Region segmentation in histopathological breast cancer images using deep convolutional neural network

CONFERENCE PAPER · APRIL 2015

DOI: 10.1109/ISBI.2015.7163815

6 AUTHORS, INCLUDING:

Hai Su

University of Kentucky

22 PUBLICATIONS **126** CITATIONS

SEE PROFILE

REGION SEGMENTATION IN HISTOPATHOLOGICAL BREAST CANCER IMAGES USING DEEP CONVOLUTIONAL NEURAL NETWORK

Hai Su[⋆]*, Fujun Liu*[†]*, Yuanpu Xie*[⋆]*, Fuyong Xing*[†]*, Sreenivasan Meyyappan*[⋆]*, Lin Yang*[⋆]

[⋆] J. Crayton Pruitt Family Dept. of Biomedical Engineering
[†] Dept. of Electrical and Computer Engineering, University of Florida, 32601, FL, USA

ABSTRACT

Computer aided diagnosis of breast cancers often relies on automatic image analysis of histopathology images. The automatic region segmentation in breast cancer is challenging due to: i) large regional variations, and ii) high computational costs of pixel-wise segmentation. Deep convolutional neural network (CNN) is proven to be an effective method for image recognition and classification. However, it is often computationally expensive. In this paper, we propose to apply a fast scanning deep convolutional neural network (fCNN) to pixel-wise region segmentation. The fCNN removes the redundant computations in the original CNN without sacrificing its performance. In our experiment it takes only 2.3 seconds to segment an image with size 1000×1000. The comparison experiments show that the proposed system outperforms both the LBP feature-based and texton-based pixel-wise methods.

1. INTRODUCTION

Breast cancer is one of the leading cancers worldwide. Meanwhile, it is the principle cause of death among women who are diagnosed cancers [1]. Pathological analysis plays a critical role in diagnosis, prognosis, and therapy planning for breast cancer. However, manual inspection of the histopathology images is not only time consuming, but also subjective. Computer-aided diagnosis (CAD) systems are promising technology for ensuring a standardized, objective pathology specimen analysis. In most CAD systems, region of interest (ROI) delineation is a prerequisite step [2].

ROI segmentation in pathological images has been widely studied. In [3], glandular components and their geometrical layout information are exploited for gland segmentation. ROI segmentation methods based on color-texture feature using expectation maximization (EM) algorithm are proposed in [4, 5]. In [6], an efficient GPU implementation of the method in [4] is presented. Foran *et al.* [7] propose to segment out the tumor region using texton features and logistic boosting. These existing methods mostly rely on hand-crafted features. The recent development of deep learning methods has shown that in many cases, handcrafted features are outperformed by the features learned from a large data set for particular tasks [8, 9, 10]. For some applications, it even achieves human-level performance [11].

In this paper, we propose a ROI segmentation scheme using a fast scanning deep convolutional neural network (fCNN) [12]. In our system, we adopt a pixel-wise classification method. An optimized forward-propagation network is utilized for classification. The raw pixel values are considered as the input. The features are learned by the network in a supervised manner. A common challenge in using CNN for pixel-wise segmentaiton is its efficiency. In the proposed system, the computational burden is significantly relieved by avoiding redundant computations in the convolutional and max-pooling layers. For our dataset, it takes only 2.3 seconds to segment an image with resolution 1000×1000.

2. METHOD

2.1. CNN Structure

An overview of the structure of the training network is shown in Fig.1. To perform pixel-wise classification, we take a small patch centered on the pixel as the input of the classifier. A convolutional layer with a kernel of size 5×5 is used to detect the low level features (*i.e.*, edges). Following the first convolutional layer a max-pooling layer with scale 2 is deployed to down-sampling the data. Next, a second convolutional layer followed by a max-pooling layer is deployed for detecting higher level image structures. The rest of the network consist of two fully connected layers with one having 64 neurons and the other having 2 neurons as the network output. In all the convolutional layers rectifier function [13] is used as the activation function. In the fully connected layers, dropout probability is set to 0.5 to prevent over fitting [14]. The convolutional layers and the max-pooling layers function as the feature extractors. The fully connected layers collectively behave as a classifier. We choose such structure because the computational cost in pixel-wise segmentation framework needs to be controlled carefully. The selected structure is moderately complex such that it provides sufficient classification performance yet introduces moderate computational cost.

This research is funded, in part, by NIH 1R01AR065479-01A1.

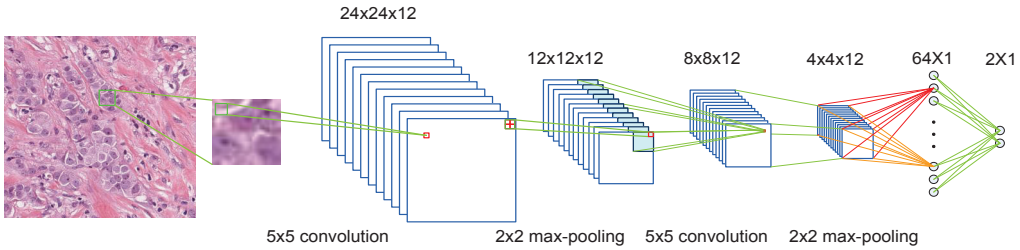


Fig. 1. A deep neural network with 7 layers. The input of the network is a 28×28 patch centered on the pixel to be classified. Besides the input layer, the network consists of two convolution layers, two max pooling layers, and two fully connected layers.

2.2. Fast Scanning CNN

Although the network model we choose has lower complexity than the model in AlexNet [14], it takes almost 1 minute to scan through an image of size 400×400 that contains 160,000 patches. Actually there is significant redundancy in the computation of the convolutional layers and the max-pooling layers when the network is used for a sliding window based segmentation task. The repeated convolutional computations and max-pooling operations reside in the overlapping region of one sliding window and its successive windows.

The strategy to avoid redundancy is to process the entire image directly. For example, the redundancy in one convolutional layer can be removed by applying the convolutional kernel to the entire image or the output of previous layers. The obtained feature map is called *extended feature map*. However, directly applying max-pooling to the entire extended feature map will lead to loss of information after the downsampling. This problem is depicted in Fig.2. An extended feature map is shown on the left hand side. A max-pooling output by applying a 2×2 max-pooling kernel to the entire extended feature map is shown on the right hand side. The color boxes correspond to sliding windows in the original image. The same color denotes the same sliding window. As one can tell that in the max-pooling output, the information in the green box is lost, *i.e.,* no pixels in the maxpooling output are generated from the green box in the extended map. In general, all the windows with their upper left most pixels located in the coordinates containing even indexes are lost in this direct max-pooling approach. This loss of information aggregates in the successive max-pooling layers in the CNN.

To preserve all the necessary information for each sliding window without redundancy, the max-pooling layer needs to be rearranged. The rearranged max-pooling layer consists of a set of *fragments*. Each fragment contains max-pooling results corresponding to a set of sliding windows. Assume the size of an input extended feature map F^l_{ext} is $d^l_{ext,v} \times d^l_{ext,h}$ at layer l and the max-pooling kernel size is k such that $\mathbf{mod}(d^l_{ext,v}, k) = 0$ and $\mathbf{mod}(d^l_{ext,h}, k) = 0$. The output of the max-pooling layer contains k^2 fragments. Each fragment is obtained as a result of max-pooling operations starting with k^2 different locations (pixels) in the upper left corner of F^l_{ext}. These locations can be represented by

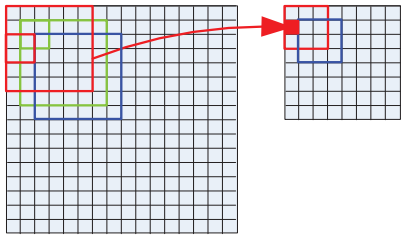


Fig. 2. A demonstration of the problem of loss of information in max-pooling layer when the downsampling is applied to the entire extended feature map directly. An extended feature map is shown on the left hand side. The max-pooling output is shown on the right hand side. The color boxes correspond to sliding windows in the original image. The same color indicates the same sliding window. As one can observe that in this approach, the information of the green box is lost after the max-pooling.

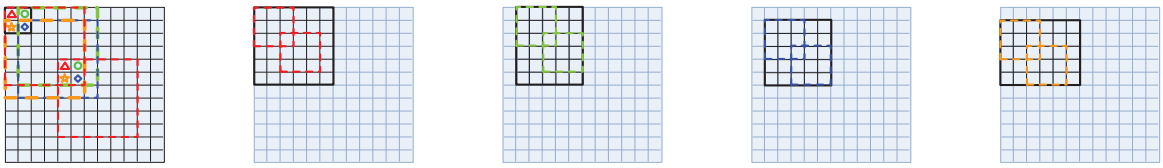
a set of 2D offsets defined by $\{\mathbf{O}(v,h) = \{0,1,\cdots,k-1\} \bigodot \{0,1,\cdots,k-1\}\}$, where $\bigodot$ denotes Cartesian product of location indices. These offsets are with respect to the upper left most pixel of F^l_{ext}. For $k = 2$, the offsets are $\mathbf{O}(v,h) = \{(0,0),(0,1),(1,0),(1,1)\}$, and the corresponding starting locations are marked out in the upper left corner in Fig.3(a) (the coordinates start with 1). A pixel at (i,j) in a fragment corresponding to the offset $\mathbf{O}(v,h)$ is the maximum value of the pixels $\{i',j'\}$ in F^l_{ext}. The correspondence can be calculated by:

$$(i-1)k+1+\mathbf{O}(v) \leq i' \leq (i-1)k+\mathbf{O}(v)+k, \quad (1)$$
$$(j-1)k+1+\mathbf{O}(h) \leq j' \leq (j-1)k+\mathbf{O}(h)+k, \quad (2)$$

where $\mathbf{O}(\bullet) = \{0,1\}$ for $k = 2$. The size of a fragment $(f_{mp,v}, f_{mp,h})$ depends on its offset and can be calculated by:

$$f_{mp,v} = \mathbf{floor}[\frac{d^l_{ext,v} - \mathbf{O}(v)}{k}], \quad (3)$$
$$f_{mp,h} = \mathbf{floor}[\frac{d^l_{ext,h} - \mathbf{O}(h)}{k}]. \quad (4)$$

An example of a pair of convolution-max-pooling layers is shown in Fig.3. Fig.3(a) is the extended feature map F^l_{ext}.



(a) (b) (c) (d) (e)

Fig. 3. An illustration of the structure of max-pooling layer in fCNN. A 2×2 max-pooling kernel is used for illustration purpose. (a) The 12×12 extended feature map computed by the previous convolutional layer. Note that this extended feature map contains all the information of the entire input map rather than just one sliding window. The color square boxes correspond to 5 sliding windows selected for this illustration; (b) A fragment that contains information of all the sliding windows whose first pixels locate in odd coordinates in (a) (numbering starting with 1). All the locations are marked with red triangle; (c)-(e) Each fragment contains all the information of the sliding windows starting with the locations marked by green circles, blue diamonds, and orange stars, respectively.

The bold boxes in Fig.3(b)-(e) are fragments obtained by the proposed max-pooling operation based on a 2×2 kernel. In Fig.3(a), the markers indicate locations of the pixels. For example, all the pixels with coordinates $\{(i,j)\|$ mod(i,2)=1, mod(j,2)=1$\}$, where i indexes the rows and j for the columns, are considered as red triangle pixels. Similarly all pixels with coordinates $\{\ (i,j)\|$ mod(i,2)=1, mod(j,2)=0$\}$ are considered as green circle pixels. To avoid information loss in max-pooling, the output are stored in different fragments depending on the location of the first pixel of the sliding window. For the sliding windows starting with red triangle pixels, the output is in the fragment shown in Fig.3(b). For the windows starting with green circle pixels, the output is in Fig.3(c). The windows corresponding to blue diamond and orange star pixels are in Fig.3(d) and (e). Two sample windows are shown in each fragment. They correspond to the eight sliding windows starting with the marked pixels in Fig.3(a). The size of the fragment varies because of the **floor**$(\cdot)$ operation. All fragments except for the one with $(0,0)$ offset tend to ignore some outermost rows and columns. This does not affect the performance because if the maximum value appears in these locations they are included in the fragment with $(0,0)$ offset.

The spatial relationships described above are deterministic for a pair of extended map and its following max-pooling layer. Such deterministic property ensures that a stacked fast scanning implementation can generate the same result as the original CNN used in sliding window framework.

3. EXPERIMENTS

3.1. Data

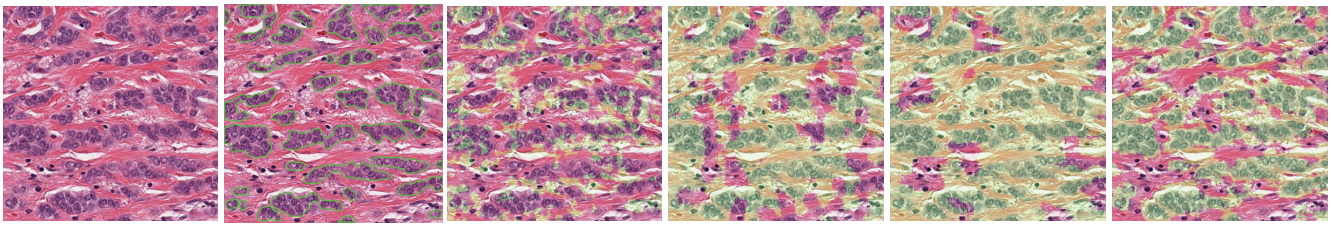
We conducted experiment with 92 images cropped from 20 patients in The Cancer Genome Atlas (TCGA) breast cancer data set. 75 are used for training and 17 are used for testing. Each image patch has the size of 1000×1000. For each pixel used for training, four gray scale patches at different scales (*i.e.,* 22×22, 28×28, and 40×40, and 48×48) are cropped and resized to the size 28×28 to increase the robustness. In total about a half million training samples are generated. A CNN with the architecture shown in Fig.1 is trained and converted into the fast scanning structure for testing. We conducted the experiment on a workstation with Xeon E5-1650 3.5GHz processor and 128 GB memory.

3.2. Experimental Results

The proposed framework is compared against three texture classification methods, including raw pixel patch with large scale support vector machine (RPLSVM) [15], local binary pattern feature with large scale SVM (LBPLSVM) ,and texton histogram with logistic booting (THLB) [7]. In RPLSVM, patches of size 28×28 are cropped. In LBPLSVM, LBP feature of a 3×3 neighborhood is calculated. In the above two methods, about half million sample are used for training. In the experiment with THLB, 32 textons are generated by applying K-means clustering on the output of 38 root filter set (RFS) filters of size 5. More than a half million training samples are used for training over 20 epoches. The sliding window size of all the experiments is set to 28.

We evaluate the accuracy, efficiency, and scalability of the proposed method. In the accuracy evaluation, we compare the methods both qualitatively and quantitatively. In Fig.4, the comparison of one randomly selected images is displayed. It can be seen that due to the intra-class variation, RPLSVM, LBPLSVM and THLB cannot produce correct ROI segmentation. Our method is robust to such variations. In quantitative comparison the performance is measured by precision $P = \frac{TP}{TP+FP}$, recall $R = \frac{TP}{TP+FN}$ and $F_1 = \frac{2PR}{P+R}$ score, where TP is true positive, FP denotes false positive, and FN represents false negative. The comparison is shown in Table 1. Our method achieves an average $F_1 = 0.85$. A box plot of F_1 score is shown in Fig.5(a). It can be seen that the three methods are sensitive to the intra-class variations present in different images. Ours is consistently robust for various testing images.

To show the excellent efficiency and scalability of fCNN, we conducted experiments with the original implementation of CNN and the proposed method using images with different

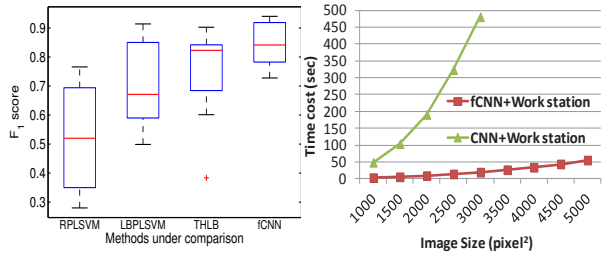


(a) (b) (c) (d) (e) (f)

Fig. 4. The qualitative comparisons of several automatic ROI (epithelial regions inclosed by green contours in (b)) segmentation results. (a) Original images; (b) The ground truth annotation; (c) RPLSVM; (d) LBPLSVM; (e) THLB; (f) fCNN.

Table 1. The quantitative comparison. Mean and variance of the precision, recall, and F1 score are evaluated.

	P_{avg}	R_{avg}	$F1_{avg}$	P_{var}	R_{var}	$F1_{var}$
RPLSVM	0.60	0.47	0.52	0.03	0.04	0.04
LBPLSVM	0.62	0.83	0.70	0.02	0.03	0.023
THLB	0.74	0.81	0.75	0.04	0.05	0.03
fCNN	0.91	0.82	0.85	0.015	0.02	0.01



(a) (b)

Fig. 5. (a): The box plot of F_1 score of different automatic ROI segmentation methods. (b) The evaluation of scalability of standard CNN and fCNN.

sizes. A CNN with the same parameter is implemented using C++ while the fCNN is implemented in MATLAB. The time cost along increase of image size is shown in Fig.5(b). It can be observed that fCNN is at least 20X faster than the original CNN. In addition, as we can see in Fig.5(b), fCNN shows much better scalability as the size of images increases.

4. CONCLUSION

In this paper, we proposed a robust, efficient and scalable method based on fast scanning deep neural network for region segmentation in histopathological breast cancer images. The proposed method is comprehensively evaluated in terms of performance, efficiency and scalability. It outperforms the three texture classification methods in segmentation accuracy and its time cost is sufficiently low to make it a desirable tool for practical biomedical image analysis.

References

[1] "Breast cancer statistics worldwide," http://www.worldwidebreastcancer.com/learn/breast-cancer-statistics-worldwide/.

[2] X. Zhang, W. Liu, M. Dundar, S. Badve, and S. Zhang, "Towards large-scale histopathological image analysis: Hashing-based image retrieval," *TMI*, vol. 34, no. 2, pp. 496–506, Feb 2015.

[3] K. Nguyen, A.K. Jain, and R.L. Allen, "Automated gland segmentation and classification for gleason grading of prostate tissue images," in *Proc. of ICPR*, Aug 2010, pp. 1497–1500.

[4] J. Kong, H. Shimada, K. Boyer, J. Saltz, and M. Gurcan, "Image analysis for automated assessment of grade of neuroblastic differentiation," in *Proc. of ISBI*, Apr 2007, pp. 61–64.

[5] H. Kong, M. Gurcan, and K. Belkacem-Boussaid, "Partitioning histopathological images: An integrated framework for supervised color-texture segmentation and cell splitting," *TMI*, vol. 30, no. 9, pp. 1661–1677, Sep 2011.

[6] A. Ruiz, J. Kong, M. Ujaldon, K. Boyer, J. Saltz, and M. Gurcan, "Pathological image segmentation for neuroblastoma using the gpu," in *Proc. of ISBI*, May 2008, pp. 296–299.

[7] D.J. Foran, L. Yang, O. Tuzel, W. Chen, J. Hu, T.M. Kurc, R. Ferreira, and J.H. Saltz, "A cagrid-enabled, learning based image segmentation method for histopathology specimens," in *Proc of ISBI*, Jun. 2009, pp. 1306–1309.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc of CVPR*, 2014.

[9] H. Chang, Y. Zhou, P. Spellman, and B. Parvin, "Stacked predictive sparse coding for classification of distinct regions in tumor histopathology," in *Proc of ICCV*, 2013, pp. 169–176.

[10] H. Suk and D. Shen, "Deep learning-based feature representation for ad/mci classification," in *Proc. of MICCAI*, vol. 8150, pp. 583–590. 2013.

[11] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," in *Proc of CVPR*, 2014.

[12] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *Proc. of ICIP*, 2013.

[13] Vinod Nair and Geoffrey E Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. of ICML*, 2010, pp. 807–814.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. of NIPS*, 2012, pp. 1097–1105.

[15] K. Zhang, L. Lan, Z. Wang, and F. Moerchen, "Scaling up kernel svm on limited resources: A low-rank linearization approach," in *Proc. of AISTATS*, 2012, pp. 1425–1434.