```
# SOURCE CODE: https://github.com/PacktPublishing/Hands-On-Computer-Vision-with-TensorFlow-2/blob/master/Chapter04/ch4_nb2_reuse_m
import tensorflow as tf
import os
from matplotlib import pyplot as plt
import math
import numpy as np
import pandas as pd
import cv2
import seaborn as sns
from tqdm import tqdm
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score, confusion_matrix
import keras
from keras.preprocessing import image
from keras.models import Sequential, Model
from keras.layers import Dropout, MaxPooling2D, Dense, Conv2D, Activation, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import ResNet50V2
import warnings
warnings.filterwarnings("ignore")

# Choosing which GPU this notebook can access
# (useful when running multiple experiments in parallel, on different GPUs):
os.environ["CUDA_VISIBLE_DEVICES"]= "0"

# Some hyper-parameters:
input_shape = [224, 224, 3] # We will resize the input images to this shape
batch_size  = 32            # Images per batch (reduce/increase according to the machine's capability)
num_epochs  = 300           # Max number of training epochs
random_seed = 42            # Seed for some random operations, for reproducibility


#SOURCE DATA: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia/data
#number of chest Xrays in train folder: 1341 normal, 3875 pneumonia
#number of chest Xrays in test folder: 234 normal, 390 pneumonia

#INSTRUCTIONS UPLOADING KAGGLE DATASET TO GOOGLE COLAB: https://www.geeksforgeeks.org/how-to-import-kaggle-datasets-directly-into-
!pip install opendatasets
!pip install pandas
import opendatasets as od
import pandas
od.download(
    "https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia/data")
```

```
Collecting opendatasets
  Downloading opendatasets-0.1.22-py3-none-any.whl (15 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from opendatasets) (4.66.1)
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (from opendatasets) (1.5.16)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from opendatasets) (8.1.7)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2023.7.22)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.31.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (8.0.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle->opendatasets) ((
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle->o
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->op
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatasets)
Installing collected packages: opendatasets
Successfully installed opendatasets-0.1.22
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.1
Please provide your Kaggle credentials to download this dataset. Learn more: http://bit.ly/kaggle-creds
Your Kaggle username: ws912345
Your Kaggle Key: ··········
Downloading chest-xray-pneumonia.zip to ./chest-xray-pneumonia
100%|██████████| 2.29G/2.29G [00:29<00:00, 84.0MB/s]
```

```
# Number of classes:
num_classes = 2
class_names = ['NORMAL', 'PNEUMONIA']
# Number of images:
num_train_imgs = 1341 + 3875
num_val_imgs = 234 + 390
train_steps_per_epoch = math.ceil(num_train_imgs / batch_size)
val_steps_per_epoch   = math.ceil(num_val_imgs / batch_size)
```

```
#CODE HELP SOURCE FROM TENSORFLOW DOCUMENTATION: https://www.tensorflow.org/api_docs/python/tf/keras/utils/image_dataset_from_dire
# CODE HELP SOURCE FROM TENSORFLOW DOCUMENTATION: https://www.tensorflow.org/tutorials/load_data/images
#MAKING THE TrainING SET
#in this project we focus on using the dataset from chest-xray train which is further subdivided into 0.2-0.8 split of training ar
#Then calling image_dataset_from_directory(main_directory, labels='inferred') will return a tf.data.Dataset that yields batches of
train_ds = tf.keras.utils.image_dataset_from_directory(
  "/content/chest-xray-pneumonia/chest_xray/train/",
  labels='inferred',
  validation_split=0.2,
  subset="training",
  seed=random_seed,
  image_size=(224, 224),
  batch_size=batch_size, shuffle=True,
    )
```

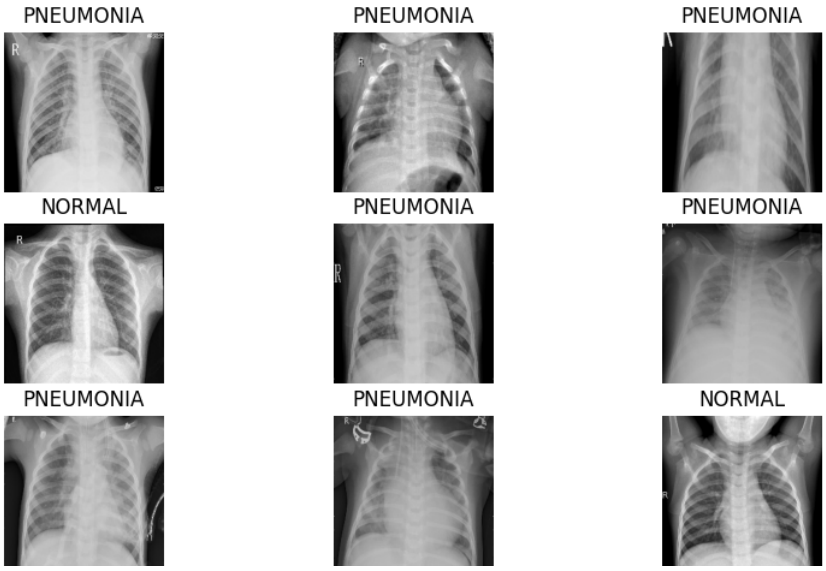
```
Found 5216 files belonging to 2 classes.
Using 4173 files for training.
```

```
#CODE HELP SOURCE FROM TENSORFLOW DOCUMENTATION: https://www.tensorflow.org/api_docs/python/tf/keras/utils/image_dataset_from_dire
# https://www.tensorflow.org/tutorials/load_data/images
#in this project we focus on using the dataset from chest-xray train which is further subdivided into 0.2-0.8 split of training ar
#Then calling image_dataset_from_directory(main_directory, labels='inferred') will return a tf.data.Dataset that yields batches of
test_ds = tf.keras.utils.image_dataset_from_directory(
  "/content/chest-xray-pneumonia/chest_xray/train/",
  labels='inferred',
  validation_split=0.2,
  subset="validation",
  seed=random_seed,
  image_size=(224, 224),
  batch_size=batch_size, shuffle=True,
    )
```

```
Found 5216 files belonging to 2 classes.
Using 1043 files for validation.
```

```
#visualizing first nine images from the training set
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(15):
    ax = plt.subplot(5, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```




```
#INITIATE RESNET
#source code: https://github.com/PacktPublishing/Hands-On-Computer-Vision-with-TensorFlow-2/blob/9a73003eff274f288d59dfb1532a5a486
model = tf.keras.applications.resnet50.ResNet50(
    include_top=True, weights=None,
    input_shape=input_shape, classes=num_classes)
model.summary()
```

```
Model: "resnet50"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0         []

 conv1_pad (ZeroPadding2D)   (None, 230, 230, 3)       0         ['input_2[0][0]']

 conv1_conv (Conv2D)         (None, 112, 112, 64)      9472      ['conv1_pad[0][0]']

 conv1_bn (BatchNormalizati  (None, 112, 112, 64)      256       ['conv1_conv[0][0]']
 on)

 conv1_relu (Activation)     (None, 112, 112, 64)      0         ['conv1_bn[0][0]']

 pool1_pad (ZeroPadding2D)   (None, 114, 114, 64)      0         ['conv1_relu[0][0]']

 pool1_pool (MaxPooling2D)   (None, 56, 56, 64)        0         ['pool1_pad[0][0]']

 conv2_block1_1_conv (Conv2  (None, 56, 56, 64)        4160      ['pool1_pool[0][0]']
 D)

 conv2_block1_1_bn (BatchNo  (None, 56, 56, 64)        256       ['conv2_block1_1_conv[0][0]']
 rmalization)

 conv2_block1_1_relu (Activ  (None, 56, 56, 64)        0         ['conv2_block1_1_bn[0][0]']
 ation)

 conv2_block1_2_conv (Conv2  (None, 56, 56, 64)        36928     ['conv2_block1_1_relu[0][0]']
 D)

 conv2_block1_2_bn (BatchNo  (None, 56, 56, 64)        256       ['conv2_block1_2_conv[0][0]']
 rmalization)

 conv2_block1_2_relu (Activ  (None, 56, 56, 64)        0         ['conv2_block1_2_bn[0][0]']
 ation)

 conv2_block1_0_conv (Conv2  (None, 56, 56, 256)       16640     ['pool1_pool[0][0]']
 D)

 conv2_block1_3_conv (Conv2  (None, 56, 56, 256)       16640     ['conv2_block1_2_relu[0][0]']
 D)

 conv2_block1_0_bn (BatchNo  (None, 56, 56, 256)       1024      ['conv2_block1_0_conv[0][0]']
 rmalization)

 conv2_block1_3_bn (BatchNo  (None, 56, 56, 256)       1024      ['conv2_block1_3_conv[0][0]']
 rmalization)

 conv2_block1_add (Add)      (None, 56, 56, 256)       0         ['conv2_block1_0_bn[0][0]',
                                                                  'conv2_block1_3_bn[0][0]']
```

```
conv2_block1_out (Activati  (None, 56, 56, 256)          0         ['conv2_block1_add[0][0]']
on)

conv2_block2_1_conv (Conv2  (None, 56, 56, 64)           16448     ['conv2_block1_out[0][0]']
D)
```

```
#standardizing the data
#standardize RGB channels from [0,255] to [0,1]
#source: https://www.tensorflow.org/tutorials/load_data/images
normalization_layer = tf.keras.layers.Rescaling(1./255)
normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
print(np.min(first_image), np.max(first_image))
```

```
0.0 1.0
```

```
#configure dataset for performance
#prevent I/O blocking when calling data from disk
#code source: https://www.tensorflow.org/tutorials/load_data/images
AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

```
#apply training now to the data using the model, use Adam optimizer and Sparse Categorical Cross Entropy for loss function, add sc
model.compile(
  optimizer='adam',
  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
  metrics=['accuracy'])
```

```
model.fit(
  train_ds,
  validation_data=train_ds,
  epochs=5
)
```

```
Epoch 1/3
131/131 [==============================] - 3769s 29s/step - loss: 0.3095 - accuracy: 0.8955 - val_loss: 3.6947 - val_accuracy
Epoch 2/3
131/131 [==============================] - 3790s 29s/step - loss: 0.1535 - accuracy: 0.9410 - val_loss: 0.6222 - val_accuracy
Epoch 3/3
131/131 [==============================] - 3698s 28s/step - loss: 0.1140 - accuracy: 0.9564 - val_loss: 18.7119 - val_accurac
<keras.src.callbacks.History at 0x7afa243a6f20>
```