RESET: Revisiting Trajectory Sets for Conditional Behavior Prediction

Julian Schmidt*†, Pascal Huissel*, Julian Wiederer*‡, Julian Jordan*, Vasileios Belagiannis‡ and Klaus Dietmayer†
*Mercedes-Benz AG, Research & Development, Stuttgart, Germany
Email: julian.sj.schmidt@mercedes-benz.com
†Ulm University, Institute of Measurement, Control and Microtechnology, Ulm, Germany
‡Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany

***Abstract*—It is desirable to predict the behavior of traffic participants conditioned on different planned trajectories of the autonomous vehicle. This allows the downstream planner to estimate the impact of its decisions. Recent approaches for conditional behavior prediction rely on a regression decoder, meaning that coordinates or polynomial coefficients are regressed. In this work we revisit set-based trajectory prediction, where the probability of each trajectory in a predefined trajectory set is determined by a classification model, and first-time employ it to the task of conditional behavior prediction. We propose RESET, which combines a new metric-driven algorithm for trajectory set generation with a graph-based encoder. For unconditional prediction, RESET achieves comparable performance to a regression-based approach. Due to the nature of set-based approaches, it has the advantageous property of being able to predict a flexible number of trajectories without influencing runtime or complexity. For conditional prediction, RESET achieves reasonable results with late fusion of the planned trajectory, which was not observed for regression-based approaches before. This means that RESET is computationally lightweight to combine with a planner that proposes multiple future plans of the autonomous vehicle, as large parts of the forward pass can be reused.**

***Index Terms*—Behavior-Based Systems, Motion and Path Planning, AI-Based Methods**

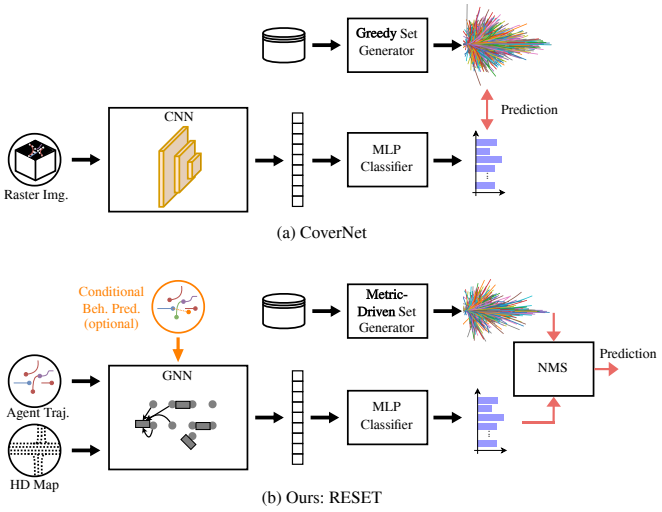


Fig. 1. Comparison of (a) CoverNet and (b) our method RESET: We use a new metric-driven algorithm for set generation, a graph-based input representation and a Non Maximum Suppression post-processing strategy. The model is also applicable to the task of conditional behavior prediction.

I. Introduction

Safe trajectory planning in complex traffic scenarios requires the autonomous vehicle to predict the future motion of surrounding traffic participants. Lately, machine learning-based prediction models, e.g., [1]–[4], have shown promising results, as they are able to efficiently leverage information provided by perception and fusion stacks in combination with information from a High Definition (HD) map. For highly interactive scenarios, it is desired to additionally condition the predictions on the planned future trajectory of the autonomous vehicle. In this way, a downstream planner can estimate the impact of its planned trajectories on surrounding agents, as predictions resulting from multiple planned trajectories can be compared to each other.

In order to make conditional behavior prediction computationally lightweight during planning, it is advantageous to inject the future trajectory of the autonomous vehicle in later stages of the model. Thereby, during the forward pass, the calculations up to the fusion point can be reused for multiple possible planned futures. Prior publications [5], [6] that compare early fusion (fusion during encoding) and late fusion (fusion prior to decoding) with regression decoders share one observation: Early fusion always outperforms late fusion by a margin, with late fusion even leading to worse predictions than without conditioning.

In this paper we do not follow recent approaches, e.g., [1], [4], [7], that typically predict coordinates or polynomial coefficients during a regression step. Instead, we revisit set-based trajectory prediction and employ it to the task of conditional behavior prediction. Set-based trajectory prediction, originally proposed in CoverNet [8], reformulates the task of trajectory prediction as a classification problem. Rather than regressing trajectories, the probability of each trajectory in a trajectory set is predicted. Without retraining and influencing the runtime or complexity, this allows, up to the size of the used set, a flexible amount of trajectories with a high probability being used as predictions. On benchmarks, CoverNet fails to achieve competitive results with respect to other recent approaches.

The research leading to these results is funded by the BMWK within the project "KI Delta Learning" (Förderkennzeichen 19A19013A).

Therefore, this paper is two-fold: Firstly, we propose a set-based trajectory prediction method that achieves similar performance to regression-based approaches. We name this method *RESET* (REvisiting SET-based trajectory prediction). RESET consists of a new metric-driven algorithm for trajectory set generation, a graph-based encoder and a Non Maximum Suppression (NMS) post-processing strategy. Fig. 1 highlights the differences to CoverNet. Secondly, we show that our method is employable to the task of conditional behavior prediction and evaluate different variants of how to inject the conditioning information into the model. While late fusion fails for regression decoders in prior work, our set-based method achieves reasonable results with late fusion.

In summary, our main contributions are:

- We revisit set-based trajectory prediction and propose a method that achieves similar performance to a regression-based model.
- We compare different injection points for conditioning the set-based prediction on the planned future trajectory of the autonomous vehicle.
- We extensively evaluate our set-based method for unconditional and conditional prediction on the large-scale Argoverse Motion Forecasting Dataset 2 [9] and show that it obtains reasonable results with late fusion.

II. Related Work

This section discusses the related work in trajectory prediction, with a special focus on goal- and set-based approaches. Additionally, we refer to prior work related to conditional behavior prediction.

A. Trajectory Prediction

Trajectory prediction is a constantly evolving research topic with a myriad of different approaches. One group of approaches directly regresses trajectories via a regression decoder [1], [2], [10], [11]. Adding an additional classification decoder allows determining the probability per predicted trajectory [1], [10]. Distribution-based approaches, on the other hand, are used to regress trajectories from a latent distribution, most commonly realized with Conditional Variational Autoencoders (CVAEs) [7], [12]–[14] or Generative Adversarial Networks (GANs) [15]. In contrast to direct regression-based approaches, when using CVAEs with a continuous latent space, it is also possible to predict agents with an arbitrary amount of modes, without having to predefine this amount prior to training [7]. This is also a strength of goal- and set-based approaches, which are discussed in-depth below.

Goal-based approaches first sample goal candidates and then score these over-sampled candidates in order to obtain endpoint predictions [3], [4], [16]. Goal candidates can either be sparsely predefined anchors [4], [16] or densely sampled [3]. To obtain trajectory predictions, an additional regression step is required. This regression step is commonly performed in a learning-based manner and uses the top-scored goal candidates.

Set-based approaches do not share the similarity of all above mentioned approaches, which is the dependency on a regression decoder. Instead, they initially generate a trajectory set containing a variety of possible trajectories and then use a classifier to determine the most likely trajectories of the generated set. PRIME [17] relies on a model-based planner to generate a trajectory set for the agent to be predicted. The generated set is based on the current state of the agent and the HD map. A subsequent learning-based classification network is then used to determine which trajectories the agent will most likely drive. This results in a two-stage approach.

CoverNet [8] uses a rasterized representation of agents and map and proposes two variants of trajectory sets: The dynamic and the fixed set. The dynamic set is generated by the rollout of a kinematic model (e.g., kinematic bicycle model) using diverse control actions. This kinematic model is initialized with the current state of the agent to be predicted. Similar to PRIME, this results in a two-stage approach, meaning that an initial model-based set generation is required prior to the learning-based classification. The fixed set, on the other hand, is the same for each agent. It is already generated from the training split before the training process and inducts prior knowledge about the dataset into the model. For this set generation, a greedy bagging algorithm is used to determine the minimal set of trajectories in a dataset covering all other trajectories. Whether a trajectory covers another trajectory is determined by checking whether the maximum point-wise ℓ^2 distance is below an acceptable error tolerance. Due to the fixed set being generated from the training dataset prior to the training process, this results in a one-stage approach.

Our approach is also one-stage, making use of fixed trajectory sets. In contrast to the rasterization-based encoding of agents and map and the greedy bagging algorithm for trajectory set generation of CoverNet, we combine a new metric-driven set generation algorithm, a graph-based encoder and an NMS post-processing strategy.

B. Conditional Behavior Prediction

Injecting additional information about the future of one or multiple agents into the prediction model is called conditioning. Most commonly, predictions are conditioned on the planned future trajectory of the autonomous vehicle. Conditional behavior prediction has been examined in the context of direct regression-based models [6], [18], [19] and distribution-based models [20], [21]. However, it has never been investigated, whether goal-based and set-based prediction approaches are also capable of using the additional information that is injected during the conditioning process.

In this work we cover this research gap for set-based approaches and investigate different ways of employing our set-based method to the task of conditional behavior prediction.

III. Method

This section describes our method RESET, consisting of the trajectory set generation, the encoder-decoder classification model and an NMS post-processing strategy. An overview

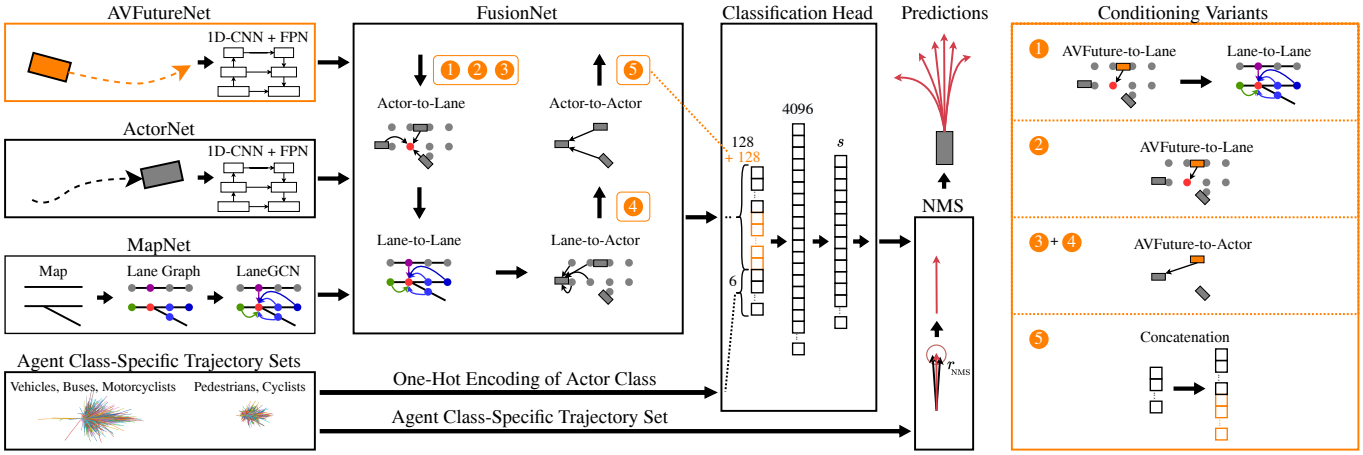


Fig. 2. Overview of RESET: Information of past agent motion (ActorNet) and the HD map (MapNet) are fused in four fusion stages (FusionNet). Subsequently, the classification layer determines a probability for each trajectory in the trajectory set. Combining these probabilities with the trajectory set leads to trajectory predictions, which are post-processed with a Non Maximum Suppression. Orange blocks indicate different variants for conditioning the predictions on the planned future trajectory of the autonomous vehicle (encoded by the AVFutureNet). Five variants, ranging from early to late fusion, are examined.

is given in Fig. 2. Note that this overview already includes different variants for conditioning the predictions on the future trajectory of the autonomous vehicle, colored in orange. Details are given in Section IV.

In general, trajectory prediction is defined as the task of predicting the future trajectory $\mathbf{T}_{\text{gt}} = \{\boldsymbol{\tau}^t\}_{t=1}^{T_f}$ of an agent. T_f defines the prediction horizon and $\boldsymbol{\tau}^t = (x^t, y^t)$ the position of an agent at timestep t in a 2D Bird's Eye View (BEV) coordinate system.

A. Trajectory Set Generation

We propose to use a metric-driven algorithm for trajectory set generation and to use different trajectory sets for different agent classes. During the training and inference, these trajectory sets are used as the classification target by the decoder. Prior to trajectory set generation, all considered trajectories are transformed into a common local coordinate system. Only future timesteps $t = 1, 2, \ldots, T_F$ are considered.

1) Algorithm: Our proposed algorithm, shown in Algorithm 1, is iterative and driven by a selected prediction metric. In our case, we use the minimum Average Displacement Error (minADE). Other metrics, such as minFDE or MR are also applicable. More details on these metrics are given in Section V-B.

The input to the algorithm are the dataset $\mathcal{D}$ and the desired final set size s. After the defined number of iterations s, the algorithm outputs the trajectory set $\mathcal{S}$.

During an initialization phase, the ADE value of each trajectory pair in the dataset $\mathcal{D}$ is calculated and stored in $\mathbf{M}_{\text{full}}$ (Line 9). $\mathbf{M}_{\text{set}}$ is initialized with arbitrary high values (Line 8). During the iterative set generation process, $\mathbf{M}_{\text{set}}$ will store the minADE for each trajectory in the dataset $\mathcal{D}$ with regard to the current trajectory set $\mathcal{S}$.

In each iteration (defined by the loop in Line 10), one trajectory $\mathbf{T}_i$ is added to the trajectory set $\mathcal{S}$. The trajectory $\mathbf{T}_i$ is selected in an optimal way with regard to the metric, in our case minADE. This is ensured by the following procedure: Each trajectory $\mathbf{T}_i$ in the dataset $\mathcal{D}$ is temporarily appended to the trajectory set (Line 14). $\mathbf{M}_i$ is calculated to store the minADE of each trajectory in the dataset $\mathcal{D}$ with regard to the current temporary trajectory set (Line 15). Averaging $\mathbf{M}_i$ results in one scalar m_i that corresponds to the average minADE that is optimally achievable with the current temporary trajectory set on the dataset $\mathcal{D}$ (Line 16). In each iteration, the

Algorithm 1 Metric-driven trajectory set generation

1: **Input**
2: $\mathcal{D}$ Dataset containing trajectories $\mathbf{T}_i$, $|\mathcal{D}| = k$
3: s Final set size
4: **Output**
5: $\mathcal{S}$ Trajectory set, $|\mathcal{S}| = s$
6: **procedure** GET_TRAJ_SET($\mathcal{D}$, s)
7: $\mathcal{S} \leftarrow \{\}$
8: $\mathbf{M}_{\text{set}} \leftarrow [\text{inf}, \text{inf}, ..., \text{inf}] \in \mathcal{R}^k$
9: $\mathbf{M}_{\text{full}} \in \mathcal{R}^{k \times k}$, where $\mathbf{M}_{\text{full}}[i, j] \leftarrow$ GET_ADE($\mathbf{T}_i, \mathbf{T}_j$)
10: **while** $|\mathcal{S}| < s$ **do**
11: Initialize empty metric memory set $\mathcal{M}_{\text{memory}} \leftarrow \{\}$
12: $i \leftarrow 0$
13: **while** $i < k$ **do**
14: $\mathbf{M}_{i,\text{combined}} \leftarrow [\mathbf{M}_{\text{full}}[i], \mathbf{M}_{\text{set}}] \in \mathcal{R}^{k \times 2}$
15: $\mathbf{M}_i \leftarrow \min_{\text{axis}=1} \mathbf{M}_{i,\text{combined}} \in \mathcal{R}^k$
16: $m_i \leftarrow \text{mean}(\mathbf{M}_i) \in \mathcal{R}$
17: $\mathcal{M}_{\text{memory}} \leftarrow \mathcal{M}_{\text{memory}} \cup \{(i, m_i, \mathbf{M}_i)\}$
18: $i \leftarrow i + 1$
19: **end while**
20: $(i, m_i, \mathbf{M}_i) \leftarrow \min_{m_i} \mathcal{M}_{\text{memory}}$
21: $\mathbf{M}_{\text{set}} \leftarrow \mathbf{M}_i$
22: $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{T}_i\}$
23: **end while**
24: **return** $\mathcal{S}$
25: **end procedure**
26:
27: **procedure** GET_ADE($\mathbf{T}_i$, $\mathbf{T}_j$)
28: **return** $\frac{1}{T_f} \sum_{t=1}^{T_f} ||\boldsymbol{\tau}_i^t - \boldsymbol{\tau}_j^t||_2$
29: **end procedure**

one trajectory $\mathbf{T}_i$ that yields the lowest optimally achievable minADE m_i (Line 20 to Line 22) is picked and added to the trajectory set $\mathcal{S}$.

2) Agent Class-Specific Trajectory Sets: We also propose to use agent class-specific trajectory sets. The motivation is simple: The motion of an agent is highly dependent on its class. For instance, vehicles typically move faster and according to a different kinematic model than pedestrians. Having predefined trajectory sets for specific agent classes therefore inducts prior knowledge into the prediction model. Algorithm 1 is still used for trajectory set generation, but the dataset $\mathcal{D}$ is limited to trajectories of agents with selected classes only.

B. Encoder

Following the principle of the Graph Neural Network (GNN)-based LaneGCN [1], we use a graph-based encoding for lane centerlines and past agent trajectories: MapNet uses layers of LaneGCN in order to extract lane node features. In the ActorNet, a 1D-Convolutional Neural Network (CNN) and Feature Pyramid Network (FPN) are used to extract features from the past trajectories of agents. These trajectories are represented as a series of displacement vectors. We extend this representation by concatenating a one-hot encoding of the corresponding agent's class. Information of map and agents are combined in the FusionNet, a stack of four interaction blocks, namely Actor-to-Lane, Lane-to-Lane, Lane-to-Actor and Actor-to-Actor. Output of the encoder is one feature vector per agent. This feature vector is aware of the motion and position of surrounding agents (social context) and the lane geometries (static context).

It is important to emphasize that the encoder is not the focus of this work. We intentionally only use the series of displacement vectors, the agent classes and the lane centerlines as input information to our model. This allows for a fair comparison to the baselines introduced in Section V-D. As demonstrated in a prior publication [22], our encoder is extendable to making use of more detailed lane information and a full tracked state, including velocity and orientation. The publication also shows that making use of such information leads to an improvement in prediction performance.

C. Decoder

The classification head is a Multilayer Perceptron (MLP) with a $\mathrm{softmax}$ output layer. The output layer size corresponds to the number of trajectories in the trajectory set. Input to the MLP are the features resulting from the encoder. Output is a probability for each trajectory in the trajectory set. During inference, up to the size of the used trajectory set, a flexible and arbitrary amount of k top-scored trajectories can be used as predictions.

D. Non Maximum Suppression

To increase the diversity in the predictions, we make use of an NMS strategy. During inference, the trajectory with the highest probability is picked first and all trajectories with endpoints that lie within a radius r_{NMS} of the first trajectory's endpoint are discarded. This procedure is repeated until the desired number of modes k are obtained.

IV. Employing the Model to Conditional Behavior Prediction

RESET is the first set-based approach that is able to perform conditional behavior prediction, where additional information about the future trajectory of the autonomous vehicle is injected. Fig. 2 highlights the blocks used for this purpose in orange. Similar to ActorNet, the future trajectory of the autonomous vehicle is encoded with a 1D-CNN and a FPN (AVFutureNet). Input to AVFutureNet is the future trajectory of the autonomous vehicle, transformed to the local coordinate system used by ActorNet and MapNet. We examine five different ways and stages (Variant 1 to 5) of fusing this information with the features of predicted agents. Variant 1 to 3 are based on early fusion, because information about the future trajectory of the autonomous vehicle is injected prior to map encoding. Variant 4 and 5 are based on late fusion, as a maximum of one Actor-to-Actor block and the MLP decoder are applied after the fusion.

Variant 1 is a lane-based fusion. Information about the future trajectory of the autonomous vehicle is first fused into the lane nodes, using an AVFuture-to-Lane block. AVFuture-to-Lane uses the same attention-based fusion that is used in the Actor-to-Lane block in later stages of the fusion cycle. Subsequently, the Lane-to-Lane mechanism distributes this information across the lane graph.

Variant 2 uses an AVFuture-to-Lane interaction block that is similar to the Actor-to-Lane block used subsequently.

Variant 3 and 4 both use an AVFuture-to-Actor interaction block that is similar to the Actor-to-Actor block. Variant 3 does this in an early fusion manner, meaning that the encoded information of AVFutureNet is initially fused to the agent-wise features resulting from ActorNet. Variant 4 does this in a late fusion manner, meaning that the encoded information is fused subsequent to the Actor-to-Lane, Lane-to-Lane and Lane-to-Actor blocks.

Variant 5 is concatenation-based, meaning that the features resulting from AVFutureNet are concatenated to the features of each agent resulting from FusionNet. This results in a larger input layer of the classification head.

V. Experiments

This section describes the evaluation of our model, including the different variants for conditional behavior prediction, on the publicly available Argoverse Motion Forecasting Dataset 2 [9].

A. Dataset

The Argoverse 2 dataset contains $199\,908$ sequences for training, $24\,988$ for validation and $24\,984$ for testing. Each sequence consists of an HD map and a $11\,\mathrm{s}$ recording of detected and tracked agents in the surrounding of an autonomous vehicle, which is sampled with $10\,\mathrm{Hz}$. Given $5\,\mathrm{s}$ of a recording,

the goal is to predict the motion of the remaining 6 s of one selected agent, namely the focal agent.

B. Metrics

Evaluation is done using standard metrics for single- ($k = 1$) and multi-modal ($k = 6$) predictions. The minimum Average Displacement Error (minADE) is the average Euclidean distance between the predicted trajectory and the ground-truth trajectory. Similarly, the minimum Final Displacement Error (minFDE) is the Euclidean distance between the predicted endpoint and the ground-truth endpoint. For multi-modal minADE and minFDE evaluation ($k > 1$), only the prediction with the smallest Euclidean endpoint distance to the ground-truth is considered. minADE and minFDE are averaged over all sequences in the dataset. Miss Rate (MR) is the ratio of sequences where none of the k predicted endpoints lies within a radius of 2 m of the ground-truth endpoint.

Turn Rate Infeasibility (TRI) [23] is used to evaluate the percentage of trajectories that are kinematically infeasible in terms of their turn radius. Including the last observed position, predicted trajectories with a radius smaller than an agent class-specific threshold are labeled as infeasible and vice versa[1].

C. Implementation Details

The model is trained with cross-entropy as a loss function. The classification target is given by the trajectory in the trajectory set that is closest to the ground-truth in terms of average displacement.

Adam optimizer [24] with a learning rate of 10^{-3} and a batch size of 32 is used. After 8 epochs, the learning rate gets adjusted to 10^{-4} and the model is trained for 8 more epochs. Similar to LaneGCN, we use a feature size of 128 for all agent and map encodings. The classification decoder has a hidden size of 4096. For the NMS, we follow an occupancy prediction approach [25] and use $r_{\text{NMS}} = 1.8\,\text{m}$. For all models, including the baselines, the optimization during the training is limited to the focal agent only.

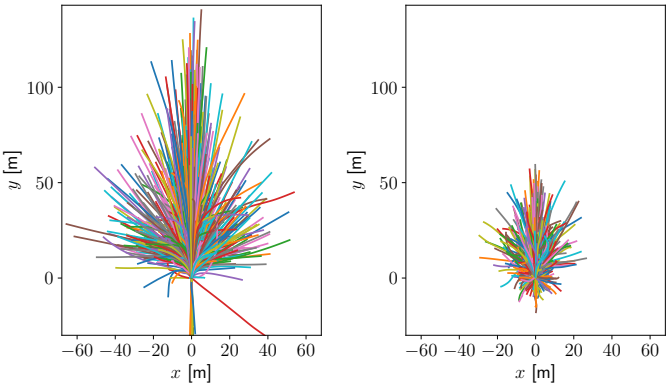
All trajectory sets are generated using a subset of 15 000 randomly picked trajectories from the training dataset. Instead of generating a separate trajectory set per agent class, we only create one trajectory set for non-vulnerable road users (vehicles, buses, motorcyclists) and one for vulnerable road users (pedestrian and cyclists).

D. Baselines

We use CoverNet and LaneGCN as baselines and reimplement them on Argoverse 2. **CoverNet** uses a BEV RGB image as the input representation. This does not allow the consideration of the dynamic state of the vehicle (including e.g., velocity). **LaneGCN** encodes the map by only using centerline information. Agent trajectories are encoded via a series of displacement vectors, meaning that a full tracked state is also not included. This baseline uses the exact same input information as our proposed RESET method. In contrast to our set-based method, it directly regresses a fixed number of trajectories.

As described in Section III-B, to ensure comparability to these baselines, our encoder only makes use of the same input information and does not include the full tracked state and lane information beyond centerlines.

[1]Used turn radius thresholds: vehicle: 1.8 m, bus: 3.0 m, motorcyclist: 0.8 m, cyclist: 0.6 m and pedestrian: 0.0 m.



(a) Non-vulnerable road users (b) Vulnerable road users

Fig. 3. Visualization of the generated agent class-specific trajectory sets for (a) non-vulnerable road users and (b) vulnerable road users.

E. Generated Trajectory Sets

Fig. 3 illustrates the agent class-specific trajectory sets for non-vulnerable road users and vulnerable road users generated with Algorithm 1. The two sets follow different velocity profiles and kinematic models. Also, there are trajectories aligned into negative y direction that go beyond the plot boundaries. During the inspection of the Argoverse 2 dataset, we found that this was from sequences where the orientation estimate during dataset recording was incorrect. In Table I, the Lower Bound minADE (LB minADE), which corresponds to the minADE that achievable in case of an optimal classification model, is listed. A general observation is that our metric-driven way of generating agent-class specific trajectory sets allows for better prediction results, provided that the classification model is able to classify the best trajectories.

F. Evaluation of Prediction Performance

This section contains quantitative and qualitative results of our method RESET. Our Python implementation achieves an average prediction time of 17.41 ms + 0.48 ms (NMS), making it suitable for real-time applications[2]. The time without NMS is independent of the number of predicted trajectories.

1) Quantitative Results: Table I lists the results of our experiments on the Argoverse 2 validation split. The results of CoverNet show a similar performance for the original CoverNet trajectory set ($\varepsilon = 3$) and our metric-driven trajectory set (1000). We argue that the achieved performance is limited by the rasterization-based input representation and the subsequent CNN. Using agent class-specific trajectory sets (1000+1000) induces more prior information and therefore slightly improves prediction performance.

[2]Test system specifications: Intel Core i9-7920X, NVIDIA GeForce RTX 2080 Ti.

TABLE I
QUANTIATIVE RESULTS ON THE VALIDATION SPLIT

Model	Type	Set size	LB minADE	$k=1$				$k=6$			
				minADE	minFDE	MR	TRI	minADE	minFDE	MR	TRI
LaneGCN	Reg.	-	-	**2.20**	5.74	**67.91**	31.53	**0.90**	**1.71**	**26.30**	27.47
CoverNet, $\varepsilon = 2$	Set	2816	0.75	3.46	8.70	78.71	32.09	1.85	3.83	44.09	30.03
CoverNet, $\varepsilon = 3$	Set	1228	0.97	3.24	8.07	79.30	31.22	1.75	3.31	45.65	28.50
CoverNet w/ our set	Set	1000	0.74	3.17	8.09	78.23	26.94	1.74	3.73	46.13	26.74
CoverNet w/ our set (agent class-specific)	Set	1000+1000	**0.73**	3.08	7.83	77.69	25.88	1.67	3.52	45.32	26.58
Ours w/ CoverNet set, $\varepsilon = 2$	Set	2816	0.75	2.38	5.84	71.23	29.54	1.42	2.56	33.01	29.01
Ours w/ CoverNet set, $\varepsilon = 3$	Set	1228	0.97	2.38	5.71	73.83	29.05	1.52	2.50	37.43	27.75
Ours w/ our set	Set	1000	0.74	2.27	5.59	69.91	25.84	1.34	2.54	35.98	25.92
Ours w/ our set (agent class-specific)	Set	1000+1000	**0.73**	2.26	**5.54**	70.55	**25.58**	1.33	2.51	35.90	26.08
Ours w/ our set (agent class-specific) + NMS	Set	1000+1000	**0.73**	2.26	**5.54**	70.55	**25.58**	1.26	2.28	31.27	**24.56**

TABLE II
SET SIZE ABLATION STUDY ON THE VALIDATION SPLIT

Set size	$k=1$				$k=6$			
	minADE	minFDE	MR	TRI	minADE	minFDE	MR	TRI
250+250	2.28	5.55	70.69	26.78	1.41	2.39	36.61	25.53
500+500	**2.23**	**5.45**	**70.01**	25.91	1.31	**2.28**	32.90	24.62
1000+1000	2.26	5.54	70.55	**25.58**	1.26	**2.28**	31.27	**24.56**
2000+2000	2.29	5.70	70.31	27.17	**1.25**	2.32	**30.61**	26.15

TABLE III
CONDITIONING VARIANT ABLATION STUDY ON THE VALIDATION SPLIT

Variant	RCC	$k=1$				$k=6$			
		minADE	minFDE	MR	TRI	minADE	minFDE	MR	TRI
1	82.73	4.37	8.88	80.26	**24.95**	2.35	3.92	44.18	**24.21**
2	82.73	2.26	5.59	70.28	26.31	1.27	2.31	31.80	24.97
3	82.69	**2.16**	**5.28**	68.63	25.89	**1.24**	**2.23**	**30.53**	25.03
4	64.31	2.17	5.29	**68.42**	26.02	1.25	2.26	30.87	25.22
$4s$	**40.50**	**2.16**	5.31	68.67	26.31	1.26	2.26	31.06	25.16
5	62.38	2.20	5.36	69.00	25.66	1.28	2.28	30.83	24.68

Independent of the trajectory set, the graph-based encoder of RESET is able to extract more relevant information about the traffic scene, which is why better results in terms of prediction metrics are achieved throughout. Also, there is a different trend with regard to the used trajectory sets: Our metric-driven way to generate trajectory sets outperforms the greedy bagging algorithm used in CoverNet for both, $\varepsilon = 3$ and $\varepsilon = 2$, by a margin. Combining it with agent class-specific trajectory sets is superior for single-mode ($k = 1$) and multi-modal ($k = 6$) prediction. Furthermore, the subsequent NMS increases prediction performance for multi-modal ($k = 6$) prediction even more, leading to a MR reduction by more than 4%. RESET with agent class-specific trajectory sets and NMS achieves the best values for $\text{minFDE}_{@k=1}$, $\text{TRI}_{@k=1}$ and $\text{TRI}_{@k=6}$. As is generally the case with GNN-based encoders [26], our encoder is extendable to make use of more detailed lane information and a full tracked state. Recent approaches making use of such information obtain even lower prediction errors [22]. Still, our chosen input representation allows for the fairest possible comparison to the baselines.

The direct regression-based approach LaneGCN, making use of the exact same input information, achieves the best multi-modal ($k = 6$) prediction results on the Euclidean metrics (minADE, minFDE and MR). However, it fails to match the set-based approaches in terms of TRI. One possible explanation is that the output is not restricted to specific feasible trajectories. Higher values in TRI possibly result in extra burdens for a downstream planner [17].

Table II ablates the performance of RESET for different set sizes. Using agent class-specific sets with the size 1000 balances single-mode ($k = 1$) and multi-modal ($k = 6$) prediction performance while achieving a low TRI.

2) Qualitative Results: Fig. 4 shows qualitative results of RESET. Sequences 1 to 3 illustrate RESET's ability to predict multi-modal trajectories for intersections with different topologies. Sequence 4 is a high-speed sequence, resulting in the predictions being bundled together. Sequences 5 and 6 show predictions of pedestrians with different velocities, resulting in probability distributions with different characteristics.

G. Evaluation of Conditional Behavior Prediction

This section contains quantitative and qualitative results of RESET for the task of conditional behavior prediction. Predictions are conditioned on the ground-truth trajectory of the autonomous vehicle.

1) Quantitative Results: Table III shows the results of the five variants for conditional behavior prediction. We introduce a new metric named Remaining Conditional Capacity (RCC). It corresponds to the ratio between all learnable model parameters used to encode the traffic scene conditioned on the future trajectory of the autonomous vehicle and the total number of model parameters. A lower RCC is especially advantageous when combining prediction with a downstream planner: For different planned trajectories of the autonomous vehicle, the values during the forward pass of the prediction module can be reused up to the fusion point. This means that late fusion is computationally cheaper than early fusion. Only the percentage of calculation measured by RCC has to be repeated during each forward pass.

Variant 3, which uses the AVFuture-To-Agent block prior to the main fusion cycle, achieves the best results in terms of prediction metrics. Variant 4 makes use of a similar AVFuture-To-Agent block, but performs the fusion in later stages of the fusion cycle, which is also reflected by the lower RCC. However, it still achieves results that are comparable to Variant 3. Prior publications [5], [6] that compare early and late fusion for conditional behavior prediction with regression decoders

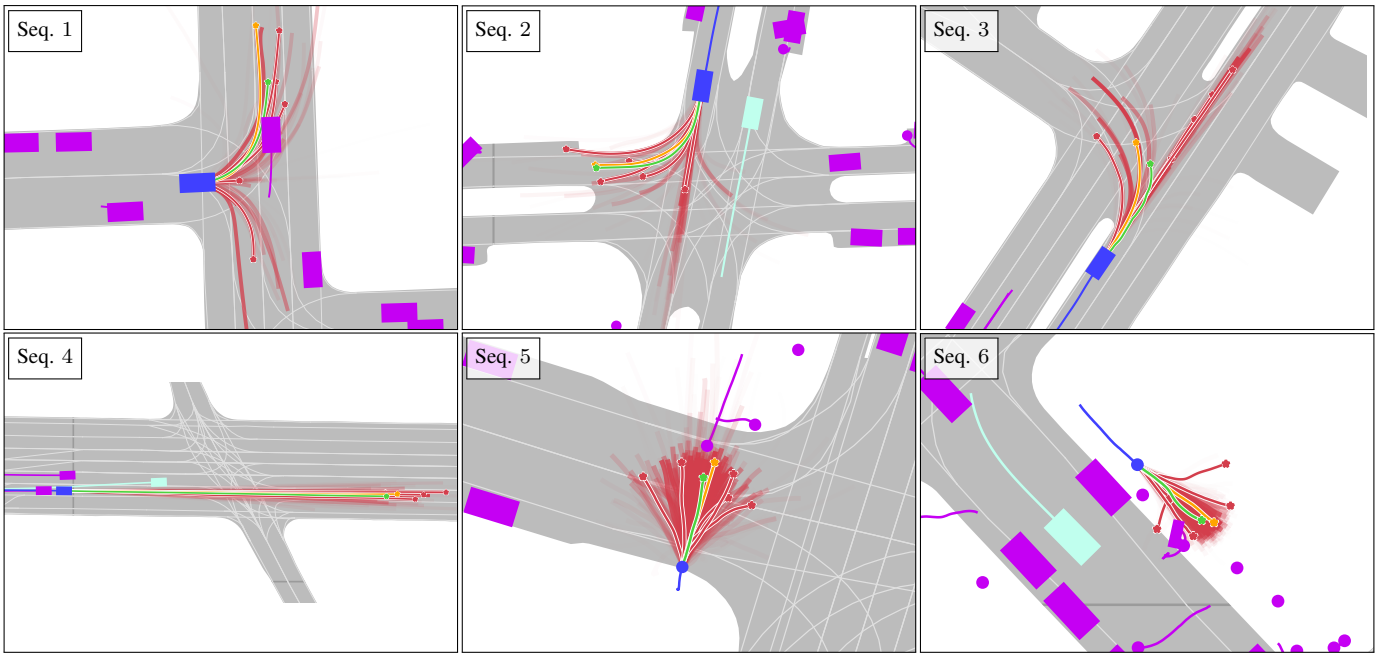


Fig. 4. Qualitative results of RESET on the Argoverse 2 validation set: The past observed trajectory of the focal agent is colored in blue, the ground-truth future trajectory in green. Predictions are colored in orange and red, with orange corresponding to the most probable future trajectory. The $k = 6$ selected modes are highlighted with a white outline, the remaining trajectories in the trajectory set obtain an alpha value that is proportional to their assigned probabilities. The past trajectory of the autonomous vehicle is colored in cyan and the past trajectories of other vehicles are colored in purple.

share the same observation: Early fusion always outperforms late fusion by a margin, with late fusion even leading to worse predictions than without conditioning. Interestingly, for our set-based method, late fusion obtains similar results than early fusion. We even test a Variant 4s that uses a decoder with a hidden size of 1024 instead of 4096 and still achieves comparable performance, while having a significantly lower RCC. Our set-based method therefore is the first approach with functioning late fusion, making it computationally lightweight to condition the predictions on multiple planned trajectories.

2) Qualitative Results: Fig. 5 compares qualitative results of RESET without (top) and with conditioning Variant 4 (bottom). Sequence 1 shows a left turn maneuver. Without conditioning, all predicted modes indicate that the predicted agent will continue driving straight. With conditioning, the model knows that the autonomous vehicle will yield at the intersection (the autonomous vehicle stands still, hence the future is not visible). The model concludes that the left turn maneuver is protected, resulting in significantly higher probabilities for left-turning trajectories. In Sequence 2, without conditioning, the agent is predicted to stop behind the autonomous vehicle or crash into it. With conditioning, the model knows that the autonomous vehicle will barely move forward, resulting in a nudging maneuver being predicted. In Sequence 3, without conditioning, the highest ranked prediction indicates that the agent will change the lane in order to drive around the parked cars in front. With the additional information that the autonomous vehicle will rapidly approach from behind, the agent is predicted to most likely wait behind the parked cars. This also corresponds to the ground-truth maneuver.

VI. Conclusion

This paper employs a set-based method to the task of conditional behavior prediction. The proposed method RESET combines a new metric-driven algorithm for set generation, a graph-based encoder and an NMS post-processing strategy. Results for unconditional prediction show that RESET is able to achieve performance comparable to that of a regression-based model. While late fusion fails in prior work, our set-based method achieves reasonable results with late fusion for conditional behavior prediction. It remains to combine our conditional set-based method with a downstream planner. This would allow for a computationally lightweight comparison of predictions resulting from different planned trajectories, allowing the planner to estimate the impact of its decisions on other traffic participants using the prediction model.

References

[1] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer, 2020, pp. 541–556.

[2] 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 *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 522–11 530.

[3] J. Gu, C. Sun, and H. Zhao, "Densetnt: End-to-end trajectory prediction from dense goal sets," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 15 303–15 312.

[4] W. Zeng, M. Liang, R. Liao, and R. Urtasun, "Lanercnn: Distributed representations for graph-centric motion forecasting," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 532–539.

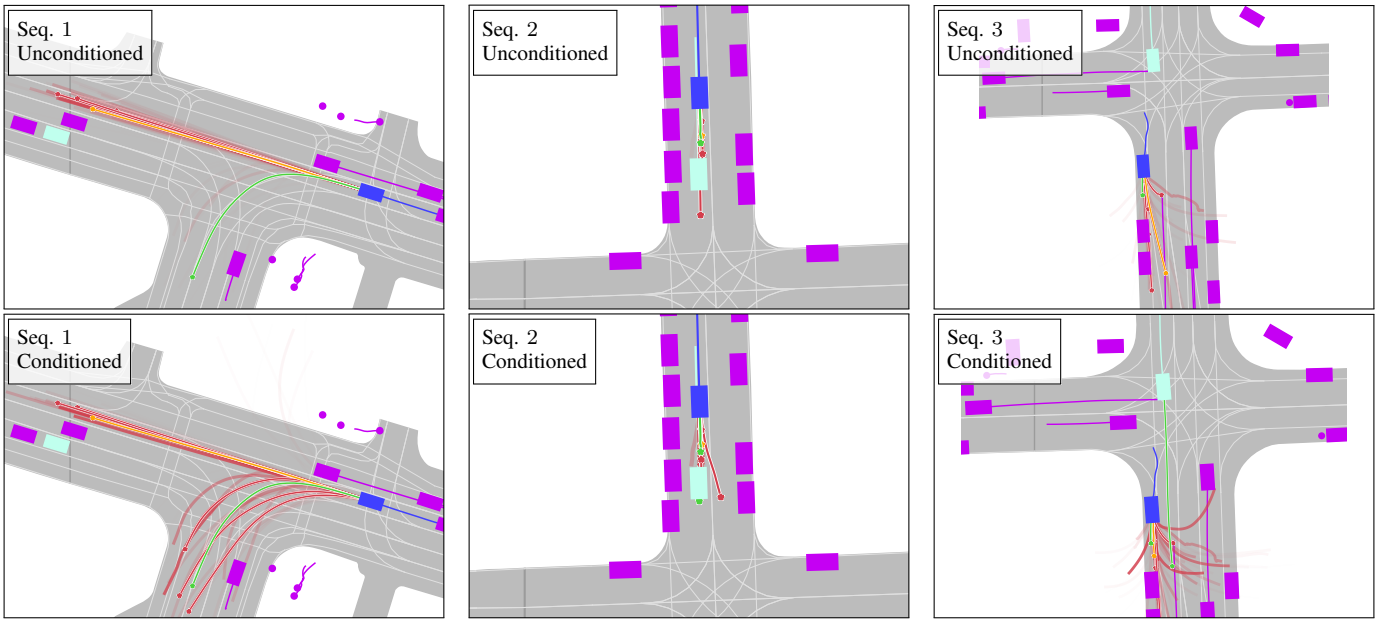


Fig. 5. Qualitative results of RESET on the Argoverse 2 validation set without conditional (top) and with conditional behavior prediction (bottom): Color codes are similar to Fig. 4. The ground-truth future trajectory of the autonomous vehicle (cyan) used for conditioning is also colored in green.

[5] Z. Huang, H. Liu, J. Wu, and C. Lv, "Conditional predictive behavior planning with inverse reinforcement learning for human-like autonomous driving," 2022.

[6] E. Tolstaya, R. Mahjourian, C. Downey, B. Vadarajan, B. Sapp, and D. Anguelov, "Identifying driver interactions via conditional behavior prediction," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 3473–3479.

[7] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, "DESIRE: distant future prediction in dynamic scenes with interacting agents," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2165–2174.

[8] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 14 062–14 071.

[9] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. Kaesemodel Pontes, D. Ramanan, P. Carr, and J. Hays, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," in *Proceedings of the Neural Information Processing (NeurIPS) Systems Track on Datasets and Benchmarks*, vol. 1, 2021.

[10] Z. Zhou, L. Ye, J. Wang, K. Wu, and K. Lu, "Hivt: Hierarchical vector transformer for multi-agent motion prediction," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8823–8833.

[11] J. Schmidt, J. Jordan, F. Gritschneder, and K. Dietmayer, "Crat-pred: Vehicle trajectory prediction with crystal graph convolutional neural networks and multi-head self-attention," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 7799–7805.

[12] N. Rhinehart, K. M. Kitani, and P. Vernaza, "R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer, 2018, pp. 794–811.

[13] C. Tang and R. R. Salakhutdinov, "Multiple futures prediction," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[14] 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*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer, 2020, pp. 624–641.

[15] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: socially acceptable trajectories with generative adversarial networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2255–2264.

[16] J. Wang, T. Ye, Z. Gu, and J. Chen, "Ltp: Lane-based trajectory prediction for autonomous driving," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 17 134–17 142.

[17] H. Song, D. Luan, W. Ding, M. Y. Wang, and Q. Chen, "Learning to predict vehicle trajectories with model-based planning," in *Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 164. PMLR, 2021, pp. 1035–1045.

[18] S. Khandelwal, W. Qi, J. Singh, A. Hartnett, and D. Ramanan, "What-if motion prediction for autonomous driving," 2020.

[19] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, D. Weiss, B. Sapp, Z. Chen, and J. Shlens, "Scene transformer: A unified multi-task model for behavior prediction and planning," in *10th International Conference on Learning Representations, ICLR 2022*, 2022.

[20] N. Rhinehart, R. Mcallister, K. Kitani, and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2821–2830.

[21] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer, 2020, pp. 683–700.

[22] C. Zhang, H. Sun, C. Chen, and Y. Guo, "Banet: Motion forecasting with boundary aware network," 2022.

[23] H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, J. Schneider, D. Bradley, and N. Djuric, "Deep kinematic models for kinematically feasible vehicle trajectory predictions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10 563–10 569.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015*, Y. Bengio and Y. LeCun, Eds., 2015.

[25] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Home: Heatmap output for future motion estimation," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 500–507.

[26] T. Monninger, J. Schmidt, J. Rupprecht, D. Raba, J. Jordan, D. Frank, S. Staab, and K. Dietmayer, "Scene: Reasoning about traffic scenes using heterogeneous graph neural networks," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1531–1538, 2023.