Blood Cells Detection Using Faster-RCNN

Sneha Raina
Department of Information Technology
Guru Gobind Singh Indraprastha University
New Delhi, India
rainasneha98@gmail.com

Abha Khandelwal
Department of Information Technology
Guru Gobind Singh Indraprastha University
New Delhi, India
khandelwal.aabha@gmail.com

Saloni Gupta
Department of Information Technology
Guru Gobind Singh Indraprastha University
New Delhi, India
saloni.gupta63@gmail.com

Alka Leekha
Department of Information Technology
Guru Gobind Singh Indraprastha University
New Delhi, India
alka.leekha@bharatividyapeeth.edu

***Abstract*— There has been an increase in the number of health issues in the recent decade with the evolution of various kinds of microorganisms and increasing pollution. The answer to prevent and protect ourselves from the diseases threatening our life lies at the microscopic level in our own blood. Blood consists of erythrocytes (RBCs), lymphocytes (WBCs) and platelets. Object detection is one of the many technologies which can help determine the different constituents of blood. It is a computer technology which deals with object occurrences of pre-defined classes such as cars, humans, buildings, etc. And has far-reaching applications such as self-driven cars, face recognition, face detection, etc. For object detection to be applicable in real-life situations, it should be fast and efficient in determining various entities in an image. There are many object detection techniques developed to make it more efficacious such as CNN and the family of R-CNN. In our research, we apply Faster R-CNN to the Blood Cell Count and Detection dataset to detect RBCs, WBCs, and Platelets. The number, shape and other meta-information or in other words any anomaly in the different constituents of blood can help in early detection of a wide array of issues and/or diseases such as leukemia, lymphoma, anemia, thrombocytopenia, leukopenia, sickle cell disease, etc. Our model detects a bounding box on the three blood constituents with a justifiable accuracy. It identifies all three constituents in the blood smear slide and forms a bounding box around them with their respective labels.**

Keywords— Machine Learning, Object Detection, Convolutional Neural Networks, Faster-RCNN, BCCD Dataset

I. INTRODUCTION

Since the concept of object detection came to light, it has come a long way with constant development and improvement in the technique of detecting objects. Object detection techniques generally fall in the category of either machine learning or deep learning. In this paper, we have reviewed different deep learning approaches such as Region-based Convolutional Neural Network (R-CNN) [1], Fast R-CNN [2], Faster R-CNN [3], etc. With each new technique, we are closer to achieving real-time rates. Object detection has applications in various fields such as face detection, video surveillance, etc. We have also demonstrated the use of object detection to detect and identify three components of blood viz White Blood Cells (WBCs), Platelets and Red Blood Cells (RBCs) in the blood cell images using Faster-RCNN. The count of RBCs, WBCs and Platelets is detrimental in determining various health issues [4].

II. LITERATURE SURVEY

Object detection is a rapidly emerging field with an exponential scope for research which has been made possible due to the groundwork laid by numerous experts.

Object detection is performed extensively with the help of deep Convolutional Neural Networks (CNN) [5]. Deep CNNs are the regularized versions of multilayer perceptrons. Layered architecture is an important aspect in object detection. CNN is a type of network where the information moves from input nodes to output nodes, i.e., only in a forward direction. For object detection using CNN, feature maps are generated. To get feature maps, images are convoluted with an activation function. The pooling layers are used to treat feature maps to get abstracted feature maps which reduces the network's spatial complexity. The process is repeated for the multiple number of filters and the feature maps are created accordingly. Finally, to get output of image recognition, fully connected layers are used to process the feature maps. The output also carries a certainty percentage for the class label predicted.

Traditionally, the approach to detecting objects was to first train object detectors on sub images and then apply them across all scales and locations exhaustively. However, this approach increased computational complexity and became more difficult with the increase in the number of classes as a separate detector was trained per class. To tackle this problem Dumitru Erhan et al. [6] proposed 'DeepMultiBox'- a detector that can be trained in a class-agonistic manner to produce a meager number of object nominees represented as bounding boxes. Each box contains a confidence score which corresponds to the probability of the box containing an object of interest. This model is capable of handling multiple numbers of the same object in an image and enables generalization across classes at the highest levels of the network, something which previous models could not do.

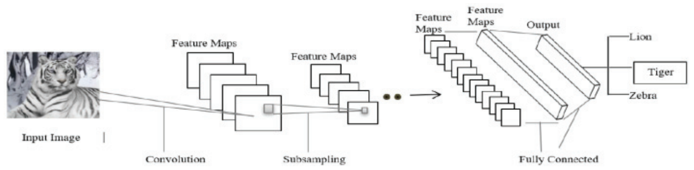


Fig. 1. Use of CNN for object detection

Ross Girshick et al. [1] introduced Regional Convolutional Neural Network(R-CNN). There was an attempt to align two of the burning topics in computer vision, image classification and object detection. The focus was on two tasks, one using a deep Network to localize objects and the second using only a small quantity of human-annotated dataset to train a high capacity model. The detection problem requires to delimitate objects inside an image. Formulating localization as a regression problem is ineffective. A substitute is to frame a sliding window detector that uses CNNs to solve this problem but CNNs with two convolutional and pooling layers are not able to maintain high resolution and CNNs with five convolution layers have a very wide receptive field in the input image which challenges the task of precise localization. The issue of CNN localization is solved by working with the model of "Recognition using Regions."

The second challenge facing detection is the insufficient classified data available to train a colossal CNN. The conventional approach to solving this would be to allow pre-training to act on information without any instruction and follow it up with supervised fine-tuning. Ross Girshick et al. [1] suggested that a supervised pre-training on a wide selection of auxiliary data superseded with fine-tuning on a smaller data set specific to a given domain is an optimal way to train high-capacity CNNs if the current data is not readily available.

But R-CNN has several shortcomings, like it took about 40-50s for prediction per image which wasn't feasible for real life application. Ross Girshick [2] proposed a Fast Region-Based Convolutional Network (Fast R-CNN) that uses a deep neural network to perform the processing of several candidate object locations and the refinement of their spatial locations to achieve precise localization of objects in a single stage instead of the traditional multi-stage approaches that are too slow.

This proposed training algorithm fixed many of the shortcomings of R-CNN [7] while being comparatively faster and more accurate. Fast R-CNN has a more eminent quality of detection mean Average Precision (mAP) and does not need any disk storage for caching of features. Training is done in a single stage using a multi-task loss and updates all the layers of the network.

The fast R-CNN takes the unprocessed picture as input and processes it using several max pooling and convolutional (conv) layers to output a feature map. From this map, a pooling layer of Region of Interest (RoI) extracts a fixed-length function vector for each object proposal. Every one of these vectors are fed in a sequence into several fully connected layers. Lastly, these layers divide into two output layers. One branch produces a probability estimate over K object classes and a background class while the other layer gives an output of 4 real numbers for each of those K object classes. These values contain the encoded information about the position of the bounding boxes.

Fast R-CNN is a faster, more accurate and more efficient successor to R-CNN and Spatial Pyramid Pooling (SPPnet). Because of Fast R-CNN, it was possible to perform experiments and find out that the detectors quality improves when sparse object proposals are used. Further development of technologies that work well with dense proposals as well as with sparse boxes will expediate object detection.

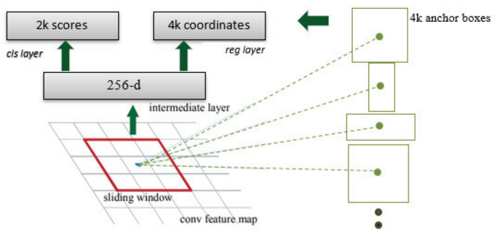


Fig. 2. Region Proposal Network

State-of-the-art networks for detection of objects make use of algorithms for proposal of region to hypothesize the locations of the objects which has proved to be a bottleneck as exposed by the reduced running time in advance like Fast R-CNN. Shaoqing Ren et al. [3] introduced Faster R-CNN which bring together Region Proposal Network (RPN) with Fast R-CNN to attain approximate real-time rates. RPN allows for area proposals that are almost cost-free by sharing full-image convolution features with the network detection. An RPN can be defined as a fully convolutional network which predicts objectness score and object bounds at each position simultaneously. End-to-end training is done on RPNs for the generation of region proposals of high quality.

The use of convolutional feature maps, which are used by region-based detectors, has been proposed for the formation of region proposals. Two additional convolutional layers are added for constructing RPNs: i) by which convolutional map positions are encoded into a short feature vector, and ii) by which regressed bounds and objectness score for k region proposals is generated as outputs at each convolutional map position. RPN is a kind of Fully Convolutional Network (FCN).

Since computation is shared with Fast R-CNN, it is assumed that a common set of convolutional layers are shared between both Fast R-CNN and RPN. The input to an RPN is an image of any size while the output is object proposals of rectangular shape with objectness score each. A sliding window is used to analyze the image. Each window is mapped to vector which is then supplied to a box-classification layer (cls) and a box-regression layer (reg).

RPN generates region proposals efficiently and accurately. A RoI pooling layer is used on the region proposals predicted, which is then used for the classification of the object within the proposed region and the bounding box offset values prediction. RPN improves the quality of region proposal and the accuracy of object detection which makes Faster R-CNN much more efficient than its predecessors. After detecting an object in an image or a video, the next step is to associate each pixel to a class label. This process is called Semantic Segmentation [8]. The labels could include an orange, truck, notebook, chair, etc. just to mention a few. We can describe semantic segmentation as an image classification problem at a pixel level.

Since Fully Convolutional Neural Networks (FCNs) have come into the picture, the accuracy for semantic segmentation has increased. But these improvements work towards the accurate prediction of a category label for every pixel and are unable to understand the relation between two neighboring pixels. Accurate semantic segmentation is the aim but it should not compromise with the speed. One of the reasons for slower speed is that it is built from CNNs and the model fails to take into account the vast quantity of training data and hence becomes a speed breaker for accuracy.

Jifeng Dai et al. [9] proposed semantic instance-aware semantic segmentation built completely on a CNN architecture in which the task was delegated into 3 sub-tasks. The first task is differentiating instances which means representing instances using bounding boxes that are unaware of the class of the foreground object. The second task is predicting masks which means estimating a 2D image whose pixel values can be used to filter each instance. And the third task is to categorize objects which means estimating a class-wise label for every mask. Each subtask is achieved through the CNN.

In the light of the above 3 subtasks, Multi-task Network Cascades (MNCs) were proposed for precise but at the same time fast, instance aware semantic segmentation. There are three stages in the network cascades each one of them addressing one subtask. The three stages have shared features which are traditional when learning from multiple tasks. Feature sharing allows to trim the test time computation, and can also raise feature learning because of basic commonality among the tasks. The proposed model is called a multitask cascade because an earlier stage is passed as an input to a later stage which forms a casual cascade. Because of casual relations among various outputs, training of a multi-task cascade is difficult. It makes the task of computing valid back-propagation tricky which in turn halts the computation of gradient terms. To solve this, a layer was built which is rotationally symmetric in terms of spatial coordinates. The proposed model was trained in an end-to-end fashion with just a single step structure. The training algorithm for a single step model generates convolutional features that are accessed by the three subtasks which leads to an uptick in precision and speed. The method was tested on the PASCAL Visual Object Classes (VOC) data set. It bagged 63.5% mean average Precision which was approximately three percent higher than the previous best results. The result was captured at the test-time speed of 360ms per input image which is two degrees quicker than the earlier systems.

Next, the need arose for correctly identifying a detection design that produces the optimal speed/accuracy/memory equilibrium for a given platform and purpose. Jonathan Huang et al. [7] deployed computer vision models in real-time with an emphasis on memory usage and running time.

A feature-for-feature comparison is inconclusive because of various classes of base feature extractors (e.g., Residual Networks, Visual Geometric Group (VGG), AlexNet), the difference in the size of the training image resolutions, and different software and hardware configurations. It is also challenging for practitioners to choose from various available architectures because Standard measures, like mean average precision (mAP), do not give an all-encompassing answer. For instance, mobile phones require some amount of main memory to use or reference and self-driving vehicles need instantaneous performance. Production systems on Server-side, in the likes used by Facebook, Snapchat or Google have more margin for optimization but still, have to work without compromising the throughput.

For the applications which require high speed and low memory, Jonathan Huang et al. [7] proposed a detector that attains impressive speeds and can be utilized on mobile devices. A detector was also proposed that accomplishes Avant grade outcomes for the Common Objects in Context (CoCo) detection task for applications where accuracy is most important.

Using a lesser region scheme for Faster Region-CNN increases speed without compromising the accuracy, which brings it at par with its faster variants, Single Shot Detector (SSD) [10] and Region-based Fully Convolutional Network (RFCN) [11]. It was also established that Single Shot Detector's utility is less proportional to the standards of feature extractor as compared to faster R-CNN and RFCN.

III. Applying Faster-RCNN to BCCD dataset

A. *Working of Faster-RCNN*

Three neural networks incorporate Faster R-CNN namely, Feature Network, Detection Network and Region Proposal Network (RPN). Feature network generates feature maps of the input image. The shape and structure of the input image is maintained by the output of this network. RPN has three convolutional layers. There is one common layer feeding into bounding box regression and classification. RPN generates Region of Interests (RoI) in the form of bounding boxes. The probability of the objects being in these bounding boxes is high. Along with the bounding boxes, a value ((1) - object in bounding box, (0) - object not in bounding box and (-1) - the box can be neglected) is generated. Around 2k region proposals are generated by RPN and top N proposals are used during the testing phase. Both the Feature Network and RPN provides input to Detection Network whose function is generation of the final class and bounding boxes. Four layers which are fully connected comprises the Detection Network. The bounding box regression layer and the classification layer share 2 common layers that are stacked together. The features in the Detection Network are cropped to help classify the bounding box inside. This cropping is done in accordance with the bounding boxes.

B. *Blood Cell Count and Detection Dataset*

The dataset [12] we used for our experiment was comprised of blood cell smear images. Blood Cell Count and Detection (BCCD) Dataset is a small-scale dataset for detection of blood cells.

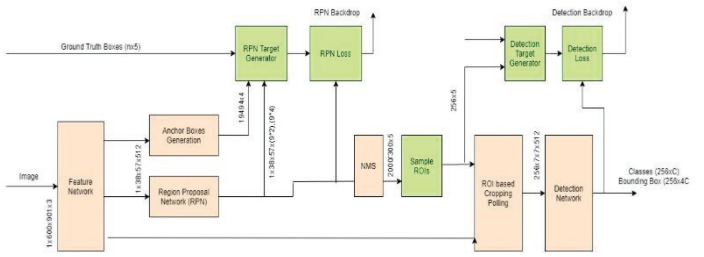


Fig. 3. Block Diagram for Faster RCNN

The original dataset has been re-organized into Pascal VOC (Visual Object Classes) [13] format. The dataset has been licensed under MIT. There are three labels in the Dataset:- Platelets, RBC (Red Blood Cell) and WBC (White Blood Cell). We split the dataset to be used as training and testing sets with 205 images in training and 72 images in testing. Training set consisted of 2806 labels.

C. *Detection Using Keras Implementaion.*

We use the Keras implementation of Faster R-CNN for detecting cellular portion of blood using blood smear slide images. Keras is written in python and runs on top of Tensorflow or Theano backend.[14]

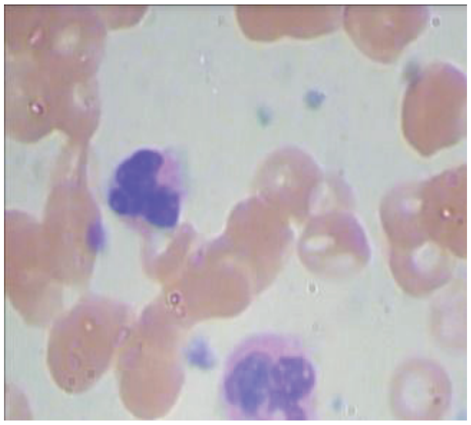
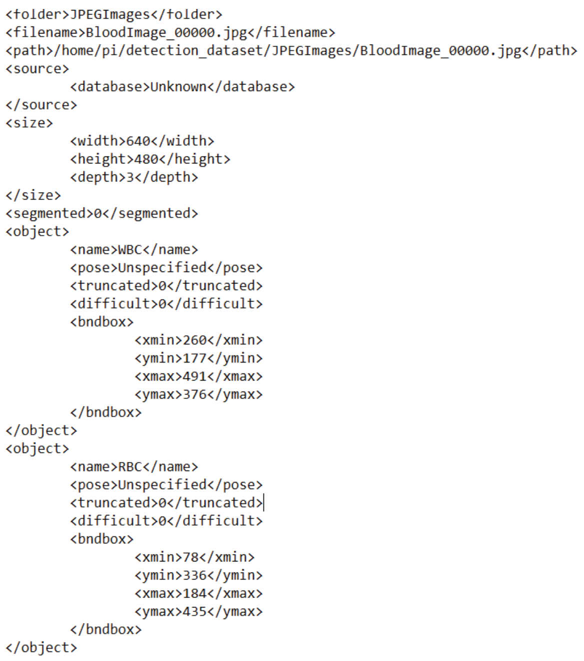


Fig. 4. Sample Image from BCCD Dataset



```
<folder>JPEGImages</folder>
<filename>BloodImage_00000.jpg</filename>
<path>/home/pi/detection_dataset/JPEGImages/BloodImage_00000.jpg</path>
<source>
        <database>Unknown</database>
</source>
<size>
        <width>640</width>
        <height>480</height>
        <depth>3</depth>
</size>
<segmented>0</segmented>
<object>
        <name>WBC</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
                <xmin>260</xmin>
                <ymin>177</ymin>
                <xmax>491</xmax>
                <ymax>376</ymax>
        </bndbox>
</object>
<object>
        <name>RBC</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
                <xmin>78</xmin>
                <ymin>336</ymin>
                <xmax>184</xmax>
                <ymax>435</ymax>
        </bndbox>
</object>
```

Fig. 5. Sample from xml file

The xml annotations were converted to combined csv file due to the flexibility offered by pandas for csv.

We used Keras version 2.2.4 with Tensorflow version 1.15.0. The Keras implementation of Faster R-CNN was cloned from GitHub repository [15]. train_frcnn.py was used to train a model. On successful training, train_frcnn.py created an hdf5 file on which the training weights were saved along with the setting of the training run as a pickle file. On running test_frcnn.py these settings were loaded and the test images passed to the model were used to perform inference from pretrained weights and a config file. The image below shows how our model works. It has constructed bounding boxes around different elements in the image with a label indicating its class.

IV. Results

Test folder passed to the model contained 72 images. The model predicted boundary boxes for three classes of cell type in blood, Platelets, RBC (Red Blood Cell) and WBC (White Blood Cell). We manually counted the number of RBCs, WBCs and Platelets in the test images and compared it to the number predicted by the model. The RBCs were predicted with 55.83% accuracy. It is because, as is evident from the above images, the red blood cells are in clusters and hence it is very difficult to put tangible boundaries on each individual erythrocyte. The WBCs were predicted with 92.10% accuracy. WBCs appear as large blots on blood smear slides and also, they are larger than Erythrocytes. Hence WBC were predicted with higher accuracy. Platelets were predicted with 68.36% accuracy. The reason for low accuracy is because platelets are faint small bright spots on a slide and are hence difficult to predict. The model was trained for 1000 epochs. Increasing the epoch size will increase the overall accuracy for all cell types.

V. Conclusion

Object Detection is one of the technologies guiding human civilization towards a better and brighter future with its many uses in the realm of image processing and computer vision. Since the introduction of the concept of object detection, many approaches have been developed to detect the object more efficiently and accurately in a video or image.

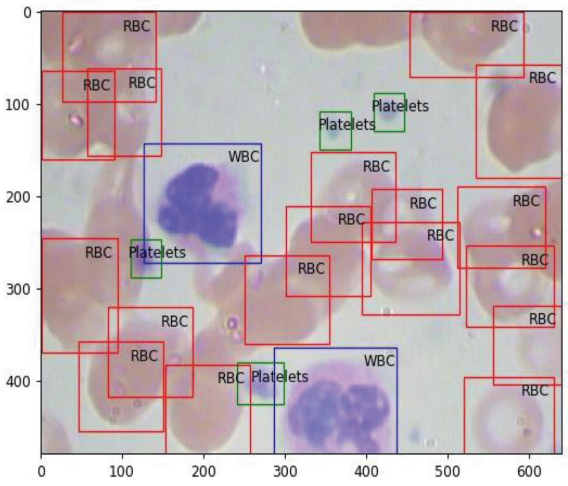


Fig. 6. Sample image after prediction

We implemented Faster R-CNN on blood cells to predict the formed elements namely Erythrocytes, leukocytes, and platelets. The implementation of Faster R-CNN on blood cell images shows that it achieves greater accuracy in detecting the boundary boxes for WBCs than RBCs which is due to number of reasons such as the size of the WBC is greater than that of the RBC, the RBCs are present in clusters in the blood which makes it difficult to get precise tangible images in blood slides. The overall accuracy of the model can be increased by increasing the number of epochs during training.

VI. FUTURE SCOPE

Although our model was successful in identifying bounding boxes around the blood smear slide images, the accuracy around the clusters of red blood cells was depressed. It can be improved in various ways which include but are not limited to increasing the number of epochs, experimenting with the layers in convolutional network etc. However, there is a conundrum to that method. By increasing the number of epochs, we can encounter over-fitting which will reduce limit accuracy. A better solution is to use advanced object detection models such as Region-based Fully Convolutional Network (R-FCN) [11] and Mask R-CNN [16].

Jifeng Dai et al. [11] discussed the use of R-FCN over the use of Fast R-CNN and Faster R-CNN to achieve more accuracy and faster output. Instead of applying a costly subnetwork on each region a hundred times, the use of position-sensitive scored maps was introduced to cope with the quandary between interpretation-invariance in classification of images and interpretation-variance in discovery of objects. These maps are crucial because the aforementioned dilemma causes an unnatural design in classification networks like Resnet which improves accuracy but at the expense of reduced speed.

A position-sensitive pooling layer of Region of Interest (RoI) guides useful data from these maps without any weight layers following. The layers are shared throughout the image but the spatial information needed for object detection is still encoded.

R-FCN is based on Resnet and has features of both RPN and R-FCN. It works in two stages- proposal and classification of regions. Region based approach is taken for greater accuracy. RPN is used in the first stage of region proposal to identify candidate regions RoIs and then R-FCN classifies the RoIs into background and object categories.

Experiments performed using R-FCN show that it's mAP (76.6%) is competitive with that of Faster R-CNN (76.4%) thus indicating that relevant spatial information for object locating was successfully encoded without the need of a learnable layer after pooling of RoI. R-FCN is a simple yet efficient framework whose accuracy is at par with Faster R-CNN, but is much faster during training and inference. It can also be applied with other region-based methods like selective Source (SS) and Edge Boxes (EB).

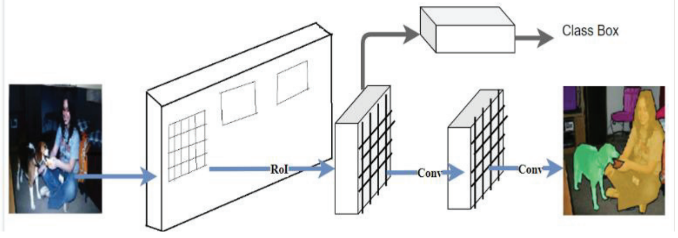


Fig. 7. The Mask R-CNN framework for segmentation of instances.[17]

Mask R-CNN is the prolongation of Faster R-CNN. Much like Faster R-CNN, Mask R-CNN works at two levels: the first level is RPN which proposes the bounding boxes for objects, the second level introduces an object mask which predicts masks for segmentation for each Region of Interest (RoI) and predicting box offset and classification alongside. This contrasts with most recent models, where mask predictions are used for classification.

Mask R-CNN uses pixel-to-pixel alignment, unlike Faster R-CNN, between input and output. It is evident in how spatial quantization is coarsely performed by RoI Pool for feature extraction. It uses RoIAlign for fixing the misalignment, which is a layer free of quantization. The exact spatial locations are preserved by RoIAlign which improves the accuracy of the mask by 10% to 50% relatively. To decouple class and mask prediction, a binary mask is predicted for each class independently and RoI classification is used to predict the category.

Kaiming He et al. [16] compared Mask R-CNN in instance segmentation to various state-of-the-art models and it has shown superior results compared to those models, including the winners of the COCO 2015 segmentation challenge which is MNC and the COCO 2016 segmentation challenge which is Fully Convolutional Instance Segmentation (FCIS). Mask R-CNN, in addition to segmentation of instances, shows top results for estimation of human pose and bounding-box object detection as well. We believe that these models would definitely take up the accuracy of our model around clusters.

REFERENCES

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv:1311.2524, 2014.

[2] Ross Girshick. Fast R-CNN. arXiv:1504.08083, 2015.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In NIPS, 2015.

[4] "American Society of Hematology." hematology.org. https://www.hematology.org/Patients/Basics/ (accessed Nov. 2019).

[5] Ajeet Ram Pathak, Manjusha Pandey, and Siddharth Rautaray. Application of Deep Learning for Object Detection. Procedia Computer Science, Volume 132, Pages 1706-1717, 2018.

[6] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable Object Detection using Deep Neural Networks. arXiv:1312.2249, 2013.

[7] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In CPVR, 2017.

[8] D. Mwiti "A 2019 Guide to Semantic Segmentation." heartbeat.fritz.ai. https://heartbeat.fritz.ai/a-2019-guide-to-semantic-segmentation-ca8242f5a7fc (accessed Nov. 2019)

[9] Jifeng Dai, Kaiming He, and Jian Sun. Instance-Aware Semantic Segmentation via Multi-task Network Cascades. arXiv:1512.04412, 2015.

[10] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In ECCV, 2016.

[11] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. arXiv:1605.06409, 2016.

[12] Shenggan, N. Chen. "BCCD Dataset. V1." February 24, 2018. Distributed by MIT. https://github.com/Shenggan/BCCD_Dataset

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, 2007.

[14] "Keras: The Python Deep Learning library." keras.io. https://keras.io/ (accessed Nov. 2019).

[15] K. Bardool. "keras-frcnn." gitHub.com. https://github.com/kbardool/keras-frcnn (accessed Nov. 2019)

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In ICCV, 2017.

[17] "COCO Dataset." cocodataset.org. http://cocodataset.org/#explore?id=239536 http://farm3.staticflickr.com/2373/2421365812_cda8476bb4_z.jpg (accessed Nov. 2019).