MM-TTA: MULTIMODAL TEST-TIME ADAPTATION WITH CONSISTENCY-BASED MODEL MERGING FOR 3D OBJECT DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Robust test-time adaptation (TTA) for 3D object detection (3DOD) plays a vital role in autonomous driving systems when facing severe domain shifts. Existing TTA methods for 3DOD mainly rely on the LiDAR modality and are vulnerable to various types of domain shifts. In this paper, we introduce a two-stage multimodal TTA method, referred to as MM-TTA, to improve the adaptation capability of 3DOD models. In the first pre-training stage, we propose the Geometric Consistency Alignment (GCA) strategy to align the 3D perception spaces of source domain multimodal data with those of the target domain, thereby preventing mismatches in parameter and feature dimensions when deploying the pretrained fusion model to the target domain, while maintaining superior performance over LiDAR-based models despite differences in camera and LiDAR configurations between domains. In the second test-time adaptation stage, we further propose the Consistency-Based Model Merging strategy (CMM) for robust transfer learning on the target domain. Instead of choosing diverse historical models, we only select the most consistent ones for model merging and use the merged model to guide the subsequent fine-tuning. This is because historical fusion models with low similarity are more likely to exhibit performance discrepancies than complementarity, which can introduce interference and degrade the merged model. To reduce the computational costs from model selection, we also propose the *sign consistency score* to measure the similarity between model pairs, which is computed only once without repeatedly accessing historical models. The CMM strategy achieves better performance than previous methods with fewer time steps and significantly reduces the computational costs introduced by model selection and numerous parameters of the fusion model itself. Our method is evaluated on five datasets and eight corruption types, demonstrating its effectiveness under various domain shifts in real-world scenarios (with a 15.24% improvement over the second-best method in a cross-dataset scenario).

1 INTRODUCTION

3D object detection is a crucial component for autonomous driving systems and gains a lot of attention in recent years with the widespread deployment of onboard cameras and LiDAR sensors. However, deploying a pretrained detector into real world may cause the failure of the perception systems due to the acute domain shifts. Domain shifts may occur when perception systems encounter varying weather conditions, different locations, sensor changes, or malfunctions, thereby posing significant challenges to the domain adaptation capability of detection models.

To address domain shifts during deployment, test-time domain adaptation (TTA) has emerged as a practical solution that seeks a balance between real-time inference demands and detection performance. TTA methods aim to adapt pretrained models to live data streams and provide real-time inference. There have been several TTA methods working on improving the model's real-time adaptation ability, as well as reducing computational costs and increasing inference speed. However, most of them are developed for 2D image detection and classification and fall short in 3D scenes, as domain shifts in 3D scenes are substantially different from those in images. Other methods de-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

veloped for 3D object detection focus solely on the LiDAR modality and are vulnerable to different types of domain shifts.

In this paper, we present a new method to address the problem of Test-Time Adaptation for 3D object detection in a multimodal fusion setting. Extensive experiments have shown the superior performance of a LiDAR-image fusion model over a LiDAR-based model. However, the fusion model is not directly deployable across domains, since different domains yield distinct parameter and feature dimensions in fusion models under varying camera and LiDAR sensor configurations. The fundamental cause lies in the discrepancy of their 3D perception spaces $\mathcal{S}$, which leads to differences in the number of model parameters and the dimensionality of intermediate features. To address this, we propose the Geometric Consistency Alignment (GCA) strategy in the pre-training stage to adjust the source domain point clouds and images so that they share the same 3D perception spaces as those in the target domain. In this way, we can ensure the fusion model $\mathcal{F}_0$ trained in the source domain (1) learns geometric information aligned with that of the target domain, (2) avoids model parameter mismatches and feature size mismatches when deployed in the target domain, and (3) preserves superior zero-shot performance in the target domain compared with a LiDAR-based model. After pre-training using the GCA strategy and obtaining $\mathcal{F}_0$, we further propose the Consistency-Based Model Merging (CMM) strategy to achieve robust TTA transfer learning. As $\mathcal{F}_0$ changes more significantly than LiDAR-based models during fine-tuning, historical models with low similarity are more likely to exhibit performance discrepancies rather than complementarity. Thus, choosing diverse historical models for model merging as in previous methods may cause interference to the merged model and lead to performance degradation. To avoid this problem, we only select the most consistent models for merging. In this way, we can ensure that the merged model generates more reliable pseudo-labels and improves its robustness against different domain shifts, so that it can better guide the subsequent fine-tuning. We also simplify the model selection process through the proposed *sign consistency score* to reduce computational costs and avoid repeatedly accessing historical models, while maintaining effective model selection. The **contributions** are:

1. This is an early attempt to introduce LiDAR-image fusion model for TTA on 3D object detection. We propose the Geometric Consistency Alignment (GCA) strategy to make it possible that the fusion model pretrained on the source domain can be directly deployed in the target domain without any modification to the model parameters while maintaining superior performance to LiDAR-based models.
2. We propose the Consistency-Based Model Merging (CMM) strategy during the TTA stage to ensure robust fine-tuning by selecting historical models which are mutually consistent with each other for model merging while reducing the computational costs of model selection and numerous parameters of the fusion model.
3. We validate our method on five representative benchmarks covering five datasets and eight corruption types. Results show that we achieve better performance on various domain shifts and reach peak performance with fewer time steps (with a 15.24% improvement over the second-best method in a cross-dataset scenario).

2 Related Work

Multimodal 3D Object Detection aims to leverage the advantages of both image and LiDAR modalities to address the drawbacks of each single modality. While image-based methods (Li et al. (2023a;b); Yang et al. (2023b; 2025a); Liu et al. (2022; 2023a); Li et al. (2024); Yang et al. (2023a)) rely on inaccurate depth estimation from images to lift 2D features into 3D space, LiDAR-based methods (Yin et al. (2021); Yan et al. (2018); Deng et al. (2021); Lang et al. (2019); Shi et al. (2020; 2023)) struggle to detect small or distant objects due to the sparse nature of point clouds. To overcome these limitations, multimodal fusion methods (Liang et al. (2022); Liu et al. (2023b); Bai et al. (2022); Yang et al. (2022b); Song et al. (2024); Huang et al. (2024); Yang et al. (2025b); Su et al. (2025)) have been proposed to leverage the accurate depth information from point clouds and the rich semantic information from images, thereby achieving superior performance over single-modality methods. However, existing multi-modal fusion methods rarely consider cross-domain detection scenarios, which raises concerns about the generalization of the multimodal fusion methods in real-world applications.

Domain Adaptation for 3D Object Detection aims to alleviate the performance degradation of pretrained 3D detectors deployed for inference in the target domain. Existing 3D domain adaptation

methods can be categorized into two paradigms: offline domain adaptation and test-time adaptation. Offline methods (Wang et al. (2020); Wei et al. (2022); Hu et al. (2023); Yang et al. (2021; 2022a); Shin et al. (2024); Wang et al. (2023); Kim et al. (2024); Chen et al. (2024a); Zhang et al. (2024); Wozniak et al. (2024); Chang et al. (2024); Zhao et al. (2026)) need to fine-tune the pretrained model with multi-round self-training on the unlabled target data before deployment for real-time inference. Despite achieving impressive performance, these offline methods are time-consuming and computationally expensive, making them unsuitable for scenarios where test data arrive in a live data stream. TTA methods (Wang et al. (2025); Liang et al. (2025)) aim to adapt the model to a live data stream, which requires the model to produce instant inference while simultaneously performing self-training. Early TTA methods (Wang et al. (2021); Niu et al. (2023); Wang et al. (2022); VS et al. (2023); Zeng et al. (2024); Goyal et al. (2022)) are developed for image detection and classification. However, these image-based TTA methods fall short in 3D scenes, as domain shifts in 3D scenes are substantially different from those in 2D images. Recently, several LiDAR-based TTA methods (Chen et al. (2024b; 2025); Yang et al. (2026)) have been proposed to address domain shifts in 3D object detection. However, these methods rely solely on the LiDAR modality, which is more vulnerable to adverse weather conditions and sensor failures.

3 METHOD

3.1 OVERALL FRAMEWORK

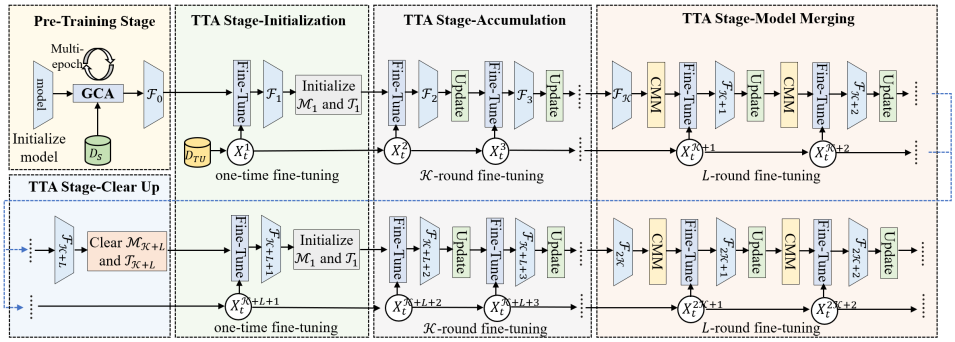

Figure 1: An overview of our framework. In the first pre-training stage, we employ the GCA strategy to pre-train the fusion model $\mathcal{F}_0$. In the second TTA stage, we finely tune $\mathcal{F}_0$ by the following four steps: *Initialization*, *Accumulation*, *Model Merging*, and *Clear Up*. We employ the CMM strategy in the *Model Merging* step to ensure stable transfer learning.

We denote the source dataset as $D_S = \{X_s^i\}_{i=1}^{N_S}$, where $X_s^i = \{P_s^i, I_s^i, L_s^i\}$, N_S is the number of samples, and P_s^i, I_s^i, and L_s^i denote the point cloud, images, and ground-truth labels of the i-th sample X_s^i, respectively. We denote the target dataset as $D_T = \{X_t^i\}_{i=1}^{N_T}$, where $X_t^i = \{P_t^i, I_t^i, \emptyset\}$, and the ground-truth labels $\{L_t^i\}_{i=1}^{N_T}$ remain unavailable during adaptation. As shown in Fig 1, our framework consists of two stage: the pre-training stage and the TTA stage.

In the pre-training stage, we pretrain a LiDAR-image fusion model $\mathcal{F}_0$ using D_S with the GCA strategy. For each input sample X_s^i, we adjust P_s^i, I_s^i, and L_s^i to ensure that they share the same 3D perception space as the samples in D_T. We train $\mathcal{F}_0$ for multi epochs until convergence. In the TTA stage, we feed each test sample X_t^i, which arrives in a live data stream, to fine-tune $\mathcal{F}_0$ for the following four steps. **(I) Initialization.** For time step $k = 1$ and an incoming sample X_t^1, we fine-tune $\mathcal{F}_0$ using X_t^1 and its pseudo-labels generated by $\mathcal{F}_0$ to obtain $\mathcal{F}_1$. We initialize two variables in GPU memory: one is the similarity matrix $\mathcal{M}_1 = (\mathrm{Sim}\langle \mathcal{F}_1, \mathcal{F}_1 \rangle)$, where $\mathrm{Sim}\langle \mathcal{F}_1, \mathcal{F}_1 \rangle = 1$; the other is the *task vector* (detailed in sec. 3.3) bank $\mathcal{T}_1 = (T_1^{'})$, where $T_1^{'}$ is the *task vector* of $\mathcal{F}_1$. **(II) Accumulation.** This step aims to retain historical models as candidates for subsequent model merging. We define $\mathcal{K}$ as the hyperparameter controlling the number of time steps to perform accumulation. For time step $1 < k < \mathcal{K}$ and an incoming sample X_t^k, we fine-tune $\mathcal{F}_{k-1}$ using X_t^k and its pseudo-labels generated by $\mathcal{F}_0$ to obtain $\mathcal{F}_k$ and $T_k^{'}$. We then use $\mathcal{T}_{k-1} = (T_i^{'})_{i=1}^{k-1}$ and $T_k^{'}$ to compute $(\mathrm{Sim}\langle T_i^{'}, T_k^{'} \rangle)_{i=1}^{k}$ with our proposed *sign consistency score* and update $\mathcal{M}_{k-1} = (\mathrm{Sim}\langle \mathcal{F}_i, \mathcal{F}_j \rangle)_{i,j=1}^{k-1}$ with these scores to obtain $\mathcal{M}_k = (\mathrm{Sim}\langle \mathcal{F}_i, \mathcal{F}_j \rangle)_{i,j=1}^{k}$. We also update $\mathcal{T}_{k-1}$ with $T_k^{'}$ to obtain $\mathcal{T}_k$. **(III) Model Merge.** This step aims to select appropriate candidates and obtain

a robust merged model to guide the subsequent fine-tuning. We define L as the hyperparameter controlling the number of time steps to perform model merging. For time step $\mathcal{K} \leq k < \mathcal{K} + L$ and an incoming sample X_t^k, we conduct model merging. We use $\mathcal{M}_{k-1}$ and our proposed CMM strategy to obtain a merged model $\mathcal{F}_{k-1}^*$. We then fine-tune $\mathcal{F}_{k-1}$ using X_t^k and its pseudo-labels generated by $\mathcal{F}_{k-1}^*$ to obtain $\mathcal{F}_k$ and $T_k^{'}$. Accordingly, we update $\mathcal{M}_{k-1}$ and $\mathcal{T}_{k-1}$ to $\mathcal{M}_k$ and $\mathcal{T}_k$, respectively. **(IV) Clear Up.** This step aims to mitigate the memory overhead caused by the increasing number of historical models. At time step $k = \mathcal{K} + L$, we clear $\mathcal{M}_k$ and $\mathcal{T}_k$, reset $k = 1$, and then repeat steps (I), (II) and (III) starting from the incoming sample $X_t^{\mathcal{K}+L+1}$.

3.2 Geometric Consistency Alignment Strategy in Pre-Training Stage

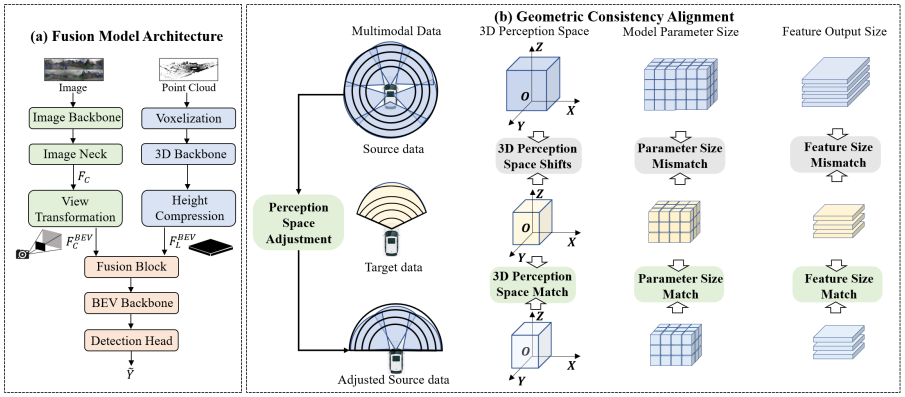

Figure 2: An overview of the fusion model architecture and our proposed Geometric Consistency Alignment (GCA) strategy. Our GCA strategy aligns the 3D perception space of D_S with that of D_T and ensures that the same set of model parameters is applicable to both domains.

In this section, we first explain why a fusion model trained on D_S cannot be directly deployed in D_T, and then discuss how to address this problem. We define the 3D perception space $\mathcal{S}$ as $\mathcal{S} = [X_1, X_2] \times [Y_1, Y_2] \times [Z_1, Z_2]$, where $X_1, Y_1, Z_1 \leq 0$ and $X_2, Y_2, Z_2 \geq 0$. We denote the 3D perception spaces of D_S and D_T as $\mathcal{S}_S$ and $\mathcal{S}_T$, respectively. Directly deploying $\mathcal{F}_0$ from D_S to D_T without aligning $\mathcal{S}_S$ and $\mathcal{S}_T$ will lead to mismatches in model parameter and feature dimensions. As shown in Fig. 2(a), $\mathcal{F}_0$ separately extracts point cloud and image features to obtain point cloud BEV features F_L^{BEV} and image features F_C, respectively. The view transformation module of $\mathcal{F}_0$ then projects F_C into spatial image BEV features F_C^{BEV} using the LSS algorithm Philion & Fidler (2020). Finally, $\mathcal{F}_0$ processes F_L^{BEV} and F_C^{BEV} in the unified BEV space and feeds the fused features into the detection head to generate prediction $\hat{Y}$. Without the alignment of $\mathcal{S}_S$ and $\mathcal{S}_T$, $\mathcal{F}_0$ will generate F_L^{BEV} and F_C^{BEV} with different dimensions when deployed on D_T. These features contain geometric information that differs substantially from that learned from D_S, thereby largely confusing the detection head and hindering the generation of correct $\hat{Y}$. Moreover, the parameters of the view transformation module are predefined according to $\mathcal{S}_S$, making them incompatible with inputs from D_T, which has a different 3D perception space.

As shown in Fig. 2(b), the goal of the GCA strategy is to ensure that all samples in D_S share the same 3D perception space as those in D_T. This ensures that (1) the model $\mathcal{F}_0$ can operate on both D_S and D_T without any modification to its parameter shapes, and (2) the geometric information learned by $\mathcal{F}_0$ from D_S is similar to that in D_T. It consists of the following three steps: (1) determine the $\mathcal{S}_T$ of D_T, (2) use $\mathcal{S}_T$ to refine the point clouds in D_S and obtain the new camera configuration of D_S under $\mathcal{S}_T$, and (3) use the refined D_S to pretrain a fusion model $\mathcal{F}_0$ with the new configuration under $\mathcal{S}_T$. The 3D perception space can be considered identical across all samples from the same dataset. **Step 1**: Given any $X_t^i \in D_T$, we obtain a point cloud subset $\tilde{P}_t^i \subseteq P_t^i$ such that each point in $\tilde{P}_t^i$ has a valid projection onto at least one of the multi-view images I_t^i. We then use the perception space of $\tilde{P}_t^i$ to determine $\mathcal{S}_T$. **Step 2**: For any $X_s^i \in D_S$, we obtain the refined point cloud subset $\tilde{P}_s^i = \tilde{P}_s^i(1) \cap \tilde{P}_s^i(2)$, where $\tilde{P}_s^i(1)$ contains points within $\mathcal{S}_T$ and $\tilde{P}_s^i(2)$ contains points with valid projections onto at least one of the multi-view images I_s^i. Since $\tilde{P}_s^i$ may cover a smaller perception space than I_s^i, it is necessary to refine I_s^i to obtain $\tilde{I}_s^i$, which has the same 3D perception

space as $\mathcal{S}_T$. $\tilde{I}_s^i = I_s^i \setminus I_{s,\text{out}}^i$, where $I_{s,\text{out}}^i$ contains the images without valid projections of points from $\tilde{P}_s^i$. Note that the selection of $\tilde{I}_s^i$ can also be considered identical across all samples from the same dataset. We also filter out labels that lie outside $\mathcal{S}_T$ to obtain the refined labels $\tilde{L}_s^i$. **Step 3:** We feed each refined sample $\tilde{X}_s^i = \{\tilde{P}_s^i, \tilde{I}_s^i, \tilde{L}_s^i\}$ to pretrain $\mathcal{F}_0$ before deploying it to D_T. We provide additional details on the selection of point clouds and images in Appendix A.3.2.

3.3 Consistency-Based Model Merging in TTA Stage

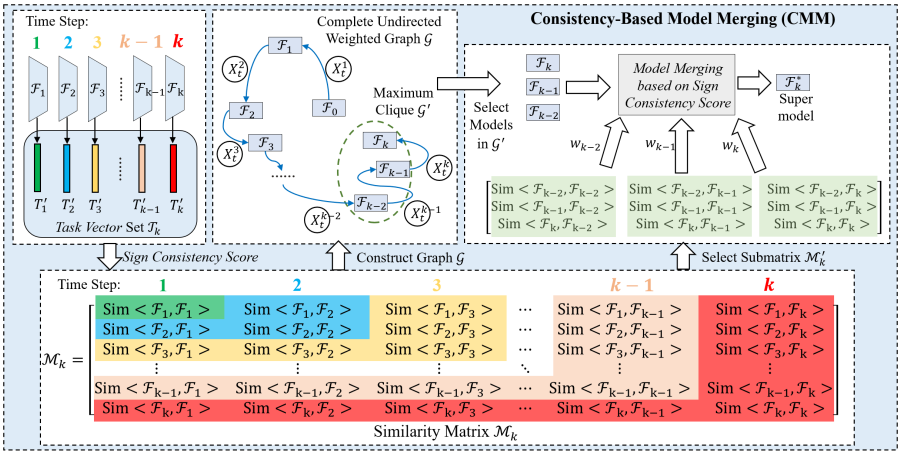

Figure 3: An overview of our proposed Consistency-Based Model Merge strategy. We use our proposed *Sign Consistency Score* to measure the similarity between each pair of models and select the most consistent models for model merging.

We propose the Consistency-Based Model Merging (CMM) strategy for the following two considerations: (1) To avoid multiple forward passes for each incoming sample from historical models or the pretrained model in previous methods (*e.g.,* MOS and CodeMerge), reduce GPU memory consumption, and improve inference speed. (2) To exploit an effective model merging strategy for $\mathcal{F}_0$ to achieve more robust transfer learning under different types of domain shifts. As shown in Fig 3, we maintain two variables in the GPU memory, one is the *Task Vector* bank $\mathcal{T}_k = (T_i^{'})_{i=1}^k$, the other is the similarity matrix $\mathcal{M}_k = (\text{Sim}\langle \mathcal{F}_i, \mathcal{F}_j \rangle)_{i,j=1}^k$. Each *task vector* $T_i^{'}$ in $\mathcal{T}_k$ is the representation of the whole weights of $\mathcal{F}_i$ in the GPU memory. We use $\mathcal{T}_k$ to compute the similarity between model pairs with our proposed *sign consistency score* and store the results in $\mathcal{M}_k$. We then construct a weighted graph $\mathcal{G}$ using $\mathcal{M}_k$ and select models in the maximum clique $\mathcal{G}^{'}$ based on *sign consistency score* for model merging. We perform model merging using models in $\mathcal{G}^{'}$ and their corresponding similarity submatrix $\mathcal{M}_k^{'}$. We provide the details in Section 3.3.1 and 3.3.2.

3.3.1 Sign Consistency Score

Given a model pair $(\mathcal{F}_i, \mathcal{F}_j)$, we denote the model parameters of $\mathcal{F}_i$ and $\mathcal{F}_j$ as θ_i and θ_j, respectively. We denote the cosine similarity between θ_i and θ_j as $\text{Cos}\langle \theta_i, \theta_j \rangle$. The most straightforward way to obtain $\text{Sim}\langle \mathcal{F}_i, \mathcal{F}_j \rangle$ is to compute $\text{Cos}\langle \theta_i, \theta_j \rangle$. However, calculating $\text{Cos}\langle \theta_i, \theta_j \rangle$ is computationally expensive, and not all parameters contribute equally to its estimation (*e.g.,* the image branch is frozen during the TTA stage for inference efficiency). Moreover, $\text{Cos}\langle \theta_i, \theta_j \rangle$ may not accurately reflect the differences between model pairs in the weight space, which could have negative impacts on the model-merging weights (as described in Section 3.3.2). To reduce computational overhead and more accurately reflect $\text{Sim}\langle \mathcal{F}_i, \mathcal{F}_j \rangle$, we (1) select only a subset of parameters, denoted by $\theta_i^{'}$ and $\theta_j^{'}$, from the linear regression layer following the feature extractor, as this layer is most relevant to the model outputs, and (2) propose the *sign consistency score*, denoted by $\text{SignSim}(\cdot)$, to estimate $\text{Sim}\langle \mathcal{F}_i, \mathcal{F}_j \rangle$. We provide additional analysis of the cosine score and the *sign consistency score* in Appendix A.1. The computation process consists of three steps. First, inspired by TIES-Merging Yadav et al. (2023), we construct two *task vectors* as $T_i = \theta_i^{'} - \theta_0^{'}$ and $T_j = \theta_j^{'} - \theta_0^{'}$. T_i and T_j reflect the element-wise changes in direction and magnitude of the fine-tuned models $\mathcal{F}_i$ and $\mathcal{F}_j$ relative to the pretrained model $\mathcal{F}_0$, respectively. Therefore, they can better represent the differences

between $\mathcal{F}_i$ and $\mathcal{F}_j$ in the weight space. Second, we further apply random linear projection to map the high-dimensional T_i and T_j into low-dimensional $T_i^{'}$ and $T_j^{'}$ for efficiency. Third, we obtain $\mathrm{Sim}\langle\mathcal{F}_i, \mathcal{F}_j\rangle$ by calculating $\mathrm{SignSim}\langle T_i^{'}, T_j^{'}\rangle$. The process can be formulated as follows:

$$\mathrm{Sim}\langle\mathcal{F}_i, \mathcal{F}_j\rangle = \mathrm{SignSim}\langle T_i^{'}, T_j^{'}\rangle = \frac{\sum_{k=1}^{D} \frac{k}{D} \min\left(\left|\tau_{i,k}\right|, \left|\tau_{j,k}\right|\right) \mathbb{I}[\mathrm{sign}(\tau_{i,k}) = \mathrm{sign}(\tau_{j,k})]}{\sum_{k=1}^{D} \frac{k}{D} \max\left(\left|\tau_{i,k}\right|, \left|\tau_{j,k}\right|\right)}, \tag{1}$$

where $T_i^{'} = \mathrm{RandProj}(\theta_{\mathrm{i}}^{'} - \theta_0^{'})$, and $T_j^{'} = \mathrm{RandProj}(\theta_{\mathrm{j}}^{'} - \theta_0^{'})$. $\frac{k}{D}$ denotes the positional weight assigned to the k-th components of $T_i^{'}$ and $T_j^{'}$. Here, $\tau_{i,k}$ and $\tau_{j,k}$ denote the k-th components of $T_i^{'}$ and $T_j^{'}$, respectively. The indicator function $\mathbb{I}\left[\mathrm{sign}(\tau_{i,k}) = \mathrm{sign}(\tau_{j,k})\right]$ takes the value 1 if $\tau_{i,k}$ and $\tau_{j,k}$ have the same sign, and 0 otherwise. $\mathrm{SignSim}\langle T_i^{'}, T_j^{'}\rangle$ ranges from 0 to 1, with a higher value indicating greater similarity between $\mathcal{F}_i$ and $\mathcal{F}_j$. Through $\mathrm{SignSim}\langle\cdot,\cdot\rangle$, we ensure that $\mathcal{F}_i$ and $\mathcal{F}_j$ receive a higher similarity score only when more of their components have the same direction and the magnitudes of these components are closer.

3.3.2 Model Selection and Merging

Given $\mathcal{M}_k = (\mathrm{Sim}\langle\mathcal{F}_i, \mathcal{F}_j\rangle)_{i,j=1}^{k}$, we construct a complete undirected weighted graph $\mathcal{G} = (V, E)$, where each node $v_i \in V$ represents $\mathcal{F}_i \in \boldsymbol{\mathcal{F}}$ ($\boldsymbol{\mathcal{F}}$ denotes the set of historical models), and each edge $(v_i, v_j) \in E$ is weighted by $\mathrm{Sim}\langle\mathcal{F}_i, \mathcal{F}_j\rangle$. We then set the threshold c to the q-th quantile of the similarity scores in $\mathcal{M}_k$. Based on this threshold, we identify a maximum clique $\mathcal{G}^{'} = (V^{'}, E^{'}) \subseteq \mathcal{G}$ such that, for all $v_i, v_j \in V^{'}$, $\mathrm{Sim}\langle\mathcal{F}_i, \mathcal{F}_j\rangle \geq c$, and for every $v_l \in V \setminus V^{'}$, there exists at least one $v_m \in V^{'}$ such that $\mathrm{Sim}\langle\mathcal{F}_l, \mathcal{F}_m\rangle < c$. We only select models that are represented by the nodes in $V^{'}$ to perform model merging. In other words, we find a maximum clique containing the largest set of models that have higher similarity to each other than to the models outside the clique. We consider models outside the clique to be outliers and exclude them from model merging. We do this for two reasons according to our observations: (1) Fusion models change more significantly than LiDAR-based models during the TTA stage, making fusion models with low similarity more likely to exhibit performance gaps rather than complementarity. (2) Merging fusion models that are highly similar to each other can facilitate more stable knowledge transfer. We provide supporting visualization results and analysis in Appendix A.2. Let $\mathcal{I}' = \{i \mid v_i \in V^{'}\}$ denote the index set of the selected models. We denote the selected model set and its corresponding similarity submatrix by $\boldsymbol{\mathcal{F}}^{'} = \{\mathcal{F}_i \mid i \in \mathcal{I}^{'}\}$ and $\mathcal{M}^{'} = (\mathrm{Sim}\langle\mathcal{F}_i, \mathcal{F}_j\rangle)_{i,j\in\mathcal{I}'}$, respectively. Inspired by TIES-Merging Yadav et al. (2023) and CodeMerge Yang et al. (2026), we adopt the sign-consistent merging strategy to obtain the super model $\mathcal{F}^*$. Instead of the *disjoint mean* from TIES-Merging or *ridge leverage score* from CodeMerge, we use $\mathcal{M}^{'}$ to aggregate the model weights. The computation process can be formulated as follows:

$$\bar{\theta}_i = \sum_{i\in\mathcal{I}'} w_i \cdot \mathbb{I}\left[\mathrm{sign}\left(\theta_i\right) = \mathrm{sign}_{\mathrm{maj}}\right] \odot \theta_i, \quad w_i = \frac{\sum_{j\in\mathcal{I}'} \mathrm{Sim}\langle\mathcal{F}_i, \mathcal{F}_j\rangle}{\sum_{l\in\mathcal{I}'}\sum_{j\in\mathcal{I}'} \mathrm{Sim}\langle\mathcal{F}_l, \mathcal{F}_j\rangle}, \tag{2}$$

where $\mathbb{I}\left[\mathrm{sign}\left(\theta_i\right) = \mathrm{sign}_{\mathrm{maj}}\right]$ is a binary mask with the same shape of θ_i. For the k-th component θ_i^k of θ_i, the k-th component of the mask is set to 1 if $\mathrm{sign}(\theta_i^k)$ matches the majority sign $\mathrm{sign}_{\mathrm{maj}}$ of the k-th components across the models in $\boldsymbol{\mathcal{F}}^{'}$, and to 0 otherwise. $\odot$ denotes element-wise multiplication. $w_i \in [0, 1]$ weights θ_i, with a higher value indicating that θ_i has a greater influence on the model parameters of $\mathcal{F}^*$. With the increasing number of historical models in $\boldsymbol{\mathcal{F}}$, the GPU memory usage of the *task vector* memory bank $\mathcal{T}$ and the similarity matrix $\mathcal{M}$ may become a burden and eventually cause the GPU to run out of memory. Moreover, relying on the merged model to guide the fine-tuning of the current model for a long period may lead to overfitting. To address the above two problems, we periodically clear $\mathcal{M}$ and $\mathcal{T}$, and repeat the processes of initialization, accumulation, and model merging, as shown in Fig. 1.

4 Experiments

4.1 Datasets, Metrics, and Baselines

We use five widely adopted multimodal 3D object detection datasets: KITTI Geiger et al. (2013), nuScenes Caesar et al. (2020), Waymo Sun et al. (2020), KITTI-C Kong et al. (2023), and nuScenes-

C Kong et al. (2023). We conduct extensive experiments on five widely studied TTA tasks, including Waymo→KITTI (W→K), nuScenes→KITTI (N→K), Waymo→nuScenes (W→N), nuScenes→nuScenes-C (N→NC), and Waymo→KITTI-C (W→KC). KITTI-C and nuScenes-C are corrupted versions of KITTI and nuScenes, respectively. These corruptions include Fog, Wet conditions (Wet.), Snow, Motion blur (Moti.), Missing beams (Beam.), Crosstalk (Cross.T), Cross-sensor (Cross.S), and Incomplete echoes (Inc.). We adopt AP_{BEV} and AP_{3D} from the KITTI evaluation metric when the target domain dataset is KITTI or KITTI-C. We adopt mAP, NDS from the nuScenes evaluation metric when the target domain dataset is nuScenes or nuScenes-C. We also adopt *Closed Gap* metric from ST3D which is defined as *Closed Gap*=$\frac{AP_{\text{model}}-AP_{\text{No Adapt.}}}{AP_{\text{oracle}}-AP_{\text{No Adapt.}}} \times 100\%$. We compare our proposed MM-TTA with 12 baselines, including **No Adapt.**, **SN** Wang et al. (2020), **ST3D** Yang et al. (2021), **Tent** Wang et al. (2021), **CoTTA** Wang et al. (2022), **SAR** Niu et al. (2023), **MemCLR** VS et al. (2023), **DPO** Chen et al. (2024b), **Reg-TTA3D** Yuan et al. (2024),**MOS** Chen et al. (2025), **CodeMerge** Yang et al. (2026), and **Oracle**. **SN** and **ST3D** are pioneer offline domain adaptation methods. **Tent**, **CoTTA**, **SAR**, **MemCLR**, **DPO**, **MOS**, and **CodeMerge** are recent state-of-the-art TTA methods. **No Adapt.** refers to directly inferring the LiDAR-based pretrained model on the target domain. **Oracle** refers to the fully supervised LiDAR-based model trained on the target domain. We use SECOND Yan et al. (2018) as the detector following previous methods and BEV-Fusion Liu et al. (2023b) as the fusion architecture. We provide additional implementation details in Appendix A.3 and experiments on other 3D detectors and fusion architectures in Appendix A.6.

4.2 Main Results and Analysis

Cross-dataset Shifts. Table 1 lists the performance of our method with pioneering offline methods and recent TTA methods on W→K, N→K and W→N tasks. As shown, for the W→K task, we achieve the highest AP_{3D} of 70.09, with 3.78 improvements over CodeMerge, and significantly close the performance gap (106.90% and 92.69%) between No Adapt. and Oracle. Despite this, there is only a slight decrease of 0.25 points in AP_{BEV} compared with CodeMerge. For the N→K task, we consistently outperform CodeMerge in AP_{BEV} and AP_{3D} by improvements of 0.56 and 6.88 points, respectively. For the W→N task, we achieve the highest mAP and NDS (0.5117 and 0.6330). Especially for NDS, we improve it by 17.46% over No Adapt. (0.5389 → 0.6330) and by 15.24% over MemCLR (0.5493 → 0.6330). We provide the full results of the W→N task with more nuScenes metrics, including mATE, mASE, mAOE, mAVE, and mAAE, in Appendix A.4. We also provide the results on multiple classes in Appendix A.5. We present the visualization results on W→K task in Fig. 4. Blue boxes denote predictions, and green boxes denote ground truth. As shown, in case 1 (marked in red), previous methods all mistakenly predict a box at the center between the two ground-truth instances, while our method successfully predicts the two ground-truth instances. In case 2 (marked in yellow), our method predicts fewer false positive instances than DPO, MOS, and CodeMerge. Moreover, for the remaining instances predicted by our method and previous methods, our predictions have boundaries closer to the ground-truth instances.

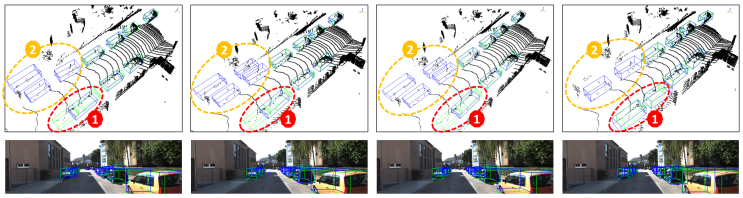

Figure 4: Visualization results on W→K task.

Corruption Shifts. Figure 5 lists the performance of our method and recent TTA methods on the N→NC task. As shown, we achieves the highest mean mAP/NDS of 0.3715/0.4690 across eight corruption types, surpassing the second-best DPO by 0.0166/0.0134. In terms of corruption type-wise performance, we consistently outperforms previous methods in both mAP and NDS in five situations, including Wet., Snow, Moti., Beam., and Inc.. Especially for Incomplete Echo (Inc.), we surpasses MOS by 0.0705 in mAP and CodeMerge by 0.0383 in NDS. We also achieve the highest NDS values of 0.3894 and 0.2691 in Fog and CrossS., respectively. There is only a slight decrease in CrossT., with mAP and NDS being 0.0193 and 0.0046 lower than those of DPO, respectively.

Cross-dataset and Corruption Shifts. Table 2 lists the performance of our method and recent TTA methods on the W→KC task. As shown, our method achieves the highest mean AP_{3D} (52.24/44.98/43.65) across eight corruption types at easy/moderate/hard difficulties, surpassing

MOS by $6.83/4.62/5.16$ points. In terms of corruption type-wise performance, our method achieves the highest AP_{3D} at all three difficulty levels in seven situations. Especially under CrossT, we achieve AP_{3D} of $73.78/63.70/64.19$, with improvements of $10.04/6.99/9.88$ points over MOS. Under Snow, although we fall behind MOS at the easy and moderate difficulty levels (by 2.60 and 1.17 points, respectively), we surpass MOS at the hard difficulty level by 0.28 points.

Table 1: TTA performance across W→K, N→K, and W→N tasks on the Car category. We report $\mathrm{AP}_{\mathrm{BEV}}/\mathrm{AP}_{3D}$ at moderate difficulty under the KITTI metric for W→K and N→K, and $\mathrm{mAP}/\mathrm{NDS}$ under the nuScenes metric for W→N. **Bold** = best; underline = second best.

Method	Venue	TTA	Waymo → KITTI		nuScenes → KITTI		Waymo → nuScenes
			$\mathrm{AP}_{\mathrm{BEV}}/\mathrm{AP}_{3D}$	Closed Gap	$\mathrm{AP}_{\mathrm{BEV}}/\mathrm{AP}_{3D}$	Closed Gap	mAP/NDS
No Adapt.	–	×	67.64 / 27.48	–	51.84 / 17.92	–	0.4739 / 0.5389
SN	CVPR'20	×	78.96 / 59.20	+72.33% / +69.00%	40.03 / 21.23	+37.55% / +5.96%	0.4817 / 0.5526
ST3D	CVPR'21	×	82.19 / 61.83	+92.97% / +74.72%	75.94 / 54.13	+76.63% / +65.21%	0.5003 / 0.5618
Oracle	–	×	83.29 / 73.45	–	83.29 / 73.45	–	0.6804 / 0.6593
Tent	ICLR'21	✓	65.09 / 30.12	-16.29% / +5.74%	46.90 / 18.83	-15.71% / +1.64%	0.5082 / 0.5486
CoTTA	CVPR'22	✓	67.46 / 35.34	-1.15% / +17.10%	68.81 / 47.61	+53.96% / +53.47%	0.4865 / 0.5363
SAR	ICLR'23	✓	65.81 / 30.39	-11.69% / +6.33%	61.34 / 35.74	+30.21% / +32.09%	0.5061 / 0.5462
MemCLR	WACV'23	✓	65.61 / 29.83	-12.97% / +5.11%	61.47 / 35.76	+30.62% / +32.13%	0.5099 / 0.5493
DPO	MM'24	✓	75.81 / 55.74	+52.20% / +61.47%	73.27 / 54.38	+68.13% / +65.66%	0.4797 / 0.5394
Reg-TTA3D	ECCV'24	✓	81.60 / 56.03	+89.20% / +62.11%	68.73 / 44.56	+53.70% / +47.97%	-
MOS	ICLR'25	✓	81.90 / 64.16	+91.12% / +79.79%	71.13 / 51.11	+61.33% / +59.78%	0.5018 / 0.5474
CodeMerge	NeurIPS'26	✓	**84.62** / 66.31	**+108.50%** / +84.47%	77.41 / 58.54	+81.30% / +73.15%	0.4987 / 0.5450
Ours	–	✓	84.37 / **70.09**	+106.90% / **+92.69%**	**77.97 / 65.42**	**+83.08% / +85.54%**	**0.5117 / 0.6330**

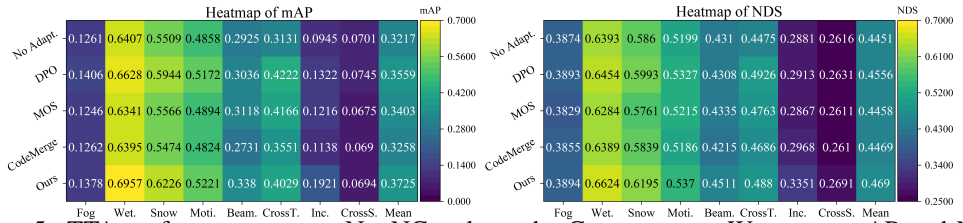

Figure 5: TTA performance across N→NC task on the Car category. We report mAP and NDS following the nuScenes metrics. Darker/lighter shades indicate lower/higher performance.

Table 2: TTA performance across W→KC task on the Car category. We report AP_{3D} at easy/moderate/hard difficulties of KITTI metric. **Bold** = best; underline = second best.

	No Adapt.	DPO	MOS	CodeMerge	**Ours**
Fog	20.25 / 17.94 / 16.85	56.16 / 50.44, 48.94	58.82 / 56.35 / 53.80	42.41 / 38.36 / 36.46	**65.91 / 58.01 / 55.45**
Wet.	30.76 / 27.74 / 26.31	62.92 / 56.35 / 53.16	70.10 / 63.51 / 60.78	67.92 / 61.32 / 56.02	**79.34 / 70.69 / 66.45**
Snow	12.44 / 13.03 / 11.40	41.33 / 40.50 / 39.06	**42.75 / 41.30** / 39.97	35.00 / 37.79 / 35.83	40.18 / 40.13 / **40.25**
Moti.	8.16 / 8.38 / 8.16	23.60 / 24.12 / 23.99	27.18 / 26.61 / 27.11	27.45 / 26.63 / 26.94	**31.39 / 32.79 / 33.81**
Beam.	18.25 / 13.57 / 12.53	48.58 / 37.06 / 36.34	45.93 / 36.05 / 34.22	39.68 / 31.80 / 28.40	**48.66 / 38.65 / 36.90**
CrossT.	23.13 / 21.71 / 20.03	61.49 / 56.46 / 53.31	63.74 / 56.71 / 54.31	53.64 / 50.14 / 49.55	**73.78 / 63.70 / 64.19**
Inc.	4.91 / 3.84 / 3.51	17.57 / 14.60 / 13.15	22.14 / 17.18 / 15.05	21.79 / 17.90 / 16.50	**34.61 / 25.10 / 23.64**
CrossS.	11.01 / 8.09 / 7.40	39.26 / 29.07 / 27.72	32.58 / 25.20 / 22.69	19.26 / 16.27 / 15.39	**44.05 / 30.74 / 28.49**
Mean	16.11 / 14.29 / 13.27	43.86 / 38.58 / 36.96	45.41 / 40.36 / 38.49	38.37 / 35.03 / 33.14	**52.24 / 44.98 / 43.65**

4.3 Ablation Study

Efficiency Analysis. We first investigates the effect of freezing different parts of the image branch and then uses the best-performing configuration to compare GPU Memory (GB) and latency (seconds per frame) with existing methods. We denote 'B', 'N', and 'V' as image backbone, neck, and view-transformation module, respectively. We denote 'freezing B', 'freezing B+N', and 'freezing B+N+V' in Fig. 6(a) as setting 1, setting 2, and setting 3, respectively. As shown, setting 1 and setting 2 outperform setting 3 by 1.13 and 5.04 points in AP_{3D}, respectively, showing the necessity of training the View-Trans module, as depth information shifts exist between domains. We choose setting 2 since it surpasses setting 1 by 3.91 points in AP_{3D}, indicating that training the neck may degrade performance. In terms of GPU memory usage, our method requires only slightly more GPU memory than previous methods (2.68–4.60 GB), and achieves lower latency than MOS (0.81 s/sample lower) and comparable latency to CodeMerge (0.06 s/sample higher).

Stability Analysis. We first evaluate our model merging strategy against other model merging strategies, and then compare our method with other methods in terms of AP_{3D} and throughput. In

Fig. 7(a) and (b), 'SW (Synergy Weights)' and 'RLS (Ridge Leverage Score)' are model merging strategies in MOS and CodeMerge, respectively, 'Cosine' refers to replacing the *sign consistency score* with the cosine score in CMM, and 'No CMM' refers to self-training without model merging. As shown in Fig. 7(a), choosing diverse models such as 'SW' and 'RLS' for model merging is suboptimal and quickly causes catastrophic forgetting, resulting in average AP_{3D} decreases of 30.0 and 19.33 points compared with CMM. Choosing consistent models with 'Cosine' alleviates catastrophic forgetting and surpasses 'No CMM' with an average AP_{3D} improvement of 3.32 points, but still shows lower performance compared with 'CMM', with an average AP_{3D} decrease of 1.63 points. 'SW' and 'RLS' even underperform 'No CMM' by 26.7 and 16.0 points in average AP_{3D}. As shown in Fig. 7(c), benefiting from GCA in the pre-training stage, we achieve nearly 16.23 higher

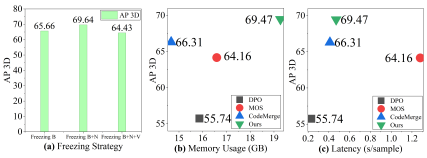

Figure 6: Efficiency analysis.

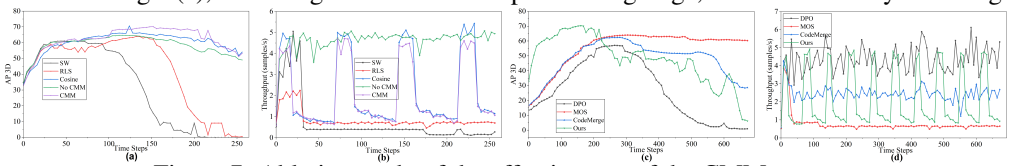

Figure 7: Ablation study of the effectiveness of the CMM strategy.

AP_{3D} than other methods at the beginning of the TTA stage. Moreover, we reach performance peak 80-144 time steps earlier than other methods and achieve the highest AP_{3D} of 70.33, indicating that our method can achieve better performance with fewer time steps. Fig. 7(d) shows the real-time throughput of different methods over time steps. Benefiting from the freezing strategy on the image branch and CMM to simplify model merging, the throughput of our method surpasses MOS by a large margin, with an average throughput gain of 1.34 samples/s. We also achieve comparable average throughput with CodeMerge, with only 0.32 fewer samples/s. DPO achieves the highest throughput with an average of 4.32 samples/s, but has significantly lower AP_{3D}.

Hyperparameter Analysis. We validate the effects of different combinations of hyperparameters $\mathcal{K}$

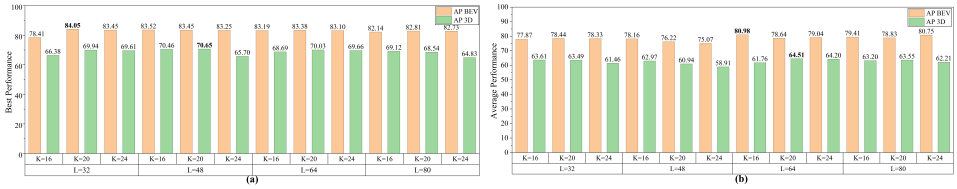

Figure 8: Ablation study of hyperparameters $\mathcal{K}$ and L of CMM.

and L in steps (II) and (III) of CMM. We report the best and average AP_{BEV}/AP_{3D} across time steps in Fig. 8(a) and (b), respectively. We achieve the highest best AP_{BEV} (84.05) when $L = 32$ and $\mathcal{K} = 20$, and the highest best AP_{3D} (70.65) when $L = 48$ and $\mathcal{K} = 20$ (as shown in Fig. 8(a)), indicating that models tend to achieve the best performance when L is relatively small ($L = 32$ or $L = 48$). However, models tend to present more stable performance when L is set to higher values. As shown in Fig. 8(b), we achieve the highest average AP_{BEV} (80.98) when $L = 64$ and $\mathcal{K} = 16$, and the highest average AP_{3D} when $L = 64$ and $\mathcal{K} = 20$. For the six combinations when L is set to 64 or 80, all of them achieve average AP_{BEV} higher than 78, and four of them achieve average AP_{3D} higher than 63. While for the six combinations when L is set to 32 or 48, only three of them achieve average AP_{BEV} higher than 78, and two of them achieve average AP_{3D} higher than 63. We provide additional analysis of the hyperparameter q and the projection dimension of the task vector in Appendix A.7 and A.8, respectively.

5 Conclusion

This paper is an early attempt to explore test-time adaptation for 3D object detection in the LiDAR-image fusion framework. We propose the GCA strategy to directly deploy the pretrained fusion model on the target domain without any modification of the model parameters while maintaining superior performance to LiDAR-based models. We further propose the CMM strategy to ensure effective and robust test-time fine-tuning while reducing the computational cost introduced by model merging. Extensive experiments demonstrate the effectiveness of our method. Future work could explore how to mitigate catastrophic forgetting and performance degradation of LiDAR-based or fusion-based models with increasing time steps.

AI USE STATEMENT

In this work, we did not use generative AI tools for any tasks requiring disclosure. Generative AI tools were used only for recommended disclosure tasks related to improving the grammatical correctness, clarity, and readability of the manuscript. All AI-assisted edits were carefully reviewed and verified by the authors. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, private data, or sensitive information. We only use publicly available datasets for evaluation. We do not identify any ethical concerns related to this research.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of our method, implementation details, and experimental settings in the main paper and supplementary text. The datasets used in our experiments and the corresponding data preprocessing procedures and evaluation protocols are publicly available. The source code and trained models will be released to facilitate future research and reproduction of our results.

REFERENCES

Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1090–1099, 2022.

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 *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11618–11628. IEEE, 2020.

Gyusam Chang, Wonseok Roh, Sujin Jang, Dongwook Lee, Daehyun Ji, Gyeongrok Oh, Jinsun Park, Jinkyu Kim, and Sangpil Kim. Cmda: Cross-modal and domain adversarial adaptation for lidar-based 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 972–980, 2024.

Shijie Chen, Junbao Zhuo, Xin Li, Haizhuang Liu, Rongquan Wang, Jiansheng Chen, and Huimin Ma. Cmt: Co-training mean-teacher for unsupervised domain adaptation on 3d object detection. In *Proceedings of the 32nd ACM international conference on multimedia*, pp. 4738–4747, 2024a.

Zhuoxiao Chen, Zixin Wang, Yadan Luo, Sen Wang, and Zi Huang. Dpo: dual-perturbation optimization for test-time adaptation in 3d object detection. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 4138–4147, 2024b.

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 *International Conference on Learning Representations*, volume 2025, pp. 92784–92808, 2025.

Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 1201–1209, 2021.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The international journal of robotics research*, 32(11):1231–1237, 2013.

Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and J Zico Kolter. Test time adaptation via conjugate pseudo-labels. *Advances in Neural Information Processing Systems*, 35:6204–6218, 2022.

Qianjiang Hu, Daizong Liu, and Wei Hu. Density-insensitive unsupervised domain adaption on 3d object detection. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17556–17566, 2023.

Junjie Huang, Guan Huang, Zheng Zhu, Yun Ye, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *arXiv preprint arXiv:2112.11790*, 2021.

Junjie Huang, Yun Ye, Zhujin Liang, Yi Shan, and Dalong Du. Detecting as labeling: Rethinking lidar-camera fusion in 3d object detection. In *European Conference on Computer Vision*, pp. 439–455. Springer, 2024.

Yecheol Kim, Junho Lee, Changsoo Park, Hyoung Won Kim, Inho Lim, Christopher Chang, and Jun Won Choi. Semi-supervised domain adaptation using target-oriented domain augmentation for 3d object detection. *IEEE Transactions on Intelligent Vehicles*, 10(1):562–572, 2024.

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 *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19937–19949. IEEE, 2023.

Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 12697–12705, 2019.

Yinhao Li, Han Bao, Zheng Ge, Jinrong Yang, Jianjian Sun, and Zeming Li. Bevstereo: Enhancing depth estimation in multi-view 3d object detection with temporal stereo. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 1486–1494, 2023a.

Yinhao Li, Zheng Ge, Guanyi Yu, Jinrong Yang, Zengran Wang, Yukang Shi, Jianjian Sun, and Zeming Li. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 1477–1485, 2023b.

Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: learning bird's-eye-view representation from lidar-camera via spatiotemporal transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

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.

Tingting Liang, Hongwei Xie, Kaicheng Yu, Zhongyu Xia, Zhiwei Lin, Yongtao Wang, Tao Tang, Bing Wang, and Zhi Tang. Bevfusion: A simple and robust lidar-camera fusion framework. *Advances in neural information processing systems*, 35:10421–10434, 2022.

Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection. In *European conference on computer vision*, pp. 531–548. Springer, 2022.

Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Aqi Gao, Tiancai Wang, and Xiangyu Zhang. Petrv2: A unified framework for 3d perception from multi-camera images. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3262–3272, 2023a.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9992–10002. Ieee, 2021.

Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *2023 IEEE international conference on robotics and automation (ICRA)*, 2023b.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *Internetional Conference on Learning Representations*, 2023.

Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *European conference on computer vision*, pp. 194–210. Springer, 2020.

Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10529–10538, 2020.

Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *International Journal of Computer Vision*, 131(2):531–551, 2023.

Sangyun Shin, Yuhang He, Madhu Vankadari, Ta-Ying Cheng, Qian Xie, Andrew Markham, and Niki Trigoni. Towards learning group-equivariant features for domain adaptive 3d detection. *Advances in Neural Information Processing Systems*, 37:18019–18035, 2024.

Ziying Song, Lei Yang, Shaoqing Xu, Lin Liu, Dongyang Xu, Caiyan Jia, Feiyang Jia, and Li Wang. Graphbev: Towards robust bev feature alignment for multi-modal 3d object detection. In *European conference on computer vision*, pp. 347–366. Springer, 2024.

Zhuoqun Su, Huimin Lu, Shuaifeng Jiao, Junhao Xiao, Yaonan Wang, and Xieyuanli Chen. Efficient multimodal 3d object detector via instance-level contrastive distillation. *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2443–2451. IEEE, 2020.

OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. `https://github.com/open-mmlab/OpenPCDet`, 2020.

Vibashan VS, Poojan Oza, and Vishal M Patel. Towards online domain adaptive object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 478–488, 2023.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7201–7211, 2022.

Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. Train in germany, test in the usa: Making 3d object detectors generalize. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11713–11723, 2020.

Yan Wang, Junbo Yin, Wei Li, Pascal Frossard, Ruigang Yang, and Jianbing Shen. Ssda3d: Semi-supervised domain adaptation for 3d object detection from point cloud. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 2707–2715, 2023.

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.

Yi Wei, Zibu Wei, Yongming Rao, Jiaxin Li, Jie Zhou, and Jiwen Lu. Lidar distillation: Bridging the beam-induced domain gap for 3d object detection. In *European Conference on Computer Vision*, pp. 179–195. Springer, 2022.

Maciej K Wozniak, Mattias Hansson, Marko Thiel, and Patric Jensfelt. Uada3d: Unsupervised adversarial domain adaptation for 3d object detection with sparse lidar and large domain gaps. *IEEE Robotics and automation letters*, 9(12):11210–11217, 2024.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

Chenyu Yang, Yuntao Chen, Hao Tian, Chenxin Tao, Xizhou Zhu, Zhaoxiang Zhang, Gao Huang, Hongyang Li, Yu Qiao, Lewei Lu, et al. Bevformer v2: Adapting modern image backbones to bird's-eye-view recognition via perspective supervision. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17830–17839, 2023a.

Huitong Yang, Zhuoxiao Chen, Fengyi Zhang, Zi Huang, and Yadan Luo. Codemerge: Codebook-guided model merging for robust test-time adaptation in autonomous driving. *Advances in Neural Information Processing Systems*, 38:49096–49123, 2026.

Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d: Self-training for unsupervised domain adaptation on 3d object detection. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10368–10378, 2021.

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*, 45(5):6354–6371, 2022a.

Lei Yang, Kaicheng Yu, Tao Tang, Jun Li, Kun Yuan, Li Wang, Xinyu Zhang, and Peng Chen. Bevheight: A robust framework for vision-based roadside 3d object detection. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21611–21620, 2023b.

Lei Yang, Tao Tang, Jun Li, Kun Yuan, Kai Wu, Peng Chen, Li Wang, Yi Huang, Lei Li, Xinyu Zhang, et al. Bevheight++: Toward robust visual centric 3d object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025a.

Zeyu Yang, Jiaqi Chen, Zhenwei Miao, Wei Li, Xiatian Zhu, and Li Zhang. Deepinteraction: 3d object detection via modality interaction. *Advances in Neural Information Processing Systems*, 35:1992–2005, 2022b.

Zeyu Yang, Nan Song, Wei Li, Xiatian Zhu, Li Zhang, and Philip HS Torr. Deepinteraction++: Multi-modality interaction for autonomous driving. *IEEE transactions on pattern analysis and machine intelligence*, 2025b.

Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11784–11793, 2021.

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 *European conference on computer vision*, pp. 197–213. Springer, 2024.

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.

Yanan Zhang, Chao Zhou, and Di Huang. Stal3d: Unsupervised domain adaptation for 3d object detection via collaborating self-training and adversarial learning. *IEEE Transactions on Intelligent Vehicles*, 2024.

Shenao Zhao, Pengpeng Liang, and Zhoufan Yang. Multi-modal assistance for unsupervised domain adaptation on point cloud 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 13226–13234, 2026.

A APPENDIX

This Appendix provides additional details, experiments, and analysis of our method, summarized as follows:

- **Sec. A.1:** Analysis of the differences between the cosine score and our proposed *sign consistency score* and the impacts of these differences.
- **Sec. A.2:** Visualization results and further analysis supporting consistency-based model merging strategy.
- **Sec. A.3:** Implementation details on training schedules (A.3.1), data and environment configurations (A.3.2), and optimization and adaptation settings (A.3.3).
- **Sec. A.4:** Full results on transfer task: Waymo→nuScenes.
- **Sec. A.5:** Results across multiple classes (Car, Pedestrian, and Cyclist) on three transfer tasks: Waymo→KITTI, nuScenes→KITTI, and Waymo→KITTI-C to address cross-dataset shifts and cross-dataset corruption shifts.
- **Sec. A.6:** Results on different 3D detectors and fusion architectures.
- **Sec. A.7:** Sensitivity analysis of the hyperparameter q.
- **Sec. A.8:** Sensitivity analysis of the projection dimension of the task vector.

A.1 DIFFERENCES BETWEEN COSINE AND SIGN CONSISTENCY SCORES AND THEIR IMPACTS

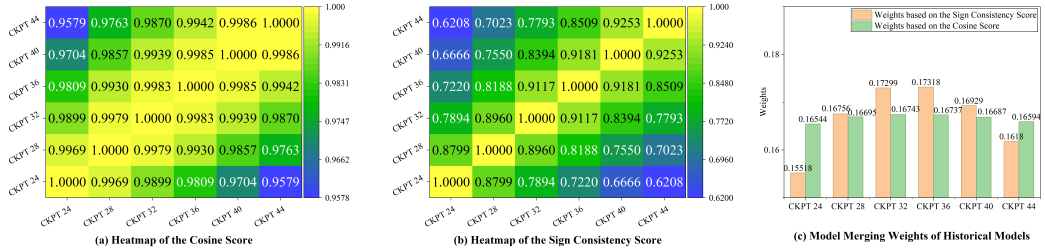

Figure 9: Visualization of the cosine score and *sign consistency score* between model pairs and the model-merging weights of historical models based on these two scores.

Fig. 9 presents the visualizations of the differences between the cosine score and the *sign consistency score* and their impacts on model merging. We denote 'CKPT 24', 'CKPT 28', 'CKPT 32', 'CKPT 36', 'CKPT 40', and 'CKPT 44' as the historical models involved in model merging at time step 44. The interval of 4 between consecutive models corresponds to the batch size of 4. As shown in Fig. 9(a), all model pairs have cosine scores greater than 0.95, which indeed demonstrates that these historical models involved in model merging are mutually consistent. However, highly similar cosine scores between model pairs (*e.g.,* the difference between the maximum and minimum cosine scores is 0.0406) result in highly similar model-merging weights across models. As shown in Fig. 9(c), the model-merging weights computed using the cosine score range from 0.16544 to 0.16743, which could be considered almost identical. This may be suboptimal for the merged model if the models involved in merging share almost identical merging weights. Ideally, among these consistent models, we expect those with higher similarity scores to other models to have higher merging weights, while those with lower similarity scores to other models should have lower merging weights. Our proposed *Sign Consistency Score* can effectively handle this problem. It ensures that model pairs sharing similar patterns obtain relatively high scores, while enlarging the score gap for model pairs with different patterns by assigning them much lower scores. As shown in Fig. 9(b), only three pairs ($\langle$CKPT 32, CKPT 36$\rangle$, $\langle$CKPT 36, CKPT 40$\rangle$, and $\langle$CKPT 40, CKPT 44$\rangle$) have sign consistency scores greater than 0.9, and the difference between the maximum and minimum *sign consistency scores* is 0.3045. For example, CKPT 36 has the highest *sign consistency scores* with other models ($\sum_{j\in\{24,28,32,36,40,44\}} \mathrm{Sim}\langle$CKPT 36, CKPT j$\rangle$=5.2215), so CKPT 36 has the highest model-merging weight of 0.17318, as shown in Fig. 9(c). CKPT 24 has the lowest *sign consistency scores* to other models ($\sum_{j\in\{24,28,32,36,40,44\}} \mathrm{Sim}\langle$CKPT 24, CKPT j$\rangle$=4.6787), so CKPT

24 has the lowest model-merging weight of 0.15518, as shown in Fig. 9(c). Results in Fig. 7(a) support our observation that replacing the *sign consistency score* with the cosine score causes a 1.63-point decrease in AP_{3D}.

A.2 Analysis of the Consistency-Based Merging Strategy

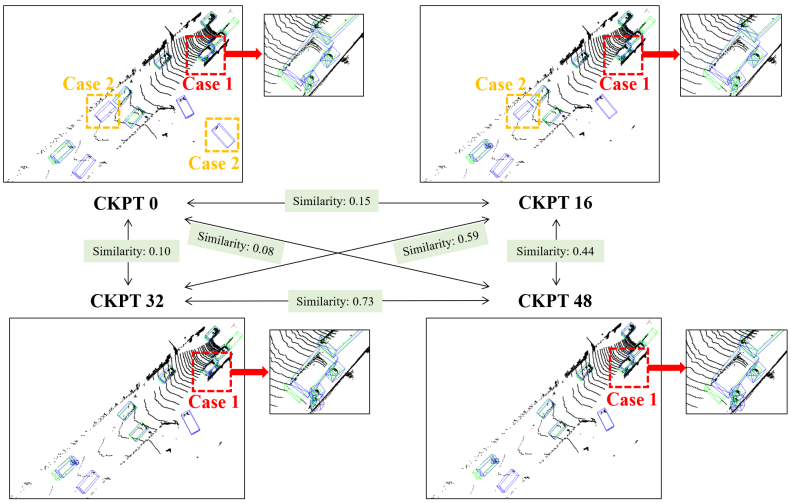

Figure 10: Visualization of performance gaps among diverse historical models.

We provide the visualization of performance gaps among diverse historical fusion models in Fig. 10. We denote 'CKPT 0', 'CKPT 16', 'CKPT 32', and 'CKPT 48' as historical models at time steps 0, 16, 32, and 48, respectively. Blue boxes denote predictions, and green boxes denote ground truth. As shown, both CKPT 0 and CKPT 16 fail to predict a Pedestrian (marked in red), while CKPT 32 and CKPT 48 successfully predict all three Pedestrians in case 1. In case 2, CKPT 0 produces one more false positive instance than CKPT 16, and two more false positives instances than CKPT 32 and CKPT 48. Although these four models have *sign consistency scores* of less than 0.75 with each other, they do not exhibit complementarity but rather performance gaps. Notably, models at early time steps tend to predict fewer bounding boxes that are consistent with the ground truth and produce more noisy boxes. Therefore, selecting diverse historical fusion models results in degradation of the merged model, as models with lower performance interfere with those with higher performance. The ablation study in Fig. 7 supports our observation, as replacing our model merging strategy with 'SW' and 'RLS', which seek model diversity, leads to inferior performance.

A.3 Implementation Details

We build our code upon the OpenPCDet Team (2020) point cloud detection framework. This framework supports a series of LiDAR-based 3D object detection methods (SECOND Yan et al. (2018), VoxelRCNN Deng et al. (2021), CenterPoint Yin et al. (2021), PointPillar Lang et al. (2019), and PVRCNN Shi et al. (2020)) and LiDAR-image fusion-based 3D object detection methods (BEVFusion Liu et al. (2023b), GraphBEV Song et al. (2024), and ICDFusion Su et al. (2025)).

A.3.1 Training Schedules

In the first pre-training stage, following the training procedure of BEVFusion, we first pretrain SECOND Yan et al. (2018) as the LiDAR-based model, and then pretrain BEVFusion with the pretrained SECOND model and the publicly available pretrained Swin-Tiny Liu et al. (2021) image backbone. For the Waymo→KITTI and Waymo→KITTI-C tasks, we pretrain SECOND Yan et al. (2018) on two NVIDIA RTX 4090 GPUs with a total batch size of 16 (8 per GPU) for 30 epochs and pretrain BEVFusion on four NVIDIA RTX 4090 GPUs with a total batch size of 4 (1 per GPU) for 30 epochs. For the nuScenes→nuScenes-C and Waymo→nuScenes tasks, we pretrain SECOND on two NVIDIA RTX 4090 GPUs with a total batch size of 16 (8 per GPU) for 30 epochs and pretrain BEVFusion on four NVIDIA RTX 4090 GPUs with a total batch size of 8 (2 per GPU) for

30 epochs. For the nuScenes→KITTI task, we pretrain SECOND on two NVIDIA RTX 4090 GPUs with a total batch size of 16 (8 per GPU) for 50 epochs and pretrain BEVFusion on four NVIDIA RTX 4090 GPUs with a total batch size of 8 (2 per GPU) for 30 epochs.

In the second test-time adaptation stage, we fine-tune the pretrained BEVFusion on one NVIDIA RTX 4090 GPU with a batch size of 4 for 1 epoch.

A.3.2 Data and Environment Configuration

Data Configuration. Waymo uses a 64-beam LiDAR and four 200-beam LiDAR sensors to collect point clouds, and five surrounding cameras (Front, Front Right, Front Left, Side Right, and Side Left) to collect images. nuScenes/nuScenes-C use a 32-beam LiDAR to collect point clouds and six surrounding cameras (Front, Front Right, Front Left, Back Right, Back Left, and Back) to collect images. KITTI/KITTI-C use a 64-beam LiDAR to collect point clouds and a stereo camera to collect images. In order to make sure the pretrained BEVFusion model is directly applicable on the target domain without modification to the number of model parameters, the GCA strategy adjusts the input of the source-domain point clouds and images in the pre-training stage. For the Waymo→KITTI and Waymo→KITTI-C tasks, for each source-domain multimodal input data from the Waymo dataset, we set the point cloud range as $[0\,\text{m}, -40\,\text{m}, -3\,\text{m}, 70.4\,\text{m}, 40\,\text{m}, 1\,\text{m}]$, and select the entire Front, Front Right, and Front Left images and half of the Side Right and Side Left images for fusion pre-training. For the nuScenes→KITTI task, for each source-domain multimodal input data from the nuScenes dataset, we set the point cloud range as $[0\,\text{m}, -40\,\text{m}, -3\,\text{m}, 70.4\,\text{m}, 40\,\text{m}, 1\,\text{m}]$, and select the entire Front, Front Right, and Front Left images for fusion pre-training. For the Waymo→nuScenes task, for each source-domain multimodal input data from the Waymo dataset, we set the the point cloud range as $[-54.0\,\text{m}, -54.0\,\text{m}, -5.0\,\text{m}, 54.0\,\text{m}, 54.0\,\text{m}, 4.8\,\text{m}]$, and select the entire Front, Front Right, Front Left, Side Right and Side Left images for fusion pre-training. For the nuScenes→nuScenes-C task, as these two domains share the same camera and point cloud configurations, we simply follow the official settings of BEVFusion.

Environment Configuration. We conduct all experiments using Python 3.8.20, PyTorch 2.1.0, CUDA 12.1, OpenPCDet 0.6.0, and spconv 2.3.6 on an Ubuntu 20.04.6 system.

A.3.3 Optimization and Adaptation Settings

Hyperparameter Settings. We set $\mathcal{K}$ to 20, L to 32, and q to 0.6 for all tasks. We adopt the Adam optimizer with a learning rate of 1×10^{-3}.

Data Augmentations. In the first pre-training stage, we adopt random world rotation, random world flip, random world scaling, and random world translation data augmentation strategies from BEVFusion Liu et al. (2023b) for all tasks. In the second test-time adaptation stage, we adopt random object scaling and random world scaling from MOS Chen et al. (2025) for all tasks.

Fine-tuning techniques In the second test-time adaptation stage, we directly apply the pseudo-labeling strategy from ST3D Yang et al. (2021) and ST3D++ Yang et al. (2022a).

A.4 Full Results On Waymo→NuScenes Task

Table 3 lists the full results of our method and previous methods on the W→N task, including mAP, NDS, mATE, mASE, mAOE, mAVE, and mAAE according to the nuScenes metrics. As shown, we achieve the highest mAP and NDS (0.5117 and 0.6330). Especially for NDS, we improve it by 17.46% over No Adapt. ($0.5389 \rightarrow 0.6330$) and by 15.24% over the second best MemCLR ($0.5493 \rightarrow 0.6330$). We also achieve the lowest mASE, mAOE, mAVE, and mAAE (0.9199, 0.9173, 0.9444, and 0.9179). For mAVE, we reduce it by 15.86% compared with CodeMerge ($1.1224 \rightarrow 0.9444$). For mAAE, we reduce it by 2.38% compared with MemCLR ($0.9403 \rightarrow 0.9179$).

A.5 Results Across Multiple Classes

Table 4 lists the performance of our method and previous methods on the Waymo→KITTI task across multiple classes. As shown, for the Car category, we achieve the highest AP_{3D} of 70.09, with an improvement of 3.78 points over CodeMerge. For the Pedestrian category, we achieve the highest AP_{BEV} and AP_{3D} of 61.38 and 57.68, with improvements of 17.72% ($52.14 \rightarrow 61.38$) and

Table 3: TTA performance across W→N task. We report mAP, NDS, mATE, mASE, mAOE, mAVE, and mAAE following the nuScenes metrics. **Bold**=best;underline=second best.

Method	Venue	TTA	mAP ↑	NDS ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
No Adapt.	–	×	0.4739	0.5389	0.9199	0.9222	0.9135	1.1275	0.9425
SN	CVPR'20		0.4817	0.5526	0.9179	0.9169	0.9129	1.1277	0.9406
ST3D	CVPR'21		0.5003	0.5618	0.9174	0.9173	0.9109	1.1349	0.9432
Oracle	–		0.6804	0.6593	0.9169	0.9145	0.9069	1.1558	0.9426
Tent	ICLR'21	✓	0.5082	0.5486	0.9222	0.9231	0.9197	1.1268	0.9406
CoTTA	CVPR'22		0.4865	0.5363	0.9228	0.9223	0.9202	1.1264	0.9417
SAR	ICLR'23		0.5061	0.5462	0.9227	0.9226	0.9210	1.1268	0.9404
MemCLR	WACV'23		0.5099	0.5493	0.9224	0.9227	0.9206	1.1265	0.9403
DPO	MM'24		0.4797	0.5394	**0.9215**	0.9207	0.9174	1.1246	0.9410
MOS	ICLR'25		0.5018	0.5457	0.9227	0.9212	0.9196	1.1283	0.9417
CodeMerge	NeurIPS'26		0.4987	0.5450	0.9225	0.9212	0.9191	1.1224	0.9415
Ours	–		**0.5117**	**0.6330**	0.9234	**0.9199**	**0.9173**	**0.9444**	**0.9179**

19.92% (48.10 → 57.68) over CodeMerge, respectively. For the Cyclist category, we achieve the highest $\mathrm{AP}_{\mathrm{BEV}}$ of 55.69, with an improvement of 1.01 compared over CodeMerge.

Table 4: Results on Waymo → KITTI across multiple classes.

Task	Method	Car	Pedestrian	Cyclist
Waymo → KITTI	No Adapt.	67.64 / 27.48	46.29 / 43.13	48.61 / 43.84
	SN	78.96 / 59.20	53.72 / 50.44	44.61 / 41.43
	ST3D	82.19 / 61.83	52.92 / 48.33	53.73 / 46.09
	Oracle	83.29 / 73.45	46.64 / 41.33	62.92 / 60.32
	Tent	65.09 / 30.12	46.21 / 43.24	46.59 / 41.72
	CoTTA	67.46 / 35.34	49.02 / 45.03	55.31 / 50.07
	SAR	65.81 / 30.39	47.99 / 44.72	47.14 / 41.17
	MemCLR	65.61 / 29.83	48.25 / 44.84	47.49 / 41.67
	MOS	81.90 / 64.16	50.25 / 46.51	54.12 / **51.36**
	CodeMerge	**84.62** / 66.31	52.14 / 48.10	54.68 / 48.11
	Ours	84.37 / **70.09**	**61.38 / 57.68**	**55.69** / 49.20

Table 5 lists the performance of our method and previous methods on the nuScenes→KITTI task across multiple classes. As shown, our method achieves the highest $\mathrm{AP}_{\mathrm{BEV}}/\mathrm{AP}_{\mathrm{3D}}$ of $77.97/65.42$, $55.71/49.64$, and $25.50/23.12$ across the three categories, respectively. Especially for the Pedestrian category, our method surpasses the second-best MOS by 29.74% (42.94 → 55.71) and 29.08% (38.45 → 49.63) in $\mathrm{AP}_{\mathrm{BEV}}$ and $\mathrm{AP}_{\mathrm{3D}}$, respectively.

Table 6 lists the performance of our method and previous methods on the Waymo→KITTI-C task across multiple corruption types for the Pedestrian category. As shown, we achieve the highest mean AP_{3D} $(45.84/41.57/38.92)$ across eight corruption types at easy/moderate/hard difficulty levels. In terms of corruption type-wise performance, our method achieves the highest $\mathrm{AP}_{\mathrm{3D}}$ at all three difficulty levels across eight corruption types. Especially under the Incomplete Echo (Inc.) situation, we achieve $\mathrm{AP}_{\mathrm{3D}}$ of $59.83/54.83/51.66$, with improvements of $7.91/6.31/6.38$ points over CodeMerge.

A.6 Results on Different 3D Detectors and Fusion Architectures

Table 7 lists the performance of our method and previous methods on the Waymo→KITTI-C task under the BEVFusion Liu et al. (2023b) architecture with VoxelRCNN Deng et al. (2021) and CenterPoint Yin et al. (2021) detectors for the Car category. VoxelRCNN is an anchor-based detector, while CenterPoint is an anchor-free detector. As shown, for VoxelRCNN, we achieve the highest AP_{BEV} and AP_{3D} of 83.07 and 71.29, and surpass MOS by 2.15 and 6.51 points, respectively. For CenterPoint, we achieve the highest AP_{BEV} and AP_{3D} of 81.12 and 63.61, surpassing MOS

Table 5: Results on nuScenes → KITTI across multiple classes.

Task	Method	Car	Pedestrian	Cyclist
nuScenes → KITTI	No Adapt.	51.84 / 17.92	39.95 / 34.57	17.61 / 11.17
	SN	40.03 / 21.23	38.91 / 34.36	11.11 / 5.67
	ST3D	75.94 / 54.13	44.00 / 42.60	29.58 / 21.21
	Oracle	83.29 / 73.45	46.64 / 41.33	62.92 / 60.32
	Tent	46.90 / 18.83	39.69 / 35.53	19.64 / 10.49
	CoTTA	68.81 / 47.61	40.90 / 36.64	20.68 / 13.81
	SAR	65.81 / 30.39	39.78 / 35.63	19.56 / 10.26
	MemCLR	65.61 / 29.83	40.31 / 35.14	19.21 / 9.66
	MOS	71.13 / 51.11	42.94 / 38.45	21.65 / 18.16
	CodeMerge	77.41 / 58.54	38.58 / 35.33	24.53 / 19.70
	Ours	**77.97 / 65.42**	**55.71 / 49.63**	**25.50 / 23.12**

Table 6: TTA performance on the W→KC task for the Pedestrian category. We report $\mathrm{AP}_{\mathrm{3D}}$ at easy/moderate/hard difficulties of KITTI metric.

	No Adapt.	DPO	MOS	CodeMerge	**Ours**
Fog	30.48 / 26.15 / 23.61	32.70 / 28.64 / 26.06	32.38 / 27.51 / 24.85	31.82 / 27.94 / 25.24	**32.76 / 30.20 / 27.82**
Wet.	49.10 / 44.44 / 41.74	51.07 / 47.45 / 44.14	51.34 / 46.88 / 43.16	52.46 / 48.62 / 45.22	**58.89 / 53.54 / 51.00**
Snow	47.22 / 42.26 / 39.19	49.59 / 43.28 / 40.86	49.42 / 44.82 / 40.79	49.25 / 43.29 / 40.27	**53.37 / 49.03 / 45.34**
Moti.	27.18 / 25.02 / 23.29	26.02 / 24.94 / 23.07	27.54 / 25.93 / 23.86	26.21 / 24.51 / 23.22	**33.92 / 30.77 / 29.32**
Beam.	32.47 / 27.89 / 25.27	33.45 / 30.34 / 30.36	34.02 / 29.73 /27.35	33.24 / 29.23 / 28.85	**39.66 / 35.04 / 33.03**
CrossT.	47.42 / 43.08 / 40.37	51.13 / 47.17 / 43.53	50.78 / 45.42 / 41.73	50.58 / 45.40 / 43.00	**56.42 / 52.06 / 49.03**
Inc.	49.28 / 44.79 / 42.21	50.42 / 46.99 / 43.89	51.06 / 46.83 / 43.31	51.92 / 48.52 / 45.28	**59.83 / 54.83 / 51.66**
CrossS.	22.46 / 18.40 / 16.08	27.90 / 24.57 / 23.27	26.74 / 22.94 / 20.89	28.62 / 24.22 / 23.05	**31.86 / 27.12 / 24.12**
Mean	38.20 / 34.00 / 31.47	40.29 / 36.67 / 34.40	40.41 / 36.26 / 33.24	40.51 / 36.47 / 34.27	**45.84 / 41.57 / 38.92**

by 3.60 points in AP_{BEV} and CodeMerge by 2.90 points in AP_{3D}. These results demonstrate the effectiveness and generality of our method on different types of detectors.

Table 8 lists the performance of our method on the Waymo→KITTI task across different architectures using the SECOND Yan et al. (2018) detector for the Car category. As shown, benefitting from the GCA strategy, all three architectures (BEVFusion, GraphBEV, and ICDFusion) achieve better performance than No Adapt. in the pre-training stage, with $1.66/2.56$, $0.60/0.82$, and $0.29/0.30$ improvements over No Adapt. in $\mathrm{AP}_{\mathrm{BEV}}/\mathrm{AP}_{\mathrm{3D}}$, respectively. This demonstrates the generalizability of our GCA strategy under different fusion architectures. In the TTA stage, BEVFusion achieves the highest $\mathrm{AP}_{\mathrm{BEV}}/\mathrm{AP}_{\mathrm{3D}}$, with significant improvements of $15.07/40.05$ over the results in the pre-training stage. However, for GraphBEV and ICDFusion, the improvements are limited. GraphBEV outperforms the results in the pre-training stage by $0.59/18.06$ in $\mathrm{AP}_{\mathrm{BEV}}/\mathrm{AP}_{\mathrm{3D}}$. ICDFusion outperforms the results in the pre-training stage by 7.22 in $\mathrm{AP}_{\mathrm{3D}}$, but shows a decrease of 2.32 points in $\mathrm{AP}_{\mathrm{BEV}}$. We attribute the performance discrepancy to the design of the view transformation module in the image branch and the fusion block. For BEVFusion, it lifts the 2D image features into 3D spatial features using the LSS Philion & Fidler (2020) algorithm and BEVPool Huang et al. (2021) to reduce the computational cost of transformation. The fusion block directly concatenates the image and LiDAR BEV features and uses a convolutional layer to extract the fused features. This simple yet effective design makes BEVFusion less sensitive to cross-domain shifts, thereby achieving the best performance. GraphBEV and ICDFusion introduce more complicated designs in the transformation module (*e.g.,* graph-based neighbor information construction in GraphBEV) and the fusion block (*e.g.,* cross linear attention fusion in ICDFusion). These techniques enable them to achieve better performance in a single domain, but make them more vulnerable to cross-domain shifts.

A.7 Sensitivity Analysis of the Hyperparameter q

Fig. 11 presents the sensitivity analysis of hyperparameter q on $\mathrm{AP}_{\mathrm{3D}}/\mathrm{AP}_{\mathrm{BEV}}$, the number of models for merging, and throughput. As shown in Fig. 11(a), we achieve the highest $\mathrm{AP}_{\mathrm{BEV}}$ of 84.37 when q is set to 0.6 and the highest AP of 70.26 when q is set to 0.7. Note that a q value lower

Table 7: Results on Waymo → KITTI using VoxelRCNN and CenterPoint Detectors. We report AP_{BEV}/AP_{3D}.

Model	No Adapt	DPO	MOS	CodeMerge	Ours
VoxelRCNN	63.47 / 26.95	78.48 / 60.21	80.92 / 64.78	77.32 / 62.73	**83.07 / 71.29**
CenterPoint	51.09 / 18.97	73.14 / 52.22	77.52 / 60.26	75.29 / 60.71	**81.12 / 63.61**

Table 8: Results of the pre-training stage and the TTA stage on Waymo→KITTI using different architectures. We report AP_{BEV}/AP_{3D}. Note that **No Adapt.** refers to directly using the LiDAR-based pretrained model for inference on the target domain, which does not involve the TTA stage.

Stage	No Adapt.	BEVFusion	GraphBEV	ICDFusion
Pre-training	67.64 / 27.48	69.30 / 30.04	68.24 / 28.30	67.93 / 27.78
TTA	- / -	**84.37 / 70.09**	68.83 / 46.36	65.61 / 35.00

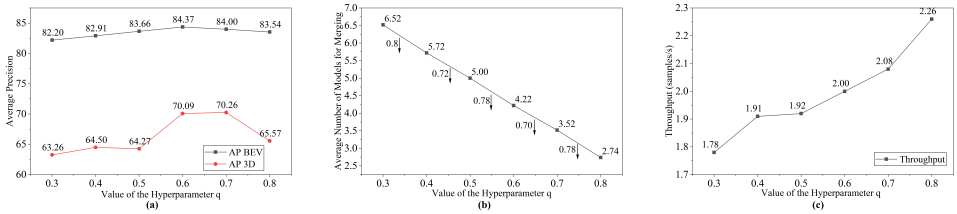

Figure 11: Sensitivity analysis of q on $\mathrm{AP}_{\mathrm{3D}}/\mathrm{AP}_{\mathrm{BEV}}$, the number of models for merging, and throughput.

than 0.6 or higher than 0.7 causes performance degradation. As shown in Fig. 11(b), when q is set to 0.5, 0.4, and 0.3, there are 0.78, 1.5, and 2.3 more models on average involved in merging compared with when q is set to 0.6. These extra models are outliers when q is set to 0.6, but participate in model merging when q is set to values lower than 0.6, which inevitably introduces inconsistent models and causes interference with the merged model. When q is set to 0.8, there are 1.48 fewer models on average for model merging, and the merged model becomes less stable without sufficient consistent models. As shown in Fig. 11(c), setting a q value lower than 0.6 also slows down the inference speed and decreases throughput, with reductions of about 0.08, 0.09, and 0.22 samples per second when q is set to 0.5, 0.4, and 0.3, respectively.

A.8 Sensitivity Analysis of the Projection Dimension of the Task Vector

Table 9: Results on Waymo → KITTI using projection matrices of different dimensions for the task vector. We report AP_{BEV}/AP_{3D}.

Dimension	512	1024	2048	4096	8192
AP_{BEV}/AP_{3D}	83.22 / 69.31	83.39 / 70.12	**84.37 / 70.09**	83.58 / 69.77	84.05 / 70.67

Table 9 lists the performance of using projection matrices of different dimensions for the task vector during the computation of the *sign consistency score*. As shown, we achieve the highest $\mathrm{AP}_{\mathrm{BEV}}$ and $\mathrm{AP}_{\mathrm{3D}}$ when we project the task vector into 2048 dimensions. Note that projecting the task vector into other dimensions only causes slight fluctuations in the final results, demonstrating that our method is robust to different projection dimensions.