CS230

Autonomous Vehicle Motion Prediction

Anirudh Kowdeed akowdeed@stanford.edu

Ratan Deep-Singh rdsingh@stanford.edu

Sophia Andrikopoulos sophiala@stanford.edu

1 Introduction

Autonomous driving is considered the future for smart vehicles. Safe autonomous driving greatly depends on the ability of autonomous vehicles to accurately predict the motion of other entities in traffic to reduce the likelihood of collision. While current technology only allows for partial automation and requires human supervision, fully autonomous vehicles would require a complete understanding of their surrounding environment. Understanding a dynamically changing environment is extremely complex. The combination of complimentary information, such as real-time data collected from Lidar and 3D cameras, driving physics, weather and road conditions, and traffic rules, can be used to estimate the motion trajectory of surrounding objects to better understand driving behavior. We aim to build upon Lyft's baseline model to more accurately predict the trajectories of other traffic participants.

2 Related Work

Much effort has gone into motion prediction for autonomous driving. Current approaches utilize Deep Learning frameworks such as convolutional neural networks (CNNs) and residual neural networks (ResNets). Revolutionary frameworks including AlexNet and MultiNet have been used for agent classification in autonomous driving, and more recently in agent motion prediction. Additionally, generative adversarial networks (GANs) have been used to predict pedestrian motion, and novel convolutional networks are often proposed to predict agent motion.

3 Dataset and Features

For this work, we use the Lyft L5 Prediction Dataset, containing more than 1,000 hours of driving data over 16,000 miles collected by Lyft's autonomous vehicles in Palo Alto, California, and a semantic map of the area. The dataset contains one frame for X timestamps. Each frame contains information about the autonomous vehicle at that timestamp, including its location, rotation, and a list of agents and traffic lights it detected at that timestamp. Agents refer to mobile entities the autonomous vehicle detected at that timestamp, such as cars and pedestrians, along with their position and unique agent ids that can be tracked between consecutive timestamps

The dataset is in the zarr format, consisting of a set of numpy structured arrays, representing 30-second chunks. The dataset is structured in four arrays: **scenes** capture information about the recording car, or "host" in a certain timeframe (frame index, host, start time, and end time); **frames** capture the information observed at a given timestamp, including the rotation and position of the recording vehicle, a reference to the agents the vehicle's sensors captured at that time, and a reference to the traffic light faces captured at that time; **agents** are objects the vehicle has detected, and contain the movement information about that agent, as well as a tracking number to associate the same agent over

CS230: Deep Learning, Spring 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

multiple frames, and an array of probabilities that map to different object classes; **faces** correspond to singular bulbs on traffic lights and contain distinct traffic light ids that map to the L5 semantic map, and the status of the specific bulb (on or off).

The chunked zarr dataset can be wrapped into an "AgentDataset", which inherits from the torch Dataset class. The L5 toolkit provides methods for loading the data into training, validation, and test sets. The validation and test sets are loaded in a "chopped" format so the models are unable to read forward to future data. As the dataset is well-established and is linked to a well-established repository, pre-processing, normalizing, or augmenting the data was deemed unnecessary.

4 Methods

For this project, we chose to focus on ResNet50, a 50-layer residual neural network pre-trained using imagenet. We used a Mean Squared Error loss function and the ADAM optimizer:

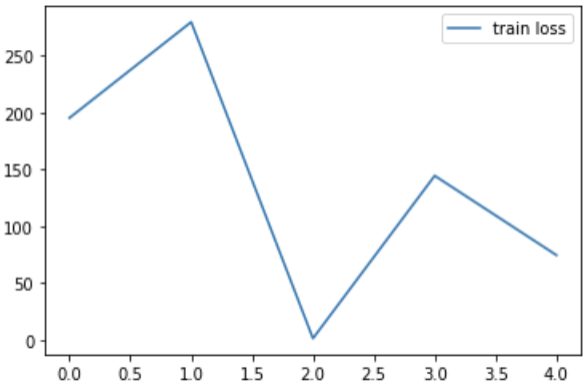
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Due to time and resource constraints, we chose to focus on finely tuning the hyperparameters of our baseline model rather than exploring more complex models.

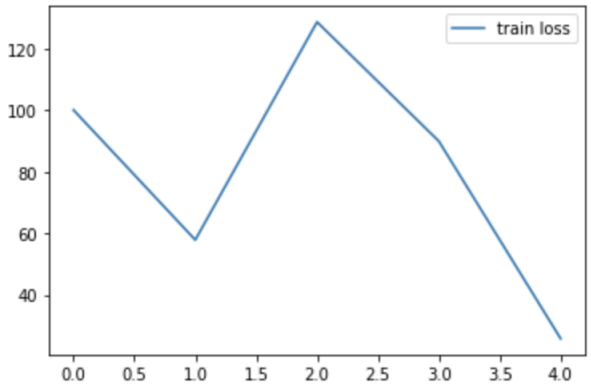
We evaluated our models by quantifying the loss and negative log likelihood of the ground truth data given the models' projections, aiming to minimize negative log likelihood. We use torchvision and Lyft's l5kit repository to build, train, and evaluate our models.

5 Experiments/Results/Discussion

We began with Lyft's pre-trained baseline model, using torchvision's default parameters, a learning rate of 0.001, and a batch size of 12. The baseline resulted in the following loss over five epochs and a negative log likelihood of 6218.98.



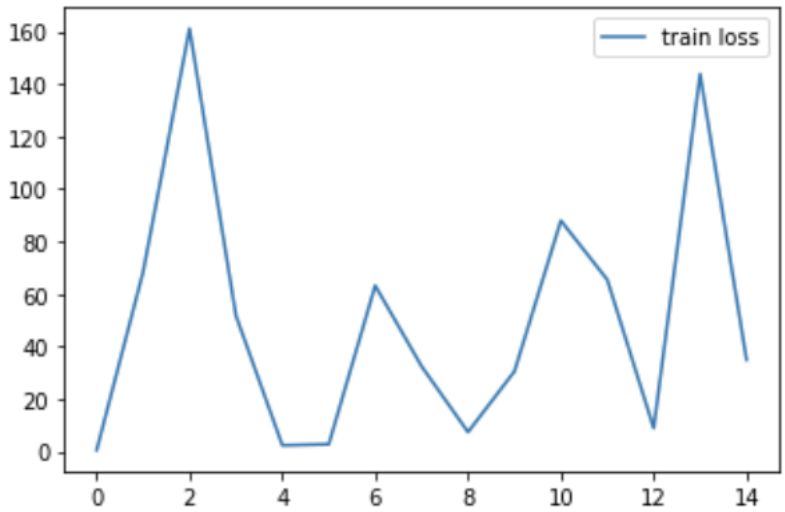
We also passed the same hyperparameters to an untrained ResNet50, resulting in the following loss over five iterations and a negative log likelihood of 6316.14.



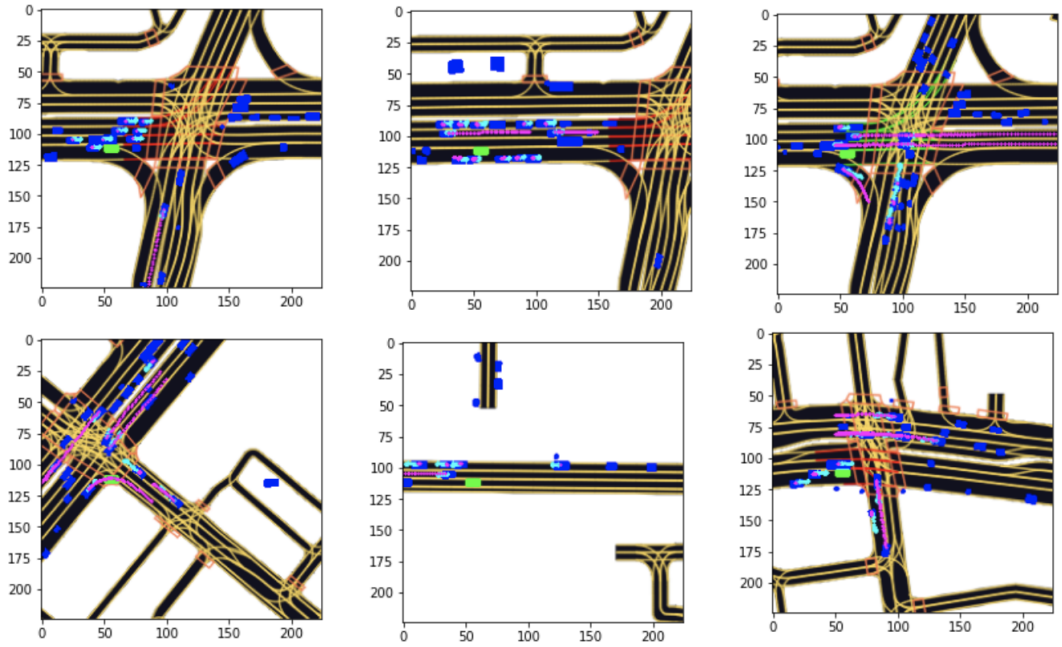
After determining that the pre-trained model resulted in an increased negative log likelihood, we chose to tune the hyperparameters of the pre-trained model and experiment with adding a dropout layer with $p = 0.5$ before the fully-connected layer. We ran the following experiments with the following results:

Batch Size	Epochs	Dropout	neg. LL
64	10	No	9230.3
64	10	Yes	12430.4
32	15	No	13283.2
32	15	Yes	72862.1
6	15	No	5852.8
6	15	Yes	4198.7
3	20	No	11145.0
3	20	Yes	9863.1

Our experiments resulted in a pre-trained ResNet50 model with $p = 0.5$ dropout before the fully-connected layer, using a batch size of 6 and training for 15 epochs. This resulted in a negative log likelihood of 4198.7, a significant decrease from our baseline. The loss for this model is shown below.



Some predicted trajectories of the final model are shown below. The predicted positions are shown in turquoise, and the actual positions are shown in pink.



6 Conclusion/Future Work

We reduced our loss and negative log likelihood significantly from baseline by using a pre-trained ResNet50 with a $p = 0.5$ dropout layer and a batch size of 6, training for 15 epochs. Decreasing batch size, increasing training epochs, and implementing dropout all improved the model to a certain extent, but there is still much to be done to improve the model. From the trajectories shown above, it seems our model functions well at lower velocities, but tends to under-predict agent motion. This is still sub-optimal, as un-predicted motion could have great consequences. With greater time and resources, we would like to further tune our hyperparameters, experiment with the addition and subtraction of layers within the ResNet50 framework, and explore other models, such as ResNet18, ResNet152, GANs, and MultiNet.

7 Contributions

All three partners conducted preliminary research, a literature review, and established the baseline model. All partners discussed the model, Sophia ran the experiments and typeset the report, and Ratan and Anirudh made the video.

References

1. Amirian, J. et. al. (2019) Social Ways: Learning Multi-Modal Distributions of Pedestrian Trajectories With GANs. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*
2. Collobert, R., Kavukcuoglu, K. & Farabet, C. (2011) Torch7: A Matlab-like Environment for Machine Learning. *BigLearn, NIPS Workshop*
3. da Costa-Luis, C. et al. tqdm: A fast, Extensible Progress Bar for Python and CLI. (2021) *Zenodo*
4. Deng, J., et al. (2009) Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*
5. Harris, C.R., Millman K.J., van der Walt, S. et al. (1995) Array programming with NumPy. *Nature* Springer Science and Business Media LLC.
6. He, K. (2017) Deep Residual Learning for Image Recognition.
7. Houston, J. et al. (2020) One Thousand and One Hours: Self-driving Motion Prediction Dataset. arXiv:2006.14480

8. Hunter, J.D. (2007) Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* IEEE COMPUTER SOC.
9. Paszke, A. et al. (2017) Automatic differentiation in PyTorch.
10. Teichmann, M. et al. (2018) MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving. *IEEE Intelligent Vehicles Symposium (IV)*
11. Wu, B. et. al. (2017) SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. *EEE Conference on Computer Vision and Pattern Recognition*