CHAPTER

7 Deep Voting and Structured Regression for Microscopy Image Analysis

Yuanpu Xie, Fuyong Xing, Lin Yang
University of Florida, Gainesville, FL, United States

CHAPTER OUTLINE

CHAPTER POINTS

- In this chapter, we present two novel frameworks for robust cell detection in microscopy images
- The presented methods provide two potential views toward object detection in a crowd scenario. Both methods go beyond the traditional sliding window based classification method
- We demonstrate that the convolutional neural network is a potentially powerful tool, which still needs future exploitation in biomedical image analysis

Deep Learning for Medical Image Analysis. DOI: 10.1016/B978-0-12-810408-8.00009-2

7.1 DEEP VOTING: A ROBUST APPROACH TOWARD NUCLEUS LOCALIZATION IN MICROSCOPY IMAGES

7.1.1 INTRODUCTION

Accurate localization of nucleus centers in microscopy images is fundamental to many subsequent computer-aided biomedical image analysis such as cell segmentation, cell counting, tracking, etc. Unfortunately, robust nucleus localization, especially in microscopy images exhibiting dense nucleus clutters and large variations in terms of both nucleus sizes and shapes, has proven to be extremely challenging [1]. In the past few years, a large number of methods have been proposed, spatial filters [2] have been applied for automatic nuclei localization, and graph partition methods have been reported in [3,4] to automatically detect cells. Since cells are approximately circular, Parvin et al. [5] have introduced a kernel based radial voting method to iteratively localize cells, which is insensitive to image noises. Several other radial voting-based methods are presented in [6–9] for automatic cell detection on histopathology images. Other localization methods based on gradient vector [10], Laplacian-of-Gaussian (LOG) with adaptive scale selection [2,11] and concave points [12,13] can also be found in the literature. However, because of the large variations in microscopy modality, nucleus morphology, and the inhomogeneous background, it remains to be a challenging topic for these non-learning methods.

Supervised learning based methods have also attracted a lot of interest due to their promising performance. For example, a general and efficient maximally stable extremal region (MSER) section method is presented in [14], in which MSER is used to efficiently generate region candidates of possible cell regions, a structured SVM [15] is then utilized to score each region, and finally the configurations of the selected regions are achieved using dynamic programming to explore the tree structure of MSER. However, in images that exhibit strong inhomogeneous background and foreground differences, the MSER detector cannot generate feasible region candidates, and thus its usage is limited. Recently, in computer vision community, codebook based hough transformation, such as implicit shape model [16], has also been widely studied and shown to be able to produce promising performance in general object detection tasks. Random forest method [17] or singular vector decomposition [18] is applied to learn a discriminative codebook. The so-called class-specific hough forest is able to directly map the image patches to the probabilistic votes and to the potential locations of the object center. However, it only associates one voting offset vector to an image patch during the training process, and in the testing stage, one patch can only vote along the directions that have already appeared in the training data. In addition, this method is not designed to detect objects that appear as a crowd, which is often the case when tackling nuclei localization in microscopy images.

Despite their success, the performance of aforementioned works heavily rely on various handcrafted features such as shapes, gradients, colors, etc. Fusion strategy is often used to combine those features for better results [19]. Such methods may work well on some specific types of microscopic images but will generally fail on others. To alleviate this problem and liberate the user from choosing proper handcrafted

features, the deep learning technique is adopted in the proposed method to free the user from laborious feature selections and parameter tuning [20–23]. Deep learning is a revived topic in recent years and achieved outstanding performance in both natural image analysis and biomedical image analysis [24–27]. Local features are typically extracted automatically from image patches via a cascade of convolutional neuron networks (CNNs). These CNNs are usually connected with fully connected networks, and the training label information is fed back to update the weights of CNNs via backpropagation [28].

Numerous works based on CNN structure can be found in the literature, with their applications ranging from fundamental low-level image processing to more sophisticated vision tasks. Farabet et al. [29] exploit a multi-scale strategy to extract hierarchical features for scene labeling task. Krizhevsky et al. [30] provide a GPU implementation of CNN and achieve record-breaking performance in ImageNet classification tasks. Besides classification, CNN has also been used as a regressor, and it is shown to be able to capture geometric information exhibited in the image. Toshev and Szegedy [31] adopt a similar architecture as that in [30] and achieve state-of-the-art performance of human pose estimation by directly doing regression on the human joints positions. A mask regression method has been proposed to do object detection in [32]. Other successful applications of deep learning in nature image analysis are reported in [32,33].

The CNN structure based learning framework has been proven successful in various biomedical image analysis tasks. Ciresan et al. [34] exploit a convolutional neural network framework to detect mitosis in breast cancer histology images. Another similar method is reported in [35] for membrane neuronal segmentation in a microscopy image. Other successful applications of deep learning in medical image analysis include Multi-Modality Isointense Infant Brain Image Segmentation [36] and predicting Alzheimer's disease using 3D convolutional neural networks [37]. Further successful applications of CNN are reported in [38–40]. It is also worth noting that image representation of biomedical images learned by convolutional neural network performs better than the handcrafted features [41–44,26]. Variations of 2D recurrent neural network [45,46] and the recently proposed fully convolutional neural network [47] have also achieved promising results in medical image analysis [48,49].

In this chapter, we present a novel deep neural network aiming at robust nucleus localization in microscopy images. Unlike traditional methods that either take advantage of CNN as a pixel-wise classifier or as a regressor on the entire image feature scope, we improve the CNN structure by introducing a hybrid nonlinear transformation capable of learning both the voting offset vectors and corresponding voting confidence jointly for robust center localization. Unlike [17,33], this method assigns more than one pair of predictions for each image patch, which, together with the specific loss function, render this method more robust and capable of handling touching cases. This model, which is named a deep voting model, can be viewed as an implicit hough-voting codebook [50], using which every testing patch votes toward several directions with specific confidence. Given a test image, we collect all the votes with quantized confidence cumulatively and estimate the voting density map in

a way similar to Parzen-window estimation [51]. Although this method is inherently suited for general object detection tasks, we focus on nuclei center localization in this work. Comparative experimental results show superiority of the proposed deep voting method.

7.1.2 METHODOLOGY

Before going into details of our proposed model, we start with a brief introduction to the widely used CNN model. A CNN framework typically contains a stack of alternating convolutional layers (C) and pooling layers (P), followed by one or more fully connected layers (FC). Convolutional layer serves the purpose to extract patterns by convolving the filter banks on the input images (or feature maps). The linear combination of the convolutional outputs are passed to a nonlinear activation function (logistic, ReLu, etc.) to generate the output feature maps. Define L as the total number of layers, with the input layer as the first layer. Define $\mathbf{f}_i^l$ as the ith feature map in lth layer. In general, we have

$$\mathbf{f}_i^l = h(\sum_j \mathbf{f}_j^{l-1} * \mathbf{k}_{ji}^l + \mathbf{b}_i^l), \tag{7.1}$$

where $\mathbf{b}_i^l$ is the bias, $\mathbf{k}_{ji}^l$ is the convolution kernel that works on the input feature maps in layer $l-1$, and h is a nonlinear activation function.

Pooling layer (P) performs down-sampling of the input feature maps by a predefined factor, the number of output feature maps is exactly the same as that of the input maps. One of the most widely used pooling layers is called a max-pooling layer, it is used to reduce the size of the feature maps and introduce local shift and translation invariance properties to the network at the same time.

Fully connected layer (FC) takes all the outputs from the previous layer and connects them to every unit in this layer. If the previous layer is a convolutional layer and consists of 2D feature maps, the conventional way is to reshape those feature maps into one feature vector. This layer is used to perform high-level inference by multiplying the input vector with the weight matrix and following by a nonlinear transformation. Actually, this type of layer can also be viewed as a special case of convolutional layer with filter size 1×1. More formally, denote by $\mathbf{W}^l$ the weights for lth layer, the output of this layer can be given by

$$\mathbf{f}_i^l = h(\mathbf{W}^l \mathbf{f}_i^{l-1} + \mathbf{b}_i^l). \tag{7.2}$$

When CNN is applied as a classifier, softmax is used at the top layer to encode the probability distribution of a certain class denoted by the corresponding unit. Let $\mathbf{h}$ denote the activation function of the nodes in penultimate layer, and denote by $\mathbf{W}$ the weight connecting the last layer and the penultimate layer. If the total input to one node of the softmax layer is $a_j = \sum_k h_k W_{kj}$, then we can have $p_j = \frac{exp(a_j)}{\sum exp(a_j)}$, and the class label can be obtained by $\hat{i} = \arg\max_i p_j$.

Denote by E the final loss function defined on a set of training data. Let Θ represent the model's parameters composed of the kernel weights and biases. The model can be learned iteratively using a stochastic gradient descent algorithm of the following form:

$$\Theta(t) = \Theta(t-1) - \mu \frac{\partial E}{\partial \Theta}, \tag{7.3}$$

where $\Theta(t)$ represents the model's parameter in the tth iteration and μ denotes the learning rate. In practice, some optimization and regularization tricks, including normalization and shuffling of the training data, adaptive learning rate, momentum, and dropout, can be used to improve the performance.

7.1.2.1 *Learning the Deep Voting Model*

Preprocessing. In the setting of this method, each image patch votes for several possible nucleus positions (the coordinates for the nucleus center are specified by voting offsets) with several specific voting confidence, a high voting confidence indicates that this image patch is highly informative to the corresponding vote. Please note that the number of votes is predefined. In this work, we propose to learn an implicit codebook using CNN for these voting offsets and corresponding voting confidence for the testing image patches.

Denote by $\mathcal{P}$ the input image patch space composed of a set of multi-channel squared image patches of size $d \times d \times c$, where d and c represent the patch height/width and the number of image channels, respectively; and by $\mathcal{T}$ the target space, defining the space of the proposed *target information*. We define the *target information* as the coalescence of voting offsets and voting confidence. For each training image patch, we first obtain the coordinates of k nucleus centers from a group of human annotation (ground-truth) that are closest to the center of the image patch. Let $\{\mathcal{D}^i = (\mathcal{P}^i, \mathcal{T}^i)\}$ represent the ith training data where $\mathcal{P}^i \in \mathcal{P}$ is the ith input image patch and $\mathcal{T}^i \in \mathcal{T}$ is the corresponding *target information*. $\mathcal{T}^i$ can be further defined as $\{\alpha_j^i, \mathbf{v}_j^i\}_{j=1}^k$, where k denotes the number of total votes from each image patch and $\mathbf{v}_j^i$ is the 2D offset vector that is equal to the displacement from the coordinate of the jth nearest ground-truth nucleus aligned to the center of patch $\mathcal{P}^i$. α_j^i represents the voting confidence corresponding to $\mathbf{v}_j^i$. In this work, we define it based on the length of the voting offset $|\mathbf{v}_j^i|$ as

$$\alpha_j^i = \begin{cases} 1, & \text{if } |\mathbf{v}_j^i| \le r_1, \\ \frac{1}{1+\beta|\mathbf{v}_j^i|}, & \text{if } r_1 < |\mathbf{v}_j^i| \le r_2, \\ 0, & \text{otherwise}, \end{cases} \tag{7.4}$$

where β represents the confidence decay ratio. r_1, r_2 are used to tune the voting range and are chosen as d and $1.5d$, respectively. Please note that (7.4) is only one of the possible definitions of α_j^i, instead, it can be defined based on many interesting

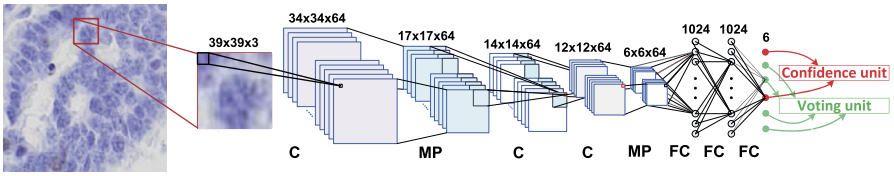


FIGURE 7.1

The architecture of the proposed deep voting model. The C, MP, and FC represent the convolutional layer, max-pooling layer, and fully connected layer, respectively. The two different types of units (voting units and weight units) in the last layer are marked with different color. Please note that in this model the number of votes for each patch (k) is 2.

properties of the training patches such as foreground area ratio, probability of being a specific class membership, distance from the patch center to the voting position, etc.

Hybrid Nonlinear Transformation. In order to jointly learn the voting offset vectors and voting confidence, we categorize the units of the output layer in the deep voting model as *voting units* and *confidence units*. As is shown in Fig. 7.1, we use two different colors to differentiate two types of units in the last layer: red nodes represent the confidence units and green units the voting units.

Since the voting offset is 2-dimensional, every voting confidence unit corresponds to 2 voting offset units. Observe that voting units can take any values and are used to specify the positions that each patch will vote to. The values for confidence units are confined to [0, 1] and are used as weights for the votes from every testing image patch. Existing activation functions (sigmoid, linear, or ReLu) associated to the output layer treat all the units in the same way and do not satisfy our needs. In order to jointly encode the geometric voting offset and the voting confidence information in the last layer of the model, we therefore introduce a **hybrid nonlinear transformation** Hy as the new activation function. Units of the output layer are treated differently based on their specific categories. For voting units, we use a simple linear activation function, for confidence units, we use the sigmoid activation function. Therefore, Hy is given by

$$Hy(x) = \begin{cases} sigm(x), & \text{if } x \text{ is the input to the } \textit{confidence units}, \\ x, & \text{otherwise}, \end{cases} \tag{7.5}$$

where $sigm$ denotes the sigmoid function. Note that, similar to sigmoid activation in neural networks, Hy here is a point-wise operator. Alternatively, Hy can be constructed using other type of activation functions based on specific needs, e.g., ReLu, tanh, etc.

Inference in Deep Voting Model. In this method, we adopt CNN architecture during the codebook learning as shown in Fig. 7.1. Denote by L the number of layers, let functions $f_1, \ldots, f_L$ represent each layer's computations (e.g., linear transformations

and convolution, etc.), and denote by $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L$ the parameters of those layers. Note that pooling layers have no parameters. This model can be viewed as a mapping function, ψ, which maps input image patches to *target information*. Denote by $\circ$ the composition of functions, then ψ can be further defined as $f_L \circ f_{L-1} \circ \cdots \circ f_1$ with parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L\}$.

Assume we are given the training data $\{\mathcal{D}^i = (\mathcal{P}^i, \mathcal{T}^i)\}_{i=1}^N$, where N is the number of training samples. The parameters of the proposed model can be evaluated by solving the following optimization problem:

$$\arg\min_{\boldsymbol{\Theta}} \frac{1}{N} \sum_{i=1}^{N} l(\psi(\mathcal{P}^i; \boldsymbol{\Theta}), T^i), \tag{7.6}$$

where l is the loss function defined on one training sample. The detailed definition of l is given in the following paragraphs. Let $\psi(\mathcal{P}^i; \boldsymbol{\Theta}) = \{w_j^i, \boldsymbol{d}_j^i\}_{j=1}^k$ denote the model's output corresponding to input $\mathcal{P}^i$, where w_j^i and $\boldsymbol{d}_j^i$ are predicted voting confidence and offsets, respectively. Recall that $\mathcal{T}^i = \{\alpha_j^i, \mathbf{v}_j^i\}_{j=1}^k$ is the *target information* of $\mathcal{P}^i$, and k represents the number of votes from each image patch. The loss function l defined on $(\mathcal{P}^i, \mathcal{T}^i)$ can be given by

$$l(\psi(\mathcal{P}^i; \boldsymbol{\Theta}), \mathcal{T}^i) = \sum_{j=1}^{k} \frac{1}{2} (\alpha_j^i w_j^i \|\boldsymbol{d}_j^i - \boldsymbol{v}_j^i\|^2 + \lambda (w_j^i - \alpha_j^i)^2), \tag{7.7}$$

where λ is a regularization factor used to tune the weights of loss coming from voting confidence and offsets. There are two benefits of the proposed loss function: (i) The $\alpha_j^i w_j^i$ in the first term penalizes uninformative votes that have either low predicted voting confidence or low voting confidence in the *target information*. This property alleviates the issue of fixed number of votes for each image patches, for uninformative votes, the loss coming from voting offsets vanishes, and this loss function degenerates into a squared error loss defined on the voting confidence. (ii) The second term not only punishes the error coming from voting confidence, it also acts as a regularization term to prevent the network from producing trivial solutions (setting all voting confidence to zero).

We use stochastic gradient descent and backpropagation algorithm [52] to solve (7.6). In order to calculate the gradients of (7.6) with respect to the model's parameters $\boldsymbol{\Theta}$, we need to calculate the partial derivatives of the loss function defined on one single example with respect to the inputs of the nodes in the last layer. As shown in Fig. 7.1, the outputs of the proposed model are organized as k pairs of {confidence units, offset units}. Let $\{I w_j^i, \boldsymbol{I} \boldsymbol{v}_j^i\}_{j=1}^k$ represent inputs to the units in the last layer for $(\mathcal{P}^i, \mathcal{T}^i)$ in the forward pass, we can easily get the model's output, $w_j^i = Hy(I w_j^i)$ and $\boldsymbol{v}_j^i = Hy(\boldsymbol{I} \boldsymbol{v}_j^i)$. The partial derivatives of (7.7) with respect to

Iw_j^i and $\boldsymbol{Iv}_j^i$ are then given as

$$\frac{\partial l(\psi(\mathcal{P}^i;\boldsymbol{\Theta}),\mathcal{T}^i)}{\partial Iw_j^i} = \frac{\partial l(\psi(\mathcal{P}^i;\boldsymbol{\Theta}),\mathcal{T}^i)}{\partial w_j^i}\frac{\partial w_j^i}{\partial Iw_j^i}$$
$$= (\frac{1}{2}\alpha_j^i\|\boldsymbol{d}_j^i - \boldsymbol{v}_j^i\|^2 + \lambda(w_j^i - \alpha_j^i))Iw_j^i(1 - Iw_j^i), \tag{7.8}$$

$$\frac{\partial l(\psi(\mathcal{P}^i;\boldsymbol{\Theta}),\mathcal{T}^i)}{\partial \boldsymbol{Id}_j^i} = \frac{\partial \boldsymbol{d}_j^i}{\partial \boldsymbol{Id}_j^i}\frac{\partial l(\psi(\mathcal{P}^i;\boldsymbol{\Theta}),\mathcal{T}^i)}{\partial \boldsymbol{d}_j^i}$$
$$= \alpha_j^i w_j^i(\boldsymbol{d}_j^i - \boldsymbol{v}_j^i), \tag{7.9}$$

where $\boldsymbol{Id}_j^i$, $\boldsymbol{d}_j^i$ and $\frac{\partial l(\psi(\mathcal{P}^i;\boldsymbol{\Theta}),\mathcal{T}^i)}{\partial \boldsymbol{Id}_j^i}$ are 2D vectors. After getting the above partial derivatives, the following inferences used to obtain the gradients of (7.6) with respect to the model's parameters are exactly the same to the classical backpropagation algorithm used in conventional CNN [52].

7.1.3 WEIGHTED VOTING DENSITY ESTIMATION

Given a testing image, denote by $\mathcal{P}(\boldsymbol{y})$ an image patch centered at position $\boldsymbol{y}$ on the testing image, $\boldsymbol{y}$ is a 2D vector representing the coordinate of the center of testing image patch $\mathcal{P}(\boldsymbol{y})$, and let $\{w_j(\boldsymbol{y}), \boldsymbol{d}_j(\boldsymbol{y})\}_{j=1}^k = \psi(\mathcal{P}(\boldsymbol{y});\boldsymbol{\Theta})$ represent the model's output for $\mathcal{P}(\boldsymbol{y})$, where $w_j(\boldsymbol{y})$ and $\boldsymbol{d}_j(\boldsymbol{y})$ represent the jth voting confidence and offset vector for patch $\mathcal{P}(\boldsymbol{y})$, respectively. Furthermore, denote by $V(\boldsymbol{x}|\mathcal{P}(\boldsymbol{y}))$ the accumulated voting score of position $\boldsymbol{x}$ coming from patch $\mathcal{P}(\boldsymbol{y})$. Similar to the kernel density estimation using Parzen-window, we can write $V(\boldsymbol{x}|\mathcal{P}(\boldsymbol{y}))$ as

$$V(\boldsymbol{x}|\mathcal{P}(\boldsymbol{y})) = \sum_{j=1}^{k} w_j(\boldsymbol{y})\frac{1}{2\pi\sigma^2}\exp(-\frac{\|(\boldsymbol{y} - \boldsymbol{d}_j(\boldsymbol{y})) - \boldsymbol{x}\|^2}{2\sigma^2}). \tag{7.10}$$

In this work, we restrict the votes contributing to the location $\boldsymbol{x}$ to be calculated from a limited area represented by $B(\boldsymbol{x})$, which can be a circle or a bounding box centered at position $\boldsymbol{x}$. To compute the final weighted voting density map $V(\boldsymbol{x})$, we accumulate the weighted votes from every testing image patch centered at $\boldsymbol{y} \in B(\boldsymbol{x})$,

$$V(\boldsymbol{x}) = \sum_{\boldsymbol{y}\in B(\boldsymbol{x})} V(\boldsymbol{x}|\mathcal{P}(\boldsymbol{y})). \tag{7.11}$$

After obtaining the voting density map $V(\boldsymbol{x})$, a small threshold $\xi \in [0, 1]$ is applied to remove the values smaller than $\xi \cdot \max(V)$. The following procedure for nucleus localization is to find all the local maximum locations in V.

In order to reduce the complexity for computing the voting density map in (7.11), we provide a fast implementation: Given a testing image, instead of following the

computation order specified by (7.10) and (7.11), we go through image patch $\mathcal{P}(\mathbf{y})$ centered at every possible position $\mathbf{y}$ in the testing image, and add weighted votes $\{w_j(\boldsymbol{y})\}_{j=1}^{k}$ to pixel $\{\boldsymbol{y}-\boldsymbol{d}_j(\boldsymbol{y})\}_{j=1}^{k}$ in a cumulative way. The accumulated vote map is then filtered by a Gaussian kernel to get the final voting density map $V(\boldsymbol{x})$.

7.1.4 EXPERIMENTS

Data and Implementation Details. The proposed method has been extensively evaluated using 44 Ki67-stained NET microscopy images. Testing images are carefully chosen to cover challenging cases like touching cells, inhomogeneous intensity, blurred cell boundaries, and weak staining.

The architecture of the model is summarized in Fig. 7.1. The patch size is set to be 39×39. The convolutional kernel sizes are set to be 6×6, 4×4, and 3×3 for the three convolutional layers, respectively. All the max pooling layers have a window size of 2×2, and a stride of 2. A dropout ratio 0.25 is used at the training stage to prevent over fitting and the learning rate is set to be 0.0001. β and λ are set to be 0.5 and 384 in (7.4) and (7.7), respectively. The threshold ξ is set to be 0.2. The Parzen-window size is 5×5, and σ is 1 in (7.10). These parameters are set either by following conventions from the existing works or by empirical selection. In fact, our method is robust to hyper-parameter selection. The implementation is based on the fast CUDA kernel provided by Krizhevsky [30]. The proposed model is trained and tested using a PC with NVIDIA Tesla K40C GPU and an Intel Xeon E5 CPU.

Evaluate the Deep Voting Model. For quantitative evaluation, we define the ground-truth region as the circular regions of radius r centered at every human annotated nucleus center. In this work, r is roughly chosen to be half of the average radius of the nucleus. True positive (TP) detections are defined as detected results that lie in the ground-truth region. False positive (FP) detections refer to detection results that lie outside of the ground-truth region. The human annotated nucleus centers that do not match with any detection result are considered to be false negative (FN). Based on TP, FP, and FN, we can calculate the recall (R), precision (P), and F_1 scores, which are given by $R = \frac{TP}{TP+FN}$, $P = \frac{TP}{TP+FP}$, and $F_1 = \frac{2PR}{P+R}$. In addition to P, R, and F_1 score, we also compute the mean μ and standard deviation σ of the Euclidean distance (defined as $\mathbf{E_d}$) between the human annotated nucleus centers and the true positive detections, and also the absolute difference (defined as $\mathbf{E_n}$) between the number of human annotation and the true positive nucleus centers.

We compare deep voting with no stride (referred to as **DV-1**), deep voting with stride 3 (referred to as **DV-3**), and standard patch wise classification using CNN (referred to as **CNN+PC**). **CNN+PC** has the same network architecture to the proposed model, except that its last layer is a softmax classifier and results in probability map for every testing image. Please note that the fast scanning strategy produces the same detection result as **DV-1**.

We also compare this algorithm with three other state of the art procedures, including kernel based Iterative Radial Voting method (**IRV**) [5], structured SVM based Non-overlapping Extremal Regions Selection method (**NERS**) [14], and Image based

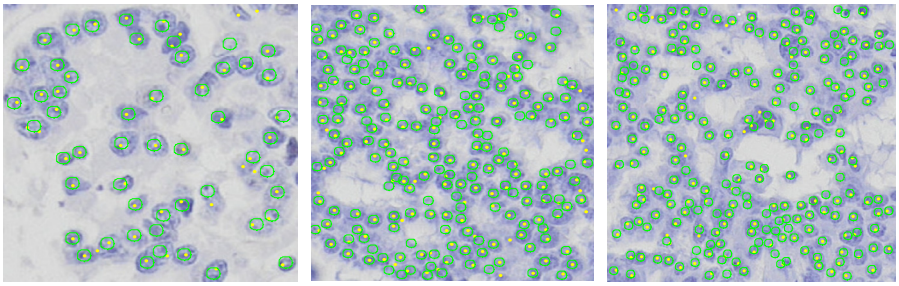


FIGURE 7.2

Nucleus detection results using deep voting on several randomly selected example images from Ki-67 stained NET dataset. The detected cell centers are marked with yellow dots. The ground-truth is represented with small green circles.

Table 7.1 Nucleus localization evaluation on the three data sets. μ_d, μ_n, and σ_d, σ_n represent the mean and standard deviation of the two criteria $\mathbf{E_d}$ and $\mathbf{E_n}$, respectively

Methods	P	R	F_1	$\mu_d \pm \sigma_d$	$\mu_n \pm \sigma_n$
DV-1	0.852	0.794	**0.8152**	2.98 ± 1.939	$\mathbf{9.667 \pm 10.302}$
DV-3	0.727	0.837	0.763	3.217 ± 2.069	17.238 ± 18.294
CNN+PC	0.673	0.9573	0.784	$2.51 \pm \mathbf{1.715}$	37.714 ± 39.397
NERS [14]	0.859	0.602	0.692	2.936 ± 2.447	41.857 ± 57.088
IRV [5]	0.806	0.682	0.718	3.207 ± 2.173	17.714 ± 16.399
ITCN [53]	0.778	0.565	0.641	3.429 ± 2.019	33.047 ± 46.425

Tool for Counting Nuclei method (**ITCN**) [53]. For fair comparison, we carefully preprocess the testing image for ITCN and IRV. We do not compare the proposed deep voting model with other methods, such as SVM and boosting using handcrafted features, because CNN is widely proven in recent literatures to produce superior performance in classification tasks.

Experimental Results. Fig. 7.2 shows some qualitative results using three randomly selected image patches. Hundreds of nuclei are correctly detected using our method. Deep voting is able to handle the touching objects, weak staining, and inhomogeneous intensity variations. Detailed quantitative results are summarized in Table 7.1. Fast scanning produces exactly the same result as **DV-1** and thus is omitted from the table. It can be observed that deep voting consistently outperforms other methods, especially in terms of F_1 score. Although NERS [14] achieves comparable P score, it does not produce good F_1 score because Ki-67 stained NET images contain a lot of nuclei exhibiting weak staining and similar intensity values between nucleus and background, leading to large FP. Due to the relatively noisy probability map, both **DV-3** and **CNN-PC** provide higher R score but with a large loss in precision. On

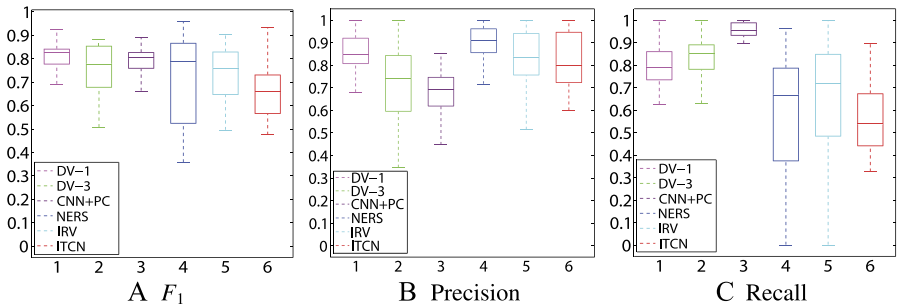


FIGURE 7.3

Boxplot of F_1 score, precision, and recall on the NET data set.

the other hand, our method, which encodes the geometric information of the nucleus centroids and patch centers, produces much better overall performance than others. The **DV-3** also gives reasonable results with faster computational time. The boxplot in Fig. 7.3 provides detailed comparisons in terms of precision, recall, and F_1 score.

The computation time for testing a 400×400 RGB image is 22, 5.5, and 31 s for our three deep voting implementations, **DV-1**, **DV-3**, and fast scanning, respectively. In our work, both **DV-1** and **DV-3** are implemented using GPU, and deep voting using fast scanning is implemented with MATLAB.

7.1.5 CONCLUSION

In this section, we proposed a novel deep voting model for accurate and robust nucleus localization. We extended the conventional CNN model to jointly learn the voting confidence and voting offset by introducing a **hybrid nonlinear activation function**. We formulated our problem as a minimization problem with a novel loss function. In addition, we also demonstrated that our method can be accelerated significantly using simple strategies without sacrificing the overall detection accuracy. Both qualitative and quantitative experimental results demonstrate the superior performance of the proposed deep voting algorithm compared with several state of the art procedures. We will carry out more experiments and test our method on other image modalities in the future work.

7.2 STRUCTURED REGRESSION FOR ROBUST CELL DETECTION USING CONVOLUTIONAL NEURAL NETWORK

7.2.1 INTRODUCTION

In microscopic image analysis, robust cell detection is a crucial prerequisite for biomedical image analysis tasks, such as cell segmentation and morphological mea-

surements. Unfortunately, the success of cell detection is hindered by the nature of microscopic images with noise coming from touching cells, background clutters, large variations in the shape and the size of cells, and the use of different image acquisition techniques.

To alleviate these problems, a non-overlapping extremal regions selection method is presented in [14] and achieves state-of-the-art performance on their data sets. However, this work heavily relies on a robust region detector and thus the application is limited. Recently, deep learning based methods, which exploit the deep architecture to learn the hierarchical discriminative features, have shown great developments and achieved significant success in biomedical image analysis [26,27]. Convolutional neural network (CNN) attracts particular attentions among those works because of its outstanding performance. Ciresan et al. adopt CNN for mitosis detection [34] in breast cancer histology images and membrane neuronal segmentation [35] in microscopy images. Typically, CNN is used as a pixel-wise classifier. In the training stage, local image patches are fed into the CNN with their labels determined by the membership of the central pixel. However, this type of widely used approach ignores the fact the labeled regions are coherent and often exhibit certain topological structures. Failing to take this topological information into consideration will lead to implausible class label transition problem [54].

In this section, we propose a novel CNN based structured regression model for cell detection. The contributions are summarized as two parts: (i) We modify the conventional CNN by replacing the last layer (classifier) with a structured regression layer to encode the topological information. (ii) Instead of working on the label space, regression on the proposed structured proximity space for patches is performed so that centers of image patches are explicitly forced to get higher value than their neighbors. The proximity map produced with our novel fusion scheme contains much more robust local maxima for cell centers. To the best of our knowledge, this is the first study to report the application of structured regression model using CNN for cell detection.

7.2.2 METHODOLOGY

We formulate the cell detection task as a structured learning problem. We replace the last (classifier) layer that is typically used in conventional CNN with a structured regression layer. The proposed model encodes the topological information in the training data. In the testing stage, instead of assigning hard class labels to pixels, our model generates a proximity patch which provides much more precise cues to locate cell centers. To obtain the final proximity map for an entire testing image, we propose to fuse all the generated proximity patches together.

CNN-Based Structured Regression. Let $\mathcal{X}$ denote the patch space, which consists of $d \times d \times c$ local image patches extracted from c-channel color images. An image patch $x \in \mathcal{X}$ centered at the location (u, v) of image I is represented by a quintuple $\{u, v, d, c, I\}$. We define $\mathcal{M}$ as the proximity mask corresponding to image I, and

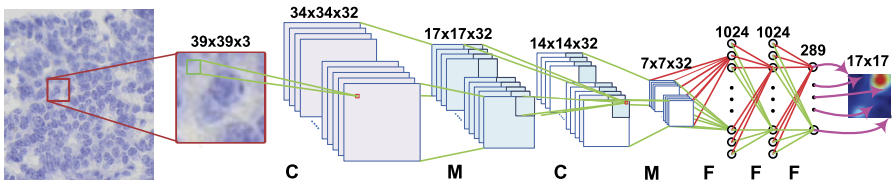


FIGURE 7.4

The CNN architecture used in the proposed structured regression model. C, M, and F represent the convolutional layer, max pooling layer, and fully connected layer, respectively. The purple arrows from the last layer illustrate the mapping between the final layer's outputs to the final proximity patch.

compute the value of the (ij)th entry in $\mathcal{M}$ as

$$\mathcal{M}_{ij} = \begin{cases} \frac{1}{1+\alpha D(i,j)}, & \text{if } D(i,j) \leq r, \\ 0, & \text{otherwise}, \end{cases} \tag{7.12}$$

where $D(i, j)$ represents the Euclidean distance from pixel (i, j) to the nearest human annotated cell center. r is a distance threshold and is set to be 5 pixels. α is the decay ratio and is set to be 0.8.

The $\mathcal{M}_{ij}$ can have values in the interval $\mathcal{V} = [0, 1]$. An image patch x has a corresponding proximity patch on the proximity mask (shown in Fig. 7.4). We define $s \in \mathcal{V}^{d' \times d'}$ as the corresponding proximity patch for patch x, where $d' \times d'$ denotes the proximity patch size. Note that d' is not necessarily equal to d. We further define the proximity patch s of patch x as $s = \{u, v, d', \mathcal{M}\}$. It can be viewed as the *structured label* of patch $x = \{u, v, d, c, I\}$.

We define the training data as $\{(\boldsymbol{x}^i, \boldsymbol{y}^i) \in (\mathcal{X}, \mathcal{Y})\}_{i=1}^{\mathcal{N}}$, whose elements are pairs of inputs and outputs: $\boldsymbol{x}^i \in \mathcal{X}$, $\boldsymbol{y}^i = \Gamma(\boldsymbol{s}^i)$, $\mathcal{N}$ is the number of training samples, and $\Gamma : \mathcal{V}^{d' \times d'} \rightarrow \mathcal{Y}$ is a mapping function to represent the vectorization operation in column-wise order for a proximity patch. $\mathcal{Y} \subset \mathcal{V}^{p \times 1}$ represents the output space of the structured regression model, where $p = d' \times d'$ denotes the number of units in the last layer. Define functions $\{f_l\}_{l=1}^{L}$ and $\{\boldsymbol{\theta}_l\}_{l=1}^{L}$ as the operations and parameters corresponding to each of the L layers, the training process of the structured regression model can be formulated as learning a mapping function ψ composed with $\{f_1, \ldots, f_L\}$, which will map the image space $\mathcal{X}$ to the output space $\mathcal{Y}$.

Given a set of training data $\{(\boldsymbol{x}^i, \boldsymbol{y}^i) \in (\mathcal{X}, \mathcal{Y})\}_{i=1}^{\mathcal{N}}$, $\{\boldsymbol{\theta}_l\}_{l=1}^{L}$ will be learned by solving the following optimization problem

$$\arg\min_{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L} \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \mathcal{L}(\psi(\boldsymbol{x}^i; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L), \boldsymbol{y}^i), \tag{7.13}$$

where $\mathcal{L}$ is the loss function which will be defined shortly.

Eq. (7.13) can be solved using the classical backpropagation algorithm. In order to back propagate the gradients from the last layer (structured regression layer) to the lower layers, we need to differentiate the loss function defined on one training sample with respect to the inputs to the last layer. Let $\boldsymbol{a}^i$ and $\boldsymbol{o}^i$ represent the inputs and the outputs of the last layer. For one training example $(\boldsymbol{x}^i, \boldsymbol{y}^i)$, we can have $\boldsymbol{o}^i = \psi(\boldsymbol{x}^i; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L)$. Denote by y^i_j, a^i_j, and o^i_j the jth element of $\boldsymbol{y}^i$, $\boldsymbol{a}^i$, and $\boldsymbol{o}^i$, respectively. The loss function $\mathcal{L}$ for $(\boldsymbol{x}^i, \boldsymbol{y}^i)$ is given by

$$\begin{aligned}\mathcal{L}(\psi(\boldsymbol{x}^i; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L), \boldsymbol{y}^i) = \mathcal{L}(\boldsymbol{o}^i, \boldsymbol{y}^i) &= \frac{1}{2}\sum_{j=1}^{p}(y^i_j + \lambda)(y^i_j - o^i_j)^2 \\ &= \frac{1}{2}\left\|(Diag(\boldsymbol{y}^i) + \lambda\mathbf{I})^{1/2}(\boldsymbol{y}^i - \boldsymbol{o}^i)\right\|_2^2, \end{aligned} \tag{7.14}$$

where $\mathbf{I}$ is an identity matrix of size $p \times p$, and $Diag(\boldsymbol{y}^i)$ is a diagonal matrix with the jth diagonal element equal to y^i_j. Since the nonzero region in the proximity patch is relatively small, our model might return a trivial solution. To avoid this problem, we adopt a weighting strategy [55] to give the loss coming from the network's outputs corresponding to the nonzero area in the proximity patch more weights. A small λ indicates strong penalization of errors coming from the outputs with low proximity values in the training data. This method is different from [55] which applies a bounding box mask regression approach on the entire image.

We choose the sigmoid activation function in the last layer, i.e., $o^i_j = sigm(a^i_j)$. The partial derivative of (7.14) with respect to the input of the jth unit in the last layer is given by

$$\frac{\partial\mathcal{L}(\boldsymbol{o}^i, \boldsymbol{y}^i)}{\partial a^i_j} = \frac{\partial\mathcal{L}(\boldsymbol{o}^i, \boldsymbol{y}^i)}{\partial o^i_j}\frac{\partial o^i_j}{\partial a^i_j} = (y^i_j + \lambda)(o^i_j - y^i_j)a^i_j(1 - a^i_j). \tag{7.15}$$

Based on (7.15), we can evaluate the gradients of (7.13) with respect to the model's parameters in the same way as [52]. The optimization procedure is based on mini-batch stochastic gradient descent.

CNN Architecture. The proposed structured regression model contains several convolutional (C), max-pooling (M), and fully-connected layers (F). Fig. 7.4 illustrates one of the architectures and mapped proximity patches in the proposed model. The detailed model configuration is: Input($49 \times 49 \times 3$) − C($44 \times 44 \times 32$) − M($22 \times 22 \times 32$) − C($20 \times 20 \times 32$) − M($10 \times 10 \times 32$) − C($8 \times 8 \times 32$) − F(1024) − F(1024) − F(289). The activation function of last F (regression) layer is chosen to be the sigmoid, and ReLu is used for all the other F and C layers. Here, the sizes of C and M layers are defined as *width* × *height* × *depth*, where *width* × *height* determines the dimensionality of each feature map and *depth* represents the number of feature maps. The filter size is chosen to be 6×6 for the first convolutional layer and 3×3 for the remaining two. The max pooling layer uses a window of size 2×2 with a stride of 2.

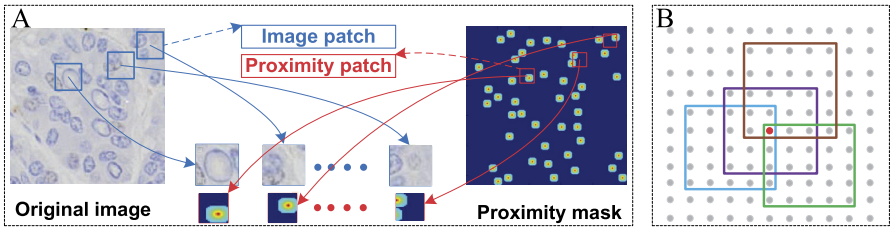


FIGURE 7.5

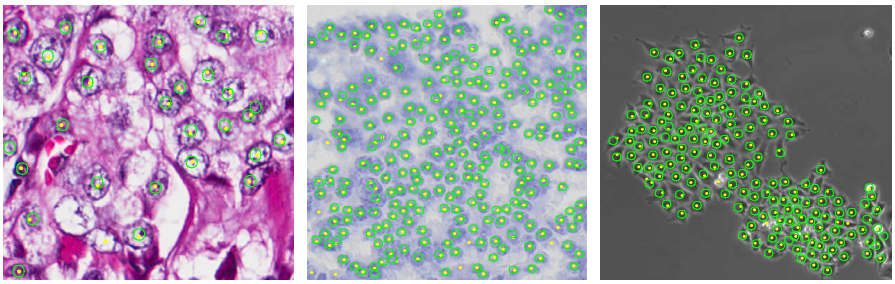
(A) The training data generation process. Each original image has a proximity mask of the same size and each local image patch has a proximity patch used as the structured *label*. (B) The fusion process. Each pixel receives predictions from it's neighborhoods. For example, the red dot collects all the predictions from its 25 neighboring pixels and an average value will be assigned as the final result. In this figure, we only display 4 out of 25 proximity patches.

Structured Prediction Fusion and Cell Localization. Given a testing image patch $x = (u, v, d, c, I)$, it is easy to get the corresponding proximity mask $s = \Gamma^{-1}(y)$, where $y \in \mathcal{Y}$ represents the model's output corresponding to x. In the fusion process, s will cast a proximity value for every pixel that lies in the $d' \times d'$ neighborhood area of (u, v), for example, pixel $(u + i, v + j)$ in image I will get a prediction s_{ij} from pixel (u, v). In other words, as we show in Fig. 7.5B, each pixel actually receives $d' \times d'$ predictions from its neighboring pixels. To get the fused proximity map, we average all the predictions for each pixel from its neighbors and calculate it's final proximity prediction. After this step, the cell localization can be easily obtained by finding the local maximum positions in the average proximity map.

Speed Up. Traditional sliding window method is time consuming. However, we have implemented two strategies to speed up. The first one comes from the property that our model generates a $d' \times d'$ proximity patch for each testing patch. This makes it feasible to skip a lot of pixels and only test the image patches at a certain stride ss $(1 \leq ss \leq d')$ without significantly sacrificing the accuracy. The second strategy, called *fast scanning* [56], is based on the fact that there exists a lot of redundant convolution operations among adjacent patches when computing the sliding-windows.

7.2.3 EXPERIMENTAL RESULTS

Data Set and Implementation Details. This model is implemented in C++ and CUDA based on the fast CNN kernels [30], and *fast scanning* [56] is implemented in MATLAB. The proposed algorithm is trained and tested on a PC with an Intel Xeon E5 CPU and an NVIDIA Tesla k40C GPU. The learning rate is set to be 0.0005 and a dropout rate of 0.2 is used for the fully connected layers. The λ is set to be 0.3 in (7.14).



A H&E breast cancer B Ki-67 stained NET C Phase-contrast HeLa

FIGURE 7.6

Cell detection results on three sample images from the three data sets. Yellow dots represent the detected cell centers. The ground truth annotations are represented by green circles for better illustrations.

Three data sets are used to evaluate the proposed method: First, The Cancer Genome Atlas (TCGA) dataset, from which we cropped and annotated 32 H&E-stained microscopy images of breast cancer cells, has a magnification of 40×. The detection task in this data set is challenging due to highly inhomogeneous background noise, a large variability of the size of cells, and background similarities. The second dataset is obtained from [14] and contains 22 phase contrast images of HeLa cervical cancer cell. These images exhibit large variations in sizes and shapes. The third dataset contains 60 Ki67-stained neuroendocrine tumor (NET), the magnification is 40×. Many touching cells, weak staining, and fuzzy cell boundaries are present in this data. Due to the limited resource of computation, we do not use cross-validation, instead all these datasets are equally split to construct training and testing sets.

Model Evaluation. Fig. 7.6 shows the qualitative detection results on three datasets. For quantitative analysis, we define the ground-truth areas as circular regions within 5 pixels of every annotated cell center. A detected cell centroid is considered to be a true positive (TP) only if it lies within the ground-truth areas; otherwise, it is considered as a false positive (FP). Each TP is matched with the nearest ground-truth annotated cell center. The ground-truth cell centers that are not matched by any detected results are considered to be false negatives (FN). Based on the above definitions, we can compute the precision (P), recall (R), and F_1-score as $P = \frac{TP}{TP+FP}$, $R = \frac{TP}{TP+FN}$, and $F_1 = \frac{2PR}{P+R}$, respectively.

We evaluated four variations of the proposed methods: (1, 2) *Structured Regression + testing with a stride ss* (SR-ss), where *ss* is chosen to be 1 for (1) and 5 for (2); (3) *CNN based Pixel-Wise Classification* (PWC), which shares a similar architecture with the proposed method except that it utilizes the softmax classifier in the last layer; and (4) *CNN based Pixel-Wise Regression* (PWR), which is similar to SR-1 but only predicts the proximity value for the central pixel of each patch.

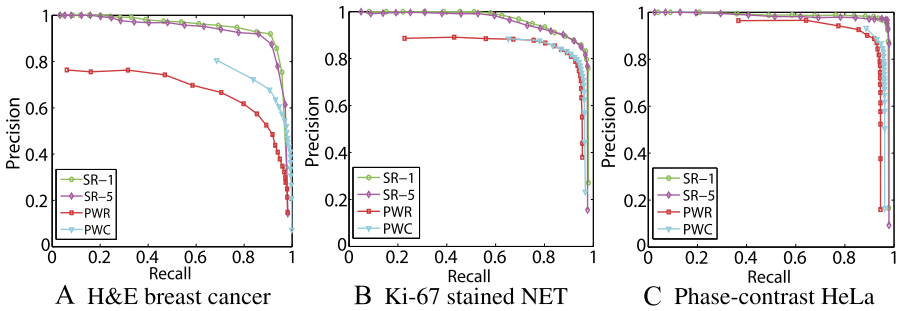


FIGURE 7.7

Precision–recall curves of the four variations of the proposed algorithm on three data sets. SR-5 achieves almost the same results as SR-1. The proposed SR-1 significantly outperforms the other two pixel-wise methods using CNN.

Fig. 7.7 shows the precision–recall curves of the four variations of the proposed method on each data set. These curves are generated by changing the threshold ζ on the final proximity maps before finding the local maximum. We can see that SR-5 achieves almost the same performance as SR-1, and both PWC and PWR don't work as well as the proposed structured regression model, especially for the H& E breast cancer data set that exhibits high background similarity and large variations in cell size. This demonstrates that introducing structured regression increases the overall performance. The computational costs for SR-1, SR-5, and *fast scanning* are 14.5, 5, and 19 s for testing a 400 × 400 RGB image. In the training stage, our model takes about 5 h to converge.

Comparison with Other Works. We also compare our structured regression model (SR) with four state-of-the-art methods, including Non-overlapping Extremal Regions Selection (NERS) [14], Iterative Radial Voting (IRV) [5], Laplacian-of-Gaussian filtering (LoG) [2], and Image-based Tool for Counting Nuclei (ITCN) [53]. In addition to Precision, Recall, and F_1 score, we also compute the mean and standard deviation of two terms: (i) the absolute difference $\mathbf{E_n}$ between the number of true positive and the ground-truth annotations, and (ii) the Euclidean distance $\mathbf{E_d}$ between the true positive and the corresponding annotations. The quantitative experiment results are reported in Table 7.2. It is obvious that our method has better performance than the above listed alternate models on all the datasets considered, especially in terms of the F_1-score. This method also exhibits strong reliability with the lowest mean and standard deviations in $\mathbf{E_n}$ and $\mathbf{E_d}$ on NET and phase contrast data sets.

7.2.4 CONCLUSION

In this section, we proposed a structured regression model for robust cell detection. The proposed method differs from the conventional CNN classifiers by introducing

Table 7.2 The comparative cell detection results on three data sets. μ_d, σ_d represent the mean and standard deviation of $\mathbf{E_d}$, and μ_n, σ_n represent the mean and standard deviation of $\mathbf{E_n}$

Data set	Methods	P	R	F_1	$\mu_d \pm \sigma_d$	$\mu_n \pm \sigma_n$
H&E breast cancer	SR-1	**0.919**	0.909	**0.913**	**3.151 ± 2.049**	4.8750 ± 2.553
	NERS [14]	–	–	–	–	–
	IRV [5]	0.488	0.827	0.591	5.817 ± 3.509	9.625 ± 4.47
	LoG [2]	0.264	**0.95**	0.398	7.288 ± 3.428	**2.75 ± 2.236**
	ITCN [53]	0.519	0.528	0.505	7.569 ± 4.277	26.188 ± 8.256
NET	SR-1	0.864	**0.958**	**0.906**	**1.885 ± 1.275**	**8.033 ± 10.956**
	NERS [14]	**0.927**	0.648	0.748	2.689 ± 2.329	32.367 ± 49.697
	IRV [5]	0.872	0.704	0.759	2.108 ± 3.071	15.4 ± 14.483
	LoG [2]	0.83	0.866	0.842	3.165 ± 2.029	11.533 ± 21.782
	ITCN [53]	0.797	0.649	0.701	3.643 ± 2.084	24.433 ± 40.82
Phase contrast	SR-1	**0.942**	**0.972**	**0.957**	**2.069 ± 1.222**	**3.455 ± 4.547**
	NERS [14]	0.934	0.901	0.916	2.174 ± 1.299	11.273 ± 11.706
	IRV [5]	0.753	0.438	0.541	2.705 ± 1.416	58.818 ± 40.865
	LoG [2]	0.615	0.689	0.649	3.257 ± 1.436	29.818 ± 16.497
	ITCN [53]	0.625	0.277	0.371	2.565 ± 1.428	73.727 ± 41.867

a new structured regressor to capture the topological information of the data. Spatial coherence is maintained across the image at the same time. In addition, our proposed algorithm can be implemented with several fast implementation options. We experimentally demonstrated the superior performance of the proposed method compared with several state-of-the-art methods. We also showed that the proposed method can handle different types of microscopy images with outstanding performance. In future work, we will validate the generality of the proposed model on other image modalities.

ACKNOWLEDGEMENTS

This research is supported by NIH grant R01 AR065479-02.

REFERENCES

1. P. Quelhas, M. Marcuzzo, A.M. Mendonça, A. Campilho, Cell nuclei and cytoplasm joint segmentation using the sliding band filter, IEEE Trans. Med. Imaging 29 (8) (2010) 1463–1473.
2. Y. Al-Kofahi, W. Lassoued, W. Lee, B. Roysam, Improved automatic detection and segmentation of cell nuclei in histopathology images, IEEE Trans. Biomed. Eng. 57 (4) (2010) 841–852.

3. E. Bernardis, S.X. Yu, Finding dots: segmentation as popping out regions from boundaries, in: CVPR, 2010, pp. 199–206.
4. C. Zhang, J. Yarkony, F.A. Hamprecht, Cell detection and segmentation using correlation clustering, in: MICCAI, vol. 8673, 2014, pp. 9–16.
5. B. Parvin, Q. Yang, J. Han, H. Chang, B. Rydberg, M.H. Barcellos-Hoff, Iterative voting for inference of structural saliency and characterization of subcellular events, IEEE Trans. Image Process. 16 (2007) 615–623.
6. X. Qi, F. Xing, D.J. Foran, L. Yang, Robust segmentation of overlapping cells in histopathology specimens using parallel seed detection and repulsive level set, IEEE Trans. Biomed. Eng. 59 (3) (2012) 754–765.
7. F. Xing, H. Su, L. Yang, An integrated framework for automatic Ki-67 scoring in pancreatic neuroendocrine tumor, in: MICCAI, 2013, pp. 436–443.
8. B. Parvin, Qing Yang, Ju Han, Hang Chang, B. Rydberg, M.H. Barcellos-Hoff, Iterative voting for inference of structural saliency and characterization of subcellular events, IEEE Trans. Image Process. 16 (3) (2007) 615–623.
9. Adel Hafiane, Filiz Bunyak, Kannappan Palaniappan, Fuzzy clustering and active contours for histopathology image segmentation and nuclei detection, in: Advanced Concepts for Intelligent Vision Systems, vol. 5259, Springer, Berlin, Heidelberg, ISBN 978-3-540-88457-6, 2008, pp. 903–914.
10. Gang Li, Tianming Liu, Jingxin Nie, Lei Guo, Jarema Malicki, Andrew Mara, Scott A. Holley, Weiming Xia, Stephen T.C. Wong, Detection of blob objects in microscopic zebrafish images based on gradient vector diffusion, Cytometry Part A 71 (10) (2007) 835–845.
11. H. Cinar Akakin, H. Kong, C. Elkins, J. Hemminger, B. Miller, J. Ming, E. Plocharczyk, R. Roth, M. Weinberg, R. Ziegler, G. Lozanski, M.N. Gurcan, Automated detection of cells from immunohistochemically-stained tissues: application to Ki-67 nuclei staining, in: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 8315, 2012, p. 3.
12. Lin Yang, Oncel Tuzel, Peter Meer, David J. Foran, Automatic image analysis of histopathology specimens using concave vertex graph, in: Medical Image Computing and Computer-Assisted Intervention – MICCAI 2008, in: Lect. Notes Comput. Sci., vol. 5241, ISBN 978-3-540-85987-1, 2008, pp. 833–841.
13. Hui Kong, M. Gurcan, K. Belkacem-Boussaid, Partitioning histopathological images: an integrated framework for supervised color-texture segmentation and cell splitting, IEEE Trans. Med. Imaging 30 (9) (2011) 1661–1677.
14. C. Arteta, V. Lempitsky, J.A. Noble, A. Zisserman, Learning to detect cells using non-overlapping extremal regions, in: MICCAI, vol. 7510, 2012, pp. 348–356.
15. Luca Bertelli, Tianli Yu, Diem Vu, Burak Gokturk, Kernelized structural SVM learning for supervised object segmentation, in: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2011, pp. 2153–2160.
16. B. Leibe, A. Leonardis, B. Schiele, Combined object categorization and segmentation with an implicit shape model, in: Proceedings of the Workshop on Statistical Learning in Computer Vision, Prague, Czech Republic, 2004.
17. Juergen Gall, Victor Lempitsky, Class-Specific Hough Forests for Object Detection, 2009, pp. 255–258.
18. Xiangfei Kong, Kuan Li, Jingjing Cao, Qingxiong Yang, Liu Wenyin, Hep-2 cell pattern classification with discriminative dictionary learning, Pattern Recognit. 47 (7) (2014) 2379–2388.

19. Hai Su, Fuyong Xing, Jonah D. Lee, Charlotte A. Peterson, Lin Yang, Automatic myonuclear detection in isolated single muscle fibers using robust ellipse fitting and sparse optimization, IEEE/ACM Trans. Comput. Biol. Bioinform. 11 (4) (2014) 714–726.
20. Kunihiko Fukushima, Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biol. Cybern. 36 (4) (1980) 193–202.
21. Sven Behnke, Hierarchical Neural Networks for Image Interpretation, vol. 2766, Springer Science & Business Media, 2003.
22. Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
23. Patrice Y. Simard, Dave Steinkraus, John C. Platt, Best practices for convolutional neural networks applied to visual document analysis, in: 2013 12th International Conference on Document Analysis and Recognition, vol. 2, IEEE Computer Society, 2003, p. 958.
24. Clement Farabet, Camille Couprie, Laurent Najman, Yann LeCun, Learning hierarchical features for scene labeling, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1915–1929.
25. A. Cruz-Roa, J. Arevalo-Ovalle, A. Madabhushi, F. Gonzalez-Osorio, A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection, in: MICCAI, vol. 8150, 2013, pp. 403–410.
26. S. Liao, Y. Gao, A. Oto, D. Shen, Representation learning: a unified deep learning framework for automatic prostate MR segmentation, in: MICCAI, vol. 8150, 2013, pp. 254–261.
27. R. Li, W. Zhang, H. Suk, L. Wang, J. Li, D. Shen, S. Ji, Deep learning based imaging data completion for improved brain disease diagnosis, in: MICCAI, 2014, pp. 305–312.
28. Y.A. LeCun, L. Bottou, G.B. Orr, K. Müller, Efficient BackProp, in: Neural Networks: Tricks of the Trade, vol. 1524, 1998, pp. 9–50.
29. Clément Farabet, Camille Couprie, Laurent Najman, Yann LeCun, Learning hierarchical features for scene labeling, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1915–1929.
30. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet classification with deep convolutional neural networks, in: NIPS, 2012, pp. 1106–1114.
31. Alexander Toshev, Christian Szegedy, DeepPose: human pose estimation via deep neural networks, in: IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1653–1660.
32. Christian Szegedy, Alexander Toshev, Dumitru Erhan, Deep neural networks for object detection, in: NIPS, 2013, pp. 2553–2561.
33. Gernot Riegler, David Ferstl, Matthias Rüther, Horst Bischof, Hough networks for head pose estimation and facial feature localization, in: Proceedings of the British Machine Vision Conference, 2014.
34. D. Ciresan, A. Giusti, L.M. Gambardella, J. Schmidhuber, Mitosis detection in breast cancer histology images with deep neural networks, in: MICCAI, vol. 8150, 2013, pp. 411–418.
35. D.C. Ciresan, A. Giusti, L.M. Gambardella, J. Schmidhuber, Deep neural networks segment neuronal membranes in electron microscopy images, in: NIPS, 2012, pp. 2852–2860.
36. W. Zhang, R. Li, H. Deng, L. Wang, W. Lin, S. Ji, D. Shen, Deep convolutional neural networks for multi-modality isointense infant brain image segmentation, in: Neuroimage, 2014.
37. A. Payan, G. Montana, Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks, 2015.

38. Yuanpu Xie, Fuyong Xing, Xiangfei Kong, Hai Su, Lin Yang, Beyond classification: structured regression for robust cell detection using convolutional neural network, in: MICCAI, 2015, pp. 358–365.
39. Yuanpu Xie, Xiangfei Kong, Fuyong Xing, Fujun Liu, Hai Su, Lin Yang, Deep voting: a robust approach toward nucleus localization in microscopy images, in: MICCAI, 2015, pp. 374–382.
40. Fuyong Xing, Yuanpu Xie, Lin Yang, An automatic learning-based framework for robust nucleus segmentation, IEEE Trans. Med. Imaging 35 (2) (2016) 550–566.
41. G. Wu, M. Kim, Q. Wang, Y. Gao, S. Liao, D. Shen, Unsupervised deep feature learning for deformable registration of MR brain images, in: MICCAI, vol. 8150, 2013, pp. 649–656.
42. A. Prasoon, K. Petersen, C. Igel, F. Lauze, E. Dam, M. Nielsen, Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network, in: Int. Conf. Med. Image Comput. Comput. Assist. Intervent. (MICCAI), vol. 8150, 2013, pp. 246–253.
43. H.R. Roth, L. Lu, A. Seff, K.M. Cherry, J. Hoffman, S. Wang, J. Liu, E. Turkbey, R.M. Summers, A new 2.5 D representation for lymph node detection using random sets of deep convolutional neural network observations, in: MICCAI, 2014, pp. 520–527.
44. A.A. Cruz-Roa, J.E.A. Ovalle, A. Madabhushi, F.A.G. Osorio, A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection, in: MICCAI, 2013, pp. 403–410.
45. Alex Graves, Santiago Fernández, Jürgen Schmidhuber, Multi-dimensional recurrent neural networks, in: ICANN, 2007, pp. 549–558.
46. Wonmin Byeon, Thomas M. Breuel, Federico Raue, Marcus Liwicki, Scene labeling with LSTM recurrent neural networks, in: CVPR, 2015, pp. 3547–3555.
47. Jonathan Long, Evan Shelhamer, Trevor Darrell, Fully convolutional networks for semantic segmentation, in: CVPR, 2015, pp. 3431–3440.
48. Yuanpu Xie, Zizhao Zhang, Manish Sapkota, Lin Yang, Spatial clockwork recurrent neural network for muscle perimysium segmentation, in: MICCAI, 2016.
49. Olaf Ronneberger, Philipp Fischer, Thomas Brox, U-Net: convolutional networks for biomedical image segmentation, in: MICCAI, 2015, pp. 234–241.
50. Angela Yao, Juergen Gall, Luc Van Gool, A hough transform-based voting framework for action recognition, in: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2010, pp. 2061–2068.
51. E. Parzen, On the estimation of a probability density function and the mode, Ann. Math. Stat. (1962) 1065–1076.
52. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, 86 (1998) 2278–2324.
53. J. Byun, M.R. Verardo, B. Sumengen, G.P. Lewis, B.S. Manjunath, S.K. Fisher, Automated tool for the detection of cell nuclei in digital microscopic images: application to retinal images, Mol. Vis. 12 (2006) 949–960.
54. P. Kontschieder, S.R. Bulò, H. Bischof, M. Pelillo, Structured class-labels in random forests for semantic image labelling, in: ICCV, 2012, pp. 2190–2197.
55. C. Szegedy, A. Toshev, D. Erhan, Deep neural networks for object detection, in: NIPS, 2013, pp. 2553–2561.
56. A. Giusti, D.C. Ciresan, J. Masci, L.M. Gambardella, J. Schmidhuber, Fast image scanning with deep max-pooling convolutional neural networks, in: ICIP, 2013, pp. 4034–4038.