In [1]:
```
import tensorflow as tf
from tensorflow.keras import models,layers
import matplotlib.pyplot as plt
from IPython.display import HTML
import numpy as np
```

In [2]:
```
BATCH_SIZE=32
IMAGE_SIZE=256
CHANNELS=3
EPOCHS=50
```

In [3]:
```
dataset=tf.keras.preprocessing.image_dataset_from_directory(
        "PlantVillage",
        seed=123,
        shuffle=True,
        image_size=(IMAGE_SIZE, IMAGE_SIZE),
        batch_size=BATCH_SIZE
)
```

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

In [4]:
```
class_names= dataset.class_names
```

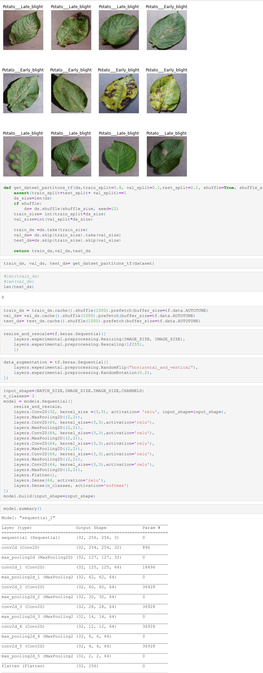
In [5]:
```
class_names
```

Out[5]:
```
['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']
```

In [6]:
```
len(dataset)
```

Out[6]:
```
68
```

In [7]:
```
plt.figure(figsize=(15, 15))
for image_batch, label_batch in dataset.take(1):
    for i in range(12):
        ax= plt.subplot(3,4,i+1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[label_batch[i]])
        plt.axis("off")
```



In [8]:
```
def get_dataset_partitions_tf(ds,train_split=0.8, val_split=0.1,test_split=0.1, shuffle=True, shuffle_size=10000):
    ds_size=len(ds)
    if shuffle:
        ds=ds.shuffle(shuffle_size, seed=12)
    train_size=int(train_split*ds_size)
    val_size=int(val_split*ds_size)

    train_ds =ds.take(train_size)
    val_ds=ds.skip(train_size).take(val_size)
    test_ds=ds.skip(train_size).skip(val_size)

    return train_ds,val_ds, test_ds
```

In [9]:
```
train_ds, val_ds, test_ds= get_dataset_partitions_tf(dataset)
```

In [10]:
```
#len(train_ds)
#len(val_ds)
len(test_ds)
```

Out[10]:
```
8
```

In [11]:
```
train_ds = train_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds= val_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds= test_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

In [12]:
```
resize_and_rescale=tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1/255),
])
```

In [13]:
```
data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2),
])
```

In [14]:
```
input_shape=(BATCH_SIZE,IMAGE_SIZE,IMAGE_SIZE,CHANNELS)
n_classes=3
model = models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32, kernel_size =(3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, kernel_size =(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, kernel_size =(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, kernel_size =(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, kernel_size =(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, kernel_size =(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])
model.build(input_shape=input_shape)
```

In [15]:
```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential (Sequential)      (32, 256, 256, 3)         0
_________________________________________________________________
conv2d (Conv2D)              (32, 254, 254, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (32, 127, 127, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (32, 125, 125, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (32, 62, 62, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (32, 60, 60, 64)          36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (32, 30, 30, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (32, 28, 28, 64)          36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (32, 14, 14, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (32, 12, 12, 64)          36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (32, 6, 6, 64)            0
_________________________________________________________________
conv2d_5 (Conv2D)            (32, 4, 4, 64)            36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (32, 2, 2, 64)            0
_________________________________________________________________
flatten (Flatten)            (32, 256)                 0
_________________________________________________________________
dense (Dense)                (32, 64)                  16448
_________________________________________________________________
dense_1 (Dense)              (32, 3)                   195
=================================================================
Total params: 183,747
Trainable params: 183,747
Non-trainable params: 0
_________________________________________________________________
```

In [16]:
```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

In [17]:
```
history = model.fit(train_ds,batch_size=BATCH_SIZE,
          validation_data=val_ds,
          verbose=1,
          epochs=50,
)
```

```
Epoch 1/50
54/54 [==============================] - 96s 2s/step - loss: 2.2730e-05 - accuracy: 1.0000 - val_loss: 0.0499 - val_accuracy: 0.9896
Epoch 2/50
54/54 [==============================] - 92s 2s/step - loss: 2.1047e-05 - accuracy: 1.0000 - val_loss: 0.0499 - val_accuracy: 0.9896
Epoch 3/50
54/54 [==============================] - 92s 2s/step - loss: 1.9533e-05 - accuracy: 1.0000 - val_loss: 0.0510 - val_accuracy: 0.9896
Epoch 4/50
54/54 [==============================] - 101s 2s/step - loss: 1.8018e-05 - accuracy: 1.0000 - val_loss: 0.0502 - val_accuracy: 0.9896
Epoch 5/50
54/54 [==============================] - 98s 2s/step - loss: 1.6284e-05 - accuracy: 1.0000 - val_loss: 0.0505 - val_accuracy: 0.9896
Epoch 6/50
54/54 [==============================] - 98s 2s/step - loss: 1.4687e-05 - accuracy: 1.0000 - val_loss: 0.0508 - val_accuracy: 0.9896
Epoch 7/50
54/54 [==============================] - 98s 2s/step - loss: 1.3467e-05 - accuracy: 1.0000 - val_loss: 0.0506 - val_accuracy: 0.9896
Epoch 8/50
54/54 [==============================] - 87s 2s/step - loss: 1.2248e-05 - accuracy: 1.0000 - val_loss: 0.0504 - val_accuracy: 0.9896
Epoch 9/50
54/54 [==============================] - 74s 1s/step - loss: 1.1165e-05 - accuracy: 1.0000 - val_loss: 0.0517 - val_accuracy: 0.9896
Epoch 10/50
54/54 [==============================] - 84s 2s/step - loss: 1.0131e-05 - accuracy: 1.0000 - val_loss: 0.0515 - val_accuracy: 0.9896
Epoch 11/50
54/54 [==============================] - 92s 2s/step - loss: 9.3206e-06 - accuracy: 1.0000 - val_loss: 0.0516 - val_accuracy: 0.9896
Epoch 12/50
54/54 [==============================] - 92s 2s/step - loss: 8.4336e-06 - accuracy: 1.0000 - val_loss: 0.0521 - val_accuracy: 0.9896
Epoch 13/50
54/54 [==============================] - 92s 2s/step - loss: 7.7749e-06 - accuracy: 1.0000 - val_loss: 0.0518 - val_accuracy: 0.9896
Epoch 14/50
54/54 [==============================] - 92s 2s/step - loss: 7.2253e-06 - accuracy: 1.0000 - val_loss: 0.0531 - val_accuracy: 0.9896
Epoch 15/50
54/54 [==============================] - 92s 2s/step - loss: 6.5023e-06 - accuracy: 1.0000 - val_loss: 0.0524 - val_accuracy: 0.9896
Epoch 16/50
54/54 [==============================] - 92s 2s/step - loss: 5.9656e-06 - accuracy: 1.0000 - val_loss: 0.0537 - val_accuracy: 0.9896
Epoch 17/50
54/54 [==============================] - 92s 2s/step - loss: 5.5763e-06 - accuracy: 1.0000 - val_loss: 0.0547 - val_accuracy: 0.9896
Epoch 18/50
54/54 [==============================] - 92s 2s/step - loss: 4.9654e-06 - accuracy: 1.0000 - val_loss: 0.0549 - val_accuracy: 0.9896
Epoch 19/50
54/54 [==============================] - 92s 2s/step - loss: 4.5929e-06 - accuracy: 1.0000 - val_loss: 0.0554 - val_accuracy: 0.9896
Epoch 20/50
54/54 [==============================] - 92s 2s/step - loss: 4.3955e-06 - accuracy: 1.0000 - val_loss: 0.0554 - val_accuracy: 0.9896
Epoch 21/50
54/54 [==============================] - 92s 2s/step - loss: 3.9753e-06 - accuracy: 1.0000 - val_loss: 0.0557 - val_accuracy: 0.9896
Epoch 22/50
54/54 [==============================] - 92s 2s/step - loss: 3.7814e-06 - accuracy: 1.0000 - val_loss: 0.0559 - val_accuracy: 0.9896
Epoch 23/50
54/54 [==============================] - 92s 2s/step - loss: 3.5453e-06 - accuracy: 1.0000 - val_loss: 0.0558 - val_accuracy: 0.9896
Epoch 24/50
54/54 [==============================] - 92s 2s/step - loss: 3.3143e-06 - accuracy: 1.0000 - val_loss: 0.0578 - val_accuracy: 0.9896
Epoch 25/50
54/54 [==============================] - 92s 2s/step - loss: 3.0155e-06 - accuracy: 1.0000 - val_loss: 0.0556 - val_accuracy: 0.9896
Epoch 26/50
54/54 [==============================] - 92s 2s/step - loss: 2.9566e-06 - accuracy: 1.0000 - val_loss: 0.0575 - val_accuracy: 0.9896
Epoch 27/50
54/54 [==============================] - ETA: 0s - loss: 1.9332e-06 - accuracy: 1.0000
```

In [18]:
```
scores = model.evaluate(test_ds)
```

```
8/8 [==============================] - 1s 313ms/step - loss: 0.0243 - accuracy: 0.9961
```

In [19]:
```
scores
```

Out[19]:
```
[0.024280546427607141, 0.99609375]
```

In [20]:
```
history
```

Out[20]:
```
<tensorflow.python.keras.callbacks.History at 0x186aaa1e20>
```

In [21]:
```
history.params
```

Out[21]:
```
{'verbose': 1, 'epochs': 50, 'steps': 54}
```

In [22]:
```
type(history.history['loss'])
```

Out[22]:
```
list
```

In [23]:
```
len(history.history['loss'])
```

Out[23]:
```
50
```

In [24]:
```
acc= history.history['accuracy']
val_acc= history.history['val_accuracy']
loss= history.history['loss']
val_loss =history.history['val_loss']
```

In [25]:
```
plt.figure(figsize=(8,8))
plt.subplot(2,1,1)
plt.plot(range(50), acc, label='Training Accuracy')
plt.plot(range(50),val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.figure(figsize=(8,8))
plt.subplot(2,1,2)
plt.plot(range(50), loss, label='Training Accuracy')
plt.plot(range(50),val_loss, label='Validation Accuracy')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
```

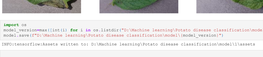
```
Text(0.5, 1.0, 'Training and Validation Loss')
```

In [26]:
```
for images_batch, labels_batch in test_ds.take(1):
    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("First image to predict")
    plt.imshow(first_image)
    print("Actual label:", class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("Predicted label:", class_names[np.argmax(batch_prediction[0])])
```

```
First image to predict
Actual label: Potato___Early_blight
predicted label: Potato___Early_blight
```



In [27]:
```
def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    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
```

In [28]:
```
plt.figure(figsize=(15,15))
for images, labels in test_ds.take(1):
    for i in range(9):
        ax=plt.subplot(3,3, i+1)
        plt.imshow(images[i].numpy().astype("uint8"))

        predicted_class, confidence =predict(model,images[i].numpy())
        actual_class = class_names[labels[i]]

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


In [29]:
```
import os
model_version=max([int(i) for i in os.listdir("C:\Machine Learning\Potato disease classification\models")] + [0])+1
model.save(f"C:\Machine Learning\Potato disease classification\models\model_version}")
```

```
INFO:tensorflow:Assets written to: C:\Machine Learning\Potato disease classification\models\1\assets
```