**Abstract:**

Tomatoes have surpassed potatoes, lettuce, and onions as the fourth most popular fresh-market vegetable. The benefits of consuming various fruits and vegetables are numerous, and tomatoes are no exception**.** As tomato demand is quite high throughout the world, we must ensure its proper cultivation and production. Numerous diseases damage tomato leaves and fruits during tomato growing. Among all diseases, tomato leaf disease is the most destructive, harming entire tomato fields. Early diagnosis of tomato disease can help us in mitigating these types of losses. I have developed a simple CNN-based technique for the early detection of tomato leaf disease in this research. Due to the model's simplicity, it requires less time to compute and provides a higher level of accuracy. I used total

**Introduction:**

As agriculture struggles to feed the world's rapidly rising population, plant disease degrades food, fiber, and biofuel crop output and quality. Losses can be catastrophic or chronic, but they account for about 42% of the production of the six most important food crops. Plant leaf diseases are one of the most damaging of all the diseases that may affect plants. Many farmers spend millions on plant leaf disease management, frequently without proper scientific assistance, which leads to poor disease control, contamination, and adverse outcomes. [1]. Most of the time, tomatoes are grown in almost every part of the world. Tomato leaf disease has a major effect on the number of tomatoes that are produced. If we identify disease early, we can avoid a big loss.

Human experts identify diseases manually and they face many difficulties

during identification. Here, identifying and classifying diseases in a precise way and quickly will be of great value. Advances in artificial intelligence research now make it possible to make automatic plant disease detection from raw images [2]. Convolutional Neural Network (CNN) has become a powerful tool for any classification task as it automatically extracts important features from images without human supervision [3]. When it comes to computing power, it has grown over time. Now, we can train a deep neural network in just a few hours.

To classify the tomato leaves diseases, I used a simple Convolutional Neural Network with three convolution layers, four dense layers and an output layer. It's a very simple model, so we can put it on any small device. I used 22948 images from the [New Plant Diseases Dataset](https://www.kaggle.com/vipoooool/new-plant-diseases-dataset)[4] to make the model. When the model was created, 18355 images were used to train it, and the rest were used for validation and testing. The model's training accuracy is 97.6% while its validation accuracy is 93.2%.

The predicted accuracy of the test data is 93.6% with an f1 score of .94. I used relu as the activation function for the convolution and the dense layer, and softmax as the classifier for the output classification.

Bacterial Spot Early Blight

**Some Photos from New Plant village dataset**

**Methods:**

My proposed model accepts images of tomato leaves as input and generates class labels for each instance. For classification, a basic CNN model was used. This is a straightforward model composed of three convolutional layers, four dense layers, and an output layer. Following each convolution layer by a max-pooling layer.

For all layers except the output layer, I applied the relu as an activation function. For disease classification, I utilized softmax as the activation for the final layer. I chose batch normalization to normalize each layer's output since it helps us to run the model quickly.

—>> model.jpg

**Convolution:**

The basic building elements of convolutional neural networks are convolutional layers.

Convolution is the simplest process of applying a filter to an input that results in activation. When the same filter is applied repeatedly to an input, a map of activations known as a feature map is created, representing the positions and strength of a recognized feature in input, such as an image [5].

Pooling is an important necessary stage that is closely connected to convolution. Pooling is mostly used to extract sharp and smooth characteristics. Additionally, this is done to minimize variance and calculations. Max-pooling enables the extraction of low-level features such as edges, points, and so on. While Avg-pooling prioritizes smooth characteristics.

In my model, I used three convolution layers followed by three max-pooling layers. For each convolution, I used a 3\*3 size kernel and a total of 64 filters in the first layer, 128 filters in the second layer, and 256 filters in the third layer.

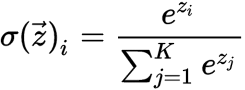
**Activation Function:**

The activation function is the nonlinear transformation of the input signal that we do. This transformed output is subsequently used as input for the following layer of neurons. Without an activation function, a neural network is just a linear regression model [6].

y = max(0, x) —-equation fro relu

I used relu as the activation function for layers in my model. ReLUs can help neural networks perform better by speeding up training. The gradient computation is really straightforward (either 0 or 1 depending on the sign of x). Additionally, a ReLU's computational phase is straightforward: all negative components are set to 0.

In the last layer, I applied the softmax function as an activation function to classify network output.



Equation softmax function—-

**Regularization:**

Regularization is a collection of Machine Learning methods for reducing generalization error. After training, most models perform excellently on a subset of the whole population, but they fail to generalize well. This is also known as overfitting. Regularization algorithms are designed to decrease overfitting while keeping the training error to a minimum.

Overfitting of neural networks may be avoided with the use of dropout regularization. When neurons and their connections are disabled, dropout occurs. All neurons must learn to generalize better to avoid the network from over-relying on a few specific neurons.To prevent our model from overfitting, I applied dropout as a regularization method. In addition, we have a number of different options to prevent overfitting.

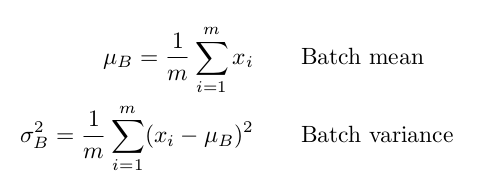
**Batch Normalization:**

Batch normalization (also known as batch norm) is a method used to make [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network) and convolutional neural networks faster and more stable through normalization of the layers' inputs by re-centering and re-scaling [7] .This stabilizes the learning process and significantly reduces the number of training epochs needed to train deep networks.

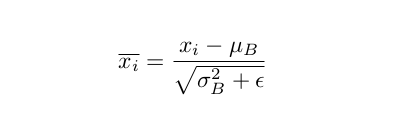
Batch normalization requires three stages. First, we must compute the mean and variance of the layers' input. In the second stage, we must normalize it using previously determined batch statistics. Finally, the normalized output must be scaled and shifted.

Equation for batch Normalization(Three equation) --—->><https://towardsdatascience.com/batch-normalization-theory-and-how-to-use-it-with-tensorflow-1892ca0173ad>

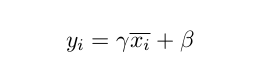
1ST EQUATION–for mean and variance



2ND EQUATION –Normalization



3RD EQUATION – Scaling and Shifting



I used Batch Normalization in every layer of convolutional neural networks and artificial neural networks.

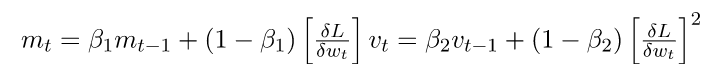
**Loss Function:**

Loss functions are used to calculate the difference between our algorithms' output and the provided target value. It's a technique for determining how effectively our algorithm represents our dataset. If our predictions are incorrect, the loss function will return a greater value. If they're really good, it'll give us a lower number.There are different types of loss functions. I used sparse categorical cross entropy for calculating loss.

**Optimizers:**

While training the deep learning model, we must adjust the weights for each epoch and optimize the loss function. An optimizer is a function or algorithm that adjusts the neural network's parameters such as weights and learning rate. As a result, it helps to reduce total loss and enhance accuracy.

There are many types of optimization algorithms. In my model, I used Adam as an optimization technique. Deep learning models can be trained using Adam instead of stochastic gradient descent (SGD). AdaGrad and RMSProp combine their greatest features in Adam, resulting in an optimization technique that can handle noisy situations with sparse gradients.

ADAM OPTIMIZER

**Experiments :**

**3.1 Experimental settings:**

Image-based disease prediction is the primary target of this research. I've developed a convolutional neural network from scratch to predict diseases. Tomato leaf disease has been detected using a variety of methods in the past. Many of them are extremely complicated and need an excessive amount of training time. To make it run faster, I went in with a more simplistic design. New plant disease dataset [4] was used for training, validation and testing purposes. There are a total of 22948 photos of tomato leaf diseases in this collection. More than 80% of the images were used for training purposes, while the rest were used for validation and training.

I used the Adam optimizer with parameters for loss optimization. In my training, there are a total of 15 epochs. Except for a few epochs, loss and accuracy decreased after iterations. My model has a total of 119,531,146 parameters, of which 119,525,642 were trainable and 5,504 were not.

—>> Learning curve

→>> loss curve

Parameters of ADAM optimizer:

| **Name** | **Value** |
| --- | --- |
| Learning Rate | 0.0001 |
| Beta1 | 0.9 |
| Beta2 | 0.999 |
| Epsilon | 1e-7 |
| decay | 1e-7 |

**Evaluation criteria:**

The three most important metrics for evaluating a classifier are accuracy, precision, and recall. The percentage of correct predictions for the test data is known as accuracy. Precision is defined as the percentage of relevant examples (true positives) among all the examples expected to belong to a specific class. The fraction of examples predicted to belong to a class compared to all of the examples that truly belong in the class is known as recall. In my model on test data I got accuracy of 93.6%, precision of .94 and recall of .94

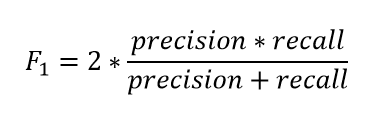
— classification report

**Accuracy = correct predictions / all predictions**

**Precision = true positives / (true positives + false positives)**

**Recall = true positives / (true positives + false negatives)**

I also used F1 score in addition to these three metrics. The F1 score is the harmonic mean of precision and recall, and it ranges from 0 to 1. My model achieved an F1 score of .94 on test data.



**Results:**

On training data, validation data my model had a sparse categorical accuracy of 97.6% and 93.2% and on test data my model had an accuracy of 93.6%. For test data, I've also used a variety of evaluation measures. On my final test data, I have a precision of .94, a recall of .94, and an F1 score of .94. I have also used a confusion matrix for proper visualization of predicted results. A confusion matrix is a summary of classification problem prediction outcomes. The number of correct and incorrect predictions is calculated and broken down by class using count values. It's easy to point out how well my model worked on test datasets based on the confusion matrix[figure].

—cinfusion matrix

**Discussion:**

With a prediction accuracy of 93.2 percent, this simple model works really well for tomato leaf diseases. This outcome is the result of a number of issues. Batch Normalization, dropout with proper likelihood, Adam optimizer with learning rate and decay, sparse category accuracy, and the relu activation function, in my opinion, all played a key role in predicting this outcome.

Batch Normalization technique helps artificial neural networks and convolutional neural networks to become faster and more stable through normalization of the layers' inputs by re-centering and re-scaling [7]. Overfitting can be overcome by dropout. The Adam optimizer can deal with noisy situations with sparse gradients, and relu activation exposes object non-linearity.

**Conclusion:**

The main goal of this research is to predict tomato leaf diseases with a very simple CNN model. I've created this simple model from scratch and it worked well on test datasets. As my model is very simple I think it will work on any lightweight devices furthermore it takes much much less time to train than usual model.

We should keep in mind that a basic model can provide better results. I believe that this study will inspire many researchers to create a model from scratch.

**Citations:**

1. <https://www.mdpi.com/journal/agriculture/special_issues/plant_disease>
2. K. Elangovan and S. Nalini, “Plant disease classification using image segmentation and SVM techniques”, International Journal of Computational Intelligence Research, vol. 13(7), pp. 1821-1828, 2017.
3. <https://arxiv.org/pdf/2109.02394.pdf>
4. <https://www.kaggle.com/vipoooool/new-plant-diseases-dataset>
5. <https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/>
6. <https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/>
7. <https://en.wikipedia.org/wiki/Batch_normalization>