Segmentation of Glands in the H&E stained Histologic images

A thesis submitted in partial fulfillment of
the requirements for the degree of

Bachelor of Technology

by

Rahul Ghosh and Ashima Jain
(Roll No. 120108027 and 120108051)

Under the guidance of
Dr. Amit Sethi

DEPARTMENT OF ELECTRONICS & ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
April 2016

CERTIFICATE

This is to certify that the work contained in this thesis entitled

Segmentation of Glands in the H&E stained Histologic images

is the work of

Rahul Ghosh and Ashima Jain
(Roll No. 120108027 and 120108051)

for the award of the degree of Bachelor of Technology, carried out in the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.

_____

Guide

Date: _____

Place: _____

DECLARATION

The work contained in this thesis is our own work under the supervision of the guides. We have read and understood the "B. Tech./B. Des. Ordinances and Regulations" of IIT Guwahati and the "FAQ Document on Academic Malpractice and Plagiarism" of EEE Department of IIT Guwahati. To the Best of our knowledge, this thesis is an honest representation of our work.

_____

Author

Date: _____

Place: _____

Acknowledgments

We are highly indebted to our guide Dr.Amit Sethi, Assistant Professor, IIT Guwahati for providing us with the opportunity to work on this project as well as for his constant support and guidance throughout the course of the project.

We would also like to extend our gratitude towards Mr. Abhishek Vahadane and Mr. Neeraj Kumar Vaid who helped us in this project by providing their valuable suggestions.

Last but not the least, our sincere thanks to all those who encouraged us to complete the project.

Abstract

Automated system of segmentation and classification of gland tissues has become a necessity of the hour provided the ever increasing rate of people suffering from cancer. In this work, we have proposed a novel technique of gland segmentation using deep learning. The scheme involves training the dataset using a convolutional neural network having two convolutional layers, one hidden layer and the final segmentation into gland or non-gland is done using logistic regression model. In contrast to the other techniques which are highly dependent on the gland structure such as presence of lumen or nuclei around the gland, our proposed method overcomes all these problems. The efficacy of the automated gland segmentation system has been evaluated by computing the validation and test error as well as other evaluation metrics like F1-score, dice index and jaccard index on the Warwick-QU dataset from the gland segmentation challenge contest at MICCAI 2015.

Contents

List of Figures

Chapter 1

Introduction

Cancer is a class of diseases characterized by out of control cell growth. According to the American Cancer Society, Cancer is the second most common cause of death in the US and accounts for nearly 1 of every 4 deaths. The World Health Organization estimates that, worldwide, there were 4 million new cancer cases and 8.2 million cancer-related deaths in 2012 (their most recent data) [1]. Cancer can occur in any part of the body, the most prevalent being lung, prostate, breast, colon and rectum. If cancer is detected, it is necessary to know its stage or how far it has spread. The diagnosis is done by a radiologist or a pathologist using the biopsy sample of the patient. The grading is done using the Gleason grading method which grades a tumor with a score between 2 to 10. This method is the addition of two scores (the most and the second most prevalent pattern in the sample) each ranging from 1 to 5. A grade 1 pattern is very well differentiated whereas grade 5 is poorly differentiated. Fig. 1.1 shows the different carcinoma patterns. In grade 1 and 2 (benign patterns), the glands have large lumen with prominent nuclei.

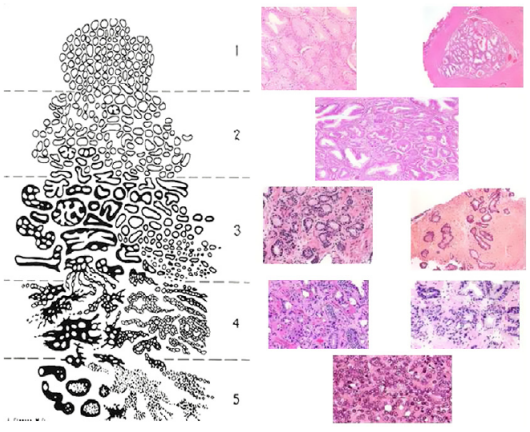


Figure 1.1: Five grades of the Gleason grading pattern applied to the prostate tissue [2].

Grade 3 glands are more circular with smaller lumen and thin nuclei boundaries. The glands start to lose their architecture as we move to grade 4 whereas grade 5 glands have poorly defined units. The tumors are graded manually by the pathologists by observing the biopsies under a microscope. This method is both tedious as well as error prone. Moreover, it also depends on the skills of the person. Hence, an automated system for cancer detection and grading is the need of the hour. Such a system should not only grade the tumors in less time but also with great precision and accuracy.

Chapter 2

Literature Review

Nguyen et al. [2] have presented a segmentation based classification method which is based on the structure of the glands. The pixels were first converted into Lab space from the RGB and then divided into 5 categories viz., stroma, nuclei, cytoplasm, lumen and mucin using the voronoi tessellations of the training points. They then created a binary image of nuclei and non-nuclei objects. The nuclei objects were enlarged by combining with the adjacent cytoplasm pixels. The enlarged nuclei objects which intersect were combined to construct the gland boundary segments. However, since the nuclei may not be densely packed everywhere on the boundary, so a complete gland boundary may not be obtained at this stage. The complete boundary was obtained by combining with the lumen. The authors created a lumen and non-lumen binary image and expanded the lumen boundary until either the boundary touches the stroma or exceeds a defined threshold and combined with the cytoplasm objects. After obtaining separate glands, fifteen features were extracted from the glands which included lumen features (area statistics, perimeter statistics, circularity statistics, percentage of the gland area to be lumen area, number of lumina in the gland), nucleus features (percentage of the gland area to be nuclei area, nucleus density), gland morphology features (average and variance of gland radius), mucin features (percentage of gland area to be mucin area). They used Adaboost, nearest neighbor, decision tree, bayes, svm and fnn to classify the glands into benign, grade 3 and 4 out of which svm and fnn produce the best results.

In [3], Gaussian filter was applied to the image which was then converted from RGB to grayscale. Thresholding was employed to separate out the lumen objects. The image was converted from RGB to Lab space and k-means clustering was applied to a* space of the image for initial segmentation of the glands. The number of clusters was two: cluster of lumen and

cytoplasm pixels, and cluster of stroma and nuclei pixels. Closing operation was applied on the initially segmented glands so as to fill any holes or gaps. Size constraint was also applied to keep only true glands. Nine features were extracted from the segmented glands, namely, average lumen area, maximum lumen area, average lumen eccentricity, average gland area, maximum gland area, average gland diameter, average gland perimeter, average gland eccentricity and gland density. Classification of the glands was done using Bayes Classifier.

Naik et al. [4] proposed a method of gland and nuclei segmentation using low-level and high-level information. In low-level information, Bayes theorem was used to obtain pixel-wise likelihood for each pixel which in turn generated likelihood scenes giving the probability of the pixel to belong to lumen, cytoplasm or nucleus class. In the detection of lumen objects, the authors took the likelihood scenes of the lumen and computed the probability of the objects actually belonging to the lumen class. The false positives were removed depending on the area of the predicted lumen region. Moreover, the non-gland regions were removed by ensuring that the detected regions were surrounded by cytoplasm regions. After the detection of lumen objects, gland segmentation was carried out by high-level information using level sets. It made use of neighboring pixels during evolution to form the target boundary. Curve evolution was done by the likelihood scene of the nucleus and the initial contour was initialized to the detected lumen object. The curve was evolved outward from lumen object within the likelihood scene of nucleus and thus the gland and nuclei boundaries were extracted. Similar to gland segmentation, nuclear segmentation was done using low-level and high-level information. From the gland and nuclei boundaries, the authors calculated 16 morphological features. Further, 51 graph-based features were taken from Voronoi diagrams, minimum spanning tree and Delaunay triangulation using the centroids of nuclei. The feature set so obtained was reduced using graph embedding, a non-linear dimensionality reduction method and then support vector machine was used to classify the glands into benign or grade 3/4.

Nguyen et al. [5] proposed 3 different techniques for segmentation and classification of prostate glands. In the lumen based grading method, glands were segmented following which 22 structural contextual features were extracted from each gland. An svm classifier was trained to remove the noisy regions from the extracted glands and then the 22 dimensional vector was used in an svm to classify glands into benign, grade 3 or 4. In the nuclei-based method, the authors first detected the tissue components of the image that is, nuclei, lumen and stroma. As nuclei are mostly circular and have small sizes, they used the radial-based symmetry to detect

nuclei. Textural features were then computed in the neighborhood of nuclei and a svm was used to differentiate between epithelial and stromal nuclei. Stromal nuclei were then discarded. Lumen and stroma were detected using k-means. The nuclei-lumina graph which depicts the relationship between nuclei and nuclei, and nuclei and lumina was constructed using two algorithms: nuclei-nuclei link creation and nuclei-lumina link creation. Around every nucleus, a conical region was formed and the nucleus closest to the central nucleus in every conical region was marked. If the line joining the two nuclei did not intersect stroma, then nuclei-nuclei link was created. For the nucleus-lumen link creation, a conical region was made around a lumen pixel lying on the lumen boundary. All the nuclei falling in the region were considered for the nucleus-lumen link creation. All the links obtained so formed the edges of the nuclei-lumina graph. To remove bad links, normalized cut method was used. The authors also computed the gland score which measures the closeness to closed chain structure. In grade 3 glands, the nuclei tend to be form more closed chain structure leading to a high gland score. On the other hand, nuclei in grade 4 glands are randomly distributed because of which they have a low gland score. The authors then fused the above two methods that is, lumen based and nuclei based and observed a higher accuracy than obtained in the two methods independently.

Nguyen et al. in [6] used k-means clustering algorithm to segment the different components of the gland (lumen, cytoplasm, nuclei etc.). A connected component algorithm on nuclei pixels and lumen pixels generates nuclei objects and lumen objects, respectively, which are used for segmentation. Nuclei-lumen association (NLA) algorithm, associates appropriate nuclei with each lumen to create a gland segment. Nuclei are searched along the normal direction of the lumen boundary contour. For gland classification the differences in structures of the three classes (artefact, normal gland and cancer gland) are used to make four sets of structural features, including 19 features, for each gland which are as follows. Set 1 (8 nuclei features), Set 2 (6 cytoplasm features), Set 3 (3 lumen shape features) and Set 4 (2 global features). The following 3 features as Neighbourhood crowdedness, Shape similarity and Size similarity are also taken for each gland segment to capture the contextual information. Finally, each gland segment can be represented by a full feature vector of dimensionality 22 (19 + 3) and for the classification purpose a SVM classifier (linear kernel, C = 1) with this feature vector is used.

Chapter 3

Convolutional Neural Network

Convolutional Neural Network (CNN) [12] is a biologically-inspired variant of MLP which finds its use mainly in the fields of image and video processing. They deploy a local connectivity pattern between neurons of adjacent layers and thus exploit the spatially-local correlation between them i.e. inputs of units in layer m are from a subset of units in layer $m-1$. This can be shown graphically as follows:

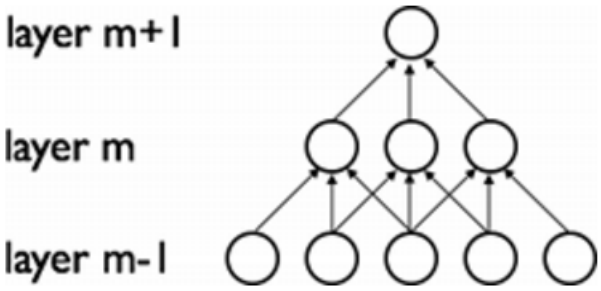


Figure 3.1: Neurons in different layers

In addition, in CNNs, each filter h_i is replicated across the entire visual field. These replicated units share the same parameterization (weight vector and bias) and form a feature map. Optimization algorithms can still be used to learn the shared parameters by only doing minor adjustments to the original algorithm. For e.g. In case of Gradient Descent, the gradient of a shared weight is simply the sum of the gradients of the parameters being shared.

Such a method of replicating units results in the detection of robust features regardless of their position in the visual field. Additionally, due to the filters sharing weights, the efficiency, both in terms of learning time as well as device memory, increases greatly due to the reduction in the number of free parameters being learnt. The constraints on the model enables CNN to

achieve better generalization on vision problems.

CNN consists of various layers. Below are presented three of these kinds:

- Convolutional layer: As the name suggests, this is the foremost layer in any CNN. It consists of a rectangular grid of neurons which is used to perform convolution on the image and the weights specify the convolutional filter. A feature map is obtained by repeated application of a function across sub-regions of the entire image, in other words, by convolution of the input image with a linear filter, adding a bias term and then applying a non-linear function. If we denote the k-th feature map at a given layer as , whose filters are determined by the weights and bias , then the feature map is obtained as follows (for non-linearities):

$$h_{ij}^{k} = tanh((W^{k} * x)_{ij} + b_{k}) \tag{3.1}$$

  Convolution can be denoted in the following way:

$$o[m,n] = f[m,n] * g[m,n] = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f[u,v]g[m-u,n-v] \tag{3.2}$$

- Pooling layer: Each convolutional layer may be followed by a pooling layer where the maximum or average of the block is taken to produce a single output from that block. This reduces the possibility of over-fitting and also ensures that the result becomes translation invariant.

- Fully connected layer: After the stack of several convolutional and pooling layers, comes the fully connected layer. It takes all the neurons in the previous layer and connects to every neuron it has. There cannot be another convolutional layer beyond the fully connected layer and it is the one responsible for classifying the data on the basis of the high-level features learnt by the previous filters.

CNNs consists of multiple convolutional layers which are collection of small filters that take only a small portion of the original image as input and process over it. The outputs of these filters of a convolutional layer are pooled to have a better representation of the given image and passed as input to the next convolution layer.

To reduce the possibility of overfitting and variances different techniques are applied:

- Dropout: In this method at any stage some of the nodes are randomly dropped which results in a reduced network which is trained in that stage. The removed nodes are later

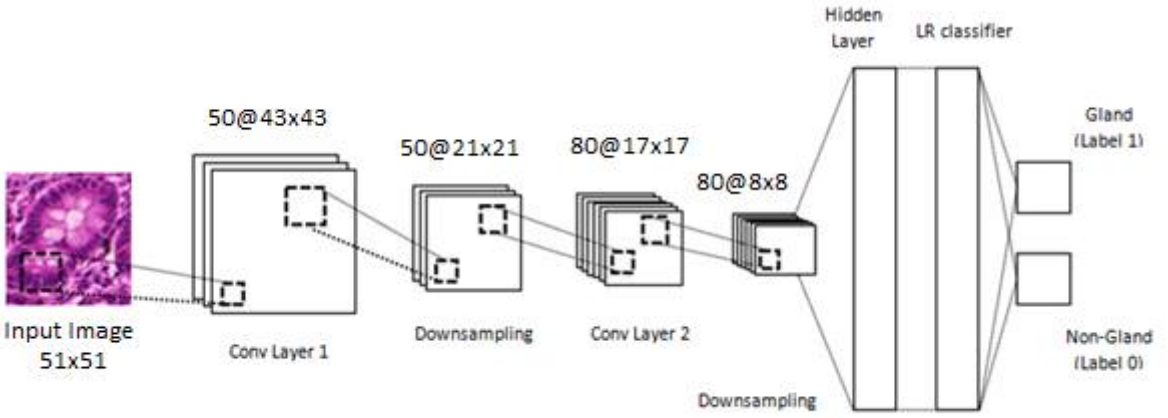


Figure 3.2: CNN Architecture

added back to the network. Fully connected layers often result in overfitting, thus by reducing the network dropout decreases the chances of overfitting.

- Rectified Linear Unit (ReLU): This is a layer of neurons that is used when we want to apply non-saturating activation function

$$f(x) = max(0, x) \tag{3.3}$$

It works by increasing the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. Usage of ReLU is preferable compared to other functions because it results in the neural network training several times faster, without making a significant difference to generalization accuracy.

The function of a filter is to extract the different aspects of the input image. Thus, these filters may encode colors or directions moving on to the basic grid or spot textures. As the layers keep increasing, the complexity of the features extracted by the filters of that layer increases. The higher level filters may start recognizing small objects present in the images.

Now these learned features can be passed through a hidden layer with a linear or nonlinear activation function. The output is used as input to a logistic regression model, multi-layer perceptron or other classifiers to solve binary or multi-class classification problems.

Chapter 4

Color Normalization

We used the color normalization technique proposed in [11]. According to the Beer-Lambert law, the attenuation of transmitted light through the tissue specimen is related to the concentration of stains. According to the law:

$$I = I_0 e^{-WH} \tag{4.1}$$

where, I is a $m \times n$ matrix with m being the number of channels and n being the number of pixels of RGB intensities in an acquired bright field microscopic image, I_0 is the illuminating light on the sample, W is called the stain colour matrix whose columns represent RGB colour of each stain. Its dimensions are $m \times r$ (r being the number of stains). Lastly, H is the stain concentration matrix, whose rows represent total amount of stained tissue. Its dimensions are $r \times n$.

Let the relative optical density be denoted as V which is given by:

$$V = logI_0 - logI \tag{4.2}$$

Hence, equation (4.1) becomes,

$$V = WH \tag{4.3}$$

The approaches to estimate W and H varies for different stain separation methods. For example, colour de-convolution [7] method achieves this by estimating W experimentally and subsequently using pseudo inverse transform to generate the value of H. On the other hand, blind colour decomposition methods estimate both unknown W and H [8]. Another method called Non-negative matrix factorization (NMF) is a good matrix factorization technique [9] which can be used for stain separation.

$$\underset{W,H}{min} \frac{1}{2} \|V - WH\|_F^2, \text{ such that } W, H \geq 0 \tag{4.4}$$

By including L1 sparseness regularization on stains H improves the above NMF objective due to the fact that a type of stain is only bound to certain biological structures. For example, in H&E, H binds to the nuclei and E to the cytoplasm and stroma, hence when both stains are separated, they should look sparse. This supports the idea of including sparsity constraints on H.

$$\underset{W,H}{min}\frac{1}{2}\|V - WH\|_F^2 + \lambda \sum_{j=1}^{r} \|H(j,:)\|_1, \text{ such that } W, H \geq 0 \tag{4.5}$$

This optimization problem is solved by a method (equation 4.5) that is related to the dictionary learning objective. Additional non-negative constraints are forced on dictionary atoms W and coefficients H. This method works by alternating between W and H which optimizes one set of parameters whilst keeping the other fixed. The publicly available SPAMS toolbox [10] is used in the implementation. The colour appearance of source W_s is changed to be as target W_t after an accurate separation of colour appearance and staining concentration for both source and target images. After this the source is combined with a normalized version of the stain concentration H_s of source image, to generate the normalized source image. This colour normalization technique changes only the stain colour appearance whilst preserving structures (as the relative stain concentration is not changed). The detailed algorithm is shown below.

Algorithm 1: Color normalization [11]

Input : source and target image, number of stains r, sparsity regularization parameter λ (default λ = 0.1)

Output: normalized source image

1 Convert source and target in to optical densities using eq.(4.2)

2 Source stain separation: $V_s = W_s H_s$ using eq.(4.5)

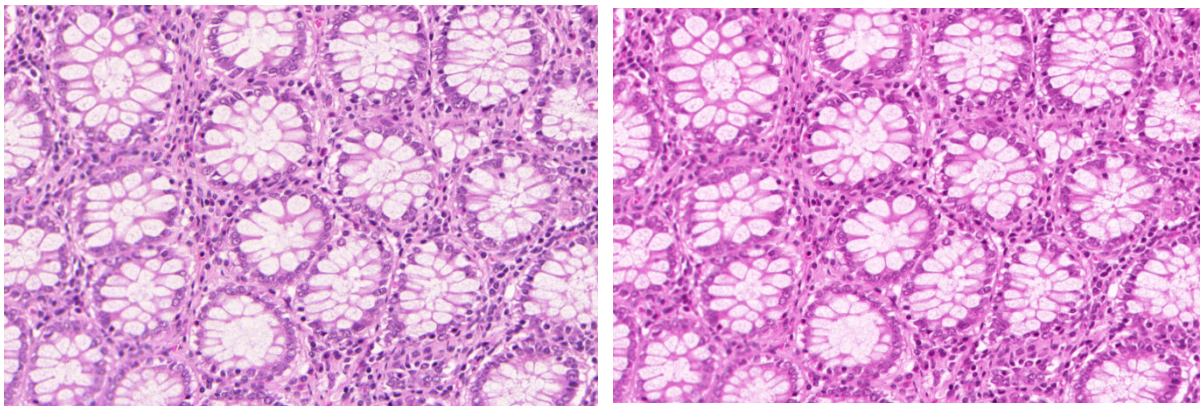
3 Target stain separation: $V_t = W_t H_t$ using eq.(4.5)

4 Adjust the dynamic range of H_s to be same as that of H_t to form the normalized source stains H_{snorm}. The dynamic range is robustly estimated with a pseudo maximum (99%).

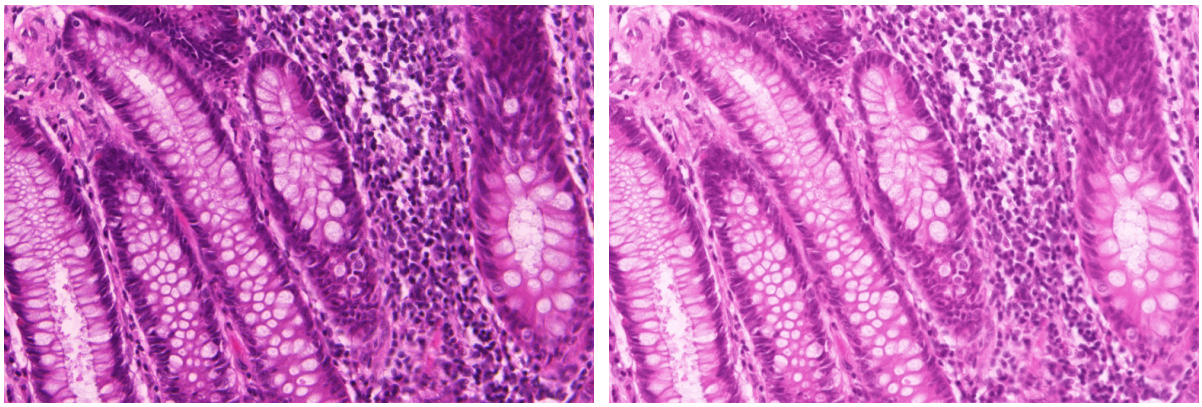
5 Colour exchange: $V_{snorm} = W_t H_{snorm}$

6 Project the V_{snorm} into RGB colour space using eq.(4.2) to get the normalized source image

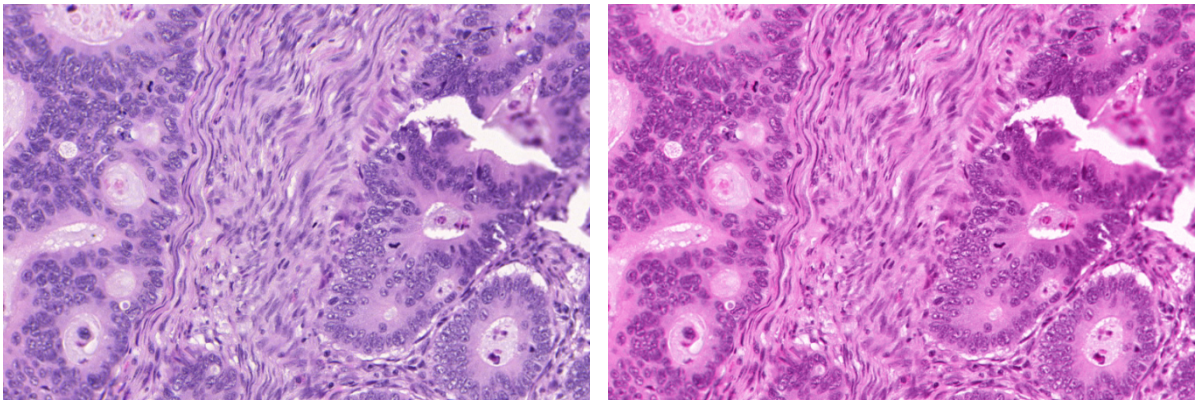
The following figure shows the result of color normalization on selected images.



(a) Before (b) After



(c) Before (d) After



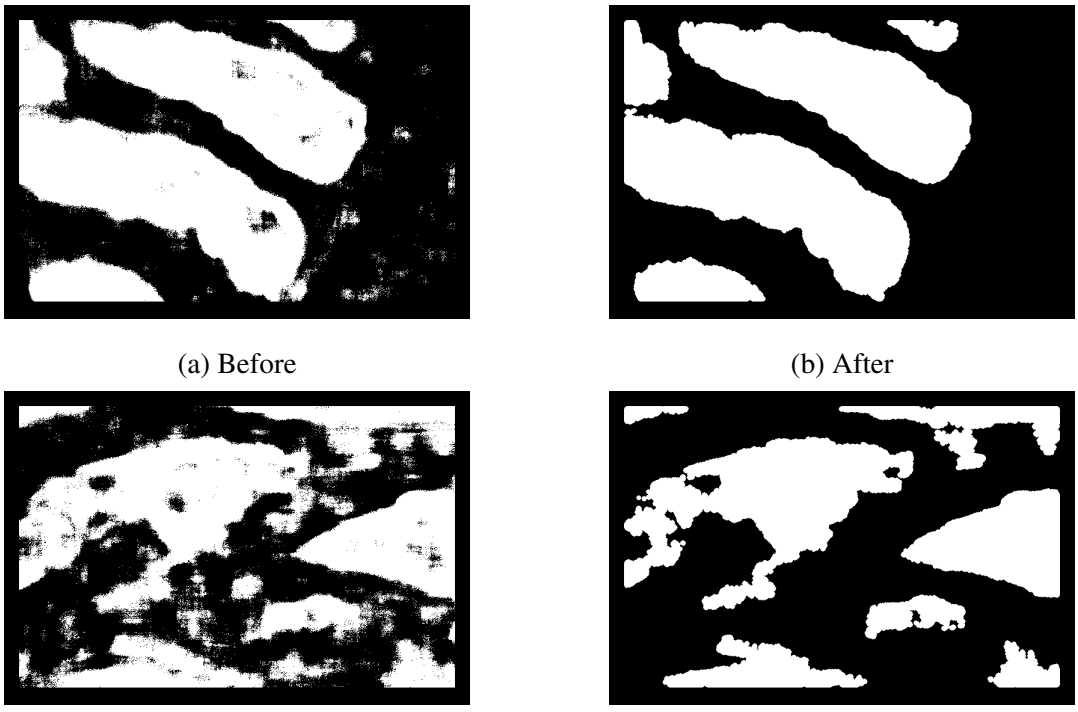
(e) Before (f) After

Figure 4.1: Images before and after color normalization

Chapter 5

Post Processing

The binary maps as predicted by the CNN were subjected to the post processing code. The binary image was opened with a disk structuring element of size 4. The holes in the resulting image were then filled. The connected components which had fewer than 1000 pixels were removed and the resulting image was then saved. Figure 5.1 shows the binary maps after post processing is done.



(a) Before (b) After

(c) Before (d) After

Figure 5.1: Images before and after post processing

Chapter 6

Evaluation Metrics

The efficiency of the proposed approach was measured using indices like F1 score, Dice index and Jaccard index both at the object level as well as the pixel level as given in [15].

The ground truth for each segmented object is the object in the manual annotation that has maximum overlap with that segmented object.

A segmented glandular object that intersects with at least 50% of its ground truth will be considered as true positive, otherwise it will be considered as false positive. A ground truth glandular object that has no corresponding segmented object or has less than 50% of its area overlapped by its corresponding segmented object will be considered as false negative.

Let TP be the number of true positives, FP be the number of false positives and FN be the number of false negatives.

For gland detection we have used F1-score which is defined by,

$$F1score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6.1}$$

where, $Precision = TP/(TP + FP)$ and $Recall = TP/(TP + FN)$.

We also calculated the values of two other evaluation metrics namely Dice index and Jaccard index. Both of these were also computed at pixel level as well as object level.

At pixel level, Dice index is calculated as

$$Dice(G, O) = \frac{2|G \cap O|}{|G| + |O|} \tag{6.2}$$

and Jaccard index as

$$Jaccard(G, O) = \frac{|G \cap O|}{|G \cup O|} \tag{6.3}$$

where, $|.|$ denotes set cardinality.

For object level segmentation accuracy, we are concerned with, how well each segmented object overlaps with the ground truth objects, and how well each ground truth object overlaps the segmented objects. Let O_i denote the set of pixels of the i^th segmented object in O and G_i denote the set of pixels of ground truth objects in G that intersect O_i. Further, let $\tilde{G}_i$ denote the set of pixels of the i^th ground truth object in G and $\tilde{O}_i$ denote the set of pixels of segmented objects in O that intersect $\tilde{G}_i$.

The object-level dice index is formulated as

$$Dice_{object}(G, O) = \frac{1}{2}[\sum_{i=1}^{n_O} \omega_i Dice(G_i, O_i) + \sum_{i=1}^{n_G} \tilde{\omega_i} Dice(\tilde{G}_i, \tilde{O}_i)] \tag{6.4}$$

and object-level jaccard index as

$$Jaccard_{object}(G, O) = \frac{1}{2}[\sum_{i=1}^{n_O} \omega_i Jaccard(G_i, O_i) + \sum_{i=1}^{n_G} \tilde{\omega_i} Jaccard(\tilde{G}_i, \tilde{O}_i)] \tag{6.5}$$

where, $\omega_i = |O_i| / \sum_{i=1}^{n_O} O_j$ and $\tilde{\omega_i} = |\tilde{G}_i| / \sum_{i=1}^{n_G} \tilde{G}_j$, n_O and n_G are the number of segmented objects and ground truth objects respectively.

All these scores have a range from 0 to 1 where 1 implies perfect segmentation.

Chapter 7

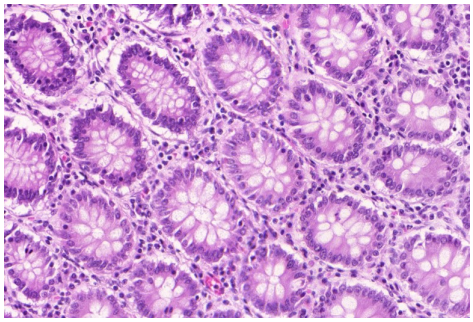
Proposed Methodology

We used the Warwick-QU dataset from the Gland Segmentation challenge contest (GlaS) at MICCAI 2015 [16] for conducting the experiments which had 37 benign and 48 malignant H&E stained biopsy slides of colon tissue in the training set and 37 benign and 43 malignant in the test set. These test images were further labelled A type which were 60 in number and the remaining 20 belonged to B type. Each image was a RGB image of size 775x522. Along with these we also had the annotated images where the background was black while the different glands were marked with different grey values.

Fig. 7.1(a) shows a sample image of the dataset. Pink portion of the image is the stroma which forms the background. The white portion in the centre of the gland is called the lumen which is surrounded by purple coloured cytoplasm. The gland boundary is marked by the round dark blue coloured epithelial nuclei. The other round objects in the stroma are called the stromal nuclei.

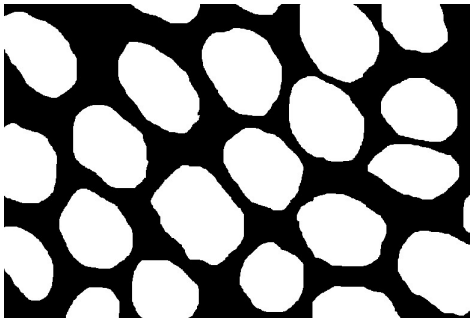
To begin with we first generated the binary image (See Fig. 7.1(c)) using the annotated image (See Fig. 7.1(b)) where every pixel with intensity greater than zero was marked as 1. Thus all the glands were represented by white colour (label 1) while the background as black (label 0). All the training and test images were color normalized using the target image which is shown in Figure 7.2. Figure 7.1(d) shows the color normalized version of the image shown in Fig. 7.1(a).

Next we loaded the complete training data set using the load_data function in which we provided the name of the appropriate directories. This function returned a list of all the images. This function can be modified to accommodate various types of datasets, like RGB images, H&E mat files and both H&E+RGB images. The implementation of the same is in Appendix I.

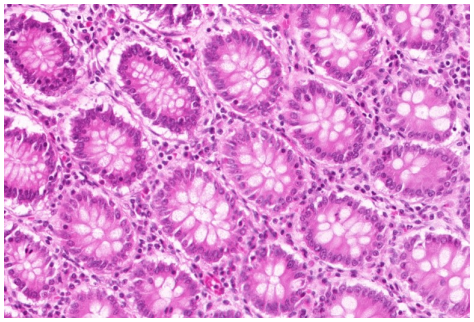


(a) Original Image

(b) Annotated Image



(c) Binary Image



(d) Color Normalized Image

Figure 7.1: Sample image from Warwick-QU Dataset

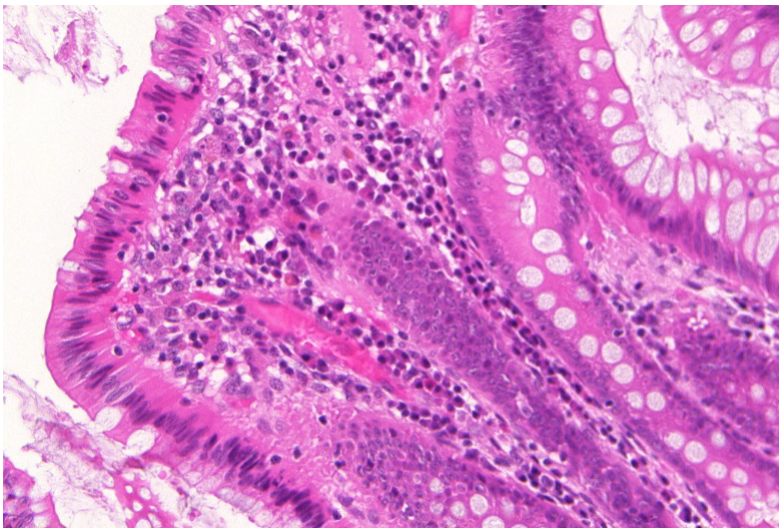


Figure 7.2: Target image used in color normalization

We used the load_img function to extract n (where n is an even number provided by the user) number of 51x51 sized patches from an image. Here a list of the patch, gland or non-gland value and the (x, y) coordinates of the center pixel are returned. The patches were chosen such that half of them belong to the gland class and half of them belong to the non-gland class. The patches were randomly chosen from the image by using a random number generator. Appendix I shows how the implementation of the function is done.

The class LeNetConvPoolLayer is used which was available publicly in the deeplearning website [12]. This class implements the convolution layers with downpooling. We added the dropout functionality which was not available earlier in it.

The Hiddenlayer class was directly taken from the deeplearning website [13]. This is an implementation of the multi-layer perceptron. This implementation already had 'tanh' activation. We added the ReLU activation unit and dropout.

The LogisticRegression class was also taken from the deeplearning website [14]. This performs logistic regression using softmax decision rule.

The proposed CNN architecture consisted of two alternatingconvolution and max-pooling layers. The upper layers were fully-connected and imitated a traditional multi-layer perceptron (hidden layer + logistic regression). The input to the first fully-connected layer was the set of all features maps at the layer below. From an implementation point of view, this means lower-layers operated on 4D tensors which were then flattened to a 2D matrix of rasterized feature maps, to be compatible with the MLP implementation.

After all the iterations, the weights of the filters which gave the lowest validation error were saved in a pickle file for further use in making the binary maps of the test images.

Next the post processing code was run and the resultant images were then used to calculate the detection and segmentation accuracy of the glands using three evaluation metrics at both pixel level and object level. These are F1-score, Dice index and Jaccard index.

Chapter 8

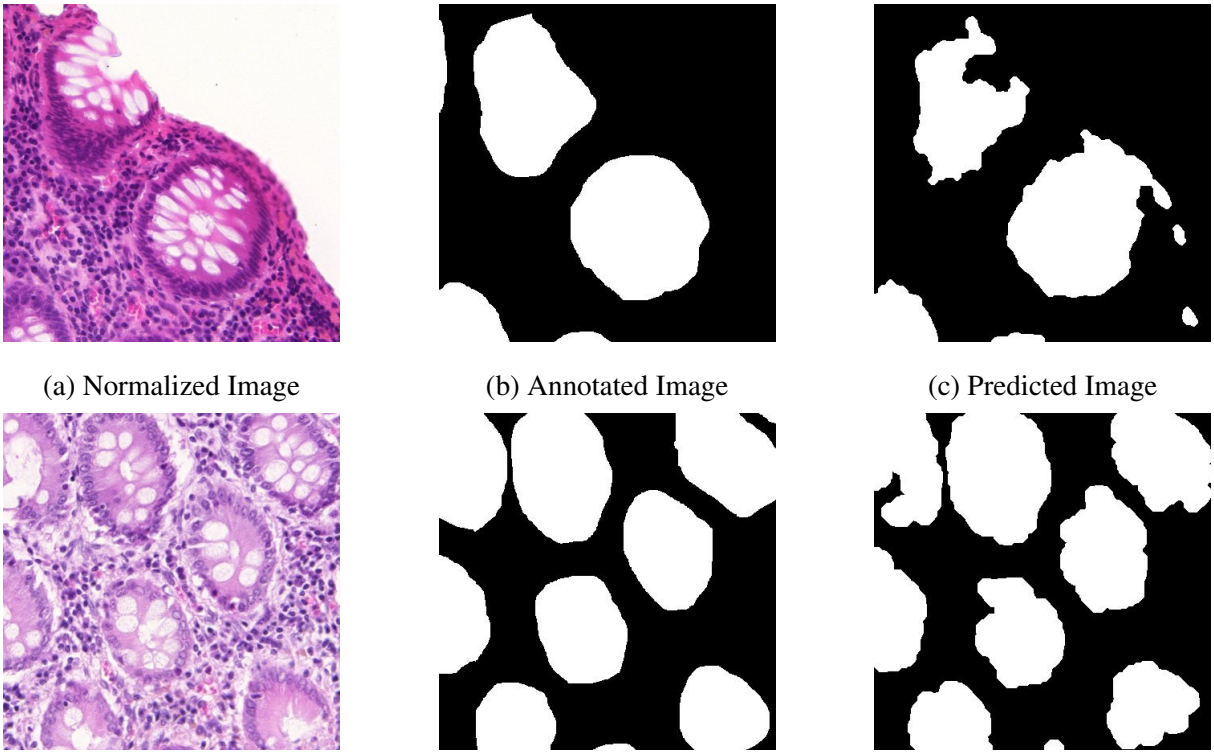
Experiments

Various experiments were conducted to determine the best possible parameters for CNN architecture.

Out of the 37 benign training images, 30 were used for the training and validation and the rest 7 images were used for testing. For predicting the class of each pixel we require the local information to be passed in to the CNN. A patch of size 31x31 was taken which is used to predict the class of the centre pixel of that patch. So we randomly selected a fixed number of patches from every image such that the ratio of gland vs non-gland centre pixels is 1:1. For prediction we used a CNN which had 2 convolutional layers, 1 hidden layer and a logistic regression layer. The size and the number of filters were experimented upon to achieve the minimum test and validation error. Pooling was done which reduced the output of the convolution layer by a factor of 2. Initially, we extracted 11760 patches from the training set and 2640 from the validation and test set. For training and testing we used batch processing with a batch size of 420 and the number of epochs was set to 200. The learning rate was set to 0.001 without the use of dynamic learning rate which yielded a validation error of 15.07% and test error of 25.54%. To provide more first level features we increased the number of filters in the first convolutional layer to which resulted in 15.34% validation error and 25% test error.

These same experiments were repeated using a dynamic learning rate initialized at 0.001 with a decrement value of (0.25*current learning rate) with 30 filters in first layer and 50 filters in the second layer achieved a validation error of 12.99% and 16.83% test error.

We played with the parameters of the CNN to achieve higher accuracy. Finally, a cnn with number of filters 50 and 80 in the first and second convolutional layer with sizes 3x3 and 5x5 respectively and a larger dataset where 117600 patches from the training set and 11760 patches



(a) Normalized Image (b) Annotated Image (c) Predicted Image

(d) Normalized Image (e) Annotated Image (f) Predicted Image

Figure 8.1: Segmentation results on only benign data

from both validation and test set was used. This architecture yielded 11.07% as the validation error and 13.62% as the test error. Figure 8.1 shows the predicted images obtained from this experiment.

After getting the assurance that the CNN was able to learn and segment glands out of the benign biopsy slides, we tried to train the CNN on both benign and malignant data. We used 60 images for training and validation part and kept 12 for testing. The same parameters and CNN architecture which gave the best results on the benign only dataset was used. Because of the more degradation in the malignant tissues, the validation as well as test error increased to 19.27% and 26.55%. The values of different parameters viz., number and size of filters, learning rate etc., were changed to obtain an optimal error rate. Choosing the number of filters in the two conv layers as 120 and 80 respectively, the error came out to be 21.78% and 28.71% while for 150 and 80 filters, we obtained 20.09% validation and 27.38% test error. Using 20 and 40 as the number of filters yielded a validation and test error of 20.08% and 27.49% respectively. The learning rate was decreased to 0.01 and the decrement value set at 0.0025. With the number of filters as 50 and 80, 14.69% and 23.83% were the resulting validation and test error. The decrement in the learning rate was further reduced to 0.1 resulting in 14% validation and 24%

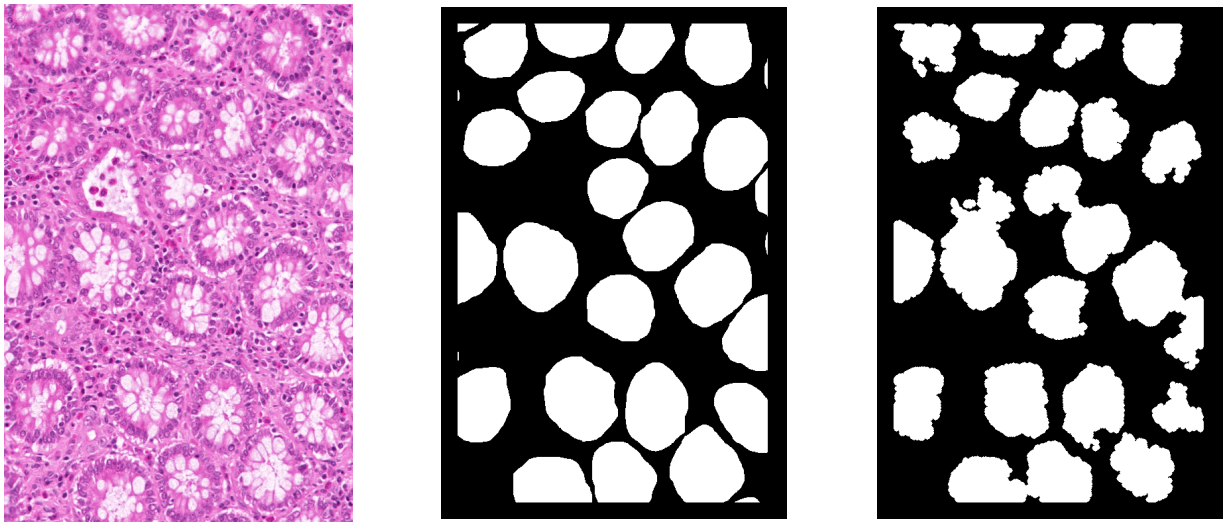
test error. Keeping the learning rate and decrement same as above but number of filters as 40 and 60, the validation and test error turned out to be 13% and 23% respectively. To increase the complexity of the features learnt by the CNN, another convolutional layer was added with the number of filters as 50, 80 and 50 each of size 3x3. The learning rate and its decrement value used was 0.01 and 0.001 respectively. 18% and 27% were the validation and test errors obtained in this experiment.

Due to the use of different microscopes/scanners or differences in tissue preparation, there might be color variations in the histologic images. As the CNN filters might also learn the color of the various entities in the slides, color normalization becomes an important technique in standardizing the color variations among the images. The training and test images were color normalized using the approach proposed in [11]. A new conv net was trained on these color normalized images with learning rate 0.0001 and decrement value 0.00025, 80 filters in the first conv layer of size 3x3 and 100 filters also of 3x3 size in the second layer. Testing on the normalized images yielded a validation error of 24.6% and test error of 27.4%. Decreasing the learning rate to 0.001 and changing the number of filter to 120 and 80 in the first and second layer respectively resulted in error rate of 21.78% in the validation set and 27.81% in the test set. Using 50 filters of size 5x5 in the first and 80 of size 5x5 in the second layer yielded 20.09% validation error and 27.38% test error. We then used 0.01 as the learning rate and 0.1 as the decrement value of the learning rate. The number of filters were changed to 50 of size 5x5 and 80 of size 3x3 in the two layers. Further, dropout was also added in the two convolutional layers as well as in the hidden layer. The values used were 0.1, 0.1 and 0.5 respectively. Relu activation was also used in this architecture. This experiment returned 15% validation error and 23% test error. As the complexity of the features learnt increases as the number of layer increases, hence more number of filters are put in the later layers. Also, the size of the filters in the first layer is more than in the second layer because more information about the surrounding is required for the identification of glands.

As the glands especially malignant are bigger in size, we decided to increase the input patch size to 51x51. Keeping the other parameters same as the previous experiment improved the result by a considerable margin. The validation error fell to 7.09% and test error to 20.3%. Another experiment was conducted with the same parameters, the only difference being the training and testing data were in the ratio 1:1. Not much difference however was observed as the validation error turned out to be 6.8% and test error as 21.75%.

Next we used the whole Warwick-QU dataset consisting of 85 (37 benign and 48 malignant) training and 80 (37 benign and 43 malignant) test images. The size of the 50 filters in the first layer was 9x9 while those of the 80 in the second was 5x5. With learning rate set at 0.01 and decrement at 0.4 and using 51x51 patch of the RGB values of the image, we got 11.97% validation error and 14.28% test error. Instead of the RGB values, the stain maps of Hematoxylin and Eosin were used with the same parameters resulting into a validation error of 12.31% and test error of 16.37%. Finally, we combined the RGB values with the H&E stain maps and used all the five of them as input to the CNN. 12.03% and 15.12% were the validation and test errors obtained in this case. The following three figures show the predicted maps generated after taking RGB, H&E and RGB+H&E values as the input.

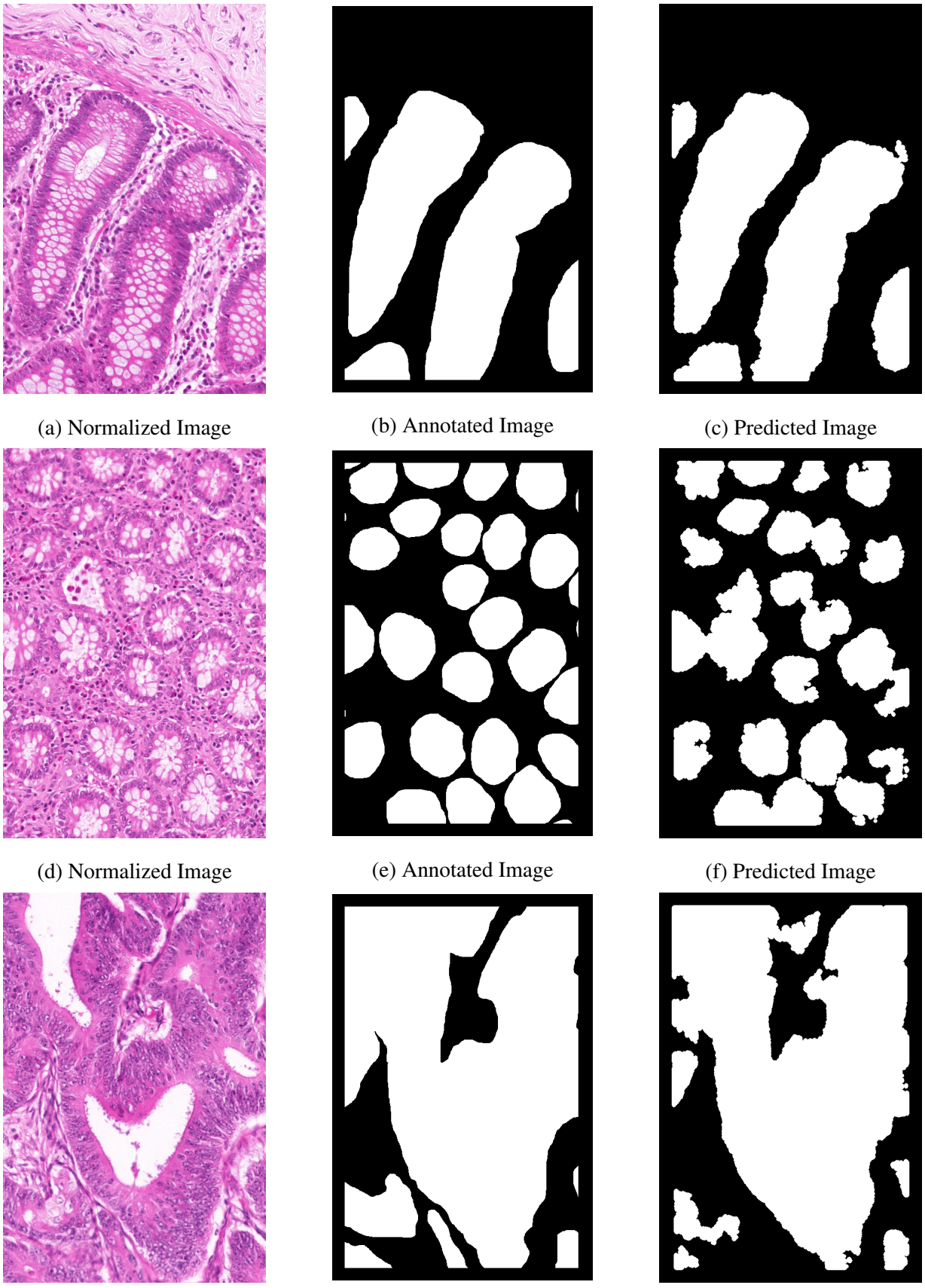
(a) Normalized Image (b) Annotated Image (c) Predicted Image



(d) Normalized Image (e) Annotated Image (f) Predicted Image

(g) Normalized Image (h) Annotated Image (i) Predicted Image

Figure 8.2: Segmentation results using RGB channels



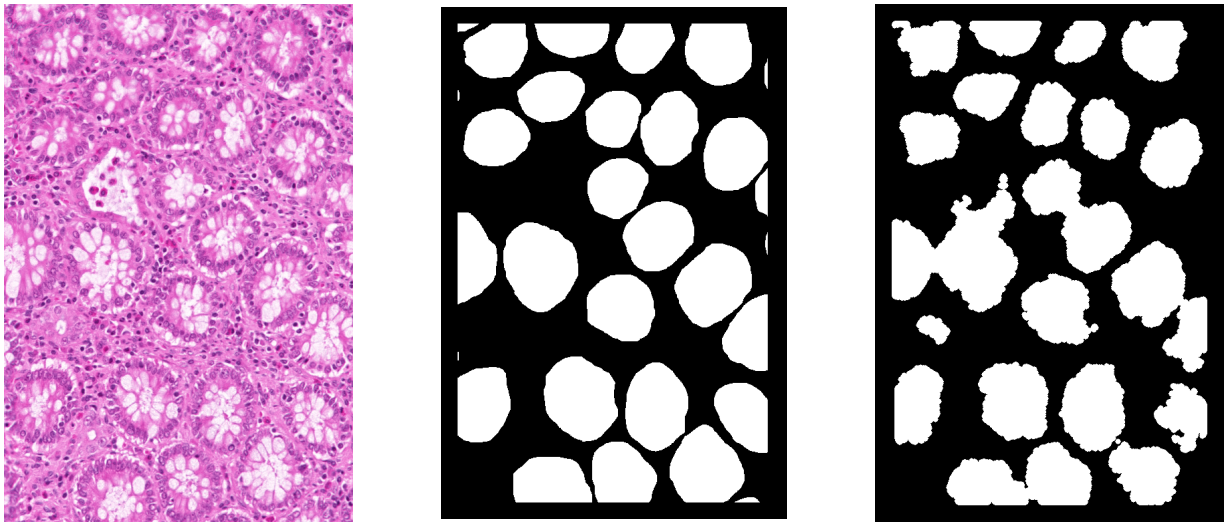
(a) Normalized Image (b) Annotated Image (c) Predicted Image

(d) Normalized Image (e) Annotated Image (f) Predicted Image

(g) Normalized Image (h) Annotated Image (i) Predicted Image

Figure 8.3: Segmentation results using H&E channels

(a) Normalized Image (b) Annotated Image (c) Predicted Image



(d) Normalized Image (e) Annotated Image (f) Predicted Image

(g) Normalized Image (h) Annotated Image (i) Predicted Image

Figure 8.4: Segmentation results using H&E and RGB channels

Chapter 9

Results

This chapter presents the results of the experiments discussed above in tabular format

The values of the different parameters used while experimenting with the 30 train and 7 test benign images have been tabulated in Table 9.1. It also contains the validation and test error obtained in each case.

Table 9.1: Results on only benign data

S.No	Dataset size (Number of patches)			Convolutional Layer 1 Filters		Convolutional Layer 2 Filters		Learning Rate	Decrement in Learning Rate	Error (in %)	
	Training	Validation	Test	Number	Size	Number	Size			Validation	Test
1.	11760	2940	2940	20	5x5	50	5x5	0.001	-	15.07	25.54
2.	11760	2940	2940	30	5x5	50	5x5	0.001	-	15.34	25.00
3.	117600	2940	2940	30	3x3	50	5x5	0.001	0.25	12.99	16.83
4.	117600	29400	29400	30	3x3	50	5x5	0.001	0.25	21.50	30.80
5.	117600	11760	11760	30	5x5	80	5x5	0.001	0.25	13.66	14.44
6.	117600	11760	11760	30	3x3	80	5x5	0.001	0.25	12.67	14.78
7.	117600	11760	11760	20	3x3	50	5x5	0.001	0.25	14.47	14.51
8.	117600	11760	11760	20	5x5	50	5x5	0.001	0.25	14.37	15.51
9.	117600	11760	11760	100	3x3	80	5x5	0.001	0.25	11.26	14.50
10.	117600	11760	11760	120	3x3	80	5x5	0.001	0.25	11.82	14.97
11.	117600	11760	11760	50	3x3	80	5x5	0.001	0.00025	11.07	13.62

The following Table 9.2 shows the results achieved on considering both the benign and malignant images that is, 60 train images and 12 test images. The batch size used here was 480. All these experiments have been conducted on non-color normalized dataset.

Table 9.2: Results on non-normalized benign and malignant data

S.No	Convolutional Layer 1 Filters		Convolutional Layer 2 Filters		Convolutional Layer 3 Filters		Learning Rate	Decrement in Learning Rate	Error (in %)	
	Number	Size	Number	Size	Number	Size			Validation	Test
1.	50	3x3	80	5x5	-	-	0.001	0.00025	19.27	26.55
2.	20	3x3	40	5x5	-	-	0.001	0.00025	20.08	27.49
3.	120	3x3	80	5x5	-	-	0.001	0.00025	21.78	27.81
4.	150	3x3	80	5x5	-	-	0.001	0.00025	20.09	27.38
5.	50	3x3	80	5x5	-	-	0.01	0.0025	14.69	23.83
6.	50	3x3	80	5x5	-	-	0.01	0.1	14.00	21.00
7.	40	3x3	60	5x5	-	-	0.01	0.1	13.00	23.00
8.	50	3x3	80	3x3	50	3x3	0.01	0.001	18.00	27.00

Table 9.3 consists of the results of the experiments done on the color normalized dataset. The number of training and testing images used are 60 and 12 respectively. The remarks column also shows the additional action undertaken in these experiments. For experiment number 4, 5 and 6, a dropout of 0.1 in the first and second convolutional layer, and 0.5 in the hidden layer was added. ReLU activation was used in the hidden layer. Instead of 31x31 size patch, 51x51 sized input is provided to the cnn in experiments 5 and 6. Moreover, in the last experiment, the size of the training and test dataset is equal.

Table 9.3: Results on color normalized benign and malignant data

S.No	Convolutional Layer 1 Filters		Convolutional Layer 2 Filters		Learning Rate	Decrement in Learning Rate	Error (in %)		Remarks
	Number	Size	Number	Size			Validation	Test	
1.	50	5x5	80	5x5	0.001	0.00025	20.09	27.38	-
2.	80	3x3	100	3x3	0.001	0.00025	24.60	27.40	-
3.	120	3x3	80	3x3	0.001	0.00025	21.78	27.81	-
4.	50	5x5	80	3x3	0.01	0.1	15.00	23.00	Dropout and ReLU added
5.	50	5x5	80	3x3	0.01	0.1	7.09	20.30	Input patch size of 51x51
6.	50	5x5	80	3x3	0.01	0.1	6.80	21.75	Equal Training and test data

The final experiments were conducted on Warwick-QU Dataset consisting of 85 training and 80 test images. The values of the different parameters used in this architecture are presented in Table 9.4.

Table 9.4: Parameters used in the experiments on Warwick-QU Dataset

Patch Size	Convolutional Layer 1			Convolutional Layer 2			Hidden Layer		Learning Rate	Decrement in Learning Rate
	Number	Size	Dropout	Number	Size	Dropout	Dropout	Activation		
51x51	50	9x9	0.1	80	5x5	0.1	0.5	ReLU	0.01	0.4

Table 9.5 shows the validation and test error as well as the object level and pixel level mean and standard deviation values of the three evaluation metrics, namely, F1-score, Dice index and Jaccard index.

Table 9.5: Results on Warwick-QU Dataset

Input	Error (in %)		F1-score				Dice Index				Jaccard Index			
			Pixel Level		Object Level		Pixel Level		Object Level		Pixel Level		Object Level	
	Validation	Test	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
RGB	11.97	14.28	0.836	0.087	0.633	0.187	0.836	0.087	0.707	0.149	0.727	0.119	0.595	0.171
H&E	12.31	16.37	0.826	0.119	0.605	0.223	0.826	0.119	0.676	0.185	0.718	0.147	0.565	0.260
RGB+H&E	12.03	15.12	0.841	0.0857	0.622	0.191	0.841	0.085	0.704	0.159	0.734	0.120	0.594	0.185

Chapter 10

Conclusions

In this work, we implemented a novel deep learning architecture using convolutional neural network to segment glands from the biopsy slides. We have used two convolutional layers, one hidden layer and a logistic regression classifier to do the final classification. From the experimental results, we can conclude that the errors obtained depend upon the parameters of the architecture of CNN. Moreover, techniques like color normalization, dropout and ReLU activation further improve the result. Results from the various experiments show the efficacy of the proposed approach over the other state-of-the-art techniques. Furthermore, by changing a few parameters, this method can be used for the segmentation of any type of gland.

Instead of a CNN, a multiscale convolutional neural network can also be used where a larger cnn can learn the surrounding features and the smaller one can concentrate upon the internal characteristics of the glands. This method can be extended to classification of the segmented glands into benign or malignant (and further into grade 3, 4 or 5).

.1 Appendix I

The following code shows the implementation of load_data function.

```
def load_data(xdirname , ydirname):
    print '...loading data'

    mat = []

    included_extensions = ['bmp']
    filelist = [fn[:-9] for fn in os.listdir(xdirname) if any([fn.endswith(
    ext) for ext in included_extensions])]
    iext = '_norm.bmp'    #normalized image
    hext = '_H.mat'       #H stain map
    eext = '_E.mat'       #E stain map
    aext = '_bin.bmp'     #binary image

    for fname in filelist:

        xname = xdirname + fname
        iname = xname + iext
        hname = xname + hext
        ename = xname + eext
        aname = ydirname + fname + aext

        H = scipy.io.loadmat(hname)     #Load H stain map
        H = H['H']
        E = scipy.io.loadmat(ename)     #Load E stain map
        E = E['E']

        xim = Image.open(iname)
        xpix = xim.load()       #Load normalized image
        yim = Image.open(aname)
        ypix = yim.load()       #Load binary image

        width,height = yim.size
        red = np.zeros((height,width))
        green = np.zeros((height,width))
        blue = np.zeros((height,width))
```

```
35          fgrnd = np.zeros((height,width))
36
37          for x in range(width):
38              for y in range(height):
39                  red[y,x] = xpix[x,y][0]
40                  green[y,x] = xpix[x,y][1]
41                  blue[y,x] = xpix[x,y][2]
42                  if ypix[x,y]:
43                      fgrnd[y,x] = 1.0
44
45          img=np.array([red,green,blue,H,E,fgrnd])        #5 input channels and
    ground truth
46          mat.append(img)
47
48      return mat
```

The following code shows the implementation of load_img function.

```
def load_img(num,img):       #num is the number of patches to be extracted
    yval = np.zeros((1,1))
    num0 = 0
    num1 = 0
    nrow = img[0].shape[0]
    ncol = img[0].shape[1]
    xgen = []
    ygen = []
    while num0<num/2 or num1<num/2:      #equal number of gland and non-
    gland patches

        x = random.randint(25,nrow-26)       #randomly select the central
    pixels
        y = random.randint(25,ncol-26)

        if(
            img[5][x,y]==0 and num0<num/2 or
            img[5][x,y]==1 and num1<num/2
        ):

            xred = img[0][x-25:x+26,y-25:y+26]       #extracting 51x51 patch
     around the central pixel
            xgreen = img[1][x-25:x+26,y-25:y+26]
            xblue = img[2][x-25:x+26,y-25:y+26]
            xh = img[3][x-25:x+26,y-25:y+26]
            xe = img[4][x-25:x+26,y-25:y+26]
            yval[0][0] = img[5][x,y]

            xred = xred.reshape((1,2601))
            xgreen = xgreen.reshape((1,2601))
            xblue = xblue.reshape((1,2601))
            xh = xh.reshape((1,2601))
            xe = xe.reshape((1,2601))

            ximg = np.concatenate([xred , xgreen , xblue , xh , xe] , axis
    =1)
            xgen.append(x)
```

```
34              ygen.append(y)
35              if num0==0 and num1==0:
36                  datax = ximg
37                  datay = yval
38              else:
39                  datax = np.concatenate([datax,ximg])
40                  datay = np.concatenate([datay,yval])
41
42              if img[5][x,y]:
43                  num1 = num1+1
44              else:
45                  num0 = num0+1
46
47      rval = [datax,datay,xgen,ygen]
48      return rval
```

Below is the code for the Lenet conv pool layer as given in [12] which has been further modified to incorporate dropout.

```
class LeNetConvPoolLayer(object):

    def __init__(self, rng, input, filter_shape,image_shape, dropout,
    poolsize=(2, 2)):
        assert image_shape[1] == filter_shape[1]
        self.input = input

        fan_in = np.prod(filter_shape[1:])

        fan_out = (filter_shape[0] * np.prod(filter_shape[2:]) /
                   np.prod(poolsize))

        W_bound = np.sqrt(6. / (fan_in + fan_out))
        self.W = theano.shared(
            np.asarray(
                rng.uniform(low=-W_bound, high=W_bound, size=filter_shape),
                dtype=theano.config.floatX
            ),
            borrow=True
        )

        b_values = np.zeros((filter_shape[0],), dtype=theano.config.floatX)
        self.b = theano.shared(value=b_values, borrow=True)

        conv_out = conv.conv2d(
            input=input,
            filters=self.W,
            filter_shape=filter_shape,
            image_shape=image_shape
        )

        pooled_out = downsample.max_pool_2d(
            input=conv_out,
            ds=poolsize,
            ignore_border=True
        )
```

```
36
37          if dropout > 0.0:
38              retain_prob = 1 - dropout
39              pooled_out *= srng.binomial(pooled_out.shape, p=retain_prob,
    dtype=theano.config.floatX)
40              pooled_out /= retain_prob
41
42          self.output = pooled_out + self.b.dimshuffle('x', 0, 'x', 'x')
43          self.output = theano.tensor.switch(self.output<0, 0, self.output)
44          self.params = [self.W, self.b]
45          self.input = input
```

The hidden layer implementation [13] with ReLU activation is as follows.

```
class HiddenLayer(object):
    def __init__(self, rng, input, n_in, n_out,dropout, W=None, b=None,
    activation=T.tanh):

        self.input = input
        if W is None:
            W_values = np.asarray(rng.uniform(
                    low=-np.sqrt(6. / (n_in + n_out)),
                    high=np.sqrt(6. / (n_in + n_out)),
                    size=(n_in, n_out)), dtype=theano.config.floatX)
            if activation == theano.tensor.nnet.sigmoid:
                W_values *= 4

            W = theano.shared(value=W_values, name='W', borrow=True)

        if b is None:
            b_values = np.zeros((n_out,), dtype=theano.config.floatX)
            b = theano.shared(value=b_values, name='b', borrow=True)

        self.W = W
        self.b = b

        lin_output = T.dot(input, self.W) + self.b

        if activation is 'relu':
            out = (theano.tensor.switch(lin_output<0, 0, lin_output))
        else:
            out = T.tanh(lin_output)

        if dropout > 0.0:
            srng = theano.tensor.shared_randomstreams.RandomStreams(
                rng.randint(999999))
            mask = srng.binomial(n=1, p=1-dropout, size=out.shape)
            self.output = out * T.cast(mask, theano.config.floatX)
        else:
            self.output = out
        self.params = [self.W, self.b]
```

The logistic regression class as taken from [14] is shown below.

```
class LogisticRegression(object):

    def __init__(self, input, n_in, n_out, W = None, b = None):

        if W is None:
                W = theano.shared(value=np.zeros((n_in, n_out),dtype=theano
    .config.floatX),name='W', borrow=True)

        if b is None:
                b = theano.shared(value=np.zeros((n_out,), dtype=theano.
    config.floatX), name='b', borrow=True)

        self.W = W
        self.b = b

        self.p_y_given_x = T.nnet.softmax(T.dot(input, self.W) + self.b)

        self.y_pred = T.argmax(self.p_y_given_x, axis=1)

        self.params = [self.W, self.b]

    def negative_log_likelihood(self, y):
        return -T.mean(T.log(self.p_y_given_x)[T.arange(y.shape[0]), y])

    def errors(self, y):
        if y.ndim != self.y_pred.ndim:
            raise TypeError('y should have the same shape as self.y_pred',
                ('y', y.type, 'y_pred', self.y_pred.type))

        if y.dtype.startswith('int'):
            return T.mean(T.neq(self.y_pred, y))
        else:
            raise NotImplementedError()
```

Bibliography

[1] "Cancer: Facts, Causes, Symptoms and Research,"
Available at http://www.medicalnewstoday.com/info/cancer-oncology

[2] K. Nguyen, B. Sabata, A. Jain, "Prostate cancer grading: Gland segmentation and structural features," *Pattern Recognition Letters*, Vol. 33, No. 7, 2012, pp. 951-961.

[3] S. N. Al-Haj Saleh, O.S. Al-Kadi, M.B. Al-Zoubi, "Histopathological Prostate Tissue Glands Segmentation for Automated Diagnosis," *IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, 2013, pp. 1-6.

[4] S. Naik, S. Doyle, S. Agner, A. Madabhushi, M. Feldman, J. Tomaszewki, "Automated gland and nuclei segmentation for grading of prostate and breast cancer histopathology," in the 5th International Symposium on Biomedical Imaging: From Nano to Macro (ISBI), 2008, pp. 284-287.

[5] K. Nguyen, A. Sarkar, A. Jain, "Prostate Cancer Grading: Use of Graph Cut and Spatial Arrangement of Nuclei," IEEE transcations on Medical Imaging, vol. 33, no. 12, 2014, pp. 2254-2270.

[6] K. Nguyen, A. Sarkar, A. Jain, "Structure and Context in Prostatic Gland Segmentation and Classification," Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2012, pp. 115-123.

[7] Arnout C Ruifrok and Dennis A Johnston, "Quantication of histochemical staining by color deconvolution.," AQCH, vol. 23, no. 4, pp. 291-299, 2001.

[8] Milan Gavrilovic et.al, "Blind color decomposition of histological images.", IEEE TMI, vol. 32, no. 6, pp. 983-994, 2013.

[9] Daniel D Lee and H Sebastian Seung, "Learning the parts of objects by non-negative matrix factorization," Nature, vol. 401, no. 6755, pp. 788-791, 1999.

[10] Julien Mairal et.al, "Online dictionary learning for sparse coding," in ICML. ACM, 2009, pp. 689-696.

[11] Vahadane, Abhishek, et al. ”Structure-preserved color normalization for histological images.” Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on. IEEE, 2015.

[12] ”Convolutional Neural Network,” Available at http://deeplearning.net/tutorial/lenet.html

[13] ”Multilayer Perceptron,” Available at http://deeplearning.net/tutorial/mlp.html

[14] ”Logistic Regression,” Available at http://deeplearning.net/tutorial/logreg.html

[15] ”Evaluation,” Available at http://www2.warwick.ac.uk/fac/sci/dcs/research/combi/research/bic/glascontest/evaluation/

[16] ”Gland Segmentation Contest,” Available at http://www2.warwick.ac.uk/fac/sci/dcs/research/combi/research/bic/glascontest/