Blood Cell detection using Singleshot Multibox Detector

Inyoung Huh
Computational Science, Mathematics and Engineering
University of California, San Diego
i1huh@ucsd.edu

***Abstract*— Automatically cell detection in microscopy images has become an important task to a wide range of biomedical research. In machine learning field, the state of the art methods for detection have been developed more than classification. In this paper, I designed Red blood cell detector using Singleshot Multibox Detection algorithm, which can detect only blood cell in an image. With more than 99% accuracy, cell detector can find red blood cell's location.**

I. INTRODUCTION

Over past few years, the neural network has evolved fast. The advent of deep learning enable machine learning not only to apply to everyday lives but also to use an interdisciplinary research. Now we can easily find the application of deep learning. Deep learning architecture has been used in many fields, such as computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics and drug design. Especially, application of deep learning in medicine is incredible. Deep learning can help doctors make faster, more accurate diagnoses. It helps doctors can see, which means that enhances doctors ability to analyze medical images. For example, by training detecting the tumor from input images, deep learning model can assist doctors to decide proper diagnosis. It is of significant interest to a wide range of medical imaging tasks and clinical applications. This paper is about cell detection motivated by this insight. Cell detection is to find whether there are certain types of cells present in an input image and to localize them in the image. This paper will begin with a discussion of Single Shot Multibox Detection algorithm. The data used in this paper is from Kaggle dataset. The input to this algorithm are blood cell images, including bounding boxes and class of blood cells in an image. The Single Shot Multibox Detection algorithm will predict the class and location of blood cell in each images.

II. RELATED WORK

In the last few decades, different cell recognition methods had been proposed. These methods can be divided into traditional methods and fully deep learning based methods. Traditional cell detection is based on two approaches: Hand-crafted feature representation+classifier. For example, Laplacian-of-Gaussian (LoG) [1] operator was famous for blob detection. Gabor filter or LBP feature [2] provide interesting texture properties and had been attempted for a cell detection task [3]. The process of this methods is, first, detection system extracts the features as the representation of input images and machine learning algorithm is applied the feature vectors to recognize the location around target cells. However, even though deep learning methods can be used as classifier, since these are mostly relied on Hand-crafted feature representation, there are always some risk that suitable features are selected by human and the features are tightly coupled with others and some limitation when massive inputs are needed. Fully deep learning approach can resolve these problems. Unsupervised learning enable to evolve itself from fixed feature towards automated learning of problem-specific features directly from training data. Therefore, users do not have to go into the elaborate procedure for the extraction of features. For example, collaboration with convolutional neural network and compressed sensing performed well for cell detection which is the first successful use of convolutional neural network with compressed sensing based output space encoding [4]. A sparse color unmixing technique with convolutional neural network presents a success for automatic immune cell[5].

III. METHODS

A. Convolutional Neural Network

Convolutional neural network is specialized in analyzing visual imagery. While traditional neural network consists of input, hidden and output layers simply, convolutional neural network consist of four main layers: Convolutional Layer, Pooling Layer, Dropout Layer and Fully-Connected Layer. These layers can be said to basic architecture of convolutional neural network. By stacking these layers, we can design a full convolutional neural network. Convolutional layer consist of a set of independent filters. When we make design choices of convolution layers, we must decide on a kernel size. The kernel is slid over image taking a "snapshot" at each step, the slide is managed with a choice of stride size, allowing each snapshot to overlap the previous. This process is the convolution of the pixels set in one snapshot with the next. A strength of the convolutional network in image classification is its use of local structure in the image to support feature learning. This ability is rooted in the overlapping of these "snapshots" in the convolution layers.

Sandwiched between convolution layers are pooling, dropout, or normalization layers depending on design choice. Pooling layer will perform a downsampling operation along the spatial dimensions. They are popularly chosen with 2×2 receptive fields. From a receptive field a representative maximum value pixel is chosen as active and is passed forward through the network, non-active pixels are effectively

zeroed. Gradients backpropogate through the active pixel. This process reduces the size of the input, not necessarily the depth though. Dropout layers are used to reduce the number of parameters and help with overfitting by reducing parameters and therefore allowable complexity of the network. Convolution layers can increase the input depth to the next layer from say, 3 initially for RGB, to 64. The added depth allows for more complex features to be learned. Associated with this is an explosion of parameters. The dropout layer will choose to randomly drop some percentage of the parameters, in effect, creating a different architecture. These changing architectures give more flexibility to what may be learned by the network. Same as traditional neural network, fully connected layer will connect every output neurons and compute the class scores.

B. VGG16

Single Shot Multibox Detector is based on convolutional network. Its base network: VGG16. VGG16 is a pre-trained model, which is already trained on a dataset and contains the weights and biases that represent the features of whichever dataset it was trained on. Learned features are often transferable to different data. Usually, when we use pre-trained model, we only use the architecture by discarding fully connected layers. Therefore, in Single Shot Mulitibox detector, VGG16 is used for base network without fully connected layer. Fully connected layers of Single Shot Multibox Detector is replaced by other network.

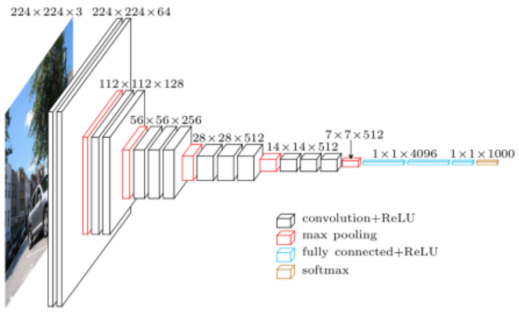


Fig. 1. Sliding Window

C. Single Shot Multibox Detector

Traditionally, image processing problem was limited to classification problem. However, there is a deeper problem: object detection. While classification is about predicting label of the object present in an image, detection goes further than that and finds locations of those objects. In image classification, we predict the probabilities of each class, while in object detection, we also predict a bounding box containing the object of that class.

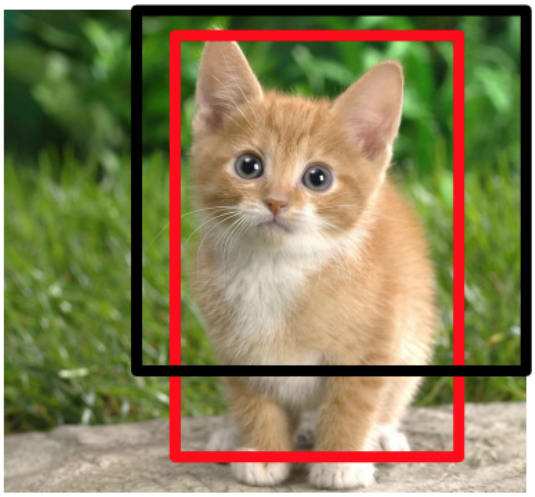


Fig. 2. Sliding Window

Single Shot Multibox Detector have great balance of accuracy and speed. As I mentioned before, base network for Single Shot Multibox is VGG16. VGG16 can be used the extract feature maps. Full connected layers, called Extra feature Layers is the key of this architecture. Basic idea of object detection is sliding window. Sliding window idea is that we crop the patches contained in the boxes, resize and insert them to convolutional network. By repeating this process with smaller window size , detector can capture objects of smaller size. However, it occurs lots of computation. In order to reduce computational cost, Single Shot Multibox Detector use default boundary boxes. Every feature map cell is matched with a set of default bounding boxes. It enable to measure different dimensions and aspect ratios. These bounding boxes are manually chosen based on IoU with respect to the ground truth was over 0.5.

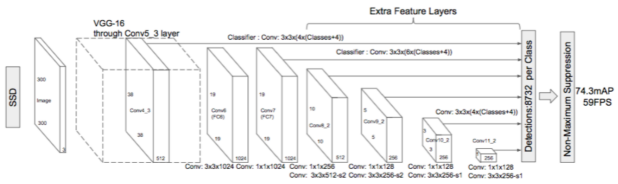


Fig. 3. Single Shot Multibox Detector Architecture

Since object detection is the process solving classification and localization problem, we should consider two type of loss: Confidence loss and location loss. Confidence loss measure how accurate the network is distinguishing the objects class. Cross-entopy is used to compute this loss. Location Loss measure how far away the networks predicted bounding boxes are from the ground truth ones from the training set. L1-norm is used for Single Shot Multibox Detector. x_{ij}^{p} = 1, 0 be an indicator for matching the i-th default box to the j-th ground truth box of category p. c indicate the softmax loss over multiple classes confidence.

We can calculate confidence loss as following.

$$L_{conf}(x,c) = -\sum_{i\in Pos}^{N} x_{ij}^{p} log(c_i^p) - \sum_{i\in Neg} log(c_i^0) \quad (1)$$

l is the predicted box and g is the ground truth box parameters. We can calculate location loss as follows.

$$L_{loc}(x,l,g) = \sum_{i\in Pos}^{N} \sum_{m\in cx,cy,w,h} x_{ij}^{k} smooth_{L1}(l-g) \quad (2)$$

Total loss can be calculated with summation of location loss and confidence loss and weight term α.

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)) \quad (3)$$

IV. Experiments/Results

A. Data

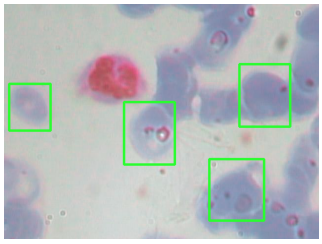
In order to train the model with Single Shot Multibox Dector, we need object image, category and its bounding boxes (x,y coordinates). The data used for this project is provided by Kaggle dataset. This project is only for detection Red Blood Cell. Therefore, class category is "Red Blood Cell". Each image contains multiple Red Blood Cells. Therefore, in one image, there are more than four bounding boxes in one image. 200 images with 1,453 labels are used as input data and it is divided into training data and test data with 8:2 ratio. Input data is as following.

filename	width	height	class	xmin	ymin	xmax	ymax
BloodImage_00000.jpg	640	480	RBC	216	359	316	464
BloodImage_00000.jpg	640	480	RBC	77	326	177	431
BloodImage_00000.jpg	640	480	RBC	540	353	640	458
BloodImage_00000.jpg	640	480	RBC	405	350	513	457
BloodImage_00000.jpg	640	480	RBC	160	72	245	177
BloodImage_00000.jpg	640	480	RBC	5	335	90	440
BloodImage_00000.jpg	640	480	RBC	540	39	640	149
BloodImage_00000.jpg	640	480	RBC	383	1	504	113
BloodImage_00000.jpg	640	480	RBC	9	82	108	168
BloodImage_00000.jpg	640	480	RBC	68	212	165	346
BloodImage_00000.jpg	640	480	RBC	171	181	264	282
BloodImage_00001.jpg	640	480	RBC	284	342	365	449
BloodImage_00001.jpg	640	480	RBC	334	366	448	454
BloodImage_00001.jpg	640	480	RBC	412	286	542	386
BloodImage_00001.jpg	640	480	RBC	435	1	569	70

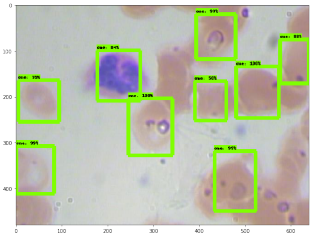
Fig. 4. Input data example

B. Experiments

Tensorflow Object Detection API provide several different models for object detection. All models are already pre-trained by COCO dataset, which is the photo collection of common objects in the world. Additionally, they are provided two type of algorithms: Mobilenet and inception version. Difference between Mobilenet and inception one is that Mobilenet is separable convolution while Inception uses standard convolution. Mobilenet is lighter than Inception since it is implemented for mobile application. Since I trained the model with local computer, Mobilenet is used for detection training. It is only for detecting single object: Blood cell. Number of class should be one. I set the batch size is 24. If batch size is too large, it occurs memory error. Initial learning rate is 0.004 and it decreases by every epoch. It is often useful to reduce learning rate as the training progresses because it helps coverage slowly and not to pass the optimal.

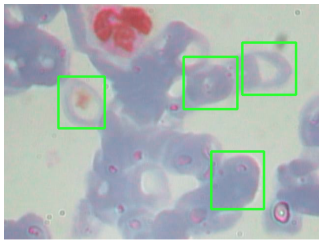


(a) $TestImage$

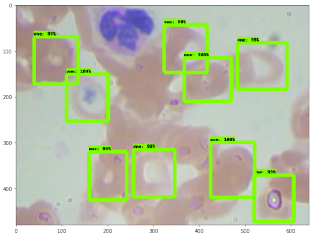


(b) $BoundingBoxesbydetector$

Fig. 5. Test Data 1

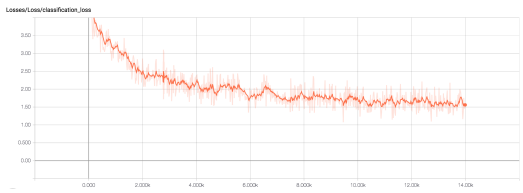


(a) $TestImage$

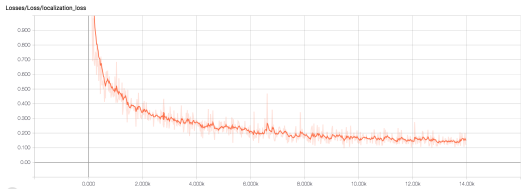


(b) $BoundingBoxesbydetector$

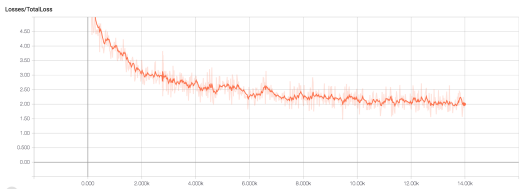
Fig. 6. Test Data 2



(a) $ClassificationLoss$



(b) $LocalizationLoss$



(c) $TotalLoss$

Fig. 7. Loss

C. Results

I trained the model 14,000K epoch for one day. We can see interesting features from the final results. First, the final image shows the more number of bounding boxes before training. In terms of accuracy, predicting the bounding boxes show good accuracy. In terms of detecting bounding boxes, it shows better results than human. In figure 4 and 5 (a), there are only four bounding boxes even though there are more blood cell in the image. However by Single Shot Multibox detector, more red blood cells can be found. It implies that cell detector trained by Single Shot Multibox Detector algorithm can outperform than human observer. It can be applied to help human by finding the feature which is hard to be distinguished by human eyes.

In figure 4 (b), it detected the wrong cell as red blood cell. I can guess two why this interesting results come out. Since the input images was limited to the blood cell but each blood cell image do not completely look same. Some images has a spot in the middle of the cell while others do not. In figure 4 (b), the wrong cell which is not blood cell has also a spot in it. Inconsistent input images may cause to detect this wrong cell as red blood cell. It also implies the risk of detection by machine.

V. CONCLUSIONS

This paper is about finding the blood cells in an image with singleshot multi box detection. Object detection is modeled as classification problem. However it includes localization problem. Therefore it predict not only objects class but also its location in the image. There are several the state-of- the-arts methods for object detection. Among these methods, Single Shot Multibox Detector is faster and more accurate than other methods. Using Single Shot Multibox Detector algorithm, I trained the model to detect the red blood cells in the image. This detector showed incredible outperforming results than human observer. I would like to improve this the two direction for future work. First one is improving this work as multi label model. Since the cell detector was trained with single label, it is difficult to distinguish two different but similar type of cells in figure 4. If this model is trained with multi-labels, it will distinguish exactly which one is red blood cell or not. Additionally, according to related works, it is still not fully deep learning algorithm. In the process to annotate bounding boxes around cells, lots of task depends on Hand-crafted feature representation. This process contains risk and limitation. We can improve this model with using fully deep learning or applying other machine learning algorithms.

REFERENCES

[1] Hui Kong, Hatice Cinar Akakin, and Sanjay E. Sarma. A generalized laplacian of gaussian filter for blob detection and its applications. IEEE Transactions on Cybernetics, 43:17191733, 2013.

[2] T. Ojala, M. Pietikinen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. Pattern Recognition, 29:5159, 1996.

[3] Yousef Al-Kofahi, Wiem Lassoued, William Lee, and Badrinath Roysam. Improved automatic detection and segmentation of cell nuclei in histopathology images. IEEE Transactions on Biomedical Engineering, 57:841852, 2010.

[4] Yao Xue and Nilanjan Ray. Cell Detection in Microscopy Images with Deep Convolutional Neural Network and Compressed Sensing.

[5] Ting Chen and Christophe Chefdhotel. Deep Learning Based Automatic Immune Cell Detection for Immunohistochemistry Images

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector.

[7] https:github.comdatitranraccoon_dataset

[8] https:github.comtensorflowmodelstreemasterresearchobject_detection