Data Mining Final Project Report

Nuclei Detection

Yining Cao yinicao@iu.edu

Chuck Jia jiac@iu.edu

Ruiyu Zhu zhu52@iu.edu

April 28, 2018

Contents

1 Objective and Background

1.1 Objective

The objective of our project is to create an algorithm that automates the detection of nuclei from medical images. Nuclei are important subjects in medical research and diagnoses, as they contain most of a cell's genetic material, organized as DNA molecules. Identifying cell nuclei is the first step for most medical analyses, because it enables researchers to isolate individual cells from samples containing multiple cells. By performing further analysis on individual cell nuclei and observing cell activities and reactions to treatments, researchers can understand the underlying biological processes and use the knowledge to design new treatments or techniques. In particular, the aspect of nuclei is critical for evaluating the existence of many diseases (e.g. cancers) and their severity.

However, identifying nuclei is very time-consuming when performed manually. Automating the nuclei detection process can free researchers in medicine and biology so that they can focus on finding cures for diseases. And it may eventually speed up the cures for various diseases as well as reduce the time-to-market for new drugs, which is currently around 10 years on average.

1.2 Background

The nucleus is a membrane-enclosed organelle found in eukaryotic cells. Most of the human body's 30 trillion cells contain a nucleus full of DNA, the genetic material that programs each cell. Nucleus' shape, size, and texture depend on the nuclei type, malignancy of the disease and the nuclei life cycle. Mitotic nuclei (MN), a stage of nuclei life circle, is another factor that may cause variations to the nucleus' appearance and increase the complexity during a nuclei detection. Among the different types of nuclei, two types are usually the object of particular interest: lymphocyte nuclei (LN) and epithelial nuclei (EN). Epithelial cells line the outer surfaces of organs and blood vessels throughout the body, as well as the inner surfaces of cavities in many internal organs. Normal EN have nearly uniform chromatin distribution with smooth boundary. In contrast, EN in high-grade cancer cells are larger in size, may have heterogeneous

chromatin distribution, irregular boundaries, referred to as nuclear pleomorphism, and clearly visible nucleoli as compared to normal EN. Lymphocyte is one of the subtypes of white blood cell in a vertebrate's immune system. LN are inflammatory nuclei having regular shape and smaller size than EN. These variations of nucleus' appearance are one of the main difficulties in nuclei detection.

There are significant number of publications devoted to the nuclei detection problem. The most commonly used image processing methods in nuclei detection, according to the review paper [1], include: thresholding, morphology, region growing, watershed, active contour models (ACMs) and level sets, K-means clustering, probabilistic models (e.g. Gaussian mixture models (GMMs)) and graph cut (e.g. normalized cut (Ncut)). A large number of publications on nuclei segmentation used the above methods, separately or in combination. The above traditional nuclei segmentation frameworks have reported good segmentation performance on LN and normal EN having regular shape, homogeneous chromatin distribution, smooth boundaries, and individual existence. However, these frameworks have poor segmentation accuracy for CN especially when CN are clustered and overlapping. When there is chromatin variations. which is common in CN, the performance is also very poor. A second generation of nuclei segmentation frameworks tackles the challenges of heterogeneity, overlapping, and clustered nuclei by using machine learning algorithms together with classical segmentation methods. For example,Wahlby et al. [8] addressed the problem of clustered nuclei and combined seeded watershed on gradient magnitude images with shape-based cluster separation method to improved segmentation. And Fatakdawala et al. [9] proposed a Expectation-Maximization (EM) driven Geodesic ACM with overlap resolution for segmentation of LN in breast cancer histopathology.

Still, most model-based approaches segment nuclei using a priori information, which may introduce a bias favoring the segmentation of nuclei with certain characteristics. More recent studies have shown that deep learning methods produce promising results in nuclei detection. Xu et al. [10] used stacked sparse autoencoder to learn a high- level representation of nuclear and non-nuclear objects in an unsupervised fashion. Xie et al. [11] proposed structural regression convolution neural networks capable of learning a proximity map of cell nuclei and was shown by the authors to provide more accurate detection results.

In this report we will try three different approaches: pure image segmentation method, image segmentation combined with SVM and neural network approach, corresponding to the 3

generations of framework mentioned in [1].

2 Data Description

Our data comes from the Kaggle.com competition "2018 Data Science Bowl: Find the nuclei in divergent images to advance medical discovery". We are using in total 2 data sets in this project: a train data set and a test data set. They both contains color images of cell nuclei.

- The train data set consists of 670 cell nuclei images along with binary mask images for each individual cell nucleus.
- The test data set consists of 65 cell nuclei images. An encoded ground truth file is given for evaluation purposes.

In light of the image data type, below we describe 3 different approaches, with connections between one another, which we used in this project.

3 The Image Segmentation Approach

In this section, we propose an image segmentation method for nuclei detection. This segmentation method can be applied directly on the test data.

3.1 Brief Introduction of the Image Processing Methods

The main image processing methods we will use here are thresholding methods (Otsu's and triangle) and the watershed method.

Thresholding is the simplest method of image segmentation, which can be used to create binary images from a grayscale image. Thresholding method is quick and easy to use, and can provide decent outcomes on images with uniform backgrounds. There are a lot of variations

of the thresholding methods. We are using the Otsu's method and the triangle method for our experiment.

Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum. In other words, Otsu's threshold method chooses the threshold to minimize the intraclass variance of the black and white pixels, see [15]. The Otsu threshold method is the built-in method in MATLAB's `graythresh` function.

While in the triangle method, we first need the histogram of an image, then a line is constructed between the maximum value b of the histogram on the gray level axis and the lowest (or highest depending on context) value a on the gray level axis where the histogram is significantly larger than 0. The distance L normal to the line and between the line and the histogram is computed for all values from a to b. The level where the distance between the histogram and the line is maximal is the threshold value (level). For more details in triangle thresholding method see [16]. This technique is particularly effective when the object pixels produce a weak peak in the histogram. In other words, triangle methods is more sensitive to the object pixels as well as noise, when compared with the Otsu's method.

One main drawback of the thresholding method is that, it can not separate overlapped clustered nucleus. To overcome this difficulty, we will introduce the watershed method (see fig below). Watershed is a segmentation method that usually starts from specific pixels called markers and gradually floods the surrounding regions of markers, called catchment basin, by treating pixel values as a local topography. Catchment basins are separated topographically from adjacent catchment basins by maximum altitude lines called watershed lines. It allows to classify every point of a topographic surface as either belonging to the catchment basin associated with one of the local minimum or to the watershed line. Details about watershed can be found in [17].

(a) original image (b) after thresholding (c) dist transform watershed

Figure 1: using watershed to separate clustered masks

3.2 Segmentation Strategy

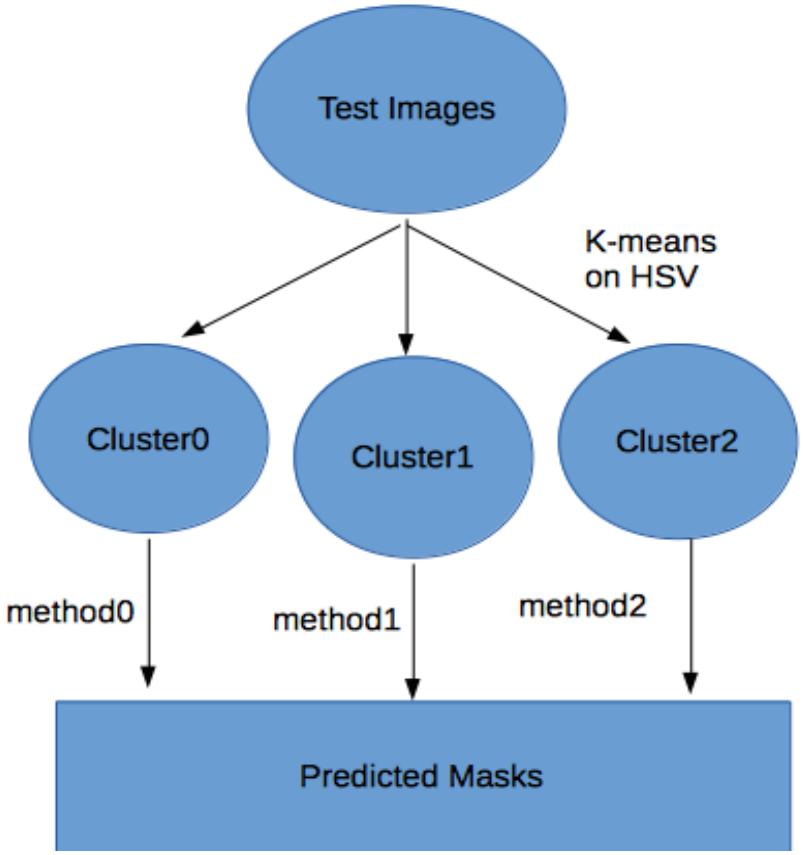


Figure 2: Overall Procedures

Considering the different types of images appeared in the test set, and the fact that different images need different segmentation skills. the first step we do is to use K-means clustering to divide the test images to 3 clusters:

- cluster0: histological slides with uniform backgrounds

- cluster1: fluorescent images
- cluster2: histological slides with multilevel backgrounds or bright field image

Then we apply different methods to each cluster. The codes is in premasknew.m.

- Method 0: On histological slides with uniform backgrounds, using Otsu thresholing can identify most object pixels, but the nucleus are usually clustered together. So the method using here is as follows:

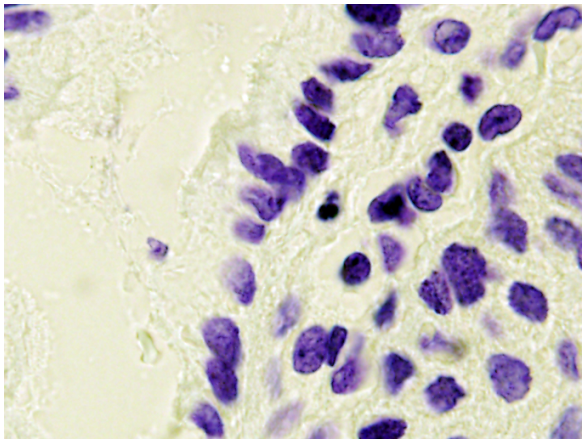
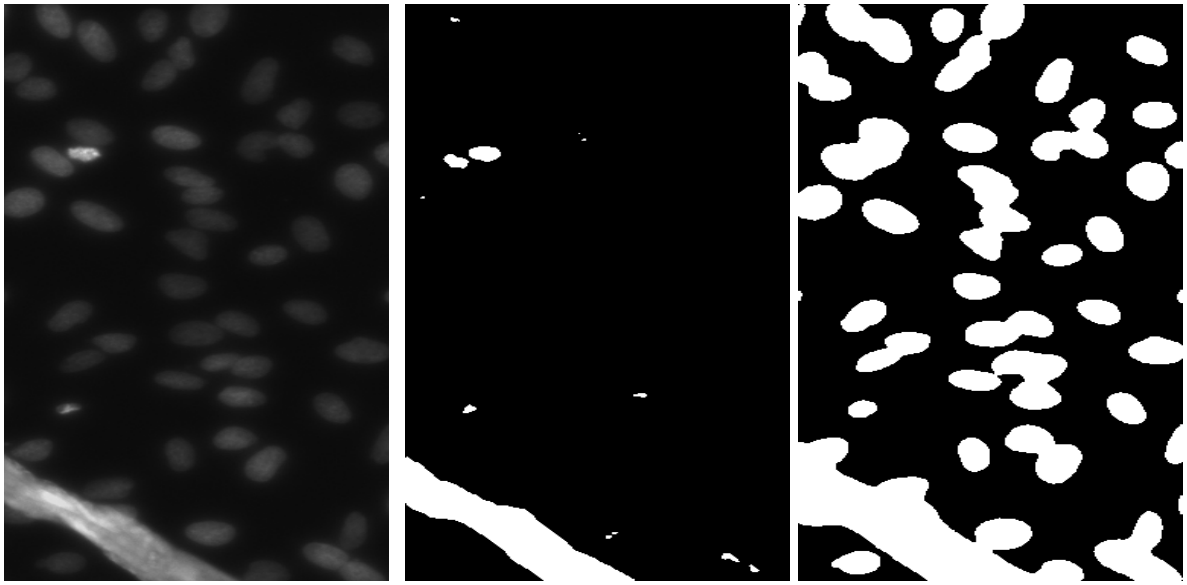


Figure 3: A typical image in cluster0

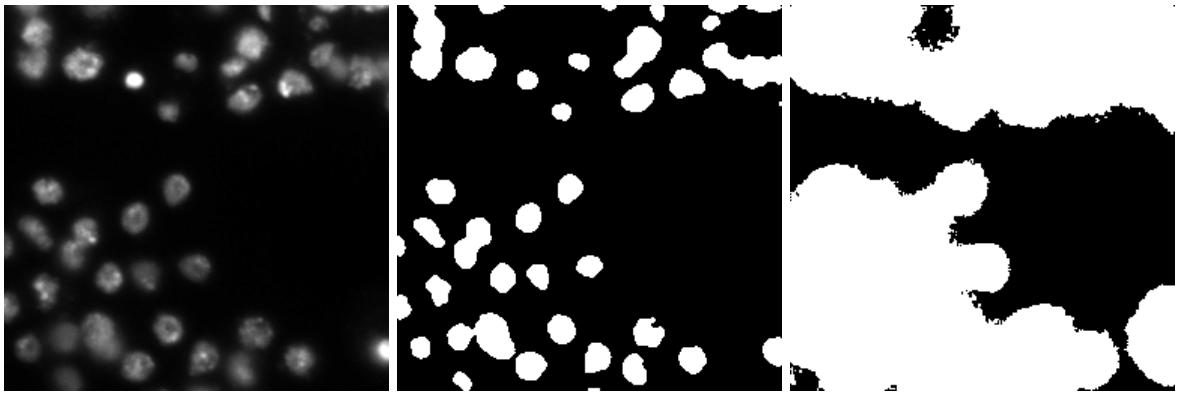
1. Read image, transform it into grayscale image using rgb2gray(), take the complement of the resulting image, so that the object is brighter than the backgrounds.
2. Apply graythresh function to the outcome of step 1 to get the thresh level, then transform the grayscale image to binary image (BW) based on the thresh level. Morphology were also used here to smooth out noise and fill in holes within each white objects.
3. Apply distance transform watershed method to separate the overlapping objects, get the separation boundary and output each connected regions as a predicted masks.

- Method 1: On fluorescent images, depending on the distribution and the brightness of the objects, we need to choose between the triangle threshold method and Ostu threshold method based on the number of connected regions generated by the two methods.



(a) original image (b) with ostu threshold method (c) with triangle method

Figure 4: example when triangle better than ostu



(a) original image (b) with ostu threshold method (c) with triangle method

Figure 5: Examples where Otsu's performs better than triangle

In general, we prefer the method that can identify more nuclei objects. However, we need to exclude the case where they are excessive number of connected regions because of noise. And the general procedures are as follows

1. Read image, transform it into grayscale image using rgb2gray(). Apply otsu and triangle method to the image to transform it into binary images BW1 and BW2 respectively, record the number of connected regions of BW1-C1 and the the number of connected regions of BW2-C2.

2. If C1>C2 or C2-C1 is too large (usually this means the triangle method identifies a lot of noise as objects) we apply distance transformation watershed on BW1 to get the predicted masks
3. Otherwise, we apply distance transformation watershed on BW2 to get the predicted masks

There is a question here, how to determine when is the value C2-C1 too large? For this part, we will record C2-C1 for all images, sort them in increasing order, and looking for a sharp change in the corresponding vector to determine the value.

- Method 2:

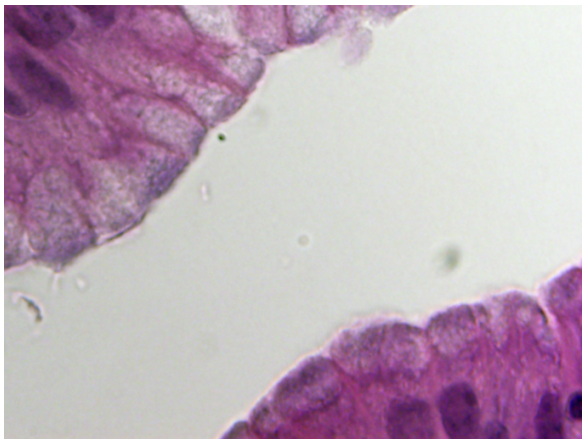


Figure 6: A typical image in cluster2

For histological slides with multilevel backgrounds or bright field images, what they have in common is that, besides the backgrounds for the cells, we also need to remove the non-nuclei materials within the cells. So we need to use multilevel thresholds here. The general steps in method 2 are the same as those for method 0, the only difference is that, instead of using graythresh in step2, we are using multithresh with 3 thresh levels here.

1. Read image, transform it into grayscale image using rgb2gray(), take the complement of the resulting image, so that the object is brighter than the backgrounds.
2. Apply multithresh function with 3 thresh levels to the outcome of step 1. Set the highest value of the return of multithresh as the desired thresh level (the brightest areas), then transform the grayscale image to binary image (BW) based on the

thresh level. Morphology were also used here to smooth out noise and fill in holes within each white objects.

3. Apply distance transform watershed method to separate the overlapping objects, get the separation boundary and output each connected regions as a predicted masks.

More segmentation examples and segmentation results are given in the section 6.

Besides giving the segmentation results directly, we also use the segmentation method to provide labeled potential masks for the SVM classification, which is discussed in the next section. Here we only give the labeled potential masks generation procedure.

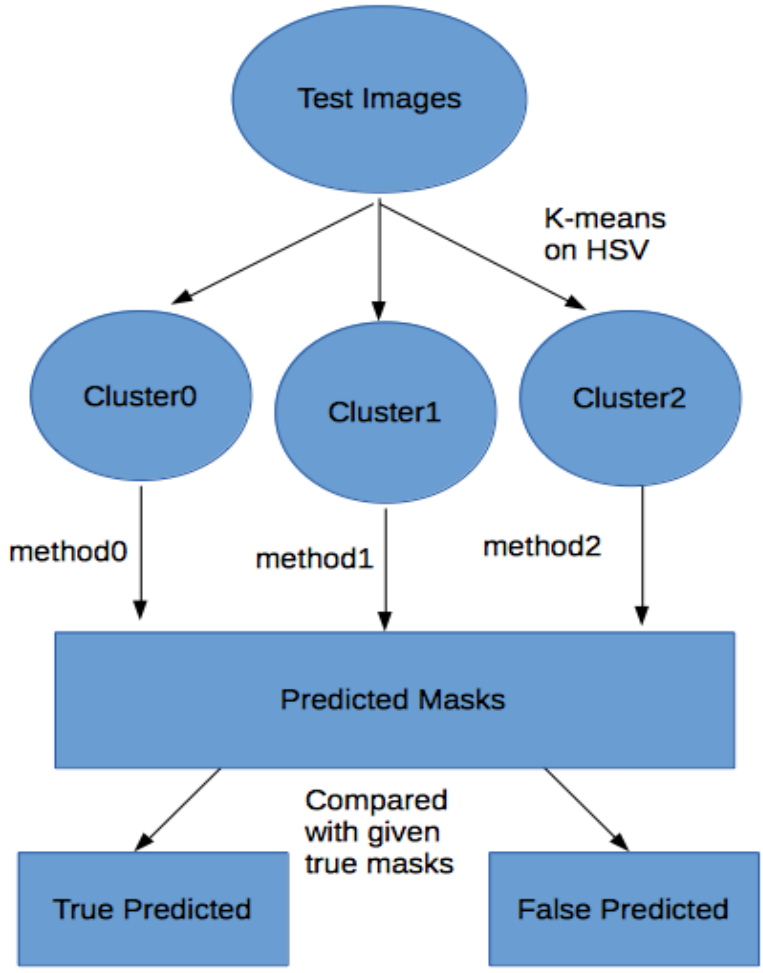


Figure 7: labeled predicted masks generation

4 The Support Vector Machine Augmented Image Segmentation Approach

The image segmentation approach produces tons of potentially accurate masks in its results. Some of them truly represent the locations of nuclei, while the others don't. To further improve our result, we would like to filter out those "false positive" masks. Our next step is to build a classifier to achieve this goal.

Our task in this step is to build a classifier to detect "correct" candidate masks. Therefore we should evaluate our classifier by its accuracy of detecting the "correct" masks instead of measuring the "quality" of the detected masks. This might introduce a gap between the intermediate outcome and our final goal. But we should not directly use the final goal in the loss function. This is because the quality of the pre-qualified candidates are limited. It is possible that some nuclei is not properly detected in the previous step. If we simply label them as "not a nuclei" then it will introduce a systematic bias to the input of the classifier.

4.1 Labels of the Pre-qualified Masks

What we get from the previous step are sets of pre-qualified masks. They are not ready for the application of any (supervised) classification algorithm, because the image masks are not yet labeled. We can't directly compare them with the ground truth masks since they are not 100% the same mask. Therefore, before training the classifier, we need to provide labels these pre-qualified masks.

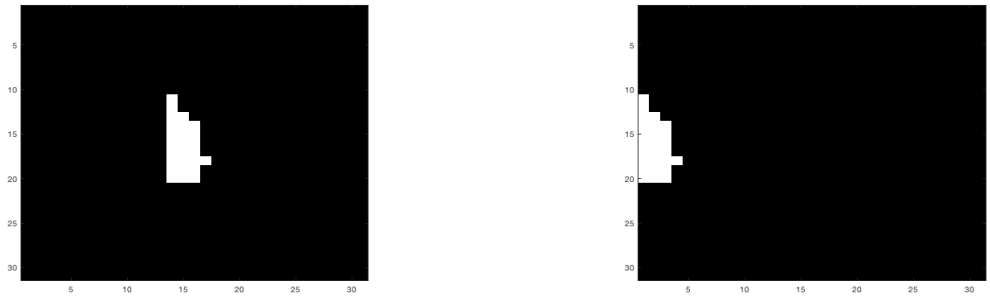
Our strategy for this task is modified from the overall evaluation strategy. The criterion of a "hit", according to the evaluation strategy, is whether the overlap part between a candidate mask and a truth ground mask is more than a threshold, say 50%. Therefore, we label a pre-qualified candidate as "positive" or "a correct mask" if it can be count as a hit with a certain threshold. After manually testing on hand-selected samples, we picked 65% as the best threshold.

4.2 Image Prepossessing

The input of this step (i.e. the output of the last step) is a set of masks for different images of different sizes. Therefore a first step of preprocessing is necessary.

4.2.1 Crop and Centering

After a quick scan of the data, we realized that the nuclei in different images are roughly of the same size. Therefore we could anticipate that it is a good strategy to “normalize” the raw data (origin masks) by cropping them into the same size and keep the “mask” at the center of the image. Unsurprisingly, this strategy is not perfect, because some of the ”potential masks” are at the edge of the image thus it is impossible to crop and then center them in the resulting images. There are two possible choices here: one is to surround the origin image with blank so any pixel in the origin image can be cropped and centered; the other strategy is to just crop & center at the most center-able pixel. Figure 8 shows the result of the two different strategies. After comparing these examples we decided on the first strategy as it would ensure the centered location of the nuclei.



(a) Fill with blank

(b) Center at closest pixel

Figure 8: Crop result of two different methods

Another issue is the size of the cropped images? On this issue, we had several concerns:

- If the size is too small, then the cropped image wouldn’t contain enough information to build the classifier.
- If the size is too large, then each sample point as the input of the SVM would have a huge

dimension, which might cause overfitting and expensive computation cost.

After scanning part of the ground truth mask, we figured out that the typical size of a real nuclei is between 15 by 15 pixels and 45 by 45 pixels. But it is not clear which combination is the best. To select the best parameter, we prepared three candidates for the cropped image size: 21 by 21 pixels, 31 by 31 pixels and 41 by 41 pixels. We then performed small scale experiments on the validation set to decide the best value for the size parameters.

4.2.2 Mask the origin image

The shape of the mask itself doesn't contain enough information for us to launch the classification. We also need to leverage the information in the origin image. Therefore, we will apply the pre-qualified masks onto the origin image before cropping & centering.

So far, the data is almost ready, the last step is to convert an image to a vector. We do this conversion with a straight forward method: convert each channel (red, blue, or green) of the image into a vector, then concatenate the three vectors into one long vector. Then, we will feed the converted image into the support vector machine. See Figure 9 for the whole pre-process phase.

4.2.3 Dimension Reduction

With some test run, we realized that the dimension of the sample points was too large and it took too long to train the model. Therefore we converted the image into grayscale before feeding into SVM.

4.3 SVM Parameter Selection

We are going to use the existing MATLAB SVM library to implement the classifier. There is one more issue before we proceed with the classifier. We still haven't decided on how we should configure our SVM classifier. For example, we have six different kinds kernel to try: "gaussian", "linear" and "polynomial" with order = 2, 3, or 4. Combined with different sizes

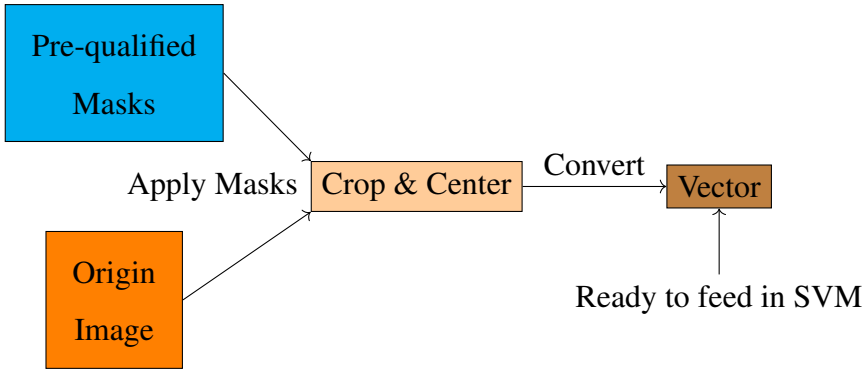


Figure 9: Preprocess Phase Diagram

of the cropped image, we now have 15 different sets of parameters to select from.

4.4 SVM Model Evaluation Method

Since we are using images and masks from the segmentation method and our target is to distinguish true positive masks from false positive masks, we will solely evaluate the performance of the SVM by the standard definition instead of plugging in the overall evaluation formula. The reasons for doing this are:

- There is no role for ground truth masks in SVM.
- The overall score of the output relies heavily on the input quality. If the input is not perfect, there is risk of overfitting.

5 The Neural Network Approach

Since the breakthrough by Krizhevsky et al. [13] in the 2012 ImageNet Large Scale Visual Recognition Challenge, the technique of deep neural networks has been widely used in the area of computer vision. While deep neural networks have achieved great success on image classification tasks, their performance on localization tasks are often less impressive. In [14], Ronneberger et al. proposed a convolutional neural network structure called "U-net" that generally performs well on biomedical image segmentation tasks. In this project, we adapted the

U-net structure to build our convolutional neural network model, which was trained and used on cell nucleus prediction.

5.1 The U-net Structure

A U-net consists of two parts: a contracting part and an expansive part. An illustration is shown below in Figure 10.

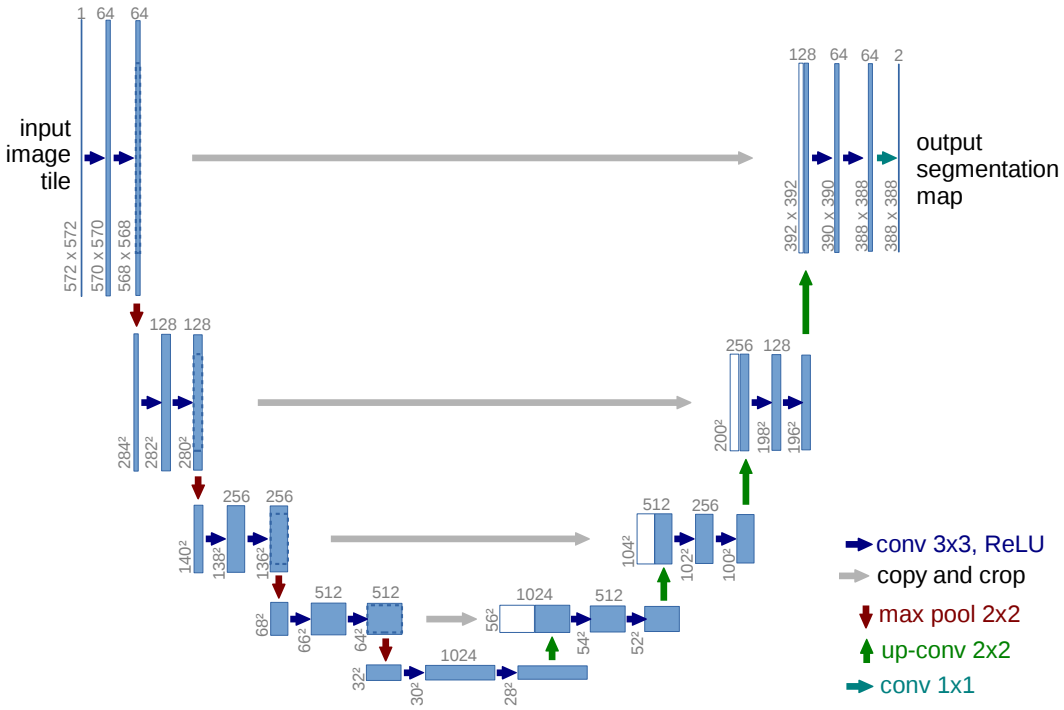


Figure 10: An example of a U-net structure. Note that the U-net we are using in the cell nuclei localization task has the same structure but with different input and layer sizes. Graph from Ronneberger et al. [14]

.

- The contracting part of the network resembles a typical feed forward convolutional neural network (CNN). The CNN can be described as a number of repeated steps, each of which involves application of convolutions, activation units, and pooling operations. More specifically, in each step, we apply two 3×3 convolutions, each one following by applying a rectified linear unit (ReLU). Then at the end of a step, we perform a 2×2 max-pooling with stride 2 downsampling. In Figure 10, the input data of the CNN is of

size 572×572 with 1 channel. The contracting part of the CNN ends with a layer of size 28×28 with 1024 channels.

- The expansive part of the network also consists of a number of repeated steps. In each step, we apply a upsampling for the feature channels, followed by a 2×2 up-convolution and a concatenation with the corresponding feature map from the contracting part of the CNN. [1] Then two convolutions of size 3×3 are applied, followed by the application of a ReLU unit. In Figure 10, we can see that the expansive part of the network ends with an output of size 388 with 64 channels.

- In addition to the contracting and expansive parts, we add a final layer which uses a 1×1 convolution to map all the feature channel vectors to the correct number of classes and then output the final result. In Figure 10, the final output of the whole U-net has size 388×388 with 2 feature channels.

In our project, the cell nuclei detection task is implemented as a classification problem with 2 classes for all pixels: either a pixel is inside a cell necleus or it is outside. Therefore, our final output of the U-net has two feature channels, i.e. a binary image indicating if each pixel is inside a nucleus.

Note that since U-net models requires a fixed size for the input data and we have images with different sizes in our data set, we need to resize the input images if they are not compatible with the U-net. Therefore, the final output of the U-net might have a different size from the original image. To generate the predicted masks for the original image, we thus need to rescale the output image to the size of the original images.

5.2 Post-processing: Segmentation

The output result for each cell nucleus image from the U-net is a binary mask image. To predict the location of each cell nucleus, we need to perform segmentation on the binary mask in order to generate individual binary masks for each cell nucleus. Then we perform segmentation

[1] Note that here due to the unmatching sizes from the two parts, cropping may be performed for the concatenation. The unmatching sizes are caused by the un-padded convolution we choose for the convolutional neural network.

Figure 11: Illustration of the task of post-processing segmentation. On the left is a sample result from the U-net output. On the right is one of the desired individual cell mask.

```
[ ]  [ ]  [ ]
   \  |  /
[ ]--[x]--[ ]
   /  |  \
[ ]  [ ]  [ ]
```

Figure 12: Illustration of connected components. In our implementation, the 8 pixels around `x` are considered as connected to `x`.

on the binary masks and generate separate masks for each cell nucleus. Figure 11 shows an example of a U-net output and an individual binary mask of a cell nucleus.

Since the output of the U-net is a binary mask, we perform the simple connected components algorithm for segmentation. The method works by assigning labels to pixels. Connected pixels are assigned the same label and disconnected pixels are assigned different labels. In our connected components method, two pixels are connected if they are vertically, horizontally, or diagonally adjacent. An illustration is shown in Figure 12.

6 Evaluation Method

The Kaggle competition provides a detailed evaluation metric (provided below)[2]. We will use the same method to evaluate the accuracy of our approach. To determine the quality of our final result, we will evaluate using k-fold validation methods on our training set. In addition, we will also use the test data set for evaluation purposes.

6.1 Evaluation Metric

For the evaluation of accuracy, we use the intersection over union (IoU) method. The IoU of a detection result D and the true cell location T, both as local images, can be calculated as

$$\text{IoU}(D,T) = \frac{\text{area}(D \bigcap T)}{\text{area}(D \bigcup T)}$$

The accuracy for a detection result on one nuclei image is calculated as the mean average IoU scores at different thresholds, as described below.

A threshold is used to determine if our predicted location for one cell nucleus is an acceptable result or a "hit". For example, for a threshold of 0.5, if the IoU between our detection and the ground truth result is greater than 0.5, then our result will considered a hit, i.e. a true positive. For a detection with an IoU score less then 0.5, the detection result for that nucleus will be considered as not acceptable, i.e. a false positive.

At each threshold level t, a precision score can be calculated by

$$\text{precision score} = \frac{TP(t)}{TP(t)+FP(t)+FN(t)}$$

where TP, FP, and FN are the numbers of true positives (TP), false negatives (FN), and false positives (FP). The TP, FN, FP numbers can be calculated by comparing the predicted result with the ground truth objects using the IoU scores with a specific threshold. The average precision of a single image is then calculated as the mean of the above precision scores at each IoU threshold:

$$\text{average evaluation score for one prediction} = \frac{1}{|\#\text{ of thresholds}|}\sum_t \frac{TP(t)}{TP(t)+FP(t)+FN(t)}$$

[2]The description of the evaluation metric below follows closely with the evaluation section in the competition webpage.

In the end, we take the mean over the evaluation scores of all image predictions, as

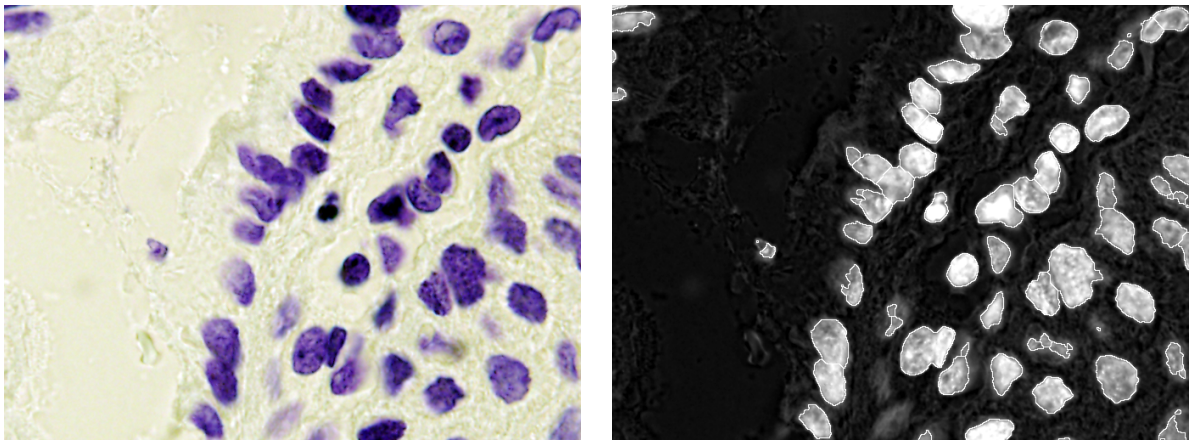
$$\text{evaluation score} = \frac{1}{N}\sum_{i=1}^{N}\text{average evaluation score}_{\text{image}_i}$$

where N is the total number of images.

7 Experiments and Results

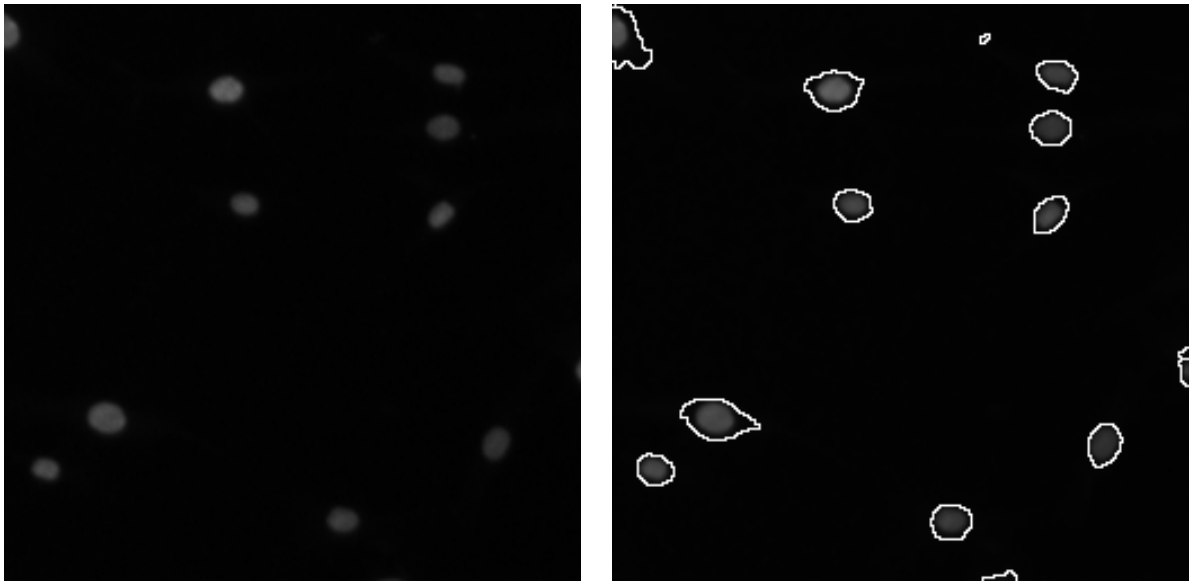
7.1 Pure Image Processing Approach

The experiment process is given in Section 3. As the pure image processing method only use the test data, cross validation does not apply here. We show some segmentation result (the segmentation boundary is set to gray level equals to 255) below and provide a table showing the accuracy at different IoU thresholds (not the threshold in image segmentation).



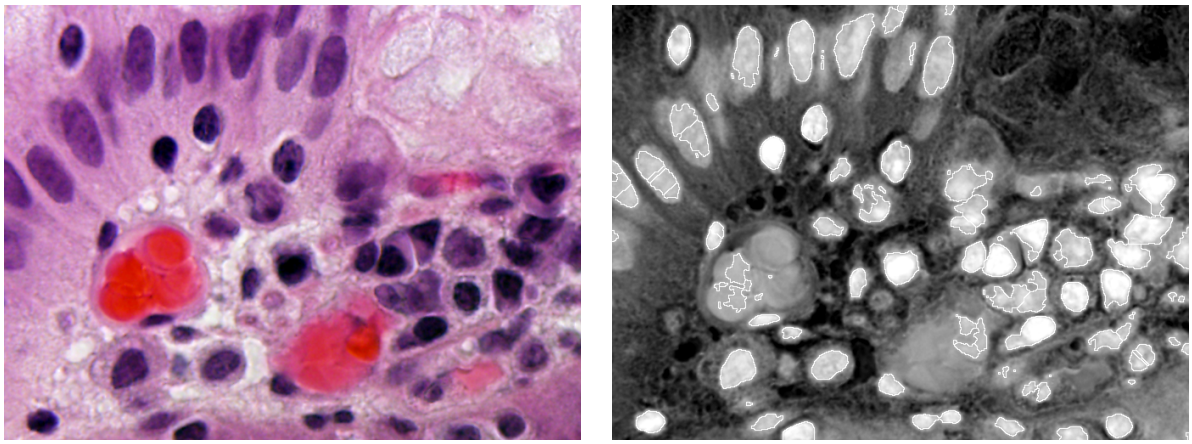
(a) original image (b) segmentation result

Figure 13: example segmentation in cluster0



(a) original image (b) segmentation result

Figure 14: Example segmentation in cluster1



(a) original image (b) segmentation result

Figure 15: Example segmentation in cluster2

Using the evaluation method in section 6, we have the following table on the accuracy.

Threshold	0.4	0.45	0.5	0.55	0.6	0.65	0.7
Evaluation Score	0.2964	0.2524	0.2134	0.1701	0.1278	0.0995	0.0666

Table 1: Evaluation scores of image processing approach. Evaluation scores are calculated as in Section 6.

Analysis of the result: From the figures of the segmentation results, we see that that the pure image processing method can identify most nuclei objects, but the evaluation score is not very high. The possible results are the follows:

1. During the segmentation process, we transformed all the images to grayscale images, there are information missing during the transformation. And the difference bewteen the background and objects may become less obvious after the transformation.

2. During the segmentation process, the first segmentation method used is threshold method, which use the local intensity of the grayscale image to determine the objects and backgrounds. And the shape of the objects is not used in this step. This is based on the assumption that the nuclei has higher intensity than the backgrounds. But for some histopathology image (for example, images in cluster2), areas outside the nucleus may have high intensity as well. Also, the above assumption may include a lot of noise points as the objects, even if we use some noise reducing filters(such as medfilt2) and morphological operations to reduce the noise.

3. The usage of distance transformed watershed method may also bring in some over segmentation of the images, namely, a correct identified objects may be divided into multiple parts (like a 'sub-mask'), resulting in a low IoU value. So a correct identified 'sub-mask' may not be count as a 'hit'.

4. Using only the image processing method, it is harder to get the same boundary as the ground truth masks. This is another reason why a correct identified object may have IoU values lower than the IoU threshold, failing to be counted as a 'hit'.

7.2 SVM-Augmented Segmentation Apporach

7.2.1 SVM results

Model Validation

We chose a set of 10 images (contain 1082 samples with 50% of them being correct masks), and run the preprocessing and classifier with different combination of parameters and evaluate

its accuracy with 5-folder cross validation. Table 2 shows the result of our validation.

Kernel	Linear	Gaussian	2-order Poly	3-order Poly	4-order Poly
21 x 21 Pixels	0.3115	0.3355	0.3216	0.4301	0.4963
31 x 31 Pixels	**0.2884**	0.3355	0.3706	0.3826	0.5111
41 x 41 Pixels	0.2994	0.3355	0.3715	0.3189	0.5591

Table 2: Model validation result (using 5-folder CV). The values are the average loss, i.e. the error rate.

From the table we can see "31 x 31 Pixels" with a 'Linear" kernel is the most promising combination. In our parameter tuning process, we would try this configuration first and then with a larger crop size, i.e. "41 x 41 Pixels" and select the best parameter combination.

SVM Results

We then used the provided SVM library to build the classifier on the pre-processed data. Each sample point is a unrolled 31 x 31 pixels RGP image, i.e. a $31^2 = 961$ dimension vector. There are 94170 samples in all while 44105 of them are labeled as "correctly pre-qualified" while the rest are labeled as "incorrectly pre-qualified". We choose linear kernel functions as suggested by the model validation experiments.

To evaluate the accuracy of our classifier, we used 5-fold cross validation. The trained model gave an average loss of 0.4082, which indicates an accuracy of 59.18%. We tried with another configuration but it failed to produce a better result (loss = 0.4357).

This result is only slightly better than a random guess and deviate what we observed in the validation experiments. The reason could be that the validation set does not represent the whole dataset well. In fact, the samples don't always show similar characteristics. Figure 16 gives some idea about the distribution of the # of "correct" and "incorrect" masks we got from each image.

We can see that there are several outliers with a spam of incorrect masks. If we remove them out, we could get an accuracy of 70.1% at the rest of the database. Figure 17 shows the distribution on the rest of the dataset. This result indicates that successfully separate three classes

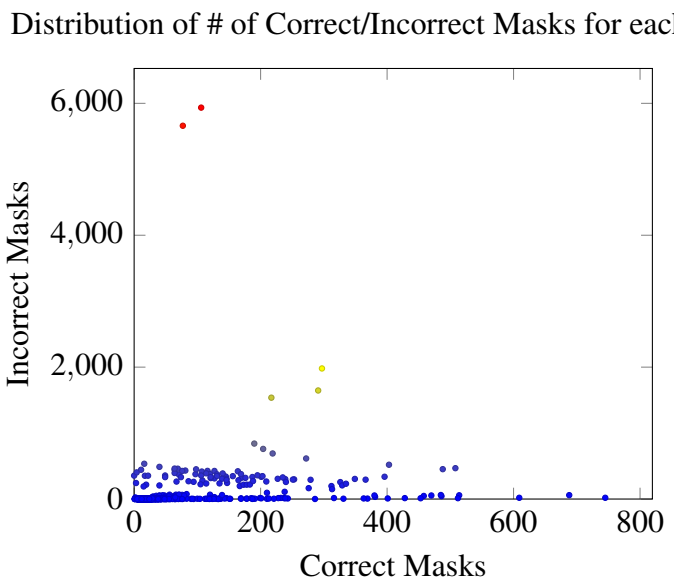


Figure 16: Distribution of # of Masks

would greatly improve our results. The three classes are defined as follows

- Most of the pre-qualified masks are correct: Figure 18
- Pre-qualified masks contains a number of mistakes: Figure 19
- Most of the pre-qualified masks are incorrect: Figure 20

We can see that each class has its own style and different classes apparently come from different source, which is one of the root cause of the low accuracy in our results. It might be a correct direction to first figure out which class a image belongs to and then build classifiers on these three classes. Since the percentage of correct masks in each class varies a lot, this approach would also make sense even with a trivial classifier in each class. However, due to our limit on time and resources we weren't able to follow this line of research.

7.2.2 Overall Performance with SVM

After filtering the masks produced by image segmentation with SVM, we used the overall evaluation method to measure the accuracy of the prediction of SVM. Table 3 shows the accuracy of this approach. This result is even worse than the segmentation before applying SVM, though SVM itself has a non-trivial accuracy.

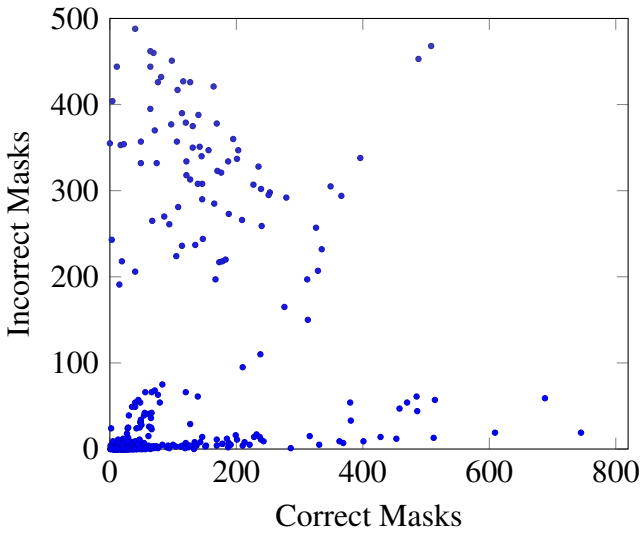
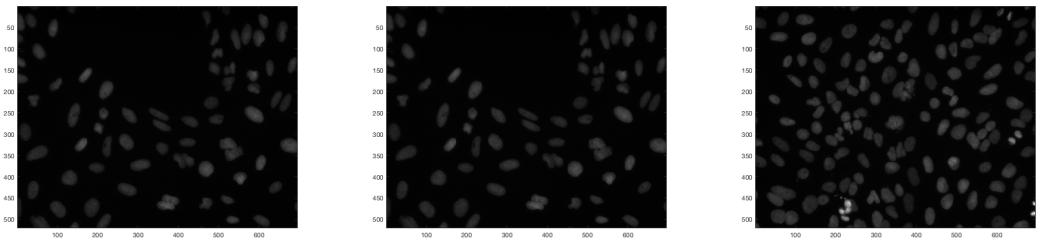


Figure 17: Distribution of # of Masks



(a) 127 correct masks
29 incorrect masks
SVM Accuracy=96.7%

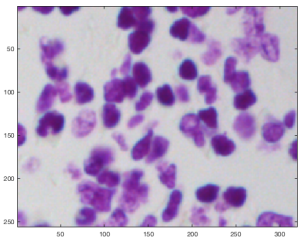
(b) 74 correct masks
2 incorrect masks
SVM Accuracy=96.1%

(c) 146 correct masks
14 incorrect masks
SVM Accuracy=96.9%

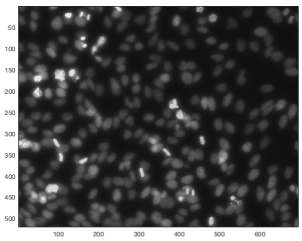
Figure 18: Representatives of class 1

Our conjunction on the reasons behind are:

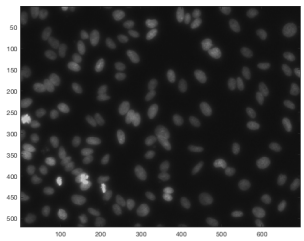
- The SVM itself is not very accuracy for the reasons discussed above.
- The goal of SVM model and the project don't work well. In a SVM, the goal is to reduce the total "error", the "error" of a sample only relies on its class and its distance to the vector. However, the evaluation of the project only focuses on whether a nuclei is correctly recognized. For example, duplicate masks of the same nuclei in the final evaluation won't introduced any improvement/penalty. However multiple duplicate in the samples will significantly affect SVM.



(a) 167 correct masks
197 incorrect masks
SVM Accuracy=60.7%

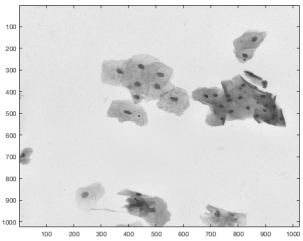


(b) 508 correct masks
468 incorrect masks
SVM Accuracy=51.3%

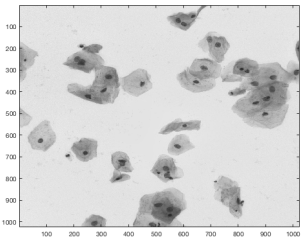


(c) 253 correct masks
298 incorrect masks
SVM Accuracy=54.9%

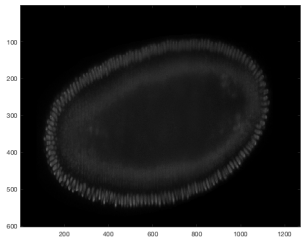
Figure 19: Representatives of class 2



(a) 77 correct masks
5660 incorrect masks
SVM Accuracy=22.3%



(b) 106 correct masks
5935 incorrect masks
SVM Accuracy=29.3%



(c) 297 correct masks
1980 incorrect masks
SVM Accuracye=10.2%

Figure 20: Representatives of class 3

7.3 Results from U-net with Post-processing Segmentation

7.3.1 Experiment Design

To fit our model, we used a U-net structure same as shown in Figure 10 but with different number of layers and input sizes. Our model is using a U-net with an input size of 256×256 and 3 channels, each corresponding to one of the three RGB channels in the original image. the middle layer between the contracting part and the expansive part has a size of 16×16 with 256 feature channels. In total our U-net network consists of 22 convolutional layers, which has in total 1,941,105 parameters.

Our code is based on the code from this article by K. Amdal-SavikKeras. We also borrowed

Threshold	0.4	0.45	0.5	0.55	0.6	0.65	0.7
Evaluation Score	0.2521	0.2234	0.1908	0.1539	0.1182	0.0915	0.0628

Table 3: Evaluation scores of SVM-augmented approach. Evaluation scores are calculated as in Section 6.

ideas from the the original U-net code by Ronneberger et al. Model parameters including the number of layers, number of feature channels are tuned to fit our task. For the training process, we chose to use the binary cross entropy combined with the IOU score as loss function. We directly used the `mean_iou` from the TensorFlow package `tensorflow.metric` to implement our IOU score. The back-propagation with stochastic gradient descent is implemented using the ADAM method.

We also added GPU support to the code to yield faster back-propagation. On the test set, the code was run in the Ubuntu 16.04 environment with Intel i7-8700k CPU and GeForce GTX 1080Ti GPU. Code converged in about 96s with 24 epochs[3].

7.3.2 Cross Validation Results

We performed 5-fold cross validation on the train set. For each validation set, we ran at most 30 epochs on the U-net model, with some of them converging early. Results are as follows.

	Fold 1	**Fold 2**	**Fold 3**	**SFold 4**	**Fold 5**
Run 1	0.3827	0.4743	0.4551	0.3343	0.3636
Run 2	0.4146	0.5105	0.4022	0.4024	0.3526
Run 3	0.2058	0.4345	0.4032	0.3740	0.4117

Table 4: Average Evaluation Scores from 5-fold Cross Validation. Here the scores are calculated as described in the Section 6, i.e. as the average score from evaluation scores using 7 different thresholds.

From the results, we can see that the quality of the results vary by a lot. This is because the quality of the convolutional neural networks models highly depend on the initial coefficients,

[3]The code also works with only CPU, but runs extremely slow with only a few cores.

as the stochastic gradient descent solver implemented in the ADAM method does not guarantee optimal solution. With some bad initialization, we can see some low quality result as in Fold 1 from Run no. 3.

7.3.3 Test Set Results

Using the whole train set to train the model, we have the following accuracy result. Evaluation scores with different thresholds are shown in Table 5.

$$\text{average evaluation score} = \frac{1}{\#\text{ of thresholds}} \sum_{\text{thr}} \frac{TP_{\text{thr}}}{TP_{\text{thr}} + FN_{\text{thr}} + FP_{\text{thr}}} = 0.4390$$

Threshold	**Evaluation Score**
0.4	0.5229
0.45	0.4956
0.5	0.4619
0.55	0.4311
0.6	0.4146
0.65	0.3871
0.7	0.3601

Table 5: 5-fold cross validation results from the U-net approach

7.4 Comparison Between Different Methods

Now we compare our 3 different approaches. On the test set, we have the following results, as shown in Table 6.

As we can see, the U-net approach stands out as the best approach for our data set. This is consistent with the reputation of the convolutional neural networks on computer vision tasks.

Method	Average Evaluation Score
Segmentation Only	0.175189
Segmentation + SVM	0.156101
U-net + Segmentation	0.439041

Table 6: Average evaluation scores from applying 3 different approaches on test set

8 Conclusions

In this project, we implemented 3 different approaches to perform localization of cell nuclei. Several observations can be made

- **The Segmentation Methods**

  For the segmentation methods of thresholding and watershed, they typically generate very good segmentation results. One disadvantage of this type of methods, from observations on our results, is that the methods generate many false positive cell nuclei. Therefore, they perform well on sensitivity scores but not as well using our evaluation scores.

- **The Support Vector Machine Method**

  We applied the SVM method on the results of the segmentation methods in hope to improve our localization quality by eliminating false positives. However, we failed to design a good SVM structure that performs well on this task. We think there are two main reasons for this.

  - The SVM itself is not very accuracy for the reasons discussed above in Section 7.2.2.
  - The goal of SVM model and the project don't work well. Either it need a sophisticated kernel function for this task, or the SVM model simply cannot capture the features that could be used to improve accuracy.

- **The U-net Method with Segmentation Post-processing**

As we observed from the result, this approach performs relatively well. This is consistent with neural networks' reputation on computer vision tasks. We believe it might be because of the high expressivity power of the neural networks. The convolutional neural networks can be very flexible in learning different type of features. One clue on this is that neural networks generally involve huge number of coefficients to provide freedom on expressing functions and maps. In our example, our U-net has almost 2 million trainable parameters[4]. One potential improvement on the neural network approach we can make for in the final task of segmentation, we can train a different neural network for the post-processing segmentation task. We believe a second neural network would be able to capture some different features from the first network such that they can achieve some goals that the first network failed to achieve. For example, one desirable goal is to be able to separate some of the cells that are too close to be separated by the simple connected component labeling algorithm.

9 Individual Tasks

- **Yining Cao:** Implemented the pure image processing approach. Applied segmentation methods directly on the test set to get predicted masks. Used image segmentation methods on the train set to provide labeled (true/ false) potential masks for the SVM method. Evaluated the accuracy of the pure image processing method.

- **Chuck Jia:** Implemented the U-net approach. Trained and applied U-net models on the project data sets in Keras using Tensorflow. Implemented the overall evaluation method in MATLAB and evaluated results on the test set from 3 different approaches.

- **Ruiyu Zhu:** Implemented the SVM augment. Trained and evaluated the model with black-box usage of image segmentation approach. Evaluated the augmented image segmentation approach on project test set.

[4]For details, see Section 7.3.1

References

[1] H. Irshad, A. Veillard, L. Roux and D. Racoceanu, *Methods for Nuclei Detection, Segmentation, and Classification in Digital Histopathology: A Review—Current Status and Future Potential*, IEEE Reviews in Biomedical Engineering, vol. 7, pp. 97-114, 2014.

[2] O. S. Al-Kadi, *Texture measures combination for improved meningioma classification of histopathological images*, Pattern Recog. vol. 43 no. 6 pp. 2043-2053 2010.

[3] H. Irshad, *Automated mitosis detection in histopathology using morphological and multi-channel statistics features*, J. Pathol. Inform. vol. 4 pp. 10–15 May 2013.

[4] P. W. Huang Y. H. Lai, *Effective segmentation and classification for HCC biopsy images*, Pattern Recog. vol. 43 no. 4 pp. 1550-1563 2010.

[5] V.-T. Ta O. Lézoray A. Elmoataz S. Schüpp, *Graph-based tools for microscopic cellular image segmentation*, Pattern Recog. vol. 42 no. 6 pp. 1113-1125 2009.

[6] O. Lezoray H. Cardot, *Cooperation of color pixel classification schemes and color watershed: A study for microscopic images*, IEEE Trans. Image Process.¡/em¿ vol. 11 no. 7 pp. 783-789 Jul. 2002.

[7] Y. Al-Kofahi W. Lassoued W. Lee B. Roysam, *Improved automatic detection and segmentation of cell nuclei in histopathology images*, IEEE Trans. Biomed. Eng. vol. 57 no. 4 pp. 841-852 Apr. 2010.

[8] C. Wahlby I. M. Sintorn F. Erlandsson G. Borgefors E. Bengtsson, *Combining intensity edge and shape information for 2D and 3D segmentation of cell nuclei in tissue sections*, J. Microsc. vol. 215 no. 1 pp. 67-76 2004.

[9] H. Fatakdawala J. Xu A. Basavanhally G. Bhanot S. Ganesan M. Feldman J. E. Tomaszewski A. Madabhushi, *Expectation maximization-driven geodesic active contour with overlap resolution (EMaGACOR): Application to lymphocyte segmentation on breast cancer histopathology*, IEEE Trans. Biomed. Eng. vol. 57 no. 7 pp. 1676-1689 Jul. 2010.

[10] J. Xu, L. Xiang, Q. Liu, H. Gilmore, J. Wu, J. Tang, and A. Madab- hushi, *Stacked sparse autoencoder (ssae) for nuclei detection on breast cancer histopathology images*, Medical Imaging, IEEE Transactions on, vol. PP, no. 99, pp. 1–1, 2015.

[11] Y. Xie, F. Xing, X. Kong, H. Su, and L. Yang, *Beyond classifica- tion: Structured regression for robust cell detection using convolutional neural network*, in Medical Image Computing and Computer-Assisted InterventionMICCAI 2015. Springer, 2015, pp. 358–365.

[12] Y. Xie, X. Kong, F. Xing, F. Liu, H. Su, and L. Yang, *Deep voting: A robust approach toward nucleus localization in microscopy images*, in Medical Image Computing and Computer-Assisted Intervention– MICCAI 2015. Springer, 2015, pp. 374–382.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, NIPS (2012), pp. 1106-1114.

[14] O. Ronneberger, P. Fischer, and T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, LNCS, Vol.9351: 234-241, 2015

[15] N. Otsu, *A Threshold Selection Method from Gray-Level Histograms*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, pp. 62-66, 1979

[16] D.W. Zack, W.E. Rogers, Latt, SA. *Automatic measurement of sister chromatid exchange frequency*, The journal of histochemistry and cytochemistry : official journal of the Histochemistry Society. 25. pp. 741-53. 1977.

[17] J. B. Roerdink and A. Meijster, *The watershed transform: Definitions, algorithms and parallelization strategies*, Fundamenta Informaticae, vol. 41, no. 1, pp. 187–228, 2000