```
! pip install -q kaggle
```

```
from google.colab import files
```

```
files.upload()
```

Browse... kaggle.json

kaggle.json(application/json) - 74 bytes, last modified: n/a - 100% done

```
Saving kaggle.json to kaggle.json
{'kaggle.json':
[illegible]
```

```
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
```

Generate Using ... a slider using jupyter widgets Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

```
! chmod 600 ~/.kaggle/kaggle.json
```

```
! kaggle datasets download -d rizwan123456789/potato-disease-leaf-datasetpld
```

```
Downloading potato-disease-leaf-datasetpld.zip to /content
 99% 37.0M/37.4M [00:01<00:00, 36.8MB/s]
100% 37.4M/37.4M [00:01<00:00, 25.6MB/s]
```

Generate Using ... print hello world using rot13 Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

```
from zipfile import ZipFile
```

```
zf = ZipFile('potato-disease-leaf-datasetpld.zip')
zf.extractall()
zf.close()
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
train_data_gen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 0.2,
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True,
    vertical_flip = True
)

val_data_gen = ImageDataGenerator(rescale=1./255)

train_data = train_data_gen.flow_from_directory('PLD_3_Classes_256/Training',
                                                target_size = (224, 224),
                                                batch_size = 32,
                                                class_mode = 'categorical')
val_data = val_data_gen.flow_from_directory('PLD_3_Classes_256/Validation',
                                                target_size = (224, 224),
                                                batch_size = 32,
                                                class_mode = 'categorical')
test_data = val_data_gen.flow_from_directory('PLD_3_Classes_256/Testing',
                                                target_size = (224, 224),
                                                batch_size = 32,
                                                class_mode = 'categorical')
```

```
Found 3251 images belonging to 3 classes.
Found 416 images belonging to 3 classes.
Found 405 images belonging to 3 classes.
```

```
val_data[0][0].shape
```

```
(32, 224, 224, 3)
```

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPool2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import CategoricalCrossentropy
```

Generate Using ... create a dataframe with 2 columns and 10 rows Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

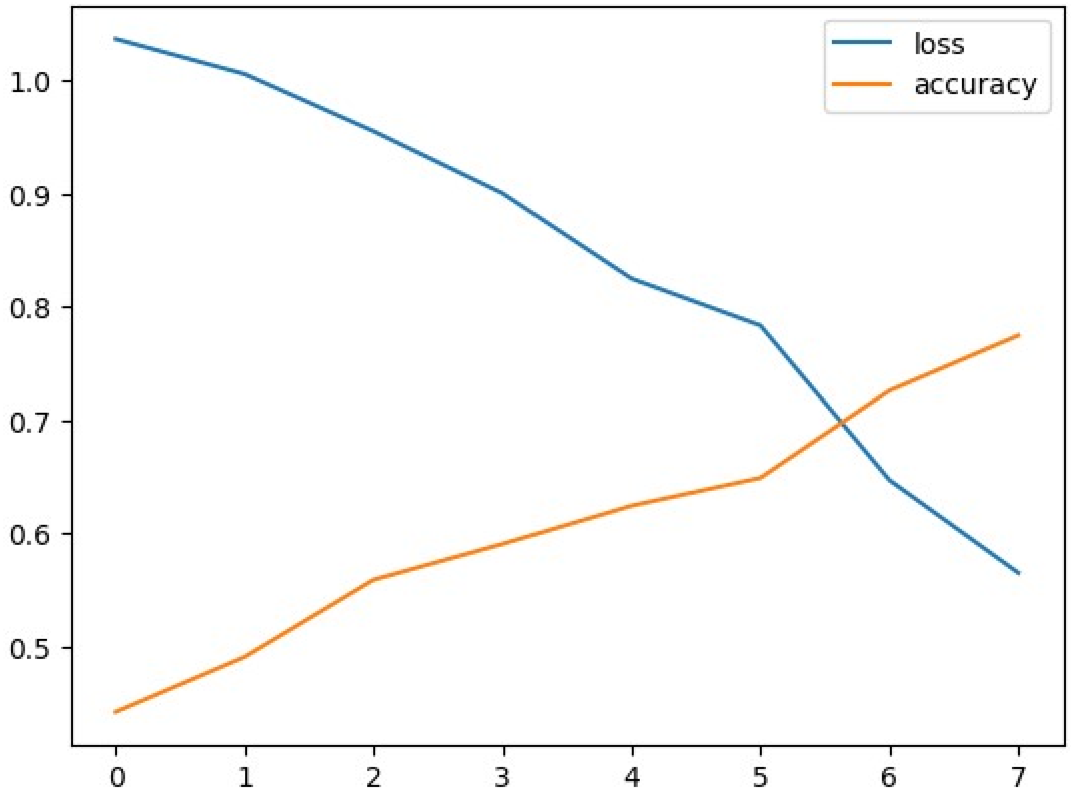
```
vgg3 = Sequential([
    Conv2D(10, 3, activation='relu', input_shape= (224, 224, 3)),
    Conv2D(10, 3, activation='relu'),
    MaxPool2D(),
    Conv2D(10, 3, activation='relu'),
    Conv2D(10, 3, activation='relu'),
    MaxPool2D(),
    Conv2D(10, 3, activation='relu'),
    Conv2D(10, 3, activation='relu'),
    Conv2D(10, 3, activation='relu'),
    MaxPool2D(),
    Flatten(),
    Dense(32, activation='relu'),
    Dense(3, activation='softmax')
])

vgg3.compile(
    loss = CategoricalCrossentropy(),
    optimizer = Adam(learning_rate=0.001),
    metrics = ['accuracy']
)

history = vgg3.fit(train_data, epochs=15, steps_per_epoch=len(train_data),
        validation_data=val_data, validation_steps=len(val_data))
```

```
Epoch 1/15
102/102 [==============================] - 45s 411ms/step - loss: 1.0278 - accuracy:
Epoch 2/15
102/102 [==============================] - 41s 404ms/step - loss: 0.8909 - accuracy:
Epoch 3/15
102/102 [==============================] - 40s 396ms/step - loss: 0.8594 - accuracy:
Epoch 4/15
102/102 [==============================] - 40s 396ms/step - loss: 0.7358 - accuracy:
Epoch 5/15
102/102 [==============================] - 41s 405ms/step - loss: 0.6728 - accuracy:
Epoch 6/15
102/102 [==============================] - 40s 395ms/step - loss: 0.6757 - accuracy:
Epoch 7/15
102/102 [==============================] - 40s 396ms/step - loss: 0.7583 - accuracy:
Epoch 8/15
102/102 [==============================] - 42s 409ms/step - loss: 0.7282 - accuracy:
Epoch 9/15
102/102 [==============================] - 42s 411ms/step - loss: 0.5964 - accuracy:
Epoch 10/15
102/102 [==============================] - 41s 403ms/step - loss: 0.5345 - accuracy:
Epoch 11/15
102/102 [==============================] - 42s 411ms/step - loss: 0.4049 - accuracy:
Epoch 12/15
102/102 [==============================] - 41s 403ms/step - loss: 0.3482 - accuracy:
Epoch 13/15
102/102 [==============================] - 42s 411ms/step - loss: 0.2718 - accuracy:
Epoch 14/15
102/102 [==============================] - 40s 396ms/step - loss: 0.2574 - accuracy:
Epoch 15/15
102/102 [==============================] - 42s 407ms/step - loss: 0.2426 - accuracy:
```

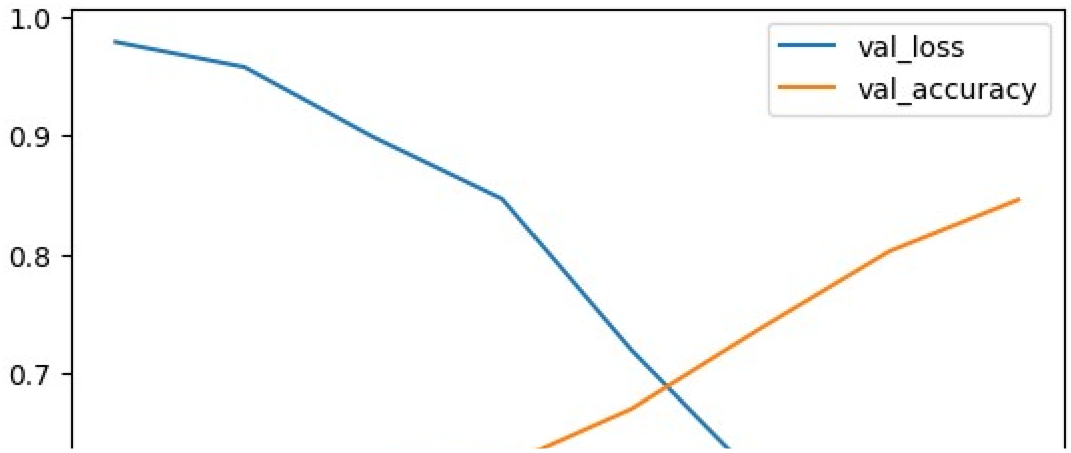
```
vgg3.evaluate(test_data)
```

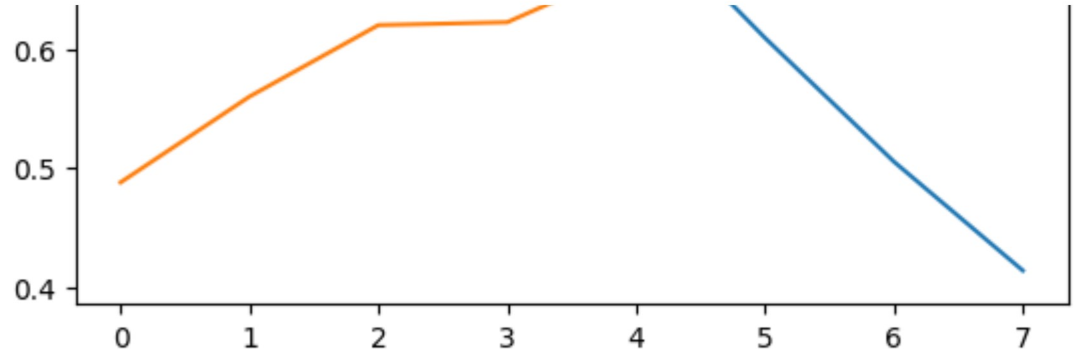
```
13/13 [==============================] - 1s 88ms/step - loss: 0.1948 - accuracy: 0.9
[0.19481322169303894, 0.9283950328826904]
```

```
import pandas as pd
import matplotlib.pyplot as plt


def plot_loss(history):
  df = pd.DataFrame(history.history)
  df[['loss', 'accuracy']].plot()

  plt.figure()
  df[['val_loss', 'val_accuracy']].plot()


plot_loss(history)
```



```
<Figure size 640x480 with 0 Axes>
```





Generate | Using ... | a slider using jupyter widgets | Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

```
import tensorflow_hub as hub


url = "https://www.kaggle.com/models/tensorflow/efficientnet/frameworks/Tenso
efficien_net_feature_extractor = hub.KerasLayer(url,
                                                trainable=False,
                                                input_shape= (224, 224)+(3,))

efficient_net = Sequential([
    efficien_net_feature_extractor,
    Dense(16, activation='relu'),
    Dense(16, activation='relu'),
    Dense(3, activation='softmax')
])

efficient_net.compile(
    loss = CategoricalCrossentropy(),
    optimizer = Adam(learning_rate=0.001),
    metrics = ['accuracy']
)

history = efficient_net.fit(train_data, epochs=15, steps_per_epoch=len(train_
        validation_data=val_data, validation_steps=len(val_data))
```

```
Epoch 1/15
102/102 [==============================] - 50s 457ms/step - loss: 1.0728 - accuracy:
Epoch 2/15
102/102 [==============================] - 46s 447ms/step - loss: 0.9553 - accuracy:
Epoch 3/15
102/102 [==============================] - 42s 408ms/step - loss: 0.7757 - accuracy:
Epoch 4/15
102/102 [==============================] - 43s 424ms/step - loss: 0.6304 - accuracy:
Epoch 5/15
102/102 [==============================] - 42s 410ms/step - loss: 0.5329 - accuracy:
```

```
Epoch 6/15
102/102 [==============================] - 42s 415ms/step - loss: 0.4650 - accuracy:
Epoch 7/15
102/102 [==============================] - 43s 417ms/step - loss: 0.4232 - accuracy:
Epoch 8/15
102/102 [==============================] - 42s 412ms/step - loss: 0.3903 - accuracy:
Epoch 9/15
102/102 [==============================] - 43s 418ms/step - loss: 0.3729 - accuracy:
Epoch 10/15
102/102 [==============================] - 42s 407ms/step - loss: 0.3507 - accuracy:
Epoch 11/15
102/102 [==============================] - 42s 412ms/step - loss: 0.3274 - accuracy:
Epoch 12/15
102/102 [==============================] - 43s 418ms/step - loss: 0.3101 - accuracy:
Epoch 13/15
102/102 [==============================] - 42s 415ms/step - loss: 0.3010 - accuracy:
Epoch 14/15
102/102 [==============================] - 42s 416ms/step - loss: 0.2936 - accuracy:
Epoch 15/15
102/102 [==============================] - 43s 419ms/step - loss: 0.2908 - accuracy:
```

```
efficient_net.evaluate(test_data)
```

```
13/13 [==============================] - 2s 154ms/step - loss: 0.2870 - accuracy: 0.
[0.2870118319988251, 0.9209876656532288]
```