Trajectory Entropy: Modeling Game State Stability From Multimodal Trajectory Prediction

Yesheng Zhang, Wenjian Sun, Yuheng Chen, Qingwei Liu, Qi Lin, Rui Zhang, and Xu Zhao, *Member, IEEE*

***Abstract*—Complex interactions among agents present a significant challenge for autonomous driving in real-world scenarios. Recently, a promising approach has emerged, which formulates the interactions of agents as a *level-k* game framework. It effectively decouples agent policies by hierarchical game levels. However, this framework ignores both the varying driving complexities among agents and the dynamic changes in agent states across game levels, instead treating them uniformly. Consequently, redundant and error-prone computations are introduced into this framework. To tackle the issue, this paper proposes an indicator termed *Trajectory Entropy* to reveal the game status of agents within the *level-k* game framework. The key insight stems from recognizing the practical relationship between the agent policy uncertainty and the associated driving complexity. Specifically, Trajectory Entropy derives an entropy-inspired indicator from confidence-weighted dispersion across multimodal trajectory predictions, capturing the stability of agent states in the game. It adopts a signal-to-noise-ratio-inspired formulation to jointly account for spatial trajectory dispersion and prediction confidence. Based on the proposed Trajectory Entropy, we refine the current *level-k* game framework through a simple gating mechanism, significantly improving overall accuracy while reducing computational costs. Our method is evaluated on the Waymo and nuPlan datasets, in terms of trajectory prediction, open-loop and closed-loop planning tasks. The results demonstrate consistent improvements over GameFormer, with prediction error reduced by up to 32.88% and planning performance improved by up to 16.48%.**

***Index Terms*—Trajectory prediction, autonomous driving, agent planning.**

I. Introduction

JOINT trajectory prediction and ego vehicle planning have been demonstrated as a promising approach to achieve intelligent Autonomous Driving (AD) [1], [2], [3], [4], [5]. The approach takes an end-to-end paradigm, enhancing the interaction between agents, which is a desirable way for autonomous driving [6].

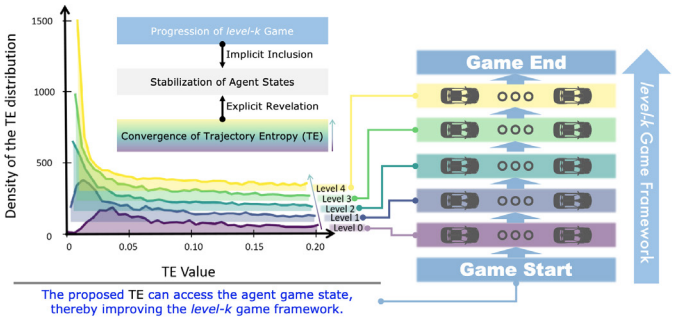


Fig. 1. **The stability of agent states within the *level-k* game can be effectively characterized by the proposed Trajectory Entropy (TE).** While recent *level-k* game framework in GameFormer is effective for joint trajectory prediction and planning, it treats all agents uniformly across game levels, leading to redundant computation. We therefore introduce TE to quantify agent states and enable adaptive processing. We analyze the TE distribution across five game levels of GameFormer on 25,000 randomly sampled Waymo scenes. The results demonstrate that the TE distribution progressively converges toward lower values as the game level increases, indicating that agents stabilize at deeper game levels. This trend validates TE as a robust indicator of game state convergence of agents, providing the foundation for our mechanism to mitigate computational redundancy by early-gating stable agents.

However, precise trajectory prediction and effective planning remain highly challenging in complex driving environments. This primarily arises from the tightly coupled nature of the AD task, that is, the trajectory of one vehicle is influenced by other vehicles, and vice versa. To address the tight-coupling issue, GameFormer [3] draws inspiration from *level-k* game theory [7] and treats ego planning on par with the trajectory prediction. It incorporates all interested agents into a hierarchical game framework with k levels. Within each level, Multimodal Trajectory Prediction (MTP) is performed for every agent. At its core, the prediction of each agent at every level only receives the game results of other agents from the preceding level as input, in line with the principles of *level-k* game theory. Consequently, this framework effectively disentangles agent interactions into different game levels, facilitating intelligent autonomous driving.

Nevertheless, in the hierarchical game framework, treating all agents equally as in GameFormer could introduce computational redundancy. In a given driving scene, agents may face varying levels of decision-making complexity due to their spatial positions and surrounding environments, such as agents in Fig. 2. This leads to different levels of complexity among agents within the game framework. For example, some agents with low driving difficulty may only require a few

Received 6 June 2025; revised 23 April 2026 and 3 July 2026; accepted 20 August 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62176156, in part by Shanghai Municipal Science and Technology Explorers Program under Grant TS251414000, and in part by Z-ONE Technology Company Ltd., Shanghai, China, under Grant SJTU-Z-ONE-2024004. The Associate Editor for this article was B. Chen. *(Yesheng Zhang and Wenjian Sun contributed equally to this work.) (Corresponding author: Xu Zhao.)*

Yesheng Zhang, Wenjian Sun, Yuheng Chen, and Xu Zhao are with the School of Automation and Intelligent Sensing, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: preacher@sjtu.edu.cn; sun.wen.jian@sjtu.edu.cn; chenyuheng@sjtu.edu.cn; zhaoxu@sjtu.edu.cn).

Qingwei Liu, Qi Lin, and Rui Zhang are with Z-ONE Technology Company Ltd., Shanghai 201100, China (e-mail: liuqingwei01@saicmotor.com; linqi01@saicmotor.com; zhangrui07@saicmotor.com).

Digital Object Identifier 10.1109/TITS.2026.3728462

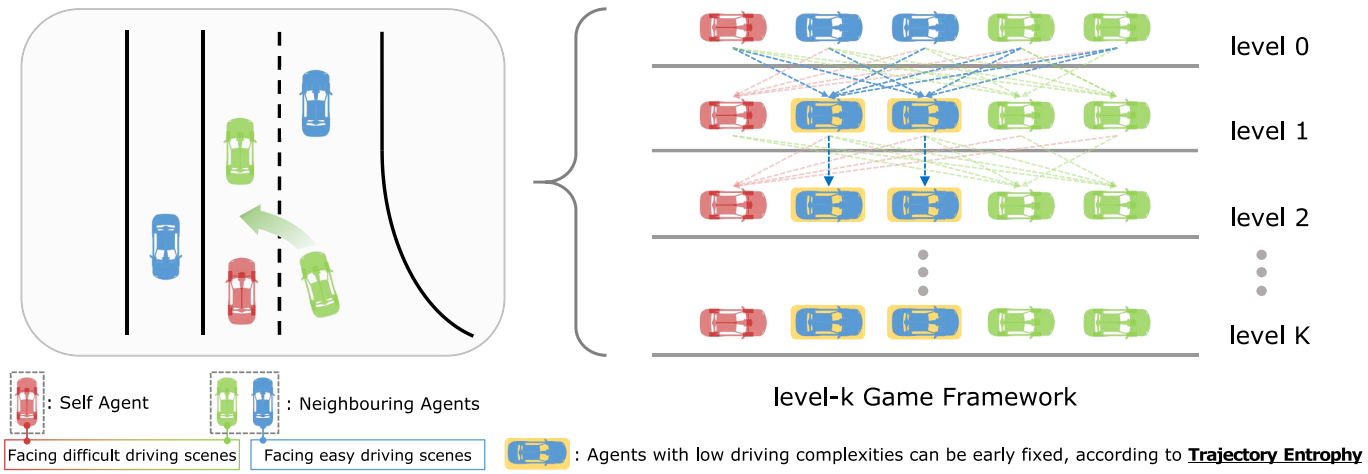


Fig. 2. **The motivation of our Trajectory Entropy.** In driving scenes, agents can face different driving complexities, as illustrated by the diverse vehicles in the diagram. Consequently, in the hierarchical game framework, these agents require tailored depths of reasoning. To this end, we propose *Trajectory Entropy* to measure the driving difficulty of agents, enabling the allocation of appropriate game depths to each agent.

game levels to determine their future trajectories. However, high-difficulty agents may need more game levels for deeper reasoning about their policy. Therefore, including agents with already stable game states in unnecessary reasoning levels not only increases computational overhead but also introduces noise into the overall framework, thereby negatively impacting the final accuracy.

This issue of redundant reasoning has also been observed in Large Language Models (LLMs), whose cascaded attention layers lead to "overthinking" [8]. To address this, DeeBert [9] and PABEE [10] introduce the "Early Exit" paradigm. Inspired by uncertainty estimation frameworks [11], [12], these methods progressively evaluate the uncertainty of intermediate attention layers, terminating inference once an uncertainty metric falls below a predefined threshold or a consensus among predictions is reached. GameFormer also predicts trajectory confidence for individual modes, rendering it compatible with this Early Exit mechanism. However, uncertainty of individual trajectory modes is insufficient to reveal the overall complexity of the driving scene. Crucially, the spatial dispersion among multimodal trajectories also serves as a physical indicator of agent stability. Therefore, to effectively mitigate "overthinking" in GameFormer, a robust uncertainty indicator is required, which should incorporate both per-mode confidence and cross-mode spatial consensus of agent trajectories.

To this end, this work introduces an indicator named *Trajectory Entropy* (TE), which can effectively reveal agent state stability/uncertainty in the *level-k* game. As illustrated in Fig. 1, we analyze the distribution of the TE value across the game levels of GameFormer using 25,000 test scenarios from the Waymo dataset. A clear downward trend in TE is observed as the game level increases, indicating that TE effectively captures the progressive stabilization of agents within the *level-k* game framework. The key insight of this ability is that the MTP results of each game round inherently reflect the corresponding game state. Particularly, MTP aims to provide multiple reasonable prediction outcomes to cover all feasible trajectories [13]. In the context of autonomous driving, the distribution of feasible trajectories for each agent potentially reflects the level of driving complexity they encounter. For instance, if few agent interactions exist, the distribution of feasible trajectories for agents is naturally concentrated near the road's centerline, according to the traffic rules, indicating low driving difficulty (see blue cars in Fig. 2). On the other hand, in complex scenes with more agent interactions, the feasible trajectory distribution of an agent is more dispersed, requiring multiple game rounds and negotiations with other agents (see red and green cars in Fig. 2). Thus, the MTP results, which model the feasible trajectory distribution, are able to offer clues of the agent states in the *level-k* game. Based on this, *Trajectory Entropy* evaluates the game states of agents by measuring the dispersion of MTP results, enabling the elimination of redundant and error-prone computation in the *level-k* game framework. This ultimately enhances the performance of trajectory prediction and planning.

Specifically, inspired by random signal processing theory [14], we treat the distances among trajectories from different modalities as signals reflecting the degree of dispersion, and link prediction confidence to the accompanying random noise. Then, theSignal-to-Noise Ratio (SNR) of this random signal could reflect the degree of dispersion in MTP, thereby serving as an indicator to reveal the game states of agents in the *level-k* game. Intuitively, this indicator serves to characterize the uncertainty and disorder of the agent's state. Drawing an analogy to information entropy, which fundamentally measures the uncertainty of information, we term this indicator *Trajectory Entropy*.

Similar to the Early Exit strategy in LLMs [9], [10], we integrate the *Trajectory Entropy* into the *level-k* game framework of GameFormer, aiming to reduce computational costs and enhance accuracy at the same time. In practice, *Trajectory Entropy* values are computed from the MTP results for agents in the first level of the game. Subsequently, agents with *Trajectory Entropy* values below a certain threshold are considered to have stable game states. Then, a simple gate is employed to fix them in subsequent games, avoiding redundant

computations on these agents and reducing uncertainty in subsequent games as well (Fig. 2 right). For unstable agents, their *Trajectory Entropy* values are repeatedly evaluated at each game level to determine whether to fix them, until the final level. With this gate module, the performance of GameFormer can be effectively enhanced.

Currently, GameFormer is the only algorithm that adopts the *level-k* game framework for joint trajectory prediction and decision-making. However, driving behavior inherently involves policy negotiation among agents, necessitating a game-theoretic framework. Therefore, the proposed *Trajectory Entropy*, as a general indicator of game state stability, is expected to be applicable to a broader range of game-theoretic frameworks in the future.

We summarize our main contributions as follows:

1) Observing the varying levels of driving complexities faced by different traffic participants, we propose treating agents differently within the *level-k* game framework of GameFormer. By adjusting the reasoning depth of agents based on their game states, we aim to reduce computational redundancy and enhance the accuracy of joint trajectory prediction and planning.
2) We introduce *Trajectory Entropy* as a bridge from MTP to the game states of agents. Drawing inspiration from signal processing theory, we extract signals reflecting uncertainty from MTP to serve as a clue for revealing the driving difficulty of agents. *Trajectory Entropy* can be used to assess the game states of agents, thereby enabling the improvement of the *level-k* game framework.
3) Through a simple Trajectory Entropy Gate mechanism, we effectively improve the *level-k* game framework of GameFormer. This approach has demonstrated competitive results and consistent improvements over GameFormer in trajectory prediction, open-loop and closed-loop planning on the WOMD [15] and nuPlan [16] benchmarks.

II. Related Work

A. Multimodal Trajectory Prediction

Trajectory prediction is a fundamental task in modern autonomous driving systems. Its objective is to anticipate the future paths of agents in the scene, providing information for planning models to ensure safe and intelligent driving. Initially, the goal of trajectory prediction was to forecast a single trajectory for the agent, referred to as discriminative trajectory prediction [13]. However, due to the inherent uncertainty of agent behavior, multiple feasible trajectories exist from start to destination. In such cases, discriminative trajectory prediction models struggle to differentiate between these feasible solutions. To address this uncertainty, multimodal trajectory prediction tasks have been introduced [17], requiring models to output multiple predicted trajectories to cover all feasible solutions. Building on this paradigm, CoverNet [18] frames prediction as a classification task over a pre-defined set of trajectory anchors. Adopting a hybrid strategy, Sun et al. [19] formulate the task by integrating clustering, classification, and synthesis. Furthermore, Deo et al. [20] introduce a policy-based approach conditioned on lane-graph traversals to yield scene-compliant multimodal predictions. Owing to the inherent interpretability, which has recently been leveraged for reasoning by large language models in [21], multimodal output has now become a default setting in trajectory prediction for autonomous driving tasks.

In order to enable models to produce multiple feasible trajectory predictions in complex environments, deep learning models are adopted in most current approaches. Initially, Long Short-Term Memory networks (LSTM) [22] and Convolutional Neural Networks (CNN) [23], [24] are applied to encode features from scenes and agents. Then, Graph Neural Networks [25], [26], [27] are used to further model the interaction between agents. With the innovation of the Transformer structure in many research fields, many works have applied it to multimodal trajectory prediction [28], [29]. However, these efforts are troubled by the inherent coupling issue of the trajectory prediction task, that is, the mutual influence between multiple intelligent agent trajectories. To address this, GameFormer [3] proposes a hierarchical trajectory refinement formulated by *level-k* game theory. It decouples interactions between agents within the same game level by predicting agent trajectories in each level based on other agents from the last game level. While GameFormer achieves leading performance, it involves redundancy in its *level-k* game refinement, as every agent is treated equally in all game levels. Due to varying environmental conditions, different agents encounter driving environments of differing complexities. This implies that the required game depth varies across agents. Thus, we combine the multimodal trajectory prediction task with the *level-k* game theory, providing an indicator of agent game-state stability. By adding a simple mask module, we effectively improve the performance of GameFormer while reducing its computational cost at the same time.

B. Joint Trajectory Prediction and Planning

The ultimate goal of trajectory prediction is to enable autonomous vehicles to make safe and efficient decisions in complex driving environments [30], [31]. The integration of trajectory prediction and planning poses a critical challenge in autonomous driving, as the trajectories of agents interact with each other [4]. Some approaches establish a sequential connection between planning and prediction [32], [33], focusing on a single vehicle and querying the trajectory prediction models of other vehicles. However, achieving reliable trajectories necessitates dense computations in this framework, significantly constraining computational speed [5].

Another promising paradigm for joint trajectory prediction and planning is to utilize a single model to implicitly consider interactions among multiple agents, providing end-to-end output that is globally consistent [4], [5], [28]. Nevertheless, this paradigm often lacks interpretability and struggles to yield reasonable results in complex scenarios. To enhance the interpretability of joint prediction and planning, GameFormer [3] introduces *level-k* game theory, explicitly modeling interactions among agents. Leveraging a level-k Transformer

Decoder, GameFormer notably enhances the accuracy of trajectory prediction and planning. Our work builds upon the *level-k* game framework of GameFormer and further refines it. By modeling the game states of agents, we eliminate redundancies in the *level-k* game, leading to enhanced precision and computational efficiency in joint trajectory prediction and planning.

C. Uncertainty-Based Early Exit Paradigm

Similar to the architecture of GameFormer, cascade attention layers in LLMs can also lead to redundant reasoning, which is referred to as "overthinking" [8]. To mitigate this issue, the Early Exit paradigm is proposed by PABEE [10] and DeeBert [9]. The core mechanism of Early Exit involves embedding "exit gates" at intermediate layers to assess the output uncertainty. If the uncertainty level is below a threshold, the model returns this output and terminates, thus saving computation and improving accuracy by mitigating "overthinking". This makes the reliable estimation of model uncertainty, often based on established frameworks [11], [12], a critical component for deciding when to exit. GameFormer also adopts the Bayesian framework [11] to achieve individual trajectory uncertainty. Thus, a straightforward approach is to perform early exit based on the trajectory uncertainty, thereby reducing the reasoning redundancy. However, the single-modal trajectory uncertainty output by GameFormer is not enough to reflect the game complexity in driving. Thus, we propose Trajectory Entropy, which integrates the learned trajectory uncertainty (as noise) and the multimodal trajectory dispersion (as signal) in a *Signal-to-Noise Ratio (SNR)* framework. This provides a link between the Bayesian outputs and the Early Exit decision gate in the *level-k* game.

III. Methodology

In this section, we first describe the preliminaries of our method, including the description of the Multimodal Trajectory Prediction (MTP) task in Sec. III-A1 and the details of our baseline, GameFormer, in Sec. III-A3. Then, we elaborate the proposed Trajectory Entropy in Sec. III-B. Finally, the improved *level-k* game framework for joint trajectory prediction and planning is presented in Sec. III-C.

A. Preliminaries

1) Multimodal Trajectory Prediction: Trajectory prediction is a pivotal module in autonomous driving systems. Initially, trajectory prediction was formulated to predict a single future trajectory $\mathbf{y}_i$ of an agent A_i from input X_i using a model $\mathcal{F}$, known as discriminative trajectory prediction.

$$\mathbf{y}_i = \mathcal{F}(X_i). \tag{1}$$

Here, the trajectory $\mathbf{y}_i$ represents a sequence of states (i.e., a 2D coordinate sequence) over the future horizon T. These coordinates are defined in a local reference frame centered at the agent A_i's last observed position (at $t = 0$).

However, due to the inherent uncertainty in agent trajectories, discriminative trajectory prediction fails to encompass all feasible solutions. This limitation hinders the network from learning a reasonable driving policy. To address this issue, Multimodal Trajectory Prediction (MTP) tasks have been introduced [17]. In this task, the model predicts a distribution covering all feasible solutions by outputting multiple trajectories with confidences for an agent, thus handling the uncertainty in trajectories.

$$\mathbf{Y}_i = \mathcal{F}(X_i). \tag{2}$$

Here, $\mathbf{Y}_i = \{(\mathbf{y}_j, c_j)\}_j^M$, where $\mathbf{y}_j$ is an individual trajectory, $c_j \in [0, 1]$ is the corresponding confidence, and M is the number of modalities. Due to enhanced interpretability and performance, this approach has become the default configuration for current trajectory prediction tasks [13].

2) Joint Trajectory Prediction and Planning: This methodology integrates the estimation of all agents' future trajectories with the generation of the ego agent's trajectory within a unified framework. Unlike serial steps in a *predict-then-plan* pipeline, the approach enables interaction-aware predictions that reflect the ego agent's influence on neighboring behaviors. Furthermore, it explicitly incorporates predictive uncertainty and multimodality into the planning objectives.

In practice, following the MTP stage, the final plan of the ego agent is determined by selecting the predicted trajectory candidate with the highest confidence score:

$$\mathbf{y}^*_{ego} = \mathbf{y}_{j^*}, \quad \text{where } j^* = *argmax_{j\in\{1,\dots,M\}} c_j,$$

and $\mathbf{y}^*_{ego}$ denotes the trajectory used for ego planning.

3) GameFormer: The joint trajectory prediction and planning contributes to the intelligent decision-making capability of the learning model. However, a primary challenge of this approach lies in the strong coupling of trajectories among agents. That is, agents mutually influence each other's decisions. To address this issue, GameFormer [3] explicitly models interactions among agents through the *level-k* game theory [34]. Specifically, it employs K Decoders $\{\mathcal{D}_{(k)}\}_0^{K-1}$ to iteratively generate trajectories for the ego vehicle and other agents. The process begins with the initial decoder $\mathcal{D}_{(0)}$, which takes as input the scene encoder's output, $\mathbf{S} \in \mathbb{R}^{N\times M\times d}$, and a learnable modality embedding, $\mathbf{I} \in \mathbb{R}^{N\times M\times d}$, where N is the number of agents. More details can be found in [3]. Each subsequent decoder, $\mathcal{D}_{(k)}$ for $k \geq 1$, refines the output by using the results from the preceding decoder, $\mathcal{D}_{(k-1)}$. Its inputs are the previously generated trajectories for all agents, $\{\mathbf{Y}_i^{k-1}\}_{i=1}^N$ (mapped to the $N \times M \times d$ dimension by an MLP), and the updated scene features, $\mathbf{S}^{k-1}$.

$$\begin{aligned} &\{\mathbf{Y}_i^k\}_{i=1}^N, \mathbf{S}^k = \mathcal{D}_{(k)}(I, \mathbf{S}), \quad k = 0 \\ &\{\mathbf{Y}_i^k\}_{i=1}^N, \mathbf{S}^k = \mathcal{D}_{(k)}(\{\mathbf{Y}_i^{k-1}\}_{i=1}^N, \mathbf{S}^{k-1}), \quad k \in \{1, \dots, K-1\} \end{aligned} \tag{3}$$

Through this approach, GameFormer decouples interactions among agents in a hierarchical game, enhancing interpretability and the overall accuracy.

However, treating all agents equally in the *level-k* games can introduce noise and computational redundancies. Due to varying environmental conditions, the driving difficulty faced by different agents varies. Some agents with lower driving difficulty can achieve stable results in the shallow-level games,

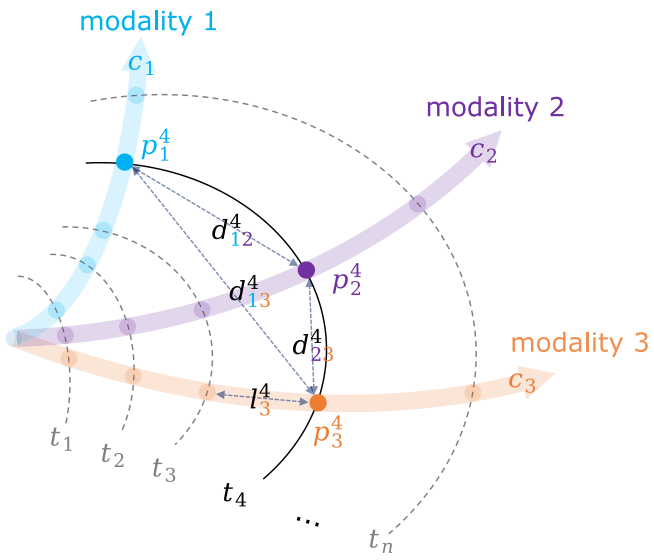


Fig. 3. **The formulation of Trajectory Entropy from Multimodal Trajectory Prediction (MTP).** Our Trajectory Entropy serves as statistical indicators of agent uncertainty based on MTP, utilizing the trajectory confidences ($\{c_i\}$), the distances between trajectory modalities ($\{d_{i,j}^t\}$) and the trajectory length within a unit time interval ($\{l_i^t\}$).

while others with greater difficulty require deeper reasoning. Therefore, it is reasonable to fix agents with lower driving difficulty in early game layers, as their trajectory prediction results are already stable. This not only reduces computational costs but also decreases overall uncertainty of the game, thereby improving accuracy. Hence, by analyzing the inherent uncertainty of multimodal trajectory prediction, this paper proposes a measure to capture the game state stability of agents (Sec. III-B). Then, the *level-k* game framework of GameFormer can be enhanced in terms of both accuracy and efficiency through a simple gate module (Sec. III-C).

B. Trajectory Entropy

To improve the performance of the *level-k* game framework [3] for AD, we construct an entropy-inspired stability indicator from Multimodal Trajectory Prediction (MTP) outputs (refer to Fig. 3). This indicator, then, can be used to capture the stability of agent states in the *level-k* game, enabling adaptive assignment of appropriate reasoning depths for different agents. This ultimately enhances the performance of trajectory prediction and planning. Next, we will provide a detailed explanation of the Trajectory Entropy.

The MTP results of an agent A can be expressed as: $\mathbf{Y} = \{(y_j, c_j)\}_j^M$, where M is the number of modalities. The $y_j = \{p_j^t\}_t^T$ represents the predicted trajectory of the j-th modality, comprising discrete 2D trajectory points $\{p_j^t = (u_j^t, v_j^t)\}_t^T$ over the future horizon T. The c_j denotes the confidence of the j-th modality trajectory, which is normalized across modalities using a `softmax` function to yield values in $[0, 1]$. For notational simplicity, we define a symbolic representation of Trajectory Entropy: $\mathcal{E}_\mathbf{Y}$.

Firstly, we simplify this temporal uncertainty measurement $\mathcal{E}_\mathbf{Y}$, by considering it at each time step t. Since the trajectories are composed of discrete trajectory points in time, we assume that the overall uncertainty $\mathcal{E}_\mathbf{Y}$ is the accumulation of the uncertainties of the trajectory point sets, $\mathcal{E}_{\mathbf{P}_t}$, where $\mathbf{P}_t = \{p_j^t, c_j\}_j^M$ denotes the trajectory point set at time step t (see Fig. 3).

$$\mathcal{E}_\mathbf{Y} = \sum_t^T \mathcal{E}_{\mathbf{P}_t}. \tag{4}$$

In other words, we adopt a temporal independent accumulation approximation, with the confidence at each time step being the same as the overall confidence. This assumption may not be flawless, due to the interrelated nature of trajectory points in a time series. However, the uncertainty in MTP is predominantly influenced by the interactions among multimodal trajectories. Moreover, our temporal aggregation ablation results (Sec. IV-D2) show that explicit temporal dependency modeling brings only marginal gains, from which we infer that the GameFormer decoder may have already implicitly encoded temporal dependencies during trajectory generation. Therefore, the temporal independent assumption is practical for the trajectory uncertainty estimation.

Then, we focus on the uncertainty of the trajectory point set $\mathcal{E}_{\mathbf{P}_t}$ at a specified time step t. Intuitively, the uncertainty of a multimodal trajectory set is related to *the distances between trajectories* and *the confidence levels of each trajectory*. We utilize them to form the core of Trajectory Entropy.

Specifically, we calculate the Euclidean distances of point pairs ($d_{ij}^t = \|p_i^t - p_j^t\|_2$) from M-modal trajectories: $\{d_{ij}^t\}_{i \neq j \in M}$. Without considering the confidences, the greater the distance between trajectory points, the more dispersed the points are, leading to higher uncertainty in trajectory prediction. Therefore, this trajectory dispersion is proportional to the uncertainty of the point set, serving as an important factor of $\mathcal{E}_{\mathbf{P}_t}$.

Taking a step further, we then incorporate the trajectory confidences $\{c_i\}_i$ into the calculation of $\mathcal{E}_{\mathbf{P}_t}$. The motivation is that pairwise spatial dispersion alone may overemphasize low-confidence trajectory modes, while the confidence scores provide a practical reliability cue for each predicted modality. Inspired by the ratio structure of SNR in random signal processing [35], we heuristically adopt a fractional form to combine these two factors: the numerator represents the pairwise spatial dispersion, analogous to signal power, and the denominator represents a confidence-based uncertainty proxy, analogous to noise power. This design makes a trajectory pair contribute strongly only when it is both spatially separated and assigned high confidence. For each pair of trajectory modalities, we empirically use the squared distance $(d_{ij}^t)^2$ to represent the magnitude of pairwise spatial dispersion at time step t. To weight this dispersion by prediction confidence, we introduce a confidence-based denominator:

$$\sigma_{ij}^2 = 1/(c_i c_j). \tag{5}$$

In this way, reliable trajectory pairs are assigned stronger influence, while uncertain trajectory pairs are down-weighted. Accordingly, we propose a pairwise uncertainty score S_{ij}^t, formulated as:

$$S_{ij}^t = \frac{(d_{ij}^t)^2}{\sigma_{ij}^2} = (d_{ij}^t)^2 c_i c_j. \tag{6}$$

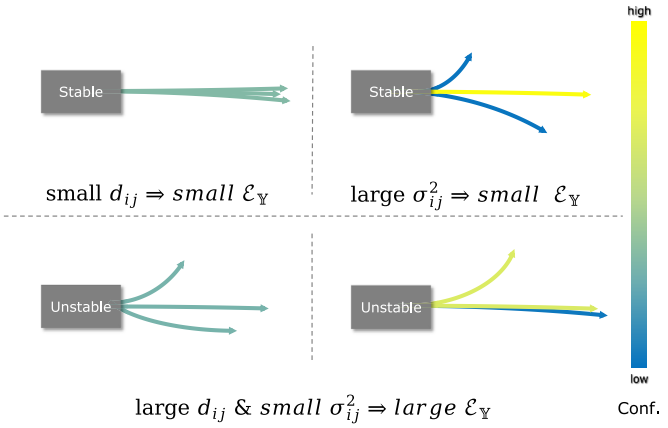


Fig. 4. **Some toy examples of Trajectory Entropy. Top:** A low Trajectory Entropy may arise from concentrated trajectories (left) or a single trajectory with high confidence (right) in MTP. Both of them exhibit narrow distribution of feasible trajectories. **Bottom:** Conversely, in cases where MTP comprises diverse trajectories with comparable confidence levels, the corresponding Trajectory Entropy is high.

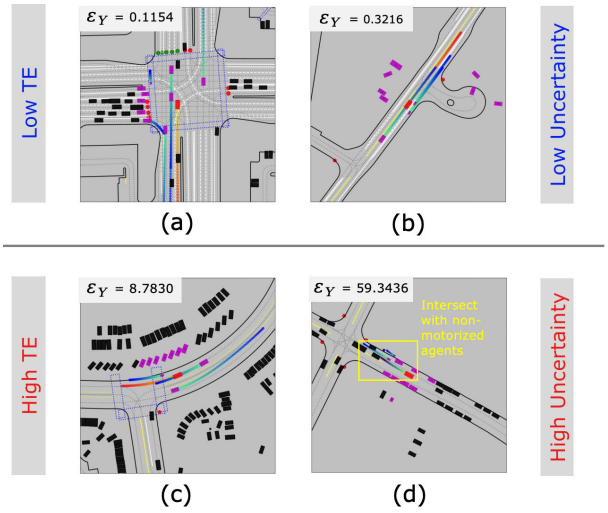


Fig. 5. **Real-scene cases illustrating TE ($\mathcal{E}_{\mathbf{Y}}$) as an indicator for agent state stability.** We collect some qualitative cases of ego agents with different TE values from the WOMD dataset. Examples with low TE (top) correspond to stable ego states in structured traffic, while high TE (bottom) arises in complex interactions, e.g., roundabouts (c) or interactions with non-motorized agents (d), indicating reduced state stability.

This formulation can be understood as a confidence-weighted multimodal trajectory dispersion. Thus, the overall uncertainty of the trajectory point set $\mathbf{P}_t$ at time step t can be represented as the sum of these scores:

$$\mathcal{E}_{\mathbf{P}_t} = \sum_{i \neq j \in M} S_{ij}^t. \tag{7}$$

Therefore, by accumulating it with a normalization factor over the prediction horizon $t \in \{1, \ldots, T\}$, we can obtain the overall MTP uncertainty, i.e., Trajectory Entropy ($\mathcal{E}_{\mathbf{Y}}$) as:

$$\mathcal{E}_{\mathbf{Y}} = \sum_{t=1}^{T} \frac{\mathcal{E}_{\mathbf{P}_t}}{E_M^t(\mathbf{l}) + \epsilon}. \tag{8}$$

Here, $E_M^t(\mathbf{l}) = \sum_j^M c_j l_j^t$ denotes the expectation of distance traveled by multimodal trajectories within a unit time duration, essentially representing the expectation of squared instantaneous speed, where $l_j^t = \|p_j^t - p_j^{t-1}\|_2^2$ (see Fig. 3). Meanwhile, ϵ is empirically set as 0.0001 to avoid numerical instability when the denominator becomes excessively small for low-speed or static agents. Variations in speed result in changes in the spatial scales of trajectory points. However, the alterations of spatial scale have been encoded in the distances. Thus, $E_M^t(\mathbf{l})$ serves as a normalization factor to remove the effect of speed variations across different time steps.

The resulting $\mathcal{E}_{\mathbf{Y}}$ has an entropy-like interpretation in the context of multimodal trajectory prediction. When the predicted modes are concentrated or dominated by high-confidence trajectories, the confidence-weighted dispersion is small, suggesting a relatively stable agent state. In contrast, when several high-confidence modes remain spatially separated, the score becomes larger, indicating higher multimodal dispersion and lower state stability. This behavior motivates the name *Trajectory Entropy*: larger values correspond to more dispersed multimodal predictions and lower agent-state stability within the *level-k* game, analogous to the notion of disorder captured by entropy. In this work, TE is therefore used as an entropy-inspired stability indicator computed from trajectory distances and modal confidences. We provide some toy examples in Fig. 4, showing that the *Trajectory Entropy* is in line with the human intuition. Moreover, we present qualitative visualizations of real-world scenes with varying ego-agent TE values in Fig. 5. These examples indicate a strong correlation between TE and driving complexity.

C. Improving Level-k Game Framework

Through Trajectory Entropy, we can utilize the uncertainty cues in MTP to assess the game states of agents, thereby enhancing the *level-k* game framework of GameFormer. Specifically, low Trajectory Entropy indicates that the model prediction for the feasible trajectory distribution of an agent is highly concentrated. In driving scenarios with structured rules, this often implies that the agent faces low driving difficulty. Therefore, within the *level-k* game, such agents could achieve stable game results in the current game level and do not require further deep reasoning. We can then fix their MTP results in order to avoid redundant computations.

This enhancement is implemented by a simple Trajectory Entropy Gate in Fig. 6, similar to the "exit gate" in [9], [10], positioned before all level decoders except for the level-0 decoder. The function of the Gate is to classify agents as active or inactive based on their Trajectory Entropy. Taking the k-th level as an example, for all active agents $\{A_i\}_i$, we only need to calculate the trajectory entropy of their MTP results before the k-th level decoder (i.e., the results of $k-1$-th level decoder): $\{\mathcal{E}_{\mathbf{Y}_i}^{k-1}\}_i$ and compare with a TE threshold $T_{\mathcal{E}}^{k-1}$. If $\mathcal{E}_{\mathbf{Y}_i}^{k-1} < T_{\mathcal{E}}^{k-1}$, the MTP result $\mathbf{Y}_i$ of the corresponding agent is directly propagated to all the subsequent game levels, and the agent is labeled as inactive. Thus, the computational redundancies can be reduced and the overall accuracy can be improved as well, as shown in our experiments. It is worth noting that the Trajectory Entropy threshold can vary across different game levels.

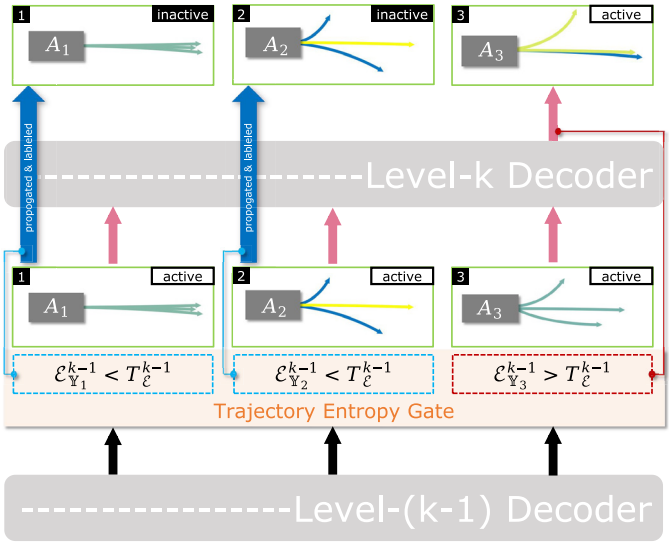


Fig. 6. **The hierarchical game framework improved by Trajectory Entropy.** Before the level-(k-1) decoder, a Trajectory Entropy Gate is set up with a specified threshold ($T_{\mathcal{E}}$), where the MTP results of active agents ($\mathbf{Y}_i$) are utilized to calculate the Trajectory Entropy ($\mathcal{E}_{\mathbf{Y}_i}$). If the $\mathcal{E}_{\mathbf{Y}_i} < T_{\mathcal{E}}$, the agent A_i is labeled as inactive. The MTP results of all the inactive agents are directly propagated to the subsequent game levels, while those of active agents are processed by the *level-k* decoder.

The intuition behind this setting is that the high game levels tend to exhibit lower TE values in the MTP results (see Fig. 1), due to their deeper reasoning. Hence, the threshold $\{T^k_{\mathcal{E}}\}_{k=0}^{K-2}$ set for the higher game levels is adjusted to a lower value, reflecting a more stringent criterion for labeling agents as inactive.

D. Adaptive Threshold Selection

Although TE can robustly capture the game state of each agent, the proposed TE gate still requires a threshold to determine when a specific agent should be frozen. In the design of TE, normalization factors are introduced to reduce the influence of trajectory scale and velocity magnitude, thereby improving generalization across different settings. However, the range of TE values may still be affected by the intrinsic driving complexity of the scene, which poses a challenge to the generalizability of the TE gate. Furthermore, since TE is accumulated over time, the total number of prediction timesteps also directly impacts its magnitude. Therefore, in this section, we present a heuristic adaptive threshold selection strategy that uses scene-level priors as a reproducible default rule.

Empirically, the magnitude of TE is mainly affected by three observable factors: (1) the number of sampled timesteps, denoted by N_{ts}; (2) the structure of GameFormer, particularly the number of decoder layers (i.e., the number of game levels), denoted by K; and (3) the difficulty of the driving scene R_H, which is calculated as the proportion of complex scenarios (i.e., intersections and roundabouts) in the target benchmark or map. We use these factors as heuristic priors for threshold initialization. They are selected because they summarize the major sources of TE scale variation across prediction horizons, model configurations, and scenario distributions. The rationale for this empirical design is detailed as follows.

First, since TE is accumulated over time, its value increases with the number of timesteps. Therefore, the TE threshold should be positively correlated with N_{ts}. **Second**, prior studies on the overthinking phenomenon in LLMs [9], [10] suggest that deeper architectures are more susceptible to redundant refinement. This implies that when GameFormer contains more decoder layers, stricter control is needed to prevent redundant refinement in deeper game levels. Accordingly, the TE threshold should be inversely related to the number of decoder layers K. **Third**, more challenging scenes generally involve greater uncertainty in agent interactions, which correspond to lower prediction confidence and larger spatial dispersion among multimodal trajectories. As a result, TE values tend to be larger in such scenes, and the threshold should therefore increase with scene difficulty. To quantify scene difficulty in benchmark evaluation, we treat roundabouts and intersections as representative hard scenes. Using prior map information, we estimate the proportion of such hard scenes R_H in the dataset. Then, the TE threshold is defined to scale positively with this proportion. Based on these empirical considerations, we propose the default adaptive threshold as follows.

$$T^i_{\mathcal{E}} = \begin{cases} \alpha \times (R_H)^3 \ \dfrac{(N_{\text{ts}})^2}{K-1}, & i = 0, \\ (1 - d \times i) \ T^0_{\mathcal{E}}, & i = \{1, \ldots, K-2\}. \end{cases} \tag{9}$$

Algorithm 1 Heuristic Adaptive Threshold Selection

Input: Prediction horizon N_{ts}, Model depth K, Scene difficulty prior R_H

Output: Array of TE thresholds $\mathbf{T}_{\mathcal{E}} = [T^0_{\mathcal{E}}, T^1_{\mathcal{E}}, \ldots, T^{K-2}_{\mathcal{E}}]$

 # Step 1: Compute the heuristic base threshold for the initial layer ($i = 0$)
$T^0_{\mathcal{E}} \leftarrow 0.1 \times (R_H)^3 \times \frac{(N_{\text{ts}})^2}{K-1}$
$\mathbf{T}_{\mathcal{E}}[0] \leftarrow T^0_{\mathcal{E}}$
 # Step 2: Compute decayed thresholds for deeper layers ($i = \{1, \ldots, K-2\}$)
for $i = 1$ **to** $K - 2$ **do**
    # Apply stricter control (lower threshold) for deeper layers
    decay_factor $\leftarrow 1 - 0.1 \times i$
    $\mathbf{T}_{\mathcal{E}}[i] \leftarrow$ decay_factor $\times T^0_{\mathcal{E}}$
end for
Return $\mathbf{T}_{\mathcal{E}}$

Here, the constant factor α, the cubic dependence on R_H, the quadratic dependence on N_{ts}, and the layer-wise decay step d are empirical choices, where the default values are $\alpha = 0.10$ and $d = 0.10$. We provide a sensitivity analysis of these factors in Sec. IV-D5. These choices encode the expected influence direction of each factor and perform well in our experiments. However, they should not be interpreted as a guarantee of optimality across all datasets or deployment scenes. For a model with K decoder layers, we define $K - 1$ thresholds $\{T^i_{\mathcal{E}}\}_{i=0}^{K-2}$. We first compute the initial threshold $T^0_{\mathcal{E}}$, and then empirically decrease the remaining thresholds with a linear step size d. In practice, one can first estimate the proportion of hard scenes in the target deployment region, and then

compute the TE thresholds according to the prediction horizon and model architecture, enabling the proposed method to be applied across different regions. When sufficient validation data are available, this adaptive threshold can be further refined through threshold sensitivity analysis. Therefore, the proposed adaptive threshold should be viewed as a practical heuristic initialization for TE gating, and its applicable boundary should be verified on the target scenario distribution before deployment. The detailed threshold selection pseudo-code is provided in Algorithm 1.

IV. Experiments

In this section, we evaluate the performance of the proposed *Trajectory Entropy* in improving the *level-k* game framework on the joint trajectory prediction and planning task. Firstly, we present the implementation details of our method in Sec. IV-A. Our experiments involve two benchmarks, i.e., the Waymo Open Motion Dataset (WOMD) [15] and nuPlan [16] datasets, both of which are used to conduct trajectory prediction (refer to Sec. IV-B) and ego planning (refer to Sec. IV-C) experiments. Further ablation studies can be found in Sec. IV-D.

A. Experimental Details

1) Model Details: Our method utilizes GameFormer as the backbone, trained on two datasets separately. Next, we present the implementation details of the model on each dataset, including its training and inference settings.

a) For Waymo Open Motion Dataset (WOMD): Following [3], we train and evaluate the original GameFormer and our improved version on the Waymo Open Motion Dataset, in terms of interaction prediction, open-loop and closed-loop planning. Specifically, for the interaction prediction task, we train two models (original GameFormer and ours) on the entire Waymo Open Motion Dataset following [3]. The joint prediction setting is applied for this task, where only 6 joint trajectories of the two agents are predicted [28]. For the planning tasks, we use the scenarios provided by GameFormer [3], including 10K 20-second randomly selected scenarios. 9K scenarios are used for training and the remaining 1K for validation, following GameFormer [3]. The models are trained on 4 NVIDIA A100 GPUs, utilizing the official implementation. Then, the joint prediction and planning performance is evaluated in 400 8-second interactive and dynamic scenarios. The GameFormer model in this dataset is set with 5 decoder layers (level-0/1/2/3/4) for planning and 4 decoder layers (level-0/1/2/3) for prediction, according to [3]. The Trajectory Entropy thresholds of our improved version are set differently in specific experiments.

b) For nuPlan: nuPlan stands as the premier large-scale benchmark for autonomous driving planning, consisting of massive driving scenes and real-world data. To train the GameFormer model, we evenly sample 60K scenes from the whole nuPlan training dataset. Then, 50K scenes are allocated for model training, while the remaining 10K scenes are reserved for trajectory prediction evaluation. The planning evaluation, on the other hand, is performed on the selected test set of nuPlan, following [39], which is a standard benchmark. The GameFormer model is implemented with 3 decoder layers (level-0/1/2). The training and inference are performed on one GeForce RTX 4090 GPU, following the hyper-parameter setting of the official implementation. During inference, the Trajectory Entropy thresholds are set according to specific driving difficulties. In practice, the thresholds are set differently in different scenes, as the scene difficulty varies in the nuPlan dataset. The specific thresholds are described in the following experimental sections. To achieve the single planning trajectory from MTP results, we use a trajectory selection approach proposed by [40], involving a score function for trajectory selection.

TABLE I

Comparison With Recent Methods on the WOMD Interaction Prediction Benchmark. The Best Results Are **Highlighted**

Model	minADE (↓)	minFDE (↓)	Miss rate (↓)	mAP (↑)
LSTM baseline [15]	1.9056	5.0278	0.7750	0.0524
Heat [27]	1.4197	3.2595	0.7224	0.0844
AIR^2 [36]	1.3165	2.7138	0.6230	0.0963
SceneTrans [28]	0.9774	2.1892	0.4942	0.1192
DenseTNT [37]	1.1417	2.4904	0.5350	0.1647
M2I [38]	1.3506	2.8325	0.5538	0.1239
MTR [29]	0.9181	2.0633	**0.4411**	**0.2037**
GameFormer [3]	0.9177	1.9791	0.4747	0.1211
Ours	**0.9092**	**1.9673**	0.4643	0.1226

2) Evaluation Metrics: In the following, we describe the evaluation metrics for prediction and planning.

a) Trajectory prediction metrics: For both WOMD and nuPlan benchmarks, we use the standard evaluation metrics [3], including minimum average displacement error (minADE), minimum final displacement error (minFDE), miss rate, and mean Average Precision (mAP).

b) Planning metrics: In open-loop planning of WOMD datasets, following [3], we use distance-based error metrics to verify the planning performance, including collision rate, miss rate, prediction and planning error (i.e. ADE and FDE). In closed-loop planning of WOMD datasets, we also follow [3], using metrics including success rate, progress along the route, comfort-related metrics (i.e. longitudinal acceleration and jerk, lateral acceleration), and position errors. In nuPlan benchmark, we conduct planning experiments under various settings, comprising open-loop and closed-loop with reactive/nonreactive agents. A comprehensive planning score is applied to performance evaluation in these settings, which is provided by the nuPlan platform.[1]

It is worth noting that, unless otherwise specified, most experimental results in Sec. IV-B and Sec. IV-C are reported from a single training and evaluation run with random seed 3407, due to computational resource limitations. To improve reproducibility, we specify the model depth, batch size, training epochs, and learning rate in the corresponding experimental subsections.

B. Trajectory Prediction Results

1) Waymo Interaction Trajectory Prediction: Firstly, we conduct the interaction prediction experiment on the WOMD,

[1]https://github.com/motional/nuplan-devkit

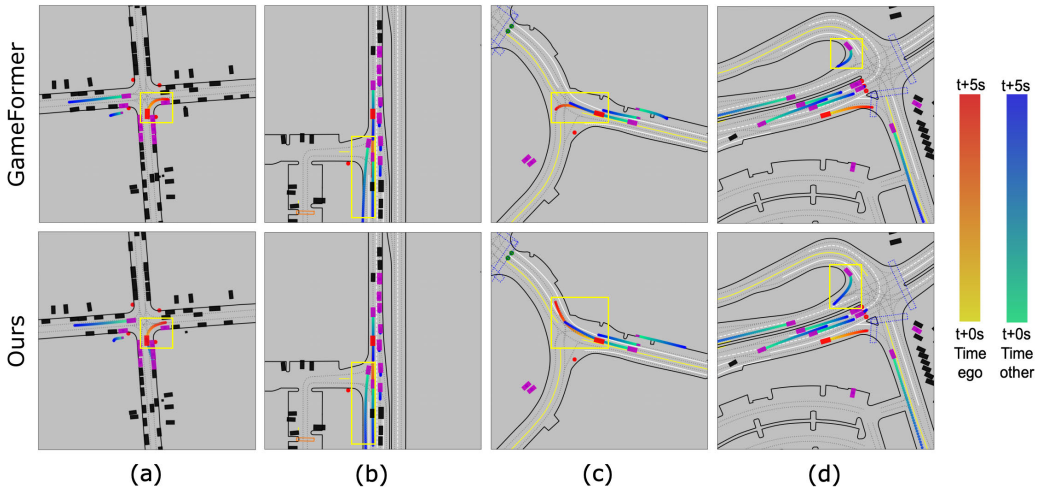


Fig. 7. **The qualitative comparison results between our method and GameFormer in the open-loop planning task on the WOMD dataset.** It can be seen that our method provides more reasonable trajectories of both ego and other agents. Specifically, our method enables collision avoidance in (a), ensures trajectory predictions remain within drivable areas in (b) and (d), and facilitates correct direction planning in (c). The red trajectory is the ego agent plan and the blue ones are the predictions of neighboring agents.

TABLE II

TRAJECTORY PREDICTION RESULTS ON THE NUPLAN BENCHMARK. THE RELATIVE CHANGE OF OUR METHOD WITH RESPECT TO THE ORIGINAL GAMEFORMER IS SHOWCASED AS SUBSCRIPTS. FOR EACH METRIC VALUE x, THE RELATIVE CHANGE IS COMPUTED AS $\Delta = (x_{\text{Ours}} - x_{\text{GameFormer}})/x_{\text{GameFormer}}$

Method	minADE (↓)	minFDE (↓)	Miss Rate (↓)	mAP (↑)
GameFormer [3]	0.4728	0.8510	0.0186	0.2064
Ours (Man. Thd.)	$0.3789_{-19.86\%}$	$0.5777_{-32.12\%}$	$0.0111_{-40.32\%}$	$0.2114_{+2.42\%}$
Ours (Ada. Thd.)	$0.3725_{-21.21\%}$	$0.5712_{-32.88\%}$	$0.0103_{-44.62\%}$	$0.2159_{+4.60\%}$

using the official setting [3], [15]. Our model predicts the joint future positions for the two interacting agents over an 8-second time horizon. Our model is trained with a batch size of 16 for 30 epochs using a learning rate of 10^{-4}. The comparison methods include representative recent approaches [3], [15], [27], [28], [29], [36], [38]. For the experiment, our Trajectory Entropy thresholds are set as $[T_{\mathcal{E}}^0 = 4.3,\ T_{\mathcal{E}}^1 = 4.2,\ T_{\mathcal{E}}^2 = 4.1]$. We report the prediction results in Tab. I. Compared with the original GameFormer, our method obtains lower minADE and minFDE, together with slight improvements in miss rate and mAP. It also achieves the lowest minADE and minFDE among the listed methods, while MTR remains stronger on miss rate and mAP. Given that several gains on this prediction benchmark are relatively small, we view these results as supportive evidence that TE improves GameFormer without compromising prediction accuracy.

2) nuPlan Trajectory Prediction: Although the nuPlan benchmark is mainly used for planning evaluation, we also conduct trajectory prediction experiments on it. We utilize models to predict the positions of neighboring agents 8 seconds into the future. The model is trained with a batch size of 32 for 100 epochs using a learning rate of 10^{-4}. In this experiment, the Trajectory Entropy thresholds are set as $[T_{\mathcal{E}}^0 = 40.0,\ T_{\mathcal{E}}^1 = 36.0]$ through our adaptive threshold selection (Ada. Thd.). Our method results are compared with the original GameFormer. The results are summarized in Tab. II. As can be seen, Trajectory Entropy Gate improves the accuracy of GameFormer in terms of all metrics. The prediction error is reduced by a large margin, e.g., up to 44.62%. This suggests that properly assigning different reasoning depths to different agents is beneficial for the overall prediction precision. By adaptively regulating the inference complexity across agents, our Trajectory Entropy Gate improves the *level-k* game process, leading to accurate trajectory prediction. We also report our results with manual threshold settings (Man. Thd.) $[T_{\mathcal{E}}^0 = 30.0,\ T_{\mathcal{E}}^1 = 20.0]$. Although this variant also outperforms GameFormer, it performs slightly worse than the adaptive threshold setting. This supports the effectiveness of our adaptive threshold selection strategy.

C. Planning Results

1) Waymo Open-Loop Planning: We conduct the open-loop planning experiment on the WOMD. Following [3], selected WOMD scenarios with a prediction/planning horizon of 5 seconds are employed in our experiments. We perform $M = 6$ MTP for 10 neighboring agents closest to the ego vehicle. The model is trained with a batch size of 32 for 20 epochs using a learning rate of 10^{-4}. A recent competitive planning method based on imitation learning is set as the baseline [4], along with the original GameFormer [3]. Our Trajectory Entropy thresholds are calculated as $[T_{\mathcal{E}}^0 = 0.5,\ T_{\mathcal{E}}^1 = 0.45,\ T_{\mathcal{E}}^2 = 0.4,\ T_{\mathcal{E}}^3 = 0.35]$ by our adaptive strategy (Ada. Thd.) in this experiment. The results are reported in Tab. III. It can be seen that our method achieves better results than the original GameFormer. Especially, a substantial decrease in collision rate is observed. Overall, the lower planning errors also demonstrate the effectiveness of our method compared with baseline methods [3], [4]. In addition, we report the results obtained using manually specified thresholds, $[T_{\mathcal{E}}^0 = 0.8,\ T_{\mathcal{E}}^1 = 0.75,\ T_{\mathcal{E}}^2 = 0.7,\ T_{\mathcal{E}}^3 = 0.65]$ (Man. Thd.). Although these empirically chosen thresholds yield better performance, our adaptive threshold selection strategy generalizes more effectively across different datasets. We also

TABLE III

EVALUATION OF OPEN-LOOP PLANNING PERFORMANCE IN SELECTED WOMD SCENARIOS. THE BEST RESULTS ARE **HIGHLIGHTED**

Method	Collision rate ↓ (%)	Miss rate ↓ (%)	Planning error ↓ (m) @1s	@3s	@5s
DIPP [4]	4.45	**7.71**	0.276	1.236	3.282
GameFormer [3]	4.39	12.11	**0.083**	0.877	2.685
Ours (Man. Thd.)	**2.65**	9.75	0.087	**0.832**	**2.513**
Ours (Ada. Thd.)	3.54	11.39	0.084	0.874	2.669

TABLE IV

EVALUATION OF CLOSED-LOOP PLANNING PERFORMANCE IN SELECTED WOMD SCENARIOS. THE **BEST** RESULTS ARE HIGHLIGHTED

Method	Success rate ↑ (%)	Progress ↑ (m)	Acceleration ↓ (m/s^2)	Jerk ↓ (m/s^3)	Lateral acc. ↓ (m/s^2)	Position error ↓ (m) @3s	@5s	@8s
DIPP [4]	85.43	45.04	0.76	3.42	1.48	2.81	6.09	12.65
GameFormer [3]	82.25	51.25	**0.54**	3.80	2.44	2.20	4.74	9.25
Ours	**87.25**	**53.31**	0.65	**1.42**	**0.26**	**2.18**	**4.64**	**8.47**

TABLE V

COMPARISON WITH RECENT METHODS ON THE NUPLAN PLANNING BENCHMARK. THE **BEST** AND SECOND RESULTS IN **HYBRID SERIES** ARE HIGHLIGHTED

Planners		Test14-random			Test14-hard		
Type	Method	OLS	NR-CLS	R-CLS	OLS	NR-CLS	R-CLS
Expert	Log-replay	100.0	94.03	75.86	100.0	85.96	68.80
Rule-based	IDM [41]	34.15	70.39	72.42	20.07	56.16	62.26
	PDM-Closed [40]	46.32	90.05	91.64	26.43	65.07	75.18
Learning-based	RasterModel [16]	62.93	69.66	67.54	52.4	49.47	52.16
	UrbanDriver [42]	82.44	63.27	61.02	76.90	51.54	49.07
	GC-PGP [43]	77.33	55.99	51.39	73.78	43.22	39.63
	PDM-Open [40]	84.14	52.80	57.23	79.06	33.51	35.83
	PlanTF [39]	87.07	86.48	80.59	83.32	72.68	61.70
	PLUTO [44]	N/A	89.90	78.62	N/A	70.03	59.74
Hybrid	PDM-Hybrid [40]	**82.21**	90.20	91.56	73.81	65.95	75.79
	Diff. Plan [45]	N/A	**94.80**	<u>91.75</u>	N/A	<u>78.87</u>	**82.00**
	PLUTO [44] w/ refine	N/A	<u>92.23</u>	90.29	N/A	**80.08**	76.88
	GameFormer [3]	79.35	80.80	79.31	<u>75.27</u>	66.59	68.83
	Ours	<u>80.70</u>	91.27	**92.38**	**76.31**	72.08	<u>77.88</u>

provide some visualization comparison between our method and GameFormer in Fig. 7. In the figure, the trajectories from our method are more reasonable than those from GameFormer, indicating that our Trajectory Entropy Gate can improve the performance of the *level-k* game framework.

2) Waymo Closed-Loop Planning: The closed-loop planning performance is also evaluated on the WOMD dataset. We use the simulation environment provided by [4], where the state of the ego agent is updated by the planning model and other agents follow logged trajectories in the dataset. The test dataset selected from the WOMD dataset is provided by DIPP [4], also recommended by GameFormer [3]. Our model is trained with a batch size of 32 for 20 epochs using a learning rate of 2×10^{-4}. Our method is compared with these recent methods [3], [4]. The results are reported in Tab. IV. It can be seen that our method surpasses other approaches in terms of most metrics, achieving the best success rate and the smallest position error. This indicates that our Trajectory Entropy Gate effectively improves the planning performance, providing safer and more comfortable driving trajectories.

3) nuPlan Planning Benchmark: The nuPlan benchmark is employed to assess the planning performance as well. We use the standard test scenes provided by [39], including **Test14-random** and **Test14-hard**. The former is sampled randomly from 14 scenario types and the latter consists of hard scenarios selected by PDM [40]. The Trajectory Entropy thresholds are set as $[T_{\mathcal{E}}^0 = 30,\ T_{\mathcal{E}}^1 = 28]$ as default. We use the same model in the nuPlan Trajectory Prediction experiments. In OLC and NR-CLS of Test14-hard, we slightly decrease $T_{\mathcal{E}}^1$ to 25 to handle more challenging scenes. We compare our methods with rule-based [40], [41], learning-based [16], [39], [40], [42], [43], [44] and hybrid planning methods [3], [40], [44]. Note that the hybrid methods involve a trajectory refinement utilizing rule-based strategies. Our method belongs to this category, the same as the original GameFormer. The results are summarized in Tab. V. It can be seen that our improved GameFormer outperforms the original version by a large margin on all scenes, especially in reactive closed-loop planning scenes, e.g., up to 16.48% (79.31 → 92.38). This indicates that the *level-k* game framework can produce

TABLE VI

ABLATION STUDY ON THE SNR-STYLE FORMULATION OF TRAJECTORY ENTROPY ON THE NUPLAN PREDICTION BENCHMARK. THE MAIN VALUE IS THE MEAN OVER THREE RANDOM SEEDS, AND THE FADED VALUE AFTER ± IS THE VARIANCE WRITTEN WITH A TWO-DECIMAL MANTISSA IN SCIENTIFIC NOTATION

Method	minADE↓	minFDE↓	MR↓	mAP↑
GameFormer	0.4600	0.7651	0.0186	0.2064
Full TE (Ours)	**0.3739**$_{\pm 9.87\times10^{-7}}$	**0.5751**$_{\pm 9.21\times10^{-6}}$	**0.0103**$_{\pm 0.00}$	**0.2171**$_{\pm 1.15\times10^{-6}}$
Distance-only TE	0.4017$_{\pm 2.29\times10^{-6}}$	0.6164$_{\pm 2.63\times10^{-5}}$	0.0127$_{\pm 3.30\times10^{-9}}$	0.2039$_{\pm 1.65\times10^{-6}}$
Confidence-only TE	0.4144$_{\pm 2.96\times10^{-6}}$	0.6420$_{\pm 4.81\times10^{-6}}$	0.0140$_{\pm 6.43\times10^{-7}}$	0.2023$_{\pm 7.00\times10^{-8}}$

TABLE VII

ABLATION STUDY ON DIFFERENT TEMPORAL AGGREGATION STRATEGIES FOR TRAJECTORY ENTROPY (TE). TIA (OUR DEFAULT) ACHIEVES A FAVORABLE BALANCE BETWEEN SIMPLICITY AND PERFORMANCE

Method	minADE↓	minFDE↓	MR↓	mAP↑
GameFormer (Baseline)	0.4728	0.8510	0.0186	0.2064
TIA (Ours)	**0.3725**	0.5712	**0.0103**	**0.2159**
Weighted TIA	0.3871	0.5956	0.0119	0.2144
TCA	**0.3725**	**0.5709**	0.0104	0.2143

more reasonable results, when agents with different driving difficulties are distinguished by our Trajectory Entropy, and assigned with different reasoning depths to mitigate the "overthinking" issue [9], [10]. Meanwhile, our method achieves comparable or better results than the recent planning methods [40], [44], [45], implying the potential of the game theory-based framework in autonomous driving planning. It is also worth noting that our method is able to perform trajectory prediction and planning at the same time, which demonstrates its advantage over planning-only methods, such as PLUTO [44] and Diff. Plan [45].

D. Model Analysis and Ablation Study

1) Ablation Study on the SNR-Style Formulation of TE: To examine whether the SNR-style TE formulation benefits from jointly using spatial trajectory dispersion and modal confidence, we compare the full TE with two simplified variants on the nuPlan prediction benchmark. The distance-only variant removes confidence weighting by setting $\sigma_{ij}^2 = 1$, so TE is computed only from pairwise spatial dispersion, $\mathcal{E}_{\mathbf{Y}} = \sum_{t=1}^{T}\left(\sum_{i\neq j} d_{ij}^2 / \left(\sum_{j}^{M} l_j^t + \epsilon\right)\right)$. The confidence-only variant removes spatial trajectory dispersion, so TE retains only the confidence-coupling term, $\mathcal{E}_{\mathbf{Y}} = \sum_{t=1}^{T}\left(\sum_{i\neq j} c_i c_j\right)$. As shown in Tab. VI, both simplified variants still improve over the original GameFormer on minADE, minFDE, and miss rate, indicating that either spatial dispersion or modal confidence alone can provide useful stability cues for TE gating. However, the full TE achieves the best overall performance across all four metrics. For example, compared with the distance-only and confidence-only variants, the full TE further reduces minADE from 0.4017/0.4144 to 0.3739. These results indicate that the complete gain comes from combining spatial trajectory dispersion with modal confidence through the SNR-style formulation.

TABLE VIII

ABLATION STUDY ABOUT THE NORMALIZATION FACTOR. THE TRAJECTORY PREDICTION EXPERIMENTS ARE CONDUCTED ON NUPLAN BENCHMARK

Method	$T_{\mathcal{E}}^0$	$T_{\mathcal{E}}^1$	minADE (↓)	minFDE (↓)	Miss Rate (↓)	mAP (↑)
GameFormer	N/A	N/A	0.4728	0.8510	0.0186	0.2064
$E_M^t(\mathbf{Y}^t)$	0.03	0.025	0.4285	0.8252	0.0902	**0.2246**
$E_M^t(\mathbf{1})$	40.0	36.0	**0.3725**	**0.5712**	**0.0103**	0.2159
$E_M^t(\mathbf{Y})$	0.06	0.05	0.6602	1.2369	0.0947	0.2037

2) Evaluation of Temporal Independence Assumption: We construct TE by independently adding the uncertainty of each time step, as shown in Eq. (8). However, the trajectory sequences exhibit temporal correlation. Therefore, in this section, we compare our default Temporal Independent Accumulation (**TIA**) in TE with two time-aware variants to see if explicit temporal modeling improves stability estimation. First, we add a time-increasing weight $\log(t+e)$ to each $\mathcal{E}_{\mathbf{P}_t}$:

$$\mathcal{E}_{\mathbf{Y}} = \sum_{t=1}^{T} \frac{\log(t+e)\mathcal{E}_{\mathbf{P}_t}}{E_M^t(\mathbf{1})}, \tag{10}$$

which is named as **Weighted TIA**. It emphasizes the uncertainty in the later stage of the trajectory. Secondly, we employ a 1D convolution (kernel size is 5 with zero padding) over the original uncertainty sequence $\{\mathcal{E}_{\mathbf{P}_t}\}_{t=1}^{T}$ to get a new sequence $\{\mathcal{E}_{\mathbf{P}_t}^C\}_{t=1}^{T}$, capturing local temporal dependencies before summation:

$$\mathcal{E}_{\mathbf{Y}} = \sum_{t=1}^{T} \frac{\mathcal{E}_{\mathbf{P}_t}^C}{E_M^t(\mathbf{1})}, \tag{11}$$

which is named as Temporal Convolution Accumulation (**TCA**). We conduct experiments on the nuPlan benchmark for trajectory prediction. The results are presented in Tab. VII. From the results, the time-aware TCA variant yields minimal improvements over our simple TIA (e.g., minFDE 0.5709 vs 0.5712). This finding suggests that summing the per-step uncertainty terms, $\mathcal{E}_{\mathbf{P}_t}$, is sufficient for stability estimation. The explicit temporal dependency modeling has limited impact on TE-based stability estimation, possibly because the GameFormer decoder has already implicitly encoded temporal dependencies during trajectory generation. Moreover, the performance degradation observed with Weighted TIA indicates that our simple unweighted summation provides a more robust estimation of stability.

3) Trajectory Normalization: There are various speed limits in different driving scenes. However, the speed of agents has less direct relation to driving difficulty. Thus, we utilize trajectory normalization to remove the effect of agent speed on judging driving difficulty, as shown in Eq. (8). There are different ways to remove the speed effect. A straightforward approach is to use the expectation of trajectory length at each time step $E_M^t(\mathbf{Y}^t) = \sum_{j}^{M} c_j |p_j^t|$ as the normalization factor, which reflects the speed differences between trajectories. However, the value of $E_M^t(\mathbf{Y}^t)$ increases with the time step, which leads to unbalanced weighting on uncertainty from the different time

TABLE IX
SENSITIVITY ANALYSIS OF DIFFERENT CONFIDENCE-TO-NOISE MAPPING FUNCTIONS ON THE NUPLAN PREDICTION BENCHMARK

Mapping Function of σ_{ij}^2	minADE ↓	minFDE ↓	MR ↓	mAP ↑
GameFormer (Baseline)	0.4728	0.8510	0.0186	0.2064
$1/(c_i + c_j)$ (Linear)	0.3765	0.5739	0.0107	0.2116
$1/c_i + 1/c_j$ (Separable)	0.3747	0.5761	**0.0103**	0.2142
$1/(c_i \cdot c_j)$ **(Ours)**	**0.3725**	**0.5712**	**0.0103**	**0.2159**

steps. Alternatively, we can use the trajectory length within a unit time duration $E_M^t(\mathbf{l})$, representing the instantaneous speed square of the agent, as the normalization factor. From another perspective, as the speed ultimately changes the final lengths of trajectories, we can directly utilize this length expectation $E_M^t(\mathbf{Y}) = \sum_j^M c_j |p_j^T|$ as the normalization factor. To investigate the efficacy of different normalization factors, we conduct experiments on nuPlan in terms of the MTP task. We experimentally set proper $\{T_{\mathcal{E}}^i\}_i$ for these normalization factors, whose performance is the best among multiple experiment rounds. The results are reported in Tab. VIII. We also provide the results of original GameFormer for comparison. It can be seen that an appropriate normalization factor is critical for model performance, as large performance gaps exist in the table. The squared instantaneous speed-related factor $E_M^t(\mathbf{l})$ obtains the best performance, which can provide balanced weights for uncertainty from all time steps. Meanwhile, TE with $E_M^t(\mathbf{Y}^t)$ can also improve the GameFormer performance, showcasing the robustness of our TE against normalization schemes.

4) Computational Cost: Our Trajectory Entropy is able to avoid unnecessary computation for agents under simple driving environments in the *level-k* game. Thus, the computational cost of original GameFormer can be reduced through our Trajectory Entropy Gate. We compare the inference time and GFLOPs of our method with the original GameFormer on both the nuPlan trajectory prediction setting and the Waymo open-loop planning setting. All runtime measurements are conducted on a workstation with an Intel Xeon Silver 4314 CPU and one NVIDIA GeForce RTX 4090 GPU. The batch size is 1, and each measured input contains one scene with 11 agents. We report the mean and standard deviation over 3 repeated inference runs. The reported time measures a single model inference pass and excludes data preprocessing and any postprocessing. No separate warm-up run is included in this measurement protocol. To provide a clear understanding of the efficiency of our proposed components, we decompose the computational costs into the three primary stages of our framework: Trajectory Entropy (TE) calculation, the Gating mechanism, and the Decoder. Also, a "Ratio (%)" column is added to the analysis. This metric represents the proportion of the total computational budget accounted for by each subcomponent.

As shown in Tab. XII, our method reduces the inference time by 14.24% on nuPlan trajectory prediction (54.07 ms vs. 46.37 ms) and by 18.54% on Waymo open-loop planning (121.90 ms vs. 99.30 ms). The GFLOPs are also reduced by 1.46% and 4.29% on the two settings, respectively. The TE calculation and gating mechanism are lightweight compared with the gated GameFormer backbone. On nuPlan, they account for only 2.02% of the total runtime and less than 0.06% of the GFLOPs. On Waymo, they account for 6.74% of the total runtime and less than 0.07% of the GFLOPs. These results indicate that the TE gate consistently improves inference efficiency across different model configurations.

5) Sensitivity Analysis of Adaptive Threshold Selection: The adaptive threshold in Eq. (9) is designed as a heuristic initialization rule. To empirically examine its sensitivity, we conduct additional experiments on the nuPlan prediction benchmark with $R_H = 0.5$ and $N_{\text{ts}} = 80$. Following Eq. (9), the sensitivity study can be written as $T_{\mathcal{E}}^0 = \alpha \times (R_H)^{p_R}(N_{\text{ts}})^{p_N}/(K-1)$ and $T_{\mathcal{E}}^i = (1 - d \times i)T_{\mathcal{E}}^0$. We sweep the color-coded components in this rule, including the constant factor α, the exponent p_R of R_H, the exponent p_N of N_{ts}, and the layer-wise decay step d. For the base-threshold formula, the default setting is $\alpha = 0.10$, $p_R = 3$, $p_N = 2$, and $d = 0.10$. We also compare with two simpler **percentile** strategies. Specifically, we randomly sample 960 validation scenes in nuPlan prediction benchmark, run GameFormer inference to obtain MTP outputs, compute the corresponding TE values without using ground-truth trajectory labels, and take the 10% and 20% percentile TE values as two initial thresholds $T_{\mathcal{E}}^0$. For each threshold setting, we repeat the experiment with three random seeds (3407, 22, and 1245234) and report the mean and variance in Tab. X.

The results show that the proposed TE gate consistently outperforms the original GameFormer under a broad range of threshold settings, demonstrating robustness. Moderate changes to the α and d lead to small performance variations, indicating that the method is not overly sensitive to these two choices in the tested range. In contrast, the exponent of N_{ts} has a clearer impact: using N_{ts}^1 produces thresholds that are too small for the 80-step prediction horizon, while using N_{ts}^3 produces excessively large thresholds, and both settings degrade the prediction metrics. The exponent of R_H controls how strongly the threshold reacts to scene difficulty; in this nuPlan setting, R_H^2 gives the best minADE/minFDE, whereas R_H^4 is too conservative and leads to worse results. The percentile-based strategies are simple alternatives, but they underperform the formula-based thresholds on minADE, minFDE, and miss rate, suggesting that directly transferring a percentile threshold from a small validation subset is less stable. Nevertheless, using a larger validation set may improve performance, albeit at a higher cost. Overall, these findings support the proposed adaptive rule as a reproducible default initialization, while also confirming that it should be treated as an empirical heuristic whose parameters may be further tuned on the target validation distribution before deployment.

E. Understanding the Confidence-to-Noise Mapping

Our TE formulation uses the confidence scores predicted by GameFormer to construct the noise power. Accordingly, it assumes that these confidence estimates are meaningfully related to trajectory accuracy. To examine this assumption, we

TABLE X

SENSITIVITY ANALYSIS OF ADAPTIVE THRESHOLD SELECTION ON THE NUPLAN PREDICTION BENCHMARK. FOR EACH REPORTED METRIC, THE MAIN VALUE IS THE MEAN OVER THREE RANDOM SEEDS (3407, 22, AND 1245234), AND THE FADED VALUE AFTER ± IS THE VARIANCE. THE PERCENTILE-BASED THRESHOLDS ARE ESTIMATED FROM 960 UNLABELED VALIDATION SAMPLES

Method	α	p_R	p_N	d	Threshold	minADE↓	minFDE↓	MR↓	mAP↑
GameFormer	–	–	–	–	–	0.4600	0.7651	0.0186	0.2064
Ours (Default)	0.10	3	2	0.10	[40.00, 36.00]	**0.3739**$\pm 9.87\times10^{-7}$	0.5751$\pm 9.21\times10^{-6}$	**0.0103**± 0.00	0.2171$\pm 1.15\times10^{-6}$
Ours	0.09	3	2	0.10	[36.00, 32.40]	0.3765$\pm 2.77\times10^{-6}$	0.5773$\pm 2.44\times10^{-6}$	0.0107$\pm 1.43\times10^{-7}$	0.2150$\pm 1.72\times10^{-6}$
Ours	0.11	3	2	0.10	[44.00, 39.60]	0.3713$\pm 7.43\times10^{-7}$	0.5718$\pm 1.92\times10^{-5}$	0.0104$\pm 8.33\times10^{-8}$	0.2197$\pm 5.08\times10^{-6}$
Ours	0.10	2	2	0.10	[80.00, 72.00]	**0.3655**$\pm 3.40\times10^{-6}$	**0.5657**$\pm 2.72\times10^{-6}$	0.0105$\pm 2.13\times10^{-7}$	0.2290$\pm 3.05\times10^{-6}$
Ours	0.10	4	2	0.10	[20.00, 18.00]	0.3858$\pm 1.84\times10^{-6}$	0.5955$\pm 2.33\times10^{-8}$	0.0113$\pm 9.00\times10^{-8}$	0.2125$\pm 4.99\times10^{-6}$
Ours	0.10	3	1	0.10	[0.50, 0.45]	0.3868$\pm 6.79\times10^{-6}$	0.6092$\pm 3.77\times10^{-5}$	0.0130$\pm 3.43\times10^{-7}$	0.1998$\pm 1.29\times10^{-5}$
Ours	0.10	3	3	0.10	[3200.00, 2880.00]	0.3866$\pm 7.63\times10^{-7}$	0.6009$\pm 7.09\times10^{-6}$	0.0122$\pm 3.00\times10^{-8}$	0.2180$\pm 5.25\times10^{-6}$
Ours	0.10	3	2	0.05	[40.00, 38.00]	0.3736$\pm 1.81\times10^{-6}$	0.5741$\pm 1.58\times10^{-6}$	**0.0103**$\pm 5.33\times10^{-8}$	0.2175$\pm 1.17\times10^{-5}$
Ours	0.10	3	2	0.20	[40.00, 32.00]	0.3734$\pm 2.29\times10^{-6}$	0.5735$\pm 3.97\times10^{-6}$	0.0104$\pm 2.43\times10^{-7}$	0.2210$\pm 5.16\times10^{-6}$
Percentile (10%)	–	–	–	–	[2.70, 2.50]	0.3913$\pm 1.20\times10^{-6}$	0.6073$\pm 1.32\times10^{-6}$	0.0144$\pm 2.80\times10^{-7}$	0.2032$\pm 5.20\times10^{-6}$
Percentile (20%)	–	–	–	–	[21.20, 17.20]	0.3860$\pm 3.43\times10^{-6}$	0.5950$\pm 5.61\times10^{-5}$	0.0114$\pm 4.33\times10^{-8}$	**0.2690**$\pm 1.03\times10^{-2}$

TABLE XI

IMPACT OF TE GATE FREEZING ON PERFORMANCE. THE EXPERIMENTS ARE CONDUCTED ON WAYMO OPEN-LOOP PLANNING BENCHMARK. WE COMPARE THE BASELINE AGAINST OUR MODEL, FURTHER DECOMPOSING OUR RESULTS INTO SUCCESS CASES (CORRECTLY FROZEN) AND FAILED CASES (INCORRECTLY FROZEN)

Method	Coll. Rate ↓	Miss Rate ↓	Plan Err@1s ↓	Plan Err@3s ↓	Plan Err@5s ↓	Pred. ADE ↓	Pred. FDE ↓
GameFormer (Baseline)	4.39%	12.11%	0.0829	0.8771	2.6850	0.8713	1.9972
Ours (Overall)	**3.54%**	**11.39%**	**0.0837**	**0.8738**	**2.6685**	**0.8501**	**1.9566**
↪ Success (94.88%)	2.56%	8.67%	0.0833	0.8071	2.4527	0.8101	1.8906
↪ Failed (5.12%)	28.08%	76.03%	0.1066	1.7749	5.8292	1.5905	3.1792

TABLE XII

COMPUTATIONAL COST COMPARISON BETWEEN OUR METHOD AND THE ORIGINAL GAMEFORMER. THE TIME AND GFLOPS REDUCTIONS IN THE SUBSCRIPTS ARE COMPUTED WITH GAMEFORMER AS THE BASELINE. FOR EACH MEASURED RUNTIME, THE VALUE BEFORE ± IS THE MEAN OVER THREE REPEATED RUNS AND THE FADED VALUE AFTER ± IS THE STANDARD DEVIATION

Benchmark	Component	Time (ms)	Ratio (%)	GFLOPs	Ratio (%)
nuPlan Traj. Pred.	GameFormer	54.07$\pm 1.7\times10^{-4}$	–	43.24	–
	Ours	**46.37**$\pm 0.077_{-14.24\%}$	**100.0**	**42.61**$_{-1.46\%}$	**100.0**
	↪ TE Calculation	0.80$\pm 3.02\times10^{-5}$	1.73	0.021	0.049
	↪ TE Gating Mechanism	0.14$\pm 1.67\times10^{-4}$	0.29	1×10^{-6}	<0.01
	↪ Gated GameFormer	45.43± 0.037	97.99	42.59	99.95
Waymo Open-loop Plan.	GameFormer	121.90± 0.77	–	41.99	–
	Ours	**99.30**$\pm 0.12_{-18.54\%}$	**100.0**	**40.19**$_{-4.29\%}$	**100.0**
	↪ TE Calculation	1.10$\pm 2.02\times10^{-8}$	1.11	0.022	0.055
	↪ TE Gating Mechanism	5.59$\pm 1.54\times10^{-4}$	5.63	1×10^{-6}	<0.01
	↪ Gated GameFormer	92.61$\pm 2.02\times10^{-2}$	93.26	40.17	99.94

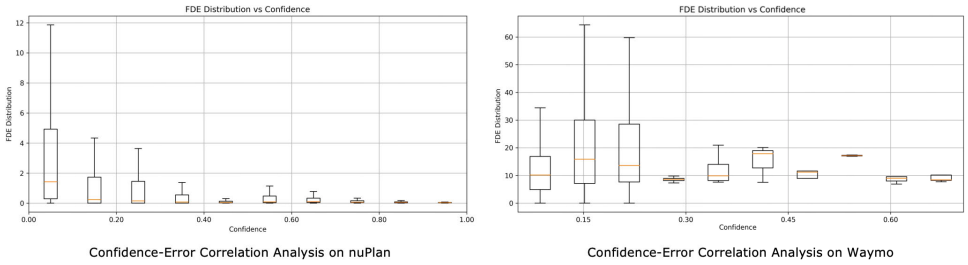


Fig. 8. **Confidence-Error Correlation Analysis of Trajectory Prediction.** We statistically analyze the correlation between the predicted trajectory confidence and trajectory accuracy on two benchmarks, by analyzing the distribution of the FDE metric of trajectories in different confidence intervals.

perform a large-scale Confidence-Error Correlation analysis on both the nuPlan (842, 688 trajectories) and Waymo (162, 162 trajectories) benchmarks by examining the distribution of Final Displacement Error (FDE) of trajectories across different trajectory confidence intervals. The results, shown in Fig. 8, indicate that the predicted confidence is empirically correlated with trajectory accuracy and therefore provides an basis for our TE.

Given this empirical correlation, we use an inverse mapping from confidence to form TE. In this work, we adopt the multiplicative form $\sigma^2 = 1/(c_i \cdot c_j)$ (Eq. (5)). To evaluate the effectiveness of this design choice, we conduct a sensitivity

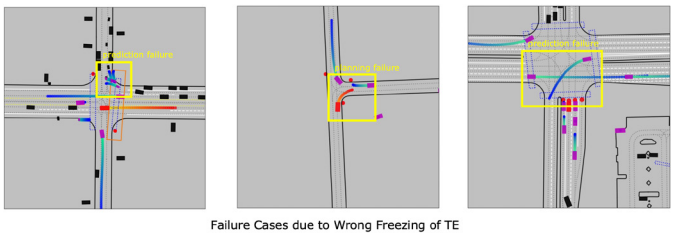


Fig. 9. **Failure cases from incorrect freezing of TE gate.** These failures mainly occur in challenging intersection scenarios, where prematurely frozen agents later become highly interactive, leading to prediction errors and unsafe plans such as collisions.

analysis using alternative mapping functions on the nuPlan prediction benchmark, including the additive form $1/(c_i + c_j)$ and the separable form $1/c_i + 1/c_j$. The results are reported in Tab. IX. Although our proposed mapping achieves the best performance, all three variants outperform the baseline GameFormer and deliver comparable gains. These findings suggest that the TE gating mechanism is robust to the specific functional form of the confidence-to-noise mapping, thereby supporting TE as a practical indicator of agent game stability.

F. Risk Analysis of Wrong Freezing

The proposed TE gate introduces an irreversible early-freezing/fixing mechanism: once an agent is identified as inactive at an intermediate level, its state is directly passed to subsequent levels without further interaction updates. While this design improves efficiency and reduces unnecessary computation in most cases, it also introduces a potential risk. If an agent is frozen early but later becomes highly interactive, the model can no longer incorporate its updated interactions, which may lead to prediction errors and eventually affect downstream planning.

To quantify this risk, we perform a comparison with the baseline GameFormer on the Waymo open-loop planning benchmark (a total of 2,800 scenes). Since the only difference between our model and GameFormer is the TE Gate mechanism, the resulting performance gap can be directly attributed to the effect of freezing. We define a *failure case* as a scene in which the TE gate freezes at least one agent prematurely and, relative to the baseline, yields worse final prediction or planning performance. Conversely, if freezing is triggered but scene-level performance is preserved or improved, the scene is categorized as a *success case*. Based on this protocol, we compute the proportions of success and failure cases, and further report their planning and prediction performance separately. Specifically, we evaluate planning using Collision Rate, Miss Rate, and Planning Error at 1s/3s/5s, and prediction using ADE and FDE. The results are summarized in Table XI.

From the table, incorrect freezing occurs in only 5.12% of the evaluated scenes, while 94.88% are categorized as success cases. However, once incorrect freezing occurs, its impact can be substantial. In the failure subset, the Collision Rate increases to 28.08%, the Miss Rate rises to 76.03%, and the Planning Error@5s reaches 5.8292, indicating that early freezing may indeed cause cascading errors in difficult interactive scenarios. The overall model still achieves better performance than the baseline on all major planning and prediction metrics, suggesting that the TE gate yields a positive effect in the vast majority of scenes. We further visualize representative failure cases in Fig. 9. These failure cases are concentrated in challenging intersection scenarios with strong and evolving multi-agent interactions. In such scenes, an agent that appears inactive at an early stage may later become critical to the ego vehicle's decision-making. Once such an agent is frozen incorrectly, the model cannot update its interaction-aware feature in subsequent levels, which may lead to inaccurate predictions and unsafe plans, such as collisions. These observations suggest that incorrect freezing is mainly concentrated in hard interactive scenes and remains relatively rare overall. Nevertheless, the irreversible freezing remains a potential risk in safety-critical driving scenarios, which is a primary limitation of our method.

Therefore, the current method is mainly suitable for offline prediction and planning evaluation, research-oriented improvement of *level-k* game reasoning, or applications with an external safety monitor. For practical deployment, this risk should be mitigated by additional safety mechanisms. For instance, a conservative gating mode could require consistently low TE values across multiple game levels before freezing an agent, thereby reducing premature freezing in rapidly changing scenes. Risk-aware thresholds could further reduce the freezing threshold, or disable freezing, in scenes involving vulnerable road users. In addition, a reactivation mechanism could unfreeze an agent when its TE value, interaction intensity, or collision risk increases after freezing. Finally, a lightweight safety checker could be applied before the final planning output to reject plans with collision, off-road, or comfort violations and fall back to the full GameFormer reasoning path. We leave the systematic design and validation of these risk-control mechanisms to future work.

V. Conclusion

This paper introduces Trajectory Entropy to assess the game states of agents in the recent *level-k* game framework, for improved precision and efficiency of joint trajectory prediction and planning. Specifically, we aim to reduce computational redundancy in the *level-k* game framework by assigning appropriate reasoning depth to agents with different driving complexities. To this end, we investigate the practical connection among MTP uncertainty, agent driving complexity, and agent game state in the *level-k* game framework. Based on this analysis, Trajectory Entropy is formulated as an entropy-inspired stability indicator that integrates confidence-weighted multimodal trajectory dispersion into an SNR-style formulation. It serves as a practical indicator for assessing agent-state stability and driving complexity within the *level-k* game. Through a simple yet effective Trajectory Entropy Gate mechanism, we improve the *level-k* game framework, reducing the computation overhead and enhancing the overall accuracy at the same time. Comprehensive experiments are conducted on Waymo and nuPlan benchmarks, in terms of trajectory prediction and open/closed-loop planning, which are key components of intelligent autonomous driving. The results demonstrate the efficacy of our method, showcasing

competitive performance on both benchmarks and enhanced computational efficiency.

One limitation is that the current TE Gate freezes agents irreversibly, so rare wrong-freezing cases may still cause safety-critical planning errors. Before deployment, TE-based early freezing should therefore be combined with risk-control mechanisms, such as conservative gating, risk-aware thresholds, agent reactivation, or safety checking. Future work will validate the above risk-control mechanisms and extend TE beyond GameFormer, where it may support early-exit rules in broader game-theoretic frameworks or stability scoring in transformer-based planners.

REFERENCES

[1] Z. Huang, H. Liu, J. Wu, and C. Lv, "Conditional predictive behavior planning with inverse reinforcement learning for human-like autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 7, pp. 7244–7258, Jul. 2023.

[2] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 683–700.

[3] Z. Huang, H. Liu, and C. Lv, "GameFormer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 3880–3890.

[4] Z. Huang, H. Liu, J. Wu, and C. Lv, "Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 11, pp. 15222–15236, Nov. 2024.

[5] Z. Huang, P. Karkus, B. Ivanovic, Y. Chen, M. Pavone, and C. Lv, "DTPP: Differentiable joint conditional prediction and cost evaluation for tree policy planning in autonomous driving," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2024, pp. 6806–6812.

[6] Y. Hu et al., "Planning-oriented autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 17853–17862.

[7] M. A. Costa-Gomes, V. P. Crawford, and N. Iriberri, "Comparing models of strategic thinking in van huyck, battalio, and Beil's coordination games," *J. Eur. Econ. Assoc.*, vol. 7, nos. 2–3, pp. 365–376, Apr. 2009.

[8] Y. Kaya, S. Hong, and T. Dumitras, "Shallow-deep networks: Understanding and mitigating network overthinking," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, K. Chaudhuri and R. Salakhutdinov, Eds. Long Beach, CA, USA: ML Research Press, Jun. 2019, pp. 3301–3310.

[9] J. Xin, R. Tang, J. Lee, Y. Yu, and J. Lin, "DeeBERT: Dynamic early exiting for accelerating BERT inference," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 2246–2251.

[10] W. Zhou, C. Xu, T. Ge, J. McAuley, K. Xu, and F. Wei, "Bert loses patience: Fast and robust inference with early exit," in *Proc. Adv. Neural Inf. Process. Syst*., vol. 33. Red Hook, NY, USA: Curran Associates, 2020, pp. 18330–18341.

[11] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[12] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–12.

[13] X. Wang, H. Yue, and Q. Yang, "Multimodal data trajectory prediction: A review," in *Proc. IEEE 10th Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/IEEE 9th Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Jul. 2023, pp. 1–5.

[14] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. New York, NY, USA: McGraw-Hill, 2002.

[15] P. Sun et al., "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2443–2451.

[16] H. Caesar et al., "NuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles," 2021, *arXiv:2106.11810*.

[17] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2255–2264.

[18] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "CoverNet: Multimodal behavior prediction using trajectory sets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14062–14071.

[19] J. Sun, Y. Li, H.-S. Fang, and C. Lu, "Three steps to multimodal trajectory prediction: Modality clustering, classification and synthesis," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 13230–13239.

[20] N. Deo, E. Wolff, and O. Beijbom, "Multimodal trajectory prediction conditioned on lane-graph traversals," in *Proc. 5th Conf. Robot Learn.*, Jan. 2022, pp. 203–212.

[21] I. Bae, J. Lee, and H.-G. Jeon, "Social reasoning-aware trajectory prediction via multimodal language model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 48, no. 6, pp. 6035–6052, Jun. 2026.

[22] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 961–971.

[23] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "HOME: Heatmap output for future motion estimation," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, Sep. 2021, pp. 500–507.

[24] H. Cui et al., "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 2090–2096.

[25] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "GOHOME: Graph-oriented heatmap output for future motion estimation," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 9107–9114.

[26] Z. Huang, X. Mo, and C. Lv, "Multi-modal motion prediction with transformer-based neural network for autonomous driving," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 2605–2611.

[27] X. Mo, Z. Huang, Y. Xing, and C. Lv, "Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9554–9567, Jul. 2022.

[28] J. Ngiam et al., "Scene transformer: A unified architecture for predicting multiple agent trajectories," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–25.

[29] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Motion transformer with global intention localization and local movement refinement," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 6531–6543.

[30] A. Cui, S. Casas, A. Sadat, R. Liao, and R. Urtasun, "LookOut: Diverse multi-future prediction and planning for self-driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 16107–16116.

[31] J. L. V. Espinoza, A. Liniger, W. Schwarting, D. Rus, and L. Van Gool, "Deep interactive motion prediction and planning: Playing games with motion prediction models," in *Proc. Learn. Dyn. Control Conf.*, 2022, pp. 1006–1019.

[32] H. Song, W. Ding, Y. Chen, S. Shen, M. Y. Wang, and Q. Chen, "PiP: Planning-informed trajectory prediction for autonomous driving," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 598–614.

[33] S. Casas, A. Sadat, and R. Urtasun, "MP3: A unified model to map, perceive, predict and plan," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14398–14407.

[34] J. Wright and K. Leyton-Brown, "Beyond equilibrium: Predicting human behavior in normal-form games," in *Proc. AAAI Conf. Artif. Intell.*, 2010, vol. 24, no. 1, pp. 901–907.

[35] B. Porat, *Digital Processing of Random Signals: Theory and Methods*. New York, NY, USA: Dover, 2008.

[36] D. Wu and Y. Wu, "AIR^2 for interaction prediction," 2021, *arXiv:2111.08184*.

[37] J. Gu, C. Sun, and H. Zhao, "DenseTNT: End-to-end trajectory prediction from dense goal sets," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Jun. 2021, pp. 15303–15312.

[38] Q. Sun, X. Huang, J. Gu, B. C. Williams, and H. Zhao, "M2i: From factored marginal trajectory prediction to interactive prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 6543–6552.

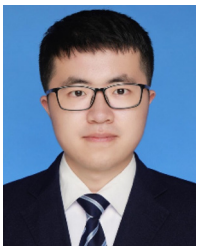
[39] J. Cheng, Y. Chen, X. Mei, B. Yang, B. Li, and M. Liu, "Rethinking imitation-based planners for autonomous driving," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2024, pp. 14123–14130.

[40] D. Dauner, M. Hallgarten, A. Geiger, and K. Chitta, "Parting with misconceptions about learning-based vehicle motion planning," in *Proc. 7th Conf. Robot Learn.*, vol. 229, J. Tan, M. Toussaint, and K. Darvish, Eds. Long Beach, CA, USA: ML Research Press, Nov. 2023, pp. 1268–1281.

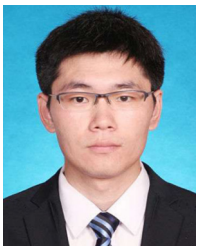
[41] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000.
[42] O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska, "Urban driver: Learning to drive from real-world demonstrations using policy gradients," in *Proc. Conf. Robot Learn.*, 2022, pp. 718–728.
[43] M. Hallgarten, M. Stoll, and A. Zell, "From prediction to planning with goal conditioned lane graph traversals," in *Proc. IEEE 26th Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2023, pp. 951–958.
[44] J. Cheng, Y. Chen, and Q. Chen, "PLUTO: Pushing the limit of imitation learning-based planning for autonomous driving," 2024, *arXiv:2404.14327*.
[45] Y. Zheng et al., "Diffusion-based planning for autonomous driving with flexible guidance," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2025, pp. 1–21.

Yesheng Zhang received the B.S. and M.S. degrees in biomedical engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2020 and 2023, respectively, where he is currently pursuing the Ph.D. degree with the School of Automation and Intelligent Sensing. His research interests include feature matching, 3D reconstruction, and autonomous driving.

Wenjian Sun received the B.S. degree from the School of Automation and Intelligent Sensing, Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2025, where he is currently pursuing the M.S. degree with the School of Automation and Intelligent Sensing. His research interests include autonomous driving and human pose estimation.



Yuheng Chen received the B.S. degree in automation and the M.S. degree in control science and engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2022 and 2025, respectively. His research interests include trajectory prediction and planning for autonomous driving.



Qingwei Liu received the Ph.D. degree in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2022. He is currently with the Automated-Driving Department, SAIC Motor, Shanghai. His research interests include integrated vehicle dynamics control and automated vehicle motion planning.

Qi Lin received the B.S. and M.S. degrees in vehicle engineering from Tongji University, Shanghai, China, in 2010 and 2013, respectively. He is currently working with the AD Department, SAIC Motor. His research interests include end-to-end planning in autonomous driving field.

Rui Zhang received the B.S. degree in mechanical engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2010, and the M.S. degree in electrical engineering from Waseda University, Tokyo, Japan, in 2013. He is currently working with the SAIC R&D Center as the ADAS Development Head.

Xu Zhao (Member, IEEE) received the Ph.D. degree in pattern recognition and intelligent system from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2011. He is currently a Full Professor with the School of Automation and Intelligent Sensing, SJTU. He was a Visiting Scholar with the Beckman Institute, University of Illinois Urbana–Champaign, Urbana, IL, USA, from November 2007 to December 2008; and a Post-Doctoral Research Fellow with Northeastern University, Boston, MA, USA, from August 2012 to August 2013. His research interests include visual analysis of human motion, machine learning, and image/video processing.