Glasgow College, UESTC

Interim Report on UESTC4006P(BEng) Final Year Project

Please start by saving this file with the name: GUID_Surname_UESTC4006P_Interim_year
******* Please add appropriate course code***

Student Name	Zilai Wei
Student Matriculation Number	2429587W
UESTC Student Number	2018190504035
Degree programme	Bachelor
Academic year	2022

Placement Company (if appropriate)	
Working Title of Project	Deep person detection in video
Name of First Supervisor	Jin Qi
Name of Second Supervisor	Joao Ponciano
Declaration of Originality and Submission Information	*I affirm that this submission is all my own work in accordance with the University of Glasgow Regulations and the School of Engineering requirements* Signed (Student) : Zilai Wei

Your report should be NO more than 8 pages in length and include the below subject headings and incorporated within this document:

Work done so far including thorough literature review (at least 4 pages)

Conclusions from initial work (at least 1 page)

Work to be done (at least 1 page)

Revised Gantt Chart

Deadlines for submission of this report

Please upload this report via the Moodle page by the deadline mentioned in Table 1 of your project handbook.

Comments from your Second Supervisor will be made via Moodle or via email.

Work done so far including thorough literature review

1. Literature Review

Before I start, I reviewed a lot references and found that Fast R-CNN and YOLO is two models most commonly used in object detection. Fast R-CNN is one of the state-of-the-art models before YOLO appears, and from the name, we can easily get one characterister of this network is fast. Although YOLO makes great process in the next several years, Fast-RCNN still have the accuracy similar to YOLO detection, like the diagram shown below[1]:

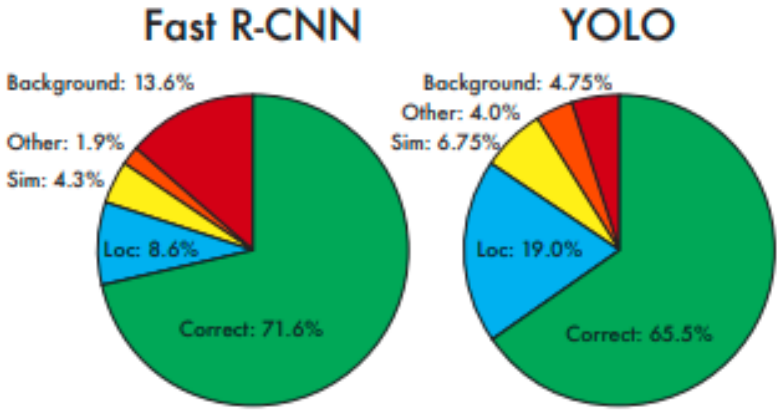

Figure 1. Error Analysis Between Fast R-CNN and YOLO

From Figure 1, we got that Fast R-CNN will have less locolization errors compared with YOLO, but it will easier to make mistakes in background.

YOLO, however, is also an extremely fast multi object detection algorithm and use CNN to detect objects. YOLO does not display the process of obtaining region proposal. Figure 2 shown the architecture for the YOLO network. In Fast R-CNN, although RPN and Fast R-CNN share convolution layer, RPN network and fast R-CNN network need to be trained repeatedly in the process of model training. Compared with the "look twice" (candidate box extraction and classification) of R-CNN series, YOLO only needs to look once, which is also the origin of the name: You Look Only Once[2]. Besides, Yolo deserves more consideration when doing real-time object detection, for Fast R-CNN will have 2-3 seconds delay for predicts.

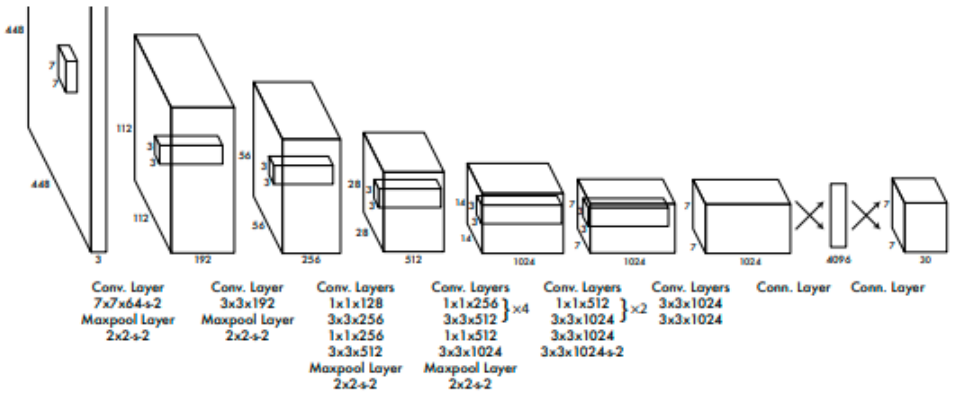

Figure 2. Architecture for YOLO Network

YOLO is also iterating and upgrading, from YOLOv1 to YOLOv5. Compared with the first generation, YOLOv5 purposes an efficient and powerful target detection model. During detector training, the effects of bag of freebies and bag of specialties methods of SOTA are verified. Alos, it improved SOTA methods to make them more effective and more suitable for single GPU training.

In conclusion, YOLOv5 has a very lightweight model size but can achieve the quite high accuracy and efficiency for real-tiem detection. So under depth consideration, YOLOv5 is the most appropriate model for this project.

2. Deep Learning Software and Environment Installation

I installed Anaconda3 at first, it contains more than 180 scientific packages such as Conda and Python and their dependencies. Through Anaconda, I can establish a deep learning environment suitable for target detection, and I built Pytorch and Paddle environment, which version is approate to my computer system and hardware configuration.

After completing the above processes, I installed a software named Pycharm, which is a Python Integrated Development Environment. It is a powerful Python compiler. After that, all code compilation and operation should be carried out on this software.

3. Data Set Collection

After configuring the basic environtment, I need to collect the data set. At first, I searched some public image data sets for face detection from Internet, but I found that in many data sets, all faces tend to be a single pose or expression, and the proportion of faces in photos is very large. However, in our daily lives, the state and expression of the human face are very easy to change. In other words, The low generalization of the model will lead to the low accuracy of the final test.

Therefore, I decided to collect my own data set. I searched all of the folders including photos in my computer, and copied 1000 photos as my data set. These 1000 pictures covered a very wide range, from kindergarten to undergraduate. The characters include me, my family, different teachers, classmates and friends. The state of the face is quite different, which have heads down, heads up, sideways faces and so on. The number of characters in each photo is also different, including single person, multiple people and group photos. In conclusion, different sizes, states faces from different people combined into my data set.

After that, I need to manually mark the faces in these 1000 pictures. It was a huge amount of work, and I used a tool named Labelimg to assist me. I marked all of the faces in a picture, and text the label name with ‘face’, as one example shown in Figure 3. When I marked the face in the picture, this tool can automatically record the four vertex coordinates of the face in my frame and convert them into a TXT file. This project cost me about more than 3 hours.

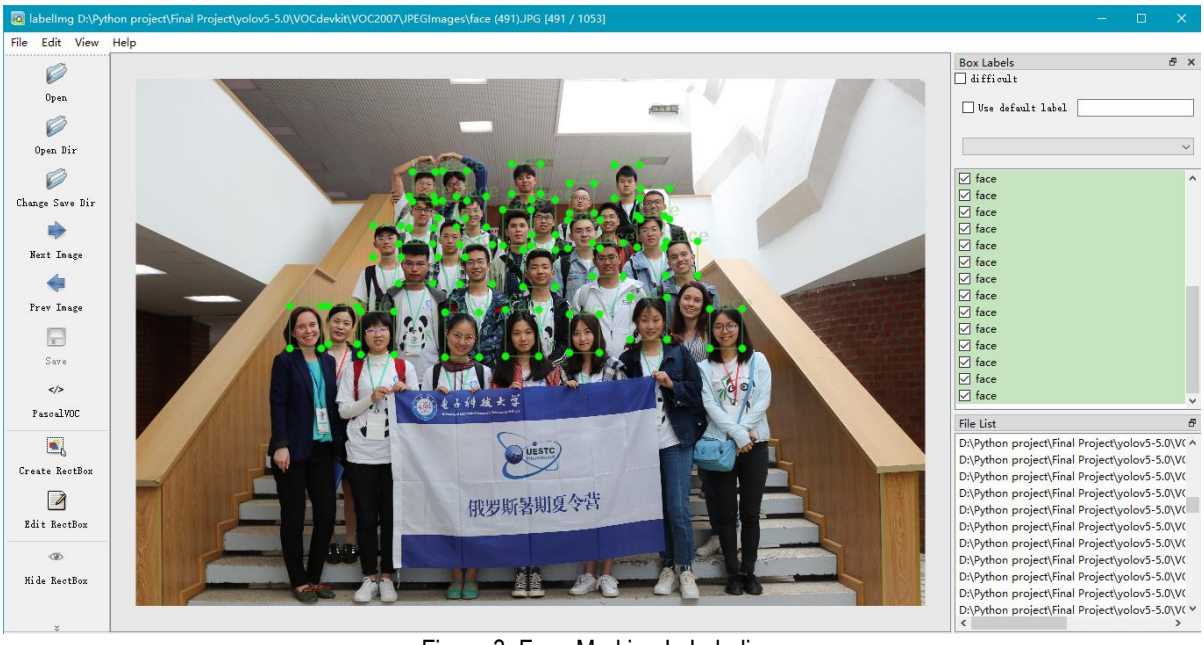

Figure 3. Face Marking In Labelimg

4. Model Training

For model training, I need to set the train set and test set at first. Almost every deep learning projects need the train set and test set. In this project, I only have one class, and I let the system automatically disrupt the previous data set in random, set 80% of them as train set and the rest as test set.

Next I need to decide the specific model in YOLOv5. There are many types of model for YOLOv5 with diferent size and parameters, as shown in Figure 4.

Model	size (pixels)	mAP^{val} 0.5:0.95	mAP^{test} 0.5:0.95	mAP^{val} 0.5	Speed V100 (ms)	params (M)	FLOPS 640 (B)
YOLOv5s6	1280	43.3	43.3	61.9	**4.3**	12.7	17.4
YOLOv5m6	1280	50.5	50.5	68.7	8.4	35.9	52.4
YOLOv5l6	1280	53.4	53.4	71.1	12.3	77.2	117.7
YOLOv5x6	1280	**54.4**	**54.4**	**72.0**	22.4	141.8	222.9
YOLOv5x6 TTA	1280	**55.0**	**55.0**	**72.0**	70.8	-	-

Figure 4. Different models for YOLOv5

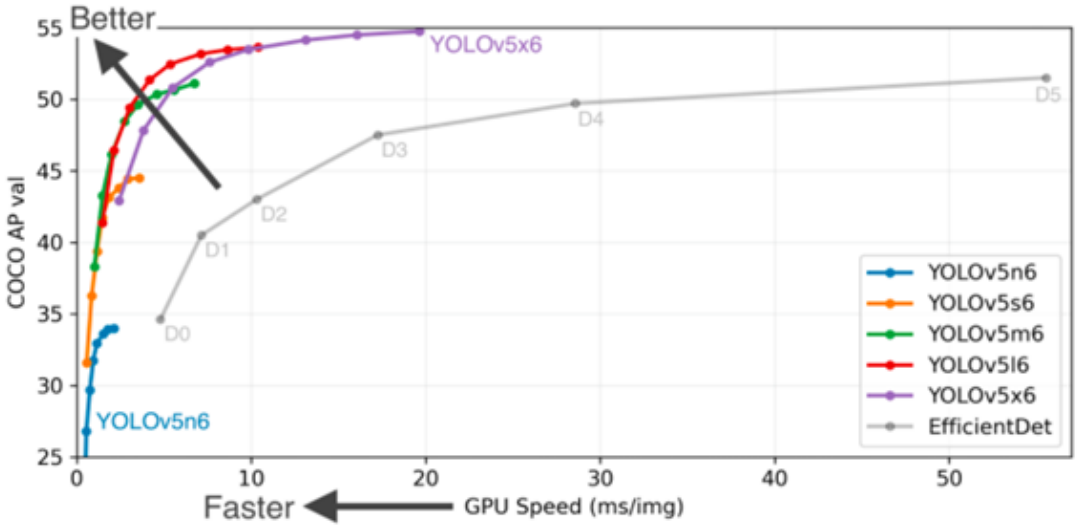

Figure 5. Performance for Different Models In YOLOv5

The Figure 5 compared the performance for the different types of model in YOLOv5. For my project, I only have one class which is ‘face’, and the data set is only about 1000 pictures, which is not a hug set. Therefore, I chose the smallest size model with great GPU speed and performance, the YOLOv5s. However, the version of the model is updated continuously, when I wrote this article, it has been updated toYOLOv5s6, as shown in Figure 4 and 5. Although the same type model with different versions are similar, YOLOv5s5 is the was the latest version when I was training.

Then I start to train the model. However, If I don't make any settings, only let the computer take a start from the head and find the weight value suitable for human faces, the model may train for more than 1000 epochs, which means a lot of time. So I will give the computer a pre weight value from github, and can make the training more efficient and accuracy. I let the epochs trained for 100 epochs, and the final training results is shown in Figure 6.

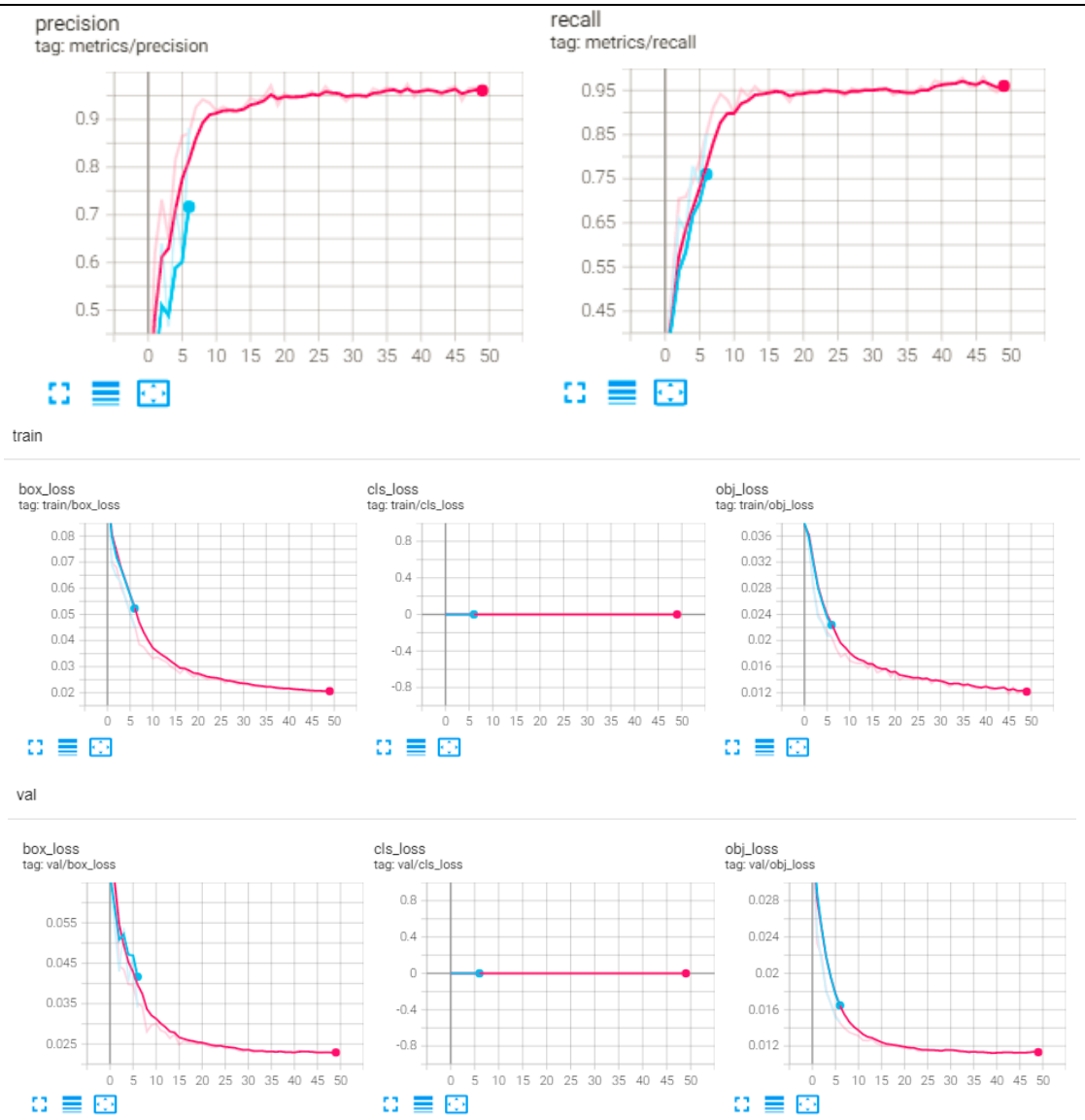

Figure 6. Training Results

From the results, the precision and recall is both approaching 1, and the train loss and valid loss is approaching 0. So the training effect is very well.

5. Final System Establishing

Aftter finishing the above works, the YOLOv5 face detection system has preliminary completed. So the final work is to test the effect of this system in practical application.

I fist use the system to detect the photos. The source codes is also published on github. Then I use the weight value document I trained above to detect the photos. One of the results is shown in Figure 5.

Figure 7. Face Detection In Photos

The system can detect the face in the photo, automatically mark them, and labelled the class name with the probability. The effect of the detection is accuracy as shown in Figure 7.

When the system could detect the static photo well, I could use it to detect the vedio. The principle for detecting in vedio is similar to do in photo. The system will fist sample each frames of the vedio as the photo, and then detect it. For YOLO detecting, it will only have very low delay time for detection, so it can achieve the real-time detection. I found a vedio, and let the system to detect it. One of the results is shown in Figure 8.

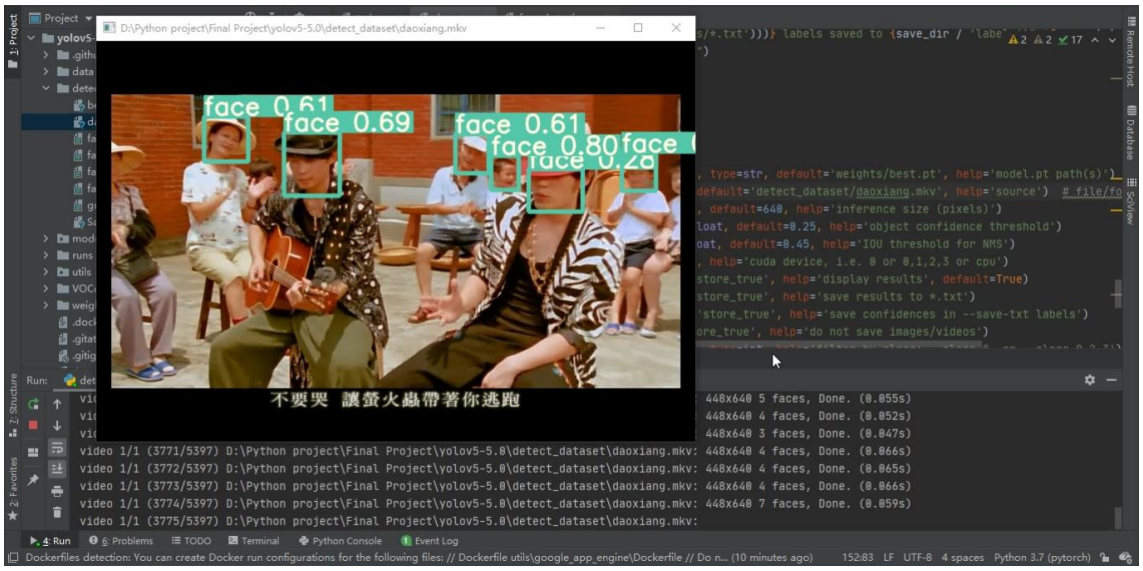

Figure 8. Face Detection In Real-time Vedio

The Figure 8 present the effect of detection for one frame in a vedio. It can detect almost all of the faces in the vedio, so the effect of the YOLOv5 face detection system can already achieve the fundamental requirements of the project.

Background Material

[1] Girshick, Ross. “Fast R-CNN.” 2015 IEEE International Conference on Computer Vision (ICCV), IEEE, 2015.

[2] Redmon, Joseph, et al. “You Only Look Once: Unified, Real-Time Object Detection.” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016.

Conclusions from initial work

In conclusion, I have already completed the full steps for building a face detection system, and the test results verified the correction of my work. Besides, the final results of this face detection system has already achieved the fundamental requirements of the project, which can achieve the real-time face detection in the vedio with a relatively great accuracy and efficiency.

However, there still exist some mistakes or shortcomings. For example, the system will miss the face in some frames in the vedio, or sometimes recogonized the wrong objects as face.

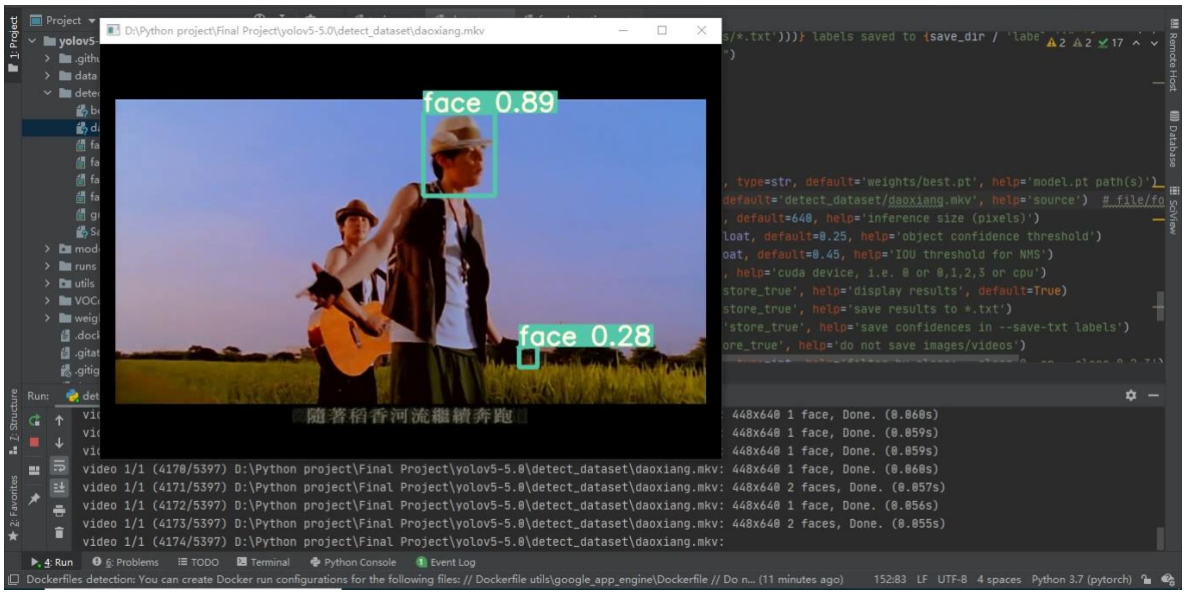

Figure 9. Detectiing Error

The Figure 9 is an example of the errors apprear in the detection. There are two faces in this frame, but it only recognized one, and a wrong place was detected. So though it achieved a quite high accurancy and recall rate at training, it will still make mistakes in detecting another documents it never trained before.

The main reason caused these errors may be the data set. When I look back for the data set collected by myself, I found I prefer to the pictures with the clear faces, and the face towards the camera accounts for a large proportion, which could be detected easily by the system, and when it need to recognise some indistinct faces, it will occurred some errors. In other words, the generalization of the model is not enough.

For the traing speed, I spent more than 11 hours for training 1000 photos in 100 epochs. The lower configuration of my laptop's graphics card and GPU limits the speed of training. In the next stage, it may spent much more time if I want to expand the data set and increase the training epochs. Therefore, I need to find a better solution to save time.

Besides, most of the work I did until now is about the how to build the YOLO detection system, and I ignored to explain the fundamental principles of the YOLO network, that is how the network extract the high-dimensional features of human face, or other classes.

In addition, the innovation of this project I did is still not enough. Like for the exist network uploaded on the github or other public websites, I could make more improvement on this basis, like optimize network parameters. I still have a lot of work to do in the following stage.

Work to be done:

I still used the software named *Worktile* to assist me to finish the tasks plan and Gantt Chart, as I did in the last working stage, which can make the plan seems more clearly.

As I introduced in the Preliminary Report, I divided all of my work into four big parts, which are Theory preparation, Network model building, Pycharm simulation and Results & Conclusion. In this stage of work, I finished some work, and combined with the above conclusion, I add three new work should be done in the next work stage.

In conclusion, in the next stage, I need to add another 1000 photos with low quality, fuzzy human face to expand my data set, in order to let the model training with more generalization and improving the precision of the detecting. Besides, I will try to find a cloud serve from our school's laboratory or from other companies, and let the train processes run on the serve, which can greatly improve the efficiency. Finally, after finishing these work, I will start to prepare for the final project, including the report writing, presentation preparing, logbook part 3 writing and so on. The specific work are listed on the Figure 10 as shown below.

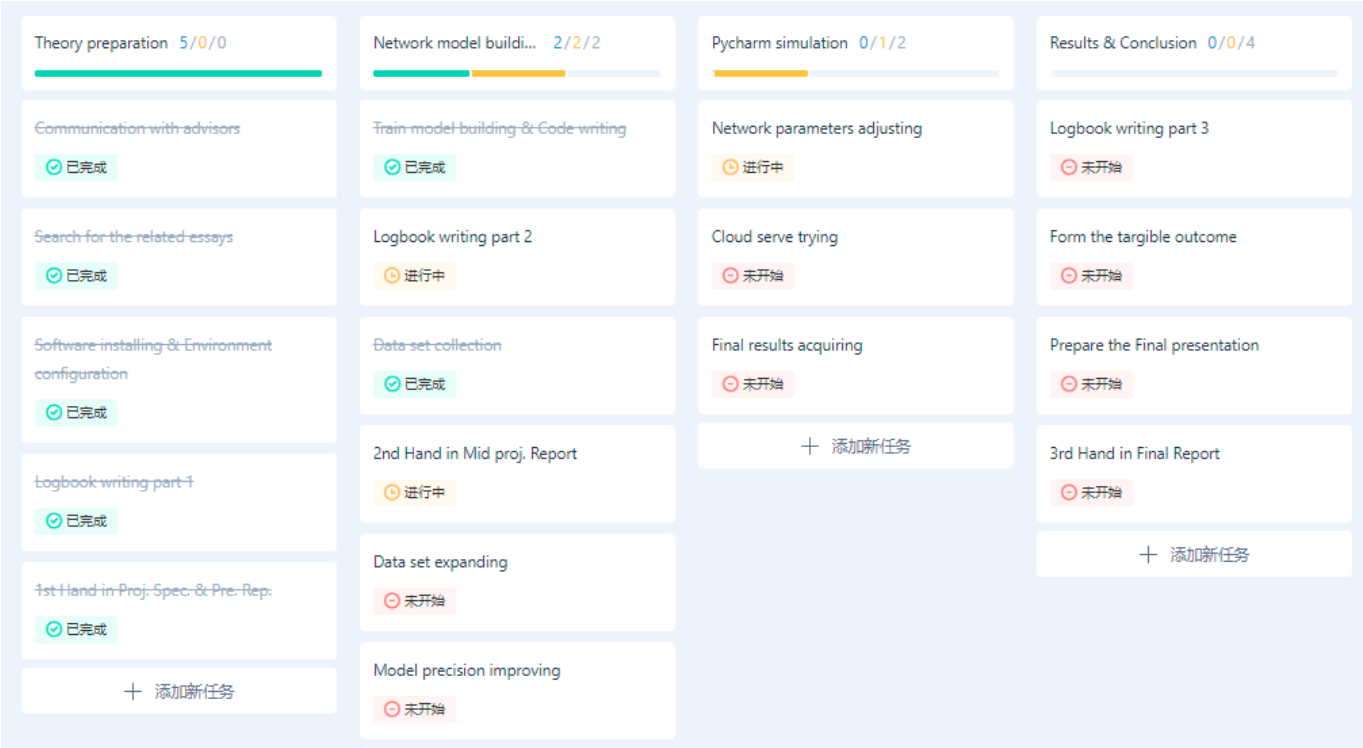

Figure 10. Work Plan listing

Both in the plan listing and the Gantt chart, the colour Green means the task has already done, Yellow means it is in progress, and Red shows that I have not begin to do it yet. When the report is submitted, the task *2st Hand in Mid proj. report* and *Logbook writing part 2* should be have done(in the colour of Green).

Revised Gantt Chart

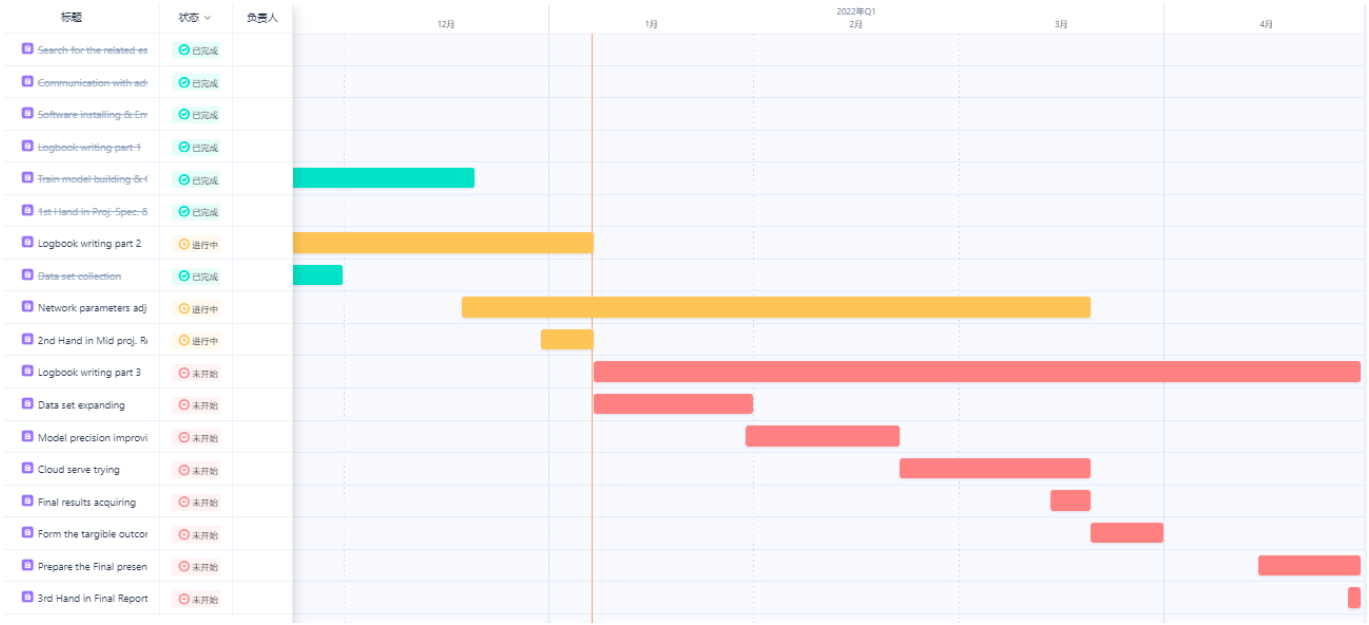

Figure 11. Gantt Chart

The Revised Gantt Chart is shown in Figure 11. Because of the page layout view limited, I mainly present the work in progress and to be done. The work I have already done is hiden, or the chart will be too small to view.

Interim Report on UESTC4006P(BEng) Final Year Project

Feedback from Second Supervisors: Second supervisors may provide their feedback by adding comments directly on Moodle taking into account the questionnaire below **or** by filling out the below form and uploading it to Moodle.

Name of Second Supervisor	

Was the report satisfactory? Yes ☐ No ☐

Are you satisfied with the (updated) scope of the project? Yes ☐ No ☐

Is the revised plan feasible? Yes ☐ No ☐

Would you like to give any suggestions/recommendations? Yes ☐ No ☐

Please write your comments in the space provided below:

Signature: **Date:**