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

Abstract. The tasks of object detection and trajectory forecasting play a crucial role in understanding the scene for autonomous driving. These tasks are typically executed in a cascading manner, making them prone to compounding errors. Furthermore, there is usually a very thin interface between the two tasks, creating a lossy information bottleneck. To address these challenges, our approach formulates the union of the two tasks as a trajectory refinement problem, where the first pose is the detection (current time), and the subsequent poses are the waypoints of the multiple forecasts (future time). To tackle this unified task, we design a refinement transformer that infers the presence, pose, and multi-modal future behaviors of objects directly from LiDAR point clouds and high-definition maps. We call this model DETRA, short for *object Detection* and *Trajectory forecasting*. In our experiments, we observe that DETRA outperforms the state-of-the-art on Argoverse 2 Sensor and Waymo Open Dataset by a large margin, across a broad range of metrics. Finally, we perform extensive ablation studies that show the value of refinement for this task and that key design choices were made.

1 Introduction

To ensure the safe deployment of autonomous driving technology, self-driving vehicles (SDVs) must be able to perceive their surroundings and anticipate potential outcomes accurately. Object detection and trajectory forecasting tasks fulfill these two capabilities in autonomy stacks. Object detection aims to recognize and localize objects in the environment, while trajectory forecasting anticipates the future behaviors of those objects in the form of trajectories. Subsequently, a planner ingests the detected objects and their future trajectories to produce a safe maneuver for the ego vehicle.

Traditional autonomous systems tackle object detection and trajectory forecasting as separate tasks, connecting them in a cascaded fashion through tracking (Figure 1a). Tracking is a narrow information bottleneck, only providing downstream forecasting with noisy past trajectories describing the motion of each object. This cascading decomposition suffers from compounding errors; *e.g.* a small estimation error in the detection heading might result in a forecasted trajectory that occupies the wrong lane, potentially causing a dangerous maneuver by the SDV such as a hard brake. This has

---

[*] Denotes equal contribution
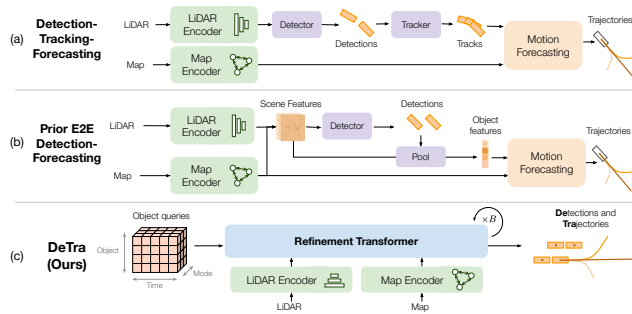
[†] Work done while at Waabi

Fig. 1: Modular detection-tracking-forecasting methods use narrow interfaces, hindering their performance due to compounding errors and a lossy information bottleneck. Prior end-to-end methods widen the interfaces and train jointly but still utilize a cascading structure, thus suffering from compounding errors. Our method tackles the problem as a unified trajectory refinement task.

motivated research to measure and incorporate perception uncertainty into forecasting [28]. Furthermore, tracking errors such as miss-associations can have catastrophic consequences, inspiring work to maintain multiple tracking hypotheses [56] throughout the entire pipeline.

Another line of work [4–6, 34, 39] has proposed a more fundamental way to address these issues by shifting to an end-to-end detection and forecasting paradigm where the two tasks share high-dimensional scene features and are jointly optimized (Figure 1b). Compared to traditional approaches where forecasting relies purely on object tracks, this paradigm can better propagate uncertainty downstream with a wider interface and intermediate representations optimized for both tasks. Nevertheless, these methods rely on a cascading inference where detection errors can still propagate to forecasting.

Our method circumvents the established cascading approach by reformulating the detection and forecasting steps into a single, more general trajectory refinement task in Bird's-Eye-View (BEV), as shown in Figure 1c. In most self-driving pipelines, the planner receives a set of object trajectories over time, making no distinction between the current time (detection) and the future (forecasting). Thus, we formulate our output to be a set of object trajectories, where each trajectory represents an object's pose in BEV from the current time into the future. In this formulation, detection is simply the particular case of the pose at the present time.

To tackle object detection and trajectory forecasting as a pose refinement problem, we utilize multi-object, multi-hypothesis and temporal learnable queries refined by attending to LiDAR point clouds and high-definition (HD) maps. However, naive global cross-attention to LiDAR and maps and self-attention between all queries are prohibitively expensive and difficult to optimize. To make cross-attention practical, we pair each of the queries with a pose that represents our belief about the BEV position of a particular *object*, at a specific *time*, for an individual future behavior *mode*, and perform local attention in a neighborhood instead. To make self-attention practical with three-dimensional queries, we factorize it into object, time, and mode attention.

We demonstrate the effectiveness of our approach in two popular self-driving datasets, Argoverse 2 Sensor and Waymo Open, where DETRA outperforms the state-of-the-art in a broad range of detection, forecasting, and joint metrics. Importantly, through ablation studies we also show that our proposed refinement mechanism is powerful, that every proposed component contributes positively to the overall method, and that key design choices were made across multiple components.

2   Related Work

Detection: Image-based object detectors traditionally leverage convolutions to extract features efficiently. These CNN-based approaches fall into two main classes: single-stage methods [37, 48, 63] that make predictions w.r.t a grid of possible centers, and two-stage detectors [2, 49] that have different networks first to generate proposals and then refine them. LiDAR-based detectors also leverage similar architectures for Bird's-Eye-View (BEV) detection [31,60,64,65]. More recently, DETR [3] redefines detection as a set prediction problem and uses a Transformer [55] architecture coupled with a bipartite matching loss. Deformable-DETR [68] leverages reference points for each of the objects so that the attention layers focus on the local context around these. DAB-DETR [36] extends this framework by feeding a DETR-based model with geometry-based anchor proposals, updating these spatial queries after each attention layer. We build on top of these works by encoding LiDAR point clouds through a CNN backbone [60] and leveraging set-based transformers [3] with deformable attention for efficient geometric priors [68].

Forecasting: High-definition (HD) maps and object past trajectories (tracks) are usually employed to forecast multiple BEV trajectories per object. The inputs can either be represented as a raster and processed with a CNN [8, 14, 45] or in vectorized form and processed with Graph Neural Networks (GNN) [13, 15, 33, 42] or Transformers [19–22,42–44,61,67]. Most prior works decode multiple futures for each object by first encoding these inputs into object features. AutoBots [21] and SceneTransformer [44] pioneered the use of mode queries: learnable parameters that separate distinct futures earlier in the network so the futures modes can interact with each other. QCNet [66] recently took this one step further, exploiting mode queries from the beginning of the network to predict an initial set of anchor trajectories, which they leverage to gather local information and refine once. However, these mode queries are stripped of temporal information, limiting their expressivity. Inspired by this trend, we leverage a volume of learnable queries with object, mode, and time dimensions from the start of the network and generalize the refinement so that it can be done iteratively without any necessary changes to the design of subsequent blocks.

End-to-end (E2E) Detection and Forecasting: Early attempts adopted a single-stage method by adding a lightweight forecasting header to an existing detection framework [6, 39]. More recent E2E detection and forecasting works use two-stage architectures [4, 5, 16, 27, 32, 34, 41, 52], where the second stage performs relational reasoning between objects and the map. While their details differ, these models share a common
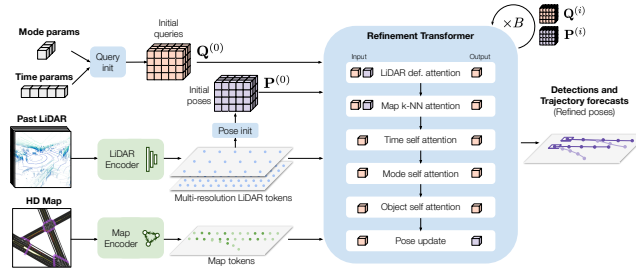
Fig. 2: DETRA uses object queries and poses to represent objects' trajectories from the present time into the future. DETRA refines the initial estimates of the object queries and poses over B blocks. Each refinement transformer block consists of several attention layers followed by a pose update. The set of poses at the end of the B-th block are the final detections and forecasts.

framework. First, an encoder extracts shared features from raw sensor data. Next, a perception module predicts a set of candidate detections from the shared features. Then, a feature vector representing the object detections is extracted from the shared features by a pooling operation. Finally, a forecasting module takes the detections and their feature vectors and predicts future trajectories. While these two-stage architectures are more powerful and can propagate uncertainty via end-to-end training, they also partially reintroduce the compounding errors from the modular pipeline by cascading the detection and forecasting stages. In contrast, our approach tackles detection and forecasting as a single trajectory refinement task by decoding detections and forecasts in parallel.

3   Unifying Object Detection and Trajectory Forecasting

For a self-driving vehicle (SDV) to plan a safe and explainable maneuver, it must attain a thorough scene understanding, including where other traffic participants are now and where they will be in the future. In other words, a self-driving vehicle must understand object trajectories over time. Because roads are reasonably flat, Bird's-Eye-View (BEV) is the de-facto representation for downstream planning in self-driving [7,18,50,51,62]. Thus, we design DETRA, an end-to-end model that turns raw observations into a set of BEV pose trajectories $\mathbf{P}$ for N objects, F future hypothesis per object, and T time steps. Each pose here is a triplet (x, y, θ) describing the object's centroid and heading (yaw). In addition to the poses, it also predicts the confidence of the object's presence, its dimensions, and the probability of each future hypothesis.

Figure 2 shows an overview of DETRA. It is composed of two parts: extracting BEV scene representations, and refining object trajectories. In Section 3.1, we explain how to extract BEV representations of the scene from LiDAR point clouds and HD maps. While this is not our main contribution, the reader should understand these representations as they are essential to the task. Then, Section 3.2 explains our core contribution: a refinement transformer that refines poses for multiple objects and hypotheses over time via attention. Finally, Section 3.3 describes the learning objective used to train our method end-to-end in a single stage.

3.1   Extracting Scene Representations from LiDAR and HD maps

To understand the environment the SDV is in, we use LiDAR as our sensory input, as it has established itself as the primary sensor in most SDV's platforms [53, 57]. In addition, our model can optionally utilize high-definition (HD) maps as an offline source of information. In contrast to previous end-to-end methods, our approach does not require the fusion of these two inputs at the scene level, which is challenging as they have a very different structure and typically results in information loss (e.g., by creating a raster of the map to fuse with BEV LiDAR representations [4, 6]). Instead, DETRA learns to attend to relevant parts of each modality for each object in its refinement transformer (explained in Section 3.2).

Next, we describe the LiDAR and Map encoders, which extract meaningful features from these two modalities and provide these as tokens with spatial locations associated with the refinement transformer for efficient attention.

LiDAR encoder: To extract rich geometric and motion features from LiDAR, we first voxelize a stack of $H = 5$ past LiDAR sweeps (0.5s) in BEV, where all points in all LiDAR sweeps are expressed in the vehicle coordinate frame at the time the last sweep was acquired [39, 60]. To obtain the voxel features, we use a PointNet [46] for the points residing inside each voxel, as done in VoxelNet [65]. We use a voxel size of 10 cm. Then, a 2D ResNet backbone [24] encodes these voxelized features to generate multi-scale BEV feature maps with resolutions of 0.4, 0.8, 1.6 m. The backbone also leverages dynamic convolution [9], squeeze-excitation [26], and multi-scale deformable attention [68], the details of which are included in the supplementary. Each multi-resolution feature map can then be used as a sequence of LiDAR tokens for cross-attention in our refinement transformer. We note that each of the tokens has an associated centroid corresponding to the position of its BEV pixel, which will be important for efficient attention.

Map encoder: To summarize the heterogeneous information in the HD map into map tokens, we use a LaneGCN [33] with GoRela [13] positional encodings. In summary, we first preprocess HD maps into lane graphs, a representation in which lane centerlines are divided into segments (each 3 m in size), and each segment center constitutes a node in the graph. Each node contains information about the length, heading, and curvature of the segment it represents and the left and right lane boundary distances and types. We connect the nodes with edges according to the topology described in the HD map (e.g., successors, neighbors). Then, a graph convolution network [29] is employed to extract map embeddings for each lane graph node. We treat each lane graph node embedding as a map token and store the corresponding coordinates for later use in efficient cross-attention.

3.2   Detection and Forecasting as Trajectory Refinement

The goal of the refinement transformer is to refine a set of object poses $\mathbf{P} \in \mathbb{R}^{N \times F \times T \times 3}$ with N objects, F future hypothesis per object, and T time steps for each trajectory. Each trajectory consists of a set of poses $\{(x_t, y_t, \theta_t) \mid t \in \{0, \cdots, T-1\}\}$ denoting

the centroid and heading of the object in BEV, where $t = 0$ is the current object pose and $t > 0$ its future poses. In addition, each future hypothesis also includes the associated mode probability. For the first time step $t = 0$, we also estimate the probability that the object exists c (i.e., object confidence) and the length and width in BEV (l, w), which are assumed constant over the future time horizon.

To jointly model the current and future poses of an object, we utilize spatial-temporal object queries $\mathbf{Q}^{(i)} \in \mathbb{R}^{N \times F \times T \times d}$ that contain d-dimensional latent representations (i.e., features) of the objects for the different future hypothesis and time steps. At the beginning of inference $(i = 0)$, these object queries are initialized from learned parameters [36] to get $\mathbf{Q}^{(0)}$. Our refinement transformer refines them iteratively using B blocks $(i \in 1 \ldots B)$, each consisting of several interleaved attention layers. Queries $\mathbf{Q}^{(i)}$ are "spatially anchored" with poses $\mathbf{P}^{(i)}$, which are also refined iteratively during the B transformer blocks. The final object poses that constitute the detections and forecasts are simply those after B transformer blocks, i.e., $\mathbf{P} = \mathbf{P}^{(B)}$.

A refinement transformer block contains A attention layers, each tasked with refining object queries by incorporating information from different inputs and other queries. Each attention layer $\mathcal{A}$ (with index a) updates the object queries as follows:

$$\mathbf{Q}_{\texttt{att}}^{(i,a)} = LN(\mathbf{Q}^{(i,a)} + \mathcal{A}(\mathbf{Q}^{(i,a)}, \cdot)), \tag{1}$$

$$\mathbf{Q}^{(i,a+1)} = LN(\mathbf{Q}_{\texttt{att}}^{(i,a)} + FFN(\mathbf{Q}_{\texttt{att}}^{(i,a)})), \tag{2}$$

where $LN(\cdot)$ denotes layer normalization and $\mathcal{A}(\mathbf{Q}^{(i,a)}, \cdot)$ indicates one of the specialized attentions illustrated in Figure 3, with $\mathbf{Q}^{(i,a)}$ as queries and any other features as keys/values. The updated queries after all the A attention layers in a transformer refinement block are $\mathbf{Q}^{(i+1)} = \mathbf{Q}^{(i,A)}$. From this point onwards, we drop the double superscript (i, a) for brevity and use (i) to refer to the queries at any point during the i-th block.

At the end of each transformer block, a pose update function $\mathcal{U}$ updates the poses given the updated queries $\mathbf{Q}^{(i+1)}$ and previous poses $\mathbf{P}^{(i+1)} = \mathcal{U}(\mathbf{Q}^{(i+1)}, \mathbf{P}^{(i)})$. The updated queries $\mathbf{Q}^{(i+1)}$ and poses $\mathbf{P}^{(i+1)}$ become the input to the next transformer block.

Next, we dive into the details of all the components.

Query initialization: We start off with learnable *mode parameters* $\mathbf{F} \in \mathbb{R}^{1 \times F \times 1 \times d}$ to consider F possible futures [21,54,66], and learnable *time parameters* $\mathbf{T} \in \mathbb{R}^{1 \times 1 \times T \times d}$ to consider multiple time steps. The intuition is that the attention layers may want to perform different reasoning for different modes or time steps. To obtain the initial query volume $\mathbf{Q}^{(0)}$ we sum the mode and time parameter tensors (with broadcasting), and expand the first dimension to the number of objects N, obtaining a tensor of size $N \times F \times T \times d$. Thus, all objects are initialized with the same query features and are only differentiated through their different initial poses $\mathbf{P}^{(0)}$, which will guide the local attention operations and iteratively update each query distinctively.

Pose initialization: Our method relies on an initial estimate of the poses, $\mathbf{P}^{(0)}$. While we can estimate this in many ways, we found it best to initialize from the output of a lightweight detector header that re-uses the LiDAR tokens at a single resolution of
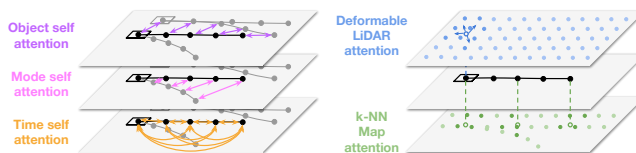
Fig. 3: A diagram of the attention mechanisms used in DETRA. We factorize self-attention in our learnable query volume into object, mode, and time axes. For efficiency and ease of optimization, we perform deformable attention to LiDAR and k-nearest neighbor attention to the map.

$0.4\,\text{m}$. The detector is a very lightweight convolutional, single-shot, anchorless detector that predicts a score heatmap and bounding box parameters per pixel. The per-pixel bounding box proposals go through NMS to yield $N_{\texttt{init}}$ objects. If $N_{\texttt{init}} > N$, we take the top-N candidates by confidence score. This detector's goal is to achieve very high recall, as detections will be refined in the refinement transformer to obtain high precision. These bounding boxes constitute $\mathbf{P}^{(0)}_{t=0}$, and we assume objects are stationary (same pose over time and mode dimensions) to initialize $\mathbf{P}^{(0)}_{t>0}$. This static initialization is not an issue since the pose volume $\mathbf{P}^{(i)}$ is dynamically refined for all time steps over different refinement transformer blocks.

LiDAR cross-attention: Object queries cross-attend to the LiDAR tokens. Since the number of LiDAR tokens N_{lidar} is considerable and fully connected attention brings unaffordable memory consumption at LiDAR feature maps with high resolution, we follow [68] and utilize deformable attention to predict multiple offsets $\mathbf{o} \in \mathbb{R}^{\ell \times 2}$ originating from the object poses $\mathbf{P}^{(i)}$, getting reference points $\mathbf{r}^{(i)} = \mathbf{P}^{(i)} + \mathbf{o}^{(i)}$. Offsets $\mathbf{o}^{(i)}$ are predicted from the object queries $\mathbf{Q}^{(i)}$, and bilinear interpolation is performed around each reference point, thus limiting the attention to 4ℓ LiDAR tokens around the pose of each object token, which yields a query-to-LiDAR attention map of $N \times 4\ell$ where $\ell \ll N_{lidar}$, significantly reducing the computational cost. For efficiency purposes, in our implementation, we only attend around the poses for the first time step $\mathbf{P}^{(i)}_{t=0}$, which is where the most relevant LiDAR evidence for a particular object lays.

Map cross-attention: Object queries cross-attend to neighboring map tokens to exploit the information in the lane graph. Since the map encoder is a graph convolutional network, each lane-graph node embedding (or map token) already encodes information about the neighborhood. Thus, attending to the map tokens far from the object poses is unnecessary, so we limit the cross-attention to the k nearest map tokens from the object pose, reducing the computation from $O(NN_{map})$ to $O(Nk)$ where $k \ll N_{map}$ (around 100 times in practice). We empirically found that limiting the attention to k-nearest map tokens greatly aids learning. Furthermore, since our transformer also has time self-attention, we do not need to attend to the map for all time steps and can update only a specific subset of the time steps in the object queries. In our final model, we attend to map for $t \in \{0, \lfloor (T-1)/2 \rfloor, T-1\}$.

Self-attention: For efficiency purposes, we factorize self-attention into *time self-attention*, *mode self-attention* and *object self-attention*. In *time self-attention*, the queries $\mathbf{Q}^{(i)}$

only attend to other queries from the same object and mode. In other words, information only propagates along the time dimension. More precisely, the sequence dimension is of size T, and $N \times F$ constitutes the batch size, allowing the model to output consistent poses over time. Similarly, in *mode self-attention*, the queries only attend to other queries from the same object and time. The intuition is to enable the model to diversely place the modes to cover possible futures while still being realistic. Finally, in *object self-attention*, each query can only attend to queries from other objects at the same time step and mode, allowing the queries to gather contextual information about neighboring traffic participants, which helps predict consistent behaviors (e.g., avoid collisions between different objects' trajectories). Over different transformer blocks, these operations are repeated, enabling DETRA to learn complex dependencies across the query volume efficiently.

Pose update: At the end of each refinement transformer block, the poses are updated. The object tokens at current time $\mathbf{Q}_{t=0}$ are used to produce object bounding boxes, and object tokens at the future time $\mathbf{Q}_{t>0}$ are used to generate multi-modal future trajectories. The detection bounding box parameters $\mathbf{D} = (x, y, l, w, \theta, c)$ are predicted as $\mathbf{D}^{(i+1)} = \mathbf{D}^{(i)} + \text{MLP}(\frac{1}{N} \sum_{f=1}^{F} \mathbf{Q}_{t=0,f}^{(i)})$, and the current pose $\mathbf{P}_{t=0}^{(i+1)}$ is updated with (x, y, θ) from the updated detections $\mathbf{D}^{(i+1)}$ for all modes. To predict the future trajectories, we use a single-layer bi-directional GRU [11] to process the temporal features for each mode and object in parallel. For each time step, an MLP predicts four outputs: the location (μ_x, μ_y) and the scale (σ_x, σ_y) of an isotropic Laplacian distribution from the GRU hidden state at that time step. The updated future poses $\mathbf{P}_{t>0}^{(i+1)}$ are composed of the Laplacian's (μ_x, μ_y), and the heading θ is computed via temporal finite differences of the waypoints. TransFuser [10] inspired using the GRU, and we find it yields smoother trajectories empirically. Finally, we also predict the score of each future as the logits of a categorical distribution with F classes with another MLP that takes the average of the temporal GRU hidden states for each object and mode as input.

3.3   Learning

All the parameters in DETRA are trained jointly from scratch in an end-to-end fashion. We stop the gradients of the poses $\mathbf{P}^{(i)}$ from flowing in between refinement transformer blocks, which we found to improve learning. We optimize a multi-task objective that is a combination of an initial pose loss $L_{\texttt{init}}$, a detection refinement loss $L_{\texttt{det}}$ and a forecasting refinement loss $L_{\texttt{for}}$, where the detection and forecasting losses are computed at every block: $L = L_{\texttt{init}} + \sum_{i=1}^{B} L_{\texttt{det}}^{(i)} + \alpha L_{\texttt{for}}^{(i)}$.

Initial pose loss: We use a binary focal loss to supervise the score heatmap. The target heatmap is 1 (positive) at pixels with a ground-truth object centroid and 0 (negative) elsewhere. Bounding box parameters are supervised with an IoU loss for the positive pixels in the target heatmap. $L_{\texttt{init}}$ is the sum of these two losses.

Detection refinement loss: For the detection task, we use a combination of a binary focal loss [35] $L_{\texttt{det}}^{\texttt{cls}}$ for classification, with an L1 loss $L_{\texttt{det}}^{\texttt{L1}}$ and a generalized intersection-over-union (gIoU) loss $L_{\texttt{det}}^{\texttt{giou}}$ to regress the parameters of bounding boxes:

Table 1: Comparing DETRA to state-of-the-art detection and forecasting models on AV2.

Argoverse2	Joint		Detection			Forecasting						
	AP ↑		AP @ IoU ↑			MR ↓		ADE ↓		FDE ↓		bFDE ↓
	Occ	Traj	@0.3	@0.5	@0.7	$K=1$	$K=6$	$K=1$	$K=6$	$K=1$	$K=6$	$K=6$
MultiPath [8]	62.7	43.7	95.5	91.7	76.3	43.3	32.9	2.12	1.03	4.81	2.26	2.87
LaneGCN [33]	64.4	41.9	95.5	91.7	76.3	41.2	27.3	2.08	0.86	4.70	1.85	2.30
SceneTF [44]	63.9	42.9	95.5	91.7	76.3	40.9	24.5	1.98	0.86	4.49	1.85	2.32
GoRela [13]	57.8	42.9	95.5	91.7	76.3	39.8	21.3	1.84	1.00	4.05	2.10	2.46
FaF [39]	64.4	39.5	94.7	90.8	74.8	40.1	22.1	1.82	0.84	4.21	1.79	2.27
IntentNet [6]	66.6	40.0	95.4	91.3	75.3	40.6	24.5	1.88	0.80	4.36	1.67	2.11
MultiPathE2E [8]	66.1	44.1	95.6	91.7	76.3	41.6	31.5	1.84	0.93	4.25	2.07	2.65
LaneGCNE2E [33]	65.9	39.2	94.8	91.0	75.0	40.7	24.5	1.93	0.82	4.51	1.73	2.17
GoRelaE2E [13]	58.4	38.3	94.7	91.1	75.5	41.3	23.2	2.36	1.35	5.06	2.74	3.12
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**

$L_{\mathtt{det}} = L_{\mathtt{det}}^{\mathtt{cls}} + \beta L_{\mathtt{det}}^{\mathtt{L1}} + \gamma L_{\mathtt{det}}^{\mathtt{giou}}$. 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].

Forecasting refinement loss: We use a simple winner-takes-all (WTA) loss on the mixture of trajectories, where an isotropic Laplacian parameterizes each waypoint and the mode probabilities are the weights of the mixture [66, 67]. We supervised the mode probabilities with a cross-entropy loss $L_{\mathtt{for}}^{\mathtt{cls}}$, where the *winner mode* is the one in which centroids are the closest to the GT. We supervise the Laplacian waypoints (location and scale) predicted for the winner mode with negative log-likelihood $L_{\mathtt{for}}^{\mathtt{nll}}$, and the ones for the other modes are left unsupervised. In sum, the forecasting loss is $L_{\mathtt{for}} = L_{\mathtt{for}}^{\mathtt{nll}} + L_{\mathtt{for}}^{\mathtt{cls}}$. Note that we only supervise the forecasted trajectories from true positive detections during training (defined as those having over 0.5 IoU with the ground truth).

4 Experiments

This section provides a comprehensive analysis of DETRA from two perspectives.

- Comparing our method to a relevant set of state-of-the-art baselines, including modular detection-tracking-forecasting methods and end-to-end detection-forecasting methods. Our results show clear improvements across all detection, forecasting, and joint metrics in two datasets: Argoverse 2 Sensor [57] and Waymo Open [53].
- Understanding the aspects that make our method attain this higher level of performance with thorough ablation studies. Our results show that (i) all presented components are necessary, (ii) refining the poses over multiple transformer blocks is very effective and benefits both detection and forecasting and (iii) we made key design choices in pose initialization as well as LiDAR and map cross attention.

We note that the novelty of DETRA is not in its particular components, but in unifying detection and forecasting as a single trajectory refinement task. Nevertheless, we think that ablating the various design choices in enlightening and important for future work. Refer to our supplementary for more implementation details, qualitative results, and ablations (number of blocks, attention ordering, dropping time/mode parameters).

Table 2: Comparing our proposed DETRA to the state-of-the-art detection and forecasting models on the Waymo Open Dataset dataset.

Waymo Open	Joint		Detection			Prediction						
	AP ↑		AP @ IoU ↑			MR ↓		ADE ↓		FDE ↓		bFDE ↓
	Occ	Traj	@0.3	@0.5	@0.7	$K=1$	$K=6$	$K=1$	$K=6$	$K=1$	$K=6$	$K=6$
MultiPath [8]	60.2	31.9	88.6	85.1	72.9	45.4	37.2	2.54	1.49	5.63	3.31	3.93
LaneGCN [33]	63.9	30.4	88.6	85.1	72.9	44.5	31.4	2.53	1.03	5.63	2.19	2.63
SceneTF [44]	64.9	31.4	88.6	85.1	72.9	44.4	27.4	2.44	1.00	5.39	2.16	2.66
GoRela [13]	61.3	29.6	88.6	85.1	72.9	41.1	19.6	2.02	1.07	4.21	2.07	2.41
FaF [39]	62.3	23.9	87.3	82.3	67.7	44.0	27.7	2.41	1.03	5.40	2.22	2.58
IntentNet [6]	62.9	24.9	87.8	83.2	68.8	43.7	27.7	2.48	1.03	5.53	2.21	2.56
MultiPathE2E [8]	57.6	32.7	88.6	84.7	71.1	44.2	34.9	2.13	1.25	4.87	2.75	3.39
LaneGCNE2E [33]	63.7	25.1	88.1	83.6	70.0	43.5	26.4	2.37	0.96	5.33	2.03	2.39
GoRelaE2E [13]	61.5	26.4	88.1	84.0	71.1	42.7	21.9	2.37	1.35	4.78	2.50	2.86
DETRA	**70.4**	**37.1**	**89.8**	**86.5**	**74.9**	**40.3**	**18.0**	**1.87**	**0.59**	**4.29**	**1.25**	**1.93**

Datasets: To show the generality of our approach across different LiDAR sensors and maps, we conduct experiments in Argoverse 2 Sensor [57] Waymo Open Dataset [53]. Argoverse 2 Sensor (AV2) contains 850 sequences collected in Austin, Detroit, Miami, Palo Alto, Pittsburgh, and Washington D.C. Each sequence lasts 15 seconds and consists of 150 frames of point clouds from two 32-beam LiDARs that provide an average of 107K points per frame. Argoverse 2 includes HD maps. We use 700 sequences for training and 150 sequences for validation. Waymo Open Dataset (WOD) [53] is made of 1150 sequences collected from 6 different cities in the U.S. Each sequence lasts 20 seconds and contains 200 frames of point clouds from five LiDAR sensors with an average of 177K points per frame.

Task settings: Provided with 0.5 seconds of LiDAR history as well as the HD map, models are tasked to detect vehicles inside the region of interest (RoI) with at least one lidar point, and forecast six possible future trajectories at 2Hz, each 5 seconds long. The RoI is a square centered around the ego vehicle of size 80×80 meters in AV2 and 150×150 meters in WOD. The rationale for having different RoIs is to keep the task complexity similar, as WOD features a denser and longer-range LiDAR than AV2.

Metrics: We consider a comprehensive set of metrics including those widely used in object detection and trajectory forecasting, and new metrics to evaluate both tasks jointly. Next, we explain them at a high level (more details in the supplementary).

- Detection: average precision at multiple IoUs (AP@IoU) measures the ability to estimate object existence (low IoUs) and localization accuracy (high IoUs).
- Forecasting: we follow the metrics proposed by Argoverse 2 Motion [57]: miss rate (MR), average displacement error (ADE), final displacement error (FDE), and its Brier variant (bFDE). Errors are reported for the most likely predicted trajectory ($K = 1$) and the best of 6 trajectories ($K = 6$). These metrics are evaluated at a common detection recall point for all models of 80% at an IoU threshold of 0.5, as proposed in previous works [4, 5, 12]. To address the fact that most objects in AV2 and WOD datasets are stationary, we bucket the objects in stationary and dynamic, calculate metrics for each bucket, and report the macro-average.
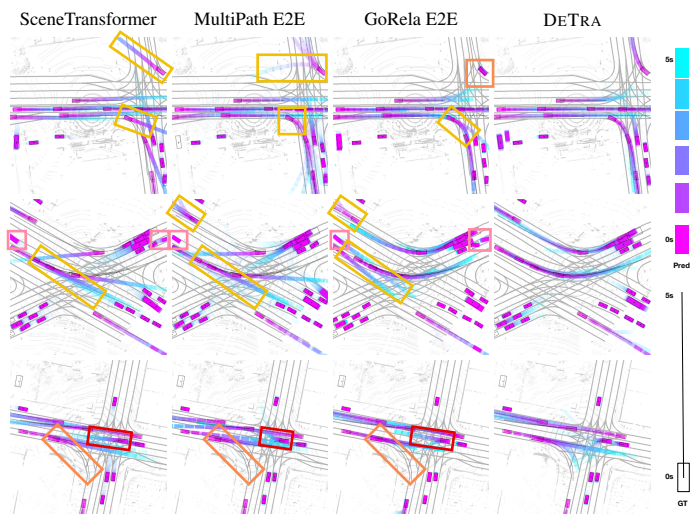
Fig. 4: Qualitative results on AV2. We highlight failure modes in the baselines that DETRA improves: inaccurate detections, off-map predictions, FN modes and FP modes.

- Joint: these metrics capture the end-to-end performance in a single metric. Drawing inspiration from occupancy forecasting works [1, 40], we compute Occupancy Average Precision (OccAP) by rendering the object detections and forecasted trajectories considering the confidence about the object's existence and the mode probabilities. In addition, we extend the Trajectory Average Precision (TrajAP) proposed in WOD Motion [17] to also penalize false positive and false negative detections.

Baselines: We benchmark against the following methods.

- Modular detection-tracking-forecasting baselines. We choose [60] as the detector architecture and modify it to predict the position at the previous time step. Following [47, 58, 59], we utilize a simple online tracker to associate detections over time leveraging the predicted previous position, and refine the tracked trajectory. For trajectory forecasting, we use MultiPath [8], LaneGCN [33], Scene Transformer [44], and GoRela [13]. The forecasting models ingest the (noisy) object tracks from the tracker and the map, and output the future trajectories.
- End-to-end (E2E) detection and forecasting. We implement FaF [39] and IntentNet [6], in which object bounding boxes and future trajectories are generated in one shot from a single convolutional network [60]. We also extend state-of-the-art forecasting approaches MultiPath [8], LaneGCN [33], and GoRela [13] by predicting object bounding boxes from the LiDAR backbone features and then cropping the feature maps with the bounding boxes to generate high-dimensional object embeddings, which inputs to the forecasting part of the end-to-end architecture.

Table 3: DeTra's self-improvement over refinement blocks on AV2. Each row evaluates the intermediate predictions at the end of the i-th block ($i = 0$ being pose initialization).

	Joint		Detection			Forecasting						
	AP ↑		AP @ IoU ↑			MR ↓		ADE ↓		FDE ↓		bFDE ↓
	Occ	Traj	@0.3	@0.5	@0.7	$K=1$	$K=6$	$K=1$	$K=6$	$K=1$	$K=6$	$K=6$
DeTra $i=0$	42.7	38.7	95.3	91.8	76.7	44.7	44.7	7.39	7.39	14.8	14.8	15.5
DeTra $i=1$	71.1	48.0	95.7	92.5	79.2	38.2	17.5	1.67	0.61	3.94	1.34	2.01
DeTra $i=2$	72.8	**50.2**	95.8	92.8	80.1	37.4	15.7	1.54	0.57	**3.58**	1.21	1.88
DeTra $i=3$	**73.0**	50.0	**95.8**	**92.9**	**80.2**	**37.0**	**15.4**	**1.54**	**0.55**	3.59	**1.19**	**1.86**

Since the LiDAR encoder is not the focus of our work, yet it influences the detection and forecasting quality, we use the same LiDAR encoder architecture in all models (including ours) to ensure a fair comparison.

Implementation Details: We utilize $N = 400$ and $N = 600$ object queries in AV2 and WOD, respectively. We use more queries in WOD to account for the larger RoI. DeTra uses $B = 3$ refinement transformer blocks. The dimensions of all embeddings are 128 ($d = 128$). We use 4-nearest map tokens ($k = 4$) for map attention. For the weights in our multi-task objective, we use $\alpha = 0.1$, $\beta = 0.01$, $\gamma = 0.1$. We train with a batch size of 16 scenes. We use AdamW [38] optimizer with a learning rate cosine decay starting at 8×10^{-4} and ending at 0, and weight decay of 10^{-4}. We train for 240K iterations on both datasets. See the supplementary for more implementation details.

Comparison against state-of-the-art: Tables 1 and 2 present detection and forecasting results on AV2 and WOD. Our model outperforms all baselines in all metrics on both datasets. On the proposed joint detection and forecasting metrics OccAP and TrajAP, DeTra outperforms the runner-up method consistently on both datasets, with a relative improvement of 9.6% and 13.4% in AV2, and 8.5% and 13.5% in WOD. Note that the runner-up method varies across datasets and metrics, but the gains over the best baseline on any metric are notable. On detection metrics, we see substantial improvements (3.9 A.P. points in AV2 and 2.0 A.P. points in WOD) at high IoU (0.7), without sacrificing any performance at low IoU (0.3), showing the power of our refinement transformer for detection. We recall that all baselines use the same LiDAR backbone architecture as DeTra, which isolates these gains to our proposed refinement transformer. On forecasting, our model achieves a relative improvement in MissRate with $K = 6$ over the best-performing baseline of 27% and 8.2% in AV2 and WOD, respectively. DeTra also achieves the best metrics when considering only the most likely future, $K = 1$. Figure 4 highlights qualitative improvements over the strongest baselines.

Self-improvement across refinement blocks: Table 3 and Figure 5 shows how DeTra's predictions —both detections and forecasts— improve over multiple rounds of our refinement transformer. Pose initialization ($i = 0$) has a reasonably high detection performance at low IoUs (AP@IoU0.3) but lacks precise localization of the bounding boxes, as shown by the lower AP@IoU0.7. Forecasting results are weak, as expected since we assume objects to be stationary at initialization. After the first refinement block ($i = 1$), detection AP@IoU0.7 is notably boosted (+2.5 points), and forecasting starts
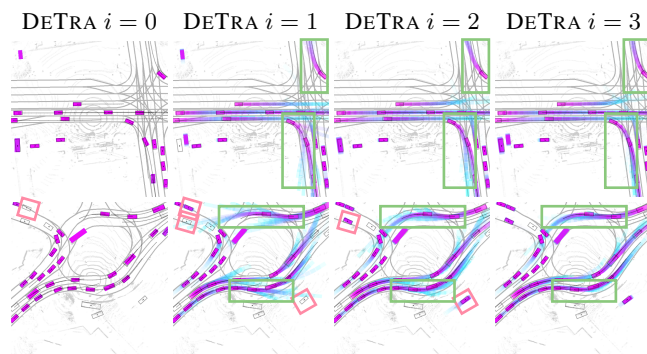
Fig. 5: Visualizing the DETRA's self improvement over refinement blocks on AV2. We highlight improved forecasting accuracy and improved detection accuracy cases.

predicting the correct future motion (much lower ADEs and FDEs). Yet, it still lacks map understanding as shown by the qualitative results and Miss Rate. The second refinement block brings further detection gains, and the forecasts better follow the lanes. Finally, while the third refinement block still provides some improvements, the returns are diminished. An attractive by-product of framing detection and forecasting as a unified trajectory refinement model is the ability to use it as an anytime predictor [23, 25, 30], i.e., a network with multiple exits where early outputs can be used if there is a need to reduce the latency/reaction time in specific situations.

Effect of pose initialization: We compare our proposed pose initialization using a lightweight detector head to a fixed grid (G) as well as grid + learnable offsets (L) in Table 4 (V_1 and V_2 respectively). We use a 20×20 grid to have $N = 400$ queries, the same number we consider in our approach. For the learned offset experiment, we allow the gradients from the i-th block to flow back to $\mathbf{P}^{(i-1)}$. We see that (i) detection-based initialization is best by a fair margin and (ii) that learned offsets from a grid are not helpful (we hypothesize this is because there is not a clear responsibility split between these learned offsets and the ones predicted in the pose update).

Effect of pose refinement: We consider a version of the model where all the refinement transformer blocks receive the initial poses $\mathbf{P}^{(0)}$ as input (V_3 in Table 4). Since DETRA and V_3 have the same number of parameters and supervision (intermediate outputs at $i < B$ are still supervised), this shows that refinement is a critical piece.

Effect of LiDAR attention: In Table 4, we consider two alternatives to the deformable LiDAR attention proposed in DETRA: no attention to LiDAR (V_4) and global attention to all the tokens in the LiDAR feature map (V_5). Both alternatives perform poorly, with detection metrics well below those of the initial detection proposals (see DETRA $i = 0$ in Table 3). With no attention (V_4), the model seems to struggle preserving the original proposals through multiple transformer blocks. Global attention (V_5) not only underperforms, but it is also computationally expensive. We hypothesize it performs poorly because there are too many LiDAR tokens in the multi-resolution feature maps,

Table 4: DETRA components ablation on AV2. ✓ means we use what is described in Section 3. × means we drop this component. For pose initialization (Init) we also consider initial poses in a grid (G), and learned offsets on top (L). The acronym Rfn stands for refinement. For cross-attention, we consider global attention to all LiDAR/Map tokens (Glb) as an alternative.

	Components							Joint		Detection			Forecasting						
	Pose		Cross-attn		Self-attn			AP ↑		AP @ IoU ↑			MR ↓		ADE ↓		FDE ↓		bFDE ↓
	Init	Rfn	Ldr	Map	Time	Mode	Obj	Occ	Traj	@0.3	@0.5	@0.7	$K=1$	$K=6$	$K=1$	$K=6$	$K=1$	$K=6$	$K=6$
V_1	G	✓	✓	✓	✓	✓	✓	70.8	46.8	94.6	91.4	78.2	37.7	16.1	1.61	0.56	3.79	1.20	1.87
V_2	L	✓	✓	✓	✓	✓	✓	68.5	46.1	92.6	89.3	76.0	37.8	16.5	1.66	0.57	3.88	1.21	1.88
V_3	✓	×	✓	✓	✓	✓	✓	70.0	44.6	95.4	91.5	75.9	39.8	21.6	1.87	0.69	4.31	1.52	2.18
V_4	✓	✓	×	✓	✓	✓	✓	42.5	14.6	70.0	52.2	35.6	-	-	-	-	-	-	-
V_5	✓	✓	Glb	✓	✓	✓	✓	48.1	16.6	77.5	52.9	31.9	-	-	-	-	-	-	-
V_6	✓	✓	✓	×	✓	✓	✓	71.2	47.9	95.4	92.5	79.7	38.7	20.2	1.71	0.66	4.01	1.47	2.14
V_7	✓	✓	✓	Glb	✓	✓	✓	71.3	46.9	95.4	92.6	79.6	39.5	20.3	1.80	0.65	4.25	1.46	2.13
V_8	✓	✓	✓	✓	×	✓	✓	62.4	40.2	95.4	92.4	79.5	43.7	31.3	4.59	1.01	9.23	2.06	2.69
V_9	✓	✓	✓	✓	✓	×	✓	**73.1**	49.7	95.7	92.9	**80.5**	37.5	16.0	1.57	0.55	3.65	1.18	1.85
V_{10}	✓	✓	✓	✓	✓	✓	×	73.0	50.0	95.7	92.7	80.1	37.6	15.8	1.61	0.55	3.77	**1.17**	**1.84**
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

which makes it very hard to find signal among the noise. Forecasting metrics are not reported for these two variants as the detector does not reach the required 80% target recall. In summary, including the inductive bias that the important LiDAR features are around the object poses $\mathbf{P}^{(i)}$ via deformable attention is very helpful.

Effect of Map attention: In Table 4, we consider no map attention (V_6) and global map attention (V_7) instead of DETRA's proposed k-nearest neighbor attention. The results are similar to those observed in LiDAR: limiting the attention to a neighborhood seems critical. Without limiting attention to map tokens in a neighborhood, this component is not effective, as shown by the very similar results of V_6 and V_7.

Effect of self-attention: V_8, V_9, and V_{10} in Table 4 drop time, mode, and object self-attention, respectively. The results show that time attention has a large impact, which makes sense since DETRA only attends to LiDAR at $t = 0$ to limit the computation, and the geometric and motion features extracted from it can only reach other time steps via time self-attention. While detection and joint metrics do not show gains for mode and object attention, most forecasting metrics show that it is beneficial to include these.

5 Conclusion

In this paper, we introduced DETRA, a model to tackle detection and forecasting as a unified trajectory refinement task. We design a refinement transformer architecture to enable self-improvement via cross attention to heterogeneous inputs as well as factorized self attention across time, mode and object dimension. Our experiments on two large-scale self-driving datasets show that our model outperforms state-of-the-art modular and E2E approaches. Importantly, extensive ablation studies back our design, showing that all the components contribute positively, our proposed pose refinement is critical, and key choices were made to aid learning with geometric priors.

References

1. Agro, B., Sykora, Q., Casas, S., Urtasun, R.: Implicit occupancy flow fields for perception and prediction in self-driving. In: CVPR (2023)
2. Cai, Z., Vasconcelos, N.: Cascade r-cnn: High quality object detection and instance segmentation. PAMI (2019)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (2020)
4. Casas, S., Gulino, C., Liao, R., Urtasun, R.: Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In: ICRA (2020)
5. Casas, S., Gulino, C., Suo, S., Luo, K., Liao, R., Urtasun, R.: Implicit latent variable model for scene-consistent motion forecasting. In: ECCV (2020)
6. Casas, S., Luo, W., Urtasun, R.: Intentnet: Learning to predict intention from raw sensor data. In: CoRL (2018)
7. Casas, S., Sadat, A., Urtasun, R.: Mp3: A unified model to map, perceive, predict and plan. In: CVPR (2021)
8. Chai, Y., Sapp, B., Bansal, M., Anguelov, D.: Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. arXiv preprint arXiv:1910.05449 (2019)
9. 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)
10. Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., Geiger, A.: Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. PAMI (2022)
11. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
12. Cui, A., Casas, S., Sadat, A., Liao, R., Urtasun, R.: Lookout: Diverse multi-future prediction and planning for self-driving. In: ICCV (2021)
13. Cui, A., Casas, S., Wong, K., Suo, S., Urtasun, R.: Gorela: Go relative for viewpoint-invariant motion forecasting. arXiv preprint arXiv:2211.02545 (2022)
14. Cui, H., Radosavljevic, V., Chou, F.C., Lin, T.H., Nguyen, T., Huang, T.K., Schneider, J.G., Djuric, N.: Multimodal trajectory predictions for autonomous driving using deep convolutional networks. 2019 ICRA pp. 2090–2096 (2018), https://api.semanticscholar.org/CorpusID:52891221
15. Deo, N., Wolff, E.M., Beijbom, O.: Multimodal trajectory prediction conditioned on lanegraph traversals. In: CoRL (2021)
16. Djuric, N., Cui, H., Su, Z., Wu, S., Wang, H., Chou, F.C., Martin, L.S., Feng, S., Hu, R., Xu, Y., Dayan, A., Zhang, S., Becker, B.C., Meyer, G.P., Vallespi-Gonzalez, C., Wellington, C.K.: Multixnet: Multiclass multistage multimodal motion prediction. IV (2021)
17. Ettinger, S., Cheng, S., Caine, B., Liu, C., Zhao, H., Pradhan, S., Chai, Y., Sapp, B., Qi, C.R., Zhou, Y., et al.: Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In: ICCV (2021)
18. Fan, H., Zhu, F., Liu, C., Zhang, L., Zhuang, L., Li, D., Zhu, W., Hu, J., Li, H., Kong, Q.: Baidu apollo em motion planner. arXiv preprint arXiv:1807.08048 (2018)
19. Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., Schmid, C.: Vectornet: Encoding hd maps and agent dynamics from vectorized representation. CVPR (2020)
20. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: Thomas: Trajectory heatmap output with learned multi-agent sampling. arXiv preprint arXiv:2110.06607 (2021)
21. Girgis, R., Golemo, F., Codevilla, F., Weiss, M., D'Souza, J.A., Kahou, S.E., Heide, F., Pal, C.: Latent variable sequential set transformers for joint multi-agent motion prediction. arXiv preprint arXiv:2104.00563 (2021)

22. Girgis, R., Golemo, F., Codevilla, F., Weiss, M., D'Souza, J.A., Kahou, S.E., Heide, F., Pal, C.: Latent variable sequential set transformers for joint multi-agent motion prediction. arXiv preprint arXiv:2104.00563 (2021)
23. Grubb, A., Bagnell, D.: Speedboost: Anytime prediction with uniform near-optimality. In: Artificial Intelligence and Statistics. pp. 458–466. PMLR (2012)
24. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
25. Hu, H., Dey, D., Hebert, M., Bagnell, J.A.: Learning anytime predictions in neural networks via adaptive loss balancing. In: Proceedings of the AAAI Conference on Artificial Intelligence (2019)
26. 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)
27. Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., Lu, L., Jia, X., Liu, Q., Dai, J., Qiao, Y., Li, H.: Planning-oriented autonomous driving. In: CVPR (2023)
28. Ivanovic, B., Lin, Y., Shrivastava, S., Chakravarty, P., Pavone, M.: Propagating state uncertainty through trajectory forecasting. In: ICRA (2022)
29. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
30. Kumbhar, O., Sizikova, E., Majaj, N., Pelli, D.G.: Anytime prediction as a model of human reaction time. arXiv preprint arXiv:2011.12859 (2020)
31. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. CVPR (2018)
32. Li, L.L., Yang, B., Liang, M., Zeng, W., Ren, M., Segal, S., Urtasun, R.: End-to-end contextual perception and prediction with interaction transformer. IROS (2020)
33. 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)
34. 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)
35. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017)
36. Liu, S., Li, F., Zhang, H., Yang, X., Qi, X., Su, H., Zhu, J., Zhang, L.: DAB-DETR: Dynamic anchor boxes are better queries for DETR. In: ICLR (2022)
37. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV (2015)
38. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
39. Luo, W., Yang, B., Urtasun, R.: Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In: CVPR (2018)
40. Mahjourian, R., Kim, J., Chai, Y., Tan, M., Sapp, B., Anguelov, D.: Occupancy flow fields for motion forecasting in autonomous driving. IEEE Robotics and Automation Letters **7**(2), 5639–5646 (2022)
41. Meyer, G.P., Charland, J., Pandey, S., Laddha, A.G., Gautam, S., Vallespi-Gonzalez, C., Wellington, C.K.: Laserflow: Efficient and probabilistic object detection and motion forecasting. IEEE Robotics and Automation Letters **6**, 526–533 (2021)
42. Mo, X., Huang, Z., Xing, Y., Lv, C.: Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network. IEEE Transactions on Intelligent Transportation Systems (2022)
43. Nayakanti, N., Al-Rfou, R., Zhou, A., Goel, K., Refaat, K.S., Sapp, B.: Wayformer: Motion forecasting via simple & efficient attention networks. arXiv preprint arXiv:2207.05844 (2022)

44. 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)
45. Phan-Minh, T., Grigore, E.C., Boulton, F.A., Beijbom, O., Wolff, E.M.: Covernet: Multi-modal behavior prediction using trajectory sets. CVPR (2019)
46. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR (2017)
47. Qi, C.R., Zhou, Y., Najibi, M., Sun, P., Vo, K., Deng, B., Anguelov, D.: Offboard 3d object detection from point cloud sequences. In: CVPR (2021)
48. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
49. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. PAMI (2015)
50. Rhinehart, N., McAllister, R., Levine, S.: Deep imitative models for flexible inference, planning, and control. arXiv preprint arXiv:1810.06544 (2018)
51. Sadat, A., Ren, M., Pokrovsky, A., Lin, Y.C., Yumer, E., Urtasun, R.: Jointly learnable behavior and trajectory planning for self-driving vehicles. In: 2019 IROS. pp. 3949–3956. IEEE (2019)
52. Shah, M., ling Huang, Z., Laddha, A.G., Langford, M., Barber, B., Zhang, S., Vallespi-Gonzalez, C., Urtasun, R.: Liranet: End-to-end trajectory prediction using spatio-temporal radar fusion. In: CoRL (2020)
53. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: CVPR (2020)
54. Varadarajan, B., Hefny, A., Srivastava, A., Refaat, K.S., Nayakanti, N., Cornman, A., Chen, K., Douillard, B., Lam, C.P., Anguelov, D., et al.: Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In: 2022 ICRA (2022)
55. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NeurIPS (2017)
56. Weng, X., Ivanovic, B., Pavone, M.: Mtp: Multi-hypothesis tracking and prediction for reduced error propagation. In: IV (2022)
57. Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J.K., et al.: Argoverse 2: Next generation datasets for self-driving perception and forecasting. arXiv preprint arXiv:2301.00493 (2023)
58. Yang, A.J., Casas, S., Dvornik, N., Segal, S., Xiong, Y., Hu, J.S.K., Fang, C., Urtasun, R.: Labelformer: Object trajectory refinement for offboard perception from lidar point clouds. In: CoRL. PMLR (2023)
59. Yang, B., Bai, M., Liang, M., Zeng, W., Urtasun, R.: Auto4d: Learning to label 4d objects from sequential point clouds. arXiv preprint arXiv:2101.06586 (2021)
60. Yang, B., Luo, W., Urtasun, R.: Pixor: Real-time 3d object detection from point clouds. In: CVPR (2018)
61. Yuan, Y., Weng, X., Ou, Y., Kitani, K.M.: Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In: ICCV (2021)
62. Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., Urtasun, R.: End-to-end interpretable neural motion planner. In: CVPR (2019)
63. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. ArXiv (2019)
64. Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T.Y., Guo, J., Ngiam, J., Vasudevan, V.: End-to-end multi-view fusion for 3d object detection in lidar point clouds. ArXiv (2019)

65. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2017)
66. Zhou, Z., Wang, J., Li, Y.H., Huang, Y.K.: Query-centric trajectory prediction. In: CVPR (2023)
67. Zhou, Z., Ye, L., Wang, J., Wu, K., Lu, K.: Hivt: Hierarchical vector transformer for multi-agent motion prediction. CVPR (2022)
68. 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)