Breast cancer severity assessment Report

Adil Ismagambetov
School of Engineering and Digital Sciences
Nazarbayev University
Astana, Kazakhstan
adil.ismagambetov@nu.edu.kz

I. Introduction

This report presents the pipeline for EE417 assignment on "Breast cancer severity assessment". First it will start from the set of morphological operators used to separate cells from the background of given image, then it will demonstrate the variables that were used as a feature vectors of cell, how the Random Forest hyperparameters were tuned and overall performance of presented model on validation and test datasets.

II. Dataset

The first task was to color the mitosis cases in a given frame. Corresponding ".csv" files were provided along with each image. In order to accomplish this, two helper functions were written. The first one is "get_coordinates" which receives a path to ".csv" file and parses the coordinates into a simple list, which is then outputted. The second function is "annotate_img" which receives an image and a coordinates list. It goes through this list, sets the corresponding coordinates RGB values to yellow, and then outputs this image.

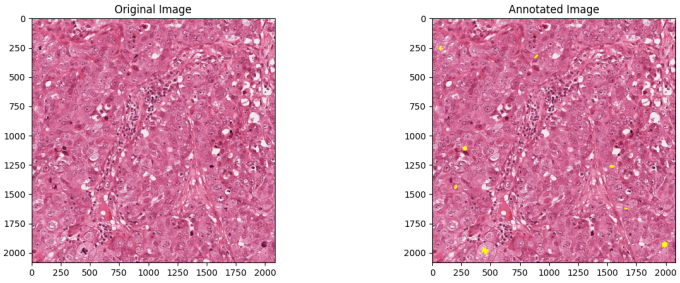

Fig. 1. Original and "Annotated" Image.

The next task was to come up with a set of morphological operators that separate cells from the background. Before applying morphological operators, the image was set to grayscale, and then to binary representation. After that, this binary image was inverted because originally cells were black. For the morphological gradient to work, cells have to have high values compared to neighboring pixels. Additionally, the "clean image" should have its cells as white (255, 255, 255) regions. Two functions with different operators were introduced. The first one - "clean_image" which applies 2 iterations of erosion with 7 by 7 structuring element and then 2 iterations of gradient with 9 by 9 SE. The second one - "clean_image_2" - applies 1 iteration of opening with 9 by 9 SE followed by 1 iteration of dilation with 7 by 7 SE.

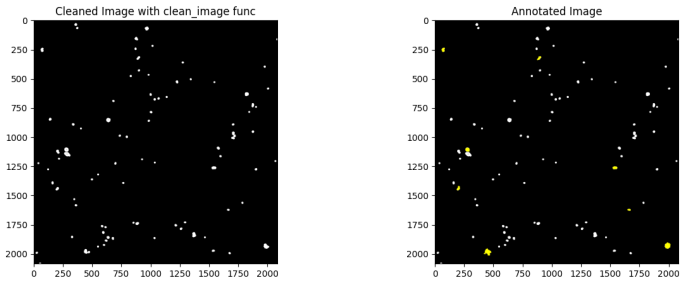

Fig. 2. How clean_image function separates cells from the background.

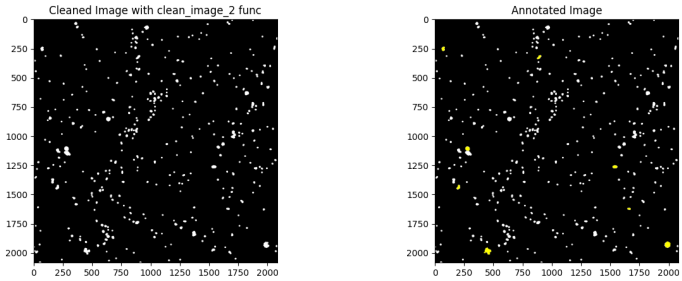

Fig. 3. How clean_image_2 function separates cells from the background.

Both functions were tested on validation data and it was observed that clean_image function, which uses morphological gradient, gives better IoU and F1-score metrics. After that, cells were extracted as connected components using built-in open-cv2 function "cv2.connectedComponentsWithStats" with connectivity of 8. Then the function "find_label" was implemented. It goes through the connected components and checks whether there are some coordinates from ".csv" file within the borders of a connected component. If it finds them, it assigns them label - 1, or mitosis case. Otherwise value 0 is given to a cell.

III. Features Vector

Once the connected components were extracted, they have to be described before feeding them in machine learning model. It was decided to use both bag of visual words and local features. In the beginning, as local features only width, height, area, average red, green, and blue channels for a cell

were used. The number of visual words was debugged along with Grid Search for Random Forest model using validation data. For this case, the best performance was recorded as 0.233 IoU score with feature vector of 80 visual words and 6 local features described above. Then it was decided to add 3 shape features (aspect ratio, compactness, and circularity) and perform Grid Search again. IoU metric was increased to 0.257 with 80 visual words and 9 local features. Other local features were tested, such as HSV and RGB histograms instead of average RGB channels. However, their performance was worse and corresponding scores are recorded in the Table. Apart from Random Forest, XGBoost model was also tested. Its performance turned out to be worse with the highest IoU score of 0.21. It is also worth noting that all the data was standardized before using it for training, validation and testing.

IV. Overall Pipeline

This is how the overall pipeline looks like: A program loops through the list that contains paths for all training images. It takes each image, gets its corresponding coordinates using "get_coordinates" function, "cleans" the image with "clean_image" method, and extracts connected components. All the connected components, or cells, are then saved in separate list along with their corresponding label. These cells are then iterated through each loop and "bag_of_visual_words" function is applied to describe them, as well as that their local features are saved. In the end, every image from dataset has its features vector as 80 visual words and 9 local features, which are width, height, area, average red, green, blue channels, aspect ratio, compactness, and circularity.

V. Resulting Model and its Performance

Resulting model was selected to be a Random Forest with these hyperparameters: {'classifier__class_weight': 'balanced', 'classifier__max_depth': 20, 'classifier__min_samples_split': 10, 'classifier__n_estimators': 200} Its performance on the testing data as follows: IoU score - 0.296 and F1-score for 1's - 0.46. It is recommended to increase the mitosis cases in the dataset by adding flipped existing cells to decrease the class imbalance. Here is the table that illustrates the performance under different features vectors.

Feature Vector	IoU Performance
30 visual words + RGB mean + width + height + area	0.214
80 visual words + RGB mean + width + height + area	0.233
80 visual words + RGB mean + width + height + area + shape features	0.257
80 visual words + 32 bins of HSV Histogram + width + height + area + shape features	0.194
80 visual words + 16 bins of HSV Histogram + width + height + area + shape features	0.194
80 visual words + 32 bins of RGB Histogram + width + height + area + shape features	0.138
80 visual words + 16 bins of RGB Histogram + width + height + area + shape features	0.167

TABLE I
IoU performance for different feature vectors

In the table 1, IoU performance is presented for validation data. Apart from features vectors, hyperparameters of Random Forest Model were also tuned using Grid Search, and their performance can be navigated in the code provided.