



# DEEP LEARNING FOR IDENTIFICATION OF PLANT LEAF DISEASES

How Data Science, AI and Machine learning support sustainable  
Kenya's agriculture

Jennifer Njeri , Kelvin Muia & Julliet Iswana,  
13 December 2023

---

The agriculture sector in Kenya is essential, contributing 20% to the GDP and indirectly affecting another 27% through related sectors. It employs over 40% of the total population and 70% of rural inhabitants. Recognizing its importance in providing livelihoods and food, the Central Bank of Kenya launched the Survey of the Agriculture Sector in July 2022. This survey, alongside the CEOs and Market Perceptions Surveys, aims to generate frequent data to inform food supply, prices, and agricultural challenges, thereby supporting monetary policy decisions.

The Economic Survey of 2022 revealed a slowdown in agricultural growth, from 5.2% in 2020 to a 0.1% contraction in 2021, mainly due to unfavorable weather, reducing crop and livestock performance. Maize production fell from 42.1 million bags in 2020 to 36.7 million in 2021, with similar trends in potatoes, beans, coffee, wheat, and tea. Factors like rising input costs, leaf rust infestation, and land use shifts to real estate contributed to this decline. In 2022, the sector contracted further in the first three quarters.

The January 2023 Agriculture Sector Survey focused on recent trends in agricultural commodity prices and outputs across the country. It also assessed key food commodity availability. The survey's areas of interest included tracking commodity prices, assessing output and acreage, farm input usage, and factors affecting production.

The improvement of crop yields in Kenya through technology is a critical issue, given that small farmers, who own less than five acres each, constitute up to 75% of the country's agricultural workforce. These farmers traditionally rely on outdated methods and lack access to modern technologies that could enhance productivity and ease their labor.

---

### **Challenges in Enhancing Farming Productivity**

Small farmers face significant challenges due to their reliance on traditional farming methods.

There is a pressing need to integrate modern technology into agriculture to increase crop yields, improve income, and reduce labor hours.

Advantech's "Farming as Business" program aims to provide data and decision-making tools to farmers in East Africa.

They offer a smartphone app providing advanced weather forecasts, market trends, and information on soil types, fertilizers, and pesticides, helping farmers make data-driven decisions.

Agriculture, a cornerstone of human sustenance, has embraced technological advancements to bolster productivity and address challenges. In this vein, the proposed project delves into the realm of plant health, specifically the early identification of leaf diseases. Acknowledging the pivotal role of timely disease detection in safeguarding crop yield, we aim to leverage deep learning, employing a deep convolutional neural network (CNN). This venture is fueled by a dataset meticulously compiled by Arun Pandian J and Geetharamani Gopal

This project seeks to leverage the power of deep learning in the field of agriculture, specifically in the area of plant disease identification.

We are working on developing a Convolutional Neural Network (CNN) model for Advantech's mobile app, aimed at helping farmers diagnose plant diseases through image recognition, which involves several steps.

This model would leverage the power of machine learning and image processing to identify various plant diseases from photos taken by farmers. This model will help in the accurate and timely detection of plant diseases which plays a crucial role in ensuring food security and sustainability. With the advent of precision agriculture and the application of AI in this field, there is an opportunity to develop automated systems for early disease detection in crops. The successful completion of this project could significantly improve disease management in crops, leading to better agricultural practices and increased yields.

---

## **Project Objective**

The primary objective of this project is to develop a CNN model capable of accurately classifying different classes of potato plant leaf diseases, as well as distinguishing healthy leaves and unhealthy leaves. The project aims to create an efficient and reliable model that can be used for automated disease detection in agricultural applications

## **Methodology**

**Data Preprocessing:** Perform additional image preprocessing as needed. Split the dataset into training, validation, and testing sets.

**Model Development:** Design and implement CNN. Experiment with different layer configurations and hyperparameters to optimize the model.

Model Training and Validation: Train the model using the training set. Validate and tune the model using the validation set to avoid overfitting.

Model Evaluation: Evaluate the model's performance on the test set using metrics like accuracy, precision, recall, and F1-score.

Deployment Strategy: Plan the integration of the model into a practical application using StreamLit where users will be able to upload images of their leaves and get a diagnosis of the plant.

Expected Outcomes: A highly accurate and reliable CNN model for the classification of plant leaf diseases. A comprehensive report detailing the model's performance, limitations, and potential applications.

---

## **DATA PREPROCESSING**

### **Augmentation vs. No Augmentation**

Our dataset comprises two sets of images: one with augmentation and another without. In the absence of augmentation, images maintain their original form, resulting in limited dataset diversity and an elevated risk of overfitting.

In contrast, augmented images undergo transformations, promoting greater diversity, improved generalization to new data, and a reduction in overfitting.

The advantages of augmentation include enhanced generalization, increased model robustness across different patterns, and improved data efficiency by expanding the dataset without the need for additional data collection; a particularly valuable asset in resource-constrained scenarios. For our work, we will utilize images with augmentation.

---

# EXPLORATORY DATA ANALYSIS

Since we were working with images the EDA involved various techniques to understand and analyze the characteristics and patterns within our image dataset.

## VISUAL INSPECTION

A subset of images was visually inspected to understand the general characteristics, such as color distribution, presence of noise, and overall image quality. We had 3 classes as per the below image Potato early blight, Potato Late blight, and Potato healthy.

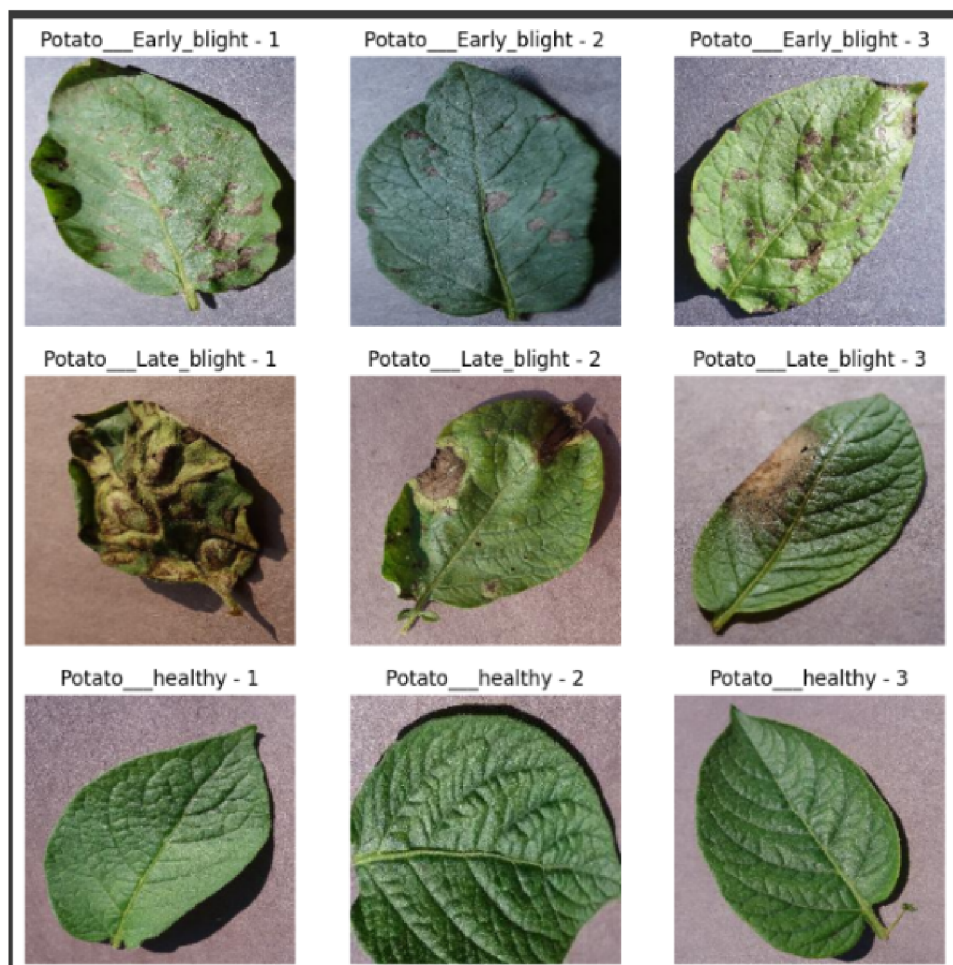


image classes

---

## Image Size Distribution

The potato leaf images in our dataset have consistent dimensions, with the same height and width. With an aspect ratio of 1.0, the images have the same height and width. To confirm this further we will check the shape of the whole dataset. There is no need to resize or reshape the images.

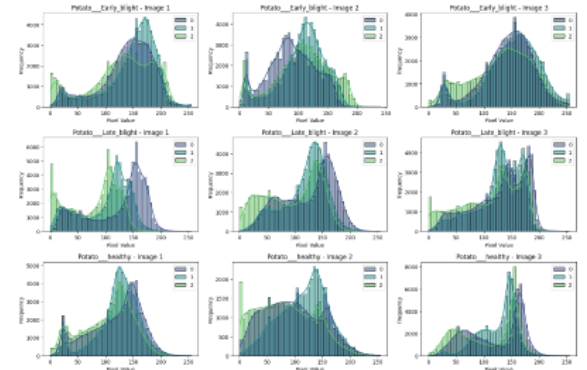




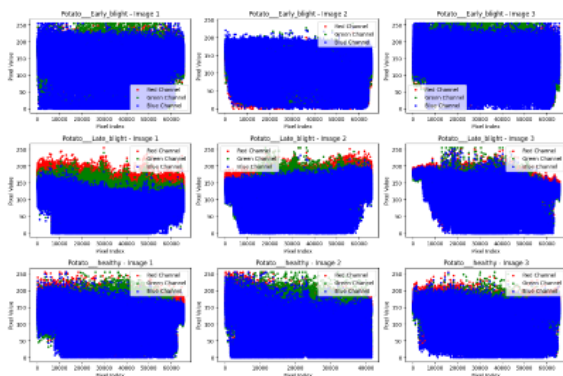
potato leaf aspect ratio

## Colour Distribution of the Images

The diverse distributions across different classes suggest variations in color characteristics and patterns unique to each class. The presence of right or left skewness and different peaks indicates the diversity of color compositions within our dataset.



color\_distribution\_for\_classes



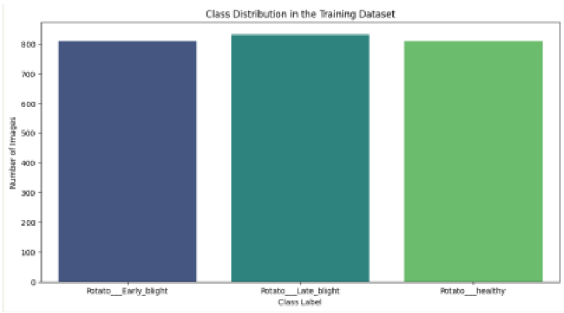
RGB Distributions

We analyzed the RGB distributions in various images to check if there's a consistent presence of green, yellow, and brown tints as is in our images. The blue channel dominance aligns with the common color characteristics of green leaves, as chlorophyll, the primary pigment responsible for photosynthesis, absorbs mostly in the blue part of the spectrum

# class distribution

Our dataset has a well-balanced distribution among the classes:

- Potato\_\_\_Early\_blight: 800 images
- Potato\_\_\_Late\_blight: 832 images
- Potato\_\_\_healthy: 800 images



class distribution

This balanced representation provides a good foundation for training our model to recognize various aspects of early blight, late blight, and healthy potato leaves.

## MODEL ARCHITECTURE.

### Basic Model - CNN

Our CNN model will use ReLU to introduce non-linearity and capture complex patterns in hidden layers and a Softmax output layer to convert raw scores into class probabilities for multi-class classification.

```
Model: "sequential"
Layer (type)                Output Shape              Param #
-----
conv2d (Conv2D)             (None, 222, 222, 32)     896
max_pooling2d (MaxPooling2D) (None, 111, 111, 32)     0
flatten (Flatten)           (None, 394272)           0
dense (Dense)               (None, 128)              50466944
dense_1 (Dense)             (None, 3)                387
-----
Total params: 50468227 (192.52 MB)
Trainable params: 50468227 (192.52 MB)
Non-trainable params: 0 (0.00 Byte)
```

Base model

Our base model which is sequential in Keras and has four layers. These layers include:

- Conv2D:** This is a convolutional layer that applies a convolution operation to the input images. It has 896 parameters.
- MaxPooling2D:** This is a pooling layer that reduces the spatial dimensions (height and width) of the input volume. It doesn't have

trainable parameters.

**Flatten:** This layer flattens the input without affecting the batch size. It also has no trainable parameters.

**Dense:** This is a fully connected neural network layer. The first dense layer in the model has 128 nodes (or neurons), and the second dense layer has 3 nodes.

The total number of trainable parameters in the base model is 50,468,227. These parameters are learned during the training process to optimize the model's performance on a specific task of classification of leaf diseases

---

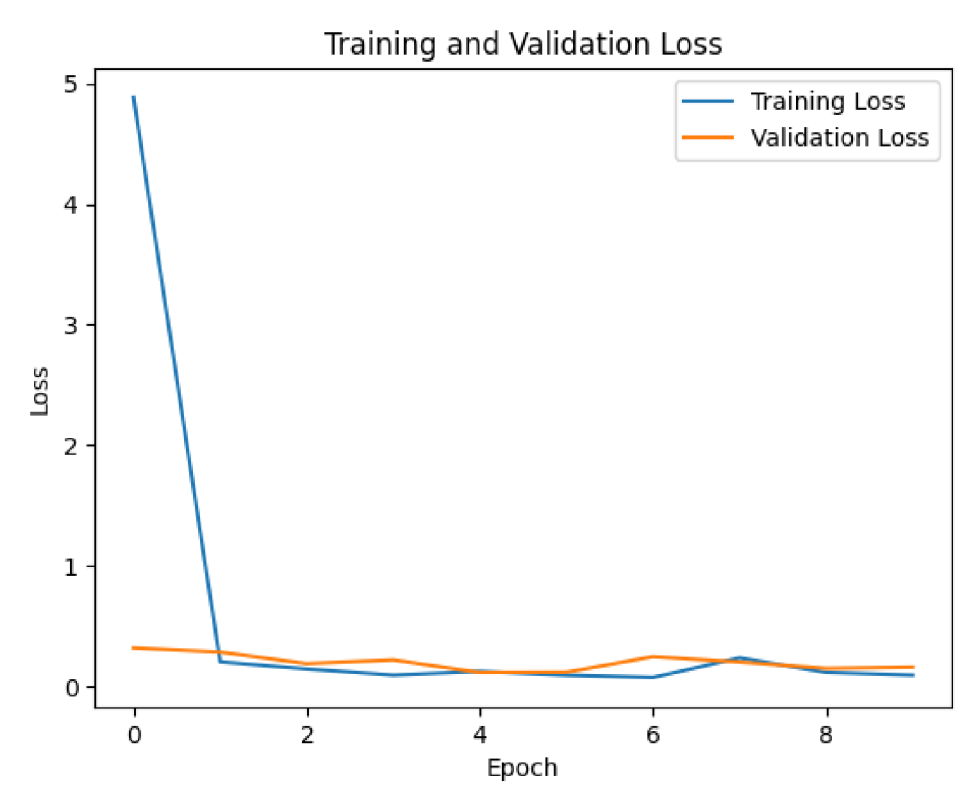
## Train, Visualize Training Loss and Training Accuracy of Base Model

The image shows the progress of our neural network base model over 10 epochs. Both the training and validation accuracy generally increase, which is a good sign. However, the training and validation loss fluctuate. This is not uncommon in neural network training and can be due to various factors like learning rate, the complexity of the model, or the nature of the data

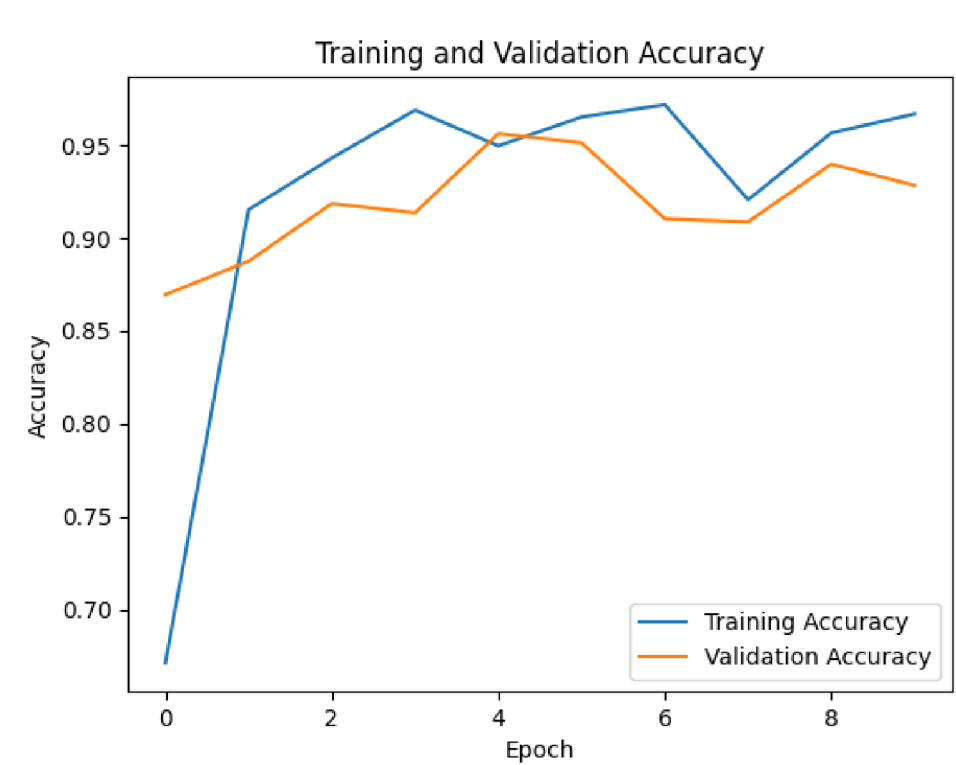
```
Epoch 1/10 [=====] - 280s 36s/step - loss: 2.8944 - accuracy: 0.4904 - val_loss: 0.3798 - val_accuracy: 0.8287
Epoch 2/10 [=====] - 238s 36s/step - loss: 0.3122 - accuracy: 0.8828 - val_loss: 0.2634 - val_accuracy: 0.9193
Epoch 3/10 [=====] - 253s 36s/step - loss: 0.2832 - accuracy: 0.9243 - val_loss: 0.2993 - val_accuracy: 0.8715
Epoch 4/10 [=====] - 226s 36s/step - loss: 0.2118 - accuracy: 0.9316 - val_loss: 0.2418 - val_accuracy: 0.8876
Epoch 5/10 [=====] - 228s 36s/step - loss: 0.1951 - accuracy: 0.9383 - val_loss: 0.2239 - val_accuracy: 0.9176
Epoch 6/10 [=====] - 234s 36s/step - loss: 0.1258 - accuracy: 0.9580 - val_loss: 0.2697 - val_accuracy: 0.8862
Epoch 7/10 [=====] - 244s 36s/step - loss: 0.1223 - accuracy: 0.9531 - val_loss: 0.2232 - val_accuracy: 0.9188
Epoch 8/10 [=====] - 230s 36s/step - loss: 0.1343 - accuracy: 0.9498 - val_loss: 0.2365 - val_accuracy: 0.9038
Epoch 9/10 [=====] - 242s 36s/step - loss: 0.0778 - accuracy: 0.9667 - val_loss: 0.1851 - val_accuracy: 0.9325
Epoch 10/10 [=====] - 234s 36s/step - loss: 0.1487 - accuracy: 0.9581 - val_loss: 0.1243 - val_accuracy: 0.9536
```

EPOCH LOG base model





base model training vs validation loss



base model training vs Validation accuracy

---

## Evaluate the Basic CNN model

```
# Evaluate the model
evaluation_results = base_cnn_model.evaluate(validation_generator)
print("Validation Accuracy: {:.2f}%".format(evaluation_results[1] * 100))
print("Validation Loss: {:.4f}".format(evaluation_results[0]))
```

```
19/19 [=====] - 20s 1s/step - loss: 0.1140 - accuracy: 0.9654
Validation Accuracy: 96.54%
Validation Loss: 0.1140
```

base model evaluation

Our CNN achieved 94.12% validation accuracy in classifying potato leaf diseases, but insights from the Confusion Matrix and Classification Report reveal variations in performance across classes.

We will leverage Ensembling using GoogleNet's Inception architecture. These steps aim to refine the model's ability to distinguish between healthy and diseased potato leaves.

---

## Deep CNN Model

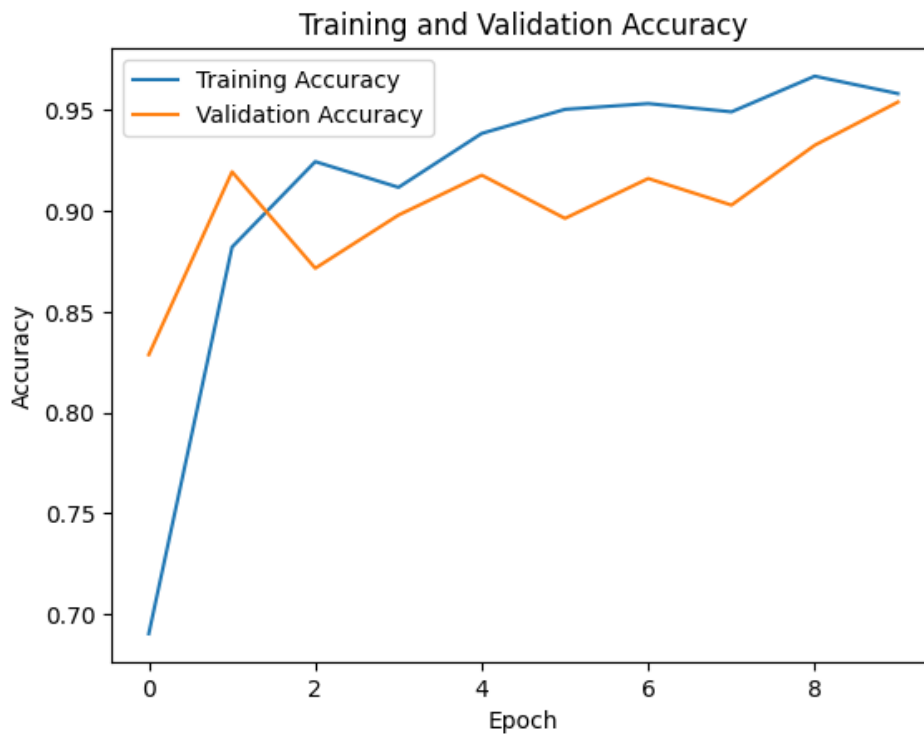
Our model consists of 11 layers. It starts with convolutional layers for feature extraction, followed by max-pooling layers to reduce spatial dimensions. Dropout layers are included for regularization. The architecture then transitions to dense layers for classification, with a total of 3 units in the output layer.

1. **Conv2D Layers:** These are convolutional layers, which apply filters to the input image to capture spatial features like edges, textures, and shapes. The output shape and number of filters increase with each layer, allowing the model to learn more complex features.
2. **MaxPooling2D Layers:** These layers reduce the spatial dimensions (height and width) of the input volume for the next convolutional layer. They help reduce computation and control overfitting by providing an abstracted form of the representation.
3. **Dropout Layers:** These are regularization layers that randomly set a fraction of input units to zero at each update during training, which helps prevent overfitting.
4. **Flatten Layer:** This layer flattens the input, converting it from a 2D matrix to a 1D vector. It's necessary before using fully connected layers.

5. **Dense Layers:** These are fully connected layers. The first Dense layer (with 32 units) serves as a hidden layer, and the final Dense layer (with 3 units) is likely the output layer, suggesting the model performs classification into three categories.
6. **Parameters:** The total number of trainable parameters is 185,667. These parameters are learned during the training process.

```
Epoch 1/10  
76/76 [=====] - 462s 5s/step - loss: 1.0797 - accuracy: 0.4137 - val_loss: 1.0464 - val_accuracy: 0.3576  
Epoch 2/10  
76/76 [=====] - 382s 5s/step - loss: 0.5198 - accuracy: 0.7048 - val_loss: 0.6156 - val_accuracy: 0.7222  
Epoch 3/10  
76/76 [=====] - 364s 5s/step - loss: 0.3686 - accuracy: 0.8943 - val_loss: 0.4879 - val_accuracy: 0.8177  
Epoch 4/10  
76/76 [=====] - 373s 5s/step - loss: 0.2399 - accuracy: 0.9834 - val_loss: 0.2604 - val_accuracy: 0.8882  
Epoch 5/10  
76/76 [=====] - 368s 5s/step - loss: 0.2644 - accuracy: 0.9383 - val_loss: 0.3564 - val_accuracy: 0.8688  
Epoch 6/10  
76/76 [=====] - 388s 5s/step - loss: 0.3886 - accuracy: 0.9243 - val_loss: 0.2835 - val_accuracy: 0.8854  
Epoch 7/10  
76/76 [=====] - 366s 5s/step - loss: 0.3887 - accuracy: 0.9348 - val_loss: 0.3511 - val_accuracy: 0.8212  
Epoch 8/10  
76/76 [=====] - 357s 5s/step - loss: 0.3612 - accuracy: 0.9387 - val_loss: 0.4873 - val_accuracy: 0.7378  
Epoch 9/10  
76/76 [=====] - 366s 5s/step - loss: 0.1717 - accuracy: 0.9330 - val_loss: 0.2888 - val_accuracy: 0.8882  
Epoch 10/10  
76/76 [=====] - 370s 5s/step - loss: 0.1333 - accuracy: 0.9470 - val_loss: 0.2524 - val_accuracy: 0.8872
```

1. **Loss:** This is a measure of how well the model is performing, with lower values indicating better performance. Your model's training loss decreased significantly from 1.0797 to 0.1333, which indicates that the model is learning effectively from the training data.
2. **Accuracy:** This measures the proportion of correctly classified instances. our model's training accuracy improved consistently, starting from 41.37% and reaching 94.70% by the end of the 10th epoch. This is a strong indication that your model is becoming more proficient at the classification task.
3. **Validation Loss and Accuracy:** These metrics are computed on a separate dataset that is not used for training, and they are critical for understanding how well your model generalizes to new data. our validation accuracy increased from 39.76% to 88.72%, which is a good sign. However, the validation loss decreased initially but then showed fluctuations, with an increase in some epochs (like epochs 5, 7, and 8).



## EVALUATION DEEP CNN

```

# evaluate the model
evaluation_results = deep_cn_model.evaluate(validation_generator)
print('Validation Accuracy: {:.2f}%'.format(evaluation_results[1] * 100))
print('Validation Loss: {:.4f}'.format(evaluation_results[0]))

10/18 [=====] - 28s 1s/step - loss: 0.2084 - accuracy: 0.9357
Validation accuracy: 93.57%
Validation loss: 0.2084

```

1. **validation Accuracy of 93.57%:** This is a high accuracy rate, indicating that our model is correctly classifying a large majority of the cases in the validation dataset. It's a strong indicator of

the model's effectiveness in understanding and categorizing the data it was tested on.

- 2. **Validation Loss of 0.1664:** This loss value is relatively low, which suggests that the model's predictions are, on average, quite close to the actual labels in the validation set.

## GoogleNet's Inception Model.

This method ensures efficient feature extraction with its inception modules, reducing parameters for computational efficiency. Its multi-pathway design captures diverse features, providing a broader receptive field and enabling effective representation learning.

By augmenting this base with a Global Average Pooling layer and a Dense layer employing softmax activation for classification, the model adapts its knowledge to the task of identifying potato leaf diseases.

An EarlyStopping callback is implemented to curb overfitting and ensure optimal generalization during the training process.

Model: "sequential\_1"

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 5, 5, 2048)	21802784
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dense_2 (Dense)	(None, 128)	262272
dense_3 (Dense)	(None, 3)	387

Total params: 22065443 (84.17 MB)  
Trainable params: 262659 (1.00 MB)  
Non-trainable params: 21802784 (83.17 MB)

Google Inception Layer

Our Google inception model is a "sequential\_1", neural network that incorporates the Inception V3 architecture as its base, followed by a global average pooling layer and two dense layers.

- 1. **Inception V3 (Functional):** This is a pre-built and pre-trained model, specifically the Inception V3 model, which is well-known for its performance in image classification tasks. It has 21,802,784 parameters, all of which are non-trainable in this specific configuration. This means the weights in this layer are



fixed during training, leveraging the knowledge it gained during pre-training on a large dataset like ImageNet.

2. **GlobalAveragePooling2D**: This layer performs global average pooling operations for spatial data. It doesn't have trainable parameters. Its purpose is to reduce the dimensionality of the feature maps from the Inception V3 layer, making the output more manageable and reducing the number of parameters needed in subsequent layers.
3. **Dense\_2**: This is a fully connected (dense) layer with 128 units of neurons. It has 262,272 trainable parameters.
4. **Dense\_3**: Another dense layer, but with 3 units or neurons, used for outputting predictions in a classification task with 3 classes. It has 387 trainable parameters.

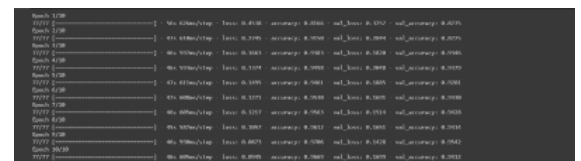
The total number of parameters in the model is 22,065,443, with 262,659 of these being trainable. The rest, 21,802,784 parameters, are part of the pre-trained Inception V3 model and are set as non-trainable in this configuration

---

## Train, Visualize Training Loss and Training Accuracy of Google Inception Model.

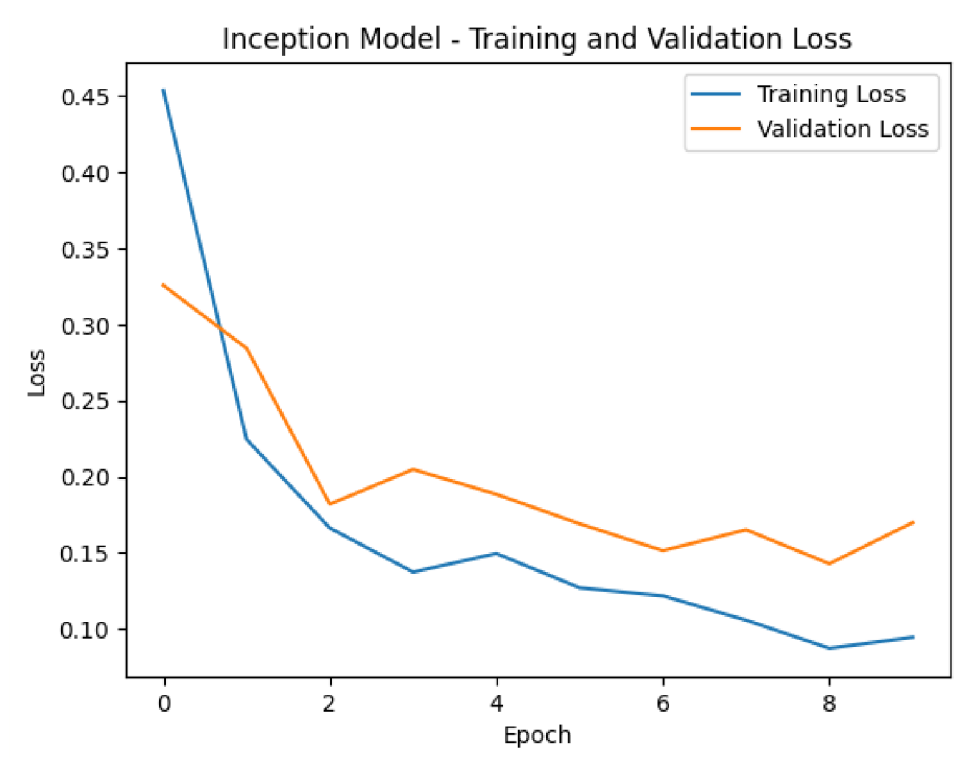
The log shows the progress of the Google inception model over 10 epochs using our dataset split into training and validation sets. the model shows a consistent

improvement in both training and validation accuracy over the epochs. The training loss decreases steadily, indicating that the model is learning effectively. The validation loss also decreases, but with some fluctuations, which is typical in training processes. The model demonstrates good generalization capabilities, as indicated by the high validation accuracy score

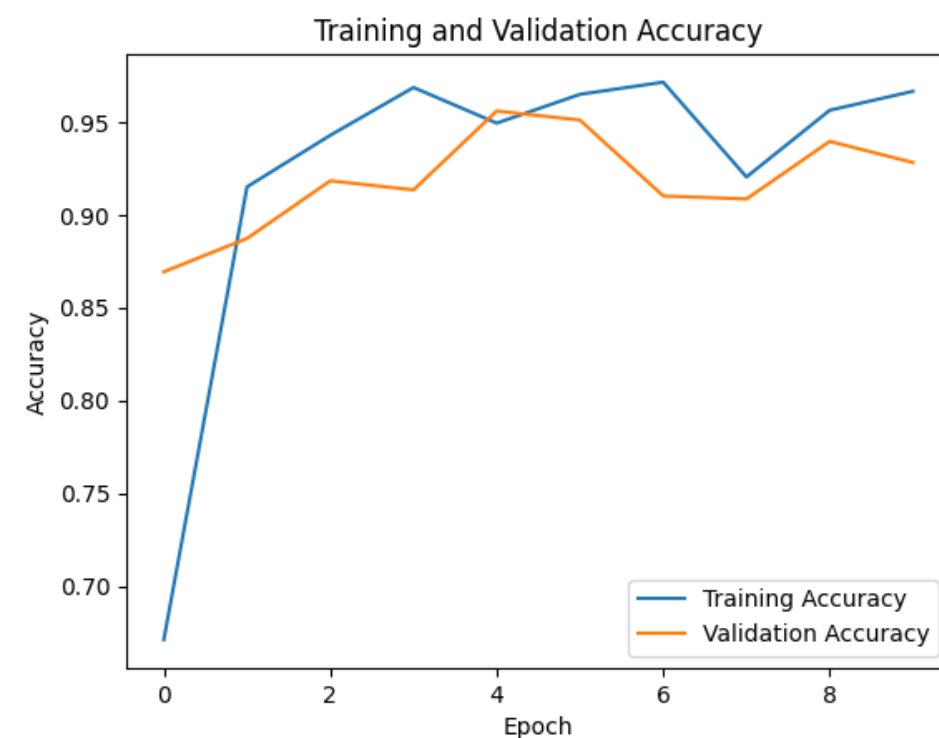


Epoch 1/10	1	64. 0.0000/1.0000	loss: 0.4036	accuracy: 0.4000	val_loss: 0.3510	val_accuracy: 0.6075
Epoch 2/10	1	64. 0.0000/1.0000	loss: 0.3076	accuracy: 0.5000	val_loss: 0.3000	val_accuracy: 0.6075
Epoch 3/10	1	64. 0.0000/1.0000	loss: 0.3043	accuracy: 0.5000	val_loss: 0.3018	val_accuracy: 0.5980
Epoch 4/10	1	64. 0.0000/1.0000	loss: 0.3104	accuracy: 0.4900	val_loss: 0.3008	val_accuracy: 0.5990
Epoch 5/10	1	64. 0.0000/1.0000	loss: 0.3105	accuracy: 0.5000	val_loss: 0.3005	val_accuracy: 0.5980
Epoch 6/10	1	64. 0.0000/1.0000	loss: 0.3171	accuracy: 0.4900	val_loss: 0.3005	val_accuracy: 0.5980
Epoch 7/10	1	64. 0.0000/1.0000	loss: 0.3171	accuracy: 0.5000	val_loss: 0.3014	val_accuracy: 0.5980
Epoch 8/10	1	64. 0.0000/1.0000	loss: 0.3051	accuracy: 0.5000	val_loss: 0.3005	val_accuracy: 0.5980
Epoch 9/10	1	64. 0.0000/1.0000	loss: 0.3071	accuracy: 0.5000	val_loss: 0.3018	val_accuracy: 0.5980
Epoch 10/10	1	64. 0.0000/1.0000	loss: 0.3008	accuracy: 0.5000	val_loss: 0.3008	val_accuracy: 0.5980

Epoch Log Google Inception



Google inception training vs validation loss



Google inception training vs validation Accuracy

```

Evaluate the Inception model

# Evaluate the model
evaluation_results_inception = inception_model.evaluate(validation_generator)

print("Validation accuracy: {:.2f}%".format(evaluation_results_inception[1] * 100))
print("Validation loss: {:.4f}".format(evaluation_results_inception[0]))

18/199 [=====] - 136s 64x64 - loss: 0.0864 - accuracy: 0.9357
Validation Accuracy: 93.57%
Validation Loss: 0.0864

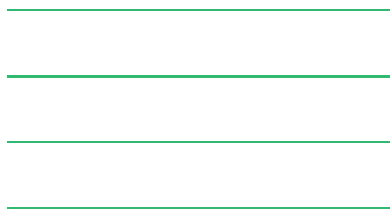
```

Inception model evaluation

## Evaluate the Google Inception Model

- **Validation Loss:** 0.26019
- **Validation Accuracy:** 88.80%

These metrics suggest that the google inception model has achieved a high level of accuracy in its predictions on the validation set, with over 92% of the predictions being correct. The loss value, which is a measure of how far off the predictions are from the actual values, is relatively low at 0.2019, further indicating good performance



## MODEL DEPLOYMENT

For the deployment of our models, we were using Google Cloud Platform (GCP) and Streamlit. we developed a Streamlit app that will act as the interface for our models. Used Python to develop the script.

We deployed the Streamlit app on a GCP service -App Engine . For App Engine, we created an app.yaml file to configure the deployment.

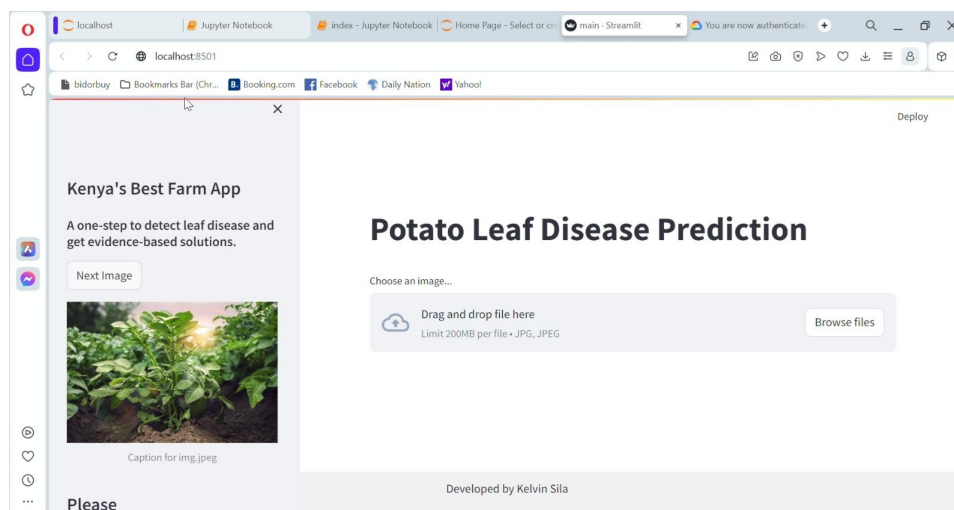
### Streamlit and GCP

We chose Streamlit for its user-friendly and rapid development capabilities, allowing us to create an intuitive web interface for our potato leaf disease detection model with minimal effort. Streamlit's Python integration and support for data visualization were crucial in seamlessly connecting with our existing machine-learning codebase. Additionally, we opted for Google Cloud Platform (GCP) for deployment due to its robust and scalable infrastructure. GCP offers reliable hosting options, efficient resource management, and integration with tools like Cloud Storage, facilitating seamless deployment and ensuring optimal performance for our potato leaf disease detection model in a cloud-based environment.

The Base CNN Model is selected for deployment due to various reasons

Rationale:

- 
- The base CNN model achieved the highest validation accuracy among the models, with an accuracy of 96.54%. This indicates that the model has demonstrated strong performance in classifying potato leaf diseases during training and validation.
- The base CNN architecture provides a solid foundation for image classification tasks. It includes convolutional layers for feature extraction, pooling layers for down-sampling, and fully connected layers for classification, making it well-suited for the task of identifying patterns in potato leaf images.
- The base CNN model is relatively simpler compared to the ensemble model. Streamlit apps benefit from simplicity to ensure efficient deployment and responsiveness, making the base CNN model a suitable choice for a straightforward deployment scenario.
- Deploying a model with high accuracy provides a positive user experience, instilling confidence in the reliability of the application for predicting potato leaf diseases.



## RECOMMENDATIONS AND CONCLUSIONS

Base CNN Model:

The Base CNN Model has delivered outstanding results by achieving a remarkably high validation accuracy of 96.54% in the intricate task of classifying potato leaf diseases. This significant accuracy underscores the model's proficiency in accurately identifying and categorizing diverse instances of potato leaf ailments. The achievement of such a high accuracy rate is indicative of the model's robust ability to detect abnormal patterns and features associated with different disease states in potato plants.

While the overall performance is promising, it is crucial to acknowledge and address the observed variations in accuracy across distinct disease classes. The model's ability to correctly identify specific diseases might be subject to fluctuations, necessitating a closer examination of its performance at a granular level. Understanding and mitigating these variations are pivotal steps in ensuring the model's reliability across the spectrum of potato leaf diseases.

#### Inception Model:

The Inception Model has demonstrated commendable performance, achieving a validation accuracy of 93.57% in classifying potato leaf diseases. This accuracy signifies the model's effectiveness in extracting distinctive features relevant to disease patterns, albeit slightly lower compared to the Base CNN Model. The marginal difference in accuracy suggests that the Inception Model could potentially offer an alternative and complementary approach to feature extraction.

To further enhance the Inception Model's accuracy, the recommendation is to embark on a fine-tuning process. Fine-tuning involves adjusting the model's hyperparameters or training on additional data to optimize its performance. This targeted optimization process aims to narrow the accuracy gap and potentially surpass the performance of the Base CNN Model. By understanding these challenges, the fine-tuning process can be more informed, addressing the root causes of misclassifications and bolstering the model's overall predictive capability.

#### Deep CNN Model:



The Deep CNN Model, while holding the promise of combining the strengths of the Base CNN and Inception models, unfortunately, demonstrates a notable drop in accuracy to 88.80%. This decline suggests challenges in effectively harnessing the individual models to achieve improved performance. The substantial decrease in accuracy raises concerns about the synergy between the constituent models within the ensemble, indicating that the combined predictive power needs to catch up to expectations.

The observed drop in accuracy prompts a critical examination of the ensemble's architecture and integration methodology. Understanding the factors contributing to this decline is crucial for rectifying and optimizing the ensemble's performance. The current combination strategy may need to leverage the strengths of the individual models more effectively, necessitating a careful reassessment of the ensemble design.

## Recommendations

In enhancing the potato leaf disease detection project, we strongly recommend the inclusion of a non-leaf class to address scenarios where the input may not be a diseased leaf. This addition will significantly improve the model's versatility, allowing it to distinguish non-leaf objects effectively. To implement this, it is crucial to design a diverse dataset representing various non-leaf instances and modify the model architecture to accommodate the new class. Fine-tuning the model on this augmented dataset will enable it to detect features associated with non-leaf objects. This strategic expansion will not only broaden the model's applicability but also contribute to a more comprehensive understanding of the broader context in which potato leaf disease detection operates.

Diversifying the negative class data is a critical step in refining the model's ability to distinguish features specific to potatoes, thereby minimizing the risk of misclassifying non-potato images. By incorporating a broader range of negative examples, the model gains exposure to a more comprehensive set of non-potato instances. This diversification facilitates a more nuanced understanding of what constitutes a non-potato object, reducing the chances of false positives and enhancing the model's overall robustness. The inclusion of varied non-potato images ensures that

the model can better discriminate between potato-related features and those unrelated to the domain, ultimately improving the accuracy and reliability of the potato leaf disease detection application.