**A System for Detecting Leaf Diseases Using Hybrid Ensemble Deep Learning Techniques**

**ABSTRACT**

Deep learning is a branch of artificial intelligence. In recent years, with the advantages of automatic learning and feature extraction, it has been widely concerned by academic and industrial circles. It has been widely used in image and video processing, voice processing, and natural language processing. At the same time, it has also become a research hotspot in the field of agricultural crop protection, such as leaf disease recognition and pest range assessment, etc. The application of deep learning in plant disease recognition can avoid the disadvantages caused by artificial selection of disease spot features, make leaf disease feature extraction more objective, and improve the research efficiency and technology transformation speed. This review provides the research progress of deep learning technology in the field of leaf disease identification in recent years. In this project, we present the current trends and challenges for the detection of crop disease using Hybrid ensemble deep Learning Technique

**CHAPTER I**

**Introduction**

Generally, a plant gets diseased when it is continually disrupted by a certain causal agent, resulting in a physiological process anomaly which disrupts the normal structure of the plant’s function, growth, among other activities. Pathological conditions and symptoms result from the disruption of one or more of a plant’s critical biochemical and physiological systems.

The occurrence and prevalence of crop diseases vary seasonally, depending on the prevalence of a pathogen, conditions of the environment, and the crops and varieties are grown. Some plant varieties are more prone to outbreaks of plant diseases than others.

## **Classification of Plant Diseases :**

Plant diseases are classed genetically based on the nature of their principal causative agent, which could be non-infectious or infectious. A pathogenic organism, such as a virus, viroid, bacterium, fungus, mycoplasma, parasitic flowering plant, or nematode causes infectious plant diseases. An agent that is infectious can replicate inside or on a host plant and spread from one vulnerable host to the next. Nonmalignant plant illnesses are caused by unfavorable growing conditions such as high temperatures, poor oxygen-moisture ratios, poisonous chemicals in the atmosphere or soil, and a nutrient deficit or excess. Because they are not organisms capable of reproducing within a host, non-infectious causal agents are non-transmissible.

In agriculture, plants can be afflicted by multiple disease-causing agents at the same time. A plant that is suffering from nutrient insufficiency or an imbalance between soil moisture and oxygen is frequently more susceptible to pathogen infection, and a plant that has been infected by one disease is often vulnerable to secondary pathogen invasion. The disease complex is a collection of all disease-causal agents that afflict a plant. Knowledge of typical growth habits, varietal traits, and the normal variability of plants within a species—as these relate to the environment under which the plants grow—is essential to diagnose a disease.

## **Causes of leaf diseases:**

Leaf disease has traditionally been classified into two types: abiotic (also known as non-infectious) and biotic (infectious). Unfavorable environmental conditions frequently result in noncommunicable diseases. Low or high temperature, excess or lack of moisture are a few examples. Infections are also commonly caused by harmful air contaminants. Chemical or metallurgical plants nearby can cause them to accumulate. The disease is usually caused by the soil’s unhealthy physicochemical composition. The latter factor is frequently the result of poor-quality herbicide treatment of fields. These examples demonstrate the importance of sustainable agriculture not only for environmental protection but also for business profitability.

Even an unfavorable light regime can have a negative impact, especially on plants grown in greenhouses. Toxins released into the soil by some embryophytes (higher plants) and fungi can also be the cause of crop diseases.

## **symptoms of plant diseases:**

An observable consequence of plant disease on the plant is referred to as a symptom. One of the symptoms could be a discernible change in the plant’s color, function or shape, as it responds to the infection. Verticillium wilt is characterized by leaf wilting, which is caused by the fungus Verticillium albo-atrium and Verticillium dahlias. Common bacterial blight symptoms on bean plants include brown necrotic lesions surrounded by a bright yellow halo at the leaf blade or center of the leaf. You do not observe the pathogen that causes the disease, but rather a symptom caused by the infection. Outlined below are examples of common signs and symptoms of fungal, bacterial, and viral plant diseases.

Crop diseases symptoms caused by viruses are typically classified into four types:  
malformations, such as abnormal shoot growth and leaf and flower distortion;  
necrosis, wilting, and the appearance of annular stripes and spots;  
dwarfism, growth retardation of both individual parts and the entire plant; and  
discoloration, such as yellowing and vein clearing.

Root crop diseases, which manifest as rotting, are a telltale sign of the presence of a virus. Some plants, however, may not show symptoms and may be latent carriers of disease. As a result, extreme vigilance is required in the fight against this type of infection.

**Plant stunting**

As you can see, there is a lot of overlap in the symptoms of viral, bacterial and fungal diseases. When an unknown problem appears in a plant, herbicide injury, abiotic diseases, and nematode problems must all be considered.

**Prevention:**

Traditional Principles of Plant Disease Control. Avoidance—prevent disease by selecting a time of the year or a site where there is no inoculum or where the environment is not favorable for infection. Exclusion—prevent the introduction of inoculum. Eradication—eliminate, destroy, or inactivate the inoculum.

**CHAPTER II**

**LITERATURE REVIEW**

**1. Title: Plant Disease Recognition: A Large-Scale Benchmark Dataset and a Visual Region and Loss Reweighting Approach**

**Author: Xinda Liu, Weiqing Min , Member, IEEE, Shuhuan Mei, Lili Wang, Member, IEEE, and Shuqiang Jiang , Senior Member, IEEE**

**Year: VOL. 30, 2021**

In this paper they first compute the weights of all the divided patches from each image based on the cluster distribution of these patches to indicate the discriminative level of each patch. Then they allocate the weight to each loss for each patch-label pair during weakly-supervised training to enable discriminative disease part learning. We finally extract patch features from the network trained with loss reweighting, and utilize the LSTM network to encode the weighed patch feature sequence into a comprehensive feature representation

**2. Title: Crop Leaf Disease Image Super-Resolution and Identification With Dual Attention and Topology Fusion Generative Adversarial Network**

**Author: QIANG DAI 1 , XI CHENG 2 , YAN QIAO 1 , AND YOUHUA ZHANG**

**Year: VOL. 8, 2020**

In this paper, they propose a generative adversarial network with dual-attention and topology-fusion mechanisms called DATFGAN. This network can effectively transform unclear images into clear and high-resolution images. Additionally, the weight sharing scheme in our proposed network can significantly reduce the number of parameters. Experimental results demonstrate that DATFGAN yields more visually pleasing results than state-of-the-art methods. Additionally, treated images are evaluated based on identification tasks

**3. Title: Tomato Leaf Disease Identification by Restructured Deep Residual Dense Network**

**Author: CHANGJIAN ZHOU 1 , SIHAN ZHOU2 , JINGE XING1 , AND JIA SONG3**

**Year: VOL. 9, 2021**

In this paper, a restructured residual dense network was proposed for tomato leaf disease identification; this hybrid deep learning model combines the advantages of deep residual networks and dense networks, which can reduce the number of training process parameters to improve calculation accuracy as well as enhance the flow of information and gradients. The original RDN model was first used in image super resolution, so we need to restructure the network architecture for classification tasks through adjusted input image features and hyper parameters. E

# 4. Title: Dynamic Forecast Integrated with Case-based Reasoning Methodology

**Author:** [**Zhengang Yang**](https://ieeexplore.ieee.org/author/37087386589)**; [Yanhui Luo](https://ieeexplore.ieee.org/author/37087386942); [Feiqi Deng](https://ieeexplore.ieee.org/author/37087069619);**[**Ling Li**](https://ieeexplore.ieee.org/author/37087386410)

**Year: - VOL. 23, 2006**

Taking cucumber downy mildew (CDM) as research object, this research explores a technology of crop disease dynamic forecast integrated with case-based reasoning (CBR) methodology. A case indexing mechanism guided by idea of vantage case is developed to speed up the retrieval process and rapidly generate a similar case set for a new case in this CBR system. The range of optimal cluster number for this application is determined through the analysis and comparison between the exhaustive search and the search using proposed indexing mechanism. The precision and recall evaluations are conducted for each testing case using the X fold cross-validation approach, the reasoning effectiveness employing different thresholds of dissimilarity distance (TDD) is figured out and the optimal TDD for this CBR system is determined.

# 5. Title:- Study on detecting system of crop disease stress with acoustic emission technology

**Author:** [**Shi-feng Yang**](https://ieeexplore.ieee.org/author/37279518300)**; [Jian-ying Guo](https://ieeexplore.ieee.org/author/37403047000); [Ji-min Zhao](https://ieeexplore.ieee.org/author/37405218700); [Huai-you Wang](https://ieeexplore.ieee.org/author/37405996600)**

**Year: - VOL. 20, 2009**

Summary form only given. A system for detecting the conditions of crop disease stress by acoustic emission technology was studied and developed, and the contrast experiment study was carried out in a greenhouse by taking a healthy tomato and a disease stressed tomato crop as the objects. The PCI-2 acoustic emission board and R15 acoustic emission sensor probes were chosen to construct the hardware detecting system, and the AEWIN software and virtual instrument technology were utilized to construct the software system, a real-time acquisition and detecting system for the information between acoustic emission and disease stress of crop was established. The results show that there are certain physiological cycle laws in the acoustic emission of healthy crop, generally the ¿double peak area¿ can appear; there occurs distortion in the acoustic emission of disease stressed crop, a sharp increase of the acoustic emission frequency can suddenly take place at some time, which implies the crop begins to show disease symptoms, and acoustic emission signals lose the regularity. By taking these as the theoretical basis, it is required to further determine the relationship between acoustic emission signals and disease degrees and physiological states of crops, and to investigate diseases forecasting and automatic spraying of agricultural pesticide variables of main crops, which will become a new approach for the physiological signal detection of crops as well as comprehensive and precise prevention and cure of diseases.

# 6. Title: -Crops Disease Diagnosing Using Image-Based Deep Learning Mechanism

**Author:** [**Hyeon Park**](https://ieeexplore.ieee.org/author/37086162068)**; [Eun JeeSook](https://ieeexplore.ieee.org/author/37086466692);**[**Se-Han Kim**](https://ieeexplore.ieee.org/author/37086159666)

**Year: - 2018**

To increase the crop productivity environmental factors or product resource, such as temperature, humidity, labor and electrical costs are important. However, above all, crop disease is the crucial factor and causes 20-30% reduction of the productivity in case of its infection. Thus, the disease of the crop is much more important factor affecting the productivity of the crops. Therefore, the farmer concentrates on the cause of the disease in the crops during its growth, but it is not easy to recognize the disease on the spot. Until now, they just relied on the opinion of the experts or their own experiences when the disease is doubtful. However, it triggers a decrease in productivity as no taking appropriate action and time. In this paper, to address this problem we provide the mechanism, which dynamically analyses the images of the disease. The analysis result is immediately sent to the farmer required the decision and then feedback from the farmer is reflected to the model. The mechanism performs the diagnosing of the disease, especially for the strawberry fruits and leaves, with data set of images using deep learning. Thus, it encourages increasing of the productivity through the fast recognition of disease and the consequent action.

# 7. Title: -Crop Disease Detection Using Deep Learning

**Author: -** [**Omkar Kulkarni**](https://ieeexplore.ieee.org/author/37088651733)

**Year: - 2018**

In recent times, drastic climate changes and lack of immunity in crops has caused substantial increase in growth of crop diseases. This causes large scale demolition of crops, decreases cultivation and eventually leads to financial loss of farmers. Due to rapid growth in variety of diseases and adequate knowledge of farmer, identification and treatment of the disease has become a major challenge. The leaves have texture and visual similarities which attributes for identification of disease type. Hence, computer vision employed with deep learning provides the way to solve this problem. This paper proposes a deep learning-based model which is trained using public dataset containing images of healthy and diseased crop leaves. The model serves its objective by classifying images of leaves into diseased category based on the pattern of defect

# 8. Title: -AI based hybrid CNN-LSTM model for crop disease prediction: An ML advent for rice crop

**Author: -**[Sonal Jain](https://ieeexplore.ieee.org/author/37088395678); [Dharavath Ramesh](https://ieeexplore.ieee.org/author/38546859500)

**Year: - 2021**

Plant disease is an extreme challenge in gaining appropriate yield and crop quality. Therefore, a pest forewarning system is advantageous in early disease prediction and controlling it by practicing suitable measures. This paper presents a pest prediction and classification model for yellow stem border (YSB) disease in rice plants. An Artificial Intelligent based prediction model developed considering historical pest and weather data of various regions of India. The proposed model is named as hybrid CNN-LSTM that combines the advantage of convolution neural network (CNN) and long short term memory network (LSTM). It is a region-specific prediction model that predicts one-month pest data based on past three-months weather and pest data. The performance of the proposed model is compared with CNN and LSTM networks. This shows the enhancement in performance while using hybrid CNN-LSTM. On the other hand, this paper also presents a generalized classification model by combining the datasets of all regions. The model predicts the disease severity for the next day based on weather and preceding day pest data. The error correcting output code (ECOC) method with SVM classifier is used for the classification of disease severity.

# 9. Title: -Disease detection in crops using remote sensing images

**Author:** [Leninisha Shanmugam](https://ieeexplore.ieee.org/author/37086321320); [A. L Agasta Adline](https://ieeexplore.ieee.org/author/37087067729); [N Aishwarya](https://ieeexplore.ieee.org/author/37073158000); [G Krithika](https://ieeexplore.ieee.org/author/37086324759)

**Year: - 2017**

This paper describes an automated diseases detection using remote sensing images. Agriculturists are facing loss due to various crop diseases. It becomes tedious to the cultivators to monitor the crops regularly when the cultivated area is huge (in acres). The most significant part of our research is early detection the disease as soon as it starts spreading on the top layer of the leaves using remote sensing images. This approach has two phases: first phase deals with training of healthy and as well as diseased datasets i.e.) the extraction of threshold values from the image, second phase deals with monitoring of crops and identification of particular disease using canny edge detection algorithm and histogram analysis and also intimate the agriculturists with an early alert message immediately.

# 10. Title: A Multi-Crop Disease Detection and Classification Approach using CNN

**Author: -** [Zubair Saeed](https://ieeexplore.ieee.org/author/37089183220); [Ali Raza](https://ieeexplore.ieee.org/author/37089182500); [Ans H. Qureshi](https://ieeexplore.ieee.org/author/37089211772); [Muhammad Haroon Yousaf](https://ieeexplore.ieee.org/author/37087890138)

**Year: - 2021**

Being an agricultural country, the well-being of plants plays a vital role in the agricultural yield. Various diseases and disorders affect the quality and quantity of the production, thus the intelligent methods for disease detection in crops is the need of hour. We have proposed a robust and generalized approach that detects diseases in multiple crops by utilizing the baselines of existing CNN models. We have proposed the variants of ResNet-152 and Inception-v3 for detection of diseases in essential crops like rice and corn. We have used publicly available datasets having three different diseases in rice and both infected and healthy leaves of corn. The proposed method has achieved accuracy of 97.81% and 97.48% by employing variants of InceptionV3 and ResNet152 respectively for corn crop. To understand the diversity of diseases, we categorize the rice disease images into major and minor subsets.The proposed ResNet152 variant has achieved accuracy of 99.10% and 82.20% for the major and minor disease subsets respectively. Experimental results of proposed approach indicate robustness in disease detection for multiple crops.

**CHAPTER III**

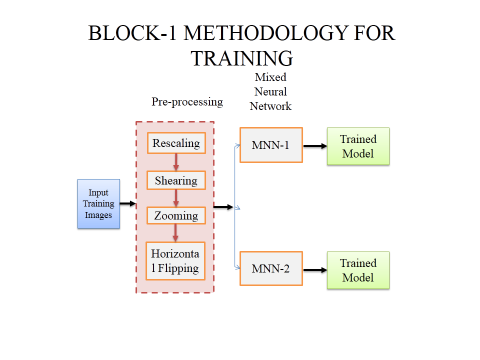
**SYSTEM IMPLEMENTATION**

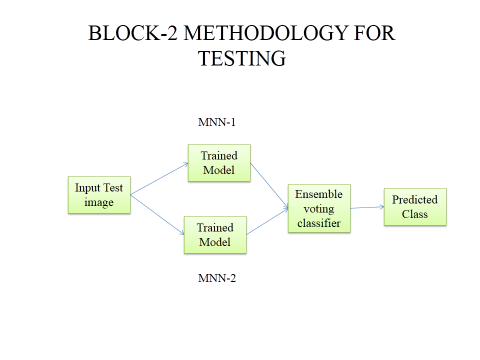
**3.1 PROPOSED SYSTEM**

In the Proposed system we used a deep Learning Algorithm called Hybrid ensemble deep Learning Technique to predict the crop disease

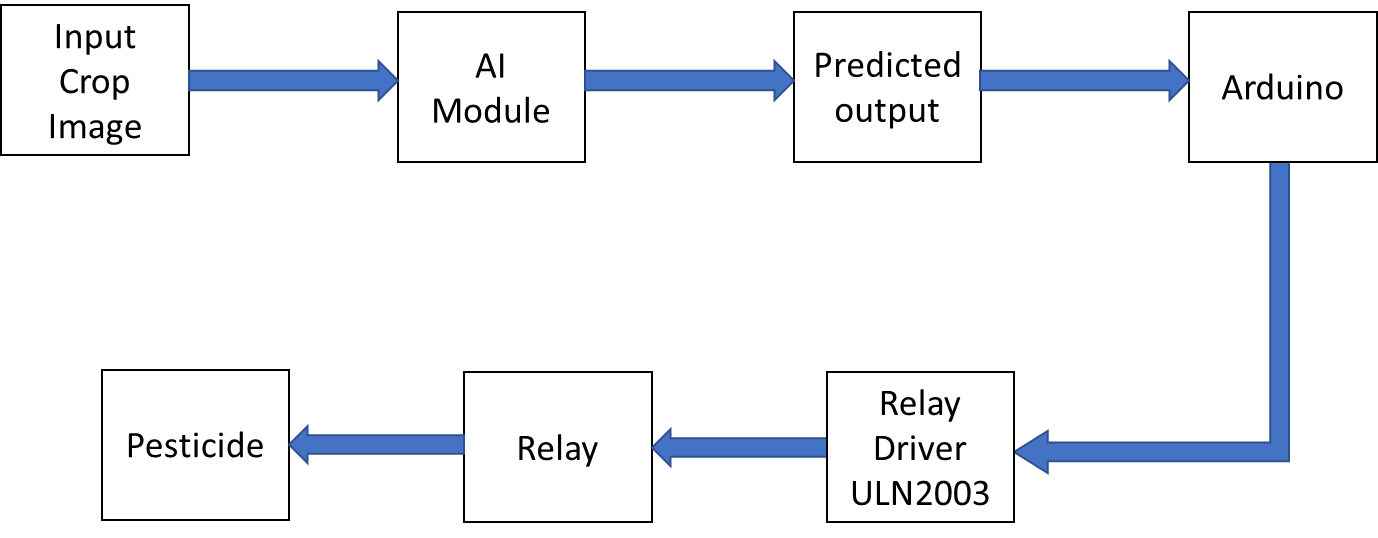
So that we can able to get the accuracy over 90%

**3.2 BLOCK DIAGRAM**

****

****

**Block-3 AI with Embedded System**



**3.2.1 Description**

* The raw input Training data's are pre-processed by using zooming, shearing, flipping and rescaling methods.
* Shear tool is used to shift one part of an image, a layer, a selection or a path to a direction and the other part to the opposite direction.
* When zooming, pixels are inserted into the image in order to expand the size of the image, and the major task is the interpolation of the new pixels form the surrounding original pixels
* A flip (mirror effect) is done by reversing the pixels horizontally or vertically. For instance, for an horizontal flip, the pixel situated at coordinate (x, y) will be situated at coordinate (width - x - 1, y) in the new image.
* Rescaling is used to rescale the data’s at each spectral dimension.
* Then the datas are given to the Mixed neural network model(MNN)
* Finally we can get the predicted Class (Good Leaf or Infected leaf) for the test image as output.
* The Input image are getted and classified using Artificial Intelligence then the classified data is feeded into embedded system, and the embedded controller triggers the pesticide pump based on the classified output.

**DEEP LEARNING**

Deep learning (also known as deep structured learning) is part of a broader family of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) methods based on [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_networks) with [representation learning](https://en.wikipedia.org/wiki/Representation_learning). Learning can be [supervised](https://en.wikipedia.org/wiki/Supervised_learning), [semi-supervised](https://en.wikipedia.org/wiki/Semi-supervised_learning) or [unsupervised](https://en.wikipedia.org/wiki/Unsupervised_learning).

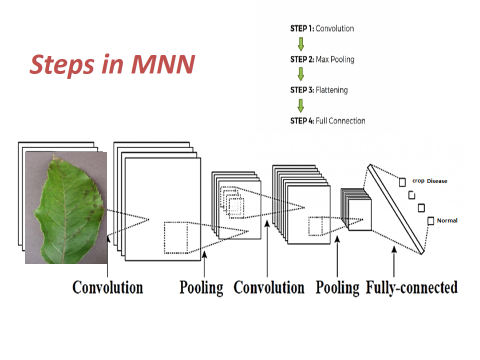
Deep learning architectures such as [deep neural networks](https://en.wikipedia.org/wiki/Deep_learning#Deep_neural_networks), [deep belief networks](https://en.wikipedia.org/wiki/Deep_belief_network), [recurrent neural networks](https://en.wikipedia.org/wiki/Recurrent_neural_networks) and [convolution neural networks](https://en.wikipedia.org/wiki/Convolutional_neural_networks) have been applied to fields including [computer vision](https://en.wikipedia.org/wiki/Computer_vision), [machine vision](https://en.wikipedia.org/wiki/Machine_vision), [speech recognition](https://en.wikipedia.org/wiki/Automatic_speech_recognition), [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), [audio recognition](https://en.wikipedia.org/wiki/Audio_recognition), social network filtering, [machine translation](https://en.wikipedia.org/wiki/Machine_translation), [bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics), [drug design](https://en.wikipedia.org/wiki/Drug_design), medical image analysis, material inspection and [board game](https://en.wikipedia.org/wiki/Board_game) programs, where they have produced results comparable to and in some cases surpassing human expert performance.

The adjective "deep" in deep learning comes from the use of multiple layers in the network. Early work showed that a linear [perceptron](https://en.wikipedia.org/wiki/Perceptron) cannot be a universal classifier, and then that a network with a nonpolynomial activation function with one hidden layer of unbounded width can on the other hand so be. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions. In deep learning the layers are also permitted to be heterogeneous and to deviate widely from biologically informed [connectionist](https://en.wikipedia.org/wiki/Connectionism) models, for the sake of efficiency, trainability and understandability, whence the "structured" part.

**MIXED NEURAL NETWORK**

In [deep learning](https://en.wikipedia.org/wiki/Deep_learning), a mixed neural network (MNN) is a class of [deep neural networks](https://en.wikipedia.org/wiki/Deep_neural_network), most commonly applied to analyzing visual imagery.[[1]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-Valueva_Nagornov_Lyakhov_Valuev_2020_pp._232%E2%80%93243-1) They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and [translation invariance](https://en.wikipedia.org/wiki/Translation_invariance) characteristics.[[2]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-:0-2)[[3]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-:1-3) They have applications in [image and video recognition](https://en.wikipedia.org/wiki/Computer_vision), [recommender systems](https://en.wikipedia.org/wiki/Recommender_system),[[4]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-4) [image classification](https://en.wikipedia.org/wiki/Image_classification), [medical image analysis](https://en.wikipedia.org/wiki/Medical_image_computing), [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing),[[5]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-5) and financial [time series](https://en.wikipedia.org/wiki/Time_series).[[6]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-Tsantekidis_7%E2%80%9312-6)

MNNs are [regularized](https://en.wikipedia.org/wiki/Regularization_(mathematics)) versions of [multilayer perceptrons](https://en.wikipedia.org/wiki/Multilayer_perceptron). Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to [overfitting](https://en.wikipedia.org/wiki/Overfitting) data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. MNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, MNNs are on the lower extreme.



**Architecture of Mixed neural network**

**Steps in MNN**

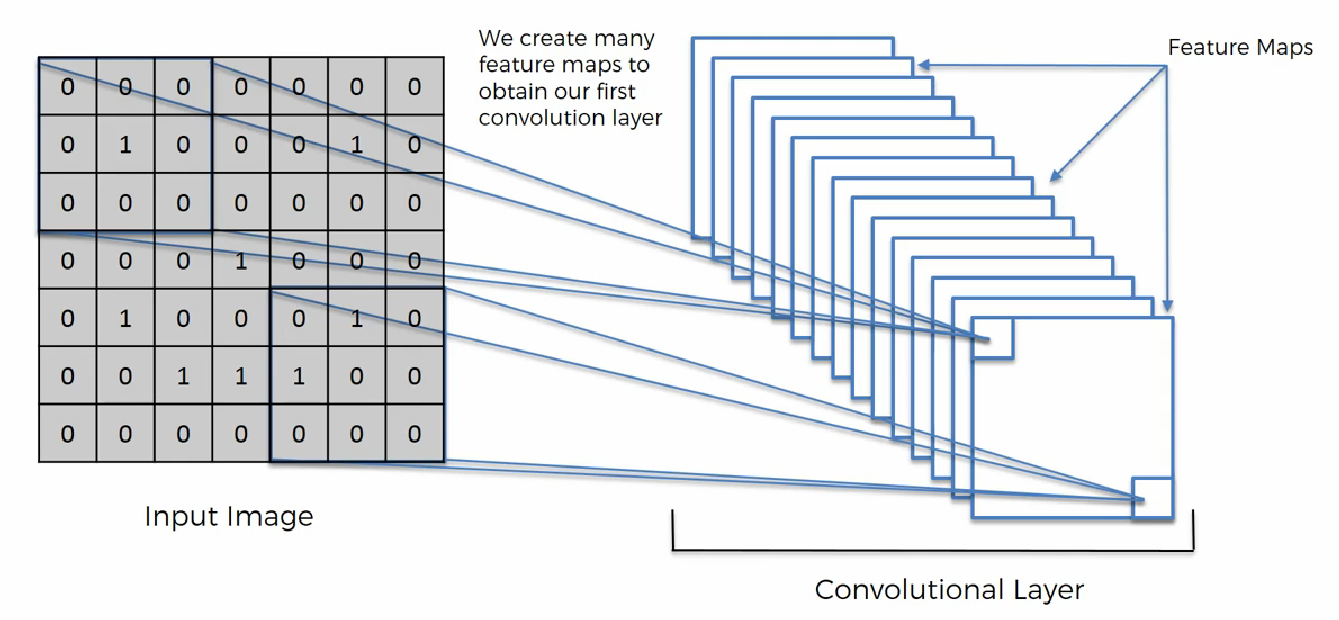
* **Step 1: Convolution**
* **Step 2: Max pooling**
* **Step 3: Flattening**
* **Step 4: Fully connection**

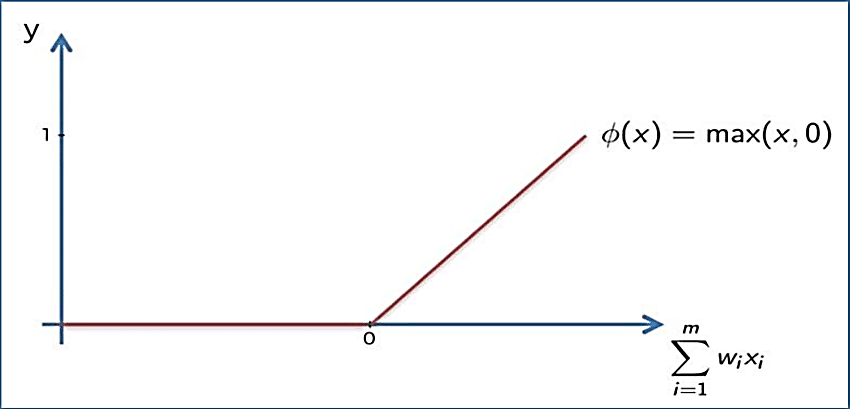
**Convolution Layer**

When programming a MNN, the input is a [tensor](https://en.wikipedia.org/wiki/Tensor) with shape (number of images) x (image height) x (image width) x ([image depth](https://en.wikipedia.org/wiki/Image_depth)). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map height) x (feature map width) x (feature map channels). A convolutional layer within a neural network should have the following attributes:

* Convolutional kernels defined by a width and height (hyper-parameters).
* The number of input channels and output channels (hyper-parameter).
* The depth of the Convolution filter (the input channels) must be equal to the number channels (depth) of the input feature map.

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.[[12]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-deeplearning-12) Each convolutional neuron processes data only for its [receptive field](https://en.wikipedia.org/wiki/Receptive_field). Although [fully connected feedforward neural networks](https://en.wikipedia.org/wiki/Multilayer_perceptron) can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters.[[13]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-13) For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. By using regularized weights over fewer parameters, the vanishing gradient and exploding gradient problems seen during [backpropagation](https://en.wikipedia.org/wiki/Backpropagation) in traditional neural networks are avoided.[[14]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-14)[[15]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-15)

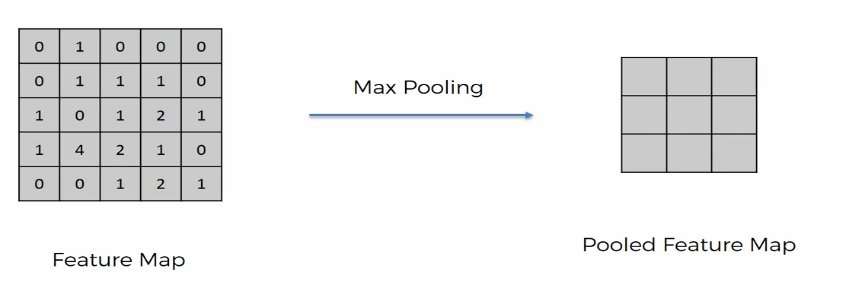




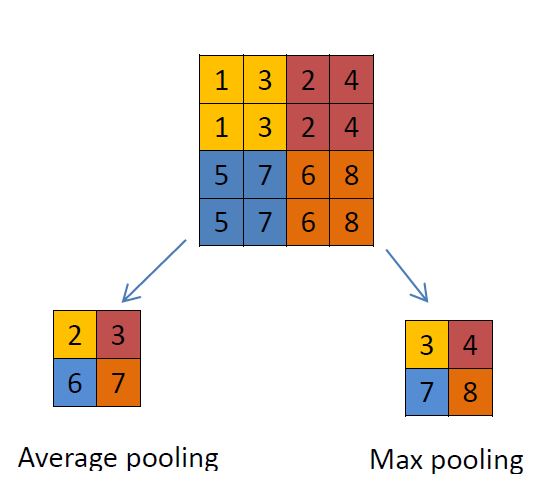
**Appling ReLu Activation function to decrease the linearity in the image, because the image originally nonlinear**

**Pooling Layer**

A **pooling** layer is another building block of a **MNN**. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. **Pooling** layer operates on each feature map independently. The most common approach used in **pooling** is max **pooling**.

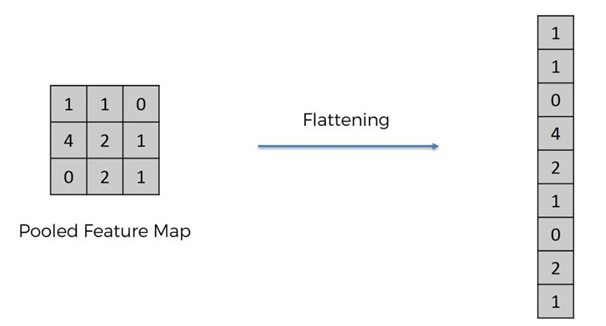


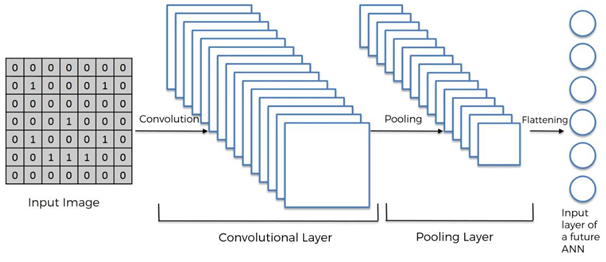
**Max / Avg. Pooling**



**Flattening**

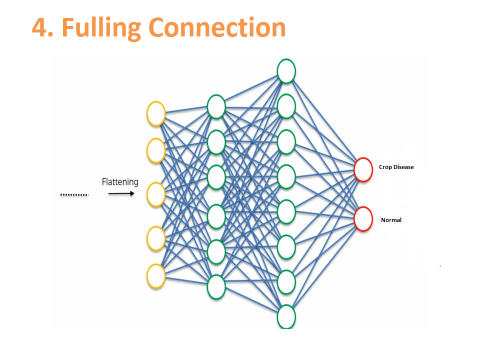
**Flattening** is converting the data into a 1-dimensional array for inputting it to the next layer. We **flatten** the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer





**Fulling Connection**

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional [multi-layer perceptron](https://en.wikipedia.org/wiki/Multi-layer_perceptron) neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.



**ENSEMBLE VOTING CLASSIFIER**

A voting ensemble (or a “*majority voting ensemble*“) is an ensemble machine learning model that combines the predictions from multiple other models.

It is a technique that may be used to improve model performance, ideally achieving better performance than any single model used in the ensemble.

A voting ensemble works by combining the predictions from multiple models. It can be used for classification or regression. In the case of regression, this involves calculating the average of the predictions from the models. In the case of classification, the predictions for each label are summed and the label with the majority vote is predicted.

* **Regression Voting Ensemble**: Predictions are the average of contributing models.
* **Classification Voting Ensemble**: Predictions are the majority vote of contributing models.

There are two approaches to the majority vote prediction for classification; they are hard voting and soft voting.

Hard voting involves summing the predictions for each class label and predicting the class label with the most votes. Soft voting involves summing the predicted probabilities (or probability-like scores) for each class label and predicting the class label with the largest probability.

* **Hard Voting**. Predict the class with the largest sum of votes from models
* **Soft Voting**. Predict the class with the largest summed probability from models.

A voting ensemble may be considered a meta-model, a model of models.

As a meta-model, it could be used with any collection of existing trained machine learning models and the existing models do not need to be aware that they are being used in the ensemble. This means you could explore using a voting ensemble on any set or subset of fit models for your predictive modeling task.

A voting ensemble is appropriate when you have two or more models that perform well on a predictive modeling task. The models used in the ensemble must mostly agree with their predictions.

Hard voting is appropriate when the models used in the voting ensemble predict crisp class labels. Soft voting is appropriate when the models used in the voting ensemble predict the probability of class membership. Soft voting can be used for models that do not natively predict a class membership probability, although may require [calibration of their probability-like scores](https://machinelearningmastery.com/calibrated-classification-model-in-scikit-learn/) prior to being used in the ensemble (e.g. support vector machine, k-nearest neighbors, and decision trees)

**CHAPTER IV**

**SIMULATION RESULTS& DISCUSSION**

**4.1 SOFTWARE DESCRIPTION**

The Python language had a humble beginning in the late 1980s when a Dutchman Guido Von Rossum started working on a fun project, which would be a successor to ABC language with better exception handling and capability to interface with OS Amoeba at Centrum Wiskunde and Informatica. It first appeared in 1991. Python 2.0 was released in the year 2000 and Python 3.0 was released in the year 2008. The language was named Python after the famous British television comedy show Monty Python's Flying Circus, which was one of Guido's favorite television programmes. Here we will see why Python has suddenly influenced our lives and the various applications that use Python and its implementations.

In this chapter, you will be learning the basic installation steps that are required to perform on different platforms (that is Windows, Linux, and Mac), about environment variables, setting up of environment variables, file formats, Python interactive shell, basic syntaxes and finally printing out formatted output.

**4.1.1 Why Python?**

Now you might be suddenly bogged with the question, why Python? According to Institute of Electrical and Electronics Engineers (IEEE) 2016 ranking Python ranked third after C and Java. As per Indeed.com's data of 2016, the Python job market search ranked fifth. Clearly, all the data points to the ever rising demand in the job market for Python. It’s a cool language if you want to learn just for fun or if you want to build your career around Python, you will adore the language. At school level, many schools have started including Python programming for kids. With new technologies taking the market by surprise Python has been playing a dominant role. Whether it is cloud platform, mobile app development, Big Data, IoT with Raspberry Pi, or the new Block chain technology, Python is being seen as a niche language platform to develop and deliver a scalable and robust applications.

Some key features of the language are:

* Python programs can run on any platform, you can carry code created in Windows machine and run it on Mac or Linux
* Python has inbuilt large library with prebuilt and portable functionality, also known as the standard library
* Python is an expressive language
* Python is free and open source
* Python code is about one third of the size of equivalent C++ and Java code
* Python can be both dynamically and strongly typed--dynamically typed means it is a type of variable that is interpreted at runtime, which means, in Python, there is no need to define the type (int or float) of the variable

**Python applications**

One of the most famous platforms where Python is extensively used is YouTube. The other places where you will find Python being extensively used are the special effects in Hollywood movies, drug evolution and discovery, traffic control systems, ERP systems, cloud hosting, e-commerce platform, CRM systems, and whatever field you can think of.

**Versions**

At the time of writing this book, two main versions of the Python programming language were available in the market, which are Python 2.x and Python 3.x. The stable release as of writing the book were Python 2.7.13 and Python 3.6.0.

**Implementations of Python**

Major implementations include CPython, Jython, IronPython, MicroPython, and PyPy.

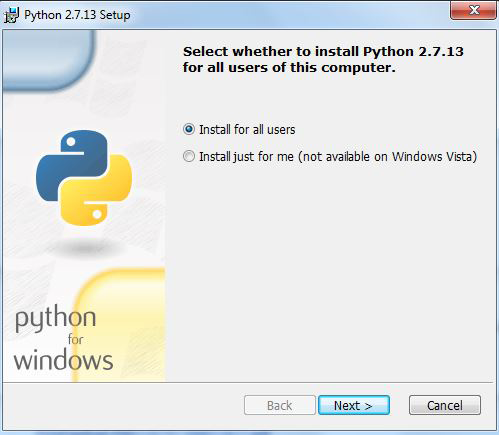
**4.1.2 Installation**

Here we will look forward to the installation of Python on three different OS platforms, namely, Windows, Linux, and Mac OS. Let's begin with the Windows platform.

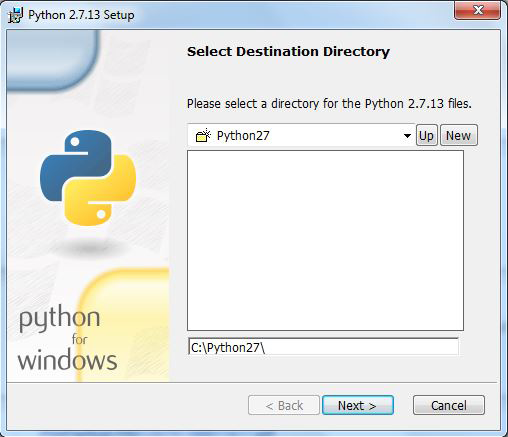
**Installation on Windows platform**

Python 2.x can be downloaded from h t t p s ://w w w . p y t h o n . o r g /d o w n l o a d s . The installer is simple and easy to install. Perform the following steps to install the setup:

1. Once you click on setup installer, you will get a small window on your desktop screen as shown here; click on **Next**:

****

2. Provide a suitable installation folder to install Python. If you don't provide the installation folder, then the installer will automatically create an installation folder for you, as shown in the following screenshot. Click on **Next**:

****

3. After completion of step 2, you will get a window to customize Python as shown in the preceding screenshot. Notice that the **Add python.exe to Path** option has been marked **x**. Select this option to add it to system path variable (which will be explained later in the chapter), and click on **Next**:

****

4. Finally, click on **Finish** to complete the installation:

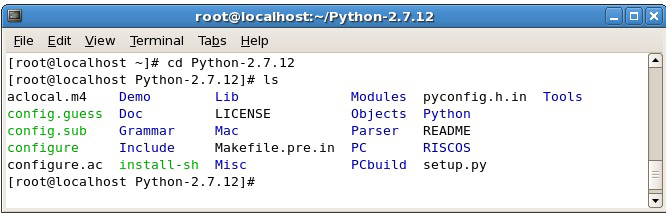
****

**Installation on Linux platform**

These days most of the Linux-based systems come preloaded with Python, so in most cases, you do not need to install it separately. However, if you do not find your desired version of Python on the Linux platform, you can download your desired version for a particular Linux platform from the site h t t p s ://w w w . p y t h o n . o r g /d o w n l o a d s /s o u r c e /. Perform the following steps:

1. Extract the compressed file using the tar –xvzf python\_versionx.x command.

2. Browse the directory of the compressed file as shown in the screenshot:

****

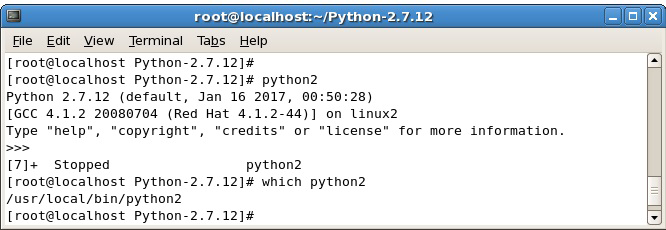
3. Run the following commands:

**[root@localhost Python-2.7.12]# ./configure**

**[root@localhost Python-2.7.12]# make**

**[root@localhost Python-2.7.12]# make install**

4. Use the command as shown in screenshot to ensure that Python is running:

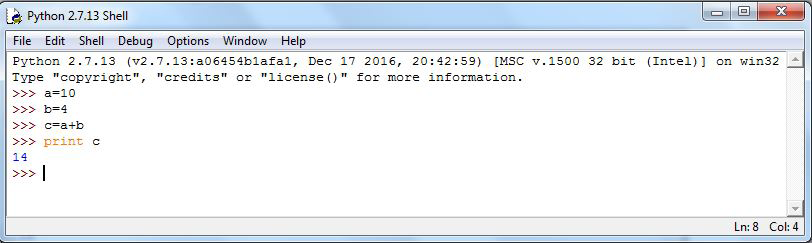
****

**4.1.3 Python file formats**

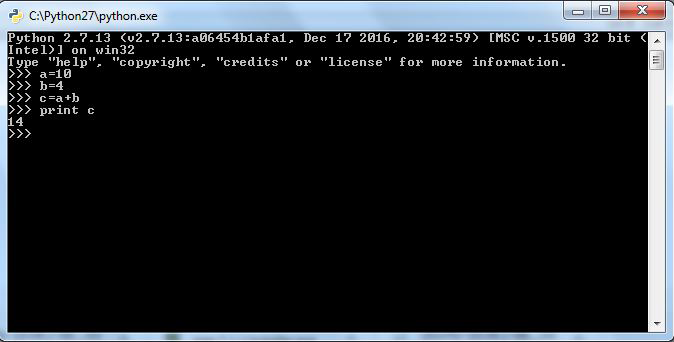
Every language understands a file format, for example, like the C language file extension is .c likewise java language has a file extension .java. The Python file extension is .py while bytecode file extension is .pyc.

**Python interactive shell**

Python interactive shell is also known as **Integrated Development Environment** (**IDLE**). With the Python installer, two interactive shells are provided: one is IDLE (Python GUI) and the other is Python (command line). Both can be used for running simple programs. For complex programs and executing large files, the windows command prompt is used, where after the system variables are set automatically, large files are recognized and executed by the system.



The preceding screenshot is what we call Python IDLE, which comes bundled with the Python installation. The next screenshot is of the command line that also comes bundled with the Python installation, or we can simply launch the Python command through the windows command line and get Python command line. For most of our programming instructions, we will be using the Python command line:

****

**4.1.4 Syntax and semantics**

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

**Indentation**

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation doesn't have any semantic meaning.

**Statements and control flow**

Python's statements include (among others):

* The **assignment** statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language. Assignment in C, e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed binding the name to the object. Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound. Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it. However at a given time a name will be bound to some object, which will have a type; thus there is dynamic typing.
* The**if**statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
* The **for** statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
* The **while** statement, which executes a block of code as long as its condition is true.
* The **try** statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
* The **raise** statement, used to raise a specified exception or re-raise a caught exception.
* The **class** statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
* The**def** statement, which defines a function or method.
* The **with** statement, from Python 2.5 released in September 2006, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing Resource Acquisition Is Initialization (RAII)-like behavior and replaces a common try/finally idiom.
* The **break** statement, exits from the loop.
* The **continue** statement, skips this iteration and continues with the next item.
* The **pass** statement, which serves as a NOP. It is syntactically needed to create an empty code block.
* The **assert** statement, used during debugging to check for conditions that ought to apply.
* The **yield** statement, which returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines.
* The **import** statement, which is used to import modules whose functions or variables can be used in the current program. There are three ways of using import: import <module name> [as <alias>] or from <module name> import \* or from <module name> import <definition 1> [as <alias 1>], <definition 2> [as <alias 2>], ....
* The**print** statement was changed to the print() function in Python 3.

**4.1.5 Python programming examples**

Hello world program:

print('Hello, world!')

Program to calculate the factorial of a positive integer:

n = int(input('Type a number, then its factorial will be printed: '))

if n < 0:

raise ValueError('You must enter a positive number')

fact = 1

i = 2

while i <= n:

fact = fact \* i

i += 1

print(fact)

CONCLUSION

In this project, a Hybrid Ensemble deep learning technique based architecture has been proposed for robust detection leaf disease. This architecture adopts an enhanced mixed neural network that is interconnected with series of skip pathway. The multi-stage approach overcomes the limitation of some deep convolutional networks in producing coarsely segmented outputs when processing challenging crop images. In this approach, the whole network is divided two MNN (mixed neural network) architectures for training,. A new method is devised to classify leaf diseases based on the results from the ensemble voting classifier. It also aims at developing an efficient system that can meet up with real time leaf disease diagnosis task.

REFERENCES

[1] Umit Atilaa, Murat Uçarb, Kemal Akyolc, and Emine Uçar, "Plant leaf disease classification using EfficientNet deep learning model," Ecological Informatics, vol.61, Article 101182, March 2021

[2] M.Akila and P.Deepan, "Detection and Classification of Plant Leaf Diseases by using Deep Learning Algorithm," International Journal of Engineering Research & Technology (IJERT), SSN: 2278-0181, 2018

[3] Srdjan Sladojevic, Marko Arsenovic, Andras Anderla, Dubravko Culibrk, and Darko Stefanovic, "Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification," Hindawi Publishing Corporation Computational Intelligence and Neuroscience, vol. 2016, Article ID 3289801, 2016

[4] S. B. Patil and S. K. Bodhe, “Leaf disease severity measurement using image processing,” International Journal of Engineering and Technology, vol.3, no.5, pp.297–301, 2011.

[5] P. Revathi and M. Hemalatha, “Identification of cotton diseases based on cross information gain deep forward neural network classifier with PSO feature selection,” International Journal of Engineering and Technology, vol.5, no.6, pp.4637–4642, 2014

[6] Kulkarni Anand H and Ashwin Patil RK, “Applying image processing technique to detect plant diseases,” International Journal of Modern Engineering Research (IJMER), vol. 2, pp. 3661-3664, 2012

[7] E. Mwebaze and M. Biehl, “Prototype-based classification for image analysis and its application to crop disease diagnosis,” Advances in Self-Organizing Maps and Learning Vector Quantization - Proceedings of the 11th International Workshop WSOM 2016, pp. 329–339, January 2016.

[8] J. R. Aduwo, E. Mwebaze, and J. A. Quinn, “Automated vision-based diagnosis of cassava mosaic disease,” Industrial Conference on Data Mining, pp. 114–122, 2010.

[9] E. Mwebaze, P. Schneider, F. Schleif, J. R. Aduwo, J. A. Quinn, S. Haase, T. Villmann, and M. Biehl, “Divergence-based classification in learning vector quantization,” Neurocomputing, vol. 74, pp. 1429– 1435, 2011.

[10] J. Tuhaise, J. A. Quinn, and E. Mwebaze, “Pixel classification methods for automatic symptom measurement of cassava brown streak disease,” Proceding of the 1st International Conference on the Use of Mobile ICT in Africa, 2014.