

ADAPT: Efficient Multi-Agent Trajectory Prediction with Adaptation

Görkay Aydemir¹ Adil Kaan Akan^{1,2} Fatma Güney^{1,2}

¹KUIS AI Center

²Department of Computer Engineering, Koç University

gorkayaydemir@gmail.com kakan20@ku.edu.tr fguney@ku.edu.tr

Abstract

Forecasting future trajectories of agents in complex traffic scenes requires reliable and efficient predictions for all agents in the scene. However, existing methods for trajectory prediction are either inefficient or sacrifice accuracy. To address this challenge, we propose ADAPT, a novel approach for jointly predicting the trajectories of all agents in the scene with dynamic weight learning. Our approach outperforms state-of-the-art methods in both single-agent and multi-agent settings on the Argoverse and Interaction datasets, with a fraction of their computational overhead. We attribute the improvement in our performance: first, to the adaptive head augmenting the model capacity without increasing the model size; second, to our design choices in the endpoint-conditioned prediction, reinforced by gradient stopping. Our analyses show that ADAPT can focus on each agent with adaptive prediction, allowing for accurate predictions efficiently. <https://KUIS-AI.github.io/adapt>

1. Introduction

A self-driving agent needs to be able to anticipate the future behavior of other agents around it to plan its trajectory. This problem, known as trajectory forecasting, is an important requirement for safe navigation. There are multiple challenges to solving this problem. First of all, traffic scenes are highly dynamic. The behavior of an agent depends not only on the scene properties, such as configurations of lanes but also on other agents, such as yielding to another vehicle that has priority. Second, multiple futures need to be predicted due to the inherent uncertainty in future predictions. While these two challenges are studied in the literature, one challenge remains mostly unresolved: The future is shaped according to *all* agents in the scene acting together. Therefore, trajectories of all agents need to be predicted as opposed to the current practice of predicting only the trajectory of a selected agent [11, 6].

The progress in trajectory forecasting has focused mainly on scene representations for predicting the trajectory of a single agent. Typically, the existing meth-

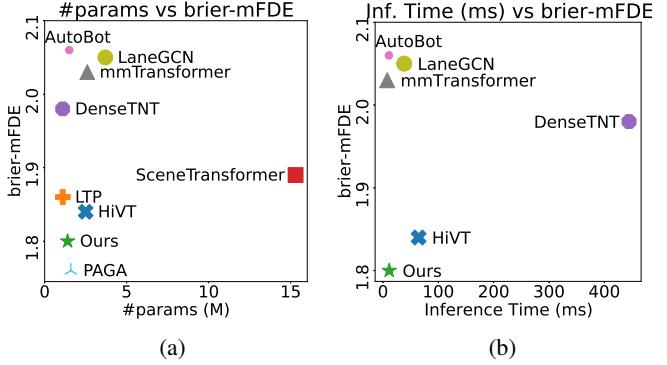


Figure 1: **Accuracy vs. Efficiency.** We plot the accuracy in terms of error (brier-mFDE) vs. the number of parameters (a) and inference time (b) on the test set of Argoverse dataset [11]. Our method achieves one of the lowest reported errors with a small number of parameters, leading to highly efficient inference time compared to methods AutoBot [22], LaneGCN [32], mmTransformer [33], DenseTNT [23], SceneTransformer [37], LTP [48], PAGA [13], and HiVT [53].

ods [32, 49, 21] follow an agent-centric reference frame where the scene is centered around the agent of interest, and everything else is positioned relative to it. This way, the prediction network is provided with the same initial state regardless of the agent’s location or orientation, providing *pose-invariance*. In other words, the scene is observed from the viewpoint of the agent of interest. In multi-agent setting, each agent has a different view of the world, and one cannot be prioritized over another as in the case of the agent-centric approach. A straightforward extension of an agent-centric approach to multi-agent is iterating the process for each agent in its own reference frame (Fig. 2). This is achieved by transforming the scene according to each agent to obtain pose-invariant features. However, this solution scales linearly with the number of agents and causes a variable inference time that cannot be afforded in the real-time setting of driving. As a solution, SceneTransformer [37] introduces a global frame that is shared across agents. In their scene-centric approach, all agents are positioned with respect to

the same reference point but at the cost of pose-invariance.

Ideally, the pose-invariance is a desirable property but for multi-agent prediction, a scene-centric approach can be preferred in real world due to efficiency concerns [37]. Then the question is how do we avoid the problems of a scene-centric approach without sacrificing efficiency? In this paper, we propose a solution to adapt to the situation of each agent with dynamic weight learning [28, 45, 43]. Dynamic networks can adjust the model structure based on input by adapting network weights according to the changes in the input states [25]. Therefore, they are well-suited for the multi-agent prediction task where each agent has a different initial state. Additionally, dynamic networks are capable of expanding the parameter space without increasing computation cost, therefore meeting the real-time requirements of our task. We learn the weights of the network that predicts the endpoints so that they can change and adapt to each agent’s reference frame. With dynamic weights, we can efficiently adapt the prediction head to each agent in a scene-centric approach without iterating over agents.

Our method is not only the first to achieve multi-agent prediction accurately and efficiently in a scene-centric approach but also one of the smallest and fastest among all trajectory prediction models including single-agent prediction. Using a goal-conditioning approach, we can easily switch between single and multi-agent prediction settings. To further enhance the performance of our model, we employ gradient stopping to stabilize the training of trajectory and endpoint prediction. This technique enables us to achieve good performance by fully leveraging the capacity of a small decoder with simple MLP layers rather than a complex one.

We show that our method outperforms the state-of-the-art methods with a fraction of their parameters in both single-agent setting of the Argoverse [11] and multi-agent setting of the Interaction [51]. On Interaction, specifically designed for evaluating multi-agent predictions, our method achieves a 1% miss rate in comparison to 5% which was the lowest achieved so far [21]. Our contributions can be summarized as follows:

- We propose a novel approach for predicting the trajectories of all agents in the scene. Our adaptive head can predict accurate trajectories by dynamically adapting to various initial states of multiple agents.
- We achieve state-of-the-art results efficiently with one of the smallest and fastest models including the ones in single-agent setting. We validate our design choices in endpoint prediction and trajectory prediction with gradient stopping for stabilized training.
- We have created a unified prediction process that can be used for both single and multi-agent settings with the same backbone by utilizing endpoint conditioning. Our method allows for easy switching between scene-centric and agent-centric reference frames, achieving

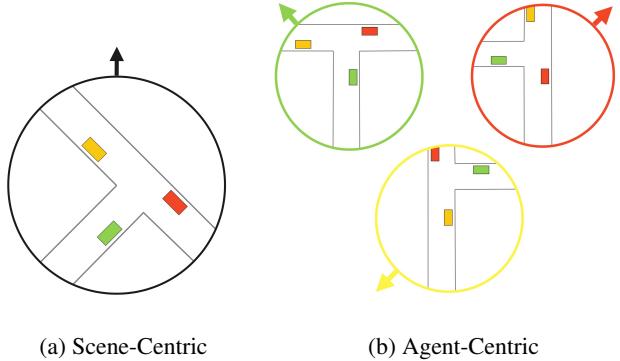


Figure 2: Scene-Centric vs. Agent-Centric Representation. In scene-centric representation (a), all elements are encoded according to the same reference point. In agent-centric representation (b), each agent is encoded in its own reference frame, leading to a complexity linear in the number of agents in multi-agent prediction.

state-of-the-art in both settings.

2. Related Work

2.1. Single-Agent Prediction

Scene Representation: In dynamic traffic scenes, representing the scene elements and modeling interactions between them play a crucial role in performance. In single-agent prediction, previous works focus on the representation of the scene and interaction modeling from the viewpoint of the agent of interest. Earlier works [12, 9, 26, 4, 31, 34] create a rasterized image to represent both the context and the interactions. The previous locations of agents are typically encoded with sequential models such as RNNs [35, 29, 1, 24, 38, 40]. Combining the two, the following works explore more specialized scene representations [44, 10, 15, 19, 39, 5, 38, 36]. In contrast to rasterized representation, Graph Neural Networks (GNNs) enable a more explicit way of modeling interactions such as with a lane graph [32] or a vectorized representation of the scene [17]. Based on their success in capturing hierarchical representations, recent works continue adapting GNNs for interaction modeling [52, 23, 50, 2]. Towards the same purpose, more recent works [53, 33, 41, 37, 21] use transformers with multi-head attention [47], whose success has been proven repeatedly [16, 8, 14, 7]. We also use a scene representation based on multi-head attention.

Attention-Based Encoding: Transformers are widely used for interaction modeling due to their ability to capture the interaction between different scene elements. VectorNet [17] uses the same attention weights for different types of elements, i.e. agents and lanes. LaneGCN [32] categorizes interactions and uses a different type of attention for

each, leading to a more specialized representation. Due to its success, the following works [33, 48, 53] continue modeling different types of interactions. Recently, factorized attention [37, 22] has been proposed to model temporal relations between scene elements efficiently. Instead of performing attention on the entire set of agents and time steps, factorized attention separately processes each axis, i.e. agent, time, and road graph elements. A similar factorization over time and agents is explored in Autobot [22] with a smaller and more efficient architecture. We also use different types of attention to model different types of interactions but enable updated information flow between different scene elements with iterative updates.

2.2. Multi-Agent Prediction

Multi-Agent Prediction: Due to the evaluation setting on publicly available datasets [11, 6], most existing works focus on predicting the trajectory of a single agent. While it led to great progress in scene representations and modeling of dynamic interactions, in real-life scenarios, the agent needs to account for the future trajectories of other agents as well. Multi-agent prediction has been recently addressed by SceneTransformer [37] with a unified architecture to jointly predict consistent trajectories for all agents. While this method can perform inference for all agents in a single forward pass, HiVTT [53] iterates over agents in an agent-centric representation, leading to the aforementioned inefficiency issues. LTP [48] follows a different approach with a lane-oriented representation to predict the most likely lane for each agent in a single pass. However, lane classification may not be as precise as regression. Our method can regress the trajectories of each agent efficiently in a single pass.

Reference Frame: The existing works represent the scene either from the viewpoint of an agent or from a fixed reference point as illustrated in Fig. 2. In the agent-centric representation [53, 22, 49, 23, 52, 32], the scene is transformed so that the agent of interest is positioned at the origin of the scene. In contrast, all elements are positioned with respect to the same reference point in a scene-centric representation [37]. This shared context representation is especially helpful for multi-agent prediction [51, 42] while the agent-centric works better for single-agent prediction [11, 6] due to the simplification of the problem. It allows focusing on a single agent without worrying about other agents except for their relation to the agent of interest. Multi-agent prediction can be performed with sequential agent-centric predictions, i.e. one agent at a time [22, 48, 23, 53, 21]. However, this straightforward extension scales linearly with the number of agents in the scene and raises efficiency concerns. We use a scene-centric representation for multi-agent predictions but adapt to each agent with dynamic weights to benefit from agent-specific features as in agent-centric representation.

3. Methodology

Given the past trajectories of all agents on a High-Definition (HD) map of the scene, our goal is to predict the future trajectories of agents in the scene. In a vectorized scene representation, we model different types of interactions between the agents and the map to obtain a representation for agents (Section 3.1). Following goal-conditioned approaches [52, 23], we first predict a possible set of endpoints. We then refine each endpoint by predicting an offset (Section 3.2). Finally, we predict the full trajectories conditioned on endpoints (Section 3.3). We stabilize training by separating endpoint and trajectory prediction with gradient detaching. Our pipeline is illustrated in Fig. 3. Our model uses small MLPs in endpoint and trajectory prediction, keeping model complexity low.

3.1. Feature Encoding

Polyline Encoding: We represent the map and the agents using a vectorized representation in a structured way. The vectorized representation initially proposed in VectorNet [17] creates a connected graph for each scene element independently. Given past trajectories of agents, $\mathcal{A} = \{\mathbf{a}_i\}$ where $\mathbf{a}_i \in \mathbb{R}^{T \times 2}$ denotes the location of agent i at previous T time steps and HD map, $\mathcal{M} = \{\mathbf{m}_i\}$ where $\mathbf{m}_i \in \mathbb{R}^{l_i \times 2}$ denotes the lane i with l_i consecutive points constituting the lane. We encode each scene element, i.e. a polyline, with a polyline subgraph. We use two separate subgraphs for the agents and lanes (Fig. 3, left), resulting in a feature vector of length d , $\mathbf{f}_i \in \mathbb{R}^d$ for each polyline.

Interaction Modelling: We model different types of interactions between the scene elements (Fig. 3, left). Following LaneGCN [32], we model four types of relations: agent-to-lane (**AL**), lane-to-lane (**LL**), lane-to-agent (**LA**), and agent-to-agent (**AA**). Using attention, we update each feature \mathbf{f}_i extracted by the polyline subgraph.

In contrast to previous work using simple attention operation [32], and given the importance of multi-head attention and feed-forward networks in understanding the relations [18], we use a Multi-Head Attention Block (MHAB) as proposed in [22]. Specifically, we update self-relations (**AA**, **LL**) using a self-attention encoder followed by a feed-forward network (FFN) and cross-relations (**AL**, **LA**) using a cross-attention encoder followed by the FFN:

$$\text{MHA}(\mathbf{f}_q, \mathbf{f}_{kv}) = \text{softmax} \left(\frac{\mathbf{Q} \mathbf{K}^T}{\sqrt{\dim_k}} \right) \mathbf{V} \quad (1)$$

$$\text{where } \mathbf{Q}, \mathbf{K}, \mathbf{V} = \mathbf{W}^q \mathbf{f}_q, \mathbf{W}^k \mathbf{f}_{kv}, \mathbf{W}^v \mathbf{f}_{kv}$$

where each \mathbf{W} is a learned projection. Similar to the original block in [47], our Multi-Head Attention Block is for-

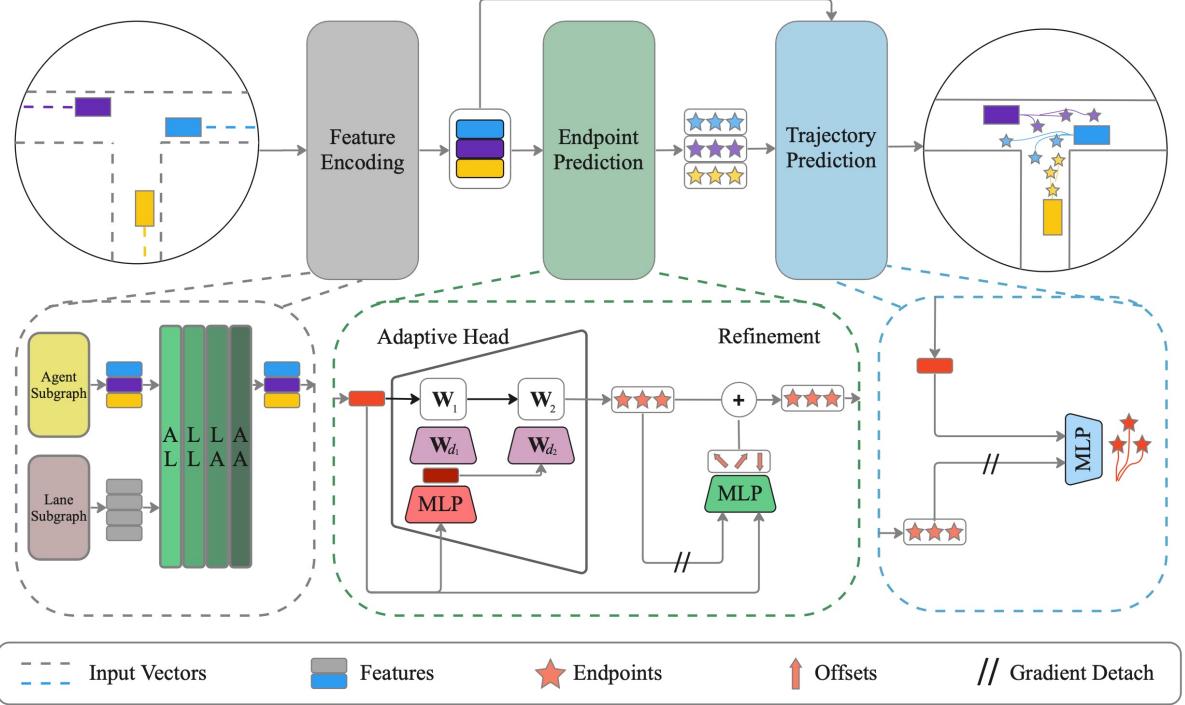


Figure 3: Overview. Our scene encoding approach involves separate polyline encoders that interact in feature encoding (**left**). To predict endpoint proposals, we utilize the endpoint head, which employs the adaptive head with dynamic weights for multi-agent prediction, without the need to transform the scene for each agent. Conversely, we use static head (simple MLP) for single-agent prediction. Then we perform endpoint refinement to improve accuracy (**middle**). Finally, we interpolate the full trajectory for each agent using the refined endpoints (**right**). By utilizing gradient detaching for endpoint and trajectory prediction modules, we achieve better performance with a small and fast architecture.

mally defined as follows:

$$\text{MHAB}(\mathbf{f}_q, \mathbf{f}_{kv}) = \text{norm}(\tilde{\mathbf{f}} + \text{FFN}(\tilde{\mathbf{f}})) \quad (2)$$

where $\tilde{\mathbf{f}} = \text{norm}(\mathbf{f}_q + \text{MHA}(\mathbf{f}_q, \mathbf{f}_{kv}))$

where the norm is the Layer Normalization [3]. Different than LaneGCN [32] applying different interaction types L times sequentially one by one, we model each interaction in order and repeat the process L times. This way, intermediate features can be updated at each iteration, and then the updated features are used to compute attention in the next iteration. Each scene element can be informed by different types of relations L times. See Supplementary for the experiment comparing the two design choices.

3.2. Endpoint Prediction

For endpoint prediction, we either use a single MLP if an agent-centric reference frame is used which might be preferred due to its advantages in single-agent prediction or an adaptive head with dynamic weights if a scene-centric reference frame is used which might be preferred due to its efficiency in multi-agent prediction. Our model uses simple linear layers for endpoint prediction rather than sophisticated modules used in previous work [37, 22].

Endpoint Prediction Head: We predict a possible set of endpoints for each agent based on the agent features from previous attention layers. We utilize two different types of heads to predict the future trajectory of a single agent in an agent-centric reference frame and future trajectories of multiple agents in a scene-centric frame. In single-agent setting, we predict the endpoints with a simple MLP, which we call a *static head*. In multi-agent setting, we train an *adaptive head* to dynamically learn the weights that predict the endpoints. Dynamic weight learning [28, 45] enables the prediction head to adapt to the situation of each agent.

$$\mathbf{W}_1 = \mathbf{W}_{d_1} \tilde{\mathbf{f}} \quad (3)$$

$$\mathbf{W}_2 = \mathbf{W}_{d_2} \tilde{\mathbf{f}}$$

$$\mathbf{F}_{d_1} = \text{ReLU}(\text{norm}(\mathbf{W}_1 \mathbf{f}))$$

$$\hat{\mathbf{y}}_{\text{pred}} = \mathbf{W}_2 \mathbf{F}_{d_1}$$

We visualize the adaptive head in Fig. 3 and explain it mathematically in (3) where \mathbf{W}_{d_1} and \mathbf{W}_{d_2} are trainable parameters and norm is the layer normalization. We process the encoded agent features concatenated with meta info, \mathbf{f} with an MLP and obtain $\tilde{\mathbf{f}}$. Meta info includes the direction and location information of the agent at prediction time.

```

def predict_trajectory(features):
    # features.shape = (N, D)
    features = features.expand(N, K, D)
    # endpoints.shape = (N, K, 2)
    endpoints = endpoint_head(features)
    # offsets.shape = (N, K, 2)
    offsets = MLP1(cat(features, endpoints.detach()))
    endpoints += offsets

    # traj.shape = (N, K, T-1, 2)
    traj = MLP2(cat(features, endpoints.detach()))
    traj = cat(traj, endpoints)

    # scores.shape = (N, K)
    scores = MLP3(cat(features, endpoints.detach()))
    scores = softmax(scores, dim=-1)

    return traj, scores

```

Figure 4: Pseudo-code for Trajectory Prediction. Given features of N agents, we first predict endpoint proposals using the endpoint head. Later, we apply a refinement on the endpoints by adding an offset. We then predict a trajectory for T steps conditioned on each endpoint. We also predict a score associated with each trajectory.

By providing the current state information to the prediction head as input, we allow the dynamic weights to adjust to the state of the agent while predicting the endpoints.

Refinement: We further refine the endpoints by predicting an offset to the initial endpoint proposals from the prediction head. Given the endpoint proposals and the features of the agent, we predict the corresponding offset for each proposal with a simple MLP. We detach the gradients of endpoints before passing them as input to decouple the training of the endpoint prediction and refinement. Intuitively, offsets that are supposed to correct the endpoints can receive an independent gradient update from the endpoints. A similar approach is used to update queries in [27].

3.3. Trajectory Prediction

Trajectory Interpolation: After obtaining the refined endpoint for each agent, we interpolate future coordinates between the initial point and the endpoint with an MLP. We detach the endpoints to ensure that weight updates for full trajectory prediction are separated from endpoint prediction. Similarly, we predict a probability for each trajectory using detached endpoints. We provide the pseudo-code for endpoint and trajectory prediction in Fig. 4. We train static and adaptive heads as the endpoint head for the agent-centric and the scene-centric reference frames, respectively.

Training: For training, we predict K trajectories and apply variety loss to capture multi-modal futures by back-propagating the loss only through the most accurate trajectory. As we predict the full trajectories conditioned on the endpoints, the accuracy of endpoint prediction is essential for full trajectory prediction. Therefore, we apply a loss

on endpoints to improve the endpoint prediction. The final term in our loss function is classification loss to guide the probabilities assigned to trajectories. In summary, we train our model using the endpoint loss, the full trajectory loss, and the trajectory classification loss. Please see Supplementary for the details of our loss functions.

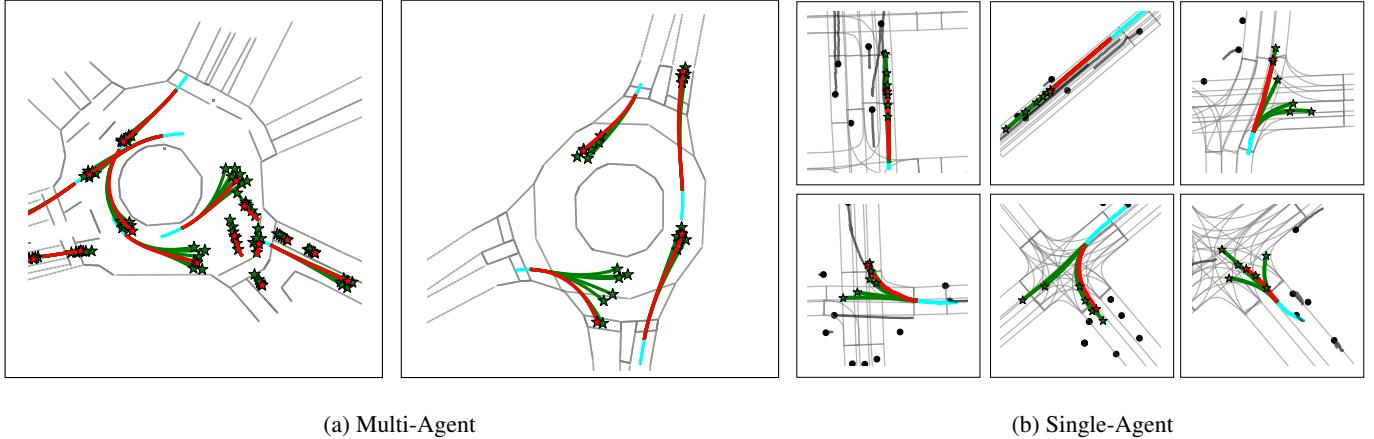
4. Experiments

4.1. Experimental Setup

Datasets: We evaluate our method in single-agent setting on Argoverse v1.1 [11] and in multi-agent setting on Interaction [51]. Argoverse, with 323,557 scenarios, is the commonly used benchmark for single-agent motion forecasting. Given the HD map and the history of agents for 2s, the goal is to predict the future locations of the agent of interest for the next 3s. Interaction contains 62,022 multi-agent scenarios with up to 40 agents per scenario. The goal is to predict the future for all agents in the scene.

Metrics: We use standard metrics including minimum Average Displacement Error (mADE_k), minimum Final Displacement Error (mFDE_k), Miss Rate (MR_k), and brier minimum Final Displacement Error (brier-mFDE_k). These metrics are calculated based on the trajectory with the closest endpoint to the ground truth over k trajectory predictions. mADE_k measures the average ℓ_2 difference between the full prediction and the ground truth, mFDE_k measures the difference between the predicted endpoint and the ground truth. MR_k is the ratio of scenes where mFDE_k is higher than 2 meters. The brier- mFDE_k is calculated as $(1 - p)^2 + \text{mFDE}_k$ where p is the probability predicted for the trajectory. In multi-agent setting, each metric is computed per agent and then averaged over all agents.

Training Details: We set the number of layers L to 3 for both polyline subgraphs and interaction modeling. We train our models with a batch size of 64 for 36 epochs. We use Adam optimizer [30] with an initial learning rate of 1×10^{-4} and 2×10^{-4} for Argoverse and Interaction experiments, respectively. We anneal the learning rate with a factor of 0.15 at the 70th and 90th percentiles. We generate lane vectors for lanes that are closer than 50m to any available agent. For data augmentation, we use random scaling in the range of [0.75, 1.25] for Argoverse and random agent drop with the probability of 0.1 for both Argoverse and Interaction experiments. We also use other agents on Argoverse as additional training data as done in previous work [53]. Specifically, we only consider agents that move by at least 6m following previous work [22, 37]. In agent-centric reference frame, we translate and rotate the scene with respect to the agent of interest. In scene-centric reference frame, we orient the scene based on the mean location of all agents.



(a) Multi-Agent

(b) Single-Agent

Figure 5: Qualitative Results. We visualize multi-agent predictions on Interaction (a) and single-agent on Argoverse (b). The predicted trajectories are shown in green, ground truth in red, past in cyan, and the trajectories of other agents in black.

	mADE _k		mFDE _k		brier-mFDE ₆	#Prm (M)
	k=1	k=6	k=1	k=6		
AutoBot [22]	-	0.89	-	1.41	-	1.5
HO+GO [20]	-	0.92	3.68	1.29	-	-
LaneGCN [32]	1.71	0.87	3.78	1.36	2.05	3.7
mmTr [33]	1.77	0.84	4.00	1.34	2.03	2.6
D-TNT [23]	1.68	0.88	3.63	1.28	1.98	1.1
THOMAS [21]	1.67	0.94	3.59	1.44	1.97	-
SceneTr [37]	1.81	0.80	4.05	1.23	1.89	15.3
LTP [48]	1.62	0.83	3.55	1.30	1.86	1.1
HiVT [53]	1.60	0.77	3.53	1.17	1.84	2.5
MP++* [46]	1.62	0.79	3.61	1.21	1.79	125
PAGA [13]	1.56	0.80	3.38	1.21	1.76	1.6
Ours (Single)	<u>1.59</u>	<u>0.79</u>	<u>3.50</u>	<u>1.17</u>	<u>1.80</u>	<u>1.4</u>

Table 1: Results on Argoverse (Test). This table shows single-agent results on Argoverse. The number of parameters is reported or calculated using the official repositories. Models with ensembles are marked with “*”.

4.2. Quantitative Results

Single-Agent Prediction on Argoverse: We compare our method in single-agent setting using an agent-centric reference frame to the state-of-the-art on test (Table 1) and validation (Table 2) sets of Argoverse. We report results without ensembles, except for Multipath++ [46] (only the result of the ensemble is reported). Our method achieves comparable results to the top-performing methods on the test set in all metrics. In particular, we approach the performance of the state-of-the-art PAGA [13] in the official metric, i.e. brier-mFDE₆ and reach the performance of HiVT [53] in other metrics by using only 56% of its parameters. On the validation set, our method performs the best in terms of mFDE₆ and MR₆. Impressively, ADAPT

achieves these results with one of the smallest and fastest models (Fig. 1).

In Table 2, we report the average runtime per scene on the validation set of Argoverse using a Tesla T4 GPU. To align the settings between different approaches and alleviate implementation differences in parallelism, we set the batch size to 1 and predict the future only for agent of interest per scene. Our method is the second fastest method, only behind mmTransformer [33] but with significantly better results than mmTransformer on both validation and test sets. Note that HiVT [53] suffers significantly in terms of inference time due to agent-centric approach where the scene is normalized for each agent iteratively. Our approach can achieve similar results, and even slightly better, on the test set with only 18% of HiVT’s inference time. We provide a comparison of methods in terms of computational complexity in Supplementary to justify our design choices in feature

	mADE ₆	mFDE ₆	MR ₆	Inf. (ms)
TPCN [49]	0.73	1.15	0.11	-
mmTrans [33]	0.71	1.15	0.11	7.66
LaneGCN [32]	0.71	1.08	-	38.37
LTP [48]	0.78	1.07	-	-
DenseTNT [23]	0.73	1.05	0.10	444.66
HiVT [53]	0.66	<u>0.96</u>	<u>0.09</u>	64.45
PAGA [13]	0.69	1.02	-	-
Ours (Single)	<u>0.67</u>	0.95	0.08	<u>11.31</u>
Ours (Multi)	0.65	0.97	0.08	11.31

Table 2: Results on Argoverse (Validation). This table shows results on the Argoverse validation set. The bottom row shows the multi-agent evaluation in scene-centric reference frame. The inference time is calculated using the official repositories in the same setting.

	mADE ₆	mFDE ₆	MR ₆	Inf. (ms)
AutoBot (J.) [22]	0.21	0.64	0.06	25.29
SceneTr [37]	0.26	0.47	0.05	-
THOMAS [21]	0.26	0.46	0.05	-
Ours (Multi)	0.16	0.34	0.01	11.10

Table 3: **Results on Interaction (Validation).** This table shows multi-agent results on Interaction. The inference time is calculated using the official repositories. The results of SceneTransformer are based on a re-implementation by authors of THOMAS [21].

encoding, contributing to our method’s efficiency. A full comparison in terms of brier-mFDE₆ vs. inference time is provided in Fig. 1b. ADAPT achieves the best performance without sacrificing efficiency.

Multi-Agent Predictions on Argoverse: We extend the Argoverse setting from single-agent in agent-centric reference frame to multi-agent in scene-centric reference frame by modifying the reference point to be the same for all agents. In this case, we use ADAPT with adaptive head instead of static head. For evaluating multi-agent predictions, we only consider the agents that are visible at all timestamps. As shown at the bottom row of Table 2, ADAPT can predict the future trajectories of all agents in scene-centric reference frame with similar accuracy to single-agent case which has the advantage of agent-centric reference frame. Please note that the inference time remains the same from single-agent to multi-agent case since we predict all future trajectories in a single pass.

Multi-Agent Prediction on Interaction: We compare our method in multi-agent setting using a scene-centric reference frame to other methods on the Interaction validation set in Table 3. Our method significantly outperforms other methods with a large gap in all metrics. Impressively, it reaches 1% miss rate, showing that our method can predict future trajectories accurately for all agents in the scene.

Robustness of Adaptive Head: To evaluate the effect of noisy input data, we conducted an experiment in multi-agent setting on Interaction. Specifically, we perturb input coordinates with noise $\mathcal{N}(0, \sigma)$ where

σ	mADE ₆	mFDE ₆	MR ₆
0	0.161	0.344	0.010
0.4	0.161	0.347	0.010
0.8	0.163	0.349	0.010
1.6	0.166	0.357	0.010
3.2	0.169	0.361	0.011

Table 4: **Robustness of Adaptive Head on Interaction (Validation).** This table shows the effect of adding noise to input coordinates on multi-agent performance.

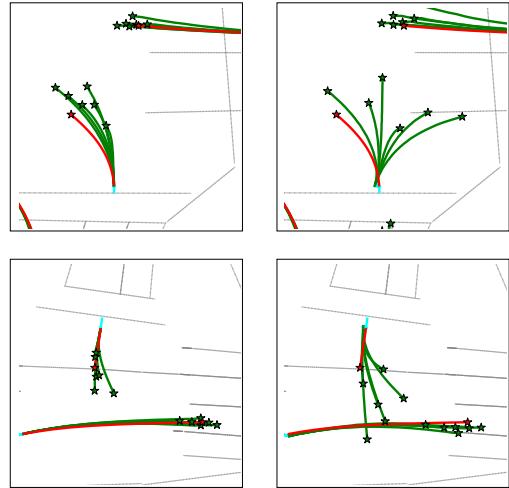


Figure 6: **Visualizing Effect of Adaptive Head.** We visualize the predictions of adaptive head (**left**) and static head (**right**). The predicted trajectories are shown in green, and ground truth in red.

$\sigma \in \{0.4, 0.8, 1.6, 3.2\}$, corresponding to an average of $\{0.32, 0.64, 1.28, 2.56\}$ meters deviation in 0.1 seconds, respectively. As shown in Table 4, the performance is quite robust to increasing noise levels in input coordinates.

4.3. Qualitative Analysis

In Fig. 5, we visualize the predictions of our model in multi-agent setting on Interaction (a) and in single-agent setting on Argoverse (b). Our model can predict accurate multi-modal trajectories for all agents in complex intersection scenarios on Interaction. In single-agent case, our model can vary predictions for the agent of interest in terms of intention and speed. Our model can successfully capture the interactions between the agents and predict futures by considering other agents, e.g. on the top left in Fig. 5b.

Visualizing Effect of Adaptive Head: To understand the importance of adaptive head, we compare the predictions of adaptive head (left) to the predictions of static head (right) in the same scenario in Fig. 6. The adaptive head significantly improves the consistency and accuracy of predictions by allowing the model to adapt to the initial state of the agent, including its rotation, location, and speed.

Understanding Dynamic Weights: To understand how the proposed adaptive prediction head changes according to each agent, we visualize dynamically learned weights for each agent by projecting them into the 3D hypersphere as shown in Fig. 7. Specifically, we project the \mathbf{W}_2 matrix (Eq. 3) for each agent to a 3-dimensional vector with PCA and normalize it to unit length as shown on top of each scene. Despite differences in absolute position, the

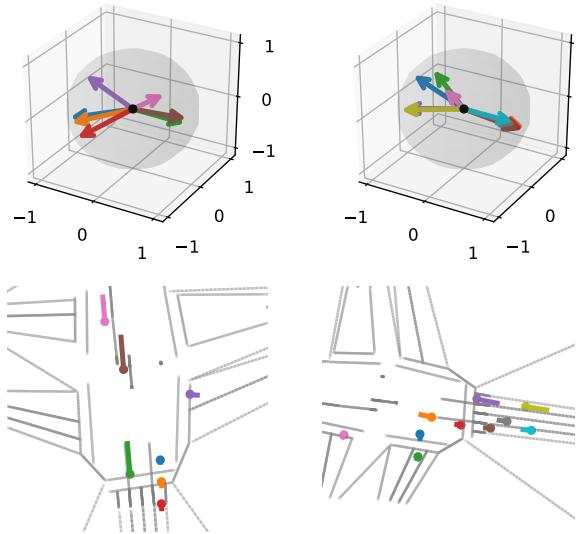


Figure 7: Visualization of Dynamic Weights. We project the dynamic weights for each agent into the 3D hypersphere (**top**) for two scenes on Interaction (**bottom**). We use the same color for the agents and their corresponding projections on the hypersphere.

learned weights for agents moving in the same lane map to similar vectors on the hypersphere. For example, the brown and green agents on the left column map to almost identical vectors, although they are spatially far from each other. A separating factor is the orientation of agents, which is preserved in the mapping. For example, the green and purple agents on the right column map along the same direction, while the orange, red, brown, and cyan agents on the opposite lane map to the opposite direction.

4.4. Ablation Study

We conduct ablation studies on the validation split of the Argoverse for single-agent setting and the validation split of the Interaction dataset for multi-agent setting.

Ablations on Single-Agent: In Table 5, we perform an ablation study on our architectural choices in single-agent

	mADE ₆	mFDE ₆	MR ₆
w/o Extended	0.694	1.000	0.090
w/o Stop Gradient	0.685	0.994	0.088
w/o Refinement	0.683	0.990	0.088
w/o Augmentation	0.675	0.974	0.089
ADAPT	0.668	0.948	0.083

Table 5: Single-Agent Ablation Study on Argoverse (Validation). This table shows the effect of removing each component on single-agent prediction on Argoverse.

	mADE ₆	mFDE ₆	MR ₆
w/o Adaptive Head	0.244	0.425	0.017
ADAPT	0.161	0.344	0.010

Table 6: Multi-Agent Ablation Study on Interaction (Validation). This table shows the effect of using the adaptive head with dynamic weights for endpoint prediction on the performance of multi-agent prediction on Interaction.

setting of the Argoverse. First, using other agents in a scene as done in previous work [53, 37, 22] improves the performance in all metrics as it provides more samples for training. Second, gradient stopping in the trajectory predictor enhances the performance by providing independent updates for endpoint refinement and trajectory scoring. Third, refinement improves the accuracy of both the endpoint and the full trajectory by improving the accuracy of the initial endpoint as well. Fourth, data augmentation increases the diversity of the training data, leading to better performance. Finally, combining all results in the best performance, proving the importance of each component and design choice.

Ablations on Multi-Agent: In Table 6, we analyze the effect of adaptive head with dynamic weights on multi-agent prediction. The results show that the performance is improved significantly with the adaptive head. This indicates that the adaptive head can adjust the weights according to the initial state of each agent.

5. Conclusion and Future Work

We propose a novel efficient framework for predicting future trajectories in both multi-agent and single-agent settings where switching between the two requires only changing the endpoint prediction head. We propose dynamic weight learning to accurately predict the endpoints of multiple agents in the same scene reference frame. We demonstrate that our model reaches state-of-the-art performance in both single-agent and multi-agent settings without increasing the model size and consequently without sacrificing efficiency in inference time.

An interesting direction for future work might be incorporating stochastic latent variables into the endpoint prediction to improve the uncertainty in future predictions, e.g. with separate latent variables for short-term and long-term goals. Another promising direction is learning the temporal dynamics of the scene to understand the relations better and improve efficiency without limiting assumptions of factorized attention. Like most of the existing work in trajectory forecasting, we assume the availability of an HD map where the past locations of agents are marked. The effect of imperfect perception on predicting future trajectories needs to be studied in future work to deploy these solutions.

References

- [1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. [2](#)
- [2] Gorkay Aydemir, Adil Kaan Akan, and Fatma Güney. Trajectory forecasting on temporal graphs. *arXiv.org*, 2207.00255, 2022. [2](#)
- [3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv.org*, 2016. [4](#), [12](#)
- [4] Yuryi Biktairov, Maxim Stebelev, Irina Rudenko, Oleh Shliazhko, and Boris Yangel. Prank: motion prediction based on ranking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. [2](#)
- [5] Thibault Buhet, Emilie Wirbel, Andrei Bursuc, and Xavier Perrotton. Plop: Probabilistic polynomial objects trajectory prediction for autonomous driving. In *Proc. Conf. on Robot Learning (CoRL)*, 2021. [2](#)
- [6] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giacarlo Baldan, and Oscar Beijbom. nuscenes: A multi-modal dataset for autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. [1](#), [3](#)
- [7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. [2](#)
- [8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. [2](#)
- [9] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Proc. Conf. on Robot Learning (CoRL)*, 2018. [2](#)
- [10] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Proc. Conf. on Robot Learning (CoRL)*, 2020. [2](#)
- [11] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. [1](#), [2](#), [3](#), [5](#)
- [12] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2019. [2](#)
- [13] Fang Da and Yu Zhang. Path-aware graph attention for hd maps in motion prediction. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2022. [1](#), [6](#)
- [14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv.org*, 2018. [2](#)
- [15] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Nitin Singh, and Jeff Schneider. Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020. [2](#)
- [16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021. [2](#)
- [17] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding HD maps and agent dynamics from vectorized representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. [2](#), [3](#), [12](#), [13](#)
- [18] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv.org*, 2020. [3](#)
- [19] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. In *Proc. IEEE Conf. on Intelligent Transportation Systems (ITSC)*, 2021. [2](#)
- [20] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output for future motion estimation. In *ICRA*, 2022. [6](#)
- [21] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Thomas: Trajectory heatmap output with learned multi-agent sampling. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2022. [1](#), [2](#), [3](#), [6](#), [7](#)
- [22] Roger Girgis, Florian Golemo, Felipe Codevilla, Martin Weiss, Jim Aldon D’Souza, Samira Ebrahimi Kahou, Felix Heide, and Christopher Pal. Latent variable sequential set transformers for joint multi-agent motion prediction. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2022. [1](#), [3](#), [4](#), [5](#), [6](#), [7](#), [8](#), [13](#)
- [23] Junru Gu, Chen Sun, and Hang Zhao. DenseTNT: End-to-end trajectory prediction from dense goal sets. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. [1](#), [2](#), [3](#), [6](#), [13](#)
- [24] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. [2](#)
- [25] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2021. [2](#)
- [26] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the road: Predicting driving behavior with a convolutional model

- of semantic interactions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2
- [27] Baoxiong Jia, Yu Liu, and Siyuan Huang. Improving object-centric learning with query optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2023. 5
- [28] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 2, 4
- [29] Siddhesh Khandelwal, William Qi, Jagjeet Singh, Andrew Hartnett, and Deva Ramanan. What-if motion prediction for autonomous driving. *arXiv.org*, 2008.10587, 2020. 2
- [30] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 5
- [31] Stepan Konev, Kirill Brodt, and Artyom Sanakoyeu. Motioncnn: A strong baseline for motion prediction in autonomous driving. *arXiv.org*, 2206.02163, 2022. 2
- [32] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 1, 2, 3, 4, 6, 13
- [33] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 3, 6
- [34] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2
- [35] Jean Mercat, Thomas Gilles, Nicole El Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Pita Gil. Multi-head attention for multi-modal joint vehicle motion forecasting. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2020. 2
- [36] Xiaoyu Mo, Yang Xing, and Chen Lv. Recog: A deep learning framework with heterogeneous graph for interaction-aware trajectory prediction. *arXiv.org*, 2012.05032, 2020. 2
- [37] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified architecture for predicting multiple agent trajectories. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2022. 1, 2, 3, 4, 5, 6, 7, 8, 13
- [38] Seong Hyeon Park, Gyubok Lee, Jimin Seo, Manoj Bhat, Minseok Kang, Jonathan Francis, Ashwin Jadhav, Paul Pu Liang, and Louis-Philippe Morency. Diverse and admissible trajectory forecasting through multimodal context understanding. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 2
- [39] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2
- [40] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 2
- [41] Haoran Song, Di Luan, Wenchao Ding, Michael Y Wang, and Qifeng Chen. Learning to predict vehicle trajectories with model-based planning. In *Proc. Conf. on Robot Learning (CoRL)*, 2021. 2, 14
- [42] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3
- [43] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2
- [44] Charlie Tang and Russ R Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2
- [45] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 2, 4
- [46] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, et al. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2022. 6
- [47] 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 (NeurIPS)*, 2017. 2, 3
- [48] Jingke Wang, Tengju Ye, Ziqing Gu, and Junbo Chen. Ltp: Lane-based trajectory prediction for autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 3, 6
- [49] Maosheng Ye, Tongyi Cao, and Qifeng Chen. TPCN: Temporal point cloud networks for motion forecasting. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 3, 6, 14
- [50] Wenyuan Zeng, Ming Liang, Renjie Liao, and Raquel Urtasun. Lanercnn: Distributed representations for graph-centric motion forecasting. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2021. 2
- [51] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kümmerle, Hendrik Königshof, Christoph Stiller, Arnaud de La Fortelle, and Masayoshi Tomizuka. INTERACTION Dataset: An INTERNATIONAL, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. *arXiv.org*, 2019. 2, 3, 5
- [52] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai,

- Cordelia Schmid, et al. TNT: Target-driven trajectory prediction. In *Proc. Conf. on Robot Learning (CoRL)*, 2021. 2, 3
- [53] Zikang Zhou, Luyao Ye, Jianping Wang, Kui Wu, and Kejie Lu. Hivt: Hierarchical vector transformer for multi-agent motion prediction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 3, 5, 6, 8, 13

Supplementary Material for “ADAPT: Efficient Multi-Agent Trajectory Prediction with Adaptation”

Abstract

In this supplementary, we present the details of implementation (Section 1) and training (Section 2), perform a complexity analysis of our method compared to other methods (Section 3), provide an additional ablation study on the model components (Section 4), identify failure cases of our method (Section 5), and finally visualize additional qualitative results (Section 6). We also visualize the result of multiple attention heads in Section 6 specializing in different areas and agents in the scene, which might help future work on interpretability.

1. Implementation Details

Scene Representation: We follow VectorNet [17] in our polyline subgraph implementation to obtain the updated node features \mathbf{v}_i of the subgraph as follows:

$$\mathbf{v}_i^{(l+1)} = \text{cat} \left(\text{MLP}_l(\mathbf{v}_i^{(l)}), \text{pool}(\text{MLP}_l(\mathbf{v}^{(l)})) \right)$$

where $\mathbf{v}_i^{(l)}$ denotes the feature vector of agent i at layer l , pool the max-pool operation, and cat the concatenation operation. We use an MLP with 2 linear layers and ReLU for non-linearity with a layer normalization [3] after the first layer. We set the layer number l to 3 and the size of the feature vector to 128 for both the agent and the lane subgraph.

Interaction Modelling: For each multi-head attention block (MHAB), we set the number of attention heads to 8 and apply a dropout rate of 0.1 to the attention probabilities. We set the size of the hidden layer in the feed-forward networks to 128 and the number of iterations L to 3.

Meta Info: Meta info includes the location of the agent at time t , $t - 1$, and the yaw angle at t . Locations are in 2D coordinates and the angle is in radians, resulting in a 5-dimensional vector. We concatenate the meta info to the corresponding agent feature before decoding.

Trajectory Predictor: For both dynamic and static heads, we use a 2-layer MLP with ReLU for the non-linearity and a layer normalization [3] after the first layer. Differently from subgraphs, we use residual connections in the last layer.

2. Training Details

As mentioned in the paper, we use the variety loss to capture multi-modal futures by calculating the loss only for the most accurate trajectory over K predicted ones. Given the ground truth trajectory $\{\mathbf{s}_t\}_{t=1}^T$ and the predicted trajectory with the closest endpoint $\{\hat{\mathbf{s}}_t\}_{t=1}^T$ for T future steps, we train our model using the endpoint loss \mathcal{L}_{end} , the full trajectory loss \mathcal{L}_{traj} , and the trajectory classification loss \mathcal{L}_{cls} . \mathcal{L}_{end} is the difference between the closest endpoint and the ground truth endpoint:

$$\mathcal{L}_{end} = \mathcal{L}_{\text{Smooth-}\ell_1}(\hat{\mathbf{s}}_T, \mathbf{s}_T) \quad (4)$$

where $\hat{\mathbf{s}}_T$ is the endpoint of $\hat{\mathbf{s}}$, i.e. the prediction at time T . \mathcal{L}_{traj} is the mean of the per-step difference between the predicted full trajectory, $\hat{\mathbf{s}}$, and the ground truth trajectory, \mathbf{s} :

$$\mathcal{L}_{traj} = \frac{1}{T} \sum_{t=1}^T \mathcal{L}_{\text{Smooth-}\ell_1}(\hat{\mathbf{s}}_t, \mathbf{s}_t) \quad (5)$$

Finally, \mathcal{L}_{cls} is the Binary Cross Entropy Loss applied to the assigned probabilities \mathbf{p} of K trajectories where the ground truth probability of the closest trajectory $\hat{\mathbf{s}}$ is set to 1 and the others to 0:

$$\mathcal{L}_{cls} = \mathcal{L}_{\text{BCE}}(\mathbf{p}, \mathbf{y}) \quad (6)$$

where \mathbf{y} denotes the ground truth probabilities assigned. Overall, our loss is the sum of these three losses:

$$\mathcal{L} = \mathcal{L}_{end} + \mathcal{L}_{traj} + \mathcal{L}_{cls} \quad (7)$$

3. Computational Complexity

In this section, we provide a comparison of the computational complexity according to the attention operations used in the existing approaches. We first define variables that define the number of elements. N , M , and T correspond to the number of agents, lane elements, and time steps, respectively. T can be decomposed into two variables, T_p and T_f , which refer to past and future time steps, respectively. In general, the number of agents dominates the computation, then, the number of lanes followed by the fixed number of time steps, e.g. $T = 50$ ($N > M > T$). While the number

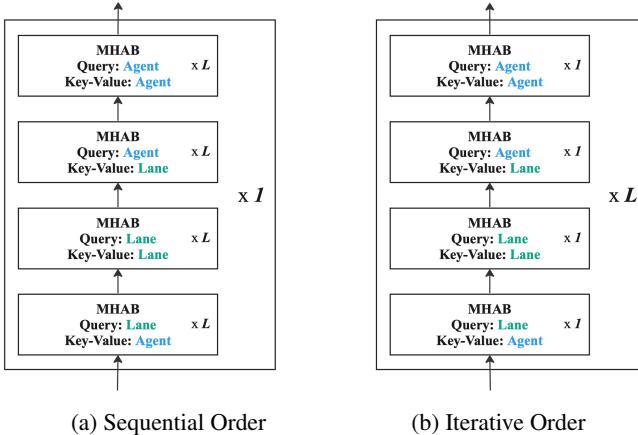


Figure 1: The Order of Attention in Interaction. Given a number of layers L , there are two ways of applying attention to model interactions: sequential and iterative. In sequential order **a**, each type of interaction is considered L times sequentially. In iterative order **b**, each type of interaction is considered once in a single pass and the pass is repeated L times.

of lanes M stays mostly uniform across scenes, the number of agents N might vary significantly even for the same scene.

As addressed in the SceneTransformer [37], directly applying attention to both time and agent axes results in high overhead, with the computational complexity of $\mathcal{O}((NT + M)^2)$ where N is the number of agents, M is the number of lane segments and T is the number of time steps including both past and future. SceneTransformer reduces it to $\mathcal{O}(NT^2 + N^2T + NTM)$ with factorized attention over time and agent axes.

Autobot [22] does not include lane elements in their factorized attention steps. Contrary to SceneTransformer, their encoding and decoding phases consider only past and future time steps, respectively, resulting in the complexity of $\mathcal{O}(NT_p^2 + N^2T_p + NT_f^2 + N^2T_f)$ where T_p denotes the number of past time steps and T_f denotes the number of future time steps.

HiVT [53] does not use the standard multi-axis factorized attention but embraces a more efficient type of temporal interaction by considering only one agent for each time step and attending to only one feature over different time steps. Since HiVT follows an agent-centric approach and calculates agent features independently from each other, considering only one agent in their local scene does not result in information loss. However, the agent-centric approach comes with the overhead of N runs of the same procedure. Considering scene normalization for each agent and global interaction in the end, HiVT has the overall complexity of $\mathcal{O}(N^2T_p + NT_p^2 + NM)$.

ADAPT has a clear advantage in terms of computational complexity over the existing approaches. Our computation is not bounded by T as our subgraphs in vectorized encoder handle the temporal reasoning. Since we calculate the attention over only agents and lanes, ADAPT has the complexity of $\mathcal{O}(N^2 + NM + M^2)$ resulting from the attention operations in the interaction modeling. Removing the number of time steps T out of the equation is the main reason behind the efficiency gain of ADAPT.

4. Quantitative Results

In this section, we present an additional ablation study to justify some minor design choices. Specifically, we investigated the effect of iterative vs. sequential order in interaction (Fig. 1) and the effect of using two separate subgraphs for encoding agents and lanes. The results in Table 1 show that the iterative attention blocks outperform their sequential counterpart. This implies that updating intermediate features at each iteration, as opposed to the attention order used in LaneGCN [32], leads to a better understanding of the relationship between agents and lanes. Furthermore, the use of separate polyline subgraphs for lanes and agents, which is in contrast to prior work [23, 17], produces better results. Overall, our decision choices on the architecture improve performance with better feature encoding.

5. Failure Cases

In this section, we provide some failure cases and investigate possible reasons. We perform the analysis on single-agent predictions on Argoverse, since the miss rate in single-agent prediction is relatively higher than multi-agent predictions on Interactions. We identify three sources of error for failure cases: erroneous data, missing rare behaviors, and inaccurate predictions.

Erroneous Data: The accuracy of the provided input trajectories in the past directly affects the future predictions, since the future predictions are trained to be consistent with the past ones. Thus, defective or unstable history data causes incorrect future predictions as shown in Fig. 2a.

	mADE ₆	mFDE ₆	MR ₆
w/o Iterative Att.	0.673	0.971	0.086
w/o Dual Subgraph	0.671	0.960	0.086
ADAPT	0.668	0.948	0.083

Table 1: Single-Agent Ablation Study on Argoverse (Val.). This table shows the effect of iterative attention and dual subgraph on the performance of single-agent prediction on the Argoverse validation set.

Moreover, defects in the future steps result in inaccurate evaluations of the predictions (Fig. 2b). Some problems such as id-switch and position-oscillation, resulting in unstable and incorrect ground truth future locations, are addressed in previous works as well [49, 41].

Additionally, accurate map information plays an important role in future predictions because it directly affects the reasoning of the model about drivable areas. In the example shown in Fig. 2c, predictions are intensified on a single mode i.e. left turn, because of the missing lane that the ground truth trajectory follows.

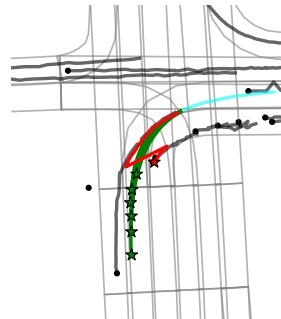
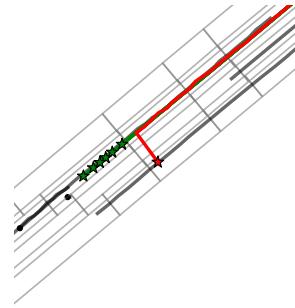
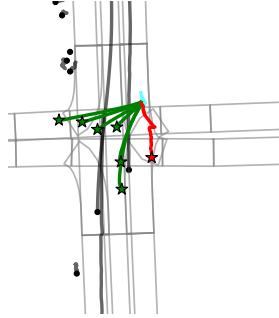
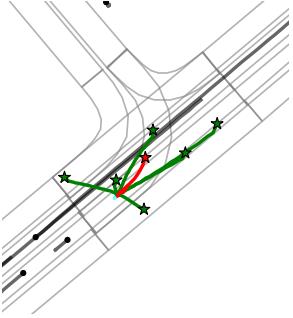
Missing a Peculiar Mode: Despite the large number of scenarios on Argoverse, some behaviors are less frequently observed such as a u-turn or an abrupt lane change. These behaviors that are rare on the training set cause the model to miss the relevant mode at test time as shown in Fig. 2d.

Precision of Predictions: Some predictions result in an error due to a lack of precision in the predicted trajectories despite correctly identifying intention. For example, in Fig. 2e, all possible paths are covered by the predictions but the difference between the closest endpoint and the ground truth endpoint is higher than the miss rate threshold.

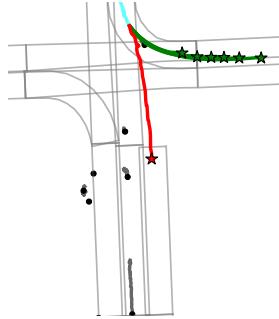
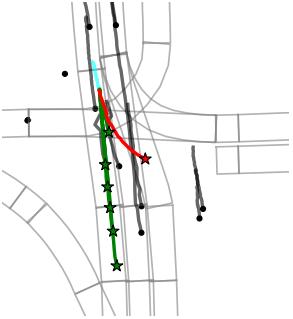
6. Qualitative Results

In this section, we provide additional qualitative results for both single-agent (Fig. 3) and multi-agent (Fig. 4) predictions of ADAPT on Argoverse and Interaction validation sets, respectively.

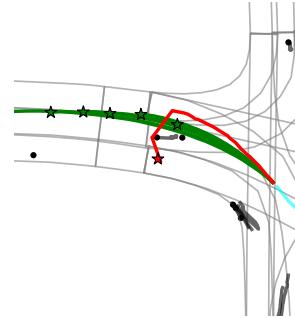
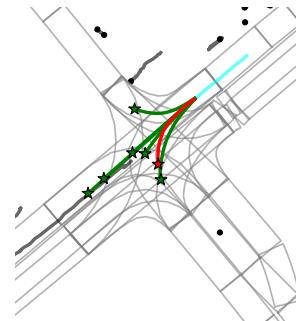
Focus of Attention Heads: In Fig. 5, we visualize attention scores from different MHAB heads that are used to update an agent (red) in scenes from the validation split of the Interaction dataset. The attention scores are gathered from **AA** and **LA** modules for agents and lanes, respectively. In the first layer of interaction (the first row), attention heads do not focus on any specific scene elements yet as this is the first step where the agent is informed by the scene. As the agent has no prior information, attending to all scene elements without focusing on any is a reasonable choice in the first layer. On the other hand, in the next layer, each attention head specializes in some part of the map. For example, in the scene given in the upper set of rows, head 2 attends to lanes in the upper left of the map whereas head 4 attends to the upper right lanes. Some **AA** heads attend only to the agent itself and do not consider any other agents, e.g. head 3 and head 4. In the last layer, heads still attend to some specific parts of the map and a subset of the agents. These visualizations confirm that different types of interactions are captured with the multi-head attention blocks of our model.



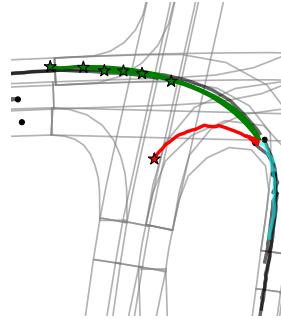
(a) **Erroneous Input Trajectories.** The errors in the given trajectories, both past, and future, result in unreasonable futures due to inconsistent and uninformative past locations.



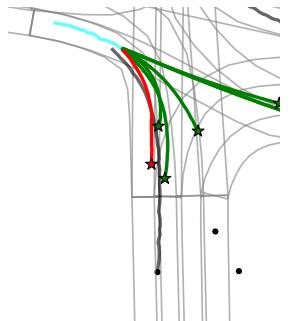
(c) **Problems in the Input Map.** The missing lane information on the map causes the model to miss a possible future path.



(b) **Erroneous Ground Truth.** Erroneous (impossible) ground truth evaluates admissible trajectories as failures.



(d) **Missing a Mode.** Rare behaviors cause the model to miss the mode corresponding to the ground truth.



(e) **Inaccurate Predictions.** The difference between the predicted and the ground truth endpoints is higher than the miss rate threshold, therefore this case is classified as a failure although ground truth intention is correctly captured by the predictions.

Figure 2: **Failure Cases on Argoverse.** We present some failure cases with their potential reasons.

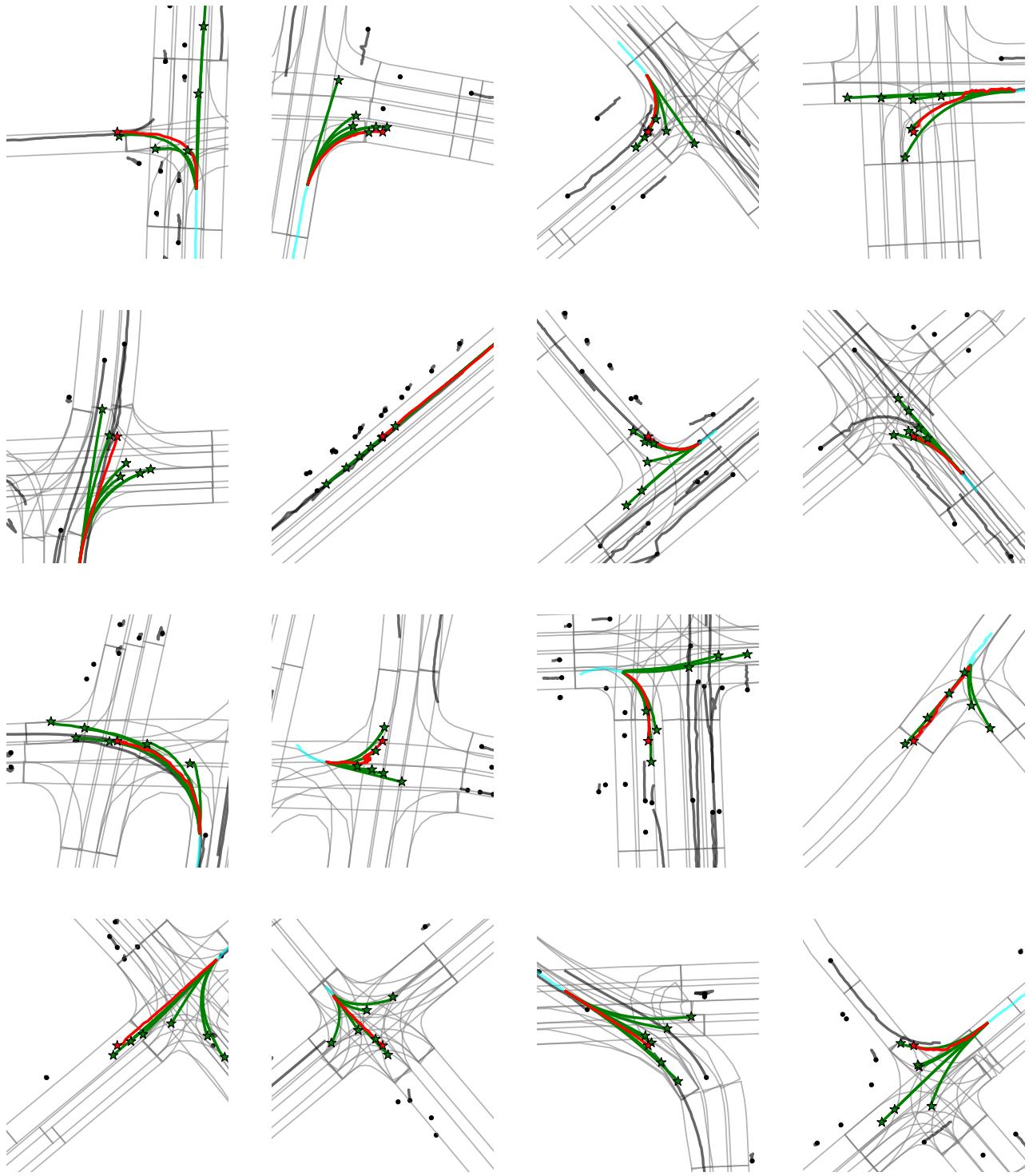


Figure 3: Additional Qualitative Results for Single-Agent Predictions on Argoverse. The red colored trajectory shows the ground truth future, the cyan shows the past trajectory of the agent of interest, and the green trajectories are the multiple predictions. Context agents are displayed in black. ADAPT can successfully predict a trajectory similar to the ground truth by also covering possible diverse trajectories for agent of interest.

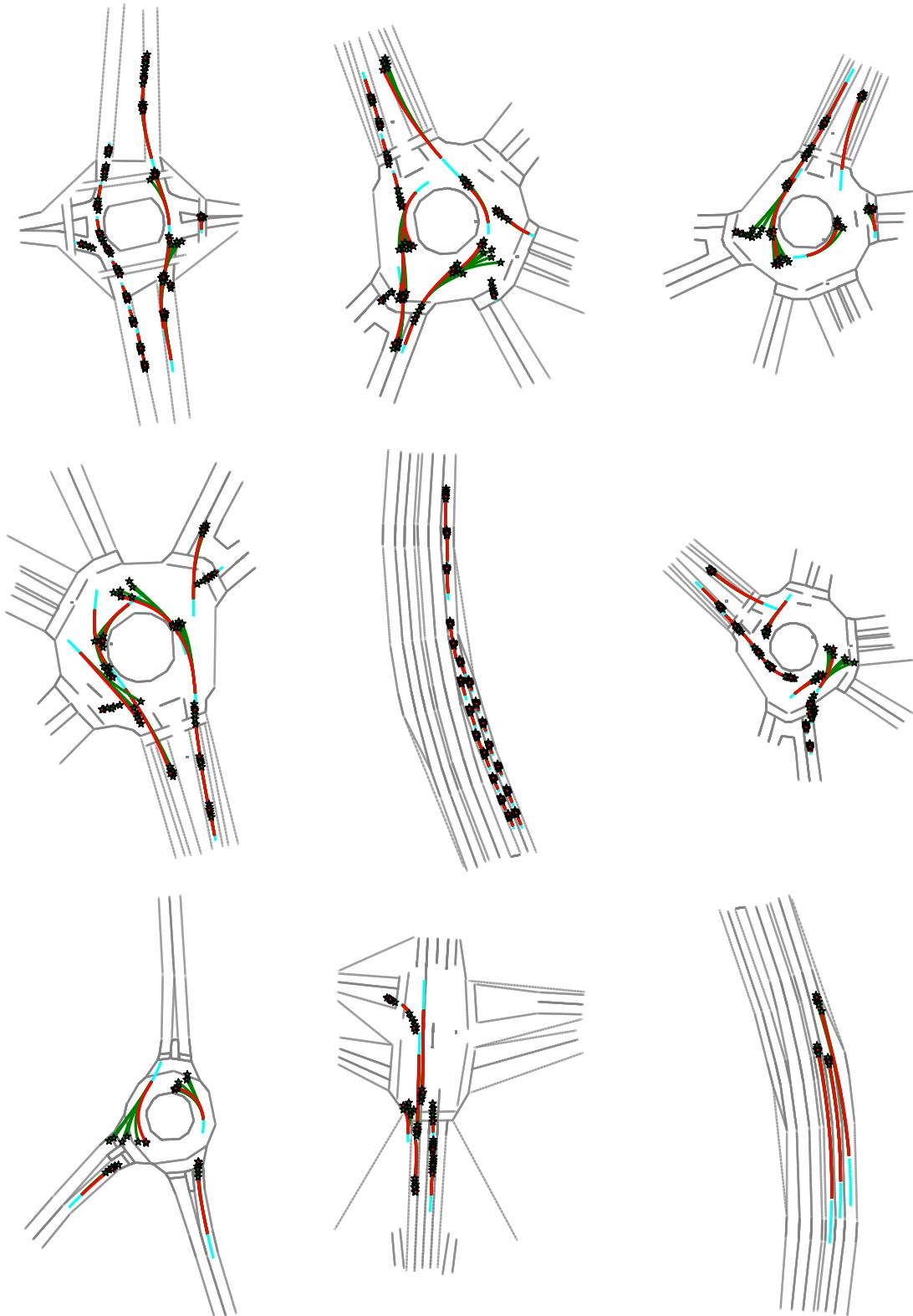


Figure 4: Additional Qualitative Results for Multi-Agent Predictions on Interaction. The red colored trajectories show the ground truth future for each agent, the cyan shows the past trajectories of the agents, and the green trajectories are the predictions. ADAPT can successfully predict futures for each agent in a single forward pass without introducing additional overhead.



Figure 5: Visualization of Attention Scores on the Interaction. We visualize multi-head attention from different layers for a selected agent (red). The attention probabilities for the agents (blue) and lanes (green) are the results of the Agent-Agent and the Lane-Agent modules, respectively. The transparency increases with lower attention probabilities. As the attention propagates towards higher layers, the attention heads specialize towards specific components such as lanes in the right turn, the vehicle in front, etc.