McGill University

Faculty of Engineering

Department of Electrical and Computer Engineering

ECSE 415 Introduction to Computer Vision Final Project: Segmentation in Medical Imaging

Final Report Group 8

Cesar Arnouk - cesar.arnouk@mail.mcgill.ca - 260847683

Andrew Geday - andrew.geday@mail.mcgill.ca - 260834063

Rodolphe Baladi - rodolphe.baladi@mail.mcgill.ca - 260829029

Ryan Tabbara - ryan.tabbara@mail.mcgill.ca - 260835629

Nabil Abdallah - nabil.abdallah@mail.mcgill.ca - 260898733

Karim Elgammal - karim.elgammal@mail.mcgill.ca - 260920556

Course given by Professor Tal Arbel - tal.arbel@mcgill.ca

Table of Contents:

I. Introduction

Computer vision has come to have great importance in the realm of medical imaging. Computer vision refers to the ability to allow computers to analyze image data and extract meaningful information. As technology continues to advance, computer vision has played an important role in advancing the field of medicine by gaining the ability to analyze medical images and make diagnoses. The project aims to explore different approaches for image segmentation in the context of medical and tumor classification. The project begins by exploring unsupervised and supervised segmentation to be able to extract certain features in a medical image. We then move on to build a classification framework to identify tumors. The framework is trained on a set of data from patient images and then tested to verify our accuracy.

The significance of the project is attributed to its potential to have real-world applications. Computer vision can help improve diagnosis by allowing medical professionals to examine a large set of data quickly and accurately to identify the presence of tumors. Moreover, computer vision can help in the early detection of tumors which may be overlooked, thus helping reduce human error. This project explores how these benefits can be harnessed through the application of computer vision in medical imaging.

II. Dataset

The dataset provided for this project includes rgb images (input to be processed). Each image has its corresponding mask (ground truth mask), which indicates the location of nuclei to be detected. Finally, each image is also matched with a csv annotations file that contains labels and location coordinates of the nuclei in the image. An example of the dataset in displayed in the figure below.

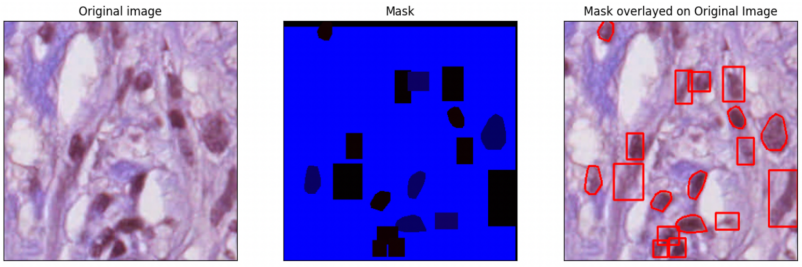

Figure 1: Original, maks, and mask overlay images

We split the data between training patients and validation patients. Each patient is identified by their unique ID. 80% of patient ids are placed in the training split (i.e. 99 unique patients) and 20% of patient ids are placed in the validation split (i.e. 25 unique patients). No other information is recorded in the training and validation splits files, because images, masks, and annotations are retrievable based on the patient ids.

III. Segmentation

A. Unsupervised Segmentation

1. Method

In order to explore the most performant unsupervised segmentation strategy in this context, we performed three main experiments. First, we performed KMean with RGB images. For different values of K, we performed KMean segmentation and computed DICE coefficients, based on the formula below. The DICE

coefficient is computed by taking twice the intersection between the segmented image and the actual mask, and dividing it by the sum of the sizes of the two sets, which results in a value between 0 and 1, with 1 indicating perfect overlap between the two sets. Based on these DICE coefficients, we deduced the value of K that optimizes RGB KMean segmentation.

$$\frac{2TP}{TP+TN+FP+FN}$$

Figure 2: DICE coefficient formula

The second experiment we performed was KMean with grayscale images. We repeated the process explained above, for RGB images. For different values of K, we performed KMean segmentation on grayscale images, then computed DICE coefficients. We then compared the grayscale results to the RGB results. Finally, the last experiment we performed was Gaussian Mixture Modeling (GMM) Segmentation on the RGB images. We started by optimizing the n_components parameter based on GMM DICE values. Then, we compared the best GMM results with the best KMean Segmentation result.

2. KMean on RGB Images

We performed KMean Segmentation on the original image for values of K ranging from 2 to 8. For each segmented image, we calculated a mask that contours features in the segmentation, and finally, we computed the DICE coefficients, as described above. Sample segmentation results for K=2 and K=4 are displayed below. Segmentation results for K=3 and K=5 can be found in the code file.

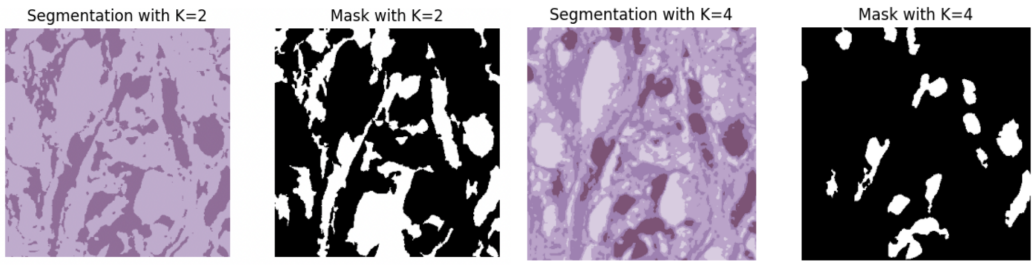

Figure 3: Sample RGB KMean Segmentation images for K=2 and K=4

Once we obtained the metrics of DICE coefficients for each value of the K parameter, we plotted the DICE scores with respect to K values for RGB KMean Segmentation. As a result, we observed that K=4 maximizes the DICE score, and therefore is the optimal value for the K parameter in this context.

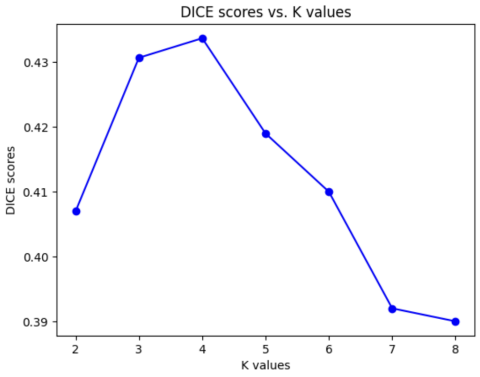

Figure 4: DICE coefficients vs K values for RGM KMean plot

For the optimized value of K is 4, we can also see that, visually, the expected box annotations match the segmented nuclei.

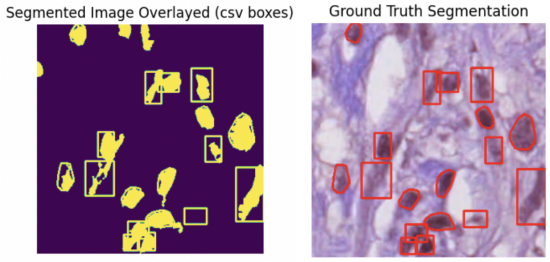

Figure 5: Optimized KMean (K=4) Segmentation and Ground Truth boxed overlaid on original image.

3. KMean on Grayscale Images

Following the same procedure as with RGB images, we performed KMean Segmentation on the grayscale images for values of K ranging from 2 to 8. For each segmented image, we calculated a mask that contours features in the segmentation, and finally, we computed the DICE coefficients. Sample segmentation results for K=2 and K=4 are displayed below.

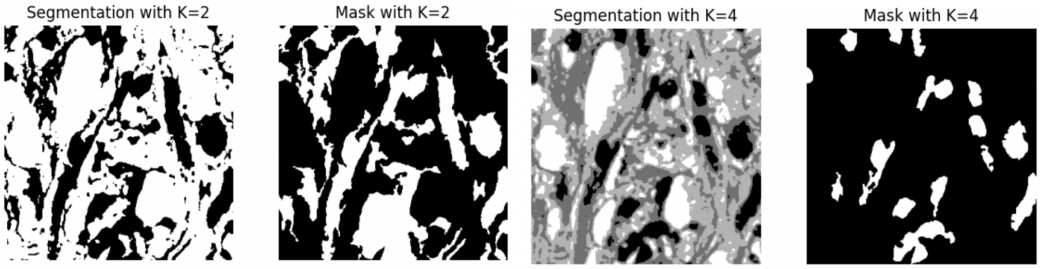

Figure 6: Sample Graysacle KMean Segmentation images for K=2 and K=4

Once we obtained the metrics of DICE coefficients for each value of the K parameter, we compared the DICE values results of grayscale images and RGB images in the plot below. As a result, we observed that we obtain very similar results between color and grayscale images. Color images seem to give slightly better results, but the best result is for grayscale K=4.

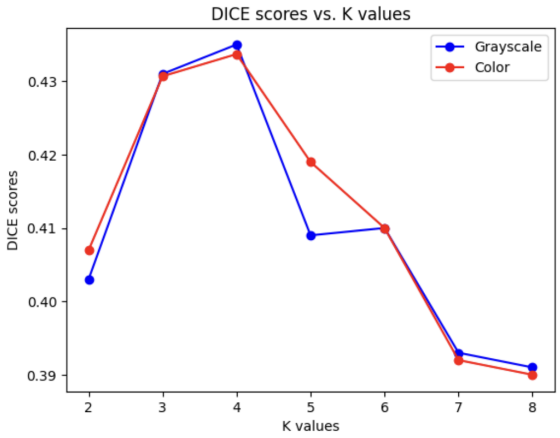

Figure 7: plot of DICE coefficients vs KMean K parameter for Grayscale and RGB images

4. Gaussian Mixture Modeling

Using the GMM Segmentation strategy, we started by optimizing the DICE values with respect to the n_components parameter. We obtained that GMM optimizes DICE with C=3. With this value, we then compared GMM results to KMean segmentation results, as shown in the figure below.

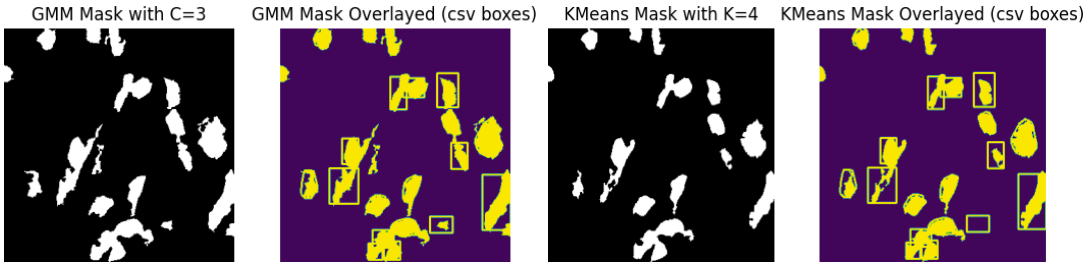

Figure 8: optimized GMM and KMean Segementations and results overlaid with ground truth boxes

In terms of DICE coefficients, GMM (C=3) obtained DICE=0.43050079871774877; whereas KMean (K=4) obtained DICE: 0.4335649318168128. The KMean result is therefore slightly higher, hence KMean Segmentation is slightly more performant than GMM Segmentation. Visually, we can see in the figure above that the GMM Segmentation preserves too many features that are not nuclei. These features are segmented-out by the KMean strategy.

B. Supervised Segmentation

1. Method

The supervised approach we are using is a Random Forest Classifier, which involves training a model on a labeled dataset (of 198 images, 2 per training patient) to predict class labels for the new data. Random Forest is effective in handling complex datasets with high-dimensional feature spaces because it is capable of capturing non-linear and interactive relationships between features. That is due to the fact that it randomly selects a subset of features at each node of each decision tree, making it less prone to overfitting and more robust to noise and missing data. Another advantage of Random Forest is its ability to provide insights into the relative importance of different features in the classification process. Random Forest is particularly suitable for medical image analysis because it can handle multi-dimensional data and has been shown to be effective in a variety of segmentation tasks, such as tumor detection and classification, brain tissue segmentation, and lesion detection. However, it is important to note that Random Forest is not without its limitations. It may struggle with imbalanced datasets, as well as having certain limitations due to a shortage in input predictors.

2. Results and Discussion

We performed the Random Forest Segmentation first by training the model on a labeled dataset (called training images), where we chose to have 2 images per patient in the training split for a total of 198 images. After training the model, the objective is to get the accuracy gotten from our classifier on the predicted mask as compared to the actual mask, and this accuracy will be based on pixel comparison.

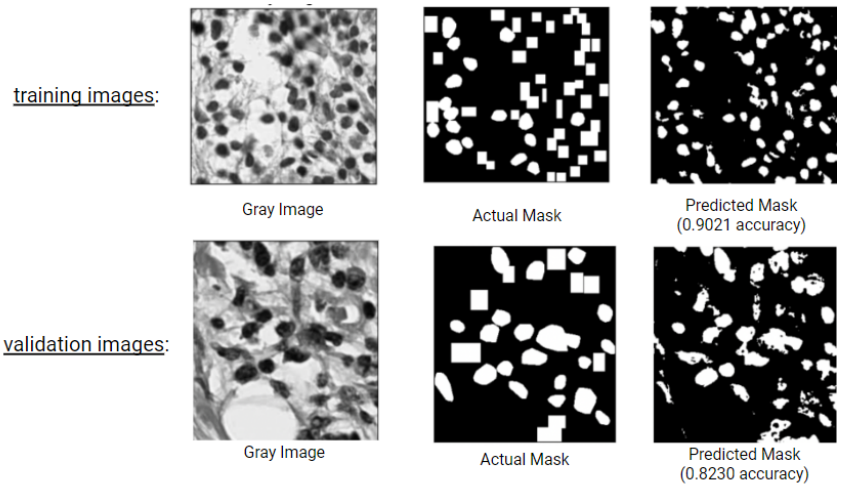

Figure 9: Sample Random Forest prediction on training and validation images

As shown in the figure above, we performed prediction of the Random Forest Classifier on some sample images. First, we selected random images from the training images, and were able to achieve an accuracy of 0.9021 on the prediction of the mask. However, when choosing a random image from the validation dataset (unseen images), we are able to achieve a lower accuracy of 0.823. This is expected since the classifier is trained on the training images and the image is basically "already seen", whereas the classifier has not seen the validation images before the prediction. More importantly, we computed the average accuracy for a set of 50 trained images which was at 0.8731 of the predicted mask, as compared to the set of 50 validation (unseen) images with an accuracy of 0.732 on average. These results are expected since the model performs well on the training data but is less accurate when applied to unseen validation data, indicating a small issue of overfitting.

C. More Discussion on Segmentation Results

After successfully implementing both unsupervised and supervised segmentation pipelines, we decided to compare the accuracy of both approaches. In order to do this, we randomly picked a sample of 30 images that were not part of the training set, computed the DICE coefficient for each image, and then computed the average value of the DICE coefficient of the 30 images. We did so for both the KMean and Random Forest segmentation methods. We obtained an average DICE Coefficient of 0.4281 when using KMean segmentation compared to 0.3719 when using Random Forest segmentation. Additionally, these results correlate well with the images obtained from both segmentation approaches, shown below.

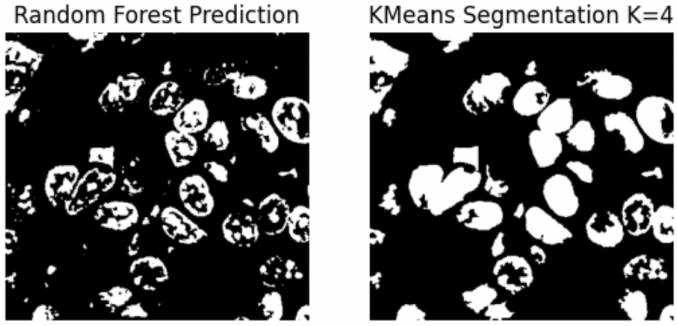

Figure 10: Sample Random Forrest prediction and KMean Segmentation results on random images

When visually comparing the two results of one of the random images used for computing the DICE coefficient in the bottom rows, the KMeans segmentation provided more accurate results that were more closely related to the actual mask (less noise and more definition than when using Random Forest). It should be noted that although these results are based on only one of the images, the other images had similar results. The point of showcasing the visual results (images) was to demonstrate how a higher DICE coefficient translates to more accurate results.

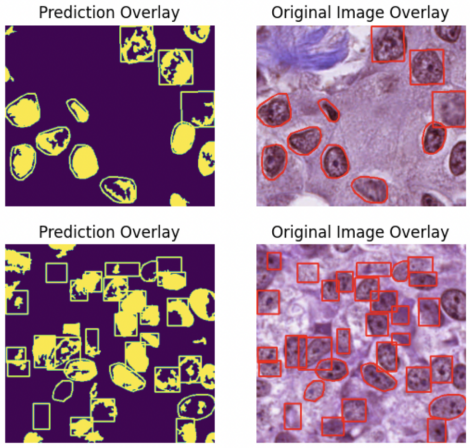

Figure 11: Prediction vs. Original images overlaid with ground truth boxes

Additionally, after determining that we were obtaining superior results using KMeans segmentation, we decided to find which physical features in the nuclei made it easier for our segmentation pipeline to detect them. As seen in the image above, we concluded that nuclei groups of larger, darker (meaning more contrast with the background) and the ones that have fewer overlapping nuclei, were segmented most successfully. The results make perfect sense because the larger a nucleus is, the more pixels there are to be grouped and countered by segmentation. Also, darker nuclei are more distinct leading to easier segmentation. Finally, when there is less clustering, the contours are more defined leading to superior segmentation results.

IV. Tumor Detection Classification Framework

A. Method

We begin by preprocessing the images to be used in our prediction model. We first convert the input images and masks to greyscale. We then multiply the mask and the original image to obtain the regions of interest. In this step, it is important to note we had to resize the image and mask such that they are the same size. We proceed by removing the background noise in the resultant image.

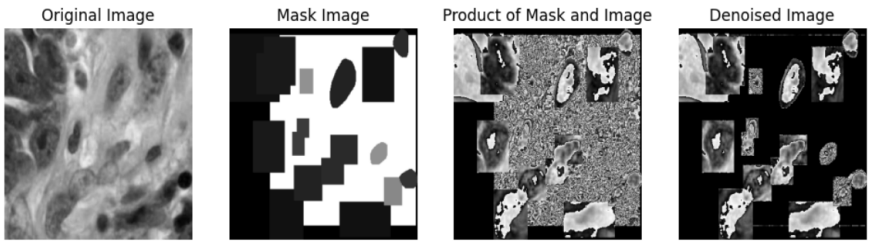

Figure 12: Steps for preparing input data for detection

Now in order to create our classification framework; we first had to find a feature extraction model that would optimize our search for tumors. According to Alhindi et al. in Comparing LBP, HOG and Deep Features for Classification of Histopathology Images [1], a combination of Local Binary Pattern (LBP) with an SVM attained the best results, when it came to dissecting cell features. LBP is a texture descriptor used in computer vision to characterize local texture patterns in images. It is a simple yet powerful method that captures the structural information of an image by analyzing the relationships between the gray levels of a pixel and its neighbours. And so, we opted to use it within our feature extraction model.

Upon experimentation with using the LBP extraction model alone, we found that we could increase the variables that are included in creating a feature; allowing us to capture more 'detail' from the image. And so, we decided to extract additional features of; intensity (mean intensity, standard deviation, skewness, and kurtosis) and shape (area, perimeter, and compactness). We then concatenated these features to arrive at our unified feature list used to train the SVM. It is also noted that our implementation utilizes the Radial Basis Function (RBF) as a kernel in our SVM. This is due to its ability to transform input data into a higher-dimensional space. By doing so, it simplifies the process of finding a linear decision boundary between different classes [2].

Next, in the Feature Extraction step, we extract the bounding box coordinates and labels from the accompanying CSV file for each image. We then extracted our concatenated features from each tumor blob and created a dataframe of said features along with their classification. Then, during the Data Preprocessing phase, we replace infinity or NaN values in the features dataframe with the mean or median of the corresponding feature and scale the features using StandardScaler to normalize the data. We then trained our SVM model to predict tumor/non-tumor labels for new, unseen images, based on our training set. We then ran our testing set through the predict function of the SVM to evaluate the performance of our model. Our predictions are then outputted as bounding boxes which are overlaid on the images, with red rectangles for tumors and blue rectangles for non-tumors.

B. Results and Discussion

1. Performance Metrics

In order to evaluate the performance metric of our detection model, we chose to create a classification report using Sklearn. By doing so we obtain the following results shown in *figure 13.*

Validation set classification report:

	precision	recall	f1-score	support
0	0.89	0.94	0.91	2241
1	0.77	0.64	0.70	728
accuracy			0.87	2969
macro avg	0.83	0.79	0.81	2969
weighted avg	0.86	0.87	0.86	2969

Test set classification report:

	precision	recall	f1-score	support
0	0.65	0.90	0.76	436
1	0.69	0.32	0.44	307
accuracy			0.66	743
macro avg	0.67	0.61	0.60	743
weighted avg	0.67	0.66	0.62	743

Figure 13: Classification report to analyze performance when images have been processed

The '0' in the classifier report represents non-tumors, while the '1' represents tumors. From the validation set classification report, we obtain information regarding precision, recall, F1-score, and support. Precision is a measure of the classifier's exactness, thus telling us what percentage of all instances classified as positive was actually correct. Recall measures the classifier's completeness, in other words, its ability to correctly identify all positive instances when predicting if a blob is a tumor. This can be thought of as what percentage of all instances classified as positive were actually positive. The F1-score is a weighted harmonic mean of precision and recall. A score closer to 1 indicates high accuracy for the classifier. Support is the number of occurrences in the dataset [3]. The accuracy F1-score allows us to measure how well our classifier functions.

The classifier report is done on both the validation set and the test set. For the validation set, we find that our classifier is better at identifying non-tumors than it is at identifying tumors. This is deduced by comparing the precision values for 0 and 1. Moreover, we see that we achieve an accuracy of 87%, indicating that our classifier can be considered reliable for finding tumors in medical images. For the test set, we now find that our precision values for detecting non-tumors (0) and tumors (1) have significantly decreased to 65% and 69%, respectively. Moreover, we now find that our accuracy is at 66%, again significantly less than with the validation set. Our below-ideal results may be a result of not having preprocessed the image enough. Moreover, experimenting with different computer vision features such as HoG and SIFT may contribute to improving the accuracy of our model.

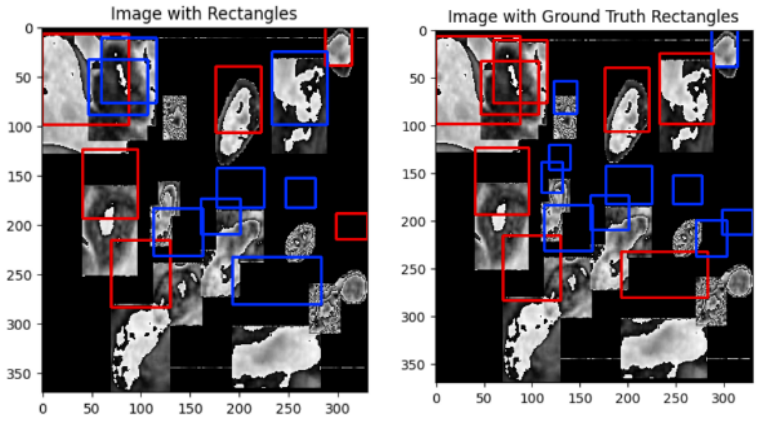

Figure 14: Results from prediction model compared to ground truth

2. Root Mean Squared Error

The root mean squared error (RMSE) is used as a metric to evaluate the accuracy of prediction models. It helps gain information on how large the difference is between the predicted values and the actual values. Thus, we would ideally want to obtain a smaller RMSE value. To calculate RMSE, we use the following formula below:

$$RMSE = sqrt(1/n * \sum(predicted - actual)^2$$

By following this formula we obtain a root mean squared error for the validation set of 0.366, seen in *figure 15.* This shows that our prediction model is relatively accurate as our value is less than 0.5.

```
Root Mean Squared Error: 0.3661309455867779
```

Figure 15: RMSE Result from model

3. Performance When Using ‘original image’

To test the performance of our model when we use only the original image, we remove all pre-processing done to the image. We then run our model and print the classification report seen in *figure 16*.

Validation set classification report:

	precision	recall	f1-score	support
0	0.76	0.99	0.86	1720
1	0.82	0.08	0.14	598
accuracy			0.76	2318
macro avg	0.79	0.54	0.50	2318
weighted avg	0.77	0.76	0.67	2318

Test set classification report:

	precision	recall	f1-score	support
0	0.60	0.98	0.75	262
1	0.71	0.05	0.10	182
accuracy			0.60	444
macro avg	0.66	0.52	0.42	444
weighted avg	0.65	0.60	0.48	444

Figure 16: Classification report to analyze performance when images have not been processed

As expected, our model performs worse if we use the 'original image'. This is because, without the blobs, it is more difficult for our model to identify tumors. This can be seen in both the validation set and the test set, where almost all the precision values for detecting tumors and non-tumors are significantly smaller. Moreover, we can see that our accuracy for the validation set has dropped from 87% to 76% and the accuracy for our test set has dropped from 66% to 60%.

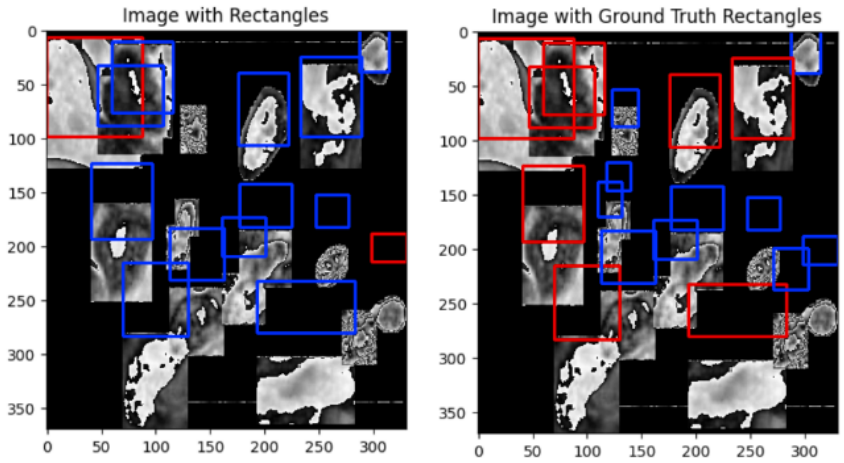

Figure 17: Results from prediction model compared to ground truth

By comparing our model predictions to the image with the ground truth rectangles we can clearly view the errors from our prediction model.

4. Segregating Different Pathological Structures

To identify which pathological structures are easiest to segregate from tumors, we need to find the fail rate associated with each structure. To do so, we first find the number of occurrences of each structure in the CSV files. For each tumor detected by our model, we check that it is indeed a tumor. If it is not a tumor, we identify what the structure is such that our program keeps track of each time that given structure is labelled as a tumor. We run this on our training set to obtain the following results in *figure 15*.

```
[ 1  0  0 31 59  0  0 15  0 30  4]
For Cell Type: apoptotic_body, Fail Rate: 50.0%
For Cell Type: ductal_epithelium, Fail Rate: 0.0%
For Cell Type: eosinophil, Fail Rate: 0.0%
For Cell Type: fibroblast, Fail Rate: 30.392156862745097%
For Cell Type: lymphocyte, Fail Rate: 36.19631901840491%
For Cell Type: macrophage, Fail Rate: 0.0%
For Cell Type: mitotic_figure, Fail Rate: 0.0%
For Cell Type: plasma_cell, Fail Rate: 57.692307692307686%
For Cell Type: tumor, Fail Rate: 0.0%
For Cell Type: unlabeled, Fail Rate: 26.08695652173913%
For Cell Type: vascular_endothelium, Fail Rate: 66.66666666666666%
```

Figure 18: Results from studying models ability to segregate different pathological structures

The array at the top of *figure 15* represents the number of times each pathological structure was identified incorrectly as a tumor. We then print out the fail rate of each pathological structure present in the training set, indicating the percentage likelihood of incorrectly identifying the structure as a tumor. We find that ductal epithelium, eosinophil, macrophage, and mitotic_figure are the easiest to segregate from tumors as they have a fail rate of 0%. This means that in no occasion were these structures incorrectly labelled as tumors. This may be because such structures contain features that our model is very good at differentiating from those of tumors. We can also see that vascular_endothelium is predicted to be a tumor 66% of the time, from which we can deduce that they have similar features with tumors that our model fails to differentiate.

V. Conclusion

In conclusion, the project has explored the potential of computer vision in medical imaging for tumor classification using different segmentation techniques. The project started by exploring unsupervised and supervised segmentation to extract certain features in a medical image and built a classification framework to identify tumors. For the image preprocessing, multiple strategies were employed: KMean segmentation, GMM segmentation, and Random Forest prediction. The different approaches were compared based on DICE scores, with the objective of optimizing these values. After completing the experiments, it was determined that the best results were obtained for K=4 in KMean segmentation. This approach allows for accurate segmentation of physical features, especially for groups of larger, darker, and disjoint nuclei. For part 5 of the project, we were able to successfully implement a prediction model to detect and count tumors. We were able to display our results by displaying medical test images and comparing what our model found to be tumors with what we actually knew were tumors. We evaluated the accuracy of our model by creating a validation classification report. We then further analyzed the results to identify which pathological structures were easiest to segregate from tumors. Overall, the project helped us better understand the significant impact computer vision can have in the field of medicine.

VI. Works Cited

[1] T. J. Alhindi, S. Kalra, K. H. Ng, A. Afrin, and H. R. Tizhoosh, "Comparing LBP, HOG and Deep Features for Classification of Histopathology Images,"

[2] C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning, vol. 20, no. 3, pp. 273-297, 1995. DOI: 10.1007/BF00994018.

[3]*Classification report*. Classification Report - Yellowbrick v1.5 documentation. (n.d.). Retrieved from https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html#:~:text=percent%20was%20correct%3F%E2%80%9D-,recall,true%20positives%20and%20false%20negatives.