Crop Protection and Advisory System Using Deep Learning

GMR Institute of Technology

A miniproject report submitted in partial fulfilment for the award of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

(2018-2019)

Submitted by

HARSHA VARDHAN N	**(16341A05C6)**
YAMINI R	**(16341A05F0)**
HIMA BINDU N	**(16341A05C3)**
SINE RAJA	**(16345A05G4)**

Under the Esteemed Guidance of

Mr. K.Somesh

Assistant Professor-CSE

GMR Institute of Technology

An Autonomous Institute Affiliated to JNTUK, Kakinada

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

(Accredited by NBA, NAAC with 'A' Grade & ISO 9001:2008 Certified Institution)

GMR Nagar, Rajam – 532 127, Andhra Pradesh, India.

April 2019

GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

GMRIT
Training Tomorrow's Engineers Today

Department of Computer Science and Engineering

CERTIFICATE

*This is to certify that the project entitled "**Crop Protection and Advisory System Using Deep Learning**"submitted by* **HARSHA VARDHAN N (16341A05C6) YAMINI R (16341A05F0) HIMA BINDU N (16341A05C3) SINE RAJA (16345A05G4)** *has been carried out in partial fulfilment of the requirement for the award of degree of* ***Bachelor of Technology*** *in* ***Computer Science and Engineering Department*** *of* ***GMR Institute of Technology****, An Autonomous Institute Affiliated to JNTUK, KAKINADA, is a record of bona fide work carried out by them under my guidance supervision. The results embodied in this report have not been submitted to any other University or Institute for award of any degree.*

Signature of Supervisor	**Signature of the HOD**
Mr. K.Somesh	**Dr. A.V.Ramana**
Asst.Professor	Professor &Head
Department of CSE	Department of CSE
GMRIT	GMRIT

ACKNOWLEDGEMENT

We would like to extend our sincere thanks to express our gratitude to the faculty of **GMRIT** for providing us an opportunity to purse B.TECH. The knowledge and experience that we gained from here during our study period has be very valuable for us.

We would like to extend our sincere thanks to our guide **Mr. K.Somesh** for his support and guidance during our project. His valuable suggestions and comments always served us as a source of inspiration and encouragement.

Our special thanks to Head of the Department, **Dr. A.V.Ramana** for his kind support, throughout our study period. We are grateful to thank the staff members of computer science engineering department for the support, during our study period.

I take privilege to thank our Principal **Dr. C. L. V. R. S. V. Prasad** and our Vice principal **Dr. J. Raja Murugadoss** who has made the atmosphere so easy to work. I shall always be indebted to them.

I would like to thank all the faculty members of the Department of Computer Science Engineering for their direct or indirect support and also all the lab technicians for their valuable suggestions and providing excellent opportunities in completion of this report.

Team Members

HARSHA VARDHAN N	**(16341A05C6)**
YAMINI R	**(16341A05F0)**
HIMA BINDU N	**(16341A05C3)**
SINE RAJA	**(16345A05G4)**

INDEX

LIST OF FIGURES

ABSTRACT

The idea is to develop a mobile app that can recognize the plant and its disease (if it is effected) by uploading the image of the plant or specific part of the plant effected by the pest. This can be done by the deep learning and image recognition; the neural network is trained with the large dataset for this task. It contains all the information about the plant problems and their treatment for the different stages of the diseases. The application can be accessed by everyone. The application User interface will be developed in such a way that illiterate farmers can also use the application effectively and this can help the farmers to very great extent. The proposed system can effectively identified different types of diseases with the ability to deal with complex scenarios from a plant's area.

CHAPTER-1

INTRODUCTION

1. INTRODUCTION

1.1 Aim:

Plant diseases are one of the major issues that directly reduce the quality of agriculture which causes severe economic losses. Crop protection is the science and practice of managing plant diseases, weeds and other pests. In order to check the condition of the plant, we develop a mobile app and website where the plant condition and its type is checked by uploading the image or specific part of the plant. Here, deep learning and image recognition is used and the neural network is trained with large dataset and the network is updated from the new images provided by the user. The application can be accessed by everyone. It contains information of plant and can give solutions for the different stages of diseases. Plant diseases are one of the major issues that directly reduce the quality of agriculture which causes severe economic losses .Crop protection is the science and practice of managing plant diseases, weeds and other pests .In order to check the condition of the plant, we develop a mobile app and website where the plant condition and its type is checked by uploading the image or specific part of the plant. Here, deep learning and image recognition is used and the neural network is trained with large dataset and the network is updated from the new images provided by the user. The application can be accessed by everyone. It contains information of plant and can give solutions for the different stages of diseases. offered to them.

1.2 Problem Statement:

As the agriculture is the main source for the country, we have to control the diseases of the crops and plants. The development of the agriculture depends on the various factors like type of soil, climate, vegetation, pesticides and insecticides, use of irrigation etc. There are many reasons or factors where the crops or plants are affected, the mostly due to pests. In order to save the crops from pests, we should follow the best and important way for saving our agriculture.

1.3 Motivations:

Globally, farmers lose 30 to 40 % of their crops due to pests and diseases according to UN Food and Agricultural Organization. Also proper crop protection is important to produce higher quality crops with minimal wastage. All types of agriculture depends on pesticides. The planet benefits by experiencing less demand for its natural resources. Since the farmers economically depends on only agriculture, so the main motto behind this pest detection is to benefit the farmers.

1.4 Objectives:

To provide the proposed pest management system as a cloud service to assist farmers by providing advisory based on pest images sent to Agro- information Cloud database through mobile devices.

To produce food of high nutritional quality in sufficient quantity.To inspect the imported agricultural commodities for preventing the introduction of exotic pests and diseases. To minimize the food wastage due to pests affect. To encourage and enhance biological cycles within the farming system, involving microorganisms, fauna, plants and animals etc. The main objective is to help farmers by improving agriculture productivity, contribute to food security and poverty.

1.5 Problem Specification:

The important aspect of Agriculture is pest management. If the farmer has best strategies

Early detection of pests, their crop would be more productive. When a crop is cultivated

Protective structures, it is proposed to use Image sensors to monitor crops. The camera captures the crop images and exports the captured images to the nearest Data Centre or cloud for further analysis. The received image would be pre-processed and analysed.

1. Image Sensor Interfacing to Cloud module: In this module Image sensor i.e. Camera of the mobile device used to capture the crop images and send to the agro-cloud system.

2. Data Centre: This is the cloud system to store the captured pest image data, processing of the image data to identify the pests present in the images and classify the identified pests and store the results.

3. User interface module: User interface from the Users to the pest management cloud so that user can access the pest management service and get the advisory information based on the provided input image.

1.6 Domain Analysis:

Agriculture in India:

At present, India holds the second position in the world in agricultural production. Agriculture plays an important role in the Indian economy. Over 70% of the rural households depend on agriculture. Agriculture employed 60% of the Indian work force and contributed 17-18% to country's GDP. The Indian food industry is poised for huge growth, increasing its contribution to world food trade every year due to its immense potential for value addition. The population in India is increasing at a high rate and puts pressure on the agriculture sector. For this, additional land is required but due to rapid growth in urban areas, agricultural land is converted into non-agricultural use.

Plant diseases:

Any disturbance that interferes with normal growth, development, function, economic value, quality of a plant is defined as plant diseases 2 types:

1Abiotic plant diseases (Non-living factors)

2. Biotic plant diseases (living factors)

1. Tomato Late Blight:

Symptoms:

Brown spots grow on the leaf margins.
White covering on the underside of the leaves.
Grey or brown wrinkled stains on fruits.
Hardened fruit flesh and fruit decay.

Explanation:

During wet weather, lesions on the lower side of the leaves may be covered with a grey to white-moldy growth, making it easier to distinguish healthy from dead leaf tissue. As the disease progresses, the foliage runs brown, curls and dries. Risk of infection is highest in mid-summer. The fungus enters the plants via wounds and rips the skin.

2.Septoria Leaf Spot

Symptoms:

Small grey circular spots with dark-brown appears.

Black dots appear in their centers.

Leaves turn slightly yellow, wither and fall off.

Explanation:

The symptoms spread upwards from the oldest to youngest growth. Small, water-soaked grey circular spots with dark-brown margins appear on the undersides of older leaves. At later stages of the disease, the spots enlarge and coalesce, and black dots appear in their centers. The same pattern may also be observed on stems and blossoms, but rarely on fruit. Heavily infected leaves will turn slightly yellow, wither, and fall off. The defoliation leads to the sun scalding of fruit.

1. Potato Late Blight:

Symptoms:

Dark brown spots develop on tips and margins of leaves.

White fungal covering grows on the underside of leaf blades.

Leaves become necrotic and die off.

Grayish-blue spots on potato tubers make them unfit to eat.

Explanation:

Dark brown spots develop on the leaves starting at the tip or the leaf margins. In humid climates, these spots become water-soaked lesions. A white fungal covering can be seen on the underside of the leaves. As the disease progresses, entire leaves become necrotic, turn brown and die off. Similar lesions develop on stems and petioles. The potato tubers have greyish-blue spots on their skin and flesh also turns brown, making them inedible. The rotting of the infested fields give off a distinctive smell.

Pest Management

Biological parameters such as pests and plant diseases which are affect the quality and quantity of agriculture production. Even we maintain agro-climatic conditions in poly-houses but sensitive with pests and plant diseases.

The warm, humid condition present in poly-house provides an excellent environment for pest development. Often, the natural enemies present in open field keep pests under control but those are not present in poly-houses. For these reasons, pest situations often develop rapidly in poly-houses. The growing conditions within the poly-house are highly favorable to arthropod pests. In India, about twenty insect and mite species have been recorded to be associated with the crops under protected environment. Some of the important pest groups are aphids, caterpillars, leaf miner, mites, thrips and whiteflies. The level of damage caused by pests tolerated up to some extent but it greatly influences the productivity of crop. The commonly observed pests in poly-houses are shown in below.

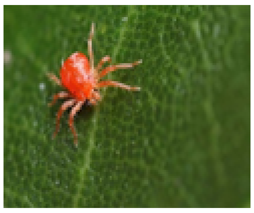


Figure 1.6.1 Mites

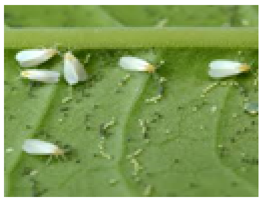


Figure 1.6.2 White Flies

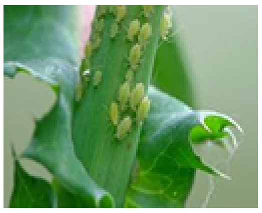


Figure 1.6.3 Aphids

Pest Management requires much more than a "see and spray" approach – use pesticides only when essential. Integrated Pest Management is a systematic approach to manage pests that combines a variety of techniques and strategies to reduce populations. The IPM describes as a pyramid strategy which is having three components namely, Avoidance of problem, early detection and curative measures. In avoidance of problem we make use of Insect-proof screens can greatly limit the movement of pests into the poly-house.

Early detection process monitors the crop regularly by a systematic inspection of the plants and its exteriors for identifying and assessing the pest problems occurred. It makes use of insect traps to early detection process. This timely crop monitoring identifies the situation of the crop. But doing it regularly with manual observation is time consuming and also there is a chance of missing some of the pest observation.

Integrated Pest Management relies on the accuracy of pest population monitoring techniques. Traditional pest monitoring technique insect traps is labour intensive and offer poor temporal resolution density in the field cannot be accurately monitored. With these reasons the use of image sensor network technologies to perform automatic pest monitoring.

1.7 Dataset Preparation:

A dataset is a collection of data. A dataset corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the data set. But here we have taken the image dataset of diseased plant leaf images of Potato and tomato.

CHAPTER-2

LITERATURE SURVEY

2. LITERATURE SURVEY

1. **Konstantinos P. Ferentinos, “Deep learning models for plant disease detection and diagnosis”, Computers and Electronics in Agriculture, 145 (2018), 311-318.**

   In this paper, convolutional neural network models were developed to perform plant disease detection and diagnosis using simple leaves images of healthy and diseased plants, through deep learning methodologies. Training of the models was performed with the use of an open database of 87,848 images, containing 25 different plants in a set of 58 distinct classes of [plant, disease] combinations, including healthy plants. Several model architectures were trained, with the best performance reaching a 99.53% success rate in identifying the corresponding [plant, disease] combination (or healthy plant). The significantly high success rate makes the model a very useful advisory or early warning tool, and an approach that could be further expanded to support an integrated plant disease identification system to operate in real cultivation conditions.

2. **Diego InácioPatrícioa, Rafael Riederb, “Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review”,Computers and Electronics in Agriculture,153 (2018), 69-81.**

   Grain production plays an important role in the global economy. In this sense, the demand for efficient and safe methods of food production is increasing. Information Technology is one of the tools to that end. Among the available tools, this paper highlights computer vision solutions combined with artificial intelligence algorithms that achieved important results in the detection of patterns in images. In this context, this work presents a systematic review that aims to identify the applicability of computer vision in precision agriculture for the production of the five most produced grains in the world: maize, rice, wheat, soybean, and barley. In this sense, and present some papers selected in the last five years with different approaches to treat aspects related to disease detection, grain quality, and phenotyping. From the results of the systematic review, it is possible to identify great opportunities, such as the exploitation of GPU (Graphics Processing Unit) and advanced artificial intelligence techniques, such as DBN (Deep Belief Networks) in the construction of robust methods of computer vision applied to precision agriculture.

3. **Ryan H.L. Ip, Li-Minn Ang,KahPhooi Senga, J.C. Brosterb, J.E. Pratley "Big data and machine learning for crop protection", Computers and Electronics in Agriculture, 51 (2018), 376-383.**

Crop protection is the science and practice of managing plant diseases, weeds and other pests. Weed management and control are important given that crop yield losses caused by pests and weeds are high.This paper first presents a brief review of some significant research efforts in crop protection using big data with the focus on weed control and management followed by some potential applications. Some machine learning techniques for big data analytics are also reviewed. The outlook for big data and machine learning in crop protection is very promising. The potential of using Markov random fields (MRF) which takes into account the spatial component among neighboring sites for herbicide resistance modeling of ryegrass is then explored. To the best of our knowledge, no similar work of modeling herbicide resistance using the MRF has been reported. Experiments and data analytics have been performed on data collected from farms in Australia. Results have revealed the good performance of our approach.

4. **G.M. Pasqual and J. Mansfield, "Development of a Prototype Expert System for Identification and Control of Insect Pests", Computers and Electronics in Agriculture, 2 (1988) 263-276.**

A prototype, rule-based expert system (PEST - Pest Expert System) was developed to evaluate how knowledge engineering techniques may be used to provide insect identification and control advice to farmers and extension workers in Western Australia. PEST uses expert system techniques to ask questions and gives recommendations and advice in a manner akin to an agricultural entomologist. This paper describes the types of decisions made and the knowledge engineering tasks undertaken during development of PEST. The methods used to acquire and represent entomological knowledge in the form of facts, rules and relationships are explained. The reasoning mechanisms employed by the system to interpret and apply rules are also discussed and an example of a session with the PEST program is presented. It is concluded that the problem of identifying insect pests and recommending control strategies is suitable for resolution by a rule-based expert system because the knowledge is of a shallow or empirical nature, the problem domain is of a manageable size, the solutions are well defined and solving the problem requires heuristic solution and the application of expertise that few people possess. PEST was implemented in LPA MacProlog TM using the University of Edinburgh standard on the Macintosh plus TM microcomputer.

5. **Javier Lacasta, F. Javier Lopez-Pellicer, Borja Espejo-García, Javier Nogueras-Iso, F. Javier Zarazaga-Soria, "Agricultural recommendation system for crop protection", 152 (2018), 82-89.**

Pests in crops produce important economic loses all around the world. To deal with them without damaging people or the environment, governments have established strict legislation and norms describing the products and procedures of use. However, since these norms frequently change to reflect scientific and technological advances, it is needed to perform a frequent review of affected norms in order to update pest related information systems. This is not an easy task because they are usually human-oriented, so intensive manual labour is required. To facilitate the use of this information, this work proposes the construction of a recommendation system that facilitates the identification of pests and the selection of suitable treatments. The core of this system is an ontology that models the interactions between crops, pests and treatments

CHAPTER 3
DESIGN AND TECHNICAL DESCRIPTION

3.1 Deep Learning:

3.1.1 Brief Theory: Deep learning (also known as deep structured learning, hierarchical learning or deep machine learning) is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing 1, with complex structures or otherwise, composed of multiple non-linear transformations.

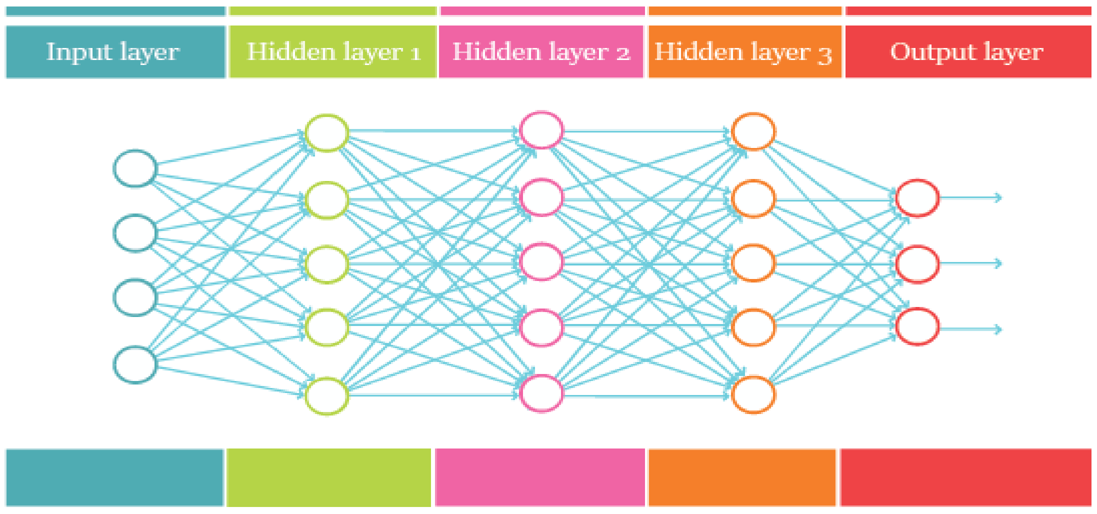


Figure 3.1.1.1 Brief Theory on Deep Learning

In Practice, defining the term "Deep": in this context, deep means that we are studying a Neural Network which has several hidden layers (more than one), no matter what type (convolutional, pooling, normalization, fully-connected etc). The most interesting part is that some papers noticed that Deep Neural Networks with the right architectures/hyper-parameters achieve better results than shallow Neural Networks with the same computational power (e.g. number of neurons or connections).

In Practice, defining "Learning": In the context of supervised learning, for example consider digits recognition in our case, the learning part consists of a target/feature which is to be predicted using a given set of observations with the already known final prediction (label). In our case, the target will be the digit (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) and the observations are the intensity and relative position of the pixels. After some training, it is possible to generate a "function" that map inputs (digit image) to desired outputs (type of digit). The only problem is how well this map operation occurs. While trying to generate this "function", the training process continues until the model achieves a desired level of accuracy on the training data.

3.1.2 Analogies

There are several ways to understand Convolutional Layers without using a mathematical approach. We are going to explore some of the ideas proposed by the Machine Learning community.

3.1.3 Instances of Neurons

When you start to learn a programming language, one of the first phases of your development is the learning and application of functions. Instead of rewriting pieces of code every time that you would, a good student is encouraged to code using functional programming, keeping the code organized, clear and concise. CNNs can be thought of as a simplification of what is really going on, a special kind of neural network which uses identical copies of the same neuron. These copies include the same parameters (shared weights and biases) and activation functions**.**

3.1.4 Location and type of connections

In a fully connected layer NN, each neuron in the current layer is connected to every neuron in the previous layer, and each connection has it's own weight. This is a general-purpose connection pattern and makes no assumptions about the features in the input data thus not taking any advantage that the knowledge of the data being used can bring. These types of layers are also very expensive in terms of memory and computation.

In contrast, in a convolutional layer each neuron is only connected to a few nearby local neurons in the previous layer, and the same set of weights is used to connect to them. For example, in the following image, the neurons in the h1 layer are connected only to some input units (pixels).

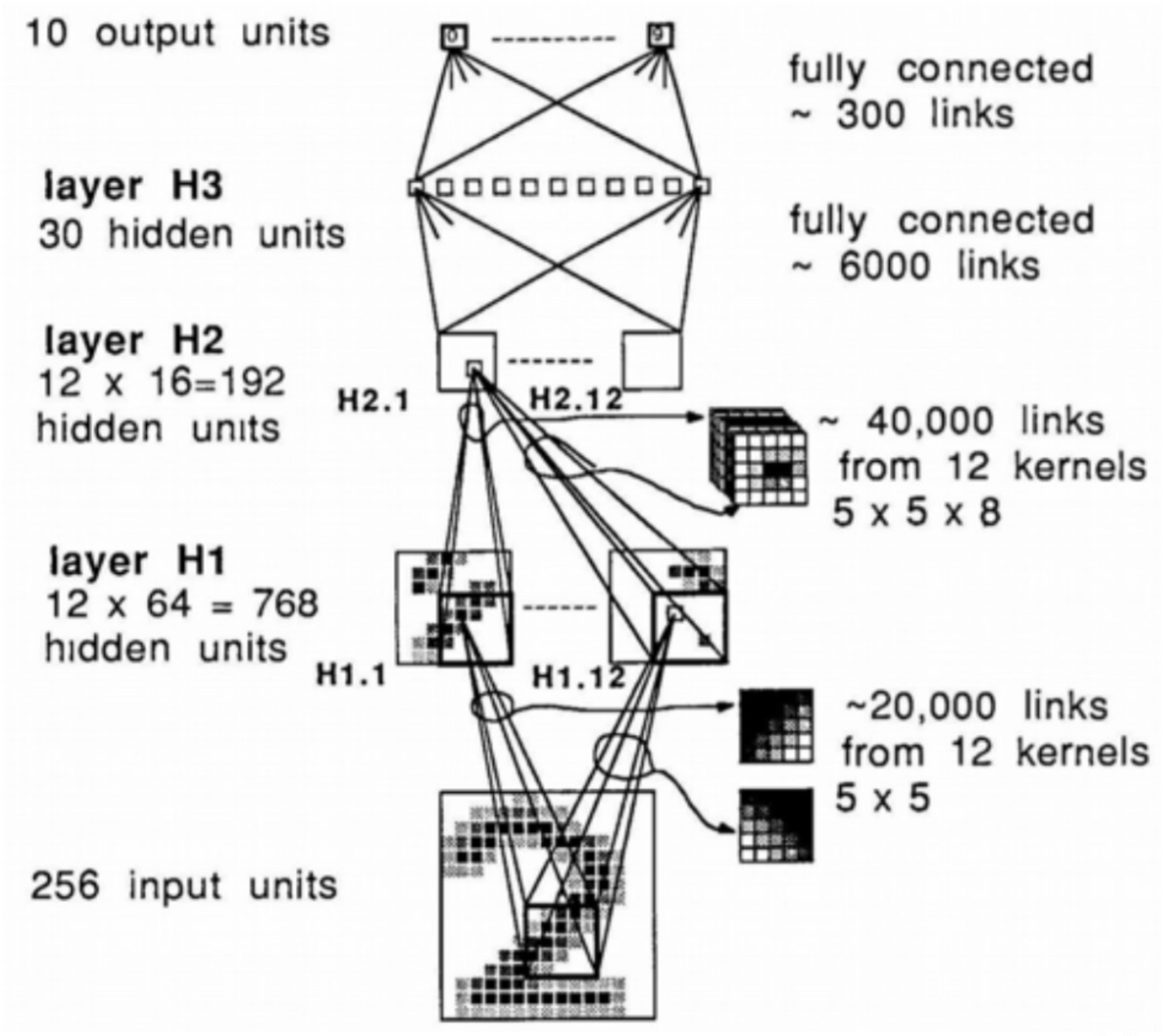

10 output units
fully connected ~ 300 links
layer H3 30 hidden units
fully connected ~ 6000 links
layer H2 12 x 16=192 hidden units
H2.1
H2.12
~ 40,000 links from 12 kernels 5 x 5 x 8
layer H1 12 x 64 = 768 hidden units
H1.1
H1.12
~20,000 links from 12 kernels 5 x 5
256 input units

Figure 3.1.4.1 Location and Types of Connections

A figure presented in one of Yann LeCun's papers. It shows the spatial relation and how the connections are modified until the output layer.

3.1.5 Feature Learning

Feature engineering is the process of extracting useful patterns from input data that will help the prediction model to understand better the real nature of the problem. A good feature learning will present patterns in a way that significantly increase the accuracy and performance of the applied machine learning algorithms in a way that would otherwise be impossible or too expensive by just machine learning itself.Feature learning algorithms finds the common patterns that are important to distinguish between the wanted classes and extract them automatically. After this process, they are ready to be used in a classification or regression problem.

The great advantage of CNNs is that they are uncommonly good at finding features in images that grow after each level, resulting in high-level features in the end. The final layers (can be one or more) use all these generated features for classification or regression.

Basically, Convolutional Neural Networks is your best friend to **automatically do Feature Engineering** (Feature Learning) without wasting too much time creating your own codes and with no prior need of expertise in the field of Feature Engineering.

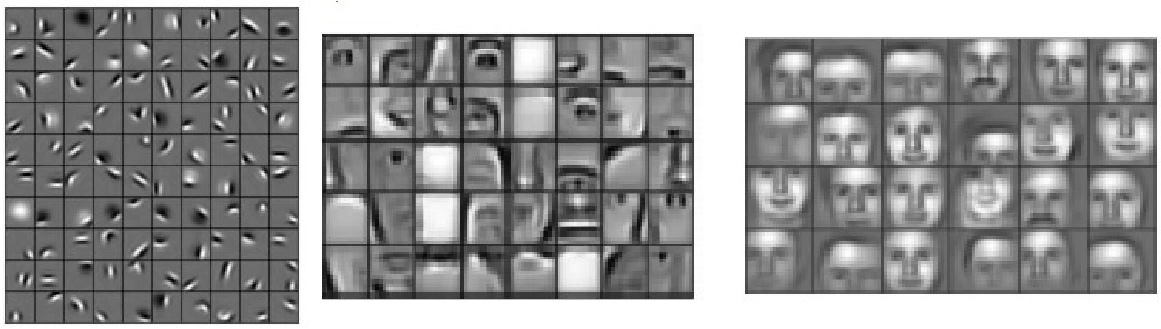


Figure 3.1.5.1 Feature Learning

Example of feature learning (automatically feature engineering), starting with simple features and ending with high-level features like human faces.

3.1.6 Image Filter

Creation of a convolved feature from of an image

The image below is a 8x8 matrix of an image's pixels, converted to binary values in the next image (left), where 1 means a white pixel and 0 a black pixel. Later we will find out that typically this is a normalization, these values can actually have different scales. The most common usage is values between 0 and 255 for 8-bit grayscale images.

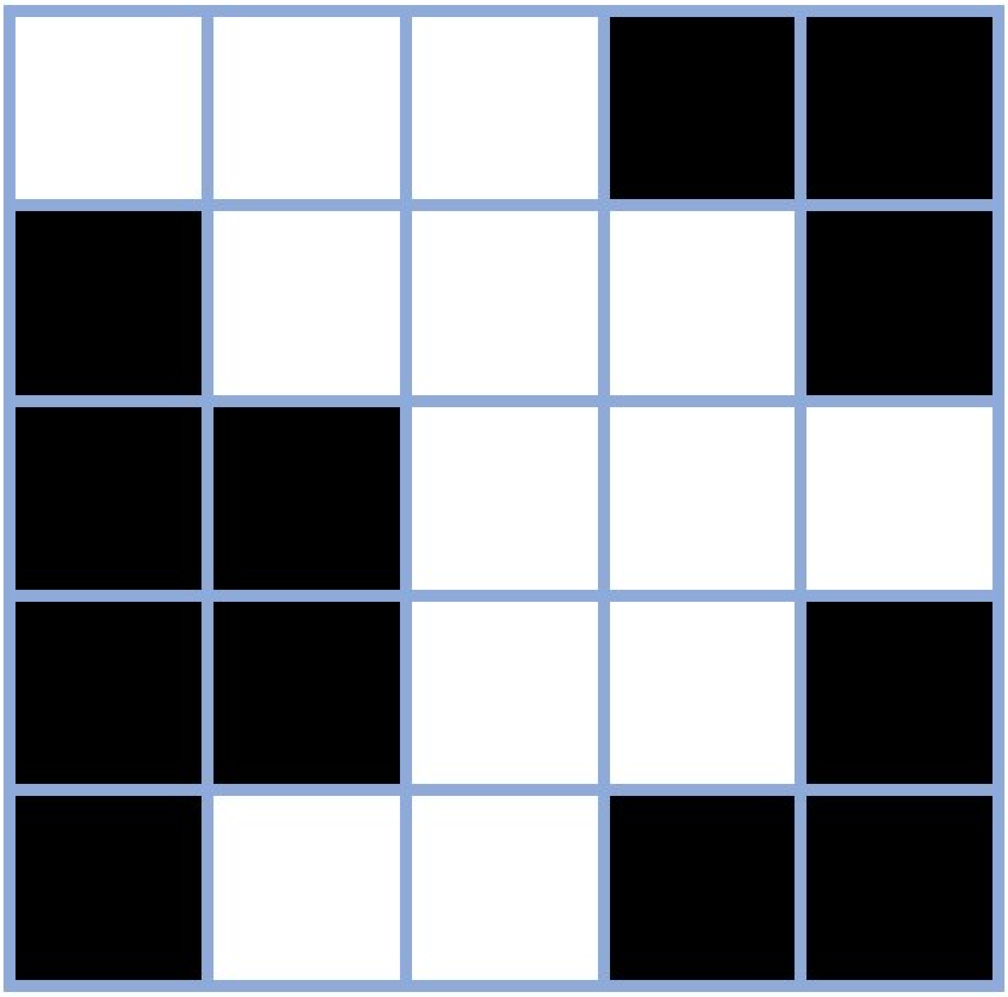


Figure 3.1.6.1 Image Filterig

An example of a low resolution image to be recognized.

In the below image, with an animation, you can see how the two-dimensional convolution operation would operate on the images. This operation is performed in most of the Deep Learning frameworks in their first phase. We need a sliding windows to create the convolved matrix:

The sliding window (a.k.a kernel, filter or feature detector) with a preset calculation ([[x1, x0,x1], [x0,x1,x0], [x1,x0,x1]]) goes through the image and creates a new matrix (feature map).

Animations showing how a kernel interact with a matrix representing an image.

In the example above we used a 3×3 filter (5x5 could also be used, but would be too complex). The values from the filter were multiplied element-wise with the original matrix (input image), then summed up. To get the full convolved matrix, the algorithm keep repeating this small procedure for each element by sliding the filter over the whole original matrix.

Illustration of the operation for one position of the kernel.

Just like the referenced example, we can think of a one-dimensional convolution as sliding function (1x1 or 1x2 filter) multiplying and adding on top of an array (1 dimensional array, instead of the original matrix).

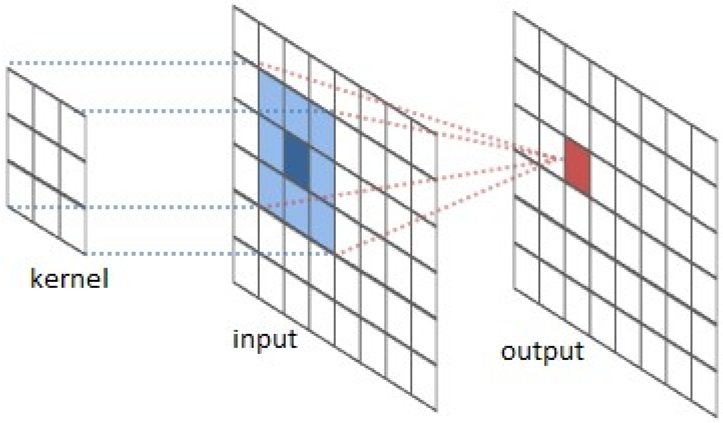
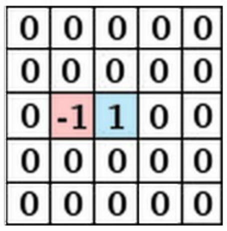


Figure 3.1.6.2 Kernel Matrix

Lets try another kernel:

Taking the values −1 and 1 on two adjacent pixels and zero everywhere else for the kernel, results in the following image. That is, we subtract two adjacent pixels. When side by side pixels are similar, this gives us approximately zero. On edges, however, adjacent pixels are very different in the direction perpendicular to the edge. Knowing that results differs from zero will result in brighter pixels, you can already guess the result of this type of kernel.



0	0	0	0	0
0	0	0	0	0
0	-1	1	0	0
0	0	0	0	0
0	0	0	0	0

Understanding the imported data

The imported data can be divided as follow:

Training: Use the given dataset with inputs and related outputs for training of NN. In our case, if you give an image that you know that represents a "nine", this set will tell the neural network that we expect a "nine" as the output.

Validation: The same as training, but now the data is used to generate model properties (classification error, for example) and from this, tune parameters like the optimal number of hidden units or determine a stopping point for the back-propagation algorithm.

Test: The model does not have access to this information prior to the testing phase. It is used to evaluate the performance and accuracy of the model against "real life situations". No further optimization beyond this point.

Creating an interactive section

You have two basic options when using TensorFlow to run your code:

- [Build graphs and run session] Do all the set-up and THEN execute a session to evaluate tensors and run operations (ops)
- [Interactive session] create your coding and run on the fly.

For this first part, we will use the interactive session that is more suitable for environments like Jupyter notebooks.

Creating placeholders

It is a best practice to create placeholders before variable assignments when using TensorFlow. Here we'll create placeholders for inputs ("Xs") and outputs ("Ys").

Placeholder 'X': represents the "space" allocated input or the images.

Placeholder 'Y': represents the final output or the labels.

dtype for both placeholders:

if you not sure, use tf.float32. The limitation here is that the later presented softmax function only accepts float32 or float64 dtype's. For more dtypes, check TensorFlow documentation

Assigning bias and weights to null tensors

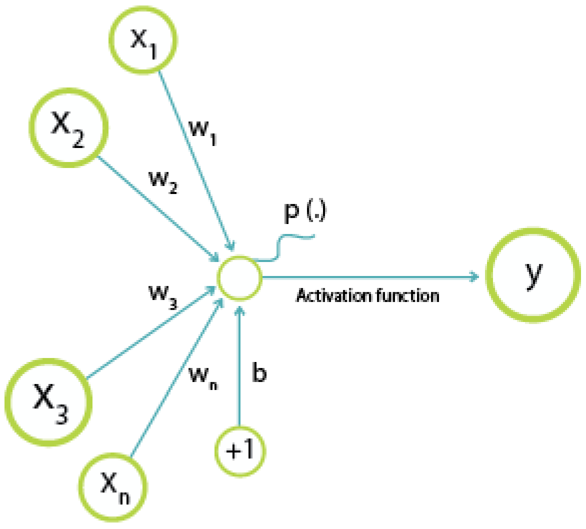
Now we are going to create the weights and biases, for this purpose they will be used as arrays filled with zeros. The values that we choose here can be critical, but we'll cover a better way on the second part, instead of this type of initialization.

Execute the assignment operation

Before, we assigned the weights and biases but we did not initialize them with null values. For this reason, TensorFlow need to initialize the variables that you assign.

Adding Weights and Biases to input

The only difference for our next operation to the picture below is that we are using the mathematical convention for what is being executed in the illustration. The tf.matmuloperation performs a matrix multiplication between x (inputs) and W (weights) and after the code add biases.



3.1.7 Softmax Regression

Softmax is an activation function that is normally used in classification problems. It generates the probabilities for the output. For example, our model will not be 100% sure that one digit is the number nine, instead, the answer will be a distribution of probabilities where, if the model is right, the nine number will have a larger probability than the other digits.

For comparison, below is the one-hot vector for a nine digit label:

0 --> 0

1 --> 0

2 --> 0

3 --> 0

4 --> 0

5 --> 0

6 --> 0

7 --> 0

8 --> 0

9 --> 1

A machine does not have all this certainty, so we want to know what is the best guess, but we also want to understand how sure it was and what was the second better option. Below is an example of a hypothetical distribution for a nine digit:

0 -->0.01

1 -->0.02

2 -->0.03

3 -->0.02

4 -->0.12

5 -->0.01

6 -->0.03

7 -->0.06

8 -->0.1

9 -->0.6

Similarly, it provides the probability of image given as an input in our model.

Logistic function output is used for the classification between two target classes 0/1. Softmax function is generalized type of logistic function. That is, Softmax can output a multiclass categorical probability distribution.

It is a function that is used to minimize the difference between the right answers (labels) and estimated outputs by our Network.

Type of optimization: Gradient Descent

This is the part where you configure the optimizer for your Neural Network. There are several optimizers available, in our case we will use Gradient Descent because it is a well-established optimizer.

Training batches

Train using mini batch Gradient Descent.

In practice, Batch Gradient Descent is not often used because is too computationally expensive. The good part about this method is that you have the true gradient, but with the expensive computing task of using the whole dataset in one time. Due to this problem, Neural Networks usually use minibatch to train.

To improve our model

Several options as follow:

- Regularization of Neural Networks using Drop Connect
- Multi-column Deep Neural Networks for Image Classification
- APAC: Augmented Pattern Classification with Neural Networks
- Simple Deep Neural Network with Dropout

Converting images of the data set to tensors

The input image is 28 pixels by 28 pixels, 1 channel (grayscale). In this case, the first dimension is the **batch number** of the image, and can be of any size (so we set it to -1). The second and third dimensions are width and height, and the last one is the image channels.

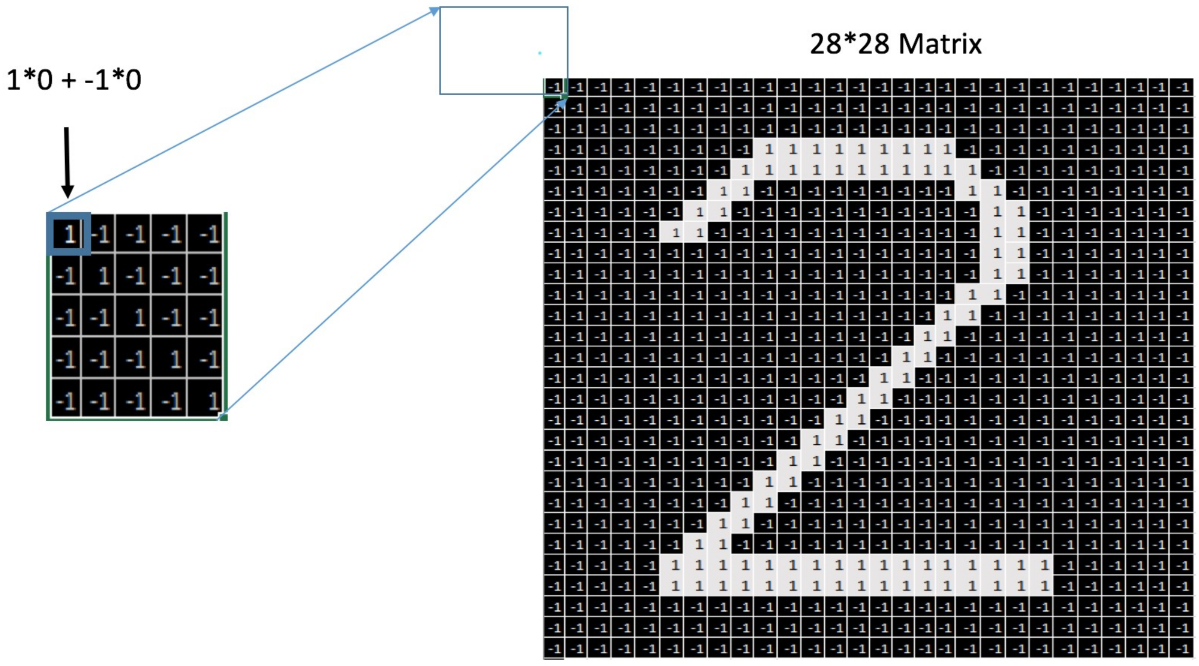
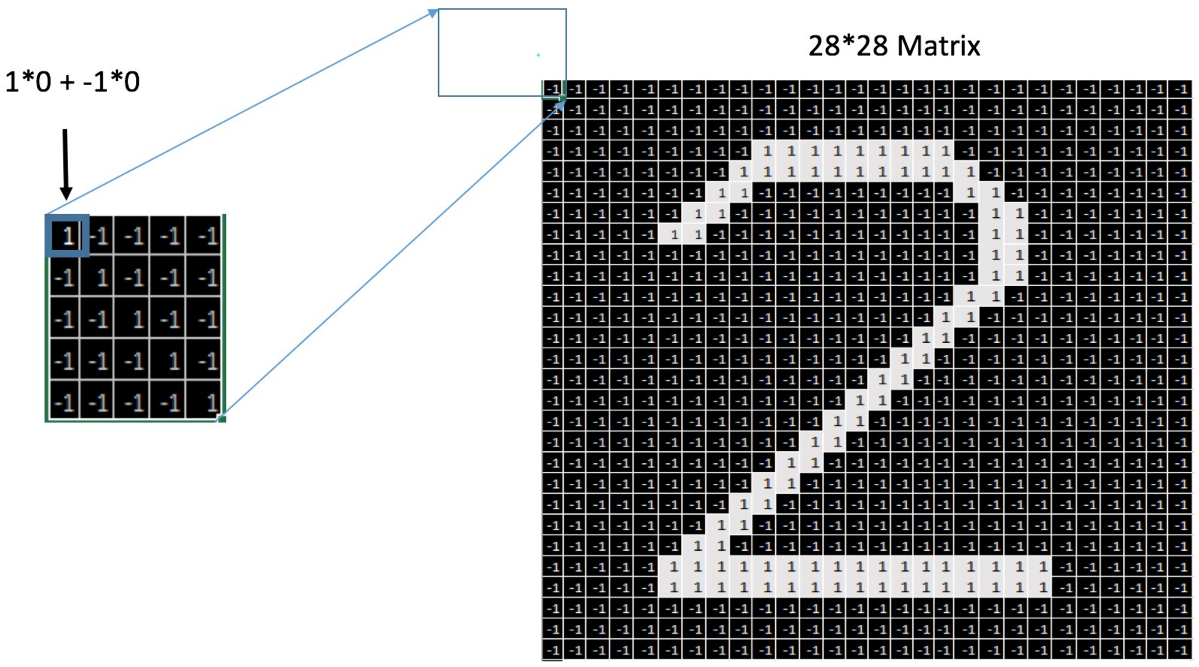


Figure 3.1.7.1 Converting images of the data set to tensors

Convolutional Layer 1

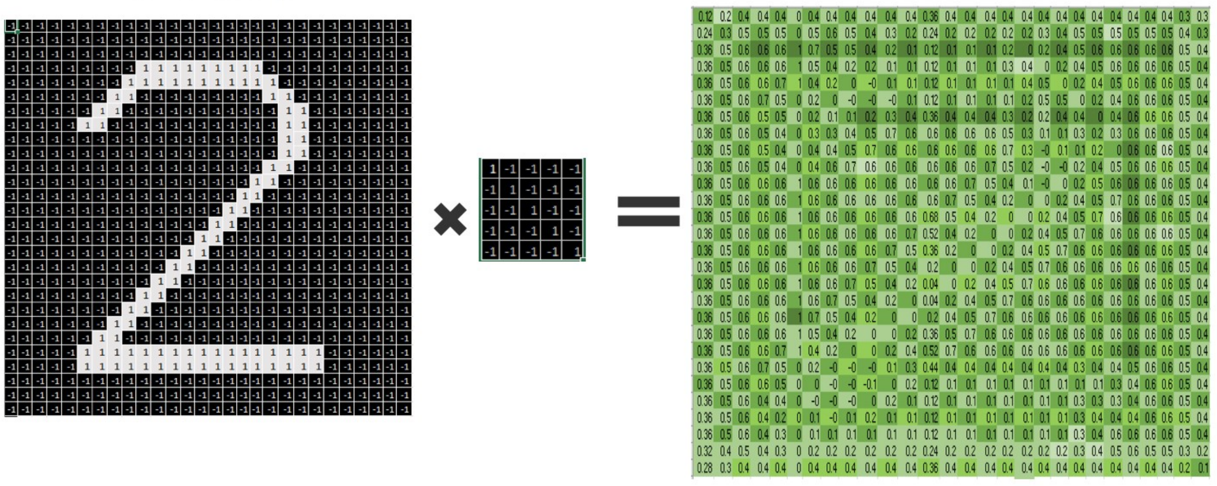
Defining kernel weight and bias

We define a kernel here. The Size of the filter/kernel is 5x5; Input channels is 1 (grayscale); and we need 32 different feature maps (here, 32 feature maps means 32 different filters are applied on each image. So, the output of convolution layer would be 28x28x32). In this step, we create a filter / kernel tensor of shape [filter height, filter width, in_channels, out channels]



Convolve with weight tensor and add biases.

To create convolutional layer, we use **tf.nn.conv2d**. It computes a 2-D convolution given 4-D input and filter tensors.



Apply the ReLU activation Function

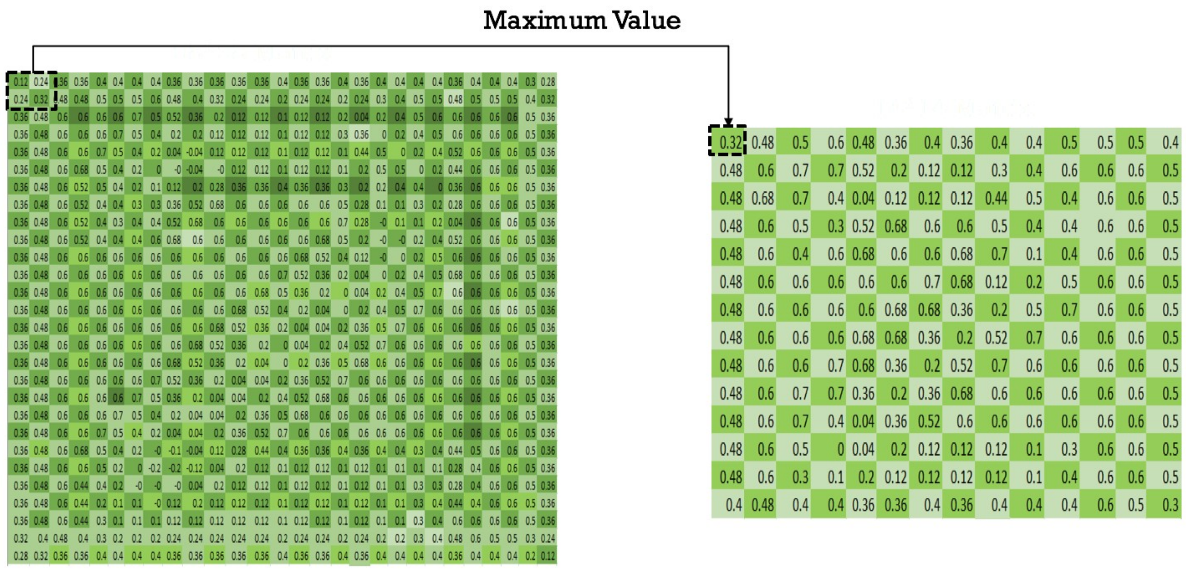
In this step, we just go through all outputs convolution layer, **convolve 1**, and wherever a negative number occurs, we swap it out for a 0. It is called ReLU activation Function.

Apply the max pooling

max pooling is a form of non-linear down-sampling. It partitions the input image into a set of rectangles and, and then find the maximum value for that region.

Let's use **tf.nn.max_pool** function to perform max pooling. **Kernel size:** 2x2 (if the window is a 2x2 matrix, it would result in one output pixel)

Strides: dictates the sliding behavior of the kernel. In this case it will move 2 pixels every time, thus not overlapping. The input is a matrix of size 28x28x32, and the output would be a matrix of size 14x14x32.



Convolutional Layer 2

Weights and Biases of kernels

We apply the convolution again in this layer. Lets look at the second layer kernel:

- Filter/kernel: 5x5 (25 pixels)
- Input channels: 32 (from the 1st Conv layer, we had 32 feature maps)
- 64 output feature maps

Notice: here, the input image is [14x14x32], the filter is [5x5x32], we use 64 filters of size [5x5x32], and the output of the convolutional layer would be 64 convolved image, [14x14x64].

Notice: the convolution result of applying a filter of size [5x5x32] on image of size

[14x14x32] is an image of size [14x14x1], that is, the convolution is functioning on volume.

Convolve image with weight tensor and add biases.

Apply the ReLU activation Function

Apply the max pooling

Fully Connected Layer

You need a fully connected layer to use the Softmax and create the probabilities in the end. Fully connected layers take the high-level filtered images from previous layer, that is all 64 matrices, and convert them to a flat array.

So, each matrix [7x7] will be converted to a matrix of [49x1], and then all of the 64 matrix will be connected, which make an array of size [3136x1]. We will connect it into another layer of size [1024x1]. So, the weight between these 2 layers will be [3136x1024],second layer should be flattened.

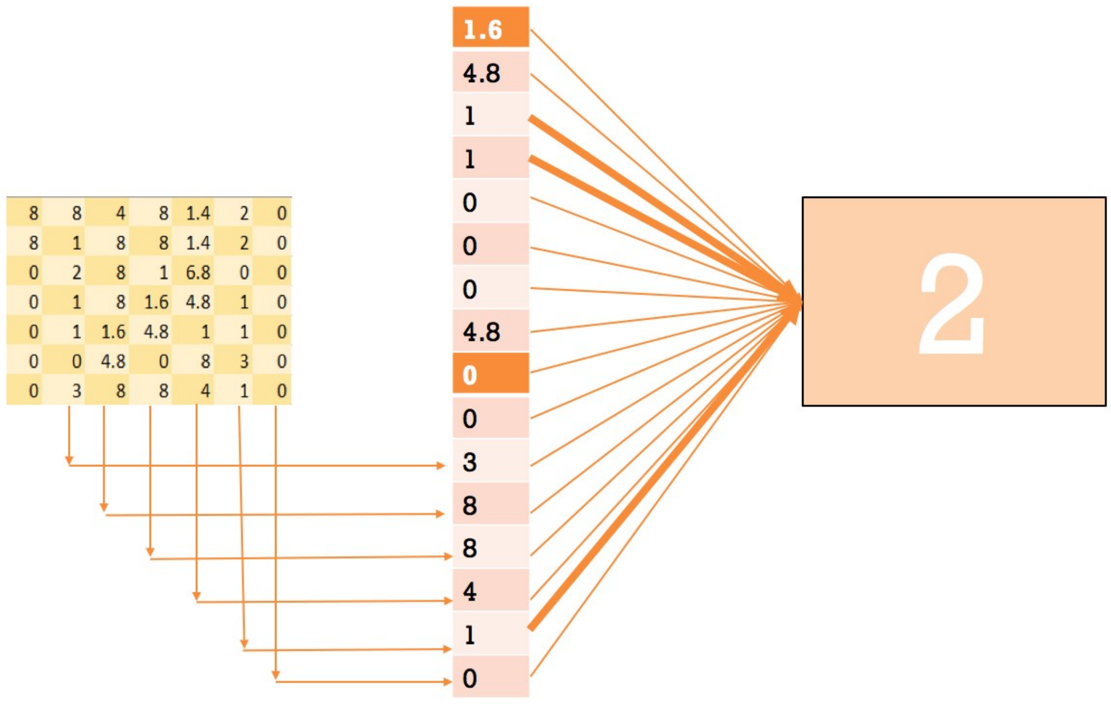


Figure 3.1.7.2 Fully Connected Layer

Summary of the Deep Convolutional Neural Network

Finally, structure of our network

0) Input

1) Convolutional and Max-Pooling

2) Convolutional and Min-Pooling

3) Fully Connected Layer

4) Processing - Dropout

5) Readout layer - Fully Connected

6) Outputs

Some functions to train the model:

Optimizer:

It is obvious that we want minimize the error of our network which is calculated by cross_entropy metric. To solve the problem, we have to compute gradients for the loss (which is minimizing the cross-entropy) and apply gradients to variables. It will be done by an optimizer: Gradient Descent.

prediction

To know how many of the cases in a mini-batch has been classified correctly and to predict the output.

accuracy

It makes more sense to report accuracy using average of correct cases.

CHAPTER:4

IMPLEMENTATION

4.1 ENVIRONMENT

After downloading the dataset, if we run the code on local machine for training we found some difficulty with system configuration so I decided to run it on the kaggle kernel. I was decided to try it out because it was easy to use, necessary packages had been installed already. Since my dataset was not available on Kaggle I had to upload.

If you won't be using Kaggle Kernel you will have to install this package

Numpy :

A library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. (Source: Wikipedia)

Sklearn :

A free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. (Source: Wikipedia)

Keras :

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, or Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. (Source: Wikipedia)

Matplotlib :

A plotting library for the Python programming language and its numerical mathematics extension.

After installing these libraries in our local machine, we need to build our model and run it on the local machine as per our requirement of the solution.

4.2 Sample Code:

```
# Here we will import the required libraries to implement the model
import numpy as np
import pickle
import cv2
from os import listdir
from sklearn.preprocessing import LabelBinarizer
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

```
EPOCHS=25
INIT_LR=1e-3
BS=32
default_image_size=tuple((256,256))
image_size=0
directory_root='../input/plantvillage/'
width=256
height=256
depth=3
```

#I converted each image to an array using the function below

```
defconvert_image_to_array(image_dir):
try:
image=cv2.imread(image_dir)
ifimageisnotNone:
image=cv2.resize(image,default_image_size)
returnimg_to_array(image)
else:
returnnp.array([])
exceptExceptionase:
print(f"Error : {e}")
returnNone
#here we have uploaded the diseased plant images dataset
#here in this model I have given 200 images from each folder
image_list,label_list=[],[]
try:
print("[INFO] Loading images ...")
root_dir=listdir(directory_root)
fordirectoryinroot_dir:
# remove .DS_Store from list
ifdirectory==".DS_Store":
root_dir.remove(directory)
forplant_folderinroot_dir:
plant_disease_folder_list=listdir(f"{directory_root}/{plant_folder}")
fordisease_folderinplant_disease_folder_list:
# remove .DS_Store from list
ifdisease_folder==".DS_Store":
plant_disease_folder_list.remove(disease_folder)
forplant_disease_folderinplant_disease_folder_list:
print(f"[INFO] Processing {plant_disease_folder} ...")
plant_disease_image_list=listdir(f"{directory_root}/{plant_folder}/{plant_disease_folde
r}/")
forsingle_plant_disease_imageinplant_disease_image_list:
ifsingle_plant_disease_image==".DS_Store":
plant_disease_image_list.remove(single_plant_disease_image)
forimageinplant_disease_image_list[:200]:
image_directory=f"{directory_root}/{plant_folder}/{plant_disease_folder}/{image}"
ifimage_directory.endswith(".jpg")==Trueorimage_directory.endswith(".JPG")==True:
image_list.append(convert_image_to_array(image_directory))
```

```
label_list.append(plant_disease_folder)
print("[INFO] Image loading completed")
exceptExceptionase:
print(f"Error : {e}")
image_size=len(image_list)
```

#Using Scikit-learn's Label Binarizer , I converted each image label to binary levels. Then I saved the label binarizer instance using pickle after which I printed the classes from the label binarizer.

```
label_binarizer=LabelBinarizer()
image_labels=label_binarizer.fit_transform(label_list)
pickle.dump(label_binarizer,open('label_transform.pkl','wb'))
n_classes=len(label_binarizer.classes_)
#printing the classes of labels
print(label_binarizer.classes_)
```

#I further pre-process the input data by scaling the data points from [0, 255] (the minimum and maximum RGB values of the image) to the range [0, 1].

```
np_image_list=np.array(image_list,dtype=np.float16)/225.0
```

# Then we performed a training/testing split on the data using 80% of the images for training and 20% for testing.

```
print("[INFO] Spliting data to train, test")
x_train,x_test,y_train,y_test=train_test_split(np_image_list,image_labels,test_size=0.2,random_state=42)
```

#now i created an image generator object which performs random rotations, shifts, flips, crops, and sheers on our image dataset. This allows us to use a smaller dataset and still achieve high results.

```
aug=ImageDataGenerator(
rotation_range=25,width_shift_range=0.1,
height_shift_range=0.1,shear_range=0.2,
zoom_range=0.2,horizontal_flip=True,
fill_mode="nearest")
```

#Next I created my model. In the model we are defaulting to "channel_last" architecture but also creating a switch for backend that support "channel_first" on the fourth line.

#Then I created the first CONV => RELU => POOL. Our CONV layer has 32 filters with a 3 x 3 kernel and RELU activation (Rectified Linear Unit).

#we apply batch normalization, max pooling, and 25% (0.25) dropout.

***#Dropout** is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. (Source : Wikipedia)*

#Next I created two sets of (**CONV** => **RELU**) * 2 => **POOL** blocks. Then only one set of **FC** (Fully Connected Layer)=> **RELU** layers

```
model=Sequential()
inputShape=(height,width,depth)
chanDim=-1
ifK.image_data_format()=="channels_first":
inputShape=(depth,height,width)
chanDim=1
model.add(Conv2D(32,(3,3),padding="same",input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3,3)))
model.add(Dropout(0.25))
model.add(Conv2D(64,(3,3),padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64,(3,3),padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Conv2D(128,(3,3),padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128,(3,3),padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(n_classes))
model.add(Activation("softmax"))
```

*#I used the Keras Adam Optimizer for my model. Training our network is initiated in **Cell 14** where we call **model.fit_generator** , supplying our data augmentation object, training/testing data, and the number of epochs we wish to train for. I used an **epoch's value of 25** for this project.*

```
opt=Adam(lr=INIT_LR,decay=INIT_LR/EPOCHS)
# distribution
model.compile(loss="binary_crossentropy",optimizer=opt,metrics=["accuracy"])
# train the network
print("[INFO] training network..."
history=model.fit_generator(
aug.flow(x_train,y_train,batch_size=BS),
validation_data=(x_test,y_test),
steps_per_epoch=len(x_train)//BS,
epochs=EPOCHS,verbose=1
)
acc=history.history['acc']
val_acc=history.history['val_acc']
loss=history.history['loss']
val_loss=history.history['val_loss']
epochs=range(1,len(acc)+1)
#Train and validation accuracy
plt.plot(epochs,acc,'b',label='Training accurarcy')
plt.plot(epochs,val_acc,'r',label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()
plt.figure()
#Train and validation loss
plt.plot(epochs,loss,'b',label='Training loss')
plt.plot(epochs,val_loss,'r',label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
#I calculated the accuracy of my model using the validation data (x_test and y_test)
created earlier. I got an accuracy score of 96.77 %.
print("[INFO] Calculating model accuracy")
scores=model.evaluate(x_test,y_test)
print(f"Test Accuracy: {scores[1]*100}")
# save the model to disk
print("[INFO] Saving model...")
pickle.dump(model,open('cnn_model.pkl','wb'))
```

To deploy our trained model for use via an API, we would do something similar to the following:

- Load our trained model
- Accept incoming data and pre-process it
- Predict using our loaded model
- Handling the prediction output.
- Make the model available for use via an API. There are many ways in which we can make a model available via an API. Some of them include :

1.Custom REST-API with Django or Flask: In this case we build a custom REST-API with either one of Django or Flask.

2. Tensor flow: we could also use Tensor flow to deploy our machine learning model using Tensor flow serving.

3.AWS Lambda/Server less: this involves the use of AWS Lambda to make your deep learning model available.

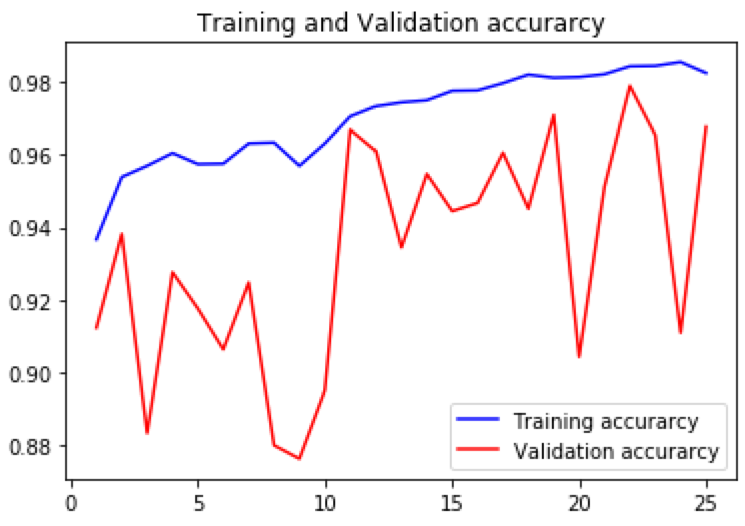
4.Kubernetes: deploy your deep learning model with Kubernetes, Docker and Flask.

CHAPTER:5

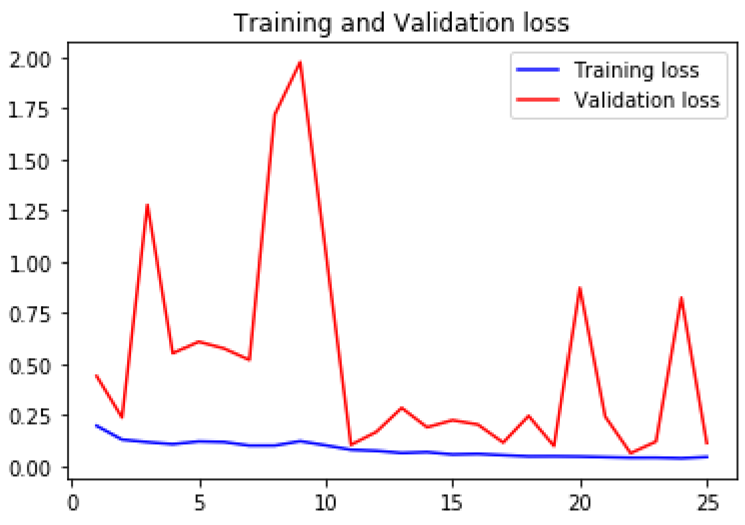
RESULTS AND DISCUSSIONS

5.RESULTS AND DISCUSSIONS

5.1 PERFORMANCE GRAPHS:



Training and validation accuracy graphs



Training and validation loss graphs

5.2 MODEL ACCURACY

```
print("[INFO] Calculating model accuracy")

scores = model.evaluate(x_test, y_test)

print("Test Accuracy: {scores[1]*100}")
```

OUTPUT:

```
[INFO] Calculating model accuracy
591/591 [==============================] - 2s 3ms/step
Test Accuracy: 96.77383080755192
```

5.3 Output screens:

5.3.1 Experimental Results of Proposed Pest detection and classification methodology

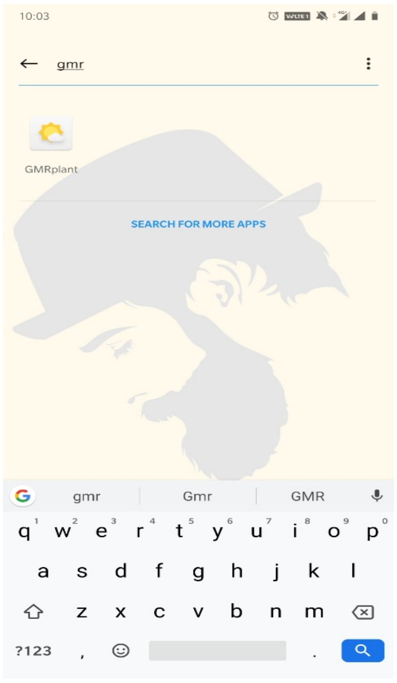


Figure 5.3.1.1 Mobile App icon

Discussion: The Mobile Application for Crop Protection and Advisory using Deep learning and Classification System shown in Fig.6.1. User can interact with the Application for more details. User click on application present in the figure to interact with the system.

- **Take Images:** By clicking this button user asked to take images**.**
- **Load Images:** By clicking this button user asked to load the images**.**
- **Test Image:** By clicking this button user asked to load the pest image to test and user shown up with advisory information.
- **About Disease:** By this button system information is displayed.

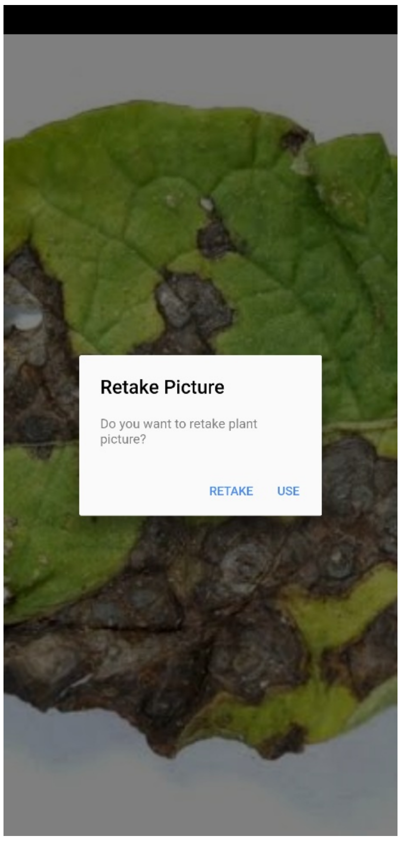


Figure 5.3.1.2 To Retake Picture

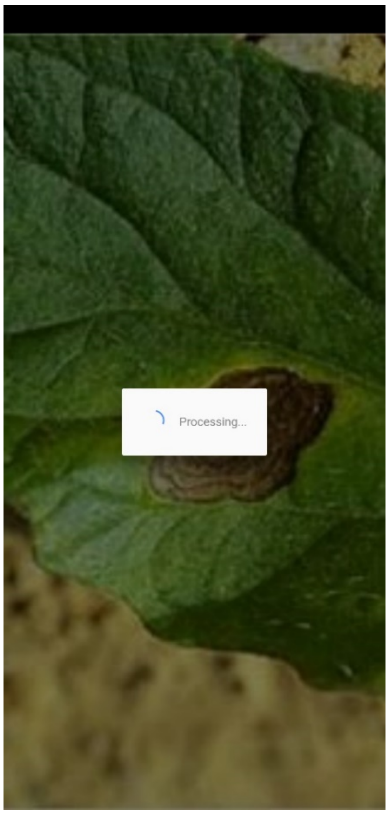
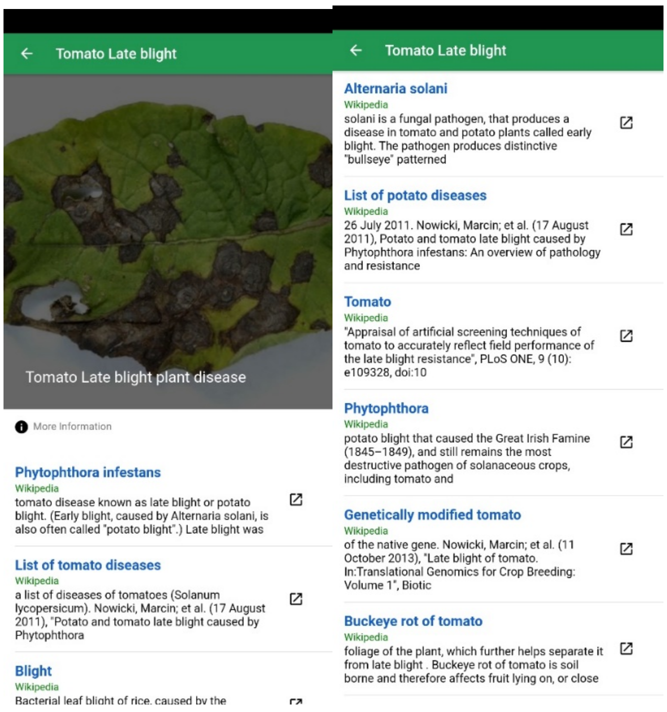


Figure 5.3.1.3 Loading the image to test and user shown up with advisory information



Phytophthora infestans
Wikipedia
tomato disease known as late blight or potato blight. (Early blight, caused by Alternaria solani, is also often called "potato blight".) Late blight was

List of tomato diseases
Wikipedia
a list of diseases of tomatoes (Solanum lycopersicum). Nowicki, Marcin; et al. (17 August 2011), "Potato and tomato late blight caused by Phytophthora

Blight
Wikipedia
Bacterial leaf blight of rice, caused by the

Alternaria solani
Wikipedia
solani is a fungal pathogen, that produces a disease in tomato and potato plants called early blight. The pathogen produces distinctive "bullseye" patterned

List of potato diseases
Wikipedia
26 July 2011. Nowicki, Marcin; et al. (17 August 2011), Potato and tomato late blight caused by Phytophthora infestans: An overview of pathology and resistance

Tomato
Wikipedia
"Appraisal of artificial screening techniques of tomato to accurately reflect field performance of the late blight resistance", PLoS ONE, 9 (10): e109328, doi:10

Phytophthora
Wikipedia
potato blight that caused the Great Irish Famine (1845–1849), and still remains the most destructive pathogen of solanaceous crops, including tomato and

Genetically modified tomato
Wikipedia
of the native gene. Nowicki, Marcin; et al. (11 October 2013), "Late blight of tomato. In:Translational Genomics for Crop Breeding: Volume 1", Biotic

Buckeye rot of tomato
Wikipedia
foliage of the plant, which further helps separate it from late blight . Buckeye rot of tomato is soil borne and therefore affects fruit lying on, or close

Figure 5.3.1.4 Getting information about the disease and the solutions.

CHAPTER-6

CONCLUSION AND FUTURE SCOPE

6.CONCLUSION AND FUTURE SCOPE

Conclusion:

In our system specialized deep learning models were developed, based on specific convolutional neural networks architectures, for the detection of plant diseases through leaves images of healthy or diseased plants. Different image processing techniques were used to detect and extract the pests in the captured image. The presented system is simple and yet efficient. The training of the models was performed using an openly available database of several photographs of leaves. The data comprises of [plant, disease] combinations, including some healthy plants. The experimental results achieved precision between 92% and 96%, for separate class tests. The final overall accuracy of the trained model was 94.3%. we have integrated the model in to mobile application for their use in mobile devices. Such devices could be either smartphones, to be used by farmers or agronomists.

Future Scope:

An extension of this study will be on gathering images for enriching the database and improving accuracy of the model using different techniques. The main goal for the future work will be developing a complete system consisting of application for smart mobile devices with features such as displaying recognized diseases in fruits, vegetables, and other plants, based on leaf images captured by the mobile phone camera. The user interface will guide the farmers to select their preferred language and post their requirements or problems in their language. The user interface also contains a personal voice assistant which guides illiterate farmers in accessing the information and services offered to them.

.

CHAPTER-7

REFERENCES

7. REFERENCES

References:

[1] Ryan H.L. Ip, Li- Minn Ang, KahPhooi Senga, J.C. Broster, J.E. Pratley, "**Big data and machine learning for crop protection**", Computers and Electronics in Agriculture 153 (2018) 69–81.

[2] G.M. Pasqual and J. Mansfield, "**Development of a Prototype Expert System for Identification and Control of Insect Pest**", Computers and Electronics in Agriculture 2 (1988) 263-276.

[3] Diego InácioPatrícioa, Rafael Riederb, "**Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review**", Computers and Electronics in Agriculture, 153 (2018) 69-81.

[4] Konstantinos P. Ferentinos, "**Deep learning models for plant disease detection and diagnosis**", Computers and Electronics in Agriculture, 145 (2018) 311-318.

[5] 5.Javier Lacasta, F. Javier Lopez-Pellicer, Borja Espejo-García, Javier Nogueras-Iso, F. Javier Zarazaga-Soria, "**Agricultural recommendation system for crop protection**", 152 (2018), 82-89.

[6] Huajian Liu, Javaan Singh Chahl, "**A multispectral machine vision system for invertebrate detection on green leaves"**, Computers and Electronics in Agriculture 150 (2018) 279-288.

[7] Zahid Iqbal, Muhammad Attique Khan, Muhammad Sharif, Jamal Hussain Shah, Muhammad Habib ur Rehman, Kashif Javed, "**An automated detection and classification of citrus plant diseases using image processing techniques: A review**", Computers and Electronics in Agriculture 153 (2018) 12-32.

[8] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. 2013.” **Overfeat: Integrated recognition, localization and detection using convolutional networks**”. arXiv:1312.6229.

[9] Simonyan, K., Zisserman, A. 2014.” Very **deep convolutional networks for large-scale image recognition**”. arXiv:1409.1556.

[10] Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., Stefanovic, D., 2016. “**Deep neural networks based recognition of plant diseases by leaf image classification**”. Computat. Intelligence Neurosci.. Article ID: 3289801

[11] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. 2015. Going deeper with convolutions. Proc. of the IEEE Conference on “**Computer Vision and Pattern Recognition**”.

[12] Yang, X., Guo, T., 2017. “**Machine learning in plant disease research**”. Europ. J. BioMed. Res. 6–9. http://dx.doi.org/10.18088/ejbmr.3.1.2016.pp6-9.