Towards Metric-Agnostic Trajectory Forecasting

Markus Knoche[1], Daan de Geus[2], and Bastian Leibe[1]

[1] RWTH Aachen University, Germany
[2] Eindhoven University of Technology, Netherlands
vision.rwth-aachen.de/TraDiE-policies

Abstract. Accurate trajectory forecasting of surrounding traffic participants is a core capability for autonomous driving, enabling vehicles to anticipate behavior and plan safe maneuvers. We observe that current state-of-the-art forecasting models on Argoverse 2 and the Waymo Open Motion Dataset tailor their training objectives to the different benchmark metrics. Because these metrics encourage conflicting behavior, we propose a paradigm change for trajectory forecasting: *training models with metric-agnostic probabilistic objectives and treating metric optimization as a downstream task applied to the predictive distribution.* Concretely, we introduce Trajectory Distribution Evaluation (TraDiE) policies, metric-specific policies that map a predictive distribution to the set of K trajectories and confidences required by trajectory forecasting metrics. We evaluate this framework by introducing DONUT-NLL, which adapts the training objective of the state-of-the-art trajectory forecasting model DONUT to directly optimize the predictive distribution. Using our policies, DONUT-NLL achieves state-of-the-art results on all metrics of the Waymo motion prediction benchmark.

1 Introduction

Trajectory forecasting is a core component of autonomous driving systems, as it enables vehicles to anticipate the behavior of other traffic participants. Because human driving is inherently non-deterministic and multimodal, it is not sufficient to predict only a single future trajectory per agent. Instead, trajectory forecasting models must capture multiple plausible futures and quantify uncertainty such that downstream applications can account for possible behaviors.

In practice, the trajectory forecasting task is defined as follows: given a road graph describing the static scene context (*e.g.*, lanes, crosswalks) and the recent state histories of relevant agents (*e.g.*, cars, pedestrians), the model must predict a set of K future trajectories over a fixed temporal horizon for each agent. Typically, $K = 6$. In addition, each trajectory gets assigned a confidence score. Together, these predictions should represent a multimodal distribution over plausible futures corresponding to the true uncertainty of the world.

In recent years, a diverse set of architectures and training objectives has been proposed for this task [11, 13, 28, 31, 38, 40]. Interestingly, we observe that these models are strongly shaped by the particular metrics that are used in the most popular benchmarks. Models evaluating on Argoverse 2 [33], whose main metric

Brier-minFDE depends on endpoint distance, often employ training objectives that closely mirror these distance-based metrics [14, 39]. Conversely, methods evaluated on the Waymo benchmark [3], which uses *soft mAP*, are optimized for broader coverage of the space of possible future endpoints, *e.g.*, by using non-maximum suppression to avoid clusters of endpoints [17, 26, 34]. Some recent papers even train multiple models to optimize different metrics [26, 30, 31].

While it is not inherently problematic to optimize for a benchmark, tightly coupling trajectory forecasting models to evaluation metrics can lead to several issues. Different benchmark metrics encode different, sometimes conflicting, goals: distance-based metrics reward precise final positions, whereas soft mAP expects broad coverage of the future space. As a result, models tuned to one metric tend to perform worse on others, and it becomes hard to assess whether improvements stem from genuinely better forecasting or from better metric-specific tuning. Moreover, benchmark metrics do not directly evaluate the model's predictive distribution and thus only provide a partial view of its quality.

In this work, we argue for a paradigm change in trajectory forecasting: *the primary training objective should be to learn a good predictive distribution, independent of the metrics used to evaluate the forecast.* Such a distribution should provide a faithful representation of uncertainty. If it is well-calibrated, it can serve as a general, metric-agnostic interface that can then be exploited by various downstream tasks, including optimizing for evaluation metrics.

We suggest that optimizing for different metrics should become part of the evaluation protocol as a post-processing step applied to the generated predictive distribution. To this end, we build upon and generalize the per-metric optimization proposed by HOME [6] by defining metric-specific policies that map these distributions to the finite set of K trajectories and confidences required by a particular metric. Our **Tra**jectory-**Di**stribution **E**valuation policies (TraDiE policies) are derived from Monte Carlo samples of the predictive distribution for common trajectory prediction metrics. Crucially, these policies are applied to the outputs of a single underlying model, without retraining. This evaluation protocol rewards the model for a good predictive distribution: if it is accurate, policies optimizing for the metric under the distribution will yield good performance.

We argue that adopting this decoupled view will allow the trajectory forecasting community to again focus on the core of the problem, namely how to represent the predictive distribution and how to improve its accuracy. This turns the choice of predictive distribution into a general design decision. It allows systematic evaluation of how future agent trajectories and their uncertainty should be represented and how trajectory forecasting models should be trained, independent of specific benchmark metrics.

As a concrete instantiation, we apply this idea to DONUT [13], a recent trajectory forecasting model that achieves state-of-the-art results on Argoverse 2. Naively applied to Waymo with its original, minFDE-related loss, DONUT's soft mAP performance is extremely poor, making it an ideal testbed to study the impact of metric-agnostic training. We replace the original loss with negative log-likelihood objectives, which directly optimize the predictive distribution, and

find that these models improve both minFDE and (soft) mAP significantly when combined with our policies. This shows that optimizing the distribution directly is indeed better than training for surrogate submetrics. We further evaluate different distribution families for representing agents' positions and show that mixtures of generalized Gaussian distributions are better suited than mixtures of Laplacian distributions or Gaussian scale mixtures. Beyond DONUT, we apply our metric-specific policies to two strong open-source baselines (MTR [26] and QCNet [38]) and show that they (i) consistently improve MTR across metrics and (ii) expose a distribution mismatch for QCNet under distance-based evaluation.

In summary, our contributions are:

1. We advocate for a paradigm shift in which trajectory forecasting models are trained with metric-agnostic probabilistic objectives that learn well-calibrated predictive distributions, and in which metric optimization is treated as a downstream step via metric-specific policies.
2. We propose sampling-based metric-specific policies for mapping predictive distributions to K trajectories and confidences, with concrete implementations for minFDE, (soft) mAP and miss rate, enabling evaluation of multiple metrics from a single predictive model without metric-specific retraining.
3. We introduce DONUT-NLL, a negative log-likelihood training scheme for DONUT, which achieves state-of-the-art results on the Waymo benchmark, demonstrating the effectiveness of optimizing the distribution directly.
4. We conduct a systematic comparison of position distributions and empirically find that generalized Gaussian distributions outperform alternatives.
5. We validate that the proposed policies are model-agnostic by applying them to MTR [26] and QCNet [38], highlighting both gains and failure modes when predictive distributions are misspecified.

2 Related Work

Early work on trajectory forecasting encodes the environment in bird's-eye-view rasterizations and applies convolutional neural networks to capture local context [1, 2, 6, 10, 15, 21, 23]. More modern approaches use sparse input representations, where agents' histories and map elements are modeled as polylines [5, 16]. On top of such structures inputs, graph neural networks have been used to capture relations between agents and the road layout [19, 35]. Current approaches commonly model agents as queries and use attention mechanisms for interaction between agents and map elements [13, 17, 18, 20, 22, 26, 28–30, 32, 38, 40].

To model uncertainty over future trajectories, earlier works [10, 15] have employed VAEs [12], which allow sampling of an arbitrary number of trajectories. Some approaches [6–8] represent the uncertainty as dense, rasterized heatmaps. Most current models predict a fixed number of possible futures, often represented as the modes of Gaussian mixtures [17, 20, 25–28, 30, 31, 34] or Laplacian mixtures [13, 32, 38–40]. For training, they commonly employ a winner-takes-all loss, in which the trajectory closest to the ground truth gets optimized with negative log-likelihood. Other approaches [14, 36, 37] directly regress the best trajectory

with an L_1 or smooth L_1 loss without defining a continuous mixture. A few works [11, 24] employ GPT-like models, which discretize sub-trajectories into a fixed number of classes and learn a categorical distribution per timestep. Such models allow sampling arbitrarily many trajectories at inference time.

As we discuss in more detail in Sec. 3.3, many of these models optimize for the main metrics of either Argoverse 2 or Waymo. As a result, models perform well on specific aspects of the trajectory prediction task, rather than improving the predictive distribution itself. Moreover, the objectives induced by these metrics are partly conflicting, so models optimized for one benchmark metric typically perform poorly on the other. In contrast, we propose to train models with metric-agnostic probabilistic objectives that focus on learning good predictive distributions, and to handle different benchmark metrics via metric-specific policies applied at evaluation time, allowing a single calibrated model to be reused across multiple metrics and downstream tasks.

The idea to decouple predictive distribution estimation from reporting has also been explored by HOME [6]. HOME outputs a dense heatmap for the endpoints and introduces a mechanism to post-process this output separately for individual metrics. However, the post-processing is tightly coupled to a specific discrete endpoint representation and cannot be used as a general, model-agnostic solution. In contrast, we derive sampling-based policies that can be applied to any probabilistic forecaster and directly follow the metric definitions without model-specific tuning. We provide a detailed comparison to HOME in App. A.

3 Evaluation Metrics

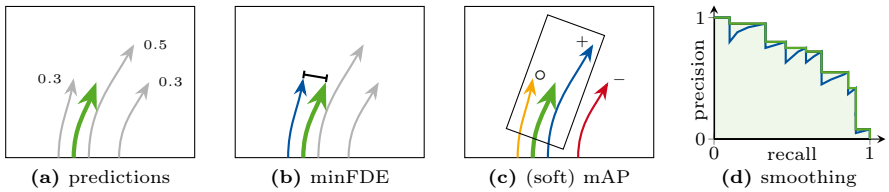
In this section, we will present common evaluation metrics for trajectory forecasting and analyze how previous methods have implicitly or explicitly adapted their optimization pipeline for either distance-based or window-based metrics. For each agent n, these metrics evaluate a set of K predicted trajectories $\{\hat{\boldsymbol{X}}_{nk}^{\text{pos}}\}_{k=1}^{K}$ where each trajectory $\hat{\boldsymbol{X}}_{nk}^{\text{pos}}$ is a sequence $(\hat{\boldsymbol{x}}_{nkt}^{\text{pos}})_{t=1}^{T}$ of 2D positions over T future timesteps, given a ground-truth trajectory $\boldsymbol{Y}_{n}^{\text{pos}} = (\boldsymbol{y}_{nt}^{\text{pos}})_{t=1}^{T}$. Some metrics additionally consider predicted confidences $\{\hat{\pi}_{nk}\}_{k=1}^{K}$ assigned to the trajectories.

3.1 Distance-Based Metrics

Distance-based metrics evaluate how close a predicted trajectory comes to the ground-truth trajectory. A common example is the minimum final displacement error (minFDE), visualized in Fig. 1b. For an agent n, it is defined as the Euclidean distance between the ground-truth endpoint $\boldsymbol{y}_{nT}^{\text{pos}}$ and the closest endpoint of K predictions $\{\hat{\boldsymbol{x}}_{nkT}^{\text{pos}}\}_{k=1}^{K}$:

$$\text{minFDE}(\boldsymbol{y}_{nT}^{\text{pos}}, \{\hat{\boldsymbol{x}}_{nkT}^{\text{pos}}\}_{k=1}^{K}) = \min_{k} \|\boldsymbol{y}_{nT}^{\text{pos}} - \hat{\boldsymbol{x}}_{nkT}^{\text{pos}}\|_2. \tag{1}$$

The minFDE for a dataset is obtained by averaging over all target agents.



(a) predictions **(b)** minFDE **(c)** (soft) mAP **(d)** smoothing

Fig. 1: Illustration of metrics. (a) Each **prediction** has a confidence assigned. **(b)** For the distance-based metric minFDE, the endpoint distance between **ground truth** and **closest prediction** is calculated. **(c)** For the window-based metric (soft) mAP, an oriented window is placed around the ground-truth endpoint. The **highest-confidence trajectory within the window** (+) counts as a true positive, **additional trajectories within the window** (∘) are either counted as false positives (mAP) or ignored (soft mAP). **Predictions outside the window** (−) count as false positives. **(d)** A **precision-recall curve** is created for each confidence score over the entire dataset and then **smoothed**; the area under the curve is the (soft) mAP.

3.2 Window-Based Metrics

Other metrics rely on windows with a specific size and orientation for evaluation. A simple example is the miss rate (MR), which counts the proportion of agents that are misses, *i.e.*, where none of the predicted endpoints are within a window W around the ground truth $\boldsymbol{y}_{nT}$. For a single agent n a miss is defined as

$$\text{Miss}(\boldsymbol{y}_{nT}, \{\hat{\boldsymbol{x}}^{\text{pos}}_{nkT}\}_{k=1}^{K}) = \mathbb{I}(\nexists k : \hat{\boldsymbol{x}}^{\text{pos}}_{nkT} \in W(\boldsymbol{y}_{nT})), \tag{2}$$

where $\mathbb{I}$ is the indicator function. Argoverse 2 defines the window to be a circle with a radius of 2 m. Waymo uses a rectangle aligned to the ground-truth heading $\boldsymbol{y}^{\text{hd}}_{nT}$ whose size depends on the agent's velocity at the last historical timestep.

Average precision (AP) is a window-based metric that also considers confidences (Fig. 1c) which are not required to sum to 1. To compute AP, predictions for all agents across the dataset are sorted based on their confidence. For each confidence level, precision and recall are computed. A prediction counts as a true positive, if its endpoint is the highest-confidence endpoint within the window W around the ground-truth endpoint. Predictions with lower confidences within the window either count as false positive (AP) or are ignored (soft AP). Predictions outside the window are considered false positives and windows not hit by any predictions become false negatives. From these, a precision-recall curve is computed (Fig. 1d), where standard object detection interpolation is applied [4].

To compute mean average precision (mAP), agents are categorized into buckets based on their ground-truth trajectories (*e.g.*, straight, left, stationary). AP is computed independently for each bucket and then averaged. Soft mAP is computed analogously and serves as the main metric for Waymo, using the same window definition as the miss rate.

Conflicting objectives. Distance-based metrics and window-based metrics reward different behaviors from a forecasting model. Distance-based metrics improve if at least one predicted trajectory is very close to the ground truth. A

model is rewarded if it puts many trajectories close together into the main mode, even if they are redundant. By contrast, window-based metrics care about coverage: models score well if they have one prediction in the ground-truth window, additional close-by trajectories either bring no benefit or are actively penalized. This incentivizes a spread between predictions. These goals are inherently conflicting: concentrating predictions to minimize distance helps minFDE but hurts coverage, whereas spreading predictions to cover many windows improves mAP or miss rate but worsens distance-based metrics.

3.3 Metric-Driven Modeling

Many current state-of-the-art models implicitly or explicitly tailor their training objectives and post-processing schemes to either distance-based or window-based metrics. Because these metrics reward conflicting behaviors, such metric-dependent training encourages models to specialize for one metric at the expense of others, rather than to learn a generally useful predictive distribution that can support diverse downstream tasks.

Models optimized to train for distance-based metrics often predict exactly $K = 6$ trajectories and directly train these outputs by minimizing a distance error between the best trajectory and the ground truth. Concretely, they apply the winner-takes-all (WTA) paradigm which selects the best mode based on Euclidean distance and regress the corresponding trajectory with an L_1 loss [14], smooth L_1 loss [36,37], a Laplacian negative log-likelihood (*i.e.*, an L_1 loss with a learned scale) [13,32,38,39], or a Gaussian negative log-likelihood (*i.e.*, a squared L_2 loss with a learned scale) [26, 30, 31]. This procedure mirrors the definition of minFDE, where only the closest prediction contributes to the metric. This improves distance-based metrics but leads to tightly clustered predictions and poor performance on window-based metrics [26, 30, 31].

Generative, GPT-like trajectory forecasting approaches, which can sample an arbitrary number of trajectories, use k-means clustering as a post-processing step to select K trajectories for reporting to the benchmarks [11, 24]. Similarly, several works apply k-means clustering on ensembles of trajectories to further reduce distance-based errors [32, 36–38]. In both cases, clustering minimizes the average squared Euclidean distance between sampled trajectories and the chosen cluster centers, which is a direct surrogate for squared minFDE under the model's predictive distribution. Again, this prioritizes minimizing distance to the ground truth over modeling the full future distribution.

Methods targeting window-based metrics adapt training and post-processing to encourage coverage of many distinct evaluation windows. A common pattern on Waymo is to output 64 candidate trajectories and then apply non-maximum suppression (NMS) as post-processing [20, 25, 26, 28, 30]. By suppressing predictions that fall close to each other, NMS decreases the number of trajectory endpoints within a window. This is rewarded by (soft) mAP, where only the highest-confidence hit within each window can contribute to the score. Some methods even make the NMS radius depend on the predicted trajectory length [17, 27, 31, 34], which approximates the velocity-dependent window size

used by Waymo and thereby aligns the post-processing even more closely with the specific definition of soft mAP. While these strategies improve mAP and miss rate, they typically worsen minFDE [26, 30, 31].

ModeSeq [40] goes one step further by aligning its training criterion precisely with the window-based metrics. In its early-match-takes-all scheme, a match is defined exactly as a true positive in the benchmark: using a circular window of radius 2 m for Argoverse 2 and a heading-aligned rectangle whose size depends on the last observed velocity for Waymo. This directly trains the model to avoid predicting multiple trajectories within the same window and tailors the optimization to the precise hit definitions of the metrics. While this improves the targeted window-based scores, it further biases the model toward the benchmark objective and away from modeling the full predictive distribution.

As many current methods are tightly coupled to particular evaluation protocols, instead of aiming for general representations, models cannot be easily applied in novel contexts. In the following, we instead advocate training with metrics-agnostic probabilistic objectives that prioritize the quality and calibration of the predictive distribution. We treat benchmark metrics as downstream problems that are addressed at evaluation time via metric-specific policies.

4 Metric-Specific Policies

Instead of adapting the training procedure to improve a certain metric, we propose to decouple the training objective from the evaluation metrics. Concretely, models should be trained to output a calibrated predictive distribution over the agents' future trajectories, which is the most general representation of uncertainty. For evaluation, we propose to continue using existing benchmarks, which require reporting K trajectories with associated confidences $\{(\hat{\boldsymbol{X}}_{nk}, \hat{\pi}_{nk})\}_{k=1}^{K}$. To obtain these, we introduce TraDiE policies for Trajectory Distribution Evaluation, which output trajectories and confidences that are optimal for a specific benchmark metric given the predictive distribution. This way, we can assess the quality of the predictive distribution through standard benchmark metrics.

Given a model providing a predictive distribution $p_n(\boldsymbol{X}_n)$ over future trajectories $\boldsymbol{X}_n$, we define a distribution-aware policy as a mapping

$$\mathcal{P} : p_n(\boldsymbol{X}_n) \mapsto \{(\hat{\boldsymbol{X}}_{nk}, \hat{\pi}_{nk})\}_{k=1}^{K}. \tag{3}$$

Ideally, such a policy produces trajectories and confidences that are optimal for a given evaluation metric under the assumption that the ground truth is drawn from the model's predictive distribution.

In the following, we design policies for the most commonly used trajectory forecasting metrics, which are evaluated at the endpoint of the forecasting horizons. For these, we only require the predictive endpoint distribution $p_{nT}(\boldsymbol{x}_{nT})$, *i.e.*, the marginal of $p_n(\boldsymbol{X}_n)$ over the final timestep T. The policies then operate on endpoints instead of full trajectories. Depending on the metric, $\boldsymbol{x}_{nT}$ is required to contain either only the positions $\boldsymbol{x}_{nT}^{\mathrm{pos}}$, or both positions and headings $\boldsymbol{x}_{nT}^{\mathrm{hd}}$. Implementation details are provided in App. C.

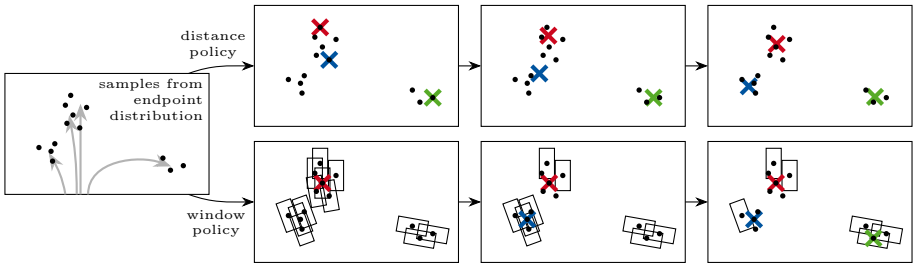


Fig. 2: Illustration of two policies. Samples from the endpoint distribution (•) of the **prediction** (left) are used to optimize three endpoints (×, ×, ×). The top row shows gradient descent for minFDE; the bottom row shows the iterative process of choosing the sample covered by the most rectangles for soft mAP.

4.1 Policies for Distance-Based Metrics

minFDE. To obtain the optimal position predictions $\{\hat{\boldsymbol{x}}^{\text{pos}}_{nkT}\}_{k=1}^{K}$ for the minFDE metric given the model's predicted distribution $p_{nT}(\boldsymbol{x}_{nT})$, we define the optimization objective for the minFDE policy as

$$\mathbb{E}_{\boldsymbol{x}_{nT}\sim p_{nT}}\left[\min_k \|\hat{\boldsymbol{x}}^{\text{pos}}_{nkT} - \boldsymbol{x}^{\text{pos}}_{nT}\|_2\right]. \tag{4}$$

This is the expected minFDE for the predictions $\{\hat{\boldsymbol{x}}^{\text{pos}}_{nkT}\}_{k=1}^{K}$ under the predictive distribution $p_{nT}(\boldsymbol{x}_{nT})$. We can optimize this by using Monte Carlo sampling to obtain a set of positions, S, from the endpoint distribution following

$$S = \{\boldsymbol{x}^{(1)}_{nT}, \ldots, \boldsymbol{x}^{(M)}_{nT}\}, \quad \boldsymbol{x}^{(m)}_{nT} \sim p_{nT}(\boldsymbol{x}_{nT}), \tag{5}$$

and minimizing the empirical approximation

$$\frac{1}{|S|}\sum_{\boldsymbol{x}_{nT}\in S} \min_k \|\hat{\boldsymbol{x}}^{\text{pos}}_{nkT} - \boldsymbol{x}^{\text{pos}}_{nT}\|_2 \tag{6}$$

by treating the endpoints $\hat{\boldsymbol{x}}^{\text{pos}}_{nkT}$ as free variables and optimizing them using gradient descent (Fig. 2, top).

4.2 Policies for Window-Based Metrics

(Soft) mAP. Computing an optimal policy for average precision is challenging, because it depends on the global ranking of all predictions across all agents and scenes. To obtain a tractable approximation, we design a per-agent heuristic. We first draw a set of candidate endpoints from the predictive distribution and associate each candidate with an evaluation window. We then greedily select endpoints that are covered by the largest number of windows, while ignoring windows that have already been hit in previous steps.

More formally, let $W_n(\boldsymbol{y}_{nT})$ denote the window defining a *hit* centered at the ground-truth endpoint $\boldsymbol{y}_{nT}^{\text{pos}}$ with orientation $\boldsymbol{y}_{nT}^{\text{hd}}$. A specific predicted endpoint $\hat{\boldsymbol{x}}_{nkT}^{\text{pos}}$ from the set of predictions $\{(\hat{\boldsymbol{x}}_{nkT}^{\text{pos}}, \hat{\pi}_{nk})\}_{k=1}^{K}$ with confidences $\hat{\pi}_{n1} \geq \hat{\pi}_{n2} \geq \cdots \geq \hat{\pi}_{nK}$ is counted as a true positive (TP) if it lies within the window $W_n(\boldsymbol{y}_{nT})$ and no prediction with higher confidence lies in the same window. Formally, we define

$$\text{TP}_{nk}(\hat{\boldsymbol{x}}_{nkT}^{\text{pos}}, \boldsymbol{y}_{nT}) = \mathbb{I}\left(\hat{\boldsymbol{x}}_{nkT}^{\text{pos}} \in W_n(\boldsymbol{y}_{nT}) \wedge \forall i < k : \hat{\boldsymbol{x}}_{niT}^{\text{pos}} \notin W_n(\boldsymbol{y}_{nT})\right). \tag{7}$$

As the ground-truth endpoint $\boldsymbol{y}_{nT}$ is not available at inference time, we use samples $\tilde{\boldsymbol{x}}_{nT}$ from the predictive distribution p_{nT} instead. The probability that $\hat{\boldsymbol{x}}_{nkT}$ is the true positive under the predictive distribution is then given by

$$\overline{\text{TP}}_{nk}(\hat{\boldsymbol{x}}_{nkT}^{\text{pos}}) = \mathbb{E}_{\tilde{\boldsymbol{x}}_{nT} \sim p_{nT}}\left[\text{TP}_{nk}(\hat{\boldsymbol{x}}_{nkT}^{\text{pos}}, \tilde{\boldsymbol{x}}_{nT})\right]. \tag{8}$$

In practice, we approximate this expectation by averaging over samples $\tilde{\boldsymbol{x}}_{nT}$ of a Monte Carlo set S (Fig. 2, bottom). To find predictions $\{\hat{\boldsymbol{x}}_{nkT}^{\text{pos}}\}_{k=1}^{K}$, a greedy heuristic iteratively picks $\hat{\boldsymbol{x}}_{nkT}^{\text{pos}}$ for $k \in \{1, \ldots, K\}$ from the same set S with

$$\hat{\boldsymbol{x}}_{nkT}^{\text{pos}} = \underset{\boldsymbol{x}_{nT} \in S}{\arg\max}\, \overline{\text{TP}}_{nk}(\boldsymbol{x}_{nT}^{\text{pos}}), \tag{9}$$

i.e., we choose the endpoint to which we assign the highest probability of becoming a true positive. We define the corresponding confidences as

$$\hat{\pi}_{nk} = \overline{\text{TP}}_{nk}(\hat{\boldsymbol{x}}_{nkT}^{\text{pos}}). \tag{10}$$

This approach has several properties that naturally align with improving average precision. By repeatedly choosing the endpoint that is most likely to be the highest-confidence true positive, we increase the chance that true positives appear earlier than false positives in the global ranking, which improves precision across confidence thresholds. The greedy selection is also a standard heuristic [9] for the maximum coverage problem: prioritizing endpoints that cover many yet-uncovered windows reduces the number of false negatives and thus improves recall and AP. Finally, because windows that have already been hit are ignored in subsequent steps, the procedure avoids placing multiple predictions in the same window, which would either be ignored (soft mAP) or penalized (mAP).

Miss rate. A forecast is considered a miss if none of the K predictions are within a window around the target. The miss rate is the proportion of misses across the dataset. Argoverse 2 defines this region as a circle with a radius of 2 m; Waymo uses the same definition as for the (soft) mAP window. Confidences do not matter for this metric. For an agent n, the probability of a hit (at least one prediction in the window) under the predictive distribution is

$$\sum_{k=1}^{K} \overline{\text{TP}}_{nk}(\hat{\boldsymbol{x}}_{nkT}^{\text{pos}}). \tag{11}$$

Thus, this metric is optimized when we select $\{\hat{\boldsymbol{x}}^{\text{pos}}_{nkT}\}_{k=1}^{K}$ to maximize Eq. (11), *i.e.*, we aim to maximize the total number of covered windows under the predictive distribution. As our greedy heuristic for (soft) mAP aims to optimize this objective, the same heuristic policy is applicable to soft mAP, mAP, and miss rate.

5 DONUT-NLL

5.1 Baseline

To study metric-agnostic training and metric-specific evaluation policies in a controlled setting, we instantiate our approach on top of DONUT [13], a decoder-only transformer for trajectory forecasting that obtains state-of-the-art results on Argoverse 2 [33]. For each agent $n \in \{1, \dots, N\}$ and each predicted trajectory $k \in \{1, \dots, K\}$, DONUT predicts a T-step future trajectory. It parameterizes positions with 1D Laplace distributions over the global x- and y-axes by predicting $\boldsymbol{\mu}^x_n, \boldsymbol{\mu}^y_n, \boldsymbol{b}^x_n, \boldsymbol{b}^y_n \in \mathbb{R}^{T\times K}$, and it models heading with von Mises distributions with parameters $\boldsymbol{\mu}^{\text{hd}}_n, \boldsymbol{\kappa}^{\text{hd}}_n \in \mathbb{R}^{T\times K}$. In addition, mixture weights $\pi_n \in \mathbb{R}^K$ are obtained from the decoder via a small prediction head, representing non-negative mode probabilities over the K trajectories.

To better align the positional uncertainty with the kinematics of the agents, we make the following adaptation to DONUT: we use the predicted mean heading $\boldsymbol{\mu}^{\text{hd}}_{nkt}$ to align the 1D distributions with the agent's local longitudinal (lg) and lateral (lt) directions, instead of aligning with global x- and y-axes. Denoting the ground-truth observation at timestep t for agent n by $\boldsymbol{y}_{nt} = (\boldsymbol{y}^{\text{lg}}_{nt}, \boldsymbol{y}^{\text{lt}}_{nt}, \boldsymbol{y}^{\text{hd}}_{nt})$, the likelihood at each timestep for a given agent and mode can be computed as

$$p_{nkt}(\boldsymbol{y}_{nt}) = \mathrm{L}(\boldsymbol{y}^{\text{lg}}_{nt} \mid \boldsymbol{\mu}^{\text{lg}}_{nkt}, \boldsymbol{b}^{\text{lg}}_{nkt}) \cdot \mathrm{L}(\boldsymbol{y}^{\text{lt}}_{nt} \mid \boldsymbol{\mu}^{\text{lt}}_{nkt}, \boldsymbol{b}^{\text{lt}}_{nkt}) \cdot \mathrm{vM}(\boldsymbol{y}^{\text{hd}}_{nt} \mid \boldsymbol{\mu}^{\text{hd}}_{nkt}, \boldsymbol{\kappa}^{\text{hd}}_{nkt}) \quad (12)$$

with $\mathrm{L}(\cdot)$ and $\mathrm{vM}(\cdot)$ denoting 1D Laplace and von Mises densities, respectively.

To optimize the mixture components for the distance-based metrics of the Argoverse 2 benchmark, DONUT uses the winner-takes-all paradigm and selects the best mode k^*_n using the average Euclidean distance over all timesteps. This component's log-likelihood is optimized via

$$\mathcal{L}_{\text{reg}} = -\sum_n \sum_{t=1}^{T} \log p_{nk^*_n t}(\boldsymbol{y}_{nt}). \quad (13)$$

A classification loss optimizes the log-likelihood of the mixture components' weights at the final timestep T, with $\bar{p}$ indicating a no-gradient version of the density:

$$\mathcal{L}_{\text{cls}} = -\sum_n \log \sum_{k=1}^{K} \pi_{nk} \bar{p}_{nkT}(\boldsymbol{y}_{nT}). \quad (14)$$

5.2 Probabilistic Training Objectives

To match our paradigm of training a model to directly optimize a predictive distribution rather than specific metrics, we adjust DONUT and propose two new variants: Traj-NLL and Step-NLL. Instead of using the winner-takes-all objective, Traj-NLL directly optimizes the predictive distribution with the per-timestep likelihood in Eq. (12) and minimizes the negative log-likelihood (NLL):

$$-\log p(\boldsymbol{Y}) = -\sum_{n} \log \sum_{k=1}^{K} \pi_{nk} \prod_{t=1}^{T} p_{nkt}(\boldsymbol{y}_{nt}). \tag{15}$$

In this formulation, the mixture weights π_{nk} are shared across all timesteps t, so each component k corresponds to a complete trajectory hypothesis.

To increase temporal flexibility, we further propose Step-NLL that defines a mixture for each timestep independently, yielding the negative log-likelihood

$$-\log p(\boldsymbol{Y}) = -\sum_{n} \sum_{t=1}^{T} \log \sum_{k=1}^{K} \pi_{nkt}\, p_{nkt}(\boldsymbol{y}_{nt}). \tag{16}$$

In this second variant, we introduce time-dependent mixture weights π_{nkt}, so that the importance of each component is allowed to vary across timesteps. This increases the expressiveness of the model, as different modes can become more or less relevant over time, but it also weakens the implicit coupling between timesteps that is present when a single π_{nk} is shared across the entire horizon.

Position distributions. Using our policies, we systematically investigate different position distribution families as drop-in replacements for DONUT's original Laplace distribution. A common choice [25, 30, 31] is the Gaussian distribution. Compared to Laplace, it has lighter tails and a less sharp peak. In preliminary experiments, Gaussians performed poorly, which is why we experiment with two generalizations of the Gaussian instead.

The generalized Gaussian distribution includes an additional parameter that controls the density's shape. Laplace and Gaussian distributions are special cases of the generalized Gaussian. We further consider discrete scale mixtures of Gaussians with a fixed number of $J = 5$ components that share a single mean but differ in scale. We provide more details in App. D.

6 Experiments

Dataset. We evaluate on the large-scale Waymo motion prediction benchmark [3], which reports a number of window-based and distance-based metrics. It consists of approximately 487k training scenes, with 44k scenes for evaluation and 45k scenes for testing. Models must predict $K = 6$ future trajectories over 8 s with assigned confidences, using 1.1 s of historical input, sampled at 10 Hz.

Additional baselines. To assess transferability beyond DONUT-NLL, we apply our metric-specific policies to the open-source trajectory forecasting models

Table 1: Study of training objectives with Laplace distributions. Applying our distance or window policies (indicated by →) combined with NLL training consistently outperforms the naive WTA predictions. Notably, naive evaluation suggests different design choices than evaluation with policies (vs.). Evaluation on Waymo [3] *val*.

Objective	Soft mAP ↑	mAP ↑	Miss rate ↓	minFDE ↓
WTA	0.3427 → **0.4764**	0.2995 → **0.4736**	0.1310 → **0.1044**	**1.0520** → 1.0824
Traj-NLL	0.3401 → **0.4921**	0.2851 → **0.4892**	0.1286 → **0.0946**	1.0874 → **1.0374**
Step-NLL	0.3595 → **0.5082**	0.3163 → **0.5053**	0.1444 → **0.0902**	1.2236 → **1.0200**

MTR [26] and QCNet [38]. MTR was originally evaluated on Waymo and is therefore optimized for window-based metrics. It predicts a 64-component Gaussian mixture and uses non-maximum suppression to select 6 trajectories. We report its original NMS outputs as the naive baseline and additionally evaluate our policies by sampling from the full 64-component mixture. By contrast, QCNet targets Argoverse 2 and directly outputs 6 trajectory modes parameterized as Laplacians. For both models, we vary the training objective and compare the original WTA losses to our probabilistic objectives (Traj-NLL and Step-NLL).

Main experiments. Our experiments are designed to answer six main questions: (i) are our TraDiE policies effective at turning a single distribution into strong performance across heterogeneous metrics, (ii) do our policies detect poorly calibrated predictive distributions, (iii) how does the training objective (WTA, Traj-NLL, Step-NLL) affect the quality of the learned predictive distribution, (iv) does optimizing for distributions instead of metrics favor different design choices, (v) which position distributions are best suited for our paradigm, and (vi) is the analysis of predictive distributions via policies model-agnostic?

In Tab. 1, we report results for the WTA, Traj-NLL, and Step-NLL training objectives, both before and after our metric-specific policies. For both NLL variants, which optimize the predictive distribution directly, we find that our policies greatly improve the corresponding metrics. The distance policy consistently reduces minFDE, and the window policy leads to substantial improvements in soft mAP, mAP, and miss rate. This confirms the first key aspect of our paradigm: given a single predictive distribution, the proposed policies can effectively optimize different metrics without retraining the underlying model.

Applying the distance policy to WTA degrades minFDE compared with naive evaluation, since WTA directly optimizes minFDE rather than a calibrated distribution. The result is therefore expected: if the predictive distribution poorly aligns with observed uncertainty, optimizing the metric under that distribution may not yield good results. This suggests that our policies expose miscalibration rather than merely improving benchmark scores through post-processing.

Comparing the different training objectives, we observe that Step-NLL consistently outperforms both WTA and Traj-NLL once TraDiE policies are applied. This directly supports our central hypothesis: optimizing the predictive distribution directly, via NLL, is more beneficial than optimizing a surrogate metric-specific loss once metric-specific policies are used at evaluation time.

Table 2: Study of position distributions using Step-NLL. Generalized Gaussians outperform the alternatives with policy optimization. Evaluation on Waymo [3] *val*.

Position Distr.	Soft mAP ↑	mAP ↑	Miss rate ↓	minFDE ↓
Laplace	0.3595 → **0.5082**	0.3163 → **0.5053**	0.1444 → **0.0902**	1.2236 → **1.0200**
Scale Mixture	0.3837 → **0.5053**	0.3533 → **0.5024**	0.1492 → **0.0909**	1.2722 → **1.0218**
Gen. Gaussian	0.3759 → **0.5101**	0.3439 → **0.5070**	0.1561 → **0.0876**	1.3127 → **1.0103**

Table 3: Additional baselines. MTR [26] consistently improves; QCNet [38] degrades for minFDE due to its poor predictive distributions. Evaluation on Waymo *val*.

Model		Soft mAP ↑	mAP ↑	Miss rate ↓	minFDE ↓
MTR	WTA	0.4288 → **0.4837**	0.4154 → **0.4798**	0.1386 → **0.0979**	1.2357 → **1.0615**
	Traj-NLL	0.3170 → **0.4422**	0.2849 → **0.4395**	0.1739 → **0.0984**	1.5428 → **1.0656**
	Step-NLL	0.2550 → **0.4873**	0.2428 → **0.4838**	0.1938 → **0.0924**	1.5520 → **1.0675**
QCNet	WTA	0.3236 → **0.4045**	0.2912 → **0.4018**	0.1628 → **0.1276**	**1.2254** → 2.2805
	Traj-NLL	0.2980 → **0.3814**	0.2475 → **0.3791**	0.1743 → **0.1343**	**1.3416** → 2.9618
	Step-NLL	0.2401 → **0.4209**	0.1964 → **0.4184**	0.1922 → **0.1216**	**1.4207** → 3.1753

We can further observe that design decisions differ between models trained to optimize metrics and models trained for distributions: without policies, the best objective for minFDE is WTA, which directly optimizes the distance-based loss. By contrast, Step-NLL outputs the best predictive distribution, as it consistently outperforms the other objectives if we apply our policies.

In Tab. 2, we investigate how the choice of position distribution influences performance under our paradigm. We compare Laplace, generalized Gaussian, and Gaussian scale mixture distributions, each combined with Step-NLL and evaluate them naively and with our metric-specific policies. Once our policies are applied, the generalized Gaussian distribution consistently outperforms the alternatives across all metrics. This suggests that its additional shape flexibility over the Laplacian is beneficial for learning a well-calibrated predictive distribution that can be effectively exploited by the downstream policy.

As before, the naive evaluation again favors different design choices. For minFDE, naive Laplace performs best, likely because its sharper shape encourages the model to place modes close to the ground-truth endpoint. In contrast, naive Gaussian scale mixtures perform best for soft mAP and mAP, presumably because their smoother and broader predictive distributions improve coverage. The consistent advantage of the generalized Gaussian after policy optimization indicates that it learns a better-calibrated predictive distribution. We provide full ablations over all combinations of losses and distribution families in App. E.

Additional baselines. To evaluate whether our findings extend to other models, we apply our policies to MTR [26] and QCNet [38] and report the results in Tab. 3. MTR improves consistently, and even outperforms the original post-processing, indicating that the distribution-aware policies transfer to a different architecture. Changing the training loss only has a minor impact in this setup.

Table 4: Comparison to state of the art on Waymo [3] *test*. We mark ensembles with E, and omit methods that train on additional data or use LiDAR.

Model		Soft mAP ↑	mAP ↑	Miss rate ↓	minFDE ↓
DriveGPT [11]		0.3795	—	0.1236	1.0609
RMP-YOLO (e2e) [30]		0.3828	0.3440	0.1354	1.0932
IMPACT (e2e) [31]		0.4434	0.4253	0.1274	<u>1.0497</u>
ModeSeq [40]		0.4487	0.4450	0.1244	1.0836
EDA [17]		0.4510	0.4401	0.1169	1.1702
UniMotion [29]		0.4642	0.4534	0.1162	1.1643
RMP-YOLO [30]		0.4673	0.4523	0.1160	1.1697
TrajFlow [34]		0.4710	0.4604	0.1162	1.1667
IMPACT [31]		0.4721	0.4609	0.1143	1.1540
ModeSeq [40] E		0.4737	<u>0.4665</u>	0.1204	1.1766
RMP-YOLO [30] E		0.4737	0.4531	<u>0.1084</u>	1.1188
IMPACT [31] E		<u>0.4801</u>	0.4598	0.1087	1.1295
DONUT-NLL	window	**0.5018**	**0.4987**	**0.0900**	1.3280
	distance	0.1649	0.1135	0.1199	**1.0304**

For QCNet, minFDE degrades significantly under the distance policy. This suggests that QCNet has not learned a well-calibrated predictive distribution. Our policies directly optimize the expectation of the metric under the model's output distribution, so if there is any discrepancy with the real-world uncertainty, the policy selects trajectories that are suboptimal in reality. We attribute this mismatch to QCNet's position distribution parameterization: its Laplacians are oriented according to the last historical heading, and therefore become misaligned with the future motion direction on curved trajectories. To compensate, QCNet inflates the uncertainty into a more circular shape, which makes the minFDE-optimal endpoints spread out. Window-based metrics are less affected because the distribution's main mass stays concentrated around each mode.

This is exactly what our policies are meant to expose: if the predictive distribution is miscalibrated, policy optimization cannot find good trajectories for real data. In this sense, policy-optimized evaluation enables a direct diagnostic of distribution quality that naive metric computation can obscure.

Comparison to the state of the art. In Tab. 4, we apply both of our metric-specific policies to DONUT-NLL with the Step-NLL loss and the generalized Gaussian position distribution, and compare with state-of-the-art methods. We find that our policies allow a single model to obtain state-of-the-art performance across both distance-based and window-based metrics with the same model weights, whereas existing methods typically perform well for only one of these when naively evaluated. DONUT-NLL outperforms IMPACT (e2e) [31] on minFDE, while using roughly 5× fewer parameters. For soft mAP, mAP, and miss rate, it even outperforms ensemble methods while using only a single model. Together, these results validate our proposed paradigm: training a metric-

agnostic predictive distribution with probabilistic objectives is an effective way to model future trajectories, and the quality of the predictive distribution can be evaluated on benchmarks using our metric-specific policies.

7 Limitations and Discussion

Our TraDiE policies provide a mechanism for adapting general-purpose trajectory prediction models that are trained to output calibrated predictive distributions to common trajectory forecasting metrics such as minFDE and soft mAP. This allows a single predictive model to be flexibly adapted to different downstream objectives without retraining. The main drawback is the additional computational cost incurred at inference time. Because our policies are intended for evaluation and not for direct deployment on autonomous vehicles, we did not focus on optimizing efficiency. In practical settings, however, an explicit trade-off between output quality and computational efficiency would be required.

Another limitation is that our results are restricted to Waymo. Preliminary experiments on Argoverse 2 suggest that Step-NLL is prone to overfitting, possibly because the step-wise mixture weights introduce additional modeling flexibility. Nevertheless, on Waymo, DONUT-NLL combined with our TraDiE policies is the only method to date that reaches state-of-the-art performance on both window-based and distance-based metrics without requiring retraining.

8 Conclusion

In this paper, we propose to shift the focus of trajectory forecasting from metric-specific training to learning well-calibrated predictive distributions and treating benchmark metrics as downstream tasks via metric-specific policies. Concretely, we design sampling-based policies for minFDE, miss rate, and (soft) mAP. We instantiate this framework with DONUT-NLL, a negative log-likelihood variant of DONUT. Our experiments on the Waymo motion prediction benchmark confirm that metric-agnostic probabilistic training, combined with our policies, yields a single model that can be adapted post hoc to diverse metrics while surpassing state-of-the-art results. Importantly, this distribution-centric view changes the preferred design choices: DONUT-NLL outperforms WTA, i.e., metric-specific training. This indicates that centering trajectory forecasting on predictive distributions fundamentally changes how forecasting models should be designed. Extending our metric-specific policies to multi-agent forecasting and implementing continuity-aware objectives are interesting directions for future work.

Acknowledgments. This work was partially funded by the BMBF project 6GEM (16KISK036K). The authors gratefully acknowledge the computing time provided to them at the NHR Center NHR4CES at RWTH Aachen University (project number p0024673). This is funded by the Federal Ministry of Research, Technology and Space, and the state governments participating on the basis of the resolutions of the GWK for national high performance computing at universities (`www.nhr-verein.de/unsere-partner`).

References

1. Chai, Y., Sapp, B., Bansal, M., Anguelov, D.: MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction. In: CoRL (2020)
2. Cui, H., Radosavljevic, V., Chou, F.C., Lin, T.H., Nguyen, T., Huang, T.K., Schneider, J., Djuric, N.: Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks. In: ICRA (2019)
3. Ettinger, S., Cheng, S., Caine, B., Liu, C., Zhao, H., Pradhan, S., Chai, Y., Sapp, B., Qi, C.R., Zhou, Y., Yang, Z., Chouard, A., Sun, P., Ngiam, J., Vasudevan, V., McCauley, A., Shlens, J., Anguelov, D.: Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset. In: ICCV (2021)
4. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. IJCV (2010)
5. Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., Schmid, C.: VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation. In: CVPR (2020)
6. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: HOME: Heatmap Output for future Motion Estimation. In: ITSC (2021)
7. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: GOHOME: Graph-Oriented Heatmap Output for future Motion Estimation. In: ICRA (2022)
8. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: THOMAS: Trajectory Heatmap Output with learned Multi-Agent Sampling. In: ICLR (2022)
9. Hochbaum, D.S.: Approximating Covering and Packing Problems: Set Cover, Vertex Cover, Independent Set, and Related Problems. In: Hochbaum, D.S. (ed.) Approximation Algorithms for NP-hard Problems, chap. 3, pp. 94–143. PWS Publishing Company (1997)
10. Hong, J., Sapp, B., Philbin, J.: Rules of the Road: Predicting Driving Behavior with a Convolutional Model of Semantic Interactions. In: CVPR (2019)
11. Huang, X., Wolff, E.M., Vernaza, P., Phan-Minh, T., Chen, H., Hayden, D.S., Edmonds, M., Pierce, B., Chen, X., Jacob, P.E., Chen, X., Tairbekov, C., Agarwal, P., Gao, T., Chai, Y., Srinivasa, S.: DriveGPT: Scaling Autoregressive Behavior Models for Driving. In: ICML (2025)
12. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
13. Knoche, M., de Geus, D., Leibe, B.: DONUT: A Decoder-Only Model for Trajectory Prediction. In: ICCV (2025)
14. Lan, Z., Jiang, Y., Mu, Y., Chen, C., Li, S.E.: SEPT: Towards Efficient Scene Representation Learning for Motion Prediction. In: ICLR (2024)
15. Lee, N., Choi, W., Vernaza, P., Choy, C.B., Torr, P.H., Chandraker, M.: Desire: Distant future prediction in dynamic scenes with interacting agents. In: CVPR (2017)
16. Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., Urtasun, R.: Learning Lane Graph Representations for Motion Forecasting. In: ECCV (2020)
17. Lin, L., Lin, X., Lin, T., Huang, L., Xiong, R., Wang, Y.: EDA: Evolving and Distinct Anchors for Multimodal Motion Prediction. In: AAAI (2024)
18. Liu, Y., Zhang, J., Fang, L., Jiang, Q., Zhou, B.: Multimodal Motion Prediction with Stacked Transformers. In: CVPR (2021)
19. Mohamed, A., Qian, K., Elhoseiny, M., Claudel, C.: Social-STGCNN: A Social Spatio-Temporal Graph Convolutional Neural Network for Human Trajectory Prediction. In: CVPR (2020)

20. Nayakanti, N., Al-Rfou, R., Zhou, A., Goel, K., Refaat, K.S., Sapp, B.: Wayformer: Motion Forecasting via Simple & Efficient Attention Networks. In: ICRA (2023)
21. Phan-Minh, T., Grigore, E.C., Boulton, F.A., Beijbom, O., Wolff, E.M.: CoverNet: Multimodal Behavior Prediction using Trajectory Sets. In: CVPR (2020)
22. Ruan, H., Yu, H., Yang, W., Fan, S., Nie, Z.: Learning Cooperative Trajectory Representations for Motion Forecasting. In: NeurIPS (2024)
23. Salzmann, T., Ivanovic, B., Chakravarty, P., Pavone, M.: Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In: ECCV (2020)
24. Seff, A., Cera, B., Chen, D., Ng, M., Zhou, A., Nayakanti, N., Refaat, K.S., Al-Rfou, R., Sapp, B.: MotionLM: Multi-Agent Motion Forecasting as Language Modeling. In: ICCV (2023)
25. Shi, C., Shi, S., Jiang, L.: MTR v3: 1st Place Solution for 2024 Waymo Open Dataset Challenge - Motion Prediction (2024)
26. Shi, S., Jiang, L., Dai, D., Schiele, B.: Motion Transformer with Global Intention Localization and Local Movement Refinement. In: NeurIPS (2022)
27. Shi, S., Jiang, L., Dai, D., Schiele, B.: MTR-A: 1st Place Solution for 2022 Waymo Open Dataset Challenge - Motion Prediction (2022)
28. Shi, S., Jiang, L., Dai, D., Schiele, B.: MTR++: Multi-Agent Motion Prediction With Symmetric Scene Modeling and Guided Intention Querying. IEEE TPAMI (2024)
29. Song, N., Jiang, J., Li, J., Zhu, X., Zhang, L.: UniMotion: A Unified Motion Framework for Simulation, Prediction and Planning. In: NeurIPS (2025)
30. Sun, J., Li, J., Liu, T., Yuan, C., Sun, S., Huang, Z., Wong, A., Tee, K.P., Ang, M.H.: RMP-YOLO: A Robust Motion Predictor for Partially Observable Scenarios even if You Only Look Once. In: ICRA (2025)
31. Sun, J., Yue, X., Li, J., Shen, T., Yuan, C., Sun, S., Guo, S., Zhou, Q., Ang Jr, M.H.: IMPACT: Behavioral Intention-Aware Multimodal Trajectory Prediction With Adaptive Context Trimming. RAL (2025)
32. Wang, M., Ren, X., Jin, R., Li, M., Zhang, X., Yu, C., Wang, M., Yang, W.: FutureNet-LOF: Joint Trajectory Prediction and Lane Occupancy Field Prediction with Future Context Encoding. In: ICRA (2025)
33. Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J.K., Ramanan, D., Carr, P., Hays, J.: Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting. In: NeurIPS Datasets and Benchmarks (2021)
34. Yan, Q., Zhang, B., Zhang, Y., Yang, D., White, J., Chen, D., Liu, J., Liu, L., Zhuang, B., Shi, S., Liao, R.: Trajflow: Multi-modal motion prediction via flow matching. In: IROS (2025)
35. Zeng, W., Liang, M., Liao, R., Urtasun, R.: Lanercnn: Distributed representations for graph-centric motion forecasting. In: IROS (2021)
36. Zhang, B., Song, N., Gao, B., Zhang, L.: Relative Position Matters: Trajectory Prediction and Planning with Polar Representation. arXiv preprint arXiv:2508.11492 (2025)
37. Zhang, B., Song, N., Zhang, L.: DeMo: Decoupling Motion Forecasting into Directional Intentions and Dynamic States. In: NeurIPS (2024)
38. Zhou, Z., Wang, J., Li, Y.H., Huang, Y.K.: Query-Centric Trajectory Prediction. In: CVPR (2023)
39. Zhou, Z., Ye, L., Wang, J., Wu, K., Lu, K.: HiVT: Hierarchical Vector Transformer for Multi-Agent Motion Prediction. In: CVPR (2022)

40. Zhou, Z., Zhou, H., Hu, H., Wen, Z., Wang, J., Li, Y.H., Huang, Y.K.: ModeSeq: Taming Sparse Multimodal Motion Prediction with Sequential Mode Modeling. In: CVPR (2025)