Tomato Leaf Disease Detection using Convolutional Neural Network

by

Md. Riazul Hasan
19101550
Md. Shajib Hossain
19101250
Md. Minhajul Islam
19101111
Md. Rejoanur Rahman Apu
19101260
Farzana Akter Moli
19101280

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
BRAC University
January 2023

Declaration

It is hereby declared that,

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Md. Riazul Hasan

19101550

Md. Shajib Hossain

19101250

Minhaj

Md. Minhajul Islam

19101111

[illegible]

Md. Rejoanur Rahman Apu

19101260

Farzana Akter Moli

Farzana Akter Moli

19101280

Approval

The thesis/project titled "Tomato leaf disease detection using Convolutional Neural Network" submitted by

1. Md. Riazul Hasan (19101550)
2. Md. Shajib Hossain (19101250)
3. Md. Minhajul Islam (19101111)
4. Md. Rejoanur Rahman Apu (19101260)
5. Farzana Akter Moli (19101280)

Of Fall, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on January 17, 2023.

Examining Committee:

Supervisor:
(Member)

Annajiat Alim Rasel

Digitally signed by Annajiat Alim Rasel
DN: cn=Annajiat Alim Rasel, o=Brac University, ou=CSE Department, email=annajiat@bracu.ac bd, c=BD
Date: 2023.01.14 23:04:00 +06'00'

Annajiat Alim Rasel

Senior Lecturer
Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)

Dr. Amitabha Chakrabarty, PhD

Associate Professor
Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD

Assistant Professor
Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

The fertile soil and easy access to water make agriculture more suitable and valuable for Bangladesh. Most people are directly or indirectly dependent on agricultural products for their livelihood. Agriculture plays an important role in the GDP of Bangladesh, which is 12.68% in 2019. According to the UN FAO, tomato is a type of vegetable that is ingested by 16% of the entire population. When analyzing the agricultural environment in Bangladesh, tomatoes are considered one of the most common vegetables. Plant infections pose a significant danger to crop production, yet timely detection remains a challenge in several regions of the world due to a lack of facilities. Climate changes are forcing us to take more care of agriculture to ensure food safety. Early detection of diseases has been made possible by current developments in computer vision. Image processing and deep learning are very useful in this situation. The object's impacted region is segmented using a bespoke threshold algorithm based on HBS (hue-based segmentation). Utilizing a color co-occurrence approach, the segmented portion's consequential selected features are recovered for edge detection. This research shows the diagnosis and detection of tomato leaf diseases involving several steps, including image capture, image pre-processing, picture segmentation, feature extraction, and classification using a Convolutional Neural Network(CNN). The proposed CNN model achieved 95% accuracy while using much fewer computational resources, which makes it easily deployable in mobile applications.

Keywords: Deep learning, Convolutional neural network, ResNet-50, Inception v3, Tomato leaf disease detection.

Acknowledgement

In the first place, we would like to thank Almighty Allah for allowing us to complete our thesis on time and without deterrent.

Having said that, we would like to express our gratitude to Annajiat Alim Rasel, our distinguished instructor and supervisor also Amitabha Chakrabarty, our distinguished instructor and co-supervisor for their unwavering support and tenacious oversight, which allowed us to complete our project. In addition, we would like to thank our supportive friends who have been there for us during the difficult times. Lastly, to our parents, without their unwavering support it may be impossible. With their gracious assistance and prayers, we are now on the verge of graduating.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and goals

Bangladesh is a densely populated country and according to a study conducted by the [72] World Bank in 2021, the density of population is over 1,278 per square kilometer of land area with a total population of 164.7 million (2020). Since the density of the population is higher than usual and with this much population it is very difficult to meet the requirement of food for this growing population day by day. However, to meet the requirements, there is no way but to increase the production of food in our country and rely less on imports from foreign countries. Fortunately, Bangladesh is an agricultural country, which makes it a bit easy to meet the requirement of food [36]. Even in 2020 according to the World Bank collection of development indicators, compiled from officially recognized sources 37.75% of the total population is somehow employed in the agriculture sector. So agriculture plays an important role in most of the sectors in Bangladesh. Therefore, talking about the agriculture sector and Bangladesh's requirement for food for the ever-growing population, rice plays the main role to meet the requirement of food. But only rice alone can not meet the demand for this huge population and it creates a huge pressure on a single crop. On the other hand, we are very aware that there are some other crops and vegetables which can decrease the demand and pressure of production of rice. That is why the government has taken some necessary steps to create diversification of food and farmer's revenue and to do this the government is trying to give incentives for vegetable production. Vegetables are an important sub-sector in agricultural GDP. Vegetables contribute 3.2 percent to the agricultural GDP [73]. The vegetable sector contributed approximately $718 million to Bangladesh's gross domestic product in 2010 (BBS, 2012). Production of vegetables has increased over the past few years here in Bangladesh [19]. According to the Yearbook of Agricultural Statistics of Bangladesh, 2011, within in just 2 years from 2009 to 2011 the total production in per acre yield has increased for over 200 kg and the number jumped into 3,378 kgs. This is why the government is planning to find a new or alternative way to meet the growing food requirements in recent years and encourage them to produce more vegetables. In Bangladesh, as the season changes, the production rate of some vegetables also changes. And vegetables are labeled as summer, winter and all season vegetables. Among all the vegetables' tomato is such a vegetable that can grow all year round, that is why this is considered as one of the most important vegetables in Bangladesh. Therefore, in this paper we are trying to find a new and automated way to detect tomato leaf disease early and easily with the help of Computer Science and Technologies so that farmers can easily take necessary steps before it can affect the other plant and prevent before the disease can bring down the production rate of tomatoes.

Though Tomatoes can grow all year round but their peak time for production is from December to March because It nurtures well under an average monthly temperature range of 21°-23 °C, commercially it may be grown at temperatures ranging from 18 °C to 27 °C. While the Detection of plant disease is an essential research topic, In this paper we have chosen to detect the tomato plant's leaf diseases because most of the plant diseases are triggered by fungi, bacteria, and viruses. Morphological changes in leaves are the primary stage of Fungi. Bacteria can generally have simpler life cycles and can be identified by morphological changes in the leaves. Therefore, our prime plan is to examine the leaves and detect if the plant has any sort of disease or not. And we have also found that according to [42] Vetal and Khule while doing their research they have found that the most common diseases for tomato plants are (a) Bacterial spot, (b) Early blight, (c) Late blight, (d) Leaf Mold, (e) Septoria leaf spot, (f) Target Spot, (g)Tomato mosaic virus, (h) Tomato yellow leaf curl virus, (i) Two-spotted spider mite. Therefore, in this paper we will be trying to find these diseases by using a method in Computer Science called Image processing using Convolutional Neural Network model.

1.2 Components of detection

The diagnostic system to detect tomato leaf disease would include the following proposed and pre-trained convolutional neural network model for data processing.

1. Convolutional Neural Network

2. ResNet-50 Architecture

3. Inception V3 Architecture

1.3 Scopes and Obstructions in Experiment

Restraints in identifying tomato leaf disease include the following:

1. We are predicting our result by collecting samples images from the PlantVillage dataset.

2. The diseases that are seen to be affecting the leaves are under experiment.

3. In our study, not all the diseases of Tomato are included.

4. The diseases that are not seen on the leaf cannot be traceable using our methodologies.

5. More than one diseases can be detected simultaneously.

6. In some cases, it is seen that some of the symptoms look familiar.

1.4 Research Objective

We got evidence of plant disease from fossils, which proves that plant diseases were there even 250 million years ago. There were no technologies and diagnosis techniques to identify the diseases. As a result, outbreaks took place due to plant diseases like rust, blight, etc. A huge number of people died from lack of food during those outbreaks. Especially, people who live in underdeveloped countries suffer the most. Now a days, diagnosticians are there to identify the diseases of crops. He examines the air, soil, and other cultural conditions of the field to determine whether the crops are healthy or not. The methods which are currently used worldwide are classified into direct and indirect methods. PCR, FISH, ELISA, IF, and FCM are included in the direct method. Thermography, Fluorescence imaging, Gas Chromatography are considered indirect methods. It extracts DNA from the plant tissue and uses it to detect the target organism. It provides several advantages like an organism needs not to be cultured, it can potentially detect a single target molecule etc. But PCR is so sensitive that a small amount of contaminant can mislead the expected result. FISH uses a fluorescently labeled probe to notice nucleic acid sequences that hybridize mainly to the complementary sequence [9]. This process follows some steps and after that fluorescence microscopy assesses the signal [5]. It is not used regularly as PCR due to a couple of limitations. Probe consumption and hybridization time are key limitations of this procedure [58]. An enzyme-labeled antigen or antibody with several antigen-antibody combinations, ELISA measures enzyme activity. This procedure, enzyme-linked immunosorbent assay (ELISA), may detect a specific viral disease only. Flow cytometry (FCM) analyzes and detects the chemical and physical characteristics of cells, and is used for cell counting, and biomarker detection [20]. The technique uses an optical method, it utilizes a laser beam as light and produces fluorescent light signals using the sample. It has widespread applications for studying eukaryotic cells [1]. Though, it is endorsed that it works efficiently in terms of identifying soil-borne bacteria [1]. Almost all of these direct approaches of identifying plant disease work for detecting specific organisms. These techniques are not versatile enough to detect multiple diseases. Besides, it needs to be tested in the laboratory to detect the organism causing the disease. In some techniques, the presence of some pollutants can affect the result badly. Also, it is not a quick process though.

Indirect Thermography is a process that makes possible identifying various plant leaf diseases by observing thermographic imaging of external temperature of plant leaves [20]. Various reports and datas have shown that many plant diseases occur due to the loss of water in stomata. This process can also detect the most amount of water without any outermost temperature influences. On top of that, thermography is a really favorable method to keep track of the heterogeneousness of disease like soil-borne pathogenic infections. Insole of being a promising technique, the implementation of thermography for detecting disease is bounded because of its sensitivity towards the change of ecological circumstances throughout the time of experimental mensuration. Besides, thermographic detection is insufficient in case of identifying the specificity of diseases. Hence the method is unable to discriminate between diseases that generate almost identical thermographic patterns [20]. Along with Thermography, processes like Fluorescence Imaging, Hyper spectral techniques, Gas Chromatography are included in the Indirect method. Fluorescence technique is used to find out the deformities in photosynthesis and to examine the bacterium infection through observing the change of photosynthetic structure and electron transport reaction [20]. But this process is not highly encouraged for experimental usage due to a couple of complications. Gas Chromatography is a process that makes it possible to detect the plant leaf diseases at various phases by collecting quantitative statistics from the volatile organic compounds(VOC) samples [20]. Because of its high specificity, Gas Chromatography can come up with more precise information about plant diseases compared to other existing direct and indirect methods. Yet, its practical implementation is limited as it is a really time-consuming process. Before analyzing the data, pre-collected VOC samples are needed for prolonged time in case of Gas Chromatography, which makes the process time-consuming.

On the other hand, we are using a couple of deep learning methods and image processing to determine whether the tomato is healthy or not. Also, it will detect diseases if the plant is unhealthy. This approach is completely different from the direct technique. This technique will resolve the issues of the direct approach to disease identification. In our model, it will first take an image of the targeted tomato. Then it will differentiate the pattern of a specific portion. After that, this pattern will be sliced into several layers. Then the deep learning architectures which are used will be run. The whole process will take a very short time compared to those direct techniques. Using our model, it will be possible to identify five different types of diseases. As the agriculture scenario of our country says, the farmers are not so familiar with modern technologies. This model will be very user-friendly even for them. As deep learning and image processing provides a feasible solution, lots of research is being made on this subject using different models of deep learning. We have selected some architectures considering the accuracy and operation parameters in priority. Inception, ResNet, and VGG are the selected ones. Among them, inception works more efficiently. There are a number of research papers where Inception, ResNet, and VGG are used. But it is hard to find one where the architecture provides satisfactory accuracy in different species of tomato. We will train and prepare the models with a large dataset. Then this model might work for different species of tomato with appropriate result accuracy.

1.5 Problem Statement

Since the density of the population is higher than usual and with this much population it is very difficult to meet the requirement of food for this growing population day by day here in Bangladesh. However, Bangladesh being an agricultural country, rice and other crops are playing a huge role in this sector. Then again we are very much aware how many other crops especially vegetables are being consumed by the people, that is why to fulfill the demand we need to work on creaisn the production rate and the quality of the most consumed vegetables among the people with the help of new technologies. Talking about vegetables, tomatoes are one of the most consumed vegetables by the people of the country, again Tomatoes also plays a huge rule on our nation's GDP too. But To produce more and get the expected result while cultivating we need to adapt more new technologies, so we have decided that to use Artificial intelligence with the help of computer science to build an automated system that can detect the diseases of tomato plants.

In most of the cases it is seen that farmers can not even detect the decease at a very early stage which results in destroying most the plants, and it decreases the production rate to a great amount. Therefore, it seems very important to detect the disease at an early stage so that the farmers can come up with a suitable solution. Our main purpose for this paper is to detect the problem at a very early stage. To accomplish our purpose for this paper, we are using image processing techniques.

[38] Durmus, Olcay and Kirci have described in their paper how they have used Deep learning methods to detect the diseases of tomato leaves which are usually occurred by using the chemical methods and pesticides, but these chemical methods increase the production rate but eventually this method of using chemicals and pesticides increases the production costs and also increases the chances of getting affected by various virus related disease's tomato leaves. So while increasing the production rate the main problem which arises here is that the time gap between getting affected by the virus and getting detected, so the main problem here to resolve is to detect the disease using our huge data sets of healthy and unhealthy leaves at a stage.

In our paper, we are trying to investigate the leaves for detecting the disease to resolve the problem that we are trying to resolve. And for this we need to use the best and advanced image processing techniques and models. [47] Turcer et.al, [37] Dhaware and Wanjale, [65] Rao and kulkarni have added in their paper that the main advantage of using CNN model is that it can extract features from raw images automatically and this is what we are actually truly interested in. To make our purpose fulfill it is necessary to create a system using which farmers can use their device to capture raw images and then using our pre-prepared datasets we can detect the features and make a good prediction of which disease the leaf is affected automatically. Then our main purpose will be served.

Another problem that farmers in our country face is that of the temperature issue while planting tomatoes. Even though there are various kinds of tomatoes, and it can be planted, and it can grow all year around but accordion to [30] Hasan Bai, they have mentioned In Bangladesh the there are two kinds of tomatoes that very much famous among the farmers which are Bari Hybrid Tomato and Summer Tomato. The authors of this paper have also studied that hybrid tomatoes can grow all year around, but the peak months for this type of tomato is from December to March. However, for Summer Tomatoes, there are some problems that arise because of extensive heat. [59] Grant has studies in her blog that Hybrid tomatoes can be affected by various diseases if the temperature goes up to 32 degree Celsius, where In Bangladesh it is very much normal for the farmers to experience temperatures going up to 35-56 degree Celsius. So this is where we are trying to make a good scope to detect the diseases for these tomato plants at a very early stage.

To conclude, the main reason why we are trying to make a system to detect the disease automatically. Because the farmers in our country rely more on chemical and pesticide based solution to increase the growth on the other hand it is not enough to depend on chemicals fully because if the disease can be detected at an early stage than it is very much possible for the farmers to take action after finding the reason behind the disease that is affecting and also possible to stop spreading the disease at a large scale.

Chapter 2

Literature Review

In this paper, they have proposed a model which can identify the disease as well as the amount of it. It works in two phases. In the first phase, the plant is recognized by preprocessing and extracting features of images using an Artificial neural network. In the second phase, they classified the disease using K-Means and ANN algorithms. In terms of grading the disease, they have considered the pixels of the diseased area. In the case of calculating the percentage of the severity of the disease, calculating only the area might not tell the actual severity of the disease. They could use the color of the pixels too. For example, Late blight produces blackish/brown spots on leaves and stems. It contains chlorotic borders in the initial stage, but the entire leaf becomes necrotic in a few days. So, in case of determining how damaged the crops are due to Late blight, using color pixels may increase the result accuracy [24].

In this paper, their proposed method automated the process of identification of 4 types of leaf diseases. CNN and LVQ are mainly used for the identification and classification of these diseases. To start convolutions in CNN they also used three different matrices for R, G, and B channels. Learning vector quantization is used for classification. By calculating the Euclidean distance, they selected the closest reference vector. The reference which is the closest to the input vector is chosen. Then they converted the R, G, and B matrices to 27×1 matrices to prepare these for the input layer of the neural network. Besides, they have used 400 images only to train the model and 100 for testing it, a larger dataset might give a more accurate result. Another point is, this model might not provide an accurate result of the severity of these four types of diseases. They could decide by counting affected pixels or analyzing the texture of the diseased area, or approaching it similarly [48].

In this system, it detects and classifies two types of grape leaf disease, which are Downy Mildew and Powdery Mildew. In the image acquisition module, grape leaf images were collected using a digital camera which is used in the dataset. They have used a Gaussian filter to decline noises from the images and K-means clustering to separate the affected area of a grape leaf. Then classification technique is used to classify the disease based on both color and texture. They have used 9 color features and 9 texture features combined to classify as well as differentiate diseases with higher accuracy. It provides 83.33% accuracy in terms of the classification of Powderly disease. But this system might have been more accurate in the case of detecting Powdery Mildew disease if they would use CNN instead of SVM [34].

In this paper, they proposed a model for detecting ten types of tomato leaf diseases using LeNet. LeNet is a modified version of CNN. Dataset was collected from the plant village repository. The size of the images of the dataset was decreased to 60 $\times$ 60 resolution to enhance the processing speed. They have taken the Z-score of the image pixels. They tried to use a simpler architecture of convolutional neural networks with fewer layers. It is constructed with an extra block of convolutional, pooling layers and activation, which is a variation of LeNet. Generally, it requires 7 layers to form LeNet-5. The layer composition is with 3 convolution layers, 2 subsampling layers, and 2 fully connected layers. Here, for feature extraction, the first two (convolutional and pooling layers) are used, and they use connected layers for classification. They applied Keras to implement the model, with a resulting 94-95% of accuracy. It has diverse uses for different diseases. This model can classify ten different diseases. They could use optimizers to increase the resulting accuracy of the model [49].

Here, the authors compared Deep learning and image processing methods with detecting citrus plant disease. They have chosen to compare SVM, RS, and SGD with Inception-v3, VGG-16, and VGG-19. They have considered several parameters like precision, f1 score, accuracy, and area under the curve for comparisons. Furthermore, they have prepared the dataset with the guidance of experts from the research center and the government. Images of healthy and infected citrus leaves were captured using a DSLR with the size of 256 $\times$ 256 pixels. To compare the methods, a confusion matrix table was prepared. It provides clear information on the right and wrong class mapping. It classifies the prediction of the taken ML and DL methods. Every ML and DL method got different confusion tables. Pie charts are also displayed to represent the information. They used k subsamples where k-1 is used for training the model and the rest is for validating the model testing. After using a similar dataset in both ML and DL models, it shows that every DL model (Inception-v3, VGG-16, VGG-19) provides a more accurate result than the ML (SVM, RS, SGD) [71].

Various kinds of plant diseases and classification techniques of ML which will be used to differentiate the diseases are discussed in this paper. Plants are mainly affected in three ways which are bacterial, fungal, and viral diseases. The diseases like leaf rust, bacterial blight, brown spot, mottle etc. are the result of bacteria, fungi, and viruses. They classified the algorithms into two types, which are -supervised and unsupervised algorithms. In the unsupervised classification section, k-means, LDA, and fuzzy C-Means are included and explained. Here, the Fuzzy C-means algorithm provides optimal results though the data is uncertain but takes a longer time as well as sensitivity to noise. Whereas K-means computationally faster but provides no guarantee of giving optimal solutions. It is also difficult to determine the number of clusters. On the other hand, ANN, CNN, NLP, and SVM use supervised data to process. KNN, which performs statistical estimation and pattern recognition, is flexible and works faster with training data which has numerous outliers, but KNN is costly in terms of computational cost. In ANN, Probabilistic Neural Network (PNN) which uses a feed-forward algorithm provides sufficient accuracy as well as works faster. Here, Fuzzy-Relevance Vector Machine provides enough accuracy with unbalanced data. It reduces the outliers and uses Bayesian inference. They showed which models work effectively in different scenarios. These results will be beneficial for any future work using these models [51].

The authors of this paper applied CNN to detect and classify the diseases of Tomato leaf. They have used 3 convolution and max pooling in the CNN model. They have collected the dataset from Plant Village. Furthermore, they have collected sufficient data for identifying 9 types of diseases which is 10000 images where 1000 are of the healthy category. To detect and differentiate the diseases with better accuracy, they have used CNN with the panda approach. Though in some cases they get 100% accuracy, 76% is the lower bound. This lower bound could be improved using optimizers or modifying Convolutional neural networking. This model can classify nine different diseases, which are useful [56].

In this paper, the proposed system detects and classifies two grape diseases which differentiate color firstly then segment disease using it and then classify the segmented disease. It recognizes color using back-propagation neural networking. BPNN basically extracts the leaf image from the background. For detecting diseases, a self-organizing feature map with a genetic algorithm for optimization is used. Besides, Gabor wavelet and support vector machines are also used to classify diseases. They could design the model for more diseases to classify than two, as there are more common diseases of a grape leaf. The average result of this model is 86.03% which could be improved. Diseased areas and textures of disease could be considered in this model to increase the accuracy of the result [11].

In this research, this model can classify 9 types of different tomato leaf diseases and can separate healthy and infected leaves. They also compared 5 CNN models. Along with 5 deep network structures (Resnet50, Xception, MobileNet, ShuffleNet, Densenet121_Xception), transfer learning is also used to reduce computational costs. During building the deep learning models, transfer learning makes it much more effective. It basically can transfer the learning from a pre-trained model to a new model. Here, Densenet_Xception provides the highest accuracy among the models, which is 97.10%. Though it has the most parameters. Whereas, ShuffleNet has the second-lowest accuracy, which is 83.68%, but the parameters are small. The accuracy of the rest of the three models Xception, Resnet50, and MobileNet are 93.17%,86.56%, and 80.11% respectively. They clearly differentiate the models with their accuracy in terms of identifying 9 different tomato leaf diseases. As their main goal was to compare the models, they could include more structure of CNN. They might take a couple of ML models and compare the accuracy of this too [60].

They proposed a deep-learning architecture which is called EfficientNet for the classification purpose. They choose both plain and segmented images for the classification. They also compared the performance of binary, six class, ten class report. In the dataset, 18,161 images of tomato leaf are used along with same number of segmented leaf masks where 20% of them were used for testing. They made 10 classes where one of the classes is healthy and rest of those are not healthy. To identify the best performance metrics they used different kinds of loss function named BCE, NLL and MSE. They used EfficientNet-B0, EfficientNet-B4, EfficientNet-B7 for the disease classification. They got 98.66% accuracy with the modified U-net segmentation model. EfficientNet-B7 showed 99.95% accuracy in binary classification. They got more accuracy with EfficientNet which is better than most of the relevant papers. This model can beat some of the existing model's performance. But this model could perform better if the number of attributes could be increased. Besides, six-class classification with Efficient-B4 should have provided more accurate results [69].

In this research paper, the author has discussed identifying diseases on plant leaves by observing the images of both healthy and infected leaves through the methodologies of deep learning, along with that, this paper also shows the development of CNN models to carry out the plant disease detection process. In order to differentiate between the healthy and infected plants, in this work we can see the development of specialized deep learning models depending on particular CNNs architectures by using the images of leaves. The author selected this basic deep learning tool 'CNNs' for performing the task because CNNs is used in cases of pattern recognition in images. The author showed some previous works that had major drawbacks, which was, the complete procedure of photographic materials only contained laboratory images instead of real conditions. However, this work overcame the drawback as it successfully used images of both real cultivation and experimental setups. The work mainly discussed the two models for accomplishing the proposal. These are convolutional neural network models and optimal deep learning model. It was confirmed that, all the models of optimal deep learning gained 100% accuracy on the experimental set. Out of total dataset of 87,848 images including 25 different plants along with 58 distinct classes, this model gave 99.53% result in observing the disease which proves that, this paper rarely has any drawback to improve [45].

The authors of this paper made a comparison of different CNNs models to identify Tomato Plant diseases which are AlexNet, Inception v3, ResNet 18, ResNet 50 and GoogleNet. The paper contained the dataset of nine various kinds of tomato diseases along with a healthy class of tomato plants. A collective mathematical analysis depending on accuracy, preciseness, specificity, sensitiveness, AVC, ROC (receiving operating characteristic) curve etc., these data were used in order to evaluate the 5 models of CNNs architectures. In the methodology segment, data acquisition, training, classification and evaluation, these 4 steps were shown that divided the whole operation. The authors described the procedure of each model individually under these 4 steps, which later gave an exact result in order to view the comparison between these 5 models in different sectors. After completing the task of methodology, the work showed different ratings on the performance based on the given statistical analysis chart for all these models. After viewing the result, the authors made it clear that, though all the models performances are almost the same, GoogleNet has the most numbers of success rate among all, that can be easily used by the farmers. GoogleNet achieved the highest AUC result which was 99.72% whereas Inception V3 and ResNet 18 had the lower results. Apart from the AUC, the other performance measures also had the same results, where the higher rate was achieved by the GoogleNet while the lower one was by Inception V3. Also, AlexNet performed with the lowest amount of time, this paper showed this as well. The authors successfully compared 5 models on different bases and showed their performances on individual stages. It would be better if the authors could compare a couple of ML models and compare the accuracy rates [64].

In this paper, the proposed system classifies tomato leaf diseases through the usage of Convolutional Neural Network. The authors of this paper analyzed various kinds of CNN architectures such as VGGNet, LeNet, ResNet50, and Xception in order to detect the diseases from tomato leaf. They tried to show the accuracy rate along with the amount of loss through all these 4 architectures. It is a good thing that all the applied CNN architectures gained more than 91 accuracy rate in the test. But in comparison with other models, VGGNet delivered the best result by gaining 99.25% test accuracy along with a very little amount of loss, which is 0.03 in color images. VGGNet also achieved the best outcome for segment images as well, here the accuracy rate and loss amount were 99.11% and 0.04 respectively. The paper tried to differentiate among these 4 architectures by dividing the whole procedure into two segments, which are for color images and for segmented images. Though the result showed VGGNet as the most useful architecture, but it cannot be used widely for being very expensive. The major drawback of this proposed system is that, the training procedure mentioned in this paper is really time-consuming along with that, it requires hardware configuration that are high-end as well. They could use the architectures which fits in smaller dataset to reduce time-consuming [55].

Here, the authors of this paper proposed to detect the diseases in tomato plants using image processing method with the help of multi-class SVM algorithm. They showed to detect four types of tomato plant diseases on the initial stage. The whole procedure was done into two steps where in the first stage they used image segmentation to separate the infected damaged areas of leaves from the other parts, and in the second stage, Multi-class SVM algorithm was used in order to classify the accurate disease. The four key diseases that they have worked on are Early Blight, septoria Leaf spot, Iron Chlorosis etc. Firstly, they used cameras with high resolution for capturing the images of leaves and later, the infected areas were separated through using segmented image feature extraction. In the next section, they calculated the overall accuracy rate for each of the mentioned diseases by using Multi-class SVM model in MATLAB 1 5a – for the disease classification process. Here the accuracy rate was overall 93.75% where Early Blight, Septoria leaf spot, Bacterial Spot etc. have given the result of 100%, 91.67%, and 91.67% respectively. They also compared their proposed model with other architectures as well, such as ANN Technique, Eigen feature, PNN (Probabilistic Neural Network). Here they prove their proposed model to achieve the best result of all. But they could use high-end CNNs architectures to get more accurate rates for up to 98-99%. Also, they could focus on classifying more diseases of tomato plants instead of only four [43].

In this paper, the authors compared among the four CNN architectures in order to select the most optimized model for the purpose of classifying as well as identifying the tomato leaf diseases. They selected VGG-16, VGG-19, ResNet and Inception V3 as CNNs models along with the usage of feature extraction and parameter-tuning. The good part for their proposed system is that, they tried to experiment on both laboratory-based dataset and real-life cultivation process. Their work on two different datasets were used to observe the models in a better way. It was noticeable that, all these models give better result on laboratory-based datasets. In the methodology part, they used the four CNNs models mentioned above with separate datasets. For the first self-collecting dataset, they collected tomato leaf data from uncontrolled natural environment to detect those data into several categories. On the other hand, they focused on identifying 4 types of tomato leaf disease from 2364 images that were contained in the lab-based dataset. They used rate of accuracy, precision, recall and F1-score, the 4 segments as the assessment criteria for the performance of each model. Later on, they calculated the values on these 4 evaluation metrics individually. Among the 4 used CNNs architectures, Inception V3 showed the best result with giving accuracy of 93.40% for laboratory-based dataset. Inception v3 gave the best performance in real-life dataset as well, though the average rate was 10%-15% lower. They represented real-world scenarios by using the self-collected dataset, which made their work different from previous works. But the models they have used cannot work efficiently on real fields. So these models cannot be approachable by the farmers [57].

Chapter 3

Datasets

3.1 Description of data

To ensure that the findings are more uniform, only photos featuring leaves were used in this study [28] [61]. As a result, the image dataset utilized is derived from the publicly available Plant Village dataset [33] [28]. Hughes and Salathe [22] generated the Plant Village dataset, which comprises 61,486 tagged photos for 14 separate species and 39 different classes containing images of healthy and sick leaf images.

We extracted characteristics from the Plant Village dataset using just 16,012 photos of tomato leaves, sorted into one healthy class and nine harmful classes [44]. Each picture is made up of one leaf and one backdrop. The following histogram(Figure 3.1) shows the uneven distribution of the images each classification from Plant Village Dataset.

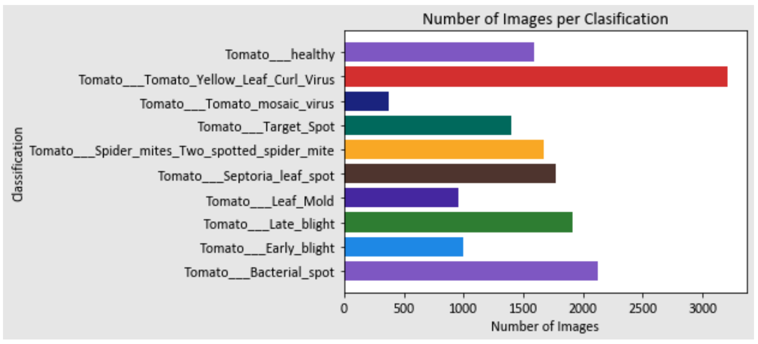


Figure 3.1: Number of images in each classification from Plant Village Dataset.

[70] The specific classifications of the dataset are as follows, after integrating the original tomato leaf photos and removing superfluous categories: Healthy, Bacterial spot, Early blight, Late blight, Leaf Mold, Septoria leaf spot, Target Spot, Tomato mosaic virus, Tomato yellow leaf curl virus, Two-spotted spider mite [44].

Table 3.1 demonstrates how well the database is organized by illness. Images were snapped using various digital equipment and cellular smartphones [28]. Approximately 25% of the photos were obtained under controlled settings, while the remaining 75% were captured under natural conditions, with the leaves not connected

Class	Unhealthy					Healthy
	Fungi	Bacteria	Mold	Virus	Mite	
Sub Classes	Early blight (1000)	Bacterial spot (2127)	Late bright (1909)	Yellow leaf curl virus (3209)	Two spotted spider mite (1676)	Healthy (1591)
	Septoria Leaf spot (1771)			Mosaic virus (373)		
	Target spot (1404)					
	Leaf mold (952)					
Total Tomato leaf images (16,012)						

Table 3.1: Details number of images of tomato leaf photos for healthy and unhealthy classifications [22].

to the host plant [28]. The size of tomato leaf images are 256x256 and all the images are in RGB color mode.

In addition to the Plant Village Dataset, 6,918 images have been added after augmentation to ensure the balance of images in the dataset. There are 6 different augmentation methods that have been applied. The methods are,

1. Scaling: the images were scaled into 256 x 256 pixels. The focus was to make the leaf stay at the center of the image.

2. Rotation: the image was rotated such that the tip of the leaf is in the upper side of the image and the axial part on the lower side of the image.

3. Noise Injection: the random variation of brightness of the images was set to an average.

4. Flipping: Some images were flipped such that the tip of the leaf is in the upper side of the image and the axial part on the lower side of the image.

5. gamma correction: The brightness and luminosity were adjusted.

6. PCA color augmentation: The intensity of RGB channels in the images were adjusted.

Due to this uneven distribution, the model could lead us to overfitting issues. The additional images were collected from another well known dataset from kaggle to the right classes. Figure 3.2 shows the distribution after additional images were added.

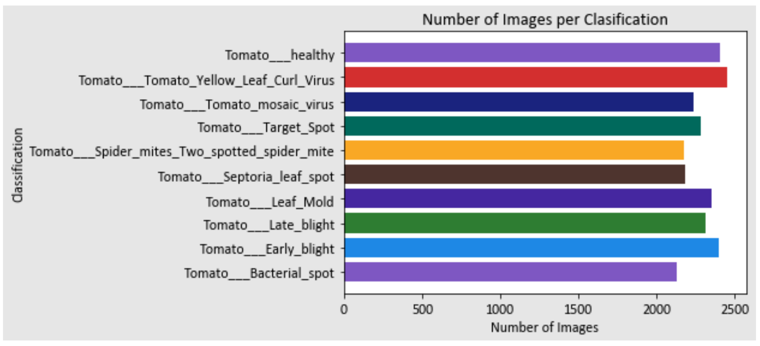


Figure 3.2: Number of images in each classification after Augmentation.

Class	Unhealthy					Healthy
	Fungi	Bacteria	Mold	Virus	Mite	
Sub Classes	Early blight (2400)	Bacterial spot (2127)	Late bright (2314)	Yellow leaf curl virus (2451)	Two spotted spider mite (2176)	Healthy (2407)
	Septoria Leaf spot (2181)					
	Target spot (2284)			Mosaic virus (2238)		
	Leaf mold (2352)					
Total Tomato leaf images (22,930)						

Table 3.2: Details of tomato leaf photos for healthy and unhealthy classifications [68] after augmentation.

[70] The specific classifications of the dataset are the same as the PlantVillage Dataset. The classifications are as follows: Healthy, Bacterial spot, Early blight, Late blight, Leaf Mold, Septoria leaf spot, Target Spot, Tomato mosaic virus, Tomato yellow leaf curl virus, Two-spotted spider mite [44].

Table 3.2 also demonstrates how many images there are in the database are organized by sickness of the leaf. Images were taken using various digital cameras and smartphones [28].

The following figure shows the random sample images of train, validation and test data. 70% data used to train the model, 20% data were used to validate the model and 10% data used to test the model.

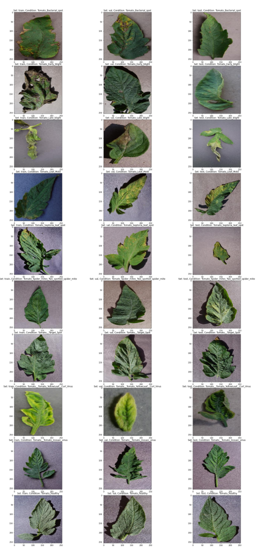


Figure 3.3: Sample images from the Augmented dataset showing trained, valid, and test data [14].

3.2 Data Pre-processing

Data preprocessing steps are the steps that we needed to cover in order to transform our image data into some useful data stream or some of the models that we are going to use.This pre-processed data is used to train our desired and used models such as CNN, Inception V3 and ResNet50 can train by extracting the features from the image dataset. Image pre-processing is a very crucial step before we can dive into training our model. It is because if the classes of dataset are distributed in an uneven way, then it is seen that our used models are unable to train themselves properly to classify the exact class while predicting.

Firstly, In order to start pre-processing before even starting to train our machine learning models our first job is Image Acquisition, here the dataset containing the images is collected from the Plantvillage tomato leaf data set containing 10 classes of tomato leaf diseases [22].

Secondly, the major step of data preprocessing is augmentation over the dataset. Here in our paper, we have already discussed in our previous section that the dataset needs to be augmented so that we can prevent the problem of uneven distribution of data. Initially, there are 6,918 images but after completing the steps of augmentation we achieved a total number of 16,012 images. Here in our models, we have used Keras's ImageDataGenerator class to generate batches of images with real-time data augmentation with parameters like width_shift, height_shift, rotation, and horizontal and vertical flip to augment our image data.

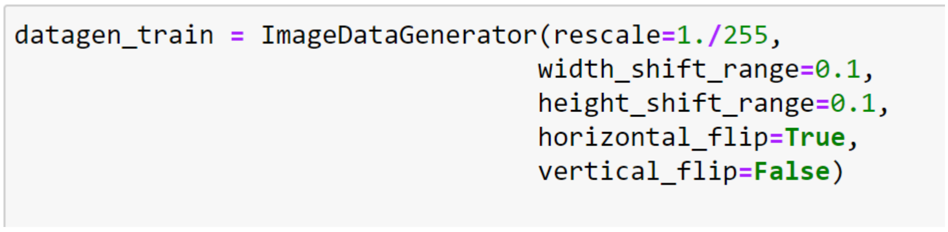

```
datagen_train = ImageDataGenerator(rescale=1./255,
                                   width_shift_range=0.1,
                                   height_shift_range=0.1,
                                   horizontal_flip=True,
                                   vertical_flip=False)
```

Figure 3.4: ImageDataGenerator class and its parameters.

In the above figure we have passed the parameter "horizontal_flip" and "height_flip" which ranges from 0 to 1 and by setting horizontal_flip = true the image data will be rotated in such a manner so that the tip of the leaves will be straightened and be on the upper side.

If the above figure 1 is considered which is actually done on our original dataset then for setting width_shift_range=0.2 the image will be shifted its x-axis 20% of its total image to the left side. An example of the result has been shown in figure 3.5 .

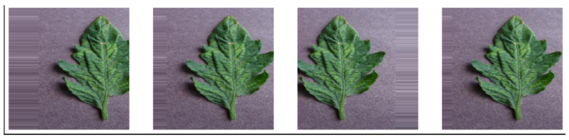


Figure 3.5: Example of images with width_shift_range =-0.2

Then again, considering the other parameter that has been passed, this parameter would also give the result of how the y-axis is being changed by setting height_shift _range=0.2. This also shifts the image by 20% to the upward to y-axis. An example

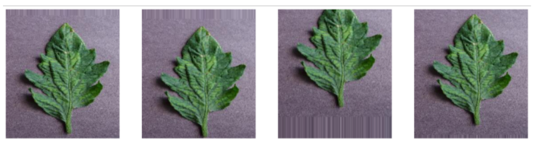


Figure 3.6: Example of images with height_range=0.2

of this changing y-axis is shown in figure 3.6 .

And lastly, the parameter that we use to augment our data is rotation_range. Rotation_range parameter is basically used to rotate an image under a given range from 0-360. Here for our usage rotaion_range=45° has been used as a result this parameter rotates each of the images by 45-degree angle. Therefore, this is how by using these parameters we have gone through with the data augmentation task.

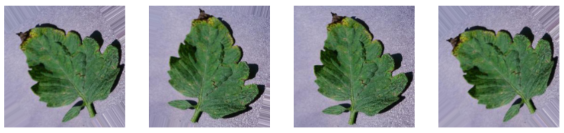


Figure 3.7: Example of images with rotation_range=45°.

Chapter 4

Methodology

4.1 Model Description

4.1.1 Convolutional Neural Network (CNN)

Around the 1980s, CNNs were first created and put to use and aside from providing the vision for robots and autonomous vehicles, ConvNets have proved effective at recognizing faces, objects, and traffic signs. Since 1988, there have been several versions of LeNet, the most recent of which, developed by Yann LeCun, is known as LeNet5 [7]. LeNet was among the earliest convolutional neural networks to advance deep learning. All major firms have been vying for this since Alex Krizhevsky deployed a convolutional neural network in the 2012 ImageNet competition [17]. Among the many significant advances in computer vision, CNNs stand out as the most important. LeNet architecture was mostly utilized for character recognition jobs in the 1990s, such as reading zip codes and other data.

Deep Convolutional Neural Network [67], which is a supervised machine learning model that has proven successful implementation in several Computer Vision and Image Processing fields [62]. CNN is commonly used for analyzing images by performing tasks related to image segmentation, classification, recognition, and retrieval [54] [63]. CNN is a multilayered hierarchical feedforward network where a set of convolutional kernels is used to execute several changes in each layer [13]. Convolutional Networks are trainable multi-layered architecture consisting of multiple layers [29].

The analysis of visual data is a common application of Convolutional Neural Networks (CNNs), a subset of deep neural networks. A convolutional neural network (CNN) is an artificial neural network (ANN) whose architecture is tailored to process data with a grid-like structure, such as an image, and thus learn spatial hierarchies of features.

Convolutional layers, pooling layers, and fully connected layers are typical components of a CNN's structure [50]. Image classification, object detection, and semantic segmentation are just some of the computer vision tasks where CNNs have proven to be highly effective. Natural language processing and voice recognition are just two more fields where they have been put to use.

Input layer

In a Convolutional Neural Network (CNN), the input layer is the first layer that receives the input data. The input data is typically an image, but it can also be other types of data such as audio, text, or time-series data. The input layer is responsible for preparing the input data for processing by the rest of the network. This may include tasks such as normalizing the data, resizing the data, or converting the data into a suitable format for the network. The input layer does not have any trainable parameters, it just passes the input data to the next layers in the network. The input layer is not always explicitly defined in the CNN architecture, it is considered as the first layer and its output is passed to the next layers.It is worth noting that the input data in CNN is typically a multidimensional array (tensor) with shape (batch_size, height, width, channels), where batch_size is the number of images in a batch, height and width are the dimensions of an image, and channels are the number of color channels in an image (for example, 3 for RGB images or 1 for grayscale images)

Convolutional Layer

The Convolution layer is the first layer of a Convolutional Neural Network (CNN), and it consists of convolution kernels (M x M) that extract patterns from the input pictures [68]. Each neuron in the figure is like a kernel that reads in an image and spits out information about it [29]. Features are extracted from the input data by a convolutional layer of a Convolutional Neural Network (CNN). Convolution is a mathematical process performed locally on the input picture by the convolutional layer, which consists of a series of filters (sometimes called kernels or weights). Afterward, the feature maps are sent on to the next layer. Tiny (3x3 or 5x5 pixel) filters are slid over the input picture to calculate dot products of filter values with the corresponding input values at each pixel. Dot product (or convolution) is an operation that modifies the feature map by multiplying it by a new value at each position. As the training progresses, the filters' weights are adjusted to reflect the information gained. Features like edges, corners, textures, and patterns are what the convolutional layer is designed to pull out of the input picture. The network may learn many characteristics by using a variety of filters on the input picture and then combining them to generate a prediction. A convolutional layer's hyperparameter is the number of filters it employs. Most CNNs employ convolutional layers in their first stages, while the network is still learning basic characteristics. The deeper the network goes, the more it is able to integrate these simple traits to learn more complex ones and make more nuanced predictions.

Nonlinear activation function

Non-linearity may be introduced into a neural network by use of a nonlinear activation function, which is a mathematical transformation added to the layer's result. Nonlinear activation functions enable a network to learn complicated patterns and carry out tasks like classification and regression by transforming the input into the desired output. The input is then fed into a nonlinear activation function after the results of a linear operation (like convolution) have been applied to it. Although smooth nonlinear functions, such as the hyperbolic tangent (tanh) or sigmoid func-

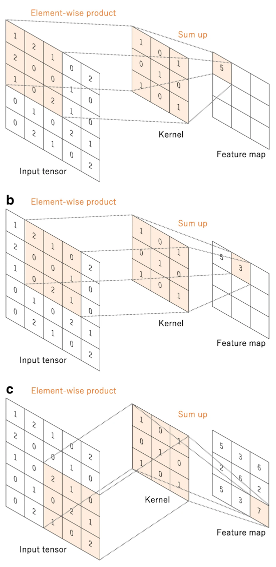


Figure 4.1: a–c An diagram of the convolution procedure using a kernel size of 3*3, with no padding, and a stride of 1.

tion, were formerly employed because they mathematically mirror the activity of biological neurons, the rectified linear unit (ReLU) is currently the most often used nonlinear activation function. f(x)=max(0,x) [39] [14] [40] [15]. The output of a layer is then subjected to one or more activation functions, the selection of which is task- and architecture-dependent. Softmax, for instance, is utilized for multi classification, whereas Sigmoid is often used for Binary classification [6] [66].

The most common nonlinear activation functions used in neural networks are:

• ReLU (Rectified Linear Unit): This function replaces all negative values with zero, allowing the network to introduce non-linearity and improve its ability to learn complex patterns.

• Sigmoid: This method is helpful for binary classification jobs since it converts the input to a value between 0 and 1.

• Tanh (hyperbolic tangent): This function maps the input to a value between -1 and 1, which is useful for regression tasks.

• Leaky ReLU: This function is similar to ReLU but it allows a small gradient when the unit is not active, this helps to avoid dying ReLU problems.

• ELU (Exponential Linear Unit): This function is similar to ReLU, but it tends to produce more negative outputs, allowing the network to learn more complex features.

• Maxout:All negative values are converted to zero in this version of the rectified linear unit (ReLU) activation function. Instead of returning zero for negative values, the Maxout activation function selects the highest positive value from the given collection.

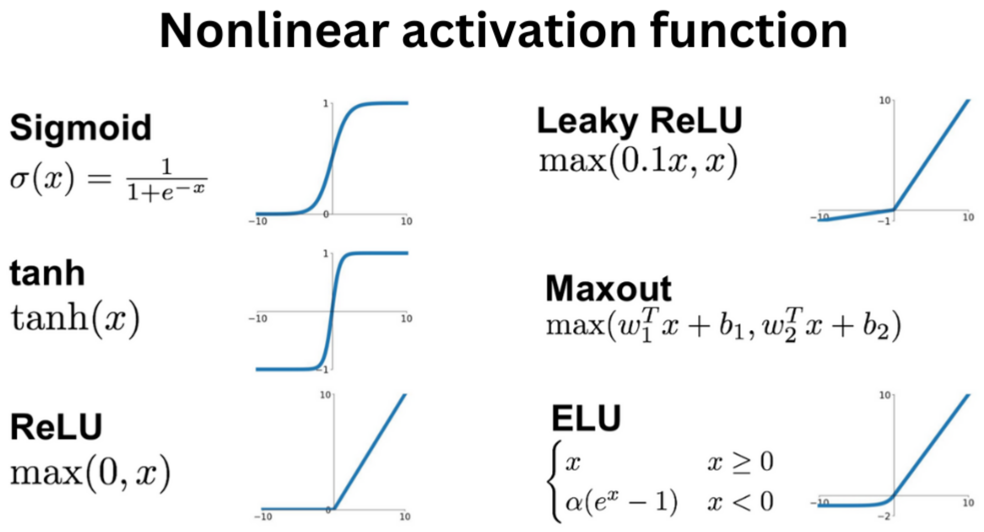


Figure 4.2: Different non-linear activation function.

Pooling layers

The relative placement of features is maintained after extraction [54]. When data is pooled, it is down-sampled, a local procedure [74]. The ascending response within the local area is then generated by performing a summing operation on comparable information within the receptive field [75].

The feature maps are down-sampled using pooling layers, decreasing their dimensionality and enabling the network to zero in on the most relevant characteristics. To make the network more computationally efficient and less susceptible to tiny translations or distortions in the input picture, pooling layers are used to minimize the spatial size of the feature maps. Because of this, overfitting is also mitigated. Note that although stride, padding and filter size are all hyperparameters in pooling operations, there is no learnable parameter in any of the pooling layers themselves. It's because convolution operations and pooling operations are quite similar [50].

There are various types of down-sampling methods used in CNN such as max pooling, average pooling, overlapping pooling etc. [75].

• Max pooling: The feature map is partitioned into many non-overlapping sections, or "pooling windows," to facilitate the process. The max pooling layer chooses the largest value in each window as the output for that window. The result is a feature map with a smaller pixel size, achieved by retaining just the most crucial details. When the network has to be resilient to slight changes in the location of an item in an image, as is the case with tasks like image classification, max pooling comes in handy. Overfitting is mitigated as a result of the feature maps' reduced dimensionality, which allows just the most crucial details to be retained.

• Average pooling: It is similar to max pooling, but it selects the average value within the window, instead of the maximum value. Average pooling is less commonly used than max pooling, as it is less robust to extreme values, which are usually the most important values for feature extraction. However, some architectures use average pooling as an alternative to max pooling, as it can help to smooth out small variations in the feature maps.

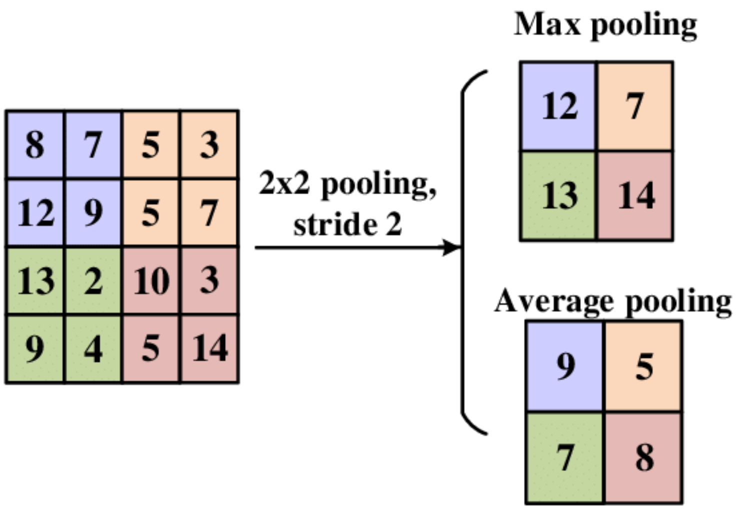


Figure 4.3: Sample diagram of pooling layer.

CNN can be overfitted with training dataset. To regulate this issue, dropout comes out. In dropout, it randomly skips some connections with a certain probability, thus the model is not overfitted and the performance of the model improves [16].

Flatten layer

The output of the preceding layers can be transformed into a one-dimensional vector by using a flatten layer. The fully connected layers that follow the convolutional layers can only handle input in a single dimension, so this step is essential.

The preceding layers' output, which normally has a three-dimensional shape (height, width, and number of channels), is transformed into a one-dimensional shape by the flatten layer (height x width x number of channels).

Fully connected layer

Since it is a global operation, it is often utilized before the output layer [41]. When building a deep neural network, it is common practice to flatten the feature maps produced by the last convolution or pooling layer, converting them into a 1D array of integers (or vector), before feeding them into one or more fully connected layers, sometimes called dense layers, in which each input is coupled to every output by a trainable weight [50].

Fully connected layers, which make up the last layers of a CNN, are in charge of creating a prediction based on the characteristics that were retrieved. A probability distribution across a number of predetermined classes is generally the result of the last fully linked layer.

Taking the output of the preceding layers, which may have a complicated structure, and turning it into a prediction or classification is the function of a fully connected layer. An optional bias term, an activation function, and a dot product between the input and a set of weights are used to achieve this.

A fully connected layer contains a set of weights, which are also known as parameters and are discovered during training. When the input is multiplied by these weights, a new set of values is produced as the dot product. These values are multiplied by the bias term, and the result is then subjected element-by-element to the activation function.

The number of outputs, which also influences the number of neurons in the next layer, is determined by the number of neurons in the completely linked layer. The dense layer may represent functions that are more complicated the more neurons there are in it.

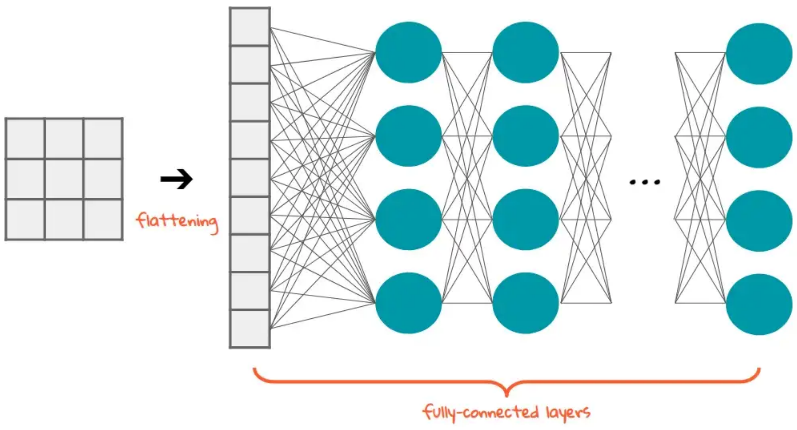


Figure 4.4: Diagram of flatten and fully connected layer.

Dropout

Dropout is a technique that can be used on any layer's output and is not thought of as a separate layer in a neural network. In order to prevent overfitting in neural networks, the regularization technique known as "dropout" randomly removes a predetermined percentage of neurons during training.

Dropout can be applied to other types of layers, such as convolutional or recurrent layers, but in practice it is typically only applied to fully connected layers. The percentage of neurons that will be dropped out during each forward pass is indicated by the hyperparameter known as the dropout rate, which can be altered depending on the architecture and the nature of the problem.

CNN can be overfitted with training dataset. To regulate this issue, dropout comes out. In dropout, it randomly skips some connections with a certain probability, thus the model is not overfitted and the performance of the model improves [16].

The completed architecture incorporates each of the aforementioned ideas into a unified whole(figure 4.5).

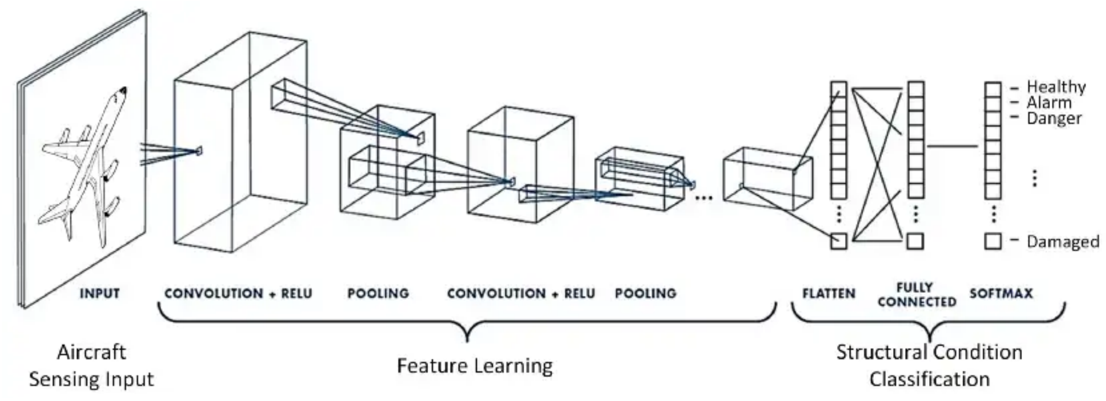
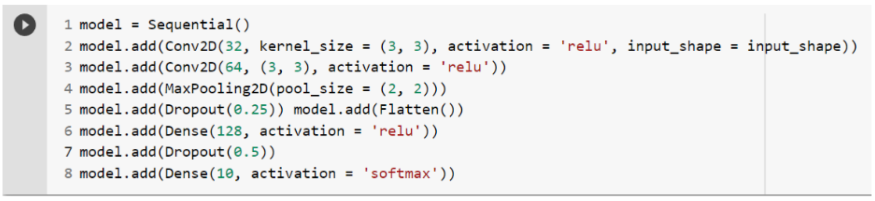


Figure 4.5: CNN architecture diagram

Sample Code



```
model = Sequential()
model.add(Conv2D(32, kernel_size = (3, 3), activation = 'relu', input_shape = input_shape))
model.add(Conv2D(64, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.25)) model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation = 'softmax'))
```

Figure 4.6: CNN architecture diagram

4.1.2 Inception V3

Inception-v3 is a supervised deep learning method that is effective at recognizing patterns with low computational cost. Introduced by Szegedy et al. in 2015 [35], inception-v3 is a popular method for training deep neural networks [32].

It was created by Google and released as a component of the Inception architecture in 2015. The network can categorize photos into 1000 different item categories after being trained on more than a million photographs from the ImageNet database. The Inception V3 architecture is distinguished by the use of batch normalization to accelerate training and decrease overfitting, as well as by the usage of Inception modules, which are tiny networks nestled within larger ones.

It was created by Google and released as a component of the Inception architecture in 2015. The network can categorize photos into 1000 different item categories after being trained on more than a million photographs from the ImageNet database. The Inception V3 architecture is distinguished by the use of batch normalization

to accelerate training and decrease overfitting, as well as by the usage of Inception modules, which are tiny networks nestled within larger ones.

To "get deeper" is the fundamental objective. The model that was suggested included 27 layers, including inception layers. The inception layer is composed of the combined filter banks from the 11 convolutional layer, the 33 convolutional layer, and the 55 convolutional layer. The combined filter banks were concatenated into a single output vector, which served as the input for the next step [35].

Inception-v3 is a newer version of the Inception neural network architecture, introduced in 2016 [35]. It has several improvements over previous versions, including the use of a more efficient factorized 7×7 convolutional layer. Inception v3 is successful in recognizing objects in the ImageNet dataset, which is more than 78%. Convolutions, max pooling, average pooling, dropout, concatenations and fully linked layers are some of the symmetric and asymmetric building pieces that make up the model. The model extensively applies batch normalization to the activation inputs. The loss is computed using Softmax [46] [52].

A number of substantial adjustments have been made to the Inception V3 model, including the following:

• Factorized Convolutions: Inception V3 uses a technique called "factorized convolutions" to reduce the number of parameters and computational cost in the convolutional layers. In addition to that, it monitors the effectiveness of the network.

The idea behind factorized convolutions is to decompose a standard convolution operation into two smaller convolution operations with fewer filters. In the case of Inception V3, factorized convolutions are used to decompose a standard 3x3 convolution into two smaller 1x3 and 3x1 convolutions. This allows the network to use fewer filters and reduce the number of parameters, while still capturing the same types of features as a standard 3x3 convolution.

Additionally, factorized convolutions are also used to decompose a standard 5x5 convolution into two smaller 3x3 and 5x1 or 1x5 convolution. This technique allows the network to use fewer filters and reduce the number of parameters, while still capturing the same types of features as a standard 5x5 convolution.

• Smaller convolutions: Increasing the number of smaller convolutions in place of larger ones will most certainly result in quicker training. Let's say a 5 x 5 filter has 25 parameters; two 3 x 3 filters working in conjunction to replace a 5 x 5 convolution have just 18 parameters (3 * 3 + 3 * 3).

Inception V3 also uses smaller convolutional filters (i.e., 1x1 convolutions) in some of its branches [53]. These 1x1 convolutions are used to reduce the dimensionality of the input before it is passed through the larger convolutional filters. This allows the network to focus on specific features while also reducing the number of parameters and computational cost.

In particular, a 1x1 convolution is used as a bottleneck layer before the 3x3 and 5x5 convolutional layers in some branches of the Inception modules. This allows the network to reduce the number of input channels and the number of parameters before passing the data through the larger filters. This in turn helps to control the overfitting by reducing the number of parameters, and also allows the network to focus on more abstract features.

In summary, the Inception V3 architecture uses a combination of smaller convolutional filters, factorized convolutions, and larger convolutional filters to extract features from the input image. By using convolutional filters with fewer layers, we may minimize the input's dimensionality and keep the number of parameters under control, both of which boost the network's computational efficiency and accuracy.

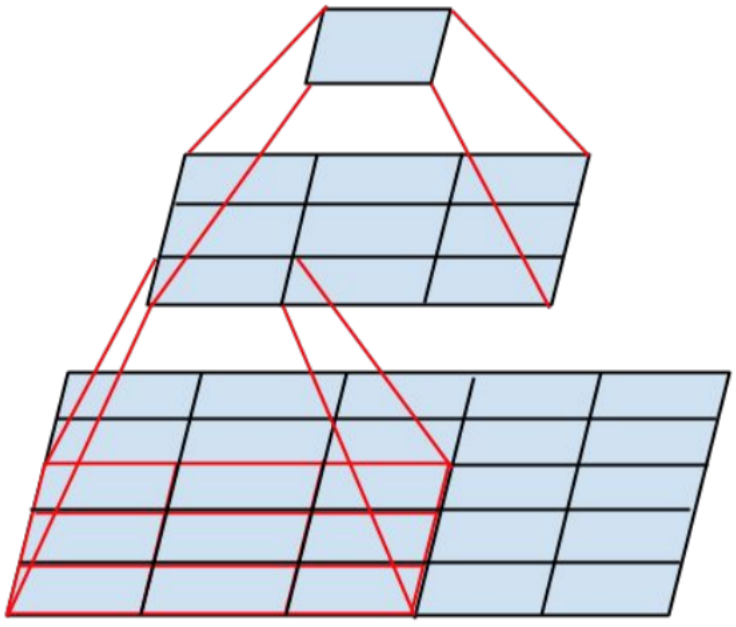


Figure 4.7: Mini-network replacing the 5 $\times$ 5 convolutions [35].

Above the fully-connected layer is a 3x3 convolution, which can be found in the center of the image. As a result of the fact that the two 3x3 convolutions may share weights with one another, it is possible to do fewer calculations.

• Asymmetric convolutions: Asymmetric convolutions, a technique where different filter sizes are used in the horizontal and vertical dimensions of the convolution. The Inception V3 architecture uses standard symmetric convolutional filters such as 1x1, 3x3, and 5x5 filters, which are applied uniformly in both the horizontal and vertical dimensions of the input. These filters are used in various branches of the Inception modules to extract features at different scales and abstraction levels.

Asymmetric convolutions have been proposed in some recent architectures as a way to further reduce the number of parameters and computational cost while still capturing the same types of features as symmetric convolution.

A 1x3 convolution might be used instead of a 3x3 convolution, and then a 3x1 convolution could be used after that. The number of parameters would be somewhat more than what was recommended for the asymmetric convolution if a 3x3 convolution was changed to a 2x2 convolution.

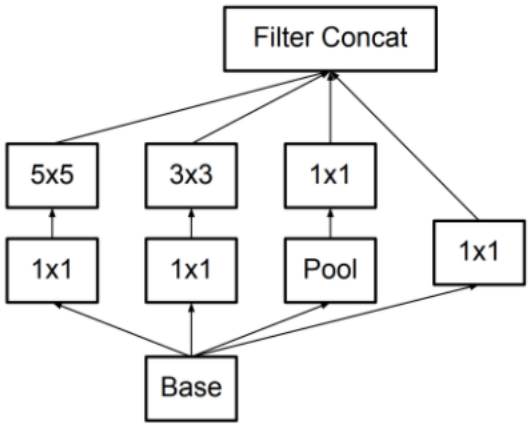


Figure 4.8: Original Inception module as described in [26].

The inception module seems to have below configuration after the first two optimization procedures have been applied.

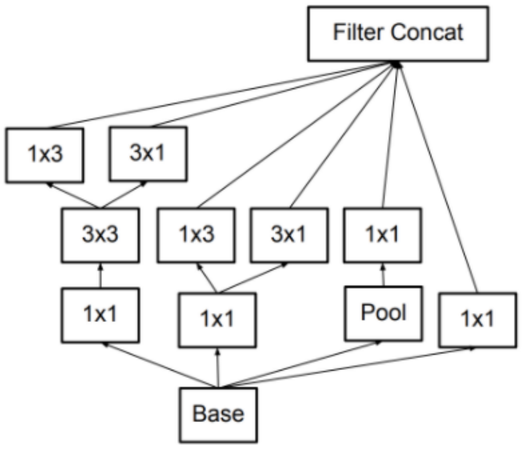


Figure 4.9: The filter bank outputs on the Inception modules have been increased [35].

In the event when the number of input and output filters is the same, the two-layer method is 33% more cost-effective for the same number of output filters.

• Auxiliary classifier: A tiny convolutional neural network (CNN) is an auxiliary classifier that is introduced between training layers. The loss that this causes is added to the loss of the main network. Auxiliary classifiers were used for the purpose of creating a deeper network in GoogLeNet; however, in Inception v3, an auxiliary classifier serves the purpose of a regularizer.

To aid in the training of the primary classifier, an auxiliary classifier was introduced to the design of Inception V3. Added on top of the primary classifier, the auxiliary classifier is a compact fully connected network that has been trained to forecast the class labels of the input pictures.

The output of the main classifier, which is a few levels before the final fully connected layer, is coupled to the auxiliary classifier during training. As a result, the primary classifier's characteristics that were retrieved earlier in the network's development may be sent to the auxiliary classifier. By giving the primary classifier an extra source of supervision, this may aid in improving its training.

Additionally, the primary classifier's weights and biases are not shared with the auxiliary classifier. The primary classifier and the auxiliary classifier are trained together, and the weights of both classifiers are updated at the same time.

Only during training is the auxiliary classifier employed; during inference, it is not used at all. The primary classifier's training is improved by the auxiliary classifier's added supervision and regularization, which helps the main classifier generalize more effectively to new data.

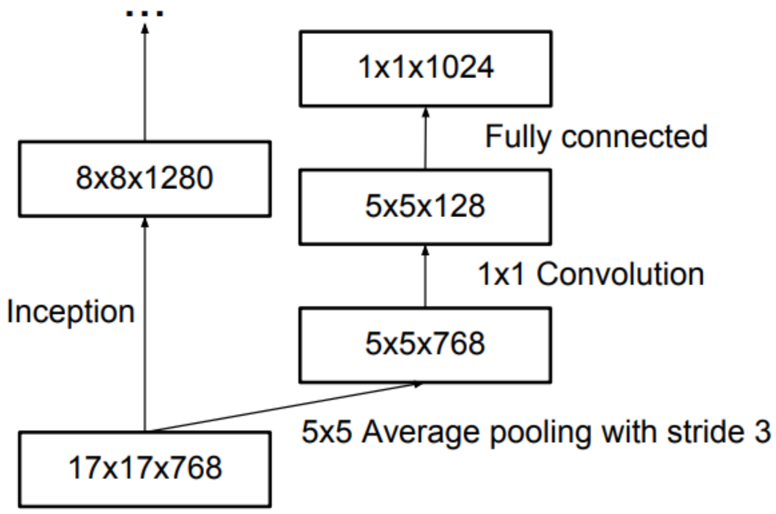


Figure 4.10: Auxiliary classifier helper above the final 17x17 layer. When the layers in the side head are Batch Normalized [23] the outcome is a 0.4% absolute improvement in top-1 accuracy. The number of 32 batches processed is shown against time below [35].

• Grid size reduction: Pooling procedures are often used to reduce the grid size. The resolution of the input picture is decreased by using grid size reduction, yet crucial details are preserved. Using max pooling layers in the network's root and Inception modules helps reduce the size of the grid.

To do this, the input picture is down - sampled by a factor of 2 in both the width and the height in the network's stem using a max pooling layer with a kernel size of 3x3 and a stride of 2. The size of the grid is virtually halved in both directions as a result of this. Some of the branches in the Inception architecture employ a max pooling layer with a kernel size of 3x3 and a stride of 1 to scale down the input resolution by a factor of 2. This may happen in either the width or the height dimension.

By reducing the resolution of the input image, the network is able to focus on the important features while reducing the computational cost and number of parameters. This is particularly useful in the early stages of the network where the spatial resolution is high, but the features are not yet well-formed, and the computational cost could be high.

Grid size reduction is also called spatial downsampling, and it is an important technique in CNN architectures, where the resolution of the input is often reduced as the features are extracted, allowing the network to focus on more abstract features while reducing the computational cost and the number of parameters. This technique can also help to reduce overfitting by reducing the number of parameters.

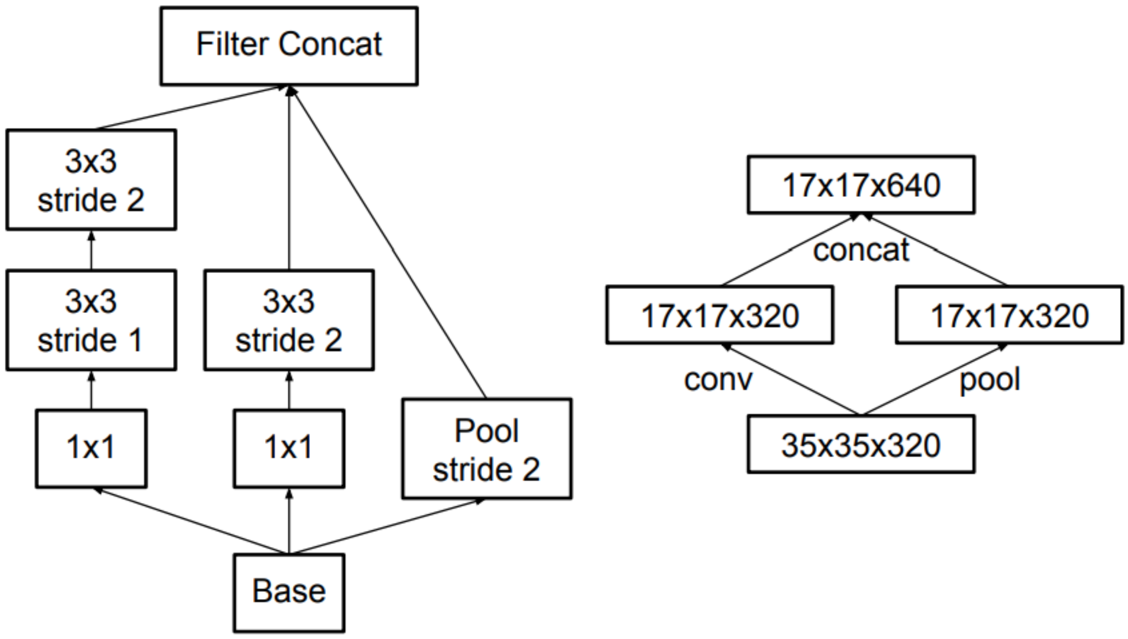


Figure 4.11: Modulator of inception that expands filter banks while decreasing grid size. It's inexpensive and gets around the representational bottleneck, just as Principle 1 [35] suggests. The solution is shown on the right, although this time it is seen through the lens of grid sizes rather than operations [35].

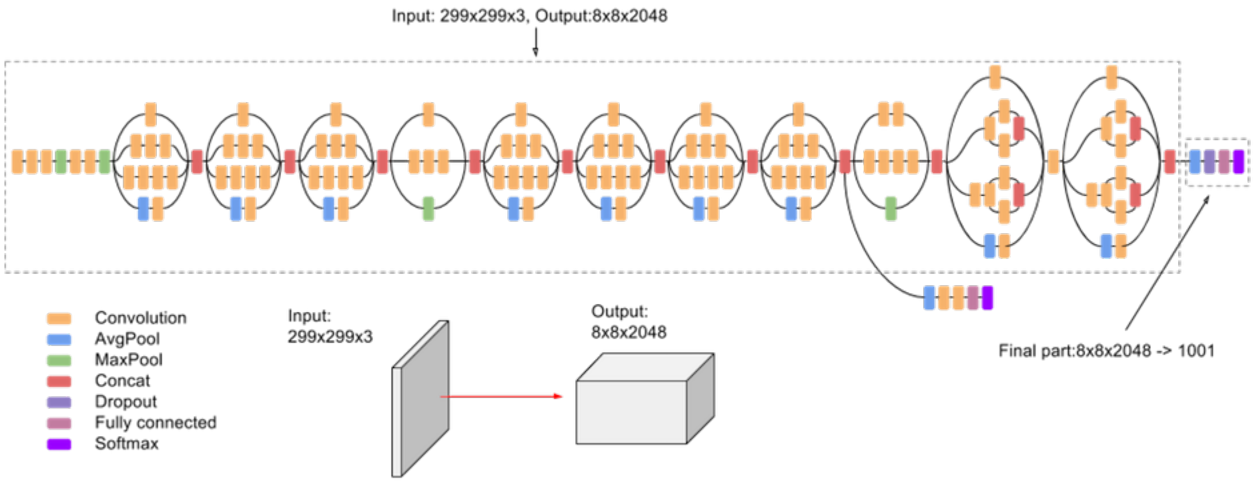
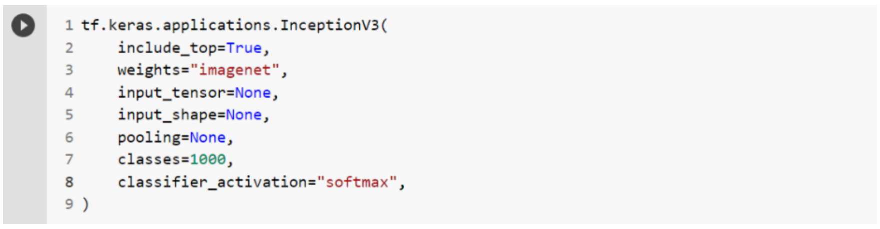


Figure 4.12: Inception v3 architecture



```
tf.keras.applications.InceptionV3(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
)
```

Figure 4.13: Inception v3 sample code.

Sample Code

4.1.3 RESNET 50

There are different kinds of convolutional neural networks and Resnet is one of them. It stands for Residual Network. This ground-breaking neural network was first described in a study by Kaiming, Xiangyu, Shaoqing, and Jian titled "Deep Residual Learning for Image Recognition" [31]. In this paper, Kaiming and et al. have shown that Resnet creates a fundamental breakthrough in neural networks as using this extremely deep neural networks model can be trained having over 150 layers. Also, This new neural architecture has helped to overcome the biggest disadvantage of convolutional neural networks, which is the vanishing Gradient Problem [25].

The base model ResNet architecture was introduced with 34 weighted layers, and it provided some easy ways to add more layers to convolutional layers to a CNN [21]. However, the Resnet50 has 50 layers of convolutional neural network, and A sort of artificial neural network known as a residual neural network constructs networks by stacking blocks of residual information.

Now, before going into how ResNet50 architecture works, we need to develop an idea of how this architecture has made a great breakthrough in the field of image Recognition. [31] LeCun et al., [2] Krizhevsky and et al. have mentioned in their paper how CNN has made a series of breakthroughs. However, recent research has shown that adding more layers can enhance the features of the image (depth). [18]

K. Simonyan and A. Zisserman and then [27] Szeged and W. Li have mentioned in their paper about the importance of network and depth in how the result changes significantly. A graph(figure 4.14) for loss error count with respect to irritations of the layers can give us an idea of how the result changes just by stacking the new layers.

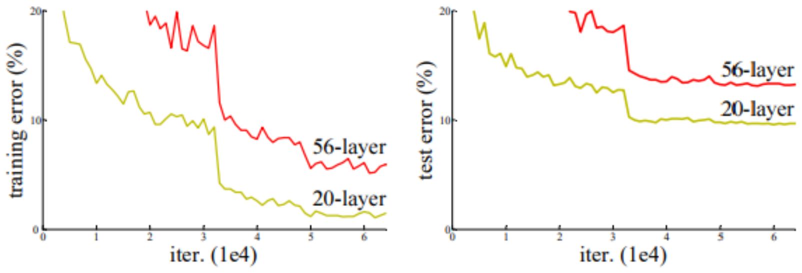
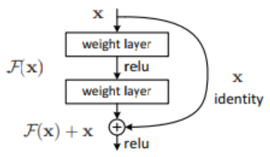


Figure 4.14: Test error (right) and training error (left) with 20-layer and 56-layer "plain" networks.

Considering figure1 we can say that the deeper the layer the higher the training error, but previously we have stated that the features of an image can be extracted more precisely by adding more layers or depth. That is why a great question arises, "Is learning better networks as easy as stacking more layers?". This is the problem of vanishing gradients that have been mentioned by [3] Bengio, P. Simard, and P. Frasconi, and later [12] Glorot and Y. Bengio in their study they have mentioned the dependencies with gradient descent while learning.

When deeper networks are able to start converging, A degradation problem gets exposed. If the accuracy saturates (perhaps unsurprisingly) and decreases rapidly as the depth of the network increases, then these decreases are not due to overfitting [30] [59], and adding more layers to the model increases training error. According to our tests, Figure 1 shows a typical example of this. Deep residual learning framework has introduced to tackle the degradation problem and this solution has mentioned in the paper submitted by [31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.

So, in this paper, the author tried to represent H(x) as desired mapping, and the mapping equation F(x)= H(x)x is used to stack nonlinear layers by mapping. Then F(x)+x is transformed to original mapping.



x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 4.15: Residual learning: a building block

This figure shows the formation of F(x)+x then again by adding more feedforward neural networks with shortcut connections the formation of F(x)+x can be achieved. This technique has already been mentioned in the recent work by [4] Bishop, [10] B.D. Ripley, [8] N.N. screwdorf. By shortcut connections, we meant skipping one or more layers. In another case, the shortcut connection performs identity mapping and one output getting from previous layers are being added to the output coming from the next layers. These shortcut connections do not add any extra parameter neither these connections add more computational complexity and just by using common libraries this can be implemented, and it does not even need to modify any solvers too.

Deep Residual Learning with shortcut identity layers

Residual learning can be applied by several stack layers and building blocks, as shown in Figure 2. [31] Kaiming, Xiangyu, Shaoqing, and Jian have shown in their paper that the block can be defined as y = F(x, Wi) + x.

Here, the input and output vectors of the layer are represented as x and y respectively for feature extraction. F(x, Wi) represents the residual map to learn. If you look at the explanation of Figure 2, there are two layers, so F = W2(W1x). And this formation was [48] V. Nair and G. E. Hinton states in his paper that stands for ReLU and that the operation F + x is performed by shortcut connections and element wise addition. And let t be her second nonlinearity after addition (that is, (y)). So you can see that the shortcut link in the above equation does not add any additional parameters or computational complexity. Comparing the simple convolutional networks and residual networks, we can see that the residual networks behave similarly to the simple convolutional neural networks with the same number of computational cost, depth and parameters. However, in this case, the dimensions of x and F must again be the same. If the dimensions are not the same, you can perform a linear projection to make them match. So we can get more information about the input image.

Here in this section of our paper, we are going to show a difference between Plain convolutional neural networks and deep residual networks, both having 34 layers. Based on the figure 4.16, we can insert shortcut connections of residual networks. Again, when output and input have the same dimensions, identity shortcuts can directly be used.

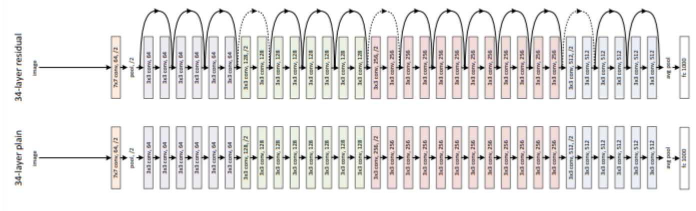
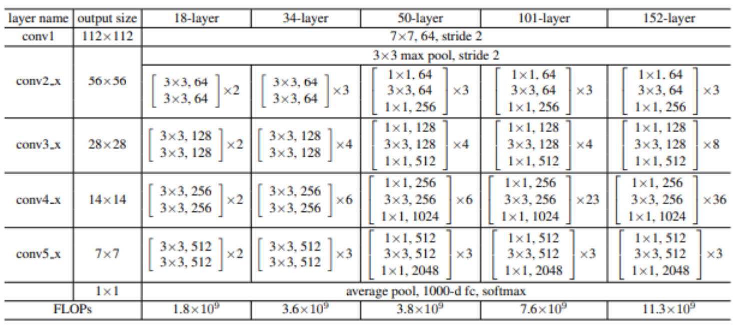


Figure 4.16: Representation of the layers of plain and residual network

Implementation of Resnet50

In the figure 4.17, the residual and plain network layers have 34 layers. First, the image is scaled down to 224x224 and the default color expansion is used here. Weights are initialized immediately after each convolutional layer is activated. Here, a stack size of SGD of 2566 is used as a minimum desired stack size and the learning rate starts at 0.1 and is divided by 10 when an error plateau is reached. The model is then trained with up to $60 \times 10\hat{4}$ iterations.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 4.17: Number of iterations as the layers increase.

Looking at figure above, we can see that the number of iterations increases as the number of layers increases. Now we can plot the error rate over the number of iterations. In contrast to a simple convolutional neural network, just skipping a few layers gives a clear picture of how well the rest of the neural network is actually performing.

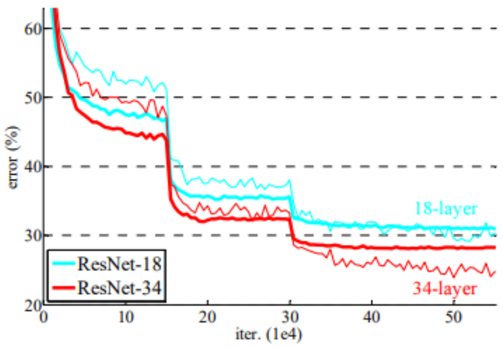


Figure 4.18: Error Rate with respect to number of iterations

As we can see (figure 4.18) that by adding the number of layers, the error rate is getting lesser and lesser. So we can take 50 layers then the error rate will come down more but here the number of iterations got increased for 50 layers, but it is at an optimal level and can be performed on almost every platform where a base model of Convolutional neural networks can be performed.

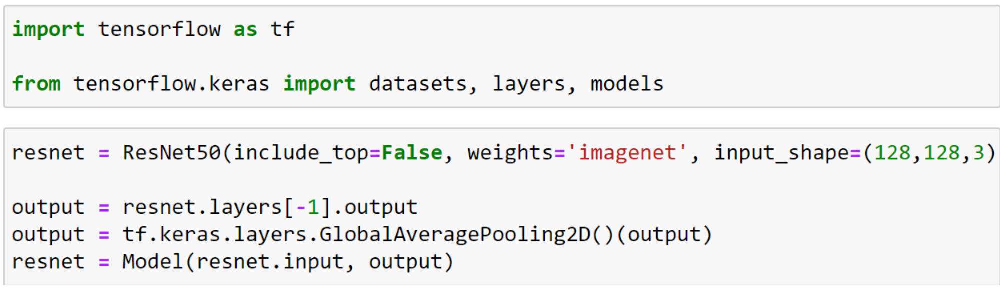
4.2 Implementation

4.2.1 Designing and Implementation of ResNet-50

ResNet, short for Residual Networks, is a neural network used as a strength for many computer vision tasks and can be trained for over 150+ layers. For our study we have also used this ResNet architecture but for our convenience, only the last 50 layers of the Residual block have been set to true for training.

Important libraries:

- Tensorflow.
- Tensorflow.keras and datasets, layers, models.



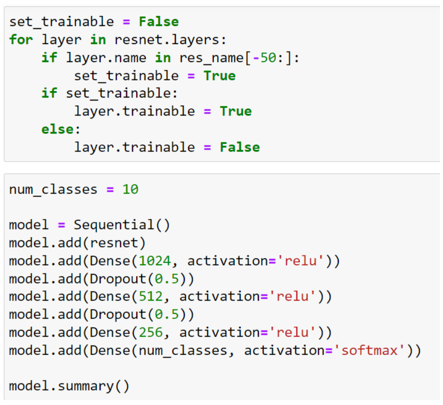
```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
```

```
resnet = ResNet50(include_top=False, weights='imagenet', input_shape=(128,128,3))

output = resnet.layers[-1].output
output = tf.keras.layers.GlobalAveragePooling2D()(output)
resnet = Model(resnet.input, output)
```

Figure 4.19: Pre-settings of ResNet-50 and important libraries

After we have imported the important libraries and now only the last 50 layers set to true. For this experiment, we have set, Epochs=100, batch size = 32 and number of classes = 10.

However, in this experiment, we created a sequential model hence we have added three Dense layers consecutively 1024, 512, and 256 with a dropout amount of 0.5 in between the dense layers. And then after the last dense layer, we have set the activation method of “Softmax” since we are dealing with multiple classes.

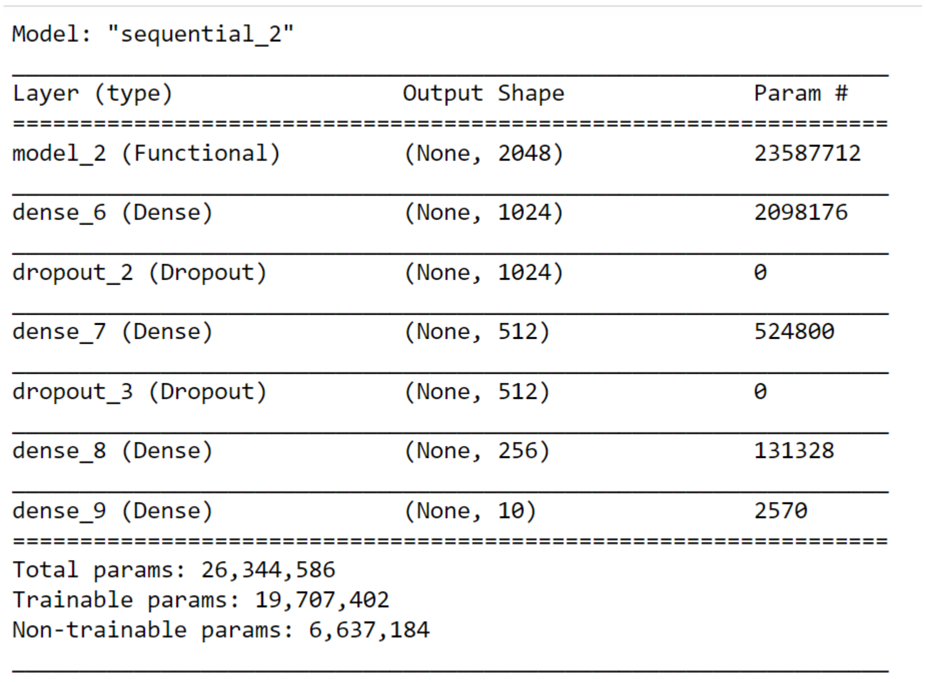


```
set_trainable = False
for layer in resnet.layers:
    if layer.name in res_name[-50:]:
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False
```

```
num_classes = 10

model = Sequential()
model.add(resnet)
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.summary()
```

Figure 4.20: ResNet-50(fine tuned) model with activation “softmax”.

Considering this figure, the total Trainable parameters are 19,707,402 which provides the model a good amount of data to be trained over the dataset. Lastly, we have



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
model_2 (Functional)         (None, 2048)              23587712
_________________________________________________________________
dense_6 (Dense)              (None, 1024)              2098176
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_7 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_8 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_9 (Dense)              (None, 10)                2570
=================================================================
Total params: 26,344,586
Trainable params: 19,707,402
Non-trainable params: 6,637,184
_________________________________________________________________
```

Figure 4.21: ResNet-50 (fine tuned) model summary.

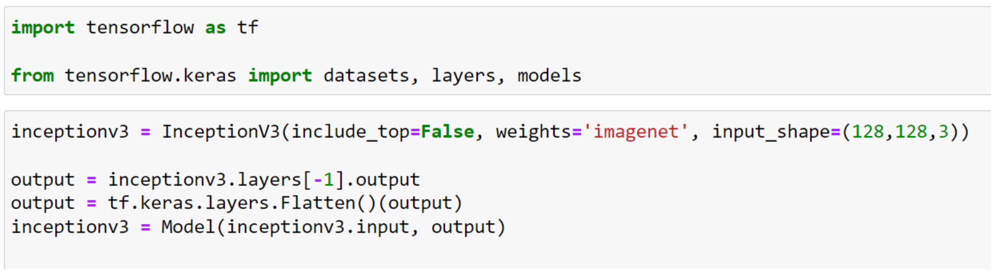
trained our model in our local machine with CPU configuration - Ryzen 9 5950x with 64gb of ram and dedicated gpu GTX Nvidia 3080ti with 12gb of memory.

4.2.2 Designing and Implementation of Inception v3

In our study we have also used this Inception architecture but for our convenience, we set all the parameters just like we have used in ResNet50. This architecture also have the same amount of layers that have been trained like we have described previously while implementing ResNet50.Though we have used Flatten() layers for output unlike Resnet-50 architecture we used GlobalAvergaePooling2D. All other settings are the same here for this fine tuned model.

Important libraries:

- Tensorflow.
- Tensorflow.keras and datasets, layers, models.

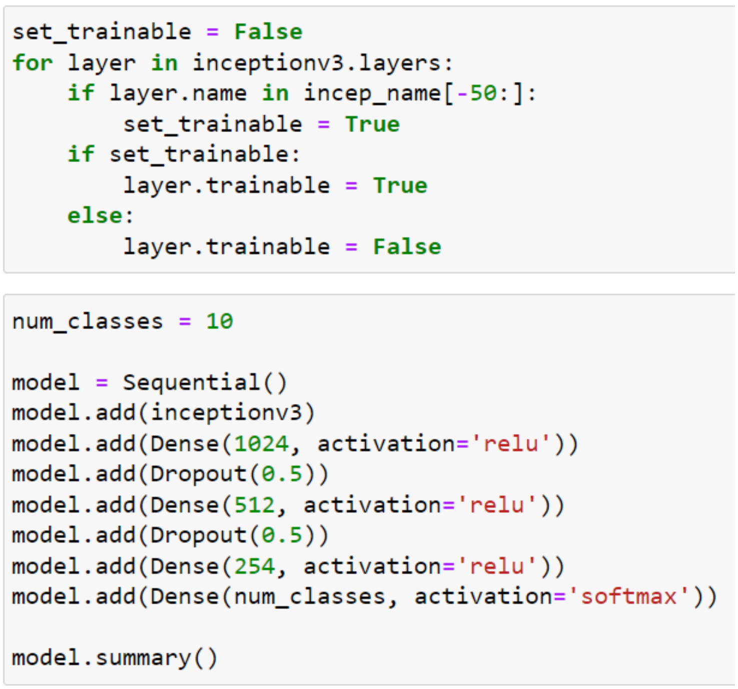


```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
```

```
inceptionv3 = InceptionV3(include_top=False, weights='imagenet', input_shape=(128,128,3))

output = inceptionv3.layers[-1].output
output = tf.keras.layers.Flatten()(output)
inceptionv3 = Model(inceptionv3.input, output)
```

Figure 4.22: Pre-settings of inception v3 and important libraries.

After we have imported the important libraries and now only the last 50 layers will be set to true. For this experiment in our study, we have set Epochs=100, batch size = 32 and Number of classes = 10. However, in this experiment, we created a sequential model hence we have added three Dense layers of 1024,512, and 256 with a dropout amount of 0.5. Here the activation method has been set to softmax since we are dealing with multiple classes.

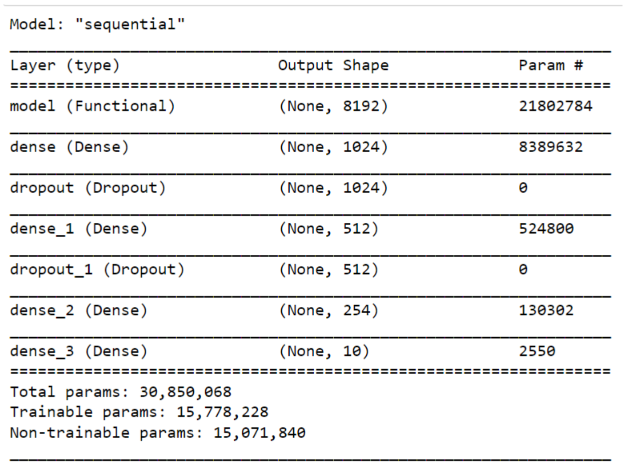


```
set_trainable = False
for layer in inceptionv3.layers:
    if layer.name in incep_name[-50:]:
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False
```

```
num_classes = 10

model = Sequential()
model.add(inceptionv3)
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(254, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.summary()
```

Figure 4.23: Inception v3 (fine tuned) model with activation “softmax”.

Here the activation method has been set to softmax since we are dealing with multiple classes.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
model (Functional)           (None, 8192)              21802784
_________________________________________________________________
dense (Dense)                (None, 1024)              8389632
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 254)               130302
_________________________________________________________________
dense_3 (Dense)              (None, 10)                2550
=================================================================
Total params: 30,850,068
Trainable params: 15,778,228
Non-trainable params: 15,071,840
_________________________________________________________________
```

Figure 4.24: Inception V3 (fine tuned) model summary.

Considering this figure, the total Trainable parameters are 15,778,228 which provides the model a good amount of parameters to get the features extracted from our dataset to be trained. Lastly, we have trained our model in our local machine with

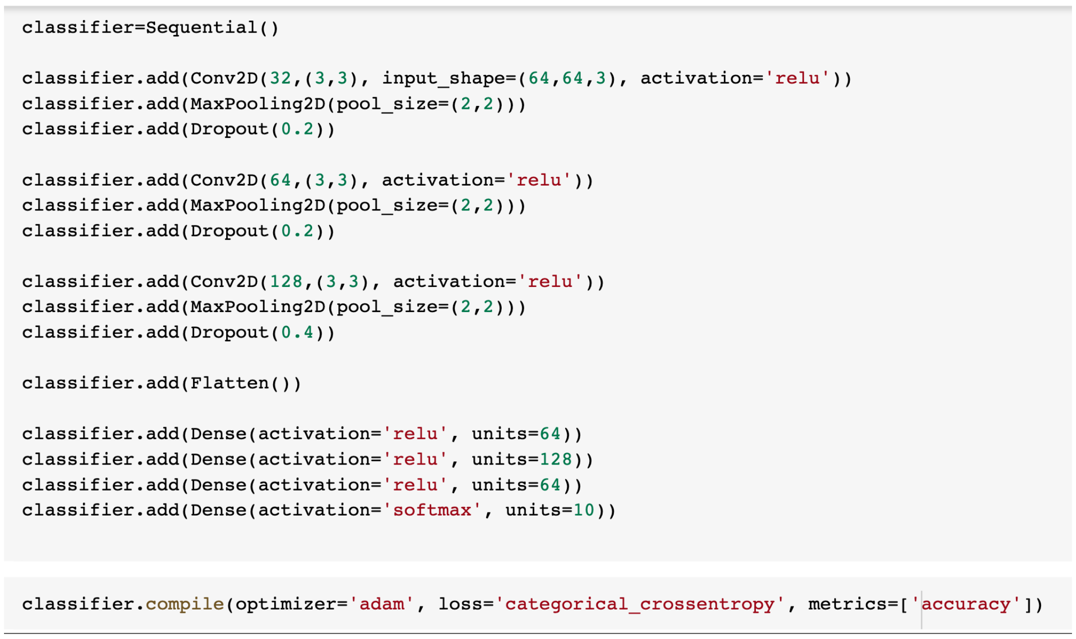
CPU configuration - Ryzen 9 5950x with 64gb of ram and dedicated GPU GTX Nvidia 3080ti with 12gb of memory.

4.2.3 Designing and Implementation of CNN

In our experiment, we have also used Convolulation Neural Network architecture model. Here have. The following import libraries have been imported to use the desired dense layer and dropout level and flatten any layer.

Important libraries:

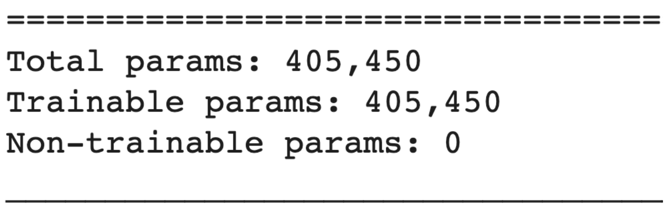
- keras.utils to_categorical.
- Tensorflow.keras.models Sequential.
- tensorflow.keras.layers Conv2D,Dense,Flatten,MaxPooling2D,Dropout.



```
classifier=Sequential()

classifier.add(Conv2D(32,(3,3), input_shape=(64,64,3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2,2)))
classifier.add(Dropout(0.2))

classifier.add(Conv2D(64,(3,3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2,2)))
classifier.add(Dropout(0.2))

classifier.add(Conv2D(128,(3,3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2,2)))
classifier.add(Dropout(0.4))

classifier.add(Flatten())

classifier.add(Dense(activation='relu', units=64))
classifier.add(Dense(activation='relu', units=128))
classifier.add(Dense(activation='relu', units=64))
classifier.add(Dense(activation='softmax', units=10))
```

```
classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Figure 4.25: CNN (proposed model) with activation “softmax”.



```
==================================
Total params: 405,450
Trainable params: 405,450
Non-trainable params: 0
__________________________________
```

Figure 4.26: Trainable Parameters for CNN(proposed model).

Considering this figure 4.26, the total trainable parameters are 405,450 which provides the model a good amount of parameters to get the features extracted from our dataset to be trained. And also trainable parameter is much more less than previous fine tuned pre-trained models.

Chapter 5

Result and analysis

5.1 Analysis of predicted results

The performance of various models in our experiments depending on our dataset is described in this section. To represent how precisely the three different methods of our proposed models can actually classify the diseases of the tomato leaves, there are some performance metrics we need to mention first.

5.1.1 Performance matrics

To represent how precisely the three different methods of our proposed models can actually classify the decreases of the tomato leaves, there are some performance metrics we need to mention first. In our study, we have constructed a confusion matrix consisting of precision, recall, and F_1 scores. These terms are - i) True positive classes(TP), ii) True negative classes (TN), iii) False positive classes (FP), iv) False Negative classes(FN). the number of positive class predictions that actually belong to the positive classes are denoted by Precision.

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

Since we have 10 different classes to classy therefore the term TP, FP would come 10 times as TP1,FP1,TP2,FP2 and so on for the rest of the classes.

Then the number of positive class predictions made out of all positive examples in the dataset is denoted as recall. Also, the ratio between the number of Positive samples correctly classified as Positive to the total number of Positive samples is denoted as Recall score.This parameter denoted the ability of the model to predict the positive samples only.Unlike the precision metric, this metric only cross-outs the negatively predicted classes and then only considers the true positive (TP) predicted classes among all the classes that have been predicted by the model. How a model performs generally across all the different classes is acutely measured by accuracy. This parameter is also important in order to elaborate on how the model is actually working. This is mainly the ratio of the number of correct predictions to the number of total predictions. This can be calculated in the equation shown below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2}$$

Lastly, the metric that we have considered in our study to show how the three different models are working if it knows as F1_Score. F1 score, is a single score that balances both the concerns of precision and recall in one number.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{5.3}$$

5.1.2 Analysis of predicted results

In this section of our study, all results using the metrics that we have discussed above will be elaborated precisely and individually for each of the models used in this study.

5.1.3 Results and Analysis From CNN

	precision	recall	f1-score	support
BACTERIAL SPOT	0.94	0.99	0.96	426
EARLY BLIGHT	0.95	0.91	0.93	480
HEALTHY	0.95	0.99	0.97	482
LATE BLIGHT	0.95	0.93	0.94	463
LEAF MOLD	0.95	0.98	0.96	471
SEPTORIA LEAF SPOT	0.96	0.91	0.94	437
SPIDER MITE	0.96	0.92	0.94	436
TARGET SPOT	0.92	0.92	0.92	457
MOSAIC VIRUS	0.97	1.00	0.98	448
YELLOW LEAF CURL VIRUS	1.00	0.98	0.99	491
micro avg	0.96	0.95	0.95	4591
macro avg	0.96	0.95	0.95	4591
weighted avg	0.96	0.95	0.95	4591
sample avg	0.95	0.95	0.95	4591

Table 5.1: Classification report of CNN(proposed model)

Among the models trained using the tomato leaf images consisting of ten classes, the experiment done using basic CNN gives a pretty good result having the F1_score of 95% which is shown in above table 5.1. Considering the above report, it is seen that all the ten classes of leaf diseases have precision score, not below 92%.

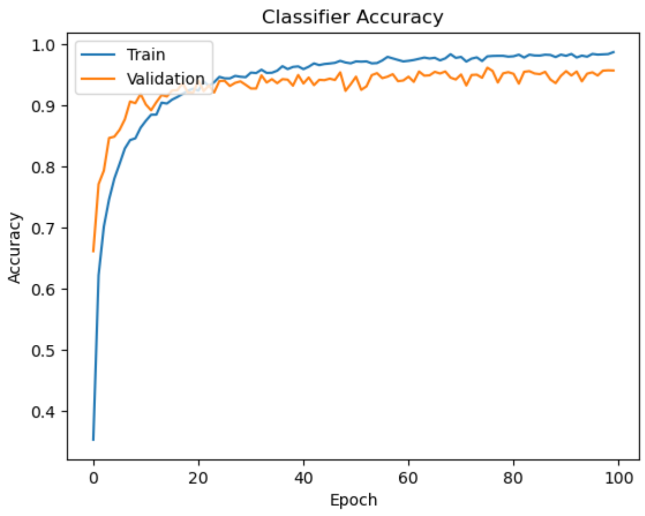


Figure 5.1: Accuracy score with respect to the number of epochs.

In the above figure 5.1 the number or epoch increases the accuracy also increases for both training and validation up to 98% for trained data and not below than 96% for validation data.

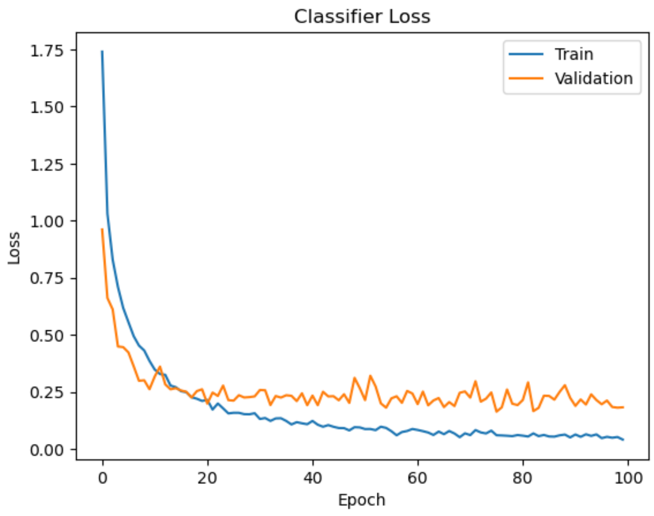


Figure 5.2: Classifier loss with respect to number of Epochs

In this experiment the classification loss of the information extracted from the dataset remains high at the initial stage but as the number of epochs increases the loss decreases in gradual way for both training data and validation data and there not many ridges to be seen here in the graph figure 5.2 shown above . Lastly to elaborate and to come to a definite decision if the model is working fine the dataset and predicting the actually classes of the images a confusion matrix is a must thing to create.

In the figure 5.3, a confusion matrix has been generated using our trained CNN model with the help of test set data to predict the classes. And it is proven by the above figure that for every class the model is giving an optimal result.

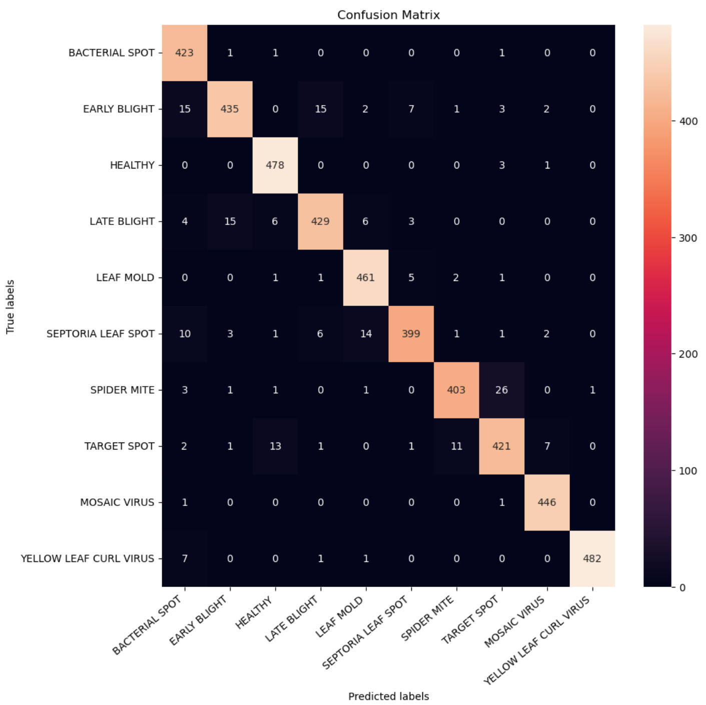


Figure 5.3: Confusion matrix for the ten classes.

5.1.4 Results and Analysis From Inception V3

The table 5.2 gives an overview of the accuracy of the precision and the F1_score. Here the precision score gives a maximum of 96% for a single class but not less than 75% for any of the ten classes. Again The overall accuracy is 87% which is a bit lower than the CNN.

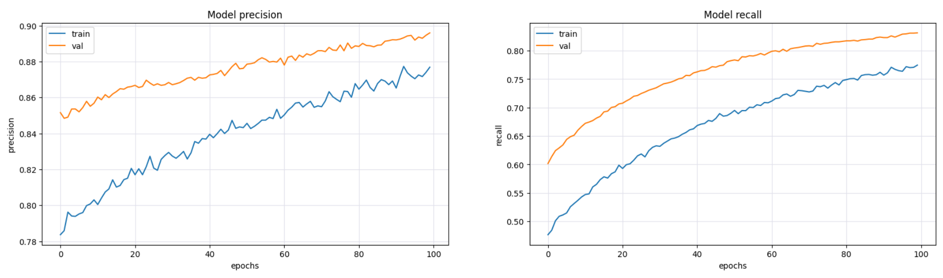


Figure 5.4: Precision and Recall score for Inception v3

In the above figure 5.4, it shows hot the precision and accuracy and recall metrics are changing in a good manner as the number of epoch is increasing. And the loss of features of the data is also decreasing as the number of epochs increases. However, a confusion matrix can state a how perfectly the model is predicting the classes from the validation data set using the test dataset.

Since the accuracy in this experiment is not much higher so it is clearly seen that not all the classes are getting predicted correctly as class 5 is getting the highest positive predicting score.

	precision	recall	f1-score	support
BACTERIAL SPOT	0.8818	0.9065	0.8940	214
EARLY BLIGHT	0.8670	0.7875	0.8253	240
HEALTHY	0.8571	0.8793	0.8681	232
LATE BLIGHT	0.9075	0.8729	0.8898	236
LEAF MOLD	0.7542	0.8265	0.7887	219
SEPTORIA LEAF SPOT	0.8578	0.8813	0.8694	219
SPIDER MITE	0.8230	0.7511	0.7854	229
TARGET SPOT	0.9672	0.9593	0.9633	246
MOSAIC VIRUS	0.9321	0.9156	0.9238	225
YELLOW LEAF CURL VIRUS	0.900	0.9669	0.9323	242
accuracy			0.8753	2302
macro avg	0.8748	0.8747	0.8740	2302
weighted avg	0.8761	0.8753	0.8750	2302

Table 5.2: Classification report of inception v3

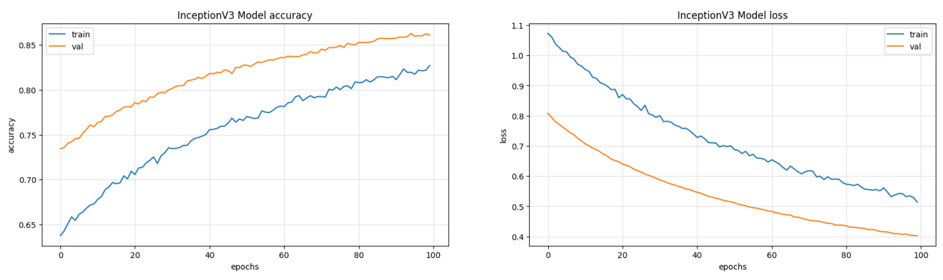


Figure 5.5: Train and Validation accuracy curve with respect to the number of epochs

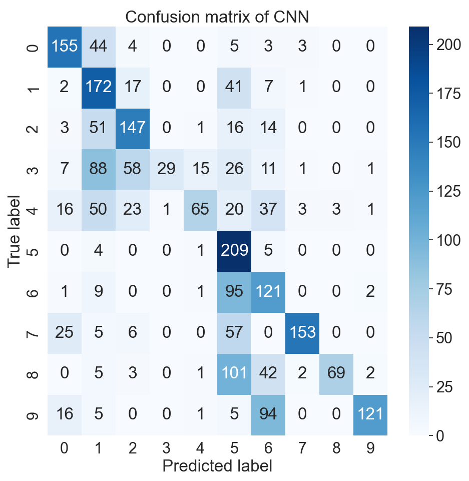


Figure 5.6: Confusion matrix for Inception V3

Here, every class has its own legend color which represents in the graph and also gives an overview of our trained model in this experiment.

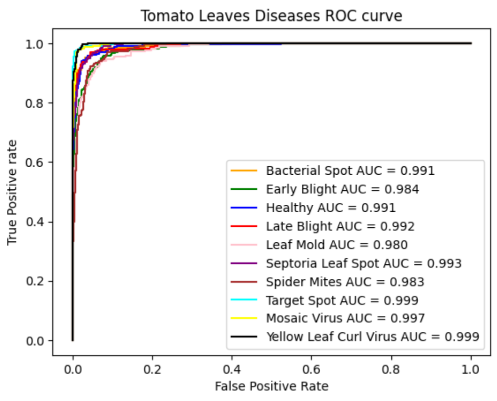


Figure 5.7: ROC curve for the classes in Inception V3

	precision	recall	f1-score	support
BACTERIAL SPOT	0.6796	0.9813	0.8031	214
EARLY BLIGHT	0.7381	0.6458	0.6889	240
HEALTHY	0.7664	0.7069	0.7354	232
LATE BLIGHT	0.9441	0.5720	0.7124	236
LEAF MOLD	0.7150	0.7658	0.6948	219
SEPTORIA LEAF SPOT	0.6181	0.8721	0.7235	219
SPIDER MITE	0.6220	0.6681	0.6442	229
TARGET SPOT	0.9691	0.7642	0.8545	246
MOSAIC VIRUS	0.9623	0.6800	0.7969	225
YELLOW LEAF CURL VIRUS	0.7621	0.9793	0.8571	242
accuracy			0.7533	2302
macro avg	0.7777	0.7546	0.7511	2302
weighted avg	0.7806	0.7533	0.7520	2302

Table 5.3: Classification report for ResNet-50

5.1.5 Results and Analysis From ResNet-50

The above table 5.3 gives an overview of the accuracy of the precision and the F1_score. Here, the precision score gives a maximum of 96% for a 2 classes, but not less than 71% for any of the nine classes. Again The overall accuracy is 75.33% which is the lowest among all the two other experiments we have shown above.

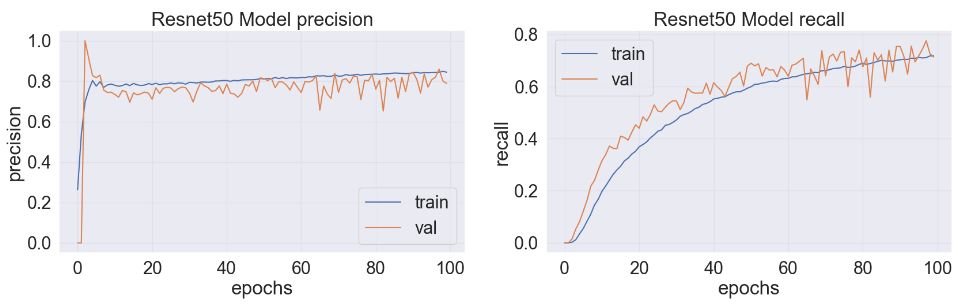


Figure 5.8: Precision and Recall curve for ResNet50(fine tuned).

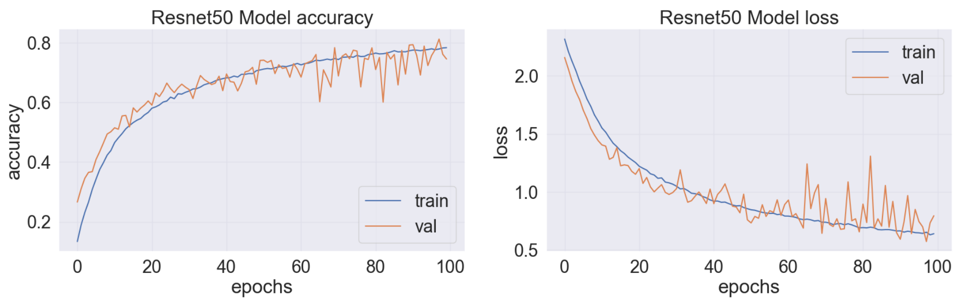


Figure 5.9: Accuracy and Loss curve of ResNet50(fine tuned).

Here in our Experiment even after training our model for 100 epochs we were unable to get good accuracy on both the train and validation dataset. Therefore, for this reason in the above figures, the line of the graphs have more ridges than we have expected.

5.2 Final analysis report among the Architectures

Since have discussed all the experiments that we are gone through for our study we can state that the maximum accuracy for our image classification we have got

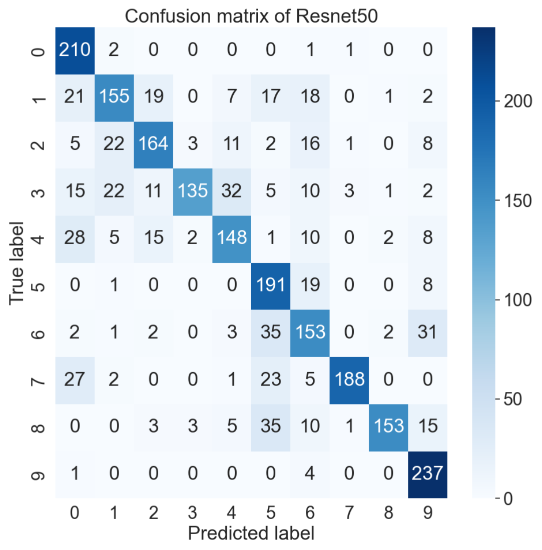


Figure 5.10: Confusion matrix for ResNet50 (fine tuned).

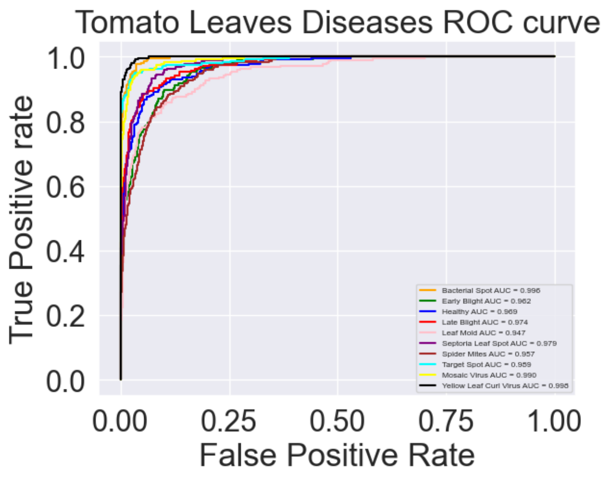


Figure 5.11: ROC curve for ResNet50 (fine tuned).

from CNN is 95%. However, the lowest accuracy was from the ResNet50 model. Also considering the figures above we also state that the lowest validation loss we have achieved was from CNN among the rest of the architecture (Inception V3 and ResNet50).

Chapter 6

Conclusion and Future Work

A greater part of Bangladeshi population depends on the agricultural sector, for which this is still one of the most important sectors of the country. Even the growth of the economy relies on various agricultural sectors. Therefore, to maintain the growth of our economy, we need to adapt advanced technologies with the help of computer science to prevent or to cure problems that arise to our farmers. And it is generally seen that even though the farmers are mostly dependent on pesticides and chemical fertilizers though this helps to prevent most of the diseases of the crops but for most of the cases the farmers are not able to detect the exact diseases by examining the leaves of a certain crop which hampers the production rate of the of crops. And can become a threat to our nation's economic growth. So in our paper we are trying to develop a system through various image processing techniques, using which we can help the farmers to detect crop's disease through their own device by providing a raw image of the selected crop's leaf. Tomato is known as a staple food as it is eaten often by the larger portion of people. Hence, in this paper we have evaluated three models known as Convolutional Neural Networks(CNN), Resnet-50, Inception-v3 for classifying 10 types of tomato leaf diseases. We implemented multiple data transformation techniques and augmentation processes. Training data is visualized for better understanding the data and performance of the models. The disease identification is evaluated by training loss, valid loss, confusion matrix, and accuracy data output from models. Although the currently existing direct and indirect models are already accessible and ready to use broadly for the detection of plant leaf diseases, those models are not much encouraged for the practical application because of having a couple of limitations. That is why we have brought different models from the available models for better accuracy rate and better effectiveness. Which has proved to obtain an accuracy rate of 75.33% in the Resnet-50 and 87.55% in Inception-v3 and 95.5% in CNN. Yet, the main drawback for our proposed methodology is that the process is a bit time-consuming and high-end hardware configuration is compulsory for model training. To solve this problem, we customized a CNN model, transformed the train images into 64 x 64 x 3, augmented the images such that the images were in focus in the images. After customizing, our models became faster and light weight which is easy to deploy.

For further work, we will be trying to extend our process for several new algorithms so that we can come up with the most favorable results compared to existing techniques. Furthermore, since we have already discussed that we are only working on the diseases that are seen on the leaf but in real a life scenario the diseases are not seen only the leaf, so we will be trying to cover more parts of the plants so that we can include more detailed dataset and classify the diseases. Again our study has been on a processed data which was collected, but we are planning to deploy our model into real life scenario where our system will be trained over mostly all the diseases of the tomato plant so that the end user can upload image as an input and the system build using our methodologies will detect instantly the diseases.Therefore, our mentioned model can be operable as a promising tool to assist and support the farmers in classifying the diseases that can be found in tomato plants.

Bibliography

[1] N. Mackenzie and A. Pinder, "Flow cytometry and its applications in veterinary medicine," *Research in veterinary science*, vol. 42, no. 2, pp. 131–139, 1987.

[2] Y. LeCun, B. Boser, J. S. Denker, *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[3] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[4] C. M. Bishop *et al.*, *Neural networks for pattern recognition.* Oxford university press, 1995.

[5] J. L. Lange, P. S. Thorne, and N. Lynch, "Application of flow cytometry and fluorescent in situ hybridization for assessment of exposures to airborne bacteria," *Applied and Environmental Microbiology*, vol. 63, no. 4, pp. 1557–1563, 1997.

[6] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[8] N. N. Schraudolph, "Centering neural network gradient factors," in *Neural Networks: Tricks of the Trade*, Springer, 1998, pp. 207–226.

[9] A. Moter and U. B. Göbel, "Fluorescence in situ hybridization (fish) for direct visualization of microorganisms," *Journal of microbiological methods*, vol. 41, no. 2, pp. 85–112, 2000.

[10] B. D. Ripley, *Pattern recognition and neural networks.* Cambridge university press, 2007.

[11] A. Meunkaewjinda, P. Kumsawat, K. Attakitmongcol, and A. Srikaew, "Grape leaf disease detection from color imagery using hybrid intelligent system," in *2008 5th international conference on electrical engineering/electronics, computer, telecommunications and information technology*, IEEE, vol. 1, 2008, pp. 513–516.

[12] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[13] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE international symposium on circuits and systems*, IEEE, 2010, pp. 253–256.

[14] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[15] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[16] G. Hinton, L. Deng, D. Yu, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[17] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[19] M. Zakaria, M. A. Aziz, M. I. Hossain, and N. M. F. Rahman, "Effects of rainfall and maximum temperature on aman rice production of bangladesh: A case study for last decade," *International Journal of Scientific & Technology Research*, vol. 3, no. 2, pp. 131–137, 2014.

[20] Y. Fang and R. P. Ramasamy, "Current and prospective methods for plant disease detection," *Biosensors*, vol. 5, no. 3, pp. 537–561, 2015.

[21] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5353–5360.

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

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[24] A. Rastogi, R. Arora, and S. Sharma, "Leaf disease detection and grading using computer vision technology & fuzzy logic," in *2015 2nd international conference on signal processing and integrated networks (SPIN)*, IEEE, 2015, pp. 500–505.

[25] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[26] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[27] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[28] J. G. A. Barbedo, "A review on the main challenges in automatic plant disease identification based on visible range images," *Biosystems engineering*, vol. 144, pp. 52–60, 2016.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[30] M. Hasan and B. HU, "Profitability of tomato production in three districts of bangladesh," *International Journal of BioResearch*, vol. 21, no. 6, pp. 1–8, 2016.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[32] G. Hua and H. Jégou, *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II.* Springer, 2016, vol. 9914.

[33] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in plant science*, vol. 7, p. 1419, 2016.

[34] P. B. Padol and A. A. Yadav, "Svm classifier based grape leaf disease detection," in *2016 Conference on advances in signal processing (CASP)*, IEEE, 2016, pp. 175–179.

[35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[36] H. A. Bekhet, A. Matar, and T. Yasmin, "Co2 emissions, energy consumption, economic growth, and financial development in gcc countries: Dynamic simultaneous equation models," *Renewable and sustainable energy reviews*, vol. 70, pp. 117–132, 2017.

[37] C. G. Dhaware and K. Wanjale, "A modern approach for plant leaf disease classification which depends on leaf image processing," in *2017 International Conference on Computer Communication and Informatics (ICCCI)*, IEEE, 2017, pp. 1–4.

[38] H. Durmuş, E. O. Güneş, and M. Kırcı, "Disease detection on the leaves of the tomato plants by using deep learning," in *2017 6th International conference on agro-geoinformatics*, IEEE, 2017, pp. 1–5.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[40] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[41] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[42] S. Vetal and R. Khule, "Tomato plant disease detection using image processing," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 6, no. 6, pp. 293–297, 2017.

[43] S. Vetal and R. Khule, "Tomato plant disease detection using image processing," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 6, no. 6, pp. 293–297, 2017.

[44] J. G. A. Barbedo, "Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification," *Computers and electronics in agriculture*, vol. 153, pp. 46–53, 2018.

[45] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and electronics in agriculture*, vol. 145, pp. 311–318, 2018.

[46] L. D. Nguyen, D. Lin, Z. Lin, and J. Cao, "Deep cnns for microscopic image classification by exploiting transfer learning and feature concatenation," in *2018 IEEE international symposium on circuits and systems (ISCAS)*, IEEE, 2018, pp. 1–5.

[47] M. Sardogan, A. Tuncer, and Y. Ozen, "Plant leaf disease detection and classification based on cnn with lvq algorithm," in *2018 3rd international conference on computer science and engineering (UBMK)*, IEEE, 2018, pp. 382–385.

[48] M. Sardogan, A. Tuncer, and Y. Ozen, "Plant leaf disease detection and classification based on cnn with lvq algorithm," in *2018 3rd international conference on computer science and engineering (UBMK)*, IEEE, 2018, pp. 382–385.

[49] P. Tm, A. Pranathi, K. SaiAshritha, N. B. Chittaragi, and S. G. Koolagudi, "Tomato leaf disease detection using convolutional neural networks," in *2018 eleventh international conference on contemporary computing (IC3)*, IEEE, 2018, pp. 1–5.

[50] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[51] L. S. P. Annabel, T. Annapoorani, and P. Deepalakshmi, "Machine learning for plant leaf disease detection and classification–a review," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, IEEE, 2019, pp. 0538–0542.

[52] T. Binoy and K. Lakshmi, "Comparative analysis of vehicle make and model recognition using deep learning techniques," in *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, IEEE, vol. 1, 2019, pp. 1298–1305.

[53] M. H. Kamrul, P. Paul, and M. Rahman, "Machine vision based rice disease recognition by deep learning," in *2019 22nd International Conference on Computer and Information Technology (ICCIT)*, IEEE, 2019, pp. 1–6.

[54] A. Khan, A. Sohail, U. Zahoora, and A. Qureshi, "A survey of the recent architectures of deep convolutional neural networks. arxiv 2019," *arXiv preprint arXiv:1901.06032*, 2019.

[55] A. Kumar and M. Vani, "Image based tomato leaf disease detection," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 2019, pp. 1–6.

[56] M. Agarwal, A. Singh, S. Arjaria, A. Sinha, and S. Gupta, "Toled: Tomato leaf disease detection using convolution neural network," *Procedia Computer Science*, vol. 167, pp. 293–301, 2020.

[57] I. Ahmad, M. Hamid, S. Yousaf, S. T. Shah, and M. O. Ahmad, "Optimizing pretrained convolutional neural networks for tomato leaf disease detection," *Complexity*, vol. 2020, 2020.

[58] E. I. Atli, H. Gurkan, H. O. Kirkizlar, *et al.*, "Pros and cons for fluorescent hybridization, karyotyping and next generation sequencing for diagnosis and follow-up of multiple myeloma," *Balkan Journal of Medical Genetics*, vol. 23, no. 2, pp. 59–64, 2020.

[59] A. Grant, *omato temperature tolerance: Best growing temp for tomatoes. gardening know how*, 2020.

[60] H. Hong, J. Lin, and F. Huang, "Tomato disease detection and classification by deep learning," in *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, IEEE, 2020, pp. 25–29.

[61] M.-L. Huang and Y.-H. Chang, "Dataset of tomato leaves," *Mendeley Data*, vol. 1, 2020.

[62] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial intelligence review*, vol. 53, no. 8, pp. 5455–5516, 2020.

[63] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial intelligence review*, vol. 53, no. 8, pp. 5455–5516, 2020.

[64] V. Maeda-Gutiérrez, C. E. Galvan-Tejada, L. A. Zanella-Calzada, *et al.*, "Comparison of convolutional neural network architectures for classification of tomato plant diseases," *Applied Sciences*, vol. 10, no. 4, p. 1245, 2020.

[65] A. Rao and S. Kulkarni, "A hybrid approach for plant leaf disease detection and classification using digital image processing methods," *The International Journal of Electrical Engineering & Education*, p. 0 020 720 920 953 126, 2020.

[66] M. S. Uddin and J. C. Bansal, *Proceedings of International Joint Conference on Computational Intelligence.* Springer, 2020.

[67] L. Alzubaidi, J. Zhang, A. J. Humaidi, *et al.*, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, no. 1, pp. 1–74, 2021.

[68] P. Bansal, R. Kumar, and S. Kumar, "Disease detection in apple leaves using deep convolutional neural network," *Agriculture*, vol. 11, no. 7, p. 617, 2021.

[69] M. E. Chowdhury, T. Rahman, A. Khandakar, *et al.*, "Automatic and reliable leaf disease detection using deep learning techniques," *AgriEngineering*, vol. 3, no. 2, pp. 294–312, 2021.

[70] M. E. Chowdhury, T. Rahman, A. Khandakar, *et al.*, "Tomato leaf diseases detection using deep learning technique," *Technology in Agriculture*, p. 453, 2021.

[71] R. Sujatha, J. M. Chatterjee, N. Jhanjhi, and S. N. Brohi, "Performance of deep learning vs machine learning in plant leaf disease detection," *Microprocessors and Microsystems*, vol. 80, p. 103 615, 2021.

[72] M. J. Alam, L. Hand, and E. Ballard, "Communication disability in bangladesh: Issues and solutions," *Speech, Language and Hearing*, pp. 1–12, 2022.

[73] T. C. Fahim and B. B. Sikder, "Exploring farmers' perception of climate-induced events and adaptation practices to protect crop production and livestock farming in the haor area of north-eastern bangladesh," *Theoretical and Applied Climatology*, vol. 148, no. 1, pp. 441–454, 2022.

[74] Z. Tao, C. XiaoYu, L. HuiLing, Y. XinYu, L. YunCan, and Z. XiaoMin, "Pooling operations in deep learning: From "invariable" to "variable"," *BioMed Research International*, vol. 2022, 2022.

[75] Z. Tao, C. XiaoYu, L. HuiLing, Y. XinYu, L. YunCan, and Z. XiaoMin, "Pooling operations in deep learning: From "invariable" to "variable"," *BioMed Research International*, vol. 2022, 2022.