```python
import tensorflow as tf
from tensorflow.keras import models,layers
import matplotlib.pyplot as plt
```

```
!wget https://github.com/uditdas84/Datasets/raw/main/PlantVillage.zip
!unzip PlantVillage.zip
```

```
--2023-07-13 17:12:17--  https://github.com/uditdas84/Datasets/raw/main/PlantVillage.zip
Resolving github.com (github.com)... 140.82.112.4
Connecting to github.com (github.com)|140.82.112.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/uditdas84/Datasets/main/PlantVillage.zip [following]
--2023-07-13 17:12:17--  https://raw.githubusercontent.com/uditdas84/Datasets/main/PlantVillage.zip
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 21315583 (20M) [application/zip]
Saving to: ‘PlantVillage.zip’

PlantVillage.zip    100%[===================>]  20.33M  --.-KB/s    in 0.1s

2023-07-13 17:12:18 (155 MB/s) - ‘PlantVillage.zip’ saved [21315583/21315583]

Archive:  PlantVillage.zip
   creating: Potato___Early_blight/
  inflating: Potato___Early_blight/034959c1-f1e8-4a79-a6d5-3c1d14efa2f3___RS_Early.B 7136.JPG
  inflating: Potato___Early_blight/042135e2-e126-4900-9212-d42d900b8125___RS_Early.B 8791.JPG
  inflating: Potato___Early_blight/0604174e-3018-4faa-9975-0be32d2c0789___RS_Early.B 7123.JPG
  inflating: Potato___Early_blight/07953ca1-8935-449f-b338-4357ed683b2d___RS_Early.B 6815.JPG
  inflating: Potato___Early_blight/08029ccc-387e-4be6-9389-04f7b82fdb2a___RS_Early.B 9130.JPG
  inflating: Potato___Early_blight/08194ca3-f0b2-4aaa-8df8-5ec5ddc6696a___RS_Early.B 8151.JPG
  inflating: Potato___Early_blight/08392b44-ecc6-4f38-8566-361b552cfe21___RS_Early.B 7393.JPG
  inflating: Potato___Early_blight/107827b3-faa5-457c-97fd-3e34d2657f6b___RS_Early.B 7162.JPG
  inflating: Potato___Early_blight/1082eee1-189d-4e0f-96b5-8b1393be4c4c___RS_Early.B 8743.JPG
  inflating: Potato___Early_blight/109730cd-03f3-4139-a464-5f9151483e8c___RS_Early.B 6738.JPG
  inflating: Potato___Early_blight/1131b92e-ef46-441e-ac5f-c18ac09bf69a___RS_Early.B 8064.JPG
  inflating: Potato___Early_blight/12429fa8-02ea-4017-88ef-b2d219b892f7___RS_Early.B 6916.JPG
  inflating: Potato___Early_blight/12826416-efc5-49d3-b615-731629c95435___RS_Early.B 7215.JPG
  inflating: Potato___Early_blight/16133ed7-f960-44a5-bd03-8e665e777363___RS_Early.B 7504.JPG
  inflating: Potato___Early_blight/17520079-9d7b-481a-bc9e-676c5404d160___RS_Early.B 6774.JPG
  inflating: Potato___Early_blight/17667077-cf29-4976-9282-359c6da25cf6___RS_Early.B 6971.JPG
  inflating: Potato___Early_blight/17756ec1-9c95-43b3-bedc-933b6e0887f3___RS_Early.B 8251.JPG
  inflating: Potato___Early_blight/17848019-6609-4cc9-b27b-c70b296ceb09___RS_Early.B 7049.JPG
  inflating: Potato___Early_blight/203357f4-1deb-42b2-99ed-32df34aa166c___RS_Early.B 6780.JPG
  inflating: Potato___Early_blight/20421747-c083-48a1-aed5-b1097ae50491___RS_Early.B 8203.JPG
  inflating: Potato___Early_blight/20978e65-f4d0-419a-bab9-f6042ab2d318___RS_Early.B 6868.JPG
  inflating: Potato___Early_blight/211094c5-4983-49ff-a92e-b992039bd048___RS_Early.B 6927.JPG
  inflating: Potato___Early_blight/23546e04-7151-4dbd-95a1-687d963eb132___RS_Early.B 7402.JPG
  inflating: Potato___Early_blight/25642761-2905-48af-b8da-1064a0f6b876___RS_Early.B 6831.JPG
  inflating: Potato___Early_blight/25703c4f-ec40-4099-b249-a4fd07b07752___RS_Early.B 7040.JPG
```

```
inflating: Potato___Early_blight/283134dd-8b32-447e-8e82-547d3b69f4d4___RS_Early.B 7494.JPG
inflating: Potato___Early_blight/29386668-721c-4359-99a6-734f1a4b096b___RS_Early.B 8362.JPG
inflating: Potato___Early_blight/29508e75-1a9d-4838-b929-cf42f8638e83___RS_Early.B 6782.JPG
inflating: Potato___Early_blight/29922d76-0eda-4e7c-89af-7688c656bfdd___RS_Early.B 8353.JPG
inflating: Potato___Early_blight/29978e78-7d4a-4fff-a659-52e45e9b96b3___RS_Early.B 7672.JPG
inflating: Potato___Early_blight/31290247-3f4f-445d-8bad-20d7dccbf979___RS_Early.B 7019.JPG
inflating: Potato___Early_blight/33019904-ac3b-4083-a192-ce4092758ddd___RS_Early.B 8344.JPG
inflating: Potato___Early_blight/349730da-a627-4da6-90d5-707c4a3dba88___RS_Early.B 7553.JPG
inflating: Potato___Early_blight/357426c8-5b7b-4d56-9cb0-13cfaecc219f___RS_Early.B 7326.JPG
inflating: Potato___Early_blight/36548ca6-33b3-4a74-9b2a-52eba7aee9a3___RS_Early.B 9205.JPG
inflating: Potato___Early_blight/3682433d-fb0a-495b-a6e8-d753542b042b___RS_Early.B 8189.JPG
inflating: Potato___Early_blight/37471260-d7b4-4ccd-901e-974332ef2eb9___RS_Early.B 7967.JPG
inflating: Potato___Early_blight/37957a08-5b06-49f7-9973-29f167dd95b8___RS_Early.B 8758.JPG
inflating: Potato___Early_blight/38757e70-6278-4961-b1f3-19fccbb085f5___RS_Early.B 6968.JPG
inflating: Potato___Early_blight/38807f81-ce49-4978-916a-a6cd2a95bfb0___RS_Early.B 8397.JPG
```

```python
image_size=256
batch_size=12
channal=3
epochs=20
```

```python
import os
os.mkdir("PlantVillage")
```

```python
import shutil

# Source path
sources = ["/content/Potato___Early_blight",
          "/content/Potato___Late_blight",
          "/content/Potato___healthy"]

# Destination path
destination = "/content/PlantVillage"

# Move the content of
# source to destination
for source in sources:
  dest = shutil.move(source, destination)

# print(dest) prints the
# Destination of moved directory
```

```python
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "PlantVillage",
    shuffle=True,
    image_size=(image_size,image_size),
    batch_size = batch_size
)
```

```
Found 1152 files belonging to 3 classes.
```

```python
class_names = dataset.class_names
class_names
```

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

```python
len(dataset)
```

```
96
```

```python
dataset
```

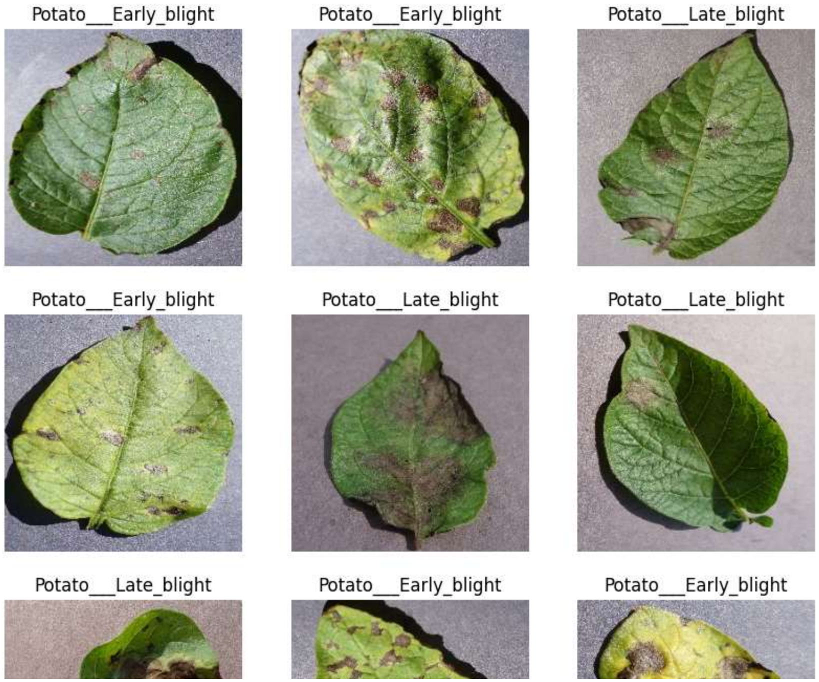
```
<_BatchDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(shape=(None,), dtype=tf.int32, name=None))>
```

```python
for image_batch, label_batch in dataset.take(1):
    print(image_batch[0])
    print(label_batch[0])
```

```
tf.Tensor(
[[[165. 168. 185.]
  [165. 168. 185.]
  [171. 174. 191.]
  ...
  [ 89.  86. 105.]
  [ 77.  74.  93.]
  [121. 118. 137.]]

 [[166. 169. 186.]
  [163. 166. 183.]
  [165. 168. 185.]
  ...
  [ 99.  96. 115.]
  [160. 157. 176.]
  [101.  98. 117.]]

 [[171. 174. 191.]
  [166. 169. 186.]
  [166. 169. 186.]
  ...
  [134. 131. 150.]
```

```
  [106. 103. 122.]
  [123. 120. 139.]]

 ...

 [[145. 146. 164.]
  [147. 148. 166.]
  [152. 153. 171.]
  ...
  [121. 116. 136.]
  [149. 144. 164.]
  [ 97.  92. 112.]]

 [[151. 152. 170.]
  [147. 148. 166.]
  [146. 147. 165.]
  ...
  [108. 103. 123.]
  [131. 126. 146.]
  [124. 119. 139.]]

 [[175. 176. 194.]
  [164. 165. 183.]
  [156. 157. 175.]
  ...
  [117. 112. 132.]
  [118. 113. 133.]
  [106. 101. 121.]]], shape=(256, 256, 3), dtype=float32)
tf.Tensor(0, shape=(), dtype=int32)
```

```
plt.figure(figsize=(10,10))
for image_batch, label_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[label_batch[i]])
        plt.axis("off")
```




```
# 80% ==> training data
# 10% ==> validation
# 10 ==> test

train_size = 0.8
train_size= len(dataset)*train_size
train_size=int(train_size)
```

```
train_ds = dataset.take(train_size)
```

```
test_ds =  dataset.skip(train_size)
len(test_ds)
```

```
20
```

```
val_size = 0.1
val_size = int(len(dataset)*val_size)
val_size
```

```
9
```

```
val_ds = test_ds.take(val_size)
test_ds = test_ds.skip(val_size)
```

```
def get_dataset_split_tf(ds,train_split=0.8,val_split=0.1,test_split=0.1,shuffle=True,shuffle_size=1000):
    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
```

```
train_ds, val_ds, test_ds = get_dataset_split_tf(dataset)
```

```
len(train_ds)
```

```
76
```

```
len(test_ds)
```

```
11
```

```
len(val_ds)
```

```
9
```

```
train_ds= train_ds.cache().shuffle(1000).prefetch(buffer_size = tf.data.AUTOTUNE)
cal_ds= val_ds.cache().shuffle(1000).prefetch(buffer_size = tf.data.AUTOTUNE)
test_ds= test_ds.cache().shuffle(1000).prefetch(buffer_size = tf.data.AUTOTUNE)
```

```
resize_and_rescale= tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(256,256),
```

```
    layers.experimental.preprocessing.Rescaling(1.0/255)
])
```

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

Model Training

```
input_shape = (batch_size,image_size, image_size,channal)
n_classes = 3

model = models.Sequential([
    resize_and_rescale,
    data_augmentation,
    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, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (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)
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (12, 256, 256, 3)         0

 sequential_1 (Sequential)   (12, 256, 256, 3)         0

 conv2d (Conv2D)             (12, 254, 254, 32)        896
```

```
 max_pooling2d (MaxPooling2D  (12, 127, 127, 32)       0
 )

 conv2d_1 (Conv2D)           (12, 125, 125, 64)        18496

 max_pooling2d_1 (MaxPooling  (12, 62, 62, 64)         0
 2D)

 conv2d_2 (Conv2D)           (12, 60, 60, 64)          36928

 max_pooling2d_2 (MaxPooling  (12, 30, 30, 64)         0
 2D)

 conv2d_3 (Conv2D)           (12, 28, 28, 64)          36928

 max_pooling2d_3 (MaxPooling  (12, 14, 14, 64)         0
 2D)

 conv2d_4 (Conv2D)           (12, 12, 12, 64)          36928

 max_pooling2d_4 (MaxPooling  (12, 6, 6, 64)           0
 2D)

 conv2d_5 (Conv2D)           (12, 4, 4, 64)            36928

 max_pooling2d_5 (MaxPooling  (12, 2, 2, 64)           0
 2D)

 flatten (Flatten)           (12, 256)                 0

 dense (Dense)               (12, 64)                  16448

 dense_1 (Dense)             (12, 3)                   195

=================================================================
Total params: 183,747
Trainable params: 183,747
Non-trainable params: 0
_________________________________________________________________
```

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

```
history = model.fit(
    train_ds,
    batch_size=batch_size,
    validation_data=val_ds,
```

```
    verbose=1,
    epochs=20,
)
```

```
Epoch 1/20
76/76 [==============================] - 24s 61ms/step - loss: 0.9820 - accuracy: 0.4879 - val_loss: 0.8429 - val_accuracy: 0.5278
Epoch 2/20
76/76 [==============================] - 4s 56ms/step - loss: 0.6700 - accuracy: 0.6985 - val_loss: 0.5356 - val_accuracy: 0.7407
Epoch 3/20
76/76 [==============================] - 3s 44ms/step - loss: 0.5038 - accuracy: 0.7851 - val_loss: 0.5032 - val_accuracy: 0.8333
Epoch 4/20
76/76 [==============================] - 3s 42ms/step - loss: 0.3743 - accuracy: 0.8575 - val_loss: 0.3419 - val_accuracy: 0.8611
Epoch 5/20
76/76 [==============================] - 4s 49ms/step - loss: 0.3151 - accuracy: 0.8695 - val_loss: 0.3610 - val_accuracy: 0.8611
Epoch 6/20
76/76 [==============================] - 3s 43ms/step - loss: 0.3242 - accuracy: 0.8673 - val_loss: 0.4001 - val_accuracy: 0.8426
Epoch 7/20
76/76 [==============================] - 3s 42ms/step - loss: 0.2756 - accuracy: 0.8882 - val_loss: 0.3198 - val_accuracy: 0.8796
Epoch 8/20
76/76 [==============================] - 4s 52ms/step - loss: 0.3047 - accuracy: 0.8772 - val_loss: 0.3169 - val_accuracy: 0.8333
Epoch 9/20
76/76 [==============================] - 3s 40ms/step - loss: 0.2511 - accuracy: 0.9013 - val_loss: 0.1647 - val_accuracy: 0.9352
Epoch 10/20
76/76 [==============================] - 3s 42ms/step - loss: 0.2543 - accuracy: 0.8904 - val_loss: 0.3072 - val_accuracy: 0.8611
Epoch 11/20
76/76 [==============================] - 4s 54ms/step - loss: 0.2622 - accuracy: 0.8925 - val_loss: 0.2246 - val_accuracy: 0.9167
Epoch 12/20
76/76 [==============================] - 3s 43ms/step - loss: 0.2424 - accuracy: 0.8991 - val_loss: 0.2619 - val_accuracy: 0.8889
Epoch 13/20
76/76 [==============================] - 3s 43ms/step - loss: 0.1852 - accuracy: 0.9232 - val_loss: 0.4155 - val_accuracy: 0.8241
Epoch 14/20
76/76 [==============================] - 3s 44ms/step - loss: 0.2276 - accuracy: 0.9101 - val_loss: 0.1347 - val_accuracy: 0.9722
Epoch 15/20
76/76 [==============================] - 5s 62ms/step - loss: 0.2377 - accuracy: 0.9024 - val_loss: 0.2312 - val_accuracy: 0.9259
Epoch 16/20
76/76 [==============================] - 3s 43ms/step - loss: 0.2344 - accuracy: 0.9035 - val_loss: 0.2371 - val_accuracy: 0.9167
Epoch 17/20
76/76 [==============================] - 3s 40ms/step - loss: 0.2372 - accuracy: 0.9090 - val_loss: 0.3107 - val_accuracy: 0.8981
Epoch 18/20
76/76 [==============================] - 4s 56ms/step - loss: 0.2415 - accuracy: 0.9090 - val_loss: 0.3004 - val_accuracy: 0.8704
Epoch 19/20
76/76 [==============================] - 3s 43ms/step - loss: 0.2288 - accuracy: 0.9123 - val_loss: 0.1904 - val_accuracy: 0.9352
Epoch 20/20
76/76 [==============================] - 3s 40ms/step - loss: 0.2069 - accuracy: 0.9112 - val_loss: 0.1093 - val_accuracy: 0.9352
```

Model Evaluation

```
scores = model.evaluate(test_ds)
```

```
11/11 [==============================] - 1s 13ms/step - loss: 0.1653 - accuracy: 0.9545
```

```
scores
```

```
[0.16530056297779083, 0.9545454382896423]
```

```
history.params
```

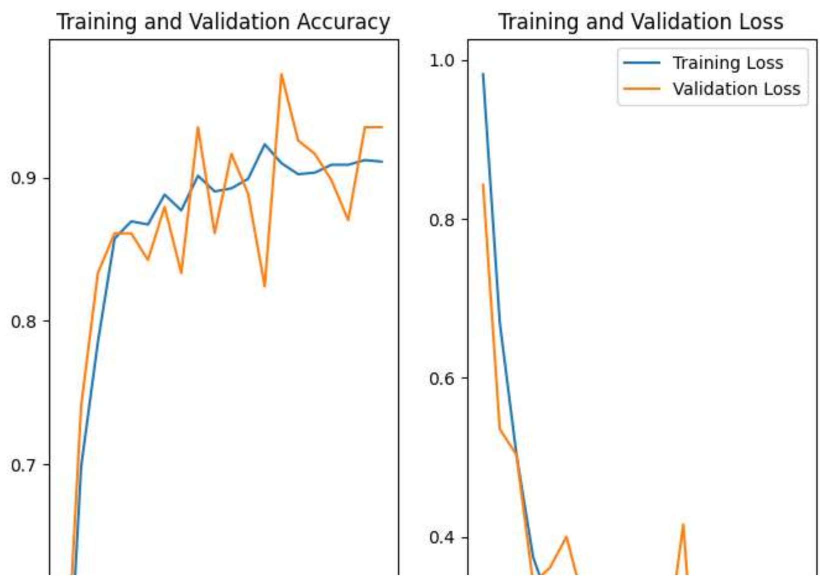
```
{'verbose': 1, 'epochs': 20, 'steps': 76}
```

```
history.history.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
```

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

plt.subplot(1, 2, 2)
plt.plot(range(epochs), loss, label='Training Loss')
plt.plot(range(epochs), val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Model Prediction

```
import numpy as np
```

```
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("actual label: ", class_names[first_label])


  print("first image to predict")
  plt.imshow(first_image)
  pred_img = model.predict(images_batch)
  print(class_names[np.argmax(pred_img[0])])
  # print(pred_img[0])

```

```
actual label:  Potato___Early_blight
first image to predict
1/1 [==============================] - 0s 27ms/step
Potato___Early_blight
```


```
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
```

```
plt.figure(figsize=(10,13))
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}\nPredicted:{predicted_class}\nConfidence:{confidence}")
    plt.axis("off")
```

```
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
```

Actual:Potato___Early_blight
Predicted:Potato___Early_blight
Confidence:99.98

Actual:Potato___Early_blight
Predicted:Potato___Early_blight
Confidence:99.87

Actual:Potato___Late_blight
Predicted:Potato___Late_blight
Confidence:99.97

Actual:Potato___Late_blight
Predicted:Potato___Late_blight
Confidence:99.98

Actual:Potato___Early_blight
Predicted:Potato___Early_blight
Confidence:83.09

Actual:Potato___Late_blight
Predicted:Potato___Early_blight
Confidence:93.63

Actual:Potato___healthy
Predicted:Potato___healthy
Confidence:99.07

Actual:Potato___Early_blight
Predicted:Potato___Early_blight
Confidence:78.98

Actual:Potato___Late_blight
Predicted:Potato___Late_blight
Confidence:99.69

```
# import os
# os.mkdir("models")
```

```
!pwd
```

```
/content
```

```

model_version=1
model.save(f"/content/models/{model_version}")
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_co
```

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

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

```
%cd /content/models
```

```
/content/models
```

```
!zip -r 1.zip 1/
```

```
  adding: 1/ (stored 0%)
  adding: 1/fingerprint.pb (stored 0%)
  adding: 1/variables/ (stored 0%)
  adding: 1/variables/variables.data-00000-of-00001 (deflated 12%)
  adding: 1/variables/variables.index (deflated 68%)
  adding: 1/keras_metadata.pb (deflated 94%)
  adding: 1/assets/ (stored 0%)
  adding: 1/saved_model.pb (deflated 89%)
```

```
from google.colab import files
files.download('1.zip')
```