ADAPT: Efficient Multi-Agent Trajectory Prediction with Adaptation

Görkay Aydemir¹, Adil Kaan Akan¹,², Fatma Güney¹,²
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 methods [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

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].
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 one of the smallest and fastest models 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 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

![Scene-Centric vs. Agent-Centric Representation](image_url)

**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.
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, HiVT [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, $A = \{a_i\}$ where $a_i \in \mathbb{R}^{T \times 2}$ denotes the location of agent $i$ at previous $T$ time steps and HD map, $M = \{m_i\}$ where $m_i \in \mathbb{R}^{l_i \times 2}$ 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 $d$, $f_i \in \mathbb{R}^d$ 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-agent (LL), lane-to-agent (LA), and agent-to-agent (AA). Using attention, we update each feature $f_i$ 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:

$$\text{MHA}(f_{q}, f_{kv}) = \text{softmax} \left( \frac{Q K^T}{\sqrt{\text{dim}_k}} \right) V$$

(1)

where $Q, K, V = W^q f_q, W^k f_{kv}, W^v f_{kv}$

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

\[
\text{MHAB}(f_{q}, f_{kv}) = \text{norm}(\tilde{f} + \text{FFN}(\tilde{f}))
\]

where \( \tilde{f} = \text{norm}(f_q + \text{MHA}(f_q, f_{kv})) \)

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.

\[
\begin{align*}
W_1 &= W_{d_1} \tilde{f} \\
W_2 &= W_{d_2} \tilde{f} \\
F_{d_1} &= \text{ReLU}(\text{norm}(W_1f)) \\
\hat{y}_{\text{pred}} &= W_2F_{d_1}
\end{align*}
\]

We visualize the adaptive head in Fig. 3 and explain it mathematically in (3) where \( W_{d_1} \) and \( W_{d_2} \) are trainable parameters and norm is the layer normalization. We process the encoded agent features concatenated with meta info, \( f \) with an MLP and obtain \( \tilde{f} \).
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 Interactive [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 \( \ell_2 \) 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 \((1−p)^2 + mFDE_k\) 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 \( 1 \times 10^{-4} \) and \( 2 \times 10^{-4} \) 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.
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.

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

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-mFDE<sub>k</sub> 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 mFDE<sub>k=6</sub> and MR<sub>6</sub>. 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 processing.

<table>
<thead>
<tr>
<th>Model</th>
<th>mADE&lt;sub&gt;k=6&lt;/sub&gt;</th>
<th>mFDE&lt;sub&gt;k=6&lt;/sub&gt;</th>
<th>MR&lt;sub&gt;6&lt;/sub&gt;</th>
<th>Inf. (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPCN [49]</td>
<td>0.73</td>
<td>1.15</td>
<td>0.11</td>
<td>-</td>
</tr>
<tr>
<td>mmTrans [33]</td>
<td>0.71</td>
<td>1.15</td>
<td>0.11</td>
<td>7.66</td>
</tr>
<tr>
<td>LaneGCN [32]</td>
<td>0.71</td>
<td>1.08</td>
<td>-</td>
<td>38.37</td>
</tr>
<tr>
<td>LTP [48]</td>
<td>0.78</td>
<td>1.07</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DenseTNT [23]</td>
<td>0.73</td>
<td>1.05</td>
<td>0.10</td>
<td>444.66</td>
</tr>
<tr>
<td>HiVT [53]</td>
<td><strong>0.66</strong></td>
<td>0.96</td>
<td><strong>0.09</strong></td>
<td>64.45</td>
</tr>
<tr>
<td>PAGA [13]</td>
<td>0.69</td>
<td>1.02</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours (Multi)</td>
<td>0.67</td>
<td><strong>0.95</strong></td>
<td><strong>0.08</strong></td>
<td>11.31</td>
</tr>
<tr>
<td>Ours (Single)</td>
<td>0.65</td>
<td>0.97</td>
<td>0.08</td>
<td>11.31</td>
</tr>
</tbody>
</table>

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.
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-mFDE 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.

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

<table>
<thead>
<tr>
<th>σ</th>
<th>mADEₜₜ</th>
<th>mFDEₜₜ</th>
<th>MRₜₜ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.161</td>
<td>0.344</td>
<td>0.010</td>
</tr>
<tr>
<td>0.4</td>
<td>0.161</td>
<td>0.347</td>
<td>0.010</td>
</tr>
<tr>
<td>0.8</td>
<td>0.163</td>
<td>0.349</td>
<td>0.010</td>
</tr>
<tr>
<td>1.6</td>
<td>0.166</td>
<td>0.357</td>
<td>0.010</td>
</tr>
<tr>
<td>3.2</td>
<td>0.169</td>
<td>0.361</td>
<td>0.011</td>
</tr>
</tbody>
</table>

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.

σ ∈ {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 interaction 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 $W_2$ 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 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.

<table>
<thead>
<tr>
<th></th>
<th>mADE₆</th>
<th>mFDE₆</th>
<th>MR₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o Extended</td>
<td>0.694</td>
<td>1.000</td>
<td>0.090</td>
</tr>
<tr>
<td>w/o Stop Gradient</td>
<td>0.685</td>
<td>0.994</td>
<td>0.088</td>
</tr>
<tr>
<td>w/o Refinement</td>
<td>0.683</td>
<td>0.990</td>
<td>0.088</td>
</tr>
<tr>
<td>w/o Augmentation</td>
<td>0.675</td>
<td>0.974</td>
<td>0.089</td>
</tr>
<tr>
<td>ADAPT</td>
<td><strong>0.668</strong></td>
<td><strong>0.948</strong></td>
<td><strong>0.083</strong></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>mADE₆</th>
<th>mFDE₆</th>
<th>MR₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o Adaptive Head</td>
<td>0.244</td>
<td>0.425</td>
<td>0.017</td>
</tr>
<tr>
<td>ADAPT</td>
<td><strong>0.161</strong></td>
<td><strong>0.344</strong></td>
<td><strong>0.010</strong></td>
</tr>
</tbody>
</table>

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.
References


[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


[38] Seong Hyeon Park, Gyubok Lee, Jimin Seo, Manoj Bhat, Mineok 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
[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
[52] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai,