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. Additionally, our method provides an interface for issuing instructions through trajectory modifications. Our experiments show that regular training of motion forecasting without instructions leads to the ability to adapt basic directional instructions to the scene context. Code: https://github.com/kit-mrt/future-motion

1. 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 three ways.

1.1. Decomposing exponential power distributions

We model the positional uncertainty per trajectory point as density of an exponential power distribution. 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

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

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 [12, 19, 20], we model the x- and y-coordinates as uncorrelated random variables. In the following, we include w in the tuple of density parameters $\boldsymbol{\phi}$.

1.2. Decomposing marginal trajectory distributions

We train a distinct decoder to perform marginal motion forecasting (i.e., per-agent). Following common practice [2, 12, 18], we predict the density of a mixture distribution at each future time step 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 [1] and express this as

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

where $\boldsymbol{x}$ is the input (Sec. 1.5), $\boldsymbol{y}$ the target vector, $t \in \{1, ..., T\}$ 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.

1.3. 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. Sec. 1.5). Afterwards, we exchange information via attention mechanisms and decode joint trajectory distributions using multi-agent mixture components (see Fig. 1). Per time step t, each mixture component is a mixture density

itself, describing one future position of each modeled agent

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

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 Fig. 1, this approximates the joint distribution of all combinations of per-agent mixture components by focusing on the diagonal query pairs in matrix form. Off-diagonal query pairs can update diagonal pairs through attention mechanisms, which compresses information from all K^2 possible combinations into K query pairs.

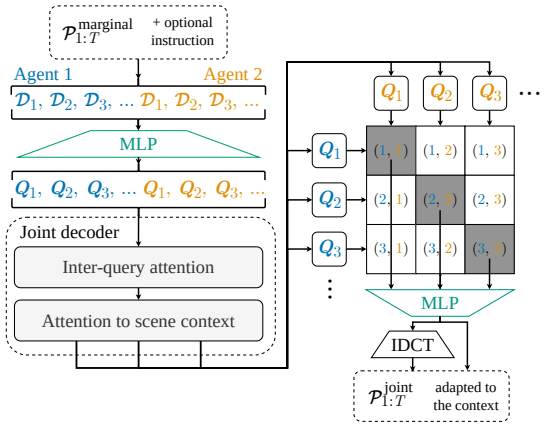

Figure 1. **From marginal to joint trajectories.** We use an MLP to generate query matrices $\boldsymbol{Q}$ from marginal trajectories and exchange information between queries and scene context with attention mechanisms. Afterwards, we decode joint trajectories $\mathcal{P}_{1:T}^{\text{joint}}$ from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

For both marginal and joint motion forecasts, we follow related methods [2, 14, 19] and use the out-most mixture weights (m_k and c_k) as confidence scores.

1.4. Compressing location parameters of densities

Most motion forecasting methods regress trajectories at a frequency of 10 Hz [17–19], 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 (Fig. 1). We hypothesize that this is a natural choice for transformer models given the use of sinusoidal positional encodings [16]. 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).

1.5. Scene encoder

We follow Gao et al. [5] 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 [16]. Following Nayakanti et al. [12], 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. [7]). Finally, we add global sinosoidal positional encodings and generate global scene context representations with further self-attention mechanisms. Our two decoders (Sec. 1.2 and Sec. 1.3) decode probabilistic motion forecasts from this scene context.

1.6. Loss function

We train our model using maximum likelihood estimation with a multi-task loss that covers the objectives described in Sec. 1.2 and Sec. 1.3. 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. We measure the distance to the ground truth using the ℓ_2-norm. For marginal forecasts $\mathcal{P}_{1:T}^{\text{marginal}}$, we select the best trajectory index $\hat{k}$ per agent. For joint forecasts $\mathcal{P}_{1:T}^{\text{joint}}$, we select the best set of trajectories at the same index $\hat{k}$ for agent pairs (cf. Fig. 1).

2. Experiments

In this section we evaluate the motion forecasting performance of our method using the Waymo Open Motion [3] dataset. Afterwards, we show that regular training of motion forecasting leads to the ability to adapt basic directional

Method	Params (active)	Anchor init.	Anchors	Vehicle		Pedestrian		Cyclist	
				mAP	% of scenes	mAP	% of scenes	mAP	% of scenes
MTR++ [15]	87M	Fixed	64	0.3303	100%	0.2088	57%	<u>0.1587</u>	16%
QCNeXt [20]		Learned	6	<u>0.3341</u>		<u>0.2346</u>		0.1369	
BeTopNet [9]	45M	Fixed	64	0.3308		0.2212		**0.1717**	
RetroMotion (SMoE)	24M	Learned	30	**0.3348**		**0.2636**		0.1284	
RetroMotion (SMoE hybrid)	24 or 45M	Mixed	94	**0.3348**		**0.2636**		0.1572	

Table 1. **Comparison of state-of-the-art methods for motion forecasting.** 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. [6]). % 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>.

instructions to the scene context. Furthermore, we compare distribution types for modeling positional uncertainty.

2.1. 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 past 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) [4] of 3 variations of our model. 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 $10\times$. All expert models are trained independently. At inference, we use a rule-based router that selects one model based on the agent type and adapt Konev [8]'s NMS to only suppress trajectories associated with lower multi-agent mixture weights $\boldsymbol{c}$ (Eq. (3)).

Training details: We sample 32 scenes 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 set $\lambda^{\text{marginal}} = 0.5$, use AdamW [10] 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.

Results: Tab. 1 presents motion forecasting metrics for interactive forecasting on the Waymo Open Motion dataset. Fully data-driven methods with learned anchor[1] 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 [11]. On average, our method outperforms related methods that require more active parameters.

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

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

2.2. Issuing instructions by modifying trajectories

In the following, we test our model's ability 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 (Fig. 1).

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 $x(t) = r \cdot \cos(t) - r, y(t) = r \cdot \sin(t)$. Similarly, we use the upper left quadrant 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 Fig. 2.

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 [3] 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. 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.

Results: Tab. 2 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.

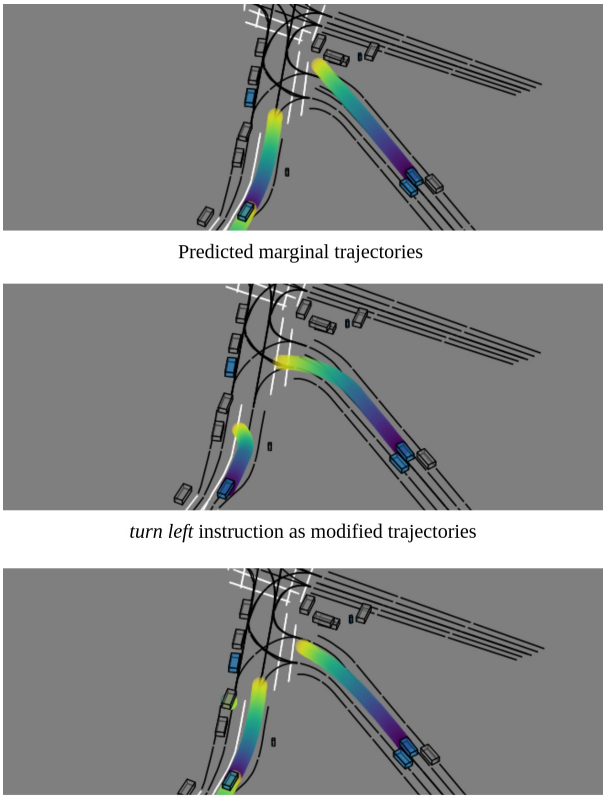

Figure 2. **Adapting a basic turn left instruction to the scene context.** The upper plot shows the default marginal trajectory forecast of our model. The middle plot shows our basic *turn left* instructions, which violate traffic rules by turning into the oncoming lanes. The lower plot shows that our 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.

We hypothesize that 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 and much higher ORP scores. This highlights the ability of our model to adapt basic directional instructions to the given scene context.

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%

Table 2. **Adapting basic directional instructions to the scene context.** As instructions, we modify marginal trajectories and evaluate the changes in joint trajectories. We report overlap rates (OR) with other agents and on-road probability (ORP) scores.

2.3. Comparing distribution types

In order 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 location parameters using DCT transforms.

We train each model configuration for 14 epochs on the Waymo Open Motion dataset and keep the remaining configurations as in Sec. 2.1. Tab. 3 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. Overall, using exponential power distributions with DCT compression leads to the best results.

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 3. **Comparing distribution types with and without DCT compression.** All configurations are evaluated on the interactive validation split of the Waymo Open Motion dataset.

3. 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 lowers the modeling burden on initial marginal forecasts and enables more accurate predictions across diverse scenarios. Furthermore, our method's ability to respond basic directional instructions reveals an emergent capability that was not explicitly trained for. 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 [13]). 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.

References

[1] Christopher M Bishop. Mixture density networks. 1994. 1

[2] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conference on Robot Learning*. PMLR, 2020. 1, 2

[3] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin 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. 2, 3

[4] William Fedus, Jeff Dean, and Barret Zoph. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*, 2022. 3

[5] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia 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. 2

[6] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024. 3

[7] Chiyu Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, Dragomir 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. 2

[8] Stepan Konev. Mpa: Multipath++ based architecture for motion prediction. *arXiv preprint arXiv:2206.10041*, 2022. 3

[9] Haochen Liu, Li Chen, Yu Qiao, Chen Lv, and Hongyang Li. Reasoning multi-agent behavioral topology for interactive autonomous driving. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 3

[10] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 3

[11] Wenjie Luo, Cheol Park, Andre Cornman, Benjamin Sapp, and Dragomir Anguelov. Jfp: Joint future prediction with interactive multi-agent modeling for autonomous driving. In *Conference on Robot Learning*, 2023. 3

[12] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023. 1, 2

[13] Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S Refaat, Rami Al-Rfou, and Benjamin Sapp. Motionlm: Multi-agent motion forecasting as language modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 4

[14] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 2022. 2

[15] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 3

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[17] Royden Wagner, Omer Sahin Tas, Marvin Klemp, Carlos Fernandez, and Christoph Stiller. Redmotion: Motion prediction via redundancy reduction. *Transactions on Machine Learning Research*, 2024. 2

[18] Zhejun Zhang, Alexander Liniger, Christos Sakaridis, Fisher Yu, and Luc V Gool. Real-time motion prediction via heterogeneous polyline transformer with relative pose encoding. *Advances in Neural Information Processing Systems*, 2024. 1

[19] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 1, 2

[20] Zikang Zhou, Zihao Wen, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Qcnext: A next-generation framework for joint multi-agent trajectory prediction. *arXiv preprint arXiv:2306.10508*, 2023. 1, 3