Interaction-Aware Motion Prediction for Autonomous Driving (Track Number 1)

Group Number 35

Bharath Raam, Radhakrishnan
A69030742

Gokul, Gandhikumar
A69036649

Ketki , Patankar
A69032361

Shrenik, Jain
A69029862

Abstract

Autonomous driving systems must accurately predict how surrounding agents will behave in dynamic and complex environments. However, traditional approaches often treat each agent independently, leading to unrealistic forecasts and conservative planning. This work proposes an interaction-aware motion prediction framework that jointly models the influence of surrounding vehicles and road geometry using a Transformer-based scene encoder and sequential decoder. Evaluated on the Waymo Interaction Prediction Dataset, our model demonstrates high accuracy in forecasting trajectories across overtaking, merging, and left-turn maneuvers. Among various decoders tested (LSTM, GRU, Transformer), LSTM yielded the best performance. Our findings suggest that capturing inter-agent dependencies and map context is critical for safe and efficient planning.

1 Introduction

Autonomous driving has the potential to transform transportation by enhancing road safety, reducing traffic congestion, and improving mobility. A central requirement for realizing these benefits is the ability to anticipate how other road users, such as vehicles, cyclists, and pedestrians, will move in the near future. This task, known as motion prediction, allows the autonomous vehicle (AV) to plan safe and efficient maneuvers by understanding its environment.

However, traditional motion prediction methods often treat other agents as independent entities whose behaviors are unaffected by the ego vehicle's actions. This assumption overlooks the inherently interactive nature of real-world traffic, where agents constantly influence one another through merging, yielding, or overtaking. The lack of interaction modeling can lead to conservative or implausible predictions, undermining both safety and performance. Furthermore, many current models separate prediction from planning, missing the opportunity to generate predictions that are aligned with the AV's intended behavior.

To address these challenges, our project introduces a Transformer-based framework for interaction-aware motion prediction that unifies scene understanding, trajectory forecasting, and planning. The framework captures temporal dynamics, agent-agent interactions, and agent-map relationships, enabling the AV to reason contextually and plan proactively. Inspired by the MotionLM paradigm (1), our approach treats motion forecasting as a structured sequence modeling task and integrates it within a receding horizon planning scheme for continuous, real-time decision-making.

Our key contributions include:

- Designing an interaction-aware decoder (using LSTM, GRU, and Transformer variants) that predicts multi-agent trajectories conditioned on the ego vehicle's future plan.

Preprint. Under review.

- Integrating a real-time cost-based planning module that evaluates predicted trajectories using metrics for safety, comfort, and goal-directedness.
- Achieving high accuracy and low displacement errors on the Waymo Interaction Prediction Open Dataset across common traffic scenarios such as overtaking, lane merging, and left turns.

2 Related work

Motion prediction for autonomous driving has been extensively studied through a variety of modeling paradigms, including policy-based learning, model-based approaches, and interaction-aware prediction frameworks. Early efforts often relied on policy-based learning, such as imitation learning (IL) (2) and reinforcement learning (RL) (3)(4)(5). While IL could mimic expert behavior, it struggled with distribution shift, which refers to the errors that arise when there is a mismatch between the training environment and the testing environment. Reinforcement learning was more robust in this regard but suffered from inefficiency due to its reliance on trial-and-error exploration, making it difficult to scale in real-world scenarios.

To overcome these challenges, researchers turned to model-based predictors that learn to forecast future agent trajectories. Early implementations of these predictors (6) (7) (8) followed an autoregressive structure, predicting one future time step at a time. However, these models often suffered from the issue of compounding errors. Specifically, small prediction errors made in early time steps would accumulate as the model progressed through the sequence, ultimately leading to inaccurate long-horizon trajectory forecasts.

To mitigate this, more recent approaches have explored multi-step look-ahead prediction, where the model generates complete future trajectories instead of predicting step-by-step. Although this improves stability and accuracy, a fundamental limitation persists: most of these models assume that neighboring vehicles behave independently of the ego vehicle's future actions. Consequently, the ego agent treats other traffic participants as static or passive obstacles. This assumption is overly simplistic and fails to capture the interactive and dynamic nature of real-world traffic, which can result in unsafe or overly conservative plans.

Recent advances have addressed this limitation by incorporating interaction-aware models that consider mutual influences between agents. These models allow for more socially compliant and context-aware predictions (9) (10). However, many existing methods still treat motion prediction and planning as separate modules, which prevents the planner from leveraging predictions in a tightly coupled manner.

In contrast, the framework proposed in this work integrates interaction-aware multi-agent prediction with real-time cost-based planning. Our model predicts how other agents will respond based on the ego vehicle's intended future path. These predictions are then used to plan safer and more effective trajectories for the ego agent. This unified approach enables better coordination between prediction and planning, advancing the state of the art in behaviorally-aware autonomous driving.

3 Methodology

Figure 1 illustrates the high-level architecture of our proposed solution. The agent, referring to the autonomous vehicle (AV) or ego agent, makes decisions at each time step by generating a set of candidate trajectories over a short prediction horizon (3 steps). For each candidate trajectory, the interaction-aware prediction model forecasts how surrounding vehicles are likely to respond. These predicted responses, along with the candidate trajectories, are passed to a trajectory evaluator, which selects the most suitable trajectory based on a cost function. The selected trajectory is then executed in the environment. After each step, the resulting state and action are stored in a training buffer, which is periodically used to retrain and improve the interaction-aware prediction model.

Our proposed approach is organized into three main components:

- **Trajectory Generator:** Generates set of candidate trajectories in Frenet space using cubic and quintic polynomials, then converts them to Cartesian coordinates.

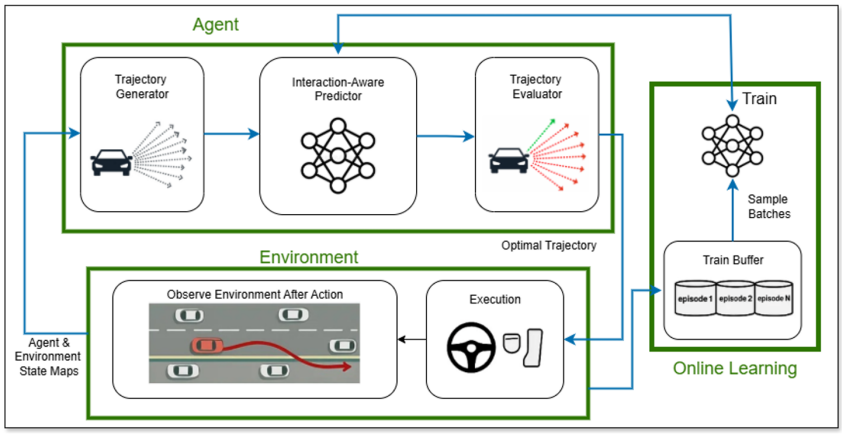

Figure 1: Proposed Solution Architecture

- **Interaction-aware Prediction Model** Predicts the response of other agents in the environment to each candidate trajectory. It consists of:
  - **Encoder:** A transformer-based scene encoder that captures temporal behaviors of agents and their interactions with both other agents and the environment by jointly modeling agent history, inter-agent dynamics, and agent-map context.
  - **Decoder:** An interaction-aware decoder that predicts the future trajectories of multiple agents conditioned on the ego vehicle's candidate plan, enabling the model to anticipate the environment's response to the autonomous vehicle's potential actions.
- **Trajectory Evaluator:** A cost-based planning module that evaluates candidate trajectories based on safety, comfort, and goal-oriented metrics using the responses predicted in previous step, and selects the trajectory that minimizes the overall cost for execution.

3.1 Trajectory Generator

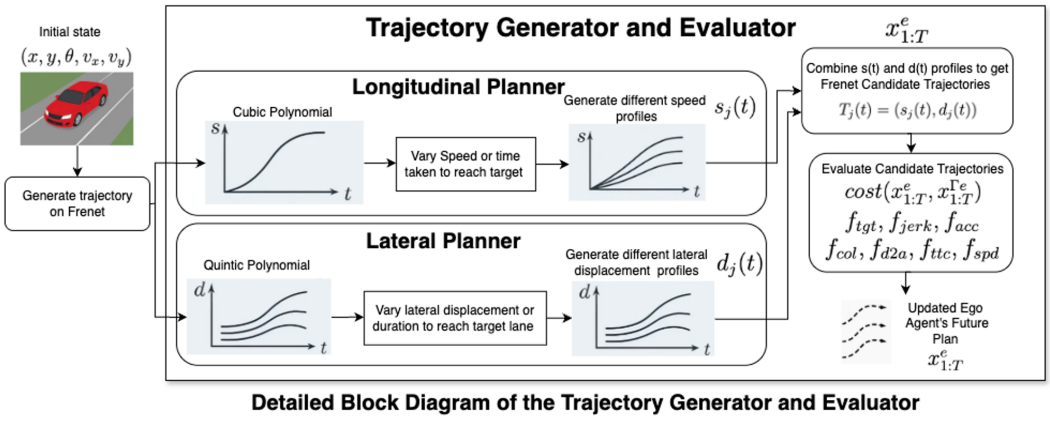

Figure 2: Detailed Block Diagram of Trajectory Generator and Evaluator

As illustrated in Figure 2, candidate trajectories are generated in the *Frenet frame*, a curvilinear coordinate system aligned with the road's geometry. This frame allows motion planning to be decoupled into two components: longitudinal (s) and lateral (d). The longitudinal component s measures distance along the road centerline, while the lateral component d captures deviation from this path. This separation simplifies trajectory generation and constraint handling: forward motion corresponds to increasing s, and lane adherence requires d to remain close to zero.

In the longitudinal direction, candidate speed profiles are generated using cubic polynomials, which satisfy four boundary conditions (initial and final positions and velocities). Cubic polynomials are

computationally efficient and sufficient for modeling smooth accelerations and decelerations, making them well-suited for trajectory planning without needing control over higher derivatives like jerk. These speed profiles transition from the vehicle's current speed to multiple target speeds and are then held constant for the remaining planning horizon. Longitudinal positions $s(t)$ are obtained by integrating these velocity profiles over time.

For lateral maneuvers, quintic polynomials are used to model lane changes. Quintics offer six degrees of freedom, enabling enforcement of boundary conditions on position, velocity, and acceleration at both the beginning and end of the maneuver. This ensures that lateral acceleration and velocity start and end at zero, leading to comfortable, dynamically feasible lane changes with minimal lateral jerk.

Each complete candidate trajectory is formed by pairing a longitudinal speed profile with a lateral maneuver, yielding a trajectory $(s(t), d(t))$ in Frenet space. These trajectories are then transformed into Cartesian coordinates $(x(t), y(t))$ for execution in the physical environment.

3.2 Interaction-aware Prediction Model

3.2.1 Scene Encoder

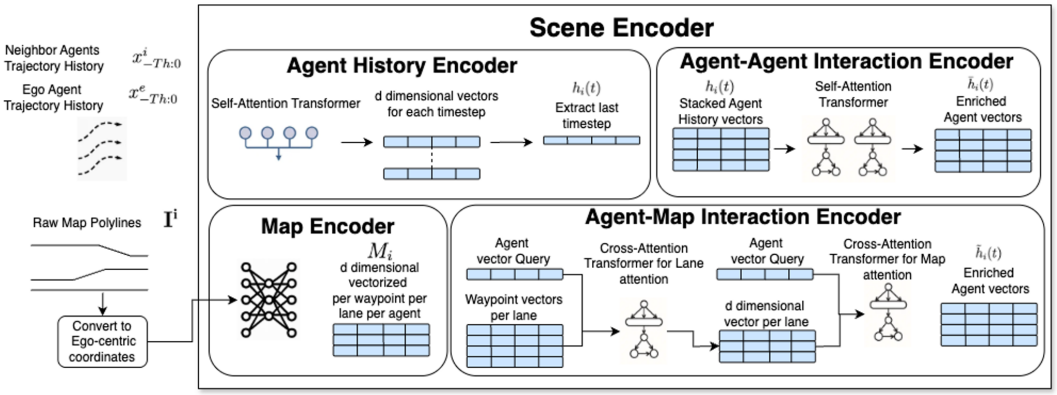

Figure 3: Detailed Block Diagram of Scene Encoder

The Scene Encoder (11) (12) is responsible for extracting rich, context-aware latent representations from both the dynamic behaviors of surrounding agents and the static structure of the driving environment. It processes two key inputs:

- **Agent History Trajectories:** These inputs represent the 11 past time steps, referred to as the *historical horizon*, for the ego vehicle and its five closest neighboring agents. At each time step, the state is encoded as a 5-dimensional feature vector comprising position (x, y), heading angle θ, and velocity components (v_x, v_y). For each agent in the batch, these sequences form a tensor of shape $(\text{Batch}, \text{Agents}, 11, 5)$. When fewer than five neighbors are available, zero-padding is applied to maintain a consistent input shape across batches.
- **Map Polylines:** The local map context for each agent comprises three distinct lanes, each represented by a sequence of 51 waypoints. Every waypoint encodes four key attributes: the spatial coordinates (x, y), the lane heading direction, and the speed limit applicable to that segment. These collectively form a tensor of shape $(\text{Batch}, \text{Agents}, 3, 51, 4)$, capturing the detailed geometric layout and traffic constraints in the agent's vicinity.

To transform these raw inputs into meaningful high-dimensional embeddings, the Scene Encoder comprises four main components. Each component is designed to independently extract features and then integrate spatial and temporal dependencies relevant for downstream trajectory prediction and planning. The components of the encoder are:

(a) Agent History Encoder: This component encodes the motion behavior of each agent based on its past 11 time steps. Initially, each 5D state vector at every time step is projected into a 128-dimensional embedding using a shared Multi-Layer Perceptron (MLP). To retain temporal order information, sinusoidal positional encoding is added to each time step. The resulting sequence of embeddings

is then passed through an 8-head self-attention Transformer. The self-attention mechanism allows the model to weigh the relevance of each historical state in context, enabling it to capture dynamic patterns such as acceleration, deceleration, lane changes, or abrupt stops. The final hidden state (corresponding to the last time step) summarizes the entire historical behavior of the agent, forming a context-rich motion representation of shape $(\text{Batch}, \text{Agents}, 128)$.

(b) Map Encoder: To encode the geometric and semantic structure of the local road network, each of the 51 waypoints per lane is independently processed through a shared MLP, which projects the 4-dimensional waypoint vector $(x, y, \text{heading}, \text{speed\_limit})$ into a 128-dimensional space. This captures features such as lane orientation, and curvature. The output is a high-dimensional tensor of shape $(\text{Batch}, \text{Agents}, 3, 51, 128)$, representing a spatially structured and semantically enriched map for each agent. This encoding is critical for understanding the driving affordances and feasible maneuvers available to each agent.

(c) Agent-Agent Interaction Encoder: Once individual agent histories are encoded, we model the interactions between the ego agent and its surrounding neighbors. The 128-dimensional embeddings of all six agents (ego + neighbors) are stacked into a single tensor of shape $(\text{Batch}, 6, 128)$. This tensor is processed by a multi-head self-attention module, which enables each agent to attend to the behaviors of every other agent in the scene. Such modeling is essential for capturing interaction effects like yielding, overtaking, merging, or crowding. To improve representation quality and training stability, we use residual connections and a second attention layer. The resulting interaction-aware embedding incorporates both individual behavior and inter-agent dynamics, which are crucial for predicting multi-agent interactions.

(d) Agent-Map Interaction Encoder: To fuse dynamic agent behavior with the static road structure, we use a cross-attention mechanism. For each agent, its interaction-aware embedding (from the agent-agent interaction encoder module) serves as a query, and the 3×51 waypoint map encoding acts as the key and value. First, positional encoding is added to the map polylines to preserve waypoint order within lanes. The cross-attention mechanism computes how strongly each agent attends to each of the 51 waypoints in each lane, resulting in an intermediate attention map of shape $(\text{Batch}, \text{Agents}, 3, 128)$. These lane-level vectors capture how relevant each lane is to the agent's current context. A secondary attention step then fuses the three lane-level vectors into a single map-aware embedding of shape $(\text{Batch}, \text{Agents}, 128)$. This final representation integrates both behavioral and spatial features, enabling downstream modules to make informed predictions and planning decisions.

3.2.2 Interaction-aware Decoder

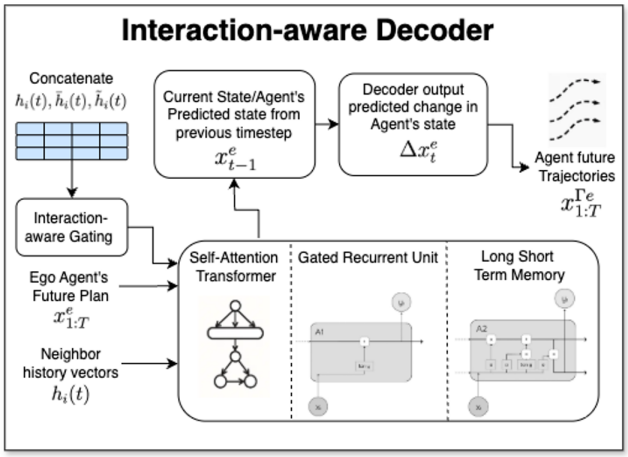

Figure 4: Detailed Block Diagram of Interaction-aware Decoder

The decoder takes as input a unified embedding for each agent, formed by concatenating the map-aware embedding (from the Agent-Map Interaction Encoder), the motion history embedding (from the Agent History Encoder), and the interaction-aware embedding (from the Agent-Agent Interaction Encoder). Together, these embeddings encode spatial context, temporal dynamics, and inter-agent

dependencies, providing the decoder with a comprehensive representation for accurate trajectory prediction.

The decoder is responsible for autoregressively predicting the future trajectories of surrounding agents over a fixed horizon of 30 time steps. The final fused agent embeddings are fed into a decoder to generate future trajectories. We evaluate three decoder architectures described below:

(a) LSTM Decoder: The LSTM decoder uses `nn.LSTMCell` with a hidden size of 384. The interaction-aware fused embedding initializes the hidden state, while the cell state is zero-initialized. At every step, the current state of the agent and the corresponding plan point are projected into a 128-dimensional space. These are fused using a sigmoid-based gating mechanism, and the resulting input is processed by the LSTM cell. The decoder then produces a position delta, which is added to the agent's current state to update its position for the next step.

The advantages of using LSTM lie in its ability to capture long-term dependencies in motion data, making it well-suited for modeling gradual driving behaviors such as smooth deceleration and lane changes. Its memory-based structure enables better understanding of temporal patterns in agent trajectories, resulting in more accurate and stable motion predictions.

(b) GRU Decoder: The GRU decoder follows a structure similar to the LSTM decoder but replaces the LSTM cell with `nn.GRUCell`, using only a single hidden state. The inputs—the current agent state and the planned waypoint—are projected and fused using the same gating mechanism. The GRU cell processes this input and outputs a hidden state, which is decoded to yield the position delta for the next time step.

The advantages of using GRU include its reduced number of parameters and lower computational overhead compared to LSTM, which leads to faster training and inference. This efficiency makes GRUs particularly suitable for real-time applications where quick and reliable motion prediction is essential.

(c) Transformer Decoder: The Transformer decoder uses a self-attention-based architecture built upon a single-layer `nn.TransformerEncoder` with 8 attention heads and a model dimension of 384. The current agent state and the corresponding planned waypoint are projected and gated, then summed with the initial embedding. Positional encoding is added to preserve temporal order. The resulting tensor is passed through the Transformer encoder, which processes the input sequence and outputs an embedding that is decoded into a position delta. The predicted state is updated and propagated to the next step.

The advantages of using the Transformer model include its ability to perform parallel computation and apply global attention across all time steps, allowing it to efficiently model long-range dependencies without relying on recurrence. This makes Transformers especially effective in complex multi-agent interaction scenarios, where understanding the influence of multiple surrounding agents over time is crucial for accurate motion prediction.

Each decoder predicts agent trajectories over a 5-second horizon, modeling both the temporal and spatial influence of other agents and the environment.

Each of these decoder variants offers distinct trade-offs between complexity and expressiveness. They all incorporate interaction-aware context, enabling more accurate and behaviorally consistent multi-agent trajectory predictions.

3.3 Trajectory Evaluator

Following the generation of candidate trajectories (see Figure 2), a cost-based planner evaluates each trajectory using an interaction-aware model that predicts how surrounding agents will respond. The planner computes a cumulative cost based on seven criteria grouped into three main categories: Goal Progress & Efficiency, Comfort & Smoothness, and Safety & Collision Avoidance.

The Goal Progress & Efficiency category includes the distance to goal f_{tgt}, which penalizes trajectories ending far from the target, and speed adherence f_{spd}, which penalizes deviations from the road's speed limit. Comfort & Smoothness covers longitudinal jerk f_{jerk}, penalizing abrupt changes in acceleration, and lateral acceleration f_{acc}, which penalizes sharp turns or sudden lateral movements. The Safety & Collision Avoidance category includes distance to other agents f_{d2a}, penalizing close proximity to surrounding vehicles, time-to-collision f_{ttc}, penalizing short intervals

before potential collisions, and a binary collision indicator f_{col} that heavily penalizes any predicted collisions. Detailed information about the above cost functions can be found in the Appendix.

Each trajectory is scored by summing these individual costs, where high penalties are given to unsafe, uncomfortable, or inefficient behaviors (e.g., high jerk, small f_{d2a}, or low f_{ttc}), and lower costs reward smoother, safer, and more goal-directed behaviors. The trajectory with the lowest overall cost is selected as the final ego plan and executed at the current decision step. This enables realistic, socially-aware, and goal-aligned behavior in dynamic driving environments.

4 Experiments

4.1 Dataset

The Waymo Open Interaction Prediction Dataset (13) is a large-scale benchmark curated to train multi-agent trajectory forecasting models for complex self-driving scenarios. It consists of 100,000 driving scenarios, with each scenario featuring the AV and another agent exhibiting interactive behavior, such as merging, yielding, or navigating an intersection, along with surrounding agents and detailed, vectorized HD maps containing lane boundaries and road geometries. The dataset also includes the actions taken by a human driver in each scenario, allowing comparison between the behavior of the AV and that of a human driver.

4.2 Driving Scenario

To rigorously evaluate our framework, we designed 3 challenging traffic scenarios within the SMARTS simulator: 1) Unsignalized Intersection - the ego vehicle must coordinate with multi-directional traffic flows; 2) Merging - starting from an on-ramp, the AV must execute a safe and timely lane change into dense highway traffic; and 3) Overtaking - the AV is required to overtake slower vehicles in its lane and return to its original path. Each scenario is carefully crafted to reflect complex real-world interactions, requiring the autonomous vehicle (AV) to anticipate agent behavior and make contextually grounded decisions. Each scenario introduces dynamic and heterogeneous traffic patterns to promote data diversity.

4.3 Simulation and Modeling Parameters

The system operates with a 3-second prediction and planning horizon, discretized into 30 timesteps at 0.1-second intervals. Each agent's historical trajectory spans 1 second (10 timesteps), and during inference, the ego vehicle predicts the future behavior of its five nearest neighboring agents. For spatial awareness, each agent is assigned a vectorized local map, comprising three nearby lane centerlines, with 50 waypoints per lane spaced 1 meter apart to capture road geometry. Trajectory generation follows a candidate-based approach, where the lowest-cost trajectory is selected and executed for 15 steps during training, while 5 steps are used in testing to maintain a controlled evaluation process.

4.4 Training Configuration

The motion prediction network is implemented in PyTorch and optimized using the Adam optimizer with an initial learning rate of 2×10^{-4}, decayed by a factor of 0.8 every 5000 gradient steps. The model is trained over 1000 episodes, with 50 gradient updates per episode, and a batch size of 32. An ϵ-greedy exploration policy is employed, where ϵ decays linearly from 1.0 to 0.05 over the first 500 episodes to balance exploration and exploitation.

4.5 Results

Figure 5 illustrates how the AV executes key maneuvers across the three driving scenarios. Video demonstrations of our simulations are made available in the Appendix. Table 1 presents a comparison of the performance (based on the Success Rate, ADE, and FDE) of the three types of decoders used in the model.

Table 1: Evaluation metrics for different decoder models

Decoder Model	Success Rate (%)	ADE (in meters)	FDE (in meters)
GRU	95.41	0.430	0.667
LSTM	97.95	0.449	0.665
Transformer	95.94	0.438	0.649

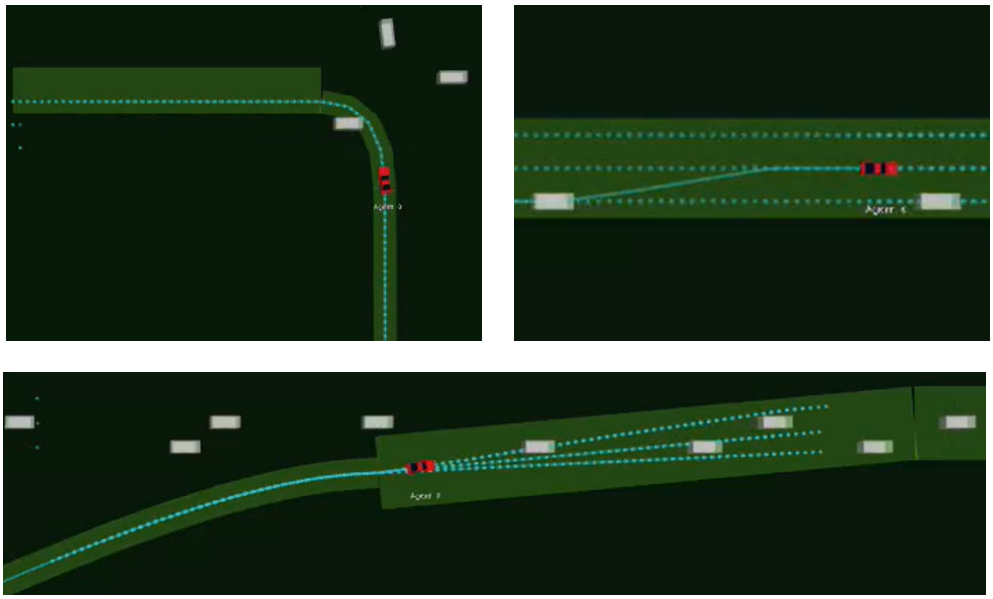

Figure 5: Screenshots from simulations showing the AV performing maneuvers: *(clockwise from top-left)* left turn, overtaking, and merging.

5 Conclusion

In this work, we developed an interaction-aware motion prediction model that integrates scene context, road geometry, and multi-agent dynamics into a unified framework. By evaluating different decoding strategies, we found that the LSTM-based decoder achieved the highest success rate (97.95%), likely due to its superior temporal memory. GRU and Transformer decoders also performed competitively, with success rates of 95.41% and 95.94%, respectively. Across all models, consistently low Average Displacement Error (ADE) and Final Displacement Error (FDE) indicate strong alignment between predicted and actual trajectories, reflecting human-like behavior prediction.

A key takeaway from this project is the importance of modeling inter-agent interactions and road semantics jointly for reliable trajectory forecasting. Decoders that capture temporal dependencies effectively tend to produce more stable and accurate predictions. However, challenges remain in generalizing to complex urban environments and adapting to highly dynamic scenes.

For future work, we can focus on extending the model to incorporate richer scene cues such as traffic lights, stop signs, and lane-level constraints. We also aim to explore end-to-end joint training of the prediction and planning modules to close the loop between intention inference and action generation. Finally, scaling the framework to handle dense, multi-intersection urban environments remains a promising and necessary direction for robust real-world deployment.

Appendix

Github Link

https://github.com/shrenik-jain/Interaction-Aware-Motion-Prediction

Demo Link

https://ucsdcloud-my.sharepoint.com/:f:/g/personal/bradhakrishnan_ucsd_edu/Ev6XNkPQg-FHjCx_9x4zKe8BISpsEbspAwbDCX8hKfhbFw?e=bmHJYE

Team Member Contribution

All the team member have completed the course and instructional assistant evaluation surveys.

Table 2: Team Member Contributions

Task	Bharath	Gokul	Ketki	Shrenik
Conceptualization	✓	✓	✓	✓
Data Curation	✓		✓	✓
Methodology	✓	✓	✓	✓
Software/Experiments	✓	✓		✓
Report Writing		✓	✓	
Poster Preparation	✓	✓	✓	✓

Additional Details for Cost Function

Each cost term in the evaluation reflects a physically grounded driving behavior and is computed at each future timestep. The goal progress term f_{tgt} measures the Euclidean distance between the final predicted position p_T and the desired goal p_{goal}:

$$f_{\text{tgt}} = \|p_T - p_{\text{goal}}\|.$$

Speed adherence is captured by f_{spd}, penalizing deviations from the legal speed limit v_{limit} through:

$$f_{\text{spd}}(t) = |v_t - v_{\text{limit}}|.$$

Comfort and smoothness are modeled using longitudinal jerk f_{jerk}, expressed as the rate of change of acceleration:

$$f_{\text{jerk}}(t) = \dot{a}_t = \frac{\Delta a_t}{\Delta t},$$

and lateral stability is governed by the steering angle δ_t and its rate of change $\dot{\delta}_t$, where:

$$f_{\text{acc}}(t) = \delta_t, \quad \dot{\delta}_t = \frac{\Delta \delta_t}{\Delta t}.$$

Safety-related costs further reinforce responsible motion planning. The distance-to-agent cost f_{d2a} penalizes close proximity to nearby agents by computing:

$$f_{\text{d2a}}(t) = \min_i \|p_t - p_t^i\|_2,$$

while a collision penalty f_{col} is enforced via a hinge loss that activates if this distance falls below a minimum safety margin ϵ:

$$f_{\text{col}}(t) = \begin{cases} \epsilon - f_{\text{d2a}}(t), & \text{if } f_{\text{d2a}}(t) \leq \epsilon \\ 0, & \text{otherwise.} \end{cases}$$

This formulation ensures the AV maintains adequate buffer space and avoids unsafe interactions. Time-to-collision f_{ttc}, although not always explicitly modeled, is commonly estimated as the ratio between relative distance and velocity and penalized when this value is critically low. The final cost

for a candidate trajectory is computed as the sum of these individual terms, weighted appropriately and aggregated over selected future timestamps. The complete objective function is given by:

$$\min \frac{1}{2} \sum_t \sum_i \left\| w^i c_t^i \right\|^2,$$

where c_t^i denotes each cost component at time t, and w^i is its corresponding weight. This multi-objective formulation balances efficiency, comfort, and safety, enabling the planner to select a socially compliant and context-aware trajectory.

References

[1] A. Seff, B. Cera, D. Chen, M. Ng, A. Zhou, N. Nayakanti, K. S. Refaat, R. Al-Rfou, and B. Sapp, "Motionlm: Multi-agent motion forecasting as language modeling," 2023. [Online]. Available: https://arxiv.org/abs/2309.16534

[2] Z. Huang, C. Lv, Y. Xing, and J. Wu, "Multi-modal sensor fusion-based deep neural network for end-to-end autonomous driving with scene understanding," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 11 781–11 790, 2021.

[3] J. Wu, Z. Huang, Z. Hu, and C. Lv, "Toward human-in-the-loop ai: Enhancing deep reinforcement learning via real-time human guidance for autonomous driving," *Engineering*, 2022.

[4] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[5] J. B. Hamrick, A. L. Friesen, F. Behbahani, A. Guez, F. Viola, S. Witherspoon, T. Anthony, L. H. Buesing, P. Veličković, and T. Weber, "On the role of planning in model-based deep reinforcement learning," in *International Conference on Learning Representations*, 2020.

[6] M. Henaff, Y. LeCun, and A. Canziani, "Model-predictive policy learning with uncertainty regularization for driving in dense traffic," in *7th International Conference on Learning Representations (ICLR)*, 2019.

[7] V. Sobal, A. Canziani, N. Carion, K. Cho, and Y. LeCun, "Separating the world and ego models for self-driving," in *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022.

[8] J. L. V. Espinoza, A. Liniger, W. Schwarting, D. Rus, and L. V. Gool, "Deep interactive motion prediction and planning: Playing games with motion prediction models," in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 1006–1019.

[9] Z. Huang, H. Liu, J. Wu, and C. Lv, "Conditional predictive behavior planning with inverse reinforcement learning for human-like autonomous driving," 2022, arXiv preprint arXiv:2212.08787.

[10] H. Song, W. Ding, Y. Chen, S. Shen, M. Y. Wang, and Q. Chen, "Pip: Planning-informed trajectory prediction for autonomous driving," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 598–614.

[11] Z. Huang, X. Mo, and C. Lv, "Multi-modal motion prediction with transformer-based neural network for autonomous driving," in *Proceedings of the 2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2605–2611.

[12] Z. Huang, H. Liu, J. Wu, and C. Lv, "Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving," 2022, arXiv preprint arXiv:2207.10422.

[13] S. Ettinger, T. Karmali, Z. Wang, V. Le, B. Ichter, S. Levine, K. Choromanski, and M. J. Kochenderfer, "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.