CSE 251B Project Final Report

Tim Kraemer
PID: A69032843
tikraemer@ucsd.edu

Mathew Raju
PID: A69032602
mraju@ucsd.edu

Angus Yick
PID: A16602157
anyick@ucsd.edu

Github Repository Link: Here

1 Introduction

The deep learning task focuses on predicting the future path of a self-driving car using past movement data from itself and nearby vehicles. Using the Argoverse 2 dataset, which includes thousands of driving scenes with multiple agents, our goal is to forecast the ego car's trajectory 60 time-steps into the future based on the first 50 time-steps of input and agent interactions.

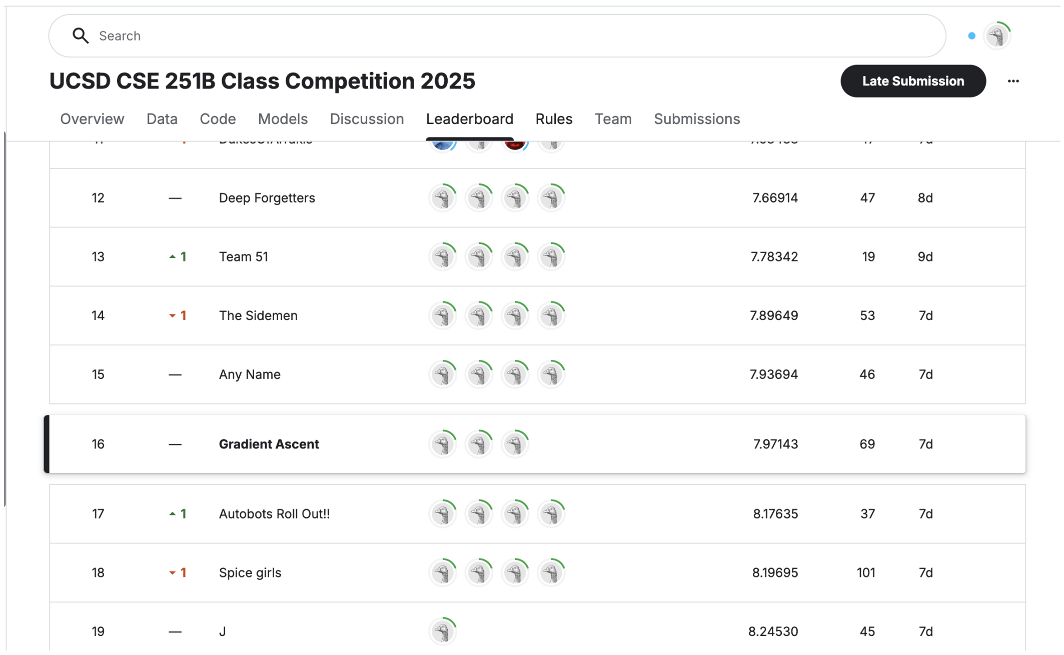
This task is important because it helps autonomous vehicles make safer, more reliable decision on the road. As self-driving cars become more common, improving their ability to predict motion will reduce accidents, protect pedestrians and passengers, and make roads safer for everyone.

As a starting point, we are provided with introductory code featuring a basic, "bare-bones" RNN LSTM model for sequential trajectory prediction. While effective as a baseline, this approach suffers from several key limitations, including: **lack of spatial awareness**, **poor modeling of multi-agent interactions**, and **difficulty with long-term prediction**. To address these challenges, our contributions focus on the following:

- **Contribution A**: Incorporation of a *Spatial Attention Mechanism* to model and extract information from agents most relevant to the ego vehicle.
- **Contribution B**: Implementation of a *Temporal Attention Mechanism* to identify and emphasize the most informative time segments within the first 50 time-steps.
- **Contribution C**: Careful use of LSTM modules as both encoders and decoders for the ego vehicle and surrounding agents, enabling better sequential understanding and prediction.

1.1 Leadboard Position

Our group **Gradient Ascent** is ranked **16th** in the overall Kaggle competition. This puts us within the top quartile of the overall class with a final validation MSE of **7.97**. A screenshot of our final leaderboard position can be seen in Figure 1.



UCSD CSE 251B Class Competition 2025

12	—	Deep Forgetters	7.66914	47	8d
13	▲ 1	Team 51	7.78342	19	9d
14	▼ 1	The Sidemen	7.89649	53	7d
15	—	Any Name	7.93694	46	7d
16	—	**Gradient Ascent**	7.97143	69	7d
17	▲ 1	Autobots Roll Out!!	8.17635	37	7d
18	▼ 1	Spice girls	8.19695	101	7d
19	—	J	8.24530	45	7d

Figure 1: Final team ranking in Kaggle competition

2 Related Work

Sequence Modeling with LSTMs: Long Short-Term Memory (LSTM) networks have been a cornerstone of sequence modeling, particularly for tasks involving temporal dependencies. Hochreiter and Schmidhuber [1] introduced LSTMs to address the vanishing gradient problem in traditional RNNs, enabling better learning of long-range dependencies. This was introduced almost 28 years ago, and since countless subsequent work has been done primarily in the field of trajectory prediction tasks. For example, work by Hyeon Park et al. (2018) describes methods of utilizing for future trajectory sequence prediction of neighboring vehicles an encoder-decoder LSTM architectures to analyze past trajectory data [2]. Additional work has also shown the promising results of using LSTM variants in other agent interaction prediction problems, such as by Bang Cheng et al. (2018), who shows the usage of *Social-Grid LSTM*, an RNN-based architecture to predict pedestrian trajectories [3]. With established significance, in our paper we describe the motivation behind utilizing LSTMs as an encoding-decoding force for self-driving vehicle trajectory prediction.

Attention Mechanism for Sequential Data: The integration of attention mechanisms with LSTMs has significantly improved performance in sequence-to-sequence tasks. Bahdanau et al. (2015) [4] first proposed additive attention for neural machine translation, enabling models to focus on relevant parts of the input dynamically. Vaswani et al. (2017) [5] later introduced multi-headed self-attention in the Transformer architecture, which has since been adapted for hybrid LSTM-attention models. Specifically for self driving vehicle problems, spatial-temporal attention mechanisms have shown countless times to elevate the long-term prediction accuracies of LSTMs through accurate modeling of spatial relationships in neighboring vehicles and temporal significance, such as by Jiang et al. (2022) [6].

3 Problem Statement

The Argoverse 2 dataset is a complex multi-dimensional dataset containing multi-agent trajectory features for self driving predictions. With 10000 training scenes and 2100 test scenes, we are tasked with making a 60 time-step trajectory prediction for the ego self-driving car, given the first 50 time-steps of data given. The dataset contains 50 distinct agents, each with their own (x, y) positions, velocity, heading, and agent type encoding.

More formally, the training set is represented as a 4-dimensional vector:

$$
\begin{aligned}
&D \in \mathbb{R}^{N \times A \times T \times F} \\
&N = 10000 \text{ (Number of training scenes)} \\
&A = 50 \text{ (Number of agents per scene)} \\
&T = 110 \text{ (Number of time-steps per agent)} \\
&F = 6 \text{ (Number of features per agent at each time step)}
\end{aligned}
$$

Each feature vector f for each agent a at each time step t for each scene d has the feature vector $f_{a,t,d} = [x, y, v_x, v_y, \theta, p]$ where:

$x, y =$ The coordinate position of the agent
$v_x, v_y =$ The directional velocity of the agent
$\theta =$ The heading angle of the agent
$p =$ The agent type
$p \in$ [vehicle, pedestrian, motorcyclist, cyclist, bus, static, background, construction, riderless bike, unknown]

Our model then predicts as output $60 \times [x, y]$ positions of the ego car's next 60 time steps. As far as pre-processing goes, we did not perform anything additional that the starter code provides to us, which includes data augmentation (randomized rotation of the (x, y) positions & randomized inversions over the horizontal axis), centering of the ego car at $(0, 0)$ and other agents relative at $t = 49$, and normalizing all other agent features other than `agent_type`. One additional pre-processing step we performed was only selecting the first 20 agent in ascending indices for the following reason:

Through an extensive data exploration phase, we noticed that most agents that tend to be the closest to the ego car happen to have smaller indices. We can see this through a distribution of average distance from the ego car at time $t = 49$, shown in figure 2:

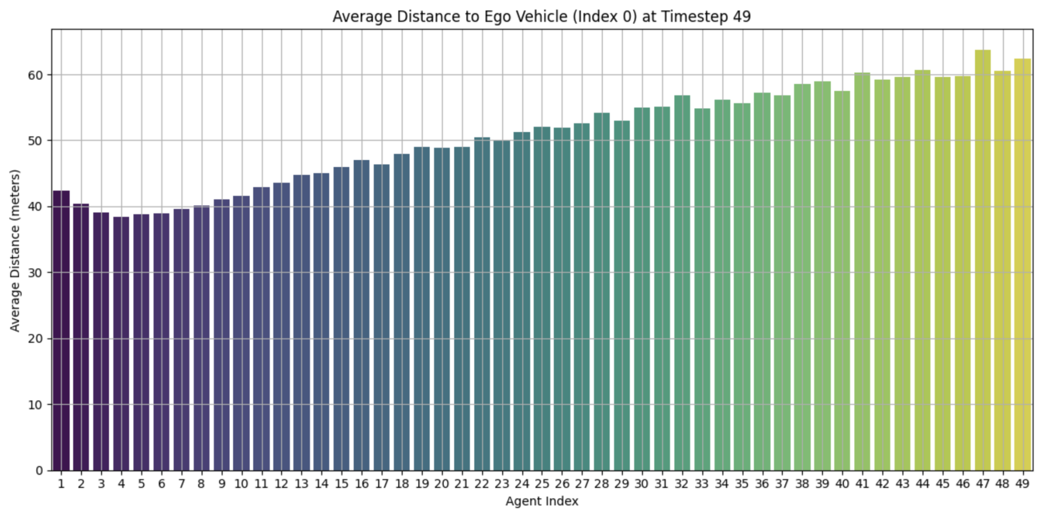


Figure 2: Distribution of average agent car distance from Ego car per index

We also discovered that the majority of vehicles that start far away from the ego car at $t = 0$ and then move closer at $t = 49$ also occur towards the front of indices, as seen in figure 3:

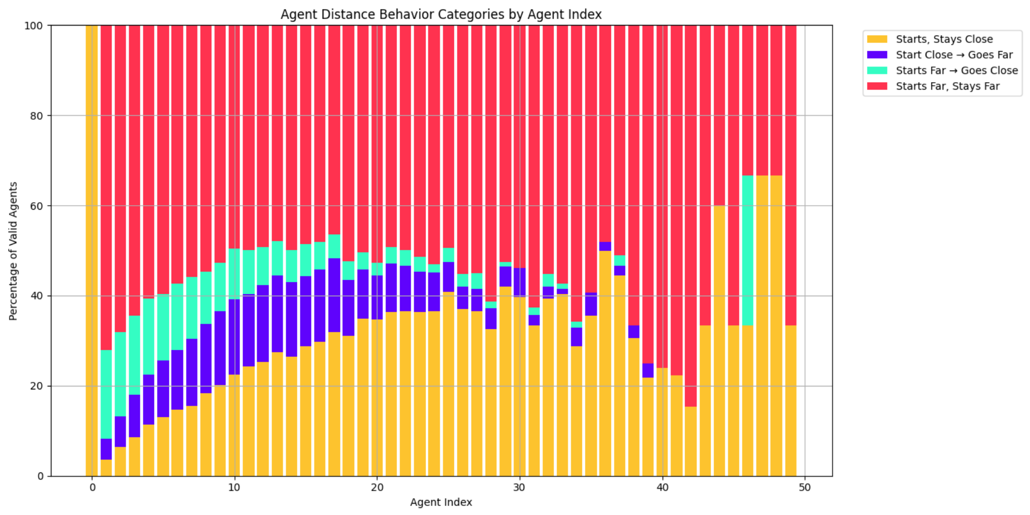


Figure 3: Distribution of agents before and after distances relative to Ego car per index

Through extensive trial and error, we concluded that subdividing and only considering the first 20 agents in the dataset capitalizes on this data distribution, and resulted in the lowest test and validation MSEs. For more consistent training, we split the training set of 10000 scenes into a training and validation set, following a standard 90% training, 10% test set sizes.

4 Methods

In this section, we present the technical details of our approach for trajectory prediction of the ego car in the Argoverse 2 dataset utilizing both spatial and temporal attention mechanisms. We define the prediction task as a sequence-to-sequence learning problem, where our goal is to predict a 60 time-step horizon given the initial 50 time-steps of data in a given scene. Our model is trained to minimize the average prediction error over the forecast horizon, using the loss function described in the following section.

4.1 Objective Function

To determine accuracy, we utilize the `Mean Squared Error (MSE)` loss function for trajectory regression, which quantifies the average squared difference between the predicted and ground truth trajectories in any given training scene. Mathematically, MSE is defined as:

$$\mathcal{L}_{MSE} = \frac{1}{T} \sum_{t=1}^{T} ||\hat{y}_t^{ego} - y_t^{ego}||^2 \tag{1}$$

where:

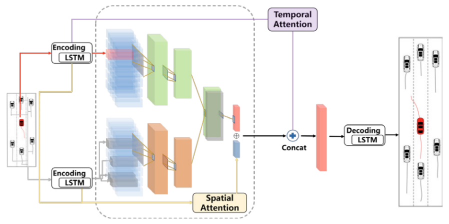
$\hat{y}_t \in \mathbb{R}^2$ represents the predicted (x, y) position of the ego car at time t

$y_t \in \mathbb{R}^2$ represents the ground truth (x, y) position of the ego car at time t

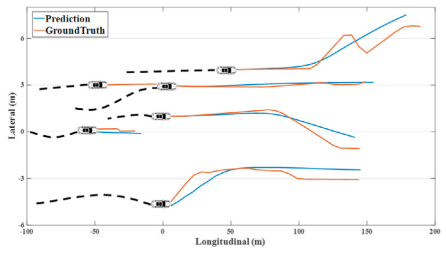
T is the length of the prediction horizon

4.2 Background and Motivation

Our model design is loosely based on the trajectory prediction model defined in *Intelligent Vehicles Trajectory Prediction with Spatial and Temporal Attention Mechanisms* by Qingyu Meng, et al. 2021 [7]. Originally developed for freeway driving scenarios, this approach integrates two attention mechanisms: (1) Spatial Attention, which captures the interactions between the target vehicle and surrounding vehicle using **cosine similarity** to weight the influence of neighboring vehicles based on their hidden states. (2) Temporal Attention, which focuses on key historical time steps of the target vehicle's trajectory, employing **CNN filters** to extract time features, and generate a context vector for important historical moments. A detailed diagram of the paper's model is seen in figure 4a, with example trajectories in figure 4b.



(a) LSTM network structure with spatial, temporal attention (Source: Meng et al.)



(b) Vehicle trajectory prediction results (Source: Meng et al.)

4.3 Model Architecture

We define our own predictor with the following components influenced by this initial work:

1. **Two LSTM encoders**: One for the ego vehicle, the other for all other agents.
2. **Sequential linear attention mechanism**
3. **1 dimensional convolution temporal attention mechanism**
4. **An LSTM decoder**
5. **A final linear singular-timestep predictor**

Figure 5 provides a clear visual representation of our developed model.

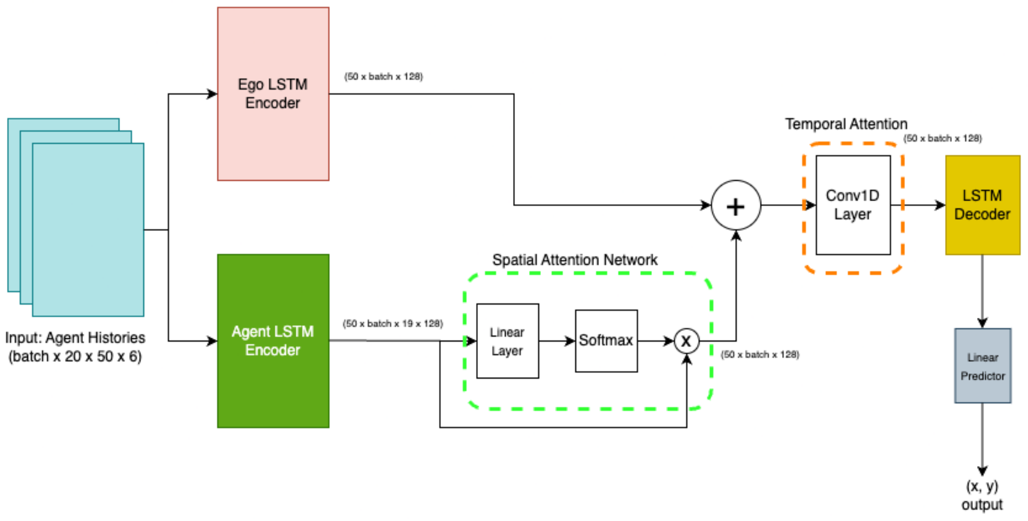


Figure 5: High level diagram of LSTM with attention model

4.3.1 LSTM encoders

The model begins by encoding the time-step information of the ego vehicle and all surrounding agents using separate LSTM encoders. Each encoder processes six input features per agent at each time step and transforms them into a hidden representation of size of 128. This results in an output tensor of shape `(50, batch_size, hidden_dim)` for the ego vehicle, and `(50, batch_size * num_agents, hidden_dim)` for the other agents.

```
# Ego Encoder
self.ego_lstm = nn.LSTM(input_size=6, hidden_size=encoder_hidden)
# Other Agents Encoder
self.nbr_lstm = nn.LSTM(input_size=6, hidden_size=encoder_hidden)
```

4.3.2 Linear Spatial Attention Mechanism

To identify which agents are most relevant to the ego car's decision making, we apply a spatial attention mechanism that allows the model to focus on the most important surrounding agents. This is designed using a sequential residual linear network that maps the encoded 128 features per agent down to 1 attention score through a hidden layer of size 64 with the `ReLU()` activation function.

These raw attention scores are then passed through a `softmax` function to produce normalized attention weights across all agents. Each agent's encoding is weighted by its corresponding attention score, and the resulting weighted agent features are aggregated. Finally, this attended agent representation is concatenated with the ego vehicle's encoded trajectory to form a joint context vector for downstream prediction.

```
# Spacial Attention Layer
self.spatial_attn = nn.Sequential(
            nn.Linear(128,64), nn.ReLU(), nn.Linear(64,1))
```

4.3.3 Temporal Attention Mechanism

To capture temporal dependencies in the initial 50 time steps, we apply a 1-dimensional convolutional layer as a temporal attention mechanism. The intuition is to allow the model to learn which time steps in the input sequence are most informative for predicting the future trajectory.

Specifically, the concatenated ego encodings with the agent encodings result in a combined embedding of size 256 per time step. This embedding is passed through a 1D convolutional layer with an input channel size of 256 into an output channel size of 128 using a kernel of size 5 with a padding size of 1, resulting in an output vector in the shape `(batch_size, 128, 50)`. The convolution operates over the temporal axis, letting the model assign explicit importance to specific time steps.

```
# Temporal Attention Layer using a 1D Convolutional Layer
self.temporal_attn = nn.Conv1d(256, 128, kernel_size=5, padding=1)
```

4.3.4 LSTM Decoder with Final Linear Predictor

Finally, to generate the future trajectory, we iteratively predict each of the next 60 time-step positions (x, y) using an LSTM decoder. This process begins by taking the final hidden embedding of size 128 from the encoder at time step $t = 49$, which serves as the initial input to the decoder. The decoder is an LSTM with a hidden size of 128, matching the architecture of the encoder.

At each decoding step, the LSTM outputs a 128-dimensional feature vector, which is passed through a residual linear network consisting of a hidden layer of size 64 with a `ReLU` activation, followed by a final linear layer that maps the features down to a 2D (x, y) coordinate. This predicted position is then used as the input for the next decoding step, forming an auto-regressive feedback loop that continues for all 60 future time steps.

```
#Decoder
self.decoder = nn.LSTM(input_size=128, hidden_size=decoder_hidden)
self.output_layer = nn.Sequential(nn.Linear(128,64), nn.ReLU(), nn.Linear(64,2))
```

4.3.5 Multi-Head Attention Mechanism (Add-On)

Later on, we added multi-head attention to capture diverse interaction patterns between the ego vehicle and surrounding agents. Unlike the original single-head spatial attention, this mechanism projects the 256-dimensional encoded features into 8 parallel attention heads (32-dimensional each), allowing the model to jointly attend to different types of agent relationships (e.g., proximity, velocity matching, and right-of-way). Each head computes scaled dot-product attention between the ego vehicle's features (queries) and all agents' features, with the final attended representation obtained by concatenating and projecting all heads' outputs.

```
#Multi-head attention
self.query = nn.Linear(128, 128)
self.key = nn.Linear(128, 128)
self.value = nn.Linear(128, 128)
attn_weights = torch.softmax(
(queries @ keys.transpose(-2,-1)) / math.sqrt(32), dim=-1
attended = (attn_weights @ values)
```

4.3.6 Learning Rate Augmentations

The training process combines warmup and plateau-based scheduling to stabilize optimization. For the first 10 epochs, the learning rate linearly increases from 0 to 1e-3, preventing early gradient instability. Subsequently, the ReduceLROnPlateau scheduler monitors validation loss, reducing the learning rate by 50% whenever the loss fails to improve for 10 consecutive epochs.

```
#Learning Rate Scheduling
warmup_epochs = 10
def lr_lambda(epoch): # Linear warmup
  return min(1.0, (epoch + 1) / warmup_epochs)
scheduler_warmup = LambdaLR(optimizer, lr_lambda)
scheduler_plateau = ReduceLROnPlateau(optimizer, mode='min', factor=0.5, patience=10)
```

4.4 Design Choices and Justification

We chose to build our model around an LSTM-based architecture due to its simplicity and well-established effectiveness in sequence modeling tasks. LSTMs are capable of capturing temporal dependencies in trajectory data, making them a reasonable baseline for motion forecasting.

However with such complex and high dimensional data, its generally impossible to expect even a highly tuned LSTM to be able to separate the importance of every interaction between all agents at every time step. This motivated our decision to incorporate attentional mechanisms, which sole purpose is to highlight the most important information between agents and within the time domain.

Additionally, one of our most important design decisions was to only apply the attention mechanisms to a subset of the agent-space in the training dataset. This significant filtering that we performed during pre-processing both stabilized learning, as well as sped up training.

5 Experiments

In this section, we delve into the experimentation and results that we collected throughout the course of the Kaggle competition. Below we walkthrough the incremental baseline phases we tested, the evaluation metrics on what is considered an "adequate" model, as well as our results and ablations.

5.1 Baselines

To understand the evolution of our approach, we delve into the various baselines in the order they were developed, each building upon the insights and limitations of the one before. Throughout the scope of the project, these are the baselines that have helped shape the given model today.

- **Linear Regression**
  We begin with a simple *Linear Regression* baseline, which takes the ego vehicle's initial (x, y) positions over the first 50 time steps and predicts a (60×2) output representing future positions over a 60-step horizon.
- **Residual Linear Network, (MLP)**
  The input to the Residual network consists of a flattened vector representing the observed positions of only the ego vehicle's 6 features over all initial 50 time steps. This vector is mapped to the 60 (x, y) predicted positions through 3 hidden layers in order of size `1024,`

`512, 256, 128`, with the accompaniment of the `LeakyReLU()` activation function, and a neuron dropout rate of `0.1` at each hidden layer.

The MLP treats the trajectory as a purely spatial pattern and does not explicitly account for the temporal nature of the sequence or the surrounding context, such as road layout or interactions with other agents. While this limits its capacity to model real-world driving dynamics, the MLP provides a fast, lightweight baseline that helps evaluate the added value of more sophisticated models that incorporate temporal dependencies or scene understanding.

- **Convolutional Neural Network (CNN)**

  The convolutional baseline improves on residual networks by better capturing local spatial patterns. It consists of: (1) two sequential 1D convolutional layers with kernel size 3—mapping 6 input features to 32, then `64` channels—with batch normalization `(size 32)`, and (2) a linear residual network for decoding.

  The residual decoder includes a hidden linear layer of size `256` with ReLU activation and `20%` dropout. While more expressive than an MLP, this model still lacks long-range temporal and contextual modeling, serving as a strong yet limited baseline.

Given the following baselines above we observed that we need not only spatial information, but it's relation to time. This prompted us to utilize models that could incorporate sequential data to work with time and space. This discovery lead to the utilization of **LSTM models**

5.2 Evaluation

For evaluating the performance of our model, we primarily utilized **Leaderboard Positioning**, Mean Squared Error **(MSE)** and Mean Absolute Error **(MAE)** as our metrics. MSE penalizes larger errors more significantly by squaring the residuals, making it particularly sensitive to outliers and useful when large deviations are undesirable. MAE, on the other hand, provides a more interpretable metric by computing the average of the absolute differences between predicted position and actual position, offering a direct measure of prediction error.

5.3 Implementation Details

Our approach formulates trajectory prediction as a sequence-to-sequence optimization problem, minimizing the mean squared error between predicted and actual trajectories with L2 regularization $\lambda = 10^{-4}$. The bulk of development and training was done on Kaggle's incorporated virtual notebook platform with available **P100** and **T4** GPUs. Although, some development was done locally on personal Apple M2, and Intel core-i7 laptops, but this development was mainly reserved for baseline models. Using Kaggle's P100 computational GPUs, we averaged around `30 seconds` per epoch, with a total computation time until early stop of around `45 minutes`.

For simplicity, we display our final hyper-parameter values for our LSTM + Attention model in table 1 below:

Table 1: Final tuned model hyperparameters

Hyperparameter	**Value/Type**
Learning Rate	1e-3
Weight Decay	1e-6
Optimizer	AdamW
Scheduler	ReduceLROnPlateau
Warmup Epochs	10
Stopping Patience	20
Learning Rate Decay Step Size	10
Epochs	200, $\sim$130 early stop
Batch Size	32
Encoder/Decoder hidden layer size	128
Random Seeding	50
Temporal Attn Kernel	5
Spatial Attn Hidden Layer Size	64

Due to the high cost and time constraint to training our model compared to the baselines, we had to be wise with which hyper parameters we experimented with to avoid wasting training time with diminishing returns. For the training loop parameter values, most of testing was done through randomly picking values and observing the convergence of the validation MSE. Hyperparameter values such as **LR** proved to be the most optimal around 1e-3, as any smaller would avoid convergence and larger values led to exploding gradients. Differences between `Adam`, `AdamW`, and `RAdam` proved minimal, so we concluded with sticking with `AdamW` for both baselines and the LSTM attention model.

As for internal parameters such as **layer sizes**, our strategy was simply to increment values until we observed diminished returns in the validation MSE. The most significant step we observed was increasing the kernel size for our temporal attention mechanism from $3 \to 5$, with and observed MSE decrease of ~ 0.5.

Additionally, we applied a gaussian smoothing to the generated trajectories for the test set using `gaussian_filter1d` from the `scipy.ndimage` library. This was done as an attempt to smooth out any noise and jagged prediction lines, and resulted in the test MSE being reduced around **0.03**.

5.4 Results

Here we present the qualitative and quantitative analysis of our baselines and tuned LSTM + Attentional Mechanism model, providing **training** and **Mean Squared Error** (MSE) and **Mean Absolute Error** (MAE). Additionally, we provide visualized trajectory predictions compared to their associated ground truth trajectory, to provide insights on what aspects of driving and decision making the ego vehicle performs well.

Table 2 displays the training MSE/MAE **(TMSE, TMAE)** and validation MSE/MAE **(VMSE, VMAE)** of the iterative baselines as well as our LSTM model that incorporates temporal and spatial attention. All models displayed utilized the same hyperparameters list in table 1. The linear regression baseline by far displayed the worst performance, plateauing at a validation MAE of ~ 300. However, the MLP, CNN and base LSTM network showed similar losses, most likely due to them only considering the ego agent. Any additional agent that was introduced in the input vector led to considerable worse loss.

Model	**TMSE**	**TMAE**	**VMSE**	**VMAE**
Linear Regression	5529.0127	43.1460	259280.5526	296.1325
Residual Network (MLP)	0.3601	0.3317	14.1186	1.8869
Convolutional Network (CNN)	0.3220	0.3166	14.0039	1.9934
Base LSTM	0.3279	0.3300	10.7563	1.6875
LSTM + Attention	0.1232	0.1802	7.6192	1.3600
Multi-headed Attention + LR augments (LSTM)	0.1442	0.1564	7.5843	1.2689

Table 2: Train MSE (TMSE), train MAE (TMAE), validation MSE (VMSE), and validation MAE (VMAE) for different models

Visualizing the decision-making of the ego vehicle, we plot the self driving car's predicted horizon trajectory tangential to the ground truth trajectory for time-steps $t = t_i > 49$ in figure 6

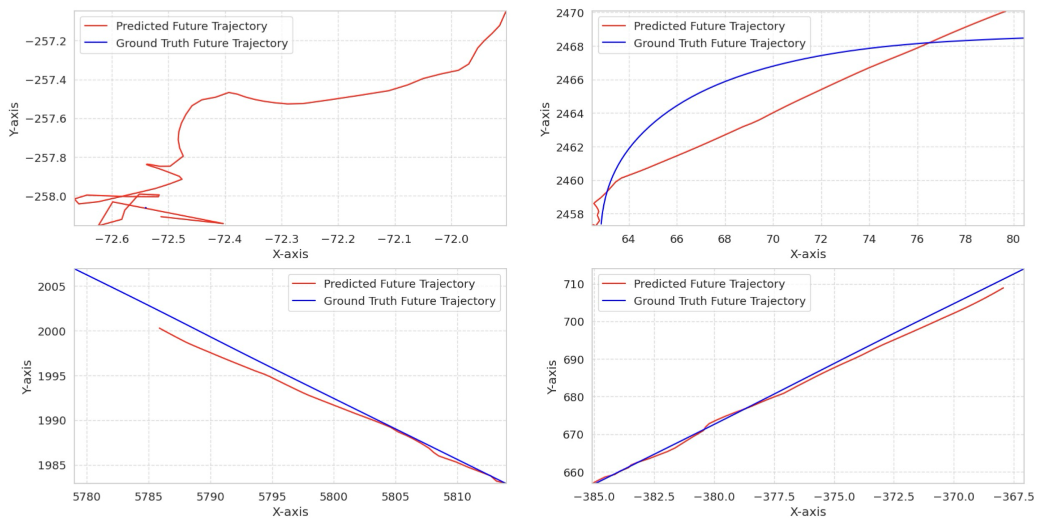


Figure 6: Ego vehicle predicted trajectory vs. ground truth trajectory for 4 validation samples

5.5 Ablations

In this subsection, we investigate which components of our LSTM+Attention model contribute the most to accurate trajectory prediction. Table 3 depicts the resulting tain MSE **(TMSE)** and validation MSE **(VMSE)** from removing different mechanisms from the model, as well as the **VMSE** difference compared to no change to the model.

Model	**TMSE**	**VMSE**	**ΔVMSE**
No change (Multi-head)	0.1442	7.5843	0
Removal of multi-head	0.1232	7.6192	+0.0349
Removal of temporal mechanism	0.0988	7.6244	+0.0401
Removal of spatial mechanism	0.1344	7.9030	+0.3187
Removal of spatial and temporal	0.2564	9.3254	+1.7411

Table 3: Effects on training MSE and validation MSE on different model ablations

We observe that both adding multiple heads to the spatial attention mechanism had minimal impact on the validation loss. Additionally, the temporal attention mechanism had little effect too. However, we can conclude that the spatial attention mechanism contributes the most to the low loss, confirming that the most important aspect of this assignment is accurately modeling relationships between the ego vehicle and other agents.

6 Conclusion

Our tuned LSTM+Attention model demonstrates strong performance on *straight-line* ground truth trajectories, closely following the expected path of the ego vehicle. However, for *more complex* trajectories—such as those involving sharp curves or requiring intricate decision-making—the model exhibits erratic behavior, as seen in the top-left and top-right trajectory figures. This is expected, and highlights an important point: regardless of tuning, there exists a lower bound on model accuracy due to the inherent difficulty of predicting complex, long-term behavior. This can be seen with the diminished returns we observe as we increase model complexity, as well as monitoring top performers on Kaggle's leaderboard.

In this deep learning task, the model predicts 60 future time-steps based on 50 historical time-steps of ego and neighboring agents. Even with sophisticated modeling of interactions, a fundamental level of uncertainty remains. For instance, if three neighboring vehicles turn right, how should the ego vehicle determine whether to follow or continue straight? While straight-line and gently curving trajectories are relatively deterministic, complex behaviors are harder to anticipate and often amount to informed guesswork. Additionally given the comparable MSE and MAE of our model to the baselines, the problem ultimately emphasized incremental model improvements and aggressive hyperparameter tuning.

In terms of **limitations**, limited computational resources became our crux, significantly hampering developmental progress due to slow training loops and finite hours on cloud based platforms such as *Google Colab* and *Kaggle*. Additionally, most research with similar problem setups seem to only handle straight line trajectory predictions, such as highway/freeway driving and lane maintenance, further supporting the limitations of the data given. However given these limitations, quantitatively our final model achieved a validation MSE of **7.5843**, with a test MSE of **7.97**, placing us **16th** overall in the class. This result reflects the effectiveness of our iterative experimentation process-across **69** total official submissions-and targeted hyperparameter tuning. Though challenges remain in modeling complex behaviors, our model's performance on straightforward cases and in general demonstrate serious potential.

Given enough resources, in the **future** we'd like to explore the impact that **Graph Neural Networks (GNNs)** would have on the training data, whether it could learn attention weights between different agents better compared to our current attention methods. Additionally, exploration into **Ensemble Methods**, **Multi-task Learning**, and **Transformer Hybrid Networks** might give better results in *learned positional embeddings* as well as better *intent modeling* of the ego vehicle.

7 Contributions

- **Tim Kraemer**
  In the first half of the Spring quarter, Tim led the charge on initial baseline modeling and testing. He also performed research on trajectory prediction model papers, creating the LSTM + Attention learning model based on Qingyu Meng, et al. (2021) and performed extensive hyperparameter training using significant Kaggle GPU hours on his account. Additionally, Tim contributed large amounts to the mid-quarter report, slide deck, and the final end-of-quarter report, specifically writing the introduction, problem statement, methods, implementation details, conclusion, and collaborating on the research and writing of the related-works section. Additionally, collected most of the data and metrics, generated all of the plots, and created the high-level model diagram.
- **Mathew Raju**
  Mathew's contributions included exploring hyperparameters for various models and looking at research papers that could help with improving our leaderboard position. This included looking at different optimizers like Radam which helped make progress early on in the quarter. He also helped develop the mid-quarter report, reported the overall strengths/weaknesses of our model in the presentation, and for the final report section helped work with evaluation, baselines and leaderboard position.
- **Angus Yick**
  Angus's contributions in the first half of the quarter included researching the topic and contributing to the mid-quarter report. His main contributions were in the second half of the quarter, creating the model with multi-headed attention and learning rate warmup + plateau. Angus also contributed to the presentation slides and the following final report sections: related work, model architecture, implementation details, and results.

References

[1] Sepp Hocrhetier and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[2] Seong Hueon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. 2018. arXiv:1802:06338v3.

[3] Bang Cheng, Xin Xu, Yujun Zeng, Junkai Ren, and Seul Jung. Pedestrian trajectory prediction via the social-grid lstm model. *ACAIT*, 2018(16), 2018.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ArXiv*, 1409, 2014.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[6] Wenxin Jiang, Han Chen, Xiu Shen, and Zhong Li. Spatial-temporal attentive lstm for vehicle-trajectory prediction. *ISPRS International Journal of Geo-Information*, 11(7):354, 2022.

[7] Qingyu Meng, Bingxu Shang, Yanran Liu, and Xu Zhao Hongyan Guo. Intelligent vehicles trajectory prediction with spatial and temporal attention mechanism. *IFAC-PapersOnLine*, 54(10):454–459, 2021.