Import all the libraries

In [1]:
```
import numpy as np
import pandas as pd
from PIL import Image
import tensorflow as tf
from sklearn.model_selection import train_test_split
import cv2
import gc
import os
import tensorflow as tf
from openslide import open_slide
import keras
from openslide.deepzoom import DeepZoomGenerator
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import applications
```

In [2]:
```
inp_size=512

eff_res_stacked = '/kaggle/input/final-dataset/final_effb0_res152_stacked.h5'
eff_vgg_stacked='/kaggle/input/final-dataset/final_effb0_vgg19_stacked.h5'
vgg_res_stacked='/kaggle/input/final-dataset/final_res152_vgg19_stacked.h5'
vgg='/kaggle/input/final-dataset/final_vgg19.h5'
all_three_stacked='/kaggle/input/final-dataset/final_effb0_res151_vgg19_stacked.h5'
res='/kaggle/input/final-dataset/final_res152.h5'
eff='/kaggle/input/final-dataset/final_effb0.h5'
```

Hyper Parameters to be Tuned

In [3]:
```
models=[]
model_list = [all_three_stacked]
epochs=40
batch_size=32
learning_rate = 1e-3
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)  #adam or sgd
```

Fixed Parameters and paths

In [4]:
```
color_normalized = 'yes'
if color_normalized == 'yes':
    dataset_directory='/kaggle/input/mayo-512720-dataset/DATASET/norm'
else:
    dataset_directory='/kaggle/input/mayo-512720-dataset/DATASET/original'

#----------------------
classMode = 'categorical'
loss = 'categorical_crossentropy'
val_split = 0
Num_tiles_to_consider_for_prediction = 3
validation_steps= 10
steps_per_epoch = 10
val_batch_size=16

train_csv='/kaggle/input/mayo-clinic-strip-ai/train.csv'
test_csv = '/kaggle/input/mayo-clinic-strip-ai/test.csv'
sample_sub_csv='/kaggle/input/mayo-clinic-strip-ai/sample_submission.csv'

train_imgs_dir = '/kaggle/input/mayo-clinic-strip-ai/train'
test_imgs_dir='/kaggle/input/mayo-clinic-strip-ai/test'
```

Load and compile the models

In [5]:
```
for i in model_list:
    models.append(keras.models.load_model(i))
```

```
2023-02-09 15:20:24.433162: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-09 15:20:24.544216: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-09 15:20:24.545028: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-09 15:20:24.546366: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized wi
th oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operation
s:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-02-09 15:20:24.546706: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-09 15:20:24.547406: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-09 15:20:24.548039: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-09 15:20:26.884611: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-09 15:20:26.885520: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-09 15:20:26.886274: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-09 15:20:26.886864: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Created device /job:localhost/repl
ica:0/task:0/device:GPU:0 with 15401 MB memory:  -> device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, co
mpute capability: 6.0
```

In [6]:
```
for i in models:
    i.compile(
        optimizer=optimizer,
        loss=loss,
        metrics=['accuracy'])
```

In [7]:
```
models
```

Out[7]:
```
[<keras.engine.functional.Functional at 0x7fd0fcead290>]
```

In [8]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=45,
    validation_split=val_split,
    width_shift_range=0.3,
    height_shift_range=0.3,
```

```
    brightness_range=[0.5,2.0],
    zoom_range = [0.5,2.0],
    horizontal_flip=True,
    vertical_flip=True
)

print(dataset_directory)
train_generator = datagen.flow_from_directory(
    dataset_directory,
    target_size=(inp_size, inp_size),
    batch_size=batch_size,
    shuffle=True,
    class_mode=classMode,
    subset='training',
)

validation_generator = datagen.flow_from_directory(
    dataset_directory,
    target_size=(inp_size, inp_size),
    batch_size=val_batch_size,
    shuffle=True,
    class_mode=classMode,
    subset='validation',
)

class_indices = train_generator.class_indices
print(class_indices)
```

```
/kaggle/input/mayo-512720-dataset/DATASET/norm
Found 4871 images belonging to 2 classes.
Found 0 images belonging to 2 classes.
{'ce': 0, 'laa': 1}
```

In [9]:

```
for i in models:
    i.fit(
        train_generator,
        steps_per_epoch=steps_per_epoch,
        validation_steps=validation_steps,
        epochs=epochs,
#         validation_data=validation_generator,
    )
```

```
2023-02-09 15:20:45.699058: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimizati
on Passes are enabled (registered 2)
Epoch 1/40
```

```
2023-02-09 15:21:02.852414: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN version 8005
```

```
10/10 [==============================] - 62s 3s/step - loss: 0.9759 - accuracy: 0.5219
Epoch 2/40
10/10 [==============================] - 35s 3s/step - loss: 1.0368 - accuracy: 0.5500
Epoch 3/40
10/10 [==============================] - 34s 3s/step - loss: 0.9221 - accuracy: 0.5322
Epoch 4/40
10/10 [==============================] - 34s 3s/step - loss: 1.0193 - accuracy: 0.5063
Epoch 5/40
10/10 [==============================] - 34s 3s/step - loss: 0.9826 - accuracy: 0.5094
Epoch 6/40
10/10 [==============================] - 33s 3s/step - loss: 0.9033 - accuracy: 0.5063
Epoch 7/40
10/10 [==============================] - 33s 3s/step - loss: 0.8305 - accuracy: 0.5469
Epoch 8/40
10/10 [==============================] - 33s 3s/step - loss: 0.7911 - accuracy: 0.5437
Epoch 9/40
10/10 [==============================] - 30s 3s/step - loss: 0.8257 - accuracy: 0.5559
Epoch 10/40
10/10 [==============================] - 31s 3s/step - loss: 0.7460 - accuracy: 0.5356
Epoch 11/40
10/10 [==============================] - 32s 3s/step - loss: 0.7944 - accuracy: 0.5281
Epoch 12/40
10/10 [==============================] - 32s 3s/step - loss: 0.7241 - accuracy: 0.5594
Epoch 13/40
10/10 [==============================] - 34s 3s/step - loss: 0.7171 - accuracy: 0.5656
Epoch 14/40
10/10 [==============================] - 32s 3s/step - loss: 0.7366 - accuracy: 0.5312
Epoch 15/40
10/10 [==============================] - 32s 3s/step - loss: 0.7310 - accuracy: 0.5531
Epoch 16/40
10/10 [==============================] - 31s 3s/step - loss: 0.7008 - accuracy: 0.5375
Epoch 17/40
10/10 [==============================] - 31s 3s/step - loss: 0.7466 - accuracy: 0.5188
Epoch 18/40
10/10 [==============================] - 31s 3s/step - loss: 0.7167 - accuracy: 0.5250
Epoch 19/40
10/10 [==============================] - 31s 3s/step - loss: 0.6928 - accuracy: 0.5219
Epoch 20/40
10/10 [==============================] - 31s 3s/step - loss: 0.6770 - accuracy: 0.5625
Epoch 21/40
10/10 [==============================] - 31s 3s/step - loss: 0.6825 - accuracy: 0.5906
Epoch 22/40
10/10 [==============================] - 31s 3s/step - loss: 0.7093 - accuracy: 0.5594
Epoch 23/40
10/10 [==============================] - 31s 3s/step - loss: 0.6993 - accuracy: 0.5437
```

```
Epoch 24/40
10/10 [==============================] - 30s 3s/step - loss: 0.7059 - accuracy: 0.5219
Epoch 25/40
10/10 [==============================] - 31s 3s/step - loss: 0.6708 - accuracy: 0.5938
Epoch 26/40
10/10 [==============================] - 31s 3s/step - loss: 0.7098 - accuracy: 0.5250
Epoch 27/40
10/10 [==============================] - 31s 3s/step - loss: 0.7003 - accuracy: 0.5406
Epoch 28/40
10/10 [==============================] - 31s 3s/step - loss: 0.6911 - accuracy: 0.5625
Epoch 29/40
10/10 [==============================] - 31s 3s/step - loss: 0.6986 - accuracy: 0.5594
Epoch 30/40
10/10 [==============================] - 30s 3s/step - loss: 0.6876 - accuracy: 0.5594
Epoch 31/40
10/10 [==============================] - 30s 3s/step - loss: 0.6901 - accuracy: 0.5406
Epoch 32/40
10/10 [==============================] - 30s 3s/step - loss: 0.7022 - accuracy: 0.5156
Epoch 33/40
10/10 [==============================] - 30s 3s/step - loss: 0.6940 - accuracy: 0.5219
Epoch 34/40
10/10 [==============================] - 30s 3s/step - loss: 0.6830 - accuracy: 0.5906
Epoch 35/40
10/10 [==============================] - 30s 3s/step - loss: 0.6845 - accuracy: 0.5469
Epoch 36/40
10/10 [==============================] - 30s 3s/step - loss: 0.6846 - accuracy: 0.5500
Epoch 37/40
10/10 [==============================] - 31s 3s/step - loss: 0.7010 - accuracy: 0.5312
Epoch 38/40
10/10 [==============================] - 30s 3s/step - loss: 0.6893 - accuracy: 0.5625
Epoch 39/40
10/10 [==============================] - 30s 3s/step - loss: 0.6959 - accuracy: 0.5375
Epoch 40/40
10/10 [==============================] - 30s 3s/step - loss: 0.6849 - accuracy: 0.5562
```

model testing

In [10]:

```
def make_test_file(x):
    return os.path.join(test_imgs_dir,x+'.tif')
test = pd.read_csv(test_csv)
test_data = pd.DataFrame({'image_id': test.image_id.apply(make_test_file)})
test_data.head()
```

Out[10]:

	image_id
0	/kaggle/input/mayo-clinic-strip-ai/test/006388...
1	/kaggle/input/mayo-clinic-strip-ai/test/008e5c...
2	/kaggle/input/mayo-clinic-strip-ai/test/00c058...
3	/kaggle/input/mayo-clinic-strip-ai/test/01adc5...

In [11]:

```
preds=[]
for x in range(int(test_data.size)):
    img_path = test_data.image_id[x]
    slide = open_slide(img_path)
    tiles=DeepZoomGenerator(slide,tile_size=inp_size,overlap=0,limit_bounds=False)
    cols,rows = tiles.level_tiles[tiles.level_count-1]
    print(x)
    temp_preds=[]
    count=0

    for row in range(0,rows,5):
        for col in range(0,cols,5):
            tile=tiles.get_tile(tiles.level_count-1,(col,row))
            tile=tile.convert("RGB")
            tile=np.array(tile)
            try:
                if tile.mean()<180 and tile.std()>50:
                    tile = np.reshape(tile, [1,inp_size, inp_size, 3])
                    p=[i.predict(tile/255) for i in models]
                    t_p = sum(p)/len(p)
                    temp_preds.append(t_p)
                    count+=1
                    if count>Num_tiles_to_consider_for_prediction:break
            except :
                pass
            if count>Num_tiles_to_consider_for_prediction:break
    if len(temp_preds) > 0:
        preds.append(sum(temp_preds)/len(temp_preds))
    else:
        preds.append([[0.5,0.5]])
    del slide
    del tiles
    gc.collect()
```

```
0
1
2
3
```

In [12]:
```
preds
```

Out[12]:
```
[array([[0.49939284, 0.50060713]], dtype=float32),
 array([[0.54945433, 0.45054564]], dtype=float32),
 array([[0.4477892, 0.5522108]], dtype=float32),
 array([[0.42307842, 0.57692164]], dtype=float32)]
```

In [13]:
```
preds = pd.DataFrame(np.concatenate(preds))
submission = pd.read_csv('/kaggle/input/mayo-clinic-strip-ai/sample_submission.csv')
submission.CE = preds.iloc[ : , : 1]
submission.LAA = preds.iloc[ : , 1: 2]
submission = submission.groupby("patient_id").mean()
submission = submission[["CE", "LAA"]].round(6).reset_index()
submission.fillna(0.5)
submission
```

Out[13]:

	patient_id	CE	LAA
0	006388	0.499393	0.500607
1	008e5c	0.549454	0.450546
2	00c058	0.447789	0.552211
3	01adc5	0.423078	0.576922

In [14]:
```
submission[["patient_id", "CE", "LAA"]].to_csv("submission.csv", index=False)
!head submission.csv
```

```
patient_id,CE,LAA
006388,0.499393,0.500607
008e5c,0.549454,0.450546
00c058,0.447789,0.552211
01adc5,0.423078,0.576922
```

In [ ]:
```

```