[Supplementary] DeTra: A Unified Model for Object Detection and Trajectory Forecasting

Sergio Casas[⋆], **Ben Agro**[⋆], **Jiageng Mao**[⋆†],
Thomas Gilles, Alexander Cui[†], **Thomas Li, Raquel Urtasun**

Waabi, University of Toronto
{sergio, bagro, tgilles, tli, urtasun}@waabi.ai

This appendix describes implementation details, including architecture and training, baselines, and metrics in Appendix A.

Additionally, we perform additional ablation studies in Appendix B to understand the effect of different numbers of transformer blocks, different orderings of the attention modules of DETRA, and varying the mode and time dimensions of the learnable query volume.

Appendix C provides visualizations of DETRA compared to the baselines on the Waymo Open Dataset (WOD).

Finally, Appendix D describes the limitations of our work and opportunities for future work.

A Implementation Details

A.1 LiDAR Encoder

The LiDAR encoder takes $H = 5$ past LiDAR sweeps to produce multi-resolution Birds Eye View (BEV) feature maps that serve as the LiDAR tokens for the refinement transformer. Below, we describe the architecture of the LiDAR encoder, illustrated in Figure 1, where we omit the batch dimension for simplicity.

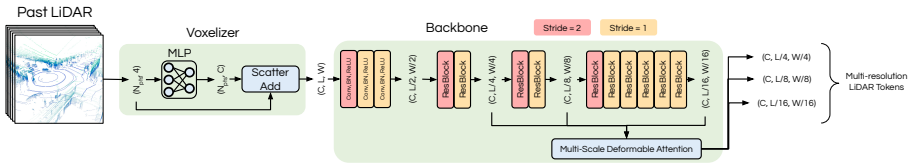

Fig. 1: The architecture of the LiDAR Encoder. The batch dimension is omitted from the tensor shapes. N_{pts} is the number of points, C is the number of channels, and L and W are the spatial dimensions in BEV. The four features of each point are (x, y, z, t).

⋆ Denotes equal contribution
† Work done while at Waabi

Voxelization: Each LiDAR sweep is a set of points with four features (x, y, z, t). A small MLP is used to encode these points as feature vectors of size $C = 128$. The LiDAR points are placed in a 2D BEV grid of shape (L, W) based on their (x, y) coordinates with a simple sum aggregator. We use an ROI of $[-40, 40]$ m on the x and y dimensions for AV2, and $[-80, 80]$ m on the x and y dimensions for WOD. We use a voxel size of $0.1\,\mathrm{m}$, resulting in $L = \frac{80}{0.1} = 800, W = \frac{80}{0.1} = 800$ for AV2, and $L = \frac{160}{0.1} = 1600, W = \frac{160}{0.1} = 1600$ for WOD. A 2D feature map of shape (C, L, W) is then generated from this BEV grid, where each grid cell has a feature vector that is the sum of the features from all points in that grid cell.

Backbone: This 2D feature map is processed by three convolution layers interleaved with batch-normalization and ReLU layers, The first convolutional layer has a stride of 2, resulting in a feature map of shape $(C, L/2, W/2)$. Then, this feature map is processed by a series of ten residual layers, labeled *ResBlock* in Figure 1, which each employ a sequence of dynamic convolution [2], batch-normalization, ReLU, dynamic convolution, batch-normalization, squeeze-and-excitation [4], and dropout operations. Each residual layer produces a feature map, and layers 0, 2, and 4 down-sample their output by a factor of 2 (using convolutions with stride 2). We extract three multi-level feature maps from the output of layers 1, 3, and 9 with shapes $(C, L/4, W/4)$, $(C, L/8, W/8)$, and $(C, L/16, W/16)$ respectively, across which information is fused with multiscale deformable attention [11] to produce three feature maps with the same shapes as the input.

A.2 Map Encoder

We follow [3, 5] for lane graph generation and map encoding. Specifically, our lane graph is constructed by selecting centerlines from HD maps at fixed intervals of $3\,\mathrm{m}$. These centerlines are divided into lane segments, each represented by a node in the graph with features including length, width, curvature, speed limit, and lane boundary type (e.g., solid or dashed). To establish connections between lane nodes, we employ four types of relationships: successors, predecessors, left neighbors, and right neighbors. We leverage the graph neural network in [3] to encode the lane graph nodes with pair-wise relative positional encodings and use the map node embeddings as the map tokens.

A.3 Initial pose header

To produce the initial poses $\mathbf{P}^{(0)}$, we use a small Feature Pyramid Network [7] where the three multi-resolution feature maps are fused into a single feature map of shape $(C, L/4, W/4)$. Then, four convolution layers interleaved with batch-normalization and ReLU layers predict a heatmap of bounding-box parameters, and a separate set of four convolutional layers predict a heatmap of detection scores. These heatmap outputs are turned into object detections via NMS with an IOU threshold of 0.1.

A.4 Metrics

OccAP: To compute this metric, we do the following:

1. Generate a ground truth spatio-temporal occupancy grid of shape (T, L, W), where T is the number of forecasted timesteps, and L and W are the spatial dimensions defined in Appendix A.1. Each grid cell is filled with 0.0 if it is unoccupied and 1.0 if it is occupied, using the ground truth future trajectories for each object to compute this occupancy.
2. Generate a predicted spatio-temporal occupancy grid of shape (T, L, W), where the values in the grid cells range from $[0.0, 1.0]$ and represent the probability that an object occupies that cell. To obtain these probabilities, we rasterize all F modes of each detected object, where each mode has an occupancy probability equal to the product of its detection confidence and mode confidence. We deal with overlapping modes appropriately, setting the probability in the grid cell equal to 1 minus the probability that no mode occupies the cell.
3. Compute the average precision (AP) of the predicted occupancy grid with respect to the ground truth occupancy grid. For 100 evenly spaced threshold values between 0 and 1, we compute the number of true positives, false positives, and false negatives and then compute the precision and recall at each threshold. We compute AP as the area under the precision-recall curve.

TrajAP: This metric closely follows the Soft AP metric used in the Waymo Motion Prediction Challenge [9][1], but we make the following modifications to adapt it to our detection-trajectory forecasting setting. First, we threshold all detections at the threshold that attains their maximum F_1 score. We consider a trajectory as a match only if its detection matches with a specific IOU score *and* its forecasted trajectory matches (using the same definition in the original Soft AP metric). The final metric we report is a macro average across future time horizons of $3\,\mathrm{s}$ and $5\,\mathrm{s}$, detection IOU thresholds of 0.5 and 0.7, and static and dynamic actors.

To compute the metric on static (or dynamic) actors only, we set the dynamic (or static) actors to "ignore", and any detection or trajectory forecast that matches with an ignore label is removed (not counted).

A.5 Baselines

Detection: All baselines use the same LiDAR encoder as DeTra (see Appendix A.1) as part of their detection network. Likewise, their detections are produced with the same header as our prior object detection header described in Appendix A.3. The detection loss is the same as our initial pose loss described in the main paper.

Tracker: To associate detections over time and create tracks as input to MultiPath, LaneGCN, SceneTF, and GoRela, we use the online heuristics tracker described in [10][2]. We perform tracking in the forward direction so it can run online and provide a fair comparison against the end-to-end methods, which also run online.

[1] https://waymo.com/open/challenges/2023/motion-prediction/

[2] see the supplementary here https://arxiv.org/pdf/2311.02007.pdf for details

Trajectory Forecasting: We faithfully reproduce the trajectory forecasting models described in each method's paper. We can directly use the tracks as input for the modular detection-tracking-forecasting models because the trajectory forecasting methods were developed to ingest tracks. For these models' End-to-end (E2E) versions, we replace per-actor track features with per-actor LiDAR features. We applied a rotated ROI pool inside each detected bounding box on the BEV feature map output from the feature-pyramid network to obtain these features.

All methods use the same prediction loss as DETRA, described in the main paper: a Laplacian winner-takes-all loss on the mixture of trajectories and a cross-entropy loss on the mode probabilities, where the winner mode is the one with trajectory waypoints closest to the ground truth, measured with an L1 loss on the Laplacian centroids (μ_x, μ_y).

Training Details: We train all methods with the same learning rate schedule, optimizer, and batch size as DETRA, with equal relative loss weighting between the heatmap detection loss and the prediction loss.

B Additional Quantitative Results

What is the effect of refinement depth? Table 1 shows the performance of DETRA with different numbers of refinement blocks. We observe that the performance of DETRA improves from 1 to 3 refinement blocks but then plateaus. We expect different training recipes, hyper-parameters, and amounts of data would be necessary to scale DETRA, which we leave as room for future work.

# of refinement blocks	Joint		Detection			Forecasting						
	AP ↑		AP @ IoU ↑			MR ↓		ADE ↓		FDE ↓		bFDE ↓
	Occ	Traj	@0.3	@0.5	@0.7	$K=1$	$K=6$	$K=1$	$K=6$	$K=1$	$K=6$	$K=6$
1	71.2	46.7	95.5	92.3	78.6	38.7	18.7	1.76	0.61	4.07	1.33	1.99
2	72.5	48.4	95.8	**93.1**	80.1	37.7	16.5	1.62	0.57	3.77	1.21	1.88
3 (DETRA)	**73.0**	**50.0**	**95.8**	92.9	80.2	**37.0**	15.4	**1.54**	0.55	**3.59**	1.19	1.86
4	72.3	49.8	95.6	92.8	**80.8**	37.2	**18.7**	1.55	**0.54**	3.63	**1.14**	**1.81**
5	72.0	47.8	95.8	92.8	79.9	37.8	16.5	1.6	0.55	3.72	1.18	1.84

Table 1: DETRA's performance using different numbers of refinement blocks on AV2. Note that unlike Table 3 in the main paper, where one version of DETRA was trained with 3 refinement blocks and evaluated after each refinement block, here we trained 5 different versions of DETRA and evaluated the final output.

Does the ordering of the attention layers matter? Table 2 shows the performance of DETRA with different orderings of the attention layers. The main finding from this is that it is important for the self-attention modules (time, mode, object) to come after the cross-attention modules (lidar and map). This result makes sense intuitively because the self-attention modules are meant to propagate sensor information (obtained in the cross-attention modules) across the query volume, and they cannot do that if applied

before the cross-attention modules. Besides that, the order within cross-attention (lidar and map) or self-attention (time, future mode, object) matters little. However, we find the selected order of lidar, map, time, future, and object to be the best in most metrics.

Ordering	Joint		Detection			Forecasting						
	AP ↑		AP @ IoU ↑			MR ↓		ADE ↓		FDE ↓		bFDE ↓
	Occ	Traj	@0.3	@0.5	@0.7	$K=1$	$K=6$	$K=1$	$K=6$	$K=1$	$K=6$	$K=6$
L, M, O, T, F	72.1	49.9	95.7	92.8	**80.5**	**36.9**	**15.2**	1.54	0.56	3.59	1.19	1.86
T, F, O, L, M	71.3	47.6	95.7	92.7	79.9	38.1	17.4	1.65	0.58	3.84	1.26	1.92
M, L, T, F, O	72.7	**50.5**	95.5	92.8	79.6	37.1	16.2	1.55	0.55	3.62	1.18	1.85
DETRA (L, M, T, F, O)	**73.0**	50.0	**95.8**	**92.9**	80.2	37.0	15.4	**1.54**	**0.55**	**3.59**	**1.19**	1.86

Table 2: DETRA's performance using different orderings of the attention blocks on AV2. Here L, M, T, F, and O stand for the *LiDAR*, *Map*, *Time*, *Mode*, and *Object* attention modules, respectively.

How does DETRA perform with different learnable query volume dimensions? Table 3 shows the performance of DETRA with different query volume dimensions. Specifically, we experiment with varying the mode dimensionality $F \in \{1, 6\}$ and time dimensionality $T \in \{1, 10\}$. Note that the output trajectories still have ten timesteps and six modes regardless of the input query volume dimensions because we adjust the output dimensionality of the MLP in the pose update accordingly. The time dimension $T = 10$ is crucial for good trajectory forecasting performance, e.g., see the OccAP and TrajAP metrics. This result makes sense because gathering information from future time queries about the map and interactions with other objects is essential for trajectory forecasting. The mode dimension $F = 6$ is also important for trajectory forecasting performance, particularly in the forecasting metrics at $K = 6$, which is expected because having the mode dimension from the beginning (as opposed to just as part of the MLP decoder) allows these modes to differentiate more from each other in meaningful ways, enabling learning of more diverse trajectory forecasts that cover multiple possible futures. Detection gets slightly degraded at high IoUs when using multiple modes or time steps, which shows that there may be room for improvement in the mode aggregation over query features and potentially using causal attention in the time dimension.

Query Volume Dims	Joint		Detection			Forecasting						
	AP ↑		AP @ IoU ↑			MR ↓		ADE ↓		FDE ↓		bFDE ↓
	Occ	Traj	@0.3	@0.5	@0.7	$K=1$	$K=6$	$K=1$	$K=6$	$K=1$	$K=6$	$K=6$
$T=1, F=1$	50.1	40.0	**95.9**	**93.0**	**80.6**	41.8	40.6	2.7	2.28	5.76	4.74	5.1
$T=10, F=1$	71.7	45.5	95.8	92.7	79.3	39.3	19.9	1.79	0.64	4.16	1.4	2.05
$T=1, F=6$	67.1	45.4	95.5	92.3	78.4	42.0	22.9	2.06	0.92	4.47	2.26	2.92
DETRA $(T=10, F=6)$	**73.0**	**50.0**	95.8	92.9	80.2	**37.0**	**15.4**	**1.54**	**0.55**	**3.59**	**1.19**	**1.86**

Table 3: DETRA's performance using different query volume dimensions on AV2.

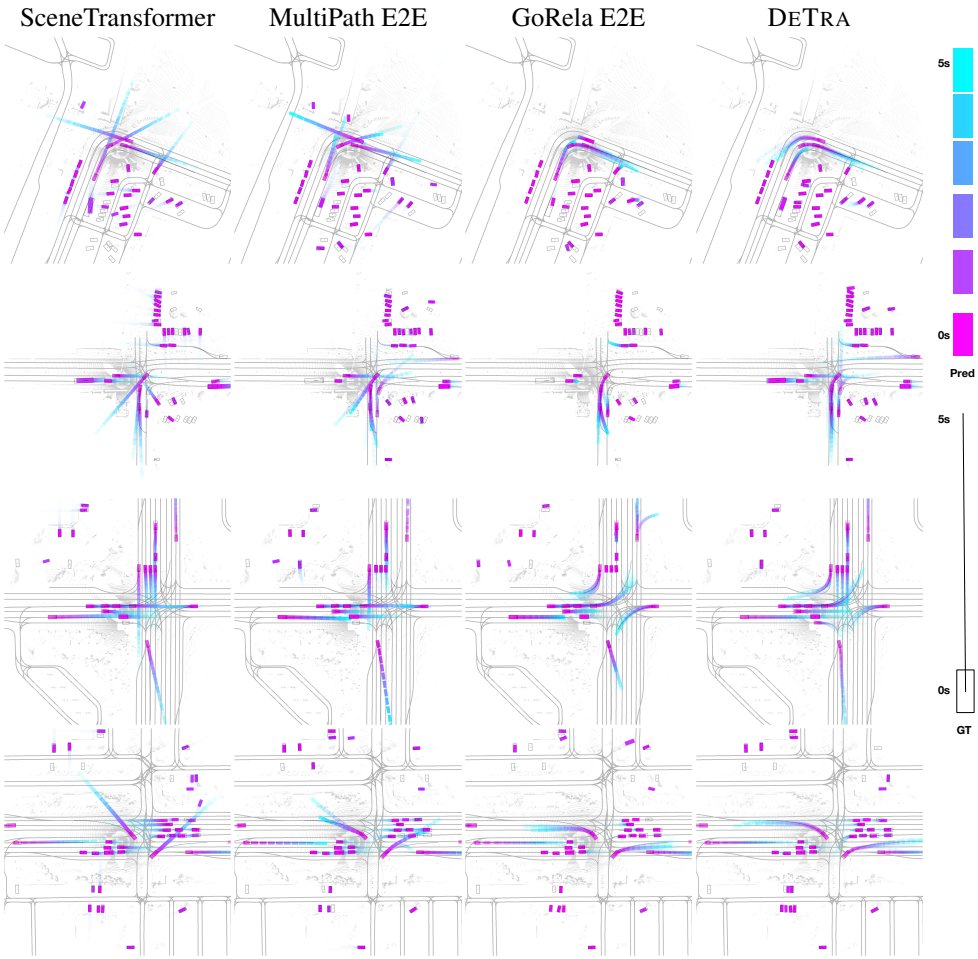

Fig. 2: Qualitative results on WOD.

C Additional Qualitative Results

Figure 2 shows qualitative results on WOD for DETRA and the baselines. Similarly to the results on AV2 shown in the main paper, DETRA exhibits more accurate trajectory forecasts and detections than the baselines.

Figure 3 shows the self-improvement of DETRA over transformer refinement blocks on WOD. Again, we see that the trajectories follow the map more accurately over transformer refinement blocks, and the detections become more accurate, with limited gains between the second-last and final refinement blocks.

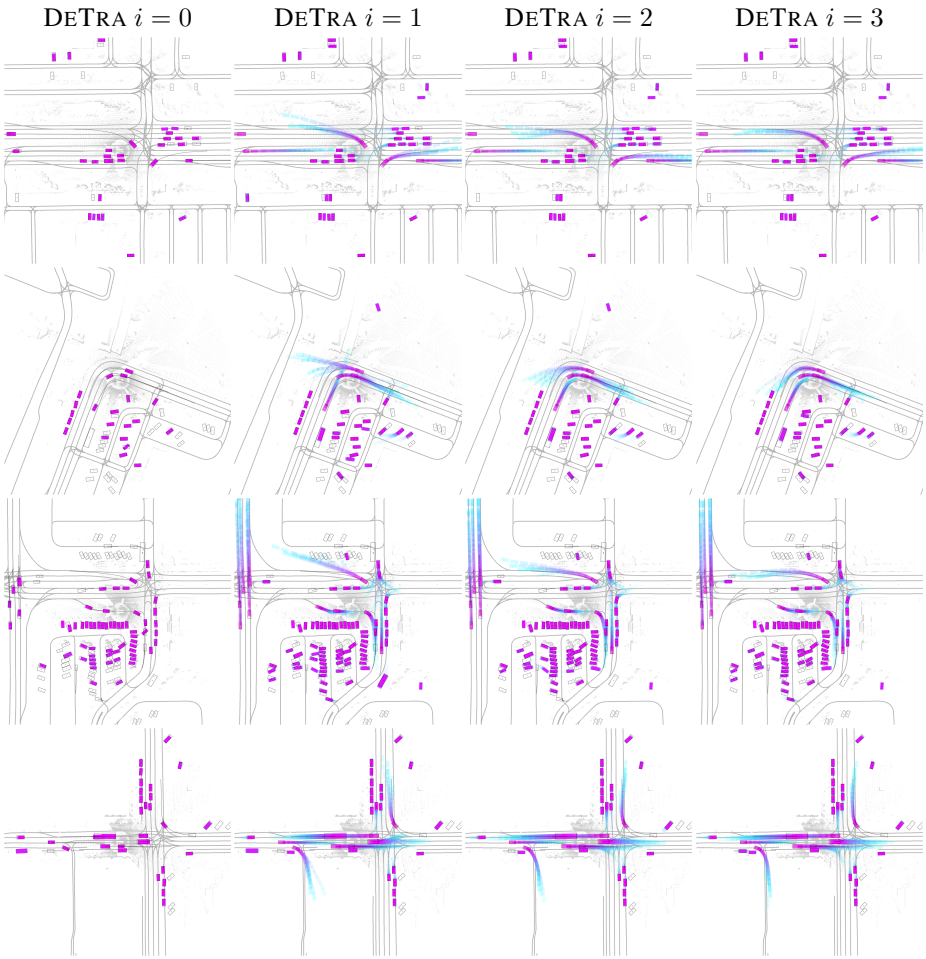

Fig. 3: Visualizing the DETRA's self improvement over refinement blocks on WOD.

D Limitations and Future Work

While our work provides a simpler approach to object detection and trajectory forecasting based on trajectory refinement that achieves better results than previous paradigms, it has several limitations.

DETRA does not consider uncertainty on its detections beyond the confidence score. This design follows the standard in detection literature, but there are likely better methods than this. Instead, future works could consider multiple detection hypotheses similar to how multiple modes are considered for motion forecasting. Our method is very suitable for this since we already have a query and pose volume with a mode dimension, but right now, all the poses at $t = 0$ are updated to be the same. The main challenge to extend detection to multi-hypothesis remains in developing strong objective functions for training and metrics for evaluation.

Another limitation is that in the current experiments, we did not consider the task of joint future forecasting across agents [1,8], only marginal forecasting for each agent. This task would be a good benchmark to evaluate the object-to-object interaction understanding of DETRA, but it is out of scope for this paper.

Finally, we did not tackle the problem of temporal consistency of detection and forecasts over multiple forward passes at consecutive frames. Traditional approaches track the detections and forecasting outputs, which could be applied to DETRA [6]. Future work may explore how to learn better temporal consistency in the DETRA framework.

Despite its limitations, we hope DETRA can lay a strong foundation for future work on end-to-end detection and forecasting.

References

1. Casas, S., Gulino, C., Suo, S., Luo, K., Liao, R., Urtasun, R.: Implicit latent variable model for scene-consistent motion forecasting. In: ECCV (2020) 8
2. Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., Liu, Z.: Dynamic convolution: Attention over convolution kernels. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11030–11039 (2020) 2
3. Cui, A., Casas, S., Wong, K., Suo, S., Urtasun, R.: Gorela: Go relative for viewpoint-invariant motion forecasting. arXiv preprint arXiv:2211.02545 (2022) 2
4. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018) 2
5. Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., Urtasun, R.: Learning lane graph representations for motion forecasting. In: ECCV. Springer (2020) 2
6. Liang, M., Yang, B., Zeng, W., Chen, Y., Hu, R., Casas, S., Urtasun, R.: Pnpnet: End-to-end perception and prediction with tracking in the loop. In: CVPR (2020) 8
7. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017) 2
8. Ngiam, J., Caine, B., Vasudevan, V., Zhang, Z., Chiang, H.T.L., Ling, J., Roelofs, R., Bewley, A., Liu, C., Venugopal, A., et al.: Scene transformer: A unified architecture for predicting multiple agent trajectories. arXiv preprint arXiv:2106.08417 (2021) 8
9. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: The waymo open motion dataset. https://waymo.com/open/motion/ (2020) 3
10. Zhang, L., Yang, A.J., Xiong, Y., Casas, S., Yang, B., Ren, M., Urtasun, R.: Towards unsupervised object detection from lidar point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9317–9328 (2023) 3
11. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020) 2