Blood Cell Count and Detection Method Based on YOLO

Baizhen Liu

College of Art and Science, New York University, New York, NY, 11101, The U.S,

Email: bl2842@nyu.edu

Abstract. Blood Cell Count and Detection (BCCD) has always been a popular topic in object detection and many researchers have applied and modified the two basic models: Faster RCNN and Yolo. However, it is still difficult to tell which model or modification would perform better on other BCCD datasets. Thus, this paper mainly focuses on finding a better model and modifications to BCCD example datasets containing 364 images of blood cells. Faster RCNN and Yolo v5 were used as the basic two models for the dataset. Through training and comparisons between the two models, the better model was chosen to make further modifications or adjustments to achieve a better maP result possible. The result shows that in this specific dataset, Yolo v5 performs better. The modified Yolo v5 model also has an improvement of 0.6 percent of map 0.5 and 0.5 percent of map 0.95 comparing to the original model, showing that modification of model configuration, model structures including head and backbone would efficiently improve the time taken for training and maP.

Keywords: Blood Cell Count and Detection (BCCD), Faster RCNN.

1. Introduction

Object detection is a very popular topic in Computer Science. As the accuracy and efficiency are both increasing, object detection is now applied to a lot of areas for example Auto-pilot in vehicles. Object detection is also widely used in the medical field for making diagnoses and detections for diseases like tumors. With the help of Object detection, a higher efficiency could be achieved.
The most two common models applied to medical object detection are Yolo and Faster RCNN. These two models, work differently and reflect differently on their results in different areas of detection.

Blood cell counting and detection is one of the most crucial tasks in medical object detection. Blood cells including Red Blood cells, White Blood cells, and Platelets detection is always an interesting and important study in medical detection. The result of the detection is a crucial part of diagnosing a various of diseases. Much research about Blood Cell Count and Detection (BCCD) has been done in the recent years and different models have been applied. For example, BCCD based on Faster RCNN by Richard Jiang and his team achieved a high precision of 98.3 percent with a 1.3 miss rate in 2019. BCCD based on Yolo v3 by Zhangzhi Yuan and his team achieved a high maP of 0.943 in 2021. Although the result of precision of Faster RCNN seems to be higher than Yolo, the time spent on detection of Faster RCNN is longer than Yolo. The comparison between Yolo v5 and Faster RCNN also have been done by Stanford University students by Ehsan Adeli and Fei Fei Li and their team based on a 194 images dataset, the results show that the Yolo v5 was outperforming Faster RCNN. However, the dataset is small so the performance of Faster RCNN and Yolo v5 is probably not so representative.

It is still unclear which one of these models works better on other BCCD datasets and how to adjust the model to fit better in order to get the best results.

This paper mainly focuses on finding the best model and fine-tuning a BCCD example dataset. The paper would use both Faster RCNN and Yolo v5 models in order to find the best of these two for this specific dataset. Then the better model would be fine-tuned to achieve the best results possible. The training data, validate data, test data and training cycles would be held the same in order to make comparison.

2. Methods

2.1 Dataset

The BCCD dataset used was primarily from Roboflow. This data was originally sourced by cosmicad and akshaylambda. The raw data contains 364 images and there are 3 classes: RBC, WBC, and Platelets. Each image contains bounding boxes for every three classes cells.

2.2 Faster RCNN

Faster RCNN is a model improved from Fast RCNN by Microsoft. Faster RCNN is consist of two networks: RPN (Region proposal network) for generating proposals of the regions and another network is for using these generated proposals to perform object detection.

Instead of using selective search used in Fast RCNN and RCNN, Faster RCNN uses RPN which is much less time-costing. RPN also shares the most computations with the object detection layer.

Anchor is also a very important key in the Faster RCNN. For each position of the images there would be 9 anchor boxes. These anchor box are defined to catch the scale and aspect ratio of the detected.

Region of interest pooling is also a very important part of the Faster RCNN. After RPN, proposed regions of different size makes it hard to make an efficient stricture to work on them. The ROI pooling reduces the feature map into the same size, which makes it much easier.

Detectron 2 is used as the basic platform of Faster RCNN in this paper. Detectron 2 is developed by Facebook and it can be easily utilized for research-first use cases and production-oriented use cases. Since the library of the Detectron 2 is built in Pytorch, the new models could also be easily implemented.

2.3 Yolo V5

Yolo was considered to be the first model to combine the steps of Bounding box estimation and object identification in one differentiable network which is end-to-end. Yolo is written and maintained under an environment called Darknet. Compared to the previous versions of Yolo, Yolo v5 firstly introduced the Focus layer which makes Yolo v5 more lightweight. The Focus layer reduces layers, parameters, FLOPS, and CUDA memory, and increase forward and backward speed while leaving almost no changes to the maP.

Yolo v5 was developed by Glenn Jocher from Ultralytics using Pytorch framework. It divides the image into a grid system (different regions) and each grid would do detections itself. Yolo v5 model contains 4 sections including input, backbone, neck, head, and a two-stage object detector.

The backbone is a section which works as a feature extractor. In this section, CSP(Cross-Stage-Partial-Network) was used. CSP improves the procession time by huge amount.

The neck is a section used for generating feature pyramid. Feature pyramid is often used to help the model to perform well when the data is unseen. In Yolo v5, PAnet is used for generating feature pyramids.

The head section is in charge of the final detection. Anchor boxes were applied on the features and the final output was generated with the probabilities of the classes, scores, and bounding boxes.

Yolo v5 also has three model configurations: small, medium, and large with different depth and width multiples.Larger depth and width require larger calculations therefore a longer time of processing time.

3. Experiment

3.1 Dataset Preparation

Data splitting was first being splitted to 300 images for training data, 50 for validation and 14 for testing. The raw data is considered to be a rather small dataset for the model, it might not reflect fully

the performance of each model. The validation images and testing images could not be augmented. However, the training images could be augmented to 600 images using Roboflow's data augmentation function. The augmentation of training data would give a more comprehensive learning process to the model. Therefore, the final dataset being put into the models is 600 images for training, 50 for validation and 14 for testing.

3.2 Faster RCNN versus Yolo v5

Experiment 1 compares the time taken and the precision of using Faster RCNN and Yolo v5. Dataset and epochs of training would be held unchanged for comparison.

After downloading Detectron 2 from GitHub and all the python packages that it requires, the modified BCCD dataset is downloaded from Roboflow (The dataset also needs to be registered). Then the Coco trainer model and the Faster_rcnn_R_50_FPN_1x are imported from Detectron 2 for the training process. The pretraining weight of the Faster RCNN is from the COCO dataset. Because the training dataset consist of 600 images and the batch is equal to 4, there are 150 iterations for 1 epoch. Then 15000 is set for the maximum iteration. The class number is 4 which is 3 classes plus 1 class for background. [https://github.com/facebookresearch/detectron2]

For Yolov5, considering the dataset used was rather small, Yolov5s configuration with a depth of 0.33 and a width of 0.50 was used in the comparison between Faster RCNN and Yolo v5. The Yolo v5 repository was copied from GitHub by ultralytics. After downloading the dataset into the Yolo v5 folder, the number of classes was defined based on the yaml file of the Yolo v5. In the configuration, the number of NC was changed from the original 80 to 3 classes to fit the dataset. All the other parts of the model remained unchanged. (batch = 16) Then for the training part, Yolov5 was trained on custom data for 100 epochs. [https://github.com/ultralytics/yolov5]

After comparing the maP 0.5 and the time taken for both models, Yolo v5 is better in both maP and time taken for this specific dataset. Therefore the next experiment would focus on improving the performance of Yolo v5.

3.3 Different Yolo v5 model configurations:

As the first experiment has shown, Yolo v5 performs better in object detection of the BCCD dataset not only in processing time but also in the accuracy and precision. So the second experiment mainly focused on selecting a better model configuration of the Yolo v5 from small(s) , medium(m) , large(l) , and extra large(x) which could perform a better precision while leave almost no change to the processing time.

In this experiment, the number of epochs and the other section of the model except for depth and width was remained unchanged. Small, medium, large ,and extra large model configurations were used on the same dataset.

The time taken for performing is simply larger when the depth and width number get larger but it didn't varies that much between each model configuration. Then maP is the focus of the comparison. By comparison between the small, medium, large, and extra large model configuration, the maP values from the highest to the lowest are large, extra large, small, and medium.

Then it is obvious that the large Yolo v5 model configuration is the most suitable model for this specific BCCD dataset, the next focus would be the possible modifications to the head and backbone of the Yolo v5.

3.4 Modification of model structure

The first modification done was to the head of the Yolo v5 structure. As the head is the part that is in charge of the input of the model, the modifications need to be done on which layers do the model choose to extract features. The original layers chosen are [17,20,23]. There are 1,3,5,7,9,13,17,20, and 23 to be chosen. Four different combinations including [1,3,5] , [3,5,7],[5,7,9],[7,9,13],[9,13,17], [13,17,20] were chosen to be studied.

The second modification was based on the result of the previous modified model, the next model would choose the modified head with the best results of maP. The number of the depth of the layers

in the backbone would be modified to achieve the best maP possible. The arrangement of the number of the backbone usually follows an ascending then descending order. The original backbone has the numbers of depth arrangement of [3,6,9,3], following the usual rule of the backbone, the modifications tried were [2,6,9,2], [1,6,9,1], [2,3,9,2],[2,9,9,2],[2,6,6,2],[2,3,6,2], and [2,9,12,2].

4. Result

Table 1: Faster RCNN vs Yolo v5

	maP 0.5	maP 0.5 ~ 0.95	Total training taken(hours)
FasterRCNN_R_50_FPN_1x	0.890	0.643	1.493
Yolo v5_s	0.932	0.627	0.279

For comparing Faster RCNN and Yolo v5, it is obvious and common that the time taken for training of Yolo v5 is much shorter than Faster RCNN. Commonly, Faster RCNN has a longer training time but a higher precision than Yolo v5, however, for this dataset, Faster RCNN is not as precise as Yolo v5 in maP 0.5 and maP 0.5 ~ 0.95.

Table 2: Different Model configurations

Different model configurations	maP 0.5	maP 0.5 ~0.95	Total training taken(hours)
Yolo v5_s	0.932	0.627	0.279
Yolo v5_m	0.927	0.621	0.317
Yolo v5_l	0.934	0.626	0.397
Yolo v5_x	0.933	0.627	1.095

Then for the comparison between the four different size model configurations of Yolo v5, the result is not quite following the common rules. For the time taken for training, the larger the model configuration is (another word, depth and width), the longer time it takes for training. Model configuration x is discarded because its training time is too long that it is even slower then Faster RCNN, and its precision numbers are not the best of the four model configurations. Model configuration m is also discarded because it is not even as precise as the small model configuration. Then, the two remaining model configuration is left for choosing. As the main purpose is to choose a model with a higher maP 0.5 and maP 0.5~ 0.95, model configuration is chosen at last for fining tuning and model structure modification.

Table 3: Modification on head of the model

Modification on head Yolo_v5_l	maP 0.5	maP 0.5 ~0.95	Total training taken(hours)
[1,3,5]	0.817	0.513	0.395
[3,5,7]	0.920	0.605	0.346
[5,7,9]	0.922	0.597	0.365
[7,9,13]	0.923	0.596	0.391
[9,13,17]	0.936	0.620	0.496
[13,17,20]	0.935	0.631	0.525
[17,20,23] (origin)	0.934	0.626	0.397

The next part is the modification done on the head of the backbone. By looking at the results, the only modification which outperform the maP of the original structure is the [13,17,20] structure. Although its training time is longer than the original structure, the maP improvement is considered to be more important. In this case, the next modification on the backbone would be based on the structure of [13,17,20].

Table 4 Modification on backbone based on head modification

Modification on backbone based on head [13,17,20]	maP 0.5	Map 0.5 ~ 0.95	Total training taken(hours)
[2,6,9,2]	0.94	0.628	0.403
[1,6,9,1]	0.934	0.631	0.388
[2,3,9,2]	0.938	0.632	0.389
[2,9,9,2]	0.931	0.619	0,418
[2,6,6,2]	0.933	0.636	0.390
[2,3,6,2]	0.94	0.627	0.373
[2,9,12,2]	0.935	0.624	0.434
[3,6,9,3] (origin)	0.935	0.631	0.525

The last part of the modification of the backbone structure is also successful. By looking at the result, [2,3,9,2] is one of the modifications which achieves both a higher maP and a shorter training time than the original structure of the backbone. Although some of the other results achieve either a higher maP 0.5 or maP 0.5 ~0.95 or a shorter training time, they are making such a general improvement as [2,3,9,2] structure.

In general, the modifications of the Yolo v5 improved maP 0.5 by 0.6 percent and map 0.95 by 0.5 percent.

5. Discussion

First, considering a comparison between the results of Faster RCNN and Yolo v5, the reason why Faster RCNN is not showing its usual advantage of being more precise than Yolo v5 is possible because the training dataset is small. Training data deficiency is going to cause the risk of overfitting. This could be improved by augmenting the training dataset into a larger number for the Faster RCNN to have a deeper study.

Second, the results of the four model configurations of the Yolo v5 show the depth and width associated with the data size. The middle configuration is not performing well possibly because the depth and width don't fit the dataset. The best depth and width range is between middle and small and middle and large because the small and large model configuration is performing better. The middle model configuration could be improved by modifying the dataset size or the depth and width.

Third, the results of the head modification show that extracting features of the model for the small dataset is better from rather lower layers. But extracting from layers that are too low may cause the precision to decrease. On the other hand, the results of the backbone structure are reflecting the fact that the original number of models of the backbone structure is too large for this dataset. By modifying three of four model numbers,3,6, and 3, lower to 2, 3,2, the precision and time taken are improved. The head modification could be improved by trying more combinations of the layer numbers or having a deeper study of the structure. More fining tuning could be done on the HPY numbers for example learning rate or anchors.

6. Conclusion

Based on research and study, it is found that in this specific BCCD dataset, Yolo v5 performs better by achieving a shorter training time and a higher maP. For model configuration, the large model configuration with a 1.0 and 1.0 depth and width has a higher maP value. Based on this model configuration, extracting features from lower layers 13, 17, and 20 and having a small model number of [2,3,9,2] would achieve a higher maP.

This reflects that with different size datasets, different models like Faster RCNN and Yolo v5 would reflect different performances. Besides, the structure inside the model also needs to be adjusted and modified to better fit the dataset. By selecting models carefully, modifying the structure, and careful adjustments, a higher accurate detection could be achieved in the future.

References

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5). UC Berkeley.

[2] Ross Girshick. 2015. Fast R-CNN. Microsoft Research.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster R-CNN: Towards Real-Time Object Detection with Regional Proposal Networks.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. 2018. Mask R-CNN. Facebook AI Research (FAIR)

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection.

[6] Ally Nakamura, Kachachan Em Chotitamnavee, Roshan Swaroop, Zane Durante, Ehsan Adeli, and Fei Fei Li. 2022. Transfer Learning for the ICU: Comparing the Performance of YOLOv5 and Faster R-CNN X101-FPN. Stanford University.

[7] Tiancheng Xia, Richard Jiang, Yong Qing Fu, and Nanlin Jin. 2019. Automated Blood Cell Detection and Counting via Deep Learning for Microfluidic Point-of-Care Medical Devices.

[8] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. 2018. Path Aggregation Network for Instance Segmentation.

[9] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh. 2019. CSPNet: A New Backbone that can Enhance Learning Capability of CNN.

[10] Md Ershadul Haque, Ashikur Rahman, Iftekhar Junaeid, Samiul UI Hoque, and Manoranjan Paul. 2022. Rice Leaf Disease Classification and Detection Using YOLOv5.