Detecting Cancer Metastases on Gigapixel Pathology Images

COMS W4995 010 Applied Deep Learning

Instructor: Joshua Gordon

Team Members: Weiwei Zhan, Haokai Zhao

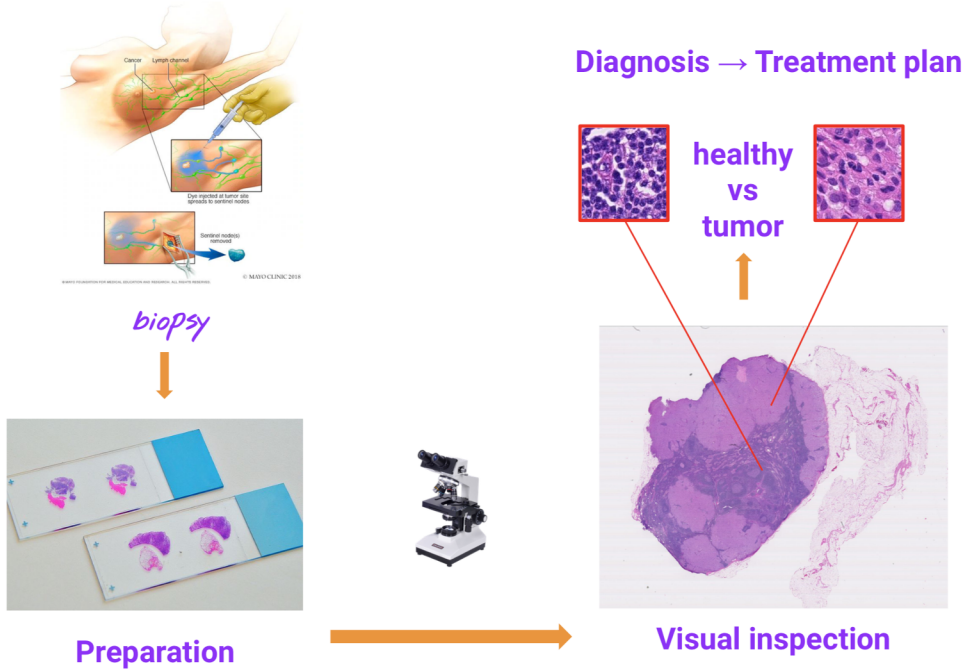
Background

Microscopic examination of breast cancer

- Time-consuming and error-prone
- Computer assisted detection

Deep CNNs have significantly improved the accuracy of computer vision tasks

Goal: to develop a CNN framework to assist breast cancer detection



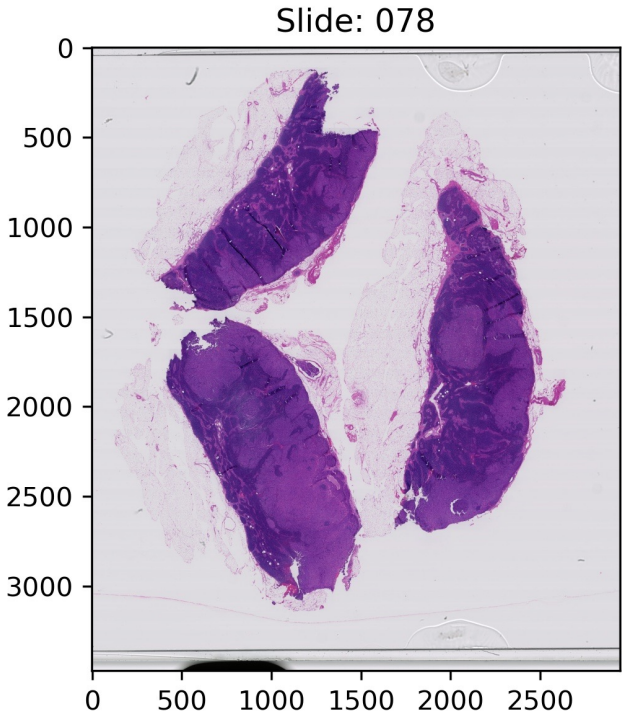
❖ From Course Slides

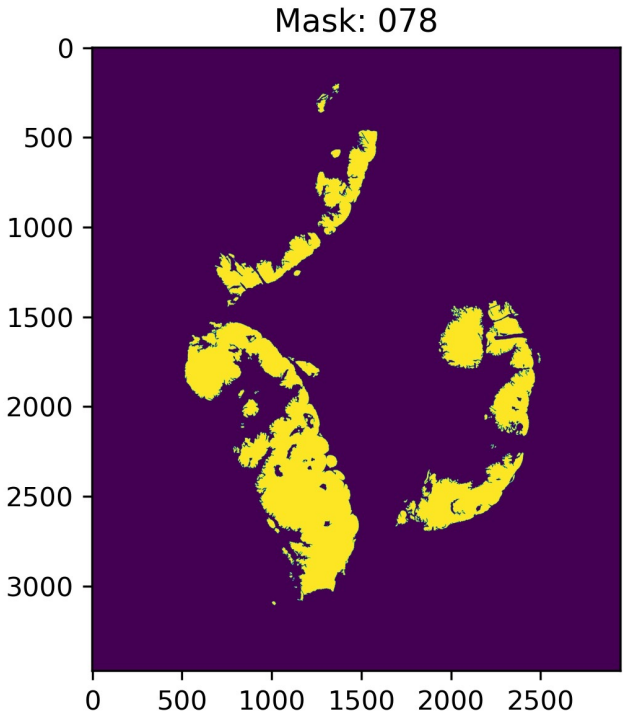
I. Preprocessing

1.1 Load the dataset

Original Dataset: Gigapixel Pathology Images (CAMELYON16 Dataset)

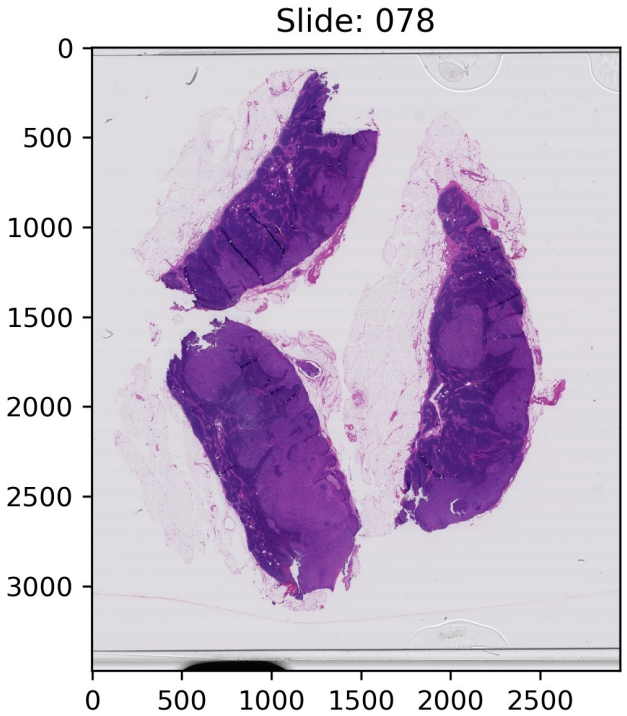
- Number of slides: 22
- Number of masks: 21 (there's no mask for slide 038)

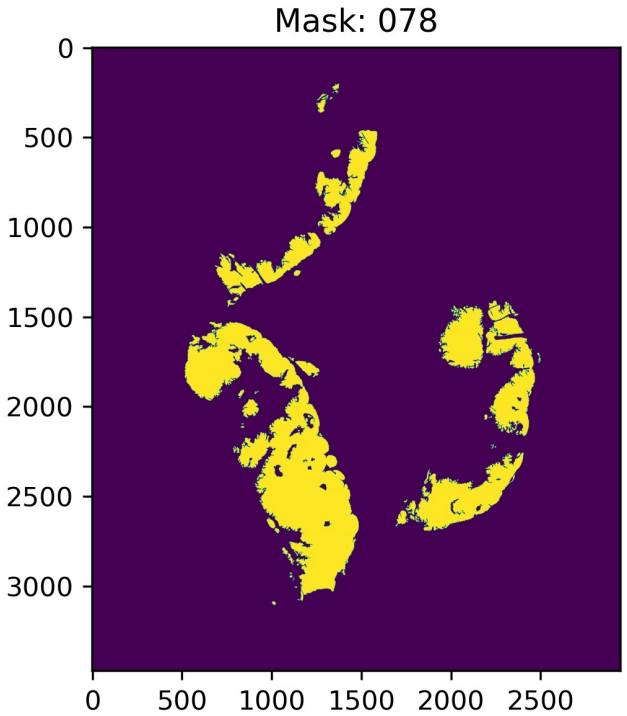




1.2 Split the Dataset & Sanity Check

- Tried to use all available slides at first, the result was not quite good, and the training process took a long time.
- Manually select slides for training, validation, and test sets
- Check whether all slides contain cancer cells

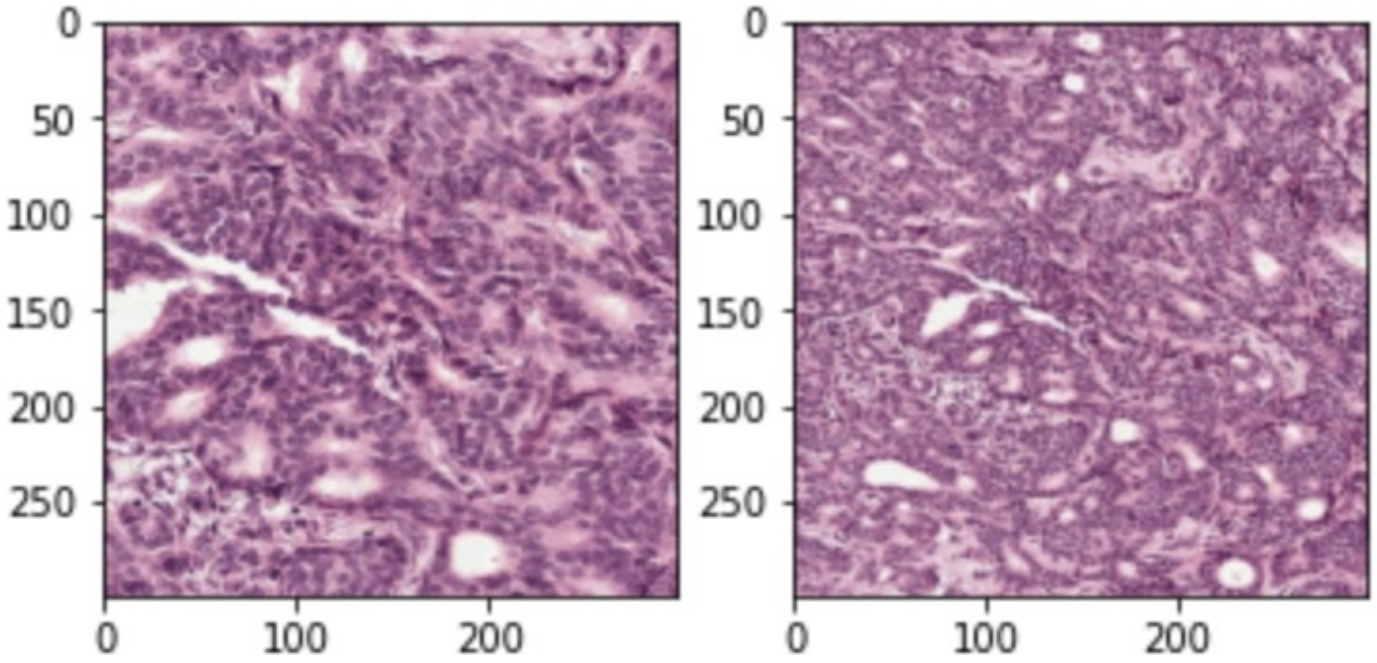




1.3 Generate Patches

Strategies Used:

- Fixed patch size (299 x 299) at two different zoom levels
- Aligned at the same center point
- Tissue percentage ≥ 30%
- Use the label of the lower zoom level (higher resolution)
- If there is any tumor pixel existing in the patch, then the label is 1 (positive)



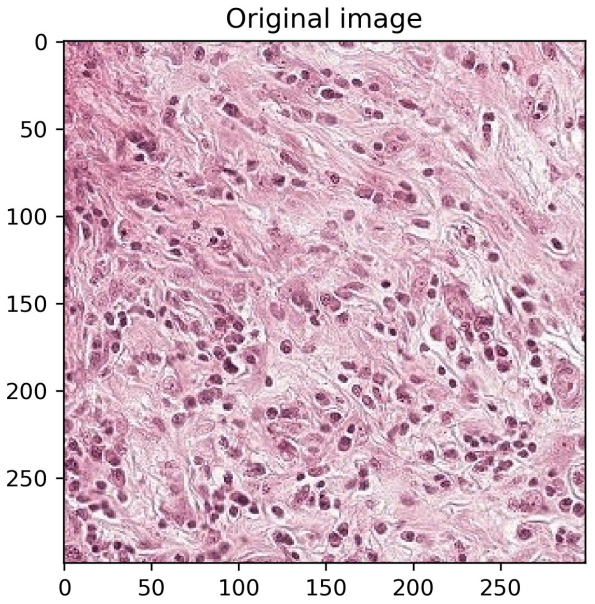
Zoom Level 2 Zoom Level 3

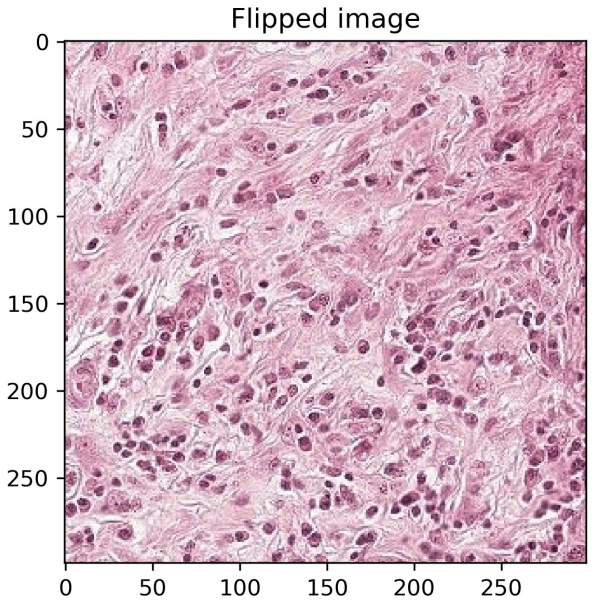
1.4 Deal with Imbalanced Data

❖ Problem: More negative samples than positive samples.

Strategies Tried:

- Data Augmentation
- Reduce the number of non-tumor samples, which has much more samples than tumor samples
- Adjust class weights at the training process
- Define custom loss functions to penalize undesired behaviors



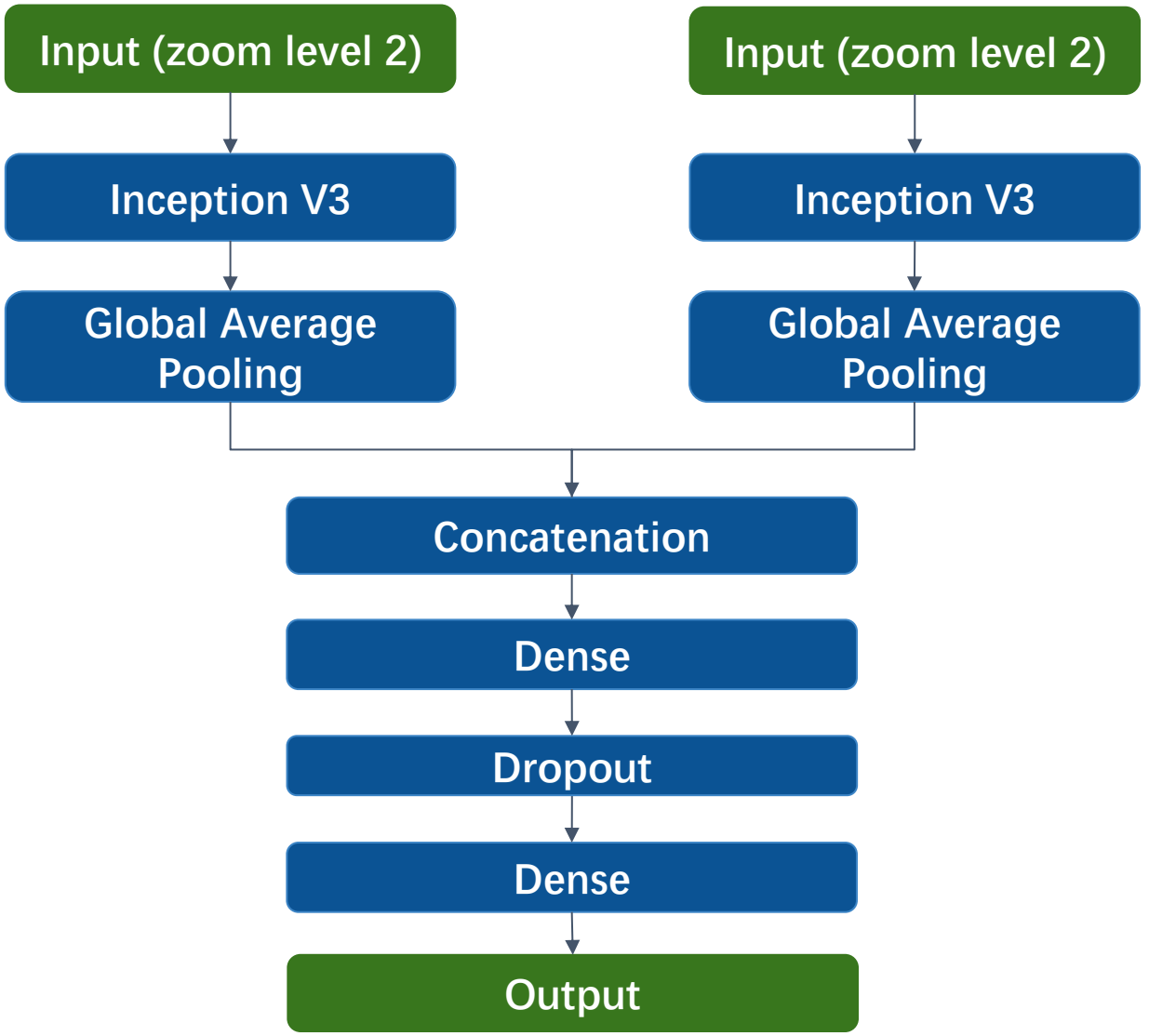


II. Model Building

2.1 Model Building

➢ **Features:**

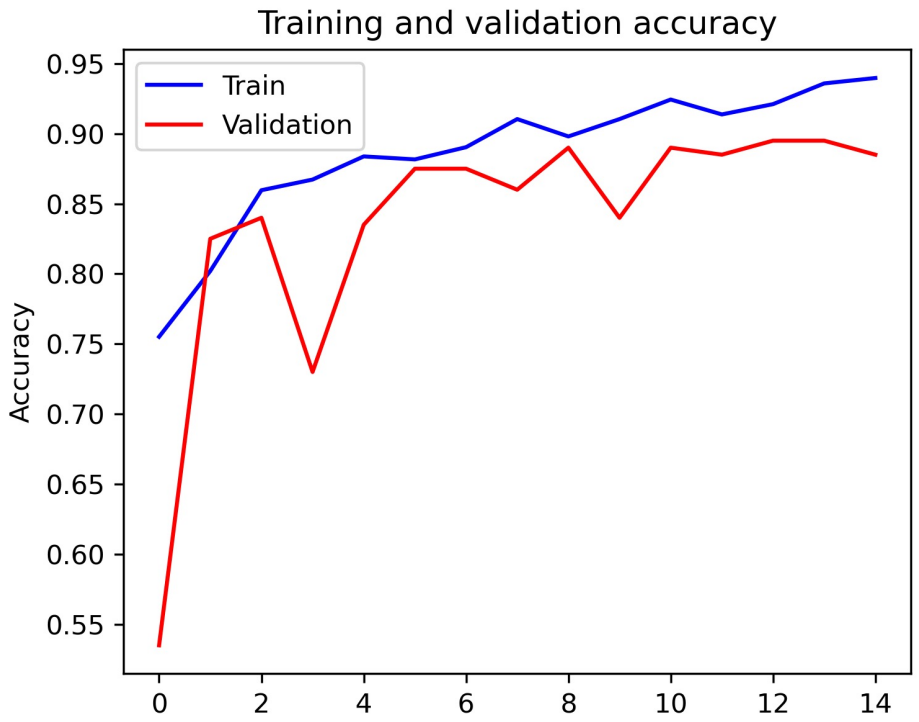
- Multi-input model
- Pretrained Inception V3 model on ImageNet (frozen)
- Global Average Pooling layers to reduce parameters
- Concatenation layer to leverage multi-scale inputs
- Dropout layer to mitigate overfitting



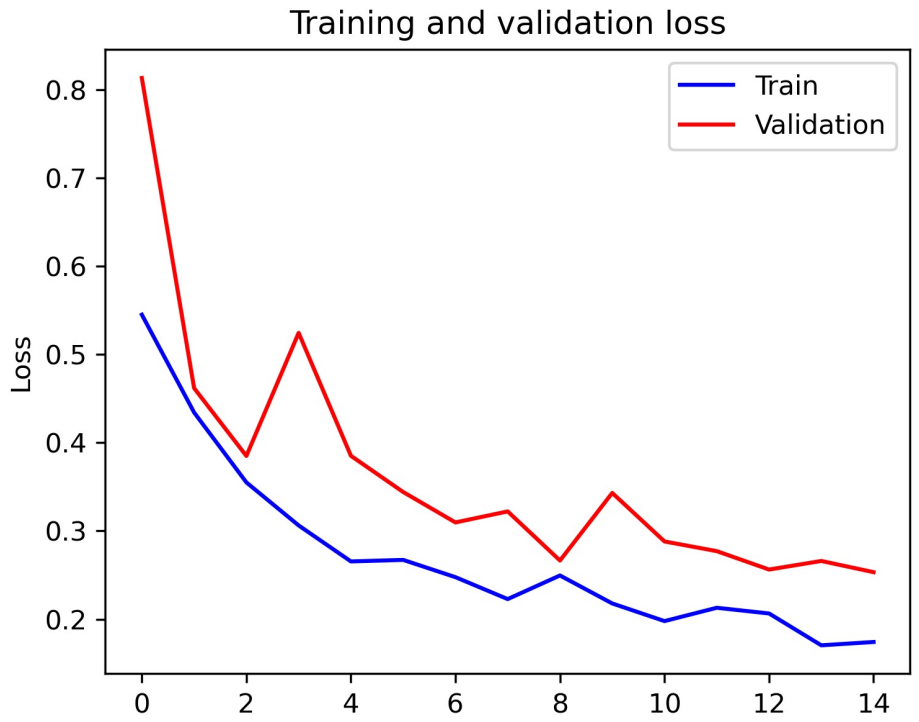
III. Model Training and Evaluation

3.1 Loss and Accuracy

- Final training accuracy: 0.9448
- Final validation accuracy: 0.8850



- Final training loss: 0.1556
- Final validation loss: 0.2533



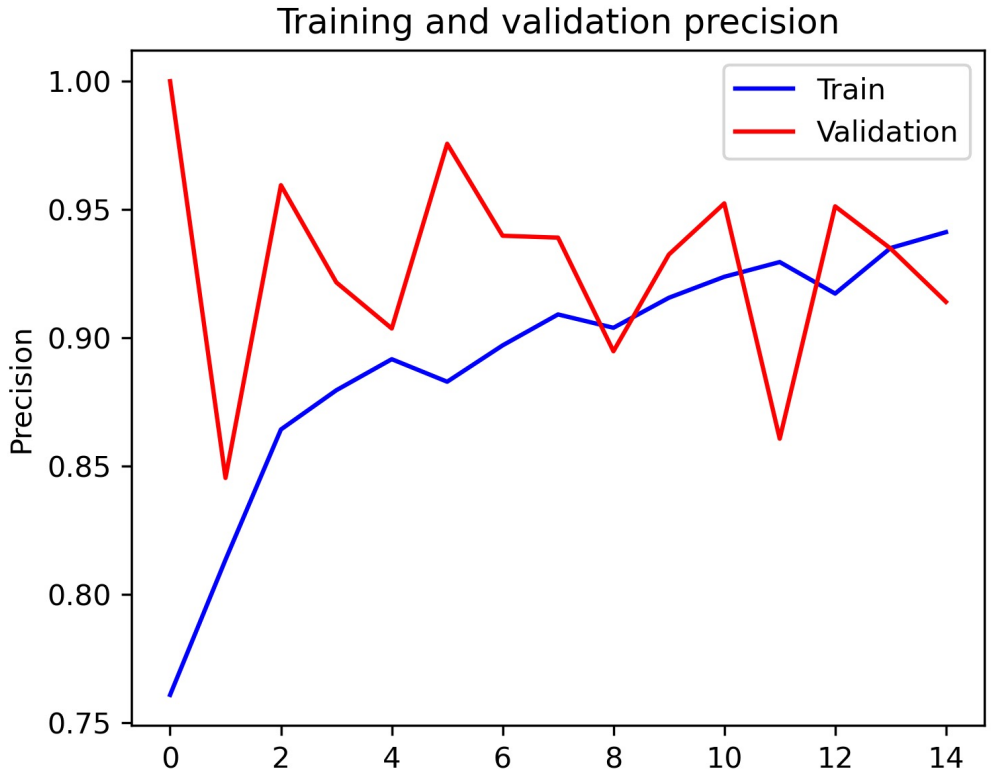
❖ More info at: https://www.tensorflow.org/tutorials/structured_data/imbalanced_data

3.2 Precision

➢ **Precision**:
The percentage of predicted positives that were correctly classified

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

- Final training precision: 0.9551
- Final validation precision: 0.9140



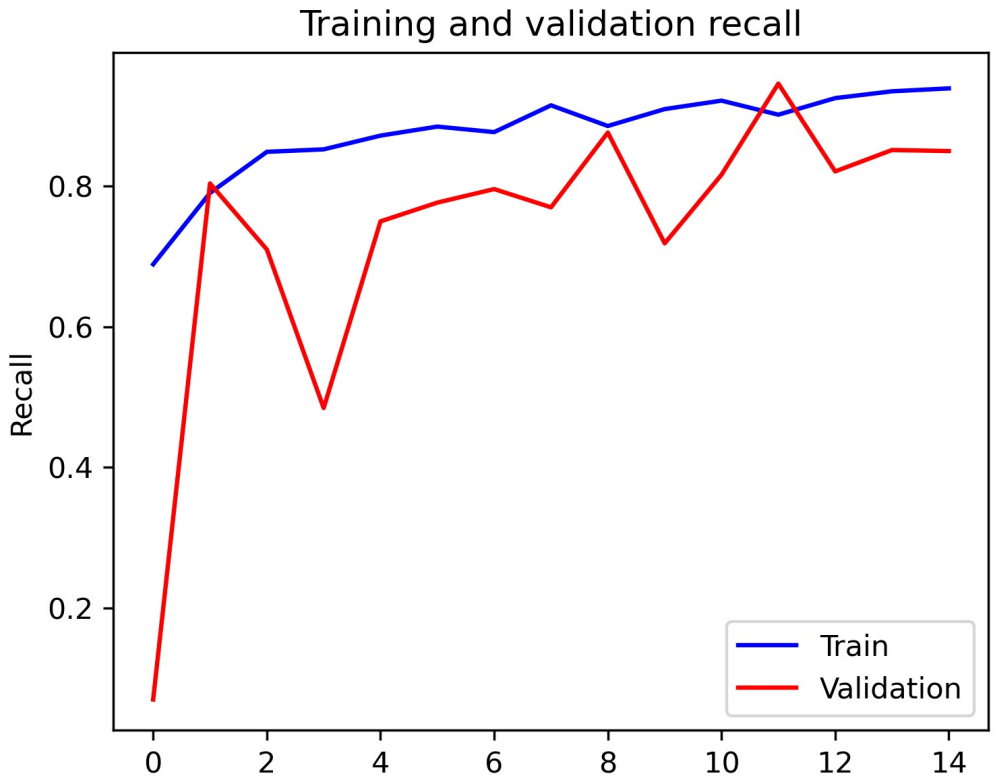
❖ More info at: https://www.tensorflow.org/tutorials/structured_data/imbalanced_data

3.3 Recall

➢ **Recall**:
The percentage of actual positives that were correctly classified

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

- Final training recall: 0.9318
- Final validation recall: 0.8500



❖ More info at: https://www.tensorflow.org/tutorials/structured_data/imbalanced_data

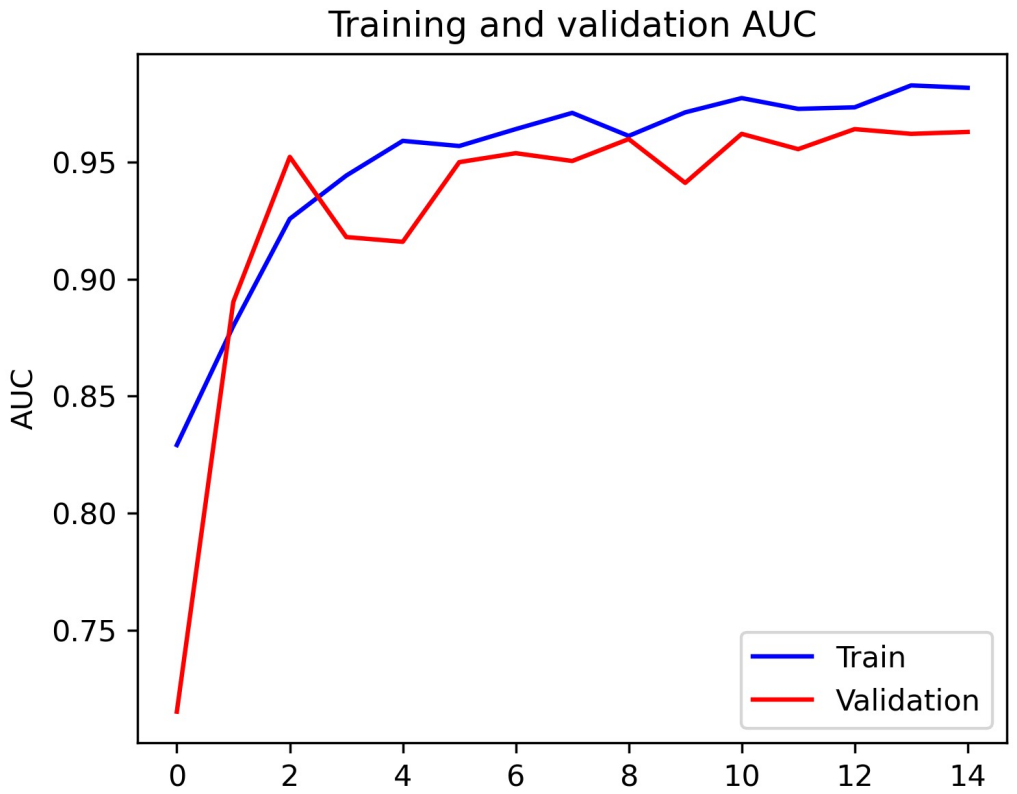
3.4 AUC (Area Under Curve)

➢ **AUC**:

AUC refers to the Area Under the Curve of a Receiver Operating Characteristic curve (ROC-AUC).

AUC ranges in value from 0 to 1.

- Final training AUC: 0.9862
- Final validation AUC: 0.9628



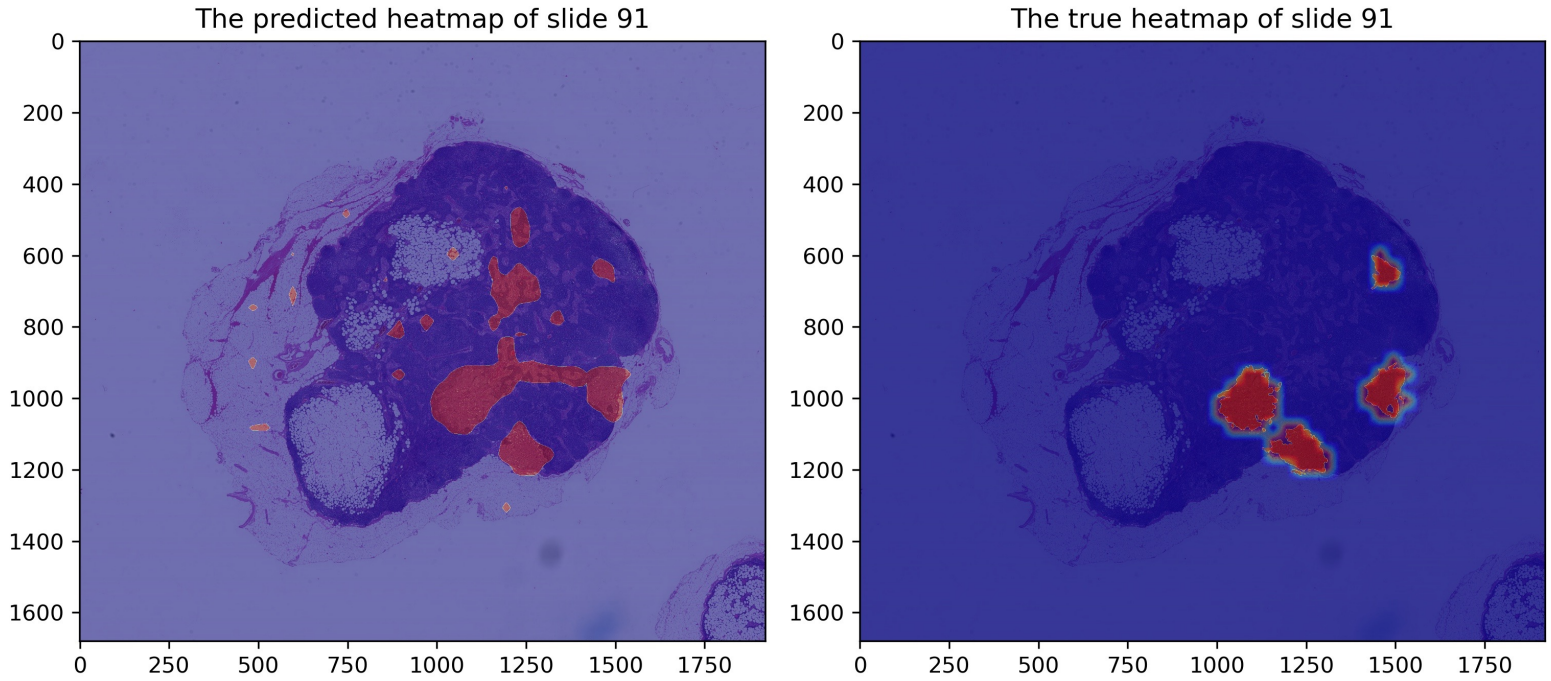
❖ More info at: https://www.tensorflow.org/tutorials/structured_data/imbalanced_data

IV. Model Predictions

4.1 Prediction Heatmaps

Evaluation Metrics:

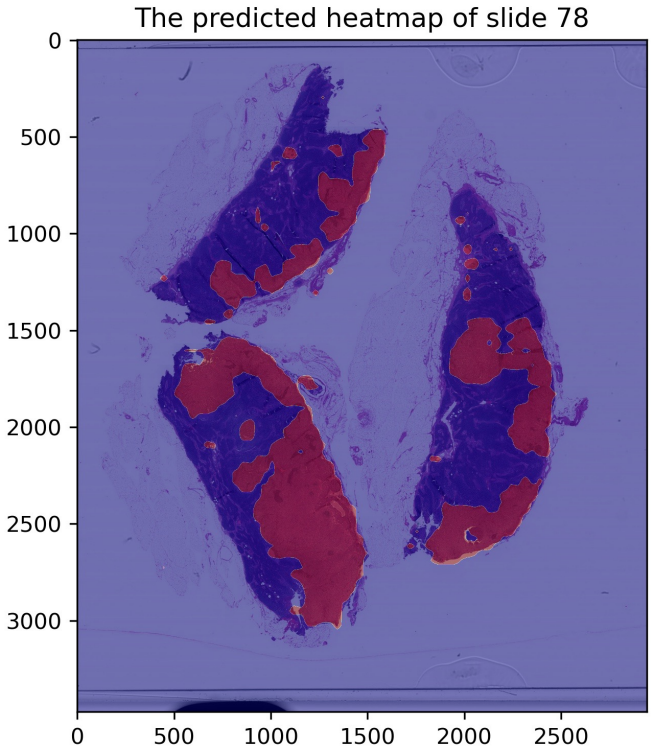
- AUC: 0.9508
- Precision: 0.4368
- Recall: 0.7451

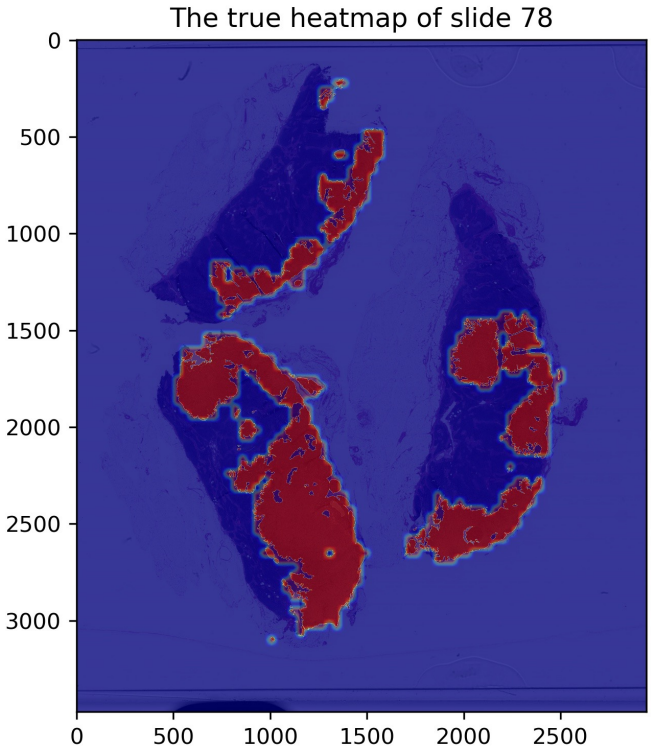


4.1 Prediction Heatmaps

Evaluation Metrics:

- AUC: 0.9811
- Precision: 0.9424
- Recall: 0.8064





Conclusions

Conclusions

In general, the model performs well, to improve it:

- Use lower level (higher resolution) slides
- Generate more patches for training (this may require more hardware resources)
- Configure the dataset for performance (cache, prefetch)
- Explore other model structures

Thank you!

Weiwei Zhan, Haokai Zhao