2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)
Venice, Italy, April 8-11, 2019

CONTEXT-REFINED NEURAL CELL INSTANCE SEGMENTATION

Jingru Yi[1], *Pengxiang Wu*[1], *Qiaoying Huang*[1], *Hui Qu*[1], *Daniel J. Hoeppner*[2], *Dimitris N. Metaxas*[1]

[1] Department of Computer Science, Rutgers University, NJ, USA
[2] Astellas Research Institute of America, IL, USA

ABSTRACT

Neural cell instance segmentation serves as a valuable tool for the study of neural cell behaviors. In general, the instance segmentation methods compute the region of interest (ROI) through a detection module, where the segmentation is subsequently performed. To precisely segment the neural cells, especially their tiny and slender structures, existing work employs a u-net structure to preserve the low-level details and encode the high-level semantics. However, such method is insufficient for differentiating the adjacent cells when large parts of them are included in the same cropped ROI. To solve this problem, we propose a context-refined neural cell instance segmentation model that learns to suppress the background information. In particular, we employ a light-weight context refinement module to recalibrate the deep features and focus the model exclusively on the target cell within each cropped ROI. The proposed model is efficient and accurate, and experimental results demonstrate its superiority compared to the state-of-the-arts. Code is available at `https://github.com/yijingru/CRNCIS-Pytorch`.

***Index Terms*—** Neural cell analysis, instance segmentation, SSD, u-net, detection, semantic segmentation

1. INTRODUCTION

In the central nervous system, neural stem cells become specified as neurons, astrocytes, and oligodendrocytes. During this process, cells are constantly sampling their environment, and making transient and long-term contacts with neighboring cells via filopodia and lamellipodia. To better understand the role of cell-cell contacts in influencing specified fates from neural stem cells, it is critical to first identify such cell interactions. With the aid of a real-time imaging system [1], this could be achieved via vision techniques. In particular, the instance segmentation serves as an important tool and can simultaneously locate and segment the cells, thus paving the way for cell analysis [2].

Neural cell instance segmentation generally consists of two tasks: cell detection and segmentation, and has been widely studied in the literature. Early works, such as [5, 6, 7], typically employ unsupervised detectors and segmentation strategies, and are therefore sensitive to intensity variations and require careful parameter tuning for every image. In recent years, motivated by the success of deep learning techniques, a number of instance segmentation methods have been developed. These approaches integrate the detection and segmentation modules into a single feed-forward network and have achieved remarkable success in natural image analysis. Representative works include MNC [8], FCIS [9] and Mask R-CNN [10]. However, these state-of-the-arts generally are ineffective under the neural cell setting for the following reasons. First, these methods are based on region proposal network (RPN), where the region proposals are resized to a small fixed size (e.g., 7×7). Consequently, they fail to preserve the tiny and slender structures of the neural cells. Second, these methods perform segmentation mainly based on deep feature maps, while ignoring the rich low-level features embedded in the shallow layers, which are beneficial to the capture of fine details of neural cells. To address such weakness, the recent work, PNCIS [2], proposes to employ the VGG16 [11]-based one-stage detector SSD [3, 12] for neural cell detection. With the predicted bounding boxes, PNCIS crops the cell ROIs from the multi-scale feature maps and applies a u-net [13] structure for segmentation. While improving the instance segmentation performance, PNCIS is ineffective in separating the neighboring cells within the same cropped region (see Fig. 1). Such false prediction oc-

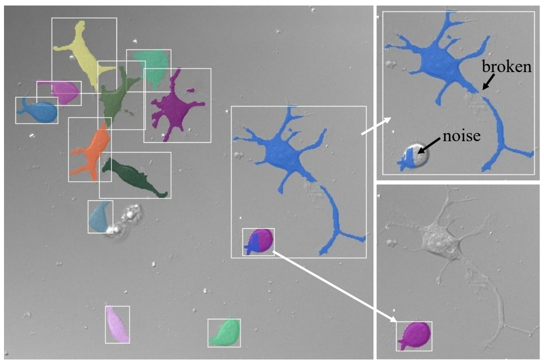

Fig. 1. The failure segmentation examples of the state-of-the-art neural cell instance segmentation method PNCIS [2]. Black arrows point to the broken segmentation and the noise induced by the adjacent cell within the same cropped region.

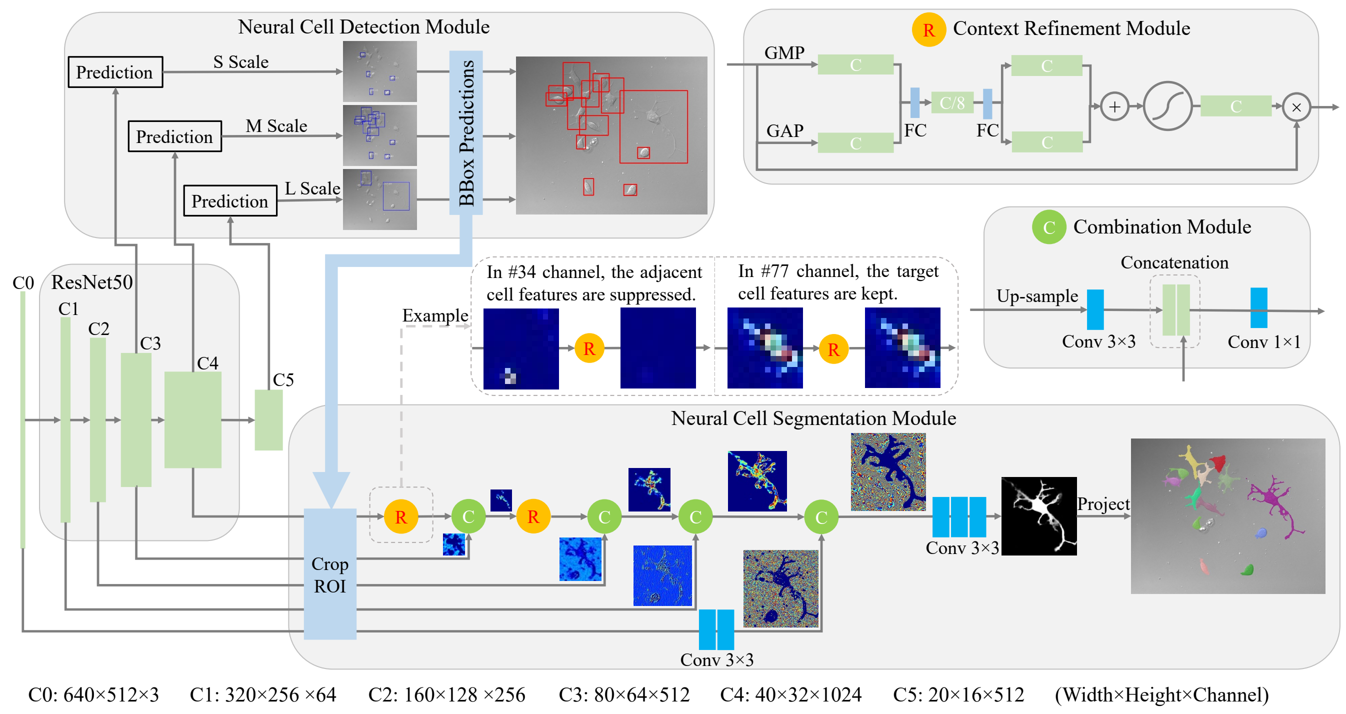

Fig. 2. Overview of the proposed context-refined neural cell instance segmentation network. The detection and segmentation modules share the feature maps. C0 represents the input image. C5 is from the original SSD [3]. C1-C4 are from ResNet50 [4]. The neural cell segmentation module crops the cell regions from C0-C4 according to the bounding box predictions. For each cell instance, the segmentation module flows the semantics from the deep layers to shallow ones through the combination modules. The context refinement module recalibrates the deep features and highlights the most salient cell.

curs when the adjacent cells are large enough to be preserved in the deep feature maps of u-net (see Fig. 2). On the other hand, PNCIS tends to generate broken and discontinuous segmentation for the fine structures of neural cells (see Fig. 1).

To solve the issues mentioned above, in this work we propose a context-refined neural cell instance segmentation model. Similar to PNCIS, the proposed model employs a joint network consisting of SSD and u-net. However, different from PNCIS, we incorporate a context refinement module that recalibrates the deep features and helps the model focus on the target cell exclusively. Therefore, our model is capable of differentiating the neighboring cells. Moreover, we apply additional three 3×3 convolutional layers on the top of the segmentation module to polish the segmentation output. Finally, we improve the model efficiency by adopting a ResNet50 [4] backbone and employing bilinear up-sampling in u-net. Compared to the existing works, the proposed model is more efficient and accurate, indicating a great potential to be applied to neural cell analysis.

2. METHODS

Fig. 2 illustrates the context-refined neural cell instance segmentation network. In particular, the model is comprised of two parts: a neural cell detection module and a neural cell segmentation module. The two modules share the same backbone feature maps and develop two different branches for the corresponding two tasks. In particular, the detection module detects the cell locations from feature maps C3, C4 and C5, where each feature map is responsible for detecting cells at a particular scale. The predicted bounding boxes are then used for cropping the cell instance regions from C0-C4. Afterward, the segmentation module receives the cropped feature maps and outputs a segmentation result for each cell instance.

2.1. Neural Cell Detection

To locate the neural cells, we utilize a one-stage object detector SSD [3] to inherit its fast detection speed. To further improve the efficiency, we employ ResNet50 [4] as the backbone instead of VGG16 [11]. The three feature maps C3, C4 and C5 are utilized to detect cells at small, medium and large scales, respectively (see Fig. 2). For each feature map, we divide it into 1×1 grids. For each grid cell, we assign N anchors with different aspect ratios and scales. The center of the grid cell represents the center of the anchors. In this work, we apply aspect ratio $a_r = \{1, 1/2, 2\}$ on C3, and $a_r = \{1, 1/2, 1/3, 2, 3\}$ on C4 and C5. For C3, C4 and C5, their minimum box scales s_{min} are 0.01, 0.1, 0.26 respectively, and the maximum box scales s_{max} are 0.1, 0.26 and 0.42. The width and height of an anchor box are $w = s\sqrt{a_r}$ and $h = s/\sqrt{a_r}$, where $s = s_{min}$. We add additional anchor boxes with $s = \sqrt{s_{min}s_{max}}$ when $a_r = 1$.

Next, we encode the ground truth boxes into the anchors and output the encoded anchors and labels. We take the encoded anchors and labels as the transferred ground truth. In particular, first, we compare each anchor with all ground truth boxes by calculating their intersection over unions (IOUs). If an IOU is greater than 0.5, the anchor box is labeled as positive, otherwise it is labeled as negative. Next, the anchor boxes are updated with the locations of their best-matched ground truth boxes. Finally, the coordinates of the encoded anchors g' are calculated by the offsets between the updated anchors g and original anchors d [3]:

$$g'_{cx} = (g_{cx} - d_{cx})/d_w, \quad g'_{cy} = (g_{cx} - d_{cx})/d_h, \tag{1}$$

$$g'_w = \log(g_w/d_w), \quad g'_h = \log(g_h/d_h), \tag{2}$$

where $(cx, cy), w, h$ represent the center coordinates, width, and height of an anchor box, respectively.

The cell detection module is trained by optimizing the following loss function:

$$L_{det} = \frac{1}{N_{pos}}(L_{locs} + L_{conf}), \tag{3}$$

where L_{locs} is the box regression loss, which is defined with smooth L_1 [14] function:

$$L_{locs} = \sum_{i \in pos} \sum_{m \in \{cx,cy,w,h\}} \text{smooth}_{L_1}(l_i^m - g_i'^m), \tag{4}$$

where l is the box predictions, and i indexes the positively labeled anchor boxes that we denote by pos, whose total number is N_{pos}. L_{conf} is the confidence loss, defined as follows:

$$L_{conf} = -\sum_i (x_i \log p_i + (1 - x_i)\log(1 - p_i)), \tag{5}$$

where x is the encoded ground truth labels, and p is the predicted box confidence.

2.2. Neural Cell Segmentation

Given a cropped cell ROI, we employ a u-net type network to extract the cell instance. However, as discussed above, in some cases the model tends to mistakenly consider the neighboring cells as foreground cell instance (see Fig. 1). One simple strategy to get rid of such error is to keep the largest connected region. However, due to imperfect segmentation, this would cause the small unconnected but essential details to be discarded as well. Instead, in this work, we propose to make the model learn to suppress the background cell signals adaptively. To this end and inspired by [15, 16, 17], we employ a context refinement module (CRM) for feature refinement.

Our key motivation and insights are illustrated in Fig. 2. Particularly, the cropped regions from C0-C4 contain two foreground neural cells: the larger one is the target cell and the smaller one is the background. As is shown, when the background cell is large enough in size, its signal is still preserved even on the small deep feature maps. Such observation motivates us to refine the high-level semantics encoded by deep feature layers, which are useful for object identification. We achieve this by applying the CRMs to the deepest cropped regions. To be specific, suppose the cropped region feature is $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$, where C is the number of feature channels, H and W are the region height and width, respectively. CRM first compresses the region features with a global max pooling (GMP) and a global average pooling (GAP), with the output denoted by $\mathbf{x}^{GMP} \in \mathbb{R}^C$ and $\mathbf{x}^{GAP} \in \mathbb{R}^C$, respectively. The GMP encourages the network to identify the most important features, such as those from target cell; while the GAP drives the network to examine the extent of the cropped region and find all distinctive parts of an object. Next, the channels of $\mathbf{x}^{GMP}$ and $\mathbf{x}^{GAP}$ are squeezed into a smaller size $C' = C/8$ by a shared fully-connected (FC) layer, which captures the most meaningful features and their variations. This is followed by a decoder FC layer for lifting the channels of $\mathbf{x}^{GMP}$ and $\mathbf{x}^{GAP}$ into the original size C. Then we combine the two features by summation $\mathbf{x}' = \mathbf{x}^{GMP} + \mathbf{x}^{GAP}$, which highlights the feature channels related to the target cell. Finally we normalize the feature vector $\mathbf{x}'$ by the Sigmoid function and perform channel-wise production between $\mathbf{x}'$ and $\mathbf{x}$ to generate the output features.

Apart from CRM, we also apply the combination modules to the relatively shallow layers, in order to flow the refined semantics from the deepest to the shallowest feature maps. As a result, the model is able to separate adjacent cells even on the shallowest layers, as is shown in Fig. 2. From Fig. 1 it can be observed that our method is able to suppress the undesired background cell and keep the target foreground cell.

The segmentation module can be trained with the following binary cross-entropy loss [2]:

$$L_{seg} = -\frac{1}{N}\sum_{i,j}(t_{ij}\log p_{ij} + (1 - t_{ij})\log(1 - p_{ij})), \tag{6}$$

where t and p represent the ground truth masks and the predicted segmentation probability maps respectively, and N is the total number of pixels considered in computation.

3. EXPERIMENTS

We use the same neural cell dataset as PNCIS [2]. The dataset is sampled from the time-lapse microscopy videos, and consists of 386 training images, 129 validation images and 129 testing images. The size of an input image is 512×640. As the dataset is small, we finetune the ResNet50 with weights pre-trained on ImageNet. The other weights of the model

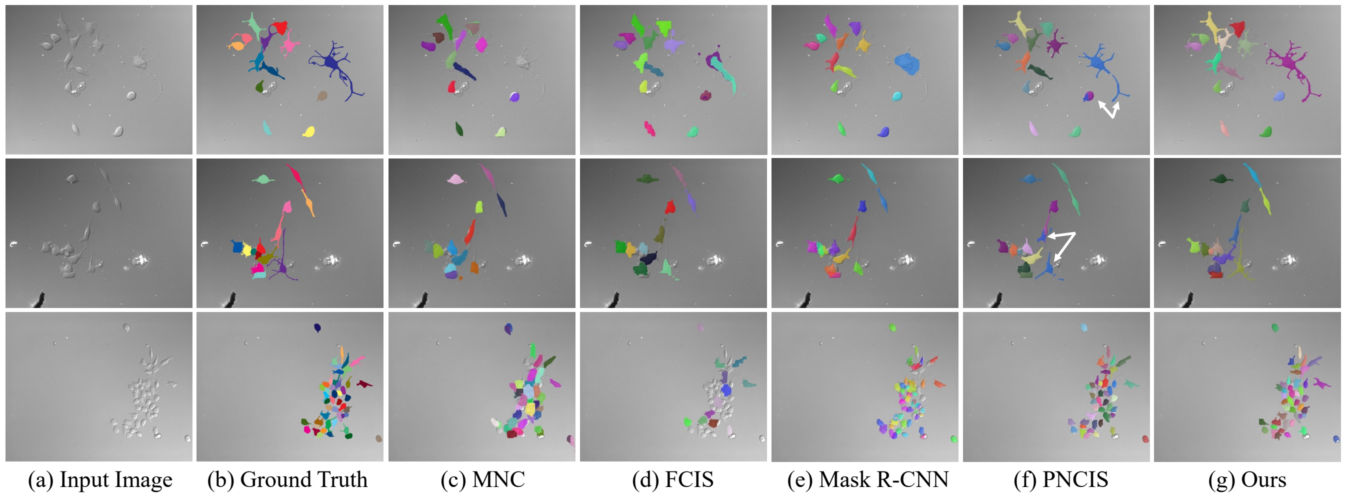

Fig. 3. Qualitative comparisons among different methods. White arrows indicate the prediction errors of PNCIS [2].

are initialized from a standard Gaussian distribution. We employ data augmentation and early-stop strategies to avoid over-fitting. The proposed model can be trained end-to-end. However, to accelerate the training, in practice we first train the detection module for 400 epochs with batch size 36 on 4 GPUs. Then we freeze its weights and train the segmentation module for 120 epochs with batch size 10 on a single GPU. The model is implemented with PyTorch [18]. We compare our method with several state-of-the-arts, including MNC [8], FCIS [9], Mask R-CNN [10] and PNCIS [2]. All methods are tested on a single NVIDIA K40 GPU.

As in [2, 8, 9, 10], we use the average precision (AP) [19] at mask-level IOU threshold of 0.5 and 0.7 (AP@0.5, AP@0.7) to evaluate the instance segmentation performance. The AP metric summarizes the shape of precision/recall curve, and measures both instance detection and segmentation accuracies. We also report the average mask-level IOU between the predicted segmentation masks and the ground truths at threshold of 0.5 and 0.7 (IOU@0.5, IOU@0.7), as in PNCIS [2].

Compared to the state-of-the-arts, the proposed method achieves the best accuracy and the highest efficiency (see Table 1). Fig. 3 illustrates the qualitative results. It can be observed that both MNC and FCIS achieve rough segmentations as they only utilize the top feature maps. Moreover, FCIS is weak in detecting small objects. In contrast, Mask R-CNN achieves better detection and segmentation performance because it distributes the anchors to multi-scale feature maps. Moreover, it adopts the feature pyramid network (FPN) to encourage the semantic communication. However, Mask R-CNN is more computationally expensive compared to the other methods, and due to the failure to consider the shallow layers which are rich in low-level details, Mask R-CNN cannot capture the slender protrusions of the neural cells. Compared to all these methods, PNCIS achieves much better performance due to the u-net structure, which explicitly utilizes the shallow layers and effectively combines the semantics from different layers. However, for a given target cell, when the neighboring cells are large enough and within the same cropped region, in general, PNCIS is unable to separate them (e.g., those pointed by the white arrows in Fig. 3). By comparison, with the proposed context refinement module, our model is able to suppress such mistakes. Besides, the segmentation results of the slender structures by our method are more continuous with fewer breaks. Finally, our model has fewer parameters (7M) than PNCIS (35M) and is quite efficient. These properties indicate that our method has a great potential to be applied to neural cell analysis.

Model	Time (s)	AP@0.5	AP@0.7	IOU@0.5	IOU@0.7
MNC [8]	0.475	48.72	11.37	62.73	75.47
FCIS [9]	0.213	66.02	7.13	64.85	75.07
Mask R-CNN [10]	0.749	66.02	32.10	72.10	79.30
PNCIS [2]	0.381	85.70	70.94	78.84	81.22
Ours	**0.092**	**89.15**	**71.48**	**79.13**	**81.47**

Table 1. Comparison of the segmentation results from different methods. Time is measured on an NVIDIA K40 GPU.

4. CONCLUSION

In this work, we propose a context-refined instance segmentation model for neural cells. With the context refinement module, our method is able to recalibrate the deep features and guide the model to suppress the background information from the adjacent cells. The proposed model is efficient in inference and achieves accurate segmentation performance. These characteristics suggest that our model could be well applied to and helpful for the study of neural cells.

5. REFERENCES

[1] Rea Ravin, Daniel J Hoeppner, David M Munno, Liran Carmel, Jim Sullivan, David L Levitt, Jennifer L Miller, Christopher Athaide, David M Panchision, and Ronald DG McKay, "Potency and fate specification in cns stem cell populations in vitro," *Cell stem cell*, vol. 3, no. 6, pp. 670–680, 2008.

[2] Jingru Yi, Pengxiang Wu, Daniel J Hoeppner, and Dimitris Metaxas, "Pixel-wise neural cell instance segmentation," in *ISBI*. IEEE, 2018, pp. 373–377.

[3] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "Ssd: Single shot multibox detector," in *ECCV*. Springer, 2016, pp. 21–37.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[5] Karin Althoff, Johan Degerman, and Tomas Gustavsson, "Combined segmentation and tracking of neural stem-cells," in *SCIA*. Springer, 2005, pp. 282–291.

[6] Amalka Pinidiyaarachchi and Carolina Wählby, "Seeded watersheds for combined segmentation and tracking of cells," in *ICIAP*. Springer, 2005, pp. 336–343.

[7] Huiming Peng, Xiaobo Zhou, Fuhai Li, Xiaofeng Xia, and Stephen TC Wong, "Integrating multi-scale blob/curvilinear detector techniques and multi-level sets for automated segmentation of stem cell images," in *ISBI: From Nano to Macro*. IEEE, 2009, pp. 1362–1365.

[8] Jifeng Dai, Kaiming He, and Jian Sun, "Instance-aware semantic segmentation via multi-task network cascades," in *CVPR*, 2016, pp. 3150–3158.

[9] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *CVPR*, July 2017, pp. 4438–4446.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask r-cnn," in *ICCV*. IEEE, 2017, pp. 2980–2988.

[11] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2014.

[12] Jingru Yi, Pengxiang Wu, Daniel J Hoeppner, and Dimitris N Metaxas, "Fast neural cell detection using light-weight ssd neural network.," in *CVPR-W*, 2017, pp. 860–864.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*. Springer, 2015, pp. 234–241.

[14] Ross Girshick, "Fast r-cnn," in *CVPR*, 2015, pp. 1440–1448.

[15] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon, "Cbam: Convolutional block attention module," in *ECCV*, 2018.

[16] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic, "Is object localization for free?-weakly-supervised learning with convolutional neural networks," in *CVPR*, 2015, pp. 685–694.

[17] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba, "Learning deep features for discriminative localization," in *CVPR*, 2016, pp. 2921–2929.

[18] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[19] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.