Image data (BRCA .svs files) directory:

/research/bsi/projects/breast/s301449.LARdl/processing/harshini/he2RNAdata1/

FPKM-UQ data (BRCA) directory:

/research/bsi/projects/breast/s301449.LARdl/processing/harshini/FPKM-UQ

Openslide Directory:

/research/bsi/tools/biotools/openslide/3.4.1/miniconda3/bin/python

Flow of TRNAsformers:

Patches -> Tissue Masks -> K-Means clustering based on Histograms->

Amirs approach(based on coordinates(special clustering)

Clusters = 49-> Features extracted based on Densenet121 -> 49\*1024->

Bags of instances created such

that there are 7 rows and 7 columns-> 224\*224 ->

with each instance 32\*32

Random Sampling done based by taking samples from each cluster ->

.npy file generated for each instance created based on resampling ->

fed into vision Transformer and train.py file is run.

Dataset creation .csv file with image\_gene, gene\_expression is created both .npy format

Amir’s Comments:

1. The preset arguments for the code should be set in arguments.py. To train the model, use the train.py. The gene\_pred\_eval.py file is for the evaluation of the gene prediction. The wsi\_classification\_eval.py is for classification task evaluation.  
   These are the main files you may need to run the experiments.  
   The model definitions are saved in the "models" folder.
2. Using Yottixel for preprocessing may give better results.

The numpy array format for the instances should be in format:

Features can be read using;

features = np.load(instance, allow\_pickle=True)[()]['feature']

{'feature': array([[0.        , 1.9029806 , 0.        , ..., 0.01972643, 0.        ,  
        0.55768365],  
       [0.        , 0.01043449, 0.        , ..., 0.0267658 , 0.7615977 ,  
        0.        ],  
       [0.        , 0.8107864 , 0.01523623, ..., 0.02649011, 0.06823773,  
        0.        ],  
       ...,  
       [0.45817223, 0.0175302 , 0.        , ..., 8.085226  , 0.04961037,  
        2.0368629 ],  
       [0.        , 0.        , 0.        , ..., 5.0472517 , 0.        ,  
        0.        ],  
       [0.        , 0.        , 0.03634823, ..., 0.14657755, 0.        ,  
        0.05196447]], dtype=float32), 'locations': {20: array([[ 420,  476],  
       [ 224,  560],  
       [ 812,  420],  
       [ 812,  560],  
       [ 784,  728],  
       [1260,  168],  
       [ 504, 1064],  
       [1176,  532],  
       [1204,  896],  
       [ 896, 1064],  
       [1484,  112],  
       [ 504, 1400],  
       [1512,  868],  
       [1568,  364],  
       [ 924, 1400],  
       [1092, 1344],  
       [1288, 1232],  
       [1876,   28],  
       [ 532, 1652],  
       [1904,  868],  
       [1988,  336],  
       [1736, 1260],  
       [ 840, 2016],  
       [2072,   84],  
       [ 672, 2296],  
       [1232, 1988],  
       [2212, 1260],  
       [2408,  560],  
       [2268,  980],  
       [ 924, 2212],  
       [1400, 2324],  
       [2240, 1596],  
       [1792, 2184],  
       [2436,  896],  
       [2604,  644],  
       [2100, 2100],  
       [2436, 1120],  
       [2408, 1680],  
       [1764, 2324],  
       [2940,  980],  
       [2576, 1764],  
       [3052, 1064],  
       [2240, 2380],  
       [2744, 1400],  
       [2184, 2128],  
       [2772, 1764],  
       [2828, 2100],  
       [2436, 2324],  
       [2856, 2296]], dtype=int32)}}

Things that need to be done:

1. Resampling features from bag of instances to create instances.

* Criteria for Resampling: Randomly picking features from each cluster.

1. Matching the instances to FPKM data based on UUID.

Code Files:

Preprocess\_numpy.ipynb: All the detailed steps for preprocessing

Preprocess\_loop.py -> code to propressing all the svs file and convert to .npy format

Problems;

Sometimes the code is giving different features after training using densenet121 though we mentioned that we need only 49 clusters.