Contents

6. IDEAL-M3D: Further Details

6.1. Problem Statement

6.1.1 Monocular 3D Object Detection (M3D).

Monocular 3D Detection (M3D) predicts categories and 3D bounding boxes $\mathcal{B}_i$ for objects in an RGB image I with intrinsics $K \in \mathbb{R}^{3\times4}$. Each $\mathcal{B}_i$ is parameterized by position $(x_i, y_i, z_i) \in \mathbb{R}^3$, dimensions $(w_i, h_i, l_i) \in \mathbb{R}^3$, orientation $R_i \in SO(3)$, and class $c_i \in \mathbb{N}$. Given a dataset $\{(I_j, K_j, \mathcal{B}(I_j))\}_{j=1}^{M}$, with $\mathcal{B}(I_j)$ as ground-truth boxes for image I_j, the goal is to train a model capable of predicting $\mathcal{B}(I)$ for any given image I. Starting solely from a 2D RGB image poses a significant challenge due to depth ambiguity.

6.1.2 Active Learning.

Active Learning (AL) starts with a small labeled dataset $\mathcal{D}_L^0 = \{(I_i, K_i, \mathcal{B}(I_i))\}$ and a large unlabeled dataset $\mathcal{D}_U = \{(I_j, K_j)\}$. In each round r, we select $\mathcal{D}_r^* = \{(I_j, K_j, \mathcal{S}_{j,r}^*)\}_{j\in\mathcal{J}_r^*}$, where $\mathcal{J}_r^*$ is the set of selected images and $\mathcal{S}_{j,r}^* \subseteq \hat{\mathcal{B}}(I_j)$ the subset of bounding boxes chosen for labeling. The oracle $\Omega : (I, \hat{\mathcal{B}}) \mapsto \mathcal{B} \cup \{\text{null}\}$, which in practice corresponds to a human annotator, refines each selected $\hat{\mathcal{B}}_{j,k} \in \mathcal{S}_{j,r}^*$, returning $\mathcal{B}_{j,k}$ or, in case it is not labeled, null. The labeled dataset is updated as $\mathcal{D}_L^r = \mathcal{D}_L^0 \cup \bigcup_{r'=1}^{r}\{(I_j, K_j, \{\mathcal{B}_{j,k}\}_{k\in\mathcal{S}_{j,r'}^*}) \mid \hat{\mathcal{B}}_{j,k} \neq \text{null}\}$, and the model is fine-tuned on $\mathcal{D}_L^r$. This repeats until the total labeled bounding boxes reach the budget $\mathcal{T} = \sum_{r=1}^{R}\sum_{j\in\mathcal{J}_r^*}|\mathcal{S}_{j,r}^*|$. In summary, AL pipelines can be seen as iterative cycles that repeat two steps: data selection for labeling, and training.

6.2. Loss Functions

We detail the loss computations for our baseline methods and omit the loss weights for clarity.

MonoLSS [23]:

$$L = L_{cls} + L_{c,o} + L_{h,w} + L_{S_{3d}} + L_\theta + L_{depth} \cdot \text{Sample } S \tag{12}$$

MonoCon [27]:

$$L = L_{cls} + L_{c,o} + L_{h,w} + L_{depth} + L_{S_{3d}} + L_\theta + L_{kp,h} + L_{kp,o} + L_{kp,co} \tag{13}$$

The individual losses can be categorized into image- and object-related losses.

6.2.1 Image-level Losses

The following losses are image-specific. Therefore, we apply the masking strategy as described in *cf*. Sec. 3.1 on both of these losses:

- L_{cls}: Gaussian kernel weighted focal loss for classification, following CenterNet [72].
- $L_{kp,h}$: (MonoCon only) Gaussian weighted focal loss for projected 3D keypoints as an auxiliary task.

6.2.2 Object-level Losses

These losses are specific to objects and do not require specialized masking:

- $L_{c,o}$: L1 Loss for offset from most confident foreground bin to precise projected 3D center
- $L_{h,w}$: L1 loss for 2D height and width
- $L_{S_{3d}}$: Dimension-loss. Dimension-aware L1 loss (L1 loss normalized by ground truth) in case of MonoCon and L1 loss in case MonoLSS
- L_{depth}: Laplacian aleatoric uncertainty loss
- L_{θ}: Multi-bin loss following Mousavian et al. [37]
- $L_{kp,o}$: L1 loss for keypoint offsets from keypoint heatmap
- $L_{kp,co}$: L1 loss for keypoint offsets from projected 3D center

6.3. Obtaining Diverse Features

Ensemble Features When extracting features for *Core-Set* Box_{3D} we orientate on our baseline detectors. The idea is simple: We use the features that lead to a bounding box prediction. For MonoLSS [23] we use the region of interest (RoI) features of dimension d x 7 x 7, while for MonoCon [27] we use the features of size d x 3 x 3. Before applying Core-Set [47] selection, the tensors are flattened into a single dimensional vector.

For our main model we employ the standard DLA-34 [61] backbone ($d = 64$). For our auxiliary models we replace the DLA-34 with the backbones of RepVIT M [55] ($d = 56$) and MobileNetv4 M [41] ($d = 48$). These models are very lightweight in terms of parameters and still offer an acceptable 2D and 3D detection performance, while being easily interchangeable with minimal code modifications (*cf*. Fig. 12).

Visual Features The extraction of visual features follows a two step process. First we mask the image, then we encode it using an off-the-shelf image autoencoder.

To ensure compatibility with Core-Set selection, all object features must share the same dimensionality, yet the pixel space of instances differs. For example, a car closer to the camera has a larger pixel height than a car further away. While resizing objects to a fixed size is a straightforward solution, we adopt a more effective strategy (*cf*. Tab. 6). Specifically, we crop each object to a fixed height and width of 320×320 pixels, centering the crop on the 2D center of the object. If the object lies near the image boundary, we apply padding using the background color. We resize the cropped region to 128x128 pixels before feeding it into the autoencoder. We utilize the autoencoder of Stable Diffusion v2-base [46] for this purpose.

Table 6. **Ablation study on the KITTI [12] validation set showing the effect of resizing**. Results are reported using $\text{NAURC}_{60\%}$ $AP_{3D|R_{40}}$ for cars (IoU 0.7) and pedestrians/cyclists (IoU 0.5). **Final AP**: Moderate AP after training on 60% of the data.

Method	Easy	Moderate	Hard	Final AP
Ours w/ object resizing	21.51	15.17	12.55	18.59
IDEAL-M3D (Ours)	**22.74**	**16.18**	**13.57**	**19.04**

Our approach ensures that the visual features effectively capture both the object's 2D size and depth through the fixed cropping strategy, which preserves relative scale and encourages depth-diversity. Additionally, the resized crop maintains the object's visual characteristics, enabling the autoencoder to learn a rich, low-dimensional representation of the visual appearance.

For segmentation [44], we utilize the SAMv2 ViT-B [44] architecture on Rope3D [60] and Waymo [51] and the larger SAMv2 ViT-L [44] variant on KITTI [12].

7. Experiments

7.1. NAURC Evaluation Metric

A central challenge in AL evaluation is to obtain a scalar, budget-aware summary that is comparable across selection paradigms. Previous work typically use two approaches: (i) training curves that plot performance against the number of labeled instances on the x-axis [13, 15, 33, 47, 58, 64, 67]; while informative, rankings often swap across budgets/time, complicating objective comparison; and (ii) fixed-budget snapshots that report performance at selected percentages of labeled data [16, 33, 35]; these yield a single value but depend on the chosen percentage and ignore the rest of the trajectory.

We introduce the Normalized Area under the Requested Curve (NAURC) to provide a fair, single-scalar comparison across image- and instance-based methods. NAURC adopts a common instance-based accounting for all approaches: for image-based selection, each selected image contributes the number of labelable instances it contains (the budget accumulates the total instances from the chosen images); for instance-based selection, the budget counts all requested instances, including requests that are later deemed non-labelable, thereby reflecting annotator verification effort. NAURC normalizes by a target budget and handles budget mismatch robustly: if a method overshoots the budget, we linearly interpolate back to it; if it undershoots, we keep the last observed performance (no extrapolation).

Empirical motivation for instance-based accounting. We analyze the relationship between actually labeled instances and the allocated budget in Fig. 5. For image-based methods, we quantify the budget at each AL iteration in terms of actually labeled boxes, since these methods request whole images. Most image-based strategies, with the exception of *Img Random*, tend to select images containing above-average numbers of instances, revealing that an image-count budget unfairly mixes annotation efforts because images vary widely in object count. For instance-based methods, the labeled ratio captures the proportion of true positive requests. Because false positive requests still require annotator verification, they count against the budget. This slightly penalizes instance-based methods compared to image-based methods. However, it also considers the annotators effort to verify that a requested object is a false positive.

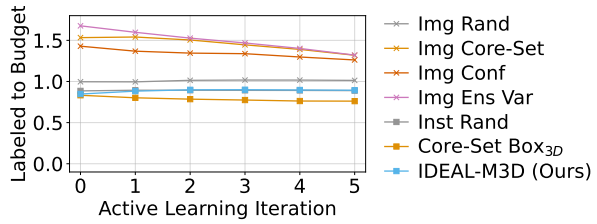


Figure 5. Ratio of labeled instances vs. instance-based budget on KITTI [12]. Most image-based methods request images with more than the average number of objects.

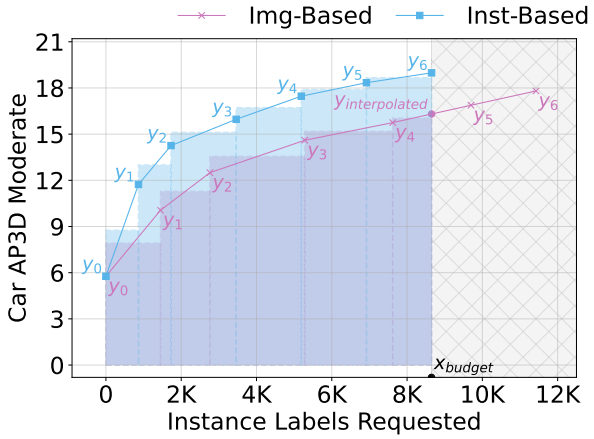


Figure 6. Visualization of the Normalized Area under the Requested Curve computation. For methods exceeding the label budget x_{budget}, the final performance metric is interpolated.

Formally, we compute the metric as the Area under the Requested Curve (AURC) up until the total requested label budget x_{budget} normalized by x_{budget} (refer to Fig. 6). The NAURC is defined as:

$$\text{NAURC}_{x_{\text{budget}}} = \frac{1}{x_{\text{budget}}} \left(\text{AURC}_{\text{final}} + \sum_{i=0}^{k} \text{AURC}_i \right), \tag{14}$$

where k denotes the last AL iteration before reaching the requested instance label budget x_{budget}:

$$k = \max \{ i \mid x_{i+1} \leq x_{\text{budget}} \}. \tag{15}$$

We compute the metric across iterations using the trapezoidal rule to calculate the AURC between consecutive data points.

$$\text{AURC}_i = \frac{(y_{i+1} + y_i)}{2} \cdot (x_{i+1} - x_i), \tag{16}$$

where y_i and y_{i+1} are the metric values at the i-th and $(i+1)$-th AL iteration and x_i and x_{i+1} are their respective requested instance labels. The AURC of the final interval is computed as:

$$\text{AURC}_{\text{final}} = \frac{(y_{\text{interpolated}} + y_k)}{2} \cdot (x_{\text{budget}} - x_k), \tag{17}$$

where $y_{\text{interpolated}}$ is the metric value at the budget point. To handle both cases where methods exceed or fall short of the target budget, we define:

$$y_{\text{interpolated}} = \begin{cases} y_k + \Delta y \cdot \frac{x_{\text{budget}} - x_k}{x_{k+1} - x_k} & \text{if } x_{k+1} > x_{\text{budget}} \\ y_k & \text{otherwise} \end{cases} \tag{18}$$

where $\Delta y = y_{k+1} - y_k$.

Compared to prior AUC-style metrics [65–67], NAURC enables fair cross-paradigm comparison across image- and instance-based methods: If a method overshoots the budget, NAURC linearly interpolates the terminal performance at x_{budget}; if it undershoots, it holds the last observed value. This removes extrapolation and prevents artificial inflation.

7.2. Implementational Details

Active Learning Parameters. The distance weights (*cf*. Eq. (5)) are determined as $\lambda_1 = \lambda_2 = \lambda_3 = \frac{1}{6}$, with λ_{vis} set to $\frac{1}{2}$ to balance the contribution of visibility metrics. The Loss weight multiplier parameter is set to $\delta = 0.2$ For adaptive sampling, we configure the time-dependent decay parameter α to 3.0 for KITTI [12] and 30.0 for Rope3D [60] and Waymo [51] to account for the different datase size. On KITTI [12], labeling proposals exclude objects with heights below 25 pixels, as ground truth instances are constrained to a minimum height of 25 pixels, which eliminates unsuitable candidates during this process. All experiments, including run time evaluations, are conducted on a single NVIDIA A40 GPU with 32GB RAM. For instance-based AL, we mitigate redundant labeling by skipping requests for objects that fall within 95% of the radius (r_x, r_y) from a previous request

and share the same predicted class. This prevents repeated annotations or requests regarding the same false-positive predictions. For masking and visual feature extraction, we leverage the 2D bounding box predictions of the main model. To accelerate computation, we apply principal component analysis (PCA) to the extracted features, compressing the dimensions while retaining at least 99% of the variance.

MonoLSS [23] Configuration. The training setup utilizes a batch size of 16 and optimizes the model using the Adam [20] optimizer with a weight decay of 1e-5. Starting from a pre-trained checkpoint, each AL iteration spans 150 epochs for the main model. During training, we initialize the learning rate at 1e-3 and decay it by a factor of 0.1 after 60% and 80% of the epochs. Additionally, the first cycle incorporates a 5-epoch cosine warmup to stabilize gradient updates. After the initial phase of training, the LSS module gets activated at the 50th epoch. To enhance model robustness, we apply comprehensive data augmentation techniques, including random horizontal flipping, shifting (W: $\pm$256 pixels, H: $\pm$77 pixels), scaling (0.6-1.4), and MixUp3D using fully labeled images.

MonoCon [27] configuration. For MonoCon, we adopt a batch size of 24 and train the model using the AdamW [29] optimizer with a weight decay of 1e-5 and a learning rate of 0.0011. The initial training phase encompasses 90 epochs, followed by an additional 30 epochs for every subsequent AL cycle. To maintain stable training, we implement gradient clipping with a norm of 35 and apply cosine learning rate scheduling. For computational efficiency, images are resized to 960$\times$640 pixels, in line with the settings from [36]. The augmentation pipeline integrates a rich variety of transformations, including photometric distortion, random shift ($\pm$ 32 pixels), horizontal flipping, and random cropping (900$\times$550 pixels). Furthermore, for Rope3D [60], we learn the $SO(3)$ orientation matrix in alignment with the GroundMix [36] approach.

Labeling radius. Also, to simplify the task for the labeler, we expect that an object lies within a depth-dependent radius (r_x, r_y), letting the user focus on a small area. Such radius is defined as:

$$r_x = H \cdot \frac{f_x}{\hat{z}}, \quad r_y = H \cdot \frac{f_y}{\hat{z}}, \tag{19}$$

where H is a scaling factor, (f_x, f_y) are the camera focal lengths, and $\hat{z}$ is the predicted depth. This ensures accurate targeting by accounting for the geometric effects of depth, as pixel-space errors decrease with distance. We define the labeling radius via $H = 2.0$, which corresponds to approximately 47 pixels for objects at a 30m distance on KITTI [12].

Instance matching between ensemble members. Using the predictions from the main model as a reference, we associate predictions from auxiliary models by calculating the 2D bounding box IoU. This computation uses a relaxed threshold of 0.5 to accommodate additional object matches. As auxiliary models are trained with fewer resources, we adjust detection thresholds to compensate for their lower capacity. Specifically, thresholds are reduced from 0.2 to 0.1 for MonoCon [27] and from 0.2 to 0.05 for MonoLSS [23]. For cases of multiple associations, we prioritize the object with the highest confidence score. This strategy significantly enhances instance coverage, particularly for rare and low-confidence objects, while maintaining diversity and minimizing discard rates.

7.3. Implementation Details of Baseline Methods

For completeness, we provide additional implementation details for a subset of our baseline methods. In our evaluation framework, we adapt image-based methods by selecting images based on their highest-scoring contained instance, while instance-based methods directly employ our labeling pipeline (Sec. 3.1) with their respective acquisition functions. To manage computational resources effectively, we implement dataset-specific sampling strategies: For datasets containing on average more than 10 objects per image (*e.g.* Rope3D [60]), we limit the maximum number of requested images to one-third of the number of instances, though this restriction is not applied to KITTI [12] due to its limited size. In the following, we provide further details on the individual baseline methods.

Augmentation Depth Variance (Augm Depth Var): This method provides a computationally efficient alternative to ensemble approaches by employing multiple augmented forward passes of a single model. The augmentation pipeline includes blur ($\sigma = 0.5$), brightness adjustments (factors: 0.5-1.5, probability: 0.2-0.5), and hue shifts (±0.1). We exclude this method for MonoCon [27] evaluations due to its built-in color augmentation during training.

CDAL [1]: This approach implements Core-Set sampling by computing class-wise confusions from softmax probabilities of both labeled data heatmap indices and predictions. The sampling strategy employs pairwise and class-wise KL divergence for measuring image similarity.

CloseDepth: Instances with low depth values are preferred for labeling.

Confidence (Conf): This uncertainty-based approach combines depth and classification confidence scores from the detector, selecting instances with lower confidence for labeling prioritization.

DDFH [4]: DDFH is the current state-of-the-art method for active learning in LiDAR-based 3D detection. The approach optimizes three key aspects: balancing class distribution, maximizing frame-level heterogeneity, and selecting

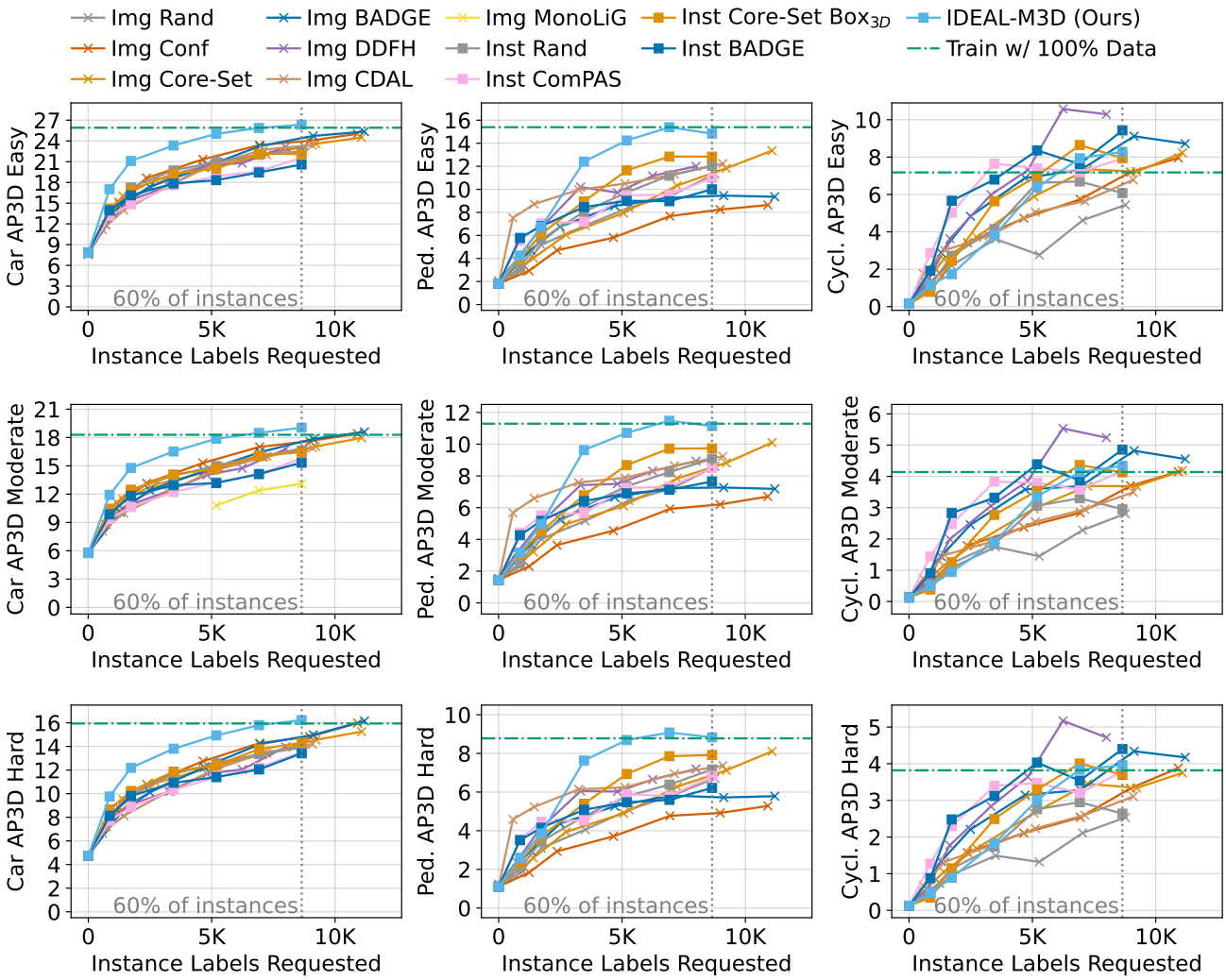


Figure 7. **AL methods evaluated on the KITTI validation set [12] for Easy, Moderate and Hard on Car (IoU=0.7), Pedestrian (IoU=0.5) and Cyclist (IoU=0.5).**

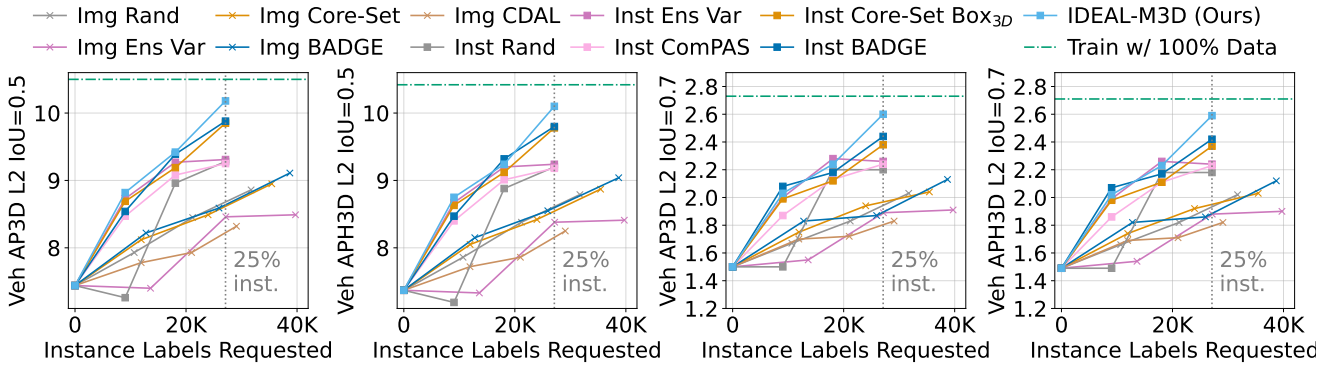


Figure 8. **AL methods evaluated on the Waymo validation set [51].**

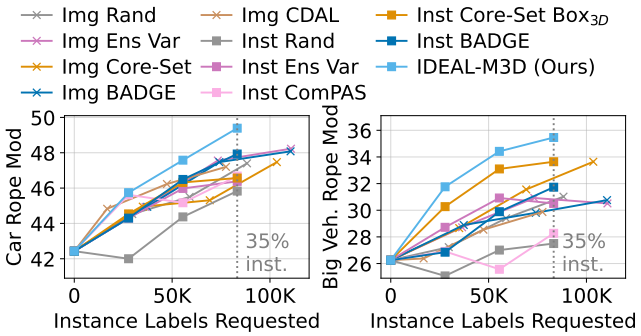


Figure 9. **AL methods evaluated on the Rope3D validation set [60]** with Rope Score at IoU 0.5.

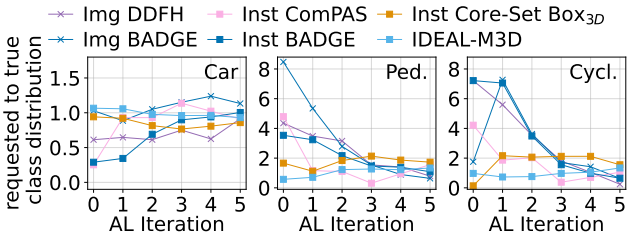


Figure 10. KITTI [12] ratio of requested AL instances to the true dataset distribution (<1: Undersampling, 1: Balanced, >1: Oversampling).

diverse instances. For instance diversity, it fits a Gaussian mixture model to t-SNE [53] compressed model features and additional geometric features. We adapt this approach to monocular 3D detection by dropping the point density information from the feature vectors.

BADGE [3]: BADGE computes gradient embeddings derived from the penultimate classification layer for each data point and selects samples via the k-MEANS++ seeding algorithm [2]. We compute gradients based on the focal loss for the predicted heatmap locations on an instance basis.

ComPAS [33]: We adapted the active learning component of ComPAS [citation] to our setting, which selects instances based on their localization and classification disagreement between a chairman and its committee members. This disagreement is measured using multiple data augmentations, including scale, shift, contrast, solarize, saturation, sharpness, and brightness.

Dropout-based Methods: We investigated dropout-based uncertainty estimation but excluded it from our final evaluation due to significant performance degradation. Specifically, when applying dropout to backbone features, we observed substantial drops in accuracy: While the baseline achieves 18.29 for Car $AP^{IoU=0.7}_{3D|R_{40}}$ Moderate, introducing dropout rates of 1%, 5%, and 10% reduces performance to 17.73, 15.81, and 13.67 on the KITTI [12] validation set, respectively (baseline: MonoLSS [23]).

Efficient AL [16]: We excluded Efficient AL [16] from our comparison due to ambiguous evaluation metrics (unspecified IoU thresholds for mAP and no differentiation between KITTI [12]'s easy, moderate, and hard instances).

Ensemble Depth Variance (Ens Depth Var): Using an ensemble of three models, we associate predictions via a 2D IoU threshold of 0.5. The uncertainty metric is derived from the variance of depth predictions relative to their mean, prioritizing instances exhibiting higher variance.

Ensemble Relative Standard Deviation (Ens Rel Std): Similar to *Ens Depth Var*, but normalizes the depth standard deviation by the mean predicted depth.

FarDepth: Instances with large depth values are preferred for labeling. Predicted instances need to have a 2D height of at least 25 pixels and a depth smaller 50m.

MonoLiG [15]: MonoLiG [15] combines AL with semi-supervised learning by leveraging an ensemble of five models alongside a LiDAR-based teacher model. The instance selection strategy integrates three uncertainty measures: the aleatoric uncertainty of the teacher model, the disagreement among ensemble members, and the teacher-student prediction inconsistency. These measures are aggregated into a unified scoring mechanism for ranking potential labeling candidates. Due to unavailability of the official implementation at submission time, we report the results from the paper. To establish a fair comparison, we estimate the number of labeled instances per AL iteration based on the dataset's average object count per image.

7.4. Comparison with AL methods

We provide additional comparisons to contextualize the main results in Tab. 2. Final accuracies at the end of AL training are summarized in Tab. 8, and the corresponding training curves are shown in Figs. 7 to 9, offering both endpoint and trajectory views of performance.

Per-class results on KITTI for pedestrian and cyclist are reported in Fig. 7 and Tab. 7. IDEAL-M3D significantly outperforms all baselines on Car and Pedestrian. While ComPAS [33], *Core-Set Box$_{3D}$* BADGE [3], and DDFH [4] report higher AP on Cyclist, this comes from a highly skewed budget allocation: they assign up to 7× more annotations to Cyclist at the expense of the other classes (*cf*. Fig. 10). In contrast, IDEAL-M3D maintains a balanced acquisition across categories, which we hypothesize emerges from the representational diversity of our ensembles. This yields stronger average performance and a more uniform gain among all classes.

7.5. Uncertainty vs. Diversity-based Methods

To understand why uncertainty-based methods are less effective for instance-based M3D (*cf*. Tab. 2), we conduct a detailed analysis comparing *Core-Set Box$_{3D}$* with three

Table 7. **AL performance on the KITTI [12] validation dataset**. Results are averaged over three rounds, each initialized from the same checkpoint. **KITTI**: We report NAURC$_{60\%}$ $AP_{3D|R_{40}}$ with IoU thresholds of 0.7 (cars) and 0.5 (pedestrians, bicyclists). **Type***: U=Uncertainty-based, D=Diversity-based, H=Hybrid.

			Car			Pedestrian			Cyclist			Average		
	Method	Type*	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Image-based	Rand	-	18.01	12.97	10.77	6.31	4.83	3.87	2.88	1.44	1.29	9.07	6.41	5.31
	Conf	U	19.72	14.20	11.80	5.56	4.31	3.47	4.19	2.11	1.88	9.82	6.87	5.72
	Ens Depth Var	U	18.19	13.00	10.77	4.36	3.47	2.74	4.15	2.09	1.89	8.90	6.19	5.13
	Augm Depth Var	U	18.24	13.16	10.90	4.53	3.61	2.86	3.34	1.67	1.49	8.70	6.15	5.08
	Core-Set [47]	D	18.80	13.50	11.34	7.20	5.59	4.47	4.88	2.45	2.23	10.29	7.18	6.01
	BADGE [3]	H	19.02	13.53	11.39	7.40	5.76	4.62	5.68	2.91	2.60	10.70	7.40	6.20
	DDFH [4]	H	17.94	12.72	10.51	8.90	6.68	5.38	6.47	3.37	3.12	11.10	7.59	6.34
	CDAL [1]	H	18.38	12.99	10.82	9.88	7.49	6.00	4.27	2.16	1.93	10.84	7.55	6.25
Instance-based	Rand	-	18.69	13.53	11.21	8.03	6.05	4.72	4.49	2.10	1.91	10.40	7.23	5.95
	Conf	U	12.10	8.93	7.60	3.71	2.90	2.30	2.96	1.42	1.27	6.26	4.22	3.87
	Ens Depth Var	U	11.88	9.07	7.82	3.66	2.82	2.22	3.67	1.75	1.59	6.40	4.55	3.88
	Augm Depth Var	U	12.25	8.96	7.64	3.15	2.42	1.94	2.80	1.36	1.24	6.07	4.25	3.61
	ComPAS [33]	U	17.31	12.31	10.48	8.04	6.24	5.01	6.29	3.18	2.87	10.55	7.24	6.12
	Core-Set [47] Box$_{3D}$	D	18.83	13.82	11.68	9.38	7.07	5.68	5.50	2.78	2.54	11.24	7.89	6.63
	BADGE [3]	H	17.39	12.63	10.74	7.98	6.13	4.88	6.55	3.32	3.04	10.64	7.36	6.22
	IDEAL-M3D (Ours)	D	22.74	16.18	13.57	11.42	8.61	6.86	4.84	2.51	2.32	**13.00**	**9.10**	**7.58**

Table 8. **Final AL performance on KITTI [12] validation, Waymo [45, 51] valiation and Rope3D [60] validation dataset**. Results are averaged over three rounds, each initialized from the same checkpoint. We report the final accuracy after training on 60% of data (KITTI) and 25% of data (Rope3D, Waymo). For image-based methods exceeding the budget we report the interpolated result. **Type***: U=Uncertainty-based, D=Diversity-based, H=Hybrid.

			KITTI [12] Car			**Waymo [51] Vehicle**				**Rope3D [60]**			
			$AP^{0.7}_{3D\|R_{40}}$			IoU=0.5		IoU=0.7		Car		Big Vehicle	
	Method	Type*	Easy	Mod.	Hard	AP$_{3D}$	APH$_{3D}$	AP$_{3D}$	APH$_{3D}$	AP$^{0.5}_{3D}$	Rope	AP$^{0.5}_{3D}$	Rope
Image-based	Rand	-	22.47	16.42	13.80	8.68	8.20	1.74	1.73	35.13	47.08	16.63	30.74
	Conf	U	23.89	17.51	14.72	8.12	8.05	1.66	1.65	34.04	46.24	17.61	31.63
	Ens Depth Var	U	22.81	16.28	13.66	8.46	8.37	8.37	1.87	35.93	47.74	16.76	30.81
	Augm Depth Var	U	22.20	16.15	13.72	-	-	-	-	-	-	-	-
	Core-Set [47]	D	23.10	16.73	14.19	8.32	8.25	1.90	1.88	34.11	46.25	18.59	32.41
	BADGE [3]	H	24.41	17.55	14.81	8.35	8.33	1.74	1.74	35.81	47.62	15.77	30.05
	DDFH [4]	H	23.13	16.77	13.96	8.22	8.15	1.85	1.84	35.70	47.54	19.02	32.71
	CDAL [1]	H	23.30	16.69	14.04	7.88	7.81	1.70	1.69	35.25	47.20	15.43	29.87
Instance-based	Rand	-	22.50	16.61	13.89	9.28	9.20	2.20	2.18	33.53	45.83	12.31	27.50
	Conf	U	13.00	10.00	8.67	9.05	8.97	2.32	2.31	34.85	46.86	14.38	29.05
	Ens Depth Var	U	11.29	9.01	8.43	9.31	9.24	2.26	2.24	34.28	46.38	16.68	30.95
	Augm Depth Var	U	9.69	7.38	6.58	-	-	-	-	-	-	-	-
	ComPAS [33]	U	21.59	15.79	13.52	9.25	9.18	2.24	2.23	34.58	46.68	13.32	28.27
	Core-Set [47] Box$_{3D}$	D	22.10	16.45	14.31	9.85	9.78	2.38	2.37	34.51	46.57	20.08	33.64
	BADGE [3]	H	20.59	15.33	13.42	9.88	9.80	2.44	2.42	36.13	47.93	17.50	31.73
	IDEAL-M3D (Ours)	D	**26.35**	**19.04**	**16.23**	**10.18**	**10.10**	**2.60**	**2.59**	**37.94**	**49.39**	**22.21**	**35.46**

uncertainty-based methods:

- *Conf*: Selects instances with lowest model confidence scores (aleatoric uncertainty)
- *Ens Rel Std*: Targets instances with highest relative depth deviation from the ensemble mean (three models)
- *Ens Depth Var*: Prioritizes instances with highest absolute depth error relative to the ensemble mean

Our analysis of the selected instances in Fig. 11 reveals

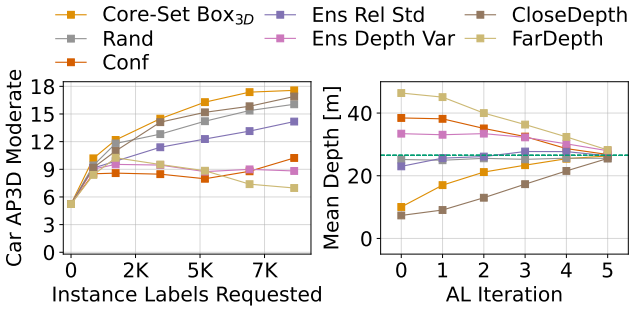


Figure 11. **Comparison of instance-based active learning methods for monocular 3D detection.** Uncertainty-based methods tend to select distant objects, which provide weaker training signals, while diversity-based approaches favor closer objects that are more informative. **Left**: The effectiveness of various selection strategies. **Right**: Distribution of selected instances, showing diversity-based methods' preference for close objects compared to uncertainty-based methods' bias toward moderately far and distant objects.

distinct selection patterns: Both *Conf* and *Ens Depth Var* demonstrate a clear bias towards distant objects, which is expected given that absolute depth errors typically increase with distance. In contrast, *Ens Rel Std* achieves more balanced sampling across different depths, while *Core-Set Box$_{3D}$* shows a preference for closer objects. We hypothesize that closer objects generally provide richer visual information due to higher pixel density and more distinct features, making them more conducive to accurate depth learning. Interestingly, diversity-based methods also tend to perform better for far away objects (hard category, *cf*. Tab. 2), even though they are undersampled during training time. To further investigate this, we introduce two simple AL strategies:

- *CloseDepth*: Prioritizes close instances for labeling
- *FarDepth*: Prioritizes most distant instances (filtered to instances ≤ 25 pixels and ≤ 50m depth to avoid false positives)

Our experiments in Fig. 11 show that *CloseDepth* performs second-best after *Core-Set Box$_{3D}$* , leading to two key insights. First, closer instances provide more effective training signals, even for detecting distant objects. Second, uncertainty-based methods are suboptimal for instance-based M3D due to their inherent bias towards moderate far and distant objects

7.6. Training time comparison with fully supervised methods

We compare total training time in Tab. 9 against fully supervised and semi-supervised baselines. Relative to MonoLSS [23], IDEAL-M3D trains for roughly twice as long. It reaches (near) fully supervised accuracy using only 60% of the labeled data on the validation and test sets. Since compute is typically far cheaper than human annotation, this is a favorable trade-off.

Compared to MonoLiG [15], IDEAL-M3D achieves higher accuracy on KITTI validation and test (*cf*. Tab. 3 and Fig. 3). It also reduces training time by about $3\times$. This efficiency stems from concrete design choices. MonoLiG trains an ensemble of five image-based student models and an additional sixth LiDAR-based model. In contrast, we train only three models. Our auxiliary ensemble components are lightweight and fast to train.

We also compare with the semi-supervised Mix-Teaching [59]. Its training time is more than $4\times$ higher. The method trains a five-model ensemble across one supervised and three semi-supervised rounds, which introduces substantial overhead. In contrast, IDEAL-M3D attains higher KITTI test accuracy without using unlabeled data and with fewer labeled samples (*cf*. Tab. 3).

Table 9. Comparison of training time of selected methods on the KITTI [12] trainval set. **SSL** denotes that the method uses semi-supervised learning. **AL** denotes that the method uses active learning.

Method	SSL	AL	**Total training time**
MonoLSS [23] (Baseline)	✗	✗	37h
IDEAL-M3D (Ours)	✗	✓	75h
MonoLiG [15]	✓	✓	240h
Mix-Teaching [59]	✓	✗	305h

7.7. Backbone ablation

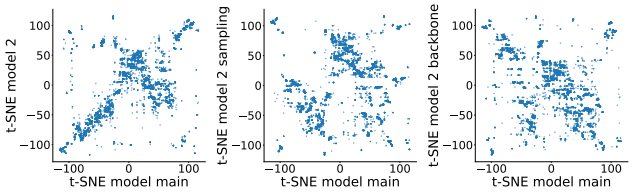
To promote ensemble diversity while keeping compute modest, we equip the auxiliary models with distinct lightweight backbones (cf. Sec. 3.3). We evaluate several candidates on KITTI [12] validation set using 100% of the training data to isolate backbone effects. Throughput and accuracy are summarized in Fig. 12.

For the main model, we retain DLA-34 [61] due to its strong accuracy, ensuring a fair and comparable reference across experiments. This backbone remains fixed in all primary results.

For the auxiliary models, we select RepViT-M1.0 [55] and MobileNetV4-Conv-M [41]. MobileNetV4-Conv-M offers the best speed–accuracy trade-off among the tested lightweight backbones, being both faster and more accurate than alternatives. ConvNeXt-Pico [28] is marginally faster than RepViT-M1.0 but is approximately 25% less accurate; we therefore prefer RepViT-M1.0. Together, RepViT-M1.0 and MobileNetV4-Conv-M provide complementary inductive biases and increase architectural diversity at low training cost.

Table 10. **Ensemble ablation study on the KITTI [12] validation set**. Results are reported using NAURC$_{60\%}$ $AP_{3D|R_{40}}$ for cars (IoU=0.7) and pedestrians/cyclists (IoU=0.5). **Final AP**: Moderate AP after training on 60% of the data.

Method	Easy	Moderate	Hard	Final AP
Ours w/o diverse backbones	20.85	15.08	12.66	18.10
Ours w/o data sampling	21.21	15.33	12.61	18.78
Ours w/ full epochs	22.38	16.03	13.42	18.42
Ours w/o random loss	22.02	15.55	12.55	18.93
IDEAL-M3D (Ours)	**22.74**	**16.18**	**13.57**	**19.04**



(a) (b) (c)

Figure 13. **t-SNE [53] visualization of RoI**. Considered features from ensemble models trained on 30% of the KITTI [12] dataset. **(a)** Identical data and backbones yield correlated features, limiting diversity. **(b)-(c)** Adaptive sampling and varied backbones (RepViT [55]) enhance feature diversity, improving ensemble effectiveness.

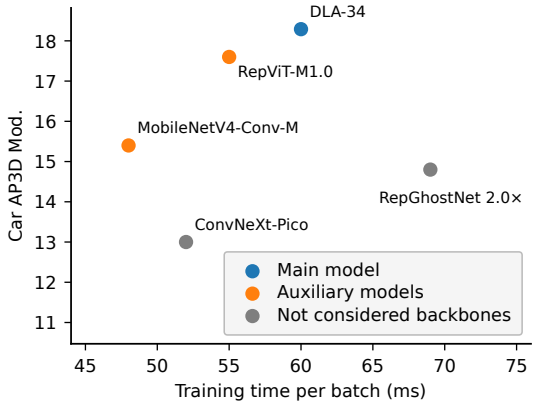


Figure 12. Comparing KITTI [12] validation set performance under diverse backbones (supervised raining with 100% of data).

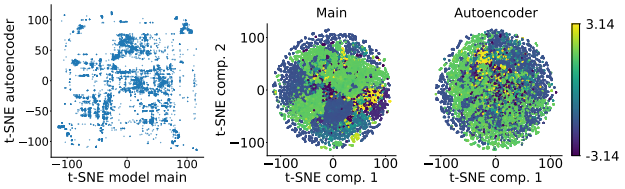
7.8. Feature diversity ablation

A key component of our approach is instance selection driven by diverse, fast-to-train feature ensembles. We increase diversity and reduce compute through four complementary mechanisms: (i) heterogeneous lightweight backbones, (ii) time-adaptive data sampling that varies the training trajectory across ensemble members, (iii) fewer training epochs for auxiliary models, and (iv) random sampling of loss weights to perturb optimization and feature emphasis across tasks/heads.

Together, these mechanisms reduce total training time by approximately 15 hours compared to a vanilla, homogeneous ensemble trained for full schedules, while improving selection quality through more diverse feature views.

The ablation in Tab. 10 supports this design. Reducing epochs on auxiliary models preserves performance to within noise levels, indicating that full schedules are unnecessary for effective feature-based selection. Adding backbone diversity, time-adaptive sampling, and random loss-weight sampling yields incremental gains in AP$_{3D|R_{40}}$ (Moderate) of +1.10, +0.85, and +0.63, respectively.

We further analyze feature diversity in Fig. 13 via t-SNE [53]. Ensembles using time-adaptive sampling and heterogeneous backbones exhibit markedly lower inter-model feature correlation than a vanilla ensemble, confirming that our mechanisms produce complementary representations that enhance the quality of selected instances.



(a) (b)

Figure 14. **t-SNE [53] visualization of features trained on 30% of the KITTI [12] dataset, highlighting the complementarity of RoI and autoencoder features. (a)** RoI and autoencoder features show low correlation, capturing distinct information. **(b)** Example: RoI features are dominated by the object orientation around the y-axis, while Stable Diffusion v2-base [46] autoencoder features are more orientation-invariant. The vehicle's orientation is shown by color.

Table 11. **Visual diversity ablation study on the KITTI [12] validation set**. Results are reported using NAURC$_{60\%}$ $AP_{3D|R_{40}}$ for cars (IoU=0.7) and pedestrians/cyclists (IoU=0.5). **Final AP**: Moderate AP after training on 60% of the data.

Method	Easy	Moderate	Hard	Final AP
Ours w/o SAMv2 [44]	17.72	13.24	11.11	16.36
Ours w/ DINOv2 [38]	17.87	13.08	11.12	16.60
IDEAL-M3D (Ours)	**22.74**	**16.18**	**13.57**	**19.04**

7.9. Visual diversity ablation

In Tab. 11, we ablate components that promote visual diversity in the feature ensemble.

Removing Segment Anything Model 2 (SAMv2) reduces AP_{3D} (Moderate) by more than 2 points. We hypothe-

size two causes. First, without SAMv2, background content leaks into the features. The same object with different backgrounds then appears artificially diverse, which dilutes foreground cues. Second, the foreground mask carries coarse geometric information. E.g, the silhouette of a car at $45°$ yaw differs from one at $0°$, which is useful for 3D selection.

Replacing the autoencoder with DINOv2 [38] features causes a similar drop. DINOv2 emphasizes global scene semantics. It captures less local, instance-centric detail [54]. The autoencoder yields object-focused representations that better support instance selection.

Fig. 14 shows t-SNE [53] visualizations of autoencoder features and detector features. The two spaces exhibit minimal correlation. This indicates that the task-agnostic autoencoder provides complementary signals to the task-specific model features.

7.10. Qualitative Results

In the subsequent pages (*cf*. Fig. 15, Fig. 16, Fig. 17) we present further qualitative results on IDEAL-M3D highlighting the prediction accuracy and selection process over time.

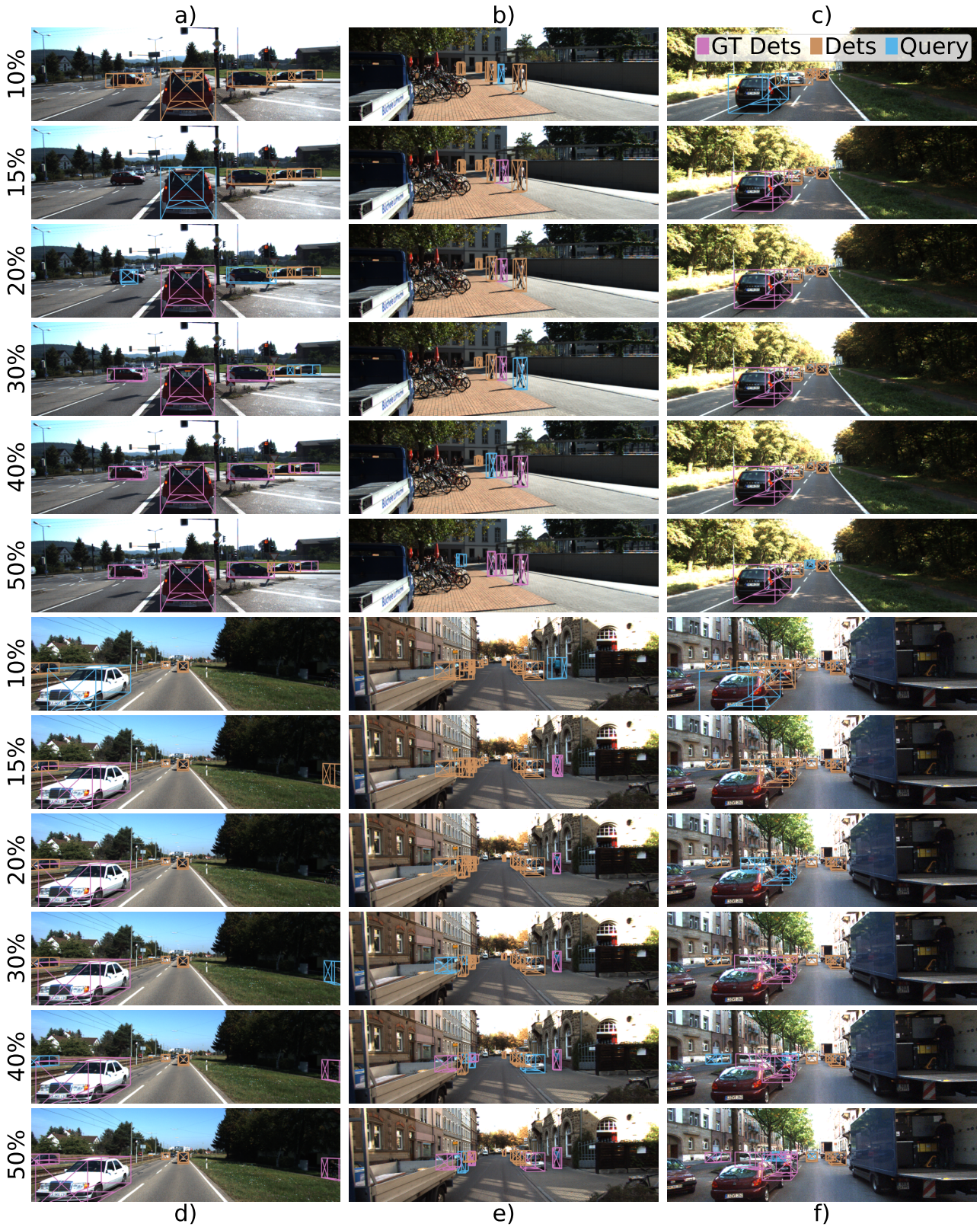


Figure 15. **Qualitative results of IDEAL-M3D on the KITTI [12] dataset showing prediction evolution over time (top to bottom)**. Pink boxes represent previously labeled instances now in the training set, cyan boxes indicate predictions selected for current labeling, and orange boxes show predictions not chosen for annotation. The progression demonstrates the model's improvement through strategic label acquisition

Figure 16. **Qualitative results of IDEAL-M3D on the Waymo [51] dataset showing prediction evolution over time (top to bottom).** Pink boxes represent previously labeled instances now in the training set, cyan boxes indicate predictions selected for current labeling, and orange boxes show predictions not chosen for annotation. The progression demonstrates the model's improvement through strategic label acquisition

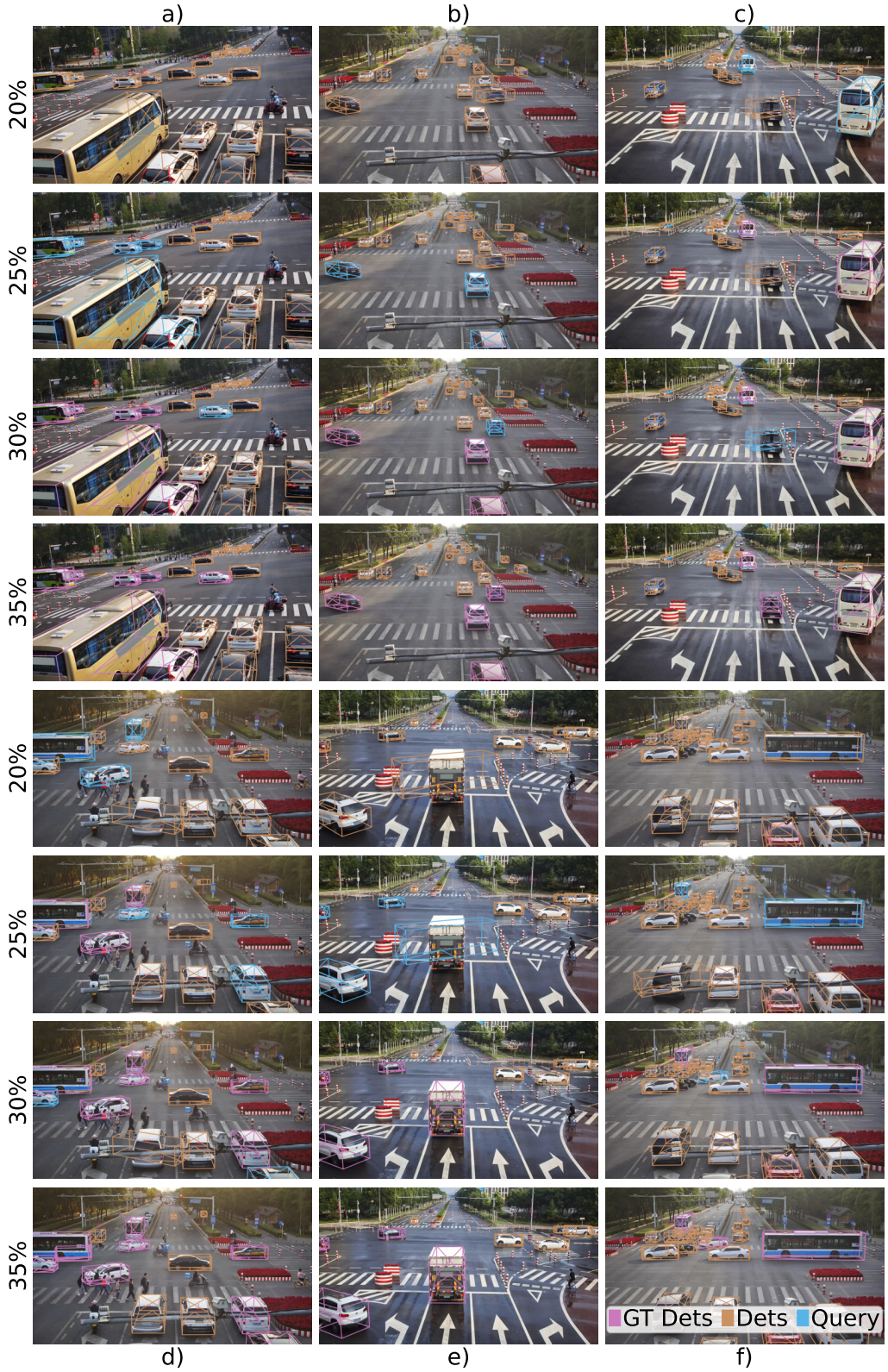


Figure 17. **Qualitative results of IDEAL-M3D on the Rope3D [60] dataset showing prediction evolution over time (top to bottom).** Pink boxes represent previously labeled instances now in the training set, cyan boxes indicate predictions selected for current labeling, and orange boxes show predictions not chosen for annotation. The progression demonstrates the model's improvement through strategic label acquisition