Histopathologic Image Classification of Benign Fibro-Osseous Lesions of the Jaws Using Deep Neural Network

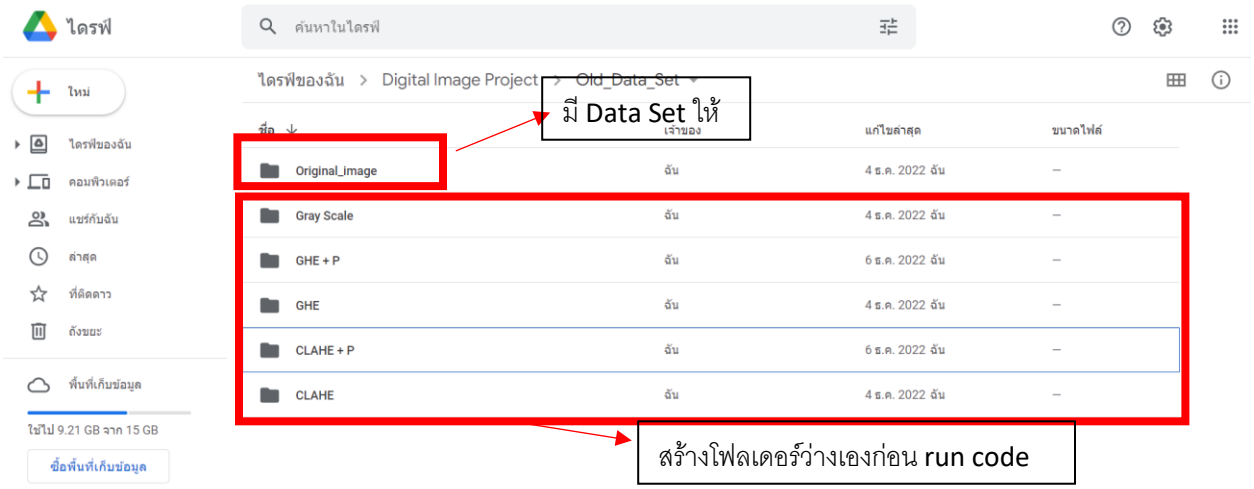
1.การอัพโหลดข้อมูลและการสร้างโฟลเดอร์ที่จำเป็นในการเก็บรูปภาพบน Google drive

อัพโหลดข้อมูลขึ้นบน Google Drive และสร้างโฟลเดอร์ที่จะใช้ในการเก็บไฟล์ข้อมูลรูปภาพลักษณะต่าง ๆ ลงบน Google Drive โดยจากงานของเราจะมีรูปภาพทั้งหมดบน Google Drive 6 แบบ รวมภาพ Original RGB Image โดย

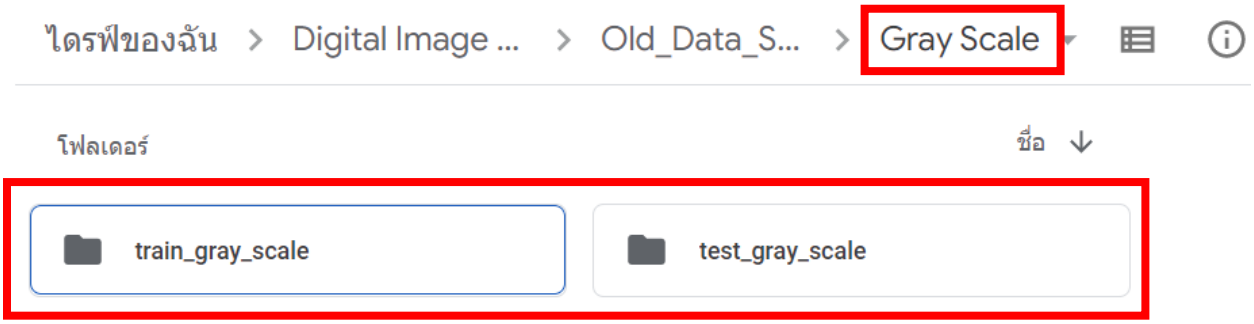
P => Piecewise Linear Contrast Stretching

GHE => Global Histogram Equalization

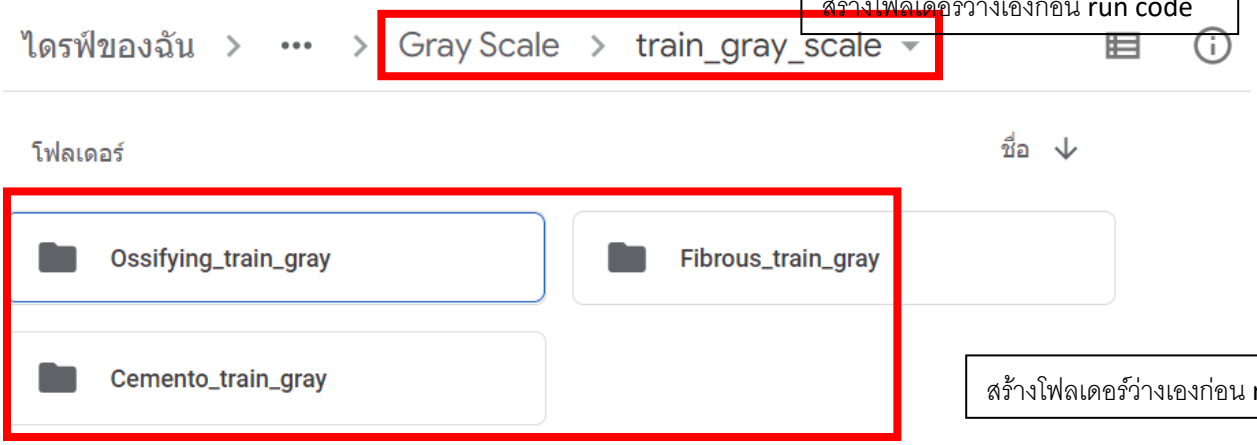
CLAHE => Contrast Limited Adaptive Histogram Equalization



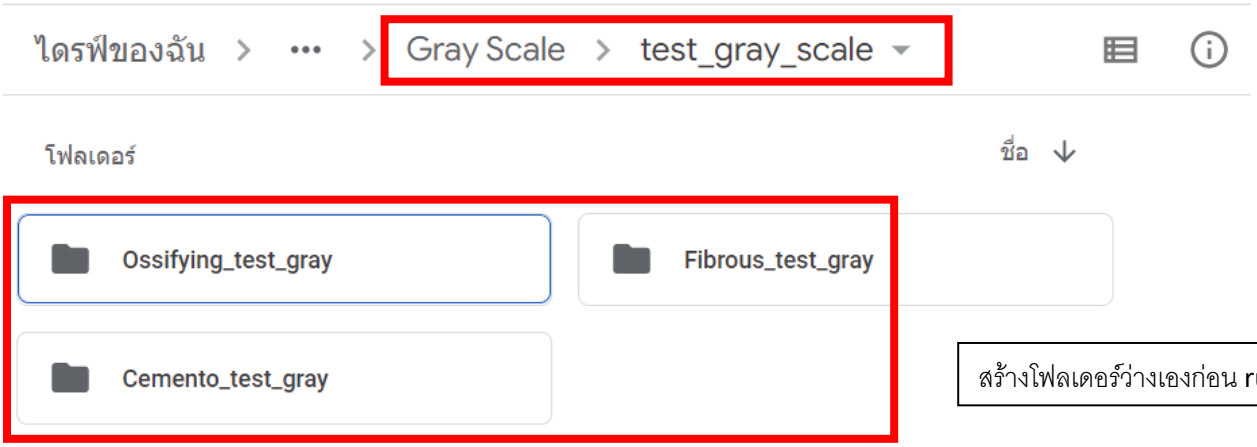
ภายในไดร์ฟจะแบ่งเป็น 2 โฟลเดอร์ คือ ข้อมูล Train และ test และภายในแต่ละโฟลเดอร์จะมีการแยกรูปภาพตามชนิดของรอยโรค คือ



สร้างโฟลเดอร์ว่างเองก่อน run code

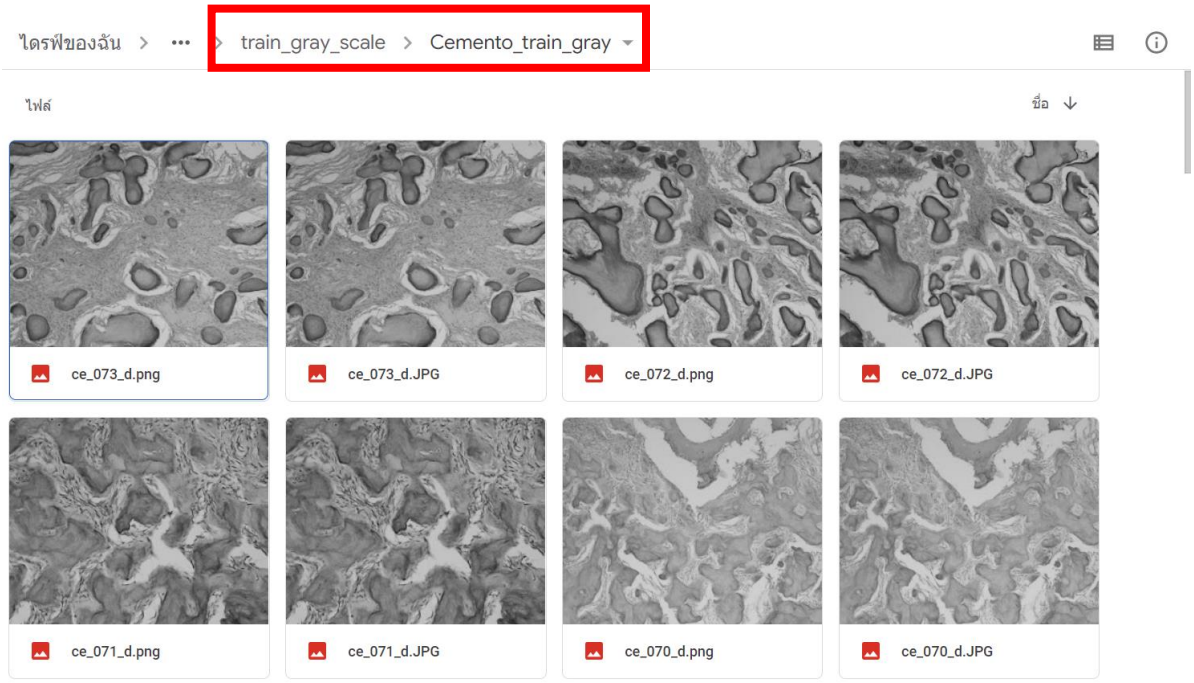


สร้างโฟลเดอร์ว่างเองก่อน run code



สร้างโฟลเดอร์ว่างเองก่อน run code

ภายในแต่ละโฟลเดอร์จะมีภาพถ่ายทางจุลพยาธิวิทยาของรอยโรคอยู่



ภาพรวมในการเข้าถึงไฟล์รูปภาพจะเป็น และจำนวนโฟลเดอร์ที่ต้องสร้างจะมี ดังนี้

RGB Image >>> Train and Test >>> Cemento and Fibrous and Ossifying (มี Dataset ให้)

Grayscale >>> Train and Test >>> Cemento and Fibrous and Ossifying

GHE >>> Train and Test >>> Cemento and Fibrous and Ossifying

CLAHE >>> Train and Test >>> Cemento and Fibrous and Ossifying

GHE + P >>> Train and Test >>> Cemento and Fibrous and Ossifying

CLAHE + P >>> Train and Test >>> Cemento and Fibrous and Ossifying

2. RUN CODE

2.1 แปลงภาพ

2.1.1 เชื่อมต่อ Colab กับ Google Drive ที่ทำการสร้างโฟลเดอร์ในการเก็บภาพถ่ายทางจุลพยาธิวิทยาไว้

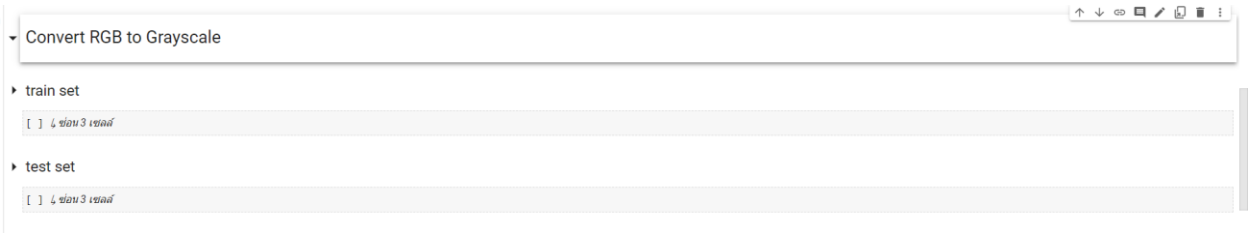
```
[1]  from google.colab import drive
     drive.mount('/content/drive')
```

2.1.2 Import library ที่จำเป็น

```
[ ]  import os
     import glob
     import keras
     import sklearn
     import cv2 as cv2
     import numpy as np
     import pandas as pd
     import PIL as Image
     import seaborn as sns
     import pathlib as path
     import tensorflow as tf
     import matplotlib as mpl
     from datetime import datetime
     import matplotlib.image as img
     from keras import backend as k
     from keras.models import Model
     import matplotlib.pyplot as plt
     from keras.utils import np_utils
     from google.colab.patches import cv2_imshow
     from tensorflow.keras.preprocessing import image
     from tensorflow.keras.datasets import fashion_mnist
     from keras.layers.convolutional import  MaxPooling2D
     from sklearn.model_selection import train_test_split
     from tensorflow.keras.callbacks import ModelCheckpoint
     from tensorflow.keras.optimizers import Adam,SGD,RMSprop
     from tensorflow.keras.applications.resnet50 import ResNet50
     from tensorflow.keras.models import Sequential, Model, load_model
     from tensorflow.keras import models, layers, optimizers, callbacks
     from sklearn.metrics import confusion_matrix, classification_report
     from  tensorflow.keras.applications.vgg16 import VGG16,preprocess_input
     from tensorflow.keras.utils import to_categorical, image_dataset_from_directory
     from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img,img_to_array
     from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D, BatchNormalization, ReLU,Flatten,Activation, GlobalAveragePooling2D, Input
     %matplotlib inline
```

2.1.3 แปลงภาพให้อยู่ในรูปแบบต่างๆตามโค้ดที่ส่งให้ โดยเปลี่ยน Path ตามที่เราใส่ไว้ใน Drive ตัวเอง

ลักษณะของ Code ตรงส่วนของการแปลงภาพเบื้องต้นเมื่อเปิดไฟล์ที่ให้ไป



มีสิ่งที่ต้องแก้คือ Path ที่ใช้เชื่อมต่อกับโฟลเดอร์ในไดร์ฟของเราที่สร้างไว้ โดยบรรทัดที่ 1 คือไดรฟ์ต้นทางที่เราจะทำการดึงภาพถ่ายทางจุลพยาธิวิทยามาแปลง และบรรทัดต่อมาคือไดรฟ์ปลายทางที่เราจะทำการจัดเก็บภาพหลังการแปลง ซึ่งจะทำเหมือนกันในข้อมูล Train และ Test ทั้ง 5 รูปแบบการแปลงภาพ

ไดรฟ์ต้นทาง

ไดรฟ์ปลายทาง

```
data_train_ce = '/content/drive/MyDrive/Digital Image Project/Old_Data_Set/Original_image/train_cell/Cemento'
gray_train_ce = '/content/drive/MyDrive/Digital Image Project/Old_Data_Set/Gray Scale/train_gray_scale/Cemento_train_gray'
try:
    makedirs(gray_train_ce)
except:
    print ("Directory already exist, images will be written in asme folder")

# Folder won't used
files = os.listdir(data_train_ce)

for image in files:
    img = cv2.imread(os.path.join(data_train_ce,image))
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) #Convert RGB image to Gray-scale
    cv2.imwrite(os.path.join(gray_train_ce,image),gray)
```

*********ในภาพคือการแปลงจาก RGB to Grayscale**********

**********ในบางรูปแบบการแปลงอาจต้องสร้างฟังก์ชั่นซึ่งสามารถกดรันตามลำดับในโค้ดที่แนบมาได้เลย******

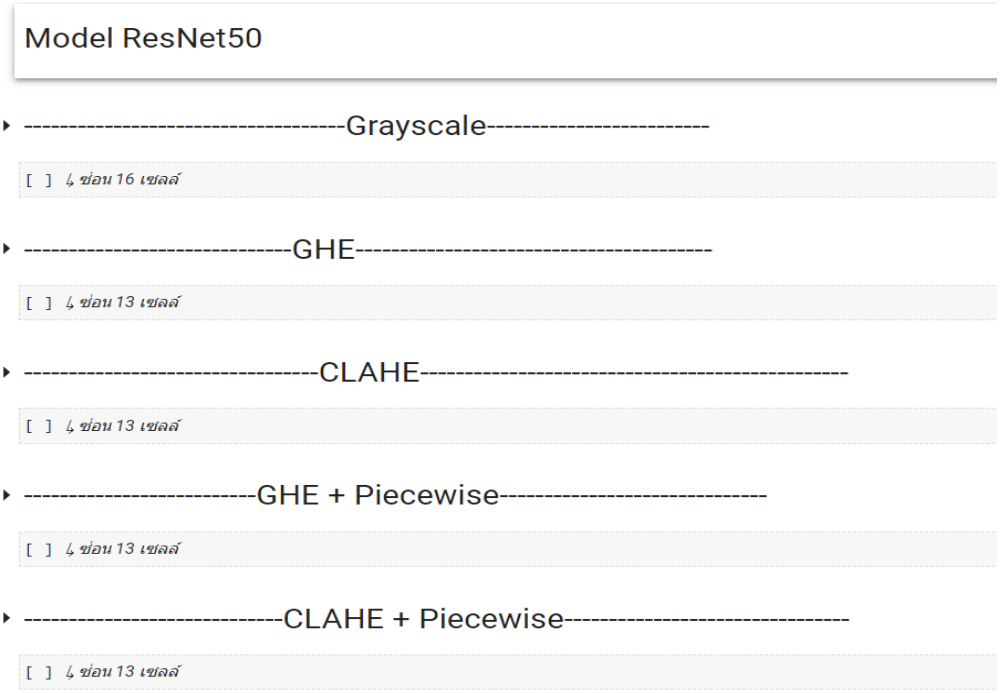
2.2. Preprocess Data

เมื่อแปลงข้อมูลเสร็จจะเป็นการแบ่งส่วนเพื่อใช้ในการ Train และ Test model สามารถปรับแก้ หรือ กดรันตามโค้ดได้เลย

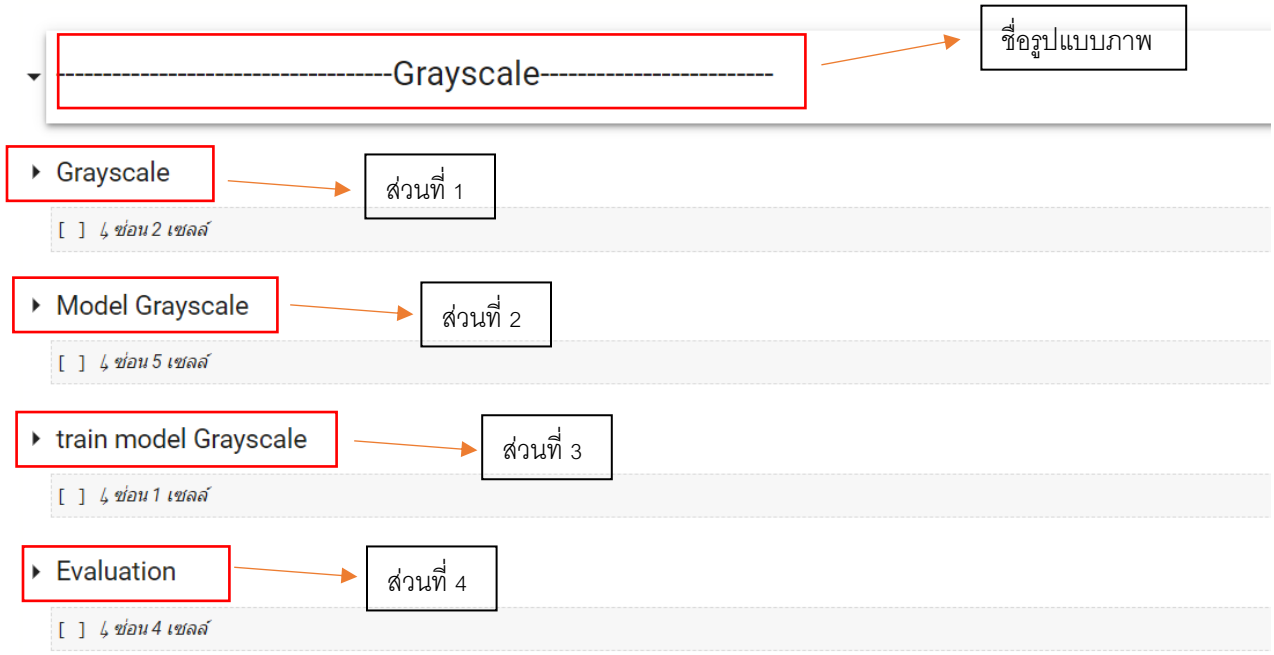
```
IMAGE_SIZE = [224,224]
train_datagen = ImageDataGenerator(
                              rescale=1./255, #rescal Grayscale
                              horizontal_flip = True,
                              vertical_flip = True,
                              validation_split = 0.2
)

validation_datagen = ImageDataGenerator(
    rescale = 1./255, validation_split = 0.2
)

test_datagen = ImageDataGenerator(
    rescale = 1./255
)
```

2.3 Model

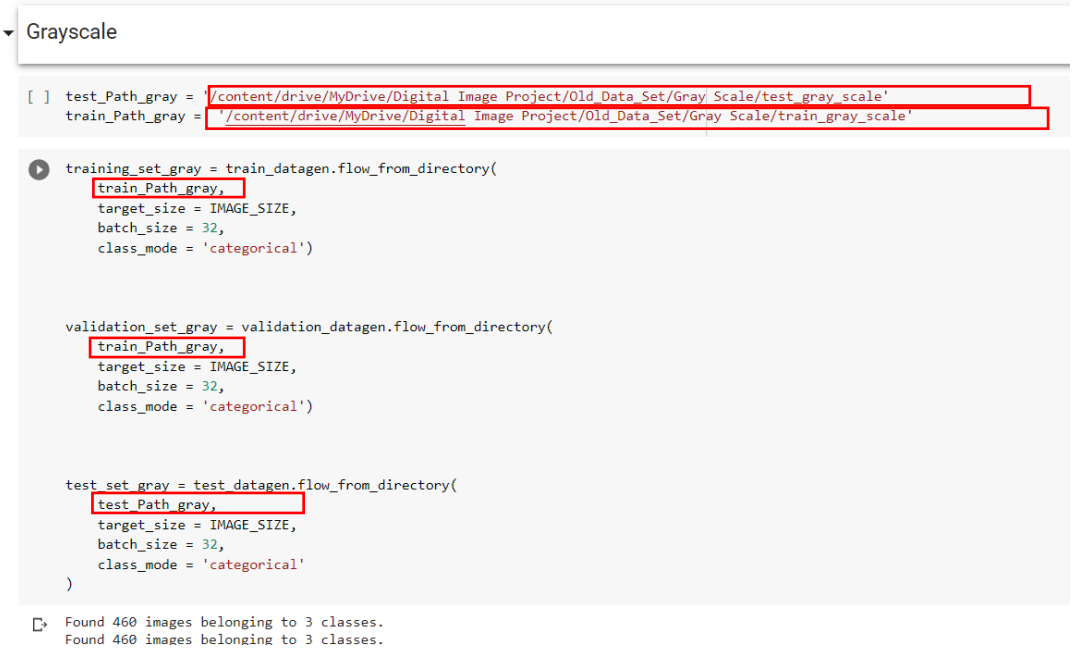
เป็นกระบวนการในการใช้ข้อมูลในรูปแบบตาง ๆ มา Train Model ซึ่งจะแบ่งออกเป็น 5 รูปแบบ ดังภาพ



ในทุก ๆ รูปแบบภาพจะมีการแบ่งส่วนเพื่อให้ง่ายต่อการ Run ดังนี้



ส่วนที่ 1 ต้องเปลี่ยน Path ของข้อมูล Train และ Test ในทุก ๆ รูปแบบของการปรับปรุงคุณภาพภาพให้ตรงกับ Location ของภาพรูปแบบนั้น ๆ (ในกรณีนี้ในภาพคือการ Train Grayscale)



ส่วนที่ 2 ตั้งแต่ส่วนที่ 2 – 4 สามารถ Run code ตามลำดับได้เลย

เป็นการเรียนใช้ ResNet50 Model Code อยู่ในส่วนของ Grayscale ซึ่งสามารถเรียกใช้ในการแปลงภาพรูปแบบอื่น ๆ ได้เลย

```
[ ]  resnet = ResNet50(
         input_shape = IMAGE_SIZE + [3], #......
         weights = 'imagenet',
         include_top = False )
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 3s 0us/step
```

```
[ ]  for layer in resnet.layers:
         layer.trainable = False
     import glob
     folders_gray = glob.glob(train_Path_gray + '/*')
     folders_gray
```

```
['/content/drive/MyDrive/Digital Image Project/Old_Data_Set/Gray Scale/train_gray_scale/Cemento_train_gray',
 '/content/drive/MyDrive/Digital Image Project/Old_Data_Set/Gray Scale/train_gray_scale/Fibrous_train_gray',
 '/content/drive/MyDrive/Digital Image Project/Old_Data_Set/Gray Scale/train_gray_scale/Ossifying_train_gray']
```

```
[ ]  x = Flatten() (resnet.output)
     prediction = Dense(len(folders_gray), activation = 'softmax')(x)
     resnet50_model = Model(inputs = resnet.input, outputs = prediction)
     resnet50_model.compile (
         loss = 'categorical_crossentropy',
         optimizer = 'adam',
         metrics = ['accuracy']
     )
```

ส่วนที่ 3 เป็นการ Train Model ที่กำหนด จำนวน epoch = 20 และ Bath size = 32 (สามารถปรับตัวเลขได้) โดยตั้งการ callback ไว้ให้ดูที่ค่า loss โดยหากค่า Loss ไม่ลดลงติดกัน 2 ครั้ง จะทำการหยุดการ Train model และ Save Epoch ที่สูงเป็นลำดับที่ 3 นับจากด้านล่าง

train model Grayscale

```
early_stop = callbacks.EarlyStopping(monitor='val_loss', patience=2)
mcp_save = ModelCheckpoint('history_resnet50_model_gray_earlystop.h5', save_best_only=True, monitor='val_loss')

history_resnet50_model_gray_earlystop = resnet50_model.fit(
                      training_set_gray,
                      batch_size=32,
                      epochs=20,
                      validation_data = (validation_set_gray),
                      callbacks=[early_stop,mcp_save]
                      )
```

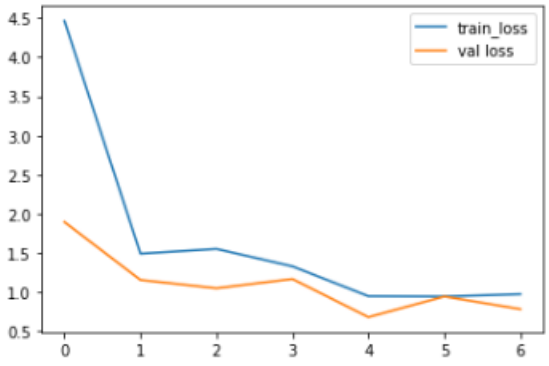
```
Epoch 1/20
15/15 [==============================] - 95s 6s/step - loss: 4.4640 - accuracy: 0.4565 - val_loss: 1.8955 - val_accuracy: 0.5848
Epoch 2/20
15/15 [==============================] - 25s 2s/step - loss: 1.4919 - accuracy: 0.5326 - val_loss: 1.1560 - val_accuracy: 0.5543
Epoch 3/20
15/15 [==============================] - 23s 2s/step - loss: 1.5525 - accuracy: 0.4935 - val_loss: 1.0503 - val_accuracy: 0.5957
Epoch 4/20
15/15 [==============================] - 25s 2s/step - loss: 1.3302 - accuracy: 0.6196 - val_loss: 1.1683 - val_accuracy: 0.6174
Epoch 5/20
15/15 [==============================] - 23s 2s/step - loss: 0.9491 - accuracy: 0.6522 - val_loss: 0.6814 - val_accuracy: 0.6913
Epoch 6/20
15/15 [==============================] - 23s 2s/step - loss: 0.9454 - accuracy: 0.6000 - val_loss: 0.9451 - val_accuracy: 0.6913
Epoch 7/20
15/15 [==============================] - 23s 2s/step - loss: 0.9750 - accuracy: 0.6304 - val_loss: 0.7836 - val_accuracy: 0.6130
```

Save

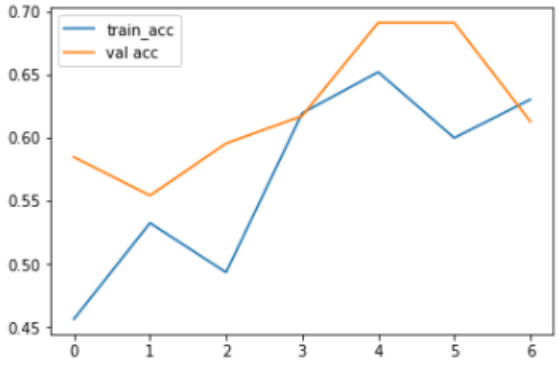
ส่วนที่ 4 เป็นการดูผลลัพธ์ความแม่นยำที่ได้จากการ Train Model ด้วยภาพถ่ายทางจุลพยาธิวิทยาในแบบต่าง ๆ

Evaluation

```
plt.plot(history_resnet50_model_gray_earlystop.history['loss'], label = 'train_loss')
plt.plot(history_resnet50_model_gray_earlystop.history['val_loss'], label ='val loss')
plt.legend()
plt.show()
```




```
plt.plot(history_resnet50_model_gray_earlystop.history['accuracy'], label = 'train_acc')
plt.plot(history_resnet50_model_gray_earlystop.history['val_accuracy'], label ='val acc')
plt.legend()
plt.show()
```



```
model_gray_earlystop = load_model('history_resnet50_model_gray_earlystop.h5')
score1 = model_gray_earlystop.evaluate(training_set_gray)
score1
```

```
15/15 [==============================] - 13s 760ms/step - loss: 0.6950 - accuracy: 0.6957
[0.6950275301933289, 0.695652186870575]
```

```
[ ] score2 = model_gray_earlystop.evaluate(test_set_gray)
    score2
```

```
2/2 [==============================] - 8s 7s/step - loss: 0.8848 - accuracy: 0.6222
[0.8848317265510559, 0.6222222447395325]
```

ในการ Train model ของการปรับปรุงภาพถ่ายทางจุลพยาธิวิทยาในรูปแบบอื่น ๆ หลังจากการเปลี่ยน Path แล้ว สามารถ Run Code ตามลำดับได้เลยเช่นเดียวกันกับ ภาพในรูปแบบ Grayscale

ปล. ชุดข้อมูลที่แสดงการรันโค้ดข้างต้นคือ ชุดข้อมูลรูปภาพ Grayscale โดยชุดข้อมูลชุดอื่น ๆ ก็จะทำการรันโค้ดตามวิธีที่แสดงข้างต้นเช่นกัน