The Cancer Genome Atlas Kidney Clear Cell Carcinoma ECE 4783 - Final Project Report

Alexander Faché
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA, USA
afache3@gatech.edu

***Index Terms*—The Cancer Genome Atlas (TCGA), kidney cancer, histopathological, necrosis, stroma, tumor, medical image processing, Reinhard's Method, PCA, KNN, SVM, Random Forest**

I. Introduction

This document will perform an in-depth analysis of key components related to clinical decision support for the The Cancer Genome Atlas project presented in ECE 4783 Introduction to Medical Image Processing. First, a discussion showing the importance of TCGA and the use of histopathological images from cancer patients will set the stage. Following this discussion, a literature review to learn more about feature extraction, supervised learning models, and implementation methods will be discussed. Next, explanations of the preprocessing and feature extraction steps required to ready our dataset for testing and training will be stepped through. To conclude this document, supervised learning results will be presented.

II. The Cancer Genome Atlas

A. Overview

In order to support clinical decisions among cancer patients, a model for diagnosis and prognosis of cancer samples will be developed. We observe the image processing pipeline: normalization, feature extraction and selection, and ultimately prediction modeling. In this document, we will discuss applying image processing techniques to cancer histopathological images in order to develop objective, reproducible decision support for the diagnosis and prognosis of kidney cancer.

B. Data

The provided data set for undergraduate students contains quality controlled tissue slide image samples of three stages of cancer: necrosis, stroma, and tumor for kidney clear cell carcinoma cells. These stages of cancer will be our labels and desired output for our trained decision support models. Necrosis classification is a result of premature death of cells. Stroma classification is healthy connective tissue cells. Tumor classification results from abnormal cell divisions that uncontrollably destroy body tissue. Fig. 1 shows sampled images of all three classes.

The quality control stage of the preprocessing began with 1000 original whole slide images (20,000 x 40,000 pixels). These images were subject to 512 x 512 pixel sub-section cropping to remove tissue folding, invalid stains, and empty regions of interest (ROI). In the end, 100 images of each class were presented.

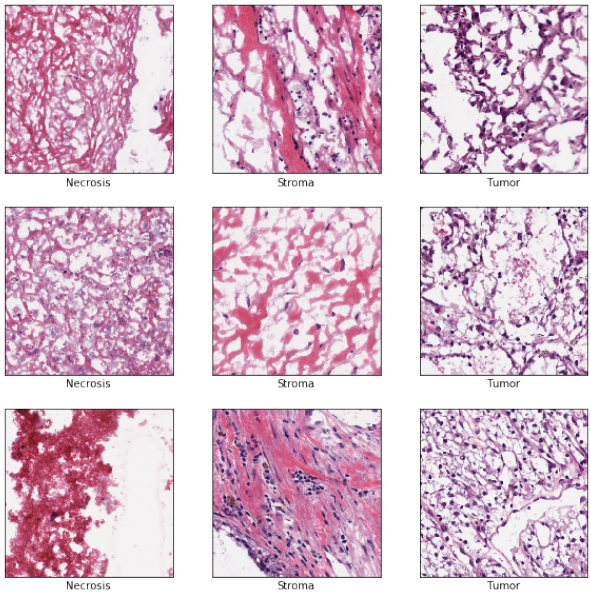

Fig. 1. Sampled images that have undergone "quality control".

C. Goal

Given the provided data set that has undergone quality control, the goal of the image processing pipeline is to create a model that actively supports clinical decision making with high precision. The ability the accurately distinct between necrosis, stroma, and tumor classifications will drive this precision.

ECE 4783 Introduction to Medical Image Processing, Spring 2020.

III. Literature Review

A. Overview

In order to gain more insight into the process of feature extraction and supervised learning methods, several papers were analyzed. The following discussion looks at color, texture, and morphological based features as well as Random Forests, K-Nearest Neighbors, Support Vector Machine, and Logistic Regression concepts.

B. Color Based Features

1) Automatic Batch-Invariant Color Segmentation of Histological Cancer Images [1]: Due to variations in tissue specimen preparation, staining, and imaging, color distribution among different samples may vary. A method for resolving such differences is the batch effect where users of the imaging equipment can incorporate domain knowledge during the segmentation process. However, human error was determined to lower objectivity, reproducibility, and speed. With a novel color normalization scheme and domain expertise, a system resistant to batch effects was developed. The proposed solution normalizes images over the color map instead of over the pixels. The issue with pixel level normalization is that it tends to lead to distorted colors in the resulting normalized image. Normalizing over the color map extracts the unique colors however, since it does not look at individual pixels, it does not include the frequency of colors. Results show that color map normalization was more accurate than the all pixels method.

2) Histological image feature mining reveals emergent diagnostic properties for renal cancer [2]: Using colorspaces from 28 different Gabor filters, this paper examines renal cancer images, traditionally a very difficult type of cancer to work with. Each filter and filtered image has a unique energy and entropy, which proved to be one of the highest weighted features. Unlike many other papers in the medical imaging field, this paper opts to classify beyond malignant and benign, but performs prediction modeling with non-binary outcomes. Fractal methods are used for texture features, while Voronoi diagrams are used for topological features.

C. Texture Based Features

1) Automatic Classification of Pathological Prostate Images Based on Fractal Analysis [3]: Standard texture based features are rated using the Gleason grading system. Using a novel texture analysis techniques based on a fractal, detection rates for pathological prostate images were improved compared to previous multiwavelets, Gabor filters, and GLCM methods. The fractal dimension feature set is smaller, however, still effective according to the Gleason grading system.

2) Automated classification of brain tumor type in whole-slide digital pathology images using local representative tiles [4]: Coarseness versus fineness, unlike some other texture features, is very difficult to distinguish by histopathologists. Using a dataset of brain tumor images to classify cancer subtypes, this paper segments its images into tiles, then performs typical analysis on the tiles and weighs the tile proper and its neighboring regions. The results of this paper boasted very high accuracy ratings in both of its examined cases of cancer subtypes with an accuracy rating greater than 95%.

D. Morphological Based Features

1) Mitosis detection in breast cancer pathology images by combining handcrafted and convolutional neural network features [5]: Hand crafting features is a time consuming process and it is sometimes difficult to get an adequate amount. With increasing amount of data available, more of these processes can be automated. In order to assist hand crafted feature creation, convolutional neural networks (CNN) aim to learn features that could otherwise not be represented through hand crafted means. It is important to note that hand crafted features are not being ignored as they better capture domain specific applications. The development of a lightweight CNN cascaded with hand crafted features is a faster approach and less computationally expensive than a stand alone CNN.

2) Object-and Spatial-Level Quantitative Analysis of Multispectral Histopathology Images for Detection and Characterization of Cancer [6]: This thesis follows a dataset of breast cancer images by creating multiple classes for object recognition. Multispectral analysis on nuclear features did not seem to provide as significant features as the non-nuclear features. Feature extraction for this thesis is very well documented, which provided the basis for many of the texture and morphological features used in this project. While this thesis did not have as many novelties as the other papers published, its documentation is much more thorough.

E. Supervised Learning Methods

1) Image Classification using Random Forests and Ferns [7]: Considers the image classification problem using the Caltech-101 and 256 image datasets. They intend to compare the performance of random forests and random ferns to a benchmark multi-way SVM. The reason for selecting a random forest and fern is that it is faster to train compared to a SVM. For the object detection process, they employ a strategy that uses a learned region of interest to eliminate regions of high visual similarity corresponding to the background of an image. Compared to a multi-way SVM, a random forest performed 80.0% vs 81.3% on the Caltech-101 dataset however it had a classification time that was 40 times quicker. The paper demonstrated that using random forests and ferns reduces training and testing costs significantly compared to a multi-way SVM, and has comparable performance.

2) Efficient kNN Classification With Different Numbers of Nearest Neighbors [8]: Uses a pairing of KNN and Decision Trees for a more optimal way of determining the number of neighbors for each test sample. Instead of class labels as the leaves of the decision tree, values of k are used instead.

3) Speech Emotion Recognition using Support Vector Machine [9]: Aims to classify emotions through speech using Support Vector Machine. Use MFCC (Mel-frequency cepstrum) and LPCC (Linear Prediction Cepstral Coefficients) algorithms. Using a trained actors, noiseless environment (LDC) and a student, noisy environment (UGA) speech datasets, they

showed that using gender dependent classification was more accurate than one-against-all methods.

4) Multiple Sclerosis Detection Based on Biorthogonal Wavelet Transform, RBF Kernel Principal Component Analysis, and Logistic Regression [10]: Uses MRI data to distinguish between multiple sclerosis and healthy patients using a combination of biorthogonal wavelet transform, RBF kernel PCA, and logistic regression. The combined method had a greater sensitivity and accuracy than 5 state-of-the-art approaches.

IV. Module 1 - Image Preprocessing

A. Overview

In the order to feed in our training samples to a model we first need to preprocess our image data. Preprocessing is important as it serves to reduce noise, perform any clean ups, as well as expand the dataset. The following sections will outline Normalization and data augmentation steps that were applied as well as the result.

B. Normalization

First color normalization will be applied to remove any side effects of lighting. The proposed implementation is Reinhard's Method which uses the CIELAB color space [11]- [12]. The CIELAB color space is represented by:

- L^* - lightness from black to white
- a^* - lightness from green to red
- b^* - lightness from blue to yellow

The implementation makes use of a target image which is used as a reference image. This image contains the distribution of color (RGB channels) that is desired to be mimicked by the source image. Because $l\alpha\beta$ is a transform of the LMS color space, RGB to LMS then LMS to $l\alpha\beta$ transformations must be performed. First, the source image is transformed from the RGB color space to LMS color space via (1):

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.5783 & 0.0402 \\ 0.1967 & 0.7244 & 0.07802 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1}$$

Then the LMS image is transformed to $l\alpha\beta$ color space via (2):

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix} \tag{2}$$

Once in LAB color space, the mean is subtracted from each channel via (3):

$$\begin{aligned} l^* &= l - \bar{l} \\ \alpha^* &= \alpha - \bar{\alpha} \\ \beta^* &= \beta - \bar{\beta} \end{aligned} \tag{3}$$

Then the data is scaled by the respective standard deviations (4):

$$\begin{aligned} l' &= \frac{\sigma_t^l}{\sigma_s^l} l^* \\ \alpha' &= \frac{\sigma_t^\alpha}{\sigma_s^\alpha} \alpha^*, \\ \beta' &= \frac{\sigma_t^\beta}{\sigma_s^\beta} \beta^* \end{aligned} \tag{4}$$

where σ_t stands for the target and σ_s stands for the source standard deviations. Finally, the modified source image is transformed from $l\alpha\beta$ to LMS via (5):

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & -2 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{3}}{3} & 0 & 0 \\ 0 & \frac{\sqrt{6}}{6} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} \tag{5}$$

Then returned to RGB color space via (6):

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4.4679 & -3.5873 & 0.1193 \\ -1.2186 & 2.3809 & -0.1624 \\ 0.0497 & -0.2439 & 1.2045 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix} \tag{6}$$

Before applying normalization on tissue cells where it may be difficult to determine the immediate effect, the normalization process was applied on a test image.

It is clear from Fig. 2 that the darkened sky and grass from the target image (top) was successfully applied to the source image (middle) resulting in the normalized source image (bottom).

After validating normalization on the test image, the normalization technique was applied using target images Fig. 3, Fig. 5, Fig. 7. The resulting sample normalizations are displayed in Fig. 4, Fig. 6, Fig. 8. From top to bottom, necrosis, stroma, and tumor samples are displayed.

Using the three target images applied to the entire 300 image data set results in 897 normalized images. The target image is removed from the normalization process.

C. Data Augmentation

In order to generalize machine learning models, large amounts of samples are required. However, copious amounts of data are not easy to come by. In order to expand the dataset without collecting more tissue samples, data augmentation is applied. This includes cropping, flipping, and rotating the previously normalized images.

Cropping: The normalized images are originally 512 x 512 pixels. 224 x 224 pixels sub regions will be taken as this is a common input layer size to convolutional neural networks (CNNs) if the approach is to be taken in the future. Five 224 x 224 pixel sub regions are selected at random from each sample image.

Flipping: To slightly alter the cropped images, independent vertical and horizontal flips will be performed. A vertical flip of an image is created by flipping the rows of an image shown via (7). Given an image I:

$$I_{new}[:: -1, :, :] = I[:, :, :] \tag{7}$$

Fig. 2. Example output after normalization.

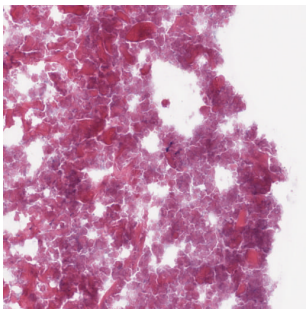

Fig. 3. Necrosis as target.

A horizontal flip of an image is created by flipping the columns of an image via (8):

$$I_{new}[:, :: -1, :] = I[:, :, :] \tag{8}$$

Rotating: Another alteration that was employed is rotation. A 90^o counter clockwise rotation is the product of first transposing the image then flipping the rows via (9)-(10):

$$I_{temp} = I^T \tag{9}$$

$$I_{new}[:: -1, :, :] = I_{temp} \tag{10}$$

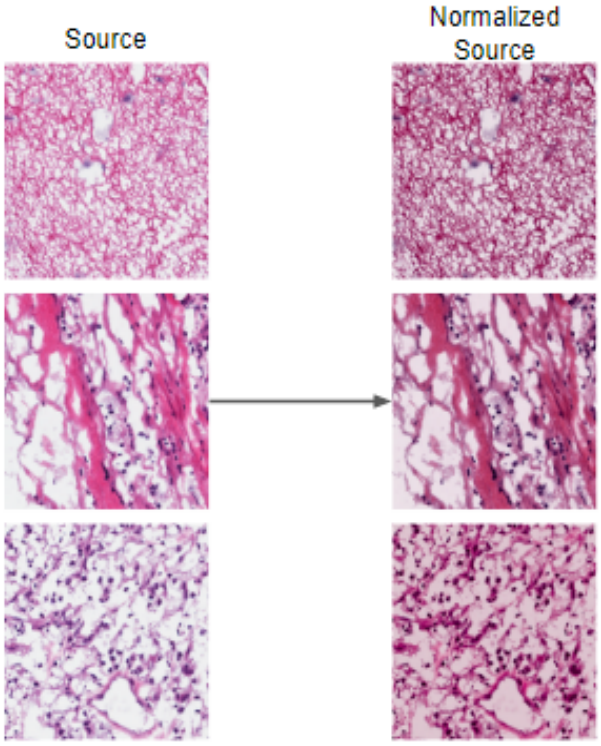

Fig. 4. Output from necrosis normalization.

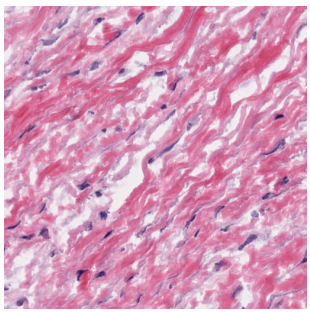

Fig. 5. Stroma as target.

A 180^o counter clockwise rotation is the product of first flipping the rows then flipping the columns via (11)-(12):

$$I_{temp}[:: -1, :, :] = I \tag{11}$$

$$I_{new}[::, :: -1, :] = I_{temp} \tag{12}$$

A 270^o counter clockwise rotation is the product of first transposing the image then flipping the columns via (13)-(14):

$$I_{temp} = I^T \tag{13}$$

$$I_{new}[:, :: -1, :] = I_{temp} \tag{14}$$

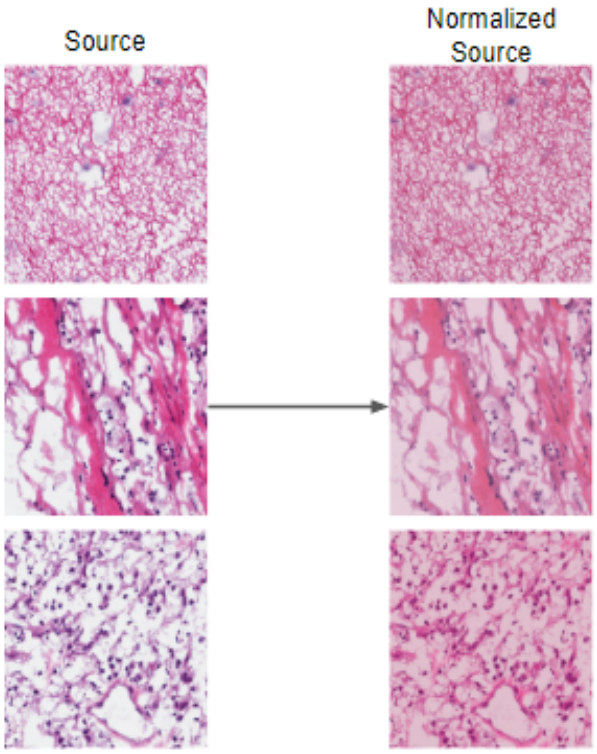

Fig. 6. Output from stroma normalization.

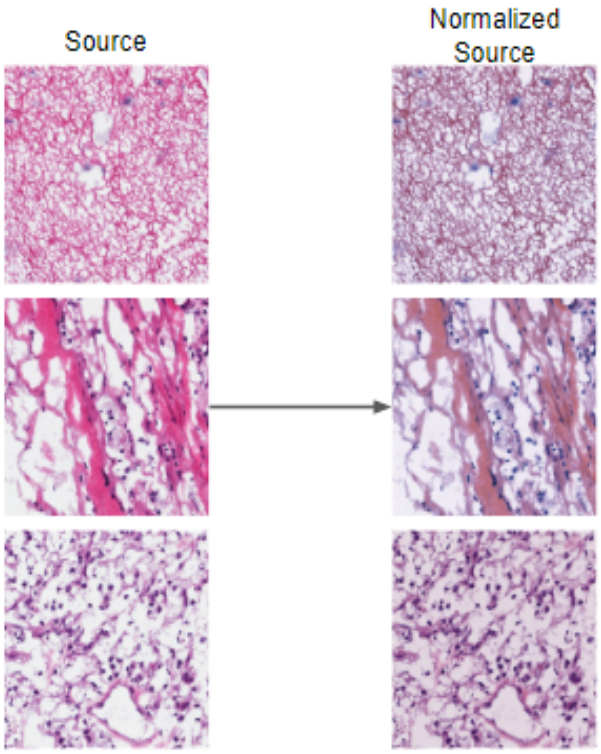

Fig. 8. Output from tumor normalization.

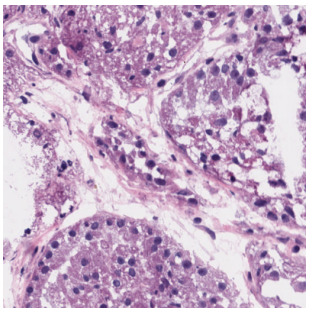

Fig. 7. Tumor as target.

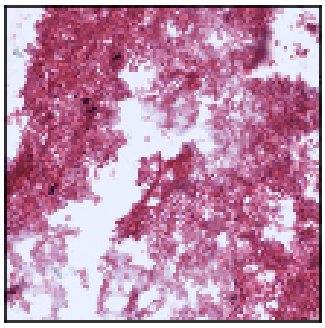

Fig. 9. Original normalized image selected for augmentation.

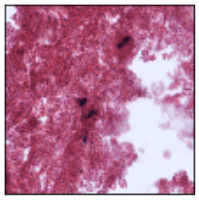

Fig. 10. Cropped sub region.

D. Results

Through two flips and three rotations, each image now has six new samples based on the cropped sub region (five augmented + one original). This process of selecting a random sub region, applying flipping, then rotating is applied a total of five times to result in 30 (5 * 6) samples for each original normalized image. Applying to all 897 images results in 26,910 images of dimension 224 x 224 pixels. A sample output for one iteration of this process is shown in Fig. 9-Fig. 12.

V. Module 2 - Feature Extraction and Selection

A. Overview

In Section IV, preprocessing and data augmentation were discussed in order to clean up the data set and expand the dataset. In this section, features will be extracted from the images to train and test machine learning models.

B. Description of Color Based Features

1) Color Spaces:

- RGB

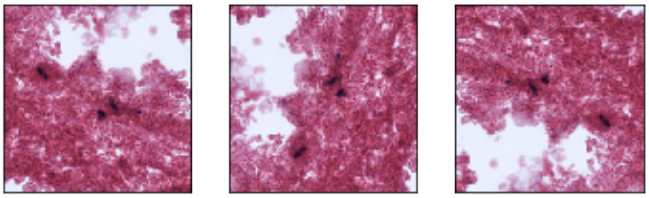

Fig. 11. Result from 90^o (left), 180^o (center), 270^o (right) counter clockwise rotations.

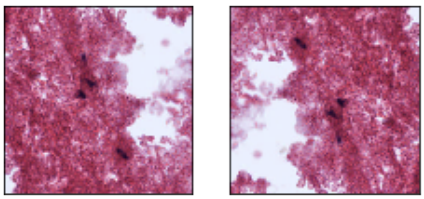

Fig. 12. Result from vertical (left) and horizontal (right) flips.

- HSV
- LAB

2) Color Channel Properties: Each color space consists of three different color channels represented by a matrix I with r rows and c columns. From each individual color channel the following properties were extracted using $scipy.stats.describe$.

- Minimum value
$$\min_{r,c} I(r,c) \tag{15}$$
- Maximum value
$$\max_{r,c} I(r,c) \tag{16}$$
- Mean value
$$\mu = \frac{1}{N}\sum_{r,c} I(r,c) \tag{17}$$
- Variance
$$\sigma^2 = \frac{1}{N}\sum_{r,c}(I(r,c)-\mu)^2 \tag{18}$$
- Skewness
  - measure of a lack of symmetry
  - degree of symmetry
  - $\widetilde{I}$ is the channel median
$$\frac{3*(\mu-\widetilde{I})}{\sigma} \tag{19}$$
- Kurtosis (Fisher)
  - measure of the pointedness of peak
  - degree of peakedness
$$E\left[\left(\frac{I-\mu}{\sigma}\right)^4\right] \tag{20}$$
- Number of pixels per channel
$$N = r*c \tag{21}$$

C. Analysis of Color Based Features

The purpose of the aforementioned statistical analyses is to identify outliers, as in theory due to the Reinhard normalization process many of these statistics should be very similar image to image. In addition, the red stain applied when capturing the tissue images aids in separating the color of the output images.

D. Description of Texture Based Features

1) Grey-Level Co-Occurance Matrix properties: GLCM = C(i, j) is determined with $skimage.feature.greycomatrix$.

- Contrast
$$\sum_{i,j} C(i,j)(i-j)^2 \tag{22}$$
- Dissimilarity
$$\sum_{i,j} C(i,j)|i-j| \tag{23}$$
- Homogeneity
$$\sum_{i,j} \frac{C(i,j)}{1+(i-j)^2} \tag{24}$$
- ASM
$$ASM = \sum_{i,j} C^2(i,j) \tag{25}$$
- Energy
$$\sqrt{ASM} \tag{26}$$
- Correlation
$$CORR = \sum_{i,j} C(i,j)\frac{(i-\mu_i)(j-\mu_j)}{\sigma_i\sigma_j} \tag{27}$$
- Inertia
$$\sum_{i,j}(i-j)^2 C(i,j) \tag{28}$$
- Entropy
$$-\sum_{i,j} C(i,j)\log_2 C(i,j) \tag{29}$$
- MaxProb
$$\max_{i,j} C(i,j) \tag{30}$$
- Cluster Shade
$$sgn(A)|A|^{1/3} \tag{31}$$
- Cluster Prominance
$$sgn(B)|B|^{1/4} \tag{32}$$

$$A = \sum_{i,j} \frac{C(i,j)(i+j-2\mu)^3}{\sigma^3\sqrt{2(1+CORR)}^3} \tag{33}$$

$$B = \sum_{i,j} \frac{C(i,j)(i+j-2\mu)^4}{4\sigma^4(1+CORR)^2} \tag{34}$$

These properties are discussed in [6], [11]- [13].

2) Shannon Entropy:

$$-\sum_i p_i \log_2(p_i) \tag{35}$$

3) Signal to Noise Ratio:

$$10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) \tag{36}$$

4) Compression Ratio:

$$CR = \frac{filesize(CompressedFile)}{filesize(OriginalFile)} \tag{37}$$

5) Laplacian of Gaussian:

$$\begin{aligned} LoG = conv2D(Laplacian, ksize = 3, \\ conv2D(Gaussian, ksize = 3, image)) \end{aligned} \tag{38}$$

E. Analysis of Texture Based Features

We primarily analyze texture based features by transforming into different domains. Two helpful transforms, not listed above, are the Discrete Cosine Transform (DCT) and Fast Fourier Transform (FFT) both of which tell us many things about the texture of the image. The DCT, often used in image compression, represents the image as a sum of cosines. The more nonzero coefficients of the cosine terms, the more complex the texture of the image is. This is also reflected in the entropy of the image. The FFT puts the image into the frequency domain, which tells us which parts of the image change quickly (i.e. pure black to pure white) and which change slowly (i.e. red to darker red), another useful component for texture analysis. In future work, autoencoders can extract more texture based features.

F. Description of Morphological Based Features

1) Symmetry: Let I_v represent a vertically flipped image and I_h represent a horizontally flipped image.

- Vertical MSE

$$MSE_v = \frac{1}{N}\sum (I - I_v)^2 \tag{39}$$

- Horizontal MSE

$$MSE_h = \frac{1}{N}\sum (I - I_h)^2 \tag{40}$$

2) Area: The total number of pixels in the image object.

$$Area = \sum_{n,m} \Omega(n,m) \tag{41}$$

3) Object Centroids:

$$\bar{x} = \sum_{n,m} n\Omega(n,m) \tag{42}$$

$$\bar{y} = \sum_{n,m} m\Omega(n,m) \tag{43}$$

4) Major and Minor Axis Lengths:

$$major = 2\sqrt{2}\sqrt{m_{xx} + m_{yy} + \sqrt{(m_{xx} - m_{yy})^2 + 4m_{xy}^2}} \tag{44}$$

$$minor = 2\sqrt{2}\sqrt{m_{xx} + m_{yy} - \sqrt{(m_{xx} - m_{yy})^2 + 4m_{xy}^2}} \tag{45}$$

given

$$m_{xx} = \frac{1}{M}\sum_M (x_i - \bar{x})^2 + \frac{1}{12} \tag{46}$$

$$m_{yy} = \frac{1}{M}\sum_M (y_i - \bar{y})^2 + \frac{1}{12} \tag{47}$$

$$m_{xy} = \frac{1}{M}\sum_M (x_i - \bar{x})(y_i - \bar{y}) \tag{48}$$

where M is the number of pixels within the object.

5) Eccentricity: A measure of conic sections that shows how elliptical an object is–how far an object is from being circular.

$$ecc = \frac{2\sqrt{(\frac{major}{2})^2 - (\frac{minor}{2})^2}}{major} \tag{49}$$

6) Orientation: A measure of the angle between the major elliptical axis and the image's original x-axis.

$$\theta_o = \arctan\left(\frac{m_{yy} - m_{xx} + \sqrt{(m_{yy} - m_{xx})^2 + 4m_{xy}^2}}{2m_{xy}}\right) \tag{50}$$

7) Convex Area: The area of the convex hull of the object (from MATLAB).

$$ConvexArea = \sum_{n,m} ConvHull\Omega(n,m) \tag{51}$$

8) Convex Deficiency: The pixels within the convex hull that are not within the object (excluding spikes).

$$ConvexDef = \frac{ConvexArea - Area}{Area} \tag{52}$$

9) Solidity: The fraction of pixels within the convex hull computed by MATLAB that are also within the object.

$$Solidity = \frac{Area}{ConvexArea} \tag{53}$$

10) Perimeter: The distance around the boundary of the object. The last coordinate is identical to the first.

$$Per = \sum_{n=1}^{N} \sqrt{(x(n+1) - x(n))^2 + (y(n+1) - y(n))^2} \tag{54}$$

11) Equivalent Diameter: The diameter of the circle that could enclose the object.

$$EquivDiameter = 2\sqrt{\frac{\pi}{Area}} \tag{55}$$

12) Sphericity: The ratio of the smallest circle that encloses the object over the largest circle that can be enclosed by the object.

$$Sphericity = \frac{min(EquivDiameter/2)}{max(EquivDiameter/2)} \tag{56}$$

13) Proportion White Pixels:

$$\frac{\sum I > 220}{N} \tag{57}$$

14) Proportion Black Pixels:

$$\frac{\sum I < 100}{N} \tag{58}$$

G. Analysis of Morphological Based Features

We chose to invest more features into morphological features than texture and color space features. The most important of these are edge detection filters such as the Sobel, Canny, and Prewitt filters. Many of the aforementioned features are derived from MATLAB's built-in object analysis using the *regionprops* function. One of our primary goals for morphological feature extraction was to identify the locations and number of nuclei within the hisopathology images. At a first glance, it seems that performing texture analysis of the results of edge detection is a strong way of locating nuclei. Taking the FFT of the Perimeter as done by L. Bouchard in her PhD thesis outperformed the Hough Circle Transform [6]. This is theorized that since the nuclei are so small that the Hough Circle Transform becomes inaccurate.

H. Feature Scaling

Feature scaling is important in assisting machine learning models. Typically models perform better and converge faster when features are of similar magnitude and are normally distributed [14]. $sklearn.preprocessing$ offers several feature scalers one of which is the MinMaxScaler. For a feature F, MinMaxScaler subtract the minimum value of the feature then divides by the feature's range (59):

$$\begin{aligned} m_n &= \min F \\ m_x &= \max F \\ F_{scale} &= \frac{F - m_n}{m_x - m_n} \end{aligned} \tag{59}$$

MinMaxScaler preserves the original distribution and returns scaled features in the range $[0, 1]$.

I. Dimensionality Reduction

1) Principle Component Analysis (PCA): Dimensionality reduction using PCA was applied to reduce the number of features and complexity of the data set. PCA uses singular value decomposition in order to transform the original features to Z space. These new features, called principal components, are optimized in order to maximize the variance. By doing so they also become linearly independent allowing us to ignore redundant features. In order to determine the optimal number of principal components to keep, a scree plot is used, Fig. 13. A scree plot shows the individual recovered variance for each principal component in descending order. Along with a plot of the cumulative recovered variance, Fig. 14, the number of principal components can be selected to meet a desired recovered variance. Note that 100% recovered variance is equivalent to using the original number of features and therefore inappropriate for dimensionality reduction. Equation (60) shows the transformation from data set X to Z space using the top K principle components. Analyzing the scree plot and cumulative recovered variance of all principal components for this dataset showed that 97% recovered variance was an appropriate threshold. As a result the top 43 principal components were selected from the original 169 features. PCA is also used to plot data in a lower dimension (2-D, 3-D) for visualization. This is done by simply keeping the top two or three principle components. Shown in Fig. 5-Fig. 16: necrosis in red, stroma in blue, and tumor in yellow.

$$\begin{aligned} U, S, V &= SVD(X) \\ Z &= X \bullet V_{0...K} \end{aligned} \tag{60}$$

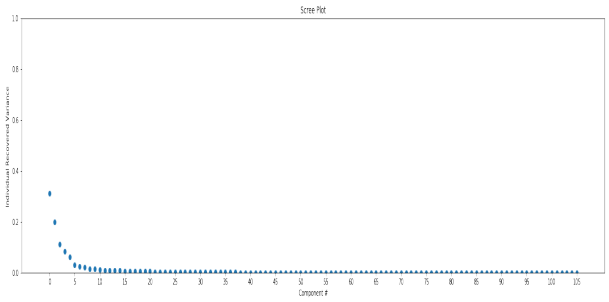

Fig. 13. Scree plot.

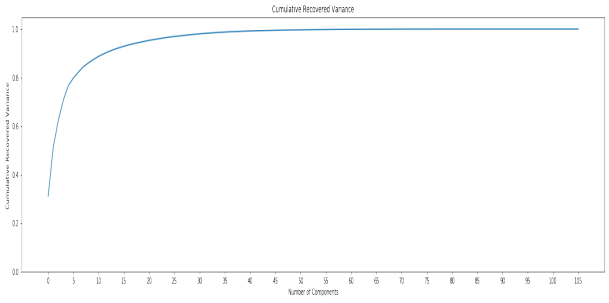

Fig. 14. Cumulative recovered variance.

2) Correlation Matrix: After performing PCA we expect to have completely independent features in order to avoid redundant features/information. As a verification of correct implementation, the original features were visualized using a correlation matrix. The different shades of red, white, and blue show some inter dependence, Fig. 17. After dimensionality reduction, a single diagonal red line in a field of blue indicates that all principal components are uncorrelated and the implementation was correct, Fig. 18.

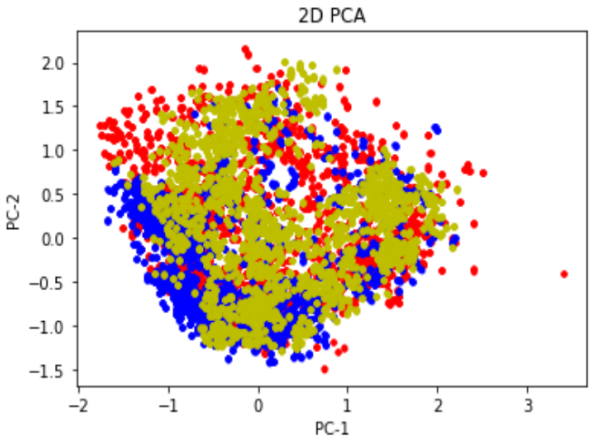

Fig. 15. Two-dimensional reduction of dataset using PCA.

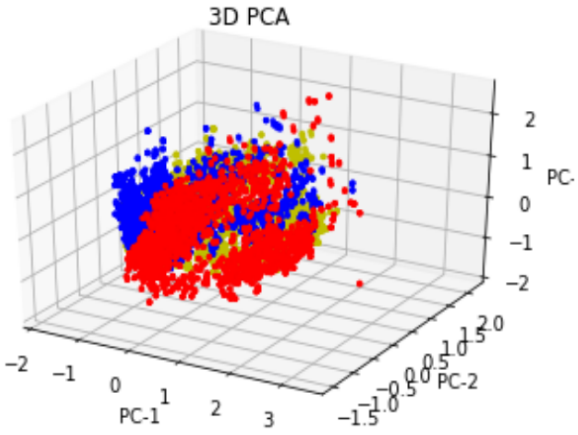

Fig. 16. Three-dimensional reduction of dataset using PCA.

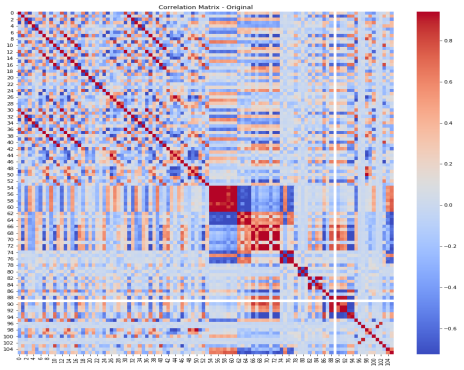

Fig. 17. Correlation matrix of original features.

VI. Module 3 - Classification for Clinical Decision Making

A. Overview

The final stage of the image processing pipeline consists of classification for clinical decision making. With knowledge of the given histopathological images, implementation of required preprocessing steps, and feature extraction analysis, machine learning models can be trained to differentiate between the three types of tissue stages: necrosis, stroma, and tumor. Several supervised machine learning models for decision boundary creation are discussed below.

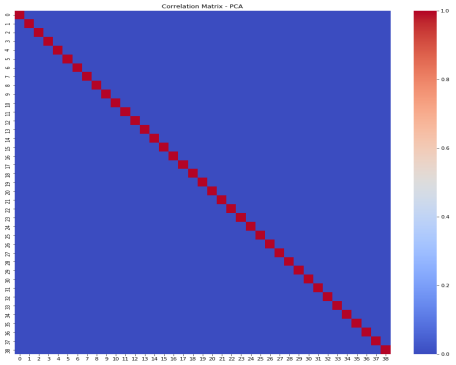

Fig. 18. Correlation matrix of Z space features after performing PCA.

B. Flowchart

The process of training supervised learning models is described by Fig. 19.

1) The dataset is comprised of 26,700 samples each with 43 features extracted from PCA. The labels are $\{0, 1, 2\}$ corresponding to $\{necrosis, stroma, tumor\}$, respectively.
2) The dataset is then split 70-30, train-test.
3) Supervised learning methods: K-Nearest Neighbors, Support Vector Machine, and Random Forest are selected for implementation.
4) Several cross-validation schemes including K-Fold and Shuffle-Split will be used to test the effectiveness of each model.
5) GridSearchCV is an exhaustive hyperparameter tuning method and will be used to determine the optimal parameters of each model.
6) Finally, performance on the test set will be measured with several metrics: Confusion Matrix, Accuracy Score, F1-Score, Matthews Correlation Coefficient, and Area Under the Curve.

C. Technical Background

Below, a quick discussion of the supervised learning models used in this analysis will highlight some of the implementation details.

1) K-Nearest Neighbors (KNN): K-Nearest Neighbors is a local classification method in that it only uses several training samples to determine the final prediction. The K stands for the number of these training samples, also known as neighbors, that will be used. The algorithm works by determining the

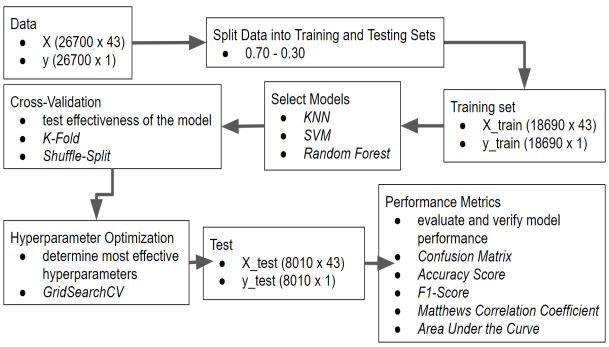

Fig. 19. Module 3 flowchart.

distance from a test data point to every data point in the training set. These distances are sorted in ascending order and the top K are selected. The resulting class label of the test data point is the mode of the selected subset. As an example, let's say we are given two classes red and blue and a green test data point, Fig. 20. With K equal to 4, a circle bounds the four nearest neighbors based on euclidean distance, Fig. 21. It is clear that there are 3 blue and 1 red label therefore classifying our green test data point as blue, Fig. 22.

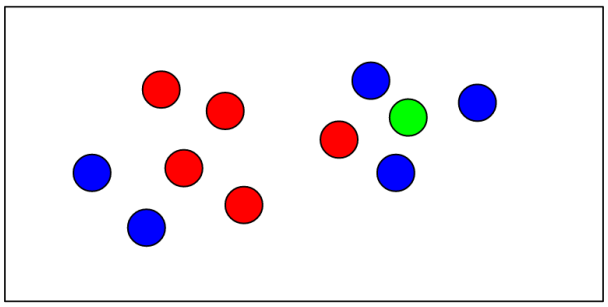

Fig. 20. Class 1 in red, class 2 in blue, test data point in green.

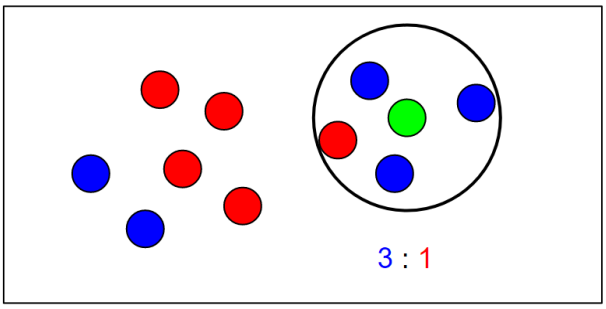

Fig. 21. With K = 4, the four closest data points are used to determine the green test data point's class.

2) Support Vector Machine (SVM): In order to create an effective decision boundary for classification, there needs to be a large enough cushion between classes such that it is robust and generalizable. Support Vector Machine provides a template to increase the decision boundary's margin because SVM creates a concrete decision boundary that only works

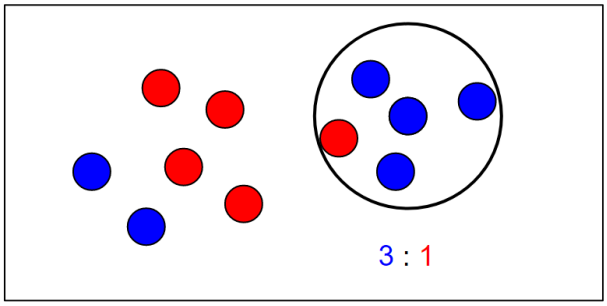

Fig. 22. Green test data point is assigned to the blue class.

for linearly separable data, methods such as soft SVM and kernels can be used for nonlinearly separable data, see Fig. 23-Fig. 24. This decision boundary is created with the help of support vectors. In determining these support vectors, (61) must be solved with (62)-(63). Because the problem is setup as a constrained optimization with respect to theta and b, a lagrangian multiplier is used. On top of this, because it is an inequality constraint and not a standard equality constraint, KKT is used. Solving the constraint using a quadratic programming package returns the initial lagrangian multiplier which in turn dictates the support vectors. In order to determine if the model correctly classifies its samples, a simple misclassification trick is used. Class labels are $\{-1, 1\}$ and the model's prediction is equivalent to (64). Therefore a correct classifcation results in the equality (65) equating to True, else False. A poor example of a decision boundary has a very small margin, Fig. 25. After applying SVM, new support vectors are determined and a better boundary is selected, Fig. 26.

$$y_i * (x_i * \theta + b) \geq 1 \tag{61}$$

$$|y_i| = 1 \tag{62}$$

$$|x_i * \theta + b| \geq 1 \tag{63}$$

$$(x_i * \theta + b) \tag{64}$$

$$y_i * (x_i * \theta + b) > 0 \tag{65}$$

3) Random Forest: Random Forests are described as Bagged Random Decision Trees. This is because, as a forest implies, multiple decision tree classifiers are collected together as one overall classifier. The bagged randomness comes from an implementation detail where only a random subset of features are used to train each decision tree. Entropy and information gain are the governing principles of decision trees. Entropy can be described as uncertainty in a prediction and is ideally close to 0 (66). Information gain can be viewed as a change in entropy (67). Therefore with a greater information gain, more uncertainty is removed. A decision tree works by recursively splitting the training set along a single feature

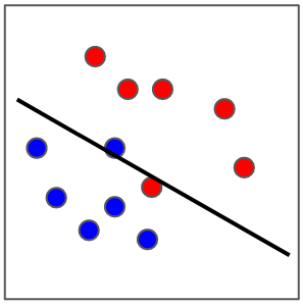

Fig. 23. Soft SVM.

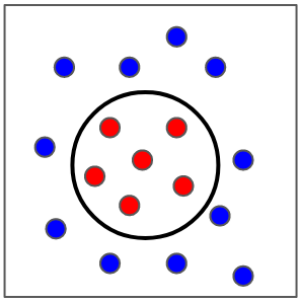

Fig. 24. Kernel.

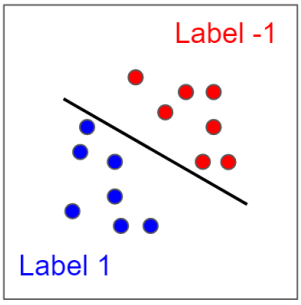

Fig. 25. Poor decision boundary margin.

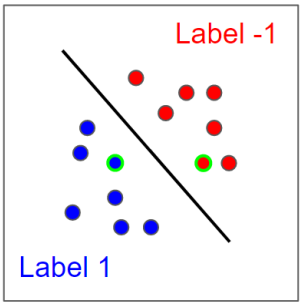

Fig. 26. Decision boundary with greater margin.

that results in the greatest information gain. Splits continue to happen until a subset of the training set can be classified as a single class resulting in the creation of a leaf node. Bagging in random forests is simply the process of training B different decision trees. When combined into an ensemble, the randomness in the decision trees reduces the variance of the model. The final prediction is then a majority voting scheme among all classifiers on a test data point (68).

$$H = -\sum_{i=1}^{M} P_i * log_2 P_i, M\, classes \tag{66}$$

$$IG = H - (H_L * P_L + H_R * P_R) \tag{67}$$

$$f(x) = majority\{f_j(x)\}|_{j=1:B}, B\, classifiers \tag{68}$$

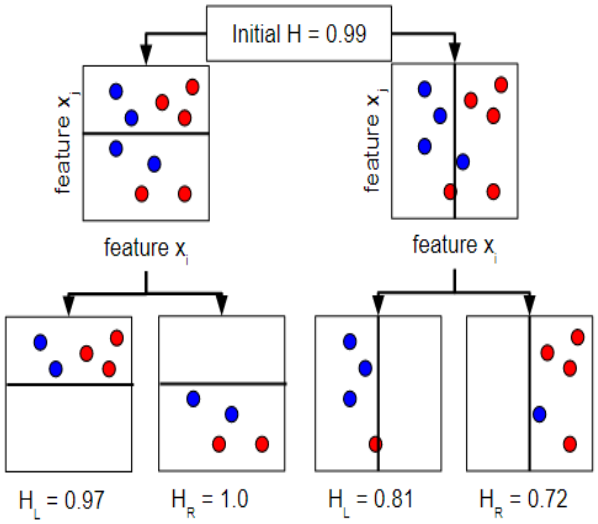

Fig. 27. Random Forest entropy and information gain splitting.

D. Cross-Validation Schemes

Cross-Validation is used to internally test the effectiveness of a model before hyperparameter optimization and training to determine if this model is worth further consideration. Two such methods will be used: K-Fold and Shuffle-Split both available through $sklearn$.

1) K-Fold: K-Fold performs K iterations of the model on the training set. In doing so, one of K training sections is used as an intermediate test set. As shown in Fig. 28, for each iteration a different subset is used as the training set.

2) Shuffle-Split: Shuffle Split performs K iterations of the model on the training set. In doing so, a random selection of training indices are used as an intermediate test set. As shown in the Fig. 29, for each iteration a different subset is randomly left out.

E. Hyperparameter Tuning

After cross-validation schemes have been applied to a bare bones model and it proves to be effective, the models parameters are optimized. An exhaustive grid search on a set of input parameters returns the estimator with the highest mean test score. Example hyperparameters that can be tuned for three models is shown below.

- KNN

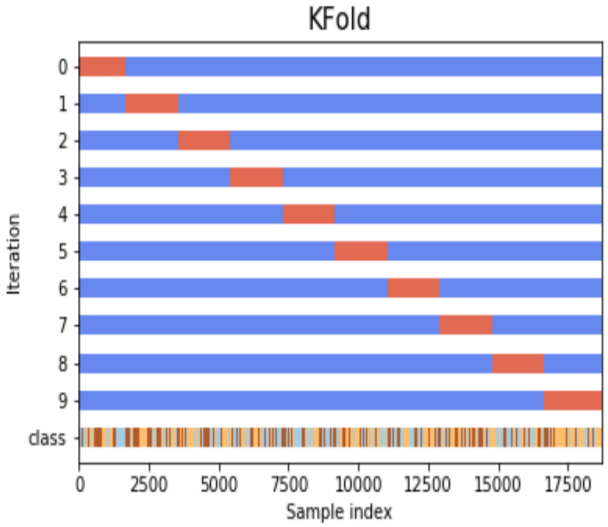

Fig. 28. Example train and test indices for K-Fold. Test indices are in orange, train indices are in blue, class row represents value of output label for each index.

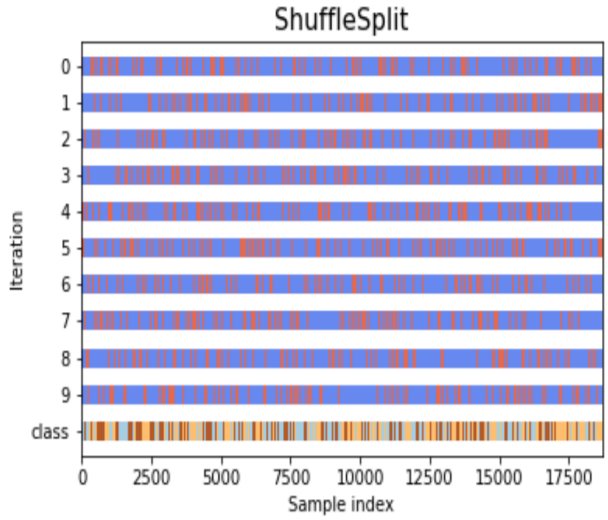

Fig. 29. Example train and test indices for Shuffle-Split.

  - number of neighbors
  - power parameter for the Minkowski distance
- SVM
  - regularization parameter
  - kernel coefficient for radial basis function
- Random Forest
  - maximum depth of the tree
  - minimum number of samples required to split an internal node

F. Performance Metrics

In order to assess each supervised learning method, a set of performance metrics will be used.

- Confusion Matrix
  - Summarizes prediction results from all classes. It return a matrix showing the number of correct classifications and number of misclassifications.
- Accuracy Score
  - The fraction of correctly classified labels.
- F1-Score
  - The weighted average between precision and recall. As a result, it captures both accuracy and number of correctly predicted labels.
- Matthews Correlation Coefficient (MCC)
  - A balanced metric taking into account TP, TN, FP, and FN.
- Area Under the Curve (AUC)
  - The ratio between true positive rate and false positive rate.

G. Results

1) K-Nearest Neighbors: Grid search was used for hyperparameter optimization. Fig. 30-Fig. 31 show mean test score compared to $n\_neighbors$ and p. Fig. 32-Fig. 33 show resulting training and testing performance metrics using $n\_neighbors = 26$ and $p = 2$.

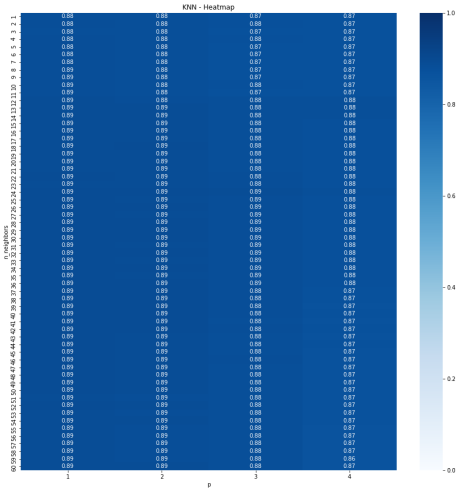

Fig. 30. KNN - heatmap showing $n\_neighbors$ vs p vs mean test score.

2) Support Vector Machine: Grid search was used for hyperparameter optimization. Fig. 34-Fig. 35 show mean test score compared to C and γ. Fig. 36-Fig. 37 show resulting training and testing performance metrics using $C = 10$ and $\gamma = 0.1$.

3) Random Forest: Grid search was used for hyperparameter optimization. Fig. 38-Fig. 39 show mean test score compared to max_depth and $min\_samples\_leaf$. Fig. 40-Fig. 41 show resulting training and testing performance metrics using $max\_depth = 18$ and $min\_samples\_leaf = 14$.

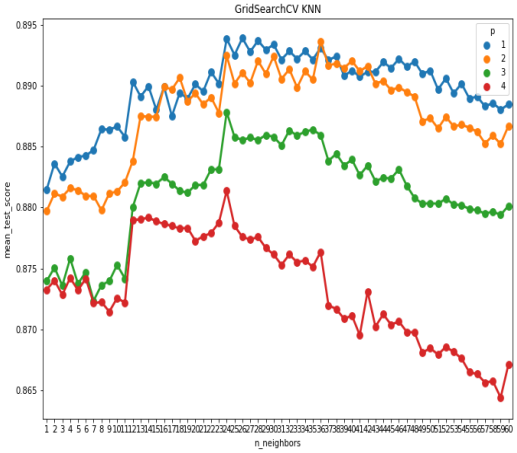

Fig. 31. KNN - plot showing $n\_neighbors$ vs p vs mean test score.

	0	1	2
0	6193	10	67
1	47	6100	63
2	22	3	6185

```
                                  Accuracy Score: 0.9887
                                      F1-Score: 0.9887
Matthews Correlation Coefficient (MCC): 0.9830
                    Area Under the Curve (AUC): 0.9830
```

Fig. 32. KNN - training performance metrics.

	0	1	2
0	2603	22	45
1	185	2249	146
2	197	25	2538

```
                                  Accuracy Score: 0.9226
                                      F1-Score: 0.9226
Matthews Correlation Coefficient (MCC): 0.8857
                    Area Under the Curve (AUC): 0.8857
```

Fig. 33. KNN - testing performance metrics.

VII. Discussion

All three methods tested showed great promise on this small data set, Fig. 42. The process of tuning KNN's hyperparameters was the simplest since it's performance dependent highly on only one feature, $n\_neighbors$, and less so on p with both SVM and Random Forest requiring more hyperparameter tuning. KNN and Random Forest were computationally much more efficient when training and predicting compared to SVM. However, SVM did have the best test results. For all three methods, testing set metrics performed 3-7% below their training counterparts, indicating some overfitting may have occurred. To overcome overfitting, KNN could be tested with a greater number of neighbors so that it isn't susceptible to local

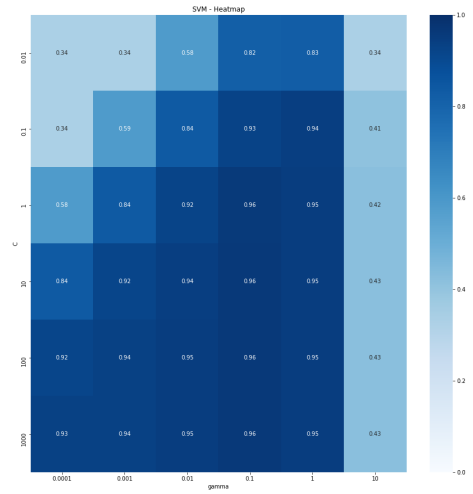

Fig. 34. SVM - heatmap showing C vs γ vs mean test score.

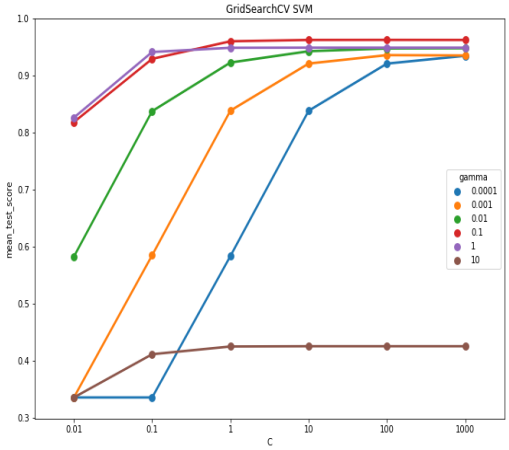

Fig. 35. SVM - plot showing C vs γ vs mean test score.

	0	1	2
0	6270	0	0
1	0	6210	0
2	6	0	6204

```
                                  Accuracy Score: 0.9997
                                      F1-Score: 0.9997
Matthews Correlation Coefficient (MCC): 0.9995
                    Area Under the Curve (AUC): 0.9995
```

Fig. 36. SVM - training performance metrics.

hotspots, with SVM enlarging γ may lead to higher accuracy using the same Radial Basis Function kerneland with Random Forests decreasing the depth of each tree and pruning could

	0	1	2
0	2603	33	34
1	36	2526	18
2	122	36	2602

Accuracy Score: 0.9652
F1-Score: 0.9653
Matthews Correlation Coefficient (MCC): 0.9480
Area Under the Curve (AUC): 0.9480

Fig. 37. SVM - testing performance metrics.

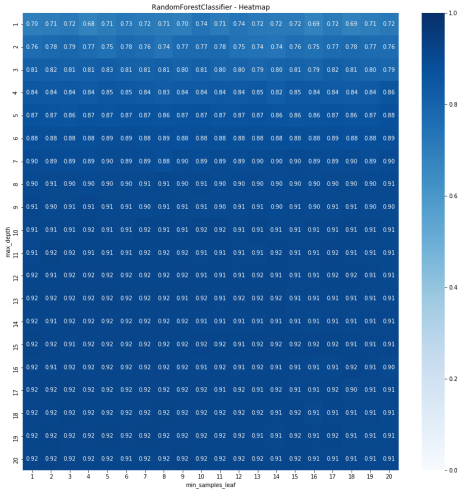

Fig. 38. Random Forest - heatmap showing max_depth vs $min\_samples\_leaf$ vs mean test score.

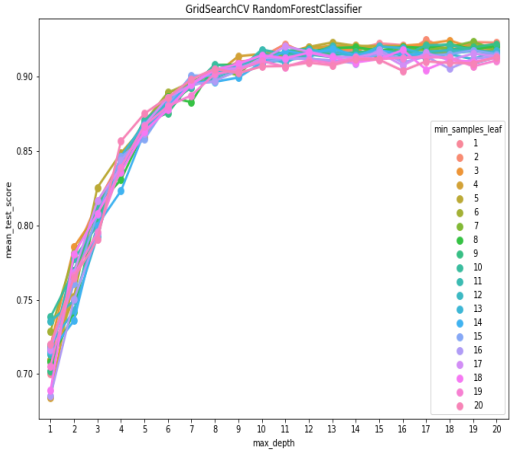

Fig. 39. Random Forest - plot showing max_depth vs $min\_samples\_leaf$ vs mean test score.

help.

	0	1	2
0	6270	0	0
1	0	6203	7
2	0	0	6210

Accuracy Score: 0.9996
F1-Score: 0.9996
Matthews Correlation Coefficient (MCC): 0.9994
Area Under the Curve (AUC): 0.9994

Fig. 40. Random Forest - training performance metrics.

	0	1	2
0	2519	57	94
1	93	2446	41
2	198	98	2464

Accuracy Score: 0.9275
F1-Score: 0.9277
Matthews Correlation Coefficient (MCC): 0.8917
Area Under the Curve (AUC): 0.8917

Fig. 41. Random Forest - testing performance metrics.

Metric (test set)	KNN	SVM	Random Forest
Accuracy	0.9226	0.9652	0.9220
F1-Score	0.9226	0.9653	0.9222
MCC	0.8857	0.9480	0.8838
AUC	0.8857	0.9480	0.8838

Fig. 42. Summary of testing results.

VIII. Future Work

Using this project as a stepping stone I would like to look into ensemble learning where different classifiers are added together similar to a Random Forest. Additionally using a convolutional neural network for feature extraction or prediction could be very revealing. Building an autoencoder may be able to provide useful features as well.

IX. Appendix

A. Cross-Validation Results

Fig. 43-Fig. 48.

B. Decision Boundary Visualization

t-SNE (t-Distributed Stochastic Neighbor Embedding) is an unsupervised method used to visualize high dimensional data and decision boundaries. Using voronoi tessellation, such decision regions can be visualised. Fig. 49-Fig. 51 visualize high dimensional decision boundaries using t-SNE on a subset of the training and testing sets due to computation limitations.

C. Sklearn

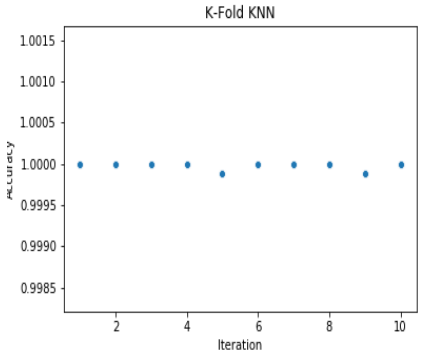

Fig. 43. KNN K-Fold.

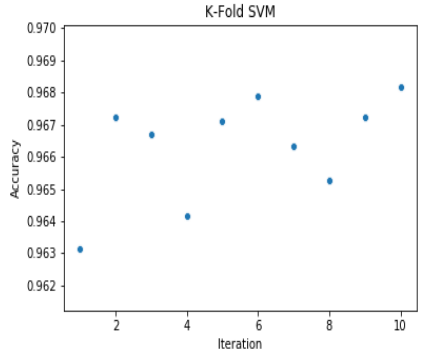

Fig. 44. SVM K-Fold.

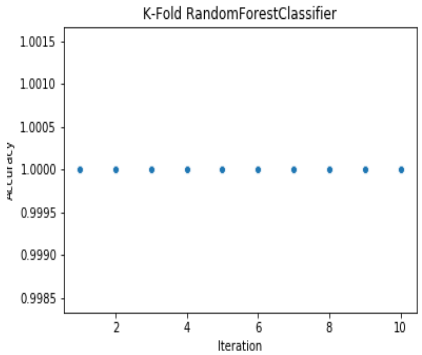

Fig. 45. Random Forest K-Fold.

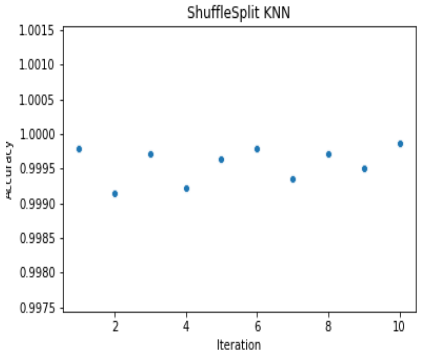

Fig. 46. KNN Shuffle-Split.

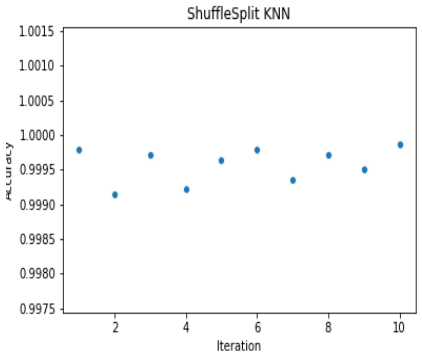

Fig. 47. SVM Shuffle-Split.

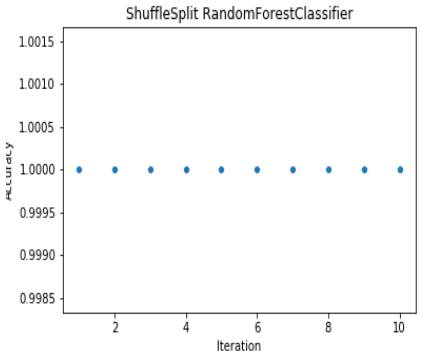

Fig. 48. Random Forest Shuffle-Split.

```
# K-Fold Cross-Validation
from sklearn.model_selection import KFold

# Shuffle Split Cross-Validation
from sklearn.model_selection import ShuffleSplit

# Exhaustive Grid Search
from sklearn.model_selection import GridSearchCV

# Performance Metrics
from sklearn.metrics import accuracy_score,
                            auc,
                            confusion_matrix,
                            f1_score,
                                matthews_corrcoef

# KNN
from sklearn.neighbors import KNeighborsClassifier

# SVM
from sklearn.svm import SVC

# Random Forest
from sklearn.ensemble import RandomForestClassifier
```

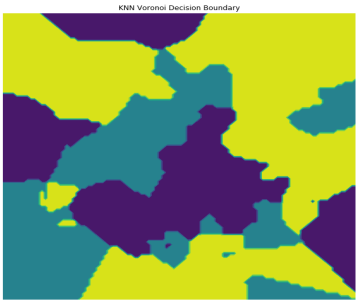

Fig. 49. KNN High Dimensional Decision Boundary.

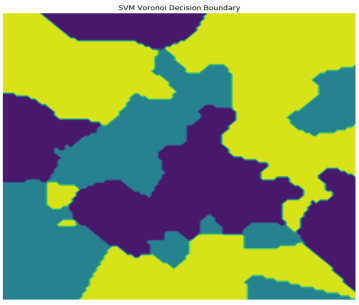

Fig. 50. SVM High Dimensional Decision Boundary.

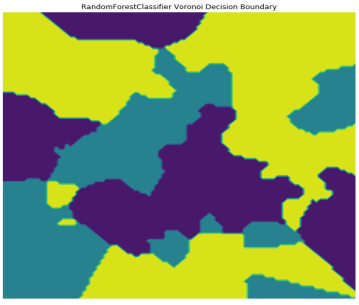

Fig. 51. Random Forest High Dimensional Decision Boundary.

REFERENCES

[1] S. Kothari, J. H. Phan, R. A. Moffitt, T. H. Stokes, S. E. Hassberger, Q. Chaudry, et al., "Automatic batch-invariant color segmentation of histological cancer images," Conf Proc IEEE Int Symp Biomed Imaging, ISBI, pp. 657-660, 2011.

[2] S. Kothari, J. H. Phan, A. N. Young, and M. D. Wang, "Histological image feature mining reveals emergent diagnostic properties for renal cancer," Conf Proc IEEE Bioinform Biomed, BIBM, 2011.

[3] P. Huang and C. Lee, "Automatic Classification for Pathological Prostate Images Based on Fractal Analysis," in IEEE Transactions on Medical Imaging, vol. 28, no. 7, pp. 1037-1050, July 2009.

[4] J. Barker, A. Hoogi, A. Depeursinge, D. L. Rubin, "Automated classification of brain tumor type in whole-slide digital pathology images using local representative tiles," Medical Image Analysis, vol. 30, pp. 60-71, 2016.

[5] H. Wang, A.C. Roa et al, "Mitosis detection in breast cancer pathology images by combining handcrafted and convolutional neural network features". Journal of Medical Imaging, 1(3), 034003 (2014).

[6] L. Boucheron, "Object-and spatial-level quantitative analysis of multispectral histopathology images for detection and characterization of cancer," PhD thesis, University of California, Santa Barbara, 2008.

[7] A. Bosch, A. Zisserman and X. Munoz, "Image Classification using Random Forests and Ferns," 2007 IEEE 11th International Conference on Computer Vision, Rio de Janeiro, 2007, pp. 1-8.

[8] S. Zhang, X. Li, M. Zong, X. Zhu and R. Wang, "Efficient kNN Classification With Different Numbers of Nearest Neighbors," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 5, pp. 1774-1785, May 2018.

[9] Manas Jain, Shruthi Narayan, Pratibha Balaji, Bharath K P, Abhijit Bhowmick, Karthik R, "Speech Emotion Recognition using Support Vector Machine," 2020, [http://arxiv.org/abs/2002.07590 arXiv:2002.07590].

[10] S. Wang et al., "Multiple Sclerosis Detection Based on Biorthogonal Wavelet Transform, RBF Kernel Principal Component Analysis, and Logistic Regression," in IEEE Access, vol. 4, pp. 7567-7576, 2016.

[11] D. Magee, D. Treanor, D. Crellin, M. Shires, K. Smith, K. Mohee, et al., "Colour Normalisation in Digital Histopathology Images," Proc. Optical Tissue Image analysis in Microscopy, Histopathology and Endoscopy (MICCAI Workshop), pp. 100-111, 2009

[12] E. Reinhard, M. Adhikhmin, B. Gooch and P. Shirley, "Color transfer between images", IEEE Computer Graphics and Applications, vol. 21, no. 4, pp. 34-41, 2001. Available: 10.1109/38.946629.

[13] "GLCM Texture Feature." GLCM Texture Feature, 1 Aug. 2019.

[14] J. Hale, "Scale, Standardize, or Normalize with Scikit-Learn", Medium, 2019. [Online].