media/user/Storage/ML/PlantVillage-Dataset/raw/color/
            Tomato___healthy"
      },
      {
        "id": 2,
        "title": "Diseased Leaves Images Location",
```

```
        "type": "INPUT",
        "value": "/media/user/Storage/ML/PlantVillage-Dataset/raw/color/
            Tomato____Late_blight"
      },
      {
        "id": 3,
        "options": [
          "lbfgs",
          "sag",
          "saga"
        ],
        "title": "Solver",
        "type": "OPTIONS",
        "value": "lbfgs"
      },
      {
        "id": 4,
        "title": "Run Cross-validation",
        "type": "OPTION",
        "value": false
      }
    ]
  },
  "buildTimestamp": 1619197878167,
  "state": "RUNNING"
}
```

Listing A.10: Example request to build a new model.

```
GET http://localhost:8000/api/models/builds/1

RESPONSE
{
  "id": 1,
  "model": {
    "id": 1,
    "modelTypeId": 1,
    "name": "Logistic Regression",
    "settings": [
      {
        "id": 1,
        "title": "Healthy Leaves Images Location",
        "type": "INPUT",
        "value": "/media/user/Storage/ML/PlantVillage-Dataset/raw/color/
            Tomato____healthy"
      },
      {
        "id": 2,
        "title": "Diseased Leaves Images Location",
        "type": "INPUT",
        "value": "/media/user/Storage/ML/PlantVillage-Dataset/raw/color/
            Tomato____Late_blight"
      },
      {
        "id": 3,
        "options": [
          "lbfgs",
          "sag",
          "saga"
        ],
        "title": "Solver",
        "type": "OPTIONS",
        "value": "lbfgs"
      },
      {
        "id": 4,
        "title": "Run Cross-validation",
        "type": "OPTION",
        "value": false
      }
    ]
  },
```

```
  "buildTimestamp": 1619197878167,
  "state": "FINISHED_SUCCESS",
  "result": {
    "buildTime": 270.97,
    "evaluationTime": 3.21,
    "classificationReport": {
      "accuracy": 0.9175,
      "classReports": [
        {
          "labelId": 0,
          "precision": 0.9,
          "recall": 0.9,
          "f1": 0.91
        },
        {
          "labelId": 1,
          "precision": 0.94,
          "recall": 0.94,
          "f1": 0.92
        }
      ]
    },
    "crossValidationReports": [],
    "labels": [
      "tomato_healthy",
      "tomato_late_blight"
    ]
  }
}
```


Listing A.11: Example of getting a finished model build.

```
POST http://localhost:8000/api/models/builds/1/prediction

BODY
--------WebKitFormBoundaryQju2spf0GON5GuyG
Content-Disposition: form-data; name="json"; filename="blob"
Content-Type: application/json

{"name":"fce9cc3f-88ab-46d6-a26f-c6ff0e7c56e7____RS_Late.B 6372.JPG","mimeType":"image/
    jpeg"}
--------WebKitFormBoundaryQju2spf0GON5GuyG
Content-Disposition: form-data; name="attachment"; filename="fce9cc3f-88ab-46d6-a26f-
    c6ff0e7c56e7____RS_Late.B 6372.JPG"
Content-Type: image/jpeg


--------WebKitFormBoundaryQju2spf0GON5GuyG--

RESPONSE
{
  "prediction": 1,
  "labels": [
    "tomato_healthy",
    "tomato_late_blight"
  ]
}
```

Listing A.12: Example of classifying a file object.

A.3 Endpoint descriptions

Endpoint	Type	Parameters	Body	Response
/config	GET			Returns the configuration(Listing A.1)
/model-builds	POST	modelId, modelTypeId, callback	settings (Listing A.4)	Builds a new model with the given settings and sends the serialized model, including the build and evaluation results, back to the caller, using the callback endpoint. For a structure of the callback endpoint, see the description of it in Table A.2
/model-builds/prediction	POST		model file, file object	Classifies a file object using the model file. Invokes `predict_file` on the model and returns the prediction(Listing A.6)

Table A.1: Description of the endpoints, exposed by the models module.

Endpoint	Type	Parameters	Body	Response
/model-types	GET			Returns a list of model types, loaded from the configuration(Listing A.2)
/models	GET	modelTypeId		Returns a list of models, loaded from the configuration (Listing A.3), filtered by modelTypeId, if available
/models/builds	POST		model (Listing A.3)	Returns a new model build (Listing A.7)
/models/builds	GET			Returns a list of model builds(Listing A.7)
/models/builds/{id}	POST		build result, model file	Updates the model build for the given id with the provided build result (Listing A.5) and persists the model file. This is the callback endpoint that is invoked by the models module
/models/builds/{id}	GET			Returns the model build for the given id(Listing A.7)
/models/builds/{id}/prediction	POST		file object	Loads the model build with the given id and returns the prediction result for a file object(Listing A.6)

Table A.2: Description of the endpoints, exposed by the core module.

APPENDIX B

Supplementary materials: source code of PDIS, models, and data sets

The file `pdis.zip` contains the source code of PDIS, including the data sets. Following data sets are available:

- the tomato late blight data set, available under `/datasets/tomato-late-blight-dataset`. It contains healthy images of tomato leaves and images of leaves that are infected with tomato late blight disease, from the PlantVillage data set[HS15]
- the corn northern leaf blight data set, available under `/datasets/corn-northern-leaf-blight-dataset`. It contains healthy images of corn leaves and images of leaves that are infected with corn northern leaf blight disease, from the PlantVillage data set[HS15]
- the cucumber thermal leaf images data set, available under `/datasets/cucumber-thermal-leaf-images-dataset`. Wang at al.[WLD+12] published six thermographic images of healthy cucumber leaves, and also another six images, which show cucumber leaves that are infected with Fusarium wilt. These images are part of this data set
- the rice remote sensing data set, available under `/datasets/rice-remote-sensing-dataset`. It contains images of a rice cultivars. They are extracted from 2 images of rice fields, provided by Zhang et al.[ZZZ+18]. Images show rice cultivars with a low, medium and high risk of sheath blight

- the wheat weather data set, available under `/datasets/wheat-weather-dataset`. Contains weather data and assessments regarding wheat stripe rust disease. The data is provided by Jarroudi et al.[JLK+20]
- the whiteflies data set, available under `/datasets/whiteflies-dataset`. It contains images of healthy leaves, randomly selected from the PlantVillage data set[HS15]. Images of whiteflies on leaves are manually collected from the internet

The following model implementations can be found:

- `models/src/models/tomato_late_blight_lr_model.py`
- `models/src/models/tomato_late_blight_svm_model.py`
- `models/src/models/tomato_late_blight_cnn_model.py`
- `models/src/models/wheat_stripe_rust_lr_model.py`
- `models/src/models/corn_northern_leaf_blight_lr_model.py`
- `models/src/models/corn_northern_leaf_blight_svm_model.py`
- `models/src/models/corn_northern_leaf_blight_cnn_model.py`
- `models/src/models/cucumber_fusarium_wilt_lr_model.py`
- `models/src/models/cucumber_fusarium_wilt_cnn_model.py`
- `models/src/models/rice_sheath_blight_cnn_model.py`
- `models/src/models/whiteflies_cnn_model.py`

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 2–5, 7, 9, 19–23, 25, 27, 31, 32, 35–38, 51, 54, 59, 63, 65, 66, 71, 77, 81, 83–86, 88, 89, 91–93, 109, 110

API Application Programming Interface. 61, 64, 91

CPU central processing unit. 66, 84

FN false negatives. 61

FP false positives. 61

GAN Generative Adversarial Network, a framework that can be used for data augmentation. 39, 40, 46–48

GPS Global Positioning System. 8

GPU graphics processing unit. 66

HSI hue, saturation, intensity - an alternative color model to RGB. 33

HSL hue, saturation, lightness - an alternative color model to RGB. 53, 54

HSV hue, saturation, value - an alternative color model to RGB. 33–35, 57

HTTP Hypertext Transfer Protocol. 64

IoT Internet of Things. 3

JSON JavaScript Object Notation. 64

NDVI Normalized Difference Vegetation Index. 85, 86

PDIS Plant Disease Identification System. 63, 64, 93, 105, 107

RAM random-access memory. 66

RGB red, green, blue - an additive color model. 17, 27, 28, 33–35, 42, 43, 48, 51, 53, 56, 57, 85, 111, 112

ROC Receiver operating characteristic. 62, 67, 72–74

SSD solid-state drive. 66

TN true negatives. 61

TP true positives. 61

YIQ an alternative color model to RGB, Y represents the perceived luminance and I and Q the luminance information. Is is used by the NTSC color TV system. 33, 35

Bibliography

[Alp14] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2014.

[Bar13] Jayme Barbedo. Automatic method for counting and measuring whiteflies in soybean leaves using digital image processing. 10 2013.

[Bar20] Jayme Barbedo. Detecting and classifying pests in crops using proximal images and machine learning: A review. *AI*, 1:312–328, 06 2020.

[BB07] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. volume 20, 01 2007.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[BM13] T.S. Bodhe and P. Mukherji. Selection of color space for image segmentation in pest detection. pages 1–7, 01 2013.

[Cra02] J.S. Cramer. The origins of logistic regression. *Tinbergen Institute, Tinbergen Institute Discussion Papers*, 01 2002.

[CS09] Anyela Camargo and Jeremy Smith. An image-processing based algorithm to automatically identify plant disease visual symptoms. *Biosystems Engineering - BIOSYST ENG*, 102:9–21, 01 2009.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, sep 1995.

[DBLJ14] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives, 2014.

[ESJT17] Mosbah El Sghair, Raka Jovanovic, and Milan Tuba. An algorithm for plant diseases detection based on color features. *International Journal of Agricultural Science*, 2, 2017.

[FNJ19] Per Frankelius, Charlotte Norrman, and Knut Johansen. Agricultural innovation and the role of institutions: Lessons from the game of drones. *Journal of Agricultural and Environmental Ethics*, 32:681–707, 11 2019.

[Fre05] David Freedman. *Statistical Models : Theory and Practice*, page 26. Cambridge University Press, August 2005.

[FRJ86] J. Fussell, Donald Rundquist, and John Jr. On defining remote sensing. 52:1507–1511, 01 1986.

[GBC+10] H. Charles J. Godfray, John R. Beddington, Ian R. Crute, Lawrence Haddad, David Lawrence, James F. Muir, Jules Pretty, Sherman Robinson, Sandy M. Thomas, and Camilla Toulmin. Food security: The challenge of feeding 9 billion people. *Science*, 327(5967):812–818, 2010.

[GPAM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[Gre14] Prof Grewal. A critical conceptual analysis of definitions of artificial intelligence as applicable to computer engineering. *IOSR Journal of Computer Engineering*, 16:09–13, 01 2014.

[GVG+17] Nathalie Gorretta, Pierre Vaysse, Michel Giraud, Christian Germain, Barna Keresztes, and Jean-Michel Roger. Near infrared hyperspectral dataset of healthy and infected apple tree leaves images for the early detection of apple scab disease. *Data in Brief*, 16, 12 2017.

[HK00] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42:80 – 86, 2000.

[HS15] David Hughes and Marcel Salathe. An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing. 11 2015.

[IJM+20] S. Iniyan, R. Jebakumar, P. Mangalraj, Mayank Mohit, and Aroop Nanda. Plant disease identification and detection using support vector machines and artificial neural networks. In Subhransu Sekhar Dash, C. Lakshmi, Swagatam Das, and Bijaya Ketan Panigrahi, editors, *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, pages 15–27, Singapore, 2020. Springer Singapore.

[Isl18] Waqar Islam. Plant disease epidemiology: Disease triangle and forecasting mechanisms in highlights. 5, 02 2018.

[JAT17] Sarah Jasim and Ali Al-Taei. A comparison between svm and k-nn for classification of plant diseases. *Diyala Journal for Pure Science*, 01 2017.

[JLK+20] Moussa Jarroudi, Rachid Lahlali, Louis Kouadio, Antoine Denis, Alexandre Belleflamme, Mustapha El Jarroudi, Mohammed Boulif, H. Mahyou, and Bernard Tychon. Weather-based predictive modeling of wheat stripe rust infection in morocco. 02 2020.

[Joh07] D. Johnson. Proximity sensors. 54:92–97, 07 2007.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[KHI+19] Ahmed Khattab, Serag Habib, Haythem Ismail, Sahar Zayan, Yasmine Fahmy, and M.M. Khairy. An iot-based cognitive monitoring system for early plant disease forecast. *Computers and Electronics in Agriculture*, 166:105028, 11 2019.

[KLM96] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *J. Artif. Int. Res.*, 4(1):237–285, may 1996.

[KPG18] Sukhvir Kaur, Shreelekha Pandey, and Shivani Goel. Plants disease identification and classification through leaf images: A survey. *Archives of Computational Methods in Engineering*, 26, 01 2018.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBH15] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[Mah15] Anne-Katrin Mahlein. Plant disease detection by imaging sensors – parallels and specific demands for precision agriculture and plant phenotyping. *Plant Disease*, 100, 09 2015.

[MGI14] Johnny Miranda, Bobby Gerardo, and Bartolome Iii. Pest detection and extraction using image processing techniques. *International Journal of Computer and Communication Engineering*, 3:189–192, 01 2014.

[MSB+17] Mohammadmehdi Maharlooei, Saravanan Sivarajan, Sreekala Bajwa, Jason Harmon, and John Nowatzki. Detection of soybean aphids in a greenhouse using an image processing technique. *Computers and Electronics in Agriculture*, 132:63–70, 01 2017.

[NAAZ20] Lawrence Ngugi, Moataz Abelwahab, and M. Abo-Zahhad. Recent advances in image processing techniques for automated leaf pest and disease recognition - a review. *Information Processing in Agriculture*, 04 2020.

[Neg01] Michael Negnevitsky. *Artificial Intelligence: A Guide to Intelligent Systems*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 2001.

[NJS+19] Koushik Nagasubramanian, Sarah Jones, Asheesh Singh, Soumik Sarkar, Arti Singh, and Baskar Ganapathysubramanian. Plant disease identification using explainable 3d deep learning on hyperspectral images. *Plant Methods*, 15, 12 2019.

[NS20] Harshita Nagar and R.S. Sharma. A comprehensive survey on pest detection techniques using image processing. pages 43–48, 05 2020.

[OSDL06] E-C Oerke, U Steiner, H.-W Dehne, and M Lindenthal. Thermal imaging of cucumber leaves affected by downy mildew and environmental conditions. *Journal of experimental botany*, 57:2121–32, 02 2006.

[OSET18] Dor Oppenheim, Guy Shani, Orly Erlich, and Leah Tsror. Using deep learning for image-based potato tuber disease detection. *Phytopathology*, 109, 12 2018.

[PCB20] Leidy Pamplona, Andres Calvo, and Arley Bejarano. Detection of foliar diseases using image processing techniques. *Revista Ceres*, 67:100–110, 04 2020.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[PMS20] P. Prathusha, K. E. Srinivasa Murthy, and K. Srinivas. Plant disease detection using machine learning algorithms. In S. Jyothi, D. M. Mamatha, Suresh Chandra Satapathy, K. Srujan Raju, and Margarita N. Favorskaya, editors, *Advances in Computational and Bio-Engineering*, pages 213–220, Cham, 2020. Springer International Publishing.

[PP20] Gayatri Pattnaik and K. Parvathi. *A Review on Advanced Techniques on Plant Pest Detection and Classification*, pages 665–673. 01 2020.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[RAPSN20] Jonathan Rocha, Marcelo Alves, Edson Pozza, and Helon Santos Neto. Detection of coffee berry necrosis by digital image processing of landsat 8 oli satellite imagery. *International Journal of Applied Earth Observation and Geoinformation*, 85:101983, 03 2020.

[SA13] R. Sathya and Annamma Abraham. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, 2, 02 2013.

[SAAB20] Muhammad Saleem, Babar Atta, Zulfiqar Ali, and Muhammad Bilal. Laser induced fluorescence spectroscopy for early disease detection in grapefruit plants. *Photochemical & Photobiological Sciences*, 19, 04 2020.

[Sam59] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[SCZZ19] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. A survey of optimization methods from a machine learning perspective, 2019.

[SK19] Gurleen Sandhu and Rajbir Kaur. Plant disease detection techniques: A review. pages 34–38, 04 2019.

[SMS20] Vikas Sharma, Aftab Mir, and Dr Sarwr. Detection of rice disease using bayes' classifier and minimum distance classifier. *Journal of Multimedia Information System*, 7:17–24, 03 2020.

[SPDWM19] Denis Shah, Premila Paul, Erick De Wolf, and Laurence Madden. Predicting plant disease epidemics from functionally represented weather series. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 374:20180273, 06 2019.

[Spe14] Sandro Sperandei. Understanding logistic regression analysis. *Biochemia medica*, 24:12–8, 02 2014.

[SRB13] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient, 2013.

[Ste97] Stephen Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62:77–89, 10 1997.

[SVN20] K. S. Sankaran, N. Vasudevan, and V. Nagarajan. Plant disease detection and recognition using k means clustering. In *2020 International Conference on Communication and Signal Processing (ICCSP)*, pages 1406–1409, 2020.

[SVR+19] GOMEZ SM, Alejandro Vergara, Henry Ruiz, Nancy Safari, Sivalingam Elayabalan, Walter Ocimati, and Guy Blomme. Ai-powered banana diseases and pest detection. *Plant Methods*, 15, 12 2019.

[Swe86] John A. Swets. Indices of discrimination or diagnostic accuracy: their rocs and implied models. *Psychological bulletin*, 99 1:100–17, 1986.

[TGL+13] Indi Trehan, Hayley Goldbach, Lacey LaGrone, Guthrie Meuli, Richard Wang, Kenneth Maleta, and Mark Manary. Antibiotics as part of the management of severe acute malnutrition. *The New England journal of medicine*, 368:425–35, 01 2013.

[tH19] Muammer turkoglu and Davut HANBAY. Plant disease and pest detection using deep learning-based features. *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES*, 27:1636–1651, 05 2019.

[VA19] Aravindhan Venkataramanan and Pooja Agarwal. Plant disease detection and classification using deep neural networks. 08 2019.

[VNF+20] Ahmad Virk, Mehmood Ali Noor, Sajid Fiaz, Saddam Hussain, Hafiz Hussain, Muzammal Rehman, Muhammad Ahsan, and Wei Ma. *Smart Farming: An Overview*, pages 191–201. 02 2020.

[VTS04] JP. Vert, K. Tsuda, and B. Schölkopf. *A Primer on Kernel Methods*, pages 35–70. MIT Press, Cambridge, MA, USA, 2004.

[WLD+12] Min Wang, Ning Ling, Xian Dong, Yiyong Zhu, Qirong Shen, and Shiwei Guo. Thermographic visualization of leaf response in cucumber plants infected with the soil-borne pathogen fusarium oxysporum f. sp cucumerinum. *Plant physiology and biochemistry : PPB / Societe francaise de physiologie vegetale*, 61, 10 2012.

[WZL+19] Xiaoping Wu, Chi Zhan, Yu-Kun Lai, Ming-Ming Cheng, and Jufeng Yang. Ip102: A large-scale benchmark dataset for insect pest recognition. pages 8779–8788, 06 2019.

[YYW+19] Ning Yang, Minfeng Yuan, Pan Wang, Rongbiao Zhang, Jun Sun, and Hanping Mao. Tea diseases detection based on fast infrared thermal image processing technology. *Journal of the Science of Food and Agriculture*, 99, 01 2019.

[ZBLN97] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained https://www.overleaf.com/project/60883f2b0845848c36eb8171optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, dec 1997.

[ZZZ+18] Dongyan Zhang, Xingen Zhou, Jian Zhang, Yubin Lan, Chao Xu, and Dong Liang. Detection of rice sheath blight using an unmanned aerial system with high-resolution color and multispectral imaging. *PLOS ONE*, 13:e0187470, 05 2018.