Min Sung Cha 85408485

Chan Woo Park 26984415

Joowon Suh X0992979

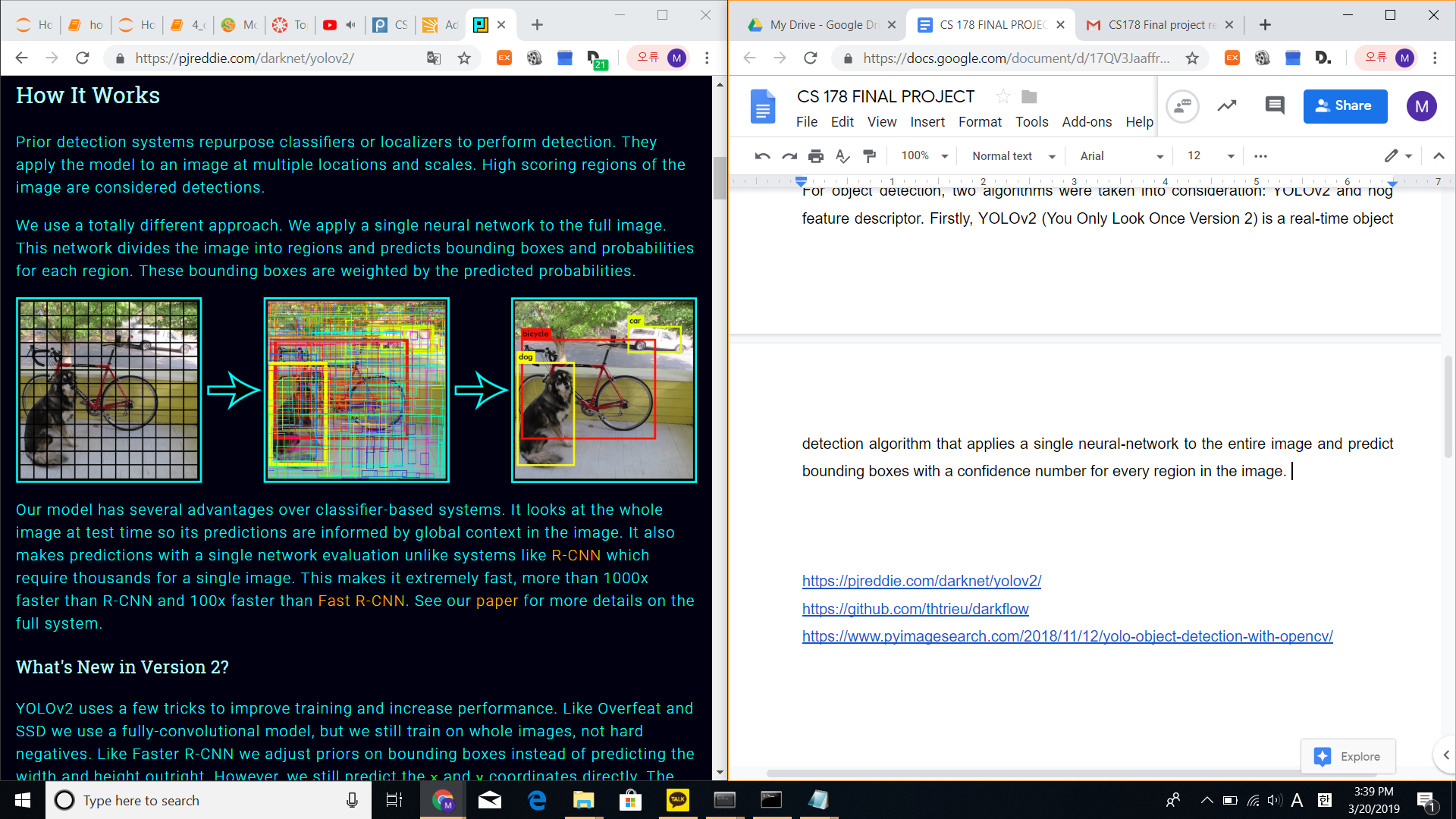
CS 178 Final Project

License Plate Detection

**Problem Definition**

The problem under concern for our final project is license plate recognition via object detection and text recognition algorithms. With the increasing popularity in the field of computer vision regarding detection and recognition of images, our goal is to implement a back-end application which is capable of recognizing the location of a motor vehicle stationed in a parking structure. As human beings, it is often the case that people forget the location his/her car was parked at. This problem is more relevant in parking structures with considerable size that the driver has never been exposed to before. In order to provide the convenience of easily locating the vehicle by minimizing time wastage, our application aims to detect the license plate number of a vehicle through a small webcam installed at every stall in the parking structure. Once our application detects the license plate through object detection, only the license plate is cropped out from the image and fed onto the Google Cloud Vision API Convolutional Neural Network. After obtaining the license plate digits, it would then ideally be stored in a database with the corresponding webcam stall location. With this, the user would be able to easily locate his/her vehicle by inputting the license plate number. Due to time constraints and requirements for this course, we focused on the back-end development and implementation of the license plate detection and text recognition which corresponds to the machine learning and artificial intelligence part of the application.

For object detection, two algorithms were taken into consideration: YOLOv2 and hog feature descriptor. Firstly, YOLOv2 (You Only Look Once Version 2) is a real-time object detection algorithm that applies a single neural-network to the entire image and predict bounding boxes with a confidence number for every region in the image (**Figure1)**.





The algorithm is able to make prediction based on the global context of the image and unlike R-CNN that require thousands of networks for a single image based on the segmented regions, predictions are made with single network evaluation. This methodology is currently considered as the state-of-art in object detection.

Secondly, hog feature descriptor, a relatively old object detection algorithm, was considered for detection. This algorithm simplifies the image by extracting the gradients which include orientation and magnitude of each edge in the image. With these features, a template is trained with the target object we look forward to detect. This template is then cross correlated across an hog image and find regions with the highest correlation with the hog template.

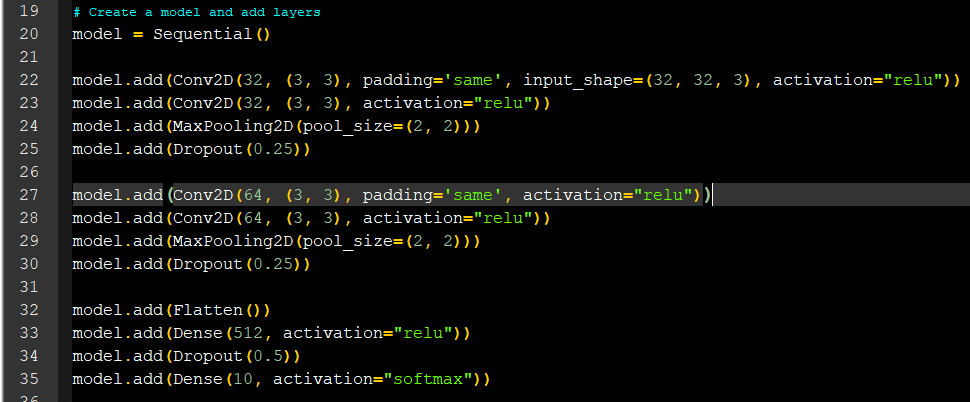
**Approach**

For license plate detection, the current state-of-the art is the research paper *Towards Human-Level License Plate Recognition* which utilizes semantic segmentation and character counting to extract the digits of a license plate in an image. A convolutional neural network is trained and harnessed to detect and recognize the digits in a license plate. Semantic segmentation is implemented by the production of a semantic map in order to recognize the initial character sequence for an input license image. A wrapped bounding box is implemented for each character and the pixels in the bounding box are labeled with the class of the corresponding character. With the semantic map of a license plate, then through character counting module is used to produce the final character sequence. Although the impressive advancements of this state-of-the-art research paper, we figured out that semantic segmentation was a computationally expensive algorithm which utilizes deep learning and convolutional neural networks that require a great amount of time. As our application is exposed with images that are fairly modest to recognize due to vehicle stability, moderate lighting, and closeness to the webcam we decided to implement hog features object detection methodology which is a computationally cheap and fast algorithm. As efficiency overweights the accuracy in our license plate recognition application, hog features object detection is a suitable approach.

Once the object detection procedure is completed, the captured images of license plates can be fed into the next step. The Keras library supports different types of models to build neural networks. One of these structures is called the Sequential model. The sequential model allows the use of multiple layers in a neural network, where each layer takes input from the previous layer and forwards processed data to the next.

Generally, each layer is responsible of capturing a specific pattern in the given image. To train a neural network to recognize texts, a series of multiple layers was combined:

* Dense layer: collect information to process and classify final output
* Convolutional layer: a filter to find patterns in given input
* Max pooling layer: merge and reduce data size
* Dropout layer: randomly drop out a percentage of the data to prevent memorization



By training the neural network with image datasets of each character of the English alphabet, the neural network should be able to find patterns corresponding to each alphabet and be able to recognize what text is visible in the input. This means finding datasets for each character and number, and performing training on each.

**Data**

The datasets that we used as inputs to our application were from the Medialab LPR Database and the OpenALPR library benchmarks. 443 images of cars and their license plates were used as training data from the Medialab LPR dataset, while 241 images were used as testing data from the OpenALPR dataset.



Medialab LPR Database: <http://www.medialab.ntua.gr/research/LPRdatabase.html>

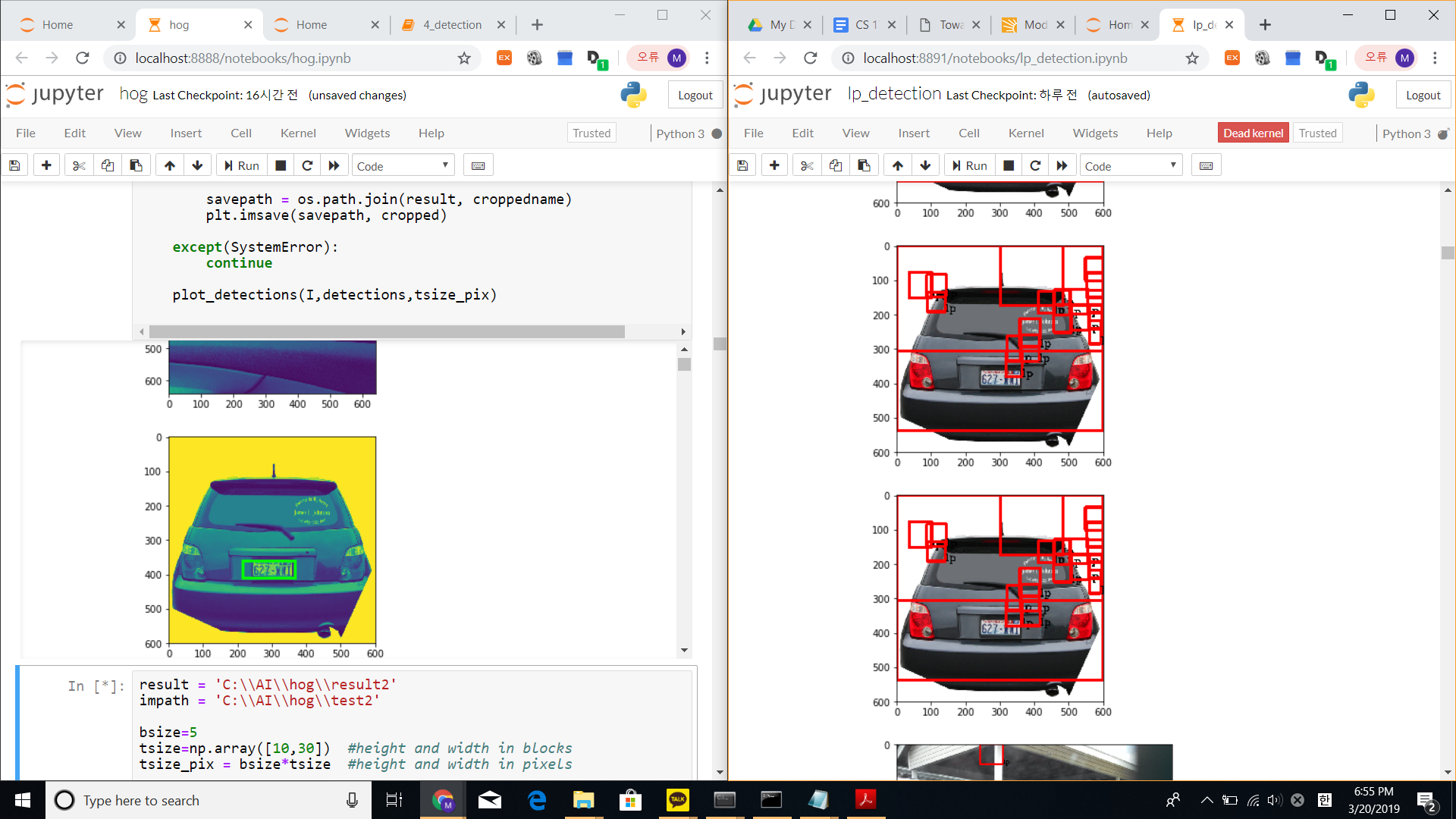


OpenALPR library: <https://github.com/openalpr/benchmarks>

Both datasets are public resources available online. The Medialab LPR dataset is a collection of images of cars from 2007, while the OpenALPR dataset is data collected from 2014.

**Analysis**

Firstly, YOLOv2 was implemented as it was considered an efficient and fast algorithm for object detection. However, after custom training the model with images of cars and their license plate bounding boxes, YOLOv2 was not able to successfully detect the license plate for any of the images provided as test set. As the training set, the OpenALPR benchmark was utilized which is a dataset that contains images of vehicles from Europe, Brazil, and U.S. It contains car images from a variety of lighting, orientation, distance, and resolution. As the test set, the Medialab LPR Database was utilized which contains a range of front and rear car images with different views, lighting, distance, blurriness, and orientation. Due to computational limits, the tiny-yolo model was used with 300 epochs to train the model to classify 1 class: license plate. After, training the model and testing it with the test data the results seen in the below figure were obtained.

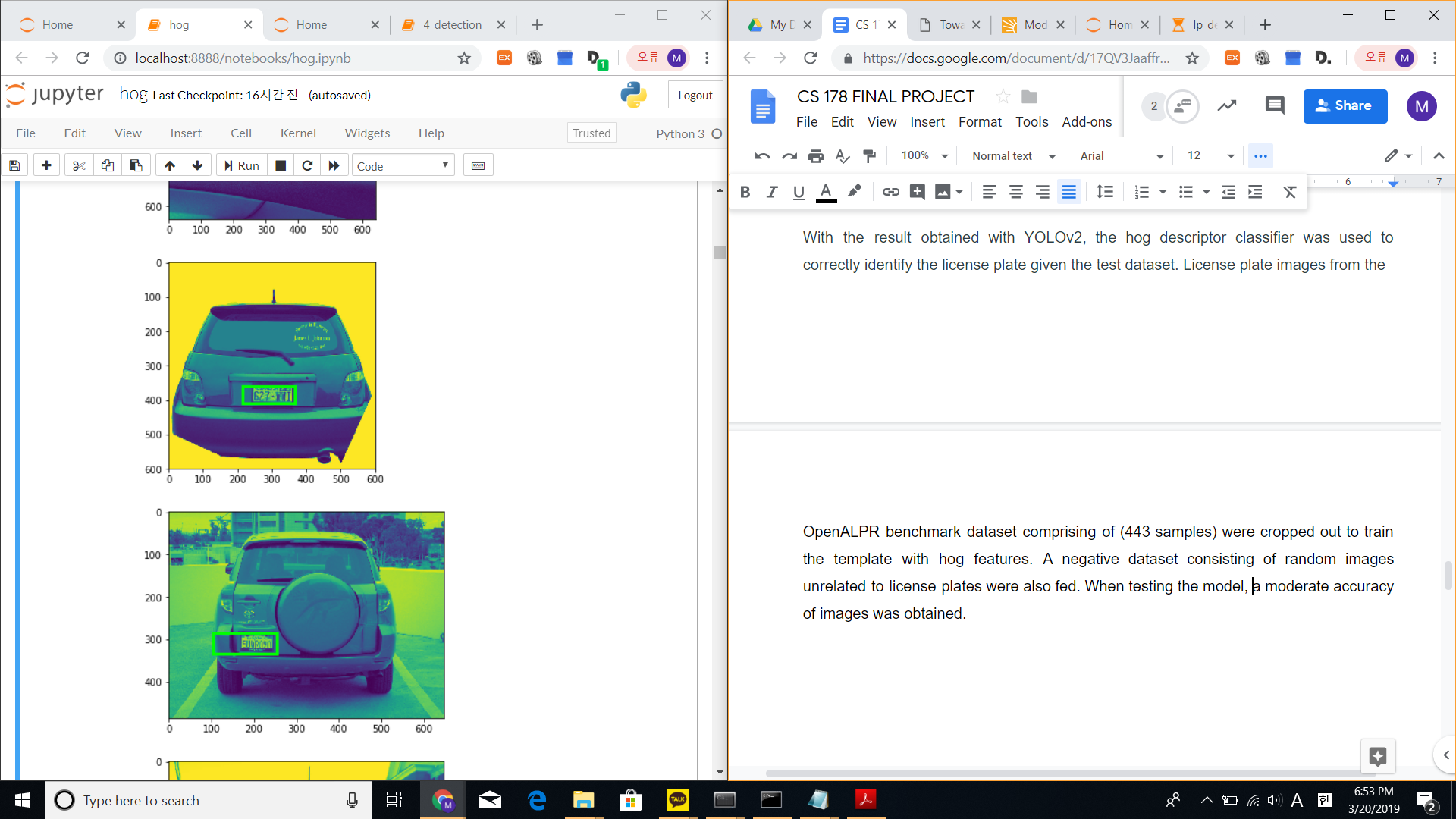




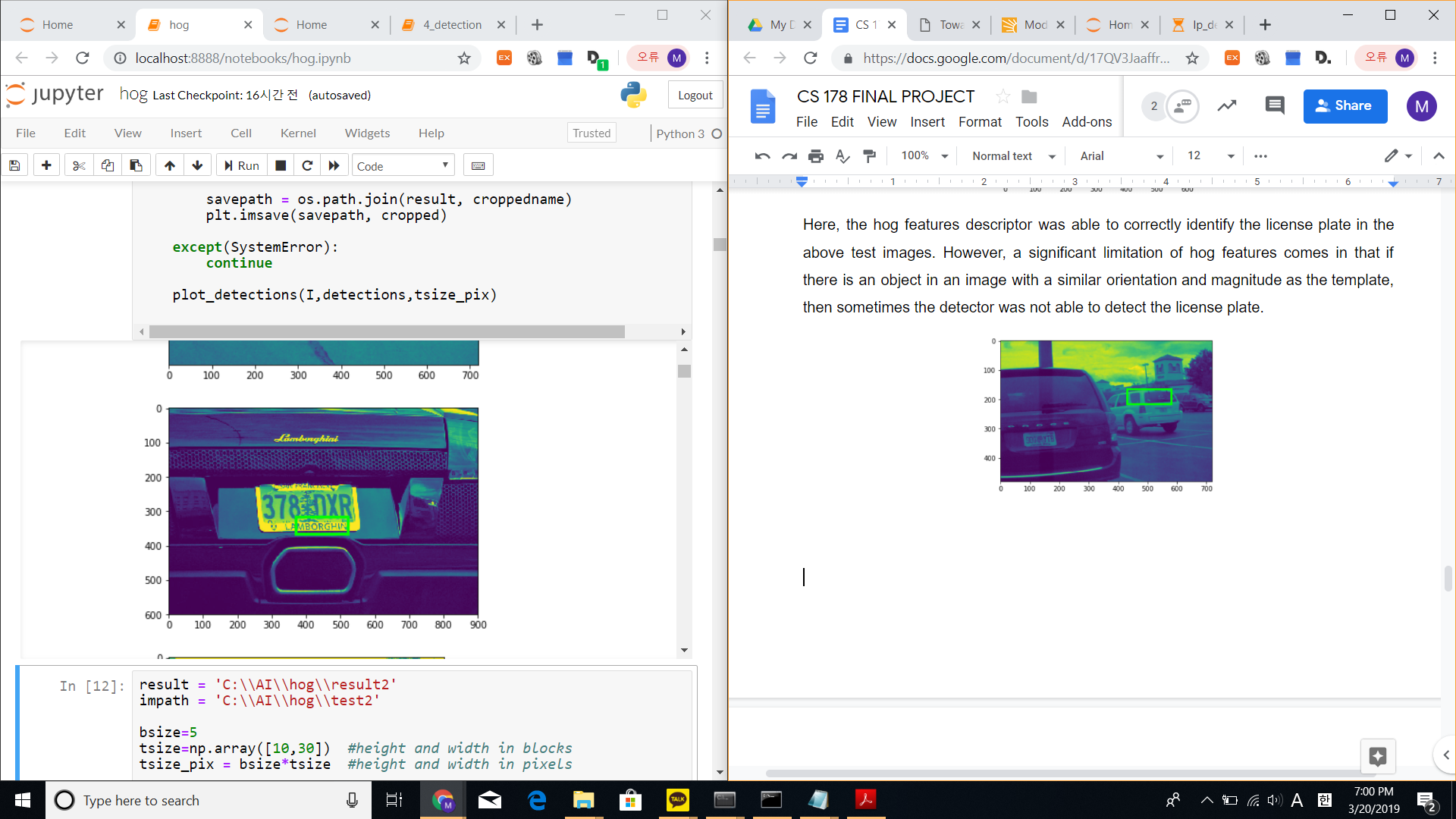
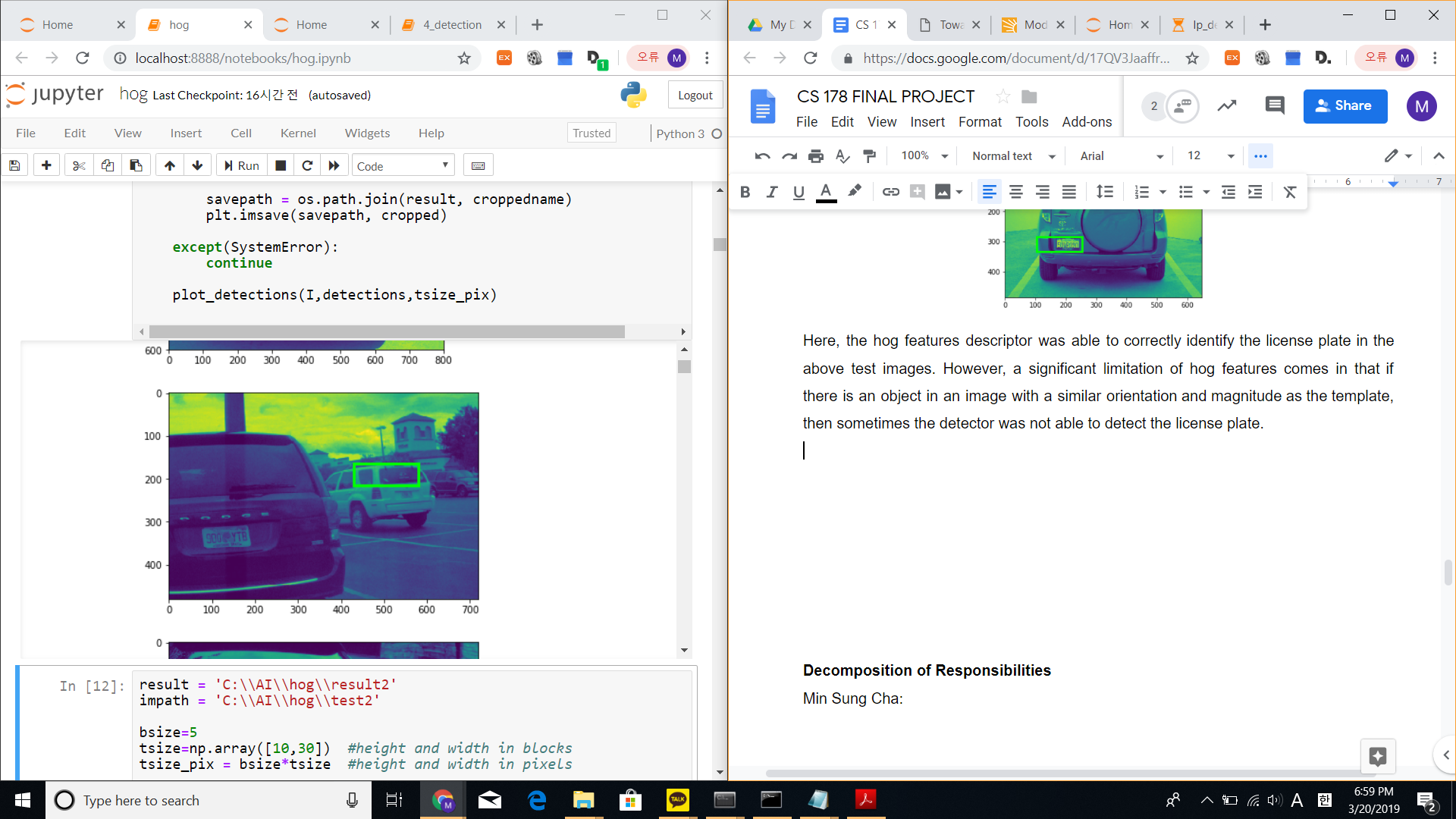
YOLOv2 was not able to detect the license plate (lp) successfully for any of the testing data set. As seen above a clear bounding box covering the license plate could not be obtained using YOLOv2. This provided insight into YOLOv2 limitations in recognizing license plates with a relatively small training data set (443 samples). An enormous amount of computational resources and graphical processing unit are required to train the model in order to correctly classify and detect an object. Therefore, in the future, a greater number of training samples coupled with more computationally capable computers and greater number of epochs could be utilized to more accurately detect license plates.

With the result obtained with YOLOv2, the hog descriptor classifier was used to correctly identify the license plate given the test dataset. License plate images from the

OpenALPR benchmark dataset comprising of (443 samples) were cropped out to train the template with hog features. A negative dataset consisting of random images unrelated to license plates were also fed. When testing the model, a moderate accuracy of images was obtained.

****

Here, the hog features descriptor was able to correctly identify the license plate in the above test images. However, a significant limitation of hog features comes in that if there is an object in an image with a similar orientation and magnitude as the template, then sometimes the detector was not able to detect the license plate accurately as seen in the below images.



Limitations in the hog features come in that images contain license plates with different sizes, but there is only one trained template to recognize the object. Therefore, in the future, our hog descriptor could be enhanced by training a variety of template sizes and resizing the image so that the license plate in the image matches with the size of the template for more accurate detection.



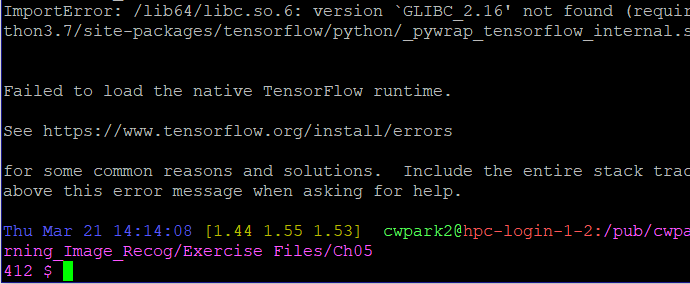
License plates from Medialab dataset



License plates from OpenALPR dataset

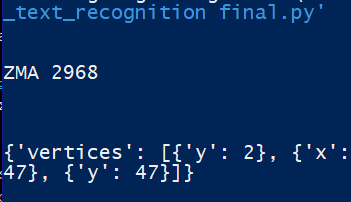
The final step was to perform text recognition on the captured license plate images. Multi layered convolutional neural network models were built with the purpose of feeding and training text image data. However, due to limitations in the resources available, there were extensive time constraints for conducting the training procedure. It was close to impossible to train the neural network in classifying all characters in the English alphabet, along with the numbers 0 through 9.

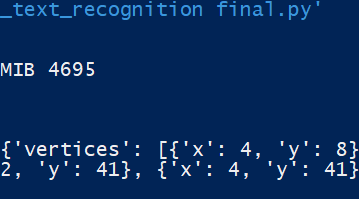
The team looked for setting up an environment for prompt and efficient training of neural networks. There were too many tasks for training the neural network to recognize the entire English alphabet and numbers. Each character and each digit was required to be trained on each isolated dataset, meaning a total of 36 datasets were needed. This was a lot of work with the environment that the team had; it had taken three days to train HOG model for object detection, and needless to say training a neural network of this scale locally on a computer was going to take longer. One solution we found to this was using the UCI High Performance Computing Cluster (HPC). HPC has about 9900 64 bit CPU cores, about 55TB of RAM, and 30 Nvidia GPUs. We were able to gain access to this remote environment with the purpose of executing all the training procedures. The team contacted administrators of the HPC to get help in understanding and learning how to utilize installed modules on the HPC.



However, another issue was brought up with the operating system that the HPC was running being too outdated to run the Python modules that used tensorflow libraries. This was because CentOS6 (HPC OS) was using GLIBC 2.12, while tensorflow required a minimum version of 2.16. The administrators were not able to help the team resolve this, and thus the team moved on to another solution.

Having failed both approaches in training a personalized neural network, the team took the solution towards utilizing publicly available pre-trained neural networks. The Google Cloud Vision utilized a similar convolutional neural network structure as mentioned, and so was a suitable replacement. By integrating the Cloud Vision API to the project, we were able to extract the text information from the license plates.





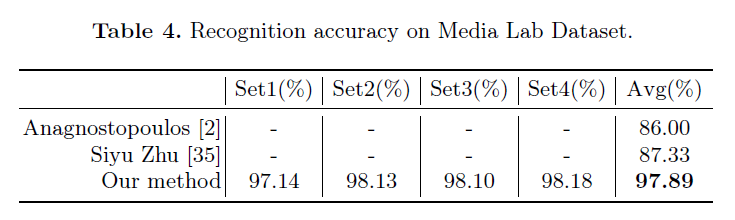
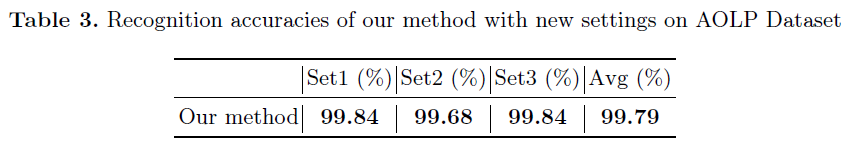
Both datasets were used for testing, because due to using Google Cloud Vision’s pre-trained neural network, there was no training data to be involved.

Table 1. Recognition accuracies on datasets

|  |  |  |
| --- | --- | --- |
| **Dataset** | **Recognized** | **Accuracy** |
| Medialab LPR | 263 / 443 | 59.37% |
| OpenALPR | 196 / 365 | 53.70% |

Performing the check on whether the returned text data from Google Cloud was correct required some manual work. There was no target value set in our data to confirm the output of the neural network. Each image filename was labeled with integers, starting from 1 through 443 or 365. A text file was written for each set with the line numbers corresponding to the image filename, holding a string value of the correct license plate number. The classification between correct and incorrect tests was then performed by comparing the output from Google Cloud to the matching line number stored in the text file.

Comparing the results to the Medialab dataset also used from the paper *Towards Human-Level License Plate Recognition*, there is a difference in accuracy by at least 38.52%.



**Decomposition of Responsibilities**

Min Sung Cha:

* Custom trained YOLOv2, a convolutional neural network for real-time object detection, with car images labeled with their corresponding bounding boxes utilizing custom weights and model. As license plate images were not available, the dataset was preprocessed in order to obtain the license plate bounding box for each image. Once the predictions were obtained the license plate prediction was cropped out and fed into the text recognition algorithm. Utilized darkflow and tensorflow.
* Wrote the entire code for HOG descriptor which preprocesses the image by calculating the gradient, orientation and magnitude of an image. The hog descriptor is calculated by dividing up the image into blocks and calculating the orientations of each gradient into 9 separate bins. Detections was implemented through cross-correlation of hog input image and hog template. Hog template was trained with 443 positive images from OpenALPR benchmark dataset which includes vehicles in Europe, Brazil, and US and 443 negative random images that do not contain the license plate. Once the detection was made cropped images of the license plate were saved into a file. Implemented using numpy, matplotlib, skimage, scipy, skimage, and cv2.

Chan Woo Park:

* Collected datasets that were used throughout the experiments. Ideally, the goal was to utilize the two public datasets that were used in the reference paper *Towards Human Level License Plate Recognition*: AOLP and Media Lab. However, the owners of AOLP had made their dataset private and required permission to share their dataset. The AOLP dataset was finally shared with the team, but not promptly enough to make through the experiments. Another type of dataset that was needed for the experiment was negative images. To highly train the neural network, there was a requirement of feeding it ‘noise data’ that did not contain the desired target images. However, using convolutional neural networks required us to have specific dimensions for these negative images as well. The solution was to use an API that generated a random image given the length and width as a POST request on [https://picsum.photos](https://picsum.photos/). By writing a Python script that made the correct API call for the right number of images, the team was easily able to get access to all needed negative images.
* Setup Google Cloud Vision account and integrate with project. The Vision API that is provided in Google Cloud services allows users to upload image data using basic API calls and get back information about the image. This included text recognition, which lifted the burden of training my own neural network from scratch. Using this transfer learning technique, I learned how to perform API calls to the Google Cloud Vision service and was able to extract text information from license plate images.

**References**

1. Mallick, Satya. “Histogram of Oriented Gradients” *Learn OpenCV*, (2016), https://www.learnopencv.com/histogram-of-oriented-gradients/
2. Marby, David. Yonskai, Nijiko. “Basic Usage” *Lorem Picsum*, https://picsum.photos/
3. Plate Recognizer, “Number Plate Datasets” *Plate Recognizer*, (2019), https://platerecognizer.com/number-plate-datasets/
4. Redmond, Joseph C. Farhadi, Ali. “YOLO: Real-Time Object Detection” *Joseph Chet Redmon*, (2016), https://pjreddie.com/darknet/yolov2/
5. Rosebrock, Adrian. “YOLO object detection with Opencv” *pyimagesearch*, (2018), https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/
6. Thtrieu. “darkflow - Translate darknet to tensorflow”, GitHub, (2017), https://github.com/thtrieu/darkflow/

1. Zhuang, J.& Hou, S. (2018), Towards Human-Level License Plate Recognition. *ECCV 2018,* Computer Vision Foundation