Potato Leaf Dataset PlantVillage

Transfer Learning models (VGG19, GoogLeNet, ResNet50)

Inspiration taken from: https://www.kaggle.com/code/sakthimurugavel/potato-leaf-disease-classifier-review-ii-final/notebook

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Import all the required dependencies

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf

from tensorflow.keras import models, layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import load_model, Model
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.resnet_v2 import ResNet50V2
from tensorflow.keras.utils import plot_model
```

Define the constants

```
BATCH_SIZE = 32
IMAGE_SIZE = 224
CHANNELS = 3
EPOCHS = 20
```

```
path = "/content/drive/MyDrive/Colab Notebooks/Dissertation/Code/PlantVillage"
```

```
dataset = tf.keras.preprocessing.image_dataset_from_directory(path,
                                                              shuffle = True,
                                                              image_size = (IMAGE_SIZE, IMAGE_SIZE),
                                                              batch_size = BATCH_SIZE
)
```

```
Found 2152 files belonging to 3 classes.
```

```
class_names = dataset.class_names
class_names # 0 = early blight, 1 = late blight, 2 = healthy
```

```
['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']
```

```
def DatasetSize(path):
    number_of_images = {} #tuple
    for folder in os.listdir(path):
        number_of_images[folder] = len(os.listdir(os.path.join(path, folder)))
    return number_of_images

data_set = DatasetSize(path)
print(data_set)
```

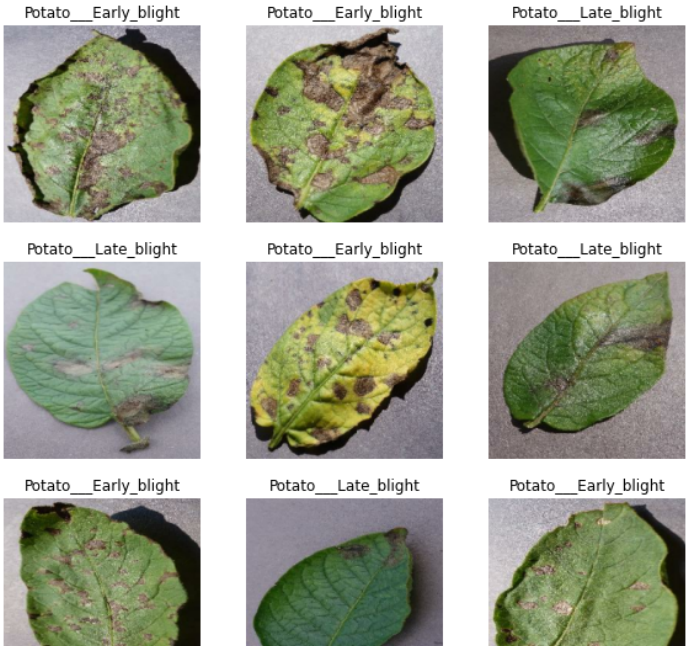
```
{'Potato___healthy': 152, 'Potato___Early_blight': 1000, 'Potato___Late_blight': 1000}
```

```
for image_batch, labels_batch in dataset.take(1):
    print(image_batch.shape)
    print(labels_batch.numpy())
```

```
(32, 224, 224, 3)
[0 1 0 0 0 0 1 0 1 1 0 0 0 0 2 1 0 1 0 1 1 1 0 2 1 1 2 0 1 1 0 1]
```

Visualise the images

```
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in dataset.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i+1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```



Split the dataset

```
# Use split folders
#splitfolders --ratio .8 .1 .1 --
```

```
!pip install split-folders
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
```

```
import splitfolders
splitfolders.ratio(path, output="output", seed=1337, ratio=(.8, 0.1,0.1))
```

```
Copying files: 2152 files [00:29, 73.42 files/s]
```

Data Preprocessing and Augmentation

```
train_datagen = ImageDataGenerator(
        rescale = 1./255,
        rotation_range = 10,
        horizontal_flip = True
)

train_generator = train_datagen.flow_from_directory("/content/drive/MyDrive/Colab Notebooks/Dissertation/Code/TransferLearning/output/tra
                                                    target_size = (IMAGE_SIZE, IMAGE_SIZE),
                                                    batch_size = 32,
                                                    class_mode = "sparse")
```

```
Found 1721 images belonging to 3 classes.
```

```
train_generator.class_indices
```

```
{'Potato___Early_blight': 0, 'Potato___Late_blight': 1, 'Potato___healthy': 2}
```

```
class_names = list(train_generator.class_indices.keys())
class_names
```

```
['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']
```

```
# Validation

validation_datagen = ImageDataGenerator(
        rescale = 1./255,
        rotation_range = 10,
        horizontal_flip = True
)

validation_generator = validation_datagen.flow_from_directory("/content/drive/MyDrive/Colab Notebooks/Dissertation/Code/TransferLearning/
                                                target_size = (IMAGE_SIZE, IMAGE_SIZE),
                                                batch_size = 32,
                                                class_mode = "sparse")
```

```
Found 215 images belonging to 3 classes.
```

```
# Test

test_datagen = ImageDataGenerator(
        rescale = 1./255,
        rotation_range = 10,
        horizontal_flip = True
)

test_generator = test_datagen.flow_from_directory("/content/drive/MyDrive/Colab Notebooks/Dissertation/Code/TransferLearning/output/test"
                                                target_size = (IMAGE_SIZE, IMAGE_SIZE),
                                                batch_size = 32,
                                                class_mode = "sparse")
```

```
Found 216 images belonging to 3 classes.
```

VGG19

```
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 3

def create_model():
    model = models.Sequential()
    model.add(VGG19(include_top = False, weights = 'imagenet', input_shape = (IMAGE_SIZE, IMAGE_SIZE, CHANNELS)))
    model.add(layers.GlobalAveragePooling2D())
    model.add(layers.Flatten())
    model.add(layers.Dense(256, activation = 'relu'))
    model.add(layers.Dropout(0.3))
    model.add(layers.Dense(n_classes, activation='softmax'))

    return model


vgg19_with_aug_model = create_model()
vgg19_with_aug_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_n
80134624/80134624 [==============================] - 4s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 7, 7, 512)         20024384

 global_average_pooling2d (G  (None, 512)              0
 lobalAveragePooling2D)

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 256)               131328

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 3)                 771
```

```
=================================================================
Total params: 20,156,483
Trainable params: 20,156,483
Non-trainable params: 0
_________________________________________________________________
```

Compiling the model and determining the accuracy

```
vgg19_with_aug_model.compile(
                          optimizer= 'adam',
                          loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
                          metrics=['accuracy'])
```

```
vgg19_with_aug_history = vgg19_with_aug_model.fit(
    train_generator,
    steps_per_epoch=54,
    batch_size=32,
    validation_data=validation_generator,
    validation_steps=7,
    verbose=1,
    epochs=EPOCHS,
)
```

```
Epoch 1/20
54/54 [==============================] - 640s 12s/step - loss: 4.5245 - accuracy: 0.4625 - val_loss: 0.8797 - val_accuracy: 0.7535
Epoch 2/20
54/54 [==============================] - 31s 565ms/step - loss: 0.8920 - accuracy: 0.5514 - val_loss: 0.8381 - val_accuracy: 0.5302
Epoch 3/20
54/54 [==============================] - 31s 570ms/step - loss: 0.8878 - accuracy: 0.5596 - val_loss: 0.7391 - val_accuracy: 0.6419
Epoch 4/20
54/54 [==============================] - 32s 596ms/step - loss: 0.6748 - accuracy: 0.7420 - val_loss: 0.5713 - val_accuracy: 0.7674
Epoch 5/20
54/54 [==============================] - 31s 577ms/step - loss: 0.5416 - accuracy: 0.7966 - val_loss: 0.4877 - val_accuracy: 0.8512
Epoch 6/20
54/54 [==============================] - 31s 576ms/step - loss: 0.4621 - accuracy: 0.8466 - val_loss: 0.7797 - val_accuracy: 0.7116
Epoch 7/20
54/54 [==============================] - 31s 574ms/step - loss: 0.4078 - accuracy: 0.8518 - val_loss: 0.4162 - val_accuracy: 0.8605
Epoch 8/20
54/54 [==============================] - 31s 576ms/step - loss: 0.3629 - accuracy: 0.8681 - val_loss: 0.4231 - val_accuracy: 0.8372
Epoch 9/20
54/54 [==============================] - 32s 595ms/step - loss: 0.4372 - accuracy: 0.8478 - val_loss: 0.3687 - val_accuracy: 0.8651
Epoch 10/20
54/54 [==============================] - 31s 575ms/step - loss: 0.3449 - accuracy: 0.8832 - val_loss: 0.3855 - val_accuracy: 0.8558
Epoch 11/20
54/54 [==============================] - 31s 575ms/step - loss: 0.3167 - accuracy: 0.8919 - val_loss: 0.2946 - val_accuracy: 0.8930
Epoch 12/20
54/54 [==============================] - 31s 575ms/step - loss: 0.2906 - accuracy: 0.8977 - val_loss: 0.3445 - val_accuracy: 0.8837
Epoch 13/20
54/54 [==============================] - 31s 576ms/step - loss: 0.3008 - accuracy: 0.8902 - val_loss: 0.2773 - val_accuracy: 0.8977
Epoch 14/20
54/54 [==============================] - 33s 597ms/step - loss: 0.2608 - accuracy: 0.9035 - val_loss: 0.3608 - val_accuracy: 0.8605
Epoch 15/20
54/54 [==============================] - 31s 574ms/step - loss: 0.2863 - accuracy: 0.8960 - val_loss: 0.3144 - val_accuracy: 0.8837
Epoch 16/20
54/54 [==============================] - 32s 593ms/step - loss: 0.2907 - accuracy: 0.8925 - val_loss: 0.2502 - val_accuracy: 0.8977
Epoch 17/20
54/54 [==============================] - 32s 588ms/step - loss: 0.2262 - accuracy: 0.9024 - val_loss: 0.2031 - val_accuracy: 0.9023
Epoch 18/20
54/54 [==============================] - 32s 595ms/step - loss: 0.1819 - accuracy: 0.9297 - val_loss: 0.3434 - val_accuracy: 0.8977
Epoch 19/20
54/54 [==============================] - 32s 579ms/step - loss: 0.1773 - accuracy: 0.9309 - val_loss: 0.3373 - val_accuracy: 0.9023
Epoch 20/20
54/54 [==============================] - 31s 574ms/step - loss: 0.1745 - accuracy: 0.9332 - val_loss: 0.1051 - val_accuracy: 0.9767
```

Producing the test accuracy and test loss

```
vgg19_with_aug_scores = vgg19_with_aug_model.evaluate(test_generator)
vgg19_with_aug_test_acc = vgg19_with_aug_scores[1]*100
vgg19_with_aug_test_loss = vgg19_with_aug_scores[0]
print("Test Loss: ", vgg19_with_aug_test_loss)
print("Test Accuracy: ", vgg19_with_aug_test_acc)
```

```
7/7 [==============================] - 107s 18s/step - loss: 0.0859 - accuracy: 0.9769
Test Loss:  0.08585644513368607
Test Accuracy:  97.68518805503845
```

VGG19 plot for accuracy vs loss

```
vgg19_with_aug_history
```

```
<keras.callbacks.History at 0x7ff2a06acfa0>
```

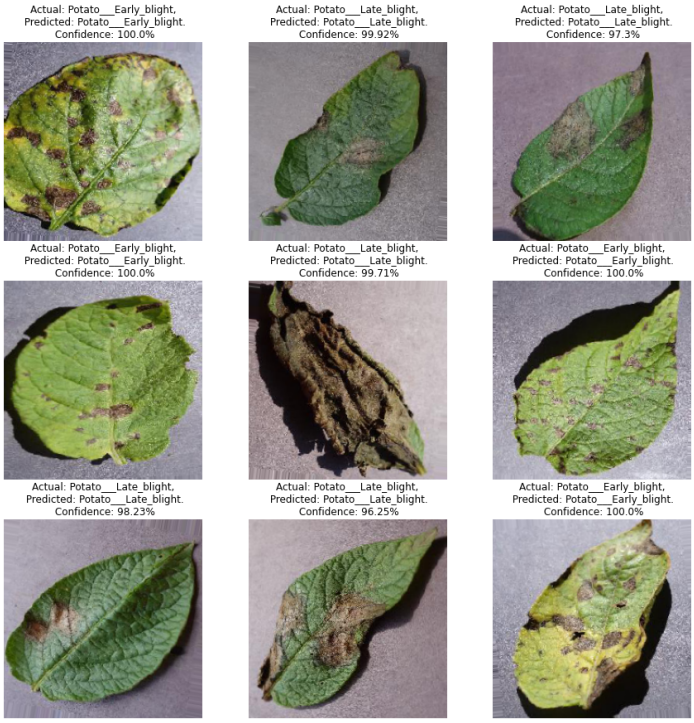
```
vgg19_with_aug_acc = vgg19_with_aug_history.history['accuracy']
vgg19_with_aug_val_acc = vgg19_with_aug_history.history['val_accuracy']

vgg19_with_aug_loss = vgg19_with_aug_history.history['loss']
vgg19_with_aug_val_loss = vgg19_with_aug_history.history['val_loss']

n = range(1, EPOCHS+1)


plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(n, vgg19_with_aug_acc, label='Training Accuracy')
plt.plot(n, vgg19_with_aug_val_acc, label='Validation Accuracy')
plt.legend(loc='best')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')

plt.subplot(1,2,2)
plt.plot(n, vgg19_with_aug_loss, label='Training Loss')
plt.plot(n, vgg19_with_aug_val_loss, label='Validation Loss')
plt.legend(loc='best')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')
plt.tight_layout()
plt.suptitle('VGG19 Model')
plt.savefig("plot_vgg19.png")
plt.show()
```

Prediction of images using VGG19

```
def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i])
    img_array = tf.expand_dims(img_array, 0)

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])), 2)
    return predicted_class, confidence


plt.figure(figsize=(15, 15))
for images, labels in test_generator:
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i])

        predicted_class, confidence = predict(vgg19_with_aug_model, images[i])
        actual_class = class_names[int(labels[i])]

        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence: {confidence}%")

        plt.axis("off")
```

```
    break

plt.savefig("VGG19_prediction images")
plt.show()
```

```
1/1 [==============================] - 1s 836ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
```



GoogLeNet (InceptionV3)

```
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 3

def create_model():
    model = models.Sequential()
    model.add(VGG19(include_top = False, weights = 'imagenet', input_shape = (IMAGE_SIZE, IMAGE_SIZE, CHANNELS)))
    model.add(layers.GlobalAveragePooling2D())
    model.add(layers.Flatten())
    model.add(layers.Dense(256, activation = 'relu'))
    model.add(layers.Dropout(0.3))
    model.add(layers.Dense(n_classes, activation='softmax'))

    return model


inceptionv3_with_aug_model = create_model()
inceptionv3_with_aug_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
 vgg19 (Functional)          (None, 7, 7, 512)         20024384

 global_average_pooling2d_1   (None, 512)              0
 (GlobalAveragePooling2D)

 flatten_1 (Flatten)         (None, 512)               0

 dense_2 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 3)                 771

=================================================================
Total params: 20,156,483
Trainable params: 20,156,483
Non-trainable params: 0
_________________________________________________________________
```

Compiling the model and determining the accuracy

```
inceptionv3_with_aug_model.compile(
                          optimizer='adam',
                          loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
                          metrics=['accuracy'])
```

```
inceptionv3_with_aug_history = inceptionv3_with_aug_model.fit(train_generator,
                                              steps_per_epoch=54,
                                              batch_size=32,
                                              validation_data=validation_generator,
                                               validation_steps = 7,
                                               verbose=1,
                                               epochs=EPOCHS)
```

```
Epoch 1/20
54/54 [==============================] - 34s 598ms/step - loss: 4.8246 - accuracy: 0.4619 - val_loss: 0.9054 - val_accuracy: 0.4651
Epoch 2/20
54/54 [==============================] - 33s 600ms/step - loss: 0.9282 - accuracy: 0.4759 - val_loss: 0.8882 - val_accuracy: 0.5767
Epoch 3/20
54/54 [==============================] - 31s 568ms/step - loss: 0.9091 - accuracy: 0.5044 - val_loss: 0.8902 - val_accuracy: 0.4651
Epoch 4/20
54/54 [==============================] - 31s 571ms/step - loss: 0.8458 - accuracy: 0.5683 - val_loss: 0.7701 - val_accuracy: 0.6140
Epoch 5/20
54/54 [==============================] - 31s 575ms/step - loss: 0.5653 - accuracy: 0.7960 - val_loss: 0.5275 - val_accuracy: 0.8047
Epoch 6/20
54/54 [==============================] - 31s 574ms/step - loss: 0.4382 - accuracy: 0.8425 - val_loss: 0.6440 - val_accuracy: 0.7628
Epoch 7/20
54/54 [==============================] - 32s 595ms/step - loss: 0.4453 - accuracy: 0.8437 - val_loss: 0.5518 - val_accuracy: 0.8326
Epoch 8/20
54/54 [==============================] - 32s 578ms/step - loss: 0.3829 - accuracy: 0.8629 - val_loss: 0.4716 - val_accuracy: 0.8233
Epoch 9/20
54/54 [==============================] - 32s 581ms/step - loss: 0.3610 - accuracy: 0.8681 - val_loss: 0.2927 - val_accuracy: 0.8837
Epoch 10/20
54/54 [==============================] - 31s 577ms/step - loss: 0.2724 - accuracy: 0.8954 - val_loss: 0.2709 - val_accuracy: 0.8884
Epoch 11/20
54/54 [==============================] - 32s 580ms/step - loss: 0.3421 - accuracy: 0.8861 - val_loss: 0.3495 - val_accuracy: 0.8605
Epoch 12/20
54/54 [==============================] - 33s 601ms/step - loss: 0.2925 - accuracy: 0.8902 - val_loss: 0.3155 - val_accuracy: 0.8930
Epoch 13/20
54/54 [==============================] - 31s 576ms/step - loss: 0.2589 - accuracy: 0.8966 - val_loss: 0.2554 - val_accuracy: 0.8930
Epoch 14/20
54/54 [==============================] - 32s 579ms/step - loss: 0.2665 - accuracy: 0.8873 - val_loss: 0.2090 - val_accuracy: 0.9256
Epoch 15/20
54/54 [==============================] - 31s 575ms/step - loss: 0.2021 - accuracy: 0.9157 - val_loss: 0.1657 - val_accuracy: 0.9488
Epoch 16/20
54/54 [==============================] - 32s 593ms/step - loss: 0.2609 - accuracy: 0.8942 - val_loss: 0.2309 - val_accuracy: 0.9349
Epoch 17/20
54/54 [==============================] - 32s 578ms/step - loss: 0.1720 - accuracy: 0.9355 - val_loss: 0.4846 - val_accuracy: 0.8372
Epoch 18/20
54/54 [==============================] - 31s 576ms/step - loss: 0.2049 - accuracy: 0.9169 - val_loss: 0.1637 - val_accuracy: 0.9442
Epoch 19/20
54/54 [==============================] - 31s 572ms/step - loss: 0.1477 - accuracy: 0.9419 - val_loss: 0.0856 - val_accuracy: 0.9860
Epoch 20/20
54/54 [==============================] - 31s 574ms/step - loss: 0.1248 - accuracy: 0.9570 - val_loss: 0.1852 - val_accuracy: 0.9349
```

Producing the test accuracy and test loss

```
inceptionv3_with_aug_scores = inceptionv3_with_aug_model.evaluate(test_generator)
inceptionv3_with_aug_test_acc = inceptionv3_with_aug_scores[1]*100
inceptionv3_with_aug_test_loss = inceptionv3_with_aug_scores[0]
```

```
print("Test Loss: ", inceptionv3_with_aug_test_loss)
print("Test Accuracy: ", inceptionv3_with_aug_test_acc)
```

```
7/7 [==============================] - 4s 499ms/step - loss: 0.1769 - accuracy: 0.9398
Test Loss:  0.17685197293758392
Test Accuracy:  93.9814805984497
```

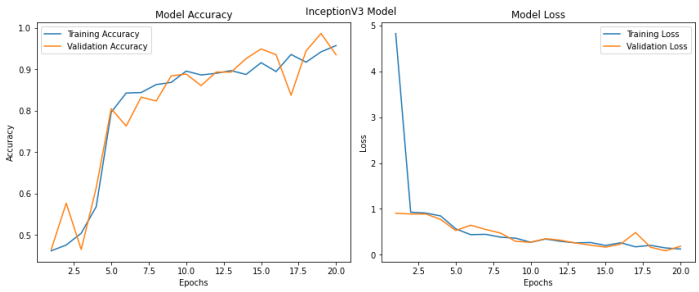
InceptionV3 plot for accuracy vs loss

```
inceptionv3_with_aug_history
```

```
<keras.callbacks.History at 0x7ff2a184b070>
```

```
inceptionv3_with_aug_acc = inceptionv3_with_aug_history.history['accuracy']
inceptionv3_with_aug_val_acc = inceptionv3_with_aug_history.history['val_accuracy']

inceptionv3_with_aug_loss = inceptionv3_with_aug_history.history['loss']
inceptionv3_with_aug_val_loss = inceptionv3_with_aug_history.history['val_loss']

n = range(1, EPOCHS+1)
```

```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(n, inceptionv3_with_aug_acc, label='Training Accuracy')
plt.plot(n, inceptionv3_with_aug_val_acc, label='Validation Accuracy')
plt.legend(loc='best')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')

plt.subplot(1,2,2)
plt.plot(n, inceptionv3_with_aug_loss, label='Training Loss')
plt.plot(n, inceptionv3_with_aug_val_loss, label='Validation Loss')
plt.legend(loc='best')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')
plt.tight_layout()
plt.suptitle('InceptionV3 Model')
plt.savefig("plot_inceptionv3.png")
plt.show()
```



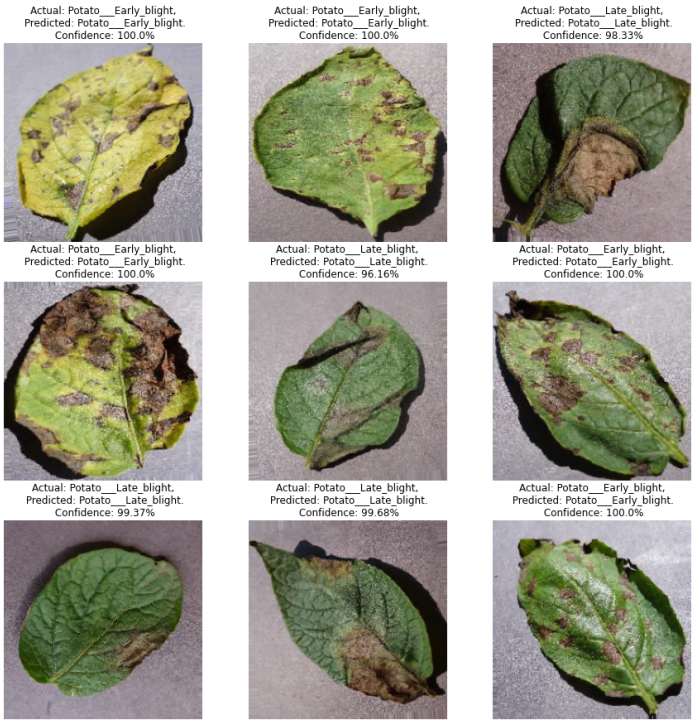
Prediction of images using InceptionV3

```
def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i])
    img_array = tf.expand_dims(img_array, 0)

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])), 2)
    return predicted_class, confidence
```

```
plt.figure(figsize=(15, 15))
for images, labels in test_generator:
    for i in range(9):
```

```
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i])

        predicted_class, confidence = predict(inceptionv3_with_aug_model, images[i])
        actual_class = class_names[int(labels[i])]

        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence: {confidence}%")

        plt.axis("off")
    break

plt.savefig("InceptionV3_prediction images")
```

```
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
```



ResNet50

```
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 3

def create_model():
    model = models.Sequential()
    model.add(VGG19(include_top = False, weights = 'imagenet', input_shape = (IMAGE_SIZE, IMAGE_SIZE, CHANNELS)))
    model.add(layers.GlobalAveragePooling2D())
    model.add(layers.Flatten())
```

```
    model.add(layers.Dense(256, activation = 'relu'))
    model.add(layers.Dropout(0.3))
    model.add(layers.Dense(n_classes, activation='softmax'))

    return model


resnet50_with_aug_model = create_model()
resnet50_with_aug_model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 7, 7, 512)         20024384

 global_average_pooling2d_2   (None, 512)              0
 (GlobalAveragePooling2D)

 flatten_2 (Flatten)         (None, 512)               0

 dense_4 (Dense)             (None, 256)               131328

 dropout_2 (Dropout)         (None, 256)               0

 dense_5 (Dense)             (None, 3)                 771

=================================================================
Total params: 20,156,483
Trainable params: 20,156,483
Non-trainable params: 0
_________________________________________________________________
```

Compiling the model and determining the accuracy

```
resnet50_with_aug_model.compile(
                           optimizer='adam',
                           loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
                           metrics=['accuracy'])


resnet50_with_aug_history = resnet50_with_aug_model.fit(train_generator,
                                                steps_per_epoch=54,
                                                batch_size=32,
                                                validation_data=validation_generator,
                                                 validation_steps = 7,
                                                 verbose=1,
                                                 epochs=EPOCHS)
```

```
Epoch 1/20
54/54 [==============================] - 33s 584ms/step - loss: 4.7703 - accuracy: 0.4765 - val_loss: 0.8929 - val_accuracy: 0.4651
Epoch 2/20
54/54 [==============================] - 32s 579ms/step - loss: 0.9059 - accuracy: 0.5305 - val_loss: 0.9359 - val_accuracy: 0.4651
Epoch 3/20
54/54 [==============================] - 32s 580ms/step - loss: 0.7356 - accuracy: 0.6758 - val_loss: 0.6046 - val_accuracy: 0.7535
Epoch 4/20
54/54 [==============================] - 31s 576ms/step - loss: 0.5764 - accuracy: 0.7745 - val_loss: 0.4729 - val_accuracy: 0.8233
Epoch 5/20
54/54 [==============================] - 32s 596ms/step - loss: 0.4802 - accuracy: 0.8350 - val_loss: 0.5267 - val_accuracy: 0.8233
Epoch 6/20
54/54 [==============================] - 32s 580ms/step - loss: 0.4954 - accuracy: 0.8175 - val_loss: 0.4201 - val_accuracy: 0.8512
Epoch 7/20
54/54 [==============================] - 31s 576ms/step - loss: 0.3886 - accuracy: 0.8693 - val_loss: 0.4567 - val_accuracy: 0.8512
Epoch 8/20
54/54 [==============================] - 31s 576ms/step - loss: 0.3994 - accuracy: 0.8530 - val_loss: 0.4022 - val_accuracy: 0.8512
Epoch 9/20
54/54 [==============================] - 31s 573ms/step - loss: 0.4256 - accuracy: 0.8454 - val_loss: 0.3522 - val_accuracy: 0.8605
Epoch 10/20
54/54 [==============================] - 32s 594ms/step - loss: 0.3206 - accuracy: 0.8844 - val_loss: 0.2911 - val_accuracy: 0.8837
Epoch 11/20
54/54 [==============================] - 31s 572ms/step - loss: 0.2664 - accuracy: 0.8972 - val_loss: 0.2723 - val_accuracy: 0.8837
Epoch 12/20
54/54 [==============================] - 31s 575ms/step - loss: 0.3458 - accuracy: 0.8786 - val_loss: 0.6042 - val_accuracy: 0.8000
Epoch 13/20
54/54 [==============================] - 31s 576ms/step - loss: 0.2962 - accuracy: 0.8896 - val_loss: 0.2661 - val_accuracy: 0.8977
Epoch 14/20
54/54 [==============================] - 32s 588ms/step - loss: 0.2333 - accuracy: 0.9082 - val_loss: 0.2118 - val_accuracy: 0.9395
Epoch 15/20
54/54 [==============================] - 31s 574ms/step - loss: 0.1955 - accuracy: 0.9245 - val_loss: 0.2012 - val_accuracy: 0.9302
Epoch 16/20
54/54 [==============================] - 31s 576ms/step - loss: 0.1487 - accuracy: 0.9431 - val_loss: 0.1523 - val_accuracy: 0.9581
Epoch 17/20
54/54 [==============================] - 31s 577ms/step - loss: 0.1157 - accuracy: 0.9587 - val_loss: 0.1085 - val_accuracy: 0.9488
Epoch 18/20
54/54 [==============================] - 31s 577ms/step - loss: 0.1796 - accuracy: 0.9320 - val_loss: 0.1079 - val_accuracy: 0.9628
Epoch 19/20
```

```
54/54 [==============================] - 32s 590ms/step - loss: 0.2214 - accuracy: 0.9175 - val_loss: 0.1238 - val_accuracy: 0.9628
Epoch 20/20
54/54 [==============================] - 31s 576ms/step - loss: 0.1110 - accuracy: 0.9617 - val_loss: 0.1938 - val_accuracy: 0.9209
```

Producing the test accuracy and test loss

```
resnet50_with_aug_scores = resnet50_with_aug_model.evaluate(test_generator)
resnet50_with_aug_test_acc = resnet50_with_aug_scores[1]*100
resnet50_with_aug_test_loss = resnet50_with_aug_scores[0]
print("Test Loss: ", resnet50_with_aug_test_loss)
print("Test Accuracy: ", resnet50_with_aug_test_acc)
```

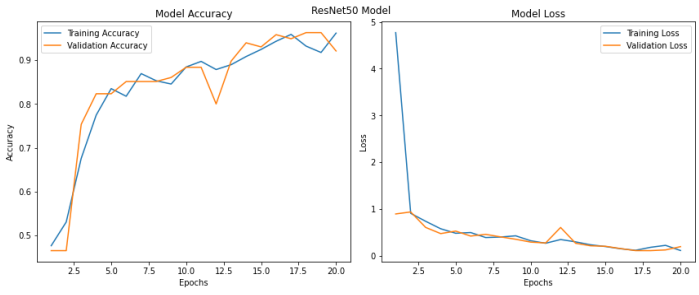
```
7/7 [==============================] - 3s 406ms/step - loss: 0.1820 - accuracy: 0.9167
Test Loss:  0.18200641870498657
Test Accuracy:  91.66666865348816
```

ResNet50 plot for accuracy vs loss

```
resnet50_with_aug_history
```

```
<keras.callbacks.History at 0x7ff2a1543cd0>
```

```
resnet50_with_aug_acc = resnet50_with_aug_history.history['accuracy']
resnet50_with_aug_val_acc = resnet50_with_aug_history.history['val_accuracy']

resnet50_with_aug_loss = resnet50_with_aug_history.history['loss']
resnet50_with_aug_val_loss = resnet50_with_aug_history.history['val_loss']

n = range(1, EPOCHS+1)


plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(n, resnet50_with_aug_acc, label='Training Accuracy')
plt.plot(n, resnet50_with_aug_val_acc, label='Validation Accuracy')
plt.legend(loc='best')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')

plt.subplot(1,2,2)
plt.plot(n, resnet50_with_aug_loss, label='Training Loss')
plt.plot(n, resnet50_with_aug_val_loss, label='Validation Loss')
plt.legend(loc='best')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')
plt.tight_layout()
plt.suptitle('ResNet50 Model')
plt.savefig("plot_resnet50.png")
plt.show()
```



Prediction of images using ResNet50

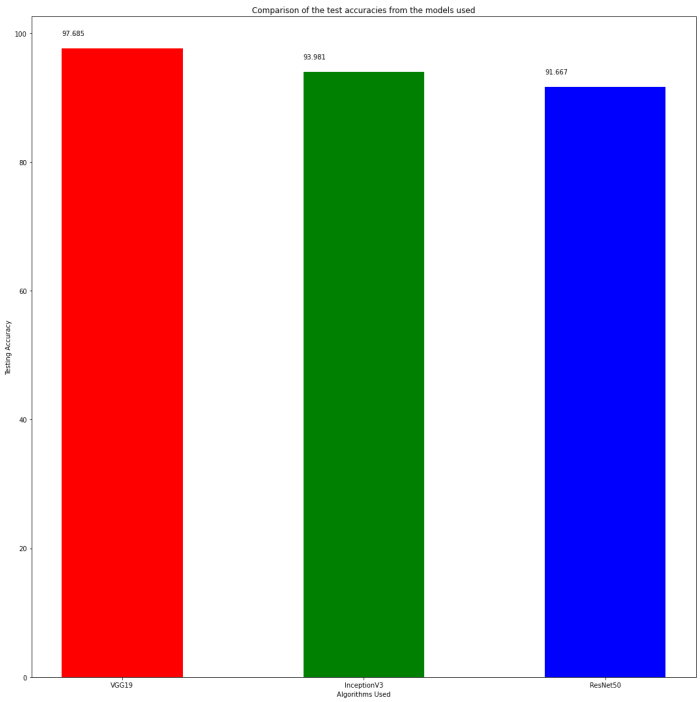
```
def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i])
    img_array = tf.expand_dims(img_array, 0)

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])), 2)
    return predicted_class, confidence


plt.figure(figsize=(15, 15))
for images, labels in test_generator:
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i])

        predicted_class, confidence = predict(resnet50_with_aug_model, images[i])
        actual_class = class_names[int(labels[i])]

        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence: {confidence}%")

        plt.axis("off")
    break

plt.savefig("ResNet50V2_prediction images")
```

```
 1/1 [==============================] - 0s 145ms/step
 1/1 [==============================] - 0s 15ms/step
 1/1 [==============================] - 0s 16ms/step
 1/1 [==============================] - 0s 17ms/step
 1/1 [==============================] - 0s 15ms/step
 1/1 [==============================] - 0s 14ms/step
 1/1 [==============================] - 0s 15ms/step
 1/1 [==============================] - 0s 17ms/step
 1/1 [==============================] - 0s 15ms/step
```

Compare the transfer learning models performances

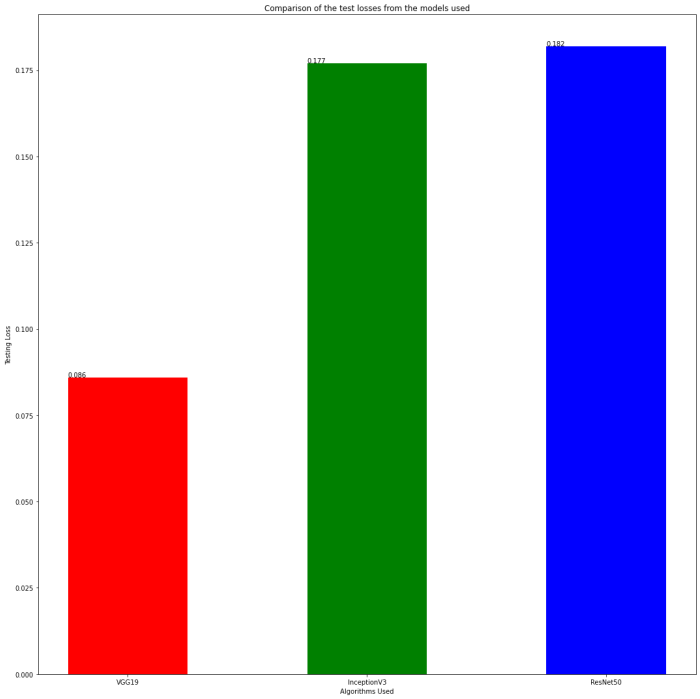
Comparing the test accuracies

```
algorithms = ['VGG19', 'InceptionV3', 'ResNet50']
accuracy = [vgg19_with_aug_test_acc, inceptionv3_with_aug_test_acc, resnet50_with_aug_test_acc]
accuracy = np.around([i for i in accuracy], 3)
colours = ['red', 'green', 'blue']

fig = plt.figure(figsize=(15,15))

# Plotting the bar chart
bars = plt.bar(algorithms, accuracy, color=colours, width = 0.5)

plt.xlabel("Algorithms Used")
plt.ylabel("Testing Accuracy")
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x(), yval+2, yval)

plt.title('Comparison of the test accuracies from the models used')
plt.tight_layout()
plt.savefig('comparison_of_test_accuracies.png')
plt.show()
```



Comparing the test loss

```
algorithms = ['VGG19', 'InceptionV3', 'ResNet50']
loss = [vgg19_with_aug_test_loss, inceptionv3_with_aug_test_loss, resnet50_with_aug_test_loss]
loss = np.around([i for i in loss], 3)
colours = ['red', 'green', 'blue']

fig = plt.figure(figsize=(15,15))

# Plotting the bar chart
bars = plt.bar(algorithms, loss, color=colours, width = 0.5)

plt.xlabel("Algorithms Used")
plt.ylabel("Testing Loss")
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x(), yval+0.00009, yval)

plt.title('Comparison of the test losses from the models used')
plt.tight_layout()
plt.savefig('comparison_of_test_losses.png')
plt.show()
```




```
!jupyter nbconvert --to html PlantVillageTransferLearning.ipynb
```

```
[NbConvertApp] Converting notebook PlantVillageTransferLearning.ipynb to html
[NbConvertApp] Writing 5359748 bytes to PlantVillageTransferLearning.html
```