Classification and Detection of Nuclei in Histology Images Using Convolutional Neural Networks

Archit Khosla, B.Sc. Computer Science[1], Juan C. Kuri, B.Sc. Computer Engineering[2]
[1]University of California San Diego, USA; [2]ESPOL, Ecuador

Abstract: *By using convolutional neural networks, also known as CNN or convnets, an artificial vision system was programmed and trained in order to classify and detect nuclei in histology images. Basically, this vision system is a proof-of-concept that could lead to the partial automatization of the work of pathologists. Thus, this vision system could potentially help many patients, save lives, reduce healthcare costs, and save the time and efforts of pathologists.*

Introduction

Motivation and Background

Today, pathologists sit under microscopes day and night to count and to distinguish nuclei in histology images. Pathologists have to do this tedious process in order to understand the diseases of patients. For example, pathologists could ask: Do patients have cancerous tumors? If so, to what extent?

This task can be very tedious and prone to human error. The more pathologists get tired, the more their capability to correctly diagnose nuclei drops. These errors could be deathly and cost the lives of patients. This process is also very slow and there is a very limited number of pathologists. The rate at which we require to process these slides is intense and automating this process could help to improve and to save many lives and the precious time of pathologists.

Moreover, automating this process could decrease the cost of lab tests, making them more accessible to poor people who desperately need lab tests and cannot afford them. Diseases and economical problems usually come together and are highly correlated.

Literature Survey

Previously, many works in this field have been done. Here are the approaches tried:

- Locality sensitive deep learning approaches to detect and classify nuclei in routine hematoxylin and eosin (H&E) stained histopathology images of colorectal adenocarcinoma, based on convolutional neural networks (CNN). [1]
- Cosatto et al. [2] detected cell nuclei using difference of Gaussian (DoG) followed by Hough transform to find radially symmetrical shapes.
- Al Kofahi et al. [3] proposed a graph cut-based method that is initialized using response of the image to Laplacian of Gaussian (LoG) filter.
- Arteta et al. [4] employed maximally stable extremal regions for detection, which is likely to fall victim to weakly stained nuclei or nuclei with irregular chromatin texture.
- Vink et al. [5] employed AdaBoost classifier to train two detectors, one using pixel-based features and the other based on Haar-like features, and merged the results of two detectors to detect the nuclei in immunohistochemistry stained breast tissue images. The performance of the method was found to be limited when detecting thin fibroblasts and small nuclei.

However, this vision system uses convnets due to their superior performance when classifying images.

Method

Pipeline

The pipeline for the visual system has the following processes, also represented in *Figure 1*:

- Reading the dataset of images.
- Splitting data into train, validation, and test datasets.
- Cropping tumor subimages from images.
- Removing subimages with black padding.
- Normalizing image pixels to [-1, 1].
- Training convnets and meta-optimizing parameters.
- Selecting the best model. (Notice that convnets automatically extract features.)
- Testing the best classifier.
- Detecting classes of nuclei in images of the test dataset.

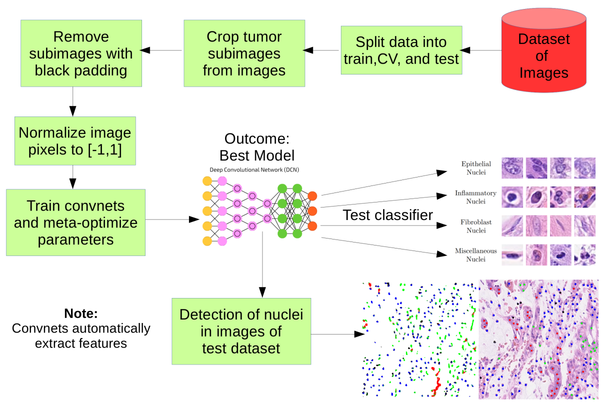
Dataset of Images
Split data into train,CV, and test
Crop tumor subimages from images
Remove subimages with black padding
Normalize image pixels to [-1,1]
Train convnets and meta-optimize parameters
Outcome: Best Model
Deep Convolutional Network (DCN)
Test classifier
Epithelial Nuclei
Inflammatory Nuclei
Fibroblast Nuclei
Miscellaneous Nuclei
Note: Convnets automatically extract features
Detection of nuclei in images of test dataset

Figure 1: Pipeline

Splitting Data into Train, Validation, and Test Datasets

Once images are read from disk, images are divided into 3 datasets and placed in their corresponding directories:

- training dataset (60%);
- cross-validation dataset (20%);
- and testing dataset (20%).

Cropping Tumor Subimages from Images

Each histology image contains many tumors like the following figure. Those tumor subimages were extracted and saved in a different directory structure to train and to test the classifier. For each annotation in the data set, a square patch of variable size was cut around annotation and used as a subimage. The patch size varied from every other integer between 12 to 32.

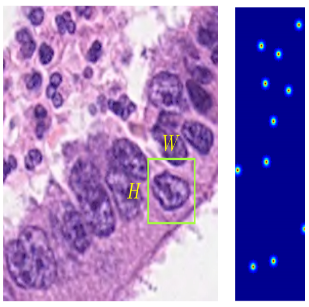

Figure 2: Example of an image and a tumor subimage [1]

In this step, an additional process to whats provided in the data is done: Cropping subimages of background which do not contain any marked nuclei. By creating a new category, the background category, the detection process reduced the number of false positives. The background category is crucial for achieving greater accuracy in detection.

Removing Subimages with Black Padding

Some subimages were nearby corners and edges and were padded with black pixels to fill the voids. As there were only a small fraction of such cases, these subimages with black padding were removed from the dataset to clean it from impurities instead increasing the complexity to find which color would not interfere with our results.

Normalizing Image Pixels to [-1, 1]

Images are stored in RGB format. RGB stands for red, green, and blue. Each RGB value is usually stored as an integer in the range [0, 255]. However, PyTorch represents image tensors with RGB values in the range [0, 1]. Deep learning experts recommend to normalize this range to [-1, 1], so that the center is located at 0. In this way, neural networks have better convergence for learning.

Training Convnets

Convnets were created using PyTorch. In the following figure, the diagram of the selected convnet is shown. Many models of convnets were explored. This model was selected due to its better performance according to the metrics. Diagrams of other models are omitted due to space limitation.

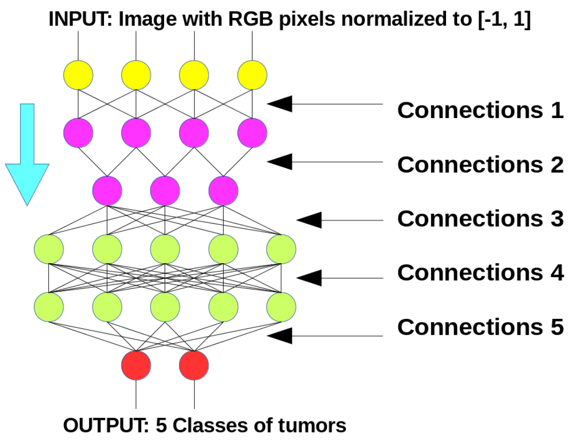
INPUT: Image with RGB pixels normalized to [-1, 1]
Connections 1
Connections 2
Connections 3
Connections 4
Connections 5
OUTPUT: 5 Classes of tumors

Figure 3: Diagram of the Convnet

Here is a brief description of the connections programmed in PyTorch, where each value is a variable and a grid search was done to find the best value:

Connections 1:
ReLU(Conv2d(in_channels = 3, out_channels = in_channels_2, kernel_size1 = 4))
MaxPool2d(kernel_size = 2, stride = 2)

Connections 2:
ReLU(Conv2d(in_channels = in_channels_2, out_channels = out_channels_2, kernel_size2 = 4))
MaxPool2d(kernel_size = 2, stride = 2)

Connections 3:
dim_size = (((patch_size – kernel_size1) / 2) – kernel_size2) / 2
ReLU(Linear(in_features = out_channels_2 * dim_size * dim_size, out_features = hidden_2))

Connections 4: ReLU(Linear(in_features = hidden_2, out_features = hidden_3))
Connections 5: Max(Linear(in_features = hidden_3, out_features = 5))

Table 1: CNN connections programmed in PyTorch

Regarding the criteria for stopping learning to prevent overfitting, *formula (1)* describes its mechanism. The convnet should stop learning when the current validation loss (VL) is relatively greater than the minimum validation loss (MinVL) found so far by an epsilon percentage. The default value is epsilon = 0.10. Each time a new minimum is found, the current convnet is entirely saved on disk in order to be loaded after learning is stopped. Statistics are computed with the best convnet found and saved.

$$(VL - MinVL)/MinVL > \epsilon \tag{1}$$

According to the stopping criteria explained above, the convnet should stop learning at epoch 34 in *graph 4*. However, the best convnet was found and saved on disk at epoch 21 because in this epoch the convnet has the minimum validation loss.

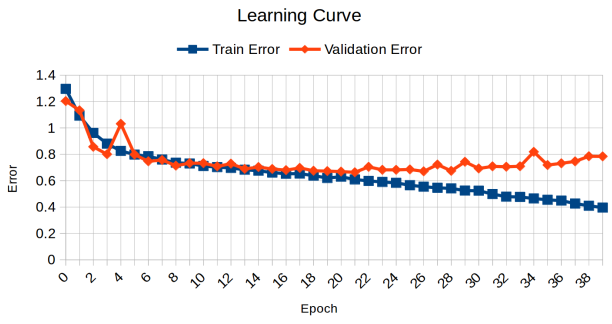

Figure 4: Example of learning curve

Meta-optimizing Parameters

Basically, there are 3 ways to meta-optimize the parameters of convnets:

- varying only 1 parameter at once and keeping the other parameters constant (Var1);
- varying all parameters at once without keeping constant parameters (VarAll);
- and varying a group of parameters at once and keeping the other parameters constant (VarN).

Based on the pros and cons mentioned in *Table 2* and due to time and computational limitations, this project was meta-optimized by using the method VarN. While the results were not perfect they were really good.

Meta-Optimization Method	Completeness	Time
Var1	Incomplete because it does not exploit synergies	Linear (Fast)
VarN	Acceptable but not complete	Sum of **low** exponentials (Reasonably slow)
VarAll	Complete with respect to the lists of values to explore	Exponential in the number of parameters (Extremely slow)

Table 2: Pros and Cons of Meta-optimization methods

Selecting the Best Model and Testing It

For each run of the meta-optimization process, a file TABLE.CSV was generated in the log/<RUN> directory. This table contains lots of columns with important information about the training process of each convnet for the input parameter combination.

At this stage one has the best model for classification, however, one should not have access to the results from the test set for the detection problem. Using, the best model could be considered cheating. Hence, the best model was chosen based on validation results only. Fscore was used to determine the best model.

Once the best model is selected for detection, then more results are computed for the classification problem to determine models performance.

Detecting classes of nuclei in images of the test dataset

In this step, this project goes one step further than the original paper [1]. In the original paper, nuclei are detected regardless of their class. In this project, nuclei are detected and classified according to their class.

Since the detection process uses no meta-parameters to optimize, it is safe to evaluate its performance directly in the test dataset. However, for the sake of completeness, the system generates detection images and statistics for both validation dataset (20%) and test dataset (20%). But, only the results of the test dataset will be shown in order to avoid selection bias, aka cheating. Why? Because meta-parameters were selected with the statistics of the validation dataset and thus, the performance in the validation dataset is much better.

Detected images were computed in this way: A sliding window was passed through the image with stride = 1, generating a winning class for each pixel, which is colored accordingly. Here are the colors associated to each class:

Class	Associated Color
Epithelial nuclei	🟥 RED
Fibroblast nuclei	🟩 GREEN
Inflammatory nuclei	🟦 BLUE
Miscelaneous nuclei	⬛ BLACK
Background	WHITE

Table 3: Classes and associated colors

The statistics of detection were computed in this way: A sliding window was passed through the image with stride = patch_size / 2, generating a winning class for each point. Then each point is associated to the nearest marked nucleus in ground truth, only if such marked nucleus is not too far. Nearest marked nuclei whose distance is greater than patch_size are not associated to points.

True positives are points whose associated nucleus match their class. True negatives are marked nuclei without associated points. False positives are points whose associated nucleus do not match their class. False positives are also points with a class and without associated nuclei. False negatives are points classified as background with an associated nucleus.

Experiments

Experimental Setup

Everything was programmed in Python. Convnets were programmed in PyTorch. Python's library Pillow was used for image operations. Big Data analytics was done with PySpark. Basically, the processes to be done in each major stage were distributed among many processors by using RDD. There are 3 major stages: Preprocessing of images, meta-optimization of learning, and detection.

The virtual machine nvidia-docker was used in order to guarantee consistency when running the project in different machines. The local machines to develop the system are:

Machine 1: Operative System: Mac OS. Processor: 2.9 GHz i7. Memory: 16GB RAM. Graphics: Intel HD graphics 630. Disks: 1536 Gb SATA and 512 SSD.

Machine 2: Operative System: Linux. Processor: 3.5 GHZ i7. Memory: 16GB RAM. Graphics: NVIDIA® GeForce® 940MX (4GB DDR3 dedicated). Disks: 1 Tb SATA and 256 SSD.

Machine 3: Operative System: Ubuntu 16.04 LTS (64 bits). Memory: 15,7 Gb RAM. Processor: AMD FX™-6300 Six-Core Processor x 6. Graphics: GeForce GTX TITAN Black/PCIe/SSE2. Disks: 868,9 Gb and 5 Tb.

Data

This dataset involves 100 H&E stained histology images of colorectal adenocarcinomas from 9 patients (the cohort), at a pixel resolution of 0.55 µm/pixel (20× optical magnification). All images have a size of 500 × 500 pixels. A total number of 29,756 nuclei were marked at the center for detection purposes. Out of which, there were 22,444 nuclei that also have an associated class label, i.e. epithelial, inflammatory, fibroblast, and miscellaneous. In total, there are 7,722 epithelial, 5,712 fibroblast, 6,971 inflammatory, and 2,039 miscellaneous nuclei (other). The nuclei that do not fall into the first three categories (i.e., epithelial, inflammatory, and fibroblast) such as adipocyte, endothelium, mitotic figure, nucleus of necrotic (i.e., dead) cell, etc. are labeled as miscellaneous. [1] The dataset can be found in the following link:

CRCHistoPhenotypes - Labeled Cell Nuclei Data
https://www2.warwick.ac.uk/fac/sci/dcs/research/tia/data/crchistolabelednucleihe/

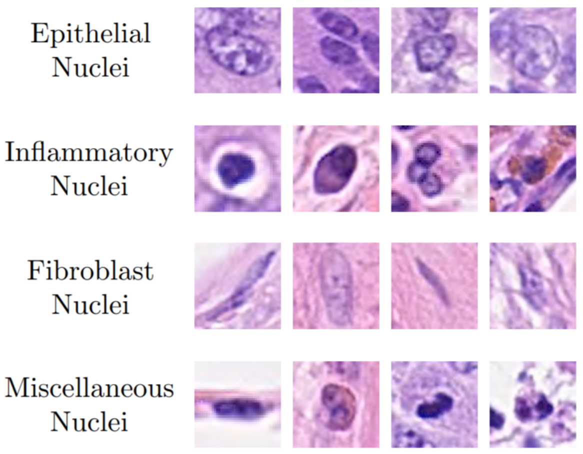

Figure 5: Classes of nuclei [1]

Baseline Experiment

Random Classifier: The dataset contains 5 classes. A model which randomly picks a class and claims that to be the result would produce a theoretical accuracy of 20% for each class:

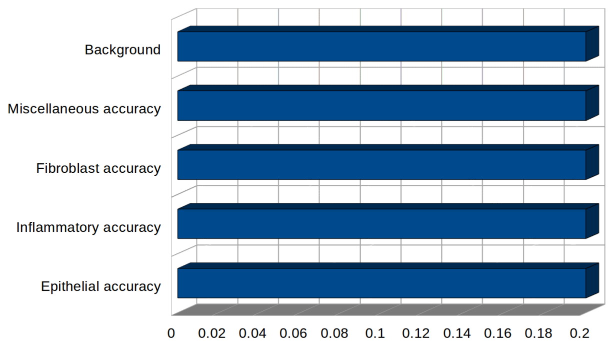

Figure 6: Theoretical statistics of random classifier

Original Paper [1]: The paper has limited results that can be directly compared with our results. However, we try our best efforts to limit and convert our results to make them apples to apples comparison. The best F-score seen in that paper was 0.784 and for detection they show that they got precision, recall, f-score as 0.758, 0.827, 0.791 respectively.

Results and Analysis for Classification

Convnets were trained by using combinations of the following values. However, to select these ranges we first ran each parameter independently to find what results better.:

Meta-parameter	List of values
batch_size	150, 250
learning_rate	0.001, 0.005, 0.01, 0.025, 0.03, 0.04, 0.05
momentum	0.2, 0.4, 0.5, 0.6, 0.7
hidden_2	100, 120, 140, 160
hidden_3	40, 60, 80, 100
in_channels_2	6, 10, 14, 16, 18
out_channels_2	8, 10, 12, 16, 20
patch_size	16, 20, 24, 28

Table 4: Meta-parameters and their lists of values

If the meta-optimization method VarAll had been used, the amount of combinations to train would have been 2 * 7 * 5 * 4 * 4 * 5 * 5 * 4 = 112,000, which are impossible to evaluate due to time and computational constraints.

Fortunately, the meta-optimization method VarN was used instead, varying only groups of parameters at once. VarN generated only 75 combinations, which is tractable. From the 75 combinations, here is a table with the 10 best combinations, based on the best F-Score of the validation dataset:

Learning rate	Momentum	Hidden 2	Hidden 3	F-Score (VALID)
0.025	0.6	140	60	0.708
0.025	0.6	120	60	0.699
0.05	0.5	140	60	0.698
0.02	0.6	140	60	0.697
0.025	0.6	120	80	0.696
0.025	0.6	100	60	0.696
0.025	0.4	120	80	0.694
0.025	0.6	140	100	0.693
0.03	0.7	140	60	0.692
0.05	0.7	140	60	0.692

Table 5: Top 10 models of convnets and their parameters

In this table, many columns where omitted for the sake of brevity. These omitted columns include batch_size (with value of 250 for the ten rows), in_channels_2 (with value of 16 for the ten rows), out_channels_2 (with value of 10 for the ten rows), and patch_size (with value of 24 for the ten rows). Moreover, the average training time was 1218.28 seconds and the average amount of epochs was 69.77.

After this long and painful meta-optimization process, the following combination of values was selected:

TrainClassifier parameters	ConvNet parameters

batch_size = 250
learning_rate = 0.025
momentum = 0.6
	hidden_2 = 140
	hidden_3 = 60
	in_channels_2 = 16
	out_channels_2 = 10
	patch_size = 24

Table 6: Parameters used by the best classifier

This combination of values was evaluated in the test dataset, producing the following results:

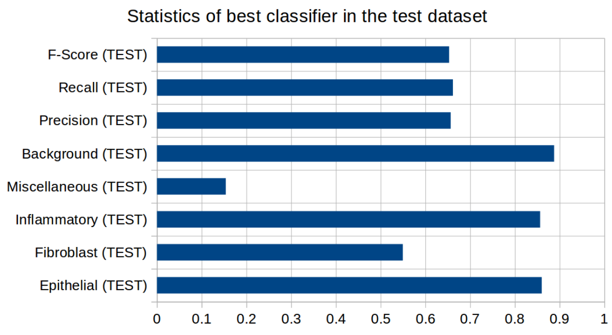

Figure 7: Statistics of best classifier in the test dataset

The accuracies of epithelial, inflammatory, and background classes are high. The accuracy of the fibroblast class is decent. And the accuracy of the miscellaneous class is bad. The F-Score is good enough. Compared to the F-Score of the original paper we did not do that well, but it was close.

Results and Analysis for Detection

Once the best classifier was selected, the detection software should use the best classifier to scan images pixel by pixel. In the 2 next figures, detection was pretty accurate. Compare the marked nuclei based on ground truth on the right of each image and the detection results on the left.

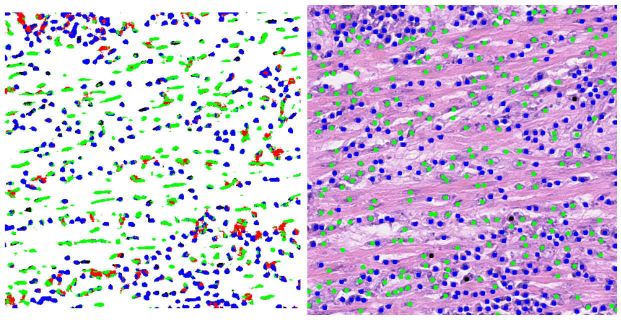
Figure 8: Example of detection in img89.png

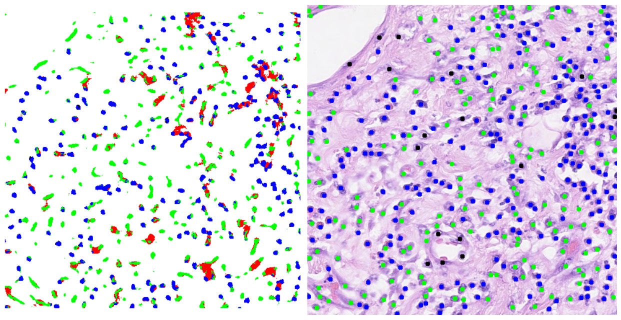
Figure 9: Example of detection in img81.png

Not all results were satisfying. In the next figure, the red class (epithelial nuclei) produced many false positives. This can be improved by exploring more the meta-parameters and the neural architectures of the convnets. More layers, deeper learning, are probably required.

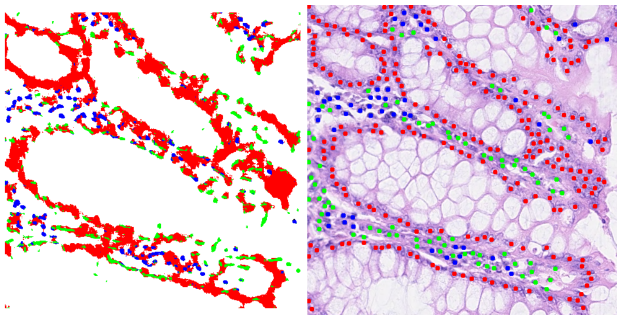
Figure 10: Example of detection in img69.png

The detection statistics of the test dataset are good. The average F-score is 0.47 and the average accuracy is 0.73, with standard deviations of 0.16 approximately. Accuracy is higher than F-score because the accuracy of the background classifier is high, which makes the amount of true negatives very big. The background classifier has a high accuracy because background is the most abundant class, covering more than 75% of the area of images approximately. More data produce more accurate classifiers.

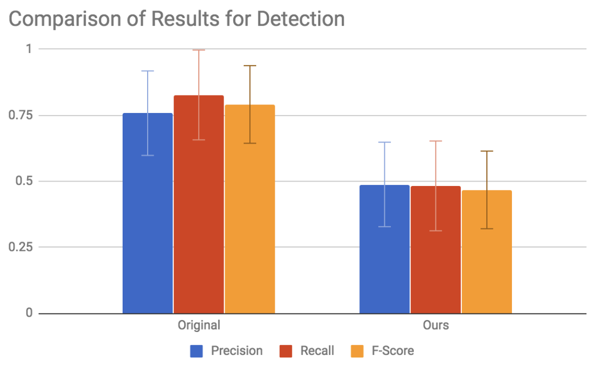

Figure 11: Detection statistics compared to original paper

Conclusion

Based on the experiments and observations, there is potential in this project. However, more exploration is required to achieve greater performance. The meta-optimization process should be better distributed among multiple processors in the cloud. The motivation to achieve better results is imperative because this will help the healthcare industry in a huge and unquantifiable way. Human lives cost more than money.

Even if this algorithm is not 100% accurate, it should be better than the average performance of humans to be adopted as a reliable and groundbreaking technology. This is a requirement for every AI system which is supposed to recommend actions and perceptions to humans.

Making the visual classifier stronger is paramount for new versions of the project. The classifier should emit confident signals when it is passed at the center of tumors, and should avoid false positives as well.

The miscellaneous category of tumors is hard to predict because this category is ill-defined. In other words, many categories of tumors that rarely appear are grouped together to form a bigger category. However, this should not be a problem for deep learning because deeper and wider convnets should be able to make a good variety of complex features capable of characterizing these ill-defined and less frequent categories. However, tuning such neural topologies is hard due to the exponential nature of meta-optimization methods and the computational cost of training neural networks.

Creating the background class definitely improved results and reduced the amount of false positives. By using only 4 classes, the initial images were totally green, implying the detector saw fibroblasts everywhere, which is wrong. When using 5 classes, the detected images improved dramatically, making the observer feel that the vision system is starting to gain visual awareness.

References

[1] Sirinukunwattana et al., “Locality Sensitive Deep Learning for Detection and Classification of Nuclei in Routine Colon Cancer Histology Images”, IEEE Transactions on Medical Imaging, 2016. (in press)

[2] E. Cosatto, M. Miller, H. P. Graf, and J. S. Meyer, “Grading nuclear pleomorphism on histological micrographs,” in Pattern Recognition, 2008. ICPR 2008. 19th International Conference on. IEEE, 2008, pp. 1–4

[3] Y. Al-Kofahi, W. Lassoued, W. Lee, and B. Roysam, “Improved automatic detection and segmentation of cell nuclei in histopathology images,” Biomedical Engineering, IEEE Transactions on, vol. 57, no. 4, pp. 841–852, 2010.

[4] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman, “Learning to detect cells using non-overlapping extremal regions,” in Medical image computing and computer-assisted intervention –MICCAI 2012. Springer, 2012, pp. 348–356.

[5] J. P. Vink, M. Van Leeuwen, C. Van Deurzen, and G. De Haan, “Efficient nucleus detector in histopathology images,” Journal of microscopy, vol. 249, no. 2, pp. 124–135, 2013

[6] A. A. Cruz-Roa, J. E. A. Ovalle, A. Madabhushi, and F. A. G. Osorio, “A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection,” in Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013. Springer, 2013, pp. 403-410.

[7] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images,” in Advances in neural information processing systems, 2012, pp. 2843–2851

[8] A. Vedaldi and K. Lenc, “Matconvnet – convolutional neural networks for Matlab,” CoRR, vol. abs/1412.4564, 2014.

[9] D. J. Hand and R. J. Till, “A simple generalization of the area under the ROC curve for multiple class classification problems,” Machine learning, vol. 45, no. 2, pp. 171–186, 2001.

[10] Y. Xie, F. Xing, X. Kong, H. Su, and L. Yang, “Beyond classification: Structured regression for robust cell detection using convolutional neural network,” in Medical Image Computing and Computer-Assisted Intervention MICCAI 2015. Springer, 2015, pp. 358–365.

[11] 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