RetroMotion: Retrocausal Motion Forecasting Models are Instructable

Royden Wagner[1] **Ömer Şahin Taş**[2] **Felix Hauser**[1] **Marlon Steiner**[1] **Dominik Strutz**[1]
Abhishek Vivekanandan[2] **Carlos Fernandez**[1] **Christoph Stiller**[1]
[1]Karlsruhe Institute of Technology [2]FZI Research Center for Information Technology

Abstract

Motion forecasts of road users (i.e., agents) vary in complexity as a function of scene constraints and interactive behavior. We address this with a multi-task learning method for motion forecasting that includes a retrocausal flow of information. The corresponding tasks are to forecast (1) marginal trajectory distributions for all modeled agents and (2) joint trajectory distributions for interacting agents. Using a transformer model, we generate the joint distributions by re-encoding marginal distributions followed by pairwise modeling. This incorporates a retrocausal flow of information from later points in marginal trajectories to earlier points in joint trajectories. Per trajectory point, we model positional uncertainty using compressed exponential power distributions. Notably, our method achieves state-of-the-art results in the Waymo Interaction Prediction dataset and generalizes well to the Argoverse 2 dataset. Additionally, our method provides an interface for issuing instructions through trajectory modifications. Our experiments show that regular training of motion forecasting leads to the ability to follow goal-based instructions and to adapt basic directional instructions to the scene context.

Code: `https://github.com/kit-mrt/future-motion`

1 Introduction

Motion in traffic scenarios ranges from complex, interactive behaviors among road users in urban environments to more uniform trajectories on highways. Data-driven methods for motion forecasting [Ngiam et al., 2022, Shi et al., 2022, Cui et al., 2023] address this with multiple choice learning [Guzman-Rivera et al., 2012, Lee et al., 2016], where choices are trajectories of future positions. In complex scenarios with many road users (i.e., agents), the distribution over future trajectories is likely multimodal. A common simplification is to model marginal distributions per agent [Nayakanti et al., 2023, Zhou et al., 2023a, Zhang et al., 2024]. To improve interaction modeling, recent methods predict joint distributions over multiple agents [Ngiam et al., 2022, Seff et al., 2023, Jiang et al., 2023]. However, when modeling the joint distribution over all agents in a scenario, the output space grows exponentially with the number of agents.

We address this with a multi-task learning method for motion forecasting. The corresponding tasks are to forecast (1) marginal trajectory distributions for all modeled agents and (2) joint trajectory distributions for interacting agents. We generate joint trajectory distributions by re-encoding marginal distributions followed by pairwise modeling. Our two-stage decoding mechanism incorporates a retrocausal flow of information from later points in marginal trajectories to earlier points in joint trajectories. This lowers the modeling burden on initial marginal forecasts and provides an interface for issuing instructions through trajectory modifications. Remarkably, regular training of motion forecasting without instructions leads to the ability to follow goal-based instructions and to adapt basic directional instructions to the scene context.

Preprint. Under review.

Recent planning algorithms for self-driving vehicles [Tas et al., 2023, Geisslinger et al., 2023, Bouzidi et al., 2024] benefit from probabilistic uncertainty estimates. We train our decoders to model positional uncertainty in future motion using maximum likelihood estimation. While related methods assume either normal [Shi et al., 2022, 2024, Zhang et al., 2024] or Laplace distributions [Zhou et al., 2023a,b], our decoders predict variably shaped exponential power distributions. Our experiments show that learned distribution shapes tend to be Laplace-like, yet outperform plain Laplace distributions in terms of forecasting accuracy. In addition, we further improve our results by compressing the location parameters of distributions with discrete cosine transforms.

Our main contributions are:

1. Our method connects marginal and joint motion forecasts through a retrocausal flow of information. This reduces the modeling burden on initial marginal forecasts and enables us to issue instructions by modifying marginal trajectories (see Figure 3).
2. We model positional uncertainty with compressed exponential power distributions, resulting in higher forecasting accuracy than with normal or Laplace distributions (see Section 4.4).

2 Related work

Marginal motion forecasting methods model future motion per agent using marginal distributions of trajectories [Nayakanti et al., 2023, Zhou et al., 2023a, Zhang et al., 2024]. Related methods extend marginal forecasting models with auxiliary goal prediction[1] objectives [Gilles et al., 2022, Shi et al., 2022, 2024] or perform conditional [Sun et al., 2022, Rowe et al., 2023] and joint forecasting [Luo et al., 2022] in a second step. With auxiliary goal prediction, interaction modeling is more implicit than with joint modeling. The JFP method [Luo et al., 2022] is most related to our method, but performs non-differentiable non-maximum suppression prior to joint modeling. This limits the information flow from marginal to joint forecasts.

Joint motion forecasting methods model future motion with joint distributions of trajectories over multiple agents. A common method is to reduce the full joint distribution of each combination of per-agent trajectories using global latent variables, i.e. learned global embeddings [Casas et al., 2020, Ngiam et al., 2022, Girgis et al., 2021, Zhou et al., 2023b]. Jiang et al. [2023] perform joint motion forecasting as denoising diffusion process and denoise sets of noisy trajectories conditioned on the scene context. Seff et al. [2023] cast joint motion forecasting as language modeling using a vocabulary of discrete motion vectors and joint roll-outs for multiple agents. These generative modeling approaches are limited to forecasts of two agents and exhibit high inference latency (e.g., 409 ms for MotionDiffuser [Jiang et al., 2023]). In contrast, our method forecasts motion for up to 8 agents and is realtime capable.

Retrocausal forecasting models incorporate a flow of information from later parts of predictions to earlier ones. Consistent causal-retrocausal modeling [Zimmermann et al., 2012] extends RNNs with a retrocausal information flow, which is directed from later states to earlier states. Consistent Koopman autoencoders [Azencot et al., 2020] retrocausally predict backward dynamics. In language modeling, bidirectional BERT models [Devlin et al., 2019, Warner et al., 2024] include information flows from later words in a sentence to earlier ones. For motion forecasting, recent self-supervised pre-training methods [Cheng et al., 2023, Lan et al., 2024, Wagner et al., 2024b] include retrocausal objectives, where earlier parts of trajectories are masked and reconstructed using later parts. In contrast to our work, none of these methods explore using retrocausal mechanisms for issuing instructions.

Instructable machine learning models[2] are designed to follow instructions effectively, with the goal of improving alignment [Sun et al., 2024], adaptability [Raad et al., 2024], or generalization [Guhur et al., 2023, Sarch et al., 2024, Kim et al., 2024]. Vision-and-language navigation (VLN) methods [Chen et al., 2021, Pashevich et al., 2021] combine language and computer vision models to follow text-based navigation instructions. While these methods are trained or fine-tuned to follow instructions using supervised learning, we show that regular training of motion forecasting leads to the ability to follow goal-based instructions and to adapt basic directional instructions to the scene context.

[1] Also referred to as dense prediction [Shi et al., 2022].

[2] Also referred to as instructable AI agents [Sun et al., 2024, Raad et al., 2024].

Compressed trajectory representations are typically generated using lossy compression, which smoothes perception errors by removing high frequency components. From a modeling perspective, compression reduces computational complexity. Related methods compress trajectories using principal component analysis [Jiang et al., 2023], discrete cosine transforms [Mao et al., 2019], eigenvalue decomposition [Bae et al., 2023], or Bézier curves [Hug et al., 2020]. Unlike Mao et al. [2019], we compress probabilistic representations of trajectories using discrete cosine transforms. In contrast to Jiang et al. [2023], Bae et al. [2023], our method is not data dependent, making it robust to dataset-specific noise.

3 Method

We forecast multiple trajectories per modeled agent. A trajectory is a sequence of future positions (x- and y-coordinates) with positional uncertainties represented as probability densities. Using mixture distributions, our method decomposes motion forecasts in the following three ways.

3.1 Decomposing exponential power distributions

We model the positional uncertainty per trajectory point as density of an exponential power distribution[3]. Platikurtic densities (i.e., with flatter peaks) and densities with wide tails result in large uncertainties close to forecasted positions, which is undesirable in subsequent planning. Therefore, we limit the shape parameter of exponential power distributions to the range of 1.0 to 2.0. We approximate this as a mixture distribution of a bivariate normal and a bivariate Laplace distribution, with the mixture density

$$\mathcal{D}(w, \boldsymbol{\phi}) = w \cdot \text{Normal}(\boldsymbol{\phi}) + (1 - w) \cdot \text{Laplace}(\boldsymbol{\phi}), \tag{1}$$

for learned weights $0 \leq w \leq 1$ and shared density parameters $\boldsymbol{\phi} = (\mu_x, \mu_y, \sigma_x, \sigma_y)$, with location parameters μ and scale parameters σ. Following common practice [Nayakanti et al., 2023, Zhou et al., 2023a,b], we model the x- and y-coordinates as uncorrelated random variables. In the following, we include w in the tuple of density parameters $\boldsymbol{\phi}$.

3.2 Decomposing marginal trajectory distributions

We train a distinct decoder to perform marginal motion forecasting (i.e., per-agent). Following common practice [Chai et al., 2020, Zhang et al., 2024, Nayakanti et al., 2023], we predict the density of a mixture distribution at each future time step t and fix mixture weights over time. The per-agent mixture components describe future positions of the same agent, but from different trajectories. Formally, we follow Bishop [1994] and express this as

$$\mathcal{P}_t^{\text{marginal}}(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta}) = \sum_{k=1}^{K} m_k(\boldsymbol{x}; \boldsymbol{\theta}) \cdot \mathcal{D}\big(\boldsymbol{y} \mid \boldsymbol{\phi}_{t,k}(\boldsymbol{x}; \boldsymbol{\theta})\big), \tag{2}$$

where $\boldsymbol{x}$ is the input (cf. Section 3.5), $\boldsymbol{y}$ the target vector (i.e., ground truth trajectory), $\boldsymbol{\theta}$ are the parameters of our model, $t \in \{1, ..., T\}$ are future time steps, K is the number of trajectories, k indexes the corresponding mixture components, and $\boldsymbol{m}$ are mixture weights. Unlike mixture weights, density parameters $\boldsymbol{\phi}$ are variable across components and time steps.

3.3 Decomposing joint trajectory distributions

We generate joint trajectory distributions by re-encoding marginal distributions followed by pairwise modeling. To effectively exchange information between agents, we transform all trajectories to the local reference frame of agent 1 and use the scene context embeddings of agent 1 (cf. Section 3.5). Afterwards, we exchange information via attention mechanisms and decode joint trajectory distributions using multi-agent mixture components (see Figure 1). Per time step t, each mixture component is a mixture density itself, describing one future position of each modeled agent

$$\mathcal{P}_t^{\text{joint}}(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta}) = \sum_{k=1}^{K} c_k \sum_{a=1}^{A} M_{k,a}(\boldsymbol{x}; \boldsymbol{\theta}) \cdot \mathcal{D}\big(\boldsymbol{y} \mid \boldsymbol{\phi}_{t,k,a}(\boldsymbol{x}; \boldsymbol{\theta})\big), \tag{3}$$

[3] Also known as symmetric generalized normal distributions.

where A is the number of agents and $\boldsymbol{M}$ is a matrix of per-agent mixture weights. We compute multi-agent mixture weights with $\boldsymbol{c} = \text{softmax}\big(\sum_{a=1}^{A} \boldsymbol{M}_{1:K,a}/\tau\big)$ and a tunable temperature parameter τ.

As shown in Figure 1, this approximates the joint distribution of all combinations of per-agent mixture components by focusing on the diagonal query pairs in matrix form. Specifically, we decode joint trajectory distributions using only the on-diagonal query pairs. Off-diagonal queries can update on-diagonal queries through attention mechanisms (inter-query attention in Figure 1), which compresses information from all K^2 possible combinations into K query pairs.

For both marginal and joint motion forecasts, we follow related methods [Chai et al., 2020, Shi et al., 2022, Zhou et al., 2023a] and use the out-most mixture weights (m_k and c_k) as confidence scores.

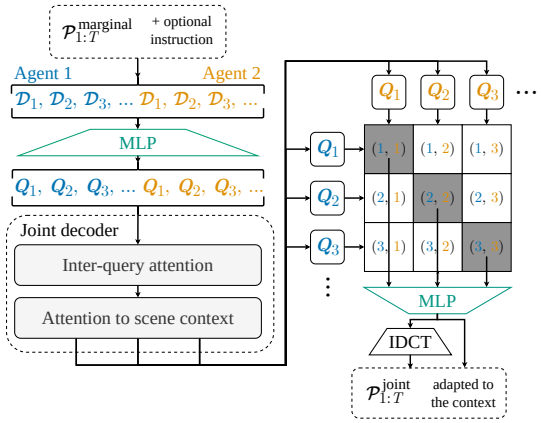


Figure 1: **From marginal to joint trajectories.** We use an MLP to generate query matrices $\boldsymbol{Q}$ from marginal trajectories and exchange information between queries and scene context with attention mechanisms. Afterwards, we decode joint trajectories $\mathcal{P}_{1:T}^{\text{joint}}$ from pairs of queries at the same index. This compresses information from all K^2 possible combinations into K query pairs.

3.4 Compressing location parameters of probability densities

Most motion forecasting methods regress trajectories at a frequency of 10 Hz [Zhou et al., 2023a, Zhang et al., 2024, Wagner et al., 2024a], allowing models to predict sudden changes between successive positions that are physically impossible yet close to the ground truth. Such forecasts resemble noisy versions of smooth ground truth trajectories. Therefore, we use a compressed probabilistic representation of trajectories without high frequency components. Specifically, we incorporate the inverse discrete cosine transform (IDCT) into our model to internally represent density location parameters as a sum of cosine functions (see Figure 1). We hypothesize that this is a natural choice for transformer models given the use of sinusoidal positional encodings [Vaswani et al., 2017]. To compress, we limit the frequencies in the IDCT to the lower end. This method is data-independent, making it invariant to dataset or setup-specific noise (e.g., produced by errors of perception models).

3.5 Scene encoder

We follow Gao et al. [2020] and represent multimodal inputs (i.e., past trajectories, lane data, and traffic light states) as polyline vectors. We sample temporal features (past positions and traffic light states) with a frequency of 10 Hz and static spatial features (lane markings and road borders) with a resolution of 0.5 meters. We generate embeddings for each modality with 3-layer MLPs, add sinusoidal positional encodings, and process the embeddings with transformer encoder modules [Vaswani et al., 2017]. Following Nayakanti et al. [2023], we initially process local agent-centric views within scenes (centered around each modeled agent) and compress them using cross-attention. We then change the batch dimension from the agent index to the scene index, and concatenate learned embeddings of Cartesian transformation matrices from agent-centric views into a global reference frame (cf. Jiang et al. [2023]). Finally, we add global sinosoidal positional encodings and generate global scene context representations with further self-attention mechanisms. Our two decoders (Section 3.2 and Section 3.3) decode probabilistic motion forecasts from this scene context.

3.6 Loss formulation

We train our model using maximum likelihood estimation with a multi-task loss that covers the objectives described in Section 3.2 and Section 3.3. Formally, we batchwise minimize the negative log-likelihood for forecasting the ground truth trajectories.

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T} -\ln\left(\mathcal{P}_{t,k=\hat{k}}^{\text{joint}}\right) - \lambda^{\text{marginal}} \ln\left(\mathcal{P}_{t,k=\hat{k}}^{\text{marginal}}\right), \tag{4}$$

where N is number of samples in a batch, $\lambda^{\text{marginal}}$ is a tunable weighting factor. We optimize this objective with multiple trajectories per agent by backpropagating only the error for the trajectories that are closest to the ground truth trajectories[4]. We measure the distance to the ground truth using the ℓ_2-norm. For marginal forecasts $\mathcal{P}^{\text{marginal}}_{1:T}$, we select the best trajectory index $\hat{k}$ per agent. For joint forecasts $\mathcal{P}^{\text{joint}}_{1:T}$, we select the best set of trajectories at the same index $\hat{k}$ for agent pairs (cf. Figure 1).

4 Experiments

In this section, we evaluate the motion forecasting performance of our method using the Waymo Open Motion [Ettinger et al., 2021] and the Argoverse 2 Forecasting [Wilson et al., 2023] datasets. Afterwards, we show that regular training of motion forecasting leads to the ability to follow goal-based instructions and to adapt basic directional instructions to the scene context. Furthermore, we analyze learned density representations using forecasting metrics, mixture weights, and metrics for neural regression collapse [Andriopoulos et al., 2025].

4.1 Interactive motion forecasting

Model configuration: We configure our marginal decoder to forecast marginal trajectories of 8 agents and our joint decoder to forecast joint trajectory sets for 2 agents per scenario. Our scene encoder processes the 128 closest map polylines and up to 48 trajectories of surrounding agents per modeled agent. We include an IDCT transform in our model that reconstructs 80 location parameters (for x- and y-coordinates) from 16 predicted DCT coefficients. We build a sparse mixture of experts model (SMoE) [Fedus et al., 2022] of 3 variations of our model (cf. Appendix A.2). All expert models are trained independently (cf. Appendix A.1). At inference, we use a rule-based router that selects one model based on the agent type and perform non-maximum suppression on joint trajectory forecasts (cf. Appendix A.3).

Results: Table 1 presents motion forecasting metrics (cf. Appendix A.5) for interactive forecasting on the Waymo Open Motion dataset. The QCNeXt model [Zhou et al., 2023b] performs strongly on the distance-based minADE and minFDE metrics. Averaged over all agent types, our RetroMotion model outperforms all other methods on the main metric mAP, indicating that our model predicts higher confidence scores for trajectories close to the ground truth.

Furthermore, we follow Ngiam et al. [2022] and evaluate the marginal predictions of our marginal decoder as joint predictions on the validation split (cf. marginal as joint configuration in Table 1). All forecasting metrics are significantly worse than for the joint predictions. This highlights the ability of our model to perform joint modeling by re-encoding and adapting the marginal predictions to the predictions of surrounding agents. Figure 2 shows qualitative results of our method on the validation split. We show joint and combined (marginal and joint, see bottom right) forecasts to highlight that our method is suitable for more complex scenarios with many agents.

Split	Agent type	Method (config)	Venue	mAP↑	minADE↓	minFDE↓	MR↓	OR↓
Test	All	Scene Transformer (joint) [Ngiam et al., 2022]	ICLR'22	0.1192	0.9774	2.1892	0.4942	0.2067
		GameFormer (joint) [Huang et al., 2023]	ICCV'23	0.1376	0.9161	<u>1.9373</u>	0.4531	0.2112
		JointMotion (HPTR) [Wagner et al., 2024b]	CoRL'24	0.1869	0.9129	2.0507	0.4763	0.2037
		MotionDiffuser [Jiang et al., 2023]	CVPR'23	0.1952	<u>0.8642</u>	1.9482	0.4300	0.2004
		JFP [Luo et al., 2022]	CoRL'22	0.2050	0.8817	1.9905	0.4233	0.1835
		MotionLM [Seff et al., 2023]	ICCV'23	0.2178	0.8911	2.0067	<u>0.4115</u>	0.1823
		MTR++ [Shi et al., 2024]	TPAMI'24	0.2326	0.8795	1.9509	0.4143	**0.1665**
		QCNeXt [Zhou et al., 2023b]		0.2352	**0.7750**	**1.6772**	**0.3838**	0.1946
		BeTopNet [Liu et al., 2024]	NeurIPS'24	0.2412	0.9744	2.2744	0.4355	<u>0.1695</u>
		RetroMotion (SMoE) [ours]		<u>0.2422</u>	0.9029	2.0245	0.4433	0.2007
		RetroMotion (SMoE hybrid) [ours]		**0.2519**	0.9256	2.0890	0.4347	0.1927
Val	All	MotionLM (single replica)	ICCV'23	0.1687	1.0345	2.3886	0.4943	-
		MTR++	TPAMI'24	<u>0.2398</u>	<u>0.8859</u>	<u>1.9712</u>	**0.4106**	-
		RetroMotion (SMoE) [ours]		**0.2515**	**0.8718**	**1.9383**	<u>0.4216</u>	0.1994
		RetroMotion (marginal as joint) [ours]		0.1797	0.8864	2.0118	0.4545	0.2200

Table 1: **Interactive motion forecasting results.** All methods are evaluated on the interactive splits of the Waymo Open Motion Dataset [Ettinger et al., 2021]. The main challenge metric is the mAP. Best scores are **bold**, second best are <u>underlined</u>.

[4]Also referred to as winner-takes-all [Shi et al., 2022] or loss with hard assignment [Zhang et al., 2024].

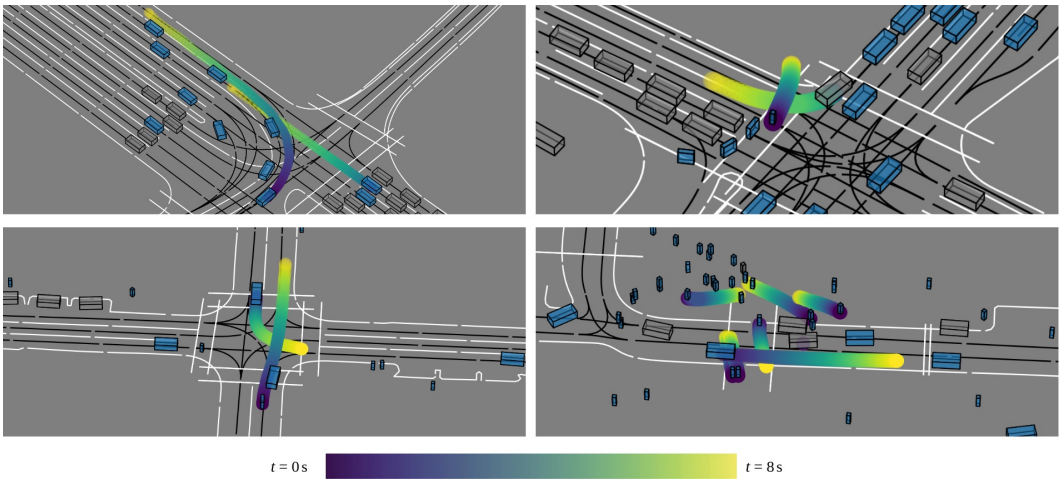


Figure 2: **Joint and combined motion forecasts of our model.** Dynamic agents are shown in blue, static agents in grey (determined at $t = 0$ s). Lanes are black lines and road markings are white lines. Top left: forecasts for two cars, top right: a car yielding to a pedestrian on a crosswalk, bottom left: forecasts for a car and a cyclist, and bottom right: combined forecasts of two cars and six pedestrians.

Since the top4-methods over all agent types in Table 1 are within one mAP, we further compare them in Table 2. Fully data-driven methods with learned anchor[5] initialization (QCNeXt and RetroMotion) tend to perform better on more common agent types, while using larger models and more anchors with fixed initialization seems to improve the results for cyclists. The comparably worse results for cyclists indicate that our method would significantly improve with more training samples, as in the $1000\times$ larger internal dataset mentioned in [Luo et al., 2022]. On average, our method outperforms related methods that require more active parameters.

Method	Params (active)	Anchor init.	Anchors	Vehicle		Pedestrian		Cyclist	
				mAP	% of scenes	mAP	% of scenes	mAP	% of scenes
MTR++	87M	Fixed	64	0.3303	100%	0.2088	57%	<u>0.1587</u>	16%
QCNeXt		Learned	6	<u>0.3341</u>		<u>0.2346</u>		0.1369	
BeTopNet	45M	Fixed	64	0.3308		0.2212		**0.1717**	
RetroMotion	24M	Learned	30	**0.3348**		**0.2636**		0.1284	

Table 2: **Per agent type comparison of the top4 methods.** All metrics are for the interactive test split of the Waymo Open Motion dataset. Params (active) gives the number of parameters that are active per agent (cf. Jiang et al. [2024]). % of scenes gives the percentage of scenes that contain at least one instance of the corresponding agent type. Best scores are **bold**, second best are <u>underlined</u>.

Our RetroMotion (SMoE hybrid) configuration in Table 1 is motivated by the fact that models with more anchors and static anchor initialization perform better for cyclists. Specifically, we build a SMoE model that uses RetroMotion-based experts for vehicles and pedestrians, along with a reproduced BeTopNet model for cyclists.

4.2 Cross-dataset generalization

We follow the evaluation protocol proposed in UniTraj [Feng et al., 2024] to measure the generalization capabilities of our method. Specifically, we configure our model to forecasts trajectories of up to 6 seconds and otherwise train the model with the same settings as in Section 4.1 and Appendix A.1. Afterwards, we evaluate this model, trained on the Waymo dataset, on the Argoverse 2 dataset.

Results: Table 3 shows the results of this experiment. For vehicle trajectories, our method outperforms MTR [Shi et al., 2022] and Wayformer [Nayakanti et al., 2023] and competes with AutoBot [Girgis et al., 2021] (lower miss rate, but higher minFDE scores). For pedestrian trajectories, our method

[5]We refer to the initial vector representation of queries used to decode trajectories as anchors.

achieves lower minFDE scores, but a higher miss rate than AutoBot. Overall, these results show that RetroMotion generalizes well and reinforce the results in Table 1.

Agent type	Method	Brier-minFDE↓	minFDE↓	MR↓
Vehicle	MTR [Shi et al., 2022]	3.63	3.14	0.44
	Wayformer [Nayakanti et al., 2023]	3.60	3.14	0.45
	AutoBot [Girgis et al., 2021]	**3.23**	**2.41**	<u>0.40</u>
	RetroMotion (ours)	<u>3.51</u>	<u>2.84</u>	**0.31**
Pedestrian	AutoBot [Girgis et al., 2021]	<u>2.22</u>	<u>1.51</u>	**0.16**
	RetroMotion (ours)	**1.97**	**1.26**	<u>0.19</u>

Table 3: **Cross-dataset generalization from Waymo Open Motion to Argoverse 2 Forecasting.** All methods are trained on Waymo Open Motion and evaluated on the validation split of the Argoverse 2 Forecasting dataset [Wilson et al., 2023]. Best scores are **bold**, second best are <u>underlined</u>.

4.3 Issuing instructions by modifying trajectories

In the following, we test our model's ability to follow goal-based instructions and to adapt basic directional instructions to the scene context. Specifically, we issue instructions by modifying predicted marginal trajectories prior to re-encoding and joint modeling (see Figure 1).

4.3.1 Goal-based instructions

In this experiment, we use the last second of the ground truth trajectories as goal-based instructions. Specifically, we replace the last second of predicted marginal trajectories with the last second of ground truth trajectories and evaluate the changes in joint trajectories ($\mathcal{P}^{\text{joint}}_{1:T}$ in Figure 1). For evaluation, we use the validation split of the Argoverse 2 Forecasting dataset and the metrics described in Appendix A.6. This evaluation is similar to the *joint-as-marginal* configuration of Ngiam et al. [2022], but with trajectory modifications as goal-based instructions. At inference, we modify either the marginal trajectory with the highest confidence score or with the lowest minFDE w.r.t. the desired goal positions, or all 6 trajectories per agent.

Results: All instruction configurations are evaluated on the validation split of the Argoverse 2 Forecasting dataset. Figure 3a presents the results. All instruction configurations improve the distance-based metrics, while modifying all trajectories leads to the most significant improvement (12% lower final displacement error). This shows that modifications to marginal trajectories affect subsequently decoded joint trajectories and can be used to issue goal-based instructions.

4.3.2 Basic directional instructions

Building on the insight that our model can follow goal-based instructions, we further evaluate basic directional instructions. We define basic directional instructions for turning left and right.

We describe both instructions with trajectories based on quarter circles. Specifically, we scale the radius r of the circle to maintain an agent's current speed. Then we use the upper right quadrant, shifted to the origin, as a *turn left* instruction, with $x(t) = r \cdot \cos(t) - r$ and $y(t) = r \cdot \sin(t)$. Similarly, we use the upper left quadrant shifted to the origin as the *turn right* instruction.

At inference, we issue the instructions by replacing the last 4 seconds in all marginal trajectories with the corresponding quarter circle, as shown qualitatively in Figure 3c.

However, such instructions are intentionally not adapted to the scene context (map and other agents). In this experiment, we evaluate the ability of RetroMotion to adapt our directional instructions to the given context. Specifically, we measure the overlap rate with other agents [Ettinger et al., 2021] of the modified trajectories used as instructions versus the subsequently decoded joint trajectories. Furthermore, we compute average on-road probability (ORP) scores, which describe the probability of trajectories staying on the road versus going off-road[6] (see Appendix A.7).

Results: Figure 3b presents the results of this experiment. Overall, higher ORP scores and lower OR scores are obtained for the *turn right* than for the *turn left* instructions. We hypothesize that

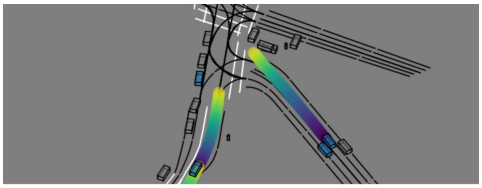
[6]Our metric is related to the drivable area compliance (DAC) metric [Chang et al., 2019].

Instruction config.	minFDE↓		minADE↓	MR↓
None	2.45		1.25	0.31
Highest confidence	2.41	-1.6%	1.24	0.31
Lowest minFDE	<u>2.28</u>	-6.9%	<u>1.19</u>	<u>0.29</u>
All trajectories	**2.16**	-11.8%	**1.15**	**0.27**

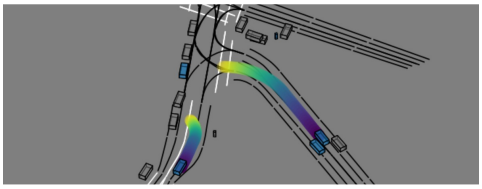
(a) **Goal-based instruction following.** As instructions, we modify marginal trajectories and evaluate the changes in joint trajectories. We modify either the trajectory with the highest confidence score or with the lowest minFDE w.r.t. the desired goal positions, or all 6 trajectories. All instruction configurations are evaluated on the validation split of the Argoverse 2 Forecasting dataset [Wilson et al., 2023]. Best scores are **bold**, second best are <u>underlined</u>.

Instruction	Eval. trajectory	OR↓		ORP↑	
turn left	basic instruction	0.23		0.64	
	adapted joint traj.	0.18	-22%	0.85	+33%
turn right	basic instruction	0.21		0.76	
	adapted joint traj.	0.18	-14%	0.91	+20%

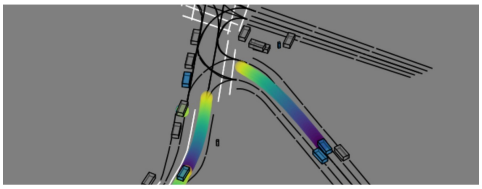
(b) **Adapting basic directional instructions to the scene context.** As instructions, we modify marginal trajectories and evaluate the changes in joint trajectories. In this experiment, we modify all 6 marginal trajectories per agent using our basic directional instructions (cf. Section 4.3.2). We report overlap rates (OR) with other agents and on-road probability (ORP) scores. All configurations are evaluated on the validation split of the Argoverse 2 Forecasting dataset.



Predicted marginal trajectories



turn left instruction as modified trajectories



Joint trajectories adapted to the scene context

(c) **Adapting a basic turn left instruction to the scene context.** The upper plot shows the default marginal trajectory forecast of our model. The middle plot shows our basic *turn left* instructions, which violate traffic rules by turning into the oncoming lanes. The lower plot shows that our model responds to this instruction by adapting the trajectory of the right vehicle to its lane (shown as black line) and reversing the instruction for the left vehicle, since turning left is not possible.

Figure 3: RetroMotion models are instructable through trajectory modifications.

this is due to the fact that the dataset mainly contains right-hand traffic scenarios, where right turns are commonly allowed and more frequent. Notably, the adjusted joint trajectories have significantly lower OR scores (22% and 14% lower) and much higher ORP scores (33% and 20% higher). This highlights the ability of our model to adapt basic directional instructions to the given scene context.

4.4 Analyzing learned trajectory representations

In order to analyze the learned representations and to perform an ablation study on our design choices, we train different configurations of our model. Specifically, we ablate modeling positional uncertainty with different probability distributions, including normal, Laplace, and exponential power distributions. We also train our model with and without compressing the location parameters of the distributions using DCT transforms.

Distribution	DCT	mAP↑	minFDE↓	minADE↓
Normal	False	0.172	2.177	0.954
Laplace	False	0.176	2.149	0.940
Laplace	True	<u>0.194</u>	<u>2.102</u>	<u>0.917</u>
Exponential power	True	**0.195**	**2.060**	**0.910**

Table 4: **Comparing distribution types with and without DCT compression.** All configurations are evaluated on the interactive validation split of the Waymo Open Motion dataset. Best scores are **bold**, second best are <u>underlined</u>.

We train each model configuration for 14 epochs on the Waymo Open Motion dataset and keep the remaining configurations as in Section 4.1 and Appendix A.1. Table 4 presents the results of this experiment. Using Laplace distributions to model positional uncertainty improves motion prediction metrics over using normal distributions. Compressing the location parameters of densities further improves the results significantly. Overall, using exponential power distributions with DCT compression leads to the best results across all metrics.

Weights in exponential power distributions: During training, we track the value of the mixture weights of normal components in exponential power distributions (see Figure 4). Initially, the weight is close to 0, because the negative loglikelihood (NLL) of a normal distribution is closely related to the mean squared error, while the NLL of a Laplace distribution is related to the mean absolute error.

Therefore, the NLL of a normal distribution is much higher for outliers (i.e., unreasonable predictions during the earlier training epochs), which is compensated by a low w value. During training, w progressively increases, while reaching higher values for predicted joint trajectory distributions than for marginal distributions. However, on average, the learned exponential power distributions tend to be Laplace-like with normal components with relatively low weights of 0.1 (joint) and 0.04 (marginal).

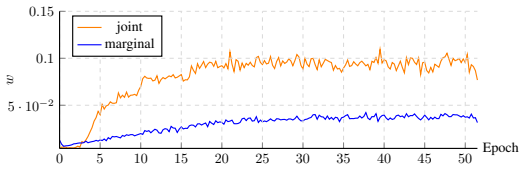


Figure 4: **Mixture weight of normal components in exponential power distributions.** During training the weight w progressively increases, while reaching higher values for joint trajectory distributions than for marginal distributions.

Feature vector collapse: We measure the feature vectors collapse metric (NRC1) [Andriopoulos et al., 2025] for feature vectors of marginal and joint trajectory distributions. The feature vectors are the last hidden states of trajectories generated by the MLP heads in our model. We compute the metric assuming either 272 or 32 principal components since there are 272 density parameters per trajectory including location parameters compressed as 32 DCT coefficients. Results in Figure 5 show an immediate collapse when a 272-dimensional subspace is assumed (left plot) and no collapse when only considering the first 32 principal components. This observation strongly suggest that the true dimensionality of features vectors lies between 32 and 272, even when considering correlation between output variables, and reinforces the results Table 4 and Figure 4. These findings support that, in addition to the DCT coefficients for location parameters, other density parameters (i.e., shape and scale) are important when regressing trajectories.

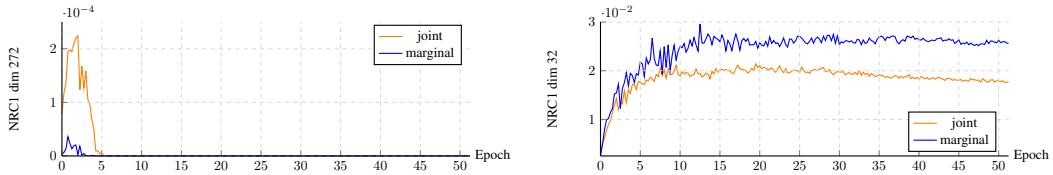


Figure 5: **Feature vectors collapse (NRC1) metrics.** The left plot shows that feature vectors collapses to a subspace spanned by less than 272 principal components. NRC1 does not decrease in the right plot indicating that the feature vectors span more than 32 dimensions.

5 Conclusion

In this work, we decompose the motion forecasting task into modeling marginal trajectory distributions for all modeled agents and joint distributions for interacting agents. Our transformer-based forecasting model incorporates a retrocausal information flow and models positional uncertainty through compressed exponential power distributions. This enables more accurate predictions across diverse scenarios in the Waymo Interaction Prediction and Argoverse 2 Forecasting datasets. Furthermore, our method's ability to respond to goal-based and basic directional instructions reveals an emergent capability that was not explicitly trained for. This suggests that standard motion forecasting training naturally develops representations that can interpret and adapt to goal-based and directional instructions within the appropriate context. This capability can improve simulation and human-AI interaction in self-driving systems, potentially allowing operators to guide vehicle behavior.

Limitations and future work: While the retrocausal information flow enables us to instruct our model, it can also lead to less realistic forecasts (cf. acausal reactions [Seff et al., 2023]). E.g., instructing a following vehicle to slow down may also slow down a leading vehicle. This may be caused by the data-driven nature of our method and many training samples where the behavior of leading and following vehicles is correlated. However, following vehicles usually react to leading vehicles, not vice versa. We leave investigating that further and mitigating such issues (e.g., by auto-regressive trajectory generation) to future research.

Broader impacts

Once deployed in self-driving cars, motion forecasting models can either improve or worsen subsequent motion planning. This can result in safer or riskier behavior on the road. Therefore, it is crucial to evaluate such models beyond datasets before deploying them.

References

G. Andriopoulos, Z. Dong, L. Guo, Z. Zhao, and K. Ross. The prevalence of neural collapse in neural multivariate regression. *Advances in Neural Information Processing Systems*, 37:126417–126451, 2025.

O. Azencot, N. B. Erichson, V. Lin, and M. Mahoney. Forecasting sequential data using consistent koopman autoencoders. In *International Conference on Machine Learning*. PMLR, 2020.

I. Bae, J. Oh, and H.-G. Jeon. Eigentrajectory: Low-rank descriptors for multi-modal trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

C. M. Bishop. Mixture density networks. 1994.

M.-K. Bouzidi, B. Derajic, D. Goehring, and J. Reichardt. Motion planning under uncertainty: Integrating learning-based multi-modal predictors into branch model predictive control. *arXiv preprint arXiv:2405.03470*, 2024.

S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 624–641. Springer, 2020.

Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conference on Robot Learning*. PMLR, 2020.

M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8748–8757, 2019.

S. Chen, P.-L. Guhur, C. Schmid, and I. Laptev. History aware multimodal transformer for vision-and-language navigation. *Advances in neural information processing systems*, 2021.

J. Cheng, X. Mei, and M. Liu. Forecast-mae: Self-supervised pre-training for motion forecasting with masked autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

A. Cui, S. Casas, K. Wong, S. Suo, and R. Urtasun. Gorela: Go relative for viewpoint-invariant motion forecasting. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

W. Fedus, J. Dean, and B. Zoph. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*, 2022.

L. Feng, M. Bahari, K. M. B. Amor, É. Zablocki, M. Cord, and A. Alahi. Unitraj: A unified framework for scalable vehicle trajectory prediction. In *European Conference on Computer Vision*, pages 106–123. Springer, 2024.

J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

M. Geisslinger, F. Poszler, and M. Lienkamp. An ethical trajectory planning algorithm for autonomous vehicles. *Nature Machine Intelligence*, 2023.

T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde. Thomas: Trajectory heatmap output with learned multi-agent sampling. In *International Conference on Learning Representations*, 2022.

R. Girgis, F. Golemo, F. Codevilla, M. Weiss, J. A. D'Souza, S. E. Kahou, F. Heide, and C. Pal. Latent variable sequential set transformers for joint multi-agent motion prediction. *arXiv preprint arXiv:2104.00563*, 2021.

P.-L. Guhur, S. Chen, R. G. Pinel, M. Tapaswi, I. Laptev, and C. Schmid. Instruction-driven history-aware policies for robotic manipulations. In *Conference on Robot Learning*, pages 175–187. PMLR, 2023.

A. Guzman-Rivera, D. Batra, and P. Kohli. Multiple choice learning: Learning to produce multiple structured outputs. *Advances in neural information processing systems*, 2012.

Z. Huang, H. Liu, and C. Lv. Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3903–3913, 2023.

R. Hug, W. Hübner, and M. Arens. Introducing probabilistic bézier curves for n-step sequence prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

C. Jiang, A. Cornman, C. Park, B. Sapp, Y. Zhou, D. Anguelov, et al. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, et al. Openvla: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*, 2024.

S. Konev. Mpa: Multipath++ based architecture for motion prediction. *arXiv preprint arXiv:2206.10041*, 2022.

Z. Lan, Y. Jiang, Y. Mu, C. Chen, and S. E. Li. SEPT: Towards efficient scene representation learning for motion prediction. In *The Twelfth International Conference on Learning Representations*, 2024.

S. Lee, S. Purushwalkam Shiva Prakash, M. Cogswell, V. Ranjan, D. Crandall, and D. Batra. Stochastic multiple choice learning for training diverse deep ensembles. *Advances in Neural Information Processing Systems*, 2016.

H. Liu, L. Chen, Y. Qiao, C. Lv, and H. Li. Reasoning multi-agent behavioral topology for interactive autonomous driving. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019.

W. Luo, C. Park, A. Cornman, B. Sapp, and D. Anguelov. Jfp: Joint future prediction with interactive multi-agent modeling for autonomous driving. In *Conference on Robot Learning*, 2022.

W. Mao, M. Liu, M. Salzmann, and H. Li. Learning trajectory dependencies for human motion prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9489–9497, 2019.

N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

J. Ngiam, V. Vasudevan, B. Caine, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, et al. Scene transformer: A unified architecture for predicting future trajectories of multiple agents. In *International Conference on Learning Representations*, 2022.

A. Pashevich, C. Schmid, and C. Sun. Episodic transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

M. A. Raad, A. Ahuja, C. Barros, F. Besse, A. Bolt, A. Bolton, B. Brownfield, G. Buttimore, M. Cant, S. Chakera, et al. Scaling instructable agents across many simulated worlds. *arXiv preprint arXiv:2404.10179*, 2024.

L. Rowe, M. Ethier, E.-H. Dykhne, and K. Czarnecki. Fjmp: Factorized joint multi-agent motion prediction over learned directed acyclic interaction graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13745–13755, 2023.

G. Sarch, S. Somani, R. Kapoor, M. J. Tarr, and K. Fragkiadaki. Helper-x: A unified instructable embodied agent to tackle four interactive vision-language domains with memory-augmented language models. *arXiv preprint arXiv:2404.19065*, 2024.

A. Seff, B. Cera, D. Chen, M. Ng, A. Zhou, N. Nayakanti, K. S. Refaat, R. Al-Rfou, and B. Sapp. Motionlm: Multi-agent motion forecasting as language modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

S. Shi, L. Jiang, D. Dai, and B. Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 2022.

S. Shi, L. Jiang, D. Dai, and B. Schiele. Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Q. Sun, X. Huang, J. Gu, B. C. Williams, and H. Zhao. M2i: From factored marginal trajectory prediction to interactive prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Z. Sun, Y. Shen, H. Zhang, Q. Zhou, Z. Chen, D. D. Cox, Y. Yang, and C. Gan. Salmon: Self-alignment with instructable reward models. In *ICLR*, 2024.

O. S. Tas, P. H. Brusius, and C. Stiller. Decision-theoretic mpc: Motion planning with weighted maneuver preferences under uncertainty. *arXiv preprint arXiv:2310.17963*, 2023.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

R. Wagner, O. S. Tas, M. Klemp, C. Fernandez, and C. Stiller. Redmotion: Motion prediction via redundancy reduction. *Transactions on Machine Learning Research*, 2024a.

R. Wagner, O. S. Tas, M. Klemp, and C. Fernandez Lopez. Jointmotion: Joint self-supervision for joint motion prediction. In *8th Annual Conference on Robot Learning (CoRL)*, 2024b.

B. Warner, A. Chaffin, B. Clavié, O. Weller, O. Hallström, S. Taghadouini, A. Gallagher, R. Biswas, F. Ladhak, T. Aarsen, et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*, 2024.

B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. *arXiv preprint arXiv:2301.00493*, 2023.

Z. Zhang, A. Liniger, C. Sakaridis, F. Yu, and L. V. Gool. Real-time motion prediction via heterogeneous polyline transformer with relative pose encoding. *Advances in Neural Information Processing Systems*, 2024.

Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang. Query-centric trajectory prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023a.

Z. Zhou, Z. Wen, J. Wang, Y.-H. Li, and Y.-K. Huang. Qcnext: A next-generation framework for joint multi-agent trajectory prediction. *arXiv preprint arXiv:2306.10508*, 2023b.

H.-G. Zimmermann, C. Tietz, and R. Grothmann. Forecasting with recurrent neural networks: 12 tricks. *Neural Networks: Tricks of the Trade: Second Edition*, pages 687–707, 2012.

A Appendix

A.1 Training details

We sample 32 scenes from the Waymo Open Motion dataset in a batch, with 8 focal (i.e., predicted) agents per scene. Specifically, we predict marginal trajectory distributions for all 8 agents and joint distributions for two interactive agents. We use Adam with weight decay [Loshchilov and Hutter, 2019] as the optimizer and a step learning rate scheduler to halve the initial learning rate of 2^{-4} every 10 epochs. We train for 50 epochs using data distributed parallel (DDP) training on 4 Ada A6000 GPUs.

A.2 Expert model configuration and routing

Our sparse mixture of experts model contains 3 expert models adapted to the 3 evaluated agent types, vehicles, pedestrians, and cyclists. Specifically, we follow the evaluation protocol of Ettinger et al. [2021] and use one expert for joint trajectory forecasts of only vehicles, one for forecasts that involve vehicles and pedestrians, and one for forecasts of vehicles and cyclists. The expert model for vehicles is trained to forecast 18 trajectories per agent, while the other experts to forecast 6 trajectories per agent. For the cyclist expert, we increase the loss weight of cyclist trajectories $10\times$.

A.3 Post-processing

As post-processing, we follow Konev [2022] and reduce the confidence scores of redundant trajectories. We adapt this mechanism to joint trajectory sets by suppressing confidence scores only if a nearby trajectory belongs to a joint trajectory set with a higher accumulated confidence score (see c in Section 3.3). For the vehicle expert, we additionally perform topk-selection to reduce the predicted 18 trajectories to 6 trajectories. We determine the suppression thresholds per expert and agent class using the training split. Furthermore, we tune the softmax temperature used to compute the accumulated confidence scores. The distance thresholds and temperatures for the results in Table 1 are:

- veh expert: veh thresh = 1.3 meters, $\tau = 0.5$
- ped expert: veh thresh = 1.4 meters, ped thresh = 1.6 meters, $\tau = 1.1$
- cyc expert (RetroMotion-based): veh thresh = 1.9 meters, cyc thresh = 1.0 meters, $\tau = 1.0$
- cyc expert (BeTopNet-based): veh thresh = 2.5 meters, cyc thresh = 2.5 meters

For the BeTopNet-based cyc expert, we use the default NMS of Shi et al. [2022].

A.4 Dataset details

We use the Waymo Open Motion dataset to evaluate the joint motion forecasts of our model. Specifically, we use the interactive test and validation splits to benchmark our model. The dataset contains diverse traffic scenarios of urban and suburban driving and annotations of interacting road users. All samples consist of 1 second of past agent states (i.e., position, velocity, acceleration, and bounding boxes) and map features such as road markings and traffic light states. The prediction targets are future trajectories of up to 8 seconds.

A.5 Motion forecasting metrics

The official challenge metrics are the mean average precision (mAP), the average displacement error (minADE), and the final displacement error (minFDE), the miss rate (MR), and the overlap rate (OR). All metrics are computed using the joint trajectory set closest to the ground truth trajectories. Furthermore, the metrics are averaged over the three prediction horizons of 3 s, 5 s, and 8 s, and the 3 agent classes (vehicles, pedestrians, and cyclists).

A.6 Cross-dataset evaluation metrics

Since UniTraj focuses on marginal forecasting per agent type, we evaluate the marginal predictions of our model. UniTraj uses the same distance-based metrics described in Appendix A.5, but evaluated

on an agent-by-agent (i.e., marginal) basis. Additionally, we report the Brier-minFDE = minFDE + $(1-p)^2$, where p is the predicted confidence score per trajectory.

A.7 On-road probability maps

We compute on-road probability maps to describe the probability of trajectories staying on the road versus going off-road. Specifically, we use a rasterized representation of drivable areas and a distance transform function to compute the on-road probability maps. The distance transform calculates, for each pixel representing a non-drivable area, the distance to the closest pixel that represents a drivable area in the raster mask. Essentially, it quantifies how far away each non-drivable pixel is from the nearest drivable region. Therefore, this metric also measures how far trajectories leave the drivable area. Figure 6 shows an on-road probability map of a scenario in the Argoverse 2 Forecasting dataset. We first compute the on-road probability score of each trajectory point. Then, we set a trajectory to off-road if the probability of any trajectory point is < 0.99, and to on-road otherwise. Finally, we compute the average on-road probability (ORP) score as the percentage of on-road trajectories over all trajectories.

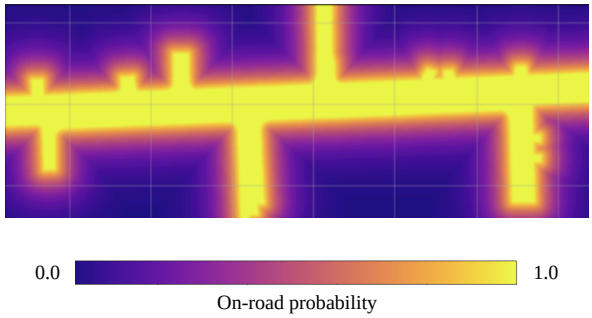


Figure 6: On-road probability map of a scenario in the Argoverse 2 Forecasting dataset.