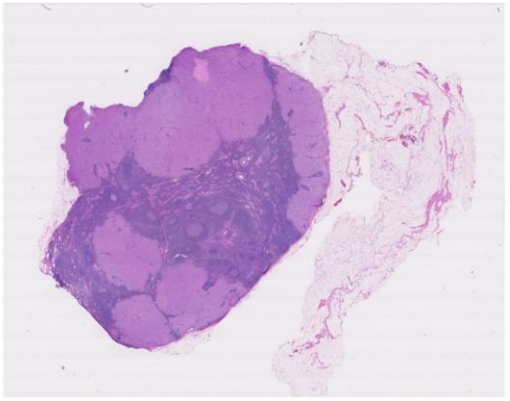
Detecting Cancer Metastases On Gigapixel Pathology Images

Aditya Natarajan (an2867)

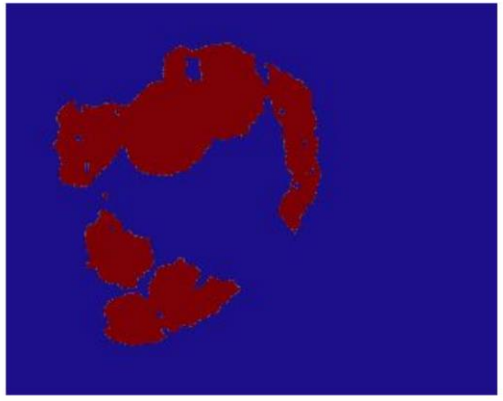
Aashish Misraa (akm2215)

PROBLEM STATEMENT

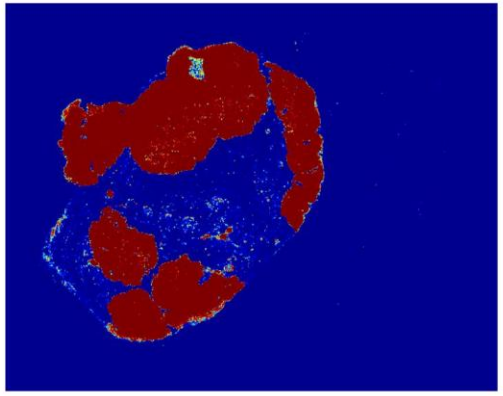
- Develop a deep learning model that outputs a heatmap showing regions of a biopsy image likely to contain cancer
- We explore different deep learning techniques to arrive at a solution that can be used to ***assist*** in diagnosis



Biopsy image

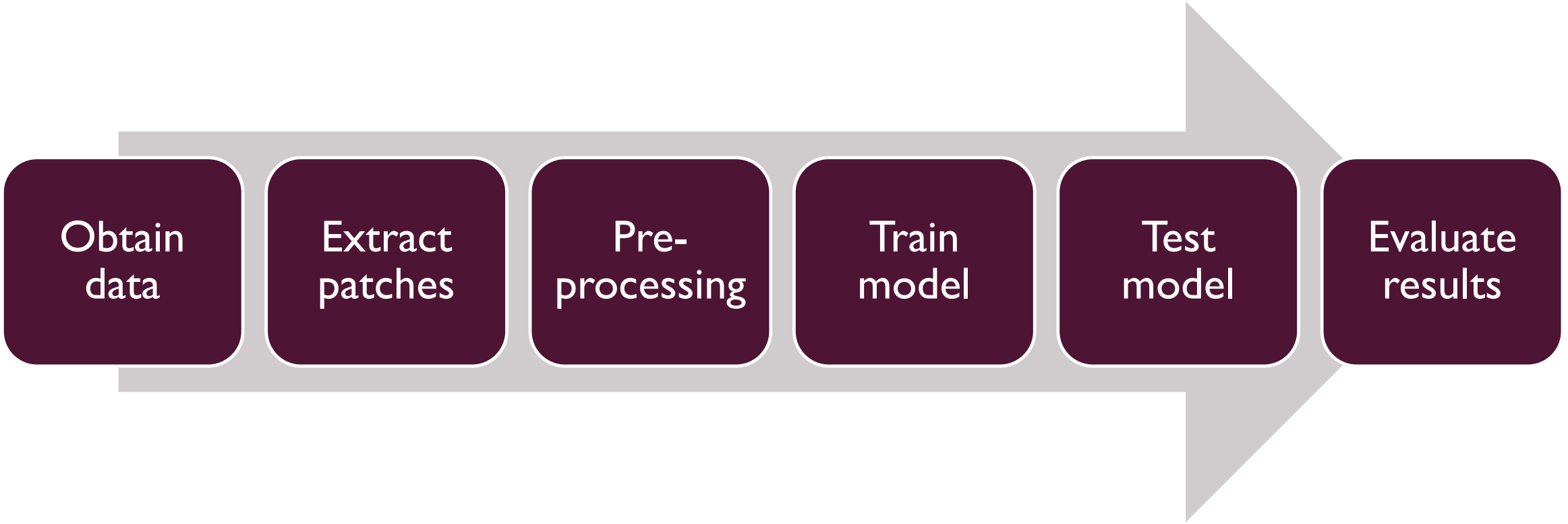


Ground truth
(from pathologist)



Model predictions

WORKFLOW


Obtain data
Extract patches
Pre-processing
Train model
Test model
Evaluate results

OBTAINING DATA

- CAMELYON 16 dataset
- The training and testing data are a single slide each (randomly chosen)
  - Training slide : tumor_078.tif (and associated mask)
  - Testing slide : tumor_110.tif (and associated mask for evaluation)
- Choose two levels per slide : levels 3 and 6 (could not use more due to memory issues!)
  - Downsample factor for level 3 is 8
  - Downsample factor for level 6 is 64

EXTRACTING PATCHES

- Patch size : 100 x 100
- Central window size : 60 x 60 (To extract the mask context, referenced from the original paper)
- Stride : 60
- Patches are extracted using a sliding window technique

PRE - PROCESSING

- Data augmentation:
  - Rotating the patch by 90 degrees (4 times, so all rotations are obtained)
- Sampling
  - To combat the class imbalance problem – there are far more non-cancer patches than there are cancer patches (anywhere between 1 : 10 to 1 : 15, depending on the extraction method)
  - The non cancer patches were added to the training data if a randomly generated choice was above a certain threshold
    - Threshold value was set through trial-and-error

PRE – PROCESSING (CONTD.)

- We finally obtain the patches across the different levels, shuffle them and create a final training set of patches and labels
  - Label 1 for tumor cell
  - Label 0 for normal cell
- The data is now ready to be used for training the model

TRAIN MODEL

- Use the ***same*** model to train patches across both levels (Deviation from the paper, which uses different InceptionV3 towers for each level)
  - Done due to memory restrictions (Colab runs out of RAM)
- Use transfer learning with InceptionV3 (as the base)
- Model is NOT trained on ImageNet weights!
- Base is followed by Global Average Pooling, 2 Dense layers(relu and sigmoid) and 0.2 Dropout

TRAIN MODEL (CONTD.)

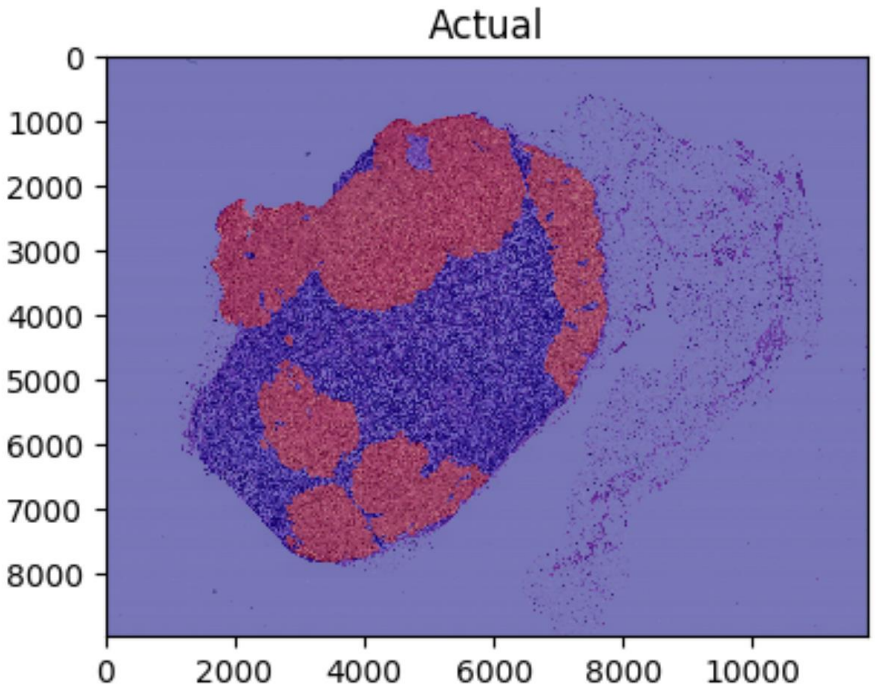
- Model parameters:
  - Optimizer : RMSprop (learning rate is 0.0001)
  - Loss : Binary Crossentropy
  - Metric : Accuracy
  - Number of epochs : 20
  - Batch size : 64 (also experiment with 32)
- Use model.fit() 's validation_split feature to divide data into 75% for train, 25% for validation

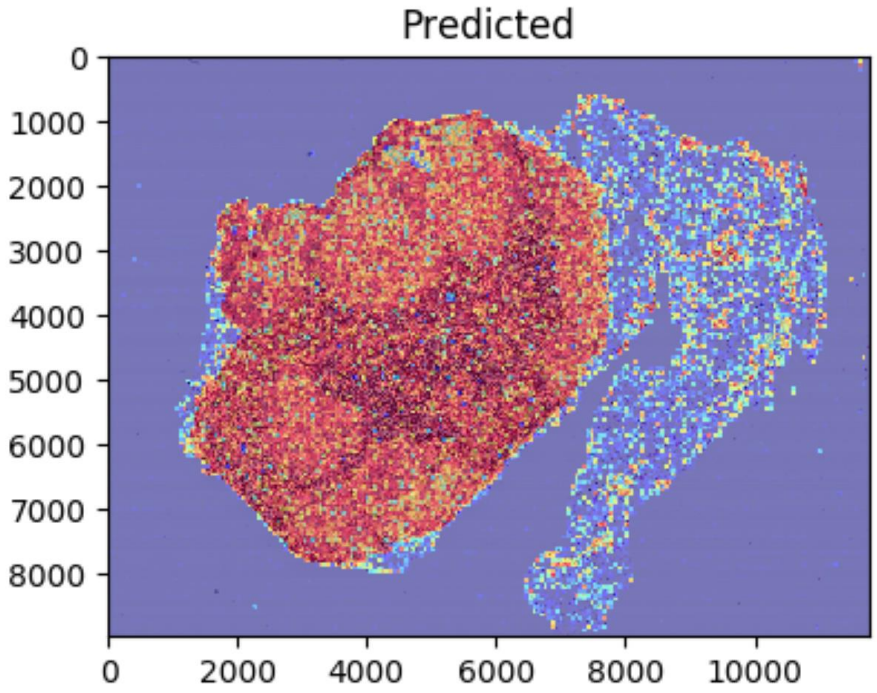
MODEL SUMMARY

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Model)         (None, 1, 1, 2048)        21802784
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 128)               262272
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 22,065,185
Trainable params: 262,401
Non-trainable params: 21,802,784
_________________________________________________________________
```

TEST MODEL

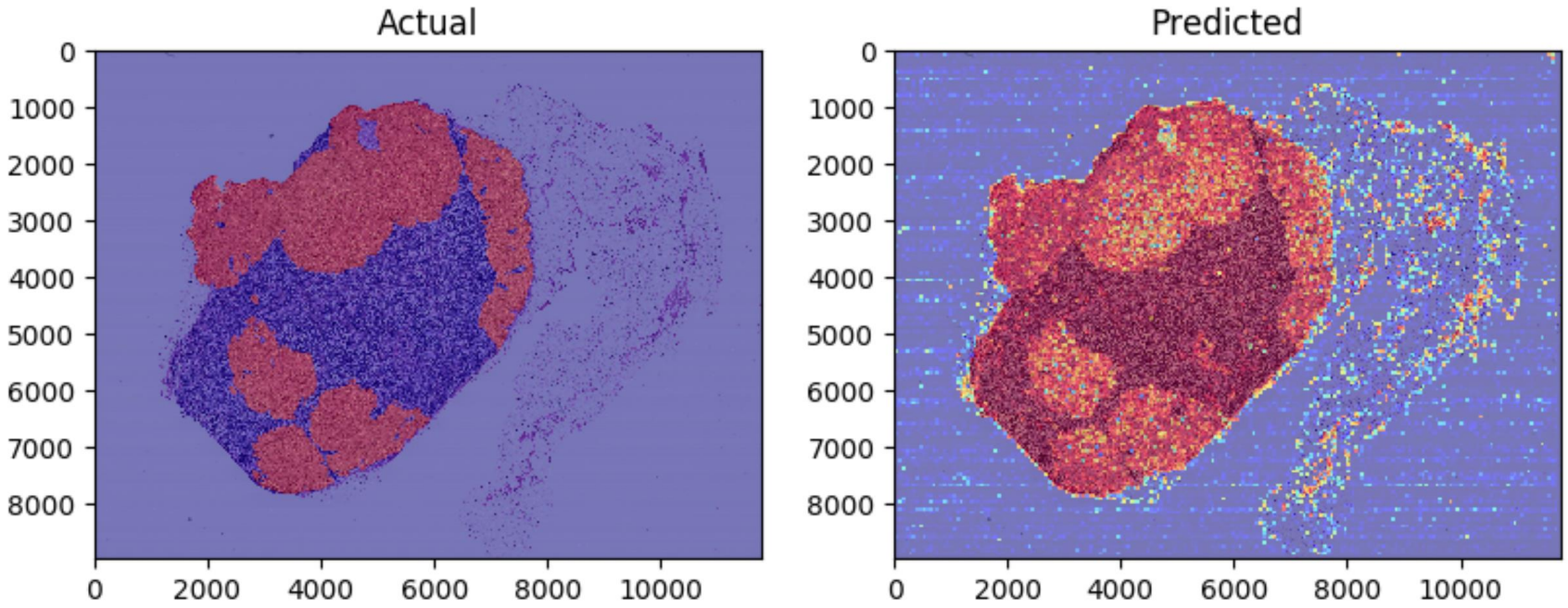
- Model predicts on a test slide at level 3 (uses a similar patch extraction approach)
- Model performance can be improved!





For 20 epochs

TEST MODEL (CONTD.)

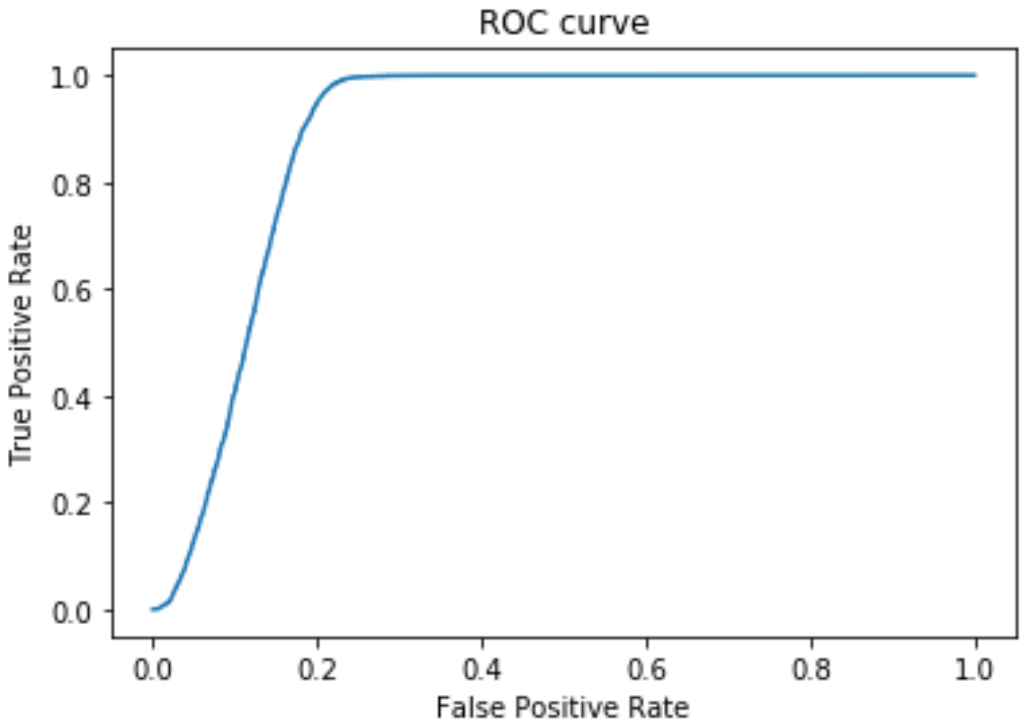


For 60 epochs

EVALUATION

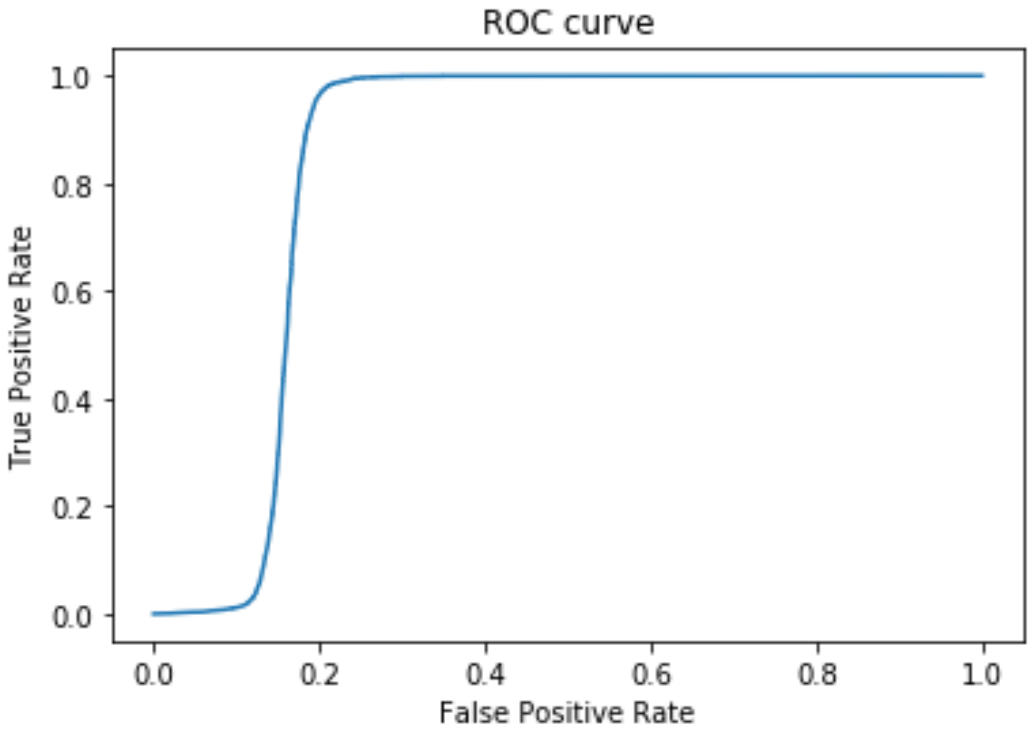
- Use Precision, Recall and AUC score
- Plot an ROC curve
- Display the confusion matrix

EVALUATION FOR 20 EPOCHS



AUC : 0.885454056201899
Threshold : 0.4063341021537781
CONFUSION MATRIX
[[69306304 19757455]
 [246256 16202945]]
True Positive : 16202945
True Negative : 69306304
False Positive : 19757455
False Negative : 246256
Precision : 0.45057744074037
Recall : 0.9850293032470088

EVALUATION FOR 60 EPOCHS



AUC : 0.8401566635430748
Threshold : 0.45884621143341064
CONFUSION MATRIX
[[70509827 18553932]
 [342333 16106868]]
True Positive : 16106868
True Negative : 70509827
False Positive : 18553932
False Negative : 342333
Precision : 0.46469983381803076
Recall : 0.9791884724370503

OBSERVATIONS AND TAKEAWAYS

- Smaller patch sizes seem to work better
  - Tried various patch sizes from 90 x 90 to 200 x 200 – smaller patches yielded best results
- Model can identify tissue correctly (no red patches outside the tissue)
- Model performs well on cancerous cells
  - Evidenced from high recall and relatively lower false negatives
- Model does not perform well on non-cancerous cells
  - Evidenced from high false positives
- KEY TAKEAWAY : Model performs reasonably well, but can be improved
  - Better to mislabel cell as cancerous than the other way round – can compromise on precision if it leads to high recall (which our model follows)

OTHER APPROACHES ATTEMPTED

- Approach : Select only patches from the slide with tissue percent above a specified threshold (say 10%)
  - Observation : Did not significantly impact performance
- Approach : Use patches from multiple slides (10+), at a higher level, say 7
  - Observation : Yielded less accurate results

OTHER APPROACHES ATTEMPTED (CONTD.)

- Approach : Train different models for different levels, and then ensemble them
  - Observation : Ran out of RAM in Colab

- Approach : Extreme case – set patch stride as 1 (Extracted 200000+ patches per slide at level 7)
  - Observation : Ran out of RAM in Colab

FUTURE WORK

- Experiment with more architectures
  - We used only VGG16 and InceptionV3 – increasing the number of epochs did not necessarily improve the performance
- Better data extraction techniques – leading to more *diverse* and *balanced* training data
- Possibly better metric ?
  - Using accuracy to train the model led to large red patches because model optimized for accuracy

Code Walkthrough