APS360 - Progress Report

1002965026 Muyi Chen
1002839690 Jianyu Wen
1003149416 Yuanzhuo Wang
1002147651 Bowen Wu

1.Brief Project Description

Vehicles have proven to be one of the most useful tools in daily lives. License plates are used to identify unique vehicles in the legal system. As camera technologies develop, it has shown great potential for Automatic License-Plate Recognition (ALPR) to replace manual plate-reading. As a result, public agencies such as the police are already making use of this technology for a variety of tasks.

We propose to make an end-to-end ALPR system that, when given plate images, records the plate numbers that were shown. We believe the trained model can be applied to the parking enforcements, the recording of data, etc. As a visual recognition project, the project can scale very big. Hence, we intend to implement an increasing number of features as time allows, such as ROI localizing and decreasing time in recognition.

This is a diagram that shows the process of extracting the plate number: Notice that some steps in the diagram are extended features we may implement if time allows.

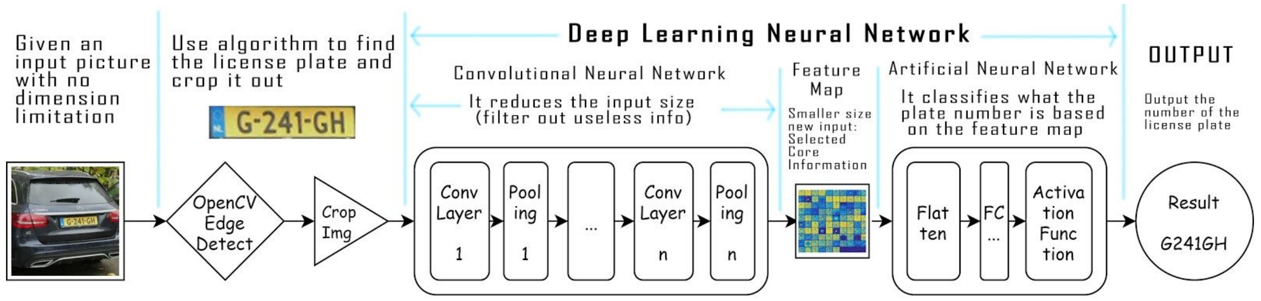

Given an input picture with no dimension limitation
Use algorithm to find the license plate and crop it out
G-241-GH
Deep Learning Neural Network
Convolutional Neural Network
It reduces the input size (filter out useless info)
Feature Map
Smaller size new input: Selected Core Information
Artificial Neural Network
It classifies what the plate number is based on the feature map
OUTPUT
Output the number of the license plate
OpenCV Edge Detect
Crop Img
Conv Layer 1
Pooling 1
...
Conv Layer n
Pooling n
Flatten
FC ...
Activation Function
Result G241GH

Figure 1. General project architecture

2. Data collection and process

There seems to be plenty of available datasets on license plates online. For the initial data loading, we have loaded a set of low quality plate images from PlateRecognizer. Unfortunately the link to the dataset we use is not available anymore after we pulled them, but the dataset should serve as a good start for us.

In this dataset, we have 182337 total images of license plates. A lot of these images are from the same plate, augmented to produce more data in quantity.

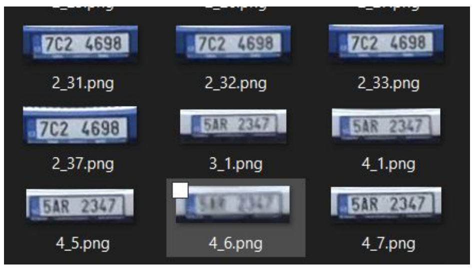
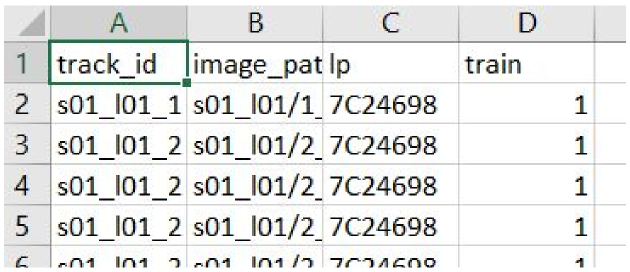


Figure 2(a).Sample data presentation



	A	B	C	D
1	track_id	image_pat	lp	train
2	s01_l01_1	s01_l01/1_	7C24698	1
3	s01_l01_2	s01_l01/2_	7C24698	1
4	s01_l01_2	s01_l01/2_	7C24698	1
5	s01_l01_2	s01_l01/2_	7C24698	1
6	s01_l01_2	s01_l01/2_	7C24698	1

Figure2(b).CSV file for labeling

The labels of said images are noted in a separate csv file. We have used the panda library to read data from the csv in the data loader.

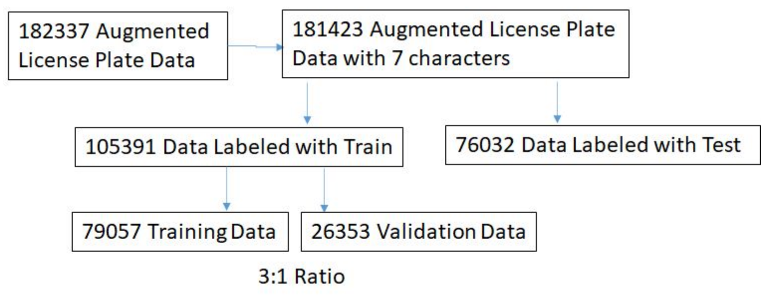

182337 Augmented License Plate Data
181423 Augmented License Plate Data with 7 characters
105391 Data Labeled with Train
76032 Data Labeled with Test
79057 Training Data
26353 Validation Data
3:1 Ratio

Figure 3. Custom dataloader flow

Using the csv document alongside with the dataset, the data loader returns data as a pair of (Image, Label). Normally we would be able to use the Pytorch's default ImageFolder data loaders, but in this case the labels come from the csv document.

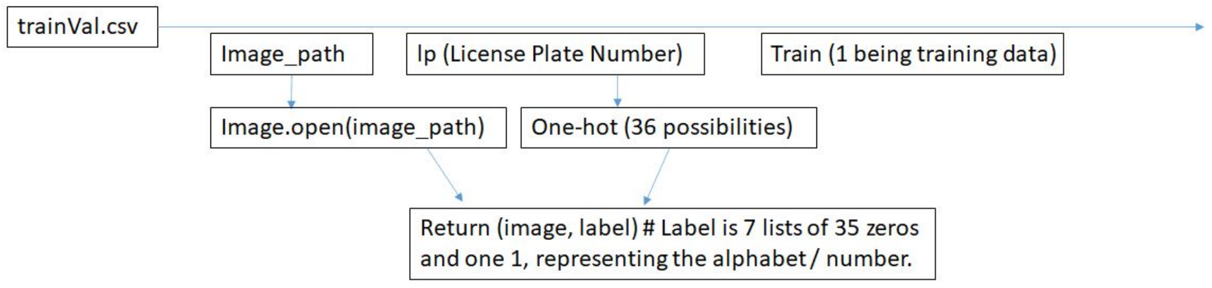

trainVal.csv
Image_path
lp (License Plate Number)
Train (1 being training data)
Image.open(image_path)
One-hot (36 possibilities)
Return (image, label) # Label is 7 lists of 35 zeros and one 1, representing the alphabet / number.

Figure 4. CSV document processing flow

We understand that for a project that has a large scale like this, the number of data we have might not satisfy an accuracy requirement. In the case of implementing the extended features as stated in our project plan, we may need to incorporate more license plate data in a similar fashion. The pictures can be from online, or be built on our own by pasting plate numbers on pictures or sceneries.

3. Baseline model

According to experiments, we have discovered that using a classifier does not fit our problem, since each of the labels is a combination of 7 characters, it will result in a model classifying a plate with a fixed combination of characters, rather than classifying each one. Using sklearn API, we work around by using a one-hot encoder and Random Forest Regression.

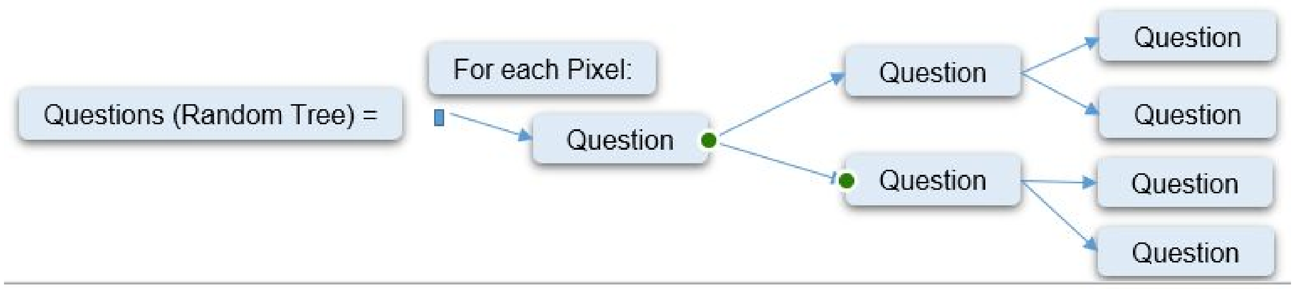
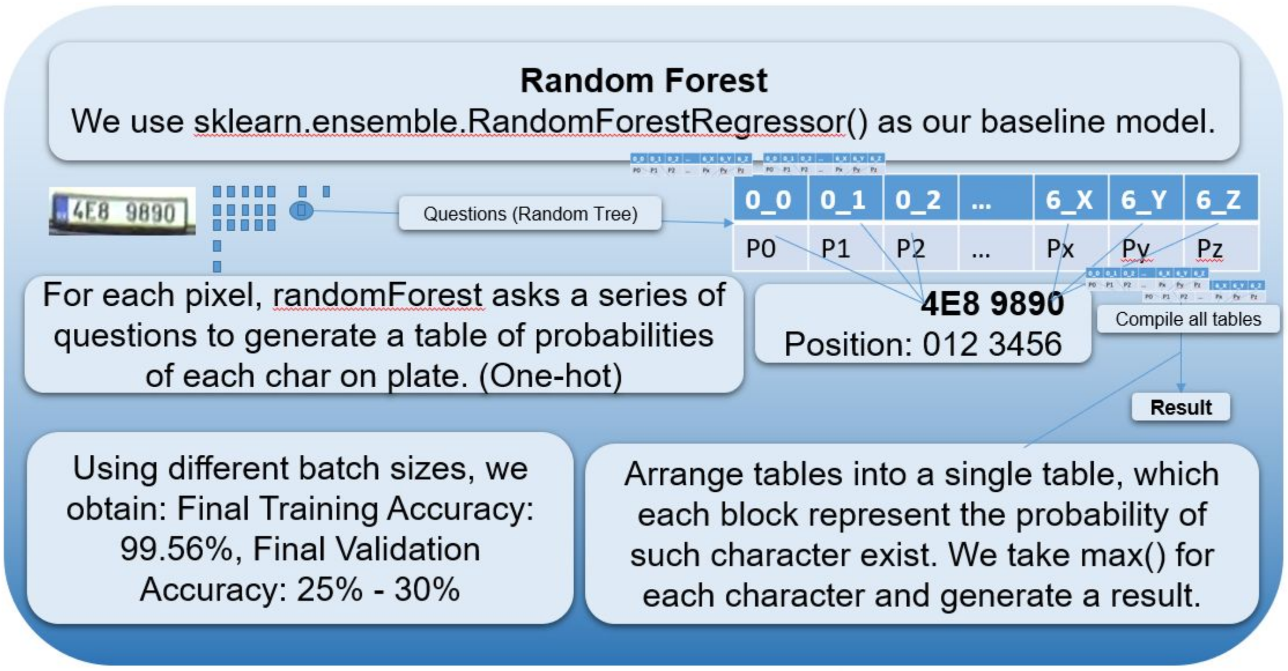


Figure 5a. Baseline Model Explanation



Random Forest

We use sklearn.ensemble.RandomForestRegressor() as our baseline model.

For each pixel, randomForest asks a series of questions to generate a table of probabilities of each char on plate. (One-hot)

Using different batch sizes, we obtain: Final Training Accuracy: 99.56%, Final Validation Accuracy: 25% - 30%

Arrange tables into a single table, which each block represent the probability of such character exist. We take max() for each character and generate a result.

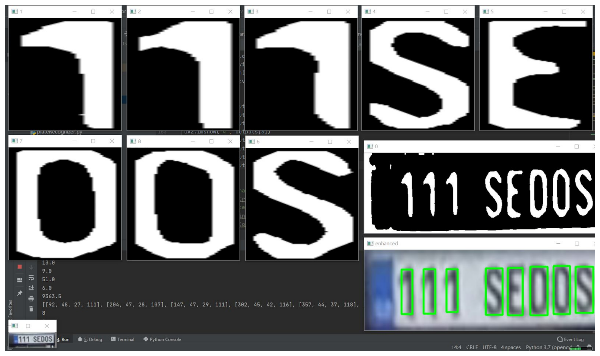
0_0	0_1	0_2	...	6_X	6_Y	6_Z
P0	P1	P2	...	Px	Py	Pz

Figure 5b. Baseline Model Explanation

4. Primary Model

- **OpenCV:**

OVERVIEW:



```
13.0
9.0
51.0
6.0
9363.5
[[92, 48, 27, 111], [204, 47, 28, 107], [147, 47, 29, 111], [302, 45, 42, 116], [357, 44, 37, 118],
8
```

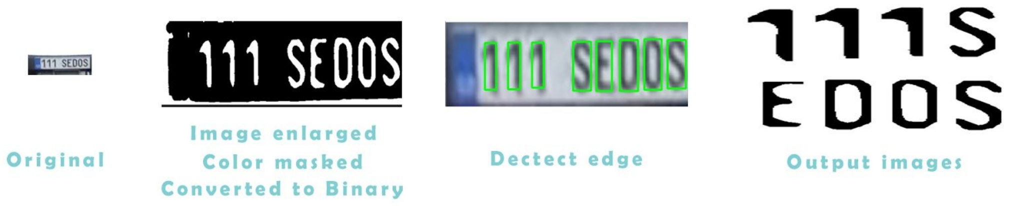


Figure 6. OpenCV extraction on example image

The first part of the model is to locate the individual numbers and characters, out of a raw image of a license plate. This part is implemented using openCV.

1. The original 115 x 38 sized image is imported and converted to 1 channel grayscale, apply gaussian blur to enhance edges of the numbers, and then use a canny edge detection filter to prepare for cropping the entire license plate out. Convert the processed image to binary by applying a color mask.

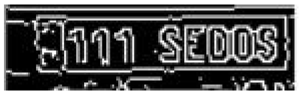


Figure 7. Canny masked image(115 x 38)

2. Apply opencv built in edge detection on the canny masked image to find the outer bounding box of the license plate.

3. Crop the license plate out and enlarged to 600 x 200 resolution, in order to perform edge detection. A simple opencv built in enlargement function is used in this part. Apply step 1 and 2 again to find the binary masked image. Crop the license plate out and binary masked image

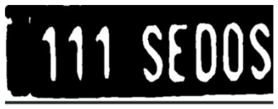


Figure 8. Binary masked image(600 x 200)

4. Find the bounding box for each letter by applying step2 and apply area constraint for each detected contour. Locate their bounding boxes on the enlarged original image.

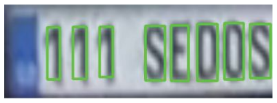


Figure 9. Bounding boxes on 600 x 200 image

5. Crop the picture of each letter out and convert them to binary. Resize each picture to 280 x 280 resolution, store them in a list for further use.

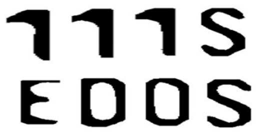


Figure 10. Letter images (280 x 280 each)

- **ML Model Architecture:**

OpenCV cropped out images of single characters will be fed into this Machine learning model:

Layer name	Tensor size	Functionality
Single char Input	1*280*280	N/A
Convolutional 1	5*276*276	Feature extraction
Max Pooling	5*138*138	Reduce dimension, mitigate overfitting

Convolutional 2	10*134*134	Feature extraction
Max Pooling	10*67*67	Reduce dimension, mitigate overfitting
Fully connected 1	1*44890	Combine Features
Fully connected 2	1*2500	Hidden layer
Prediction	47	Output

Table 1: Layer configuration

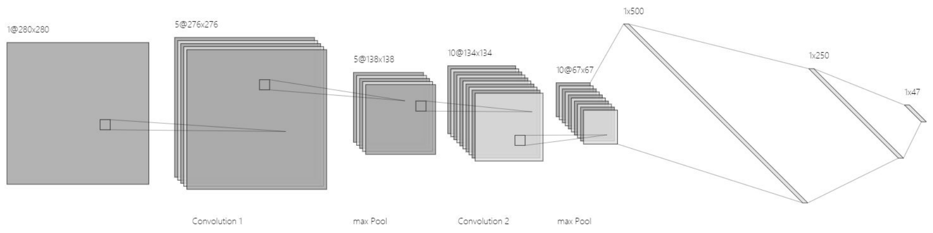


Figure 11(a). Deep learning architecture

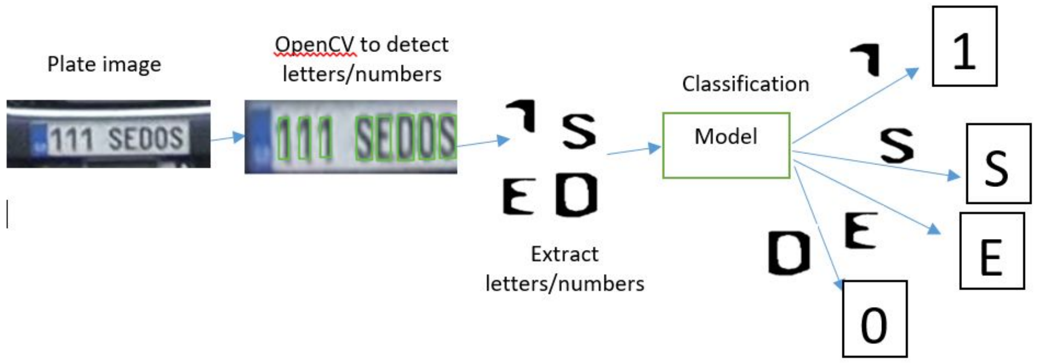
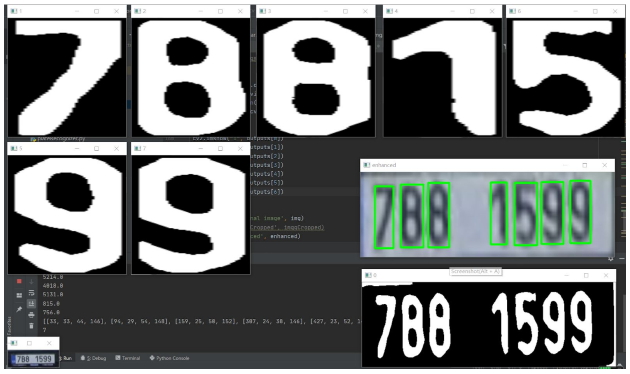


Figure 11(b).Primary model workflow

5. Result

1. OpenCV accuracy result:

To test OpenCV accuracy, 5238 test samples with 7 letters are parsed into the cv model. If the model recognizes exactly all 7 letters from the input image, it is considered a pass.



```
5214.0
4018.0
5131.0
815.0
756.0
[[33, 33, 44, 146], [94, 29, 54, 148], [159, 25, 50, 152], [307, 24, 38, 146], [427, 23, 52, 1
7
```

Figure 12. Sample openCV extraction

Total tested images	No letters found	Less than 7 letters	**Exact 7 letters found**
5238	2225	1380	1633

Table 2. OpenCV extraction result statistic

The test result indicates a 31.2% recognition accuracy. This may be due to the fact that letters are detected by filtering contours in a specific area range. This range is manually tuned. The inaccuracy of range threshold will introduce error in character detection. Currently, a rough threshold is applied which needs to be improved by trial-and-error.

2. **Classification model result on single character:**
   - **Hyperparameter & Final train accuracy**

learning rate	**batch_size**	**epochs**	**Final train accuracy**	**Final validation accuracy**
0.0001	32	10	0.99745	0.57143

Table 3. hyper-parameter & final accuracy of single char recognition model

   - **Training curves:**

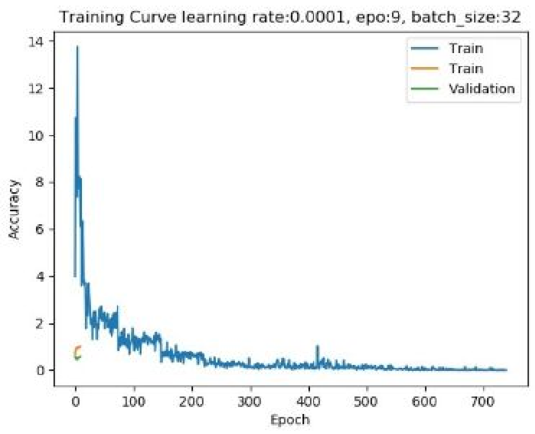


Figure 13(a). Training Loss

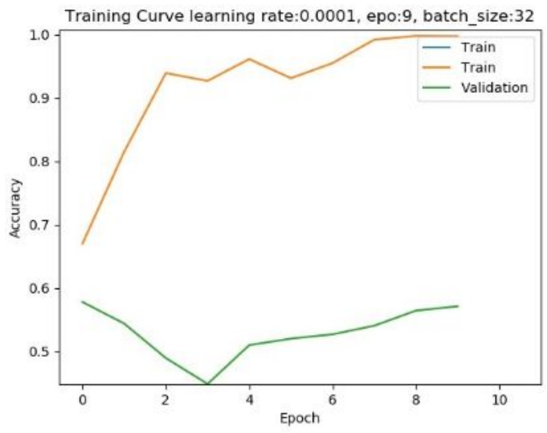
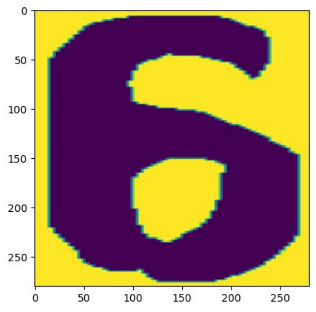


Figure 13(b): Training & Validation Curve

- **Example Image and model prediction**

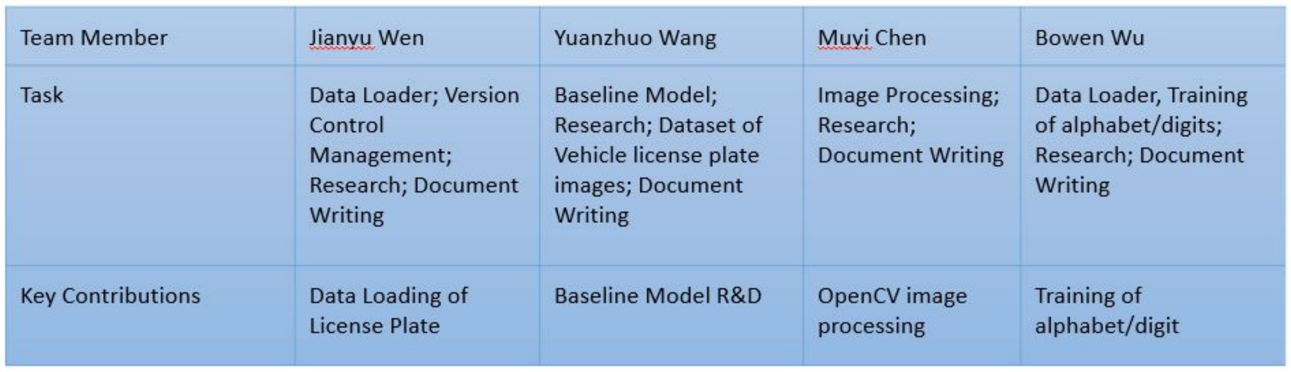


Model prediction: tensor([6]), label tensor([6])

Figure 14. Example input from OpenCV

6. Project Progress

- **Task Done:**



Team Member	Jianyu Wen	Yuanzhuo Wang	Muyi Chen	Bowen Wu
Task	Data Loader; Version Control Management; Research; Document Writing	Baseline Model; Research; Dataset of Vehicle license plate images; Document Writing	Image Processing; Research; Document Writing	Data Loader, Training of alphabet/digits; Research; Document Writing
Key Contributions	Data Loading of License Plate	Baseline Model R&D	OpenCV image processing	Training of alphabet/digit

Figure 15.Task Completed

We started to work on the project on July 9th. Since then, we have consistently communicated with each other in voice meetings on WeChat, and everyone is committed to work on the project with 100% effort.

After some discussions with the TAs, we discovered that our full project could potentially be too ambitious for the course. We also started with the project late due to being busy with PEY works, so we are likely behind the schedule on completing everything that has been proposed in the Project Proposal. We have since amended some of our objectives to meet our time restraints, and changed our destinations to sub-goals.

- **Project Plan:**

We have around 20 days before our presentation, and some of us are still busy with PEY work. We propose to put our objectives in steps, and make more features available if time allows.

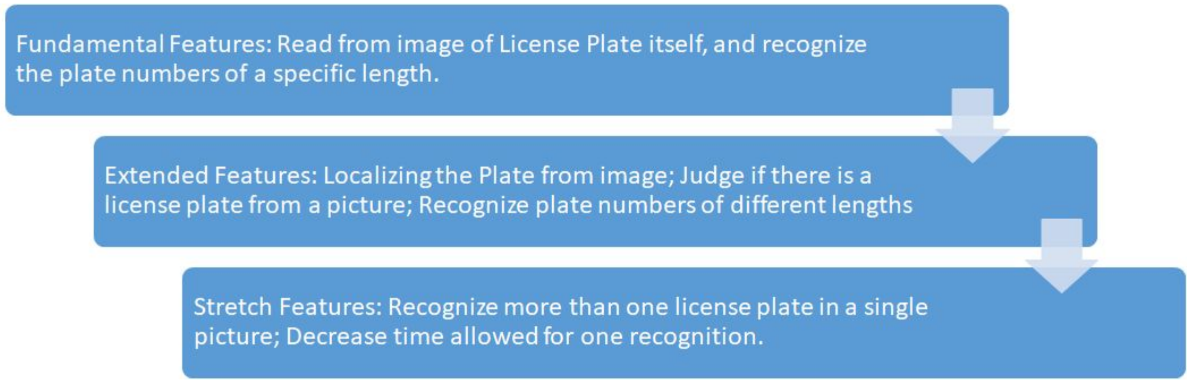
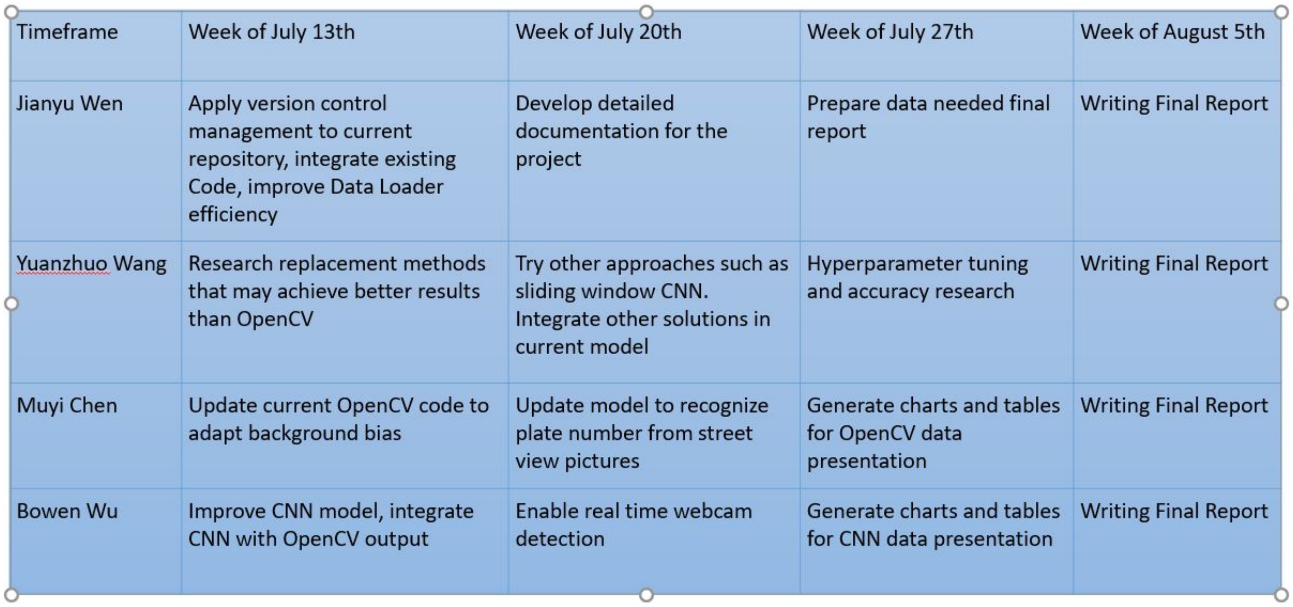

Fundamental Features: Read from image of License Plate itself, and recognize the plate numbers of a specific length.
Extended Features: Localizing the Plate from image; Judge if there is a license plate from a picture; Recognize plate numbers of different lengths
Stretch Features: Recognize more than one license plate in a single picture; Decrease time allowed for one recognition.

Figure 16. Future Feature Diagram

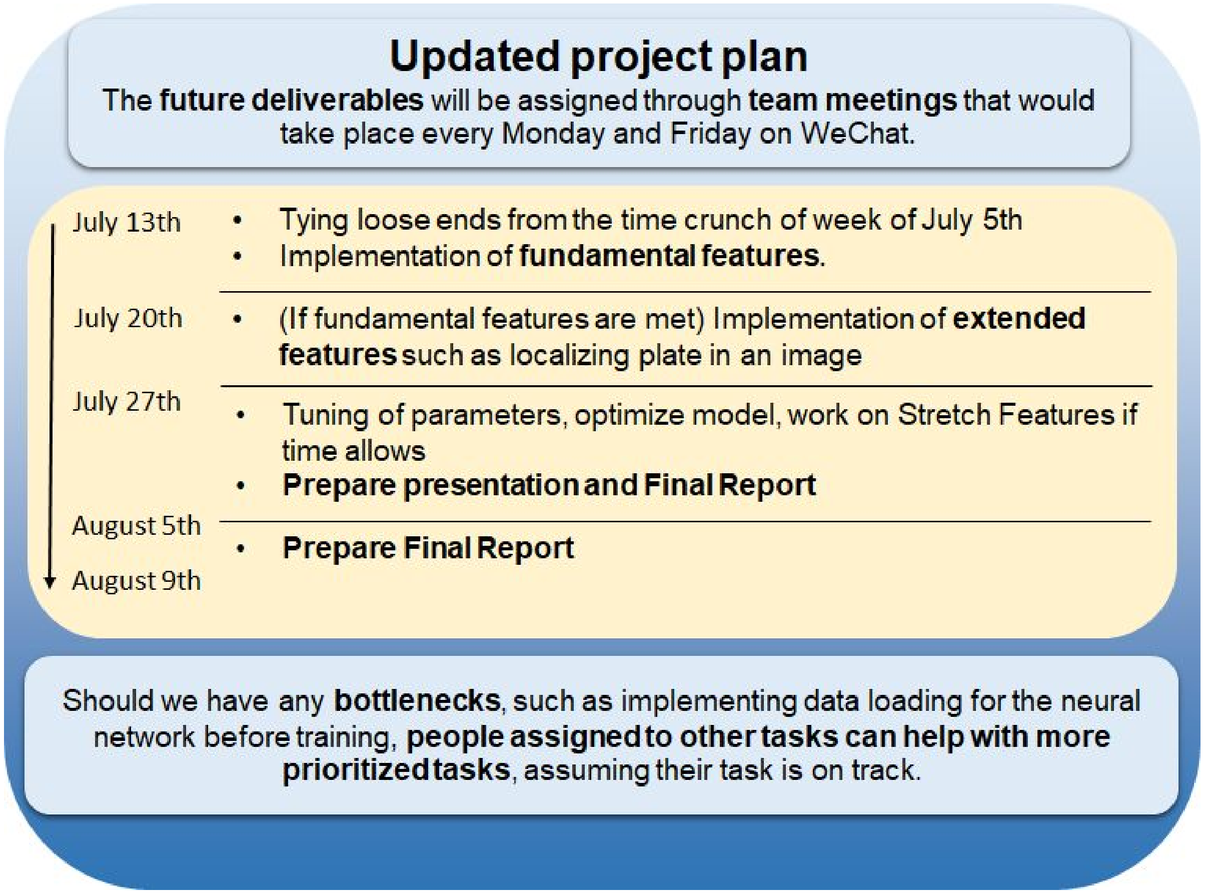
In the diagram shown above, we promise to deliver the fundamental features, while working on extended features to the best of our abilities. Stretch features should only be worried about after a majority of extended features are implemented and if time permits.

- **Future Task Assignments:**



Timeframe	Week of July 13th	Week of July 20th	Week of July 27th	Week of August 5th
Jianyu Wen	Apply version control management to current repository, integrate existing Code, improve Data Loader efficiency	Develop detailed documentation for the project	Prepare data needed final report	Writing Final Report
Yuanzhuo Wang	Research replacement methods that may achieve better results than OpenCV	Try other approaches such as sliding window CNN. Integrate other solutions in current model	Hyperparameter tuning and accuracy research	Writing Final Report
Muyi Chen	Update current OpenCV code to adapt background bias	Update model to recognize plate number from street view pictures	Generate charts and tables for OpenCV data presentation	Writing Final Report
Bowen Wu	Improve CNN model, integrate CNN with OpenCV output	Enable real time webcam detection	Generate charts and tables for CNN data presentation	Writing Final Report

Figure 17. Future Task



Updated project plan

The **future deliverables** will be assigned through **team meetings** that would take place every Monday and Friday on WeChat.

July 13th
- Tying loose ends from the time crunch of week of July 5th
- Implementation of **fundamental features**.

July 20th
- (If fundamental features are met) Implementation of **extended features** such as localizing plate in an image

July 27th
- Tuning of parameters, optimize model, work on Stretch Features if time allows
- **Prepare presentation and Final Report**

August 5th
- **Prepare Final Report**

August 9th

Should we have any **bottlenecks**, such as implementing data loading for the neural network before training, **people assigned to other tasks can help with more prioritized tasks**, assuming their task is on track.

Figure 18. Project timeline

Redundancy and risk control:

Problem	Computing power limit	Version control and mis-communication	Model low accuracy	Internal milestones not met	Team member failed to complete task
Solution	We will use a local terminal with good hardware to train the model, instead of using google colab. GPU: RTX2070 CPU: I79750H	A team member is assigned for integrating all codes who performs version control. All team members will be coding under one regulated structure.	A team member is assigned to research better models and add new solutions to existing models. Solution not limited to current choice.	A team meeting is required every Monday and Friday and each member must report status. All status will be recorded and tracked.	Members will be split in groups of twos and will be reporting to the team every Monday and Friday. If a member is failing his task, the other members should be responsible to help and report ASAP.

Table 4. Redundancy & risk control