UNIVERSITY OF THE PELOPONNESE & NCSR "DEMOCRITOS"
MSC PROGRAMME IN DATA SCIENCE

AI-based Plant Pathology

Kyriakos Zorbas

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science

Supervisor: Theodoros Giannakopoulos
Research Associate, NCSR Demokritos

Athens, April 2021

AI-based Plant Pathology

Kyriakos Zorbas

MSc. Thesis, MSc. Programme in Data Science

University of the Peloponnese & NCSR “Democritos”, April 2021

UNIVERSITY OF THE PELOPONNESE & NCSR "DEMOCRITOS" MSC
PROGRAMME IN DATA SCIENCE

AI-based Plant Pathology

Kyriakos Zorbas

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science

Supervisor: Theodoros Giannakopoulos
Principal Researcher of Multimodal Machine Learning

Approved by the examination committee in April, 2021.

Theodoros Giannakopoulos	Anastasia Krithara	Christos Trifonopoulos
Research Associate, NCSR Demokritos	Research Associate, NCSR Democritos	Assistant Professor, Department of Informatics and Telecommunications, University of Peloponnese.

Athens, April 2021

UNIVERSITY OF THE PELOPONNESE & NCSR "DEMOCRITOS" MSC PROGRAMME IN DATA SCIENCE

Declaration of Authorship

(1) I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

(2) I confirm that this thesis presented for the degree of Bachelor of Science in Informatics and Telecommunications, has

   (i) been composed entirely by myself

   (ii) been solely the result of my own work

   (iii) not been submitted for any other degree or professional qualification

(3) I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

. . . . . .

Kyriakos Zorbas

Acknowledgment

First of all, I would like to thank Theodoros Giannakopoulos who was the supervisor of my work and gave me the opportunity to prepare my thesis on a very interesting subject expanding my knowledge. His guidance was crucial to the completion of my thesis work. At the same time, I would like to thank the teachers Mrs. Anastasia Krithara and Mr. Christos Tryfonopoulos for their participation in the three-member examination committee. Finally, I would like to thank my parents in particular for their help and support but also all my friends who were by my side during my student years helping everyone in their own way.

Kyriakos Zorbas

Athens, April 2020

Περίληψη

Ο συνδυασμός της συνεχούς αύξησης του πληθυσμού και της σταδιακής ερήμωσης των περιοχών της Γης λόγω της κλιματικής αλλαγής θα οδηγήσει στην αδυναμία παροχής τροφής σε όλους τους ανθρώπους στο μέλλον, οπότε για να βοηθήσουμε την παραγωγή τροφίμων να αυξηθεί πρέπει να βρούμε λύσεις. Μία λύση είναι να βοηθήσουμε τους αγρότες να εντοπίσουν πολύ γρήγορα εάν η σοδειά τους είναι άρρωστη για να αναλάβουν δράση πριν να είναι πολύ αργά. Λαμβάνοντας υπόψη αυτό αυτή η διατριβή θα μελετήσει τη βιβλιογραφία και θα αναπτύξει μια προσέγγιση που βασίζεται σε Τεχνητή Νοημοσύνη για την αναγνώριση ανωμαλιών στον φύλλωμα από διαφορετικές ασθένειες. Ο στόχος είναι να βοηθήσουμε τον άνθρωπο για το εάν θα ψεκάσει ένα φυτό για μια συγκεκριμένη ασθένεια. Δεδομένου ότι υπάρχει περιορισμένος αριθμός δεδομένων, οποιαδήποτε λύση δεν μπορεί να βασιστεί αποκλειστικά σε βαθιές αρχιτεκτονικές που χρειάζονται σχετικά τεράστια σύνολα δεδομένων εκπαίδευσης. Αντ 'αυτού, η διατριβή θα εξετάσει τη χρήση μιας προσέγγισης πολλαπλών μοντέλων, όπου το καθένα χρησιμοποιώντας διαφορετικές μεθοδολογίες μηχανικής μάθησης, όπως η μεταφορά μάθησης που χρησιμοποιεί γνώσεις που αποκτήθηκαν από άλλους τομείς. Ειδικότερα, το πρώτο κομμάτι θα είναι ένας αλγόριθμος ανίχνευσης αντικειμένων το οποίο αναγνωρίζει εάν υπάρχει ένα "φύλλο" μέσα σε μια εικόνα και ως δεύτερο κομμάτι θα είναι ένας αλγόριθμος που αποφασίζει για κάθε φύλλο που βρέθηκε από το προηγούμενο κομματι, εάν είναι υγιές ή άρρωστο και αν το φύλλο βρέθηκε ότι είναι άρρωστο, το κατηγοριοποιεί μεταξύ τριών κοινών ασθενειών των φύλλων. Για λόγους απλότητας, το πρώτο κομμάτι θα αναφέρεται ως CP1 και το δεύτερο ως CP2.

Abstract

The combination of continuous population growth and the gradual desertification of areas of the Earth due to climate change will lead to the inability to provide food for all people in the future, so in order to help the food production to increase we have to find solutions. One solution is to help farmers to detect very fast if their crop is diseased in order to take action before it's too late.Given that in mind this thesis will study the literature and develop an AI-based approach for recognizing anomalies in the canopy, resulting from different diseases. The goal is to support actuation decisions e.g. whether to spray a plant for a given disease. Given that there is a limited amount of properly annotated ground truth data, any solution cannot rely solely on deep architectures that need relatively huge training datasets. Instead the thesis will examine the use of a multi-model approach, where a set of components, each using different machine learning methodologies such as transfer learning, uses knowledge gained from other domains. More specifically, the first component will be an object detection algorithm which recognizes if there is a “leaf” inside an image and as second component will be an algorithm that decides for every leaf, found from the previous component, if it is healthy or diseased and if the leaf found to be diseased it categorizes it among three common leaf diseases. For simplicity reasons the first component will be referred to as CP1 and the second as CP2.

Contents

List of Tables

List of Figures

List of Abbreviations

CP1	First Component of thesis's Architecture
CP2	Second Component of thesis's Architecture
CNN	Convolutional Neural Network
ResNet	Residual Network
AI	Artificial Intelligence
ANN	Artificial Neural Network
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
R-CNN	Region Based Convolutional Neural Network
SVM	Support Vector Machines
GPU	Graphics Processing Unit
CPU	Central Processing Unit
MSE	Mean Squared Error
MAE	Mean Absolute Error
COCO	Common Objects in Context
SGD	Stochastic Gradient Descent
ROC	Receiver Operating Characteristic
UI	User Interface
API	Application Programming Interface
JS	JavaScript
ReLU	Rectified Linear Unit
LSTM	Long Short-Term Memory

RNN	Recursive Neural Networks

Chapter 1

Introduction

1.1 Motivation & Contribution

Ever since man began to establish the first permanent settlements in the Neolithic era, agriculture has been the main source of food for the ever-increasing human population. According to the International Food and Agriculture Organization [1], 90% of the world's population could not survive without the agricultural products produced by the agriculture, livestock and fisheries sectors, most importantly agriculture [2]. Food production, however, is confronted with current issues such as overpopulation [3] and climate change [4]. With the increase of the population, the need for greater food production increases, which can be achieved with more modern ways of production and care of the fruits but also with the increase of the arable land. Most of the arable land is already used for crops and finding additional areas involves deforestation and other natural areas with negative effects on the environment. According to the International Union for Conservation of Nature, more than half of the world's rainforests have been deforested since 1960, and today more than one hectare of rainforest is being destroyed or severely degraded every second [5]. Unfortunately, not only is the expansion of arable land not easily feasible and sustainable, but, in addition, climate change and soil pollution are gradually reducing the number of existing ones, as there is a gradual desertification of areas previously used for agricultural purposes [6]. In addition, rising temperatures change the suitability of an area for growing certain fruits and lead to a reduction in crop production, a phenomenon that Chuang Zhao examines for the 4 most common fruits in agriculture (wheat, rice, corn, soybeans)[7]. The above makes it necessary to continuously and extensively monitor and record globally grown fruits, in order to facilitate the management of food production, decision-making and the definition of future strategies.

The goal of this thesis is to support farmers whether to spray a plant for a given disease. In order to achieve this we will examine the use of a multi-model approach, where a set of components, each using different machine learning methodologies such as transfer learning, uses knowledge gained from other domains. More specifically, the first component will be an object detection algorithm which recognizes if there is a “leaf” inside an image and as second component will be an algorithm that decides for every leaf, found from the previous component, if it is healthy or diseased and if the leaf found to be diseased it categorizes it among three common leaf diseases. For simplicity reasons the first component will be referred to as CP1 and the second as CP2. The following image describes the flow of our components.

Additionally we created a web application as a proof of concept that our presented solution of the above described problem is working.Lastly it is important to say that everything in this thesis like the used datasets and the base codes , are open and shareable in order for anyone to check the validity of the results and to contribute furthermore if wishes.

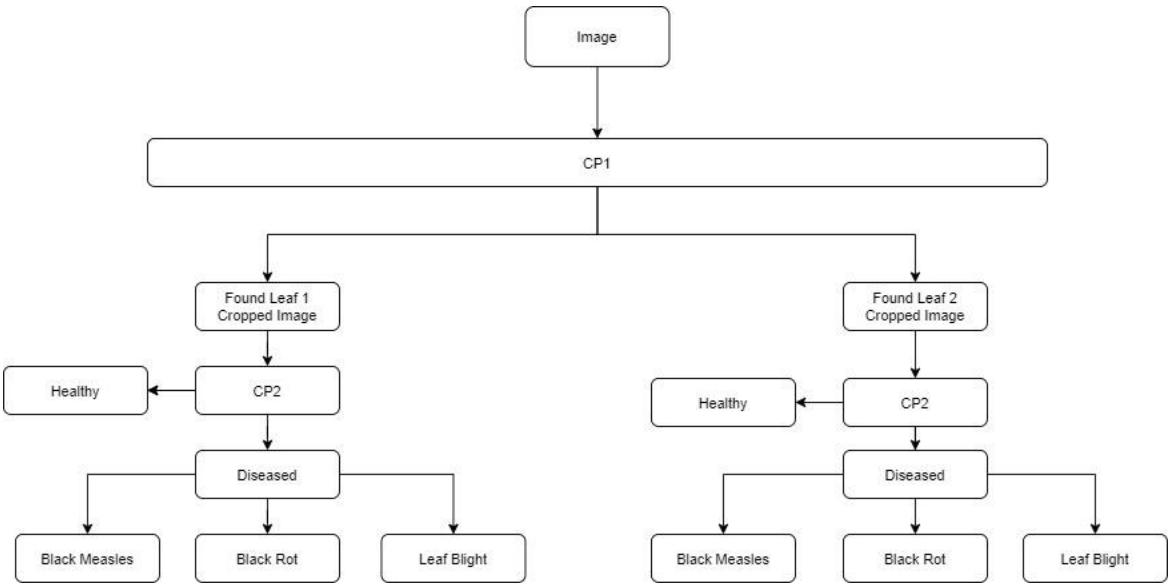


Figure 1: Thesis Architecture Workflow

1.2 Thesis Organization

This thesis is organized in 8 chapters.

Chapter 1: A brief reference is made to the motivations and objectives of this thesis.

Chapter 2: An introduction in Data Science and Machine Learning.

Chapter 3: A presentation of Neural Networks Principles.

Chapter 4: A brief description of Convolutional Object Detection & Classification which are the core of CP1 and CP2 accordingly.

Chapter 5: An overview of the datasets that are used in this thesis.

Chapter 6: Info about the implementation of CP1 and CP2 and about the created web application. It also contains a short guide for the deployment of the created components.

Chapter 7: A description about the methodology of CP1 and CP2 among the presentation of the experiments that took place with their results.

Chapter 8: A concise overview of the results and some thoughts about future work.

Chapter 2

Machine Learning & Neural Networks Principles

2.1 What is Machine Learning ?

Machine Learning is a subfield of Artificial Intelligence, which deals with the construction of models that implement specific algorithms that use experimental data to make useful predictions; or conclusions. These models must have the ability to improve their performance when they receive additional data without having to be programmed from the beginning. Mitchell [8] has given the following definition to machine learning:"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

The training process that is followed is to feed the model with data, usually numerical. Then, vectors are created from this data, the so-called Feature Vectors with which appropriate algorithms are fed. Finally, metrics are used to calculate the reliability of each model. Machine learning algorithms are divided in 3 big categories:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

2.2 Machine Learning Categories

The separation of the categories of machine learning arises from the way in which the system interacts with its environment during training, in other words. the way in which feedback is given during the learning.

2.2.1 Supervised Learning

In Supervised Learning the purpose is to approach a function that connects input to output using existing input-output value pairs (data-tags)[9]. The set of these pairs is called the set of training and the process of calculating such a function from the above set is called training. The ultimate goal of supervised learning is the calculation of a function that sufficiently generalizes to the input data so that it is able to assign to the correct outputs new input data which it had not used during training.

2.2.2 Unsupervised Learning

Unlike the previous category of machine learning, unsupervised learning does not use input-output value pairs. Its purpose is to identify patterns in the input without feedback from output values [10]. A typical example of unsupervised learning is the problem of clustering. This method aims grouping input data by creating groups whose members are present similar to each other, but differ significantly from members of other groups.

2.2.3 Reinforcement Learning

Reinforcement learning differs from the other categories because of a technique that uses "rewards" and "punishments" from the environment and the model interactions with it [11]. The choices made to lead to an outcome are judged whether they had a positive or negative contribution to that outcome. In this way the model can choose whether to make the same choices again or to follow a different path of action, which may differ in one or more choices [12].

2.3 Neural Networks

Neural Networks are characterized by their architecture, the site function they perform and their training method. The network architecture determines the arrangement of neuron connections as well as the number and type of neurons. Neurons are organized in the form of layers [13]. If there are several layers in the network then we refer to it as deep learning. Deep learning is part of a larger family of approaches to machine learning based on artificial neural networks with representational learning (also known as or hierarchical learning or deep structured learning) [14][15]. In fields such as computer vision, audio recognition, speech recognition, natural language processing, machine vision, social network filtering, medical image analysis, machine translation, drug design, bioinformatics, content inspection and inspection, deep-learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been implemented [16].

2.3.1 Artificial neural networks

Artificial Neural Networks is a supervised machine learning algorithm that has to simulate the function of the human brain. Their structural elements are the artificial neurons, which have been created based on the human neuron. In Figure 2 shows the architecture of an artificial neuron.

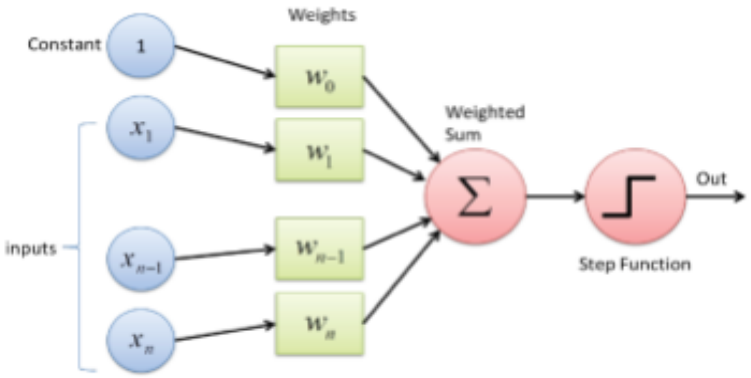


Figure 2: Artificial Neuron Architecture (source:towardsdatascience.com)

Xi are the inputs of the neuron, which are multiplied by the weights Wi and sum together with a polarity W0 (often found with the symbol b). The final sum is fed to an activation function f and depending on the sum value, the output is 0 or 1, or as it is otherwise called the neuron is deactivated or activated. The complete mathematical expression is as follows:

$$y = f(\sum_{i=1}^{n} x_i w_i + b)$$

In Figure 3 the activation function is the simple step. There are several more functions, the choice of which depends on the nature of the problem and its designer neural network. Given the binary output of the artificial neuron it is obvious that it has the ability, appropriately adjusting the wi weights to sort the input data into two classes if these are linearly separable [7].

The aforementioned artificial neuron is called Perceptron and Rosenblatt in 1958 was the first to propose him [18]. The combination of many Perceptrons can create a field of simple neurons, which functions as both input and output of the network [19], the so-called a single-level Artificial Neural Network. Every neuron in this network is independent of the others, so his learning is done independently of the others neurons. However, Minsky-Papert in 1969 demonstrated that the Artificial Neural Networks of a level are not capable of solving nonlinear problems [20], which they can achieve with the Multilevel Neural Networks.

2.3.2 Multilayer Neural Networks

In order to solve more complex problems with great complexity, it is necessary to combine many different neurons with each other, creating multilevel architectures Neurons Perceptrons and are called Multi-Layer Perceptrons (MLPs). In MLP the neurons are organized by levels which are divided into 3 categories:

- Input layer
- Hidden layer
- Output layer

In Figure 3 the input neurons are marked in yellow, the hidden neurons with blue and the output neurons with green. Every artificial neuron in the network works like referred to in section 2.1.1. A feature of Artificial Neural Networks separation is the way it is connected between neurons of all levels. The resulting categories are:

- **Fully Connected:** networks in which each neuron of a level connects to all neurons of the next level.
- **Partially Connected:** networks that exist at each level neurons that are not connected to all the neurons of the next level.
- **Feedforward:** networks in which neural connections do not create connection circles. That is, no neuron promotes its output in neurons of previous levels.
- **Feedback:** networks that, unlike the previous category, have neurons that advance their output to previous level neurons.

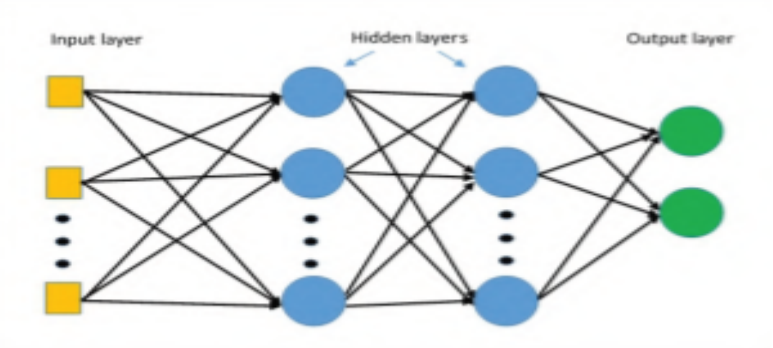


Figure 3: Multi-Layer Perceptron (source:deepai.org)

2.3.3 Convolutional Neural Networks

Convolutional neural networks (convnets or CNNs) are specialised feed-forward neural networks. Primarily used for computer vision. The key difference between the convolutional Artificial Neural Network (ANN) and the Convolution Neural Network (CNN) is that only the last layer of a CNN is completely connected, while in ANN, as seen in Figure 4, each neuron is connected to all other neurons [21][22].

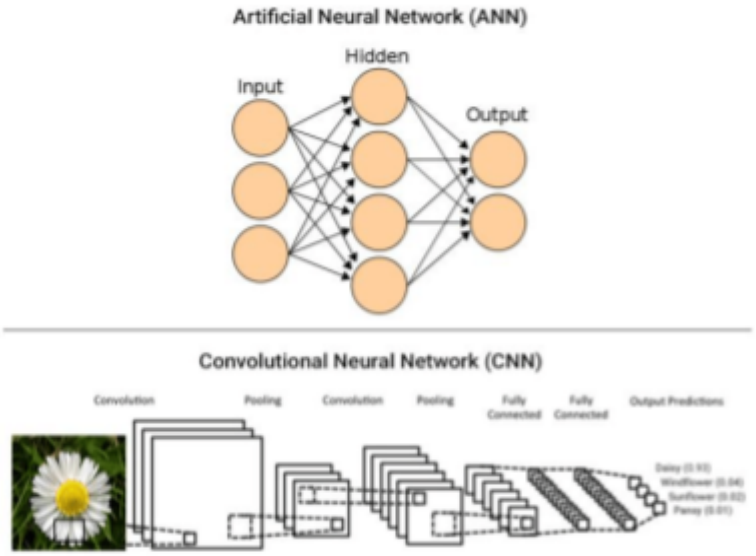


Figure 4: Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) (source:www.researchgate.net)

2.3.4 Activation Functions

The Activation Function is a function that takes the output as input from the sum of the neuron value and generates an output value that will be forwarded as input to the subsequent neurons. Its usefulness lies in the fact that the value of the sum in one neuron can be any number. Using the activation function We can transfer this value at any time it deems most convenient, so that it is easy to interpret the result. Essentially this function

works as a filter on produced prices. The following are some of the most common activation functions.

- **Sigmoid function:** This function converts its input into space (0, 1) without ever being able to get the values 0 and 1. The normalization of values in the period leads to the absence of significant value differences at the exit of the function. This phenomenon is called Vanishing Gradient. The graphics representation of the function is shown in Figure 5.

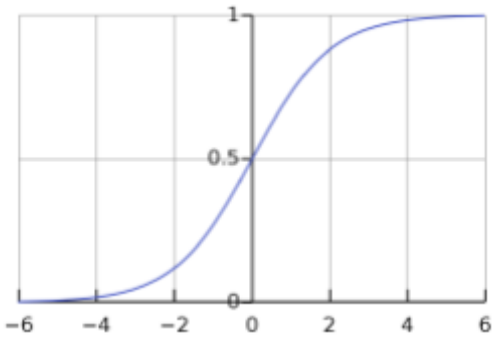


Figure 5: Sigmoid Function (source: wikipedia.org)

and its mathematical formula is as follows:

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

- **Hyperbolic Tangent Function:** This function is similar to the sigmoid with the difference that the values correspond to the interval (−1, 1).

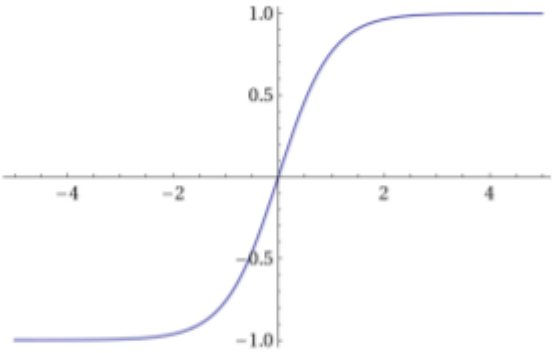


Figure 6: Hyperbolic Tangent Function (source: oreilly.com)

Its mathematical formula is as follows:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- **Rectified Linear Unit, ReLU:** This function is the most common activation function in Neural Network applications. It consists of a part zeroing of values for negative inputs, and a linear function for positive inputs. This function is ideal for problems which do not include large entry values.Also does not face the Vanishing Gradient problem that the two previous functions have. One of its negatives is inactivation of all neurons that have negative values, which can greatly affect the solution of the problem, depending of course on the nature of the problem itself. Its graph is as follows.

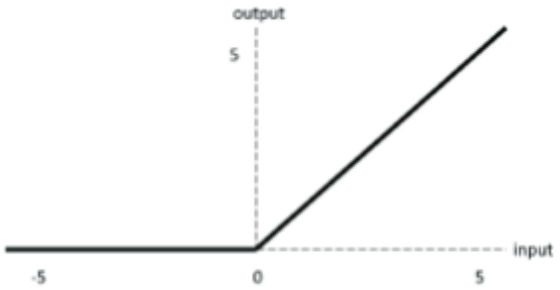


Figure 7: Rectified Linear Unit (source: researchgate.net)

Its mathematical formula is as follows:

$$f(x) = (0, \max)$$

2.3.5 Cost Functions

Cost Functions are used to control the performance of Neural Networks. The validation is made with data of the input samples and their expected output value. Thus, it is possible to monitor the improvement of the network and its adaptation to minimize the mistakes it makes. The comparison is made between values predicted by the model and the actual values. Some cost functions are below.

- **Mean Squared Error (MSE):** Calculate the average of squares of errors.The mathematical formula is as follows:

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Pi)^2$$

where Yi are the actual values and Pi are the forecasts.

- **Mean Absolute Error (MAE):** Calculates the average absolute value of errors.

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}|Y_i - Pi|$$

where Yi are the actual values and Pi are the forecasts.

- **Cross Entropy Loss:** It is the most common cost function in classification problems. Its function is to compare the probable distributions of the predicted values and of real prices. The more the two distributions diverge, the greater is its price, so the goal is to get as low prices as possible so the two distributions can be identified. Characteristic of this function is the fact that it imposes a large penalty on predictions with a high degree of confidence (confident predictions) but they are wrong. The mathematical formula is as follows:

$$J(\boldsymbol{\theta}) = H(p,q) = -\sum_{x \in X} p(x)\log(q(x))$$

where p, q are the probable distributions of the actual values and the predicted values respectively.

- **SVM Loss (Hinge loss):** Occurs in categorization problems and the purpose of the definition is that the sum of the correct predictions is greater than its sum wrong. As its name suggests, it is used in Vector Machines Support. It is characterized by the fact that it is not productive but it is very easy to calculate. The mathematical formula is as follows:

$$J(\boldsymbol{\theta}) = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Where θ is the vector of the parameters of each network.

2.3.6 Backpropagation Algorithm

This algorithm is a member of the Optimization algorithm category and is from the most famous. It was first proposed by Kelley in 1960 [23] and is widely used in the Learning also seen in the Feed Forward Neural Networks. The algorithm implements a small modification to each weight of the network, taking into account the error that occurs for a specific input, the corresponding desired output and the network recall. Its application is done in epochs, that is, running through a predetermined one way all input data each epoch. The network weights are adjusted accordingly how much they contributed to the overall network error. This adjustment is made following the opposite direction from the data flow.

The two main ways of presenting data on the network are the following:

- Incremental Training: In this case the input data are presented separately in the network and for each one the weights are modified.

- Batch Training: In this case everything is presented first input data once, changes in weights are calculated for each input and at the end the weights are updated.

The total error for all the examples is the sum of their squares errors of output neurons. An algorithm termination condition is also defined, which is either the total error falling below a specified threshold or the a predetermined number of training sessions at the beginning of the whole process.

2.3.7 Optimization Algorithms

Optimization Algorithms are used to maximize or minimize a function. In this case, this function is some cost function, which must be minimized. In Neural Networks the weights of the hidden layer have a very key role in the final performance, therefore it is necessary to continuously renew them until we reach the desired result. This is the purpose of the algorithms in this category. The following is the most widely used technique of Gradient Descent and some variants of it. This method aims to minimize the cost function using its first partial derivative. In each iteration of the algorithm it is subtracted from each parameter of the network the partial derivative multiplied by a numeric parameter called learning rate. The mathematical formula is as follows:

$$\theta_{j,t+1} = \theta j, t - \alpha \frac{\partial}{\partial \theta j, t} J(\boldsymbol{\theta})$$

In the general case the gradient descent algorithm is executed after a feed forward has been made for all the data in the network.Often, however, part of the data set is fed to the network before an update is made.Based on this criterion we distinguish the following basic cases:

- **Batch Gradient Descent:** All input samples are fed to the grid and synchronously The parameters are updated. In this case, it may be unnecessary calculations, as

slopes are often calculated for similar examples before each parameter update. Of course, it is certain that it will find the total Most of the Cost Function if used on a convex error surface. In cases where the input data is too much, problems are created in calculation due to computational resources.

- **Stochastic Gradient Descent-SGD:** In contrast to the previous case here the network parameters are updated for each input sample separately. In the approach unnecessary calculations do not happen, but problems arise when approaching at a local minimum of the curve and especially when the slopes of the minimum are steep, resulting in oscillation on the slopes of this area. It is also sensitive to noisy input samples as the network parameters are updated based on "bad" input samples delay the process of finding the minimum. This method was proposed by Robbins in 1951 [24].

- **Mini-Batch Gradient Descent:** This method is characterized by random sample selection but in larger groups. It tries to combine randomness and fast calculation of the stochastic case using more input data so that it reduces noise from individual noisy input samples.

2.6 Evaluation Metrics

Evaluation Metrics are very important for drawing conclusions in order to be able to compare different models with each other in terms of effectiveness there. There is no strict rule for the use of each metric as to how to evaluate one model depends on the nature of the problem being addressed. These metrics are applied to the evaluation data (test set).

- **Accuracy:** It is the most basic metric evaluation of a model. The measure expresses the success rate of the model in classifying the samples into the correct ones categories on the data set.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predicitons\ made}$$

This metric is avoided when our data classes are not balanced as well; the correct predictions of the larger class can overshadow the incorrect predictions of the smaller classes. For example, suppose there is a problem with three classes of which the Class A has 980 samples, Class B has 10 samples and Class C has 10 samples. If the model predicts for each sample that it is in class A then the metric accuracy will have a value 980/1000 = 0.98 or 98%. The example shows that it is not possible to extract information on the classification of data by class and is not the appropriate measure for evaluation of such a model. This is a problem if it is important to anticipate the other classes are correct. For this reason, two new metrics are defined, precision and recall.

Before defining these two evaluation metrics, some classes are first defined based on the predicted and the actual class to which the samples in question belong. The definitions will be given for a two-class problem (Positive, Negative) but can easily be extended and for problems of more classes.

- **True Positive (TP):** The set of samples for which the prediction is the Positive

category and the actual category is the Positive. So, it's done correctly.

- **True Negative (TN):** The set of samples for which the prediction is the Negative category and the actual category is the Negative. So, it takes the correct classification.

- **False Positive (FP):** The set of samples for which the prediction is the Positive category and the actual category is the Negative. Therefore, buried classification.

- **False Negative (FN):** The set of samples for which the forecast is the Negative category and the actual category is the Positive. So, it takes place incorrectly.

The following are the definitions of Precision and Recall.

- **Precision**: It is the ratio of the correct prediction results of a class to the total number of forecasts in this class.

$$Precision = \frac{TP}{TP + FP}$$

This metric summarizes the model's ability to return samples related to that class as results.

- **Recall:** It is the ratio of the correct prediction results of a class to the total number of samples in this class.

$$Recall = \frac{TP}{TP + FN}$$

This metric summarizes the model's ability to find all samples of a particular class.

It is desirable that both metrics have high values. Usually, however, there is a trade-off between them, but without excluding the fact that the proposed model has a solution to return high prices on both. One metric that combines the two previous metrics is the F1-Score.

- **F1 Score**: It is the harmonious average of Precision and Recall.

$$F1 = \frac{Precision \cdot Recall}{Precision + Recall}$$

The higher this metric the better the results the model returns.

In the case of multiple classes in which an imbalance occurs, often used a variant of F1-Score that is called micro-F1. The calculation is performed by first summing the

variables TP, FP, FN from all classes, the metric Precision and Recall, now called micro-Precision and micro-Recall respectively and finally the metric micro-F1 is calculated.

2.7 Regularization and Dropout

Regularization is a set of techniques that can prevent overfitting in deep neural networks can learn very complicated relationships between their inputs and outputs but they are likely to quickly overfit a training dataset with few examples. In order to reduce overfitting there have been developed some methods such as stopping the training when performance on a validation dataset is starting to get worse or weight penalties of several kinds like L1 and L2 regularization.

- **L1 regularization:** is a way to penalize the parameter size and to reduce the gradient by a constant factor equal to sign (Wi).

$$\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1 = \sum_i |w_i|$$

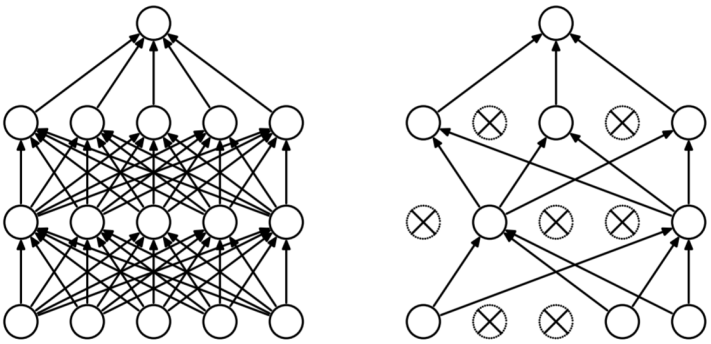
  Equivalent to MAP estimation with Laplace prior.

- **L2 regularization:** Drives the weights closer to the origin shrinks the weights vector by a constant factor on each training step.

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

  Equivalent to MAP estimation with Gaussian prior.

Aside from L2 and L1 regularization, another well-known and effective regularization method is known as dropout regularization. Dropout regularization is a straightforward operation. Dropout suggests that during preparation, with any probability P, a neuron of the neural network is switched off typically by multiplying their outputs by zero.



(a) Standard Neural Net (b) After applying dropout.

Figure 8: Neural Network before and after dropout (source: https://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf)

Dropout can be seen as practical computationally inexpensive bagging for deep neural networks.

Chapter 3

Convolutional Object Detection & Classification

In this Chapter we address and compare various methods of object detection, using neural convolutional networks. We are going to, in fact, look at methods that combine region proposal classification(also called regions of interest) with CNNs. We also take a look at how classification works and we emphasize in hierarchical classification.

3.1 R-CNN

R-CNN stands for Region Based Convolutional Neural Networks.Krizhevsky [25] achieved positive outcomes with CNNs in 2012 and after that the next year a method [26] was published by Girshick to generalize these findings to object detection. This approach is called R-CNN ("CNN with region proposals").

3.1.1 General Overview

There are several phases of R-CNN forward computation, shown in figure 9. In the first place, it generates regions of interest(RoIs). Category independent RoIs are bounding boxes with a high probability of containing an interesting object.Next, to extract features from each regional proposal, a convolutional network is used. To fit the input size of the CNN, the sub-image stored in the bounding-box is distorted and then fed to the network. The features are input to support vector machines (SVM) that provide the final classification, after the network has extracted features from the input.

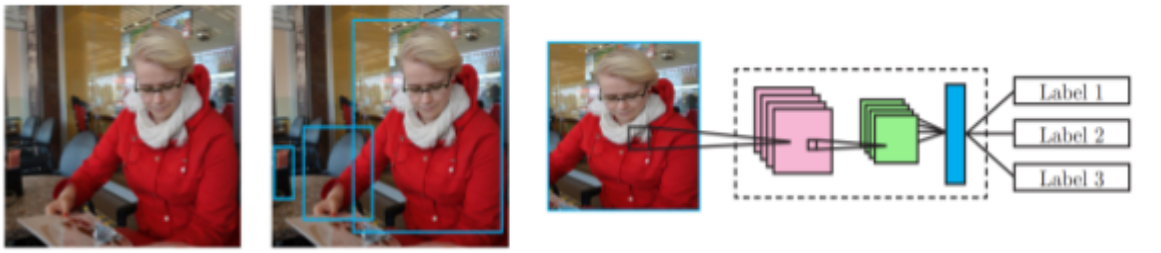


Figure 9: Stages of R-CNN forward computation (source: www.semanticscholar.org)

This technique - method is trained in multiple stages, starting with the convolutional network [20]. After the CNN has been trained, the SVMs are fitted to the CNN features. Eventually, the region proposal generating method is trained.

3.1.2 Disadvantages

R-CNN is an important technique since the first practical solution for object detection using CNNs was presented. Being the first, it has several disadvantages that later approaches have built on. Girshick mentions three major R-CNN concerns in his 2015 paper for Fast R-CNN[27]. First, and maybe most important, the detection of objects is sluggish, requiring for each picture nearly a minute, even on a GPU. This is because the CNNforward computation is performed separately for every object proposal.This happens even if the proposals originate or overlap each other from the same picture.

3.2 Fast R-CNN

Grishick in 2015 found a more efficient method for object detection and named that method Fast R-CNN [20]. The core idea is to perform the forward pass of the CNN for the entire image, instead of performing it separately for each region of interest.

3.2.1 General Overview

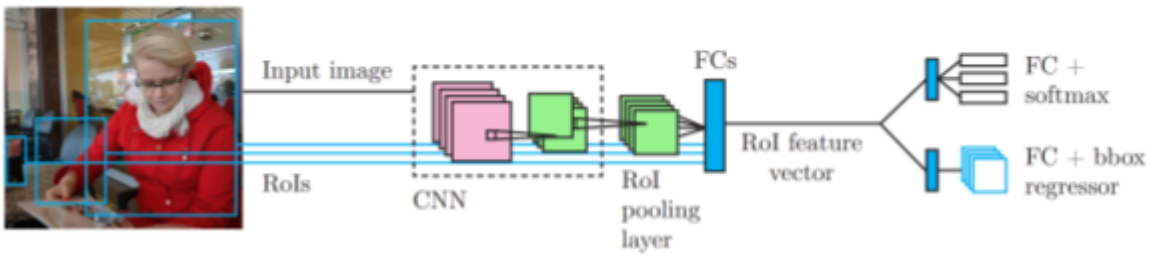


Figure 10: Stages of Fast R-CNN forward computation (source: www.semanticscholar.org)

In figure 10 we can see the general structure of Fast R-CNN. The method receives as input an image plus regions of interest computed from the image. As in R-CNN, the RoIs are generated using an external method. The image is processed using a CNN that includes a lot of convolutional and max pooling layers. The convolutional feature map that is generated after these layers is input to a RoI pooling layer. This extracts a fixed-length feature vector for each RoI from the feature map. The feature vectors are then input to fully connected layers that are connected to two output layers: a softmax layer that produces probability estimates for the object classes and a real-valued layer that outputs bounding box coordinates computed using regression (meaning refinements to the initial candidate boxes).

3.2.2 Performance

As authors claim, using the same feature map for each RoI, Fast R-CNN provides significantly shorter classification time compared to regular R-CNN, taking less than a second on a state-of-the-art GPU [27]. The overall computation time begins to depend significantly on the efficiency of the area proposal generation method as the detection time decreases. Thus, the RoI generation can shape a bottleneck of computation[28]. In

addition when there are many regions of interest increases the evaluation time.Finally, Classification time can be accelerated by approximately 30% if the fully-connected layers are compressed using truncated singular value decomposition [27]. This results in a slight decrease in precision, however.

3.2.3 Disadvantages

Fast R-CNN does not have as many disadvantages as its predecessor R-CNN, but it also uses the Selective Search Algorithm, which is slow and time-consuming and it takes around 2 seconds in a CPU implementation to detect objects in image that often do not function properly with huge real-life datasets [29].

3.3 Faster R-CNN

Ross Girshick, Shaoqing Renet et al. in 2015 found a more efficient method for object detection and named that method Faster R-CNN [27] we can consider it as an integrated method [23].In this thesis it's the main algorithm that it is used.

3.3.1 General Overview

The innovative idea from the authors was that in order to generate region proposals, we can use feature maps generated from object detection networks as well. This part of the network is called region proposal network (RPN) and it is used instead of a selective search algorithm. RPN has proven to be very efficient till now. The authors used Fast R-CNN architecture for the detection network.

3.3.2 Architecture and Performance

This algorithm is much faster than the previous ones (figure 11).

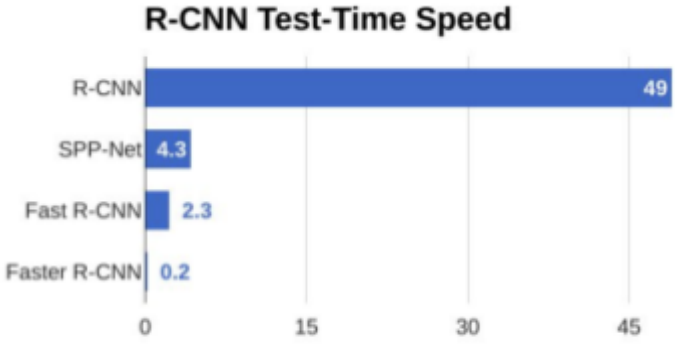


Figure 11: Comparison of test-time speed of object detection algorithms (source: www.towardsdatascience.com)

Faster R-CNN is composed of three main parts. These parts are the convolution layers, the RPN and classes,bounding boxes.This algorithm's network is trained by alternating between training for RoI generation and detection. First, two separate networks are trained. After that, these networks are combined and fine tuned. While fine tuning is taking place, certain layers are kept fixed and certain layers are trained in their turn.As input the trained networks receives an image where the feature maps are generated. As said earlier

these features are fed to the region proposal network. The RPN output with the feature maps are the input for the final detection layers where eventually, the final classifications are computed from these layers with the help of a RoI pooling layer [30]. The performance boost is coming from the fact that the conputaniom of the region proposals on a CNN can be realizable on a GPU, rather than older RoI generation methods such as selective search, which is mentioned earlier, are implemented on a CPU.

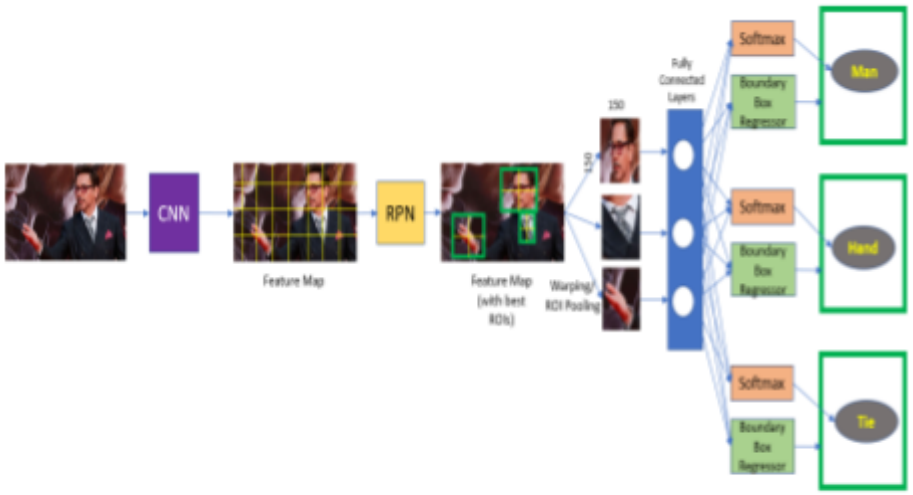


Figure 12: Faster R-CNN (source: www.towardsdatascience.com)

3.4 Classification

The process of predicting the class of given data points is known as classification. Classes are also known as goals, marks, or groups. The role of approximating a mapping function (f) from discrete input variables (X) to discrete output variables is known as classification predictive modeling (y).An example of a classification problem is to categorize an incoming email to “spam” or not. Classification belongs to the category of supervised learning where the targets also provided with the input data, also there are two types of learners in classification as eager learners and lazy learners.

Lazy learners simply store the training data and wait until testing data appear. When it does, classification is conducted based on the most related data in the stored training data. Compared to eager learners, lazy learners have less training time but more time in predicting.Famous algorithms that belonging there are the k-nearest neighbor and Case-based reasoning. On the other hand eager learners construct a classification model based on the given training data before receiving data for classification. It must be able to commit to a single hypothesis that covers the entire instance space. Due to the model construction, eager learners take a long time to train and less time to predict. Known algorithms in this type are the Decision Tree and Naive Bayes [31].

One sub category of a classification task is the hierarchical classification which is a classification splitted in more than one step. For example CP2 is an hierarchical classification task because the first step is to categorize the leaf among “healthy” or “diseased” and the second step if the leaf is categorized as “diseased” is to further categorize the leaf into one of the following three categories “Black Measles”,”Black Rot”,”Leaf Blight”.These categories are common leaf diseases.[32]

Chapter 4

Datasets - Preprocess

4.1 Datasets

As already mentioned in this thesis two components developed. The first component which is an object detection, used for training the Leaf Dataset (see 4.1.1) and the second component, which is about classifying health and diseased leaves, used for training the rest of the Datasets that are described below (see 4.1.2 - 4.1.7). In our experiments for CP2 we tried to keep a balanced number of images per class.

4.1.1 Leaf Dataset

The content of this dataset is images of plants, trees, or even individual leaves. Each image has bounding boxes annotated around each leaf, but only for those that can be seen easily. This dataset has 1140 images with 5346 annotated leafs.[33]

Leaf Dataset				
Species	**Size**	**Classes**	**Format**	**Resolution**
Random plants and trees	1140	1	JPG	1024x1024

Table 1: Leaf Dataset description.

Figure 13: Leaf Dataset sample image(source : https://www.kaggle.com/alexo98/datasets)

4.1.2 Plant Village Dataset

The Plant Village Dataset consists of 54303 healthy and unhealthy leaf images divided into 38 categories by species and disease. Each category has one class of healthy leaf images and at least one class of images with diseased leaves [34]. From this dataset the species that were eventually used can be shown below (table 3).

Plant Village Dataset				
Species	**Size**	**Classes**	**Format**	**Resolution**
Various plants	54303	38	JPG	256x256

Table 2: Plant Village dataset description.

Plant Village Dataset Details		
Species	**Size**	**Classes**
Apple	3171	4
Cherry	1906	2
Corn	3852	4
Grape	4062	4
Peach	2567	2
Pepper	2475	2
Potato	2152	3
Strawberry	1565	2
Tomato	18160	10

Table 3: Plant Village Dataset details.

Figure 14: Plant Village Dataset sample image (source : https://www.tensorflow.org/datasets/catalog/plant_village)

4.1.3 Plant Pathology 2020 Dataset

The Plant Pathology 2020 Dataset provides also annotated field images that depict real-life symptom images of multiple apple foliar diseases, with variable illumination, angles, surfaces, and noise.This dataset contains 3642 images, one class of healthy leaf images and three classes of images with diseased leaves [35].

Plant Pathology 2020 Dataset				
Species	**Size**	**Classes**	**Format**	**Resolution**
Apple	3642	4	JPG	2048x1365

Table 4: Plant Pathology 2020 Dataset description.

Figure 15: Plant Pathology 2020 Dataset sample image
(source : https://www.kaggle.com/c/plant-pathology-2020-fgvc7/overview)

4.1.4 Embrapa WGISD Dataset

The Embrapa WGISD dataset provides annotated field images of five different grape varieties and it consists of .This dataset contains 300 high resolution images ,with five classes of images with healthy leaves [36].

Embrapa WGISD Dataset				
Species	**Size**	**Classes**	**Format**	**Resolution**
Grape	300	5	JPG	5184x3456

Table 5: Embrapa WGISD Dataset description.

Figure 16: Embrapa WGISD Dataset sample image(source : https://zenodo.org/record/3361736).

4.1.5 Coffee Dataset

The Coffee Dataset consists of 1747 healthy and unhealthy leaf images of coffee. It has one class of healthy leaf images and four classes of images with diseased leaves [37].

Coffee Dataset				
Species	**Size**	**Classes**	**Format**	**Resolution**
Coffee	1747	5	JPG	2048x1024

Table 6: Coffee Dataset description.

Figure 17: Coffee Dataset sample image(source : https://drive.google.com/file/d/15YHebAGrx1Vhv8-naave-R5o3Uo70jsm/view).

4.1.6 COCO 2017 Dataset

COCO, short for Common Objects in Context, is a large image recognition/classification, object detection, captioning, and segmentation dataset and it is one of the most popular open source object recognition databases used to train deep learning programs. It contains complex everyday scenes of common objects in their natural context [38]. This Dataset is used by CP1 in order to evaluate how Transfer Learning [39] performs in our task.

COCO Dataset			
Train Images	**Test Images**	**Validation Images**	**Resolution**
118000	41000	5000	640x480

Table 7: COCO 2017 Dataset description.

Figure 18: COCO 2017 Dataset sample image (source : https://www.researchgate.net/figure/Example-images-from-MS-COCO-dataset_fig2_341596604).

4.1.7 Production Dataset

In order to evaluate our components in production level (CP1 and CP2) we gathered a small dataset with real life images of plants. That was a necessary step in order to approach a more realistic output.

Production Dataset	
Validation Images	**Resolution**
50	1536x2048

Table 8: Production Dataset description.

Figure 19: Production Dataset sample image.

4.2 Data Preprocess

4.2.1 Data preparation

Except from the Leaf dataset that it used only by CP1, the rest datasets had their images resized to the shape 256×256 pixels. This low resolution selected mainly because if we were operating on an image that was 1024 by 2048 pixels for example, each layer could have millions of nodes.Also because of memory limitations in the available hardware that used in this thesis, it wasn't possible to handle images with higher resolution.Another action that was made in datasets of CP1 and CP2 was to transform all the images into grayscale, in order to compare the results in between color and gray images. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel, which reduces computation complexity.

4.2.2 Data cleaning

Data cleaning is a critically important step in any machine learning project. In text data, there are many different statistical analysis and data visualization techniques that can be used in order to clean a dataset in contrast to datasets with images, as in our case, the cleaning is mainly done manually and usually by domain experts. The quality of data cleansing has a direct impact on the accuracy of the derived models and conclusions.Data cleaning main goal is to fill , fix or remove data.As John Adair says "garbage in, garbage out" [40], the meaning behind this phrase is that incorrect or poor-quality input will produce faulty output. In our case, our datasets were already "clean", the only cleaning was to remove some blur and duplicate images among removing some corrupted images due to resolution downscale that was made in the previous step(4.2.1).

4.2.3 Data augmentation

In image datasets, data augmentation is used for increasing the size of a dataset. It is the process where new images are generated by slightly distorting the original images. It can also act as a regularizer and helps reduce overfitting when training a machine learning model. In machine learning, as well as in statistics, overfitting appears when a statistical model describes random noise or error rather than underlying relationship [41]. There are many modifications that we can do to images, some of them are introduced in the below list.

- Resize
- Horizontal or vertical flip
- Deform
- Add blur
- Modify colors
- Rotate
- Zoom in or zoom out
- Increase or reduce brightness

In our case in order to increase our dataset we used the rotation among the horizontal and vertical flip techniques.

Chapter 5

Implementation and Deployment of the Application

5.1 Machine Learning Components Code Base

The experiments were developed in Python 3.8 using Jupyter Notebooks [42] for ease. All the experiments regarding the CP1 component were conducted using an open source library named PyTorch [43]. On the other hand the CP2 component was built using an open source library named Keras [44]. Finally, in table 9 are presented the basic characteristics of the machine that is used in order to train all of the experiments.

Hardware and Software	Characteristics
Memory	16.0 GB
Processor	Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
Graphics	NVIDIA GeForce GTX 960M
Operating System	Linux Ubuntu 18.02 64 bits

Table 9: Basic machine characteristics.

5.2 API

In order to demonstrate and evaluate this thesis' main concept an API [45] was developed. The API is written in NodeJs[46] and takes as input an image (with leaves in order to have meaningful results) and returns a response in JSON [47] format that contains the corresponding results, a sample response is shown in figure 20. The endpoint of the API is “http://dev.scio.services:7001/sso” and as already said, it takes as argument the path of an image. In order to transfer the image to the server we have to encode it with base64, we can achieve that using a python library named accordingly. Below we can see a python script example of how we can call the API.

```
import requests
import json
import base64

with open("/content/image_sample.jpg", "rb") as img_file:
    my_string = base64.b64encode(img_file.read())

base64Image = ""+my_string.decode('utf-8')
```

```
data = {'base64Image': base64Image}
json.dumps(data)

url = 'http://dev.scio.services:7001/sso'
headers = {'Content-type': 'application/json', 'Accept': 'text/plain'}
response = requests.post(url, data=json.dumps(data), headers=headers)

print(response.text)
```


```
{
  "bbox": [
    [ 3,34,500,794],
    [ 272, 34, 977, 556]
  ],
  "totalLeafs": "2",
  "healthy": "0",
  "diseased": "2",
  "blackrot": "0",
  "blackMeasles": "0",
  "leafBlight": "2",
  "healthyPercentage": "0.034",
  "diseasedPercentage": "0.966"
}
```


Figure 20: API json format response.
An explanation of the results can be found in table 10.

Key	Value
bbox	An array that contains the bounding boxes of the found leaves
totalLeafs	The amount of total leaves that are found.
healthy	How many of the found leaves are Healthy
diseased	How many of the found leaves are Diseased
blackrot	How many of the found leaves has the black rot disease
blackMeasles	How many of the found leaves has the black measles disease
leafBlight	How many of the found leaves has the leaf blight disease
healthyPercentage	This percentage is calculated by dividing the total number of leaves with CP2s prediction

	number about the healthy class
diseasedPercentage	This percentage is calculated by dividing the total number of leaves with CP2s prediction number about the diseased class

Table 10: API Results explanations

5.3 Demo Application

In order to demonstrate and evaluate this thesis' main concept, an application was developed. The application has two parts, the user interface (UI) and the API part that was explained earlier. The UI is written in React Js[48]. The web application can be seen from the following figures (figure 21, 22, 23) where we uploaded an image containing two "diseased" leaves (figure 22), the results are shown in figure 23. It also can be accessed in this url "http://dev.scio.services:7000/", which will be active until the end of 2021.

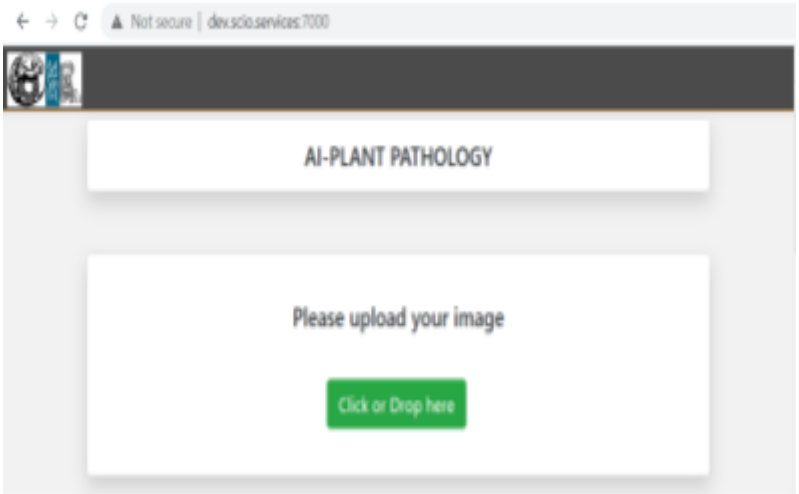


Figure 21: Application Main Page

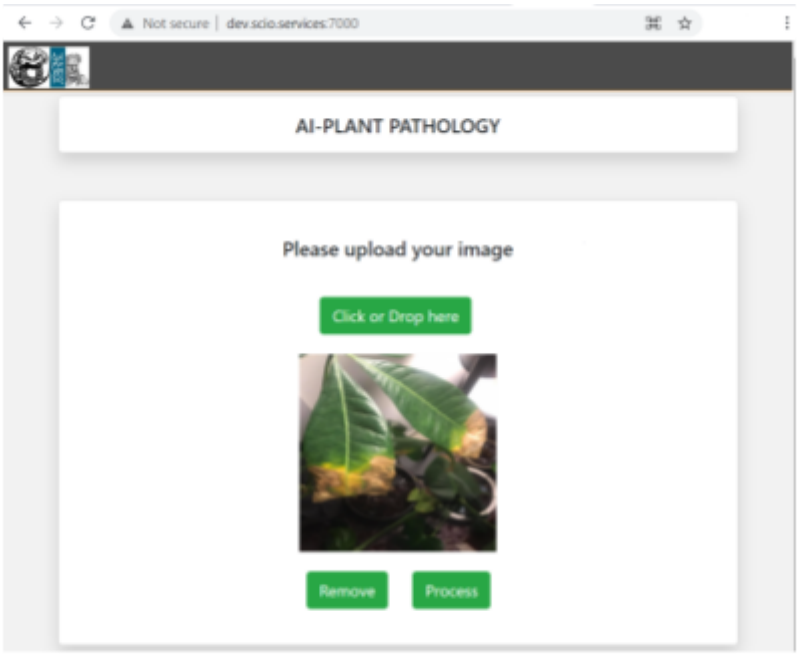


Figure 22: Application Page with inserted image.

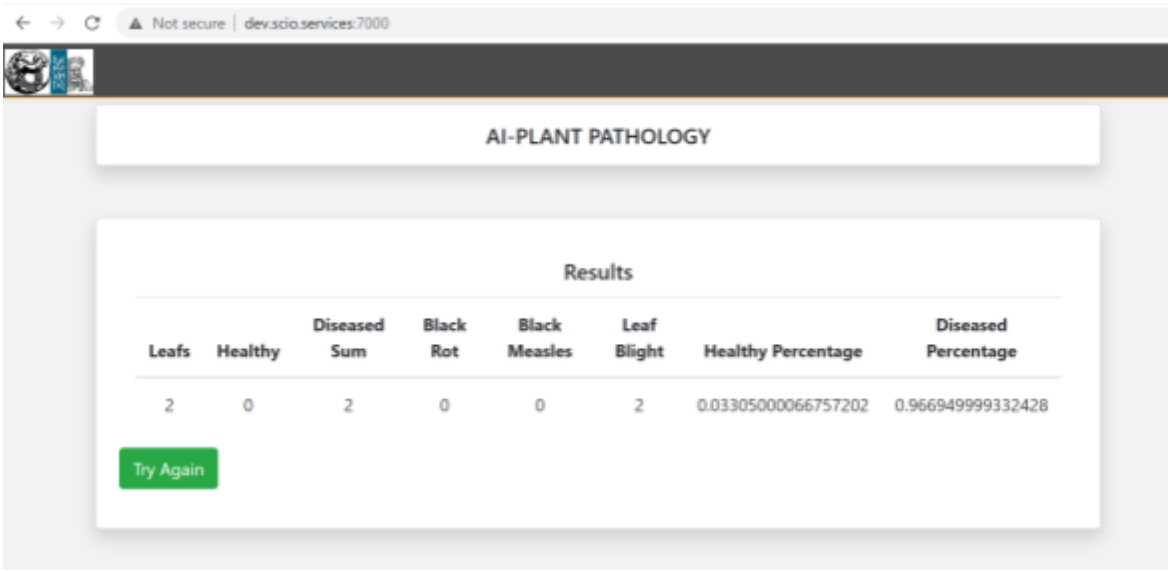


Figure 23: Application results page.

5.4 Deployment & Open Source Code

In order to contribute to the community everything in this thesis is publicly accessible from a Github [49] repository that has been created for this cause. Also in order to help someone to run the described application and its components to his infrastructure, another repository in Dockerhub [50][51] has been created that contains a containerized version of our components. Now it follows a small guide about the deployment using Docker technology. Docker [52] in simple words it's a container engine that is used for running applications. It is very useful because there is no need to install anything else like libraries or other technologies because everything is inside a docker container.

Prerequisites

- Physical or Virtual machine of Linux distribution.
- Install Docker.
- Install docker-compose.

Now in order to run our API and our application we create a docker-compose.yml file that contains configuration about the listening ports of our services, this file should be like the following figure 24.

```
version: '2.2'
services:
  plant-pathology-api:
    container_name: plant-pathology-api
    image: zorbaskyriakos/plant-pathology-api
    ports:
      - "7001:7001"
  plant-pathology-ui:
    container_name: plant-pathology-ui
    image: zorbaskyriakos/plant-pathology-ui
    ports:
      - "7000:7000"
```

Figure 24: docker-compose.yml file example

After the creation of the above file, we have to execute the following command in order to start our services:

- docker-compose up -d

Chapter 6

Methods - Experiments -Results

6.1 CP1 Methodology & Experiments

CP1 is our first component and it is trained for finding leaf or leaves inside an image. The approach for CP1 is to evaluate ResNet and AlexNet architectures among the Faster-RCNN algorithm. We also examine training the model from scratch or using transfer learning from the "COCO" dataset. Also, another aspect that was examined was the performance between color and gray scale images. Finally the dataset was splitted in two parts (train - test). The size of the split can depend on the size and specifics of the dataset, so two different cases were examined. Next we will describe the two architectures that we mentioned above.

- **Residual Network:** (ResNet) is a Convolutional Neural Network (CNN) architecture which can support hundreds or more convolutional layers .Resnet models were proposed in "Deep Residual Learning for Image Recognition" [53]. While previous CNN architectures had a drop off in the effectiveness of additional layers, ResNet can add a large number of layers with strong performance.

- **AlexNet:** is a CNN architecture which uses GPU to boost performance and is considered one of the most popular for object detection tasks. This architecture consists of 5 convolution layers, followed by 3 fully connected layers ,and finally ending with a softMax layer [54].

In order to sum up the following experiments were examined.

1) **Deep learning architecture:**
   - ResNet
   - AlexNet

2) **Training mechanism:**
   - Transfer Learning
   - Training from Scratch

3) **Dataset type:**
   - Gray scale
   - Color

4) **Choice Of Training - Testing set distribution:**
   - Train: 80%, Test: 20%
   - Train: 60%, Test: 40%,

In order to have a fair comparison between the results of all the experimental configurations, we used the same hyper-parameters across all the experiments.

- Learning rate : 0.005
- Momentum : 0.9
- Weight decay: 0.0005

- Batch size : 16
- Solver type : Stochastic Gradient Descent

Each of these 16 experiments runs for a total of 25 Epochs, where one epoch is defined as the number of training iterations in which the particular neural network has completed full pass of the whole training set. The choice of 25 epochs was made based on the empirical observation that in all of these experiments, the learning always converged well within 25 epochs across all the experiments. Finally, the methodology about the transfer learning with the COCO dataset (4.1.6), the first step was to take the network with its structure and its pretrained weights. We removed the last layer which is also the output layer and we added a new one. So the "second" training was made with our other dataset (see 4.1.1) and in our case we didn't modify the weights of the pretrained network only our newly added layers which as already mentioned it is a small dataset and that's why we chose this technique.

6.2 CP1 Results & Evaluation

In order to evaluate our experiments, the metric that is used was the F-measure or balanced F-score ($F1$ - score) which is the mean of precision and recall [55] (figure 25). $F1$ score reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Figure 25: F1 Score formula (source : towardsdatascience.com).

		AlexNet		Resnet	
		Transfer	**Scratch**	**Transfer**	**Scratch**
Grayscale	**80% - 20%**	0.8175	0.7375	0.8374	0.7675
	60% - 40%	0.8678	0.8248	**0.8846**	0.8352
Color	**80% - 20%**	0.7754	0.7175	0.7811	0.7504
	60% - 40%	0.8345	0.8145	0.8447	0.8114

Table 11: MeanF1 score across various experimental configurations at the end of 25 epochs.

In order to refer to a specific experiment from the above table the following notation is used Architecture:TrainingMechanism:DatasetType:Train-Test-Set-Distribution.
The winning configuration (marked with green color in table 11) was the following **ResNet::TransferLearning::GrayScale::60–40**, and the worse was AlexNet::FromScratch::Color::80–20.

After finding the best configuration we tried to tune our hyper parameters in order to achieve a better F1 score. As a tuning technique we used Grid search which is a process

that searches exhaustively through a manually specified subset of the hyper parameter space of the selected algorithm [56]. The final hyper parameters that are shown below achieved only a small change in our best F1 score only by +0.0081% :

- Learning rate : 0.004
- Momentum : 0.85
- Weight decay: 0.0005
- Batch size : 16
- Solver type : Stochastic Gradient Descent

So the final F1 score of our winning algorithm was **0.8927%.**

The last step for evaluating the CP1 was to validate it with the Production dataset(see 5.1.7). The F1 score was lower by -0.0603 % which sums up to **0.8324%**.

6.3 CP2 Methodology & Experiments

CP2 is our second component and it is trained for classifying an image that contains a leaf if it is healthy or diseased, if the component predicts the leaf inside the image is "diseased" then it classify it further with available categories being "leaf blight" , "black measles" , "black rot". For that reason we have 2 classifiers, the first one is a binary classifier that predicts if a leaf is healthy or not and we will analyze in section 6.3.1 and after we have a multiclass classifier that predicts in which "diseased" category our leaf belongs , accordingly we will analyze it in section 6.3.2

6.3.1 CP2 binary classifier

In our network of this binary classifier we used 26 layers. The main idea was convolution => relu => pool => dropout and in each new convolution layer we doubled the number of filters.We did that because as we move forward in the layers the patterns get more complex, and we want to capture as many combination patterns as possible. We also tried with more layers with the same logic but it led to overfitting. The activation functions that we used were "relu" [57] and "softmax" [58]. We used relu because it had higher accuracy, and it was much faster during training than swiss or sigmoid. Furthermore we used the softmax function in the last layer because we want the neural network to predict probability scores during the classification tasks. In other words the softmax activation function forces the values of output neurons to take values between zero and one, so they can represent probability scores in a binary class classification. We also tried Kullback Leibler Divergence Loss but with no better accuracy. We also used L2 regularisation in order to avoid overfitting with regularization factor = 0.001. We also tried L1 regularization and a combination of L1 and L2 but the accuracy wasn't better [59]. We used RMSprop optimizer [60], which is an optimizer that utilizes the magnitude of recent gradients to normalize the gradients. We used RMSprop optimizer because it led to better accuracy than "AdaDelta" [61] and "SGD" [62]. Finally we validated the model with 15% data after every epoch which is also shuffled after each epoch. The Hyper-parameters we used were :

- Batch size = 32
- Epochs = 100
- Optimizer Learning rate = 0.0001

- Loss Function = binary cross entropy
- Weight Decay = 1e-6

We also have to note that we used as input the Gray scale type of dataset, because it led to better accuracy.

Layer (type)	Output Shape	Param #
conv2d_69 (Conv2D)	(None, 32, 32, 32)	896
activation_85 (Activation)	(None, 32, 32, 32)	0
max_pooling2d_69 (MaxPooling	(None, 16, 16, 32)	0
dropout_71 (Dropout)	(None, 16, 16, 32)	0
conv2d_70 (Conv2D)	(None, 16, 16, 64)	18496
activation_86 (Activation)	(None, 16, 16, 64)	0
max_pooling2d_70 (MaxPooling	(None, 8, 8, 64)	0
dropout_72 (Dropout)	(None, 8, 8, 64)	0
conv2d_71 (Conv2D)	(None, 8, 8, 128)	73856
activation_87 (Activation)	(None, 8, 8, 128)	0
max_pooling2d_71 (MaxPooling	(None, 4, 4, 128)	0
dropout_73 (Dropout)	(None, 4, 4, 128)	0
conv2d_72 (Conv2D)	(None, 4, 4, 256)	295168
activation_88 (Activation)	(None, 4, 4, 256)	0
max_pooling2d_72 (MaxPooling	(None, 2, 2, 256)	0
dropout_74 (Dropout)	(None, 2, 2, 256)	0
conv2d_73 (Conv2D)	(None, 2, 2, 512)	1180160
activation_89 (Activation)	(None, 2, 2, 512)	0
max_pooling2d_73 (MaxPooling	(None, 1, 1, 512)	0
dropout_75 (Dropout)	(None, 1, 1, 512)	0
flatten_8 (Flatten)	(None, 512)	0
dense_16 (Dense)	(None, 512)	262656
activation_90 (Activation)	(None, 512)	0
dropout_76 (Dropout)	(None, 512)	0
dense_17 (Dense)	(None, 10)	5130
activation_91 (Activation)	(None, 10)	0

Total params: 1,836,362
Trainable params: 1,836,362
Non-trainable params: 0

Figure 26: Network overview of binary classifier.

6.3.2 CP2 multiclass classifier

In the network of this multiclass classifier we tried the same logic as the binary classifier that we saw in the previous section (6.3.1) but it led to overfitting. In order to solve that made some modifications to our network, one of them was to decrease the number of layers from 26 to 18 but with the same logic which again was convolution => relu => pool => dropout and in each new convolution layer we doubled the number of filters. We did that because as we explained in the previous section while moving forward in the layers the patterns get more complex, and we want to capture as many combination patterns as possible. The activation functions that we used were "relu" and "softmax" [63]. Furthermore we used the softmax function in the last layer for the same reason we did with the CP2 binary classifier (6.3.1). The loss function that we chose was the Categorical Cross Entropy [64] because our problem is multi-class classification. To avoid overfit other than decreasing the number of layers we also use L2 regularisation with regularization factor = 0.001. Finally, the optimizer that gave us better results was the "SGD", and we validated the model with 15% data after every epoch which is also shuffled after each epoch.

The Hyper-parameters we used were :

- Batch size = 32
- Epochs = 100
- Optimizer Learning rate = 0.0001
- Loss Function = categorical cross entropy
- Weight Decay = 1e-6

In this model we used as input the images that were predicted with the label "diseased" from the previous binary classifier.

```
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_66 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
activation_80 (Activation)   (None, 32, 32, 32)        0
_________________________________________________________________
max_pooling2d_66 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
dropout_67 (Dropout)         (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_67 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
activation_81 (Activation)   (None, 16, 16, 64)        0
_________________________________________________________________
max_pooling2d_67 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
dropout_68 (Dropout)         (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_68 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
activation_82 (Activation)   (None, 8, 8, 128)         0
_________________________________________________________________
max_pooling2d_68 (MaxPooling (None, 4, 4, 128)         0
_________________________________________________________________
dropout_69 (Dropout)         (None, 4, 4, 128)         0
_________________________________________________________________
flatten_7 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_14 (Dense)             (None, 128)               262272
_________________________________________________________________
activation_83 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_70 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_15 (Dense)             (None, 10)                1290
_________________________________________________________________
activation_84 (Activation)   (None, 10)                0
=================================================================
Total params: 356,810
Trainable params: 356,810
Non-trainable params: 0
```

Figure 27: Network overview of multiclass classifier.

6.4 CP2 Results & Evaluation

In order to evaluate our experiments, the metric that was used in both of our classifiers was the “accuracy” which calculates how often predictions equal labels.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Figure 28: Accuracy formula (source : towardsdatascience.com).

More analytically, among other things in 6.4.1 we will see the accuracy of our binary

classifier and accordingly in 6.4.2 about our multiclass classifier.

6.4.1 CP2 binary classifier

In this section we can see some plots of our binary classifier.

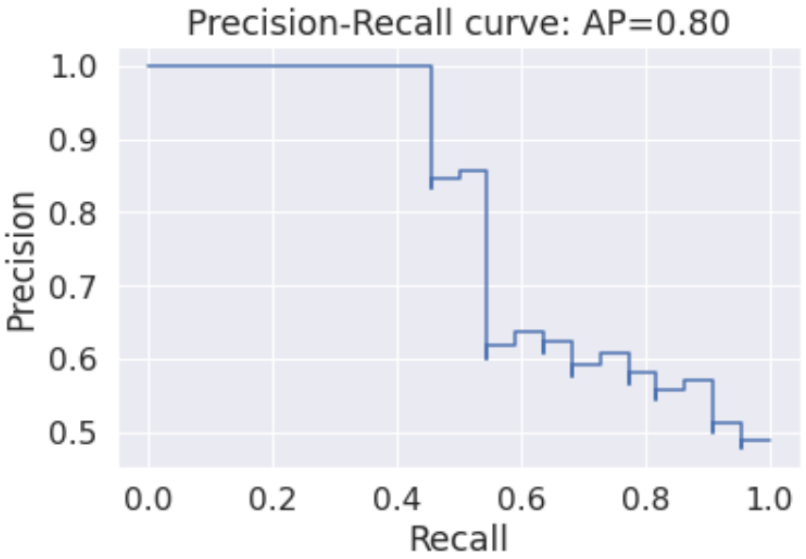


Figure 29: Binary classifier Precision-Recall curve

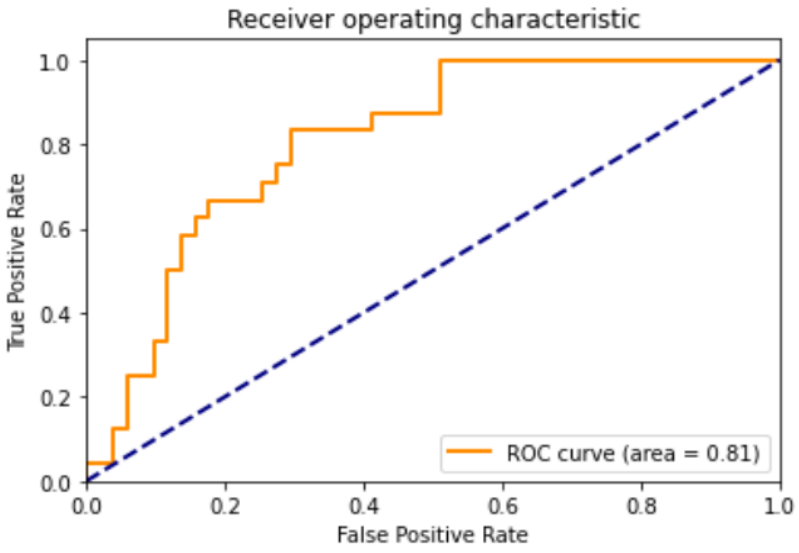


Figure 30: Binary classifier ROC curve

Our final model achieved accuracy 0.889 % in training and **0.866 %** in the test dataset.

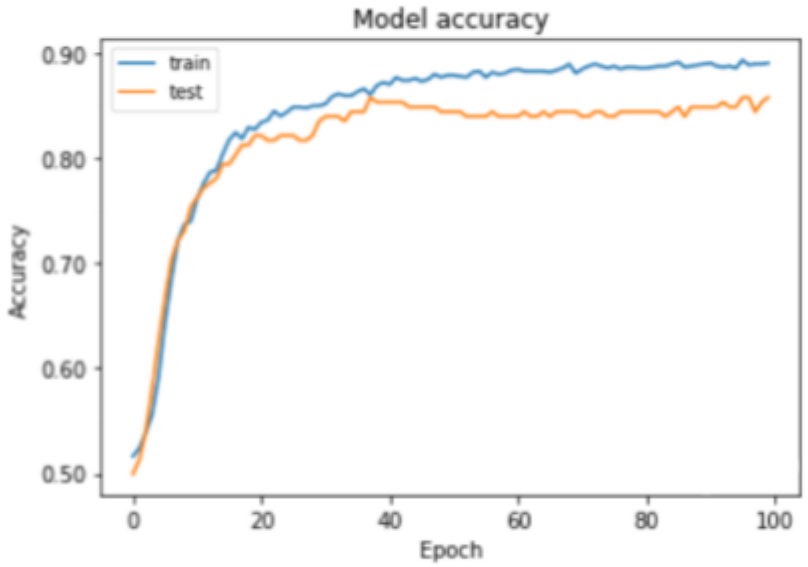


Figure 31: Binary classifier accuracy

As we did in the CP1 the last step for evaluating the CP2 binary classifier was to validate it with the Production dataset(see 4.1.7). The Accuracy score was a little lower and more specifically was **0.831 %**.

6.4.2 CP2 multiclass classifier

In this section we can see some plots of our multiclass classifier.

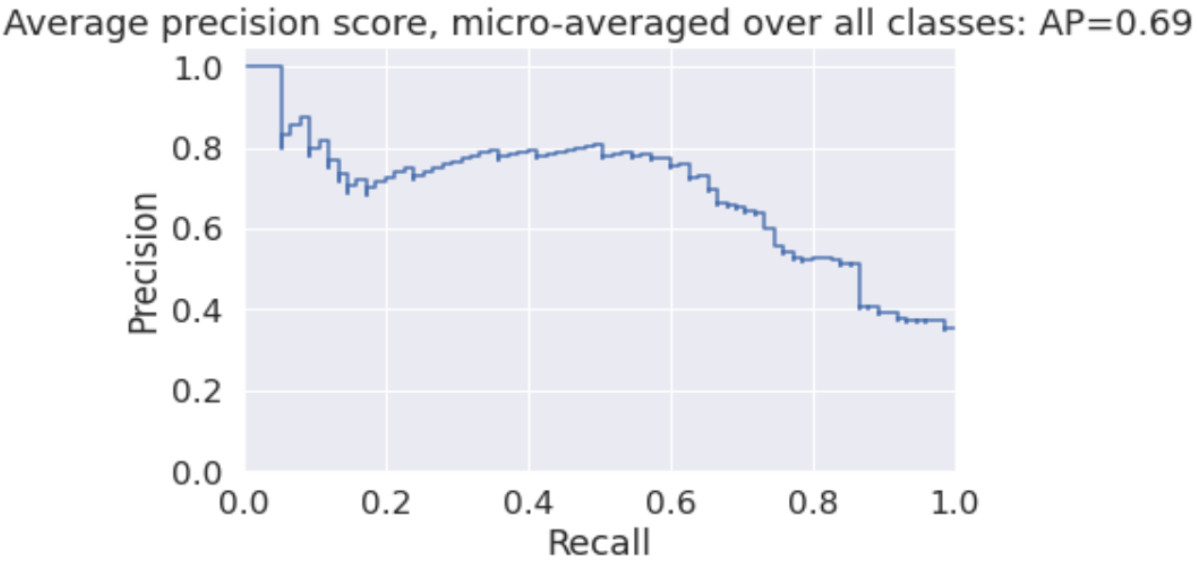


Figure 32: Multiclass classifier Precision-Recall curve

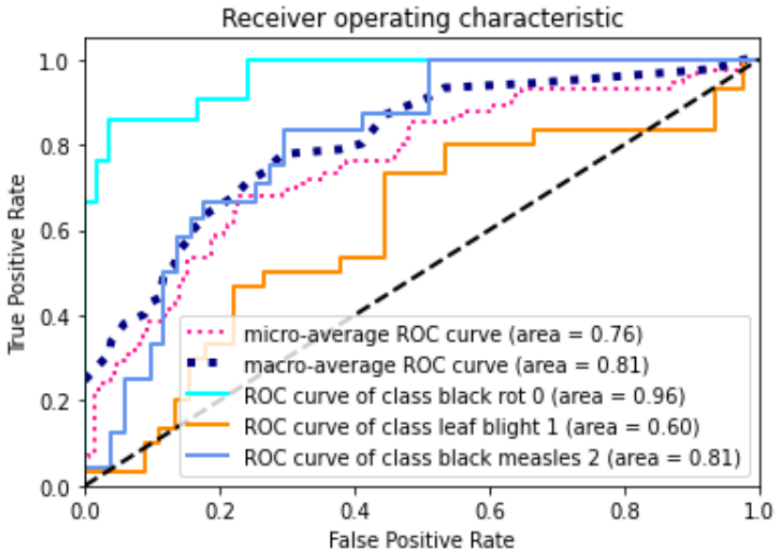


Figure 33: Multiclass classifier ROC curve

Our final model achieved accuracy 0.799 % in training and **0.766** in the test dataset. We have to note that without the use of convolutional neural networks the best accuracy that we managed to achieve was 0.58.

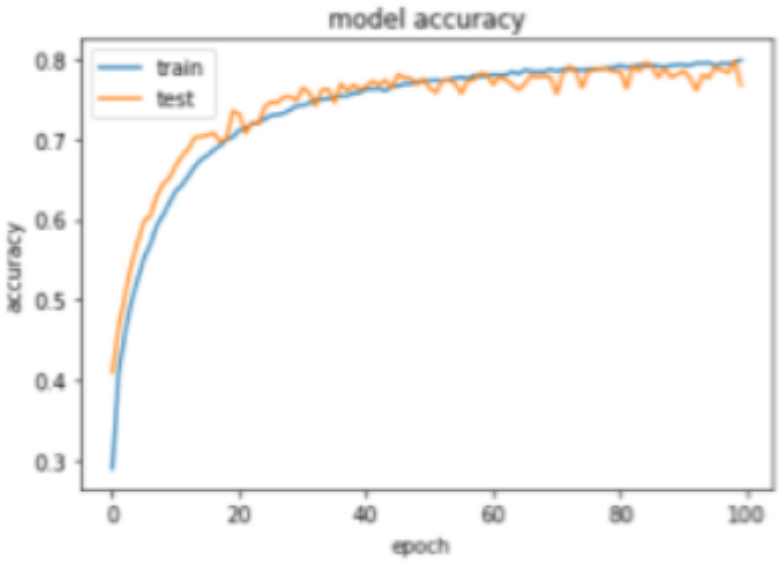


Figure 34: Multiclass classifier accuracy

Once again the last step for evaluating the CP2 multiclass classifier was to validate it with the Production dataset(see 4.1.7). The Accuracy score was a little lower and more specifically was **0.704 %**.

Chapter 7

Epilogue

7.1 Summary and conclusions

In this thesis we presented a solution for recognizing leafs from images and classifying them among some of the most known leaf diseases. Also an end to end web app has been created. There are various methods in computer vision plant disease identification and classification systems, but this research area is still underdeveloped. Furthermore, there are no commercial options available on the market, with the exception of those concerned with plant species identification based on leaf photographs. The developed components were able to detect leaf presence and distinguish between healthy leaves and 3 different diseases, which can be visually diagnosed. The complete workflow was described, respectively, from gathering the images used for training and validation to image preprocessing and augmentation and finally the procedure of training CP1 and CP2 and fine-tuning. Different test cases were performed in order to check the performance. Importantly, while the training of the model takes a lot of time(multiple hours on a high performance GPU computer),the classification itself is very fast(less than a few seconds on a CPU),and can thus easily be implemented on a smartphone.This presents a clear path toward smartphone-assisted crop disease diagnosis on a massive global scale of newly created model. New plant disease image database was created, more than 60,000 images using appropriate transformations. The CP1 achieved F1 score in validation dataset 0.89% and in production dataset 0.83%.In the other hand CP2 managed to achieve overall accuracy 0.86% in validation dataset and 0.83% in production dataset. Fine-tuning has not shown significant changes in the overall F1 score in CP1 or accuracy in CP2, but the augmentation process had greater influence to achieve respectable results. As far as we know, no connection with similar findings using the exact approach has been made because the presented procedure has not been applied in the field of plant disease identification.

7.2 Related Work

To date, much work has been done to classify the type of a leaf and in particular to classify their diseases, which use machine learning algorithms but also deep learning. Widespread algorithms and architectures are used in the various approaches, as well as specially designed or adapted architectures. Random Forest (RF), Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN) algorithms have prevailed in the most recent research. The use of these algorithms is often combined with other approaches to data preprocessing, such as the selection of special features that make class separation easier or the reduction of available features to minimize computational costs. without losing the accuracy of the results [65]. In addition to machine learning algorithms, deep learning approaches using neural networks have been tested. The use of Artificial Neural Networks (ANN), although it brings relatively good results, does not manage to exceed the success rates of the aforementioned machine learning algorithms [66]. In addition, it is interesting to use Convolutional Neural Networks (CNN) with one-dimensional (1-D) and two-dimensional (2-D) convulsions. The application of one-dimensional convolutions seems to produce better results both from the SVM and RF algorithms and from implementations with Long Short-Term Memory (LSTM) and the XGBoost algorithm. In

another study [67] the implementation of a two-dimensional convergent neural network led to better results than the one-dimensional one. Of particular interest are implementations that combine neural networks to create unique architectures. For example, the combination of CNN with Recursive Neural Networks (RNN) [68]. In such implementations the goal is to make the most of the properties of the different networks, extracting as much information as possible. The results of the above research show that such specially designed implementations are able to produce better results than machine learning algorithms and simple neural network implementations.

7.3 Future Work

As future work could be the following. An extension of this study will be on gathering images for enriching the database and improving accuracy of the model using different techniques of fine-tuning and augmentation. The main goal for the future work will be developing a complete system consisting of server side components containing a trained model and an application for smart mobile devices with features such as displaying recognized diseases or giving to users the option to verify the results. Having the user to verify the results will lead to retraining our components based on user opinion, in order for this to be successful the user must be a domain expert. Furthermore, future work will involve spreading the usage of the model by training it for plant disease recognition on wider land areas, combining aerial photos of orchards and vineyards captured by drones and convolution neural networks for object detection. By extending this research, in fruits, vegetables , and other plants , based on leaf images captured by the mobile phone camera.

This application will serve as an aid to farmers (regardless of the level of experience),enabling fast and efficient recognition of plant diseases and facilitating the decision-making process when it comes to the use of chemical pesticides. Furthermore, future work will involve spreading the usage of the model by training it for plant disease recognition on wider land areas, combining aerial photos of orchards and vineyards captured by drones and convolution neural networks for object detection. Finally by extending this research, we hope to achieve a valuable impact on sustainable development, affecting crop quality for future generations.

Bibliography

[1]FAO, Agriculture, food and water: A contribution to the World Water Development Report, English, FAO AQUASTAT REPORTS. Rome, Italy: FAO, 2003, isbn: 92-5-104943-2. [Online]. Available: http://www.fao.org/publications/card/en/c/Y4683E/ (visited on 05/02/2021).

[2]FAO Statistical Yearbook 2012 - Europe and Central Asia Food and agriculture, English. Rome, Italy: FAO, 2013, isbn: ISBN 978-92-5-107427-5. [Online]. Available: http://www.fao.org/3/i3138e/i3138e00.htm (visited on 05/02/2021).

[3] A. Obaisi, "OVERPOPULATION: A THREAT TO SUSTAINABLE AGRICULTURE AND FOOD SECURITY IN DEVELOPING COUNTRIES? A Review," 2017. doi: 10.13140/RG. 2.2.20613.04325. [Online]. Available: https://www.researchgate.net/publication/ 324792051 _ OVERPOPULATION _ A _ THREAT _ TO _ SUSTAINABLE _ AGRICULTURE _AND_FOOD_SECURITY_IN_DEVELOPING_COUNTRIES_A_Review (visited on 05/02/2021)

[4]R. Bhadouria, R. Singh, V. K. Singh, A. Borthakur, A. Ahamad, G. Kumar, and P. Singh, "Chapter 1 - Agriculture in the Era of Climate Change: Consequences and Effects," in Climate Change and Agricultural Ecosystems, K. K. Choudhary, A. Kumar, and A. K. Singh, Eds., Woodhead Publishing, 2019, pp. 1–23, isbn: 978-0-12-816483-9. doi: 10.1016/B978-0-12-816483 - 9 . 00001 - 3. [Online]. Available: http : / / www . sciencedirect . com / science /article/pii/B9780128164839000013.

[5]Deforestation and forest degradation, en, Library Catalog: www.iucn.org, Nov. 2017. [Online]. Available: https://www.iucn.org/resources/issues- briefs/deforestation- and forest-degradation (visited on 05/02/2021).

[6]O. Milman, "Earth has lost a third of arable land in past 40 years, scientists say," en-GB, The Guardian, Dec. 2015, issn: 0261-3077. [Online]. Available: https : / / www . theguardian . com/environment/2015/dec/02/arable-land-soil-food-security-shortage (visited on 05/02/2021).

[7]C. Zhao, B. Liu, S. Piao, X. Wang, D. B. Lobell, Y. Huang, M. Huang, Y. Yao, S. Bassu, P. Ciais, J.-L. Durand, J. Elliott, F. Ewert, I. A. Janssens, T. Li, E. Lin, Q. Liu, P. Martre, C. Müller, S. Peng, J. Peñuelas, A. C. Ruane, D. Wallach, T. Wang, D. Wu, Z. Liu, Y. Zhu, Z. Zhu, and S. Asseng, "Temperature increase reduces global yields of major crops in four independent estimates," Proceedings of the National Academy of Sciences, vol. 114, no. 35, pp. 9326–9331, 2017, Publisher: National Academy of Sciences _eprint: https://www.pnas.org/content/114/35/9326.full.pdf, issn: 0027-8424. doi: 10.1073/pnas.1701762114. [Online]. Available: https://www.pnas. org/content/114/35/9326.

[8]T. M. Mitchell, Machine Learning, 1st ed. New York, NY, USA, 1997, p. 2.

[9]Stuart J. Russell, Peter Norvig (2010) Artificial Intelligence: A Modern Approach, Third Edition, Prentice Hall ISBN 9780136042594.

[10]Hinton, Geoffrey; Sejnowski, Terrence (1999). Unsupervised Learning: Foundations of Neural Computation. MIT Press. ISBN 978-0262581684.

[11]Kaelbling, Leslie P.; Littman, Michael L.; Moore, Andrew W. (1996). "Reinforcement Learning: A Survey". Journal of Artificial Intelligence Research. 4: 237–285. arXiv:cs/9605103. doi:10.1613/jair.301. S2CID 1708582. Archived from the original on 2001-11-20.

[12]van Otterlo, M.; Wiering, M. (2012). Reinforcement learning and markov decision processes. Reinforcement Learning. Adaptation, Learning, and Optimization. 12. pp. 3–42. doi:10.1007/978-3-642-27645-3_1. ISBN 978-3-642-27644-6.

[13]Silver, David; Huang, Aja; Maddison, Chris J.; Guez, Arthur; Sifre, Laurent; Driessche, George van den; Schrittwieser, Julian; Antonoglou, Ioannis; Panneerselvam, Veda (January 2016). "Mastering the game of Go with deep neural networks and tree search". Nature. 529 (7587): 484–489.

[14]Bengio, Y.; Courville, A.; Vincent, P. (2013). "Representation Learning: A Review and New Perspectives". IEEE Transactions on Pattern Analysis and Machine Intelligence. 35 (8): 1798–1828. arXiv:1206.5538. doi:10.1109/tpami.2013.50. PMID 23787338. S2CID 393948.

[15]Schmidhuber, J. (2015). "Deep Learning in Neural Networks: An Overview". Neural Networks. 61: 85–117. arXiv:1404.7828. doi:10.1016/j.neunet.2014.09.003. PMID 25462637. S2CID 11715509.

[16]Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffry (2012). "ImageNet Classification with Deep Convolutional Neural Networks" (PDF). NIPS 2012: Neural Information Processing Systems, Lake Tahoe, Nevada.
[10]S. Haykin, Neural Networks: A Comprehensive Foundation, 1st. USA: Prentice Hall PTR,1994, isbn: 978-0-02-352761-6.

[17]F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” Psychological Review, vol. 65, no. 6, pp. 386–408, 1958, Place: US Publisher: American Psychological Association, issn: 1939-1471(Electronic),0033-295X(Print). doi: 10. 1037/h0042519.

[18]E. Kussul, T. Baidyk, L. Kasatkina, and V. Lukovich, “Rosenblatt perceptrons for handwritten digit recognition,” in IJCNN 01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222), ISSN: 1098-7576, vol. 2, Jul. 2001, 1516–1520 vol.2. doi:10.1109/IJCNN.2001.939589.

[19]M. Olazaran, “A Sociological Study of the Official History of the Perceptrons Controversy:” en, Social Studies of Science, June. 2016, Publisher: SAGE PublicationsLondon. doi: 10.1177/030631296026003005. [Online]. Available: https://journals.sagepub.com/doi/10.1177/030631296026003005 (visited on 05/04/2020).

[20]Zhang, Wei (1988). "Shift-invariant pattern recognition neural network and its optical architecture". Proceedings of Annual Conference of the Japan Society of Applied Physics.

[21]Zhang, Wei (1990). "Parallel distributed processing model with local space-invariant interconnections and its optical architecture". Applied Optics. 29 (32): 4790–7. Bibcode:1990ApOpt..29.4790Z. doi:10.1364/AO.29.004790. PMID 20577468.

[22]H. J. Kelley, “Gradient Theory of Optimal Flight Paths,” ARS Journal, vol. 30, no. 10, pp. 947–954, Oct. 1960, Publisher: American Institute of Aeronautics and Astronautics. doi: 10.2514/8.5282. [Online]. Available: https://arc.aiaa.org/doi/10.2514/8.5282

[23]H. Robbins and S. Monro, “A Stochastic Approximation Method,” EN, Annals of Mathematical Statistics, vol. 22, no. 3, pp. 400–407, Sep. 1951, Publisher: Institute of Mathematical Statistics, issn: 0003-4851, 2168-8990. doi: 10.1214/aoms/1177729586. [Online]. Available: https: //projecteuclid.org/euclid.aoms/1177729586.

[24]Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (2012), pp. 1097–1105.

[25]Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (2014), pp. 580–587.

[26]Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision (2015), pp. 1440–1448.

[27]Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (2015), pp. 91–99

[28] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” International Journal of Computer Vision (IJCV), 2013.

[29]Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (2015), pp. 91–99.

[30]Deng, Zhipeng & Sun, Hao & Zhou, Shilin & Zhao, Juanping & Lei, Lin & Zou, Huanxin. (2018). Multi-scale object detection in remote sensing imagery with convolutional neural networks. ISPRS Journal of Photogrammetry and Remote Sensing. 145. 10.1016/j.isprsjprs.2018.04.003.

[31]https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623. Retrieved 22 April 2021.

[32]https://www.planetnatural.com/pest-problem-solver/plant-disease/

[33]Retrieved from https://www.kaggle.com/alexo98/datasets

[34]Retrieved from https://www.tensorflow.org/datasets/catalog/plant_village

[35]Retrieved from https://www.kaggle.com/c/plant-pathology-2020-fgvc7/overview

[36]Retrieved from https://zenodo.org/record/3361736

[37]Retrieved from https://drive.google.com/file/d/15YHebAGrx1Vhv8-naave-R5o3Uo70j

[38]Retrieved from https://cocodataset.org/

[39]West, Jeremy; Ventura, Dan; Warnick, Sean (2007). "Spring Research Presentation: A Theoretical Foundation for Inductive Transfer". Brigham Young University, College of Physical and Mathematical Sciences. Archived from the original on 2007-08-01. Retrieved 2007-08-05.

[40]Adair, John (2009-02-03). The Art of Creative Thinking: How to be Innovative and Develop Great Ideas. Kogan Page Publishers. ISBN 9780749460082.

[41]D. M. Hawkins, “The problem of over-fitting,” Journal of Chemical Information and Computer Sciences, vol. 44, no. 1, pp. 1–12,2004.

[42]Retrieved from https://jupyter.org/

[43]Yegulalp, Serdar (19 January 2017). "Facebook brings GPU-powered machine learning to Python". InfoWorld. Retrieved 11 December 2017.

[44]Retrieved from https://keras.io/about/

[45]Retrieved from https://www.hubspire.com/resources/general/application-programming-interface/

[46]Retrieved from https://nodejs.org/en/

[47] "The JSON Data Interchange Format" (PDF). ECMA International. October 2013. Retrieved 12 February 2021.

[48]https://reactjs.org/

[49]https://github.com/KyriakosZorbas/DatascienceMscThesis

[50]https://hub.docker.com/repository/docker/zorbaskyriakos/plant-pathology-ui

[51]https://hub.docker.com/repository/docker/zorbaskyriakos/plant-pathology-ui

[52]https://www.docker.com/

[53]He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2015). Deep Residual Learning for Image Recognition. 7.

[54]Retrieved from https://cocodataset.org/

[55]Goutte, Cyril & Gaussier, Eric. (2005). A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. Lecture Notes in Computer Science. 3408. 345-359. 10.1007/978-3-540-31865-1_25.

[56]"Complexity of brute force search". coursera. Retrieved 14 June 2020.

[57]Vinod Nair and Geoffrey Hinton (2010). Rectified Linear Units Improve Restricted Boltzmann Machines (PDF). ICML.

[58]Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). "6.2.2.3 Softmax Units for Multinoulli Output Distributions". Deep Learning. MIT Press. pp. 180–184. ISBN 978-0-26203561-3.

[59]Ng, Andrew Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance (PDF). Proc. ICML.

[60]Retrieved from https://keras.io/api/optimizers/rmsprop/

[61]Retrieved from https://keras.io/api/optimizers/adadelta/

[62]Retrieved from https://keras.io/api/optimizers/sgd/

[63]Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). "6.2.2.3 Softmax Units for Multinoulli Output Distributions". Deep Learning. MIT Press. pp. 180–184. ISBN 978-0-26203561-3.

[64] Retrieved from https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy

[65]M. Zaefizadeh, A. Jalili, Majid Khayatnezhad, Roza Gholamin, andT. Mokhtari. Comparison of multiple linear regressions (mlr) and artificial neural network (ann) in predicting the yield using its components in the hulless barley.Advances in Environmental Biology, 5:109–113,01 2011

[66]R. B. Arango, I. Díaz, A. M. Campos, E. R. Canas, and E. F. Combarro, “Automatic arable land detection with supervised machine learning,” Earth Science Informatics, 2016. doi: 10.1007/s12145-016-0270-6.

[67]L. Zhong, L. Hu, and H. Zhou, “Deep learning based multi-temporal crop classification,” en,Remote Sensing of Environment, vol. 221, pp. 430–443, Feb. 2019, issn: 0034-4257. doi: 10.1016 / j . rse . 2018 . 11 . 032. [Online]. Available: http : / / www . sciencedirect . com /science/article/pii/S0034425718305418 (visited on 01/05/2021).

[68]Alberto Gonzalez-Sanchez, Juan Frausto-Solis, and Waldo Ojeda. Predictive ability of machine learning methods for massive crop yield prediction.Spanish Journal of Agricultural Research, 06 2014.