Detecting Cancer Metastases on Gigapixel Pathology Images

COMS4995 APPLIED DEEP LEARNING

Chloe Liu (yl4616)

Agenda

- Project Introduction
  - Motivation
  - Data
  - Method
- Data Preprocessing
- Model Implementation
  - Transfer learning
  - Single-scale
  - Multi-scale
- Evaluation and Discussion

Project Introduction

Problem and data

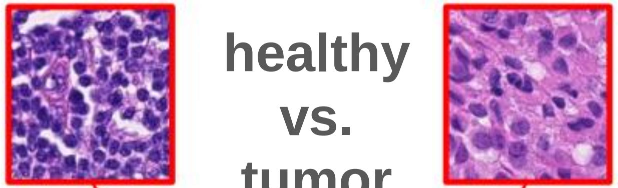
Motivation and goals

- High misdiagnosis rate in breast cancer biopsy interpretation
  - Especially when tumors are very small
  - But metastasis detection is vital
- We want to help:
  - Improve diagnosis accuracy
  - Raise productivity, save time and cost
  - Increase consistency

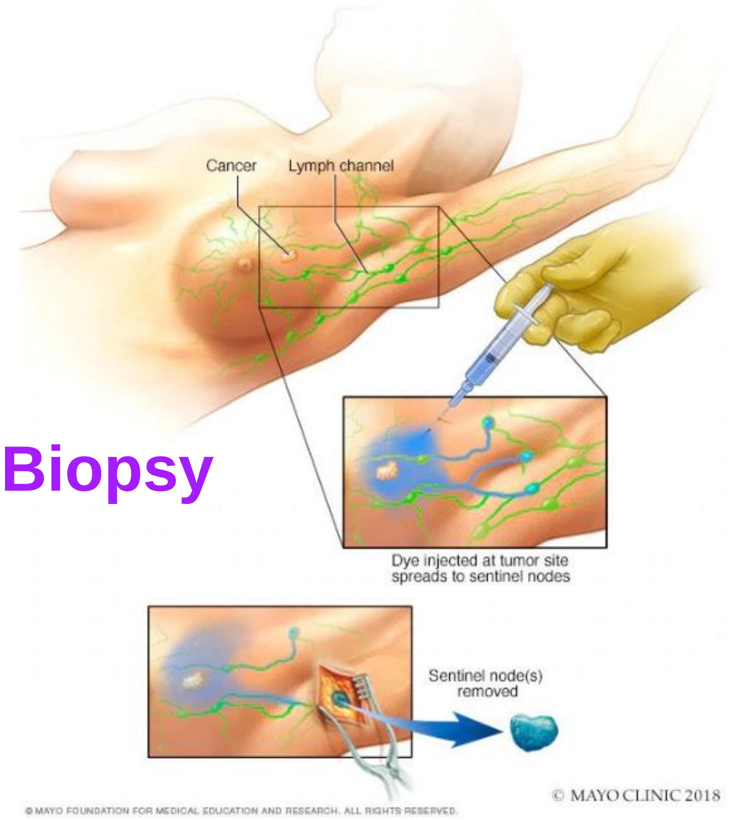
Cycle

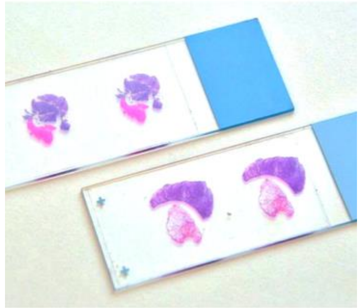
Treatment

Diagnosis

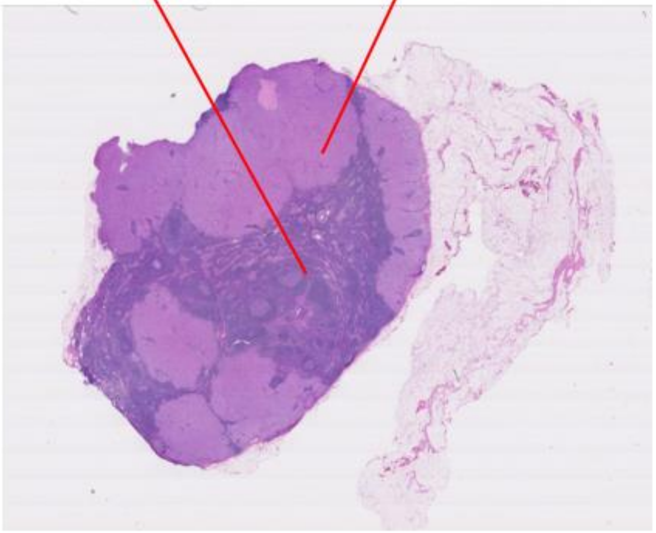


Biopsy





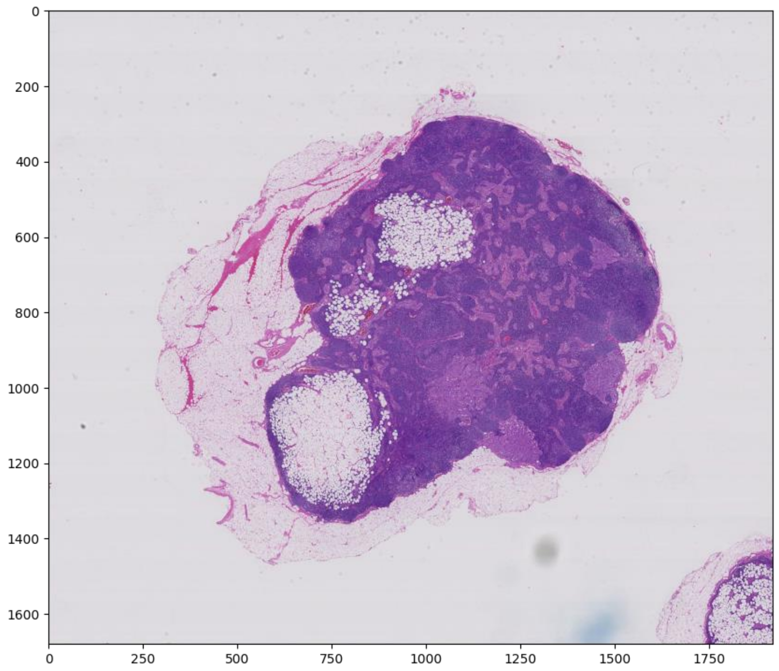
Preparation

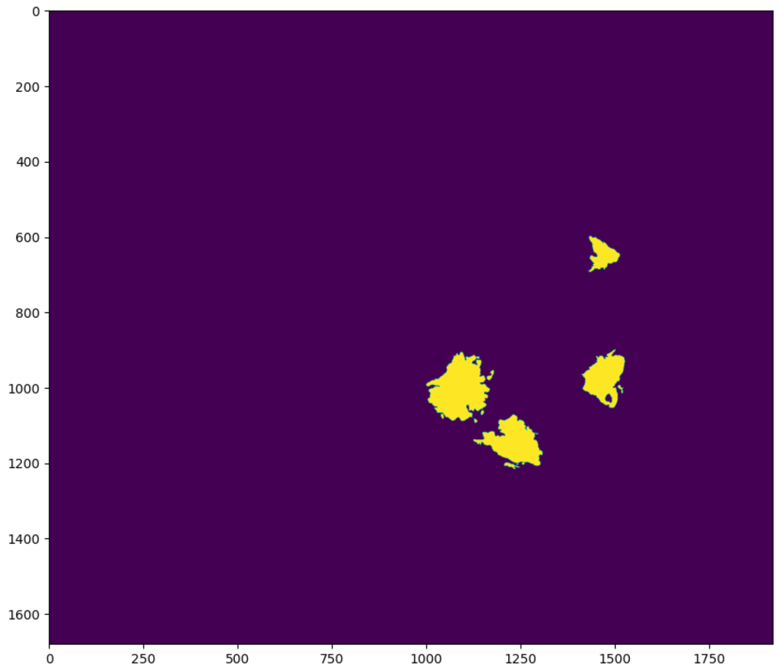


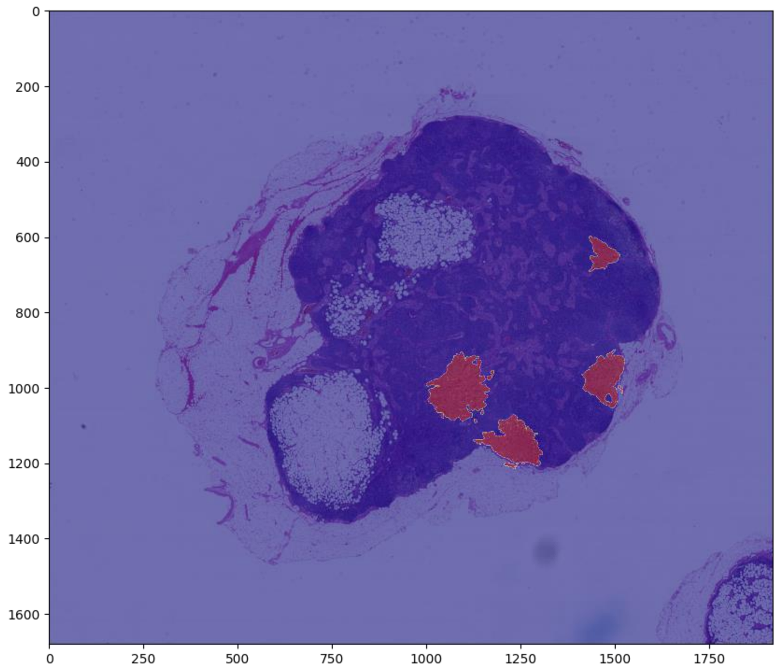
Visual inspection

Dataset – CAMELYON16

Slide image + **Tumor mask** = **Masked image**



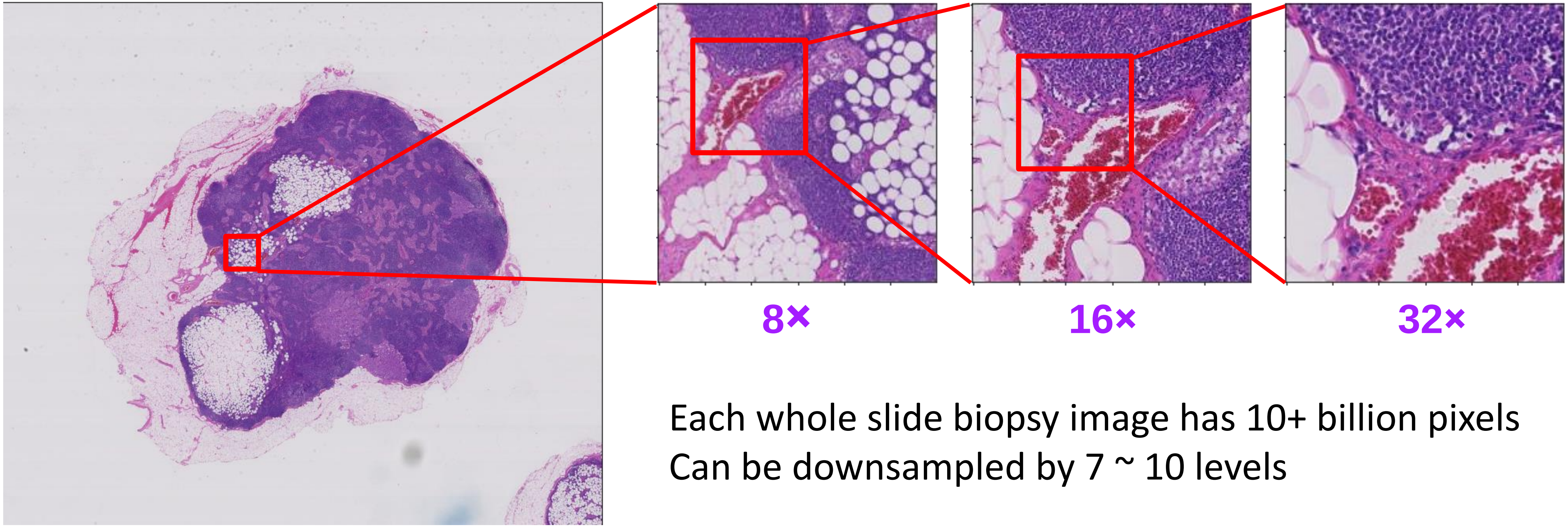




https://camelyon16.grand-challenge.org/

Data used in this project was provided by Prof. Joshua Gordon.

Dataset – CAMELYON16

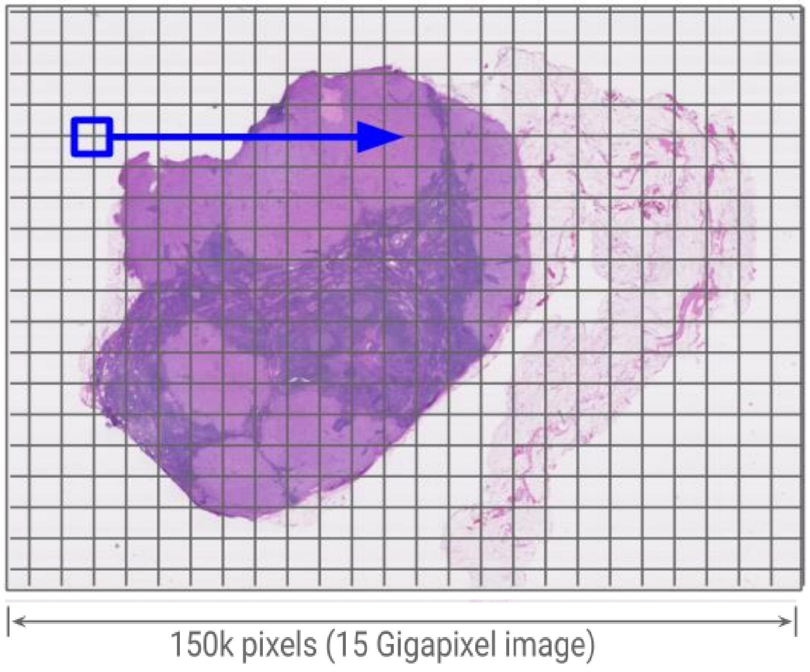


Each whole slide biopsy image has 10+ billion pixels
Can be downsampled by 7 ~ 10 levels

https://camelyon16.grand-challenge.org/
Data used in this project was provided by Prof. Joshua Gordon.

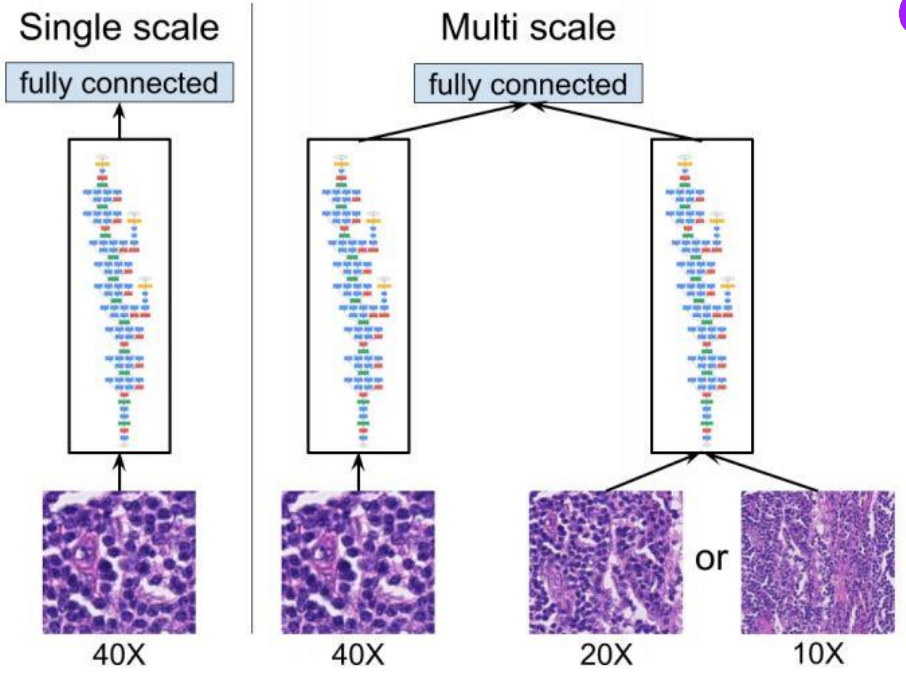
Method (from research papers)

Patch based approach

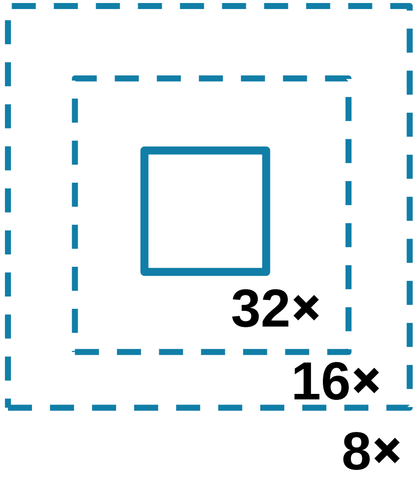


https://arxiv.org/abs/1703.02442

Multi-scale model



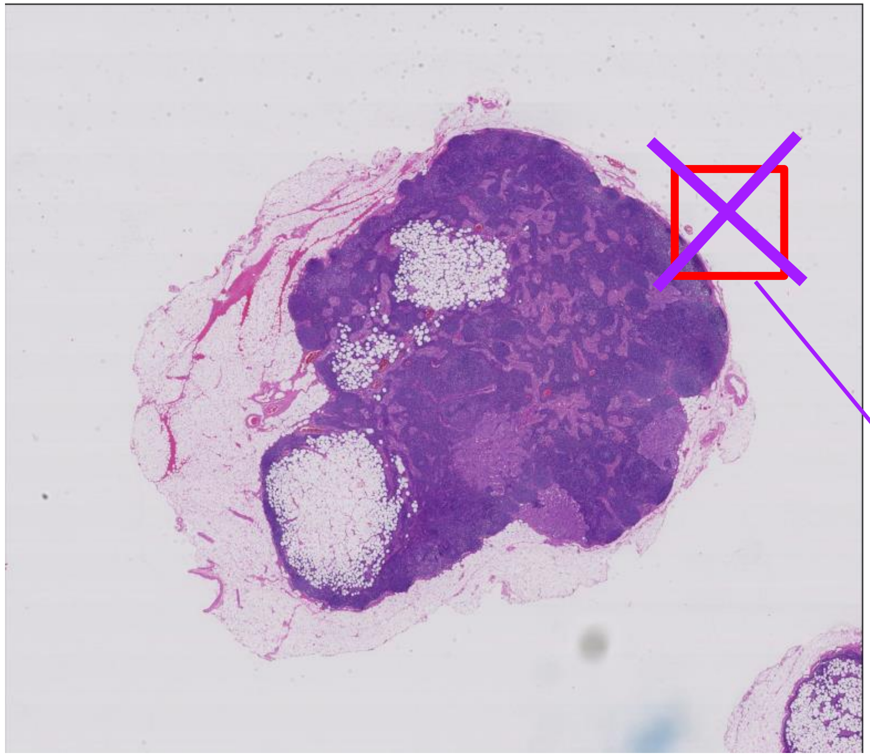
Surrounding context



Data Preprocessing

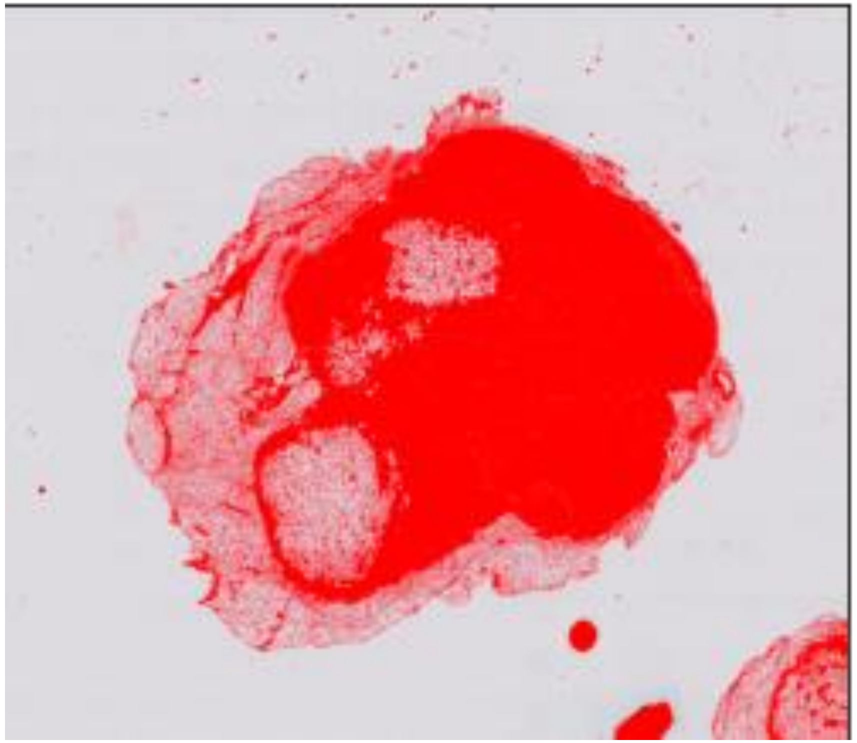
Patch extraction and tissue filtering

From multiple slides, on level 3 and level 4



This patch is not considered as only a few pixels contain tissues

Tissue mask



Resampling

- RandomOverSampler():

  Random oversampling

- SMOTETomek():

  A combination of oversampling and undersampling

https://imbalanced-learn.org/stable/user_guide.html

Model Implementation

Deal with data imbalance

- Data augmentation
  - Flip, rotate, zoom, …
- Resampling
  - Random oversampling
  - SMOTE oversampling + undersampling
- Class weights <-- model.fit()
- Output bias <-- model output Dense(1) layer
- Classification metrics

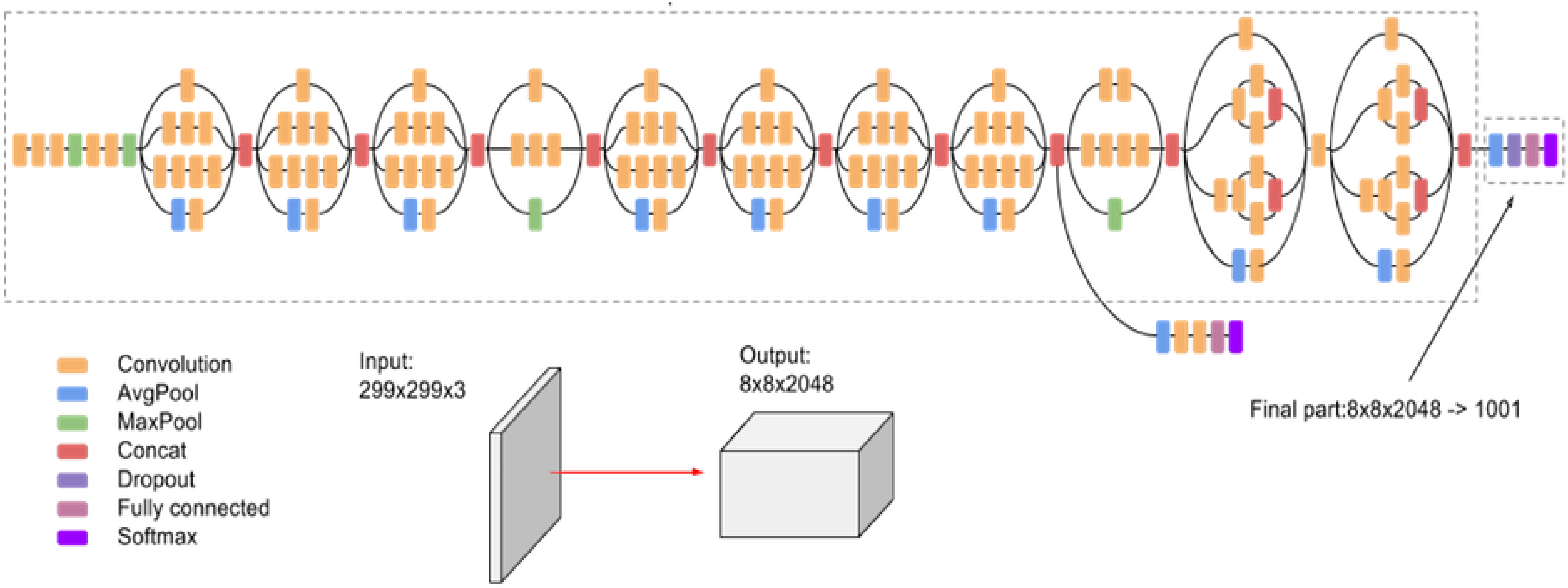
https://www.tensorflow.org/tutorials/structured_data/imbalanced_data

Transfer learning with fine tuning

InceptionV3

(by Christian Szegedy

et al., 2015)

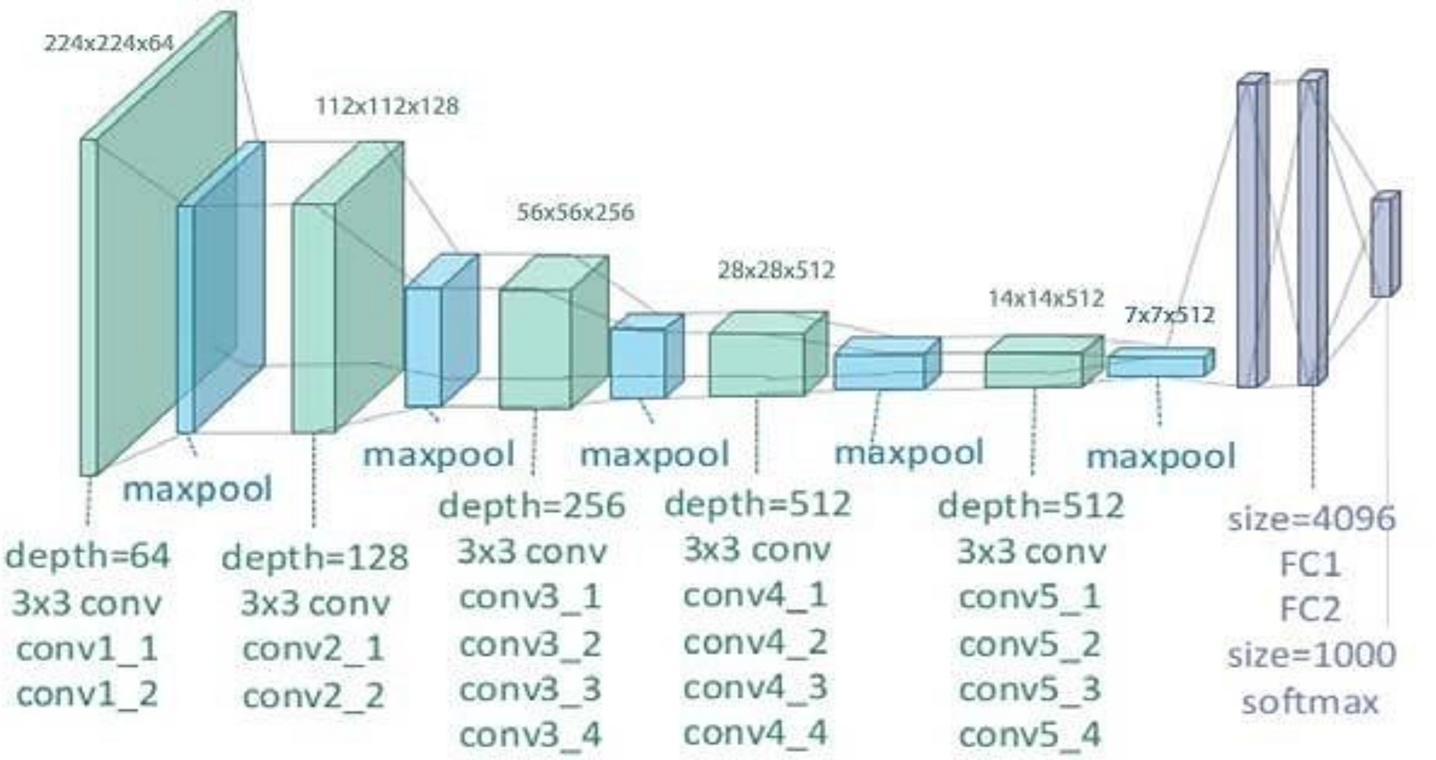


https://arxiv.org/abs/1512.00567

Transfer learning with fine tuning

VGG19

(by Karen Simonyan, et al., 2014)



https://arxiv.org/abs/1409.1556

InceptionV3

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_19 (InputLayer)        [(None, 128, 128, 3)]     0
_________________________________________________________________
tf.math.truediv_7 (TFOpLambd (None, 128, 128, 3)       0
_________________________________________________________________
tf.math.subtract_7 (TFOpLamb (None, 128, 128, 3)       0
_________________________________________________________________
data_augmentation (Sequentia (None, 128, 128, 3)       0
_________________________________________________________________
dropout (Dropout)            (None, 128, 128, 3)       0
_________________________________________________________________
inception_v3 (Functional)    (None, 2, 2, 2048)        21802784
_________________________________________________________________
global_average_pooling2d (Gl multiple                  0
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 2049
=================================================================
Total params: 21,804,833
Trainable params: 21,770,401
Non-trainable params: 34,432
```

VGG19

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_15 (InputLayer)        [(None, 128, 128, 3)]     0
_________________________________________________________________
tf.__operators__.getitem_5 ( (None, 128, 128, 3)       0
_________________________________________________________________
tf.nn.bias_add_5 (TFOpLambda (None, 128, 128, 3)       0
_________________________________________________________________
data_augmentation (Sequentia (None, 128, 128, 3)       0
_________________________________________________________________
dropout (Dropout)            (None, 128, 128, 3)       0
_________________________________________________________________
vgg19 (Functional)           (None, None, None, 512)   20024384
_________________________________________________________________
global_average_pooling2d (Gl multiple                  0
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 513
=================================================================
Total params: 20,024,897
Trainable params: 20,024,897
Non-trainable params: 0
```

Preprocess_input specified by each architecture

Trials in parameter search

Datasets

- Original unbalanced
- Oversampled
- Over + undersampled

Models

- InceptionV3
- VGG19

Layers

- Data augmentation
- MaxPool

…

Optimizers

- SGD
- Adam
- RMSprop

Learning rates

- 1e-5
- 1e-4

…

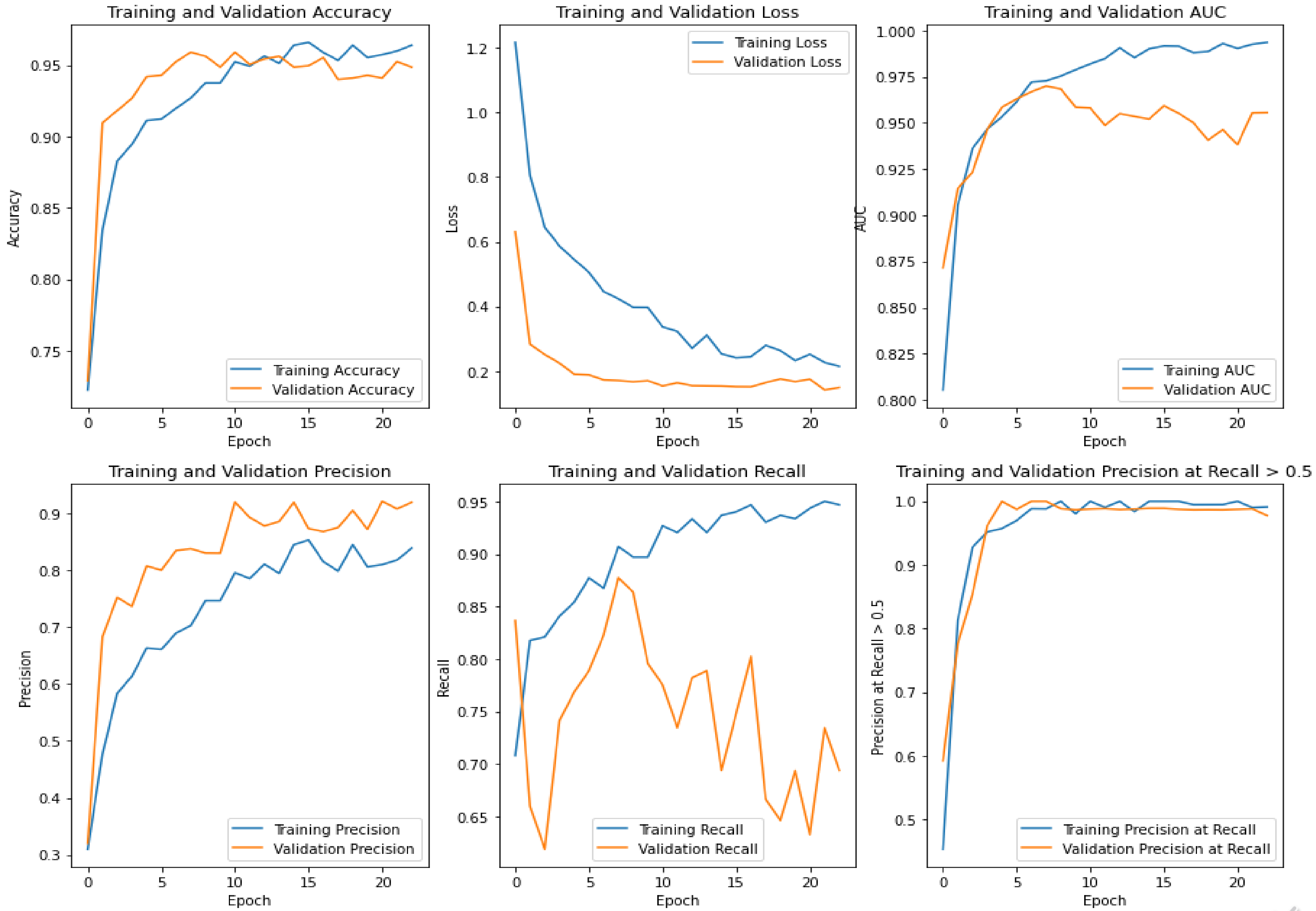
… …

- Output bias
- Class weights

…

Metrics

- Accuracy
- Precision
- Recall
- AUC



Evaluation and Discussion

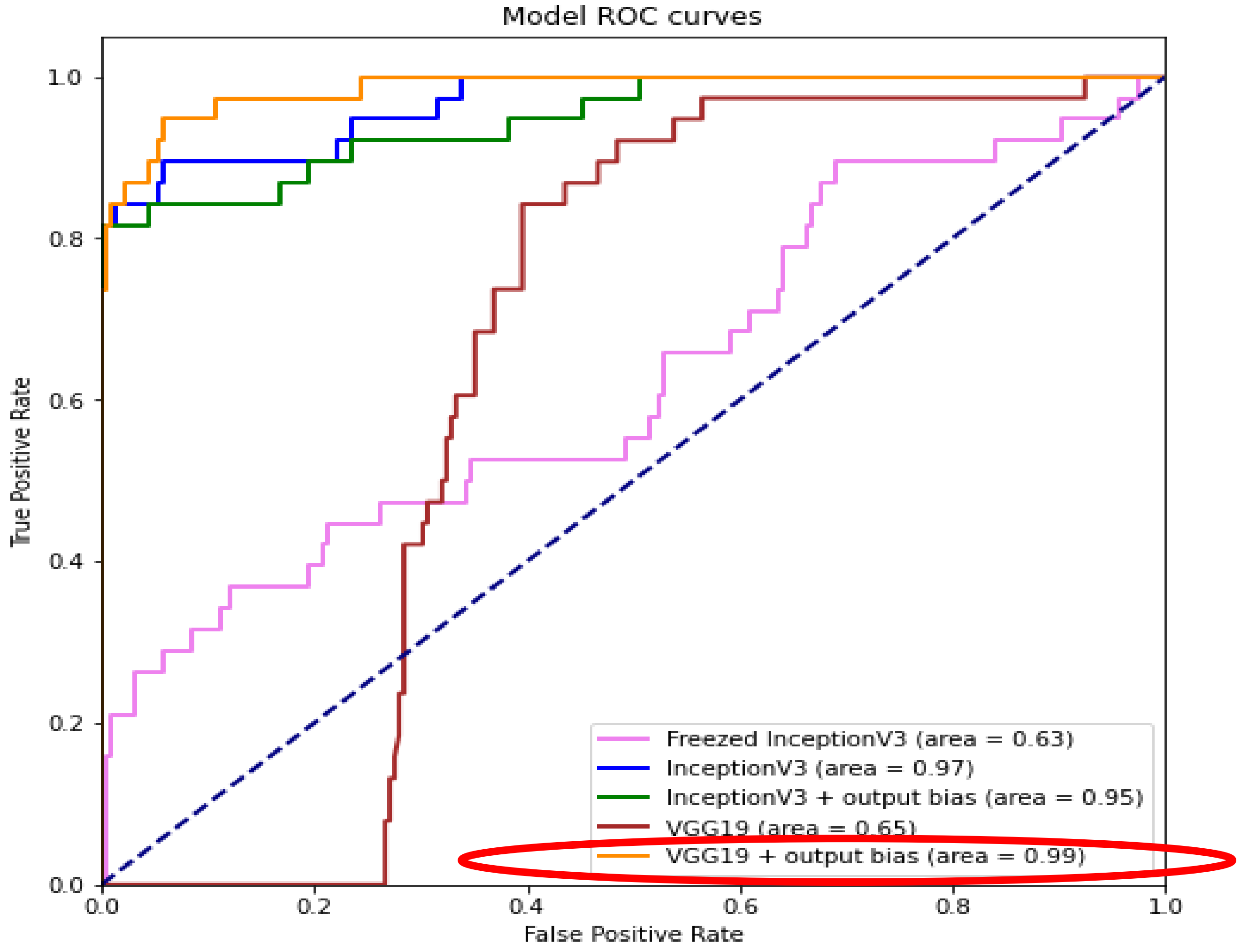
Evaluation

```
Train set ['031','064','091']
```

- Best model for level 3, 4:

Retrained VGG19
+ output bias
+ class weights

- Robust to data imbalance
- AUC = 0.99 on test dataset



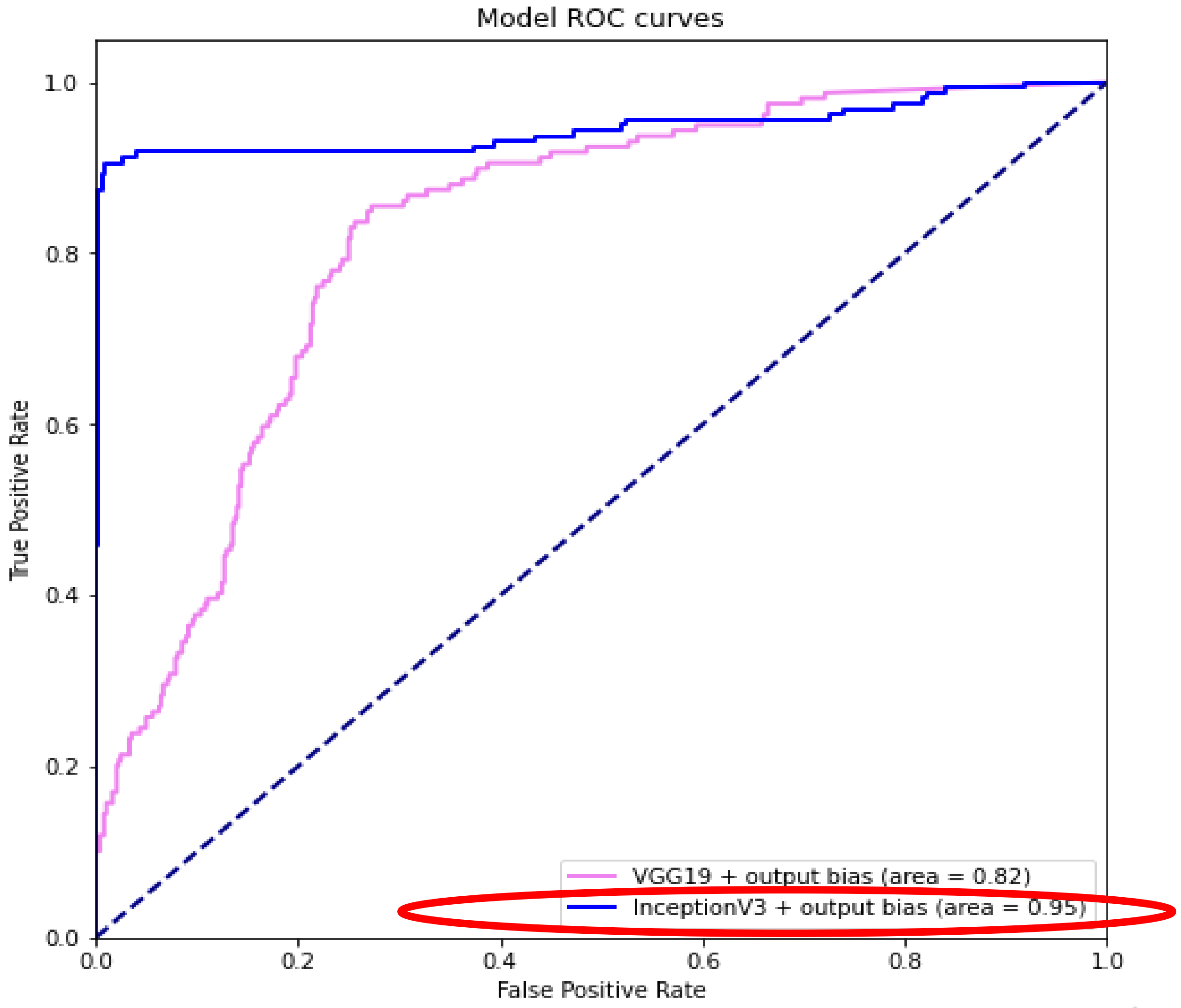
Evaluation

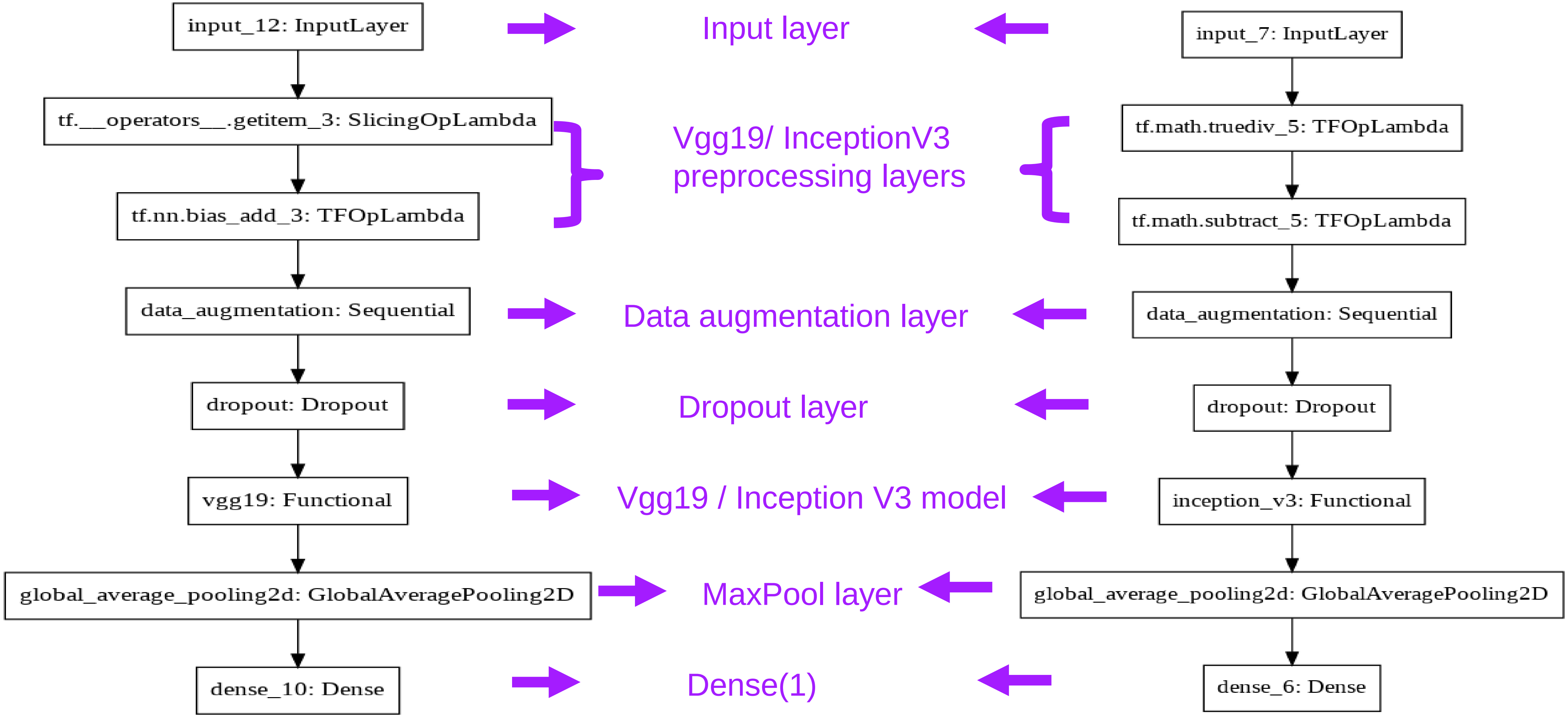
```
Train set ['031','064','091','016','078']
```

- Best model for level 3:

Retrained InceptionV3
+ output bias
+ class weights

- Robust to data imbalance
- AUC = 0.95 on test dataset

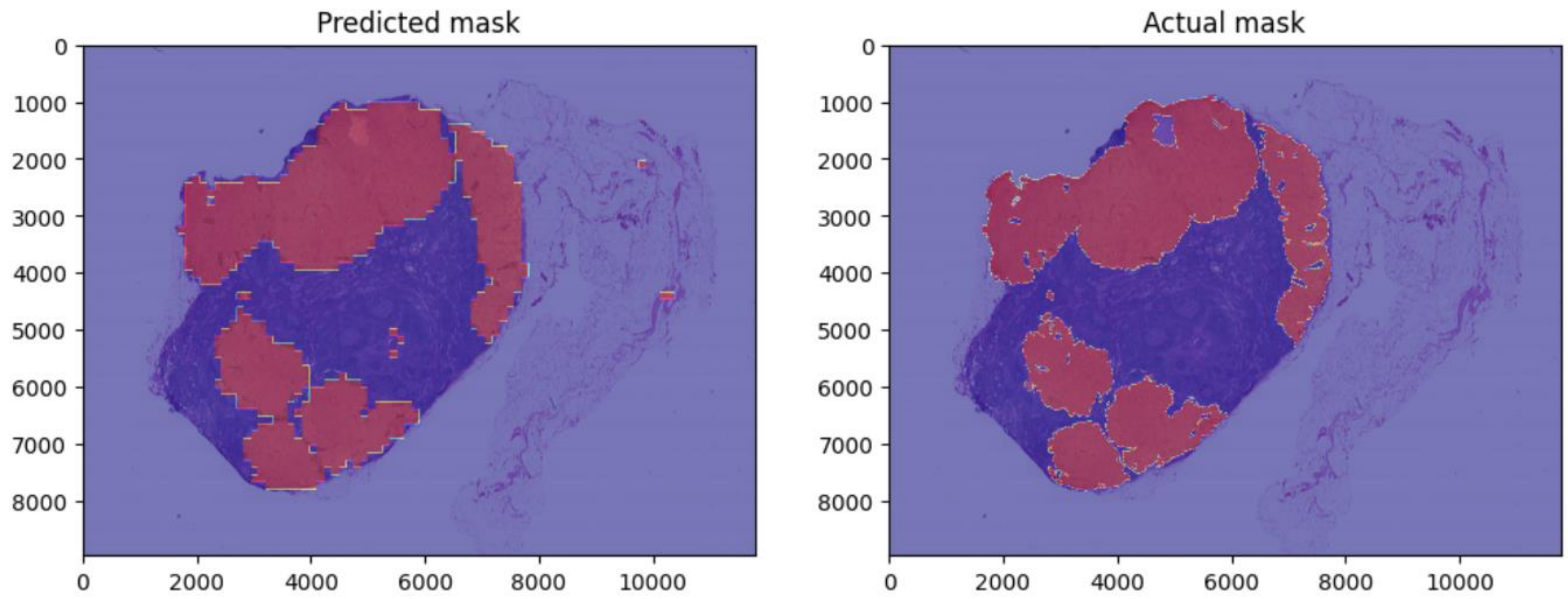



input_12: InputLayer
tf.__operators__.getitem_3: SlicingOpLambda
tf.nn.bias_add_3: TFOpLambda
data_augmentation: Sequential
dropout: Dropout
vgg19: Functional
global_average_pooling2d: GlobalAveragePooling2D
dense_10: Dense
Input layer
Vgg19/ InceptionV3
preprocessing layers
Data augmentation layer
Dropout layer
Vgg19 / Inception V3 model
MaxPool layer
Dense(1)
input_7: InputLayer
tf.math.truediv_5: TFOpLambda
tf.math.subtract_5: TFOpLambda
data_augmentation: Sequential
dropout: Dropout
inception_v3: Functional
global_average_pooling2d: GlobalAveragePooling2D
dense_6: Dense

```
Test set ['110','101','096']
```

Results (Predicted by VGG-l3 model)

Test image 110

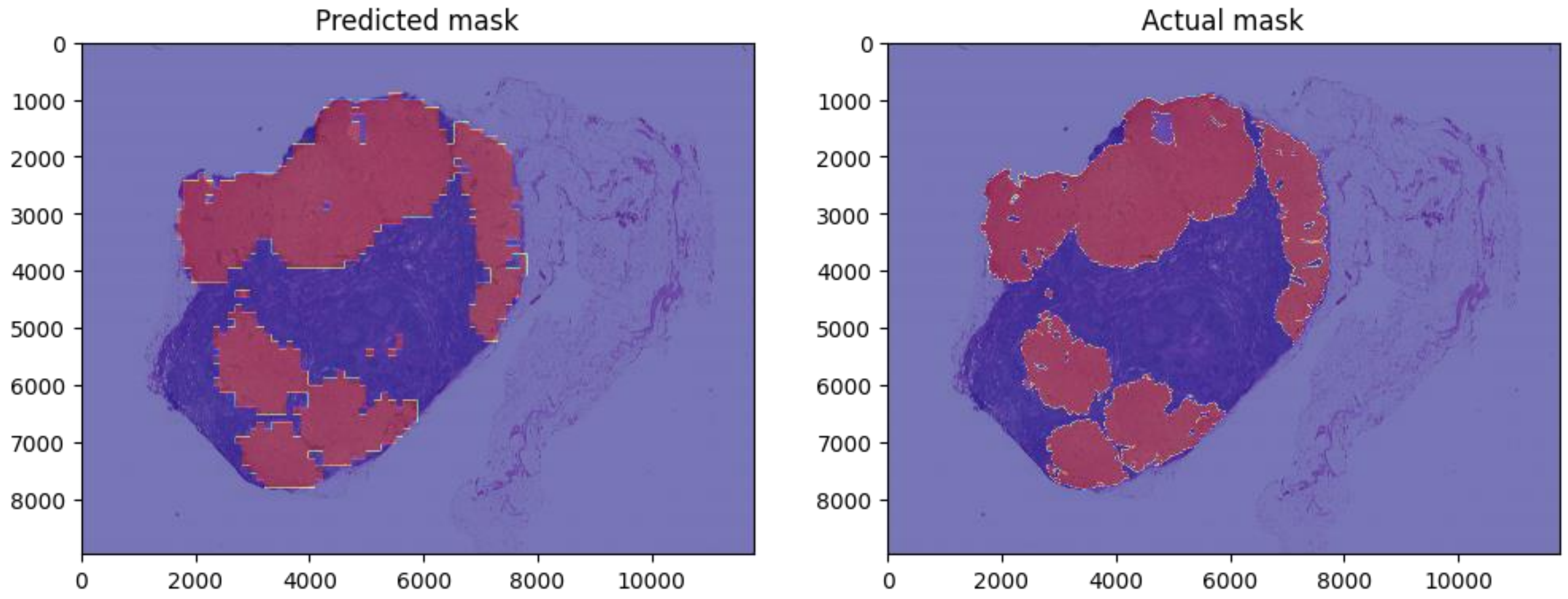



```
Classification report:
Accuracy: 0.9425
Precision: 0.9927
Recall: 0.9124
F1-score: 0.9508
```

Test set [‘110’,’101','096']

Results (Predicted by InceptionV3-l3 model)

Test image 110



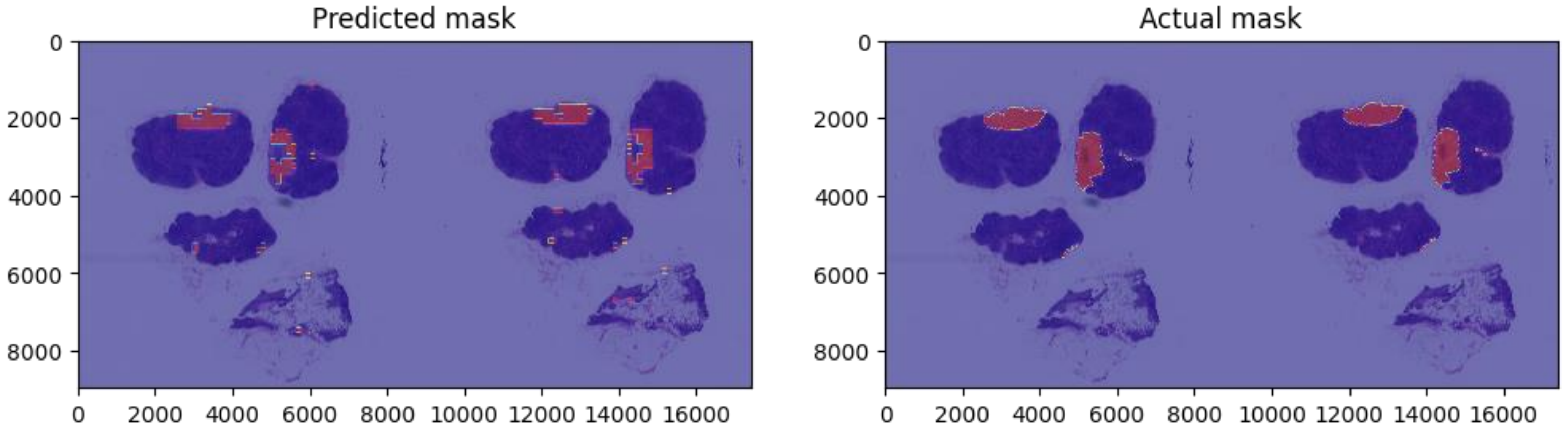
Classification report:
Accuracy: 0.9312
Precision: 0.9944
Recall: 0.8922
F1-score: 0.9405

Test set ['110','101','096']

Results (Predicted by VGG-l3 model)

Test image 101

Classification report:
Accuracy: 0.9244
Precision: 0.9145
Recall: 0.5073
F1-score: 0.6526

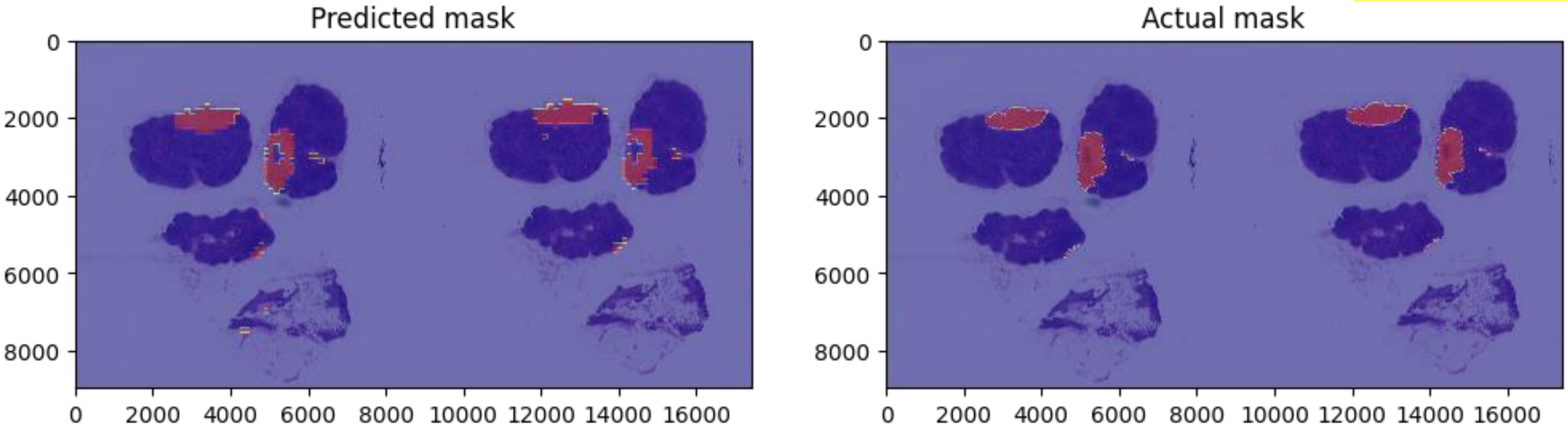


Test set [‘110’,’101',’096']

Results (Predicted by InceptionV3-l3 model)

Test image 101

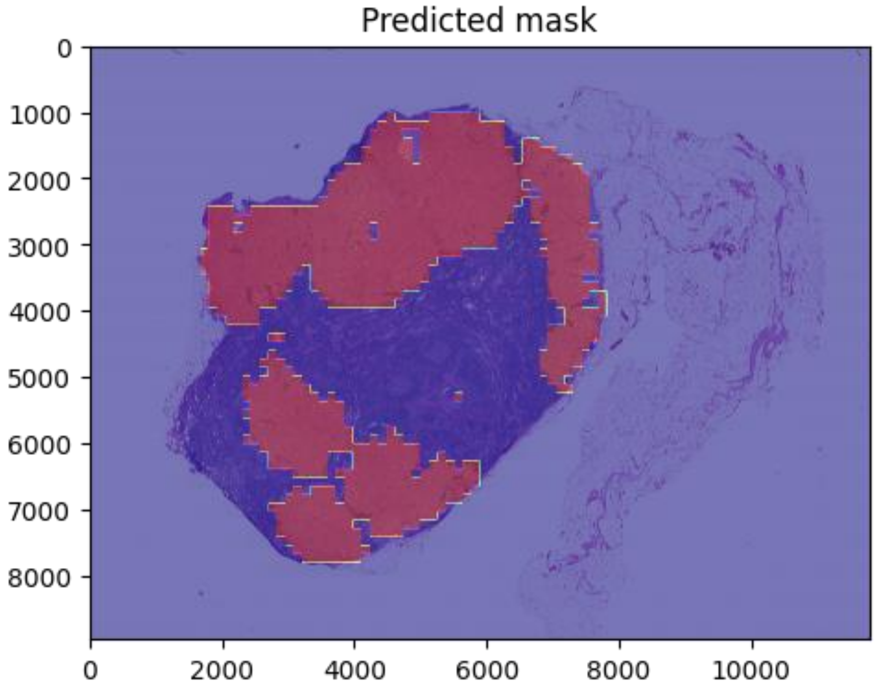
```
Classification report:
Accuracy: 0.9663
Precision: 0.9727
Recall: 0.781
F1-score: 0.8664
```

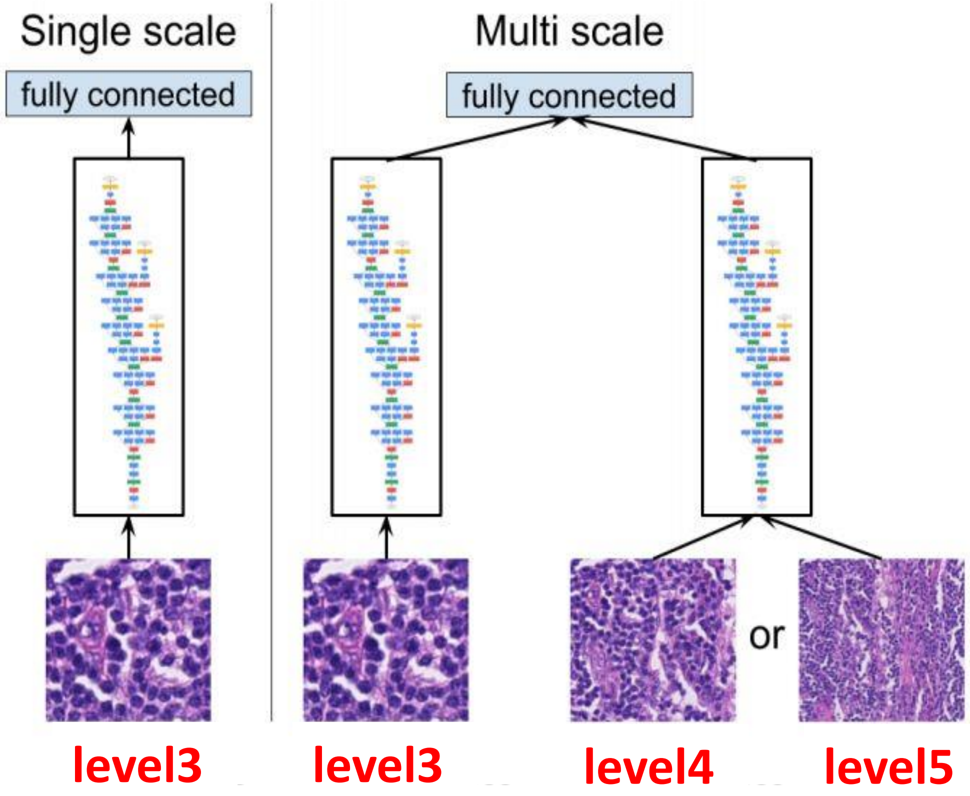


Test set ['110','101','096']

Results (Predicted by Multi-scale model of level 3, 4)

Test image 110





Classification report:
Accuracy: 0.9159
Precision: 0.9971
Recall: 0.8644
F1-score: 0.926

Conclusion

- The single-scale models on level 3 generated from both VGG19 and InceptionV3 architectures can generalize well
- The multi-scale model on level 3 & 4 offers more reliable diagnosis, which maybe a better option in practice
- Tensorflow provides a variety of excellent options to deal with data imbalance
  - Class weights,
  - Output bias,
  - Data augmentation, etc….
- These imbalance treatments perform better then resampling in this case

Future work

- There might be more accurate algorithms if decreasing the slide window size, e.g. 50 * 50 patch
- More explorations can be made on Multi-scale prediction models
- Simpler neural networks might also perform well, i.e. lighter models with fewer layers and weights
- It worths exploring more directed/targeted algorithms for medical imaging

Thanks!