```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
from IPython.display import HTML
```

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
BATCH_SIZE = 32
IMAGE_SIZE = 256
CHANNELS = 3
EPOCHS = 50
```

```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "drive/MyDrive/Dataset",
    seed=123,
    shuffle=True,
    image_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE
)
```

```
Found 881 files belonging to 3 classes.
```

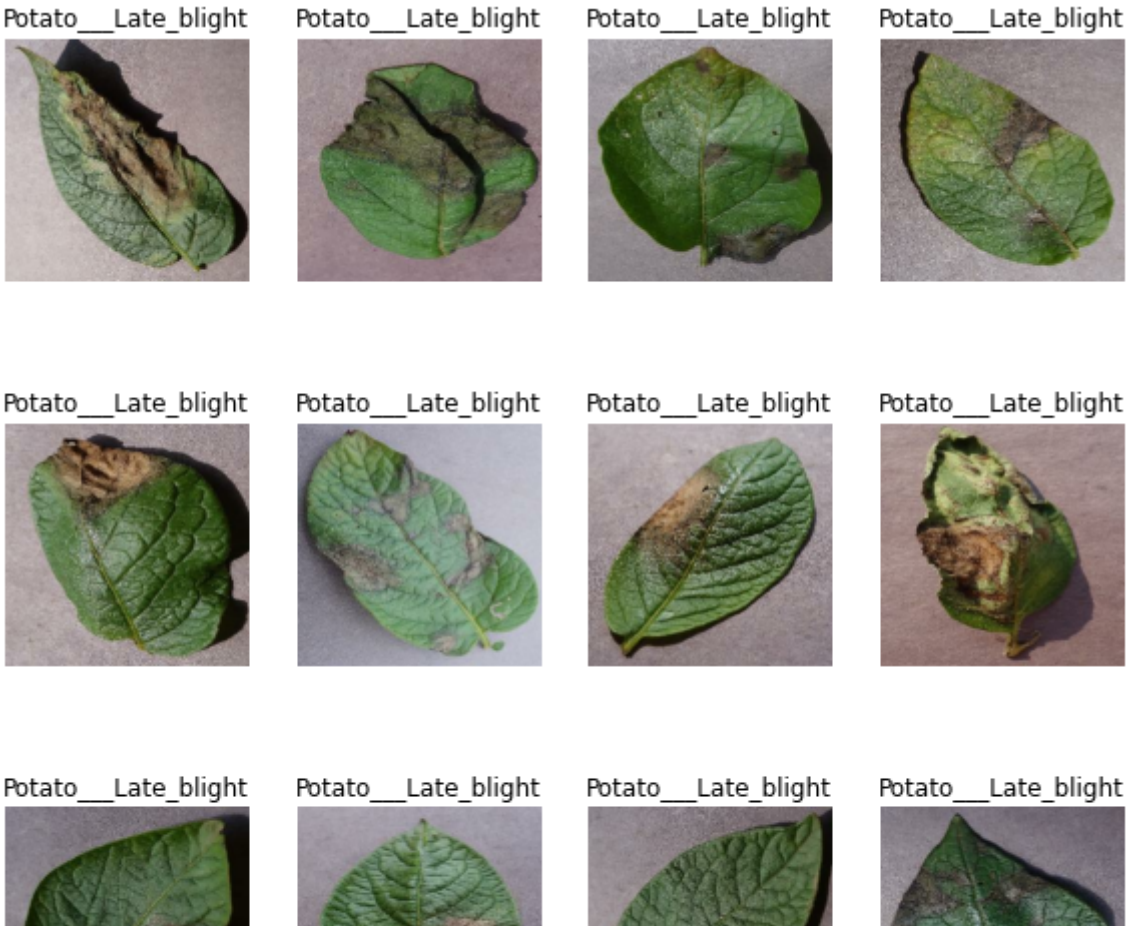
```
class_names = dataset.class_names
class_names
```

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

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

```
(32, 256, 256, 3)
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

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




```
len(dataset)
```

```
28
```

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

```
22.400000000000002
```

```
train_ds = dataset.take(54)
len(train_ds)
```

```
28
```

```
test_ds = dataset.skip(54)
len(test_ds)
```

```
0
```

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

```
2.8000000000000003
```

```
val_ds = test_ds.take(6)
len(val_ds)
```

```
0
```

```
test_ds = test_ds.skip(6)
len(test_ds)
```

```
0
```

```
def get_dataset_partitions_tf(ds, train_split=0.8, val_split=0.1, test_split=0.1, shuffle=
    assert (train_split + test_split + val_split) == 1

    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_partitions_tf(dataset)
```

```
len(train_ds)
```

```
22
```

```
len(val_ds)
```

```
2
```

```
len(test_ds)
```

```
4
```

```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_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)
```

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

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

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

```
train_ds = train_ds.map(
    lambda x, y: (data_augmentation(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
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, (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)      (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
_________________________________________________________________
```

```
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=50,
)
```

```
Epoch 22/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 22/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 23/50
```

```
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 24/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 25/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 26/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 27/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 28/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 29/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 30/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 31/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 32/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 33/50
22/22 [==============================] - 72s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 34/50
22/22 [==============================] - 72s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 35/50
22/22 [==============================] - 72s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 36/50
22/22 [==============================] - 72s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 37/50
22/22 [==============================] - 72s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 38/50
22/22 [==============================] - 74s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 39/50
22/22 [==============================] - 72s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 40/50
22/22 [==============================] - 72s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 41/50
22/22 [==============================] - 72s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 42/50
22/22 [==============================] - 72s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 43/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 44/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 45/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 46/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 47/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 48/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 49/50
22/22 [==============================] - 71s 3s/step - loss: 0.0000e+00 - accuracy
Epoch 50/50
```

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

```
4/4 [==============================] - 3s 776ms/step - loss: 0.0000e+00 - accuracy: 1
```

```
scores
```

```
[0.0, 1.0]
```

```
history
```

```
<keras.callbacks.History at 0x7efbeb67c750>
```

```
history.params
```

```
{'epochs': 50, 'steps': 22, 'verbose': 1}
```

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

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

```
type(history.history['loss'])
```

```
list
```

```
len(history.history['loss'])
```

```
50
```

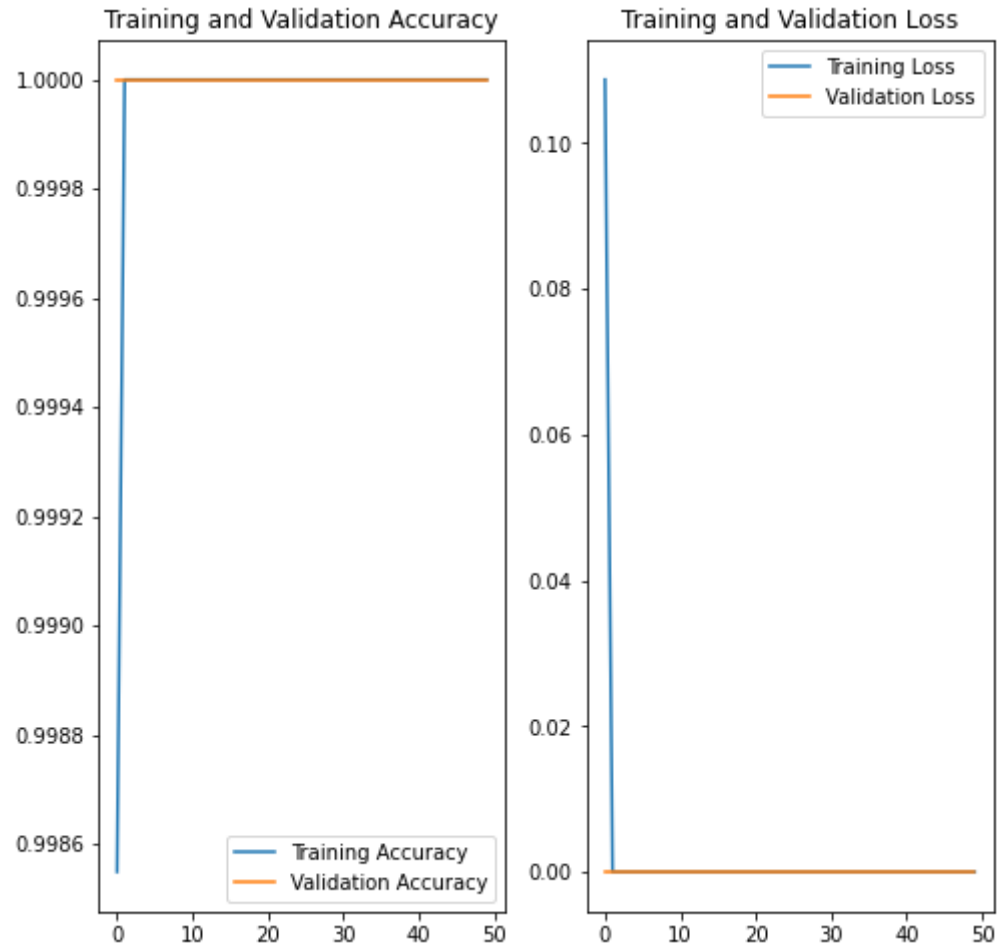
```
history.history['loss'][:5]
```

```
[0.10873198509216309, 0.0, 0.0, 0.0, 0.0]
```

```
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()
```

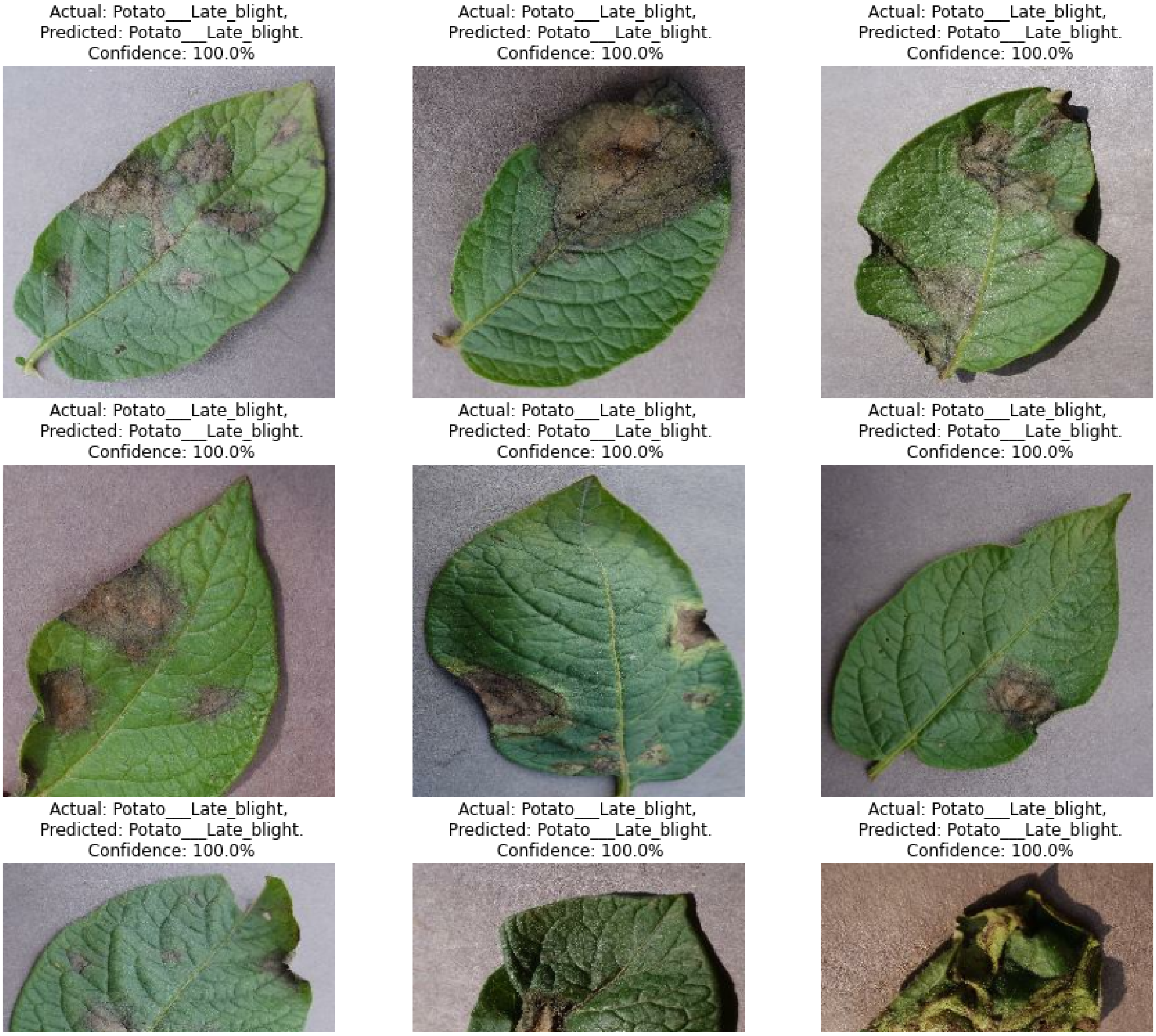



```
import numpy as np
for images_batch, labels_batch in test_ds.take(1):

    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("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___Late_blight
predicted label: Potato___Late_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=(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: {confi

        plt.axis("off")
```




```
import os
model_version = max([int(i) for i in os.listdir("../models") + [0]])+1
model.save(f"../models/{model_version}")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-44-3408744e39da> in <module>()
      1 import os
----> 2 model_version = max([int(i) for i in os.listdir("../models") + [0]])+1
      3 model.save(f"../models/{model_version}")

FileNotFoundError: [Errno 2] No such file or directory: '../models'
```

SEARCH STACK OVERFLOW

```
model.save("potatoes.h5")
```

0s completed at 8:25 PM