Fast Neural Cell Detection Using Light-Weight SSD Neural Network

Jingru Yi
Department of Computer Science
Rutgers University, NJ, USA
jy486@cs.rutgers.edu

Pengxiang Wu
Department of Computer Science
Rutgers University, NJ, USA
pw241@cs.rutgers.edu

Daniel J. Hoeppner
Lieber Institute for Brain Development
Hopkins Medical Campus, MD, USA
daniel.hoeppner@libd.org

Dimitris Metaxas
Department of Computer Science
Rutgers University, NJ, USA
dnm@cs.rutgers.edu

Abstract

Identifying the lineage path of neural cells is critical for understanding the development of brain. Accurate neural cell detection is a crucial step to obtain reliable delineation of cell lineage. To solve this task, in this paper we present an efficient neural cell detection method based on SSD (single shot multibox detector) neural network model. Our method adapts the original SSD architecture and removes the unnecessary blocks, leading to a light-weight model. Moreover, we formulate the cell detection as a binary regression problem, which makes our model much simpler. Experimental results demonstrate that, with only a small training set, our method is able to accurately capture the neural cells under severe shape deformation in a fast way.

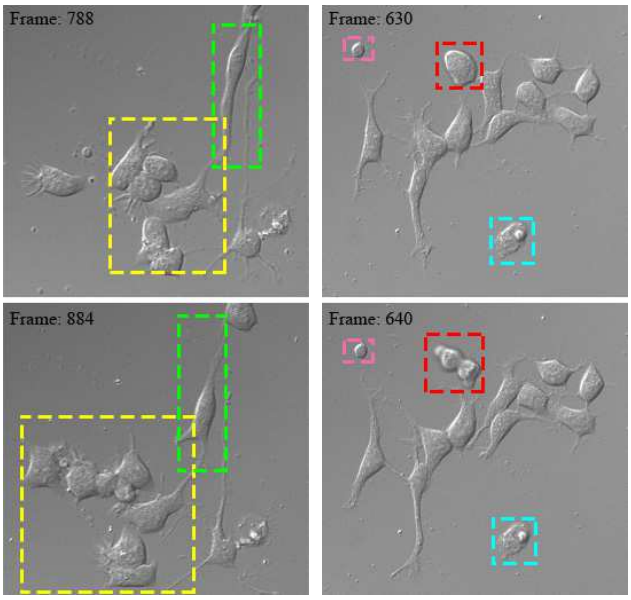
1. Introduction

The establishment of lineage path from a single neural stem/progenitor cell to the production of neurons, astrocytes, and oligodendrocytes is critical to the study of normal brain development [8]. In the lineage history, neural cells divide themselves through mitosis, and physically interact with each other through filopodia and lamellipodia, aiming to establish local niches for differentiation, promote maturation and ultimately produce neuronal synapses.

One way to investigate the lineage path for neural cells is by recording the cell fate transitions in time-lapse videos. Then different types of neural cell descendants in the video are detected, which is a crucial pre-requisite step to classifying the cells, capturing interactions between them and identifying the cell lineage. Conventionally, this is manually performed by experts who examine the time-lapse videos frame by frame. However, manual annotation suffers from the issues such as considerably time-consuming and limited reproducibility. Therefore, automatic cell detection methods are highly demanded to improve the efficiency and reduce the workload on researchers.

Nevertheless, this task is quite challenging due to the severe aggregation and shape distortion of cells, and the sharp increase of cell numbers (Figure 1). In addition, the unnoticeable background changes (e.g., lighting condition), as well as the existence of dead cells and impurities in the background, pose another challenge to detection. These factors render traditional handcrafted feature-based detection methods, such as HOG feature-based detector [2], powerless for coping with this problem. .

Recent years have witnessed the great success of deep neural networks (DNNs) in object detection [5, 4, 11, 9, 10, 7]. Unlike handcrafted features, the features of DNNs are built through learning and therefore are more general and scalable, leading to better performance in practice. One groundbreaking work was by Girshick *et al*. [5], who proposed a R-CNN method which combines classical region proposals with convolutional neural networks (CNNs) for robust object detection and classification. R-CNN achieved great success and was further improved in [4, 11]. However, while accurate, the family of R-CNN methods still remain too computationally expensive and too slow for real-time applications. To overcome this problem, Redmon *et al.* [9] proposed to predict the object bounding box directly with a novel YOLO detection system, which is much simpler and faster than R-CNN while achieving higher precision. In [7], Liu *et al.* [7] went a step further and presented a single shot multibox detector (SSD) method which eliminates proposal generation and subsequent resampling stages, and is able to obtain higher accuracy and speed than YOLO system. In this paper, we employed a modified light-weight SSD model for neural cell detection. The adapted model formulates the cell detection as a binary regression problem, and



(a) (b)

Figure 1: Illustration of neural cell detection challenges. (a) In the process towards maturation, neural cells tend to stretch their filopodia and lamellipodia, distort themselves (green box) and aggregate (yellow box), posing challenges to robust cell detection. (b) To form a neural network, they need to perform mitosis (red box), which leads to a large change of appearance. Besides, the existence of dead cells (cyan box) and impurities (pink box) also adds challenges to the extraction of cells from background.

is much smaller in size than the original version.

One issue with DNNs is the heavy demand for training data. In the presence of small dataset, such as our neural cells (with only 1595 frames), the model performance could decrease sharply. To deal with this problem, we transfered the learned weights from pre-trained model (e.g., VGG-16 [13] for our case) to our adapted version, and fine-tuned it on the neural cell images. This transfer learning process enables our cell detection model to inherit the learned concept of 'objectness' [6] from the trained SSD model, and thereby helps our model capture the features of neural cells with only a small training set. Experimental results demonstrate that our method is able to detect the cells in a fast way with relatively high accuracy.

The rest of this paper is organized as follows. We present the details of our cell detection model in Section 2. Experimental results and corresponding discussion are offered in Section 3. Our work is summarized in Section 4.

2. Methods

2.1. Network architecture

The neural network architecture of our method is shown in Figure 2. For each input image, it is formed as a mean-subtracted $300 \times 300 \times 3$ tensor. Then a series of feed-forward convolutional layers are applied to the input image. The bottom part of the network (Block 1 to Block 5) is based on VGG-16 [13], for which the pre-trained weights on ImageNet [12] are available. We transfer these learned weights to the bottom part and fine-tune them on our neural cell images. Compared to the original SSD model, our adapted version removes several intermediate layers and is therefore much smaller in size. In particular, our network model is 1 billion FLOPs (in terms of multiplication) smaller than that of the original version.

In order to perform detection on multiple scales, layers from Block 4 (38×38) and Block 7 (19×19) are extracted as our feature maps. For each feature map, it contains a particular number of default bounding boxes, which are of certain scales and aspect ratios (see Section 2.2). Two different kinds of filters, i.e., localization and objectness, are applied to the two feature maps to predict the bounding box offsets as well as the objectness scores. Finally, a sigmoid layer is added to the end of objectness prediction layer to restrict the objectness score to $[0, 1]$.

2.2. Default box proposals

Two commonly used strategies for generating possible object locations are sliding windows and region proposals. However, the sliding window method suffers from extremely high computational cost in training and prediction. When there are many choices of window aspect ratios and scales, the computation would become infeasible. By contrast, region proposals work better but are still expensive in computation [11, 14]. To solve this issue, SSD model employs default boxes [7], which largely improves the speed vs accuracy trade-off.

In default box strategy, the feature maps (38×38 and 19×19) are divided into grids with blocks of size 1×1, and each of these blocks determines the centers of default boxes. Specifically, the centers of default boxes are set to $(cx_d, cy_d) = (\frac{i+0.5}{l}, \frac{j+0.5}{l})$, $i, j = 0, 1, 2, \cdots, l-1$, where l is the size of feature map. Note that it is possible that different default boxes correspond to one block center, depending on the box width and height settings.

During the process towards maturation, neural cells tend to become slender (the green box in Figure 1) so that they could actively communicate with other cells around and finally form a neural network. According to this property, we choose the aspect ratios $a_r \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$ for 38×38 feature map, and $a_r \in \{1, 2, 3, 4, 5, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}\}$ for 19×19 feature map. For the scales of bounding boxes, they can be

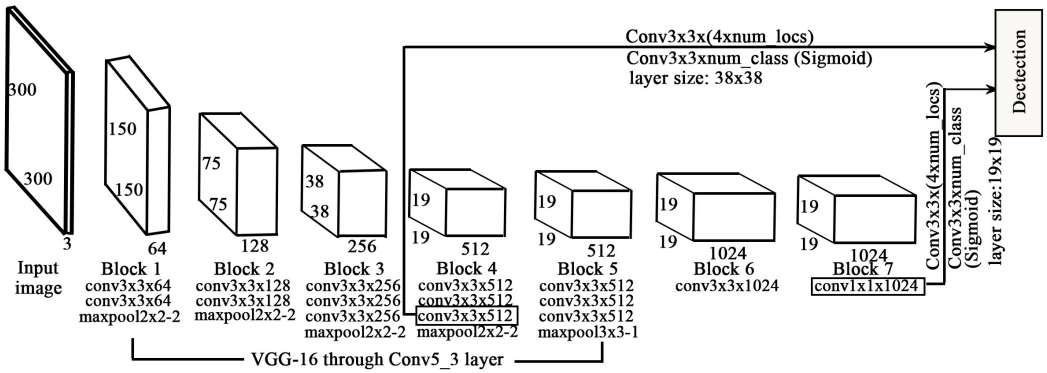


Figure 2: The overview of detection network architecture. Two feature maps (38×38 and 19×19) are utilized to predict the location offset and objectness score for each cell. Here, num_locs indicates the number of bounding boxes, which depends on the bounding box scale and aspect ratio specified for a particular layer; and num_class is the number of cell classes (which is 1 in this paper).

chosen to be $s_k \in \{0.07, 0.15, 0.33\}$, where k is the index of feature maps. We use $s_3 = 0.33$ here because for aspect ratio of $a_r = 1$, we also add a default box whose scale is $s'_k = \sqrt{s_k s_{k+1}}$ [7].

After setting the scales and aspect ratios for the default boxes, we can then calculate their widths and heights as $w_k = s_k\sqrt{a_r}$ and $h_k = s_k/\sqrt{a_r}$, respectively. Thus in total we have 6 default boxes for each block center in 38×38 feature map, and 10 default boxes per block in 19×19 feature map. Our next step is to encode the groundtruth bounding boxes as default boxes.

2.3. Encode groundtruth boxes

After obtaining default boxes from section 2.2, we need to encode the groundtruth bounding boxes as default boxes such that we can get their offsets relative to the grid block centers, the corresponding default box labels and their objectness scores. In this way we transform the groundtruth bounding box into a form that can be fed into our SSD model for training.

The transformation steps are as follows. First, we compare the groundtruth box with each default box by calculating their jaccard index. If the jaccard value is greater than the ignore threshold (0.45), then the default box is just the corresponding encoded result. In this case, the objectness label of the default box will be set to 1, the jaccard value will be kept as the objectness score of the default box, and the location offsets between groundtruth box and encoded default box will also be recorded. However, it is possible that two groundtruth boxes would match the same default box. In this situation, we just keep the one with higher jaccard value. Therefore, to sum up, the encoded groundtruth vector consists of location offset ($g = (cx, cy, w, h)$), objectness score ($p \in [0, 1]$) and label ($x \in \{0, 1\}$). Note that the location offsets are calculated as [7]:

$$cx = (cx_g - cx_d)/w_d \tag{1}$$

$$cy = (cy_g - cy_d)/h_d \tag{2}$$

$$w = \log(\frac{w_g}{w_d}) \tag{3}$$

$$h = \log(\frac{h_g}{h_d}) \tag{4}$$

where (cx, cy) is the center of the encoded box, and (w, h) represents its width and height. The subscript index g and d refer to the groundtruth box and default box, respectively.

2.4. Loss function

The total loss is a weighted sum of location offset loss and the objectness score loss:

$$L = \frac{1}{N}(L_{obj} + \alpha L_{loc}). \tag{5}$$

where N is the number of matched default boxes. The location offset loss is defined as [4, 7]:

$$L_{loc} = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} \text{smooth}_{L_1}(l_i^m - g_i^m) \tag{6}$$

$$\text{smooth}_{L_1}(z) = \begin{cases} 0.5z^2 & \text{if } |z| < 1 \\ |z| - 0.5 & \text{otherwise} \end{cases} \tag{7}$$

where i is the index of default box with label 1, and l and g refer to predicted box offsets and encoded box offsets, respectively. The objectness score loss is calculated by the

binary cross-entropy:

$$L_{obj} = -\sum_{i \in Pos}^{N} x_i \log(p_i) - \sum_{j \in Neg}^{N} (1 - x_j) \log(1 - p_j) \tag{8}$$

where p is the objectness score of the predicted box, and $x \in \{0, 1\}$ is the encoded label, which indicates the objectness of default boxes. Hard negtive mining [7] is utilized when calculating the objectness score loss for the negtive default boxes. In this way we keep a balance between the positive and negative training examples.

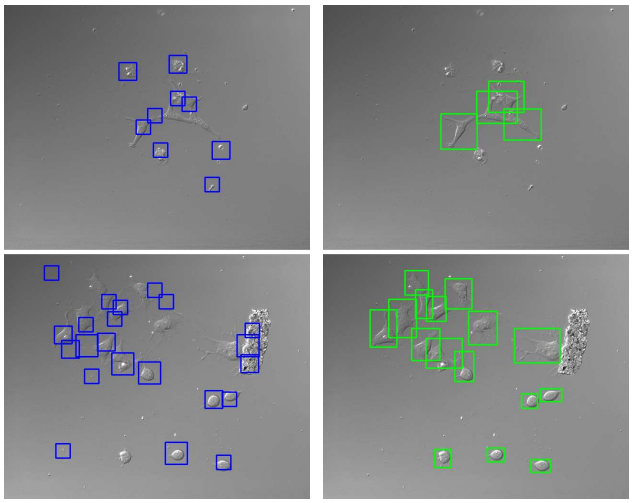
3. Experiments

The neural cell data used in our experiment came from a series of time-lapse microscopy videos, from which we sampled 2658 images in total. Among these images, 60% are used for training, 20% for validation, and 20% for testing. Considering that our training set is relatively small (with only 1595 images), we additionally performed data augmentation by random cropping and flipping. In the prediction part, boxes with objectness scores greater than 0.4 are accepted as cell bounding boxes. Non-max suppression with 0.5 threshold is utilized to get rid of similar bounding boxes. Our model was implemented in Python using Keras library [1], and was trained and tested on a single Nvidia K40 GPU.

The detection accuracy is calculated according to [3]. If the jaccard overlap between the predicted bounding box and the groundtruth box exceeds 0.5, the predicted box will be considered as correct. The accuracy for testing images (531 frames in total) is 83%, and the detection speed is 10 FPS on average. We believe that with more powerful hardware (e.g., Nvidia Titan X used by original SSD model [7]), the detection speed would be further increased.

Figure 3 demonstrates several examples of our detection results. It can be observed that our method is able to gracefully deal with cell aggregation and shape deformation. We also show several failure cases in Figure 3 (pointed to by arrows), which are probably caused by small number of feature maps or the lack of scale and aspect ratio options for default boxes. One way to resolve this issue is to enrich the available scale and aspect ratio specifications, which, however, also means the increased need for training data.

To illustrate the advantage of our light-weight SSD model over handcrafted feature-based detection method, in Figure 4 we compare our model with classical HOG-based SVM detector [2]. It can be observed that HOG detector is quite sensitive to impurities and fails to distinguish between dead and normal cells, whereas our method is comparatively more robust and handles the background interference well. The precision and recall for HOG detector are 60% and 44%, while for our light-weight SSD model they are 83% and 96% respectively, outperforming HOG detector by a large margin. Moreover, our method is $67\times$ (0.1s per frame) faster than HOG detector (6.72s per frame).



(a) (b)

Figure 4: Comparison of the detection results by (a) HOG-based SVM detector and (b) our light-weight SSD model. Compared to HOG detector, our model handles the interference of impurities and dead cells better, and captures the target cells in a more accurate way.

4. Conclusion

In this paper, we applied a modified light-weight SSD model to the problem of neural cell detection. Compared to the original SSD architecture, our adapted version is much smaller in size. Besides, our model formulates the cell detection as a binary regression problem, making it simpler to deploy. Experimental results demonstrate that our method is able to detect the neural cells in an accurate and fast way, thereby setting the foundation for future cell classification and lineage establishment.

References

[1] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893 vol. 1, June 2005.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[4] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

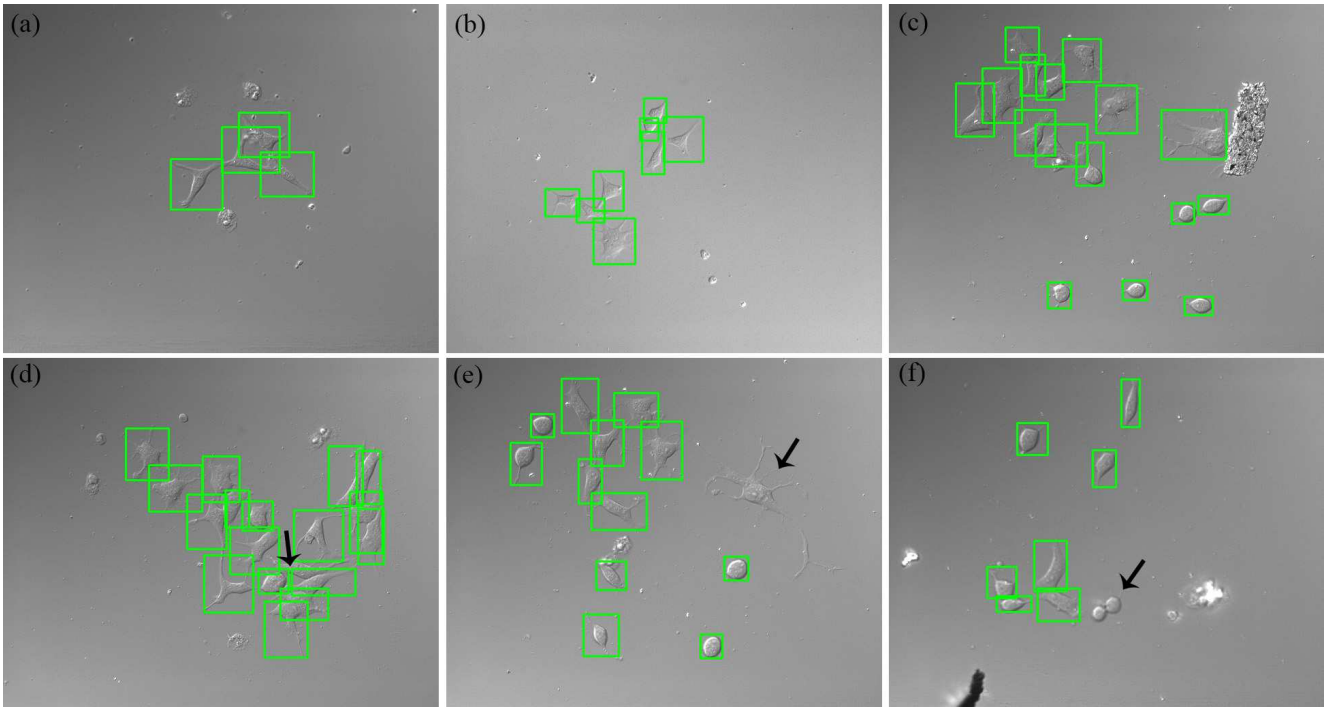


Figure 3: Detection results. Top row: successful examples, where the predicted bounding boxes for neural cells are represented by green boxes. Bottom row: examples with failure cases, which are pointed to by black arrows. In particular, (d) and (e) illustrate respectively the cells that are too long and large to capture, while (f) provides a failure instance where the two newly generated cells from mitosis are too small to recognize.

[5] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[6] S. D. Jain, B. Xiong, and K. Grauman. Pixel objectness. *CoRR*, abs/1701.05349, 2017.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[8] R. Ravin, D. J. Hoeppner, D. M. Munno, L. Carmel, J. Sullivan, D. L. Levitt, J. L. Miller, C. Athaide, D. M. Panchision, and R. D. McKay. Potency and fate specification in cns stem cell populations in vitro. *Cell Stem Cell*, 3(6):670–680, 2004.

[9] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[10] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[11] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[14] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.