Published in final edited form as:

Med Image Comput Comput Assist Interv. 2015 October ; 9351: 374–382. doi:

10.1007/978-3-319-24574-4_45

Deep Voting: A Robust Approach Toward Nucleus Localization in Microscopy Images

Yuanpu Xie¹, Xiangfei Kong¹, Fuyong Xing², Fujun Liu², Hai Su¹, and Lin Yang¹

¹J. Crayton Pruitt Family Department of Biomedical Engineering, University of Florida, FL 32611, USA

²Department of Electrical and Computer Engineering, University of Florida, FL 32611, USA

Abstract

Robust and accurate nuclei localization in microscopy image can provide crucial clues for accurate computer-aid diagnosis. In this paper, we propose a convolutional neural network (CNN) based hough voting method to localize nucleus centroids with heavy cluttering and morphologic variations in microscopy images. Our method, which we name as deep voting, mainly consists of two steps. (1) Given an input image, our method assigns each local patch several pairs of voting *offset* vectors which indicate the positions it votes to, and the corresponding voting *confidence* (used to weight each votes), our model can be viewed as an implicit hough-voting codebook. (2) We collect the weighted votes from all the testing patches and compute the final voting density map in a way similar to Parzen-window estimation. The final nucleus positions are identified by searching the local maxima of the density map. Our method only requires a few annotation efforts (just one click near the nucleus center). Experiment results on Neuroendocrine Tumor (NET) microscopy images proves the proposed method to be state-of-the-art.

1 Introduction

Nuclei localization in microscopy image is the foundation of many subsequent biomedical image analysis tasks (e.g., cell segmentation, counting and tracking, etc). However, robust and accurate nucleus localization, especially for microscopy images that exhibit dense nucleus clutters and large variations in both nucleus sizes and shapes, has been a challenging task for a long time. In the past few years, a large number of methods have been proposed, including kernel based radial voting [10], spatial filtering [1] and graph partition based methods [3]. 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 approaches.

Supervised learning based methods have also attracted a lot of interests due to their promising performance. For example, a general and efficient maximally stable extremal region (MSER) section method is presented in [2]. However this method requires a robust MSER detector and thus the usage is limited. Codebook based hough transformation has also been widely studied and shown to be able to produce promising performance. In [6], a class specific random forest method is applied to learn a discriminative codebook. However, it only associates one voting *offset* vector to a 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.

Recently, deep learning has been revived and achieved outstanding performance in both natural and biomedical image analysis tasks [12,11,5]. In this work, we extend the traditional CNN model to learn the voting *offset* vectors and voting *confidence* jointly. Different from [6,11], our method assigns more than one pairs of prediction for each image patch, which together with our specific loss function render our method more robust and capable of handling touching cases. As shown in Fig.1, our deep voting model can be viewed as an implicit hough-voting codebook, via which, each testing patch can vote towards several directions with specific voting *confidence*. We then collect all the weighted votes from all the image patches in an additive way, and estimate the voting density map in a way similar to Parzen-window estimation. Comparative experimental results using several datasets show superiority of the proposed deep voting method.

2 Methodology

2.1 Learning The Deep Voting Model

Preprocessing—In our method, each patch votes to several possible nucleus positions (the coordinates for the nucleus center specified by voting *offsets*) with voting *confidence*. We propose to learn an implicit codebook for these voting *offsets* and corresponding voting *confidence*. During the training stage, we denote the input image patch space as $\mathcal{P}$, composed of a set of image patches with size $d \times d$, and the target space as $\mathcal{T}$, 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 patch, we first get its k nearest nucleus positions from a group of human annotated nucleus centers (ground-truth) on training images. Then, corresponding *target information* is computed. Let $\{\mathcal{D}^i = (\mathcal{P}^i, \mathcal{T}^i)\}$ represent the training set where $\mathcal{P}^i \in \mathcal{P}$ is the i -th input image patch and $\mathcal{T}^i \in \mathcal{T}$ is the corresponding *target information*. $\mathcal{T}^i$ is further defined as $\{\alpha_j^i, \mathbf{v}_j^i\}_{j=1}^k = 1$, where k denotes the number of total votes from each image patch and $\mathbf{v}_j^i$ is the 2D *offset* vector equals to the displacement from the j -th nearest ground-truth nucleus aligned to the center of patch $\mathcal{P}^i$. α_j^i represents the voting *confidence* corresponding to $\mathbf{v}_j^i$. We define it based on the length of the voting *offset* $|\mathbf{v}_j^i|$:

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

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 α_j^i is given the flexibility to associate

with many properties of the training patches such as foreground area ratio, class membership, distance to the voting position, etc.

Hybrid Non-linear Transformation—We categorize the units of the output layer in our CNN as *voting units* and *confidence units* as shown in Fig.1. Thus, voting *offset* vectors and voting *confidence* are jointly learned. *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 each vote. Existing activation functions (sigmoid, linear, or ReLu) associated to the output layer do not satisfy our needs. We therefore introduce a **hybrid non-linear transformation** Hy as our new activation function. Units of the output layer are treated differently based on their category:

$$Hy(x) = \begin{cases} \text{sigm}(x), & \text{if } x \text{ is the input to the } \textit{confidence units}, \\ x, & \text{otherwise,} \end{cases} \quad (2)$$

where sigm denotes the sigmoid function, Hy takes a vector as its input and computed the value in element-wise. For example, for a vector $\mathbf{a} = \{a_i\}_{i=1}^m$, $Hy(\mathbf{a})$ equals to $\{Hy(a_i)\}_{i=1}^m$.

Inference In Deep Voting Model—Denote L as the number of layers, let functions $f_1, \dots, f_L$ represent each layer's computations (e.g., linear transformations and convolution, etc.), and denote $\theta_1, \dots, \theta_L$ as the parameters of those layers, our model can be viewed as a mapping function: ψ , which maps input image patches to *target information*. Denote $\circ$ as the composition of functions, ψ can be further defined as $f_L \circ f_{L-1}, \dots, \circ f_1$ with parameters $\Theta = \{\theta_1, \dots, \theta_L\}$.

Given a set of training data $\{\mathcal{P}^i = (\mathcal{P}^i, \mathcal{T}^i)\}_{i=1}^N$, where N is the number of training samples. We evaluate the proposed model's parameters by solving

$$\arg \min_{\Theta} \frac{1}{N} \sum_{i=1}^N l(\psi(\mathcal{P}^i; \Theta), T^i), \quad (3)$$

where l is a proper loss function defined in the following sections. Let

$\psi(\mathcal{P}^i; \Theta) = \{w_j^i, d_j^i\}_{j=1}^k$ denote the model's output corresponding to the input $\mathcal{P}^i$, where w_j^i and d_j^i are predicted voting *confidence* and *offsets*, respectively. Recall that

$\mathcal{T}^i = \{\alpha_j^i, \mathbf{v}_j^i\}_{j=1}^k$ the *target information* of $\mathcal{P}^i$ and k represents the number of votes from each training image patch. The loss function defined on one training example $(\mathcal{P}^i, \mathcal{T}^i)$ is given by

$$l(\psi(\mathcal{P}^i; \Theta), \mathcal{T}^i) = \sum_{j=1}^k \frac{1}{2} \left(\alpha_j^i w_j^i \| \mathbf{d}_j^i - \mathbf{v}_j^i \|^2 + \lambda (w_j^i - \alpha_j^i)^2 \right), \quad (4)$$

where λ is a regularization factor. There are two benefits of the proposed loss function: (1) The first term punishes uninformative votes that have either low predicted voting *confidence* or low voting *confidence* in the *target information*. (2) The second term acts as a regularization term to prevent the network producing trivial solutions (setting all voting *confidence* to zero).

The optimization problem defined in (3) is highly non-convex. We use the back propagation algorithm [9] to solve it. In order to calculate the gradient of (3) with respect to the model's parameters Θ , 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.1, the outputs of the proposed model are organized as k pairs of $\{\text{confidence units, offset units}\}$.

Let $\{Iw_j^i, Iv_j^i\}_{j=1}^k$ represent the inputs to the units in the last layer for one training sample $(\mathcal{P}^i, \mathcal{T}^i)$, we can easily get $w_j^i = Hy(Iw_j^i)$ and $v_j^i = Hy(Iv_j^i)$. The partial differentiations of (4) with respect to Iw_j^i and Iv_j^i are then given as

$$\begin{aligned} \frac{\partial l(\psi(\mathcal{P}^i; \Theta), \mathcal{T}^i)}{\partial Iw_j^i} &= \frac{\partial l(\psi(\mathcal{P}^i; \Theta), \mathcal{T}^i)}{\partial w_j^i} \frac{\partial w_j^i}{\partial Iw_j^i} \\ &= \left(\frac{1}{2} \alpha_j^i \| \mathbf{d}_j^i - \mathbf{v}_j^i \|^2 + \lambda (w_j^i - \alpha_j^i) \right) Iw_j^i (1 - Iw_j^i), \\ \frac{\partial l(\psi(\mathcal{P}^i; \Theta), \mathcal{T}^i)}{\partial Id_j^i} &= \frac{\partial \mathbf{d}_j^i}{\partial Id_j^i} \frac{\partial l(\psi(\mathcal{P}^i; \Theta), \mathcal{T}^i)}{\partial \mathbf{d}_j^i} = \alpha_j^i w_j^i (\mathbf{d}_j^i - \mathbf{v}_j^i), \end{aligned} \quad (5)$$

where Id_j^i , $\mathbf{d}_j^i$ and $\frac{\partial l(\psi(\mathcal{P}^i; \Theta), \mathcal{T}^i)}{\partial Id_j^i}$ are 2D vectors. After getting the above partial derivatives, the following inferences used to obtain the gradients of (3) with respect to the model's parameters are exactly the same to the classical back propagation algorithm used in conventional CNN [9].

2.2 Weighted Voting Density Estimation

Given a testing image, denote $\mathcal{P}(\mathbf{y})$ as a testing patch centered at position $\mathbf{y}$ on the testing image, and let $\{w_j(\mathbf{y}), \mathbf{d}_j(\mathbf{y})\}_{j=1}^k$ represent our model's output for $\mathcal{P}(\mathbf{y})$, where $w_j(\mathbf{y})$ and $\mathbf{d}_j(\mathbf{y})$ represents the j -th voting *confidence* and voting *offset* vector for patch $\mathcal{P}(\mathbf{y})$, respectively. Furthermore, denote $V(\mathbf{x} | \mathcal{P}(\mathbf{y}))$ as the accumulated voting score of position $\mathbf{x}$ coming from patch $\mathcal{P}(\mathbf{y})$. Similar to the kernel density estimation using Parzen-window, we write $V(\mathbf{x} | \mathcal{P}(\mathbf{y}))$ as

$$V(x|\mathcal{P}(y)) = \sum_{j=1}^k w_j(y) \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|(\mathbf{y} - \mathbf{d}_j(\mathbf{y})) - \mathbf{x}\|^2}{2\sigma^2}\right). \quad (6)$$

In this paper, we restrict the votes to a location $\mathbf{x}$ to be calculated from a limited area $B(\mathbf{x})$, which can be a circle or a bounding box centered at position $\mathbf{x}$. To compute the final weighted voting density map $V(\mathbf{x})$, we accumulate the weighted votes:

$$V(\mathbf{x}) = \sum_{\mathbf{y} \in B(\mathbf{x})} V(\mathbf{x}|\mathcal{P}(\mathbf{y})). \quad (7)$$

After obtaining the voting density map $V(\mathbf{x})$, 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 maxima in V .

2.3 Efficient Nuclei Localization

Traditional localization methods with sliding window are computationally expensive. However, there are two methods to boost the testing speed of the proposed algorithm. The first strategy, which is called fast scanning [7], achieves three orders of magnitude speedup compared to standard sliding window. The basic idea is to remove the redundant convolution operations appeared in adjacent testing patches by doing the convolutional operations on the entire image. The second strategy is straight-forward: instead of performing the testing on the entire image for every sliding window, we can perform the computation in an s stride manner. Both of the methods are evaluated in the experiment part and are proven to be efficient.

In order to reduce the complexity for computing the voting density map in (7), we provide a fast implementation: Given a testing image, instead of following the computation order of (6) and (7), we go through every possible patch $\mathcal{P}(\mathbf{y})$ centered at position $\mathbf{y}$, and add weighted votes $\{w_j(\mathbf{y})\}_{j=1}^k$ to pixel $\{\mathbf{y} - \mathbf{d}_j(\mathbf{y})\}_{j=1}^k$ in a cumulative way. The accumulated vote map are then filtered by a Gaussian kernel to get the final voting density map $V(\mathbf{x})$.

3 Experiments

Data and Implementation Details

The proposed method has been extensively evaluated using 44 Ki67-stained NET microscopy images (400×400 pixels). Testing images are carefully chosen to cover challenging cases like touching cells, inhomogeneous intensity, blurred cell boundaries, and weak staining.

The architecture of our model is summarized in Fig.1. The patch size is set as 39×39 . The convolutional kernel sizes are set as 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 as 0.0001. β and λ are set to be 0.5 and 384 in (1) and (4), respectively. The threshold ξ is set to be 0.2. The Parzen-window size is 5×5 and σ is 1 in (6). These parameters are set either by following conventions from the existing works or by empirical selection. In fact, our method is robust not sensitive to parameters. Our implementation is based on the fast CUDA kernel provided by Krizhevsky [8]. The proposed model is trained and tested using a PC with NVIDIA Tesla K40C GPU and a 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 centers. In our case, r is roughly chosen to be half of the average radius of all 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 results 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 E_d) between the human annotated nucleus centers and the true positive detections, and also of the absolute difference (defined as E_n) between the number of human annotation and the true positive nucleus centers.

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

We also compare our algorithm with three state of the arts, including kernel based Iterative Radial Voting method (**IRV**) [10], structured SVM based Non-overlapping Extremal Regions Selection method (**NERS**) [2], and Image based Tool for Counting Nuclei method (**ITCN**) [4]. 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

Figure 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 1. Fast scanning produces exactly the same result as **DV-1** and thus is omitted in the table. It can be observed that deep voting consistently outperforms other methods, especially in terms of F_1 score. Although NERS [2] achieves comparable P score, it does not produce good F_1 score because Ki-67 stained NET

images contain a lot of nucleus exhibiting weak staining and similar intensity values between nucleus and background, leading to a large number of *FP*. Due to the relatively noisy probability map, both **DV-3** and **CNN-PC** provide higher *R* score but with a large sacrifice of precision. On the other hand, our method, which encodes the geometric information exhibited between 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.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 seconds 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.

3.1 Conclusion

In this paper, we propose a novel deep voting model for accurate and robust nucleus localization. We extend the conventional CNN model to jointly learn the voting *confidence* and voting *offset* by introducing a hybrid non-linear activation function. We formulate our problem as a minimization problem with incorporating a novel loss function. In addition, we also demonstrate 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 arts. We will carry out more experiments and test our method on other image modalities in the future work.

References

1. Al-Kofahi Y, Lassoued W, Lee W, Roysam B. Improved automatic detection and segmentation of cell nuclei in histopathology images. *TBME*. 2010; 57(4):841–852.
2. Arteta C, Lempitsky V, Noble JA, Zisserman A. Learning to detect cells using non-overlapping extremal regions. *MICCAI*. 2012; 7510:348–356.
3. Bernardis E, Yu SX. Finding dots: segmentation as popping out regions from boundaries. *CVPR*. 2010:199–206.
4. Byun J, Verardo MR, Sumengen B, Lewis G, Manjunath BS, Fisher SK. Automated tool for the detection of cell nuclei in digital microscopic images: application to retinal images. *Mol Vis*. 2006; 12:949–960. [PubMed: 16943767]
5. Ciresan D, Giusti A, Gambardella LM, Schmidhuber J. Mitosis detection in breast cancer histology images with deep neural networks. *MICCAI*. 2013; 8150:411–418.
6. Gall J, Lempitsky V. Class-specific hough forests for object detection. *CVPR*. 2009:1022–1029.
7. Giusti A, Ciresan DC, Masci J, Gambardella LM, Schmidhuber J. Fast image scanning with deep max-pooling convolutional neural networks. *ICIP*. 2013:4034–4038.
8. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. *NIPS*. 2012:1106–1114.
9. Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. 1998; 86:2278–2324.
10. Parvin B, Yang Q, Han J, Chang H, Rydberg B, Barcellos-Hoff MH. Iterative voting for inference of structural saliency and characterization of subcellular events. *TIP*. 2007; 16:615–623.
11. Riegler G, Ferstl D, Rüther M, Bischof H. Hough networks for head pose estimation and facial feature localization. *BMVC*. 2014

12. Toshev A, Szegedy C. Deeppose: Human pose estimation via deep neural networks. 2014:1653–1660.

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Fig. 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.

Fig. 2.

The 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.

Fig. 3.
The boxplot of F_1 score, precision, and recall on the NET data set.

Table 1

The 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	9.667 ± 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 [2]	0.859	0.602	0.692	2.936 ± 2.447	41.857 ± 57.088
IRV [10]	0.806	0.682	0.718	3.207 ± 2.173	17.714 ± 16.399
ITCN [4]	0.778	0.565	0.641	3.429 ± 2.019	33.047 ± 46.425