Comparison of Different Classifiers with Active Learning to Support Quality Control in Nucleus Segmentation in Pathology Images

Si Wen, M.S.[1], Tahsin M. Kurc, Ph.D[1], Le Hou, M.S.[1], Joel H. Saltz, MD, Ph.D[1], Rajarsi R. Gupta, MD[1], Rebecca Batiste, MD[2], Tianhao Zhao, MD[2], Vu Nguyen, M.S.[1], Dimitris Samaras, Ph.D[1], Wei Zhu, Ph.D[1]
[1]Stony Brook University, Stony Brook, NY, USA; [2]Stony Brook School of Medicine, Stony Brook, NY, USA

Abstract

Segmentation of nuclei in whole slide tissue images is a common methodology in pathology image analysis. Most segmentation algorithms are sensitive to input algorithm parameters and the characteristics of input images (tissue morphology, staining, etc.). Because there can be large variability in the color, texture, and morphology of tissues within and across cancer types (heterogeneity can exist even within a tissue specimen), it is likely that a set of input parameters will not perform well across multiple images. It is, therefore, highly desired, and necessary in some cases, to carry out a quality control of segmentation results. This work investigates the application of machine learning in this process. We report on the application of active learning for segmentation quality assessment for pathology images and compare three classification methods, Support Vector Machine (SVM), Random Forest (RF) and Convolutional Neural Network (CNN), for their performance improvement and efficiency.

Introduction

Review of glass tissue slides under a high-power microscope by a pathologist is the de facto standard for cancer diagnosis.[1] This manual process is labor intensive and time consuming[2], and is not scalable to large image datasets, which are desirable in research studies. With the development of digital pathology, computer algorithms can be used to automate the manual segmentation and characterization of structures and computation of imaging features for classification of images and patients.[3] Computerized detection and segmentation of nuclei and cells is one of the core operations in tissue image analysis, and a variety of nucleus segmentation methods[1, 4, 5] have been developed. Nevertheless, due to the wide variety of nucleus appearances and inter-cell structures across tissues and within a tissue, it is not uncommon that a nucleus segmentation algorithm with a specific set of parameters will produce accurate segmentations in some images and image regions, but will not perform as well in others. For example, when a segmentation algorithm[6] discriminates between background tissue and target nuclei via a threshold (or gain) parameter, the quality of the segmentation correlates with the choice of the threshold value. If a low threshold value is chosen, the algorithm will delineate nuclear boundaries well when nuclei are dark in color and there is high contrast between background (tissue) and nuclei. However, it may miss nuclei when there is low contrast and nuclei are light colored (because of staining, missing chromatin, etc.), resulting in under-segmentation of nuclei in some regions. If a high threshold value is applied, the algorithm can segment nuclei that the low threshold value will miss, but the high threshold value will likely result in over-segmentation in some regions (incorrectly detecting non-nuclear material and objects). In our previous study[7], we have proposed a pipeline to predict whether a region would be segmented well, not under- or over-segmented, by a segmentation algorithm with a specific threshold value. *In the current work, we investigate the application of machine learning algorithms and an active learning process to carry out a quality assessment of analysis results and select the best results in image regions, when an image has been analyzed with both a low threshold value and a high threshold value. The active learning process enables iterative improvement of the initial decision model when there are not enough labeled data ("ground truth data") initially to develop a generalized model.*

The traditional approach to automatically classifying image regions into different categories is to develop a supervised machine learning model based on a pre-labeled training set. The trained classifier is then tested on another set of data which is independent from the training set to examine its ability to predict the correct labels for unseen data. To ensure the model is capable of generalizing and is not overfitted, the training set should be representative and balanced[8] and as large as possible. This increases the difficulty of generating a proper labeled training set, especially when the image labeling process is manual and time consuming. Active learning, which

involves querying for more labeled data by an objective algorithm, has been shown to mitigate this problem.[9,10] With an active learning framework, a sampling strategy is applied to select relatively small set of data iteratively for experts to label. The selected samples are most likely to improve classifier performance. For each active learning iteration, the classifier is updated by the feedback from the expert and then all unlabeled data are re-evaluated for their ability to further improve the classifier.

Active learning has been successfully applied in many image classification problems.[11,12,13] In the field of biomedical image classification, it was utilized for either generating better feature representation for images[14,15] or creating classification models[16,17,18,19] iteratively. These applications are either at nuclei/cell level to detect a specific type of cell[14,15,17,18] or at patch level to select regions of interest (ROI)[16,19]. A variety of classification methods have been employed, such as linear classifiers (logistic regression[15] and Support Vector Machine[14]), ensemble methods (Probabilistic Boosting Tree[16], Random Forest[18], and Gentle Boosting[17]) and deep learning method (Bayesian CNN[19]).

Our process of quality assessment is carried out at the image patch level. That is, a whole slide tissue image is partitioned into patches. To determine whether segmentation results from low/high threshold values should be selected for a given patch, a classification model is trained with intensity and texture features extracted from the image patches. By incorporating active learning, a relatively small set of patches with ground truth labels is required to train a classification model. Then among the unlabeled patches, we select a small number of patches that can best improve the current classification model and label them. The initial model is updated by adding these new labeled patches into the original training set. We iteratively select unlabeled patches and update the current model with new labeled patches until the performance of the classifier is satisfied. We employ and compare three classification methods: Support Vector Machine (SVM), Random Forest (RF) and Convolutional Neural Network (CNN) for their performance improvement and efficiency. Each of the classifiers represents a different type of classification method: SVM is a linear classifier; RF is a representative of ensemble methods; and CNN is a deep learning method. All of them are commonly used supervised machine learning models.

Our work differs from the prior work in that the previous studies, to the best of our knowledge, did not investigate the application of active learning in the assessment of nucleus segmentation quality and did not compare the performance of different classifiers. We carried out experiments using datasets from two different cancer types: Breast Cancer (BC) and Pancreatic Cancer(PC). In our experimental evaluation, CNN has achieved the best classification performance, while taking the longest processing time. The performance in test accuracy of SVM improved the largest with active learning among the three classification methods. SVM was the most sensitive to the size of the training set. The execution time of RF was the smallest for each active learning iteration, but required the largest number of patches in total to develop a relatively generalized model.

Methods

Our nucleus segmentation quality control process consists of two steps: image data processing and active learning, as illustrated in Figure 1. The first step (orange parts in Figure 1) generates the labels and features for the traditional supervised learning, and the second step (green parts in Figure 1) shows how active learning is applied to update the classification model interactively. In the following subsections, we will first describe our active learning framework (Fig 1.d-f). We will then present the user interface for visualizing segmentation results (Fig 1.a) and two user interactive parts: marking up regions (Fig 1.b) and labeling patches (Fig 1.f). Lastly, we will describe our feature extraction method (Fig 1.c) and three classification methods (Fig 1.d).

Active Learning

Active learning is a kind of semi-supervised learning in which a learning algorithm is allowed to interactively ask a user to provide new labels for samples. The aim of the interactive learning is to achieve high accuracy using as few labeled samples as possible. In this way, the cost for obtaining labeled sample is minimized.[9] The simplest and most commonly used query strategy is uncertainty sampling. The samples that the classifier get most confused with are chosen to query for the labels. Updated by these new labeled uncertain samples, rather than by some random labeled samples, the classifier can improve most.[20] Since the model used in our experiment is a binary classifier, instead of multi-class one, we can simply use the predicted positive probability to calculate the uncertainty measure.

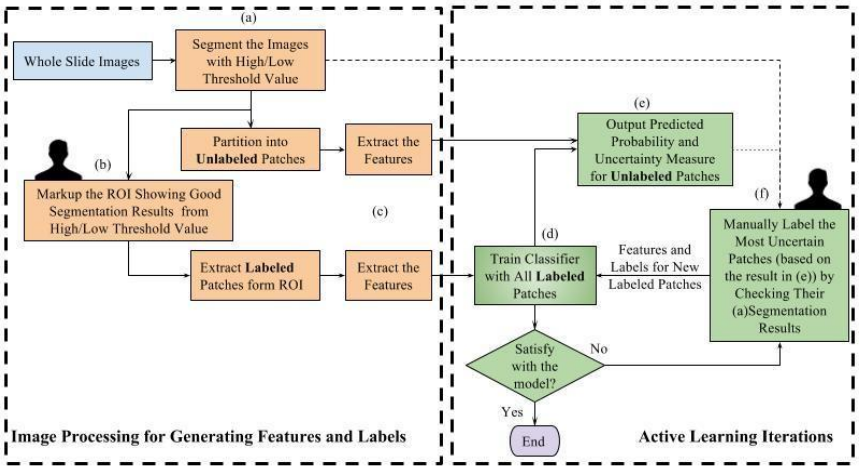

Figure 1. Workflow of active learning to support quality control in nucleus segmentation. (After generating (a)segmentation results for the whole slide images, a small portion of the images are selected to (b)markup regions showing good segmentation result from high/low threshold value and extract patches from the labeled regions; the other images are partitioned into unlabeled patches. Using the labels and (c)features extracted from the labeled patches, (d)a classifier is trained. If the users are not satisfied with the current model, they can select the unlabeled patches with less (e)uncertainty measure and (f)manually label them. Then, the (d) classifier is updated with both the original labeled patches and the new labeled patches until a satisfied model achieves)

Suppose we use $\boldsymbol{L}$ and $\boldsymbol{U}$ to denote the labeled and unlabeled dataset respectively. For each sample $\boldsymbol{x_i} \in \boldsymbol{L}$, there is a corresponding label $\boldsymbol{y_i} \in \{\boldsymbol{1}, \boldsymbol{0}\}$ to denote which class the labeled sample belongs to. After a classification model $\boldsymbol{f}$is trained on $\{\boldsymbol{x_i}, \boldsymbol{y_i}|\boldsymbol{x_i} \in \boldsymbol{L}\}$, it is tested on the unlabeled samples $\boldsymbol{x_i} \in \boldsymbol{U}$ and output a posterior probability $\boldsymbol{p_i} = \boldsymbol{P}\{\boldsymbol{f}(\boldsymbol{x_i}) = \boldsymbol{1}\} \in [\boldsymbol{0}, \boldsymbol{1}]$ for each unlabeled sample. The uncertainty measure is then defined as:

$$\boldsymbol{c_i} = -|\boldsymbol{p_i} - \boldsymbol{0.5}| \in [-\boldsymbol{0.5}, \boldsymbol{0}], \boldsymbol{for}\ \boldsymbol{x_i} \in \boldsymbol{U}$$

where $|\cdot|$ denote the absolute operator. This means that the sample with higher uncertainty measure, that is probability close to 0.5, are the ones that the classifier is more uncertain with. In our experiment, each time we label $\boldsymbol{h}$ samples. Let $\boldsymbol{n}$ be the number of samples in $\boldsymbol{U}$, then the selected samples are $\{\boldsymbol{x_i}|\boldsymbol{c_i} > \boldsymbol{c_{(n-h)}}\}$, where $\boldsymbol{c_{(n-h)}}$ denote the $\boldsymbol{n-h}$th order statistics of $\{\boldsymbol{c_i}\}$and, which is the $\boldsymbol{n-h}$th smallest uncertainty measure.

In our process as shown in Figure 1, the green parts ((d), (e), (f)) represent the iterative active learning framework. The current classifier $\boldsymbol{f}$ can output predicted probabilities $\boldsymbol{p_i}$ by passing the features for unlabeled patches through it. The uncertainty measure $\boldsymbol{c_i}$ is then computed and the most uncertain ones are sampled for manually labeling. The new labeled patches, combined with the previous labeled ones, are used to update the classifier.

User Interface for Visualizing Segmentation Result and Interactive Classification

For all the whole slide images (H&E stained) in our study, a computerized nucleus segmentation method [6] was applied. The method performs color-normalization in the L*a*b color space on input images using a properly stained template image. It then extracts the Hematoxylin (stained on nuclei mainly) channel through a color decomposition process. A localized region based level set method with a user-defined threshold value determines the contour of each nucleus. The threshold value is highly correlated with the quality of segmentation result.

To better visualize the segmented images and compare the segmentation results from algorithms with different threshold parameters, we used a web-based suite of tools, called the QuIP software (Figure 2(a)). QuIP is designed to support analysis, management, and query of pathology image data and image analysis results (http://quip1.bmi.stonybrook.edu/; https://github.com/SBU-BMI/quip_distro.git). The web interfaces and applications are backed by a database. The users can view high resolution whole slide tissue images and segmentation results from algorithms with different parameters, overlaid on the image as polygons with different

colors (Figure 2(b)), using the caMicroscope application [21]. After comparing the different segmentation results, the users can markup region of interest (ROI) using rectangular or free-hand drawing tools, annotate the regions with a label, and save the results in the database. The color of the user-drawn polygon is consistent with the color of segmentation result that is selected for that region (Figure 2(a)).

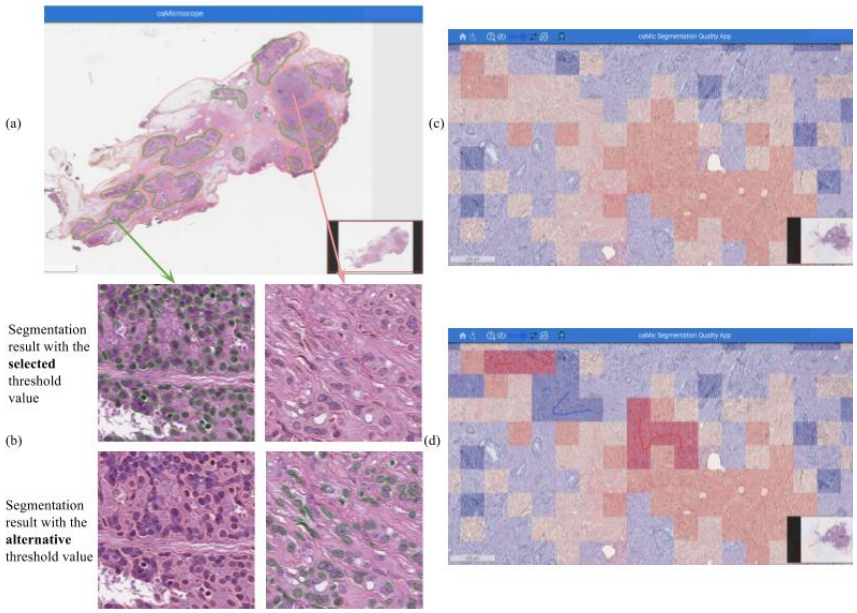

Figure 2. Viewing images with segmentation results, marking up ROI and labeling uncertain samples. ((a) regions chosen for each threshold value. The pink color represents the segmentation result from high threshold value and the green color represent the one from low threshold value; (b) sample patches from two different colored regions – the first row presents the result from the selected threshold value, and the second row shows the one from the alternative threshold value; (c) heatmap to show the location of uncertain samples. The light blue (or orange) ones are the uncertain samples, the blue (or red) ones are patches with high predicted probability to choose result from low (or high) threshold value; (d) labeling the uncertain samples with blue line means the high threshold and red line means low threshold and the color for the patch changes to the corresponding color.)

From the marked ROI, 512-by-512 non-overlapping patches (Figure 2(b)) are extracted to consists of the initial data set. After feature extraction and classifier training processes, a classifier is generated for predicting whether a high or low threshold value should be selected for any 512-by-512 tissue image patch. We partition a whole slide image with no markups into 512-by-512 non-overlapping patches. The predicted results for the whole slide image can be shown as heatmap (Figure 2(c)). The blue ones indicate the patches with high confidence to be labeled with low threshold value and the red ones are highly confident of being labeled with high threshold value. The light blue and orange ones are the uncertain samples with predicted positive probability close to 0.5. The light blue ones have the probability greater than 0.5 and the orange ones have the probability less than 0.5. The user can label uncertain patches with a blue line indicating the patch should be segmented with high threshold value or a red line representing low threshold value. At the end of the labeling, both the patches and their new labels are saved to the database.

Feature Extraction

We employed patch level intensity and texture features which were examined in our previous work[7] for nuclei segmentation quality control. There are two sets of features. The first set of feature contains 16 features from 3 groups: pixel statistics, gradient statistics and edge. The pixel statistics feature group contains the basic statistics (mean, standard deviation, minimum, maximum, skewness, and kurtosis) of the raw pixel value; the gradient statistics feature group contains the basic statistics of gradient value; and the last feature group contains the features

for the pixels on the edge of image patch. A total of 32 features are computed from this set; 16 for the red channel and 16 for the blue channel. We used features extracted from the blue and red channels as the image analysis and feature extraction pipeline we used in an earlier work[7] worked with these channels. The second set consists of the mean and standard deviation of the intensity values of the red, green and blue channels. These features were only used for SVM and RF, but not for CNN. The CNN has its own convolution layers to extract more problem specific features. Therefore, for each active learning iteration, the features extracted by CNN might be different, while the features for SVM and RF remain the same.

Classification Methods

In this paper, we mainly focus on the comparison of three types of classification methods: Support Vector Machine (SVM), Random Forest (RF) and Convolutional Neural Network (CNN). SVM is a supervised learning method. It finds hyperplanes in a high-dimensional space that maximizes the distance to the nearest training data point of any classes. We used the package Scikit-learn[22] in python to implement SVM with the radial basis function kernel. The kernel parameter gamma and cost were selected by 5-fold cross validation. The parameter selection was done for each active learning iteration to achieve better performance. Random Forest is an ensemble learning method for classification that works by bagging multiple decision trees and outputting the classification label by taking the majority vote. Each decision tree is built on a bootstrap sample of the training data using a randomly selected subset of variables.[23] We used the implementation of Random Forest in the package Scikit-learn[22] in python. In our experiments, we set the number of trees as 1,000, since based on our previous experience, the forest with 10,000 trees did not perform better than the forest with 1,000 trees.

The architecture of our CNN is shown in Table 1. The network takes input images of size 100-by-100 pixels in RGB along with the binary segmentation mask as the fourth channel. In this way, the location of each pixel is spatially aligned with its segmentation prediction. Based on the observation that, the quality of segmentation result is determined on *average*, how each nucleus is segmented. we design the network architecture to explicitly extract features at patch-level and make a final prediction based on the *averaged* feature. In particular, the size of the receptive field of a neuron in the last convolutional layer roughly equals to a typical size of nuclei. Then the global average pooling layer extracts globally averaged features. Finally, the last two fully connected layers perform classification. By doing the global average pooling, our network explicitly ignores large-scale structures which are irrelevant for predicting the results. This is done in order to prevent over-fitting.

Table 1. The architecture of our 15-layer network. We use batch normalization[24] followed by leaky ReLU[25] in all layers, except that for the output layer, we use the sigmoid activation function.

Layer	Output size	Filter size, stride
Input	100x100x4	-
Conv.	100x100x100	5x5, 1
Conv.	100x100x120	5x5, 1
Average Pooling	50x50x120	2x2, 2
Conv	50x50x240	3x3, 1
Conv.	50x50x320	3x3, 1
Average Pooling	25x25x320	2x2, 2
Conv	25x25x640	3x3, 1
Conv.	25x25x1024	3x3, 1
Conv.	25x25x640	1x1, 1
Conv.	25x25x100	1x1, 1
Conv. (repeat for 4 times)	25x25x320	1x1, 1
Global Average Pooling	1x1x320	25x25, 25
Fully Connected	1x1x100	-
Fully Connected	1x1x80	-
Output	1x1x1	-

The implementation details are as follows. We apply four types of data augmentation on-the-fly. First of all, we randomly rotate an input image by +/- 45 degrees. Secondly, we randomly crop a 100-by-100 sub-image out of the input 256-by-256-pixel image. Then, we randomly perturb the Hematoxylin and Eosin intensities. Finally, we randomly mirror and/or transpose the image. During test time, given an input image of 256-by-256 pixels (down sampled from our original 512-by-512 patches), we extract 25 overlapping sub-images of 100-by-100 pixels, obtain the predicted probabilities for all 25 sub-images and average them as the final prediction. For optimization, we train the network for 80 epochs, using stochastic gradient descent with learning rate 1×10^{-3} and momentum 0.985. After the initial training, the parameters at the last epoch are saved. For the next active learning iteration, the parameters are initialized with the saved ones and the learning rate are reduced by 0.1. We implemented the network using Theano[26].

Results and Discussions

We evaluated the proposed process using two sets of whole slide images obtained from two different cancer types: Breast Cancer (BC) and Pancreatic Cancer (PC). The size of our pre-labeled dataset for each cancer type was listed in Table 2. The patches in training set and test set were from independent sets of whole slide images. To suite for the active learning framework, the training set was regarded as the labeled data set and the test set was regarded as the unlabeled one. Since we need to compare the classification performance of each classifier, the test data set was labeled in advance for convenience. The labels for the test set were only used to examine the performance of the classifier and were not involved in the model training and the uncertainty sampling processes. After the most uncertain samples had been selected from the test set, they were moved to the training set for the active learning iterations to update the classifier.

Table 2. List of information of training and test dataset

# of patches (# of images)	Breast Cancer(BC)		Pancreatic Cancer(PC)	
	Low threshold	High threshold	Low threshold	High threshold
Training set	2605 (13)	3113 (8)	2456 (6)	1393 (9)
Test set	2463 (12)	3209 (7)	2465 (5)	1424 (8)

Performance of the Classifiers and Effectiveness of Active Learning Framework

The Receiver Operating Characteristic (ROC) curves for the initial classification model based on the original training set were shown in Figure 3. Since the classification result of CNN was not stable across different runs due to the randomness in data augmentation process, we presented the result for three runs of CNN (labeled as 'CNN-1', 'CNN-2', and 'CNN-3') with the same original data set and hyper parameter setting. The performance of the three runs of CNN for BC were roughly the same, while one of the runs for PC ('CNN-1') performed much worse than the other two runs. This may be caused by the different size of the initial training set for the two cancer types. The total number of patches for PC training set was only about two-third of that for BC. This shows that the CNN model generates more stable results for a larger data set. From the shape of ROC curves and the Area Under Curve(AUC) shown in the legend, all the CNN runs performed much better than SVM and RF for both cancer types. The RF achieved better performance than SVM for BC, but worse performance for PC. However, both SVM and RF performed even worse than random guessing. The 'S'-shaped ROC curves of SVM and RF for BC and the inverse 'S'-shaped ROC curves for PC were caused by the imbalance of the two categories in the initial training set. There were more negative samples (patches with high threshold value) for BC, whereas more positive ones for PC. For a weak classifier, like the initial RF and SVM models, there is more chance that the negative samples were labeled as positive for BC and the opposite for PC. Therefore, the false positive rate for BC increases rapidly when the decision threshold (the threshold value for the positive probability) is high, while the true positive rate for PC increases slowly when the decision threshold is low.

Table 3 lists the test accuracies of the initial classifiers and the classifiers trained after moving 100/200/400 most uncertain patches to the training set. In general, for both cancer types, the test accuracies of the three classification methods increased significantly (by at least 12%) after adding only 100 patches to the training set. This shows the

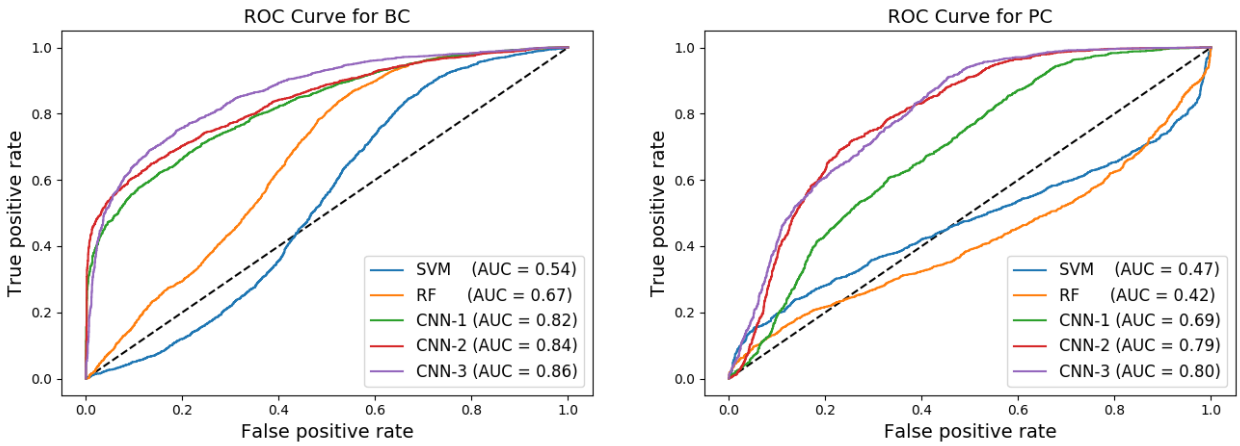

Figure 3. Comparison of ROC curves for different classifiers for both BC and PC

active learning framework is very effective for developing more generalized classifiers. As more patches were added to the training set, the test accuracy of the classifiers continued to increase. To compare among the three classification methods, the CNN runs got at least 7% better in test accuracy than SVM and RF in most of the cases. The exceptions occurred when CNN-1 and CNN-3 compared with SVM for PC. In that case, the SVM achieved higher improvement (its test accuracy increased by 33.93%) after adding 100 most uncertain patches to the training set. In fact, by adding the same number of patches, SVM was always the one with the most increase in test accuracy, indicating that the active learning framework was most effective for SVM in our experiments.

Table 3. Comparison of performance of different classifiers for both BC and PC

Test Accuracy (%)	Breast Cancer(BC)					Pancreatic Cancer(PC)				
	SVM	**RF**	**CNN-1**	**CNN-2**	**CNN-3**	**SVM**	**RF**	**CNN-1**	**CNN-2**	**CNN-3**
Initial	50.58	58.34	71.28	70.79	75.11	47.96	43.28	66.21	74.08	73.59
Add 100 patches	75.29	77.46	85.82	85.37	87.92	81.89	67.91	82.13	91.29	86.20
Add 200 patches	76.90	80.15	86.11	87.87	90.79	82.92	71.40	87.96	92.41	86.83
Add 400 patches	82.85	82.68	90.67	91.33	92.66	87.39	73.17	92.43	94.64	92.75

Efficiency of Active Learning Framework for the Classifiers

As shown in the last row of Table 3, after adding 400 most uncertain patches, all the CNN runs achieved 90% test accuracy, but SVM and RF needed more iterations to achieve the same high accuracy. Figure 4 shows how many patches in total were needed to reach 90% test accuracy for each classification method with active learning. For each active learning iteration, 100/200/400 most uncertain patches in the remaining test set were added to the training set. When more patches were added in a single iteration, the active learning process required fewer iterations to learn an improved model.

The total number of patches in the active learning phase may still be large. In order to reduce the manual work for labeling patches, a relatively small number of patches should be selected for each active learning iteration. Among all the classification methods, RF required the largest number of patches to get to 90% accuracy and CNN needed the smallest.

Although the CNN achieved the highest test accuracy and approached 90% with fewer patches for both BC and PC, its processing time (seen in Table 4) was much longer than the processing times of SVM and RF. The processing time for training a CNN model is the total processing time for finishing 80 epochs. The number of epochs was

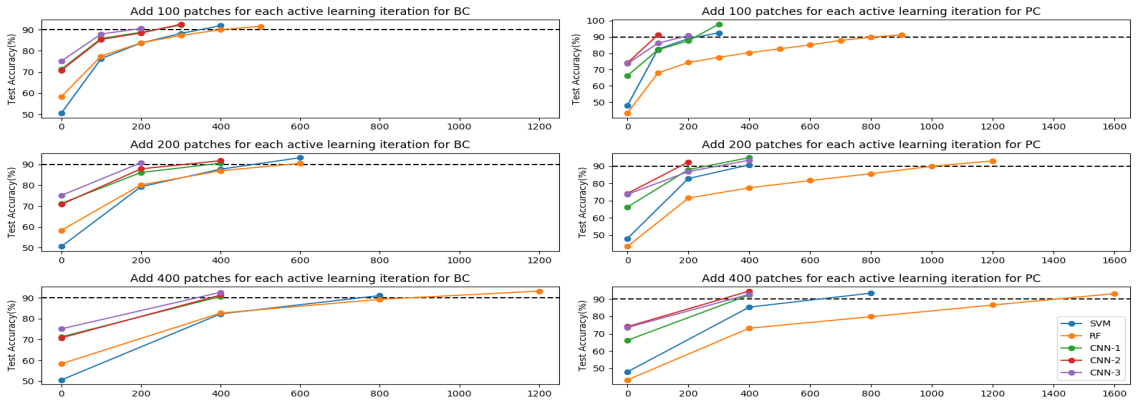

Figure 4. The test accuracy changes with the total number of patches added by active learning for both BC and PC

chosen as 80 to ensure the convergence of the model. In fact, Figure 5 shows that our CNN models almost converged after around 20 epochs, which still is nearly 5 hours, for both cancer type. RF was the fastest classification algorithm in our experiment. It took less than 1 minutes for both cancer type. Though it required 9 iterations to achieve 90% accuracy by adding 100 patches each iteration for PC, it took less than 5 minutes to finish the iterations. SVM spent about 16 minutes for BC and 6 minutes for PC. Since our initial training set for BC was larger than that of PC, all the classification method used longer time for building up models for BC than for PC. However, SVM was the most sensitive to input data size and took more than twice as much time for BC as for PC.

Table 4. Comparison of processing time of different classifiers for both BC and PC

Processing Time (second)	**Breast Cancer(BC)**				
	SVM	**RF**	**CNN-1**	**CNN-2**	**CNN-3**
Initial	950.15	33.31	107,973.619	116,783.285	108,506.769
Add 100 patches	909.96	34.34	108,687.878	106,496.861	107,924.15
Add 200 patches	864.73	34.77	107,611.661	106,756.751	106,154.06
Add 400 patches	937.02	35.68	106,532.782	107,620.047	106,215.291
Processing Time (second)	**Pancreatic Cancer(PC)**				
	SVM	**RF**	**CNN-1**	**CNN-2**	**CNN-3**
Initial	398.24	19.23	73,974.748	79,892.922	74,208.552
Add 100 patches	450.18	19.69	73,636.762	72,718.406	74,578.636
Add 200 patches	387.58	20.09	72,147.099	72,860.227	73,726.653
Add 400 patches	604.91	21.24	77,015.413	75,224.745	72,458.709

Generally, it took about 1 to 1.5 hours for marking up ROIs in a whole slide image as shown in Figure2(a). Our experiment result shows that by labeling ROIs in training image dataset (21 whole slide images for BC or 15 images for PC, as seen in Table2) and labeling no more than 1600 additional patches in test image dataset, we can build up a quality control pipeline to automatically label the patches from 19 images for BC or 15 images for PC with more than 90% accuracy. On average, it took 10 minutes to manually re-label 20 patches with high/low segmentation thresholds.

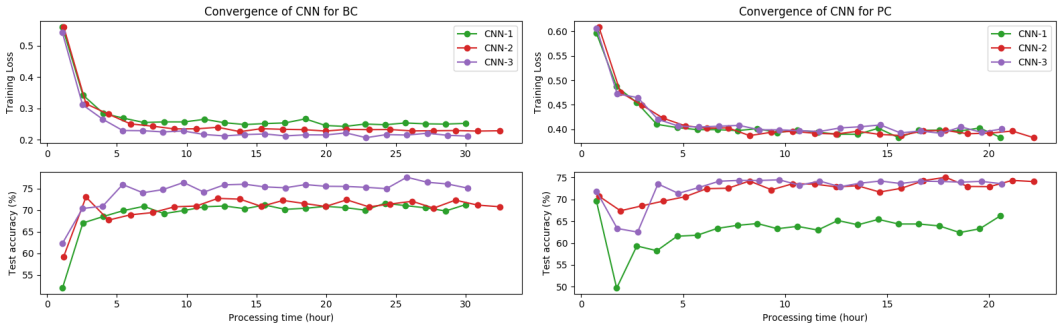

Figure 5. Convergence of CNN for both BC and PC. (The upper subplots present the change of training loss over time and the lower subplot shows the change of test accuracy over time. The time between two dots in the plot represent processing time for 4 epochs)

Overall, CNN achieved the best classification performance, while taking the longest processing time. The performance of SVM improved the most with active learning. SVM was the most sensitive to the size of the training set. RF spent the least amount of time for each iteration, but required the largest number of patches in total to develop a relatively generalized model. These results indicate a trade-off between effectiveness and efficiency. In our experiment, it took 30 seconds to label a PC patch with high/low segmentation threshold value -- we labeled 100 patches for each active learning iteration. Thus, it takes RF about 5 minutes to train all the classifiers and 450 minutes to label the required set of patches. The total processing time for SVM, on the other hand, is the sum of 25 minutes for classifiers training and 150 minutes for getting labels. These results show RF is slower than SVM.

Conclusion

In this paper, we propose an active learning process for nuclei segmentation quality assessment, which is employed by three classification methods: Support Vector Machine (SVM), Random Forest (RF) and Convolutional Neural Network (CNN). The performance and efficiency of the three classification methods was compared on patches extracted from whole slide images of two cancer types: Breast Cancer (BC) and Pancreatic Cancer(PC). All the classifiers improved effectively by adding a small amount of new labeled patches iteration by iteration. CNN outperformed the other two classification methods by developing a model with test accuracy of 90% through fewer active learning iterations. However, CNN requires much longer time for each iteration than SVM or RF does. This is expected because CNN re-extracts features in each iteration, whereas SVM and RF use texture features extracted a priori. In future we plan to explore ways of accelerating CNN with small reduction in accuracy; for example, by setting the parameters for convolutional layers in CNN and updating only the fully connected layers for each active learning iteration. We will also investigate hyper-parameter optimizations to select good set of parameters for CNN in a future work. Lastly, we plan to extend the current active learning framework on binary classifiers to multi-class classifiers and investigate if active learning can be used efficiently and effectively to select the best analysis results from several choices.

The active learning and classification codes used in this paper are available at https://github.com/SBU-BMI/quip_quality_analysis.git.

Acknowledgments. We would like to thank Dr. Yi Gao for providing code for segmentation algorithm used in the experiments.

References

1. Irshad H, Veillard A, Roux L, Racoceanu D. Methods for nuclei detection, segmentation, and classification in digital histopathology: a review—current status and future potential. IEEE reviews in biomedical engineering. 2014; 7:97-114.
2. Cohen S. Biophotonics in Pathology: Pathology at the Crossroads: IOS Press; 2013.
3. Rosai J, Ackerman LV. The pathology of tumors, part III: grading, staging and classification. CA: a cancer

journal for clinicians. 1979; 29:66-77.
4. Gurcan MN, Boucheron L, Can A, Madabhushi A, Rajpoot N, Yener B. Histopathological Image Analysis: A Review. IEEE Reviews in Biomedical Engineering. 2009; 2:147–71.
5. Xing F, Yang L. Robust nucleus/cell detection and segmentation in digital pathology and microscopy images: a comprehensive review. IEEE reviews in biomedical engineering. 2016; 9:234-63.
6. Gao Y, Ratner V, Zhu L, Diprima T, Kurc T, Tannenbaum A, et al. Hierarchical nucleus segmentation in digital pathology images. In SPIE Medical Imaging.: International Society for Optics and Photonics 2016;979117.
7. Wen S, Kurc TM, Gao Y, Zhao T, Saltz JH, Zhu W. A methodology for texture feature-based quality assessment in nucleus segmentation of histopathology image. Journal of Pathology Informatics. 2017; 8:38.
8. Kubat M, Matwin S. Addressing the curse of imbalanced training sets: one-sided selection. ICML. 1997; 97:179-186.
9. Settles B. Active Learning Literature Survey. Computer Sciences Technical Report. University of Wisconsin--Madison; 2009.
10. Cohn D, Atlas L, Ladner R. Improving generalization with active learning. Machine learning. 1994; 15:201-221.
11. Li X, Guo Y. Adaptive active learning for image classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013;859-866.
12. Demir B, Bruzzone L. A novel active learning method in relevance feedback for content-based remote sensing image retrieval. IEEE Transactions on Geoscience and Remote Sensing. 2015; 53:2323-2334.
13. Wang K, Zhang D, Li Y, Zhang R, Lin L. Cost-effective active learning for deep image classification. IEEE Transactions on Circuits and Systems for Video Technology. 2016.
14. Kutsuna N, Higaki T, Matsunaga SaOT, Yamaguchi M, Fujii H, Hasezawa S. Active learning framework with iterative clustering for bioimage classification. Nature communications. 2012; 3:1032.
15. Padmanabhan RK, Somasundar VH, Griffith SD, Zhu J, Samoyedny D, Tan KS, et al. An active learning approach for rapid characterization of endothelial cells in human tumors. PloS one. 2014; 9:e90495.
16. Doyle S, Monaco J, Feldman M, Tomaszewski J, Madabhushi A. An active learning based classification strategy for the minority class problem: application to histopathology annotation. BMC bioinformatics. 2011; 12:424.
17. Jones TR, Carpenter AE, Lamprecht MR, Moffat J, Silver SJ, Grenier JK, et al. Scoring diverse cellular morphologies in image-based screens with iterative feedback and machine learning. Proceedings of the National Academy of Sciences. 2009; 106:1826-1831.
18. Nalisnik M, Gutman DA, Kong J, Cooper LA. An interactive learning framework for scalable classification of pathology images. In Big Data (Big Data), 2015 IEEE International Conference on. 2015;928-935.
19. Gal Y, Islam R, Ghahramani Z. Deep Bayesian Active Learning with Image Data. arXiv preprint arXiv:1703.02910. 2017.
20. Lewis DD, Gale WA. A sequential algorithm for training text classifiers. In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval.: Springer-Verlag New York, Inc. 1994;3-12.
21. Sharma A, Kazerouni A, Saghar YN, Commean P, Tarbox L, Prior F. Framework for data management and visualization of the national lung screening trial pathology images. Pathology Informatics Summit. 2014.
22. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011; 12:2825-2830.
23. Breiman L. Random forests. Machine learning. 2001; 45:5-32.
24. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning. 2015;448-456.
25. Xu B, Wang N, Chen T, Li M. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853. 2015.
26. Al-Rfou R, Alain G, Almahairi A, Angermueller C, Bahdanau D, Ballas N, et al. Theano: A Python framework for fast computation of mathematical expressions. arXiv preprint arXiv:1605.02688. 2016.