

```

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
from tensorflow.keras import utils
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D, AveragePooling2D

imagesize = 256 ,batchsize = 32 ,epoch = 50

dataset = tf.keras.utils.image_dataset_from_directory(
    '/content/drive/MyDrive/PlantVillage',
    labels='inferred',
    label_mode='int',
    class_names=None,
    color_mode='rgb',
    batch_size=batchsize,
    image_size=(imagesize,imagesize),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation='bilinear',
    follow_links=False,
    crop_to_aspect_ratio=False,)

    Found 2152 files belonging to 3 classes.

class_names= dataset.class_names
class_names

    ['Potato__Early_blight', 'Potato__Late_blight', 'Potato__healthy']

len(dataset)

    68

plt.figure(figsize=(10,10))
for image_batch, label_batch in dataset.take(1):
    for i in range(12):
        plt.subplot(3,4,i+1)
        # print(image_batch.shape)
        # print(image_batch[0].numpy())
        # print(label_batch.numpy())
        plt.imshow(image_batch[i].numpy().astype('uint8'))
        plt.axis('off')
        plt.title(class_names[label_batch[i]])

```

Potato__Early_blight



Potato__Late_blight



Potato__Early_blight



Potato__Early_blight



Potato__Late_blight



Potato__Late_blight



Potato__Early_blight



Potato__Late_blight



```
def get_dataset_partitions(dataset, train_size = 0.8, val_size = 0.1,
                           shuffle = True, shuffle_size = 10000):
    if shuffle:
        dataset = dataset.shuffle(shuffle_size, seed=12)

    train_data = int(len(dataset)*train_size)
    train_ds=dataset.take(train_data)

    test_ds= dataset.skip(train_data)

    val_data = int(len(dataset)*val_size)
    val_ds= dataset.take(val_data)

    test_ds= test_ds.skip(val_data)

    return train_ds, val_ds, test_ds
```

   

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

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

resize_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Rescaling(1.0/255),
    layers.experimental.preprocessing.Resizing(imagesize, imagesize)
])

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

model = Sequential()

model.add(resize_rescale)
model.add(data_augmentation)
model.add(Conv2D(32, kernel_size=(3,3),strides=(1,1), padding='valid',
                  activation='relu', input_shape= (imagesize,imagesize,3)))

model.add(MaxPooling2D(pool_size=(2,2), strides=1,padding='valid'))

model.add(Conv2D(64, kernel_size=(3,3),strides=(1,1), padding='valid',
                  activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=1,padding='valid'))

model.add(Conv2D(64, kernel_size=(3,3),strides=(1,1), padding='valid',
                  activation='relu', input_shape= (imagesize,imagesize,3)))

model.add(MaxPooling2D(pool_size=(2,2), strides=1,padding='valid'))
```

```
model.add(Conv2D(64, kernel_size=(3,3),strides=(1,1), padding='valid',
                activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=1,padding='valid'))

model.add(Conv2D(64, kernel_size=(3,3),strides=(1,1), padding='valid',
                activation='relu', input_shape= (imagesize,imagesize,3)))
model.add(MaxPooling2D(pool_size=(2,2), strides=1,padding='valid'))

model.add(Conv2D(16, kernel_size=(3,3),strides=(1,1), padding='valid',
                activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2), strides=1,padding='valid'))

model.add(Flatten())

model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='softmax'))

model.build(input_shape = (batchsize,imagesize,imagesize,3))

model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
=====		
sequential (Sequential)	(32, 256, 256, 3)	0
sequential_1 (Sequential)	(32, 256, 256, 3)	0
conv2d (Conv2D)	(32, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(32, 253, 253, 32)	0
conv2d_1 (Conv2D)	(32, 251, 251, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(32, 250, 250, 64)	0
conv2d_2 (Conv2D)	(32, 248, 248, 64)	36928
max_pooling2d_2 (MaxPooling2D)	(32, 247, 247, 64)	0
conv2d_3 (Conv2D)	(32, 245, 245, 64)	36928
max_pooling2d_3 (MaxPooling2D)	(32, 244, 244, 64)	0
conv2d_4 (Conv2D)	(32, 242, 242, 64)	36928
max_pooling2d_4 (MaxPooling2D)	(32, 241, 241, 64)	0
conv2d_5 (Conv2D)	(32, 239, 239, 16)	9232
max_pooling2d_5 (MaxPooling2D)	(32, 238, 238, 16)	0
flatten (Flatten)	(32, 906304)	0
dense (Dense)	(32, 64)	58003520
dense_1 (Dense)	(32, 64)	4160
=====		
Total params: 58,147,088		
Trainable params: 58,147,088		
Non-trainable params: 0		

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

```
history=model.fit(train_ds,
                  epochs = 100,
                  batch_size = batchsize,
                  verbose = 1,
                  validation_data = val_ds,
                  )

Epoch 72/100
54/54 [=====] - 30s 560ms/step - loss: 0.0918 - accuracy: 0.9641 - val_loss: 0.1310 - val_accuracy: 0.9688
Epoch 73/100
54/54 [=====] - 30s 560ms/step - loss: 0.1827 - accuracy: 0.9346 - val_loss: 0.1436 - val_accuracy: 0.9583
Epoch 74/100
54/54 [=====] - 30s 562ms/step - loss: 0.1057 - accuracy: 0.9578 - val_loss: 0.2154 - val_accuracy: 0.9219
Epoch 75/100
54/54 [=====] - 30s 557ms/step - loss: 0.1171 - accuracy: 0.9595 - val_loss: 0.1036 - val_accuracy: 0.9688
Epoch 76/100
54/54 [=====] - 30s 556ms/step - loss: 0.0845 - accuracy: 0.9705 - val_loss: 0.1738 - val_accuracy: 0.9479
Epoch 77/100
54/54 [=====] - 30s 558ms/step - loss: 0.0908 - accuracy: 0.9676 - val_loss: 0.1582 - val_accuracy: 0.9635
Epoch 78/100
54/54 [=====] - 30s 563ms/step - loss: 0.0923 - accuracy: 0.9688 - val_loss: 0.1815 - val_accuracy: 0.9375
Epoch 79/100
54/54 [=====] - 30s 555ms/step - loss: 0.1304 - accuracy: 0.9589 - val_loss: 0.4998 - val_accuracy: 0.8958
Epoch 80/100
54/54 [=====] - 30s 560ms/step - loss: 0.1621 - accuracy: 0.9456 - val_loss: 0.1601 - val_accuracy: 0.9583
Epoch 81/100
54/54 [=====] - 30s 553ms/step - loss: 0.1177 - accuracy: 0.9601 - val_loss: 0.1239 - val_accuracy: 0.9583
Epoch 82/100
54/54 [=====] - 30s 561ms/step - loss: 0.1149 - accuracy: 0.9554 - val_loss: 0.1675 - val_accuracy: 0.9583
Epoch 83/100
54/54 [=====] - 30s 556ms/step - loss: 0.1227 - accuracy: 0.9560 - val_loss: 0.1799 - val_accuracy: 0.9531
Epoch 84/100
54/54 [=====] - 30s 555ms/step - loss: 0.1498 - accuracy: 0.9485 - val_loss: 0.1567 - val_accuracy: 0.9375
Epoch 85/100
54/54 [=====] - 30s 555ms/step - loss: 0.1333 - accuracy: 0.9462 - val_loss: 0.1965 - val_accuracy: 0.9167
Epoch 86/100
54/54 [=====] - 30s 555ms/step - loss: 0.1010 - accuracy: 0.9670 - val_loss: 0.1285 - val_accuracy: 0.9635
Epoch 87/100
54/54 [=====] - 30s 556ms/step - loss: 0.0812 - accuracy: 0.9722 - val_loss: 0.1789 - val_accuracy: 0.9323
Epoch 88/100
54/54 [=====] - 30s 562ms/step - loss: 0.0830 - accuracy: 0.9705 - val_loss: 0.1326 - val_accuracy: 0.9531
Epoch 89/100
54/54 [=====] - 30s 556ms/step - loss: 0.0800 - accuracy: 0.9722 - val_loss: 0.1365 - val_accuracy: 0.9740
Epoch 90/100
54/54 [=====] - 30s 555ms/step - loss: 0.1216 - accuracy: 0.9543 - val_loss: 0.1579 - val_accuracy: 0.9323
Epoch 91/100
54/54 [=====] - 30s 555ms/step - loss: 0.0828 - accuracy: 0.9751 - val_loss: 0.0968 - val_accuracy: 0.9740
Epoch 92/100
54/54 [=====] - 30s 556ms/step - loss: 0.0668 - accuracy: 0.9751 - val_loss: 0.1331 - val_accuracy: 0.9688
Epoch 93/100
54/54 [=====] - 30s 554ms/step - loss: 0.0737 - accuracy: 0.9751 - val_loss: 0.1482 - val_accuracy: 0.9583
Epoch 94/100
54/54 [=====] - 30s 556ms/step - loss: 0.1065 - accuracy: 0.9606 - val_loss: 0.1860 - val_accuracy: 0.9635
Epoch 95/100
54/54 [=====] - 30s 556ms/step - loss: 0.0795 - accuracy: 0.9711 - val_loss: 0.1233 - val_accuracy: 0.9583
Epoch 96/100
54/54 [=====] - 30s 555ms/step - loss: 0.1394 - accuracy: 0.9549 - val_loss: 0.1161 - val_accuracy: 0.9688
Epoch 97/100
54/54 [=====] - 30s 562ms/step - loss: 0.0875 - accuracy: 0.9676 - val_loss: 0.1019 - val_accuracy: 0.9688
Epoch 98/100
54/54 [=====] - 30s 563ms/step - loss: 0.0771 - accuracy: 0.9693 - val_loss: 0.0998 - val_accuracy: 0.9740
Epoch 99/100
54/54 [=====] - 30s 554ms/step - loss: 0.0730 - accuracy: 0.9734 - val_loss: 0.1807 - val_accuracy: 0.9427
Epoch 100/100
54/54 [=====] - 30s 553ms/step - loss: 0.0850 - accuracy: 0.9728 - val_loss: 0.1415 - val_accuracy: 0.9688

model.evaluate(test_ds)

8/8 [=====] - 8s 153ms/step - loss: 0.0720 - accuracy: 0.9727
[0.0719895213842392, 0.97265625]
```

✓ 8s completed at 02:43

