**CSE / EEE / ETE 499B (Section 02)**

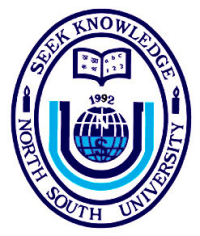
**Research Report (CO7)**

**Project Title: Comparative Analysis of Different CNN Architectures on Potato Leaf Disease Detection and Classification using Transfer Learning Approach**

**Submitted To**

**Dr. Shazzad Hosain (SZZ)**

**Date: 13/06/2023**



**Group-07**

**Members**

|  |  |
| --- | --- |
| **ID** | **Name** |
| 1911350642 | **MD. Fatin Habib Nihal** |
| 1921855042 | **Raihan Mahmud Tahir** |
| 1922013642 | **Sabiha Akter Shorna** |

***Abstract*— *This research conducts an in-depth analysis of some of the most widely used and important Convolutional Neural Network models in today's fast-paced world, where new techniques and models are being created daily to improve prediction accuracy and computational performance of the neural network comprehensively. The nine most common convolutional neural networks (CNNs), such as DenseNet121, VGG16, ResNet50, VGG19, EfficientNetB0, Xception, NASNetMobile, InceptionV3, and MobileNet for potato leaf disease detection and identification are examined in depth in this research along with a variety of data. We looked at and examined the nine CNN models' performance in terms of prediction accuracy, f1 score, precision, recall, and roc score. According to the findings, using transfer learning approach provides us the highest prediction measurement in Xception and MobileNet architectures, with NASNetMobile having the lowest.***

***Keywords— CNN, Plant leaf disease detection, image processing, f1 score, precision, recall, transfer learning***

# **Introduction**

Plant diseases have a devastating impact on crop yield and quality, leading to substantial economic losses and endangering global food production. For effective disease control and the prompt deployment of effective therapies, early detection and correct identification of plant diseases are essential. Neural networks have become effective tools for automated image-based disease classification due to the rapid improvements in deep learning. The comparison of several neural network architectures used with the PlantVillage dataset is the main objective of this study.

The PlantVillage dataset is made up of a sizable number of high-resolution photos showing various plant species, a wide variety of illnesses, and various environmental conditions. It is a priceless tool for developing and testing machine learning models created especially for identifying plant diseases.

To conduct a thorough comparative analysis, we consider a range of widely adopted neural network architectures, including traditional convolutional neural networks (CNNs) such as VGG19, Xception, NASNetMobile, InceptionV3, ResNet50, MobileNetas well as more recent advancements like DenseNet121 and EfficientNetB0.

We assess this neural network design's performance using a variety of criteria, such as classification accuracy, precision, recall, F1-score, macro average, and weighted average.

# **Literature Review**

The study R. Meena Prakash et al. implemented in [1] can assess and categorize citrus leaf diseases. The entire system is composed of four steps: feature extraction, image preprocessing, segmentation using k-means clustering, and classification.

A detection method for grape leaf disease based on texture analysis and pattern recognition was created by Harshal Waghmare et al. in [2]. Support vector machines are used to classify the data. There are two main diseases that affect grape plants: downy mildew and black rot. The accuracy of the suggested method was 96.6%.

Using image processing and a neural network, Sandika Biswas et al. [3] devised a system to assess the severity of the potato late blight disease on potato leaves. Fuzzy C-means clustering divides the disease-affected area and adds background to the photos that have similar color characteristics.

Back Propagation Neural Network (BPNN) was used as a classifier by Libo Liu et al. in [4] to develop a system for distinguishing damaged and non-diseased portions of rice leaves. Here, the color characteristics of healthy and disease-affected areas served as input values to BPNN. With BPNN, they were able to identify more than 400 photos of healthy and unhealthy rice leaves with a 90% accuracy rate.

An efficient and error-free leaf disease detection method for potato plants was created by Monzurul Islam et al. in [5] using image processing and machine learning. Their automated method has been tested on more than 300 photos and can distinguish between potato leaves with disease and those that are healthy using the publicly available dataset "Plant Village." The system's accuracy is 95%. Their upcoming effort will focus on creating a system that uses a smartphone.

Malvika Ranjan et al. [6] proposed a straightforward cotton plant disease detection system that makes use of the plant's leaf picture. A picture of the diseased leaf is taken. Then, a distinction between samples that are healthy and those that are sick is carried out utilizing a variety of image processing and Artificial Neural Networks (ANN). The accuracy of this ANN classification is 80%.

With the aid of a slightly modified model of the Convolutional Neural Network (CNN) named LeNet, Prajwala TM et al. proposed a study in [7] to detect and identify illnesses in tomato leaves. The neural network model uses an automatic feature extraction strategy to aid with categorization. The practicality of the neural network is demonstrated by the suggested system's average accuracy of 94–95% in identifying and detecting the leaves.

With the aid of a slightly modified model of the Convolutional Neural Network (CNN) named LeNet, Prajwala TM et al. proposed a study in [7] to detect and identify illnesses in tomato leaves. The neural network model uses an automatic feature extraction strategy to aid with categorization. The practicality of the neural network is demonstrated by the suggested system's average accuracy of 94–95% in identifying and detecting the leaves.

The classification of three prevalent paddy leaf diseases (Brown spot, Leaf blast, and Bacterial blight) and recommendations for fertilizers or pesticides were made by Farhana Tazmim Pinki et al. in [9]. Visual contents (texture, color, and shape) are used as characteristics in this automated system to segment the disease-affected region using K-means clustering. Following that, a support vector machine classifier does the classification. The system's overall accuracy is 92.06%.

An automated Deep Convolutional Neural Network (D-CNN) based technique was created by Md. Rasel Howlder et al. in [10] identify the three main guava leaf diseases: rust, whitefly, and algal leaf spot. They developed a dataset with 2705 photographs divided into four groups—three infected and one healthy leave category—for their study. This method's eleven-layer D-CNN model, which was created using the AlexNet framework, is presented. The proposed approach generated results with a 98.74% accuracy on average.

Cotton leaf diseases have been categorized exactly by Namrata R. Bhimte et al. in [11]. To extract the portion of a leaf image impacted by the disease, K-means segmentation, a color-based segmentation technique, is used. The segmented portion of a picture is classified based on the extraction of pertinent attributes like color and texture. The classification's accuracy has been scored as high as 98.46%.

In [12], Md. Selim Hossain et al. proposed an image processing method that distinguished between and categorized two tea plant illnesses, namely brown blight and algal leaf diseases. For the purpose of identifying these disorders, a support vector machine classifier was employed. The classification procedure included eleven features, and the system provided an accuracy of greater than 90%.

# **Methodology**

## **CNN Model Architecture**

For processing and evaluating visual data, such as photographs, CNN is a potent deep-learning architecture. Convolutional, pooling, and fully connected layers are some of the layers found in CNNs, which enable the network to automatically learn hierarchical representations of the input data. While the pooling layers downsample the feature maps to reduce the spatial dimensions, the convolutional layers add filters to capture local patterns and features. To create predictions, the fully connected layers combine the learned features. By achieving state-of-the-art performance in image classification, object recognition, and image segmentation, CNNs have transformed the field of computer vision, making them essential tools in a wide range of applications, including autonomous driving, medical imaging, and video analysis. Three layers make up a convolutional neural network.

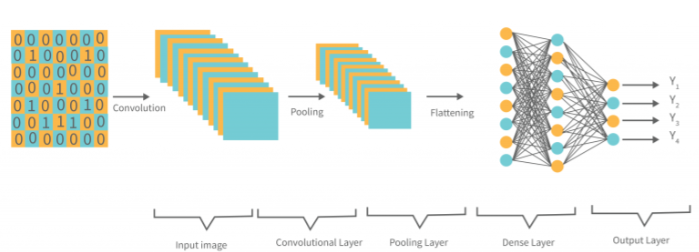


Figure 1: CNN Architecture

The Convolutional Layer: It is the cornerstone of CNN and is responsible for carrying out convolutional operations. The element in this layer that does the convolution operation (matrix) is the Kernel/Filter. The kernel makes horizontal and vertical adjustments depending on the stride rate until the entire image is scanned. While the kernel is smaller than a picture, it is deeper. The kernel height and width will be modest in size if the image has three (RGB) channels, but the depth will cover all three.

Pooling layer: It is in charge of minimizing dimensionality. It helps to lessen how much computing power is needed to process the data. Maximum and average pooling are the two categories into which pooling can be classified. Max pooling returns the highest value from the kernel-covered region of the image. Average pooling gives the average of all the values in the area of the picture covered by the kernel.

Fully Connected Layer (FC): Each input is tied to each neuron in the fully connected layer (FC), which operates with a flattened input. The flattened vector is then transmitted via a few more FC layers, where the usual mathematical functional operations are carried out. At this moment, the classification process begins. If FC layers are present, they are often located near the end of CNN architectures.

## **Transfer Learning Approach**

Transfer learning is the process of applying the skills of a machine learning model that has already been trained to a new, unrelated task. Utilizing what has been learned in one task to enhance generalization in another is the core idea behind transfer learning. We move the weights that a network learns through doing successive tasks. For instance, neural networks in computer vision often attempt to identify edges in the earlier layers, forms in the middle layer, and certain task-specific properties in the latter layers. Only the latter layers are retrained during transfer learning; the early and intermediate levels are utilized. It assists in making use of the labeled data from the initial training task.

Sometimes it is ineffective if the classes for a new problem cannot be distinguished by the features learned by the classification layer, which is the bottom layer. Through the process of transfer learning, we attempt to apply as much of the information from the prior task that the model was trained on to the current task as feasible. More learning occurs as a result of it. With less data, a model may now provide more accurate results.

As opposed to conventional systems, it can reach the desired performance more quickly. Depending on the data and challenge, this knowledge might take many different forms. To reduce training time and improve neural network performance, transfer learning has been applied.

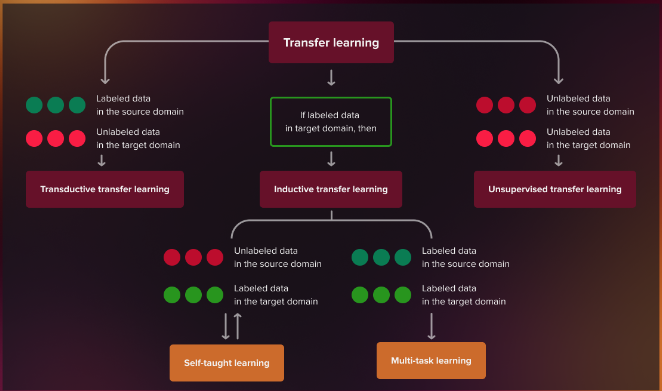


Figure 2: Transfer Learning

* 1. ***Dataset Tracing***

The dataset is a collection of data. It can be in any format, image, text, or in tabular form. For doing machine learning and deep learning research, Kaggle is a well-known dataset platform. Datasets that have been published here can be found in both public and private settings. For this study, we used a publicly available dataset from Kaggle called PlantVillage, which is a collection of photos of plant leaves categorized according to several categories. The collection includes 2152 photos of potato plant leaves.

Early Blight Healthy Late Blight

The potato sample image has three classes: Potato Early Blight, Potato Healthy, and Potato Late Blight. The sample images vary from 152 images to 1000 images. The train test validation split is shown in the below table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Label | Class | Number | Train | Validation | Test |
| 1 | Early Blight | 1000 | 800 | 100 | 100 |
| 2 | Healthy | 152 | 122 | 15 | 15 |
| 3 | Late Blight | 1000 | 800 | 100 | 100 |
| Total |  | 2152 | 1722 | 215 | 215 |

* 1. ***Dataset Preprocessing***

Data preprocessing is a stage in the data mining and data analysis process that takes raw data and converts it into a format that the deep learning architecture can understand and evaluate. CNN architectures employ picture datasets for deep learning. The preprocessing approach converts raw input leaf image datasets into appropriate process datasets format in order to improve the quality of the leaf pictures and remove unwanted areas from the leaf images. The process of image preprocessing is shown below:

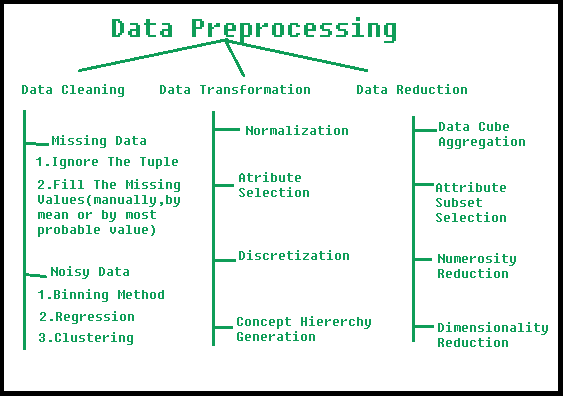


Figure 3: Data Preprocessing

During the data cleaning process, we try to detect and repair flaws or inconsistencies in the dataset. Data integration is merging data from numerous data conflicts and arranging a cohesive representation of data. Because of the varying feature dimensions, huge datasets require more storage space and are more complex to compute. A vast volume of data is reduced during the data reduction process in order to improve picture processing performance and efficiency. Data transformation processes include data smoothing, aggregation, feature building, data normalization, and discretization to reduce the dependability of characteristics in data assessment structures and units for data image conversion. These leaf image files are scaled and translated into the needed dimension of an architecture for analysis of training and testing datasets. As a result, preprocessing approaches can help prepare datasets for identifying leaf diseases using leaf image databases.

Our project's dataset preprocessing phase involves a number of processes that are taken to get the data ready for training, validation, and testing. The dataset is divided into training, validation, and testing sets as the first stage. The dataset is segmented based on a predetermined train size of 0.8, totaling 'len(dataset)' samples. The first 54 samples are used to construct the training set, designated by 'train\_ds,' while the remaining samples are used to build the testing set, denoted by 'test\_ds.'

A val\_size of 0.1 (10% of the dataset) is supplied to further divide the testing set and produce a validation set. The first six samples from the test set are taken to create the validation set, or "val\_ds," while the remaining samples are left in the test set.

The function 'get\_dataset\_partitions\_tf' is used to enable effective data processing; it accepts the original dataset as input along with the necessary training, validation, and test splits. If necessary, the function shuffles the dataset and determines the sizes of each partition in accordance. The necessary samples are taken from the shuffled dataset to construct the resulting partitions, train\_ds, val\_ds, and test\_ds.

The dataset is subjected to extra preprocessing processes after partitioning to improve its appropriateness for training. Each of the train\_ds, val\_ds, and test\_ds are prefetched using the 'tf.data.AUTOTUNE' argument, shuffled with a buffer size of 1000, and cached for quicker access during training. This optimization allows for the effective use of hardware resources and aids in ensuring a constant stream of data throughout training.

The TensorFlow Sequential API is then used to build a data transformation pipeline, denoted by the variable'resize\_and\_rescale'. The Resizing and Rescaling preparation layers, respectively, are used in this pipeline to resize the input images to a predetermined IMAGE\_SIZE and scale their pixel values to the range of 0 to 1. These adjustments make sure that all images have uniform dimensions and normalized pixel values, which helps the model to stabilize and converge throughout training.

* 1. ***Dataset Augmentation***

In image processing, augmentation is a highly common approach. It is engaged in modifying and making leaf image representation easier to recognize leaf disease. It enables us to obtain data that is more varied in nature than what is already there, which improves the training set and, in turn, the model that is being trained. Utilizing flipping, cropping, and rotating techniques, as well as color modification, the augmentation process is used to downsize the original leaf picture collection and convert it to RGB. The enhanced leaf photos, on the other hand, were developed in order to preserve the balanced quality and size of images in the healthy and unhealthy leaf datasets.

Our project pipeline's data augmentation comprises two operations: random rotation and random horizontal and vertical flipping. Before supplying the input images to the model for training, several operations are applied to them.

By randomly flipping the photos along both the horizontal and vertical axes, the random horizontal and vertical flipping process helps bring diversity into the dataset and lessens the sensitivity of the model to object orientation.

The random rotation operation rotates the images randomly within a certain range. Each image will be randomly rotated by up to 0.2 radians in this instance because the rotation range is set to 0.2 radians. The model can learn robust features that are rotation-invariant thanks to this augmentation strategy.

The data augmentation pipeline is then applied to each image in the dataset using the train\_ds dataset, which comprises the training data. By using data\_augmentation(x, training=True), the lambda function applies the data augmentation operations to each input image x and its related label y. Because it guarantees that the augmentation is only used during training and not inference, the training=True argument is crucial.

In order to prefetch and load the augmented data concurrently while the model is training, the prefetch() method is finally used. This improves the training pipeline by lowering I/O latency.

* 1. ***Model Architecture:***

Using the model function and the argument weights='imagenet,' the multiple CNN models that have been pre-trained on the ImageNet dataset (transfer learning) are loaded. The final fully connected layer of the original model, which is in charge of the 1000 ImageNet classes, will not be included because the include\_top parameter is set to False. The model will be employed as a feature extractor instead. The Sequential model is then developed. This layer stack is linear. The base\_model (CNN model) is added as the Sequential model's first layer. This basically links the model's output—the retrieved features—to the following layers. Before connecting to the dense (completely connected) layers, the multidimensional output of the CNN model must be flattened into a 1D vector, which is added after the base model. The addition of a dense layer with 64 units and a ReLU activation function. The flattened input characteristics are subjected to non-linear modifications on this hidden layer. The final step is the addition of a Dense layer with n\_classes units and a softmax activation function. This layer serves as the model's output layer and outputs probabilities for each class. The projected probabilities are ensured to add up to 1 using the softmax activation function. To complete the model's development and define the input shape, the build method is invoked. To identify the model's intermediate layers' forms, this is necessary.

Overall, we build a sequential model using a combination of different CNN architectures as a feature extractor and additional dense layers for classification. The model receives input images of a given size, processes them through the base model to extract features, flattens the output, adds additional transformations through thick layers, and ultimately generates probabilities for each class.

* 1. ***Classification-based CNN models:***

The key purpose of this paper is to compare different types of CNN architectures according to different measures like accuracy, precision, recall, f1 score, etc. For the comparative analysis, we choose nine different CNN architectures. Those are: VGG19, VGG16, Xception, NASNetMobile, InceptionV3, ResNet50, MobileNet well as more recent advancements like DenseNet121 and EfficientNetB0. These nine different architectures have been applied to a total of 2152 images of plant leaves. One of the most popular pre-trained architectures for many variants of computer vision applications is VGG. Three completely linked layers are present at the conclusion of every version. Residual Networks (ResNet) is one of the intricate designs that best exemplifies how complicated a deep learning architecture can be. As the fundamental component of the ResNet architecture, ResNet is made up of several successive residual modules. A CNN architecture called Inception Network was created by Google researchers. The inception module, a revolutionary technique they created, represents a significant departure from the sequential designs we had previously seen. A thick block in the DenseNet architecture where every CNN layer is linked to every descendent layer.

* 1. ***Performance Evaluation***

20 epochs were used to train each pre-trained model on the dataset. For every epoch, we train the model with 32 batch sizes of images. For faster training and testing, we use prefetch and shuffle to introduce randomness into the training, validation, and testing data. The F1 score, recall, precision, and accuracy are used to evaluate the model's performance, together with the macro average and weighted average.

# **Performance Analysis**

* 1. ***Evaluation Metrics***

The following parameters are used to assess the classification performance of the pre-trained models on the testing data, where TP, TN, FP, and FN are defined as True Positive, True Negative, FP, and FN, respectively.

*Precision:*

It defines the probability that the early blight classification made by the classifier is actually an infected leaf or not.

*Accuracy:*

It's the ratio of the correct predictions to the total predictions. It measures how accurately the model can predict results.

*Recall:*

It's the ratio between the actual truly predicted value by the classifier and the total number of positive values.

*Macro Average:*

The arithmetic mean of each class's precision, recall, and f1 score makes up the macro average. When it's necessary to treat all classes equally, we utilize macro average scores to assess the classifier's overall performance in comparison to the most typical class labels.

*Weighted Average*:

The weighted-average F1 score, precision, and recall are calculated by averaging each per-class score while taking into consideration each class's support. The term "weight" essentially refers to the share of support for each class in relation to the total worth of support.

* 1. ***Experimental Results***

The classification performance of transfer learning models of different CNN architectures is given below:

*Accuracy& ROC*:

|  |  |  |
| --- | --- | --- |
| Architectures | Accuracy | ROC |
| VGG19 | 92.1% | 99.1% |
| VGG16 | 92.9% | 99.2% |
| MobileNet | 100% | 100% |
| Inception V3 | 99.2% | 100% |
| Xception | 100% | 100% |
| NASNetMobile | 50% | 45.8% |
| ResNet50 | 96.8% | 98.7% |
| DenseNet 121 | 99.6% | 100% |
| EfficientNetB0 | 95.3% | 99.09% |

Here we can see that among these nine architectures, Mobile Net and Xception architecture give the highest accuracy and ROC value which is 100%, which means these two architectures can detect the images successfully. Among this architecture, the lowest accuracy is 50%, and the lowest ROC value is 45.8% which is provided by the NASNetMobile.

*Precision:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Architecture | Precision | | | | |
|  | Early Blight | Healthy | Late Blight | Macro Avg | Weighted Avg |
| VGG19 | 91% | 77% | 96% | 88% | 93% |
| VGG16 | 92% | 76% | 97% | 88% | 93% |
| MobileNet | 100% | 100% | 100% | 100% | 100% |
| Inception V3 | 98% | 100% | 100% | 99% | 99% |
| Xception | 100% | 100% | 100% | 100% | 100% |
| NASNetMobile | 100% | 100% | 50% | 81% | 75% |
| ResNet50 | 97% | 85% | 98% | 94% | 97% |
| DenseNet121 | 100% | 95% | 100% | 98% | 100% |
| EfficientNetB0 | 96% | 100% | 95% | 97% | 95% |

Here, MobileNet and Xception architecture give the highest precision for all three classes among these nine architectures, which is 100%. It means these two architectures detect images accurately as their macro score, and the weighted score is 100% as well. NASNetMobile will not predict classes accurately as it gives the lowest value of precision, macro-score, and weighted average. It has high precision in Early Blight and healthy classes but the lowest precision in the Late Blight class, which is 50%. So, it will not predict Late blight class properly.

*Recall:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Architecture | Recall | | | | |
|  | Early Blight | Healthy | Late Blight | Macro Avg | Weighted Avg |
| VGG19 | 96% | 94% | 88% | 93% | 92% |
| VGG16 | 98% | 89% | 89% | 92% | 93% |
| MobileNet | 100% | 100% | 100% | 100% | 100% |
| Inception V3 | 100% | 100% | 98% | 99% | 99% |
| Xception | 100% | 100% | 100% | 100% | 100% |
| NASNetMobile | 00% | 00% | 100% | 33% | 50% |
| ResNet50 | 99% | 94% | 95% | 96% | 97% |
| DenseNet121 | 100% | 100% | 99% | 100% | 100% |
| EfficientNetB0 | 99% | 67% | 96% | 100% | 100% |

For Recall, NASNetMobile has the lowest values of macro-score and weighted average. This means the architecture cannot predict correctly. All other architectures of these nine have a high value of Recall, which means these work well on prediction. EfficientNetB0 has low precision for healthy classes, and NASNetMobile has very low precision for early blight and healthy classes. So, these two architectures will not work well in the classes.

*F1 Score:*

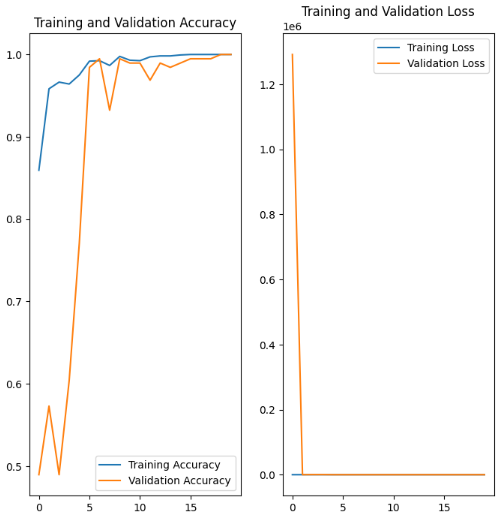
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Architecture | F1 Score | | | | |
|  | Early Blight | Healthy | Late Blight | Macro Avg | Weighted Avg |
| VGG19 | 94% | 85% | 92% | 90% | 92% |
| VGG16 | 95% | 82% | 93% | 90% | 93% |
| MobileNet | 100% | 100% | 100% | 100% | 100% |
| Inception V3 | 99% | 100% | 99% | 99% | 99% |
| Xception | 100% | 100% | 100% | 100% | 100% |
| NASNetMobile | 00% | 00% | 67% | 22% | 33% |
| ResNet50 | 98% | 89% | 97% | 95% | 97% |
| DenseNet121 | 100% | 97% | 100% | 99% | 100% |
| EfficientNetB0 | 97%% | 80% | 95% | 91% | 95% |

Here, Dense Net, Xception, and MobileNet architectures give 100% F1- scores which assure that these will predict well on all three classes. NASNetMobile gives a very low score of 0% for two classes and 67% for the late blight class. It will not predict classes properly. This architecture will give many errors in predicting each class.

* 1. ***Graph Analysis:***

The graphs of training and validation accuracy, and loss is given below:

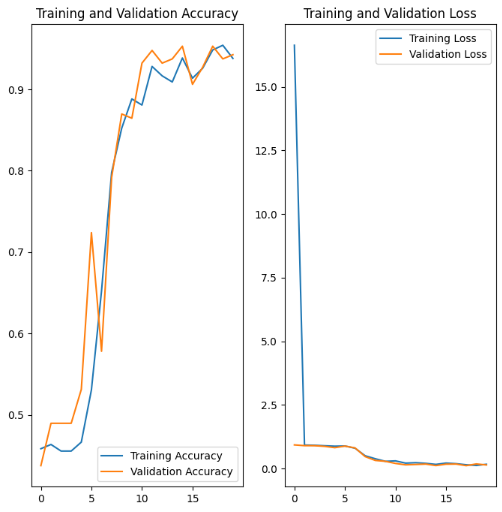
*DenseNet:*



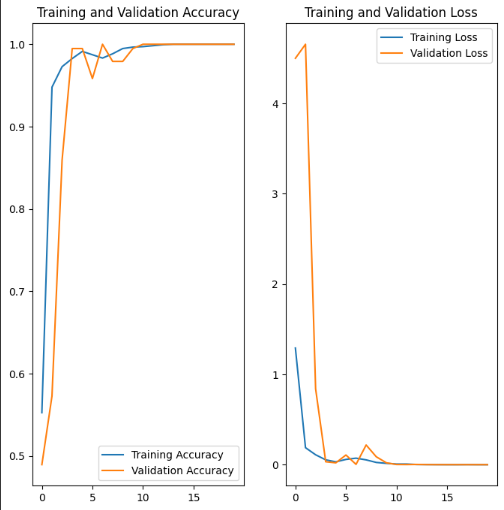
*ResNet50:*



*Vgg16:*



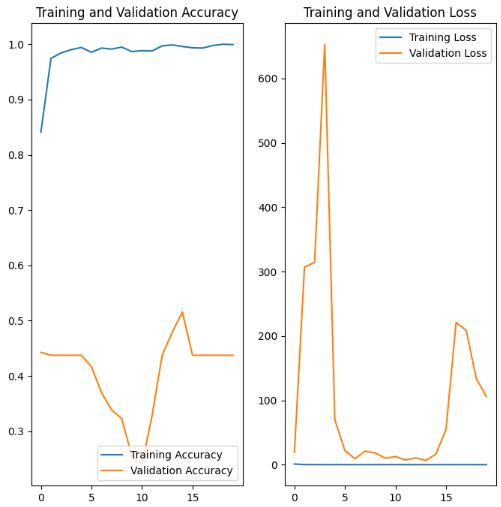
*Xception:*



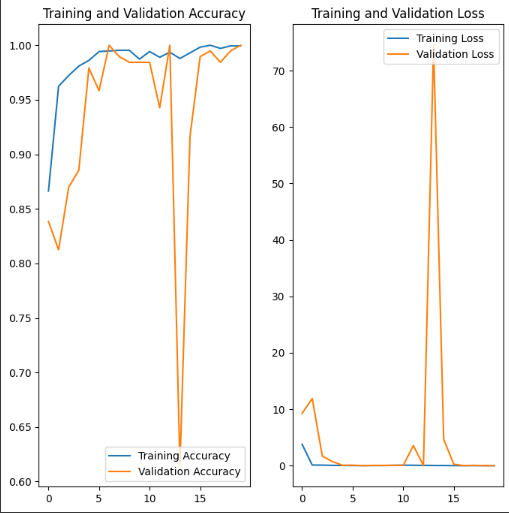
*Vgg19:*



*NasNetMobile:*



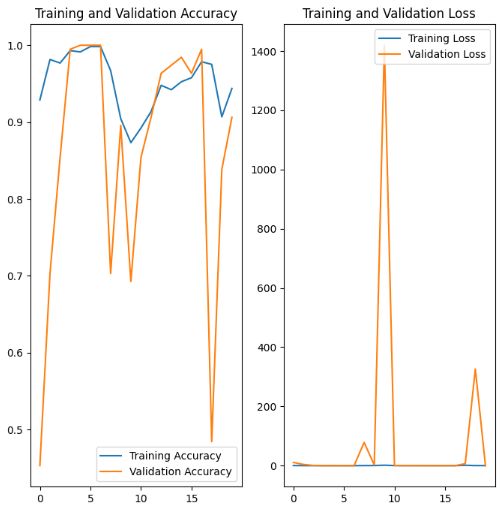
MobileNet:



*InceptionV3:*



*EfficientB0:*



After studying these graphs thoroughly, we can see that whilst most of the models’ curves have increased in Training and Validation Accuracy steadily and decreased in Training and Validation loss with each running epoch, some architectures have shown signs of instability. ResNet50, NasNetMobile, and EffecientNetB0 have irregular curves, which indicate overfitting, or the learning rate may be too high. Regularization techniques, adjusting the learning rate, and early stopping may combat these instabilities.

* 1. ***Comparative Analysis:***

After careful comparison, we can see that Xception and MobileNet are working exceptionally well in relation to the other architectures, as we can see the high accuracy, roc, precision, recall, and F1 score they produce. Overall NasNetMobile is the worst-performing architecture due to its abysmal scores all around.

# **Future Work**

Future research can go in a number of directions to improve the functionality and applicability of the system for classifying potato leaf disease that is based on transfer learning and different CNN architectures.

Although transfer learning was successful in this investigation, further improvement may be possible by tweaking the chosen base models. The model can be made to respond more specifically to the classification job for potato leaf disease by enabling the pre-trained layers' weights to be changed during training.

*Data Augmentation Methods*: For the current investigation, fundamental data augmentation methods, including random flipping and rotation, were used. Further research into cutting-edge augmentation techniques like scaling, picture translation, and Gaussian noise injection may lead to better generalization and enhanced model performance.

*Hyperparameter Optimization*: The effectiveness of the models is significantly impacted by the selection of hyperparameters. It is possible to discover the ideal set of hyperparameters for each base model and maybe enhance their performance by carrying out a systematic hyperparameter search, such as grid search or Bayesian optimization.

*Ensemble Methods*: Combining the predictions of multiple models through ensemble techniques, such as majority voting or stacking, could potentially enhance the overall classification performance. By leveraging the diversity and complementary strengths of different CNN architectures, ensemble methods can provide more robust and accurate predictions.

*Additional Classes and Datasets*: Adding more classes to the classification system and gathering larger and more varied datasets will help produce a more thorough and workable answer. The addition of new classes, including early blight, late blight, and normal, makes disease diagnosis more precise and thorough.

# **Conclusion**

In this study, we assessed the performance of nine different CNN architectures for classifying potato leaf disease and examined the efficacy of transfer learning. We learned a lot about each model's capabilities and suitability for the task at hand through rigorous trial and review. We determined the top-performing CNN architecture for classifying potato leaf disease based on evaluation measures such as accuracy, F1 score, recall, precision, and ROC. The outcomes proved the effectiveness of transfer learning in utilizing the knowledge of previously trained models and customizing it to the particular categorization task. The results emphasize the significance of choosing an acceptable base model and optimizing it for performance. Furthermore, the generalization capability and accuracy of the models were enhanced by the use of data augmentation methods and careful evaluation of hyperparameters. Future work to further improve the classification method is still possible, though. Accuracy and robustness can be increased by investigating fine-tuning, sophisticated data augmentation techniques, hyperparameter optimization, ensemble approaches, and increasing the dataset and classes. Overall, the created system for classifying potato leaf diseases, which is based on transfer learning and several CNN architectures, yields encouraging results and may find use in agricultural contexts. This approach can help farmers and researchers take timely and focused actions to reduce crop damage and increase agricultural productivity by precisely recognizing and diagnosing potato leaf diseases.

##### References

1. R. M. Prakash, G. P. Saraswathy, G. Ramalakshmi, K. H. Mangaleswari, and T. Kaviya, "Detection of leaf diseases and classification using digital image processing," 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, 2017, pp. 1–4.
2. H. Waghmare, R. Kokare, and Y. Dandawate, "Detection and classification of diseases of the Grape plant using opposite color Local Binary Pattern feature and machine learning for automated Decision Support System,” 2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN), Noida, 2016, pp. 513–518.
3. S. Biswas, B. Jagyasi, B. P. Singh and M. Lal, “Severity identification of Potato Late Blight disease from crop images captured under uncontrolled environment,” 2014 IEEE Canada International Humanitarian Technology Conference—(IHTC), Montreal, QC, 2014, pp. 1–5.
4. L. Liu and G. Zhou, “Extraction of the Rice Leaf Disease Image Based on BP Neural Network,” 2009 International Conference on Computational Intelligence and Software Engineering, Wuhan, 2009, pp. 1–3.
5. M. Islam, Anh Dinh, K. Wahid, and P. Bhowmik, "Detection of potato diseases using image segmentation and multiclass support vector machine," 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, ON, 2017, pp. 1–4.
6. Malvika Ranjan, Manasi Rajiv Weginwar, NehaJoshi, A.B. Ingole, “Detection and Classification of Leaf Disease Using Artificial Neural Network,” International Journal of Technical Research and Applications, 2015, pp. 331–333.
7. P. Tm, A. Pranathi, K. SaiAshritha, N. B. Chittaragi, and S. G. Koolagudi, “Tomato Leaf Disease Detection Using Convolutional Neural Networks,” 2018 Eleventh International Conference on Contemporary Computing (IC3), Noida, 2018, pp. 1–5.
8. Aparajita, R. Sharma, A. Singh, M. K. Dutta, K. Riha, and P. Kriz, “Image processing based automated identification of late blight disease from leaf images of potato crops,” 2017 40th International Conference on Telecommunications and Signal Processing (TSP), Barcelona, 2017, pp. 758–762.
9. F. T. Pinki, N. Khatun, and S. M. M. Islam, "Content-based paddy leaf disease recognition and remedy prediction using support vector machine," 2017 20th International Conference of Computer and Information Technology (ICCIT), Dhaka, 2017, pp. 1–5.
10. M. R. Howlader, U. Habiba, R. H. Faisal, and M. M. Rahman, "Automatic Recognition of Guava Leaf Diseases using Deep Convolution Neural Network," 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox’sBazar, Bangladesh, 2019, pp. 1–5.
11. N. R. Bhimte and V. R. Thool, “Diseases Detection of Cotton Leaf Spot Using Image Processing and SVM Classifier,” 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2018, pp. 340–344.
12. S. Hossain, R. M. Mou, M. M. Hasan, S. Chakraborty, and M. A. Razzak, "Recognition and detection of tea leaf's diseases using support vector machine," 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA), Batu Feringghi, 2018, pp. 150–1545.