Proposed Solutions for Machine Learning in Digital Pathology

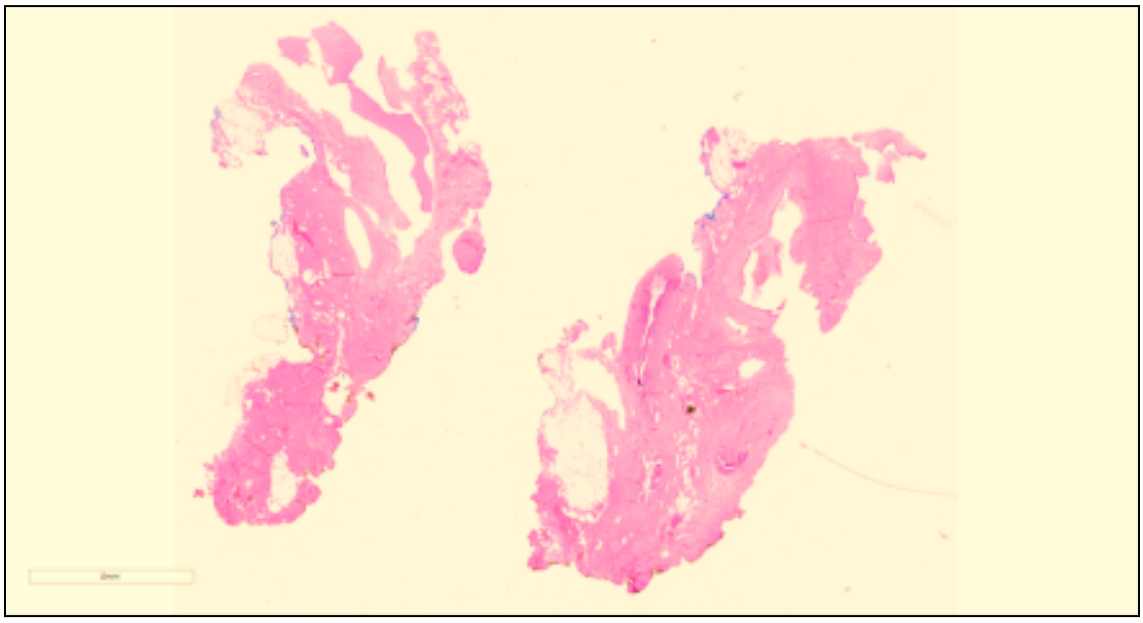
Leo Berman, Albert Bulik, Yuan Nghiem

In order to save the most breast cancer patients, the time to treatment has to be minimized. The best way we can contribute is by reducing the time to treatment by reducing the burden on individual pathologists. Pathologists examine biopsies and tomographic data to identify visual and molecular indicators of malignancy in a patient. The schooling and training required to examine biopsies is significant (14 years), and the pool of trained pathologists is expected to diminish.[1][2] Our goal is to train a machine learning model on a large corpus of breast cancer biopsy data to reduce the burden on individual pathologists, and as a result the overall hospital.

To achieve our goal we have several requirements. Firstly, we require training data. This is fulfilled by annotated, digitized, breast cancer biopsy slides in the Temple University Health Digital Pathology Corpus. Secondly, we require a powerful computer to train our model. This is fulfilled by the high-performance Neuronix computing cluster maintained by Temple's Neural Engineering Data Consortium. Lastly, we require a flexible test bench to evaluate a variety of pre-processors, post-processors and training algorithms. This has been partially fulfilled, and is a work in progress. Ultimately, the chosen combination of algorithms will be the one resulting in the lowest error rate relative to human-annotated biopsy slides.

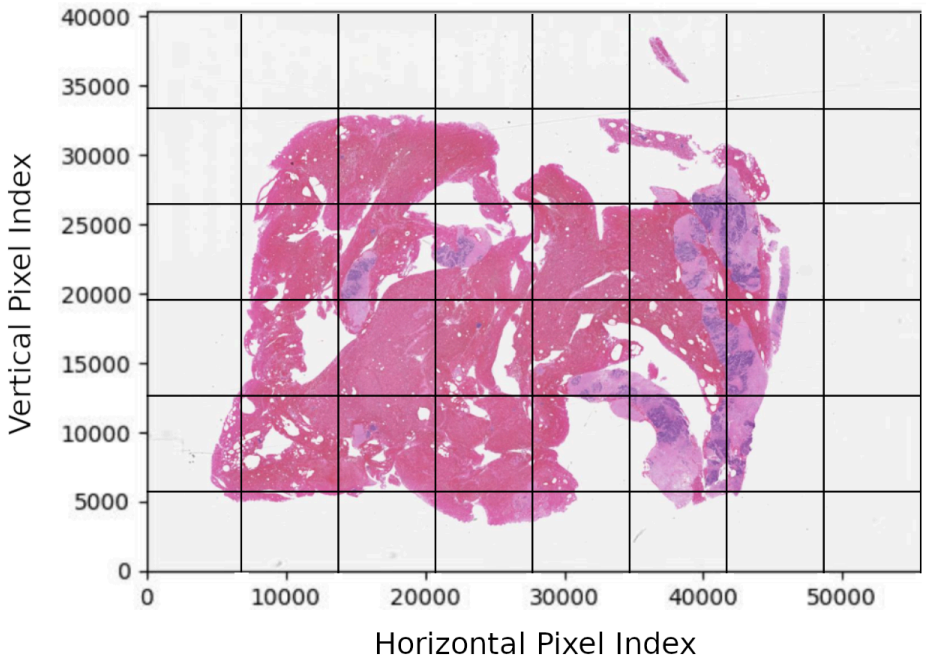
Biopsy slides are formatted as high resolution image ScanScope Virtual Slide (SVS) files. Breast cancer biopsies (Fig. 1) are photographed at multiple resolutions and multiple focal depths.[1] The images are compressed using Joint Photographic Experts Group (JPEG) compression and annotations are formatted in Extensible Markup Language (XML). Each slide contains one to six tissue samples.

Figure I: Digitized biopsy slide featuring two tissue samples.



Our progress thus far has been to segment each biopsy image into a uniform grid of squares that we term ‘frames’ (Fig. 2), with the intent of training on frames rather than whole-slide images (WSI). The advantage of frame-level classification over WSI is that it provides more feedback about the location of a malignancy rather than classifying a whole tissue sample as malignant or non-malignant.

Figure II: Segmentation of a biopsy slide into a uniform grid of frames. (Note: the real frames are significantly smaller)

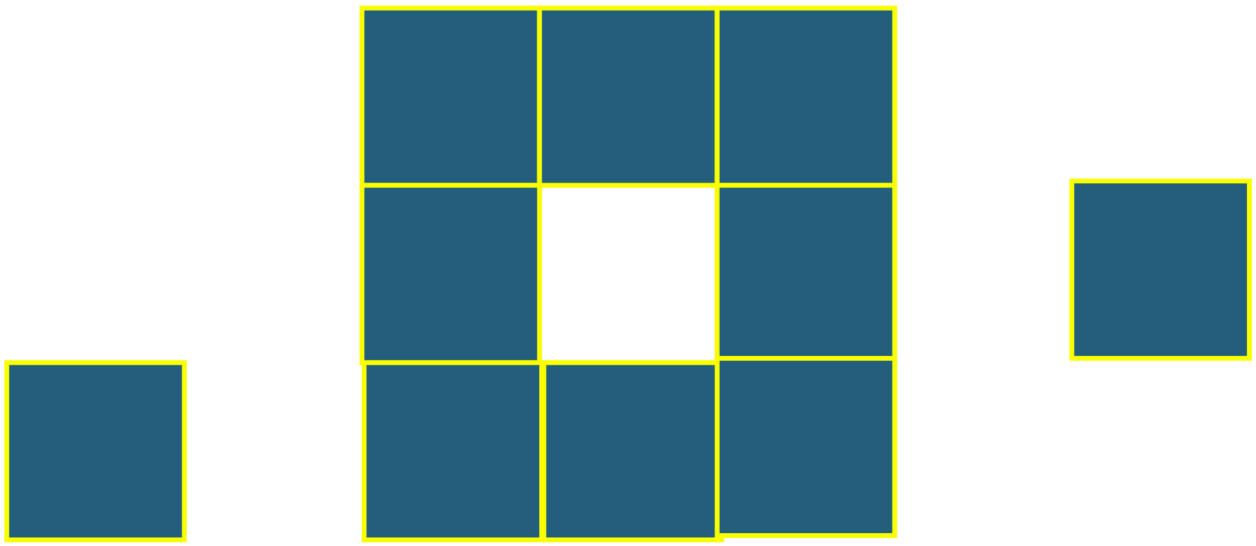


Frame-level classification comes at the cost of imbalancing malignant and non-malignant classes. Because there are a greater number of non-malignant frames in each biopsy slide, the model will tend to skew in favor of non-malignant classification.

One potential solution to the problem of unbalanced data is to artificially balance the two classes. We may downsample the number of non-malignant frames or upsample by duplicating a random selection of malignant frames. Each of these methods has its caveats; downsampling discards useful training data and upsampling introduces no new information. Our preferred alternative is to lower the malignancy probability threshold for the classifier and let the examining pathologist analyze frames above the stated threshold without downsampling/upsampling.

Another pitfall of frame-level classification concerns frame continuity. Fig. 3 represents potential output of our classifier, where the blue areas are frames above a malignancy probability threshold (e.g., $P > 0.5$) and white areas are below the threshold. In the center of Fig. 3 is a region bounded by malignant frames that contains a frame below the threshold but inside the bounding region. If a tumor has extended to such a boundary, then naturally the interior must be malignant as well. Our task is to identify interior regions and reclassify them with the appropriate malignant class.

Figure III: Segmented grid demonstrating low malignancy probability(white) and high malignancy probability (blue) classses.



There are many potential methods of separating bounded from unbounded regions with varying levels of complexity. We may inject a separate machine learning model into the processing pipeline, trained solely on identifying bounded shapes, use semantic segmentation, a deep-learning method of identifying and separating objects, or heuristic methods of parsing the matrix against simple logic. The simplest solution is the easiest to debug and requires the fewest compute cycles, so we opt to use a four-way flood-fill algorithm.

To implement the flood-fill algorithm, we start by filling in the white area in Fig. 3, then inverting the colors white and blue. This exposes the interior region of the bounded figure, which can be applied to the original image as a mask. Flood-fill algorithms are commonly used in graphics editors' paint-bucket tools. We have tested a recursive stack-based implementation of flood-fill (iterating and appending pixel locations to a data structure), but other implementations exist and each has its trade-off in terms of time, memory and processor usage. For example, a span-filling flood-fill algorithm fills an area row-by-row. Span-fill is a lot faster than recursion, but consumes more memory and time as it revisits filled pixels. The exact implementation of flood-fill will be chosen based on time, processor and memory usage after testing.

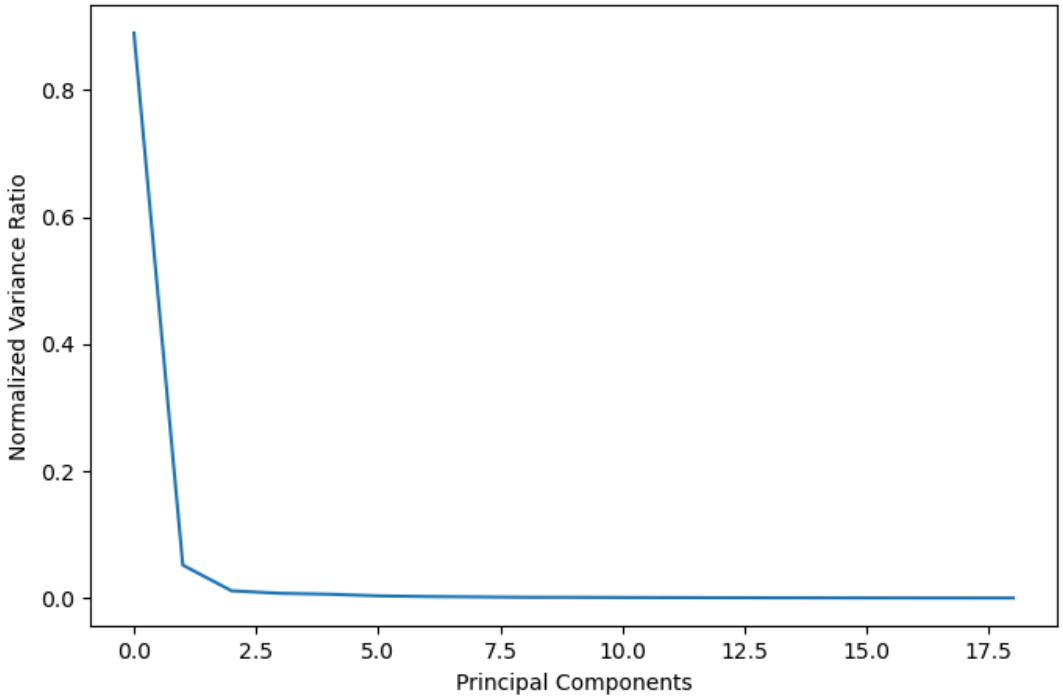
In order to actually start classifying frames, we face the problem of feature extraction. In other words, how physical data is represented as numbers for training, compression, or transformation. Good feature extraction leads to better training outcomes. Bitmap images encode information into four values per pixel: red, green, blue and alpha (transparency) values. Our pre-processor pipeline applies a 2-dimensional discrete cosine II transform (DCT-II) to map each color and transparency from the pixel domain to the discrete frequency domain. This frequency domain is the basis of our features for our machine learning model due to its ability to allow us to greatly reduce the size of feature vectors while maintaining significant information by trimming frequencies. While this loses granularity of the data, it

allows us to create a temporary shortcut for dimensionality reduction which could possibly be better refined using PCA.

This dimensionality reduction is necessary as each biopsy slide is stored as an image of 50,000 horizontal pixels, 50,000 vertical pixels, and 4 colors (red, green, blue, alpha/transparency). If we apply 2-dimensional DCT-II to a single slide, we end up with 4 x 50k x 50k potential features in the frequency domain. Without trimming frequencies as mentioned before, training a model on the full dataset would take much longer than we have this semester. There exists a phenomenon in math and science known as the curse of dimensionality. In exceedingly high dimensions, distances between points are equal. As a rule-of-thumb, the number of features on which a model is trained should be the square root of the number of test cases in the training set. This motivates us to reduce the number of features while maintaining as much variance or information in each feature as possible. The class of algorithms devoted to this problem are known as 'dimension reduction' algorithms.

The most popular dimension reduction algorithm is principal component analysis (PCA). PCA works by realigning the axes of the feature space to the line of best fit for each combination of features. In practice, PCA produces an equal number of features to the generating set, each of which is a linear combination of the generating features. The new features are known as 'principal components' and are ordered by descending variance (Fig. 7). This allows the data analyst to pick and choose the principal components that best retain the predictive ability of the original feature set.

Figure VII: Principal component analysis applied to high-resolution JPEG photo of a tiger.

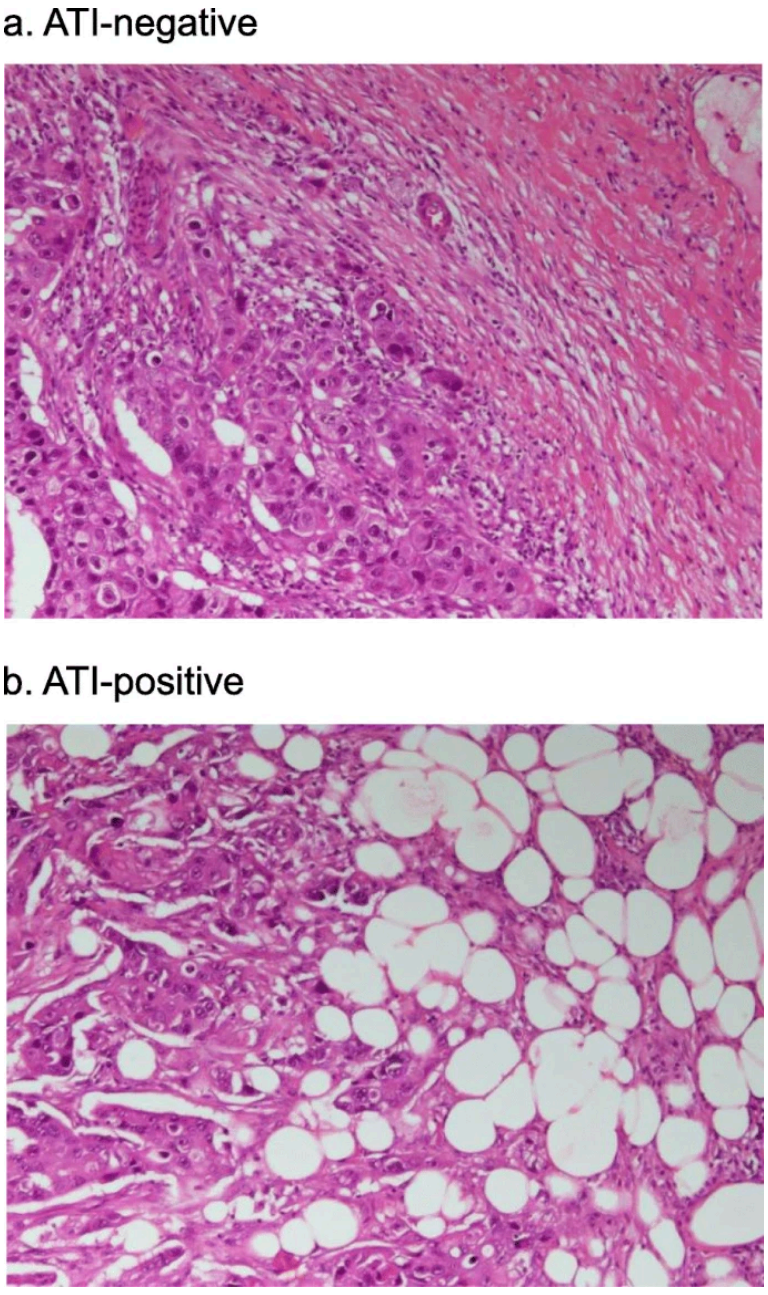


Other dimension reduction algorithms include Uniform Manifold Approximation and Projection (UMAP), which is similar to PCA but maps to nonlinear functions, and Feature Ordering by Conditional Independence (FOCI), which orders classes of features (not just the features themselves) by predictive ability. Which dimension reduction algorithm to choose for our purpose is not obvious, but we plan to

test PCA and UMAP on the training set, and weigh them by their abilities to discriminate between classes and by their computation times.

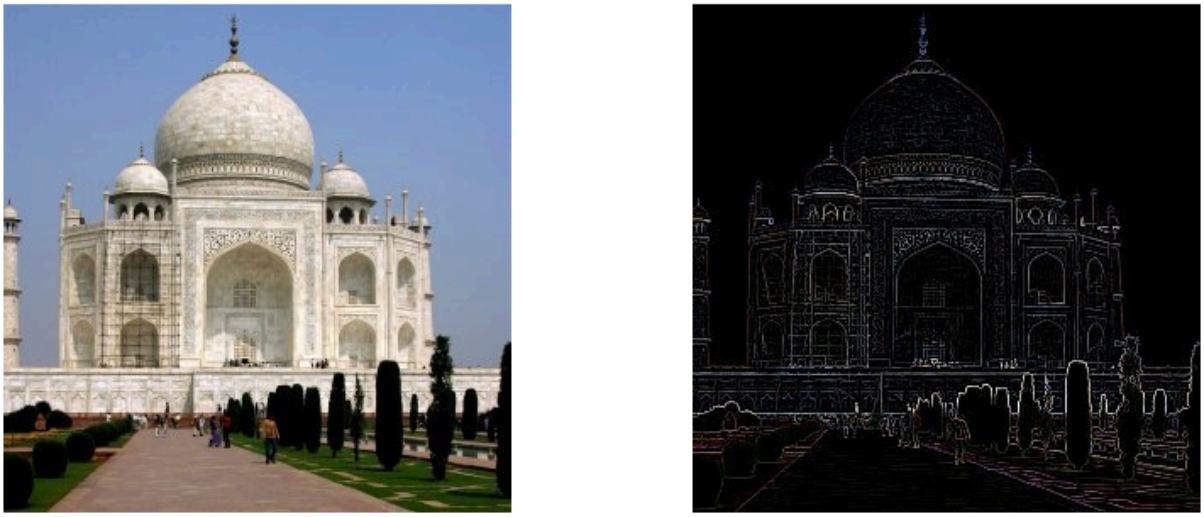
Of particular interest to us is the issue of separating areas of tissue with different biological functions (for example, epithelial and connective tissue). Different types of tissue have different appearances. By separating tissue, we may determine the boundaries of a malignancy, as certain types of tissue (e.g., adipose/fatty tissue) are more likely to become malignant than others (Fig. 4). Separating tissue also allows us to accurately annotate malignancies for the reviewing pathologist.

Figure IV: Adipose tissue invasion in negative and positive cases of triple-negative breast cancer.[3]



To separate different types of tissue, we plan to test different image filters. Filters for consideration include gradience, divergence, curl, Sobel operators or Laplacian filters. Filters may be employed separately or in succession, and may be used to annotate images or aid in feature extraction. The utility of each operator varies by application. For example, The Laplacian operator is useful for edge detection (Fig. 5).

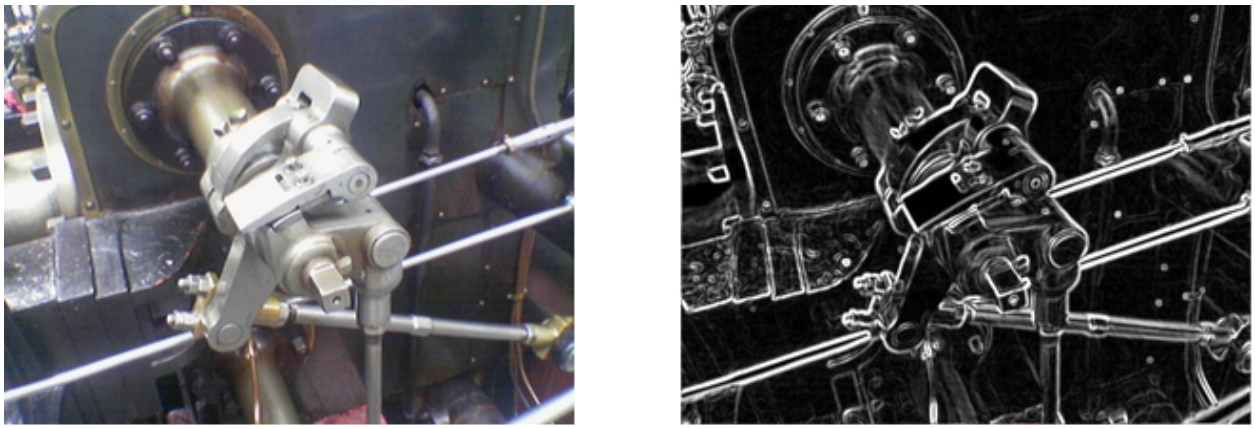
Figure V: Application of the Laplacian filter to a photo of the Taj Mahal.[4]



Original image Filter “Laplace” applied

Likewise, the Sobel operator is a convolutional algorithm used to approximate a gradient of image intensity (Fig. 6).

Figure VI: Application of the Sobel filter to a photo of an engine.[5]



Original Image Filter "Sobel" Applied

By comparing different intensities, we may rigorously separate different areas of an image. The Sobel operator works by taking the first derivative of image intensity, against which it convolves a small kernel in a raster pattern, allowing the algorithm to create well-defined borders.[6] The Laplacian filter uses the second derivative of image intensity. This allows the Laplacian filter to define inward and outward

borders. In other words, where the Sobel operator reveals a single border as two pixels wide, the Laplacian filter produces an outer and inner border, each one pixel wide.[7]

After extracting features from our training data, we must train the machine learning model. Currently, we are using the random forest (RNF) supervised ensemble method, which generates a decision tree seeded from the training data. Each decision, or split, in the tree minimizes the p-value (probability of the null hypothesis–the hypothesis that asserts there is no correlation between features and malignancy). Decision trees may also be generated via Gini impurity, entropy or information gain. Random forests, in particular, randomly sample the features for each split from the total set of features. We also employ a tactic called bootstrapping, sampling multiple datasets from the training set. Each decision tree is assigned its own bootstrapped dataset. When a prediction is made, each decision tree votes on the classification. Then, the model generates a confidence percentage for each class using these votes.[8]

Moving forward, we plan to use a Convolutional Neural Network (CNN). A CNN uses a construct called convolutional Layers. Each layer extracts features and condenses them into a smaller subset of features by a technique known as max-pooling. The training loop repeats until the model is able to narrow the features down to a list of classes. When fed a piece of evaluation data, it will repeat this process and return a confidence percentage for each class.[9] The training algorithm chosen will be the one that minimizes the model's error rate relative to human-annotated biopsy slides.

Two methods we can use to fine tune some of the model hyperparameters are the gradience of the loss function and divergence of datasets. Using gradience allows us to utilize feedback from the models in the form of a loss function. By taking the derivative of a loss function for each parameter, we can attempt to refine our model. Since the slope isn't guaranteed to have a single critical value, it may also involve random perturbation to make sure the model doesn't get stuck at a non-optimized parameter value.[10] On the other hand, divergence is a way for us to compare datasets. Using this method we can compare two datasets to see how much information is lost between the two. Since we will possibly be utilizing a method of breaking the dataset down into small subsets using a technique called bootstrapping or bagging, it's important that we ensure these subsets of data minimize the loss of information between them in order to reduce bias.[11]

In summation, the next goals moving forward are to complete the pipeline for training and scoring our model and testing. Then, from there we can begin to test methods for dimensionality reduction, and hyper-tuning parameters. Finally, we will prepare a user-friendly GUI to be used in future presentations.

References

[1] Nabila Shawki *et al.*, "The Temple University Hospital Digital Pathology Corpus," *Springer eBooks*, pp. 69–106, Jan. 2020, doi: https://doi.org/10.1007/978-3-030-36844-9_3.

[2] Nirag Jhala, "Digital pathology: Advancing frontiers," *IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, Dec. 2017, doi: https://doi.org/10.1109/spmb.2017.8257013.

[3] J. Yamaguchi *et al.*, "Active behavior of triple-negative breast cancer with adipose tissue invasion: a single center and retrospective review," *BMC Cancer*, vol. 21, no. 1, Apr. 2021, doi: https://doi.org/10.1186/s12885-021-08147-2.

[4] "8.4. Laplace," *Gimp.org*, 2024. https://docs.gimp.org/2.10/en/gimp-filter-edge-laplace.html (accessed Sep. 16, 2024).

[5] Wikipedia Contributors, "Sobel operator," *Wikipedia*, Nov. 22, 2019. https://en.wikipedia.org/wiki/Sobel_operator

[6] "How the Sobel Operator Works – Automatic Addison." https://automaticaddison.com/how-the-sobel-operator-works/

[7] Bernd Jähne, *Digital image processing*. Berlin ; New York: Springer, 2005, pp. 345–347.

[8] Scikit-learn, "sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.3 documentation," *Scikit-learn.org*, 2018. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[9] https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns (accessed Sep. 16, 2024).

[10]J. Lagos, "Understanding Gradients in Machine Learning - Analytics Vidhya - Medium," *Medium*, Oct. 23, 2020. https://medium.com/analytics-vidhya/understanding-gradients-in-machine-learning-60fff04c6400 (accessed Sep. 16, 2024).

[11]Kumar Vishwesh, "Understanding KL Divergence in Machine Learning: Applications and Improving Model Accuracy - Data Science Stunt," *Data Science Stunt*, Apr. 29, 2023. https://datasciencestunt.com/kl-divergence-machine-learning/ (accessed Sep. 16, 2024).