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

```
     Mounted at /content/gdrive
```

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#import utils
import os
%matplotlib inline


from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Input, Dropout,Flatten, Conv2D
from tensorflow.keras.layers import BatchNormalization, Activation, MaxPooling2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
#from tensorflow.keras.utils import plot_model

from IPython.display import SVG, Image
#from livelossplot import PlotLossesTensorFlowKeras()
import tensorflow as tf
print("Tensorflow version:", tf.__version__)
```

```
     Tensorflow version: 2.9.2
```

```
base_dir = 'gdrive/MyDrive/InsectsData-main/InsectsData-main/tmp/'
train_dir = os.path.join(base_dir,'train')
test_dir = os.path.join(base_dir,'test')
print(test_dir)
```

```
     gdrive/MyDrive/InsectsData-main/InsectsData-main/tmp/test
```

```
img_size = 64
batch_size = 64
datagen_train = ImageDataGenerator(horizontal_flip=True)
train_generator = datagen_train.flow_from_directory(train_dir,
                                                    target_size = (img_size, img_size),
                                                    color_mode='grayscale',
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    shuffle=True)
datagen_validation = ImageDataGenerator(horizontal_flip=True)
validation_generator = datagen_validation.flow_from_directory(test_dir,
                                                    target_size = (img_size, img_size),
                                                    color_mode='grayscale',
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    shuffle=False)
```

```
     Found 1493 images belonging to 3 classes.
     Found 753 images belonging to 3 classes.
```

```
from tensorflow.keras.metrics import CategoricalAccuracy
from sklearn.metrics import f1_score, precision_score, recall_score, confusion_matrix
model = Sequential()

#1 - conv

model.add(Conv2D(64,(3,3),padding='same',input_shape=(64,64,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

#2 - conv
model.add(Conv2D(128,(5,5),padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

#flatten
model.add(Flatten())
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(3,activation='softmax'))

opt = Adam(lr=0.0005)

model.compile(optimizer=opt,loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 64, 64, 64)        640

 batch_normalization_4 (Batc  (None, 64, 64, 64)       256
 hNormalization)

 activation_4 (Activation)   (None, 64, 64, 64)        0

 max_pooling2d_2 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 dropout_4 (Dropout)         (None, 32, 32, 64)        0

 conv2d_3 (Conv2D)           (None, 32, 32, 128)       204928

 batch_normalization_5 (Batc  (None, 32, 32, 128)      512
 hNormalization)

 activation_5 (Activation)   (None, 32, 32, 128)       0

 max_pooling2d_3 (MaxPooling  (None, 16, 16, 128)      0
 2D)

 dropout_5 (Dropout)         (None, 16, 16, 128)       0

 flatten_1 (Flatten)         (None, 32768)             0

 dense_3 (Dense)             (None, 256)               8388864

 batch_normalization_6 (Batc  (None, 256)              1024
 hNormalization)

 activation_6 (Activation)   (None, 256)               0

 dropout_6 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 512)               131584

 batch_normalization_7 (Batc  (None, 512)              2048
 hNormalization)

 activation_7 (Activation)   (None, 512)               0

 dropout_7 (Dropout)         (None, 512)               0

 dense_5 (Dense)             (None, 3)                 1539

=================================================================
Total params: 8,731,395
Trainable params: 8,729,475
Non-trainable params: 1,920
_________________________________________________________________
```

```
epochs = 15
steps_per_epoch = train_generator.n//train_generator.batch_size
validation_steps = validation_generator.n//validation_generator.batch_size

checkpoint = ModelCheckpoint('model_weights.h5',monitor='val_acc',
                            save_weights_only=True,mode='max',verbose=1)

reduce_lr = ReduceLROnPlateau(monitor='val_loss',factor=0.1,patience=2,min_lr=0.00001,mode='auto')
callbacks = [checkpoint, reduce_lr]

history = model.fit(x=train_generator,
                   steps_per_epoch=steps_per_epoch,
                   epochs=epochs,
                   validation_data=validation_generator,
                   validation_steps=validation_steps,
                    callbacks = callbacks )
```

```
Epoch 1/15
23/23 [==============================] - ETA: 0s - loss: 0.4910 - accuracy: 0.8062
Epoch 1: saving model to model_weights.h5
23/23 [==============================] - 151s 7s/step - loss: 0.4910 - accuracy: 0.8062 - val_loss: 60.5465 - val_accuracy: 0.38
Epoch 2/15
23/23 [==============================] - ETA: 0s - loss: 0.1532 - accuracy: 0.9489
Epoch 2: saving model to model_weights.h5
23/23 [==============================] - 45s 2s/step - loss: 0.1532 - accuracy: 0.9489 - val_loss: 16.8514 - val_accuracy: 0.369
Epoch 3/15
23/23 [==============================] - ETA: 0s - loss: 0.0817 - accuracy: 0.9741
Epoch 3: saving model to model_weights.h5
23/23 [==============================] - 44s 2s/step - loss: 0.0817 - accuracy: 0.9741 - val_loss: 10.1586 - val_accuracy: 0.367
Epoch 4/15
23/23 [==============================] - ETA: 0s - loss: 0.0448 - accuracy: 0.9888
Epoch 4: saving model to model_weights.h5
23/23 [==============================] - 43s 2s/step - loss: 0.0448 - accuracy: 0.9888 - val_loss: 5.0080 - val_accuracy: 0.3750
Epoch 5/15
23/23 [==============================] - ETA: 0s - loss: 0.0441 - accuracy: 0.9874
Epoch 5: saving model to model_weights.h5
23/23 [==============================] - 44s 2s/step - loss: 0.0441 - accuracy: 0.9874 - val_loss: 3.0021 - val_accuracy: 0.4830
Epoch 6/15
23/23 [==============================] - ETA: 0s - loss: 0.0288 - accuracy: 0.9923
Epoch 6: saving model to model_weights.h5
23/23 [==============================] - 43s 2s/step - loss: 0.0288 - accuracy: 0.9923 - val_loss: 4.9353 - val_accuracy: 0.4048
Epoch 7/15
23/23 [==============================] - ETA: 0s - loss: 0.0198 - accuracy: 0.9958
Epoch 7: saving model to model_weights.h5
23/23 [==============================] - 43s 2s/step - loss: 0.0198 - accuracy: 0.9958 - val_loss: 0.1757 - val_accuracy: 0.9233
Epoch 8/15
23/23 [==============================] - ETA: 0s - loss: 0.0103 - accuracy: 1.0000
Epoch 8: saving model to model_weights.h5
23/23 [==============================] - 43s 2s/step - loss: 0.0103 - accuracy: 1.0000 - val_loss: 0.0620 - val_accuracy: 0.9872
Epoch 9/15
23/23 [==============================] - ETA: 0s - loss: 0.0095 - accuracy: 0.9986
Epoch 9: saving model to model_weights.h5
23/23 [==============================] - 44s 2s/step - loss: 0.0095 - accuracy: 0.9986 - val_loss: 0.0049 - val_accuracy: 1.0000
Epoch 10/15
23/23 [==============================] - ETA: 0s - loss: 0.0086 - accuracy: 0.9979
Epoch 10: saving model to model_weights.h5
23/23 [==============================] - 43s 2s/step - loss: 0.0086 - accuracy: 0.9979 - val_loss: 0.0036 - val_accuracy: 1.0000
Epoch 11/15
23/23 [==============================] - ETA: 0s - loss: 0.0043 - accuracy: 1.0000
Epoch 11: saving model to model_weights.h5
23/23 [==============================] - 43s 2s/step - loss: 0.0043 - accuracy: 1.0000 - val_loss: 0.0037 - val_accuracy: 1.0000
Epoch 12/15
23/23 [==============================] - ETA: 0s - loss: 0.0092 - accuracy: 0.9965
Epoch 12: saving model to model_weights.h5
23/23 [==============================] - 43s 2s/step - loss: 0.0092 - accuracy: 0.9965 - val_loss: 0.2070 - val_accuracy: 0.8977
Epoch 13/15
23/23 [==============================] - ETA: 0s - loss: 0.0027 - accuracy: 1.0000
Epoch 13: saving model to model_weights.h5
23/23 [==============================] - 43s 2s/step - loss: 0.0027 - accuracy: 1.0000 - val_loss: 0.0951 - val_accuracy: 0.9545
Epoch 14/15
23/23 [==============================] - ETA: 0s - loss: 0.0050 - accuracy: 1.0000
Epoch 14: saving model to model_weights.h5
23/23 [==============================] - 43s 2s/step - loss: 0.0050 - accuracy: 1.0000 - val_loss: 0.0594 - val_accuracy: 0.9787
Epoch 15/15
```

```
#print(model.summary())
print(history.history)
```

```
{'loss': [0.4910242557525635, 0.15324920415878296, 0.08167506754398346, 0.04483981430530548, 0.04407018423080444, 0.028758831322193
```

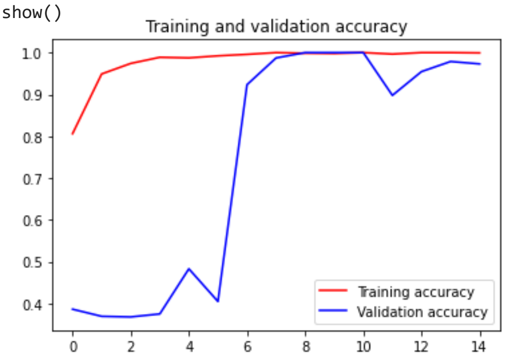
```
print(history.history.keys())
```

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

```
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()
plt.show()
```



```
<Figure size 432x288 with 0 Axes>
```

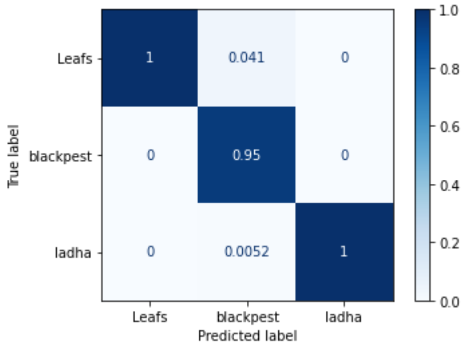
```
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix, plot_confusion_matrix

Y_pred = model.predict_generator(validation_generator)
y_pred = np.argmax(Y_pred, axis=1)
labels = ["Leafs", "blackpest", "ladha"]

cm = confusion_matrix(validation_generator.classes, y_pred, normalize='pred')
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)

disp.plot(cmap=plt.cm.Blues)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: UserWarning: `Model.predict_generator` is deprecated and wi
```



```
model.save('model_weights.h5')
```

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
import os
import cv2
import tensorflow as tf
model = tf.keras.models.load_model('model_weights.h5')
def prepare(filepath):
    IMG_SIZE = 64
    img_array = cv2.imread(filepath, cv2.IMREAD_GRAYSCALE)
    new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
    return new_array.reshape(-1, IMG_SIZE, IMG_SIZE, 1)

img_path = 'gdrive/MyDrive/InsectsData-main/InsectsData-main/dataset/sample/horti_vegetables_chilli_clip_image003.jpg'
img = image.load_img(img_path, target_size=(64, 64,1))
img_array = image.img_to_array(img)
prediction = model.predict([prepare(img_path)])
print(prediction)
```

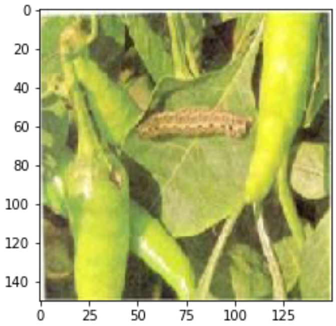
```
1/1 [==============================] - 0s 106ms/step
[[1.4317456e-04 9.9761510e-01 2.2416795e-03]]
```

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

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

```
img = image.load_img(img_path, target_size=(150, 150,1))
plt.imshow(img)
classes = {1:'ladha',2:'leaf',3:'black pest'}
classes_x=np.argmax(prediction,axis=1)
print(classes[int(classes_x)])
```

```
     ladha
```



```
print("Training set accuracy :", end = " ")

print(history.history['accuracy'][-1])

print("Test data  accuracy:", end = " ")

print(history.history['val_accuracy'][-1])
```

```
     Training set accuracy : 0.9993001818656921
     Test data  accuracy: 0.9730113744735718
```

✓ 0s completed at 5:37 PM