**Readme:**

### **ResNet50\_pretrained\_trained.ipynb**

This notebook file contains code to run the ResNet50 pretrained model, as well as Full training of the model.

def model\_dataset\_epoch(model, train\_set, val\_set, test\_set, epoch, folder\_path, plot\_title="")

This function takes the model, training, validation, and test dataset, number of epochs, and folder path where to store the model.

We can define pretrained model using this function -

model = resnet50\_only()

And, to train the whole model, we can use -

model = resnet50\_only(train=True)

Running the following marked cells in the notebook will give the test results shown in the table of our report.

* ResNet50 (Pretrained): 40X, 100X, 400X, 200X
* ResNet50 (Trained): 40X, 100X, 400X, 200X

### **ResNet50\_Inception.ipynb**

This notebook contains the code to run our proposed lightweight model.

def resnetinnet\_builder(finetune=False, model=None, trainable\_blocks=None, print\_trainable=False)

Using this function we can create our lightweight model of different configurations. Trainable\_blocks will contain the block that we want to train in the Resnet part.

def lr\_schedule(epoch, lr\_warmup\_epochs = 25, lr\_min = 1e-5, lr\_max = 5e-5, lr\_exp\_decay = 0.8)

This function is used for learning rate scheduler in the fine-tuning part.

def model\_dataset\_epoch(model, train\_set, val\_set, test\_set, epoch, folder\_path, strategy="train", plot\_title="")

Using this function, we can train a model for the given epochs and save the model.

The following marked cells in the notebook will re-create the test results, learning curve, confusion matrix, and ROC-AUC plots shown in the report.

* 400X : Configuration - 1,2,3,4
* 40X : Configuration - 1,2,3,4
* 100X : Configuration - 1,2,3,4
* 200X : Configuration - 1,2,3,4