MAD: <u>M</u>emory-<u>A</u>ugmented <u>D</u>etection of 3D Objects
Supplementary Material

In this appendix, we describe additional implementation details relevant to MAD including architecture, training, any implementation details for the 3D detectors, additional quantitative and qualitative results, and limitations and areas for future work.

6 Additional Implementation Details

In this section, we describe some additional implementation details of MAD — its architecture and training — and implementation details for our off-the-shelf 3D detectors.

6.1. Architecture

Trajectory Forecasting Header: Each refinement block predicts trajectory forecasts from the refinement features $\mathbf{Q}^{\mathrm{ref}(i)} \in \mathbb{R}^{N^{\mathrm{ref}} \times T_f+1 \times d}$ using a single-layer bi-directional GRU [9]. The output of the GRU is a sequence of 2D BEV positions for each actor of shape $\mathbb{R}^{N^{\mathrm{ref}} \times T_f \times 2}$, and we use finite differences to calculate the heading. The result is trajectories $\mathbf{T}^{\mathrm{ref}(i)} \in \mathbb{R}^{N^{\mathrm{ref}} \times T_f \times 3}$, describing each objects' BEV pose $\{(x, y, \theta)_{t+s_f}, \ldots, (x, y, \theta)_{t+s_f+T_f-1}\}$. Note while the GRU sequence length is $T_f + 1$, we only use the last T_f outputs as the future trajectory forecasts.

6.2. Training

Augmentations: When training MAD we adopt standard data augmentations used in prior works [62, 80, 81], including random translations in the x, y, and z directions in the ego coordinate frame with a standard deviation of 0.5m, random rotation around the z-axis (in the BEV plane) uniformly sampled from $[-\pi/4, \pi/4]$, random scaling sampled uniformly from $[0.95, 1.05]$, and flipping with {no flip, x flip, y flip, x = y flip} occurring with equal probability. We do not use any ground-truth sampling [71] because it is difficult to make consistent and realistic over time and thus incompatible with memory training.

Loss Functions: Recall our loss function described in Sec. 3.2,

$$L = L_{\mathrm{rescore}}(\mathbf{C}^{\mathrm{merge}}) + \sum_{i=1}^{I} L_{\mathrm{det}}(\mathbf{B}^{\mathrm{ref}(i)}, \mathbf{C}^{\mathrm{ref}(i)}) + L_{\mathrm{for}}(\mathbf{T}^{\mathrm{ref}(i)}), \tag{1}$$

which is a combination of a rescoring loss L_{rescore}, a detection refinement loss L_{det}, and a forecasting refinement loss L_{for}, where the detection and forecasting losses are computed at every refinement block. For brevity, in the above formula we omit the labels in the inputs to the above loss functions. The labels consist of the bounding boxes for L_{rescore} and L_{det}, and the ground truth future trajectories for L_{for}.

Our detection refinement loss $L_{\mathrm{det}}(\mathbf{B}^{\mathrm{ref}(i)}, \mathbf{C}^{\mathrm{ref}(i)})$, includes a binary focal loss $L_{\mathrm{det}}^{\mathrm{cls}}$ for classification, and an L1 loss $L_{\mathrm{det}}^{\mathrm{L1}}$ and IoU loss $L_{\mathrm{det}}^{\mathrm{IoU}}$ to regress the bounding box parameters: $L_{\mathrm{det}} = \lambda_{\mathrm{cls}} L_{\mathrm{det}}^{\mathrm{cls}} + \lambda_{\mathrm{L1}} L_{\mathrm{det}}^{\mathrm{L1}} + \lambda_{\mathrm{IoU}} L_{\mathrm{det}}^{\mathrm{IoU}}$. For $L_{\mathrm{det}}^{\mathrm{cls}}$ we use focal loss parameters $\alpha = 0.5, \gamma = 2.0$. We use loss weightings of $\lambda_{\mathrm{cls}} = 1.0, \lambda_{\mathrm{L1}} = 0.1$, and $\lambda_{\mathrm{IoU}} = 4.0$. To calculate the targets for these losses, we first match the detections to the ground truth bounding boxes through bipartite matching as proposed in DETR [3]. We use the same costs for bipartite matching as in DETR except that we omit the L1 component of the box cost, and use 3D IoU instead of 2D IoU.

The rescoring loss L_{rescore} is the same as the detection refinement loss described above but without regression $\lambda_{\mathrm{L1}} = \lambda_{\mathrm{IoU}} = 0$.

6.3. 3D Detector Implementation Details

We use the official implementations for all 3D detectors in our experiments: single stage CenterPoint [75][3], HEDNet [81] and SAFDNet [80][4], FCOS3D [64][5], and BEVMap [6][6].

For the LiDAR based models (CenterPoint [75], HEDNet [81], SAFDNet [80]), we train with their official configurations for the WOD. Neither FOCS3D nor BEVMap provide official configurations for training on AV2; we train with the "front ring camera" on AV2 for 3 epochs with a batch size of 16, using the proposed learning rate scheduler.

To obtain $\mathbf{Q}^{\mathrm{det}}$ from the 3D detectors, for each object proposal $(x, y, z, l, w, h, \theta)$ in $\mathbf{B} \in \mathbb{R}^{N \times 7}$, we interpolate the features map directly before the detection header at the projected object centroid (x, y, z). Concretely, for detectors that produce BEV feature maps before their header (e.g., CenterPoint [75], HEDNet [81], SAFDNet [80], BEVMap [6]), we perform bi-linear interpolation of the feature maps at the BEV object centroid (x, y). For detectors that use image-view feature maps, we perform bi-linear interpolation at the object centroid (x, y, z) projected onto the image plane. While not experimented with in this work, we could similarly use tri-linear interpolation for detectors that use 3D feature maps.

7 Additional Quantitative Results

In this section, we provide additional quantitative experiments on MAD, including full results on WOD leaderboard, evaluating the performance using different memory horizons, investigating if our proposed memory improves

---

[3] https://github.com/tianweiy/CenterPoint
[4] https://github.com/zhanggang001/HEDNet
[5] https://github.com/jjw-DL/mmdetection3d_Noted/
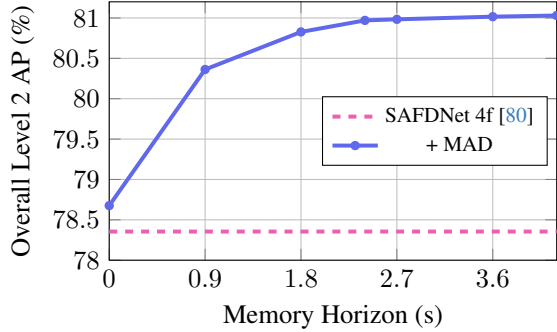[6] https://github.com/mincheoree/BEVMap

Figure 3. The effect of the memory horizon on the WOD validation set. We keep the memory target time stride $s_m = 0.3s$ fixed, while increasing the number of target timestamps $T_m \in \{0, 3, 6, 8, 9, 14\}$. The base detector is SAFDNet 4f [80].

Figure 4. The effect of randomly dropping memory target timestamps $\mathcal{T}_m$ at inference time on the WOD validation set. The base detector is SAFDNet 4f [80].

trajectory forecasting performance, evaluating the effect of randomly dropping memory at inference time, ablating self-attention in the refinement transformer, and additional training design experiments — training with back-propagation through time (BPTT) and accumulating gradients.

Full Leaderboard Results: In Tab. 8 we show the performance of MAD when enhancing SAFDDet 4f [80] on the WOD test set for all actor classes, comparing against other methods on the leaderboard that do not use ensembles, test-time augmentations, or future sensor data. For methods that provide no information on these criteria, we give them the benefit of the doubt and include them. We see that MAD is strong across all actor classes, particularly on pedestrians and cyclists. Our intuition is that these actor classes are smaller and more prone to limited LiDAR observations due to occlusions or distance, so our memory mechanism is particularly beneficial.

In Tab. 9 we compare against prior methods for learned temporal fusion on the WOD validation and test sets. We find that MAD outperforms prior works by a large margin on all actor classes.

Performance at different memory horizons: In Fig. 3, we plot detection performance as a function of the memory horizon $T_m s_m$ used at inference time. As described in Sec. 3.2, the model is trained with a variable number of target past timestamps $T_m \in \texttt{uniform}(\{6, 7, 8, 9, 10\})$, with a variable stride $s_m \in \texttt{uniform}(\{0.2s, 0.3s, 0.4s\})$, meaning the memory horizon during training varies from 1.2s to 4.0s. To generate this plot, we run multiple evaluations on the WOD validation set, each with a different number of target timestamps $T_m \in \{0, 3, 6, 8, 9, 14\}$, keeping the memory target time stride $s_m = 0.3s$ fixed. We find that the performance increases rapidly when first introducing memory, and then increases more slowly and saturates around 2.7s, due to the redundancy in most memory proposals.
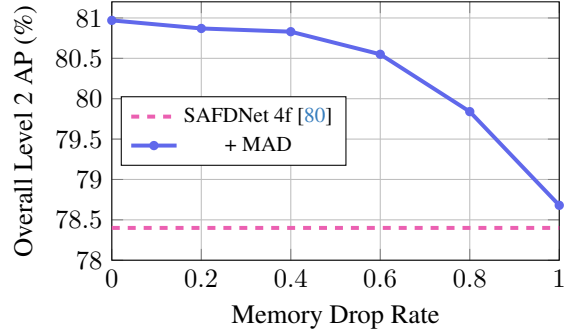
Performance across memory dropping rates: To show the importance of the memory at inference time, we investigate how the performance of MAD changes when we artificially drop memory proposals at inference time. We randomly drop memory target past timestamps $\mathcal{T}_m$ with a given probability, and Figure 4 plots model performance as a function of this dropping probability. Our method is relatively robust up to a dropping probability of 0.6 because much of the information provided by the memory proposals is redundant. Removing all target timestamps $\mathcal{T}_m = \emptyset$ causes the performance to drop near the base detector, as expected. The refinement transformer without any memory (dropping probability = 1.0) results in *slight* improvements over the base detector. We hypothesize this is because our refinement transformer without memory is the second stage of a two-stage detector, which have been shown to improve performance [75].

Forecasting Results: While our focus is task of 3D object detection, MAD also performs trajectory forecasting, which is an important and well-studied sub-task of autonomous driving systems [5, 10, 11, 44, 45, 59]. In Tab. 10, we investigate the effect of our memory mechanism on trajectory forecasting performance on the WOD validation set by training MAD to enhance HEDNet 1f and HEDNet 4f [81] on the WOD validation set. We also train a version where we remove the memory from the refinement transformer; no memory proposals $\mathcal{P}^{\text{mem}}$ (using $\mathcal{P}^{\text{ref}(0)} = \mathcal{P}^{\text{det}}$), and no memory cross-attention, so that the model is still able to generate trajectory forecasts but can't leverage memory inputs. The metrics in Tab. 10 are computed at a common detection recall point of 80% at an IoU threshold of 0.5. Miss-rate (MR) measures the percentage of predicted trajectory endpoints that are more than 2m away from the ground truth trajectory endpoint. Average displacement error (ADE) measures the average Euclidean distance between the predicted trajectory and the ground truth trajectory across all waypoints in the trajectory, and final dis-

Method	Overall L1 AP	Overall L1 APH	Overall L2 AP	Overall L2 APH	Vehicle L1 AP	Vehicle L1 APH	Vehicle L2 AP	Vehicle L2 APH	Pedestrian L1 AP	Pedestrian L1 APH	Pedestrian L2 AP	Pedestrian L2 APH	Cyclist L1 AP	Cyclist L1 APH	Cyclist L2 AP	Cyclist L2 APH
CenterFormer [82]	82.3	80.9	77.6	76.3	85.4	84.9	78.7	78.3	85.2	82.5	80.1	77.4	76.2	75.3	74.0	73.2
BEVFusion [36]	82.7	81.4	77.7	76.3	84.9	84.6	77.9	77.5	84.7	82.0	79.1	76.4	78.5	77.5	76.0	75.1
MSF [15]	83.1	81.7	78.3	77.0	86.1	85.7	79.2	78.8	86.0	83.1	80.6	77.8	77.3	76.4	75.1	74.3
FSD++ [13]	83.5	82.1	78.4	77.1	84.5	84.1	77.1	76.7	84.5	81.6	79.0	76.2	81.4	80.5	79.2	78.3
Octopus_Noah	83.1	81.7	78.7	77.3	85.5	85.1	80.2	**79.8**	84.3	81.2	78.7	75.7	79.5	78.8	76.9	76.3
SEED-L [38]	83.5	82.1	78.7	77.3	84.3	83.9	77.5	77.1	85.2	82.3	79.9	77.0	80.9	80.1	78.7	77.3
LION [37]	83.7	82.4	78.7	77.4	84.7	84.3	77.2	76.9	87.2	84.5	81.9	79.3	79.2	78.3	76.9	75.9
VeuronNet3D	83.7	82.2	79.1	77.7	85.7	85.2	79.1	78.7	85.3	82.3	80.3	77.4	80.1	79.1	77.9	76.7
HIAC	84.0	82.6	79.1	77.8	86.2	85.8	79.4	79.0	86.1	83.4	80.7	78.1	79.6	78.6	77.2	76.3
InceptioLidar	83.8	82.5	79.2	77.8	-	-	-	-	-	-	-	-	-	-	-	-
VADet	84.1	82.8	79.4	78.2	86.4	85.9	79.8	79.4	85.7	83.3	80.4	78.1	80.1	79.2	77.9	76.4
MT3D	85.0	83.7	80.1	78.7	**86.8**	**86.4**	79.8	79.5	86.9	84.1	81.4	78.6	81.4	80.5	78.9	77.4
LIVOX Detection	84.8	83.5	80.2	79.0	-	-	-	-	-	-	-	-	-	-	-	-
MAD (Ours)	**86.0**	**84.3**	**81.8**	**80.2**	86.5	85.9	**80.3**	79.7	**87.9**	**85.4**	**83.4**	**80.8**	**83.5**	**81.6**	**81.8**	**79.9**

Table 8. A snapshot of the WOD 3D detection leaderboard https://waymo.com/open/challenges/2020/3d-detection/. We exclude entries that state they use ensembles, test-time augmentations, or are offline (use future sensor data). "Ours" is using SAFDNet 4f as the 3D detector. Overall APH L2 is the ranking metric. Dashes "-" denote unavailable results due to missing link on the WOD leaderboard.

	Method	Overall L1 AP	Overall L1 APH	Overall L2 AP	Overall L2 APH	Vehicle L1 AP	Vehicle L1 APH	Vehicle L2 AP	Vehicle L2 APH	Pedestrian L1 AP	Pedestrian L1 APH	Pedestrian L2 AP	Pedestrian L2 APH	Cyclist L1 AP	Cyclist L1 APH	Cyclist L2 AP	Cyclist L2 APH
Validation	3D-MAN [74]	-	-	-	-	74.5	74.0	67.6	67.1	-	-	-	-	-	-	-	-
	MoDAR [30]	-	-	-	72.5	81.0	80.5	73.4	72.9	83.5	79.4	76.1	72.1	-	-	-	-
	LEF [17]	79.6	79.2	71.4	70.9	-	-	-	-	-	-	-	-	-	-	-	-
	MPPNet [8]	81.6	81.1	76.0	74.8	82.7	82.3	75.4	75.0	84.7	82.3	77.4	75.1	77.3	78.7	75.1	74.5
	MSF [15]	82.2	80.7	76.8	75.5	82.8	82.0	75.8	75.3	85.2	82.2	78.3	75.6	78.5	77.7	76.3	75.5
	PTT [20]	82.7	80.7	77.7	75.7	83.7	83.2	76.3	75.8	85.9	83.0	78.9	76.0	78.5	77.8	76.0	75.3
	MAD (Ours)	**85.8**	**84.2**	**81.0**	**79.4**	84.2	83.6	77.4	76.8	87.9	85.4	82.2	79.7	85.3	83.7	83.3	81.7
Testing	3D-MAN [74]	49.6	48.1	44.8	43.4	78.7	78.3	70.4	70.0	70.0	66.0	64.0	60.3	49.6	48.1	44.8	43.4
	MPPNet [8]	81.8	80.6	76.9	75.7	84.3	83.9	77.3	76.9	84.1	81.5	78.4	75.9	77.1	76.4	74.9	74.2
	MSF [15]	83.1	81.7	78.3	76.9	86.1	85.7	79.2	78.8	85.9	83.1	80.6	77.8	77.3	76.4	75.7	74.3
	MAD (Ours)	**86.0**	**84.3**	**81.8**	**80.2**	86.5	85.9	80.3	79.7	87.9	85.4	83.4	80.8	83.5	81.6	81.8	79.9

Table 9. Comparison of our method against various methods for learned temporal fusion on WOD. "Ours" is using SAFDNet 4f.

Detector	Memory	MR (%)	ADE (m)	FDE (m)
HEDNet 1f	✗	23.4	2.5	4.81
HEDNet 1f	✓	18.1	0.89	2.05
HEDNet 4f	✗	18.4	0.86	2.06
HEDNet 4f	✓	**17.2**	**0.72**	**1.75**

Table 10. Measuring the effect of memory on trajectory forecasting performance on the Vehicle class in the WOD validation set. Metrics are computed on 5s forecasts at 0.5 Hz, at a recall threshold of 80% with an IoU threshold of 0.5. Miss rate (MR) is computed at a distance threshold of 2 m.

placement error (FDE) measures the Euclidean distance between the predicted trajectory endpoint and the ground truth trajectory endpoint. We find that memory improves trajectory forecasting performance for both HEDNet 1f and HEDNet 4f. Note that, because HEDNet 4f uses multiple past LiDAR sweeps, it has better sensor evidence on the motion of vehicles. This is why it results in better trajec-

tory forecasting performance than HEDNet 1f; the interpolated features $\mathbf{Q}^{\text{det}}$ have more motion information which is useful to the refinement transformer. Notably, HEDNet 1f with memory matches or outperforms HEDNet 4f without memory, showing the strength of our memory mechanism in capturing object motion information.

Additional training design experiments: In Tab. 11 we investigated additional training procedures for MAD. For the experiments in this table we train for 45k iterations to reduce costs. For fair comparisons, all methods in Tab. 11 were trained with a fixed chunk length of 144 frames and no cache. Row 3 uses the architecture of MAD described in Sec. 3.1. There are four alternative design questions we investigated:

- If we accumulate gradients over the chunk length and update the model parameters once per chunk, does it improve performance? Our hypothesis was that this could reduce over-fitting to sequences of correlated training examples. However, comparing row 0 to row 3 of Tab. 11,

	Accumulating Gradients	$\mathbf{Q}^{\text{ref}}$	BPTT	AP L1	APH L1	AP L2	APH L2
0	✓	✗	✗	83.8	82.2	78.7	77.1
1	✗	✓	✗	85.2	83.7	80.3	78.8
2	✗	✓	✓	**85.3**	83.8	80.4	78.9
3	✗	✗	✗	85.3	**83.8**	**80.4**	**79.0**

Table 11. Comparison of different training strategies on the WOD validation set. All trainings are for 45k iterations using a chunk length of 144 and no cache.

we find that accumulating gradients hurts performance. For this experiment, while we use the same number of model forward passes (45k), there are less model parameter updates (only once per chunk), resulting in an undertrained model, even if the learning dynamics are better. Training with 45k model parameter updates steps in this setting would multiply the training duration and cost by the chunk length, which is prohibitively expensive.

- Can we improve performance by initializing the memory proposal features $\mathbf{Q}^{\text{mem}}$ with refinement features in the memory bank $\{\mathbf{Q}^{\text{ref}}_{t-s_m}, \ldots \mathbf{Q}^{\text{ref}}_{t-s_m T_m}\}$? Recall from Sec. 3.1 that $\mathbf{Q}^{\text{mem}}$ is calculated from the boxes $\mathbf{B}^{\text{mem}}$, confidence scores $\mathbf{C}^{\text{mem}}$, and trajectory forecasts $\mathbf{T}^{\text{mem}}$. However, the memory also stores past refined query features $\{\mathbf{Q}^{\text{ref}}_{t-s_m}, \ldots \mathbf{Q}^{\text{ref}}_{t-s_m T_m}\}$, and our hypothesis was that initializing $\mathbf{Q}^{\text{mem}}$ with these refined query features could improve performance because they contain richer learned information. To test this, we compute $\mathbf{Q}^{\text{mem}}$ using $\mathbf{Q}^{\text{ref}}$ and adding an encoding of the box, confidence score, and trajectory forecasts (the same way we computed the memory proposal features described in Sec. 3.1). However, comparing row 0 to row 3 of Tab. 11, we find that using the refined query features does not improve performance over our proposed $\mathbf{Q}^{\text{mem}}$ described in Sec. 3.1. We think this is because the distribution of these features is changing over training, which makes them difficult to learn with.

- To address the lack of improvement from using the refined features, we investigated using BPTT through these features to allow the model to learn to output refinement features that are relevant for future inferences. Due to GPU memory constraints, we could only back-propagate through one past inference. Unfortunately, we found that this did not improve performance significantly over not using BPTT (2 vs 1 in Tab. 11), or over not using refinement features to initialize the memory features (2 vs 3 in Tab. 11). We have two hypotheses for why this is: The first is that the position, size, confidence, and trajectory information in the memory features are important and easy for the refinement transformer to use, overpowering the information in the refinement features. Secondly, BPTT through one step might not be enough to see improvements. This is an area for future work.

Self Attention	Veh. AP	Ped. AP	Cyc. AP
✗	76.8	81.9	82.2
✓	**77.0**	**82.3**	**83.3**

Table 12. Ablating the use of self attention in the refinement transformer on the WOD validation set. All metrics are L2.

Model	GigaFLOPs	GPU Mem (GB)
SAFDNet 4f	880	2.5
MAD Module	50	2.8

Table 13. FLOPs and peak memory measurements per forward pass for SAFDNet 4f and MAD.

Effect of self-attention: Table 12 illustrates the effect of using self-attention in the refinement transformer, which brings a small but consistent performance improvement. Recall that the memory cross attention only allows each query $\mathbf{Q}^{\text{ref}(i)}$ to attend to the nearest $k = 4$ memory proposals features $\mathbf{Q}^{\text{mem}}$. Self-attention lets queries see information from more distant memory proposals because information about memory proposals can propagate through all query features which can be anywhere in the scene. Furthermore, self-attention allows for modelling interactions between actors, and prior works have shown it to improve forecasting performance [4], which would in turn improve MAD's detection performance through the memory. Both of these factors could explain the improved performance of MAD.

Computational Overhead: We report total FLOPs and peak memory measurements per forward pass, taking the maximum across 10 random sequences on WOD in Tab. 13. We use FLOPs because it is more hardware and implementation agnostic. There is minimal computational overhead from MAD relative to the base detector. We also note that in most traditional self-driving systems, the 3D detector would be followed by a forecasting module, which MAD can replace.

Performance on Stationary Objects Table 14 shows the performance of MAD on stationary objects on WOD (≤ 0.5 m/s). As expected, forecasting provides minimal benefit for

| | **Stationary** | | |
	Veh. AP	Ped. AP	Cyc. AP
SAFDNet 4f (Base)	73.8	54.1	34.3
MAD No Forecasting	**74.9**	**56.2**	35.9
MAD	74.8	56.1	**36.9**

Table 14. Performance on stationary objects on WOD (≤ 0.5 m/s).

detecting stationary objects, and both versions of MAD provide improvements over the base detector.

8 Additional Qualitative Results:

See Figs. 5 and 6 for additional qualitative results on the WOD validation set. We show the memory bank with the boxes $\mathbf{B}^{\text{ref}}_{t_m}$ with all target timestamps $\mathcal{T}_m$ overlayed, the memory proposal bounding boxes $\mathbf{B}^{\text{mem}}$ (recall these are computed with interpolation along the trajectory forecast in them memory bank), the detection proposal bounding boxes $\mathbf{B}^{\text{det}}$, and model output bounding boxes $\mathbf{B}^{\text{ref}}$. To visually compare the detection proposals and MAD's outputs, we filter their boxes with a threshold on the confidence scores, using the threshold that attains the max F1 score on the WOD validation set (this threshold is computed separately for the detection proposals and MAD's outputs). This is a fair comparison as during deployment we need to choose a single operating point for the detector. The memory boxes are thresholded at a confidence score of 0.1, and colored by their age $t_m - t$. We list some interesting themes which we circle in the figure:

a) Improved recall on stationary or slow moving vehicles.
b) Improved recall on fast moving vehicles.
c) Accurate memory trajectory forecast and interpolated position.
d) Improved recall on pedestrians or cyclists.

We notice that the memory has excellent recall on most objects, and that for most objects the interpolated boxes (and thus trajectory forecasts) are quite accurate.

8.1. Limitations and Areas for Future Work

Trajectory Forecasting: MADs trajectory forecasts for turning vehicles can be inaccurate. For example, in Scene #4 of Fig. 5, there is a right-turning vehicle which has inaccurate interpolated memory. This is because we do not use any map information in our model, and we only predict a single future trajectory for each agent. Works in trajectory forecasting often incorporate map information and predict multiple possible future trajectories for each agent [4, 10]. There is room for future work on bringing these improvements to MAD.

Using Refinement Features and BPTT: We think that using refinement features stored in the memory in conjunction with BPTT should improve performance. For example,

these features could store more nuanced information about object appearance and uncertainty, which would be relevant for deciding if current evidence from the detection proposals is similar to evidence in the memory. Unfortunately, we did not see improvements when using the refinement features or BPTT in our experiments (Tab. 11), which leaves room for future improvements.

Data Limitations: We observe that on WOD and AV2, there are many cases where labels disappear, possibly due to occlusion. An example of this is shown in Fig. 7, where we see a group of parked vehicles at time t, but they are gone by time $t + 0.3s$, although they are stationary and in the region of interest. This provides confusing supervision for temporal models like MAD, as it must learn when to recall an object or not, possibly based on estimated visibility. A dataset with labels that are temporally consistent would be beneficial for future work in temporal object detection.

Generalization Across Base Detectors: As discussed in Sec. 3.2, currently MAD is trained separately for each base detector. Developing a version of MAD that can be trained once and generalize across multiple base detectors is an exciting direction for future work and holds promise for broader applicability.
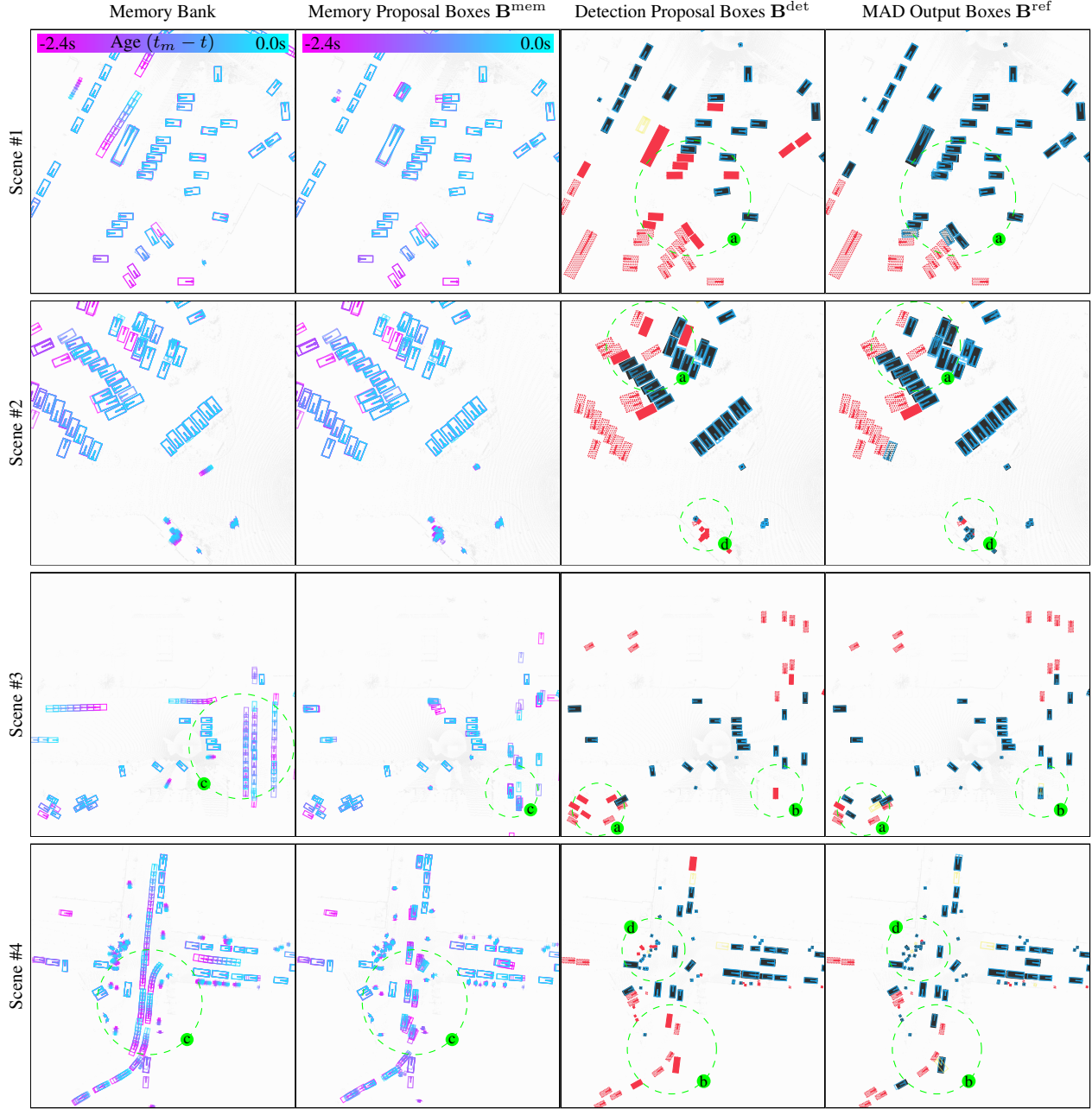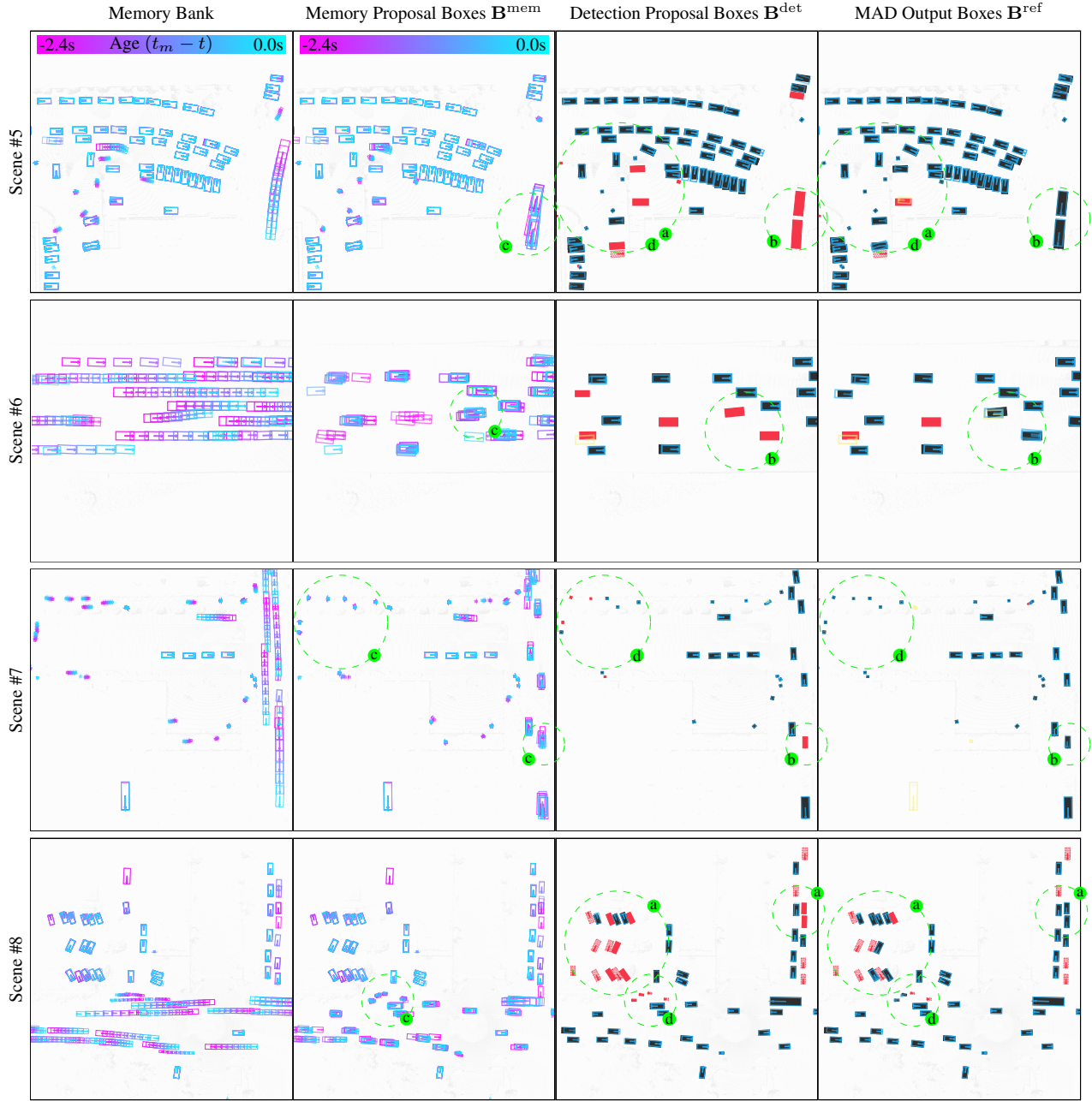
Figure 5. Visualization of bounding box proposals of MAD enhancing CenterPoint 1f [75] on the WOD validation set. We threshold at the max f1 score for the detection proposals and model outputs. Filled boxes are labels: black are matched labels, red boxes are false negatives, and dotted boxes have < 1 LiDAR point observation. Box outlines are predictions: blue boxes are true positive proposals, and yellow boxes are false positives. Annotations which we refer to in the main text are in green. For computing the matching boxes in visualization, we use IoU thresholds of 0.5, 0.1, and 0.1 for Veh., Ped., and Cyc., respectively.

6

Figure 6. Visualization of bounding box proposals of MAD enhancing CenterPoint 1f [75] on the WOD validation set. We threshold at the max f1 score for the detection proposals and model outputs. Filled boxes are labels: black are matched labels, red boxes are false negatives, and dotted boxes have < 1 LiDAR point observation Box outlines are predictions: blue boxes are true positive proposals, and yellow boxes are false positives. Annotations which we refer to in the main text are in green. For computing the matching boxes in visualization, we use IoU thresholds of 0.5, 0.1, and 0.1 for Veh., Ped., and Cyc., respectively.
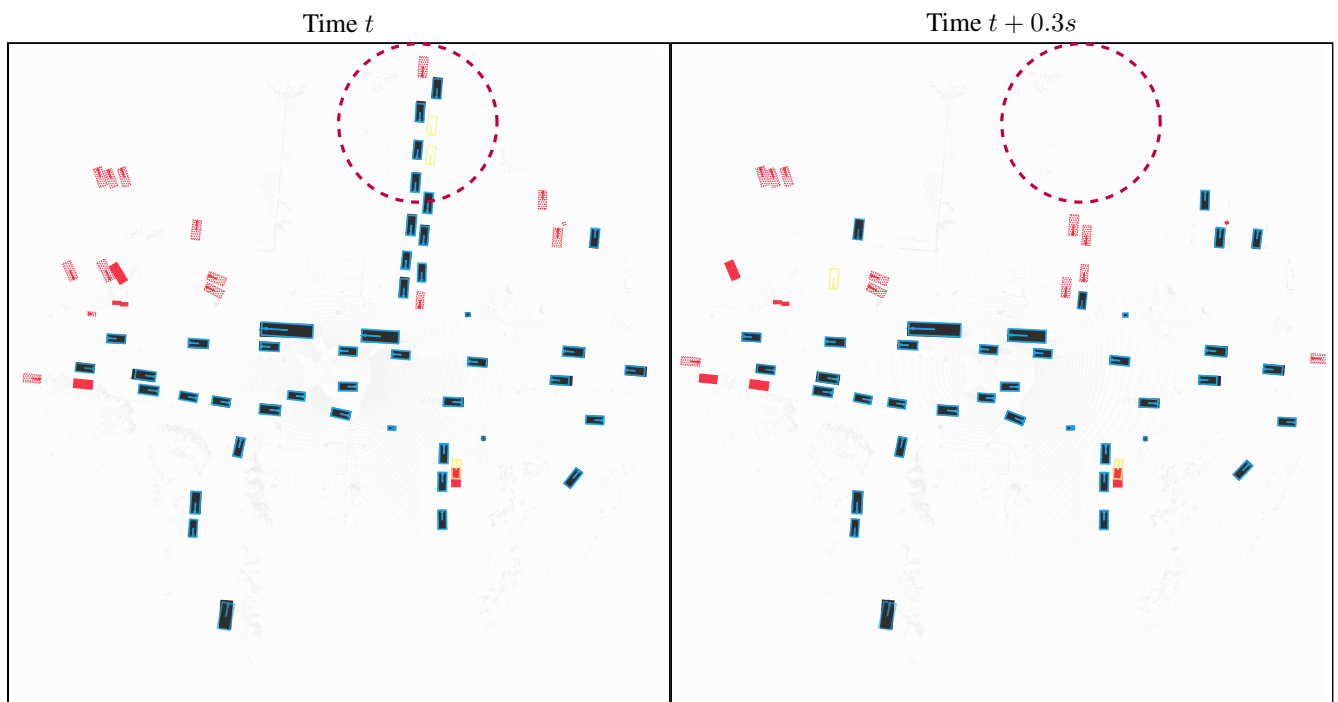
7

Figure 7. An illustration of disappearing labels, circled in purple, from the WOD dataset. Labels are shown with filled boxes (same as Figs. 5 and 6).