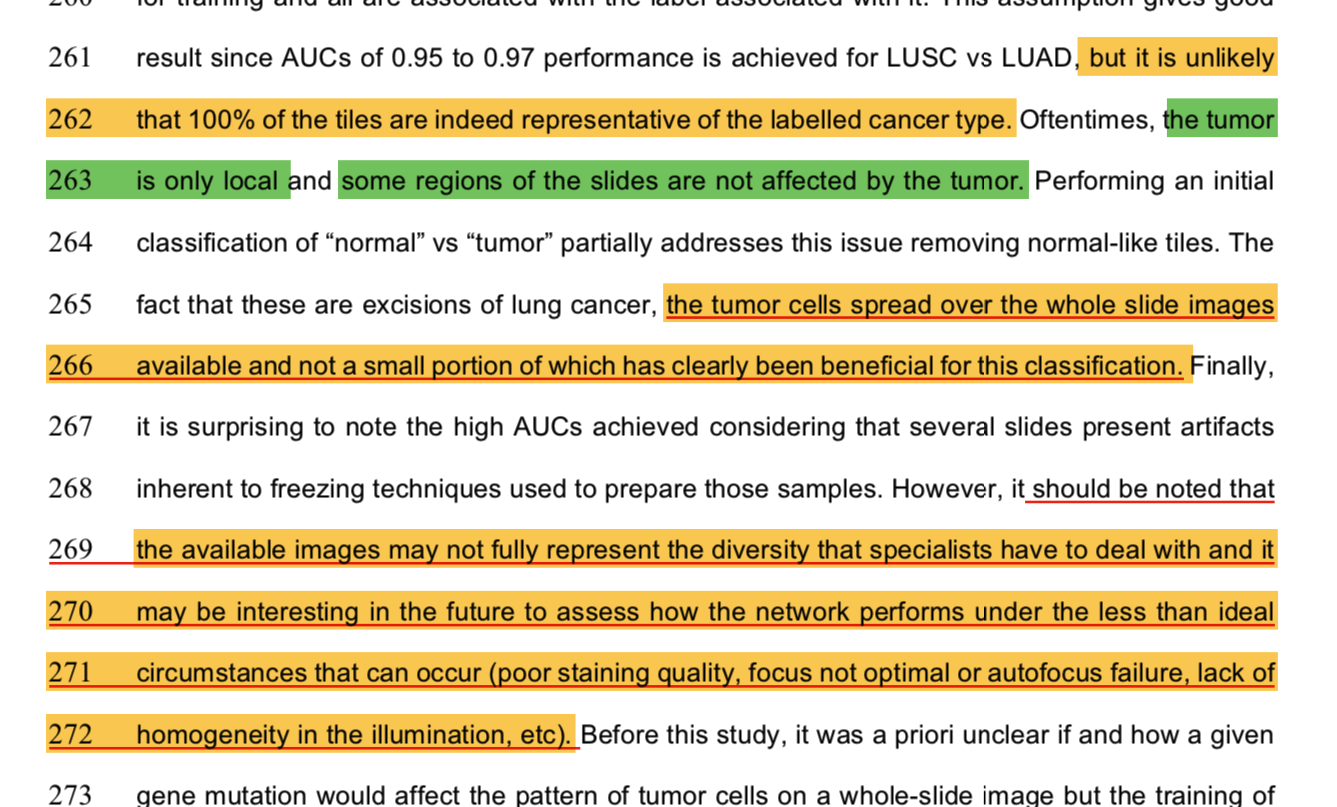
Predicting gene mutational status from whole slide images.

Feb28th/2020

* The result from rerunning the model by changing the preprocessing parameters:
  + Not much change for micro and macro AUC/ROC values.
  + Still have NA values for each Class files.
  + Reasons:
    - Computational:

ROC/AUC calculation script in: </nfs/home/xwang/DeepPATH/DeepPATH\_code/03\_postprocessing/0h\_ROC\_MultiOutput\_BootStrap.py>

* + - * Is not working / not generalize well for our datasets, this may due to fact that differences in image generations. Somehow this script does not have generalizability for our image dataset from Providence.
    - Biological:
      * The model trained on TCGA datasets and tested on their own testing set for different biopsy images. They may figure out the same preprocess step for images they trained on. For example, their images may obtain from similar lung cancer stage or microenvironment.
      * Biologically, the staining process on these images may be different and the cancer stages for these images may be different in clinical definition that causing bad classifying results.



* **Next Step:** Gene mutation prediction from image.
  + Only using LUAD tiles for gene mutation prediction to see what’s the results.

From <https://github.com/ncoudray/DeepPATH/tree/master/DeepPATH_code/example_TCGA_lung>:

To process mutations of LUAD images, there are different ways to do it.

First, to extract probability of LUAD tiles on all LUAD tiles, we'll run them through the above classifier:

Sort the tiles, assigning them all to "test".

"""

mkdir r2\_LUAD\_segmentation

cd r2\_LUAD\_segmentation

python 00\_preprocessing/0d\_SortTiles.py --SourceFolder='../512px\_Tiled/' --Magnification=20.0 --MagDiffAllowed=0 --SortingOption=3 --PatientID=12 --nSplit 0 --JsonFile='../downloaded\_data/metadata.cart.2017-03-02T00\_36\_30.276824.json' --PercentTest=100 --PercentValid=0

"""

Since Normal and LUSC do not interest us, delete their content (the content only - not the folder - the number of folders in that directory is used to identify the total number of possible classes):

"""

rm -rf TCGA-lUSC/\*

rm -rf Solid\_Tissue\_Normal/\*

"""

convert to TFRecord:

"""

mkdir r2\_TFRecord\_test

python 00\_preprocessing/TFRecord\_2or3\_Classes/build\_TF\_test.py --directory='r2\_LUAD\_segmentation/' --output\_directory='r2\_TFRecord\_test' --num\_threads=1 --one\_FT\_per\_Tile=False --ImageSet\_basename='test'

"""

**Segment the LUAD tiles using the checkpoint giving the best validation/test AUC.**

"""

export CHECKPOINT\_PATH='r1\_results'

export OUTPUT\_DIR='r2\_test'

export DATA\_DIR='r2\_TFRecord\_test'

export LABEL\_FILE='labelref\_r1.txt'

# Best checkpoints

declare -i count=69000

declare -i NbClasses=3

# create temporary directory for checkpoints

mkdir -p $OUTPUT\_DIR/tmp\_checkpoints

export CUR\_CHECKPOINT=$OUTPUT\_DIR/tmp\_checkpoints

export TEST\_OUTPUT=$OUTPUT\_DIR/test\_$count'k'

mkdir -p $TEST\_OUTPUT

ln -s $CHECKPOINT\_PATH/\*-$count.\* $CUR\_CHECKPOINT/.

touch $CUR\_CHECKPOINT/checkpoint

echo 'model\_checkpoint\_path: "'$CUR\_CHECKPOINT'/model.ckpt-'$count'"' > $CUR\_CHECKPOINT/checkpoint

echo 'all\_model\_checkpoint\_paths: "'$CUR\_CHECKPOINT'/model.ckpt-'$count'"' >> $CUR\_CHECKPOINT/checkpoint

# Test

python 02\_testing/xClasses/nc\_imagenet\_eval.py --checkpoint\_dir=$CUR\_CHECKPOINT --eval\_dir=$OUTPUT\_DIR --data\_dir=$DATA\_DIR --batch\_size 300 --run\_once --ImageSet\_basename='test\_' --ClassNumber $NbClasses --mode='0\_softmax' --TVmode='test'

# wait

mv $OUTPUT\_DIR/out\* $TEST\_OUTPUT/.

"""

Retrieve the mutation information from the GDC website. In this particular example, we use mutect2 "masked somatic mutations". We **label samples/slides as mutated with respect to every gene if** it had a non-silent mutation. We used maftools to parse the Mutect2 variants from TCGA which by default uses Variant Classifications with High/Moderate variant consequences. These include: "Frame\_Shift\_Del", "Frame\_Shift\_Ins", "Splice\_Site", "Translation\_Start\_Site", "Nonsense\_Mutation", "Nonstop\_Mutation", "In\_Frame\_Del", "In\_Frame\_Ins", "Missense\_Mutation". We then picked the top 10 "known cancer genes" (https://cancer.sanger.ac.uk/census) with respect to the number of (non-silent) mutation across our dataset, excluding genes like TNN which are known to be frequently mutated in general (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4267152/). **We can then generate a label file where the first column is the slide ID, and the second the mutation name (if a slide has several mutations, then it will have several lines - example in file attached labels\_r3.txt), and a reference file with the list of possible mutations (see labelref\_r3.txt).**

sort the LUAD tiles identified as LUAD intro a train, valid a test set for mutation analysis.

"""

mkdir r3\_LUAD\_sorted

cd r3\_LUAD\_sorted

python ../00\_preprocessing/0d\_SortTiles.py --SourceFolder='../512px\_Tiled\_NewPortal/' --Magnification=20 --MagDiffAllowed=0 --SortingOption=10 --PatientID=-1 --PercentTest=15 --PercentValid=15 --nSplit 0 --outFilenameStats='../r2\_test/test\_69000k/out\_filename\_Stats.txt'

"""

Convert to TFRecord:

"""

# valid

python 00\_preprocessing/TFRecord\_multi\_Classes/build\_TF\_test\_multiClass.py --directory='r3\_LUAD\_sorted/512px\_Tiled\_NewPortal/' --output\_directory='r3\_TFRecord\_valid' --num\_threads=1 --one\_FT\_per\_Tile=False --ImageSet\_basename='valid' --labels\_names='labelref\_r3.txt' --labels='labels\_r3.txt' --PatientID=14

# test

python 00\_preprocessing/TFRecord\_multi\_Classes/build\_TF\_test\_multiClass.py --directory='r3\_LUAD\_sorted/512px\_Tiled\_NewPortal' --output\_directory='r3\_TFRecord\_test' --num\_threads=1 --one\_FT\_per\_Tile=False --ImageSet\_basename='test' --labels\_names='labelref\_r3.txt' --labels='labels\_r3.txt' --PatientID=14

# train

python 00\_preprocessing/TFRecord\_multi\_Classes/build\_image\_data\_multiClass.py --directory='r3\_LUAD\_sorted/512px\_Tiled\_NewPortal' --output\_directory='r3\_TFRecord\_train' --train\_shards=1024 --validation\_shards=128 --num\_threads=16 --labels\_names='labelref\_r3.txt' --labels='labels\_r3.txt' --PatientID=14

"""

**train the model with 10-class sigmoid classifier:**

"""

bazel-bin/inception/imagenet\_train --num\_gpus=4 --batch\_size=400 --train\_dir="r3\_results\_train" --data\_dir="r3\_TFRecord\_train" --ClassNumber=10 --mode='1\_sigmoid' --NbrOfImages=326613 --save\_step\_for\_chekcpoint=815 --max\_steps=81501

"""

**once the checkpoints start being saved, we can start runing the valid and test sets:**

"""

export CHECKPOINT\_PATH='full\_ath\_to/r3\_results\_train/'

export OUTPUT\_DIR='full\_path\_to/r3\_valid'

export DATA\_DIR='r3\_TFRecord\_valid'

export LABEL\_FILE='labelref\_r3.txt'

# create temporary directory for checkpoints

mkdir -p $OUTPUT\_DIR/tmp\_checkpoints

export CUR\_CHECKPOINT=$OUTPUT\_DIR/tmp\_checkpoints

# check if next checkpoint available

declare -i count=815

declare -i step=815

declare -i NbClasses=10

while true; do

echo $count

if [ -f $CHECKPOINT\_PATH/model.ckpt-$count.meta ]; then

echo $CHECKPOINT\_PATH/model.ckpt-$count.meta " exists"

# check if there's already a computation for this checkpoinmt

export TEST\_OUTPUT=$OUTPUT\_DIR/test\_$count'k'

if [ ! -d $TEST\_OUTPUT ]; then

mkdir -p $TEST\_OUTPUT

ln -s $CHECKPOINT\_PATH/\*-$count.\* $CUR\_CHECKPOINT/.

touch $CUR\_CHECKPOINT/checkpoint

echo 'model\_checkpoint\_path: "'$CUR\_CHECKPOINT'/model.ckpt-'$count'"' > $CUR\_CHECKPOINT/checkpoint

echo 'all\_model\_checkpoint\_paths: "'$CUR\_CHECKPOINT'/model.ckpt-'$count'"' >> $CUR\_CHECKPOINT/checkpoint

# Test

python /gpfs/scratch/coudrn01/NN\_test/code/DeepPATH/DeepPATH\_code/02\_testing/xClasses/nc\_imagenet\_eval.py --checkpoint\_dir=$CUR\_CHECKPOINT --eval\_dir=$OUTPUT\_DIR --data\_dir=$DATA\_DIR --batch\_size 200 --run\_once --ImageSet\_basename='valid\_' --ClassNumber $NbClasses --mode='1\_sigmoid' --TVmode='test'

# wait

mv $OUTPUT\_DIR/out\* $TEST\_OUTPUT/.

# ROC

export OUTFILENAME=$TEST\_OUTPUT/out\_filename\_Stats.txt

python /gpfs/scratch/coudrn01/NN\_test/code/DeepPATH/DeepPATH\_code/03\_postprocessing/0h\_ROC\_MultiOutput\_BootStrap.py --file\_stats=$OUTFILENAME --output\_dir=$TEST\_OUTPUT --labels\_names=$LABEL\_FILE

else

echo 'checkpoint '$TEST\_OUTPUT' skipped'

fi

else

echo $CHECKPOINT\_PATH/model.ckpt-$count.meta " does not exist"

break

fi

# next checkpoint

count=`expr "$count" + "$step"`

done

# summarize all AUC per slide (average probability) for class 1:

ls -tr $OUTPUT\_DIR/test\_\*/out2\_roc\_data\_AvPb\_c1a\* | sed -e 's/k\/out2\_roc\_data\_AvPb\_c1a/ /' | sed -e 's/test\_/ /' | sed -e 's/\_/ /g' | sed -e 's/.txt//' > $OUTPUT\_DIR/valid\_out2\_AvPb\_AUCs\_1.txt

# summarize all AUC per slide (average probability) for macro average:

ls -tr $OUTPUT\_DIR/test\_\*/out2\_roc\_data\_AvPb\_macro\* | sed -e 's/k\/out2\_roc\_data\_AvPb\_macro\_/ /' | sed -e 's/test\_/ /' | sed -e 's/\_/ /g' | sed -e 's/.txt//' > $OUTPUT\_DIR/valid\_out2\_AvPb\_AUCs\_macro.txt

# summarize all AUC per slide (average probability) for micro average:

ls -tr $OUTPUT\_DIR/test\_\*/out2\_roc\_data\_AvPb\_micro\* | sed -e 's/k\/out2\_roc\_data\_AvPb\_micro\_/ /' | sed -e 's/test\_/ /' | sed -e 's/\_/ /g' | sed -e 's/.txt//' > $OUTPUT\_DIR/valid\_out2\_AvPb\_AUCs\_micro.txt

ls -tr $OUTPUT\_DIR/test\_\*/out2\_roc\_data\_AvPb\_c2\* | sed -e 's/k\/out2\_roc\_data\_AvPb\_c2/ /' | sed -e 's/test\_/ /' | sed -e 's/\_/ /g' | sed -e 's/.txt//' > $OUTPUT\_DIR/valid\_out2\_AvPb\_AUCs\_2.txt

ls -tr $OUTPUT\_DIR/test\_\*/out2\_roc\_data\_AvPb\_c3\* | sed -e 's/k\/out2\_roc\_data\_AvPb\_c3/ /' | sed -e 's/test\_/ /' | sed -e 's/\_/ /g' | sed -e 's/.txt//' > $OUTPUT\_DIR/valid\_out2\_AvPb\_AUCs\_3.txt

ls -tr $OUTPUT\_DIR/test\_\*/out2\_roc\_data\_AvPb\_c4\* | sed -e 's/k\/out2\_roc\_data\_AvPb\_c4/ /' | sed -e 's/test\_/ /' | sed -e 's/\_/ /g' | sed -e 's/.txt//' > $OUTPUT\_DIR/valid\_out2\_AvPb\_AUCs\_4.txt

ls -tr $OUTPUT\_DIR/test\_\*/out2\_roc\_data\_AvPb\_c5\* | sed -e 's/k\/out2\_roc\_data\_AvPb\_c5/ /' | sed -e 's/test\_/ /' | sed -e 's/\_/ /g' | sed -e 's/.txt//' > $OUTPUT\_DIR/valid\_out2\_AvPb\_AUCs\_5.txt

ls -tr $OUTPUT\_DIR/test\_\*/out2\_roc\_data\_AvPb\_c6\* | sed -e 's/k\/out2\_roc\_data\_AvPb\_c6/ /' | sed -e 's/test\_/ /' | sed -e 's/\_/ /g' | sed -e 's/.txt//' > $OUTPUT\_DIR/valid\_out2\_AvPb\_AUCs\_6.txt

ls -tr $OUTPUT\_DIR/test\_\*/out2\_roc\_data\_AvPb\_c7\* | sed -e 's/k\/out2\_roc\_data\_AvPb\_c7/ /' | sed -e 's/test\_/ /' | sed -e 's/\_/ /g' | sed -e 's/.txt//' > $OUTPUT\_DIR/valid\_out2\_AvPb\_AUCs\_7.txt

ls -tr $OUTPUT\_DIR/test\_\*/out2\_roc\_data\_AvPb\_c8\* | sed -e 's/k\/out2\_roc\_data\_AvPb\_c8/ /' | sed -e 's/test\_/ /' | sed -e 's/\_/ /g' | sed -e 's/.txt//' > $OUTPUT\_DIR/valid\_out2\_AvPb\_AUCs\_8.txt

ls -tr $OUTPUT\_DIR/test\_\*/out2\_roc\_data\_AvPb\_c9\* | sed -e 's/k\/out2\_roc\_data\_AvPb\_c9/ /' | sed -e 's/test\_/ /' | sed -e 's/\_/ /g' | sed -e 's/.txt//' > $OUTPUT\_DIR/valid\_out2\_AvPb\_AUCs\_9.txt

ls -tr $OUTPUT\_DIR/test\_\*/out2\_roc\_data\_AvPb\_c10\* | sed -e 's/k\/out2\_roc\_data\_AvPb\_c10/ /' | sed -e 's/test\_/ /' | sed -e 's/\_/ /g' | sed -e 's/.txt//' > $OUTPUT\_DIR/valid\_out2\_AvPb\_AUCs\_10.txt

"""

A similar code can be used for the test check by modifying the corresponding options and inputs.

**labelref\_r3.txt**

EGFR

FAT1

FAT4

KEAP1

KRAS

LRP1B

NF1

SETBP1

STK11

TP53

run1b\_10way\_MutationClassifier

**labels\_r3.txt**

TCGA-95-7039-0 TP53

TCGA-95-7039-0 LRP1B

TCGA-95-7039-0 KRAS

TCGA-95-7039-0 EGFR

TCGA-95-7039-0 FAT1

TCGA-95-7567-0 TP53

TCGA-95-7567-0 LRP1B

TCGA-95-7567-0 KRAS

TCGA-95-7567-0 FAT4

TCGA-05-4427-0 TP53

TCGA-05-4427-0 LRP1B

TCGA-05-4427-0 KRAS

TCGA-05-4427-0 SETBP1

TCGA-64-5778-0 TP53

TCGA-64-5778-0 LRP1B

TCGA-64-5778-0 KRAS

TCGA-55-8507-0 TP53

TCGA-55-8507-0 LRP1B

TCGA-55-8507-0 FAT4

TCGA-55-8507-0 STK11

TCGA-MN-A4N4-0 TP53

TCGA-MN-A4N4-0 LRP1B

TCGA-05-4382-0 TP53

TCGA-05-4382-0 LRP1B

TCGA-05-4382-0 FAT4

TCGA-05-4382-0 EGFR

TCGA-05-4382-0 SETBP1

TCGA-05-4398-0 TP53

TCGA-05-4398-0 LRP1B

TCGA-05-4398-0 FAT1

TCGA-05-4398-0 SETBP1

TCGA-49-AAR4-0 TP53

TCGA-49-AAR4-0 LRP1B

TCGA-49-AAR4-0 FAT1

* Figure out if we need at least one gene mutation label for each tile name.
  + Try first with tile labels only from tiles had KRAS mutation.
  + Then, if not working properly, use other gene name to represent tiles without KRAS mutation.
* Need to pull out the KRAS mutation Patient ID ---- also Image ID ---- Tile ID list / dict / table. In order to generate this label file.