Autonomous Driving (Motion Prediction)

Fuchen Shen (fs2756) Zidong Guo (zg2417)

***Abstract*—In this project, we built a model for the Motion Prediction part of Autonomous Driving. We focused on using bird-eye-view information to predict the motion of Autonomous vehicles and surrounding agents, including pedestrians, cars, motorcycles, and bicycles. Our project is based on the Lyft level 5 dataset. We used the training, validation, and testing data in the data set and implemented our project with the visualization by partly utilizing the l5kit, which is an AV library for the Lyft level 5 dataset. We modified the training original Resnet training model and add our own methods by ensembling other Convolutional Neural Networks such as EfficientNet with the ResNet. We also tested multiple CNNs such as Resnet34, EfficentNet B3 to B5, and various Densenet to achieve the highest performance. We finally got an enhanced model which has higher computing speed and less loss based on the Metrics we use, the MSE loss, and Negative loglikelihood loss. With the final model, we implemented our model with test data and predicted the motion with trajectories of AVs in specific scenes such as crossroads with or without traffic lights. We also presented our results on a website using the Bootstrap 5 framework with python Flask and deployed it on Amazon Cloud so that everyone can view it through the provided URL.**

I. Introduction

A. Background

The idea of self-driving vehicles dates back much further than Google's research in the present day. In fact, the concept of an autonomous car dates back to Futurama, an exhibit at the 1939 New York World's Fair. General Motors created the exhibit to display its vision of what the world would look like in 20 years, and this vision included an automated highway system that would guide self-driving cars. While a world filled with robotic vehicles isn't yet a reality, cars today do contain many autonomous features, such as assisted parking and braking systems. Meanwhile, work on full-fledged autonomous vehicles continues, with the goal of making driving a car safer and simpler in the coming decades.[1]

With the evolution of Artificial Intelligence and its related technologies, people are giving their trust to machines with their reliable functionalities and stable performances. At present, many vehicles on the road are considered to be semi-autonomous due to safety features like assisted parking and braking systems, and a few have the capability to drive, steer, brake, and park themselves.

Autonomous vehicles are expected to bring with them a few different benefits, but the most important one is likely to be improved safety on the roads. The number of accidents caused by impaired driving is likely to drop significantly, as cars can't get drunk or high like human drivers can. Self-driving cars also don't get drowsy, and they don't have to worry about being distracted by text messages or by passengers in the vehicle. And a computer isn't likely to get into an accident due to road rage. A 2015 National Highway Traffic Safety Administration report found that 94 percent of traffic accidents happen because of human error: By taking humans out of the equation, self-driving vehicles are expected to make the roads much safer for all.[2]

B. Problem Formulation

There are three main components of the normal build of an Autonomous Vehicle, which we can see in figure 1. First is perception. Perception means we can let the AV knows what is surrounding it. This is usually done by sensors such as a gyro sensor, speed sensor, and camera. We decided to use the bird-eye view (figure2)as the information obtained from perception. The bird-eye view contains information about:

- Label of passengers (car, bicycle, and pedestrian).
- Road Information (pedestrian crossing and direction.
- Status of traffic light.
- Location and timestamp

All these information can be gathered into a single image by l5kit library

The next two tasks are prediction (determining what will happen next) and planning (deciding what the AV is going to do in the future). We're focusing on this second task, prediction.

Since AVs need to be able to make predictions about the future — something drivers do subconsciously all the time. Imagine an AV trying to turn left while another car is approaching from the opposite direction. In order for the AV to perform this maneuver safely, it needs to determine whether the other car will turn right or continue driving straight and interfere with the left turn. This is exactly what motion prediction is about.[3]

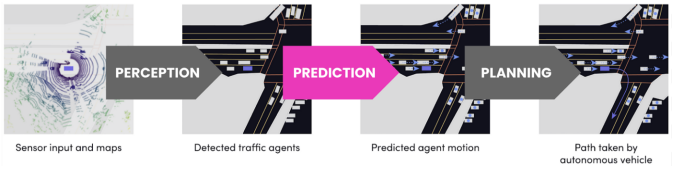

Figure 1. Three main components of AV stacks.

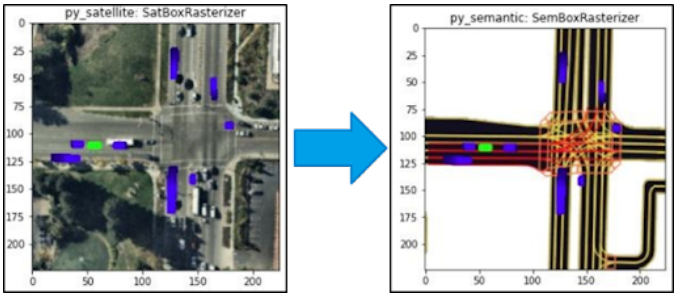

Figure 2. Bird-eye view

Today, models for motion prediction and planning are mainly built using rule-based systems. However, the future is uncertain and rules may not always scale well with uncertainty. As you add other agents into the mix, the number of rules and costs grow exponentially. A deep supervised learning approach could address that.Thus, there is a need for us to build an Autonomous Driving Motion Prediction model using Convolutional Neural Networks so that the prediction results can evolve within the input data and scaling to a large datasets.

C. Task Goals

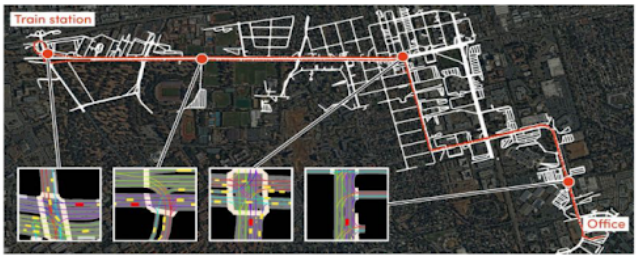

Figure 3. Sample of Motion Prediction

Our task goals contain follows:

- Predict nearby agents motions of the autonomous vehicle over next 5 seconds given their previous 1 seconds positions.
- Use Convolutional Neural Networks to build the model and trained it with provided dataset. Visualize the prediction result on semantic map.

To achieve these goals, we want to successfully implement following features:

- Examine the Dataset and filter the data.
- Turn prediction, planning and simulation problems into data problems and train them on real data.
- Use neural networks to model key components of the Autonomous Vehicle (AV) stack.
- Use historical observations to predict future movement of cars around an AV.

We want to use the neural network to create a prediction model which is trained by the Lyft Level 5 dataset. The model has functionality that can predict nearby agents motions of the autonomous vehicle over next 5 seconds given their previous positions. We use the trained prediction model to run with the dataset and compare with the real test data. Show the result as trajectories and images as Figure 4

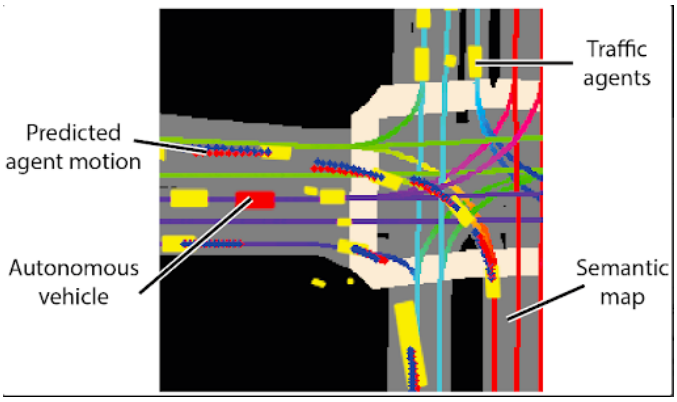

Figure 4. Project Overview

D. Challenges

During this project, we will also face many challenges. The most important is the Big data 3V, which are Volume, Velocity, and Variety. For Volume, since we are going to use a large dataset, we will have to handle a big number of data and our model should be able to process it. For Velocity, we aim to build a real-time motion prediction model to satisfy the requirements of Autonomous Driving because the prediction has to be very fast in reality, otherwise a traffic accident may be caused. For variety, our model also has to fit with various scenes such as highways, crossroads, etc. We also have to choose appropriate architecture for prediction speed and capability.

II. Data

A. Concepts and terms

To define the problem of motion prediction in self-driving cars and to comprehend the provided data, it is crucial to define and understand technical terms in the field:

- Semantic Map: Every type of modeling for self- driving cars requires high-resolution maps. The semantic maps are crucial for the autonomous vehicle from an operational and safety point of view. The semantic maps are providing a better understanding of AV's from their surrounding environment. The semantic maps consist of several layers such as road networks, building in 3D, lane layer, pathways and etc.
- Target Vehicles (TVs) are those vehicles which their behavior is important to be predicted by the model and the self-drive car is in interaction with them.
- Ego Vehicle (EV): A self-drive car which is moving while observing the behavior of the TVs
- Sensors: There are three major sensors installed on self-drive cars, and they are cameras, Radar, and LiDAR. All three sensors are used for generating Lyft Level5

dataset as they all provide crucial location and movement information. The cameras are available to display highly detailed images from the surrounding environment from the AV and this would enable the self-drive cars to detect the surrounding objects and classify them. Radar sensor is working based on the doppler effect, and it mainly serves the purpose of quickly getting the distance and contour of the surrounding objects from the different direction. LiDAR sensors, on the other hand, can provide higher resolution and accuracy comparing with Radar as it offers a full 360-degree map around the vehicle, generating data to localize the movement of the vehicle within the detected environment[4].

B. Dataset

Level 5 dataset, realized by Lyft, is the largest publicly released dataset of its kind. It includes over 55,000 human-labeled 3D annotated frames, a drivable surface map, and an underlying HD spatial semantic map to contextualize the data. This was collected by a fleet of 20 Lyft autonomous vehicles with LiDAR, RADAR and Cameras equipped along a fixed route in Palo Alto, California, over a four-month period. For this task, we focus on the motion prediction part of Lyft Level 5 dataset, which in total has about 80 GB of training dataset and 8.2 GB of validation dataset. It consists of 170,000 scenes, where each scene is 25 seconds long and captures the perception output of the self-driving system, which encodes the precise positions and motions of nearby vehicles, cyclists, and pedestrians over time.

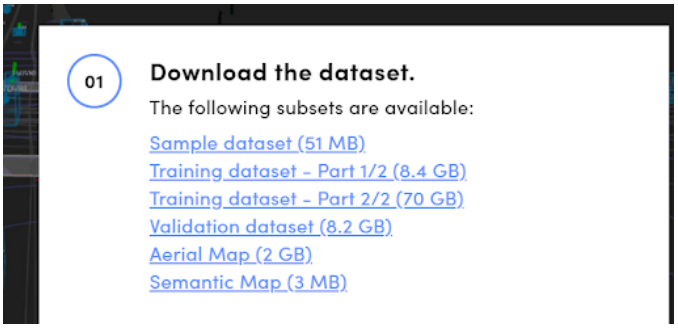

Figure 5. Lyft Level5 Dataset Overview

Lyft Dataset is organized in a hierarchical order of Scenes > Frames > Agents > Traffic Light Phase where each scenes are linked to different semantic map. Scenes includes references to its corresponding frames in terms of the start and end index within the frames array; frames includes all information that was observed at a given time stamp with regard to agents and traffic light; agents includes Ego and traffic agents, with entry describes the object in terms of its attributes such as position and velocity, and gives the agent a tracking number to track it over multiple frames and its most probable label.

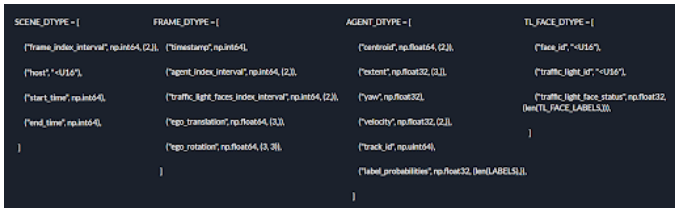

Figure 6. Labels contained in the Dataset

III. Method

A. Base Model

The Base Model we choose is the ResNet 34. ResNet is the model in which layers are in residual connections. As we can see in the figure shown below. The idea is used to reduce the vanishing gradient problem. If the gradient of F(x) which goes through two weight layers is near to zero, we still have a gradient because of another output x that bypasses the weight layers.

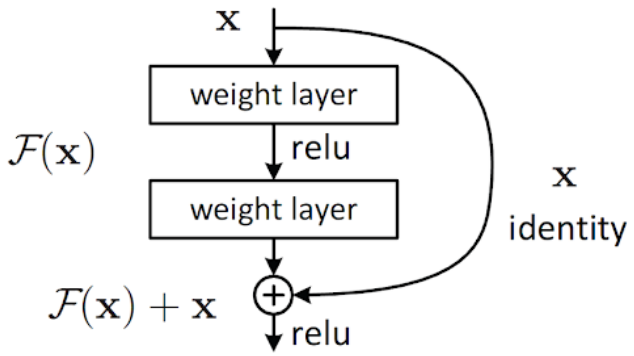

Figure 7. Residual Connection.

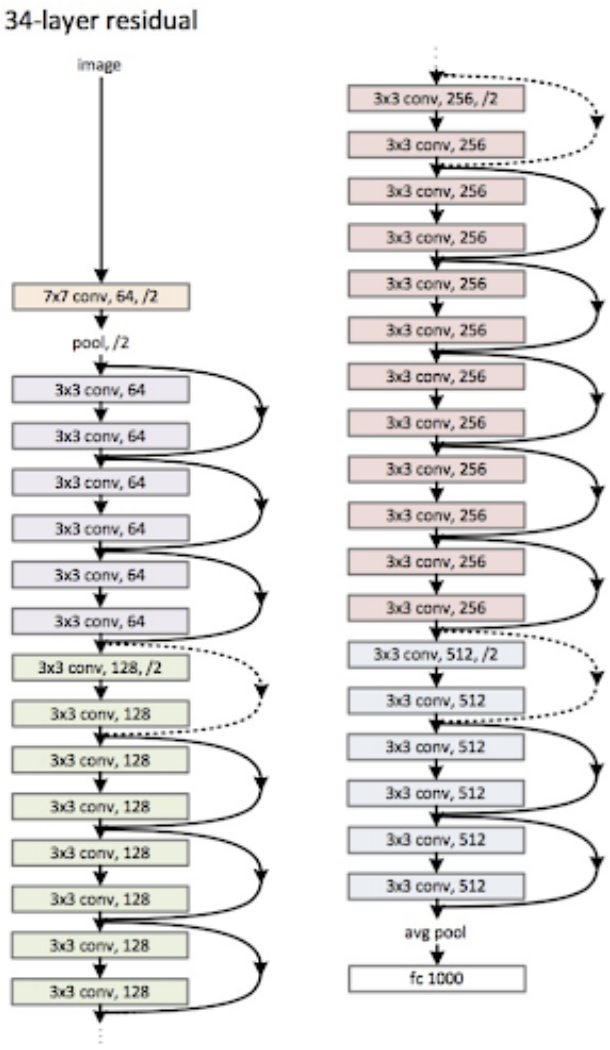

Figure 8. Resnet Structure.[4]

For the Resnet architecture, it initially contains 64 filters with kernel size of 7x7 followed by a max pooling layer. The methodology used in our base model is to use data frames of the previous 1 second to predict the motion of surrounding agents of the Autonomous Vehicle over the next 5 seconds. To achieve this, we will use 10 frames with a time distance of 0.1s as the input of the agent's position. Similarly, also 10 frames for the AV of the previous 1 seconds. With position we can calculate the motion of the agents. We also need the current position of both agents and AV, and three channels of the semantic map contains R, G, B. Thus, the total input will be 25 channels. For output, we will have 50 frames for each of the agents and AVs, the total number of output channels will be 100 channels (Houston). The input and the final layer of the Resnet model is replaced to the above dimension specification with added nn.conv2d and nn.linear. The second CNN architecture we use is EfficientNet. It is a pre-trained network and it is often used for multi target image classification.

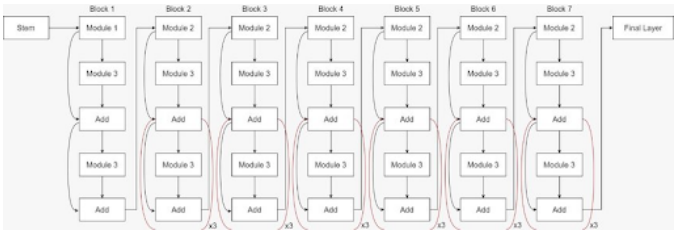

Figure 9. EfficientNet Architecture.

We tried multiple versions of the Resnet and Efficient, including Resnet 34, Resnet 101, Resnet 152, EfficientNet B3, EfficientNet B5. We compared these networks performance based on the figure shown below:

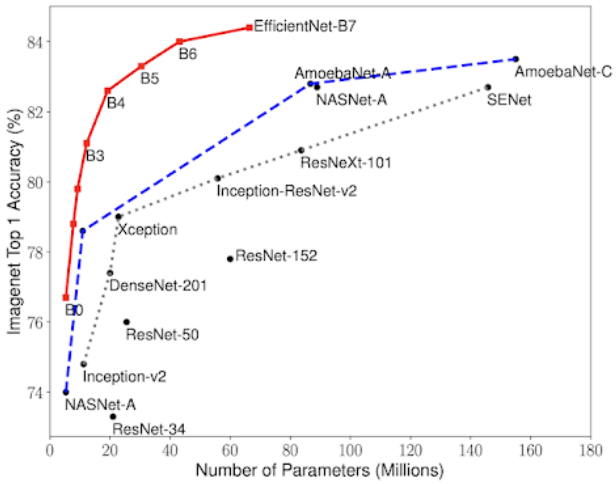

Figure 10. Model Comparison.

Since our model contains parameters near 20 to 40, EfficientNet will have better accuracy over resnet. However, since EfficientNet is more complicated than ResNet, the runtime for it will also be pretty long.

B. Metrics

We used two loss functions, the first is L2 loss, which is the mean square error loss. The second is the Cross-Entropy loss, which is the Negative log likelihood.

$$loss(x,y)=\frac{1}{n}\sum_{i=1}^{n}\begin{cases}0.5\times(y_i-f(x_i))^2, & if\ |y_i-f(x_i)|<1\\ |y_i-f(x_i)|-0.5, & otherwise\end{cases}$$

Figure 11. L2 loss.

$$\begin{cases}NLL\times penalize\times arccos(\frac{|dx|}{\sqrt{dx^2+dy^2}}) & arccos(\frac{|dx|}{\sqrt{dx^2+dy^2}})\geq 0.5\\ NLL\times penalize\times 0.5 & arccos(\frac{|dx|}{\sqrt{dx^2+dy^2}})<0.5\end{cases}$$

Figure 12. NLL loss.

Comparing L2 loss and NLL loss, we finally choose to use the NLL loss because it is better for classification of the multi-trajectory of our model.

Given the ground truth trajectory GT and K predicted trajectory hypotheses hypothesisk,k=1,...,K, we compute the likelihood of the ground truth trajectory under the mixture of Gaussians with the means equal to the predicted trajectories and the Identity matrix as a covariance. For every hypothesis we provide a confidence value c(k), such that kc(k)=1. This metric can be further decomposed into the product of 1-dimensional Gaussians, and we get just a log sum of the exponents. Note that in the evaluation metric we ignore the constant normalizing factors of the Gaussian distributions.

C. Ensemble

Currently, to reduce the generalization error of the prediction, we ensemble the models with stacking algorithm as shown in figure 8 where we trained and stored each model with the data separately and stack them with additional NN layers. In particular we first concatenate flattened prediction of the models with dim=(50 future frame * 2 xy * 3 modes + 3 conf.) * 2 models=606, then let the flattened data go through nn.linear(606, 4096), ReLU, nn.linear(4096, 303). The emsembling process tested through two stacking of different Resnet and Efficientnet architectures and the improvement in the negative loglikihood loss is significant as shown in the result section.

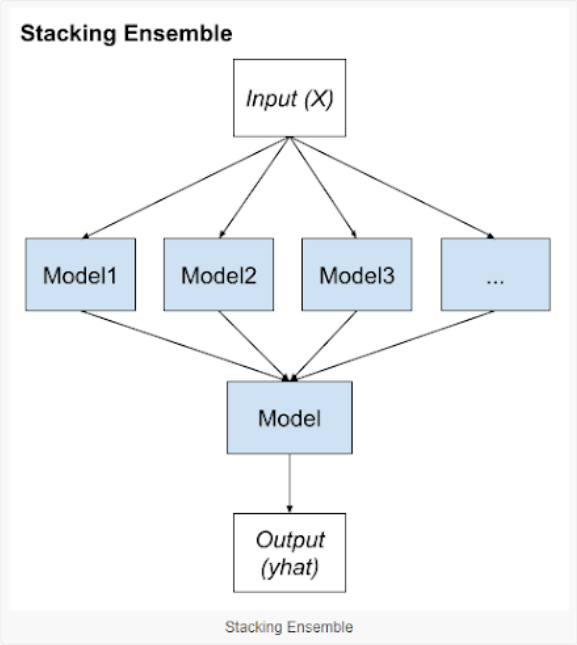

Figure 13. Ensemble through stacking.[5]

IV. Results

Since the data is very large and consume a lot of time to train the deep learning model, we limited the epochs and the batch size of the input data. The results are shown in figure 9 with the ensemble network of ResNet 34 and EfficientNet B3 having the lowest NLL Loss for validation dataset. However, the NLL Loss can be further reduced if enough iterations and epochs are made in the training step.

DL Model	Optimizer	Epoch	Batch_Size	Validation NLL Loss
Baseline ResNet 34	Adam	30	64	33.5
ResNet 50	Adam	30	64	33.16
ResNet 152	Adam	25	32	35.77
EfficientNet B3	Adam	25	32	28.31
EfficientNet B5	Adam	25	32	27.98
ResNet34+EfficientNet B3				22.47
ResNet34+EfficientNet B5				23.3

Figure 14. NLL loss results with different model.

We calculated the result on Google Colab platform since the GPU on our own computer is not powerful enough. The models were trained on a single Nvidia Tesla V100 GPU. . We adjusted the epoch and batch size because the GPU memory will exceed the limit if we have large batch size and epoch. Each model training time various greatly with upwards of 16 hours individually. Other models like DenseNet and Resnext were tested; however, the training will exceed the 32 GB VRAM available to the GPU. We can see the ensemble of Resnet34 and Efficientnet B3 has the best performance since it has the lowest NLL loss. The reason we did not use the DenseNet is due to the limitation of GPU memory since every batch size we tried will exceed the memory limit and crash the program. As result, we finally use the ensemble model of Resnet34 and EfficientNet B3 as our prediction model and implemented this model later.

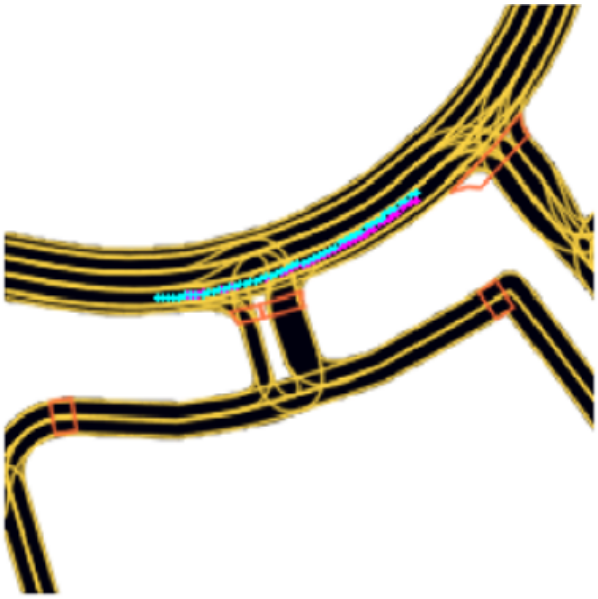

Figure 15. Visualization of ground truth (pink) and model prediction (teal)

We can also visualize results from the ego (AV) point of view for those frames of interest (the 100th of each scene) through the use of rasterizer and draw trajectory functions

provided within the l5 kit and setting the corresponding agent. The above figure demonstrates the visualizatin of one of the semantic map scene with the pink line indicating the ground truth of the AV's trajectory and the teal line indicating the predicted trajectory using the ensemble network of Resnet 34 and efficientnet B3. As we can see, although the two trajectories are not completely in line with each other, the movement of the AV stll falls within the same lane as shown.

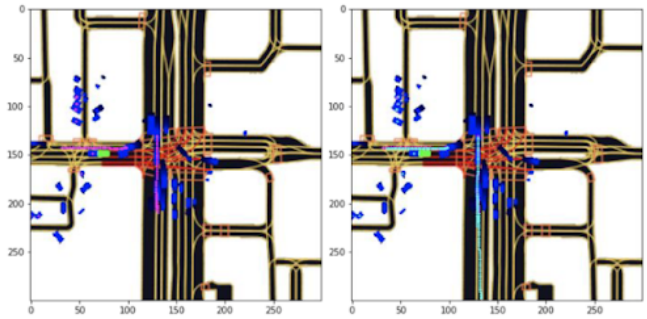

Figure 16. Visualization with traffic light (red horizontal green vertical)

With the implemented ensemble network, we can also visualize the trajectory of the agents when there are traffic signs label involved and as figure 11 demonstrates, the agents on the red light lane stop moving in the predicted frames while the agents on the green light lane moves forward.

V. Implementation

A. Implementation Architecture

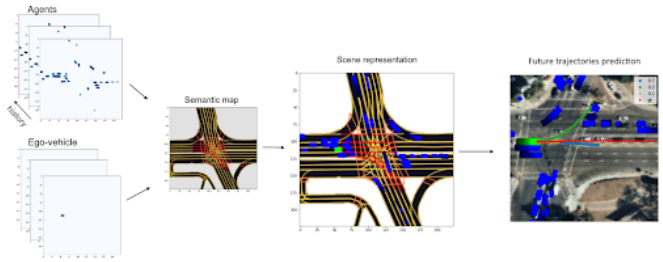

Figure 17. Implementation Architecture.

For implementation, we first load our Data with training dataset and finish running of our model training. Then, we load the trained model with testing dataset and get the prediction motions and trajectories of both agents and Egos in certain scenes. Then we load our semantic map and put the predicted data into the semantic map to obtain a scene representation. Finally, we run the rasterizer with build-in function provided by l5kit to visualize the future trajectory prediction of AVs.

B. Front End

The Web Application is written based on Python Flask, which is a micro web framework. The website page is written with Bootstrap 5 that conveniently offers free and open-source CSS code that directed at responsive web development. It contains HTML, CSS and JavaScript-based design templates for various interface component. The website we design shows a brief summary of our project regarding the background, dataset and methodology, as well as the visualization of predicted trajectories of the Ego vehicle as well as other travelling agents with the Ensemble NN we trained. In the visualization part, we show how the model predict the agents' trajectory for different scenario including Green Light, Red Light and curved lane. A video sequence is also generated on the website with motion prediction on an entire scene of moving agents by combining the movement predictions across multiple semantic maps and it demonstrates the model's capability of following the lane within the semantic maps as well as avoiding collisions between different agents.

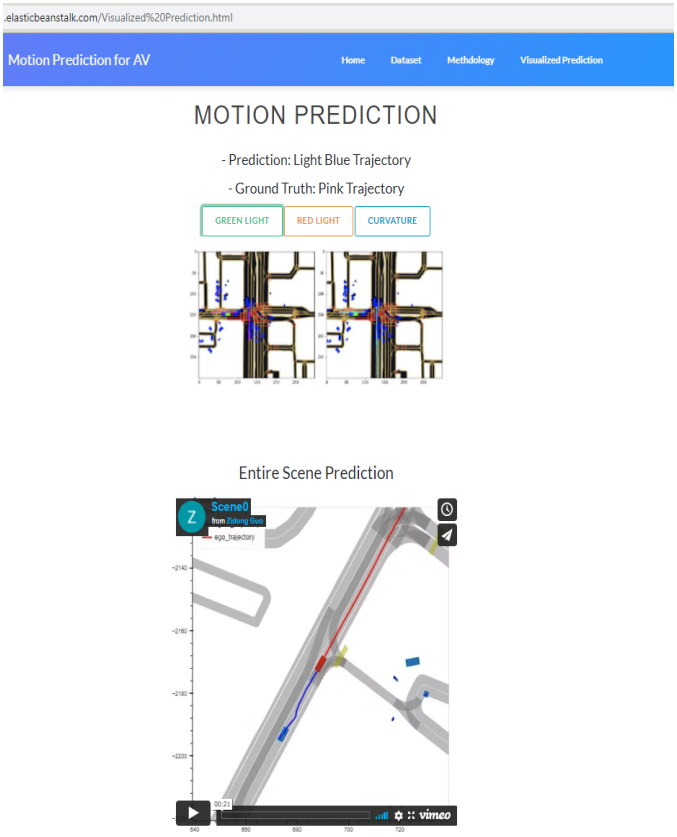

Figure 18. Web Scene Visualization.

In order to make the website available to everyone, the website is deployed on Amazon Cloud with the particular one being AWS Elastic Beanstalk as Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Our webpage can be accessed at:
http://motionprediction-env.eba-duhhamxy.us-east-1.elasticbeanstalk.com/index.html

VI. Conclusion & Future Work

In conclusion We created a better stacking algorithms motion prediction model based on l5 dataset. We constructed a motion picture for visualizing the entire scene. We made the user can visualize their AV's motion and trajectories.

For future work, we want to ensemble predicted results with multiple CNNs to achieve best performance and improve accuracy with model modification and other ensemble methods. Moreover, to speed up training, we will possibly resize/augment the input scenes. As suggested by the professor, we will also work on the implementation of a front end website showing the prediction of our algorithms based on Lyft's data and how the movement of the agents changes with the different parameters involved (traffic light, stop signs and other travel agents like pedestrian and cyclists).

VII. Github Repo

https://github.com/FuchenShen141/Autonomous_Driving_Lyft.git

References

[1] "History of the autonomous car," Apr 2020. [Online]. Available: https://www.titlemax.com/resources/history-of-the-autonomous-car/

[2] W. Brenner and A. Herrmann, "An overview of technology, benefits and impact of automated and autonomous driving on the automotive industry," *Digital marketplaces unleashed*, pp. 427–442, 2018.

[3] W. P. L. 5, "How to build a motion prediction model for autonomous vehicles," Oct 2021. [Online]. Available: https://medium.com/wovenplanetlevel5/how-to-build-a-motion-prediction-model-for-autonomous-vehicles-29f7f81f1580

[4] S. Paravarzar and B. Mohammad, "Motion prediction on self-driving cars: A review," *arXiv preprint arXiv:2011.03635*, 2020.

[5] "Prediction," Jul 2021. [Online]. Available: https://level-5.global/data/prediction/