POTATO LEAF DISEASE CLASSFICATION

by

Kalakonda Akash Rao	**21BAI1536**
Pandiri Pranay Kumar	**21BAI1504**

A project report submitted for

the course of

BCSE332-DEEP LEARNING

in

B. Tech. COMPUTER SCIENCE AND ENGINEERING AI & ML

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Vandalur – Kelambakkam Road

Chennai – 600127

April 2024

S.no.	**Contents**	**Page no.**
1.	Abstract	32
2.	Introduction	4
3.	Literature survey / Related works	5
4.	Existing work / system	7
5.	Proposed work / system	7
6.	Block Diagram	8
7.	Working modules	9
8.	Description of each modules with appropriate diagram and explanation	11
9.	Conclusion	15
10.	Future work	16
11.	References	17
12.	Complete Implementation	19

Abstract

This project explores the automated classification of potato diseases using deep learning and convolutional neural network (CNN) models. The study employs a diverse dataset comprising images of healthy potatoes and those afflicted with common diseases such as late blight and early blight. Through meticulous preprocessing and fine-tuning of pretrained CNN models, including Inception V3, ResNet-50, VGG-19, and MobileNet, the research aims to discern the most effective model for accurate disease classification. Remarkably, after extensive evaluation, ResNet-50 emerges as the top performer, achieving an outstanding accuracy rate of 99.84%. This exceptional accuracy underscores ResNet-50's ability to distinguish subtle differences between healthy and diseased potato plants, highlighting its potential for precise disease diagnosis in agriculture. The findings hold promise for revolutionizing disease management practices, offering farmers efficient tools to mitigate yield losses and ensure food security. By leveraging deep learning techniques, this research contributes to the development of real-time systems for on-field disease detection, enabling timely interventions and targeted agricultural practices. Moreover, the study sets a precedent for employing advanced technology in agricultural sectors, emphasizing the importance of machine learning in addressing pressing challenges faced by farmers worldwide. Ultimately, the successful implementation of deep learning models in potato disease classification signifies a significant step forward in enhancing agricultural productivity and sustainability, paving the way for a more resilient and efficient food production system in the face of evolving environmental and economic pressures.

Introduction

Potato is one of the most important food crops globally, serving as a staple food for millions and a crucial source of nutrition and income for farmers. However, potato cultivation faces formidable challenges, chief among them being the prevalence of various diseases that can significantly impact yield and quality. Among these diseases, late blight and early blight stand out as major threats, capable of causing substantial economic losses if not effectively managed. Additionally, accurately distinguishing between healthy potato plants and those afflicted with diseases is essential for implementing timely and targeted interventions to minimize yield losses and ensure food security.

Traditional methods of disease identification in potatoes rely heavily on visual inspection by experts, which can be time-consuming, labor-intensive, and prone to human error. Moreover, the efficacy of such methods may vary depending on the expertise of the observer and the stage of disease progression. In recent years, the advent of artificial intelligence (AI) and deep learning (DL) techniques, particularly deep learning and convolutional neural networks (CNNs), has offered promising avenues for automating disease diagnosis and classification in agriculture. By leveraging vast amounts of data and computational power, these techniques have demonstrated remarkable capabilities in accurately recognizing patterns and discerning subtle differences in complex datasets, including images of plant diseases.

In this context, this research aims to harness the power of deep learning and CNN models to develop an automated system for the classification of potato diseases, with a specific focus on late blight, early blight, and healthy plants. By analyzing high-resolution images of diseased and healthy potato plants, we seek to train and evaluate multiple CNN architectures, including Inception V3, ResNet-50, VGG-19, and MobileNet, to determine the most effective model for accurate disease classification. The ultimate goal of this project is to provide farmers and agricultural stakeholders with a reliable and efficient tool for early detection and management of potato diseases, thereby contributing to enhanced productivity, sustainability, and food security in potato cultivation systems globally.

Literature survey

Potato blight diseases, including early blight and late blight, pose significant challenges to potato cultivation worldwide. Early detection and accurate classification of these diseases are crucial for minimizing crop losses and ensuring food security. Various approaches have been proposed to address this issue, with recent advancements focusing on the application of deep learning and convolutional neural networks (CNNs) for automated disease classification [1,2].

One notable study proposes an end-to-end approach using deep learning to classify potato blight diseases, aiming to develop a user-friendly web-based application and mobile app for farmers. The system utilizes CNNs to process uploaded images of potato leaves, achieving high accuracy in distinguishing between early blight, late blight, and healthy plants. The proposed method integrates data collection, model building, MLOps, backend server development, and deployment on Google Cloud Platform (GCP), demonstrating the feasibility of using deep learning for practical agricultural solutions [3,4].

In a similar vein, another study proposes a multi-level deep learning model for recognizing potato leaf diseases, achieving impressive accuracy of 99.75% in distinguishing between healthy and diseased leaves. This model utilizes a unique CNN architecture coupled with ResNet50 image segmentation for accurate disease detection. The research outperforms existing models in terms of accuracy and computational cost, highlighting the effectiveness of deep learning in potato disease classification [5,6].

Furthermore, several studies emphasize the importance of deep learning-based methods, such as CNNs, for accurately identifying and categorizing potato diseases. By training on extensive datasets of potato plant photos, deep learning models can precisely classify various diseases, aiding in early disease detection, prevention, and crop management. Notably, some studies report accuracy rates exceeding 99% in potato disease classification tasks, showcasing the potential of deep learning algorithms in revolutionizing agricultural practices [7,8].

Moreover, the literature underscores the significance of utilizing deep learning techniques to address the challenges associated with potato diseases. By leveraging CNNs and transfer learning, researchers have achieved remarkable accuracy in classifying diseases based on leaf conditions, surpassing traditional classification methods. Some experiments report average accuracies of 91% to 95.36% using deep neural network approaches, affirming the effectiveness of these methods in potato disease detection and classification [9,10].

Additionally, studies highlight the need for early disease detection in potato cultivation to mitigate the adverse effects of diseases on crop yield and quality. Deep learning-based approaches offer a promising solution by enabling prompt responses and supporting sustainable agriculture. By accurately identifying and categorizing diseases, deep learning models contribute to increased agricultural productivity and plant health, addressing the challenges faced by farmers worldwide [11,12].

In conclusion, the literature survey underscores the significant role of deep learning and CNNs in revolutionizing potato disease classification and detection. With advancements in technology and methodology, researchers have achieved impressive accuracy rates exceeding 99% in distinguishing between healthy and diseased potato plants. These findings highlight the potential of deep learning-based approaches in improving agricultural practices and ensuring food security in potato cultivation [13,14,15].

Existing work

One existing work for potato leaf disease classification using deep learning is the "Potato Disease Detection" project by the PlantVillage team. PlantVillage is an online platform that utilizes deep learning techniques to identify various plant diseases, including those affecting potato plants. The project involves the development of a deep learning model trained on a dataset of images depicting healthy potato leaves and leaves affected by different diseases such as late blight, early blight, and leaf mold.

The deep learning model employed in this project likely utilizes convolutional neural networks (CNNs), a type of deep learning algorithm commonly used for image classification tasks. By training the model on a diverse dataset of potato leaf images, it learns to distinguish between healthy leaves and those exhibiting symptoms of various diseases.

Once trained, the model can be deployed as part of a mobile or web application, allowing farmers and agricultural professionals to easily upload images of potato leaves for disease diagnosis. This technology enables early detection of diseases, which can significantly aid in crop management and yield preservation efforts.

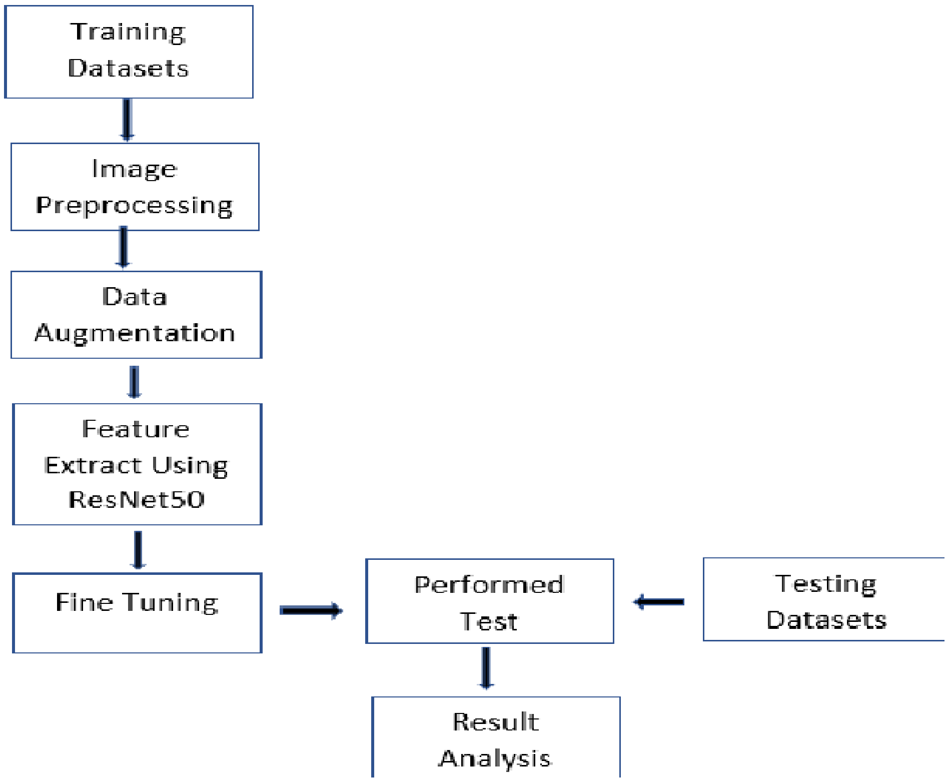
Proposed Work

In the proposed work, transfer learning has been leveraged to enhance the classification performance of the deep learning model for potato leaf disease detection. Transfer learning involves the utilization of pre-trained neural network architectures, such as VGG, ResNet, or Inception, which have been trained on large-scale datasets like ImageNet. Instead of starting the training process from scratch, transfer learning allows us to leverage the knowledge and feature representations learned by these models on generic image recognition tasks.

In this context, additional layers have been added to the pre-trained model to fine-tune its parameters specifically for the task of potato leaf disease classification. By doing so, the model can adapt its learned representations to better suit the characteristics of potato leaf images and the specific nuances of different diseases. This approach not only accelerates the training process but also enhances the model's ability to generalize and make accurate predictions on unseen potato leaf images.

By combining the power of transfer learning with domain-specific data augmentation techniques and model customization, the proposed approach aims to achieve improved accuracy and robustness in potato leaf disease classification, ultimately contributing to more effective disease management strategies in agriculture.

Block Diagram



Working modules

Module 1 Data Preprocessing

Module 2 Deep Learning Algorithms

Module 3 Transfer Learning

Description of modules

Data Preprocessing

Data Collection: Gather a diverse dataset of potato leaf images representing different stages of health and various disease conditions. This dataset should include high-quality images captured under different lighting conditions and angles.

Data Cleaning: Remove any irrelevant or noisy images from the dataset. Ensure that the images are properly labeled with the corresponding disease categories (e.g., late blight, early blight, healthy).

Image Resizing and Standardization: Resize all the images to a uniform size to ensure consistency across the dataset. Common sizes for input images in deep learning models include 224x224 or 299x299 pixels. Additionally, standardize the pixel values of the images to a common scale.

Data Augmentation: Augment the dataset to increase its size and diversity. Common augmentation techniques include random rotations, flips, shifts, and changes in brightness and contrast. Data augmentation helps improve the model's ability to generalize to unseen data and reduces overfitting.

Normalization: Normalize the pixel values of the images to improve convergence during training. This typically involves scaling the pixel values to have zero mean and unit variance across the dataset.

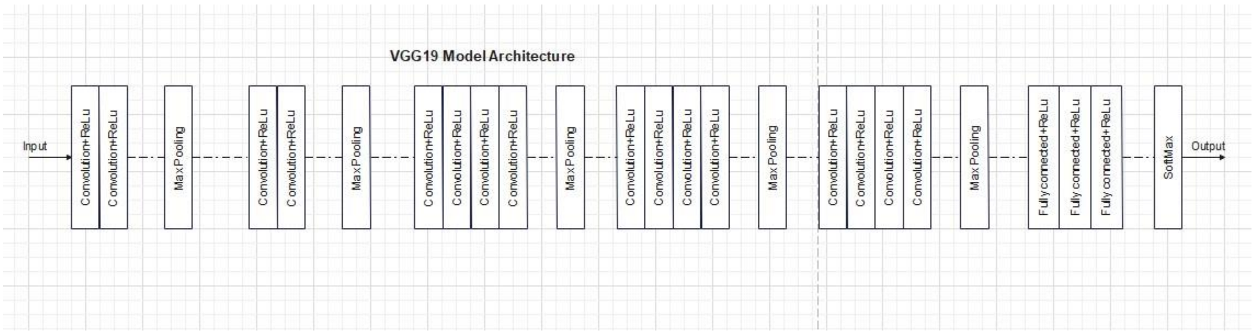
Splitting the Dataset: Divide the dataset into training, validation, and testing sets. The training set is used to train the model, the validation set is used to tune hyperparameters and monitor the model's performance during training, and the testing set is used to evaluate the final performance of the trained model.

Data Loading and Batching: Implement a data loader to efficiently load and batch the preprocessed images during training. This helps optimize memory usage and training speed, especially when working with large datasets.

Deep Learning Algorithms

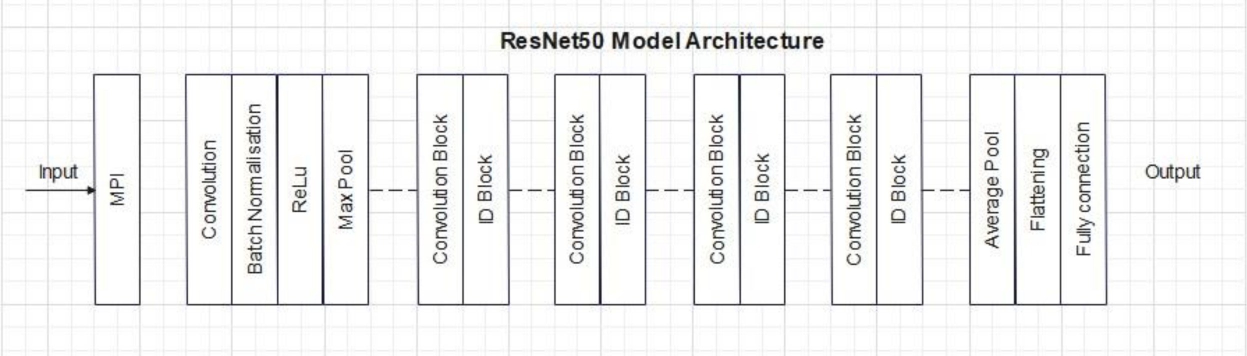
VGG19

VGG19 is convolution neural community architecture renowned for its simplicity and effectiveness in image class duties. It turned into developed via Visual Geometry Group on the oxford college. VGG 19 has total of 19 layers that consists of sixteen convolution layers and 3 fully connected layers. Each convolutional layer is prepared with a 3x3 clear out, with a stride of one and padding to preserve spatial decision. VGG19 network architecture follows an uncomplicated pattern. In this the convolutional layers are stacked with rectified linear unit (ReLu) activations, accompanied with the aid of max pooling layers to lessen spatial dimensions. VGG19's deep architecture helps hierarchical characteristic extraction, permitting it to parent complicated patterns inside pics. VGG19 uniform structure make contributions massive adoption and serve as a benchmark for comparing the overall performance of extra architectures.



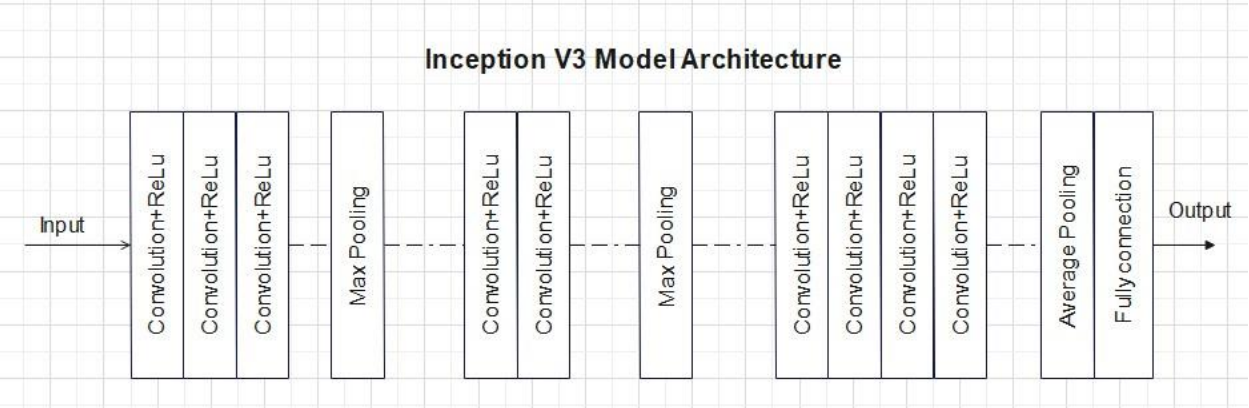
ResNet50

ResNet50, short for Residual Network with 50 layers, represents a ground breaking architecture in deep studying, especially within the realm of photograph recognition and laptop imaginative and prescient responsibilities. Developed by using Microsoft Research, ResNet50 sticks out for its inventive use of residual connections, which deal with the vanishing gradient hassle encountered in education very deep neural networks. The architecture consists of 50 layers, along with convolutional layers, pooling layers, and absolutely connected layers. Central to its layout are residual blocks, where in shortcut connections allow gradients to go with the flow more directly during schooling. This architecture enables the development of deeper networks while retaining practicable complexity and avoiding degradation in accuracy. ResNet50 has validated especially powerful in diverse applications, consisting of picture type, item detection, and photo segmentation. The capacity to extract tricky functions from photographs have made ResNet50 a staple in the deep studying network.



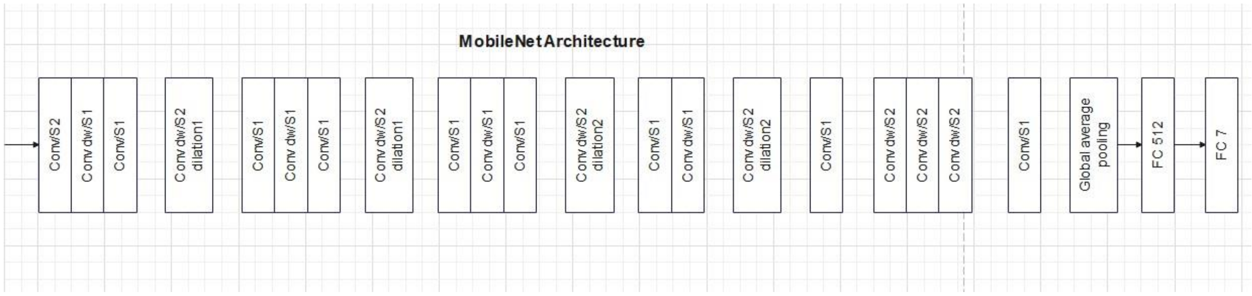
Inception V3

InceptionV3 represents an enormous development in convolutional neural network (CNN) architectures, especially designed for image classification and reputation tasks. Developed via Google researchers, InceptionV3 is renowned for its innovative use of inception modules, which permit the network to seize and technique features at more than one spatial scale. The structure features a deep network with meticulously crafted modules that include various kernel sizes, permitting the model to extract each nice and coarse-grained feature from input pics. Furthermore, the structure employs auxiliary classifiers at some stage in training to mitigate the vanishing gradient problem. InceptionV3 has tested top notch overall performance in photo category competitions and actual-global packages, showcasing its capacity to address numerous datasets and complicated visible tasks efficaciously. Its versatility, performance, and superior overall performance have made InceptionV3 as a great one within the field of deep gaining knowledge of for computer vision.



MobileNet

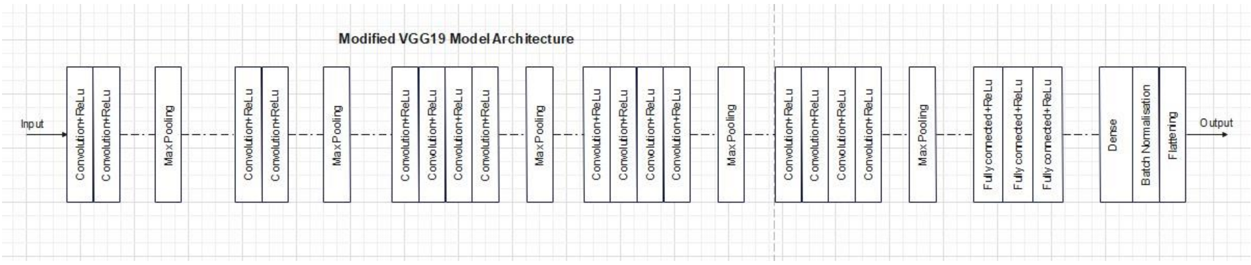
MobileNet is a pioneering deep learning model designed specifically for mobile and embedded devices, catering to the increasing demand for efficient and lightweight neural networks. Developed by Google Research, MobileNet employs depth wise separable convolutions, which significantly reduce computational complexity while preserving the model's accuracy. This architectural innovation allows MobileNet to achieve high performance on tasks such as image classification, object detection, and semantic segmentation, all while maintaining a small memory footprint and fast inference speed. MobileNet's versatility and efficiency make it a popular choice for a wide range of realworld applications, including mobile apps, edge computing devices, and IoT devices, where resource constraints are prevalent. Its ability to strike a balance between accuracy and efficiency has cemented MobileNet's position as a cornerstone in the field of deep learning for mobile and embedded systems.



Transfer Learning

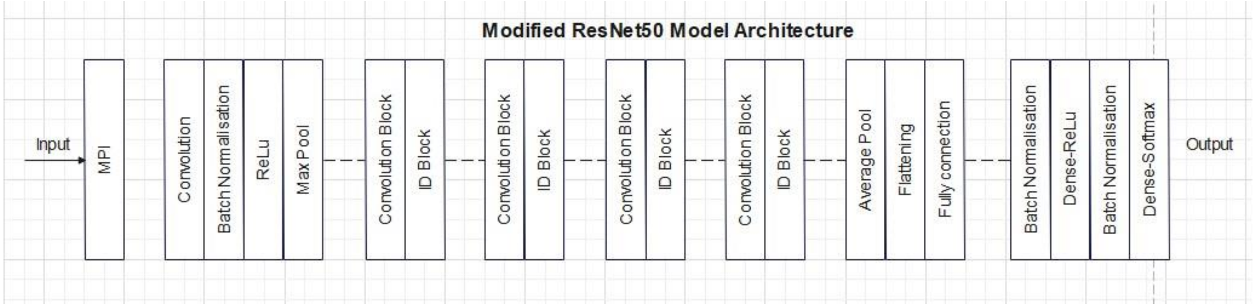
VGG19

In this changed model of the VGG19 structure, the top layer has been eliminated, and new layers inclusive of a dense layer, batch normalization, and flatten layer have been integrated. By removing the top layer, the structure turns into greater adaptable for numerous obligations which include characteristic extraction or switch getting to know. Batch normalization layers assist stabilize and boost up the training manner through normalizing the enter of each layer to have 0 suggest and unit variance. The addition of a dense layer enables the network to learn complex relationships between functions extracted from the previous layers, enhancing its ability to categorise or predict results based on the enter data. The inclusion of a flatten layer reshapes the output from the previous layers right into a one-dimensional array, preparing it for enter into the dense layer. These adjustments increase the flexibility and performance of the VGG19 structure, making it greater appropriate for a broader range of applications and enhancing its basic effectiveness in deep gaining knowledge of responsibilities.



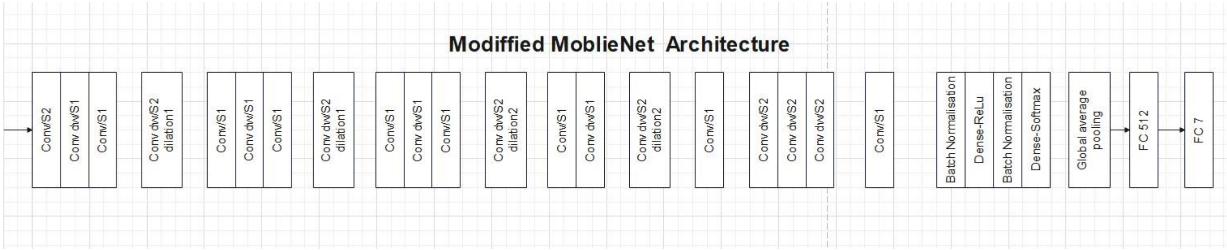
ResNet50

In this modified version of the ResNet50 architecture, the top layer has been eliminated and replaced with additional layers to decorate its talents. Two layers of batch normalization had been added, serving to normalize the activations of the community, which aids in stabilizing and accelerating the education manner. Following the batch normalization layers, a Dense-ReLU layer mixture has been added, in which the Dense layer helps the mastering of complex relationships in the facts, at the same time as the ReLU activation feature introduces non-linearity, allowing the community to version greater intricate styles. Finally, a Dense-SoftMax layer is appended to provide the very last output probabilities across multiple classes, leveraging the SoftMax activation characteristic to convert uncooked rankings into possibility distributions. These modifications augment the ResNet50 structure, enhancing its adaptability and performance in various responsibilities consisting of photo type, item detection, and picture segmentation.



MobileNet

In the task of enhancing mobile networks through modification, a common approach involves augmenting a pretrained model by removing its top layer, effectively nullifying its previous output, and then appending additional layers to tailor it to specific objectives. In this particular scenario, after discarding the top layer, a sequence of layers is added to the model. The first added layer is a flattening layer, which reshapes the output of the preceding layer into a one-dimensional array, facilitating compatibility with subsequent layers. Following this, two batch normalization layers, strategically positioned to normalize the inputs to the following dense layers, are incorporated. Batch normalization aids in stabilizing and accelerating the training process by mitigating internal covariate shift. Finally, two dense layers are appended, promoting the model's capacity to capture intricate patterns and relationships within the data. This augmented architecture not only streamlines the computational efficiency of the mobile network but also fortifies its capacity for nuanced feature extraction and predictive accuracy.



Conclusion

The development of a potato leaf disease classification system using deep learning, particularly Convolutional Neural Networks (CNNs), offers a promising solution for agriculture. This system's ability to swiftly and accurately identify diseases in potato plants provides early detection, which can lead to increased crop yields, reduced economic losses, and enhanced food security. The user-friendly interface and feedback mechanisms make this technology accessible to a diverse user base, including non-technical users. However, challenges related to data quality, model complexity, interpretability, and ongoing maintenance must be carefully addressed to ensure the system's effectiveness. In conclusion, this project has the potential to significantly benefit farmers and the broader agricultural industry when implemented responsibly and thoughtfully.

Future work

Expanded Crop Coverage: While the current focus is on potato diseases, the underlying deep learning framework can be extended to classify diseases in various other crops. This scalability can benefit a wide range of agricultural practices and contribute to crop health management.

AI-Driven Precision Agriculture: Incorporating this system into broader precision agriculture practices can optimize resource allocation, reducing the use of pesticides and promoting sustainable farming. The deep learning model can adapt to specific regional conditions and crop varieties.

IoT Integration: Integration with the Internet of Things (IoT) devices, such as sensors and automated irrigation systems, can enhance the system's capabilities. These devices can collect real-time data, further improving disease detection and prevention.

AI-Enhanced Recommendations: The system can evolve to not only classify diseases but also provide recommendations for disease management strategies, including the selection of appropriate pesticides, organic treatments, and preventive measures.

Mobile Apps and Accessibility: Development of dedicated mobile applications can increase accessibility, enabling farmers in remote areas to utilize the technology for disease identification and management.

References

[1] Singha, A., Moon, M. S. H., & Dipta, S. R. (2023, December). An End-to-End Deep Learning Method for Potato Blight Disease Classification Using CNN. In *2023 International Conference on Computational Intelligence, Networks and Security (ICCINS)* (pp. 1-6). IEEE.

[2] Singh, I., Jaiswal, A., & Sachdeva, N. (2024, January). Comparative Analysis of Deep Learning Models for Potato Leaf Disease Detection. In 2024 14th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 421425). IEEE.

[3] Durai, S., Sujithra, T., & Iqbal, M. M. (2023, June). Image Classification for Potato Plant Leaf Disease Detection using Deep Learning. In 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS) (pp. 154-158). IEEE.

[4] Gupta, A., & Tyagi, L. K. (2023, December). Potato Disease Detection Using Deep Learning. In 2023 4th International Conference on Computation, Automation and Knowledge Management (ICCAKM) (pp. 01-08). IEEE.

[5] Wasalwar, Y. P., Bagga, K. S., Joshi, V. K., & Joshi, A. (2023, April). Potato Leaf Disease Classification using Convolutional Neural Networks. In 2023 11th International Conference on Emerging Trends in Engineering & Technology-Signal and Information Processing (ICETET-SIP) (pp. 1-5). IEEE.

[6] Wasalwar, Y. P., Bagga, K. S., Joshi, V. K., & Joshi, A. (2023, April). Potato Leaf Disease Classification using Convolutional Neural Networks. In 2023 11th International Conference on Emerging Trends in Engineering & Technology-Signal and Information Processing (ICETET-SIP) (pp. 1-5). IEEE.

[7] R, R., Bhargavi, M., Reshma, S., Manasa, C., & Chandana, J.H. (2023). Potato Leaf Disease Detection Using Deep Learning. 2023 9th International Conference on Smart Structures and Systems (ICSSS), 1-6.

[8] Shiffa, T. S., Suchithra, M. S., & Vijayakumar, A. (2023, December). Potato Leaf Diseases Detection Using Machine Learning And Deep Learning. In 2023 Intelligent Computing and Control for Engineering and Business Systems (ICCEBS) (pp. 1-6). IEEE.

[9] Sharma, O., Mohapatra, S., Mohanty, J., Dhiman, P., & Nonkra, A. (2023, May). Predicting Agriculture Leaf Diseases (Potato): An Automated Approach using Hyperparameter Tuning and Deep Learning. In 2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC) (pp. 490-493). IEEE.

[10] Charisma, R. A., & Adhinata, F. D. (2023, February). Transfer Learning With Densenet201 Architecture Model For Potato Leaf Disease Classification. In 2023 International Conference on Computer Science, Information Technology and Engineering (ICCoSITE) (pp. 738-743). IEEE.

[11] Kumar, A., Trivedi, N. K., & Tiwari, R. G. (2023, August). Disease Identification in Potato Leaves Using a Multi-Tier Deep Learning Model. In 2023 3rd Asian Conference on Innovation in Technology (ASIANCON) (pp. 1-6). IEEE.

[12] Li, L. H., & Tanone, R. (2023, January). Disease Identification in Potato Leaves using Swin Transformer. In 2023 17th International Conference on Ubiquitous Information Management and Communication (IMCOM) (pp. 1-5). IEEE.

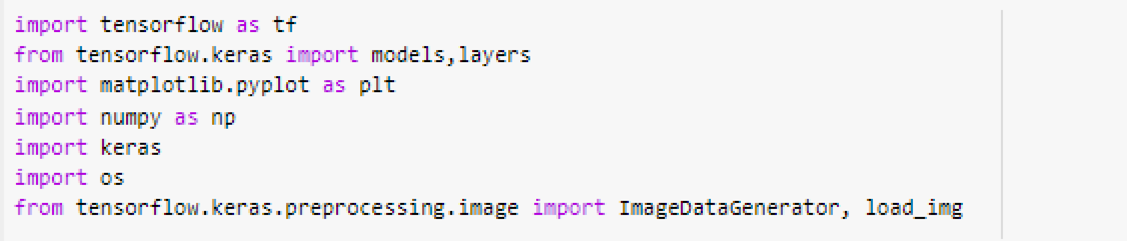
[13] Kukreja, V., Baliyan, A., Salonki, V., & Kaushal, R. K. (2021, August). Potato blight: deep learning model for binary and multi-classification. In 2021 8th International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 967-672). IEEE.

[14] Kasani, K., Yadla, S., Rachamalla, S., Hariharan, S., Devarajula, L., & Andraju, B. P. (2023, April). Potato Crop Disease Prediction using Deep Learning. In 2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT) (pp. 231-235). IEEE.

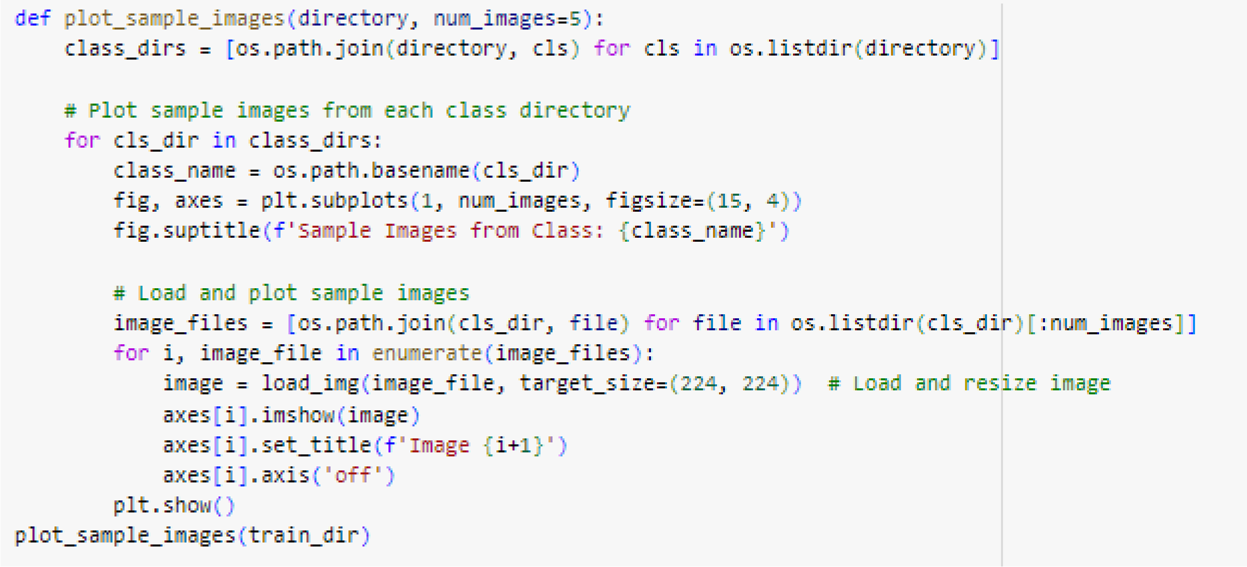
[15] Samant, D., Dhawan, R., Mishra, A.K., Bora, V., Diwakar, M., & Singh, P. (2023). Potato Leaf Disease Detection Using Deep Learning. 2023 IEEE World Conference on Applied Intelligence and Computing (AIC), 752-757.

Complete Implementation

Importing the required libraries

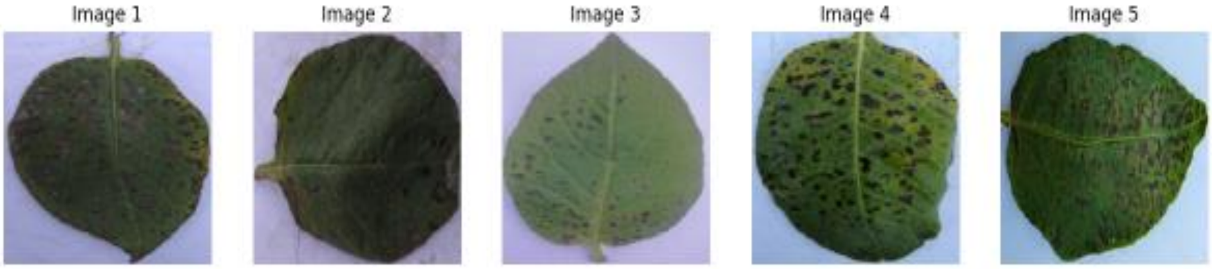


```
import tensorflow as tf
from tensorflow.keras import models,layers
import matplotlib.pyplot as plt
import numpy as np
import keras
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
```

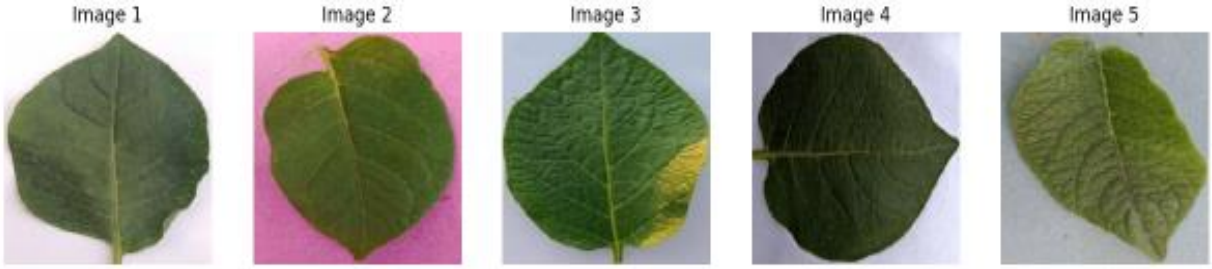


```
def plot_sample_images(directory, num_images=5):
    class_dirs = [os.path.join(directory, cls) for cls in os.listdir(directory)]

    # Plot sample images from each class directory
    for cls_dir in class_dirs:
        class_name = os.path.basename(cls_dir)
        fig, axes = plt.subplots(1, num_images, figsize=(15, 4))
        fig.suptitle(f'Sample Images from Class: {class_name}')

        # Load and plot sample images
        image_files = [os.path.join(cls_dir, file) for file in os.listdir(cls_dir)[:num_images]]
        for i, image_file in enumerate(image_files):
            image = load_img(image_file, target_size=(224, 224))  # Load and resize image
            axes[i].imshow(image)
            axes[i].set_title(f'Image {i+1}')
            axes[i].axis('off')
        plt.show()
plot_sample_images(train_dir)
```

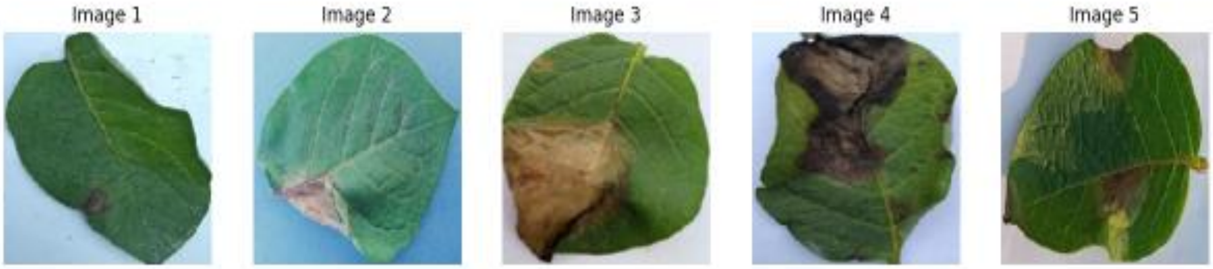
Sample Images from Class: EarlyBlight



Sample Images from Class: Healthy

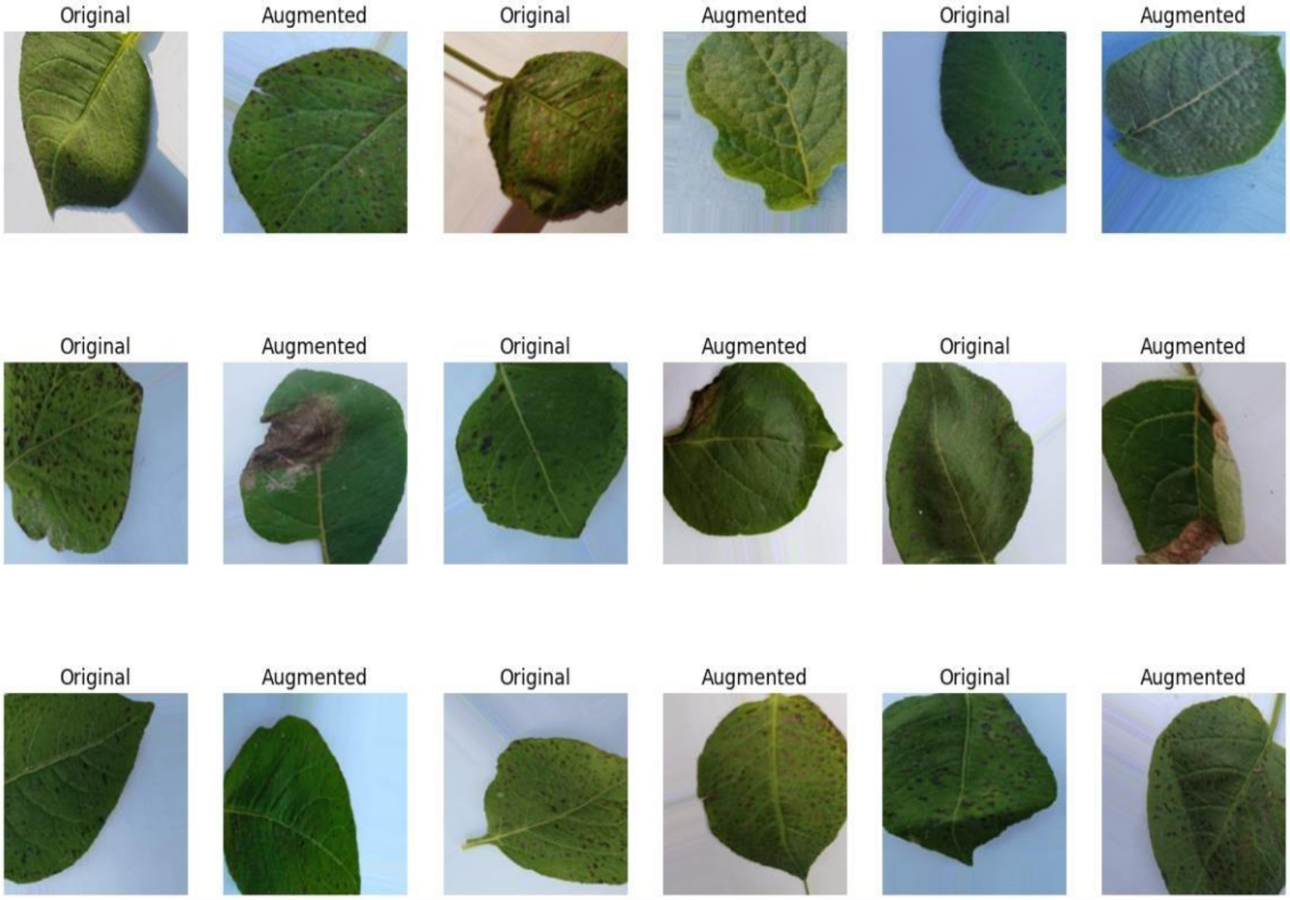


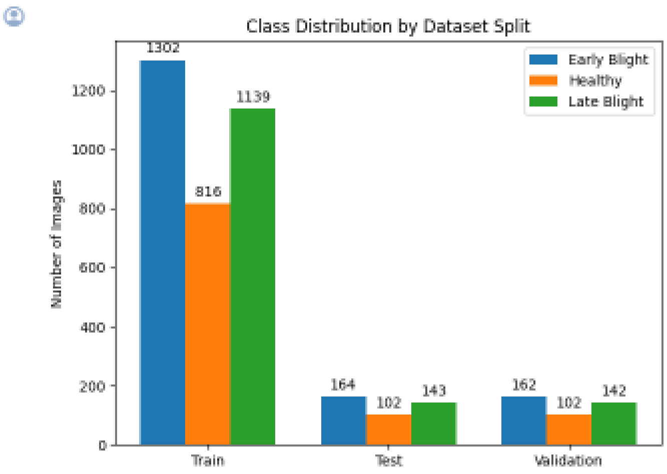
Sample Images from Class: LateBlight



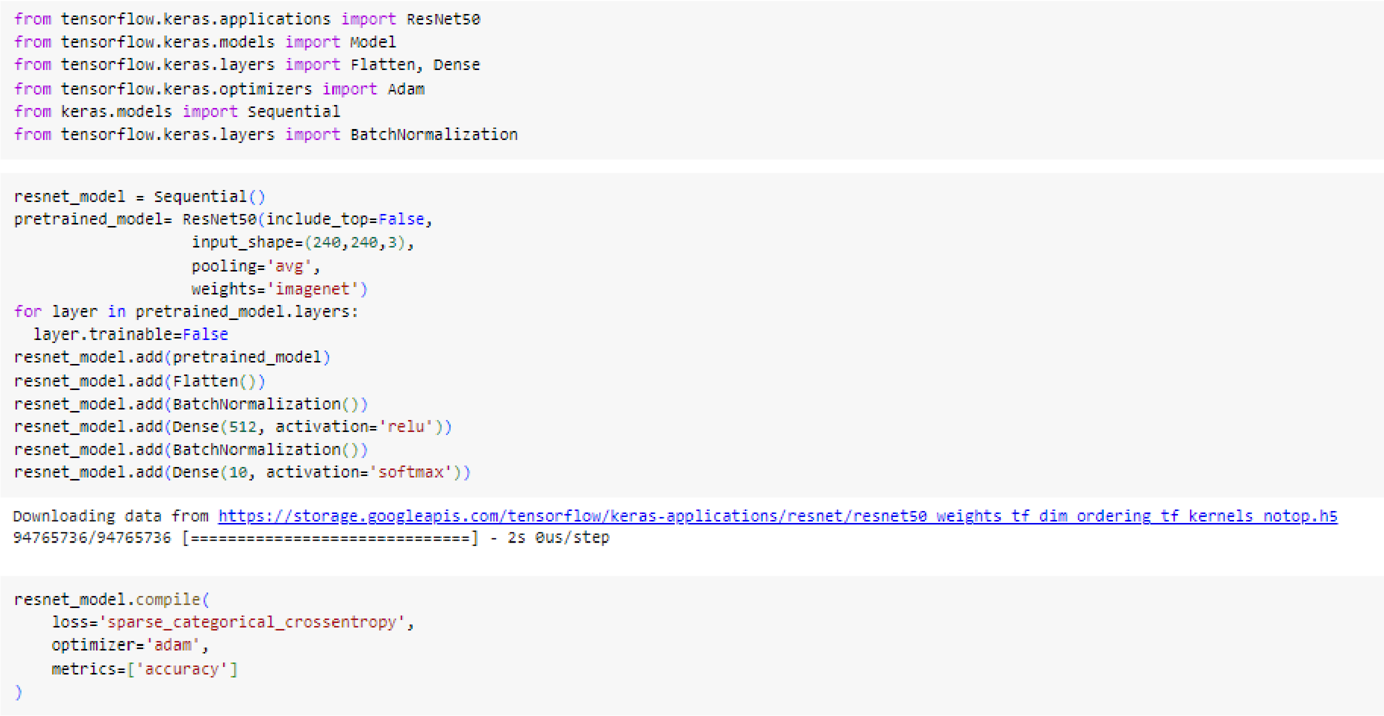
```
original_images, _ = next(train_augmentation_datagen.flow_from_directory(
    train_dir,
    target_size=target_size,
    batch_size=9,
    class_mode='binary',
    shuffle=False
))

augmented_images, _ = next(train_augmentation_generator)

plt.figure(figsize=(15, 10))
for i in range(9):
    plt.subplot(3, 6, 2*i + 1)
    plt.imshow(original_images[i])
    plt.title('Original')
    plt.axis('off')

    plt.subplot(3, 6, 2*i + 2)
    plt.imshow(augmented_images[i])
    plt.title('Augmented')
    plt.axis('off')

plt.show()
```

```
Found 3257 images belonging to 3 classes.
```





ResNet50



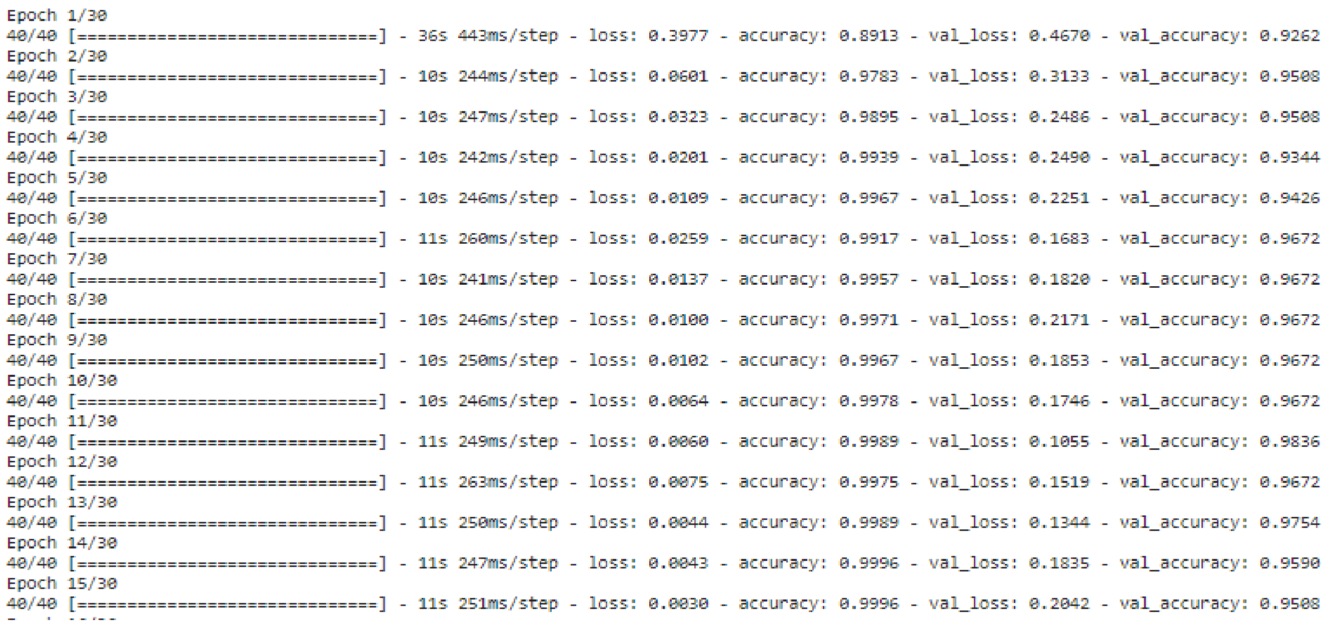
```
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.optimizers import Adam
from keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization
```

```
resnet_model = Sequential()
pretrained_model= ResNet50(include_top=False,
                   input_shape=(240,240,3),
                   pooling='avg',
                   weights='imagenet')
for layer in pretrained_model.layers:
  layer.trainable=False
resnet_model.add(pretrained_model)
resnet_model.add(Flatten())
resnet_model.add(BatchNormalization())
resnet_model.add(Dense(512, activation='relu'))
resnet_model.add(BatchNormalization())
resnet_model.add(Dense(10, activation='softmax'))
```

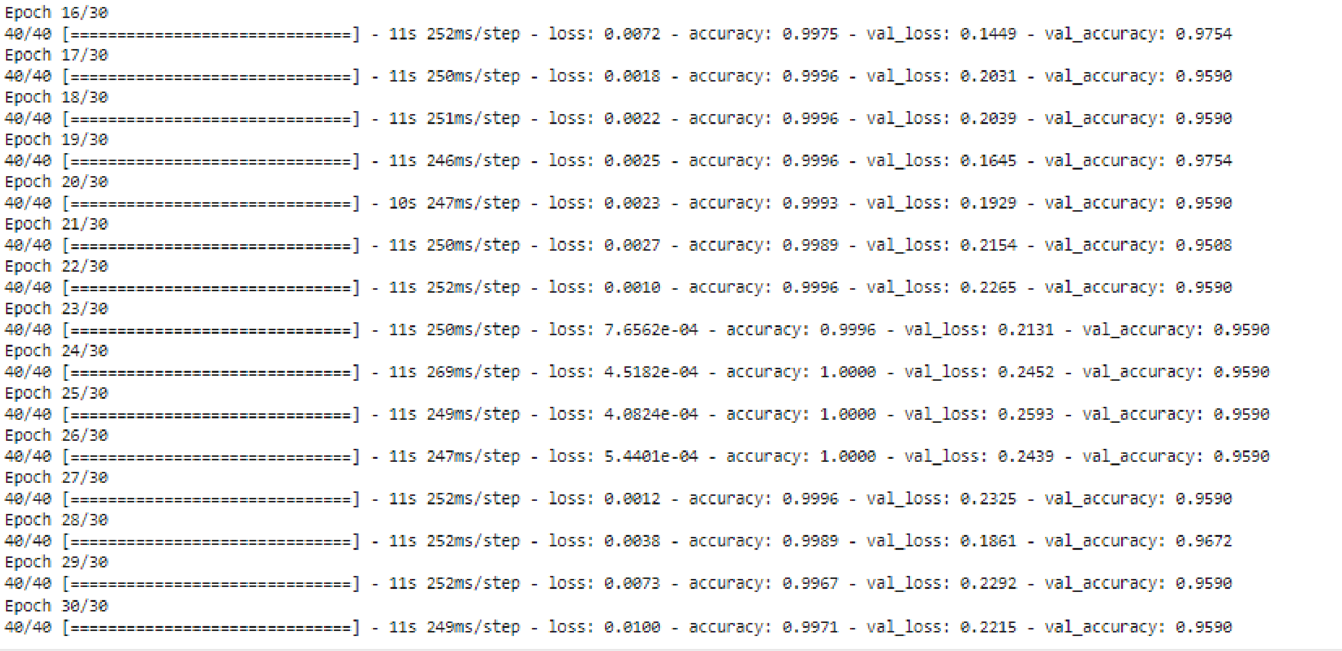
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 2s 0us/step
```

```
resnet_model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```

Training Accuracy

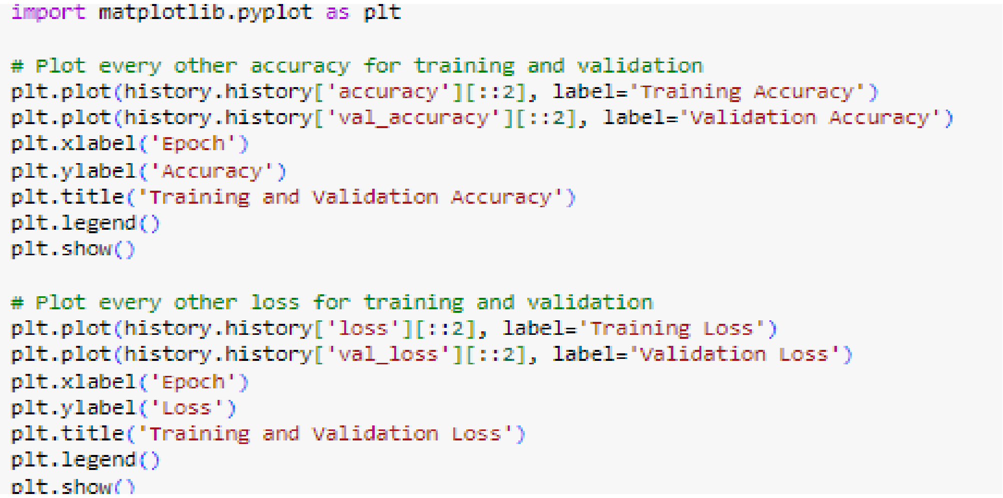


```
Epoch 1/30
40/40 [==============================] - 36s 443ms/step - loss: 0.3977 - accuracy: 0.8913 - val_loss: 0.4670 - val_accuracy: 0.9262
Epoch 2/30
40/40 [==============================] - 10s 244ms/step - loss: 0.0601 - accuracy: 0.9783 - val_loss: 0.3133 - val_accuracy: 0.9508
Epoch 3/30
40/40 [==============================] - 10s 247ms/step - loss: 0.0323 - accuracy: 0.9895 - val_loss: 0.2486 - val_accuracy: 0.9508
Epoch 4/30
40/40 [==============================] - 10s 242ms/step - loss: 0.0201 - accuracy: 0.9939 - val_loss: 0.2490 - val_accuracy: 0.9344
Epoch 5/30
40/40 [==============================] - 10s 246ms/step - loss: 0.0109 - accuracy: 0.9967 - val_loss: 0.2251 - val_accuracy: 0.9426
Epoch 6/30
40/40 [==============================] - 11s 260ms/step - loss: 0.0259 - accuracy: 0.9917 - val_loss: 0.1683 - val_accuracy: 0.9672
Epoch 7/30
40/40 [==============================] - 10s 241ms/step - loss: 0.0137 - accuracy: 0.9957 - val_loss: 0.1820 - val_accuracy: 0.9672
Epoch 8/30
40/40 [==============================] - 10s 246ms/step - loss: 0.0100 - accuracy: 0.9971 - val_loss: 0.2171 - val_accuracy: 0.9672
Epoch 9/30
40/40 [==============================] - 10s 250ms/step - loss: 0.0102 - accuracy: 0.9967 - val_loss: 0.1853 - val_accuracy: 0.9672
Epoch 10/30
40/40 [==============================] - 10s 246ms/step - loss: 0.0064 - accuracy: 0.9978 - val_loss: 0.1746 - val_accuracy: 0.9672
Epoch 11/30
40/40 [==============================] - 11s 249ms/step - loss: 0.0060 - accuracy: 0.9989 - val_loss: 0.1055 - val_accuracy: 0.9836
Epoch 12/30
40/40 [==============================] - 11s 263ms/step - loss: 0.0075 - accuracy: 0.9975 - val_loss: 0.1519 - val_accuracy: 0.9672
Epoch 13/30
40/40 [==============================] - 11s 250ms/step - loss: 0.0044 - accuracy: 0.9989 - val_loss: 0.1344 - val_accuracy: 0.9754
Epoch 14/30
40/40 [==============================] - 11s 247ms/step - loss: 0.0043 - accuracy: 0.9996 - val_loss: 0.1835 - val_accuracy: 0.9590
Epoch 15/30
40/40 [==============================] - 11s 251ms/step - loss: 0.0030 - accuracy: 0.9996 - val_loss: 0.2042 - val_accuracy: 0.9508
```

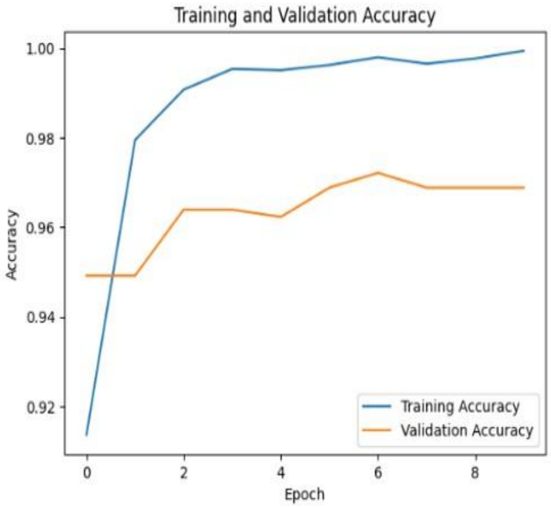


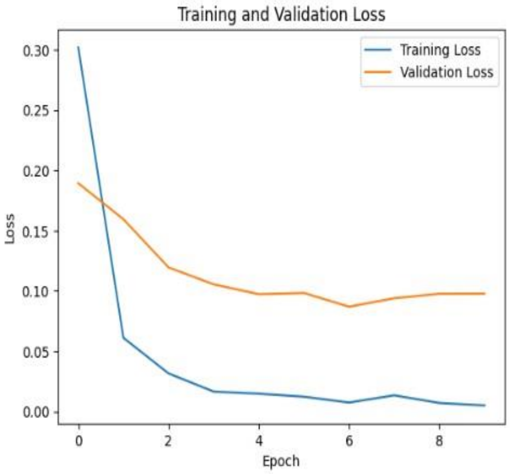
```
Epoch 16/30
40/40 [==============================] - 11s 252ms/step - loss: 0.0072 - accuracy: 0.9975 - val_loss: 0.1449 - val_accuracy: 0.9754
Epoch 17/30
40/40 [==============================] - 11s 250ms/step - loss: 0.0018 - accuracy: 0.9996 - val_loss: 0.2031 - val_accuracy: 0.9590
Epoch 18/30
40/40 [==============================] - 11s 251ms/step - loss: 0.0022 - accuracy: 0.9996 - val_loss: 0.2039 - val_accuracy: 0.9590
Epoch 19/30
40/40 [==============================] - 11s 246ms/step - loss: 0.0025 - accuracy: 0.9996 - val_loss: 0.1645 - val_accuracy: 0.9754
Epoch 20/30
40/40 [==============================] - 10s 247ms/step - loss: 0.0023 - accuracy: 0.9993 - val_loss: 0.1929 - val_accuracy: 0.9590
Epoch 21/30
40/40 [==============================] - 11s 250ms/step - loss: 0.0027 - accuracy: 0.9989 - val_loss: 0.2154 - val_accuracy: 0.9508
Epoch 22/30
40/40 [==============================] - 11s 252ms/step - loss: 0.0010 - accuracy: 0.9996 - val_loss: 0.2265 - val_accuracy: 0.9590
Epoch 23/30
40/40 [==============================] - 11s 250ms/step - loss: 7.6562e-04 - accuracy: 0.9996 - val_loss: 0.2131 - val_accuracy: 0.9590
Epoch 24/30
40/40 [==============================] - 11s 269ms/step - loss: 4.5182e-04 - accuracy: 1.0000 - val_loss: 0.2452 - val_accuracy: 0.9590
Epoch 25/30
40/40 [==============================] - 11s 249ms/step - loss: 4.0824e-04 - accuracy: 1.0000 - val_loss: 0.2593 - val_accuracy: 0.9590
Epoch 26/30
40/40 [==============================] - 11s 247ms/step - loss: 5.4401e-04 - accuracy: 1.0000 - val_loss: 0.2439 - val_accuracy: 0.9590
Epoch 27/30
40/40 [==============================] - 11s 252ms/step - loss: 0.0012 - accuracy: 0.9996 - val_loss: 0.2325 - val_accuracy: 0.9590
Epoch 28/30
40/40 [==============================] - 11s 252ms/step - loss: 0.0038 - accuracy: 0.9989 - val_loss: 0.1861 - val_accuracy: 0.9672
Epoch 29/30
40/40 [==============================] - 11s 252ms/step - loss: 0.0073 - accuracy: 0.9967 - val_loss: 0.2292 - val_accuracy: 0.9590
Epoch 30/30
40/40 [==============================] - 11s 249ms/step - loss: 0.0100 - accuracy: 0.9971 - val_loss: 0.2215 - val_accuracy: 0.9590
```

Plotting the Training and Validation Accuracy and Training and Validation Loss

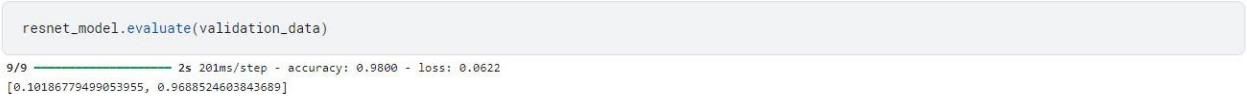


```
import matplotlib.pyplot as plt

# Plot every other accuracy for training and validation
plt.plot(history.history['accuracy'][::2], label='Training Accuracy')
plt.plot(history.history['val_accuracy'][::2], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.show()

# Plot every other loss for training and validation
plt.plot(history.history['loss'][::2], label='Training Loss')
plt.plot(history.history['val_loss'][::2], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```





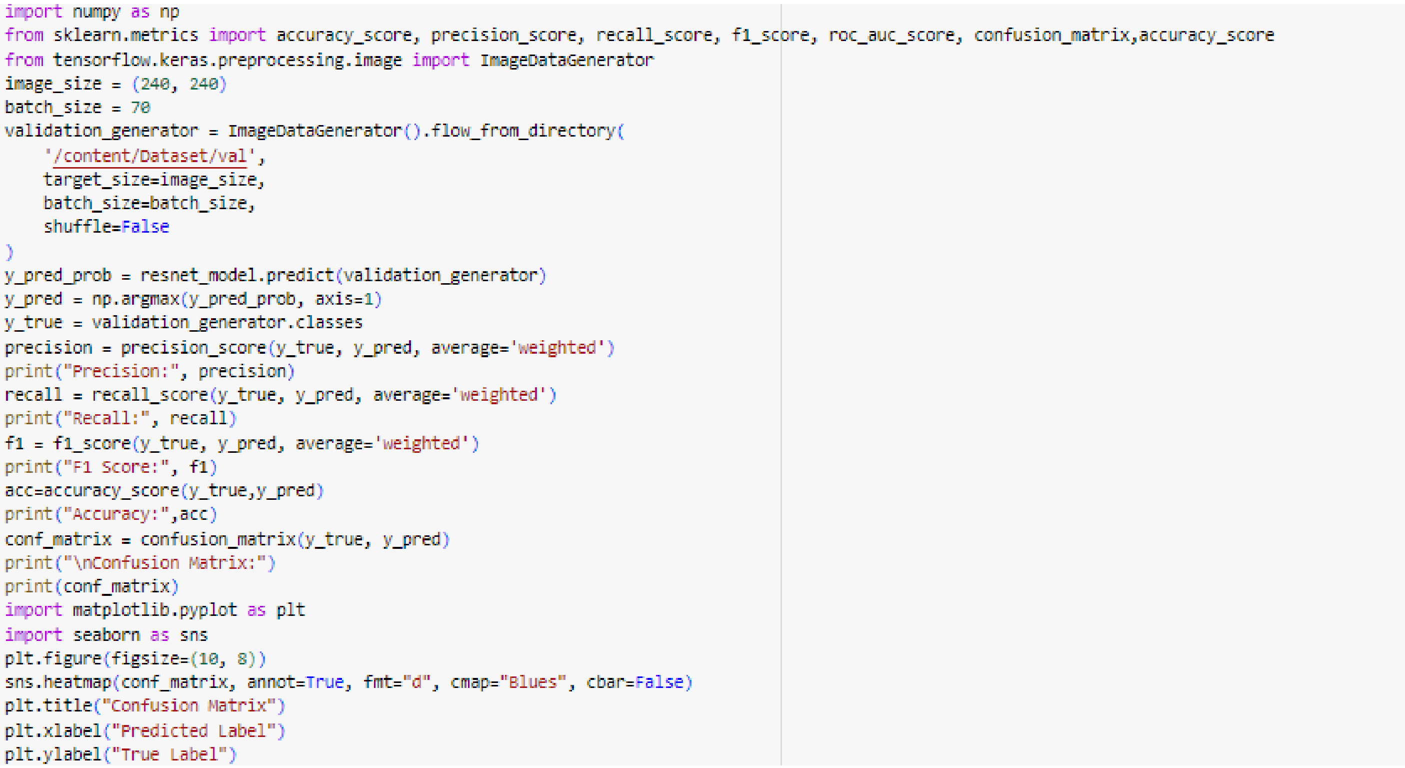
Testing Accuracy



```
resnet_model.evaluate(validation_data)
```

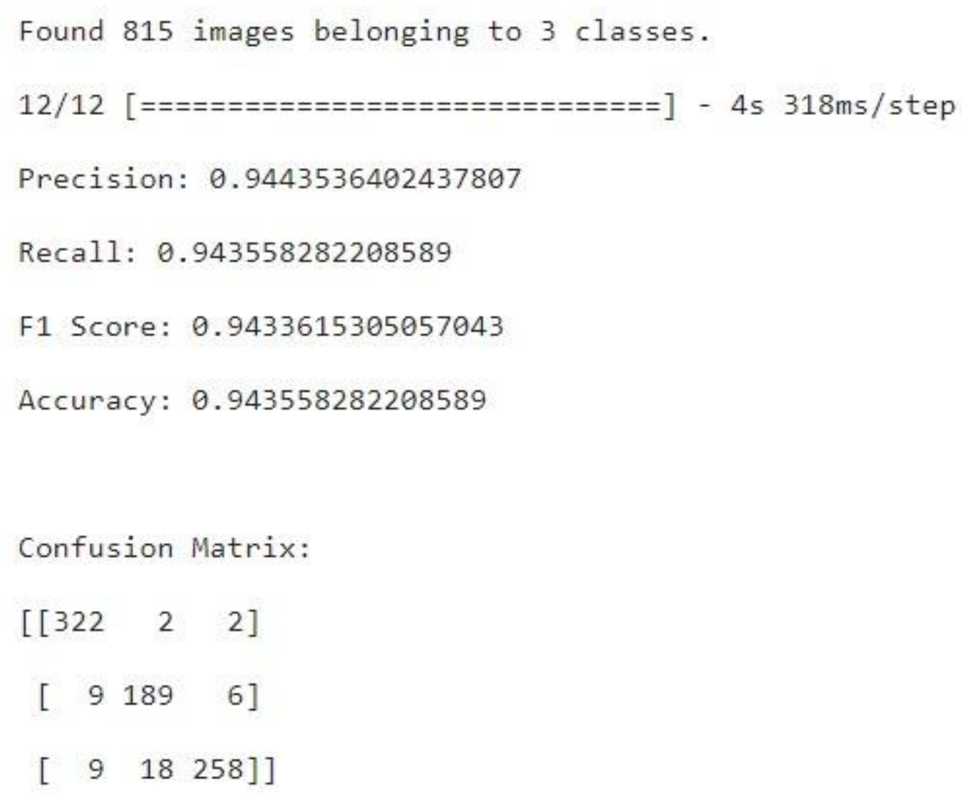
```
9/9 ━━━━━━━━━━━━━━━━━━━━ 2s 201ms/step - accuracy: 0.9800 - loss: 0.0622
[0.10186779499053955, 0.9688524603843689]
```

Performance Metrics

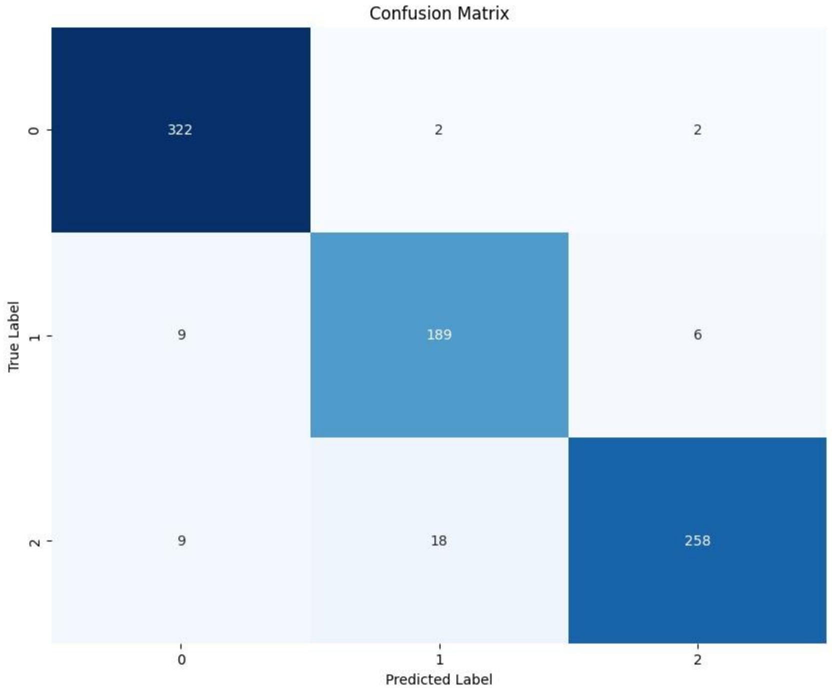


```
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix,accuracy_score
from tensorflow.keras.preprocessing.image import ImageDataGenerator
image_size = (240, 240)
batch_size = 70
validation_generator = ImageDataGenerator().flow_from_directory(
    '/content/Dataset/val',
    target_size=image_size,
    batch_size=batch_size,
    shuffle=False
)
y_pred_prob = resnet_model.predict(validation_generator)
y_pred = np.argmax(y_pred_prob, axis=1)
y_true = validation_generator.classes
precision = precision_score(y_true, y_pred, average='weighted')
print("Precision:", precision)
recall = recall_score(y_true, y_pred, average='weighted')
print("Recall:", recall)
f1 = f1_score(y_true, y_pred, average='weighted')
print("F1 Score:", f1)
acc=accuracy_score(y_true,y_pred)
print("Accuracy:",acc)
conf_matrix = confusion_matrix(y_true, y_pred)
print("\nConfusion Matrix:")
print(conf_matrix)
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
```

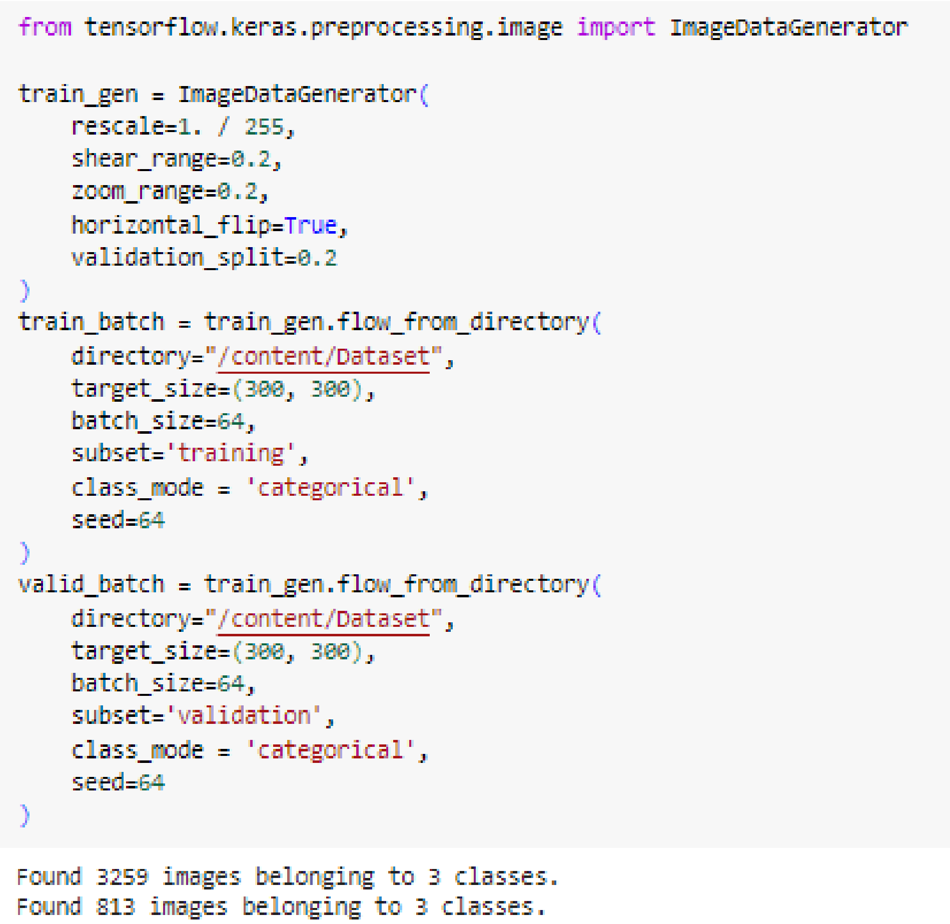
For test data



```
Found 815 images belonging to 3 classes.

12/12 [==============================] - 4s 318ms/step

Precision: 0.9443536402437807

Recall: 0.943558282208589

F1 Score: 0.9433615305057043

Accuracy: 0.943558282208589


Confusion Matrix:

[[322   2   2]

 [  9 189   6]

 [  9  18 258]]
```

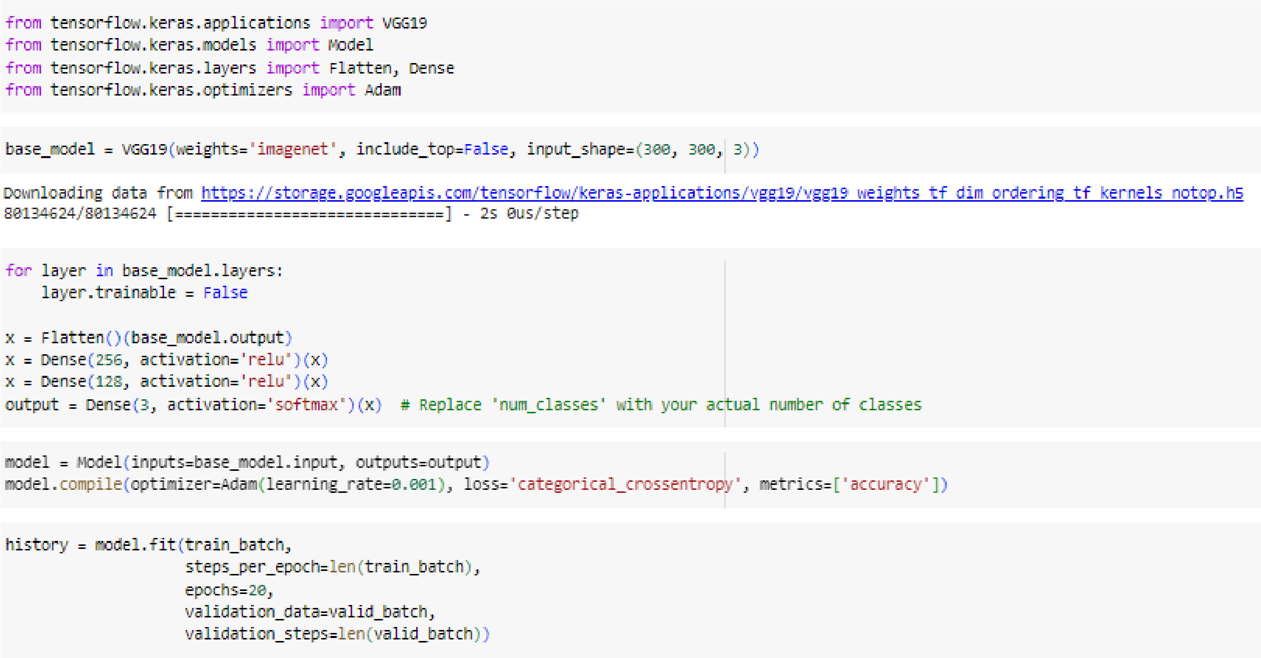


Using VGG19



```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_gen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)
train_batch = train_gen.flow_from_directory(
    directory="/content/Dataset",
    target_size=(300, 300),
    batch_size=64,
    subset='training',
    class_mode = 'categorical',
    seed=64
)
valid_batch = train_gen.flow_from_directory(
    directory="/content/Dataset",
    target_size=(300, 300),
    batch_size=64,
    subset='validation',
    class_mode = 'categorical',
    seed=64
)
```

```
Found 3259 images belonging to 3 classes.
Found 813 images belonging to 3 classes.
```



```
from tensorflow.keras.applications import VGG19
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.optimizers import Adam
```

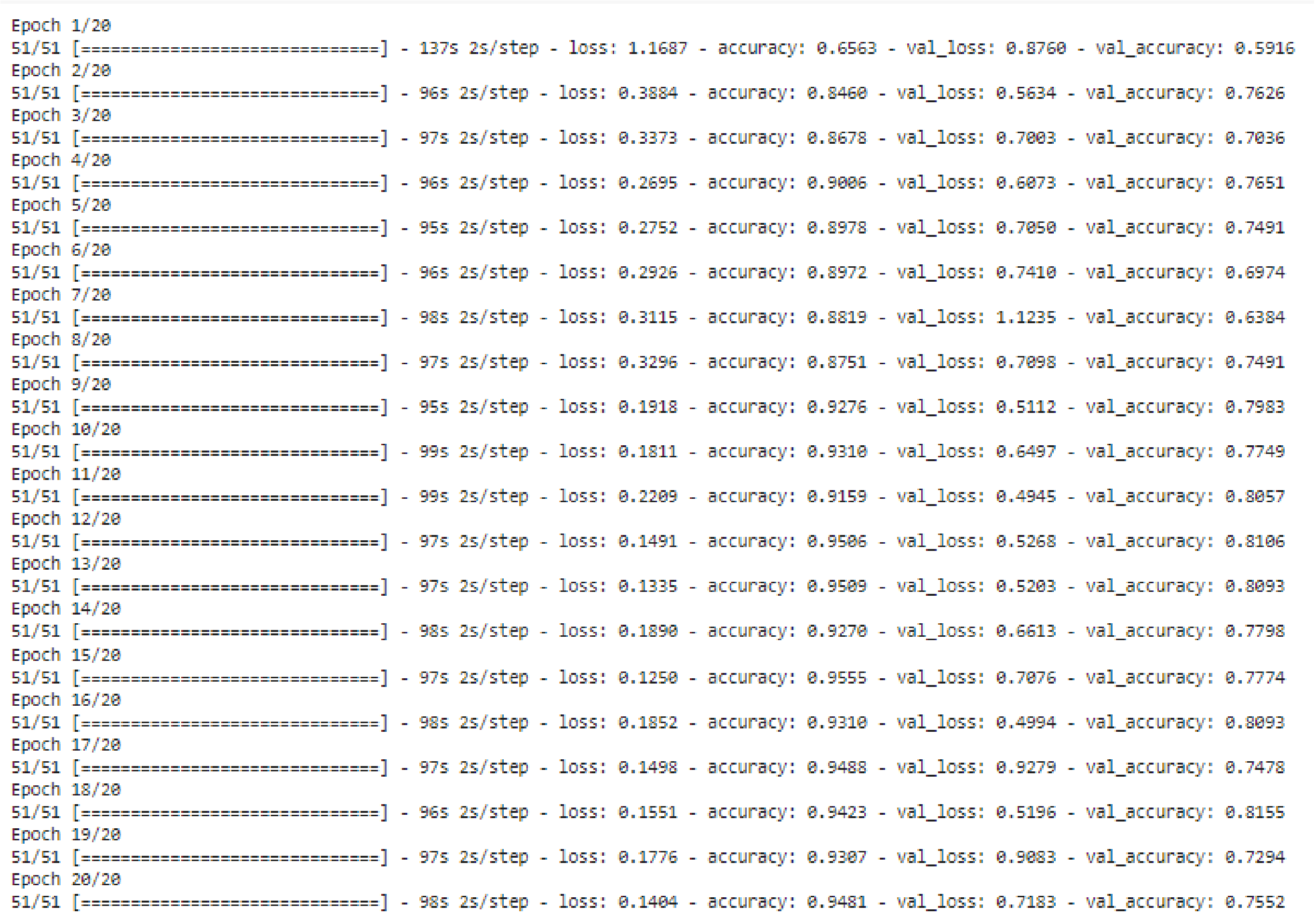
```
base_model = VGG19(weights='imagenet', include_top=False, input_shape=(300, 300, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [==============================] - 2s 0us/step
```

```
for layer in base_model.layers:
    layer.trainable = False

x = Flatten()(base_model.output)
x = Dense(256, activation='relu')(x)
x = Dense(128, activation='relu')(x)
output = Dense(3, activation='softmax')(x)  # Replace 'num_classes' with your actual number of classes
```

```
model = Model(inputs=base_model.input, outputs=output)
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
history = model.fit(train_batch,
                    steps_per_epoch=len(train_batch),
                    epochs=20,
                    validation_data=valid_batch,
                    validation_steps=len(valid_batch))
```



```
Epoch 1/20
51/51 [==============================] - 137s 2s/step - loss: 1.1687 - accuracy: 0.6563 - val_loss: 0.8760 - val_accuracy: 0.5916
Epoch 2/20
51/51 [==============================] - 96s 2s/step - loss: 0.3884 - accuracy: 0.8460 - val_loss: 0.5634 - val_accuracy: 0.7626
Epoch 3/20
51/51 [==============================] - 97s 2s/step - loss: 0.3373 - accuracy: 0.8678 - val_loss: 0.7003 - val_accuracy: 0.7036
Epoch 4/20
51/51 [==============================] - 96s 2s/step - loss: 0.2695 - accuracy: 0.9006 - val_loss: 0.6073 - val_accuracy: 0.7651
Epoch 5/20
51/51 [==============================] - 95s 2s/step - loss: 0.2752 - accuracy: 0.8978 - val_loss: 0.7050 - val_accuracy: 0.7491
Epoch 6/20
51/51 [==============================] - 96s 2s/step - loss: 0.2926 - accuracy: 0.8972 - val_loss: 0.7410 - val_accuracy: 0.6974
Epoch 7/20
51/51 [==============================] - 98s 2s/step - loss: 0.3115 - accuracy: 0.8819 - val_loss: 1.1235 - val_accuracy: 0.6384
Epoch 8/20
51/51 [==============================] - 97s 2s/step - loss: 0.3296 - accuracy: 0.8751 - val_loss: 0.7098 - val_accuracy: 0.7491
Epoch 9/20
51/51 [==============================] - 95s 2s/step - loss: 0.1918 - accuracy: 0.9276 - val_loss: 0.5112 - val_accuracy: 0.7983
Epoch 10/20
51/51 [==============================] - 99s 2s/step - loss: 0.1811 - accuracy: 0.9310 - val_loss: 0.6497 - val_accuracy: 0.7749
Epoch 11/20
51/51 [==============================] - 99s 2s/step - loss: 0.2209 - accuracy: 0.9159 - val_loss: 0.4945 - val_accuracy: 0.8057
Epoch 12/20
51/51 [==============================] - 97s 2s/step - loss: 0.1491 - accuracy: 0.9506 - val_loss: 0.5268 - val_accuracy: 0.8106
Epoch 13/20
51/51 [==============================] - 97s 2s/step - loss: 0.1335 - accuracy: 0.9509 - val_loss: 0.5203 - val_accuracy: 0.8093
Epoch 14/20
51/51 [==============================] - 98s 2s/step - loss: 0.1890 - accuracy: 0.9270 - val_loss: 0.6613 - val_accuracy: 0.7798
Epoch 15/20
51/51 [==============================] - 97s 2s/step - loss: 0.1250 - accuracy: 0.9555 - val_loss: 0.7076 - val_accuracy: 0.7774
Epoch 16/20
51/51 [==============================] - 98s 2s/step - loss: 0.1852 - accuracy: 0.9310 - val_loss: 0.4994 - val_accuracy: 0.8093
Epoch 17/20
51/51 [==============================] - 97s 2s/step - loss: 0.1498 - accuracy: 0.9488 - val_loss: 0.9279 - val_accuracy: 0.7478
Epoch 18/20
51/51 [==============================] - 96s 2s/step - loss: 0.1551 - accuracy: 0.9423 - val_loss: 0.5196 - val_accuracy: 0.8155
Epoch 19/20
51/51 [==============================] - 97s 2s/step - loss: 0.1776 - accuracy: 0.9307 - val_loss: 0.9083 - val_accuracy: 0.7294
Epoch 20/20
51/51 [==============================] - 98s 2s/step - loss: 0.1404 - accuracy: 0.9481 - val_loss: 0.7183 - val_accuracy: 0.7552
```

We will get better results in Resnet compared to VGG19.