
 CodeMerge: Codebook-Guided Model Merging for Robust Test-Time Adaptation in Autonomous Driving

Huitong Yang Zhuoxiao Chen Fengyi Zhang Zi Huang Yadan Luo*

UQMM Lab, The University of Queensland

{huitong.yang, zhuoxiao.chen, fengyi.zhang, helen.huang, y.luo}@uq.edu.au

Abstract

Maintaining robust 3D perception under dynamic and unpredictable test-time conditions remains a critical challenge for autonomous driving systems. Existing test-time adaptation (TTA) methods often fail in high-variance tasks like 3D object detection due to unstable optimization and sharp minima. While recent model merging strategies based on linear mode connectivity (LMC) offer improved stability by interpolating between fine-tuned checkpoints, they are computationally expensive, requiring repeated checkpoint access and multiple forward passes. In this paper, we introduce CodeMerge, a lightweight and scalable model merging framework that bypasses these limitations by operating in a compact latent space. Instead of loading full models, CodeMerge represents each checkpoint with a low-dimensional fingerprint derived from the source model’s penultimate features and constructs a key-value codebook. We compute merging coefficients using ridge leverage scores on these fingerprints, enabling efficient model composition without compromising adaptation quality. Our method achieves strong performance across challenging benchmarks, improving end-to-end 3D detection 14.9% NDS on nuScenes-C and LiDAR-based detection by over 7.6% mAP on nuScenes-to-KITTI, while benefiting downstream tasks such as online mapping, motion prediction and planning even without training. The code is released at <https://github.com/UQHTy/CodeMerge>.

1 Introduction

Real-world autonomous driving scenarios often encounter rapid and unpredictable environmental variations, such as sudden adverse weather conditions (*e.g.*, fog, snow) or sensor malfunctions (*e.g.*, dropped frames, missing beams) arising from LiDAR and camera systems, as illustrated in Fig. 1. These abrupt disruptions momentarily render 3D perception modules partially or fully “blind”, propagating erroneous decision-making downstream and leading to severe safety hazards in the end-to-end autonomous driving (AD) pipeline. Consequently, a critical yet unresolved question emerges: *Can perception models efficiently and robustly adapt onboard to address such unforeseen distributional shifts?*

Test-time adaptation (TTA) offers a promising direction by enabling models to adapt online during inference. Prior TTA approaches typically handle shifts by aligning BatchNorm statistics [44, 34], enforcing consistency through data augmentations [46], or minimizing sharpness via adversarial perturbations [14, 34]. Nonetheless, when directly extending them to complex tasks such as 3D detection, these approaches often suffer from brittle optimization dynamics and fall into sharp local minima, which can lead to the loss of previously acquired generalization and the ability to cope with future task distributions [9].

*Corresponding author.

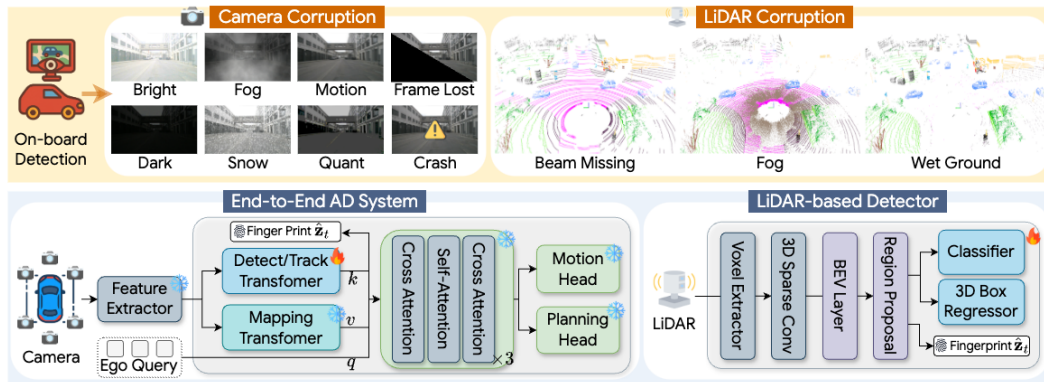


Figure 1: **Overview of real-world test-time shifts (top) and 3D perception systems considered in this work (bottom).** We study test-time adaptation (TTA) in two settings: (1) an end-to-end autonomous driving system and (2) a modular LiDAR-based detector, both affected by adverse weather and sensor failures. CodeMerge enables efficient TTA by leveraging compact fingerprints to guide model merging.

Recent studies improve *long-term* adaptation stability by leveraging model merging techniques [24] grounded in linear mode connectivity (LMC), which posits that models fine-tuned on different target samples but initialized from *the same* pretrained source model are “*linearly connected*” in weight space. Thus, interpolating between such models has been shown to produce reliable pseudo-labels and alleviate model collapse issues in TTA [34]. Techniques such as Mean Teacher and its variants exponentially average the weights of past models, but often underutilize valuable diversity across past checkpoints. More recently, Model Synergy (MOS) [8] extends this idea by maintaining a buffer of Top- K important checkpoints and dynamically merging them using synergy weights, computed via kernel similarities among each model’s predictions of the current test sample. While effective, MOS incurs significant overhead from repeatedly checkpoint loading and performing K forward passes for calculating synergy weights, limiting its scalability in high-throughput driving applications.

In this work, we introduce a codebook-guided model merging (CodeMerge) approach for adapting 3D perception modules against various shifts at test time. The core idea is to represent each fine-tuned checkpoint $\Phi_{\Theta(t)}$ by a compact “fingerprint” derived from the source model’s penultimate activations. These fingerprints serve as keys in a model codebook, mapping to their corresponding checkpoint weights. Crucially, correlations in this low-dimensional fingerprint space reliably mirror those in the high-dimensional weight space (see Fig. 3), enabling informed merging decisions without loading full model parameters. CodeMerge employs ridge leverage scores to rank the informativeness of fingerprints, a technique theoretically linked to approximations of the inverse Hessian in the parameter space. This procedure needs memory that scales only with the fingerprint dimension and adds negligible latency, yet it lifts end-to-end 3D detection NDS by 14.9%, tracking AMOTA on the nuScenes-C corruption benchmark by 19.3%, and LiDAR-based detection 3D mAP by 7.6% on the challenging nuScenes-to-KITTI shift. These improvements seamlessly propagate to downstream motion prediction and planning modules without modification or additional training.

2 Preliminaries

We begin by formalizing the problem setting for test-time adaptation (TTA) in 3D object detection and reviewing model merging strategies that exploit linear mode connectivity in such context.

Task Formulation. Let $\Phi_{\Theta(0)} = \phi_{\Theta(0)} \circ h_{\Theta(0)}$ denote a pretrained 3D object detection model, comprising a feature extractor $\phi_{\Theta(0)}(\cdot) : \mathbf{X} \mapsto \mathbf{Z} \in \mathbb{R}^d$ maps an input $\mathbf{x} \in \mathbf{X}$ (e.g., a point cloud or multi-view images) to a latent feature map $\mathbf{z} \in \mathbf{Z}$, and the head regresses 3D boxes $h_{\Theta(0)}(\cdot) : \mathbf{Z} \mapsto \mathcal{Y} \in \mathbb{R}^7$. The goal of TTA is to sequentially adapt the model to a stream of unlabeled target-domain inputs $\mathcal{D}_{\text{tar}} = \{\mathbf{x}_t\}_{t=1}^T$, which may exhibit significant distributional shifts or corruptions. The online adaptation must follow in a single forward-pass setting, incrementally evolving the model parameters $\Theta^{(0)} \rightarrow \Theta^{(1)} \rightarrow \dots \rightarrow \Theta^{(t)}$ to improve detection over time.

Linear Mode Connectivity (LMC). LMC [24, 12, 52] refers to the empirical property that two models $\Theta^{(1)}$ and $\Theta^{(2)}$ trained from a shared initialization (or sufficiently close regions in weight space), can be connected by a “*linear path*” without significant loss degradation. Formally, for any

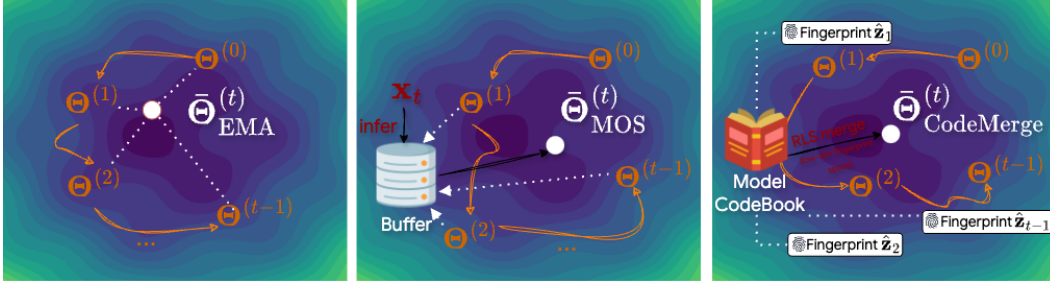


Figure 2: **Conceptual comparison of model merging strategies for TTA.** Unlike EMA (left), which ignores model behavior, or MOS (middle), which requires multiple inferences to compute merging weights, CodeMerge (right) leverages ridge leverage scores in a compact fingerprint space to efficiently guide model merging.

$\lambda \in [0, 1]$,

$$\mathcal{L} \left((1 - \lambda)\Theta^{(1)} + \lambda\Theta^{(2)} \right) \approx (1 - \lambda)\mathcal{L}(\Theta^{(1)}) + \lambda\mathcal{L}(\Theta^{(2)}). \quad (1)$$

This property facilitates efficient model merging through linear interpolation.

Implication for Model Merging in TTA. If LMC holds true between each pair of successive parameters $(\Theta^{(t-1)}, \Theta^{(t)})$ fine-tuned from $\Theta^{(0)}$, then their interpolated model should yield low loss. This underpins methods like *Mean Teacher* shown in Fig. 2, in which teacher models are recursively updated with an exponential moving average (EMA) with a decay factor $\beta \in (0, 1)$:

$$\bar{\Theta}_{\text{EMA}}^{(t)} = \beta \bar{\Theta}_{\text{EMA}}^{(t-1)} + (1 - \beta)\Theta^{(t)} \Rightarrow \bar{\Theta}_{\text{EMA}}^{(t)} = (1 - \beta) \sum_{i=0}^t \beta^{t-i} \Theta^{(i)}. \quad (2)$$

Under LMC, this leads to approximately linear combinations of multi-task losses:

$$\mathcal{L}(\bar{\Theta}_{\text{EMA}}^{(t)}) \approx (1 - \beta) \sum_{i=0}^t \beta^{t-i} \mathcal{L}(\Theta^{(i)}). \quad (3)$$

This shows that averaging can reduce variance from balancing multi-task losses. However, EMA’s coefficients are solely based on time steps rather than model behavior, making it potentially suboptimal.

In contrast, MOS [8] (middle in Fig. 2) adaptively merges model parameters by solving a kernel-weighted least squares problem over a buffer of K candidate checkpoints $\{\Theta^{(i)}\}_{i=1}^K$. Given a test batch $\mathbf{x}_t$, the merged model is computed as:

$$\bar{\Theta}^{(t)} = \sum_{i=1}^K \tilde{w}_i^{(t)} \Theta^{(i)}, \text{ where } \tilde{w}_i^{(t)} = \frac{\sum_j [\mathbf{K}^{(t)}]_{ij}^{-1}}{\sum_{i', j'} [\mathbf{K}^{(t)}]_{i'j'}^{-1}}, \quad (4)$$

$$\mathbf{K}_{ij}^{(t)} = \text{Sim}(\Phi_{\Theta^{(i)}}(\mathbf{x}_t), \Phi_{\Theta^{(j)}}(\mathbf{x}_t)) \cdot \text{Sim}(\phi_{\Theta^{(i)}}(\mathbf{x}_t), \phi_{\Theta^{(j)}}(\mathbf{x}_t)), \quad (5)$$

where kernel matrix $\mathbf{K}^{(t)} \in \mathbb{R}^{K \times K}$ captures pairwise similarity between model outputs under the current batch. To evaluate $\tilde{w}_i^{(t)}$, MOS requires K forward passes over $\mathbf{x}_t$, making it more computationally intensive and thus hard to scale up the horizon K in TTA.

3 Our Approach

We introduce CodeMerge, a codebook-guided model merging scheme for efficient TTA in 3D object detection *without* triggering repeated inference across past models. To achieve this, we construct a model codebook (Sec. 3.1), where each checkpoint is represented by a compact fingerprint derived from intermediate features of a fixed source model. During inference, we compute curvature-aware ridge leverage scores (Sec. 3.2) in the fingerprint space. Finally, we perform a sign-consistent weighted merge of top-scoring candidate models (Sec. 3.3), promoting both stability and diversity.

3.1 Model CodeBook

At each step t , we maintain a model codebook for *all* past checkpoints along the adaptation trajectory, denoted as:

$$\mathcal{C}^{(t)} = \{\hat{\mathbf{z}}_i : \Theta^{(i)}\}_{i=1}^{t-1}. \quad (6)$$

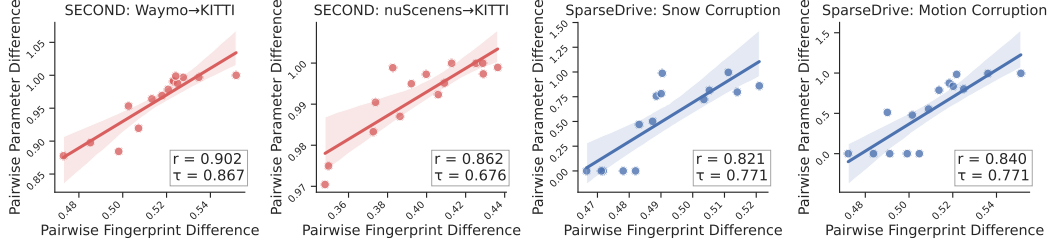


Figure 3: Pairwise fingerprint differences correlate strongly with model weight differences (Pearson r and Kendall Tau $\tau > 0.7$) across SparseDrive [41] and SECOND [55], showing that the low-dimensional fingerprint space reliably reflects parameter space structure.

Each entry is a *key-value* pair, where the key $\hat{\mathbf{z}}_i \in \mathbb{R}^{d'}$ is a low-dimensional fingerprint and the value $\Theta^{(i)}$ is the corresponding checkpoint fine-tuned at time step i . To compute the key $\hat{\mathbf{z}}_i$, we extract intermediate features from the i -th input batch $\mathbf{x}_i$ using a pretrained feature extractor $\phi_{\Theta^{(0)}}$ and randomly project them to a low-dimensional subspace for efficiency:

$$\hat{\mathbf{z}}_i = \text{RandProj}(\phi_{\Theta^{(0)}}(\mathbf{x}_i)). \quad (7)$$

Here, $\text{RandProj}(\cdot) : \mathbb{R}^d \mapsto \mathbb{R}^{d'}$ is implemented via a fixed Gaussian projection matrix where $d' \ll d$ ensures the keys are compact. As the test-time adaptation progresses, we update the codebook incrementally by appending new pairs, *i.e.*, $\mathcal{C}^{(t+1)} \leftarrow (\hat{\mathbf{z}}_t, \Theta^{(t)})$.

3.2 Curvature-Aware Merge Scores

To determine which checkpoints in the codebook should be merged at time step t , we first compute a merge score for each checkpoint $\Theta^{(i)} \in \mathcal{C}^{(t)}$ using the ridge leverage score.

Definition 1 (Ridge Leverage Score (RLS)). Let $\hat{\mathbf{Z}}_{t-1} = [\hat{\mathbf{z}}_1, \dots, \hat{\mathbf{z}}_{t-1}] \in \mathbb{R}^{(t-1) \times d'}$ be the matrix of all stored keys (fingerprints), where $\hat{\mathbf{z}}_i$ be the fingerprint of the i -th candidate model $\Theta^{(i)}$. We define the ridge leverage scores of the fingerprint $\hat{\mathbf{z}}_i$ as

$$s_i^{(t)} = \hat{\mathbf{z}}_i^\top \left(\frac{1}{K} \hat{\mathbf{Z}}_{t-1}^\top \hat{\mathbf{Z}}_{t-1} + \lambda I \right)^{-1} \hat{\mathbf{z}}_i,$$

where λ is a regularization parameter. A high leverage score indicates $\hat{\mathbf{z}}_i$ is influential and lessly observed within the current feature space defined by past direction.

Theoretical Analysis. We now connect this leverage score to the inverse of curvature through the lens of LMC. We begin by revisiting the LMC assumption (Eq. (1)) through a second-order Taylor expansion around $\Theta^{(0)}$:

$$\mathcal{L}(\Theta^{(i)}) \approx \mathcal{L}(\Theta^{(0)}) + \nabla \mathcal{L}^\top \delta_i + \frac{1}{2} \delta_i^\top H \delta_i, \text{ with } H := \nabla_\theta^2 \mathcal{L}(\Theta^{(0)}), \quad (8)$$

where $\delta_i := \Theta^{(i)} - \Theta^{(0)}$ refers the model update direction and $\mathbf{H}$ is the Hessian at $\Theta^{(0)}$. In this view, the curvature along δ_i is quantified by the quadratic term $\delta_i^\top H \delta_i$. Its inverse $\delta_i^\top H^{-1} \delta_i$ suggests δ_i explores a novel region of the loss landscape, making it an indicator for selecting diverse checkpoints.

However, computing the full Hessian in high-dimensional parameter space is impractical, especially in TTA tasks. However, considering that 3D object detection models commonly use linear layers as final regression heads, we can effectively analyze curvature through the simpler and analytically tractable ridge regression setting. Specifically, assume a linear regression head parameterized by weights $w \in \mathbb{R}^d$ and a fixed feature extractor $\phi(\cdot)$, yielding a ridge regression objective of the form:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \|w^\top \phi(\mathbf{x}_i) - \mathbf{y}_i\|^2 + \lambda \|w\|^2, H_w = 2 \left(\frac{1}{K} \mathbf{Z}^\top \mathbf{Z} + \lambda I \right), \quad (9)$$

where H_w is Hessian matrix in parameter space. More precisely, this reveals the inverse of parameter-space curvature is linked to the proposed ridge leverage score under the low-rank surrogate $\hat{\mathbf{Z}}_{t-1}^\top \hat{\mathbf{Z}}_{t-1}$:

$$\mathbf{z}_i^\top H_w^{-1} \mathbf{z}_i = \mathbf{z}_i^\top \left(\frac{2}{K} \mathbf{Z}_{t-1}^\top \mathbf{Z}_{t-1} + 2\lambda I \right)^{-1} \mathbf{z}_i \propto s_i^{(t)}. \quad (10)$$

Empirical analysis (see Fig. 3) confirms that fingerprint vectors strongly correlate (Pearson correlation and Kendall Tau scores often exceeding 0.7) with parameter deltas, confirming that the geometry of fingerprint space reliably mirrors that of parameter space.

3.3 Model Merging

To perform stable model merging, we select top- K high-scoring checkpoints based on ridge leverage scores, yet their associated parameter directions may exhibit destructive interference. To resolve such conflicts, we adopt a sign-consistent merging inspired by [54], which aligns model parameters based on majority sign consensus before merging. Let $\{\Theta^{(i)}\}_{i=1}^K$ denote the top- K selected checkpoints and $\{s_i^{(t)}\}_{i=1}^K$ their corresponding leverage scores. For each parameter dimension j , we compute the majority sign $\text{sign}_{\text{maj}}(j) := \text{mode}(\{\text{sign}(\Theta_j^{(i)})\}_{i=1}^K)$, and zero out inconsistent components. The merged model is then given by:

$$\bar{\Theta}^{(t)} = \sum_{i=1}^K \tilde{s}_i^{(t)} \cdot \mathbb{I}[\text{sign}(\Theta^{(i)}) = \text{sign}_{\text{maj}}] \odot \Theta^{(i)}, \quad \tilde{s}_i^{(t)} = \frac{s_i^{(t)}}{\sum_{j=1}^K s_j^{(t)}}, \quad (11)$$

where $\odot$ denotes element-wise multiplication, and $\mathbb{I}[\cdot]$ is a binary mask that retains only parameters aligned with the majority sign. This sign-consistent merge ensures coherent parameter updates and stabilizes adaptation under distribution shifts.

Optimization. Following the protocol in [8], we use the merged model to generate pseudo-labeled bounding boxes for self-training the LiDAR-based detector online. In realistic end-to-end AD systems (see Fig. 1), perception, mapping, and planning modules are often integrated into a monolithic architecture. For efficiency, we freeze all components except for the 3D box regression head. Experiments show that CodeMerge not only improves detection performance but also yields gains in downstream mapping and planning *without* requiring additional training or modifications (Table 2).

4 Experiments

4.1 Experimental Setup

Datasets and Tasks. We conduct comprehensive experiments across five benchmarks for end-to-end autonomous driving and outdoor 3D object detection: **KITTI** [13], **KITTI-C** [25], **Waymo** [40], **nuScenes** [4], and **nuScenes-C** [53]. For test-time adaptation in end-to-end autonomous driving, we pre-train models on the nuScenes driving benchmark and adapt them to eight real-world corruptions in nuScenes-C: Motion Blur (Motion), Color Quantization (Quant), Low Light (Dark), Brightness (Bright), Snow, Fog, Camera Crash (Crash), and Frame Lost. For LiDAR-based 3D object detection, we first tackle cross-dataset adaptation (Waymo $\rightarrow$ KITTI, nuScenes $\rightarrow$ KITTI) following [56, 57, 6], addressing both object-level shifts (*e.g.*, scale and point density) and environmental differences (*e.g.*, deployment location, beam configuration). We then evaluate adaptation to sensor failures and weather effects via KITTI $\rightarrow$ KITTI-C, covering Fog, Wet Conditions (Wet.), Snow, Motion Blur (Moti.), Missing Beams (Beam.), Crosstalk (Cross.T), Incomplete Echoes (Inc.), and Cross-Sensor (Cross.S). The detailed evaluation metric and implementation details can be found in Appendix A.1.

Baselines. We compare the proposed CodeMerge against a broad range of methods: (i) **No Adapt.**, the pretrained model evaluated directly on the target datasets; (ii) **SN** [48], a *weakly supervised DA* technique that rescales source objects using target size statistics; (iii) **ST3D** [56], the first *UDA* method for 3D detection, employing multi-epoch self-training with pseudo labels; (iv) **Tent** [44], an *TTA* approach that minimizes prediction entropy; (v) **CoTTA** [46], which combines mean-teacher supervision with stochastic augmentations for *TTA*; (vi) **SAR** [34], enhancing Tent by sharpness-aware and reliability-aware entropy minimization; (vii) **MemCLR** [43], the first *online TTA* method that uses memory-augmented mean-teacher for 2D detection; (viii) **Reg-TTA3D** [58], which regularizes 3D box regression by enforcing noise-consistent pseudo labels during *3D TTA*; (ix) **MOS** [8], dynamically fusing a bank of top- K checkpoints through kernel-based synergy for *3D TTA*; (x) **DPO** [9], flattening the test-time loss landscape via dual perturbations for *3D TTA*; (xi) **Oracle**, a *fully supervised* model trained with annotated target datasets.

Table 1: **Perception and tracking results** of the end-to-end SparseDrive model [41] with and without TTA on the **nuScenes-C** [53] validation set under different corruptions at the highest severity level. The best results for each metric and corruption are highlighted in **bold**.

	CORRUPTION	METHOD	3D OBJECT DETECTION						MULTI-OBJECT TRACKING			
			mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓	AMOTA↑	AMOTP↓	Recall↑
IMAGE DEGRADATION	MOTION	No Adapt.	0.1468	0.3136	0.7792	0.2908	0.8048	0.4835	0.2398	0.0896	1.7983	0.1837
		Tent [44]	0.2462	0.4113	0.6802	0.2839	0.6039	0.3243	0.2264	0.1736	1.5122	0.2918
		MOS [8]	0.2611	0.4125	0.6848	0.2827	0.6588	0.3455	0.2087	0.1902	1.5239	0.3332
		Ours	0.2759	0.4206	0.6697	0.2815	0.6437	0.3618	0.2169	0.2192	1.5485	0.3456
	QUANT	No Adapt.	0.2022	0.3767	0.7095	0.2896	0.6478	0.3814	0.2160	0.1548	1.5398	0.2873
		Tent [44]	0.1424	0.3043	0.6527	0.4169	0.6032	0.5758	0.4200	0.0981	1.6930	0.1788
		MOS [8]	0.2560	0.4172	0.6781	0.2848	0.6115	0.3103	0.2231	0.2096	1.5195	0.3287
		Ours	0.2742	0.4331	0.6575	0.2764	0.5903	0.3018	0.2137	0.2339	1.4868	0.3330
ILLUMINATION SHIFT	DARK	No Adapt.	0.1386	0.2804	0.7375	0.4180	0.6880	0.6285	0.4164	0.1169	1.7520	0.1995
		Tent [44]	0.1266	0.2795	0.7243	0.4116	0.6396	0.6474	0.4151	0.0776	1.7014	0.1697
		MOS [8]	0.1726	0.3500	0.7482	0.2920	0.6570	0.4202	0.2459	0.1399	1.7148	0.2153
		Ours	0.2060	0.3727	0.7206	0.2852	0.6782	0.3993	0.2196	0.1762	1.6333	0.2557
	BRIGHT	No Adapt.	0.3300	0.4641	0.6355	0.2749	0.6084	0.3013	0.1892	0.2829	1.4257	0.3982
		Tent [44]	0.2557	0.4289	0.6345	0.2896	0.5666	0.3143	0.1848	0.1879	1.4836	0.3002
		MOS [8]	0.3595	0.4825	0.6100	0.2757	0.6053	0.2908	0.1909	0.3126	1.3566	0.4387
		Ours	0.3692	0.4939	0.6138	0.2779	0.5343	0.2885	0.1928	0.3317	1.3389	0.4632
ADVERSE WEATHER	SNOW	No Adapt.	0.0970	0.2206	0.7974	0.4586	0.9349	0.6614	0.4264	0.0469	1.8822	0.1070
		Tent [44]	0.1417	0.2791	0.7312	0.4165	0.6904	0.6714	0.4077	0.0779	1.7440	0.1838
		MOS [8]	0.1478	0.3207	0.7740	0.2995	0.7092	0.5211	0.2284	0.0887	1.7828	0.1747
		Ours	0.1828	0.3581	0.7558	0.2930	0.6009	0.4604	0.2222	0.1136	1.7119	0.2293
	FOG	No Adapt.	0.3162	0.4612	0.6295	0.2775	0.5727	0.2984	0.1910	0.2756	1.4469	0.3859
		Tent [44]	0.2964	0.4515	0.6372	0.2837	0.5190	0.3149	0.2121	0.2312	1.4311	0.3623
		MOS [8]	0.3362	0.4690	0.6339	0.2797	0.5798	0.2961	0.2019	0.2907	1.3833	0.4007
		Ours	0.3421	0.4761	0.6184	0.2739	0.5597	0.2995	0.1981	0.2997	1.3749	0.4124
SENSOR FAILURES	CRASH	No Adapt.	0.0785	0.2753	0.6467	0.4060	0.6078	0.5953	0.3840	0.0670	1.8241	0.1519
		Tent [44]	0.0722	0.2679	0.7426	0.3469	0.6294	0.6658	0.2976	0.0462	1.9007	0.1155
		MOS [8]	0.0702	0.2659	0.7614	0.3460	0.6169	0.6685	0.2990	0.0454	1.8978	0.1155
		Ours	0.0973	0.3288	0.6979	0.2889	0.6061	0.4175	0.1876	0.0810	1.8372	0.1550
	LOST	No Adapt.	0.0886	0.3109	0.7314	0.2792	0.6206	0.4717	0.2310	0.0549	1.7638	0.1644
		Tent [44]	0.0372	0.2371	0.8386	0.2913	0.7439	0.7068	0.2337	0.0029	1.9856	0.0406
		MOS [8]	0.0479	0.2116	0.8913	0.3464	0.7567	0.8008	0.3281	0.0131	1.9670	0.0624
		Ours	0.1172	0.3292	0.7638	0.2787	0.5810	0.4461	0.2243	0.0700	1.7605	0.1788
AVERAGE	No Adapt.	0.1747	0.3378	0.7083	0.3368	0.6856	0.4777	0.2867	0.1361	1.6791	0.2347	
	Tent [44]	0.1648	0.3325	0.7052	0.3426	0.6245	0.5276	0.2997	0.1119	1.6815	0.2053	
	MOS [8]	0.2028	0.3551	0.7269	0.3205	0.6633	0.4829	0.2711	0.1599	1.6461	0.2532	
	Ours	0.2334	0.4016	0.6872	0.2819	0.5993	0.3719	0.2094	0.1907	1.5865	0.2966	

4.2 Main Results and Analysis

TTA on End-to-End Autonomous Driving. We comprehensively evaluate our CodeMerge method on nuScenes-C [53] with the end-to-end SparseDrive model [41], covering five downstream tasks: 3D detection, multi-object tracking, online mapping, motion prediction, and trajectory planning under diverse corruptions. Table 1 shows CodeMerge consistently outperforms all baselines, including No Adapt, Tent, and the state-of-the-art MOS [8] in averaged results. In 3D detection, we boost mAP by **33.6%** over no adaptation (0.1747 $\rightarrow$ 0.2334) and by 13.3% over MOS. CodeMerge also reduces mASE by 4.4% relative to MOS, and lowers mAVE by 19%. Under the *Bright* corruption, CodeMerge improves mAP by 11.9% over no adaptation, with consistent gains in other metrics. In multi-object tracking, CodeMerge improves AMOTA by 19.3%, reduces AMOTP by 13.8%, and raises recall by 16.5% when compared with the SOTA baseline, MOS. Notably, under the most safety-critical *Lost* scenario, the proposed method achieves the highest recall (0.1788) and lowest tracking error among all methods. Although only perception weights are adapted, downstream tasks benefit markedly. As reported in Table 2. CodeMerge increases online mapping mAP by **42.3%** (0.2009 $\rightarrow$ 0.2859) over no adaptation, with **+45.7%** on lane boundaries and **+39.5%** on obstacles, especially **+94.2%** under *Dark*. For motion prediction, mADE and mFDE fall by 9.3% and 9.7% compared to no adaptation, respectively, while EPA (higher is better) rises by 13.8%. For planning, average lateral deviation falls 8.3% (0.7923 m $\rightarrow$ 0.7266 m) and collision risk drops 6.1% compared to no adaptation. These consistent gains achieved without touching non-perception modules, confirm

Table 2: **Impact of TTA on downstream modules of end-to-end SparseDrive [41].** We evaluate online mapping, motion prediction, and trajectory planning on the **nuScenes-C** [53] under the highest severity of various corruptions. These modules are not fine-tuned; all performance gains stem from TTA applied to the detection module. Best results per metric and corruption are shown in **bold**.

	CORRUPTION	METHOD	ONLINE MAPPING				MOTION PREDICTION				PLANNING	
			AP _{ped} ↑	AP _d ↑	AP _b ↑	mAP↑	mADE↓	mFDE↓	MR↓	EPA↑	L2-Avg↓	CR-Avg↓
IMAGE DEGRADATION	MOTION	No Adapt.	0.1988	0.2343	0.1999	0.2110	0.8630	1.3483	0.1750	0.2616	0.7877	0.2150
		Tent [44]	0.3425	0.3794	0.3876	0.3698	0.7786	1.1825	0.1520	0.3712	0.6474	0.0900
		MOS [8]	0.3452	0.3943	0.4012	0.3802	0.7348	1.1278	0.1560	0.3742	0.6694	0.1340
		Ours	0.3660	0.4212	0.4283	0.4052	0.7264	1.1200	0.1570	0.3945	0.6580	0.1100
	QUANT	No Adapt.	0.1742	0.2317	0.2069	0.2043	0.7620	1.1734	0.1526	0.3204	0.7301	0.1590
		Tent [44]	0.1526	0.2153	0.2088	0.1922	0.8489	1.3551	0.1602	0.2987	0.6966	0.1200
		MOS [8]	0.2346	0.3208	0.2918	0.2824	0.7040	1.0822	0.1445	0.3668	0.6848	0.1180
		Ours	0.2600	0.3445	0.3267	0.3104	0.7002	1.0859	0.1454	0.3840	0.6762	0.1250
ILLUMINATION SHIFT	DARK	No Adapt.	0.1173	0.2038	0.1812	0.1675	0.8428	1.3255	0.1714	0.2757	0.7535	0.2760
		Tent [44]	0.2116	0.2560	0.2481	0.2386	0.8603	1.3314	0.1786	0.2722	0.7049	0.1230
		MOS [8]	0.2261	0.3090	0.2892	0.2748	0.7956	1.2443	0.1730	0.3066	0.6824	0.1360
		Ours	0.2825	0.3637	0.3291	0.3251	0.7493	1.1639	0.1644	0.3397	0.6602	0.1170
	BRIGHT	No Adapt.	0.3777	0.4847	0.4833	0.4486	0.6646	1.0246	0.1369	0.4468	0.6306	0.1260
		Tent [44]	0.3550	0.4342	0.4591	0.4161	0.6882	1.0739	0.1369	0.3978	0.6487	0.0950
		MOS [8]	0.4053	0.4960	0.5127	0.4713	0.6468	1.0031	0.1357	0.4593	0.6243	0.1230
		Ours	0.4305	0.5224	0.5398	0.4976	0.6504	1.0122	0.1392	0.4680	0.6209	0.0940
ADVERSE WEATHER	SNOW	No Adapt.	0.0061	0.0322	0.0369	0.0250	1.0643	1.7042	0.1930	0.2113	0.8897	0.4310
		Tent [44]	0.1083	0.1320	0.1359	0.1254	0.9147	1.4192	0.1753	0.2804	0.7552	0.1320
		MOS [8]	0.1237	0.1564	0.1545	0.1448	0.8736	1.3476	0.1737	0.2994	0.7684	0.1920
		Ours	0.1134	0.1812	0.1740	0.1562	0.8074	1.2589	0.1717	0.3135	0.7634	0.1900
	FOG	No Adapt.	0.3600	0.4649	0.4076	0.4109	0.6482	0.9904	0.1347	0.4380	0.6257	0.1050
		Tent [44]	0.3786	0.4492	0.4438	0.4239	0.6861	1.0631	0.1405	0.4182	0.6533	0.0870
		MOS [8]	0.4161	0.4950	0.4785	0.4632	0.6549	1.0087	0.1401	0.4539	0.6225	0.1060
		Ours	0.4276	0.5022	0.4843	0.4714	0.6501	1.0008	0.1394	0.4557	0.6200	0.1100
SENSOR FAILURES	CRASH	No Adapt.	0.1029	0.1019	0.0618	0.0889	0.8662	1.3375	0.1652	0.1920	0.9276	0.3740
		Tent [44]	0.0431	0.0764	0.0141	0.0445	0.8691	1.3548	0.1710	0.1771	0.8852	0.7040
		MOS [8]	0.0394	0.0706	0.0100	0.0400	0.8878	1.3895	0.1766	0.1721	0.8977	0.7300
		Ours	0.0727	0.1154	0.0279	0.0720	0.8302	1.3022	0.1637	0.1974	0.8539	0.6300
	LOST	No Adapt.	0.0892	0.0388	0.0250	0.0510	1.0327	1.4772	0.1740	0.1826	0.9932	0.4830
		Tent [44]	0.0431	0.0547	0.0163	0.0380	1.4194	2.1114	0.2383	0.0737	0.9985	0.6220
		MOS [8]	0.0180	0.0153	0.0038	0.0124	1.5468	2.3163	0.2155	0.0873	1.0628	0.7340
		Ours	0.0723	0.0503	0.0250	0.0492	1.0004	1.4304	0.1739	0.0952	0.9600	0.6610
	AVERAGE	No Adapt.	0.1783	0.2240	0.2003	0.2009	0.8430	1.2976	0.1629	0.2911	0.7923	0.2711
		Tent [44]	0.2044	0.2497	0.2392	0.2311	0.8832	1.3614	0.1691	0.2862	0.7487	0.2466
		MOS	0.2260	0.2822	0.2677	0.2586	0.8555	1.3149	0.1644	0.3150	0.7515	0.2841
		Ours	0.2531	0.3126	0.2919	0.2859	0.7643	1.1718	0.1568	0.3312	0.7266	0.2546

that the proposed lightweight, fingerprint-guided merging framework stabilizes the detector and unlocks robust performance across all autonomous driving tasks.

TTA on LiDAR-based Detection. We examine CodeMerge’s performance in 3D object detection across two distinct types of domain shifts: Cross-dataset (Waymo → KITTI, nuScenes → KITTI) and Corruption-induced shifts (KITTI → KITTI-C). (1) **Cross-dataset** (Table 3). Compared with the non-adapted model, CodeMerge lifts AP_{BEV} by 25.1% and AP_{3D} by 141% on Waymo → KITTI, closing 108.5%/84.5% of the domain gap and even surpassing the multi-epoch

Table 4: **TTA results on KITTI-C.** We evaluate the LiDAR-based SECOND detector [55] under the highest severity level of various corruptions, reporting AP_{3D} (hard).

	No Adapt.	Tent [44]	CoTTA [46]	SAR [34]	MemCLR [43]	DPO [9]	MOS [8]	Ours
Fog	68.23	68.73	68.49	68.14	68.23	68.72	69.11	75.96
Snow	59.07	59.50	59.45	58.78	58.74	60.80	62.72	63.53
Inc.	25.68	26.44	27.85	26.42	27.47	27.16	34.53	32.18
CrossT.	75.49	74.67	72.22	74.51	74.25	75.52	75.47	75.76
Moti.	38.21	38.15	38.62	38.12	37.57	38.71	40.59	44.87
CrossS.	41.08	41.17	40.80	40.63	40.90	42.09	43.68	42.36
Wet.	76.25	76.36	76.43	76.23	76.25	76.89	77.79	79.82
Beam.	53.93	53.85	53.98	53.75	53.49	54.06	55.91	57.26
Mean	54.74	54.86	54.73	54.57	54.61	55.49	57.48	58.97

ST3D and fully supervised Oracle in AP_{BEV}. On nuScenes → KITTI, it narrows the gap by 81.3%/73.15%, again outperforming the strongest TTA baselines (MOS, DPO) and exceeding

Table 3: **TTA results for LiDAR-based 3D detection across different datasets.** We report AP_{BEV} / AP_{3D} (moderate). “Oracle” = fully-supervised on target; **Bold** = best; underline = second best.

METHOD	VENUE	TTA	WAYMO → KITTI		NUSCENES → KITTI	
			AP_{BEV} / AP_{3D}	Closed Gap	AP_{BEV} / AP_{3D}	Closed Gap
No Adapt.	–		67.64 / 27.48	–	51.84 / 17.92	–
SN [48]	CVPR’20	×	78.96 / 59.20	+72.33% / +69.00%	40.03 / 21.23	+37.55% / +5.96%
ST3D [56]	CVPR’21		82.19 / 61.83	+92.97% / +74.72%	75.94 / 54.13	+76.63% / +65.21%
Oracle	–		83.29 / 73.45	–	83.29 / 73.45	–
Tent [44]	ICLR’21		65.09 / 30.12	–16.29% / +5.74%	46.90 / 18.83	–15.71% / +1.64%
CoTTA [46]	CVPR’22		67.46 / 35.34	–1.15% / +17.10%	68.81 / 47.61	+53.96% / +53.47%
SAR [34]	ICLR’23		65.81 / 30.39	–11.69% / +6.33%	61.34 / 35.74	+30.21% / +32.09%
MemCLR [43]	WACV’23	✓	65.61 / 29.83	–12.97% / +5.11%	61.47 / 35.76	+30.62% / +32.13%
DPO [9]	MM’24		75.81 / 55.74	+52.20% / +61.47%	<u>73.27</u> / <u>54.38</u>	<u>+68.13%</u> / <u>+65.66%</u>
Reg-TTA3D [58]	ECCV’24		81.60 / 56.03	+89.20% / +62.11%	68.73 / 44.56	+53.70% / +47.97%
MOS [8]	ICLR’25		<u>81.90</u> / <u>64.16</u>	<u>+91.12%</u> / <u>+79.79%</u>	71.13 / 51.11	+61.33% / +59.78%
Ours	–		84.62 / 66.31	+108.50% / +84.47%	77.41 / 58.54	+81.30% / +73.15%

Table 5: Ablation study on different checkpoint selection strategies, number of checkpoints to merge (K), and random projection dimension (d') on **nuScenes-C** [53] (motion blur at the heaviest level).

MERGE	K	PROJ.-D	DETECTION		TRACKING		MAPPING		MOTION		PLANNING	
			mAP↑	NDS↑	AMOTA↑	AMOTP↓	mAP↑	AP_{ped} ↑	mADE↓	mFDE↓	L2-Avg↓	CR-Avg↓
Random	5	–	0.2740	0.4185	0.2152	1.5461	0.4011	0.3678	0.7251	1.1192	0.6631	0.1120
Recent	5	–	0.2480	0.3985	0.1866	1.6040	0.3748	0.3410	0.7368	1.1436	0.6795	0.1490
EMA	–	–	0.2478	0.3992	0.1869	1.6016	0.3748	0.3413	0.7375	1.1447	0.6778	0.1250
KMeans++	5	1024	0.2746	0.4192	0.2157	1.5490	0.4010	0.3678	0.7246	1.1182	0.6625	0.1050
Leverage	5	1024	0.2851	0.4264	0.2241	1.5206	0.4103	0.3713	0.7228	1.1146	0.6504	0.1090
Leverage	3	1024	0.2655	0.4122	0.2077	1.5630	0.3623	0.3928	0.7407	1.1461	0.6651	0.1200
Leverage	9	1024	0.2818	0.4231	0.2195	1.5240	0.4167	0.3814	0.7180	1.1066	0.6534	0.1030
Leverage	5	256	0.2749	0.4176	0.2168	1.5488	0.4010	0.3678	0.7228	1.1142	0.6615	0.0960
Leverage	5	512	0.2708	0.4142	0.2117	1.5525	0.3991	0.3695	0.7378	1.1428	0.6588	0.1170
Leverage	5	2048	0.2799	0.4207	0.2140	1.5224	0.4033	0.3630	0.7324	1.1204	0.6488	0.0950

ST3D by +1.9% AP_{BEV} and +8.1% AP_{3D} . **(2) Corruption-induced** (Table 4). Against KITTI → KITTI-C corruptions, CodeMerge raises mean AP_{3D} by +7.7% over no adaptation and +2.6% over the best prior TTA baseline. Under *Fog* and *Wet* corruption, gains are pronounced: +9.9% (75.96 vs. 69.11) and +2.6% (79.82 vs. 77.79), respectively, indicating enhanced resilience to visibility and environment degradations. These results demonstrate that our latent-space, fingerprint-guided merging not only closes cross-domain gaps more effectively than existing TTA methods but also surpasses dedicated domain adaptation approaches, providing robust performance across diverse and challenging environments.

4.3 Ablation and Sensitivity Study

Impact of Checkpoint Selection Strategy. In Table 5, we compare five strategies for choosing $K = 5$ checkpoints under heavy Motion Blur: Random sampling, Recent (the latest five), KMeans++ clustering in feature space, Exponential Moving Average (EMA) model, and our Leverage-score ranking. Random yields a reduced detection mAP of 0.2740, weaker tracking (AMOTA = 0.2152), and planning (CR-Avg = 0.1120). Recent performs worst across all tasks (mAP 0.2480, AMOTA 0.1866, CR-Avg 0.1490), indicating catastrophic forgetting when only the newest checkpoints are merged. KMeans++ yields a marginal 0.17% lift in NDS over Random and reduces collision risk by 6.3%, reflecting its ability to capture diverse feature modes. However, KMeans++ is still outperformed by the proposed method (–3.8% mAP for detection), highlighting that pure feature clustering cannot match the important informativeness captured by leverage-score ranking. Furthermore, CodeMerge ($K = 5$, projection dimension $d' = 1024$) consistently outperforms EMA across all tasks: Detection mAP/NDS +0.0373/+0.0272; Tracking AMOTA +0.0372 and AMOTP –0.0810 (↓ better); Mapping mAP/ AP_{ped} +0.0355/+0.0300; Motion mADE/mFDE –0.0147/–0.0301; Planning L2-Avg/CR-Avg –0.0274/–0.0160. These gains indicate that our fingerprint-guided merging, which selects complementary checkpoints, offers stronger generalization than the uniform, time-local smoothing performed by EMA. Overall, the proposed Leverage-score selection consistently achieves the best results by explicitly identifying the most informative, complementary checkpoints carrying long-term knowledge.

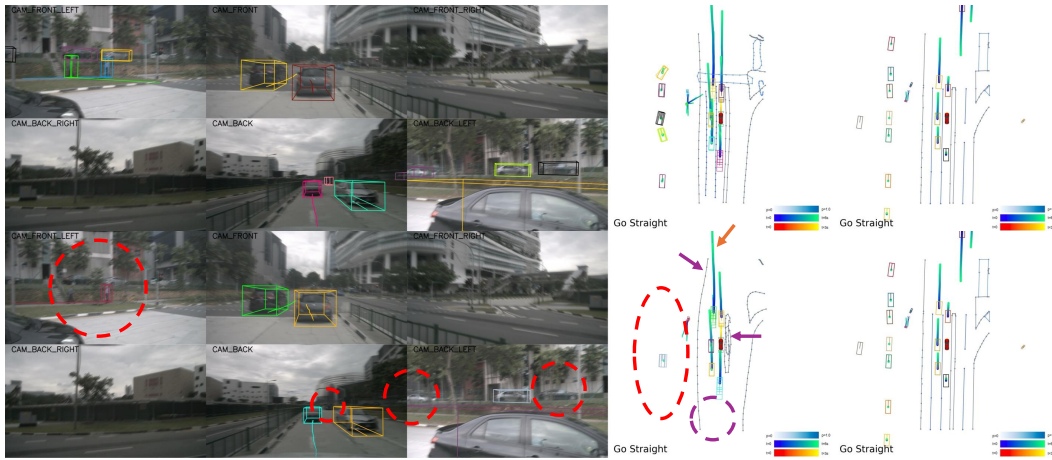


Figure 4: **Visualization of outputs of SparseDrive (bottom) and after CodeMerge adaptation (upper) under severe motion blur.** TTA greatly improves detection by capturing more true positive instances, which consequently enhances downstream mapping and planning accuracy (right).

Impact on Number of Merged Checkpoints. Table 5 compares selecting $K = 3, 5$, or 9 checkpoints (with $d' = 1024$) for model merging under Motion Blur corruption. With only $K = 3$, detection mAP drops from 0.2851 to 0.2655, and tracking AMOTA falls from 0.2241 to 0.2077, indicating insufficient coverage of knowledge diversity. Increasing to $K = 9$ recovers much of this gap (mAP 0.2818, AMOTA 0.2195) but yields only marginal gains, in mapping mAP (0.4167 vs. 0.4103 at $K = 5$). The near-parity between $K = 5$ and 9 suggests redundant information beyond five checkpoints. Balancing performance and memory fingerprint, we thus adopt $K = 5$ in all experiments.

Impact on Dimension of Random Projection. We additionally examine the effect of varying the random projection dimension d' among $\{256, 512, 1024, 2048\}$. As Table 5 shows, at $d = 256$, the performance is only slightly below that of $d' = 1024$ (mAP 0.2749 vs. 0.2851; NDS 0.4176 vs. 0.4264), demonstrating that very compact fingerprints still capture most of the critical variability. At $d' = 2048$, results nearly match the $d = 1024$ but at twice the memory cost. Therefore, $d' = 1024$ offers the best trade-off between performance and fingerprint.

Efficiency Analysis. We further compare the GPU memory and total TTA runtime of *CodeMerge* against MOS on both the SECOND and SparseDrive detectors. As reported in Table 6, with SECOND, MOS consumes 17.4 GiB and requires 1,813 seconds per adaptation run, whereas CodeMerge uses only 16.0 GiB (**-8.0%**) and completes in 1,054 seconds (**-41.8%**). The savings are even more pronounced on SparseDrive: MOS demands 39.4 GiB and 37,619 seconds, while CodeMerge needs just 29.8 GiB (**-21.0%**) and 27,359 seconds (**-37.48%**). These gains arise from our fingerprint-guided merging, which projects each checkpoint into a compact embedding and computes leverage weights on the fly (requiring only one extra forward pass), rather than loading and forwarding K full models as MOS does. This design drastically reduces memory footprint and latency, making CodeMerge well-suited for real-time autonomous driving applications.

Table 6: GPU memory (MiB) and total TTA runtime (s) for a single TTA run.

Model	Method	GPU Memory	Runtime
SECOND	MOS	17,411	1,813
	Ours	16,041	1,054
SparseDrive	MOS	39,420	37,619
	Ours	29,868	27,359

Quantitative Analysis. We visualize predictions with CodeMerge (top row) against the non-adapted SparseDrive baseline (bottom row) in Fig. 4 to illustrate how on-the-fly merging enhances every stage of the end-to-end pipeline. In detection and tracking, CodeMerge produces tight, correctly aligned 3D boxes, but the baseline suffers a large number of missed or misplaced detections (highlighted in red dashed circles). In mapping, our method reconstructs dense, straight-lane boundaries and curb lines, validating its ability to preserve semantic consistency. In contrast, the baseline yields sparse, crooked lanes and missing curbs (highlighted in purple circles/arrows), degrading map fidelity. Finally, CodeMerge’s planned trajectory remains centered in the lane and safely avoids dynamic objects, while the baseline’s path drifts toward the curb (highlighted in an orange arrow) and even intersects an oncoming track, demonstrating unsafe behavior. In summary, these qualitative results confirm that

leveraging compact fingerprints and leverage-score-guided merging yields better detections, more robust tracking, and safer trajectories under severe real-world corruptions.

5 Conclusion

In this work, we address the challenge of online adaptation to domain shifts for both LiDAR-based and vision-centric end-to-end AD detection under extreme conditions. Our proposed CodeMerge framework effectively mitigates cross-dataset and corruption-induced distribution shifts, while reducing GPU memory consumption and inference latency by approximately 27% compared to state-of-the-art TTA methods. Notably, other downstream modules, such as mapping and planning, receive performance improvements without task-specific fine-tuning due to enhanced detection outputs. However, this study represents an early attempt to address robustness in end-to-end AD, and major experiments have been primarily conducted on the SparseDrive architecture. The primary bottleneck remains that popular architectures, such as UniAD and VAD, experience over tenfold performance degradation on nuScenes-C, hindering effective adaptation training. Future work will investigate strategies to further accelerate adaptation and enhance robustness under dynamic driving conditions.

6 Acknowledgements

This work was partially supported by ARC DE240100105, DP240101814, DP230101196, DP230101753, BA24006, DE250100363, IH230100013.

References

- [1] Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha S. Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *Proc. International Conference on Learning Representations (ICLR)*, 2023.
- [2] Devansh Arpit, Huan Wang, Yingbo Zhou, and Caiming Xiong. Ensemble of averages: Improving model selection and boosting performance in domain generalization. In *Proc. Annual Conference on Neural Information Processing (NeurIPS)*, 2022.
- [3] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, June 21-24, 2010, Haifa, Israel.
- [4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020.
- [5] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 295–305, 2022.
- [6] Zhuoxiao Chen, Yadan Luo, Zheng Wang, Mahsa Baktashmotlagh, and Zi Huang. Revisiting domain-adaptive 3d object detection by reliable, diverse and class-balanced pseudo-labeling. In *Proc. International Conference on Computer Vision (ICCV)*, pages 3691–3703, 2023.
- [7] Zhuoxiao Chen, Yadan Luo, Zixin Wang, Zijian Wang, Xin Yu, and Zi Huang. Towards open world active learning for 3d object detection. *CoRR*, abs/2310.10391, 2023.
- [8] Zhuoxiao Chen, Junjie Meng, Mahsa Baktashmotlagh, Yonggang Zhang, Zi Huang, and Yadan Luo. MOS: model synergy for test-time adaptation on lidar-based 3d object detection. In *Proc. International Conference on Learning Representations (ICLR)*, 2025.
- [9] Zhuoxiao Chen, Zixin Wang, Yadan Luo, Sen Wang, and Zi Huang. DPO: dual-perturbation optimization for test-time adaptation in 3d object detection. In *Proc. ACM International Conference on Multimedia (MM)*, pages 4138–4147, 2024.
- [10] Nico Daheim, Thomas Möllenhoff, Edoardo M. Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. In *Proc. International Conference on Learning Representations (ICLR)*, 2024.

- [11] Marius-Constantin Dinu, Markus Holzleitner, Maximilian Beck, Hoan Duc Nguyen, Andrea Huber, Hamid Eghbal-zadeh, Bernhard Alois Moser, Sergei V. Pereverzyev, Sepp Hochreiter, and Werner Zellinger. Addressing parameter choice issues in unsupervised domain adaptation by aggregation. In *Proc. International Conference on Learning Representations (ICLR)*, 2023.
- [12] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proc. International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 3259–3269, 2020.
- [13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.
- [14] Taesik Gong, Yewon Kim, Taekyung Lee, Sorn Chottananurak, and Sung-Ju Lee. Sotta: Robust test-time adaptation on noisy data streams. In *Proc. Annual Conference on Neural Information Processing (NeurIPS)*, 2024.
- [15] Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and J Zico Kolter. Test time adaptation via conjugate pseudo-labels. In *Proc. Annual Conference on Neural Information Processing (NeurIPS)*, 2022.
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*.
- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [18] Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *Proc. International Conference on Learning Representations (ICLR)*, 2023.
- [19] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. VAD: vectorized scene representation for efficient autonomous driving. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 8306–8316. IEEE, 2023.
- [20] Jincen Jiang, Qianyu Zhou, Yuhang Li, Xinkui Zhao, Meili Wang, Lizhuang Ma, Jian Chang, Jian Jun Zhang, and Xuequan Lu. Pcotta: Continual test-time adaptation for multi-task point cloud understanding. In *Proc. Annual Conference on Neural Information Processing (NeurIPS)*, 2024.
- [21] Ian T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 1986.
- [22] Sanghun Jung, Jungsoo Lee, Nanhee Kim, Amirreza Shaban, Byron Boots, and Jaegul Choo. Cafa: Class-aware feature alignment for test-time adaptation. In *Proc. International Conference on Computer Vision (ICCV)*, pages 19060–19071, 2023.
- [23] Adilbek Karmanov, Dayan Guan, Shijian Lu, Abdulmotaleb El-Saddik, and Eric P. Xing. Efficient test-time adaptation of vision-language models. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14162–14171, 2024.
- [24] Byungjai Kim, Chanho Ahn, Wissam J. Baddar, Kikyung Kim, Huijin Lee, Saehyun Ahn, Seungju Han, Sungjoo Suh, and Eunho Yang. Test-time ensemble via linear mode connectivity: A path to better adaptation. In *Proc. International Conference on Learning Representations (ICLR)*, 2025.
- [25] Lingdong Kong, Youquan Liu, Xin Li, Runnan Chen, Wenwei Zhang, Jiawei Ren, Liang Pan, Kai Chen, and Ziwei Liu. Robo3d: Towards robust and reliable 3d perception against corruptions. In *Proc. International Conference on Computer Vision (ICCV)*, pages 19937–19949, 2023.

- [26] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64, 2025.
- [27] Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, and Xinggang Wang. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*. Computer Vision Foundation / IEEE, 2025.
- [28] Yadan Luo, Zhuoxiao Chen, Zhen Fang, Zheng Zhang, Mahsa Baktashmotlagh, and Zi Huang. Kecor: Kernel coding rate maximization for active 3d object detection. In *Proc. International Conference on Computer Vision (ICCV)*, pages 18233–18244, 2023.
- [29] Yadan Luo, Zhuoxiao Chen, Zijian Wang, Xin Yu, Zi Huang, and Mahsa Baktashmotlagh. Exploring active 3d object detection from a generalization perspective. In *Proc. International Conference on Learning Representations (ICLR)*, 2023.
- [30] Yadan Luo, Zijian Wang, Zhuoxiao Chen, Zi Huang, and Mahsa Baktashmotlagh. Source-free progressive graph learning for open-set domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):11240–11255, 2023.
- [31] Muhammad Jehanzeb Mirza, Jakub Micorek, Horst Possegger, and Horst Bischof. The norm must go on: Dynamic unsupervised domain adaptation by normalization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14745–14755. IEEE, 2022.
- [32] Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, Pang Wei Koh, Jesse Dodge, and Pradeep Dasigi. Merge to learn: Efficiently adding skills to language models with model merging. In *Findings of the Association for Computational Linguistics (Findings of EMNLP)*, pages 15604–15621, 2024.
- [33] Shuaicheng Niu, Chunyan Miao, Guohao Chen, Pengcheng Wu, and Peilin Zhao. Test-time model adaptation with only forward passes. In *Proc. International Conference on Machine Learning (ICML)*, 2024.
- [34] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiqian Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *Proc. International Conference on Learning Representations (ICLR)*, 2023.
- [35] Haoxuan Qu, Xiaofei Hui, Yujun Cai, and Jun Liu. LMC: large model collaboration with cross-assessment for training-free open-set object recognition. In *Proc. Annual Conference on Neural Information Processing (NeurIPS)*, 2023.
- [36] Alexandre Ramé, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. In *Proc. Annual Conference on Neural Information Processing (NeurIPS)*, 2022.
- [37] Dominik Scherer, Andreas C. Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks - ICANN 2010 - 20th International Conference, Thessaloniki, Greece, September 15-18, 2010, Proceedings, Part III*, 2010.
- [38] Hajin Shim, Changhun Kim, and Eunho Yang. Cloudfixer: Test-time adaptation for 3d point clouds via diffusion-guided geometric transformation. In *Proc. European Conference on Computer Vision (ECCV)*, 2024.
- [39] Ziyang Song, Caiyan Jia, Lin Liu, Hongyu Pan, Yongchang Zhang, Junming Wang, Xingyu Zhang, Shaoqing Xu, Lei Yang, and Yadan Luo. Don’t shake the wheel: Momentum-aware planning in end-to-end autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*. Computer Vision Foundation / IEEE, 2025.

- [40] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2443–2451, 2020.
- [41] Wenchao Sun, Xuewu Lin, Yining Shi, Chuang Zhang, Haoran Wu, and Sifa Zheng. Sparsedrive: End-to-end autonomous driving via sparse scene representation. In *Proc. International Conference on Robotics and Automation (ICRA)*, 2025.
- [42] Yun-Yun Tsai, Fu-Chen Chen, Albert Y. C. Chen, Junfeng Yang, Che-Chun Su, Min Sun, and Cheng-Hao Kuo. GDA: generalized diffusion for robust test-time adaptation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23242–23251, 2024.
- [43] Vibashan VS, Poojan Oza, and Vishal M. Patel. Towards online domain adaptive object detection. In *Proc. Winter Conference on Applications of Computer Vision (WACV)*, pages 478–488, 2023.
- [44] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *Proc. International Conference on Learning Representations (ICLR)*, 2021.
- [45] Haiyang Wang, Chen Shi, Shaoshuai Shi, Meng Lei, Sen Wang, Di He, Bernt Schiele, and Liwei Wang. DSVT: dynamic sparse voxel transformer with rotated sets. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 13520–13529. IEEE, 2023.
- [46] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7191–7201, 2022.
- [47] Shuai Wang, Daoan Zhang, Zipei Yan, Jianguo Zhang, and Rui Li. Feature alignment and uniformity for test time adaptation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20050–20060, 2023.
- [48] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark E. Campbell, Kilian Q. Weinberger, and Wei-Lun Chao. Train in germany, test in the USA: making 3d object detectors generalize. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11710–11720, 2020.
- [49] Yanshuo Wang, Ali Cheraghian, Zeeshan Hayder, Jie Hong, Sameera Ramasinghe, Shafin Rahman, David Ahmmedt-Aristizabal, Xuesong Li, Lars Petersson, and Mehrtash Harandi. Backpropagation-free network for 3d test-time adaptation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [50] Zixin Wang, Yadan Luo, Liang Zheng, Zhuoxiao Chen, Sen Wang, and Zi Huang. In search of lost online test-time adaptation: A survey. *International Journal of Computer Vision*, 133(3):1106–1139, 2025.
- [51] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proc. International Conference on Machine Learning (ICML)*, volume 162, pages 23965–23998, 2022.
- [52] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7949–7961, 2022.
- [53] Shaoyuan Xie, Lingdong Kong, Wenwei Zhang, Jiawei Ren, Liang Pan, Kai Chen, and Ziwei Liu. Benchmarking and improving bird’s eye view perception robustness in autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(5):3878–3894, 2025.

- [54] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.
- [55] Yan Yan, Yuxing Mao, and Bo Li. SECOND: sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.
- [56] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. ST3D: self-training for unsupervised domain adaptation on 3d object detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10368–10378, 2021.
- [57] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d++: denoised self-training for unsupervised domain adaptation on 3d object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.
- [58] Jiakang Yuan, Bo Zhang, Kaixiong Gong, Xiangyu Yue, Botian Shi, Yu Qiao, and Tao Chen. Reg-tta3d: Better regression makes better test-time adaptive 3d object detection. In *Proc. European Conference on Computer Vision (ECCV)*, volume 15101, pages 197–213, 2024.
- [59] Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15922–15932, 2023.
- [60] Longbin Zeng, Jiayi Han, Liang Du, and Weiyang Ding. Rethinking precision of pseudo label: Test-time adaptation via complementary learning. *Pattern Recognition Letters*, 177:96–102, 2024.
- [61] Shuangfei Zhai, Hui Wu, Abhishek Kumar, Yu Cheng, Yongxi Lu, Zhongfei Zhang, and Rogério Schmidt Feris. S3pool: Pooling with stochastic spatial sampling. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017.
- [62] Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: test time robustness via adaptation and augmentation. In *Proc. Annual Conference on Neural Information Processing (NeurIPS)*, 2022.
- [63] Bowen Zhao, Chen Chen, and Shu-Tao Xia. Delta: degradation-free fully test-time adaptation. In *Proc. International Conference on Learning Representations (ICLR)*, 2023.
- [64] Tianpei Zou, Sanqing Qu, Zhijun Li, Alois Knoll, Lianghua He, Guang Chen, and Changjun Jiang. HGL: hierarchical geometry learning for test-time adaptation in 3d point cloud segmentation. In *Proc. European Conference on Computer Vision (ECCV)*, 2024.

A Technical Appendices and Supplementary Material

We include additional technical details in the following appendices:

- Section A.1 (Implementation Details): Describes the full experimental setup, including training schedules and hyperparameter configurations.
- Section A.2 (Evaluation Metrics): Provides definitions and explanations for all evaluation metrics used across detection, tracking, mapping, and planning tasks.
- Section A.3 (Additional Visualizations): Presents qualitative results and visual comparisons illustrating the adaptation performance of the end-to-end AD system under various distribution shifts.
- Section A.4 (Related Work): Summarizes the relevant literature in TTA and model merging.
- Section A.5 (Runtime): Reports the computational cost and per-frame latency of each TTA step.
- Section A.6 (Random Projection): Evaluates the determinism and robustness of our random projection strategy.
- Section A.7 (Fingerprint Generation): Demonstrates that our fingerprint generation strategy significantly outperforms alternative methods.
- Section A.8 (3D Applicability Analysis): Assesses CodeMerge across other detectors beyond SECOND, confirming its strong model-agnostic applicability.
- Section A.9 (3D Robustness Analysis): Analyses the robustness of CodeMerge in heavy distribution shift occurs (e.g., Fog & LiDAR dropout).
- Section A.10 (Backbone Adapted Analysis): Validates robustness performance when adaptation is applied solely to the backbone.
- Section A.11 (End-to-end Generalizability Analysis): Verifies CodeMerge’s extensibility and transferability across diverse end-to-end architectures.

A.1 Implementation Details

For the end-to-end autonomous driving task, we employ ResNet50 [17] as the backbone network to uniformly process image data from both nuScenes and nuScenes-C [53] datasets. All input images are resized to 256×704. We use a 900×256 instance query as input to the transformer layers. Our optimization strategy utilizes the AdamW optimizer, configured with a weight decay of 0.001 and an initial learning rate of 1×10^{-7} . To balance computational efficiency and prediction accuracy, we apply a random projection module to reduce the dimensionality of query features extracted from the pretrained model, resulting in a compact 1024-dimensional feature vector, and manage predictions through a model bank with a limited capacity of five models. Through self-supervised training on detection and tracking heads, the model accurately predicts ten classes as well as the associated instance IDs. For the point cloud detection tasks, we adopt the SECOND [55] as our pretrained model. We configure the training with a batch size of 8, a learning rate of 0.01, and a weight decay of 0.01. Additionally, we utilize a 900×256 dimensional 3D feature vector as input to the leverage module, enabling efficient and effective model merging.

A.2 Evaluation Metrics in End-to-End AD

We follow standard evaluation protocols to assess each task module for end-to-end AD system. **Detection Metrics.** We use nuScenes metrics, including mean Average Precision (mAP) and five error-based scores: mean Average Translation Error (mATE), Scale Error (mASE), Orientation Error (mAOE), Velocity Error (mAVE), and Attribute Error (mAAE). Together, they evaluate spatial, geometric, and semantic aspects of 3D box predictions. The nuScenes Detection Score (NDS) aggregates these metrics into a single score for holistic performance evaluation.

Tracking Metrics. Tracking performance is measured using Average Multi-Object Tracking Accuracy (AMOTA), Precision (AMOTP), and Recall. These metrics capture association quality, localization precision, and coverage of tracked instances.

Online Mapping Metrics. We compute class-wise Average Precision (AP) for static map elements (e.g., lane dividers, crossings, road boundaries) and report mean AP across categories to reflect mapping accuracy and consistency.

Motion Prediction Metrics. We evaluate prediction with best-of- K trajectory metrics: minimum Average Displacement Error (minADE), minimum Final Displacement Error (minFDE), and Miss Rate (MR). We also report End-to-end Prediction Accuracy (EPA), which reflects cascading errors across detection, tracking, and forecasting stages.

Planning Metrics. We assess planning quality using two key indicators: collision rate, which measures the frequency of collisions during trajectory execution, and L2 distance to goal, which quantifies the Euclidean distance between the final position and the intended goal. Together, these metrics reflect the safety and goal-reaching accuracy of the planned motion.

A.3 More Visualizations

In Fig. 5, we present additional visualized predictions from both the non-adapted SparseDrive and the SparseDrive model adapted at test-time using the proposed CodeMerge, illustrating performance across a broader range of corruptions.

A.4 Related Work

Test-time adaptation (TTA) dynamically updates models during deployment to mitigate distribution shifts [26, 50]. Early approaches primarily focus on tuning BatchNorm layers via entropy minimization, adaptive moment estimation, global statistic alignment, or loss landscape smoothing [44, 31, 59, 63, 34, 14, 33]. Subsequent methods explore self-training with confidence-filtered pseudo-labels [15, 30, 60], feature-level consistency or contrastive regularization [5, 22, 47, 38, 64, 49], robustness through data augmentation [62, 42], and leveraging guidance from language models [23]. Extending TTA to more challenging perception tasks (e.g., image- or LiDAR-based object detection [28, 29, 7]), MemCLR aligns 2D detector features using a memory-augmented teacher-student framework [43]; DPO stabilizes LiDAR-based detection via dual perturbation optimization [9]; Reg-TTA3D generates noise-consistent pseudo-labels to supervise low-confidence 3D boxes using high-confidence ones [58]; and MOS enhances adaptation stability by dynamically merging the top- K diverse checkpoints for supervision [8]. Despite their effectiveness, existing methods adapt only perception tasks, while adapting unified end-to-end autonomous driving systems at test time remains unexplored.

Model Merging studies how weight-space operations can effectively compose, refine, or repair vision models through checkpoint averaging, gradient matching, or arithmetic edits to task-specific weight vectors [32, 51, 10, 18, 1, 54]. Recent literature highlights the effectiveness of these merging techniques in enhancing generalization across tasks such as zero-shot learning [52], open-set learning [35], domain adaptation and generalization [36, 2, 11], and cross-domain tasks involving 3D LiDAR point clouds [20, 8]. In this work, we build upon the strengths of model merging techniques to enable efficient, on-the-fly adaptation within end-to-end autonomous driving pipelines.

A.5 Computational cost of each TTA step

The runtime reported in Table 6 is the total time aggregated over all adaptation steps across the entire test set. Below, we report GPU memory usage and per-frame latency alongside $\mathbf{AP}_{3D}$ with SECOND on the **nuScenes** $\rightarrow$ **KITTI** transfer in Table 7.

Table 7: Comparison of different test-time adaptation methods on GPU memory usage, runtime per frame, and 3D detection performance ($\mathbf{AP}_{3D}$).

Method	GPU Memory (MiB)	Runtime (s / Frame)	$\mathbf{AP}_{3D}$
TENT	10,832	0.26	18.83
CoTTA	15,099	0.15	47.61
MOS	17,411	0.49	51.11
CodeMerge	16,041	0.28	58.54

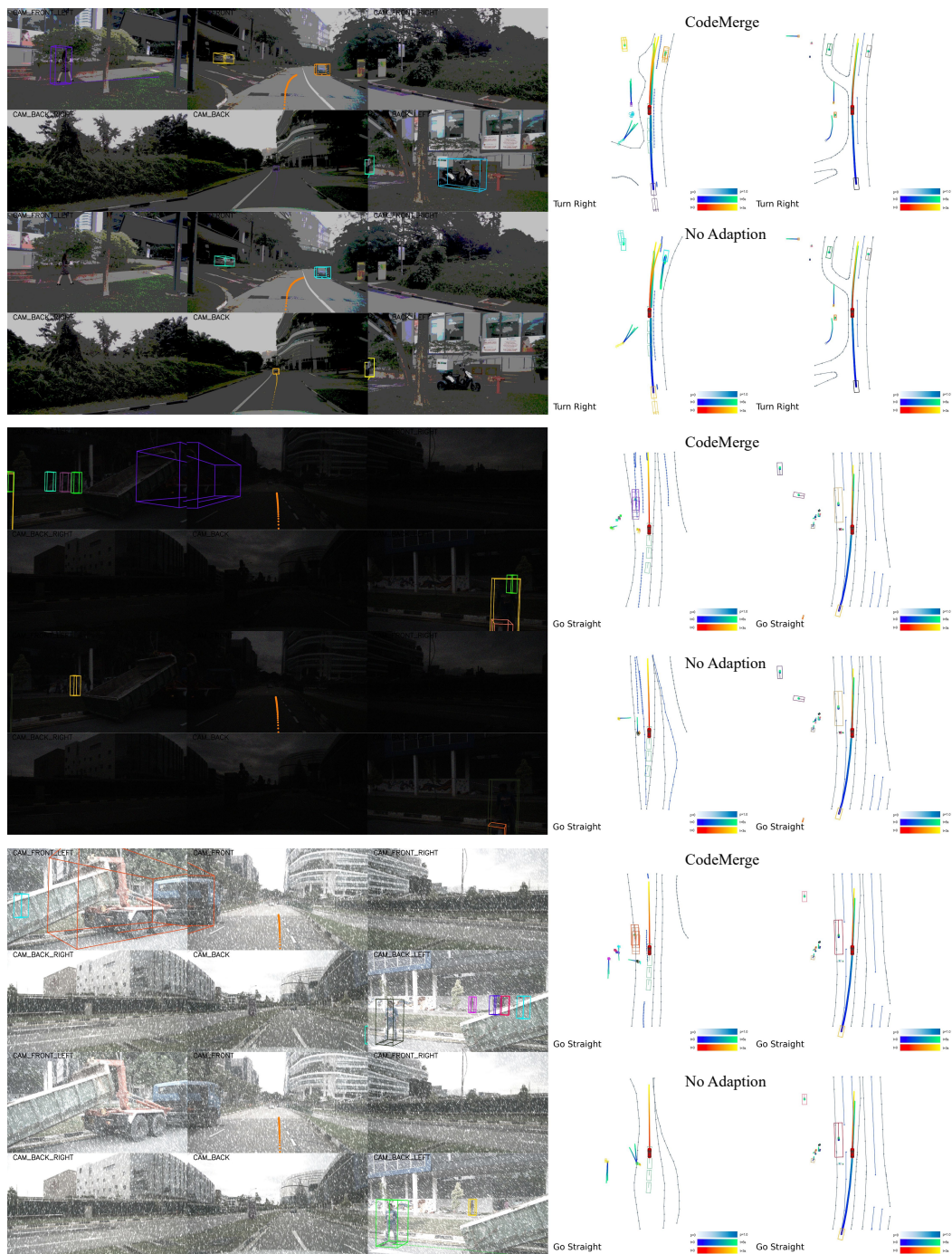


Figure 5: Visualization of outputs of SparseDrive (bottom) and after CodeMerge adaptation (upper) under severe ColorQuant, LowLight, and Snow. TTA greatly improves detection by capturing more true positive instances.

Table 8: **Robustness under different random-projection seeds (RP1–RP3).** Small variations across perception, tracking, mapping, motion prediction, and planning indicate minimal sensitivity to projection randomness.

Seed	3D OBJECT DETECTION							MULTI-OBJECT TRACKING			
	mAP $\uparrow$	NDS $\uparrow$	mATE $\downarrow$	mASE $\downarrow$	mAOE $\downarrow$	mAVE $\downarrow$	mAAE $\downarrow$	AMOTA $\uparrow$	AMOTP $\downarrow$	Recall $\uparrow$	IDS $\downarrow$
RP1	0.3735	0.4946	0.6137	0.2786	0.5468	0.2935	0.1892	0.3347	1.3359	0.4714	869
RP2	0.3693	0.4935	0.6149	0.2786	0.5353	0.2925	0.1897	0.3319	1.3372	0.4756	1256
RP3	0.3630	0.4920	0.6068	0.2764	0.5315	0.2837	0.1962	0.3252	1.3411	0.4553	892

Seed	ONLINE MAPPING				MOTION PREDICTION				PLANNING	
	AP _{ped} $\uparrow$	AP _d $\uparrow$	AP _b $\uparrow$	mAP $\uparrow$	mADE $\downarrow$	mFDE $\downarrow$	MR $\downarrow$	EPA $\uparrow$	L2-Avg $\downarrow$	CL $\downarrow$
RP1	0.4305	0.5224	0.5398	0.4976	0.6504	1.0122	0.1392	0.4680	0.6209	0.094%
RP2	0.4229	0.5172	0.5349	0.4917	0.6415	0.9964	0.1362	0.4682	0.6219	0.110%
RP3	0.4106	0.5109	0.5198	0.4804	0.6406	0.9948	0.1356	0.4660	0.6255	0.125%

Speed comparison. Compared with MOS [8], CodeMerge reduces per-frame latency by **~43%** ($0.49 \rightarrow 0.28$ s) while improving AP_{3D} by **+14.5%**. Relative to TENT [44], it yields a **+210.9%** AP_{3D} gain at essentially the same latency (0.26 vs. 0.28 s per frame). Compared with CoTTA [46], CodeMerge adds only **+0.13 s per frame** yet improves AP_{3D} by **+22.9%**.

Memory comparison. CodeMerge uses less GPU memory than MOS (16,041 vs. 17,411 MiB) and slightly more than CoTTA (15,099 MiB), while achieving the highest AP_{3D} among all methods. Compared with TENT, it requires **+5,209 MiB (~5.1 GiB)** but maintains comparable runtime and delivers markedly larger performance gains (**+210.9%**).

Overall, CodeMerge offers strong accuracy–efficiency trade-offs on SECOND [55] for the **nuScenes** $\rightarrow$ **KITTI** setting, it achieves the highest AP_{3D} with competitive latency and memory, making it practical for real-time test-time adaptation (TTA) in dynamic deployment environments.

A.6 Random Projection Determinism and Robustness

We instantiate a fixed Gaussian random projection once before TTA and keep it frozen throughout adaptation. It is applied to the pretrained source feature extractor’s intermediate activations $\phi_{\Theta(o)}(x)$ to produce low-dimensional fingerprints $\hat{z}$ (refer to Eq. (7)), ensuring all checkpoints are embedded into a shared, comparable space across time steps. Because both the fingerprinting features (from the frozen feature extractor) and the projection matrix are fixed, this mapping is *training-free* and *lightweight*.

As shown in Table 8, we verify robustness by resampling the projection matrix with three seeds (RP1–RP3) and re-running the full pipeline. The results vary only slightly: For perception, NDS($0.4920 \rightarrow 0.4946$) and mAP($0.3630 \rightarrow 0.3735$); For tracking, AMOTA($0.3252 \rightarrow 0.3347$); For planning, L2-Avg($0.6209 \rightarrow 0.6255$). These small ranges indicate that the inherent randomness of the projection has a minor impact on end-to-end AD performance.

Discussion This stability is expected because (i) the projection is a fixed Gaussian linear map instantiated once and kept frozen during TTA, applied to features from the fixed source encoder to produce low-dimensional fingerprints (Eq. (7)), ensuring all checkpoints are embedded in a shared space; and (ii) the fingerprint space faithfully mirrors parameter-space geometry (Fig. 3; Pearson r and Kendall τ typically > 0.7), so different random draws that preserve this linear embedding yield similar merge decisions. Overall, repeated experiments with different seeds exhibit only minor fluctuations, consistent with our design rationale for a stable, geometry-preserving random projection.

A.7 Alternative Fingerprint Generation Strategies

Lightweight pooling-based strategies. We first investigate a suite of computationally efficient feature-aggregation approaches, such as **MaxPool** [37], **AdaptiveAvgPool** [16], **Lp-Pooling** [3], and **Fractional-Max Pooling** [61]. While computationally lightweight, these approaches compress features into coarse statistics and underperform on the **Waymo** $\rightarrow$ **KITTI** transfer task. Our fixed

Table 9: **Comparison of fingerprint generation strategies.** Random projection (RP) achieves the highest AP_{3D} on Waymo $\rightarrow$ KITTI, showing that simple pooling operations lose geometric fidelity required for reliable merging.

Method	RP	MaxPool	AdaptiveAvgPool	Lp-Pooling	Fractional-Max
AP_{3D}	63.23	58.58	60.34	61.40	61.78

Table 10: **TTA with DSVT on Waymo $\rightarrow$ KITTI.** CodeMerge yields the best AP across both BEV and 3D metrics, indicating strong transferability beyond SECOND.

TTA Methods (w. DSVT)	$AP_{BEV} \uparrow$	$AP_{3D} \uparrow$
No Adapt	65.06	27.14
Tent	63.94	31.07
CoTTA	66.63	34.51
SAR	66.12	37.45
DPO	75.46	45.06
MOS	77.38	57.41
CodeMerge	79.88	61.06

Gaussian Random Projection (RP) achieves $AP_{3D} = 63.23$, outperforming the pooling variants by +4.6%, +2.88%, +1.82%, and +1.44%, respectively in Table 9. This result suggests that simple pooling loses the discriminative structure required for behavior-aware merging.

Learning-based dimensionality reduction methods. We also consider learning-based techniques such as PCA [21], which requires fitting or updating decompositions on streaming features. However, these incur significant computational overhead at every TTA step, making them incompatible with our single-pass, real-time adaptation setting.

Pooling is high-throughput yet discards geometry and discriminative structure, while PCA-like methods impose computational costs incompatible with real-time operation. Our fixed random projection preserves model-space geometry, satisfies real-time efficiency requirements, and empirically delivers the strongest accuracy among all tested strategies.

A.8 Applicability beyond SECOND

To assess generality, we further evaluate CodeMerge on DSVT [45] under the **Waymo $\rightarrow$ KITTI** transfer in Table 10. CodeMerge attains **79.88 AP_{BEV} / 61.06 AP_{3D}** , outperforming the strongest baseline MOS [8] by **+3.2%** and **+6.4%**, respectively, and surpassing **No Adapt** by **+125%** in AP_{3D} . Because CodeMerge builds on training-free random-projection fingerprints and codebook-guided merging rather than architecture-specific modules, these results indicate our TTA is model-agnostic and transfers effectively to detectors beyond SECOND.

A.9 Robustness under Fog & LiDAR Dropout

We conduct additional experiments adapting SECOND from **KITTI $\rightarrow$ KITTI-C** under **heavy fog** with increasing point dropout (0% $\rightarrow$ 25% $\rightarrow$ 50%). We report AP at the moderate level in Table 11. With stronger dropout, performance degrades gradually rather than catastrophically: AP_{3D} drops from 75.96 $\rightarrow$ 73.87 $\rightarrow$ 71.95 (−4.01%, $\approx$ 5.3% relative), and AP_{BEV} from 88.16 $\rightarrow$ 85.73 $\rightarrow$ 85.04 (−3.12%, $\approx$ 3.5% relative). This smooth, monotonic decline suggests that our checkpoint merging remains reliable even under compounded shifts.

A.10 Backbone-only Adaptation

End-to-end AD framework. To verify the effect of merging and keep TTA lightweight, we freeze all modules except the 3D detection head. As shown in Table 12, we additionally enable **backbone (ResNet) adaptation** and observe negligible deltas relative to head-only TTA (e.g., NDS −0.0034, mAP −0.0055, AMOTA −0.0028, map mAP −0.0097, L2-Avg −0.0023), indicating that CodeMerge remains effective even when only the backbone is updated.

Table 11: **Fog & LiDAR point-dropout robustness on KITTI → KITTI-C (moderate)**. Performance decays smoothly as dropout increases, indicating stable ranking/merging.

Dropout Ratio	0%	25%	50%
$AP_{3D} \uparrow$	75.96	73.87	71.95
$AP_{BEV} \uparrow$	88.16	85.73	85.04

Table 12: **Head-only vs. backbone-adapted TTA**. Enabling ResNet backbone adaptation yields only minor differences, indicating CodeMerge remains effective with backbone updates.

ColorQuant	3D OBJECT DETECTION							MULTI-OBJECT TRACKING			
	mAP $\uparrow$	NDS $\uparrow$	mATE $\downarrow$	mASE $\downarrow$	MAOE $\downarrow$	MAVE $\downarrow$	MAAE $\downarrow$	AMOTA $\uparrow$	AMOTP $\downarrow$	Recall $\uparrow$	IDS $\downarrow$
CodeMerge + det head	0.2742	0.4331	0.6575	0.2764	0.5903	0.3018	0.2137	0.2339	1.4868	0.3330	490
CodeMerge + ResNet	0.2687	0.4297	0.6574	0.2778	0.5906	0.3092	0.2119	0.2311	1.4799	0.3468	510

ColorQuant	ONLINE MAPPING			MAP	MOTION PREDICTION				PLANNING	
	$AP_{ped} \uparrow$	$AP_d \uparrow$	$AP_b \uparrow$	mAP $\uparrow$	minADE $\downarrow$	minFDE $\downarrow$	MR $\downarrow$	EPA $\uparrow$	L2-Avg $\downarrow$	CL $\downarrow$
CodeMerge + det head	0.2600	0.3445	0.3267	0.3104	0.7002	1.0859	0.1454	0.3840	0.6729	0.106%
CodeMerge + ResNet	0.2407	0.3452	0.3161	0.3007	0.7087	1.0930	0.1453	0.3793	0.6752	0.102%

3D detection framework. Following prior works [8, 9], we also consider adapting the entire backbone (encoder, detection head, and all batch-norm layers). These studies have shown that, for test-time adaptation in 3D detection, full-backbone adaptation typically surpasses BN-only updates; CodeMerge fits seamlessly into this setting as well.

A.11 Generalizability Beyond SparseDrive

Early end-to-end models under corruption. Some previous end-to-end architectures (e.g., VAD [19]) collapse on corrupted scenes even before any TTA is applied. Under **heavy snow**, VAD attains only **0.0168 mAP** and **0.0353 NDS** for detection in Table 13, indicating that the base model’s predictions are already unreliable and nearly collapsed. In this regime, online self-supervision becomes untrustworthy, so any TTA method has little signal to learn from. This reflects a robustness limitation of the base model, not of TTA or CodeMerge. Even so, attaching CodeMerge to VAD yields small but consistent gains across perception and planning, and does not degrade VAD’s performance, underscoring that the bottleneck lies in model brittleness rather than our adaptation mechanism.

Table 13: **VAD under Snow (severe)**. Detection collapses pre-TTA; CodeMerge yields small but consistent improvements without degradation.

Method	3D OBJECT DETECTION							MULTI-OBJECT TRACKING			
	mAP $\uparrow$	NDS $\uparrow$	mATE $\downarrow$	mASE $\downarrow$	MAOE $\downarrow$	MAVE $\downarrow$	MAAE $\downarrow$	AMOTA $\uparrow$	AMOTP $\downarrow$	Recall $\uparrow$	IDS $\downarrow$
VAD	0.0168	0.0353	0.9691	0.8577	0.9713	1.1064	0.9327	–	–	–	–
VAD+CodeMerge	0.0176	0.0356	0.9689	0.8576	0.9719	1.1019	0.9337	–	–	–	–

Method	ONLINE MAPPING			MAP	MOTION PREDICTION				PLANNING	
	$AP_{ped} \uparrow$	$AP_d \uparrow$	$AP_b \uparrow$	mAP $\uparrow$	minADE $\downarrow$	minFDE $\downarrow$	MR $\downarrow$	EPA $\uparrow$	L2-Avg $\downarrow$	CL $\downarrow$
VAD	0.0006386	0.0014679	0.0002426	0.0007830	2.3891	3.7662	0.3022	0.0479	1.6230	0.01320
VAD+CodeMerge	0.0005710	0.0015413	0.0002724	0.0007949	2.3265	3.6645	0.2974	0.0519	1.6057	0.01307

Recent architectures with non-collapsed baselines. When the backbone remains minimally competent under corruption, CodeMerge is highly effective and broadly applicable. On DiffusionDrive [27] and MomAD [39], CodeMerge delivers consistent improvements across detection, tracking, mapping, motion, and planning, and reduces collision rate under both **Brightness** and **Snow** corruptions, indicating scalability beyond SparseDrive.

Table 14: **Brightness corruption (severe)**. CodeMerge improves DiffusionDrive and MomAD across modules and reduces collision (CL).

Method	3D OBJECT DETECTION							MULTI-OBJECT TRACKING			
	mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓	AMOTA↑	AMOTP↓	Recall↑	IDS↓
DiffusionDrive	0.3278	0.4588	0.6402	0.2782	0.6473	0.3005	0.1851	0.2818	1.4334	0.4076	787
DiffusionDrive+CodeMerge	0.3580	0.4845	0.6210	0.2747	0.5767	0.2811	0.1916	0.3150	1.3529	0.4638	873
MomAD	0.3340	0.4711	0.6276	0.2734	0.5931	0.2900	0.1747	0.2944	1.4241	0.4101	731
MomAD+CodeMerge	0.3664	0.4952	0.6092	0.2741	0.5496	0.2670	0.1803	0.3341	1.3447	0.4737	824

Method	ONLINE MAPPING			MAP	MOTION PREDICTION				PLANNING	
	AP _{ped} ↑	AP _d ↑	AP _b ↑	mAP↑	mADE↓	mFDE↓	MR↓	EPA↑	L2-Avg↓	CL↓
DiffusionDrive	0.3651	0.4819	0.4794	0.4421	0.6624	1.0282	0.1380	0.4448	0.6001	0.089%
DiffusionDrive+CodeMerge	0.4125	0.5162	0.5333	0.4873	0.6582	1.0261	0.1380	0.4648	0.5919	0.062%
MomAD	0.2490	0.1808	0.3157	0.2485	1.3402	2.2509	0.2212	0.3965	7.3178	8.500%
MomAD+CodeMerge	0.2755	0.1976	0.3365	0.2699	1.1330	2.2389	0.2211	0.4147	7.3171	8.438%

Table 15: **Snow corruption (severe)**. CodeMerge improves DiffusionDrive and reduces collision; MomAD also benefits across modules.

Method	3D OBJECT DETECTION							MULTI-OBJECT TRACKING			
	mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓	AMOTA↑	AMOTP↓	Recall↑	IDS↓
DiffusionDrive	0.1006	0.2373	0.7811	0.3794	0.9105	0.7007	0.3585	0.0503	1.8648	0.1064	305
DiffusionDrive+CodeMerge	0.1797	0.3492	0.7389	0.2956	0.6673	0.4621	0.2424	0.1078	1.7171	0.1785	681

Method	ONLINE MAPPING			MAP	MOTION PREDICTION				PLANNING	
	AP _{ped} ↑	AP _d ↑	AP _b ↑	mAP↑	mADE↓	mFDE↓	MR↓	EPA↑	L2-Avg↓	CL↓
DiffusionDrive	0.0106	0.0426	0.0512	0.0348	1.0603	1.7077	0.1951	0.2063	0.9300	0.447%
DiffusionDrive+CodeMerge	0.1010	0.1713	0.1688	0.1471	0.8086	1.2838	0.1730	0.3070	0.7730	0.215%

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: The abstract and introduction clearly state the core contributions: a compact fingerprint-guided model merging framework, its theoretical grounding via leverage scores, and demonstrated improvements (*e.g.* >100% gap closure on Waymo→KITTI and 7.7% AP boost under KITTI → KITTI-C)—all of which are borne out by the experiments.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We have included detailed limitations in the conclusion section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (*e.g.*, independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, *e.g.*, if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper clearly states all assumptions underlying its theoretical results: linear mode connectivity of fine-tuned checkpoints, the Gaussian–Newton approximation of the Hessian, and regularized leverage-score derivation, including the spectrum bound and sampling error analyses that connect fingerprint correlations to optimal merging coefficients.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides exhaustive implementation details (frameworks, hardware, batch sizes, hyperparameters). All datasets, corruption settings, and model checkpoints are specified, and we commit to releasing our code and pretrained codebook, ensuring full experimental reproducibility.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will publicly release our full implementation, including data preprocessing scripts, checkpoint codebook, and evaluation pipelines, alongside the pretrained SparseDrive weights and nuScenes/Waymo split configurations.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The paper details all training and testing settings, including dataset splits, optimizer type, learning rates, batch sizes, and checkpoint schedules, in both the main text and appendix, ensuring readers have sufficient information to reproduce every experimental result.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: While our results clearly demonstrate consistent gains, we did not report error bars, confidence intervals, or statistical tests across multiple runs.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We report in Section 4 that all experiments run on a single NVIDIA RTX A6000 GPU with 48 GiB of memory. We specify batch sizes for each task, and the frameworks used (SparseDrive / OpenPCDet), and provide per-checkpoint merging overhead. This level of detail is sufficient for reproducing both runtime and memory requirements.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Our work uses only publicly available datasets (KITTI, nuScenes, Waymo) and standard benchmarks under their license terms. No personal or sensitive data is involved, and all experimental protocols conform to the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: There is no societal impact of the work performed

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We build on standard open-source assets—SparseDrive and OpenPCDet (Apache 2.0) and the KITTI, Waymo, and nuScenes datasets (with their respective academic licenses)—all of which are explicitly cited in Section 4.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA] .

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.