In [1]:
```
!nvidia-smi
from platform import python_version
print(python_version())
```

```
Tue Jul 11 19:17:26 2023
+---------------------------------------------------------------------------------
-------+
| NVIDIA-SMI 535.54.06              Driver Version: 536.40       CUDA Version: 1
2.2     |
|-----------------------------------------+----------------------+----------------
-------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncor
r. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Comp
ute M. |
|                                         |                      |
MIG M. |
|=========================================+======================+================
=======|
|   0  NVIDIA GeForce RTX 3060        On  | 00000000:01:00.0  On |
N/A |
| 45%   43C    P8              10W / 170W |   1029MiB / 12288MiB |      6%      D
efault |
|                                         |                      |
N/A |
+-----------------------------------------+----------------------+----------------
-------+

+---------------------------------------------------------------------------------
-------+
| Processes:
|
|  GPU   GI   CI        PID   Type   Process name                            GPU
Memory |
|        ID   ID                                                             Usag
e      |
|=================================================================================
=======|
|  No running processes found
|
+---------------------------------------------------------------------------------
-------+
3.11.3
```

In [2]:
```
import cv2
import copy
from ViT_CX.ViT_CX import ViT_CX
import numpy as np
import torch
import torch.nn as nn
from tqdm import tqdm
from torchvision import transforms
import torchvision.transforms as tt
from torchvision.utils import make_grid
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
#from pytorch_pretrained_vit import ViT
from matplotlib import pyplot as plt
```

```
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import KFold
```

In [3]:
```
print(torch.__version__)
```

```
2.0.1
```

In [4]:
```
# Model Hyperparameters
batch_size = 8 # 8 is maximum
```

In [5]:
```
# Device selection
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)

def to_device(data, device):
    """Move tensor(s) to chosen device"""
    if isinstance(data, (list,tuple)):
        return [to_device(x, device) for x in data]
    return data.to(device, non_blocking=True)

class DeviceDataLoader():
    """Wrap a dataloader to move data to a device"""
    def __init__(self, dl, device):
        self.dl = dl
        self.device = device

    def __iter__(self):
        """Yield a batch of data after moving it to device"""
        for b in self.dl:
            yield to_device(b, self.device)

    def __len__(self):
        """Number of batches"""
        return len(self.dl)
```

```
Using device: cuda
```

In [6]:
```
# Data transforms (normalization & data augmentation)
stats = (0.5, 0.5)

# Test Data transforms
test_tfms = tt.Compose([tt.Resize((224, 224), antialias='True'), tt.ToTensor(),

# Load Dataset from folder
data_dir = './data/pwcdata'
test_ds = ImageFolder(data_dir+'/test', test_tfms)

# PyTorch data loaders
test_dl = DataLoader(test_ds, batch_size*2, num_workers=2, pin_memory=False)

# Move data loaders to device
test_dl = DeviceDataLoader(test_dl, device)
```

In [7]:
```
# loss function
criterion = nn.CrossEntropyLoss()

def t_evaluate(model, test_dl):
    #model.eval()
    with torch.no_grad():
        epoch_val_accuracy = 0
```

```
        epoch_val_loss = 0
        for data, label in tqdm(test_dl):
            data = data #.to(device)
            label = label #.to(device)

            val_output = model(data)
            val_loss = criterion(val_output, label)

            acc = (val_output.argmax(dim=1) == label).float().mean()
            epoch_val_accuracy += acc / len(test_dl)
            epoch_val_loss += val_loss / len(test_dl)

    return {'test_loss': epoch_val_loss.item(), 'test_acc': epoch_val_accuracy.i
```

In [8]:

```
def test_model(model, test_dl):
    CM=0
    #model.eval()
    with torch.no_grad():
        for data, label in tqdm(test_dl):
            data = data #.to(device)
            labels = label #.to(device)

            outputs = model(data) #file_name
            preds = torch.argmax(outputs.data, 1)
            CM+=confusion_matrix(labels.cpu(), preds.cpu(),labels=[0,1])

        tn=CM[0][0]
        tp=CM[1][1]
        fp=CM[0][1]
        fn=CM[1][0]
        acc=np.sum(np.diag(CM)/np.sum(CM))
        sensitivity=tp/(tp+fn)
        precision=tp/(tp+fp)

        print('\nTestset Accuracy(mean): %f %%' % (100 * acc))
        print()
        print('Confusion Matirx : ')
        print(CM)
        print('- Sensitivity : ',(tp/(tp+fn))*100)
        print('- Specificity : ',(tn/(tn+fp))*100)
        print('- Precision: ',(tp/(tp+fp))*100)
        print('- NPV: ',(tn/(tn+fn))*100)
        print('- F1 : ',((2*sensitivity*precision)/(sensitivity+precision))*100)
        print()

    return acc, CM
```

In [9]:

```
def predict_image(img, model):
    # Convert to a batch of 1
    xb = to_device(img.unsqueeze(0), device)
    # Get predictions from model
    yb = model(xb)
    # Pick index with highest probability
    _, preds  = torch.max(yb, dim=1)
    # Retrieve the class label
    # test_ds.classes[label]
    return test_ds.classes[preds]
```

In [10]:
```
# Load black box model for explanations
model = torch.load('./data/pwc_ViT_timm_pure_10_folds')
```

In [11]:
```
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=FutureWarning)
```

In [12]:
```
tn_save_path = f'./data/vitcx_img/pure/tn/'
tp_save_path = f'./data/vitcx_img/pure/tp/'
fp_save_path = f'./data/vitcx_img/pure/fp/'
fn_save_path = f'./data/vitcx_img/pure/fn/'
CM = [[0,0],[0,0]]
tn = 0
tp = 0
fp = 0
fn = 0
#model.eval()
#with torch.no_grad():
for i in tqdm(range(0, 1144)):
    #print(f'img : {i}')
    img, label = test_ds[i]
#         img = to_device(data.unsqueeze(0), device) #.to(device)
#         labels = label #.to(device)
    img = to_device(img.unsqueeze(0), device)
#         yb = model(xb)
#         _, preds  = torch.max(yb, dim=1)

    outputs = model(img) #file_name
    preds = torch.argmax(outputs.data, 1)
    target_layer=model.blocks[-1].norm1
    result=ViT_CX(model,img,target_layer,target_category=None,distance_thresholc

    if ((label == 0) and (preds == 0)):
        # true negative
        tn +=1
        img_path = tn_save_path + f'{i+1:05d}' + '.png'
        plt.imsave(img_path, result, cmap='jet')
    elif ((label == 1) and (preds == 1)):
        # true positive
        tp += 1
        img_path = tp_save_path + f'{i+1:05d}' + '.png'
        plt.imsave(img_path, result, cmap='jet')
    elif ((label == 0) and (preds == 1)):
        # false positive
        fp += 1
        img_path = fp_save_path + f'{i+1:05d}' + '.png'
        plt.imsave(img_path, result, cmap='jet')
    elif ((label == 1) and (preds == 0)):
        # false negative
        fn += 1
        img_path = fn_save_path + f'{i+1:05d}' + '.png'
        plt.imsave(img_path, result, cmap='jet')
```

```
100%|██████████████████████████████████████████████████| 1144/11
44 [06:19<00:00,  3.01it/s]
```

In [13]:
```
CM[0][0] = tn
CM[1][1] = tp
CM[0][1] = fp
```

```
CM[1][0] = fn
print(CM)
```

```
[[540, 21], [63, 520]]
```

In [14]:
```
acc=np.sum(np.diag(CM)/np.sum(CM))
sensitivity=tp/(tp+fn)
precision=tp/(tp+fp)

print('\nTestset Accuracy(mean): %f %%' % (100 * acc))
print()
print('Confusion Matirx : ')
print(CM)
print('- Sensitivity : ',(tp/(tp+fn))*100)
print('- Specificity : ',(tn/(tn+fp))*100)
print('- Precision: ',(tp/(tp+fp))*100)
print('- NPV: ',(tn/(tn+fn))*100)
print('- F1 : ',((2*sensitivity*precision)/(sensitivity+precision))*100)
print()
```

```
Testset Accuracy(mean): 92.657343 %

Confusion Matirx :
[[540, 21], [63, 520]]
- Sensitivity :  89.19382504288164
- Specificity :  96.2566844919786
- Precision:  96.11829944547135
- NPV:  89.55223880597015
- F1 :  92.52669039145907
```

In [ ]:
```
# t_result = t_evaluate(model, test_dl)
# t_result
```

In [ ]:
```
#test_model(model, test_dl)
```

In [ ]:
```
# img, label = test_ds[0]
# plt.imshow(img.permute(1, 2, 0).clamp(0, 1))
# plt.axis('off')
# print('Label:', test_ds.classes[label], ', Predicted:', predict_image(img, moa
```

In [16]:
```
# # Perform ViT-CX
# target_layer=model.blocks[-1].norm1
# result=ViT_CX(model,img.unsqueeze(0),target_layer,target_category=None,distanc
```

In [ ]:
```
# %matplotlib inline
# fig, ax = plt.subplots(1,2)
# ax[0].axis('off')
# ax[1].axis('off')
# ax[0].imshow(img.permute(1, 2, 0).clamp(0, 1))
# ax[1].imshow(result, cmap='jet', alpha=1)
# plt.show()
```

In [ ]:
```
# img, label = test_ds[1100]
# plt.axis('off')
# plt.imshow(img.permute(1, 2, 0).clamp(0, 1))
# print('Label:', test_ds.classes[label], ', Predicted:', predict_image(img, moa
```

In [ ]:
```
# # Perform ViT-CX
# target_layer=model.blocks[-1].norm1
# result=ViT_CX(model,img.unsqueeze(0),target_layer,target_category=None,distanc
```

In [ ]:
```
# %matplotlib inline
# fig, ax = plt.subplots(1,2)
# ax[0].axis('off')
# ax[1].axis('off')
# ax[0].imshow(img.permute(1, 2, 0).clamp(0, 1))
# ax[1].imshow(result, cmap='jet', alpha=1)
# plt.show()
```