Individual Report

Chao Zhou

Our purpose of the final project is very straight forward which is adapt the object detection algorithm on the red blood cell dataset and fine tune it on this adapted dataset. Since the complexity of the Faster-RCNN model, especially it has multi task losses, we cannot find any pre implemented version built in any frameworks including Keras, Tensorflow and Pytorch. We find one self-implemented Keras code using Tensorflow as backend and decided to use this code as our starting point. Since understanding the neural network in detail took a lot of work, everyone in the group read many papers on this algorithm so we can have idea on how the model works. My responsibility in the team is understanding the whole git repo we cloned and know how the code works so that we can adapt this code on our dataset.

Since there are really a lot of code in the repo, the author wrote their own resnet50, batch normalization, data generator and so on. Basically the code did not use much of the built in function in Tensorflow framework. I really need to understand how the paper is implemented using that code. At the very beginning, the code does not work since the environment is not correct. I did try different version of Tensorflow, Keras and Cuda to get the correct final version to let the code run on GPU correctly. The code has separate training, testing and evaluating code, and neither of them is working correctly at the very beginning. There are different kinds of error in the code. My task was ensure the algorithm was really running correctly on logic, we feel we cannot depend on the origin work since the code is rarely documented. I read all the different python codes to make sure the origin code is correct. This process took the majority of the time.

Since the code is documented, I have to figure out what the input should look like. And instead of passing path of image to the model directly, we first need to encode the object in the image and its corresponding bounding boxes into a text file in the following format: “filepath, label, x1,x2,y1,y2”. Then when we run the model, it will first parse the text file I encoded then according to its path read all the images and divided the into training and testing data set. Then use pickle file to record how the dataset is divided so that the model can know what part of the image dataset is testing dataset. Since I never encounter this way of dividing data into training and testing, I have to read the code to figure it out. I also fixed one problem in training code so that the model can load pre-trained resnet50 weights and bias correctly which improves the mean average precision a lot. Also, in the testing code, the origin code does not even freeze the network weight when testing which is very incorrect. In the measure_map.py file which is intended to evaluate the result of our model also have bugs in it, I also fixed it to make the code work. I’m also in charge of running the model on AWS cloud. So I have to learn terminal skills to install the right environment to let the code run correctly. After finishing those technical details, I tried using resnet50 both as feature extractor and fine-tuning the resnet50 on its pre-trined weights. I find out the overall mean average precision are very similar. I also tried data augmentation in the code, since we are doing object detection, we not only should augment the origin image but also should augment the bounding boxes. I did flipping and rotation for the the dataset but the score has decreased compared with the data without augmentation for the same other hyper-parameters. I suspect maybe since the augmented data act as regularization so we need to train more epochs on the augmented data. But our computing power is limited, we do not have enough time to train the model since a simple 20 epochs would take 12 hours.

We used a holdout set to test our model, we can see the bounding box is drawn really tightly on each blood cell. We also used mean average precision to evaluate the object detection result since it is the common practice in object detection area. The following figure is

cells	AP
ring	0.4595
difficult	0.9888
trophozoite	0.9640
leukocyte	0.8202
schizont	0.9734
gametocyte	1
mAP = 0.8677	

produced by the measure_map.py file, it tells us how each category performs in the model. We can see it has some good score except the ring class, then we visualize some of the ring image, and feel like even has human being it is very hard to identify the difference between ring and some other classes. Although we do not have any previous professional training experience on blood cell image.
One thing worth-noticing is that the ring class is not the most minority class, which kind of proves our opinion on it is hard to identify this class of images.

The majority of my time contributes to understand the faster-rcnn and make the code work. In this process, I realize that to be able to fine tune a model, we really need to understand how it works at first, then we can have more creative idea to deal with the problem. I also learn we can have creative way to make use of the existing model. Like in the faster-rcnn model, it is essential CNNs + Fully connected layer. We also learned that transfer learning is very important in computer vision tasks, we can really build new work on previous people's work so we can solve variety of problems. In the future, I know plotting loss function is very important for helping us to see how the training process goes. There are so many hyper-parameter in our model, after understanding the model we could pick to tune some important parameters first. And finally, the more data does not necessarily mean better model but a better model definitely need more data!

I did not calculate this score, since the whole code was taken from the GitHub and I then debugged it and added documentation for it.

https://arxiv.org/abs/1506.01497
https://arxiv.org/abs/1512.03385
https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/
https://zhuanlan.zhihu.com/p/31426458
https://www.analyticsvidhya.com/blog/2018/11/implementation-faster-r-cnn-python-object-detection/
https://github.com/rbgirshick/py-faster-rcnn
https://www.kdnuggets.com/2017/10/deep-learning-object-detection-comprehensive-review.htm
https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173l