D²Drive: Training-Free Dynamic Inference for End-to-End Autonomous Driving

Junxuan Liu[1], Bin Hu[1(✉)], Jinlai Zhang[2(✉)], Qi Xiong[2], Lin Hu[2]

[1] School of Computer Science and Technology, Changsha University of Science and Technology, Changsha, 410114, Hunan, China
[2] College of Mechanical and Vehicle Engineering, Changsha University of Science and Technology, Changsha, 410114, Hunan, China

Abstract. End-to-end autonomous driving frameworks have achieved strong performance in perception, prediction, and planning. However, their deep interaction architectures often introduce substantial computational overhead and inference latency, which hinders real-time deployment. To improve inference efficiency without retraining, we propose D²Drive, a training-free dynamic inference framework for end-to-end autonomous driving. D²Drive reduces redundant computation by exploiting feature stability during inference. It contains two complementary components. First, a dynamic exit mechanism terminates deeper computation when feature differences between adjacent layers become sufficiently small. Second, an FFN partial computation strategy selectively performs FFN operations on tokens with larger variations across layers, while stable tokens bypass redundant FFN computation. These two designs reduce computation at both the layer and token-computation levels. Unlike pruning, quantization, and retraining-based acceleration methods, D²Drive does not modify model parameters, task heads, or training objectives. Therefore, it can be directly applied to pretrained autonomous driving frameworks during inference. We evaluate D²Drive on four representative end-to-end autonomous driving frameworks, including UniAD, VAD, SparseDrive, and MomAD. The evaluation covers GFLOPs for the whole model, computation inside selected modules, planning accuracy, collision rate, tracking quality, and detection performance, providing a systematic protocol for analyzing the balance between efficiency and accuracy in training-free dynamic inference for autonomous driving systems.

Keywords: dynamic inference, training-free acceleration, early exit, partial computation, end-to-end autonomous driving

1 Introduction

Recent end-to-end autonomous driving frameworks integrate perception, prediction, and planning in unified pipelines[5]. Methods such as UniAD[16], VAD[19], SparseDrive[28], and MomAD[27] perform well in complex driving scenarios, but their deep interaction modules also bring considerable computation and latency. Reducing

redundant inference while preserving downstream driving performance is therefore essential for real-time deployment.

Existing acceleration approaches include pruning[12], quantization[17], knowledge distillation[13], token pruning[25,22], and token merging[1]. Many of them are effective, but retraining, calibration, or architecture-specific adaptation limits their use with pretrained autonomous driving frameworks. Redundancy across deep interaction layers has also received less attention.

We propose D²Drive, a training-free dynamic inference framework that combines adjacent-layer feature-difference based dynamic exit with token-variation based FFN partial computation. The exit mechanism skips unnecessary deeper interaction layers, while partial FFN computation processes only informative high-variation tokens. Since D²Drive acts only during inference, pretrained parameters, task heads, and training objectives remain unchanged.

We evaluate D²Drive on UniAD[16], VAD[19], SparseDrive[28], and MomAD[27] using whole-model GFLOPs, module-level computation, planning accuracy, collision rate, tracking quality, and detection performance, so that efficiency gains can be considered together with task preservation.

The main contributions of this paper are summarized as follows:

- We propose D²Drive, a training-free dynamic inference framework for end-to-end autonomous driving. It improves inference efficiency without retraining, parameter pruning, or modifying task heads.
- We introduce a dynamic exit mechanism that reduces redundant deep interaction computation based on feature stability across adjacent layers.
- We propose an FFN partial computation strategy that selectively preserves informative tokens and skips redundant FFN computation for stable tokens.
- We establish an evaluation protocol across multiple frameworks, covering computation for the whole model, computation inside selected modules, and planning, tracking, and detection metrics to analyze the balance between efficiency and accuracy in training-free dynamic inference. enumerate

2 Related Work

End-to-end autonomous driving frameworks rely on deep interaction architectures for multi-view perception, temporal modeling, and planning-oriented decision making. These architectures improve performance but raise inference cost. We review related work on efficient inference acceleration, dynamic inference and early exit, and end-to-end autonomous driving.

2.1 Inference Acceleration

Existing acceleration approaches can generally be divided into model compression methods and dynamic computation methods. Model compression techniques, including pruning[12], quantization[17], and knowledge distillation[13], reduce computational

cost by removing redundant parameters, lowering numerical precision, or transferring knowledge to smaller models. Although effective in many scenarios, these methods usually require additional retraining, calibration, or architecture-specific adaptation.

Another line of research reduces computation dynamically during inference, especially for Transformer-based visual models[30,10]. DETR[3] and Deformable DETR[34] show the flexibility of query-based interaction, but repeated attention and FFN operations remain costly. Token pruning and reorganization methods reduce this cost by removing, merging, or reorganizing less informative tokens. DynamicViT[25], EViT[22], ATS[11], and V2Drop[4] demonstrate the effectiveness of token-level dynamic computation.

These methods show that visual models contain substantial token-level redundancy. Most, however, reduce tokens within each layer and pay less attention to redundancy across adjacent deep interaction layers. D²Drive considers both sources of redundancy during inference without modifying pretrained model parameters.

2.2 Dynamic Inference and Early Exit

Dynamic inference adapts computational cost according to input characteristics or intermediate representations. Early-exit methods terminate inference once representations become sufficiently reliable. BranchyNet[29], Shallow-Deep Networks[20], DeeBERT[31], FastBERT[23], PABEE[33], and DynaBERT[14] reduce unnecessary computation through side branches, confidence estimates, prediction stability, or adaptive depth. DeeR-VLA[32] further extends dynamic inference to multimodal embodied AI.

Most early-exit methods rely on confidence estimation, auxiliary classifiers, or learned routing policies. D²Drive instead uses feature differences between adjacent layers as the exit criterion, avoiding extra supervision and remaining compatible with pretrained autonomous driving frameworks.

2.3 End-to-End Autonomous Driving Frameworks

End-to-end autonomous driving has become an important research direction[5]. ST-P3[15], TransFuser[9], InterFuser[26], TCP[6], BEVFormer[21], and BEVFusion[24] improve perception, prediction, planning, or sensor fusion in unified pipelines. UniAD[16], VAD[19], VADv2[7], PPAD[8], SparseDrive[28], and MomAD[27] further advance planning-oriented or sparse end-to-end driving.

Although these frameworks achieve strong planning and perception performance, they still depend on deep hierarchical interaction modules [35-38] with substantial inference cost. Existing studies mainly improve efficiency through sparse representation design, BEV aggregation, or architecture-level optimization. D²Drive instead targets unused computation redundancy inside pretrained models and exploits it across both layers and tokens during inference.

3 Method

3.1 Overview

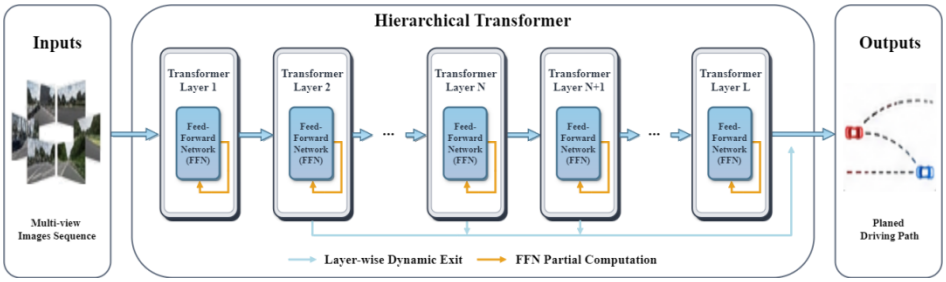

Fig. 1. Overview of the proposed D²Drive framework.

End-to-end autonomous driving frameworks usually use deep hierarchical interaction modules for perception, prediction, and planning. These modules improve representation capability but introduce considerable inference cost. Adjacent layers often become similar in later stages, and only a subset of tokens changes significantly across layers, indicating redundancy both across layers and within FFN modules.

D²Drive therefore contains two complementary components: dynamic exit, which reduces redundant layer execution according to adjacent-layer feature differences, and FFN partial computation, which selectively processes high-variation tokens. Unlike retraining- or pruning-based methods, it is inserted directly into pretrained frameworks at inference time.

Fig. 1 illustrates the overall framework. Given multi-view driving images, the original model extracts hierarchical interaction features. D²Drive then evaluates adjacent-layer feature differences for dynamic exit and computes token variation scores for partial FFN execution, reducing layer-level and token-level computation while preserving the original architecture and task heads.

3.2 Dynamic Exit Based on Feature Differences

Deep interaction layers often stabilize as inference proceeds. When adjacent-layer variation is small, deeper layers may contribute limited new information while still consuming computation. We use this behavior to design the feature-difference based dynamic exit mechanism shown in Fig. 2.

Let F_l-1 and F_l denote the output features of the (l-1)-th and l-th interaction layers:

$$F_{l-1}, F_l \in \mathbb{R}^{B\times N\times C}, \tag{1}$$

where B, N, and C represent the batch size, token number, and channel dimension, respectively. To measure the information variation introduced by the current layer, we compute the feature difference between adjacent layers as

$$D_l = \frac{1}{B}\sum_{b=1}^{B} d\left(F_l^b, F_{l-1}^b\right), \tag{2}$$

where d(·) denotes a feature distance metric. In practice, cosine distance or L2 distance can be used. When cosine distance is adopted, the feature difference is computed as

$$d\left(F_l^b, F_{l-1}^b\right) = 1 - \frac{1}{N}\sum_{i=1}^{N} \frac{\left\langle F_l^{b,i}, F_{l-1}^{b,i} \right\rangle}{\left\| F_l^{b,i} \right\|_2 \left\| F_{l-1}^{b,i} \right\|_2}. \tag{3}$$

Since different layers may exhibit different feature distributions and convergence characteristics, we assign an independent threshold τ_l to each candidate exit layer. Based on the difference in Eq. (2), the exit decision is formulated as

$$Exit(l) = \begin{cases} 1, & D_l < \tau_l, \\ 0, & D_l \geq \tau_l. \end{cases} \tag{4}$$

If the exit condition in Eq. (4) is satisfied, inference terminates at the current layer and the corresponding feature is directly used for downstream tasks. Otherwise, the model continues to deeper layers. If no exit condition is triggered, the final-layer output is returned.

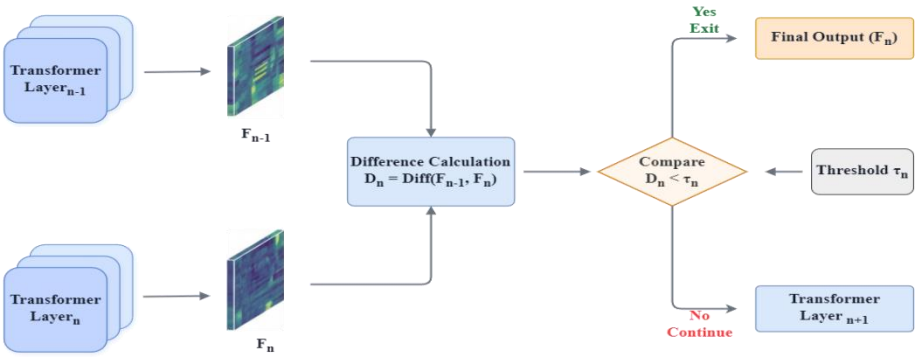

Fig. 2. Dynamic exit based on adjacent-layer feature differences.

Unlike conventional early-exit methods based on auxiliary classifiers or confidence estimates, this strategy directly measures feature stability across layers. It requires neither additional training objectives nor architectural changes, and can be applied to pre-trained autonomous driving frameworks during inference.

3.3 FFN Partial Computation Based on Token Variation

Although dynamic exit reduces the number of executed interaction layers, FFN modules inside the remaining layers still account for substantial computation. Tokens also vary unevenly across adjacent layers: some contain important updates, whereas others remain stable. Processing all tokens equally therefore introduces avoidable FFN computation.

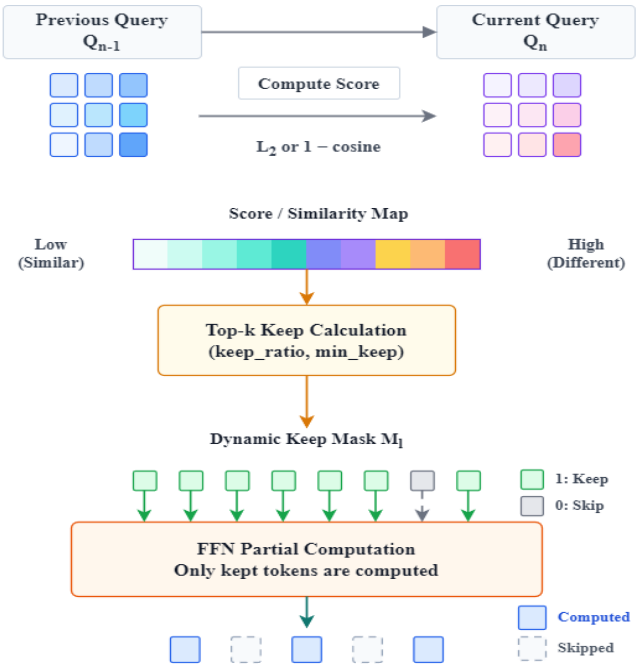

Fig. 3. FFN partial computation based on token variation.

We address this issue with an FFN partial computation strategy based on token variation. Tokens with larger cross-layer variation are preserved for FFN computation, while stable tokens bypass redundant FFN operations. As shown in Fig. 3, given the current query Q_l and the previous query Q_l-1,

$$Q_l, Q_{l-1} \in \mathbb{R}^{B\times N\times C}, \tag{5}$$

we first compute variation scores for each token. For the i-th token of the b-th sample, the score can be computed using L2 distance:

$$s_l^{b,i} = \|Q_l^{b,i} - Q_{l-1}^{b,i}\|_2, \tag{6}$$

or cosine distance:

$$s_l^{b,i} = 1 - \frac{\langle Q_l^{b,i}, Q_{l-1}^{b,i}\rangle}{\|Q_l^{b,i}\|_2 \|Q_{l-1}^{b,i}\|_2}. \tag{7}$$

A larger score indicates that the token undergoes a larger feature update and should therefore be preserved for FFN computation. Given a keep ratio r and a minimum keep number K_min, the number of preserved tokens is determined by

$$K = max(K_{min}, \lfloor rN \rfloor). \tag{8}$$

The top-K tokens with the highest scores are selected to construct a binary keep mask:

$$M_l^{b,i} = \begin{cases} 1, \text{i} \in \text{TopK}(s_l{}^b, K), \\ 0, \qquad \text{otherwise.} \end{cases} \tag{9}$$

Only tokens selected by the keep mask participate in FFN computation, while the remaining tokens bypass the FFN and directly preserve their original features. The resulting output feature is expressed as

$$X_l^{b,i} = M_l^{b,i} \cdot FFN\left(X_l^{b,i}\right) + (1 - M_l^{b,i}) \cdot X_l^{b,i}. \tag{10}$$

Thus, FFN computation is concentrated on informative high-variation tokens, and stable tokens avoid unnecessary FFN updates. The strategy changes only FFN execution; attention, normalization, residual connections, and task heads remain unchanged, making integration possible without retraining.

3.4 Training-free Dynamic Inference Procedure

During inference, D²Drive processes the interaction layers of a pretrained autonomous driving framework sequentially. The original model parameters, attention operations, normalization layers, residual connections, and task heads are kept unchanged. Therefore, the proposed strategy only changes the execution path during inference rather than modifying the model architecture.

For each executed layer, attention follows the original framework. Before the FFN module, D²Drive computes token variation from adjacent queries and builds a keep mask. High-variation tokens enter FFN computation, while stable tokens bypass redundant FFN operations and preserve their input features.

After the current layer finishes execution, the feature difference between adjacent layers D_l is evaluated and compared with the corresponding layer-specific threshold τ_l. If the exit condition is satisfied, inference terminates at the current layer and the current feature representation is used for downstream prediction. Otherwise, the model proceeds to the next layer. When no early exit is triggered, the final-layer feature is used as the output of the original model.

The procedure reduces computation at two levels: dynamic exit skips redundant layers, and partial FFN computation lowers token-level FFN cost inside executed layers. Both operations are applied only during inference, so pretrained end-to-end driving models can be used without retraining.

4 Experiments

4.1 Dataset

All experiments use the nuScenes full dataset[2], a large-scale autonomous driving benchmark with 1,000 scenes collected in Boston and Singapore. Each scene lasts about 20 seconds and provides synchronized multi-sensor data from surround-view cameras, LiDAR, radar, GPS, and IMU.

We use the nuScenes dataset to evaluate the effectiveness and generality of D²Drive across different end-to-end autonomous driving frameworks. Following the standard setting of existing autonomous driving methods, the dataset is split into training, validation, and testing subsets. Since D²Drive is a training-free dynamic inference framework, no additional model retraining is introduced during the acceleration process. Instead, the proposed strategy is directly applied to pretrained models during inference on the nuScenes validation set.

4.2 Experimental Setup

We summarize the experimental settings from four aspects: platform, implementation settings, evaluation metrics, and baseline models.

Platform. Experiments were conducted on a Linux server with eight NVIDIA Tesla P40 GPUs and CUDA 11.8. Models were implemented in PyTorch using the official codebases, evaluation settings, and pretrained checkpoints of UniAD[16], VAD[19], SparseDrive[28], and MomAD[27]. D²Drive was applied only during inference.

Implementation Settings. We evaluate D²Drive on UniAD[16], VAD[19], SparseDrive[28], and MomAD[27] using the original pretrained models. Dynamic exit is applied between adjacent interaction layers, and FFN partial computation is applied to FFN modules. Thresholds and keep ratios are selected according to each framework's layer structure and feature distribution.

Evaluation Metrics. We evaluate both efficiency and task performance. GFLOPs measure computational cost. Planning is measured by L2 displacement error and object-box collision rate at multiple horizons[2,16,19]. Perception tracking is evaluated with AMOTA, AMOTP, and RECALL following the nuScenes tracking protocol[2], and NDS follows the official detection protocol[2]. Bench2Drive[18] further motivates evaluating multiple driving abilities rather than a single efficiency or accuracy indicator.

Baseline Models. UniAD[16], VAD[19], SparseDrive[28], and MomAD[27] are used as baselines. For each framework, we compare the original inference process with the D²Drive-accelerated version, focusing on GFLOPs reduction and changes in planning, tracking, and detection metrics.

4.3 Main Results

To evaluate the effectiveness of D²Drive, we conduct experiments on four representative end-to-end autonomous driving frameworks, including UniAD[16], VAD[19], SparseDrive[28], and MomAD[27]. For each framework, we compare the original pretrained model with its accelerated versions using ATS[11] and D²Drive. Since all acceleration strategies are applied during inference, the pretrained weights of the original models are kept unchanged.

Table 1 reports whole-model GFLOPs. ATS serves as a token-level acceleration baseline, while D²Drive exploits both adjacent-layer feature redundancy and token redundancy inside FFN modules. The results show whether training-free dynamic inference can reduce computation consistently across different architectures.

Table 1. Overall GFLOPs comparison across representative end-to-end autonomous driving frameworks.

Model	Method	Modules	GFLOPs	GFLOPs Red.
UniAD[16]	Baseline	--	1608.6492	--
	ATS[11]	Perception	1600.2797	8.3695
	D²Drive	Perception	1585.7596	**22.8896**
VAD[19]	Baseline	--	903.7724	--
	ATS[11]	Perception	895.7056	8.0668
	D²Drive	Perception	892.0622	**11.7102**
SparseDrive[28]	Baseline	--	199.6199	--
	ATS[11]	Det. + Map	198.3478	1.2721
	D²Drive	Det. + Map	192.8166	**6.8033**
MomAD[27]	Baseline	--	199.6213	--
	ATS[11]	Det. + Map	198.2763	1.3450
	D²Drive	Det. + Map	194.1237	**5.4976**

4.4 Results Analysis

1) Overall Efficiency: Table 1 presents the overall GFLOPs comparison across different end-to-end autonomous driving frameworks, including UniAD[16], VAD[19], SparseDrive[28], and MomAD[27]. Because the proposed method is applied only during inference, the pretrained model parameters and task heads remain unchanged. The computation reduction therefore comes from dynamically skipping redundant execution rather than compressing or retraining the original models.

Compared with ATS[11], D²Drive additionally exploits adjacent-layer feature stability and FFN token redundancy. It therefore reduces computation at both the layer and token levels, rather than relying only on token sampling.

2) Module-Level Reduction: D²Drive reduces computation through dynamic exit across layers and partial FFN computation within executed layers. The two mechanisms are complementary: dynamic exit skips deeper interaction blocks once features stabilize, while partial FFN computation restricts FFN operations to high-variation tokens. Table 2 further reports module-level reductions. Unlike Table 1, it lists the original GFLOPs of the optimized modules and normalizes the savings within these modules, making the reductions from ATS and D²Drive easier to compare.

For VAD[19] and UniAD[16], the comparison is conducted on perception transformer modules, which are responsible for multi-view feature interaction and temporal reasoning. For SparseDrive[28] and MomAD[27], we report the results on the detection and map branches, where repeated interaction modules still introduce noticeable computation.

Table 2. Module-level GFLOPs reduction of the optimized components.

Model	Module	Module GFLOPs	ATS Red. (%)	D²Drive Red. (%)	Gain (pp)
UniAD[16]	Perception	407.2691	2.06	5.62	**3.57**
VAD[19]	Perception	297.3951	2.71	3.94	**1.22**
SparseDrive[28]	Det. + Map	40.6437	3.13	16.74	**13.61**
MomAD[27]	Det. + Map	40.6437	3.51	13.53	**10.02**

3) Accuracy Preservation: In addition to computational efficiency, we further evaluate whether D²Drive preserves the original task performance after acceleration. Since different autonomous driving frameworks focus on different downstream tasks and adopt different evaluation protocols, we conduct the analysis using both common planning metrics and framework specific perception metrics.

For planning evaluation, all frameworks are assessed using L2 displacement error and object-box collision rate at 1s, 2s, and 3s horizons. These metrics evaluate trajectory accuracy and planning safety over both short- and long-term horizons.

Table 3. Planning performance under different inference acceleration strategies.

Model	Method	L2 1s ↓	L2 2s ↓	L2 3s ↓	L2 Avg. ↓	Obj. Box Col. 1s ↓	Obj. Box Col. 2s ↓	Obj. Box Col. 3s ↓	Obj. Box Col. Avg. ↓
UniAD[16]	Baseline	0.4421	0.8786	1.5177	0.9461	0.130	0.200	0.530	0.287
	ATS[11]	0.5455	1.1034	1.8712	1.1734	0.130	0.370	0.850	0.450
	D²Drive	0.5046	0.9772	1.6802	**1.0540**	0.100	0.230	0.430	**0.253**
VAD[19]	Baseline	0.4083	0.6979	1.0510	0.719	0.068	0.171	0.407	0.215
	ATS[11]	0.6050	1.0933	1.6857	1.1280	0.244	0.381	0.824	0.483
	D²Drive	0.4302	0.7940	1.2613	**0.8285**	0.107	0.200	0.612	**0.306**
SparseDrive[28]	Baseline	0.2965	0.5766	0.9519	0.6083	0.010	0.054	0.221	0.095
	ATS[11]	0.3379	0.6638	1.0951	0.6989	0.020	0.137	0.573	0.243
	D²Drive	0.3166	0.6259	1.0429	**0.6618**	0.010	0.029	0.254	**0.098**
MomAD[27]	Baseline	0.2736	0.5429	0.9085	0.5750	0.020	0.103	0.303	0.142
	ATS[11]	0.3072	0.6003	0.9940	0.6338	0.029	0.107	0.404	0.180
	D²Drive	0.2823	0.5627	0.9420	**0.5957**	0.029	0.117	0.378	**0.175**

For perception evaluation, UniAD[16], SparseDrive[28], and MomAD[27] are evaluated with AMOTA, AMOTP, RECALL, and NDS. Since VAD[19] does not include a dedicated multi-object tracking branch, only NDS is reported following its official protocol.

We include Adaptive Token Sampling (ATS)[11] as a representative token sampling baseline. Unlike ATS, D²Drive jointly uses adjacent-layer feature stability and FFN token redundancy, enabling computation reduction at both the layer and token levels.

Table 4. Perception performance under different inference acceleration strategies.

Model	Method	AMOTA ↑	AMOTP ↓	RECALL ↑	NDS ↑
UniAD[16]	Baseline	0.3857	1.3174	0.4706	0.4501
	ATS[11]	0.2537	1.4370	0.3698	0.3867
	D²Drive	**0.3454**	**1.3351**	**0.4354**	**0.4264**
VAD[19]	Baseline	--	--	--	0.4602
	ATS[11]	--	--	--	0.3272
	D²Drive	--	--	--	**0.4624**
SparseDrive[28]	Baseline	0.3687	1.2573	0.5006	0.5233
	ATS[11]	0.1102	1.6493	0.2734	0.3765
	D²Drive	**0.3214**	**1.2673**	**0.4502**	**0.5019**
MomAD[27]	Baseline	0.3655	1.2802	0.4772	0.5105
	ATS[11]	0.1070	1.6656	0.3187	0.3579
	D²Drive	**0.3585**	**1.2736**	**0.5088**	**0.5072**

Tables~3 and~4 compare the task performance of the original models and their accelerated variants.

For planning evaluation, the key indicators are the variations in L2 error and object-box collision rate after acceleration. Smaller changes in these metrics indicate that the generated ego trajectories remain close to the original baselines and do not introduce obvious additional safety risks. Compared with ATS, D²Drive reduces computation through both dynamic exit and FFN partial computation, which helps preserve planning-related scene information while removing redundant inference operations.

For perception evaluation, AMOTA, AMOTP, RECALL, and NDS are used to check whether acceleration affects temporal association quality, localization precision, tracking recall, and overall detection capability. Since VAD does not provide explicit tracking outputs, we report NDS as its perception-related metric following the official evaluation protocol.

The preservation of task performance comes from conservative computation skipping: dynamic exit is triggered only when adjacent features become stable, and partial FFN computation keeps high-variation tokens for further processing.

Overall, the results indicate that D²Drive removes redundant computation while preserving task-relevant information more reliably than the token-only baseline.

4) Qualitative Visualization: To further examine whether D²Drive preserves the original driving behavior after acceleration, we provide qualitative visualization results on UniAD[16] in representative nuScenes[2] scenes. UniAD contains dense multi-view perception, temporal reasoning, prediction, and planning modules, making it a suitable example for observing whether inference acceleration changes task-relevant outputs.

We compare the original baseline with Adaptive Token Sampling (ATS)[11] and D²Drive. The selected examples include an urban intersection, a scene with vulnerable road users and public transport, and a cluttered road segment with multiple surrounding agents. The results are shown in Figs. 4-6.

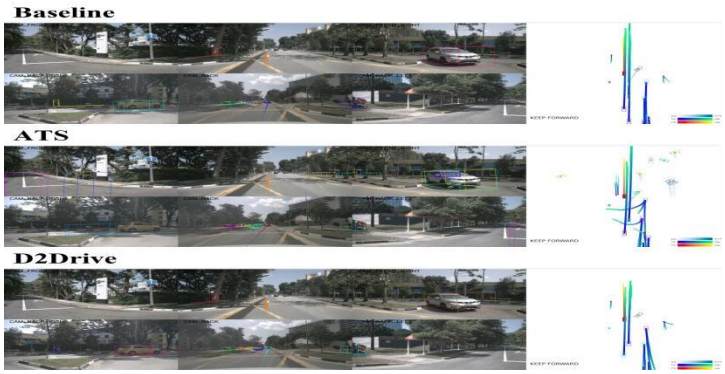

Fig. 4. Qualitative comparison of UniAD[16] on an urban intersection scene from nuScenes[2]. Rows show Baseline, ATS[11], and D²Drive.

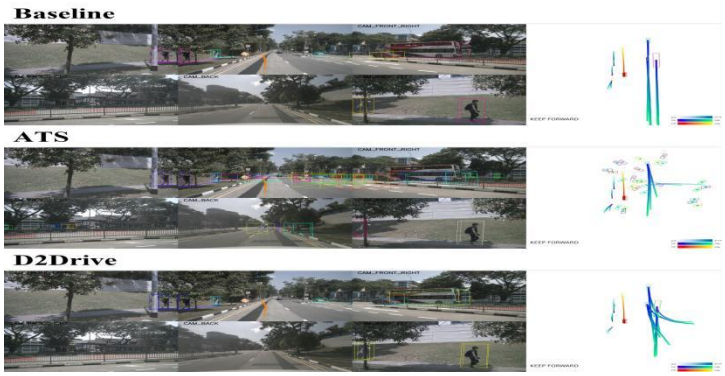

Fig. 5. Qualitative comparison of UniAD[16] on a nuScenes[2] scene with pedestrians and public transport. Rows show Baseline, ATS[11], and D²Drive.

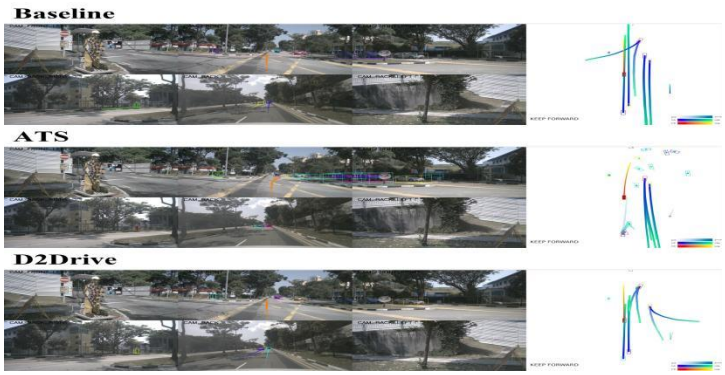

Fig. 6. Qualitative comparison of UniAD[16] on a cluttered nuScenes[2] road scene. Rows show Baseline, ATS[11], and D²Drive.

As shown in Figs. 4-6, D²Drive preserves planning trajectories and perception outputs close to the UniAD baseline across different urban scenes. ATS produces more scattered object hypotheses in these examples, while D²Drive better maintains task-relevant BEV structures during acceleration.

5) Model-wise Discussion: GFLOPs reduction varies across models because their computation is distributed differently. D²Drive benefits models with deep interaction blocks, dense token communication, or expensive FFN layers more strongly, whereas architectures with sparse representations or lightweight branches may show smaller whole-model reductions.

For VAD[19] and UniAD[16], perception transformer and similar dense feature aggregation modules play an important role in multi-view perception, temporal modeling, and planning-oriented reasoning. These modules usually contain multiple interaction layers and dense communication among image, BEV, agent, or map tokens, which contribute a considerable portion of the overall computational cost. Therefore, applying dynamic exit and FFN partial computation to these modules can effectively reduce redundant computation. However, these models also rely heavily on high-quality intermediate representations for downstream tasks such as planning and tracking. As a result, the exit thresholds and token keep ratios should be carefully selected to avoid prematurely discarding useful features.

For SparseDrive[28] and MomAD[27], the baseline architectures already introduce sparse scene representations, query-based object modeling, or computation designs for individual branches. These mechanisms reduce part of the redundant spatial computation before applying D²Drive, leading to relatively lower baseline computational cost. Consequently, the relative GFLOPs reduction may be smaller than that of models with denser interaction structures. Nevertheless, D²Drive can still provide additional acceleration by exploiting two types of redundancy that remain in these frameworks: representation stability across adjacent interaction layers and token redundancy within FFN modules. This observation shows that even sparse or modular architectures still contain exploitable redundancy during inference.

The optimization effect depends on the computational bottleneck of each model. D²Drive brings larger gains when deep interaction and FFN modules dominate the cost. If backbones, task heads, or sparse operations dominate, whole-model GFLOPs reduction may be smaller even though the optimized modules still save computation.

D²Drive is therefore most useful when computation is concentrated in deep interaction and FFN modules. Since it does not require retraining, pruning, or task-head modification, it provides a practical inference-time acceleration option for pretrained autonomous driving frameworks.

4.5 Ablation Experiments

We isolate the two components through ablation experiments. Dynamic exit and FFN partial computation are enabled separately as Diff-Exit and Partial-FFN to examine their individual effects on computation reduction and accuracy preservation.

Following common practice, UniAD[16] is selected for detailed ablation because it contains dense perception and interaction modules. GFLOPs measure efficiency, while planning and perception metrics evaluate accuracy preservation under the same protocol as the main experiments.

1) Effect of Dynamic Exit: We first evaluate the proposed dynamic exit strategy by enabling only the Diff-Exit module while disabling FFN partial computation. The objective is to analyze whether redundant interaction layers can be adaptively skipped without introducing noticeable performance degradation. All experiments in this ablation study are conducted on UniAD[16], which contains dense interaction modules for perception, prediction, and planning.

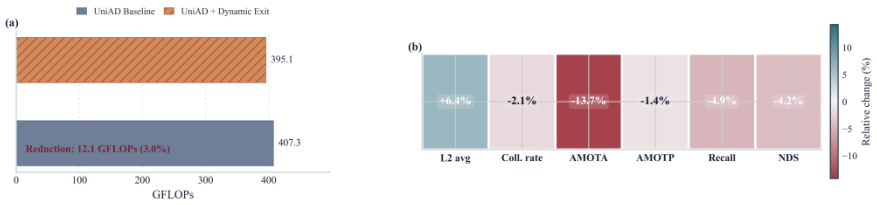

Fig. 7. Ablation study of dynamic exit on UniAD[16].

Fig. 7 shows the effect of enabling Diff-Exit alone, focusing on whether redundant layers can be skipped while keeping metric variations within an acceptable range.

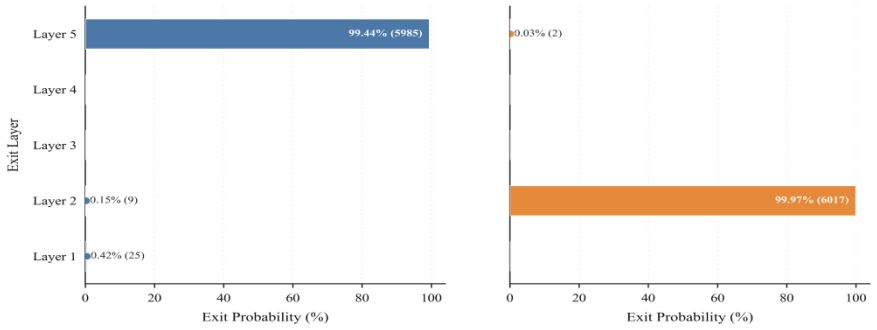

Fig. 8. Exit-layer distribution on UniAD[16].

This analysis is motivated by the observation that deeper interaction layers often generate highly similar feature representations once the scene semantics have become sufficiently stable. Continuing full-depth inference may therefore introduce redundant computation with limited additional benefit. By monitoring feature differences between adjacent layers during inference, Diff-Exit aims to terminate computation adaptively and reduce redundancy across layers.

To further analyze the inference behavior of Diff-Exit, we visualize the distribution of exit layers in Fig. 8.

2) Effect of FFN Partial Computation: Next, we evaluate the effectiveness of the proposed FFN partial computation strategy by enabling only the Partial-FFN module while disabling dynamic exit. Unlike Diff-Exit, which reduces redundancy across interaction layers, Partial-FFN focuses on reducing redundant computation over tokens within each executed layer. All experiments are conducted on UniAD[16], following the same evaluation protocol as the main experiments.

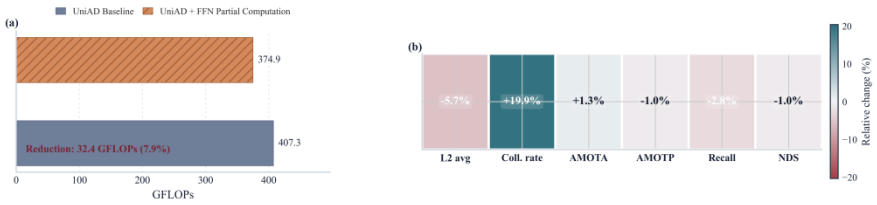

Fig. 9. Ablation study of FFN partial computation on UniAD[16].

Fig. 9 shows the effect of enabling Partial-FFN alone when the dynamic exit module is disabled. This isolated setting helps determine whether FFN reduction over tokens can provide useful savings without causing large variations in trajectory prediction or perception quality.

The saving mainly comes from reducing redundant FFN operations on low variation tokens. Since FFN modules account for a considerable proportion of computation in deep interaction architectures, selectively processing informative tokens can lower the overall computational cost. Meanwhile, preserving tokens with large query variations helps maintain important scene semantics and dynamic object information.

Fig. 10. Average token keep ratio across UniAD[16] perception encoder layers.

We further visualize the average token keep ratio across interaction layers in Fig. 10.

Fig. 10 shows that the average keep ratio varies across perception encoder layers. This layer-wise variation suggests that useful token updates are not uniformly distributed, supporting dynamic token selection instead of identical FFN computation for all tokens.

5 Conclusion

We propose D²Drive, a training-free dynamic inference framework for end-to-end autonomous driving. The proposed method reduces redundant computation in pretrained models without requiring retraining, parameter pruning, or modification of task heads. D²Drive consists of two complementary components: a dynamic exit mechanism that terminates inference when representations in adjacent layers become sufficiently stable, and an FFN partial computation mechanism that selectively processes tokens with larger variations across layers.

We conduct experiments on UniAD[16], VAD[19], SparseDrive[28], and MomAD[27] to evaluate computation for the whole model, computation inside selected modules, and planning, tracking, and detection performance. This evaluation design makes it possible to analyze the balance between efficiency and accuracy of D²Drive across both dense interaction frameworks and sparse frameworks with branch-based computation. The model analysis also clarifies that the acceleration effect depends on the computation distribution of each architecture, especially the proportion of cost located in deep interaction and FFN modules.

Future work will study adaptive threshold selection and task-aware token selection to improve robustness across driving scenarios. We will also evaluate deployment on edge devices and autonomous driving platforms to measure practical latency, memory efficiency, and real-time performance.

6 References

1. Bolya, Daniel, Fu, Cheng-Yang, Dai, Xiaoliang, Zhang, Peizhao, Feichtenhofer, Christoph, Hoffman, Judy. Token Merging: Your ViT but Faster. International Conference on Learning Representations. 2023.
2. Caesar, Holger, Bankiti, Varun, Lang, Alex H., Vora, Sourabh, Liong, Venice Erin, Xu, Qiang, Krishnan, Anush, Pan, Yu, Baldan, Giancarlo, Beijbom, Oscar. nuScenes: A multimodal dataset for autonomous driving. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.
3. Carion, Nicolas, Massa, Francisco, Synnaeve, Gabriel, Usunier, Nicolas, Kirillov, Alexander, Zagoruyko, Sergey. End-to-End Object Detection with Transformers. European Conference on Computer Vision. 2020.
4. Chen, Junjie, Liu, Xuyang, Wen, Zichen, Wang, Yiyu, Huang, Siteng, Chen, Honggang. Variation-aware Vision Token Dropping for Faster Large Vision-Language Models. 2025.
5. Chen, Li, Wu, Penghao, Chitta, Kashyap, Jaeger, Bernhard, Geiger, Andreas, Li, Hongyang. End-to-end Autonomous Driving: Challenges and Frontiers. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2024.

6. Chen, Li, Wu, Penghao, Jia, Xiaosong, Qiao, Yu, Yan, Junchi, Li, Hongyang. Trajectory-Guided Control Prediction for End-to-End Autonomous Driving: A Simple yet Strong Baseline. Advances in Neural Information Processing Systems. 2022.
7. Chen, Shaoyu, Jiang, Bo, Gao, Hao, Liao, Bencheng, Xu, Qing, Zhang, Qian, Huang, Chang, Liu, Wenyu, Wang, Xinggang. VADv2: End-to-end vectorized autonomous driving via probabilistic planning. arXiv preprint arXiv:2402.13243. 2024.
8. Chen, Zhili, Ye, Maosheng, Xu, Shuangjie, Cao, Tongyi, Chen, Qifeng. PPAD: Iterative Interactions of Prediction and Planning for End-to-End Autonomous Driving. European Conference on Computer Vision. 2024.
9. Chitta, Kashyap, Prakash, Aditya, Jaeger, Bernhard, Yu, Zehao, Renz, Katrin, Geiger, Andreas. TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2023.
10. Dosovitskiy, Alexey, others. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. International Conference on Learning Representations. 2021.
11. Fayyaz, Mohsen, Kouhpayegani, Soroush Abbasi, Jafari, Farnoush Rezaei, Sommerlade, Eric, Joze, Hamid Reza Vaezi, Pirsiavash, Hamed, Gall, Juergen. Adaptive Token Sampling for Efficient Vision Transformers. European Conference on Computer Vision. 2022.
12. Han, Song, Pool, Jeff, Tran, John, Dally, William. Learning both Weights and Connections for Efficient Neural Networks. Advances in Neural Information Processing Systems. 2015.
13. Hinton, Geoffrey, Vinyals, Oriol, Dean, Jeff. Distilling the Knowledge in a Neural Network. arXiv preprint arXiv:1503.02531. 2015.
14. Hou, Lu, Huang, Zhiqi, Shang, Lifeng, Jiang, Xin, Chen, Xiao, Liu, Qun. DynaBERT: Dynamic BERT with Adaptive Width and Depth. Advances in Neural Information Processing Systems. 2020.
15. Hu, Shengchao, Chen, Li, Wu, Penghao, Li, Hongyang, Yan, Junchi, Tao, Dacheng. ST-P3: End-to-End Vision-Based Autonomous Driving via Spatial-Temporal Feature Learning. European Conference on Computer Vision. 2022.
16. Hu, Yihan, Yang, Jiazhi, Chen, Li, Li, Keyu, Sima, Chonghao, Zhu, Xizhou, Chai, Siqi, Du, Senyao, Lin, Tianwei, Wang, Wenhai, Lu, Lewei, Jia, Xiaosong, Liu, Qiang, Dai, Jifeng, Qiao, Yu, Li, Hongyang. Planning-oriented Autonomous Driving. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.
17. Jacob, Benoit, Kligys, Skirmantas, Chen, Bo, Zhu, Menglong, Tang, Guangrun, Howard, Andrew, Adam, Hartwig, Kalenichenko, Dmitry. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018.
18. Jia, Xiaosong, Yang, Zhenjie, Li, Qifeng, Zhang, Zhiyuan, Yan, Junchi. Bench2Drive: Towards Multi-Ability Benchmarking of Closed-Loop End-To-End Autonomous Driving. Advances in Neural Information Processing Systems. 2024.
19. Jiang, Bo, Chen, Shaoyu, Xu, Qing, Liao, Bencheng, Chen, Jiajie, Zhou, Helong, Zhang, Qian, Liu, Wenyu, Huang, Chang, Wang, Xinggang. VAD: Vectorized Scene Representation for Efficient Autonomous Driving. Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023.
20. Kaya, Yigitcan, Hong, Sanghyun, Dumitras, Tudor. Shallow-Deep Networks: Understanding and Mitigating Network Overthinking. International Conference on Machine Learning. 2019.
21. Li, Zhiqi, Wang, Wenhai, Li, Hongyang, Xie, Enze, Sima, Chonghao, Lu, Tong, Yu, Qiao, Dai, Jifeng. BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers. European Conference on Computer Vision. 2022.

22. Liang, Youwei, Ge, Chong, Tong, Zhan, others. EViT: Expediting Vision Transformers via Token Reorganizations. International Conference on Learning Representations. 2022.
23. Liu, Weijie, Zhou, Peng, Zhao, Zhe, Wang, Zhiruo, Deng, Haotang, Ju, Qi. FastBERT: A Self-Distilling BERT with Adaptive Inference Time. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020.
24. Liu, Zhijian, Tang, Haotian, Amini, Alexander, Yang, Xinyu, Mao, Huizi, Rus, Daniela, Han, Song. BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation. 2023 IEEE International Conference on Robotics and Automation. 2023.
25. Rao, Yongming, Zhao, Wenliang, Zhu, Zheng, others. DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. Advances in Neural Information Processing Systems. 2021.
26. Shao, Hao, Wang, Letian, Chen, Ruobing, Li, Hongsheng, Liu, Yu. Safety-Enhanced Autonomous Driving Using Interpretable Sensor Fusion Transformer. Conference on Robot Learning. 2023.
27. Song, Ziying, Jia, Caiyan, Liu, Lin, Pan, Hongyu, Zhang, Yongchang, Wang, Junming, Zhang, Xingyu, Xu, Shaoqing, Yang, Lei, Luo, Yadan. Don't Shake the Wheel: Momentum-Aware Planning in End-to-End Autonomous Driving. arXiv preprint arXiv:2503.03125. 2025.
28. Sun, Wenchao, Lin, Xuewu, Shi, Yining, Zhang, Chuang, Wu, Haoran, Zheng, Sifa. SparseDrive: End-to-end autonomous driving via sparse scene representation. 2025 IEEE International Conference on Robotics and Automation (ICRA). 2025.
29. Teerapittayanon, Surat, McDanel, Bradley, Kung, H. T.. BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. 2016 23rd International Conference on Pattern Recognition. 2016.
30. Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, others. Attention Is All You Need. Advances in Neural Information Processing Systems. 2017.
31. Xin, Ji, Tang, Raphael, Lee, Jaejun, Yu, Yaoliang, Lin, Jimmy. DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020.
32. Yue, Yang, Wang, Yulin, Kang, Bingyi, Han, Yizeng, Wang, Shenzhi, Song, Shiji, Feng, Jiashi, Huang, Gao. DeeR-VLA: Dynamic Inference of Multimodal Large Language Models for Efficient Robot Execution. Advances in Neural Information Processing Systems. 2024.
33. Zhou, Wangchunshu, Xu, Canwen, Ge, Tao, McAuley, Julian, Xu, Ke, Wei, Furu. BERT Loses Patience: Fast and Robust Inference with Early Exit. Advances in Neural Information Processing Systems. 2020.
34. Zhu, Xizhou, Su, Weijie, Lu, Lewei, Li, Bin, Wang, Xiaogang, Dai, Jifeng. Deformable DETR: Deformable Transformers for End-to-End Object Detection. International Conference on Learning Representations. 2021.
35. Zhou S, Zhang X, Chu X, et al. Fastpillars: A deployment-friendly pillar-based 3d detector. IEEE Transactions on Circuits and Systems for Video Technology, 2025.
36. Zhou S, Yuan Z, Yang D, et al. Pillarhist: A quantization-aware pillar feature encoder based on height-aware histogram, Proceedings of the computer vision and pattern recognition conference. 2025: 27336-27345.
37. Zhou S, Li L, Zhang X, et al. LiDAR-PTQ: Post-Training Quantization for Point Cloud 3D Object Detection, The Twelfth International Conference on Learning Representations. 2026.
38. Li, T., Wang, G., Yu, B., Liu, Y., & Sun, W. Don't let the information slip away. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8504-8513).2026.