2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

MultiPath++: Efficient Information Fusion and Trajectory Aggregation for Behavior Prediction

Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S. Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, Benjamin Sapp

***Abstract*— Predicting the future behavior of road users is one of the most challenging and important problems in autonomous driving. Applying deep learning to this problem requires fusing heterogeneous world state in the form of rich perception signals and map information, and inferring highly multi-modal distributions over possible futures. In this paper, we present MultiPath++, a future prediction model that achieves state-of-the-art performance on popular benchmarks. MultiPath++ improves the MultiPath architecture [34] by revisiting many design choices. The first key design difference is a departure from dense image-based encoding of the input world state in favor of a sparse encoding of heterogeneous scene elements: MultiPath++ consumes compact and efficient polylines to describe road features, and raw agent state information directly (e.g., position, velocity, acceleration). We propose a context-aware fusion of these elements and develop a reusable *multi-context gating* fusion component. Second, we reconsider the choice of pre-defined static anchors, and develop a way to learn latent anchor embeddings end-to-end in the model. Lastly, we explore ensembling and output aggregation techniques—common in other ML domains—and find effective variants for our probabilistic multimodal output representation. We perform an extensive ablation on these design choices, and show that our proposed model achieves state-of-the-art performance on the Argoverse Motion Forecasting Competition [10] and the Waymo Open Dataset Motion Prediction Challenge [13].**

I. Introduction

Modeling and predicting the future behavior of human agents is a fundamental problem in many real-world robotics domains. For example, accurately forecasting the future state of other vehicles, cyclists and pedestrians is critical for safe, comfortable, and human-like autonomous driving. However, behavior prediction in an autonomous vehicle (AV) driving setting poses a number of unique modeling challenges:

1) *Multimodal output space*: The problem is inherently stochastic; it is impossible to truly know the future state of the environment. This is exacerbated by the fact that other agents' intentions are not observable, and leads to a highly multimodal distribution over possible outcomes (e.g., a car could turn left or right at an intersection). Effective models must be able to represent such a rich output space with high precision and recall matching the underlying distribution.

2) *Heterogenous, interrelated input space*: The driving environment representation can contain a *highly heterogeneous* mix of static and dynamic inputs: road network information (lane geometry and connectivity, stop lines, crosswalks), traffic light state information, and motion history of agents. Driving situations are often *highly interactive*, and can involve many agents at once (*e.g.* negotiating a 4-way stop with crosswalks). This requires careful modeling choices to capture interactions and relations between an agent and other agents as well as road elements. Effective models must capture not only the interactions between the agents in the scene, but also the relationships between the road elements and the behavior of agents given the road context.

The novel challenges and high impact of this problem have naturally garnered much interest in recent years. There has been a rich body of work on how to model agents' futures, their interactions, and the environment. However, there is little consensus to date on the best modeling choices for each component, and in popular benchmark challenge datasets [6], [10], [39], [13], there is a surprisingly diverse set of solutions to this problem; for details see Table I.

The MultiPath framework [34] addresses the multimodal output space challenge above by modeling the highly multimodal output distributions via a Gaussian Mixture Model. It handles a common issue of *mode collapse* during learning by using static trajectory anchors, an external input to the model. This practical solution gives practitioners a straightforward way of ensuring diversity and an extra level of modeler control via the design of such anchors. The choice of a GMM representation proved to be an extremely popular, appearing in many works—see Table I, "Trajectory Distribution", where "Weighted set" is a special case of GMMs where only means and mixture weights are modeled.

The MultiPath input representation and backbone draws heavily upon the computer vision literature. By rasterizing all world state in a top-down orthographic view, MultiPath and others [29], [23], [20], [35], [25], [9], [12], [38] leverage powerful, established CNN architectures like ResNet [19], which offer solutions to the heterogeneous interrelated input space: the heterogeneous world state is mapped to a common pixel format, and interactions occur via local information sharing via convolution operations. While convenient and established, there are downsides to such rasterization: (1) There is an uneasy trade-off between resolution of the spatial grid, field of view, and compute requirements. (2) Rasterizing is a form of manual feature engineering, and some features may be inherently difficult to represent in such a framework (*e.g.* radial velocity). (3) It is difficult to capture long range interactions via convolutions with small receptive fields. (4) The information content is spatially very sparse, making a dense representation a computationally wasteful choice.

[1] Waymo, 1600 Amphitheatre Pkwy, Mountain View, California, USA. {balakrishnanv, hefny, avikalp, krefaat, nigamaa, cornman, kanchen, bdou, cplam, dragomir, bensapp}@waymo.com

2022 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9681-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRA46639.2022.9812107

Method	Road Enc.	Motion Enc.	Interactions	Decoder	Output	Trajectory Distribution
Jean [26]	–	LSTM	attention	LSTM	states	GMM
TNT [40]	polyline	polyline	maxpool, attention.	MLP	states	Weighted set
LaneGCN [24]	GNN	1D conv	GNN	MLP	states	Weighted set
WIMP [21]	polyline	LSTM	GNN+attention.	LSTM	states	GMM
VectorNet [15]	polyline	polyline	maxpool,attention	MLP	states	Single traj.
MultiPath [34]	raster	raster	conv	MLP	states	GMM w/ static anchors
CoverNet [29]	raster	raster	conv	lookup	states	GMM w/ dynamic anchors
DESIRE [23]	raster	GRU	spatial pooling	GRU	states	Samples
RoadRules [20]	raster	raster	conv	LSTM	states	GMM
SocialLSTM [1]	–	LSTM	spatial pooling	LSTM	states	Samples
SocialGan [18]	–	LSTM	maxpool	LSTM	states	Samples
MFP [35]	raster	GRU	RNNs+attn	GRU	states	Samples
MANTRA [25]	raster	GRU	–	GRU	states	Samples
PRANK [2]	raster	raster	conv	lookup	states	Weighted set
IntentNet [9]	raster	raster	conv	conv	states	Single traj.
SpaGNN [8]	raster	raster	GNN	MLP	state	Single traj.
Multimodal [12]	raster	raster	conv	conv	states	Weighted set
PLOP [5]	raster	LSTM	conv	MLP	state poly	GMM
Precog [32]	raster	GRU	multi-agent sim.	GRU	motion	Samples
R2P2 [31]	raster	GRU	–	GRU	motion	Samples
HYU_ACE [28]	raster	LSTM	attn	LSTM	motion	Samples
Trajectron++[33]	raster	LSTM	RNNs+attention	GRU	controls	GMM
DKM [11]	raster	raster	conv	conv	controls	Weighted set
MultiPath++	polyline	LSTM+MCG	LSTM+MCG	MCG	state/control/poly	GMM

TABLE I

A SURVEY OF RECENT WORK IN BEHAVIOR PREDICTION, CATEGORIZED BY ROAD ENCODING, MOTION HISTORY ENCODING, AGENT INTERACTION ENCODING, TRAJECTORY DECODING, OUTPUT REPRESENTATION, AND FUTURE TRAJECTORY DISTRIBUTION. MCG IS DESCRIBED IN SECTION II-B.

In this paper, we introduce **MultiPath++**, which builds upon MultiPath, taking its output GMM representation and concept of anchors, but introducing key upgrades in how to represent and combine highly heterogeneous world state inputs and model interactions between state elements:

- Instead of MultiPath's rasterization-and-CNN approach, MultiPath++ directly processes sparse raw inputs. We represent road elements as polylines, agent history as a sequence of physical state encoded with RNNs, and agent interactions as RNNs over the state of neighbors relative to each ego-agent. These choices result in a compact representation that improves computational efficiency and captures long-range dependencies.
- We explicitly model relationships between road elements and agents instead of only encoding them independently. We propose a novel form of context awareness we call *multi-context gating* (MCG), in which sets of elements have access to a summary context vector upon which encodings are conditioned. MCG is a generic component that is applied throughout our model. It can be viewed as an efficient form of cross-attention, whose efficiency/quality trade-off depends on the size of the context vector.
- We also explore improvements in trajectory modeling, comparing representations based on kinematic controls, and/or polynomials as a function of continuous future time. We further demonstrate a way to learn latent representations of anchors and show they outperform the original static anchors of MultiPath, while simplifying model creation to a single-step process.
- Finally, we find significant additional gains on public benchmarks by applying ensembling techniques to our models. Unlike models with static anchors, the latent anchors of a MultiPath++ ensemble are not in direct correspondence. Furthermore, a lot of popular behavior prediction benchmarks have introduced metrics such as miss-rate (MR) and mean Average Precision (mAP), which require the ability to model diverse outcomes with few trajectories and differ from pure trajectory distance error capturing the average agent behavior. With the above in mind, we formulate the problem of ensembling the results of several models as one of greedy iterative clustering, which maximizes a probabilistic objective using Expectation Maximization [3].

As of November 1, 2021, MultiPath++ ranks 1^{st} on the Waymo Open Motion Dataset leaderboard[1], and 4^{th} on the Argoverse Motion Forecasting Competition[2]. We offer MultiPath++ as a reference set of design choices, empirically validated via ablation studies, that can be adopted, further studied and extended by the behavior modeling community.

II. MODEL ARCHITECTURE

Figure 1 depicts the proposed MultiPath++ model architecture. Like MultiPath [34], the model consists of an encoder and a predictor head which conditions on anchors and outputs the distribution of agent positions at future time steps.

MultiPath used a top-down image based representation for all input modalities (e.g., agents' tracked state, road network information), and a CNN encoder. In contrast, MultiPath++ has encoders processing each input modality and converting it to a compact and sparse representation; the

[1]https://waymo.com/open/challenges/2021/motion-prediction/

[2]https://eval.ai/web/challenges/challenge-page/454/leaderboard/1279

different modality encodings are later fused using a multi-context gating (MCG) mechanism.

A. Input Representation

MultiPath++ consumes the following input modalities:

- *Agent state history*: a state sequence describing the agent trajectory for a fixed number of past steps.
- *Road network*: The roadgraph is composed of polylines, each of which corresponds to an object such as a lane or a road boundary. We represent the roadgraph as a set of line segments extracted from the input polylines.
- *Agent interactions*: For a given ego agent, we consider all neighboring agents. For each neighboring agent, we extract features in the ego agent's coordinate frame, such as relative orientation, distance, history and speed. As a special case, we dedicate independent parameters for modeling the AV interactions in the same way.

The "Encoder" block in Figure 1 fuses these modalities into an encoding per agent, as described below.

B. Multi Context Gating for fusing modalities

In this section we focus on how to combine the different input modality encodings in an effective way. Other works use a common rasterized format [34], [38], a simple concatenation of encodings [23], [32], [33], or employ attention [27], [35], [15], [24]. We propose an efficient mechanism for fusing information we term *multi-context gating* (MCG), and use MCG blocks throughout the MultiPath++ architecture.

Given a set of elements $\mathbf{s}_{1:N}$ and an input context vector $\mathbf{c}$, a context gating (CG) block assigns an output $\mathbf{s}'_{1:N}$ to each element in the set, and computes an output context vector $\mathbf{c}'$ as follows: A single CG block is implemented via

$$\begin{array}{rclcrcl} \tilde{\mathbf{s}}_i & = & \mathrm{MLP}(\mathbf{s}_i), & & \tilde{\mathbf{c}} & = & \mathrm{MLP}(\mathbf{c}), \\ \mathbf{s}'_i & = & \tilde{\mathbf{s}}_i \odot \tilde{\mathbf{c}}, & & \mathbf{c}' & = & \mathrm{Pool}(\mathbf{s}'_{1:n}), \end{array} \tag{1}$$

where $\odot$ denotes element-wise product and Pool is a permutation-invariant pooling layer such as max or average pooling. In the absence of an input context, we simply set $\tilde{\mathbf{c}}$ to an all-ones vector in the first context gating block.

Note that both $\mathbf{s}'_i$ and $\mathbf{c}'$ depend on all inputs. Also, the output does not depend on the ordering of the elements. The size of the set n can vary across calls to $\mathrm{CG}(\cdot,\cdot)$. These properties make CG a set function [37], which is needed to encode a variable, unordered set of road elements and agents. CG can be viewed as an approximation to cross-attention; rather than each of n elements of a set attending to all m elements of another set, CG summarizes the latter set with the *single* context vector $\mathbf{c}$. The computational complexity of CG is $O(n)$, as opposed to $O(mn)$ in cross-attention. [3]

We stack multiple CG blocks by incorporating running-average skip-connections, as is done in residual networks [19]:

$$\bar{\mathbf{s}}^k_{1:n} = \tfrac{1}{k}\textstyle\sum_{j=1}^{k} \mathbf{s}^j_{1:n} \tag{2}$$

$$\bar{\mathbf{c}}^k = \tfrac{1}{k}\textstyle\sum_{j=1}^{k} \mathbf{c}^j \tag{3}$$

$$\left(\mathbf{s}^{k+1}_{1:n}, \mathbf{c}^{k+1}\right) = \mathrm{CG}\left(\bar{\mathbf{s}}^k_{1:n}, \bar{\mathbf{c}}^k\right). \tag{4}$$

We use $\mathrm{MCG}_N(\cdot,\cdot)$ to denote a stack of N CG blocks.

C. Encoders

We detail the specific encoders shown in Figure 1.

Agent history encoding. The agent history encoding is obtained by concatenating the output of three sources:

1) LSTM on the history features from H time steps ago to the present time: $(\mathbf{x}_t, \mathbf{y}_t)_{t=-H:0}$.
2) LSTM on the *difference* in the history features $(\mathbf{x}_t - \mathbf{x}_{t-1}, \mathbf{y}_t - \mathbf{y}_{t-1})_{t=-H+1:0}$.
3) MCG blocks applied to the set of history elements. Each element in the set consists of a historical position and time offset in seconds relative to the present time. The context input here is an all-ones vector with an identity context MLP. Additionally we also encode the history frame id as a one hot vector to further disambiguate the history steps.

We denote the final embedding, which concatenates these three state history encodings, as $\phi^{(\mathrm{state})}$.

Agent interaction encoding. For each modeled agent, we build an interaction encoding by considering each neighboring agent ν's past state observations: $\{\mathbf{x}^\nu_{-H}, \ldots, \mathbf{x}^\nu_{-1}, \mathbf{x}^\nu_0\}$. We transform ν's state into the modeled agent's coordinate frame, and embed it with an LSTM to obtain an embedding $\phi^{(\mathrm{interaction})}_\nu$. Note this is similar to the ego-agent history embedding but instead applied to the *relative coordinates of another agent*.

By doing this for n neighboring agents we obtain a set of interaction embeddings $\phi^{(\mathrm{interaction})}_{i=1:n}$. We fuse neighbor information with stacked MCG blocks as follows

$$\begin{aligned} &(\phi'^{(\mathrm{interaction})}_{1:n}, \phi'^{(\mathrm{state})}) = \\ &\quad \mathrm{MCG}_N(\phi^{(\mathrm{interaction})}_{1:n}, [\phi^{(\mathrm{state})}, \phi^{(\mathrm{interaction})}_{\mathrm{AV}}]), \end{aligned} \tag{5}$$

where the second argument is the input context vector to MCG, which in this case is a concatenation of the modeled agent's history embedding, and the AV's interaction embedding. In this way we emphasize the AV's representation as a unique entity in the context for all interactions.

Road network encoding. We use the polyline road element representation discussed in Section II-A as input. Each line segment is parameterized by its start point, end point and the road element semantic type Υ (*e.g.*, *Crosswalk*, *SolidDoubleYellow*,... *etc*). For each agent of interest, we transform the closest $P = 128$ polylines into their frame of reference and call the transformed segment $\mathbf{p} = (\mathbf{a}, \mathbf{b})$. Let $\mathbf{r}$ be the closest point from the agent to the segment, and $\mathbf{a}^\perp$ be the unit tangent vector at $\mathbf{a}$ on the original curve. Then we represent the agent's spatial relationship to the segment via the vector $[||\mathbf{r}||_2, \mathbf{r}/||\mathbf{r}||_2, (\mathbf{b}-\mathbf{a})/||\mathbf{b}-\mathbf{a}||_2, ||\mathbf{b}-\mathbf{a}||_2, ||\mathbf{b}-\mathbf{r}||_2, \mathbf{a}^\perp, \mathrm{OneHotEncoding}(\Upsilon)]$. These feature vectors are

[3]Note that CG may require a larger dimensionality of representation compared to cross-attention, since an entire set needs to be embedded in a context vector. In that sense CG allows us to trade off the representational power of full cross-attention with computational efficiency.

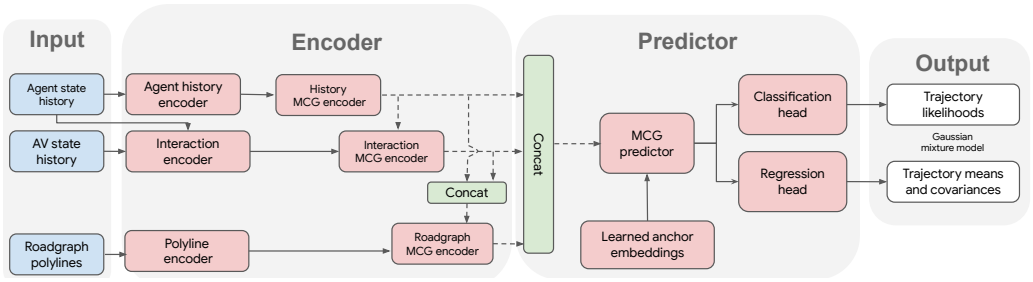

Fig. 1. MultiPath++ Model Architecture. MCG denotes Multi-Context Gating, described in Section II. Blocks in red highlight portions of the model with learned parameters. Dotted inputs to the MCG denotes context features. Each of the encoder MCG outputs aggregated embeddings (one per agent) as shown by dotted arrows. On the other hand, the predictor MCG outputs one embedding per trajectory per agent

each processed with a shared MLP, resulting in a set of agent-specific embeddings per road segment, which we denote by $\phi_{1:P}^{(\text{road})}$. We fuse road element embeddings with the agent history embedding $\phi^{(\text{state})}$ using stacked MCG blocks to enrich the road embeddings with dynamic state information.

$$(\phi_{1:P}^{\prime(\text{road})}, \phi^{\prime(\text{state})}) = \text{MCG}_N(\phi_{1:P}^{(\text{road})}, \phi^{(\text{state})}) \quad (6)$$

D. Output representation

MultiPath++ predicts a distribution of future behavior parameterized as a Gaussian Mixture Model (GMM), as is done in MultiPath [34] and other works [26], [29], [4].

E. Prediction architecture with learned anchor embeddings

The goal of the Predictor (Figure 1) is to predict M trajectories, with likelihoods and waypoint uncertainties.

To capture multimodal outcomes, Multipath employed a static set of *anchor trajectories* as pre-defined modes that applied to all scenes. This requires a large number of anchors since most modes are not a good fit to any particular scene. It also required a 2-phase learning process (first estimating the modes from data, then training the network).

Instead, MultiPath++ learns anchor embeddings as part of the overall model training. We interpret these embeddings $\mathbf{e}_{1:M}$ as anchors in latent space, and construct our architecture to have a one-to-one correspondence between these embeddings and the output trajectory modes. The vectors $\mathbf{e}_{1:M}$ are trainable model parameters that are input-independent. This has connections to DETR[7] and MANTRA[25], wherein a fixed set of embeddings are retrieved and decoded into trajectories.

We concatenate the embeddings $\phi^{\prime(\text{interaction})}$, $\phi^{\prime(\text{road})}$ and $\phi^{\prime(\text{state})}$ to obtain a fixed-length vector $\phi^{(\text{combined})}$ for each modeled agent. We use this as context in stacked MCG blocks that operate on the set of anchor embeddings $\mathbf{e}_{1:M}$, with a final MLP that predicts all GMM output parameters using a bivariate Gaussian: with mean, standard deviation in x and y, xy correlation, and mode likelihood:

$$(\phi_{1:M}^{(\text{output})}, \_) = \text{MCG}(\mathbf{e}_{1:M}, \phi^{(\text{combined})}),$$

$$(\mu, \sigma_x, \sigma_y, \rho_{xy}, \log q)_{1:M} = \text{MLP}(\phi_{1:M}^{(\text{output})}),$$

where the outputs are respectively the mean, standard deviation in x, standard deviation in y, correlation coefficient in x and y, and the log of the component likelihood.

F. Internal Trajectory Representation

We model the future position and heading of agents, along with agent-relative longitudinal and lateral Gaussian uncertainties. We parameterize the trajectory using $x(t), y(t), \theta(t), \sigma_{lng}(t), \sigma_{lat}(t)$—position, heading, and standard deviation for longitudinal and lateral uncertainty.

The most popular approach in the literature is to directly predict a sequence of such states at uniform time-discretization. Here we also consider two non-mutually exclusive variants. (1) We can represent functions over time as polynomials, which add an inductive bias that ensures a smooth trajectory. (2) Instead of directly predicting $\{x(t), y(t), \theta(t)\}$, we can predict the underlying kinematic control signals, which can then be integrated to evaluate the output state. In this work, we experiment with predicting the acceleration $a(t)$ and heading change rate $\dot{\theta}(t)$, and integrating them to recover the trajectory[4].

III. Ensembling predictor heads via bootstrap aggregation

Ensembling is a powerful and popular technique in many machine learning applications. For example, ensembling is a critical technique for getting the best performance on ImageNet [19]. By combining multiple models which are to some degree complementary, we can enjoy the benefits of a higher capacity model with lower statistical variance.

We apply bootstrap aggregation (bagging) [14] to our predictor heads by training E such heads together. To encourage models learning complementary information, the weights of the E heads are initialized randomly, and an example is used to update the weights of each head with a 50% probability.

Unlike scalar regression or classification, it is not obvious how to combine output from different heads in our case—each is a Gaussian Mixture Model, with no correspondence of mixture components across ensemble heads. Furthermore, we consider allowing each predictor head to predict a richer output distribution with more modes $L > M$; where M is fixed as a requirement for the task (and is used in benchmark metrics calculations).

To combine $E \times L$ trajectories into a final M output trajectories, we first union the set of all trajectories and

[4]we apply this integration on the rear-end of the vehicle as an approximation of the rear-axle.

renormalize their likelihoods. We then initialize M modes via a greedy set cover algorithm and refine them with a few rounds of the Expectation Maximization algorithm for GMMs [3]. This approach can be efficiently implemented in our model's compute graph, allowing us to train this aggregation end-to-end as a final layer in our deep network.

IV. Experiments

A. Datasets

The Waymo Open Motion Dataset (WOMD) [13] consists of 1.1M examples time-windowed from 103K 20s scenarios. The dataset is derived from real-world driving in urban and suburban environments. Each example for training and inference consists of 1 second of history state and 8 seconds of future, which we resample at 5Hz. The object-agent state contains attributes such as position, agent dimensions, velocity and acceleration vectors, orientation, angular velocity, and turn signal state.

The Argoverse dataset [10] consists of 333K scenarios containing trajectory histories, context agents, and lane centerline inputs for motion prediction. The trajectories are sampled at 10Hz, with 2 seconds of past history and a 3-second future prediction horizon.

B. Metrics

We use minimum distance error (minDE), minimum average distance error (minADE), miss rate (MR), overlap, and mean Average Precision (mAP) as defined in the Waymo Open Motion Dataset release [13].

C. MultiPath baseline

We include a reference MultiPath model where the input and backbone are faithful to the original paper [34] for a point of comparison. All settings are the same with the exception of using a simpler *splat rendering* [41] approach for rasterization, which can take advantage of hardware acceleration as part of the model's compute graph.

D. External benchmark results

On Argoverse, MultiPath++ achieves top-5 performance on most metrics (Table II). Our technique is ranked 1^{st} on all metrics on Waymo Open Motion Dataset [13] (Table III). The tested model is based on the best configuration in Table IV, where the outputs from multiple ensemble heads are aggregated as described in Section III. On WOMD, we also see that the original MultiPath model, even with the refinement of learned anchors and ensembling, is outperformed by more recent methods. We note that MultiPath is the best performing top-down rendering model; every known method which outranks it uses sparse representations.

E. Qualitative Examples

Figure 2 shows examples of Multipath++ on WOMD scenes, demonstrating the ability to handle different road layouts and agent interactions. Figure 3 shows how aggregation removes unrealistic trajectories, while producing diverse geometry with calibrated likelihoods.

Argoverse leaderboard ($k = 6$, $d = 2m$, $t = 3s$)			
	brier-minDE	MR	minADE
LaneGCN [24]	2.059	0.163	0.868
DenseTNT [17]	1.976	0.126	0.882
HOME + GOHOME [16]	1.860	**0.085**	0.890
TPCN++ [36]	1.796	0.116	**0.780**
MultiPath++(ours)	1.793	0.132	0.790
QCraft Blue Team	**1.757**	0.114	0.801

TABLE II

Comparison with select state-of-the-art methods on Argoverse (k = number of trajectories, d = max distance for no miss, t = trajectory time duration).

Waymo Open Motion Prediction ($k = 6$, $t = 8s$)				
	minADE	MR	Overlap	mAP
MultiPath [34]	0.880	0.345	0.166	0.409
SceneTransformer [27]	0.612	0.156	0.147	0.279
DenseTNT [17]	1.039	0.157	0.178	0.328
MultiPath++	**0.556**	**0.134**	**0.131**	**0.409**

TABLE III

Comparison with published state-of-the-art methods on WOMD public leaderboard. k is the number of trajectories and t is the trajectory time horizon.

F. Ablation Study

In this section we evaluate our design choices through an ablation study. Table IV summarizes ablation results. In the following subsections we discuss how our architecture choices affect the model performance.

1) Set Functions: MultiPath++ uses MCG in two ways: (1) encoding a set of elements (e.g. agents, roadgraph segments) into a single feature vector, and (2) converting the set of learned anchors, together with the encoded feature vector as a context, into a corresponding set of trajectories with likelihoods. We experimented with other set functions:

- MLP+MaxPool: We replace the MCG road network encoder with a MLP+MaxPool applied on points rather than polylines, inspired by PointNet [30]. We use a 5 layer deep MLP and RELU activations.
- Equivariant DeepSet [37]: The equivariant set function is represented as a series of blocks, each involving an element-wise transformation followed by pooling to compute the context. We use a DeepSet of 5 blocks in the predictor.
- Transformers [22]: We replace the gating mechanism on polylines with self-attention. For decoding, we used cross attention where the queries are the learned embeddings and the keys are the various encoder features.

2) Trajectory representation: As mentioned in Section II-F, we experiment with predicting polynomial coefficients for the trajectory, as well predicting kinematic control signals. We found that polynomial representations hurt performance, counter to conclusions made in PLOP [4]. We note that, in the PLOP datasets, we need to predict 4s into the future. For such short futures, polynomial representations can be more suitable. In our case, we do not see much gains from using the polynomial representation, possibly due to the larger dataset size and longer-term prediction horizon of 8s.

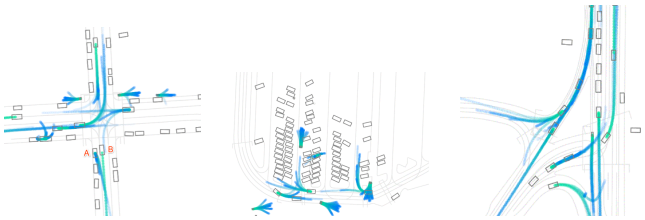

Fig. 2. Examples of MultiPath++ predictions for 8 seconds in WOMD scenes (intersection, parking lot, and atypical roadgraph). Hue indicates time horizon while transparency indicates predicted probability. Rectangles indicate vehicles while small squares indicate pedestrians.

	minDE	minADE	MR	mAP
Original MultiPath	4.752	1.796	0.749	–
Set Function				
MLP+MaxPool	2.693	1.107	0.528	0.367
DeepSet	2.562	1.055	0.5	0.368
Transformer	2.479	1.042	0.479	0.3687
1 MCG block	2.764	1.15	0.55	0.312
5 stacked MCG blocks⋆	**2.305**	**0.978**	**0.44**	**0.393**
Trajectory Representation				
Polynomial	2.537	1.041	0.501	0.368
Control	2.319	0.987	0.449	0.386
Raw coordinates⋆	**2.305**	**0.978**	**0.44**	0.393
Raw coordinates w/ heading	2.311	**0.978**	0.443	**0.395**
Ensembling				
$E = 1, L = 6$	2.333	0.982	0.410	0.240
$E = 5, L = 6$	**2.18**	**0.948**	**0.395**	0.297
$E = 1, L = 64$	2.487	1.057	0.473	0.367
$E = 5, L = 64$ ⋆	2.305	0.978	0.44	**0.393**
Anchors				
Static k-means anchors	2.99	1.22	0.578	0.324
Learned anchors⋆	**2.305**	**0.978**	**0.44**	**0.393**

TABLE IV

MODEL ABLATION ON WOMD VALIDATATION SET. METRICS ARE PARAMETERIZED BY $k = 6, t = 8s$, AND $d = 2m$.
⋆ DENOTES THE REFERENCE CONFIGURATION: ROAD ENCODING, STATE HISTORY ENCODING AND INTERACTION ENCODING AS DESCRIBED IN SECTION II.

The controls-based output works better in distance metrics than polynomial representations, which suggests it is a more beneficial inductive bias. Overall, our results suggest that the simple sequence of raw coordinates trajectory representation works best for distance-based metrics. However, these unconstrained representations have a non-trivial rate of kinematic infeasibility. We compute the circumradius constituting any 3 consecutive points on a trajectory, and consider this turning radius infeasible if smaller than the turning radius of a midsize sedan (3.5 meters). The rate of turning radius infeasibility for the relevant methods in Table IV are Polynomial=1.92%, Control=1.22%, Raw coordinates=1.08% and Raw coordinates w/ heading=1.04%. However, with the latter, we find that the predicted heading turning radius is inconsistent with the waypoint circumradius estimate by $\geq .05$ radians 9.9% of the time, but is never inconsistent in the Controls method (by construction).

3) Ensembling: We explore ensembling as proposed in Section III. We use E heads, each of which produces L trajectories. All $E \cdot L$ trajectories are aggregated into $M = 6$ trajectories, as required for the WOMD submission.

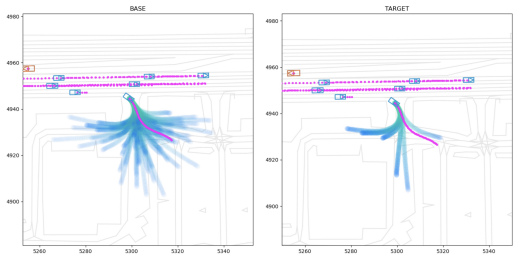

Fig. 3. Illustration of weeding out unrealistic trajectories after aggregation. Blue color depicts the predicted trajectories, and purple color depicts the logged ground-truth.

G. Discussion

First, we remark that MultiPath++ is a significant improvement over MultiPath. As discussed, they differ in many design dimensions, the primary being the change from top-down rasterization representation to a sparse element-based representation with agent-centric coordinate systems. Other design choices are validated in the following discussion.

We find that MLP+MaxPool performs the worst among all set function variants as expected due to limited capacity. DeepSet is able to outperform MLP+MaxPool. Also increasing the depth of the MCG gives consistently better results owing to effective increase in capacity and flow of information across skip connections. We get the best performance by increasing the depth of the MCG to 5 layers.

We find that learning anchors ("Learned anchors") is more effective than using a set of anchors obtained a priori via k-means. This runs counter to the original findings in the MultiPath paper [34] that anchor-free models suffer from mode collapse. The difference could possibly be due to the richer and more structured inputs, improved model architecture, and larger batch sizes in MultiPath++. We leave more detailed ablations on this issue between the two approaches to future work.

We compare the baseline of directly outputting a single head with 6 trajectories ($E = 1, L = 6$), to training 5 ensemble heads ($E = 5, L = 6$). We see that ensembling significantly improves most metrics, and particularly minDE, for which this combination is best. We also train a model with a single head that outputs 64 trajectories, followed by our aggregation method that reduces them to 6 ($E = 1, L = 64$). Compared to our initial baseline, this model significantly improves mAP, but regresses the average trajectory distance metrics $minDE$, and even $minADE$ a little bit. This suggests that the different metrics pose different solution requirements. As expected, our aggregation criterion is well suited to preserving diversity, while straight-up ensembling is better at capturing the average distribution. Finally, our experiment ($E = 5, L = 64$) with more ensemble heads and more predictions per ensemble combines the strengths of both techniques, obtaining a strictly superior performance in the majority of metrics compared to the baseline.

REFERENCES

[1] Alexandre "Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *CVPR*, 2016.

[2] Yuriy Biktairov, Maxim Stebelev, Irina Rudenko, Oleh Shliazhko, and Boris Yangel. Prank: motion prediction based on ranking. *arXiv preprint arXiv:2010.12007*, 2020.

[3] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[4] Thibault Buhet, É. Wirbel, and Xavier Perrotton. Plop: Probabilistic polynomial objects trajectory planning for autonomous driving. *ArXiv*, abs/2003.08744, 2020.

[5] Thibault Buhet, Emilie Wirbel, and Xavier Perrotton. Plop: Probabilistic polynomial objects trajectory planning for autonomous driving. *arXiv preprint arXiv:2003.08744*, 2020.

[6] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing.

[8] Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In *IEEE Intl. Conf. on Robotics and Automation*. IEEE, 2020.

[9] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conf. on Robot Learning*, 2018.

[10] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *CVPR*, 2019.

[11] Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Jeff Schneider, David Bradley, and Nemanja Djuric. Deep kinematic models for kinematically feasible vehicle trajectory predictions. In *IEEE Intl. Conf. on Robotics and Automation*, pages 10563–10569. IEEE, 2020.

[12] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *IEEE Intl. Conf. on Robotics and Automation*, 2019.

[13] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset, 2021.

[14] Jerome H Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. springer open, 2017.

[15] 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 *CVPR*, 2020.

[16] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output for future motion estimation. *arXiv preprint arXiv:2109.01827*, 2021.

[17] Junru Gu, Qiao Sun, and Hang Zhao. Densetnt: Waymo open dataset motion prediction challenge 1st place solution. *CoRR*, abs/2106.14160, 2021.

[18] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *CVPR*, 2018.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[20] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *CVPR*, 2019.

[21] Siddhesh Khandelwal, William Qi, Jagjeet Singh, Andrew Hartnett, and Deva Ramanan. What-if motion prediction for autonomous driving. *ArXiv*, 2020.

[22] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR, 2019.

[23] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip H S Torr, and Manmohan Chandraker. DESIRE: Distant future prediction in dynamic scenes with interacting agents. In *CVPR*, 2017.

[24] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. *arXiv preprint arXiv:2007.13732*, 2020.

[25] Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. Mantra: Memory augmented networks for multiple trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7143–7152, 2020.

[26] Jean Mercat, Thomas Gilles, Nicole Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Gil. Multi-head attention for joint multi-modal vehicle motion forecasting. In *IEEE Intl. Conf. on Robotics and Automation*, 2020.

[27] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, David Weiss, Benjamin Sapp, Zhifeng Chen, and Jonathon Shlens. Scene transformer: A unified multi-task model for behavior prediction and planning. *CoRR*, abs/2106.08417, 2021.

[28] Seong Hyeon Park, Gyubok Lee, Manoj Bhat, Jimin Seo, Minseok Kang, Jonathan Francis, Ashwin R Jadhav, Paul Pu Liang, and Louis-Philippe Morency. Diverse and admissible trajectory forecasting through multimodal context understanding. *arXiv preprint arXiv:2003.03212*, 2020.

[29] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. CoverNet: Multimodal behavior prediction using trajectory sets. *arXiv:1911.10298*, 2019.

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[31] Nicholas Rhinehart, Kris Kitani, and Paul Vernaza. R2P2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *ECCV*, 2018.

[32] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. PRECOG: Prediction conditioned on goals in visual multi-agent settings. In *Intl. Conf. on Computer Vision*, 2019.

[33] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. *arXiv preprint arXiv:2001.03093*, 2020.

[34] Benjamin Sapp, Yuning Chai, Mayank Bansal, and Dragomir Anguelov. MultiPATH: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conf. on Robot Learning*, 2019.

[35] Yichuan Charlie Tang and Ruslan Salakhutdinov. Multiple futures prediction. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

[36] Maosheng Ye, Tongyi Cao, and Qifeng Chen. Tpcn: Temporal point cloud networks for motion forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11318–11327, 2021.

[37] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[38] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *CVPR*, 2019.

[39] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kümmerle, Hendrik Königshof, Christoph Stiller, Arnaud de La Fortelle, and Masayoshi Tomizuka. INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. *arXiv:1910.03088*, 2019.

[40] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia

Schmid, et al. Tnt: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294*, 2020.

[41] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001.