Efficient Motion Prediction: A Lightweight & Accurate Trajectory Prediction Model With Fast Training and Inference Speed

Alexander Prutsch[1], Horst Bischof[1] and Horst Possegger[1]

***Abstract*— For efficient and safe autonomous driving, it is essential that autonomous vehicles can predict the motion of other traffic agents. While highly accurate, current motion prediction models often impose significant challenges in terms of training resource requirements and deployment on embedded hardware. We propose a new efficient motion prediction model, which achieves highly competitive benchmark results while training only a few hours on a single GPU. Due to our lightweight architectural choices and the focus on reducing the required training resources, our model can easily be applied to custom datasets. Furthermore, its low inference latency makes it particularly suitable for deployment in autonomous applications with limited computing resources.**

I. INTRODUCTION

Predicting the motion of other traffic agents is an essential task for the efficient and safe operation of self-driving robots. This applies to a wide range of use cases, *i.e.,* autonomous robots in intralogistics and autonomous vehicles in street traffic. Motion prediction algorithms use environment data, *e.g.,* road typologies and historical movement data of agents. Hence, methods are usually built on top of basic perception tasks like object detection and tracking, and combine them with static information, *i.e.,* high-definition (HD) maps. Subsequently, autonomous systems use the predicted future trajectories of other traffic participants as input to planning modules. This allows the autonomous system to proactively react to its environment. As a benefit, self-driving vehicles can move more fluently in traffic and increase safety by taking the future behavior of others into account.

Extensive research on autonomous driving cars yields great results for trajectory prediction methods. In addition, various large-scale benchmark datasets are available for evaluating motion prediction models on street traffic, *e.g.,* Argoverse 1 (AV1) [1], Argoverse 2 (AV2) [2], nuScenes [3] and Waymo Open Motion Dataset (WOMD) [4]. In recent years, transformer-based encoder-decoder methods, *e.g.,* [5], [6], [7], [8], [9], showed a performance advantage over methods that are CNN-based, *e.g.,* [10], [11], graph-centric methods, *e.g.,* [12], [13], or recurrent neural networks-based approaches, *e.g.,* [14].

While yielding highly accurate predictions and overall excellent results, transformer-based methods are resource-demanding, especially for training the respective models as outlined *e.g.,* in [6], [15], [16], [17]. This yields to manifold practical implications: without significant resource investments, it is impossible to train these methods on custom datasets. Also, adapting the models to use-case-specific requirements is difficult. Furthermore, the deployment on embedded hardware, *e.g.,* for use on mobile robots, is non-trivial.

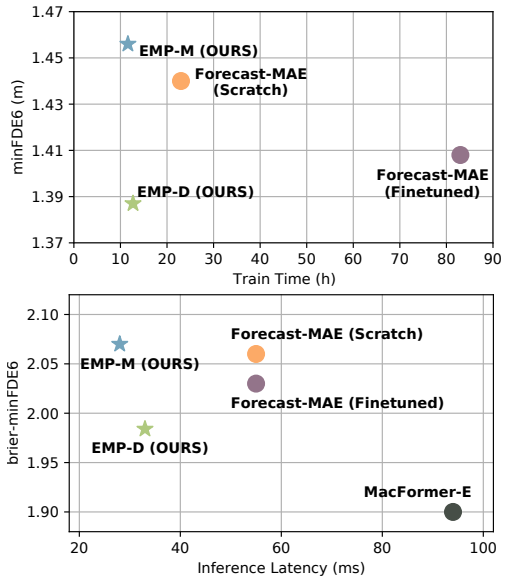

Fig. 1. Resource and accuracy trade-off of our model compared to Forecast-MAE [8] and MacFormer [18]. Top: Required training time on a single NVIDIA V100 GPU vs. the minFDE_6 on the Argoverse 2 validation set. Bottom: Inference latency for predicting 32 scenarios on a single NVIDIA RTX 2080 TI vs. the brier-minFDE_6 on the AV2 test set.

To overcome the shortcomings of resource-demanding trajectory prediction models, we propose a new **e**fficient **m**otion **p**rediction model (EMP). The design of EMP focuses on maximizing performance given a restricted training infrastructure. We show that our method achieves competitive results on the challenging AV2 dataset, despite training only for a few hours on a single NVIDIA V100 GPU with 16 gigabytes VRAM. We also highlight that our model achieves excellent inference speed on different GPU architectures.

Our model architecture builds on recent advances [8], [9], which show that excellent performance can also be achieved using only simplistic network blocks. This contrasts with other previous works, such as [6], [15], where improvements were mainly achieved by complex model design or explicit modeling of previous knowledge, *i.e.,* goal state candidates.

Compared to other methods that also put a focus on inference speed (*e.g.,* [18], [16], [18]), we explicitly also emphasize training speed. Compared to Forecast-MAE (one of the few methods which can be trained on a single

[1] Alexander Prutsch, Horst Bischof and Horst Possegger are with the Institute of Computer Graphics and Vision, Graz University of Technology. Corresponding author: alexander.prutsch@tugraz.at

GPU with reasonable batch-size), we successfully speed-up training by nearly 100% while achieving even better scores. In addition, we perform our main evaluation on the much more complex AV2, whereas many highly efficient methods, *e.g.,* [19] are only evaluated on the simpler AV1 dataset.

Our model architecture is based on standard transformer blocks for encoding agent histories, road topology and scene information. For decoding the future trajectories and confidence scores, we experiment with different decoder architectures. We compare the use of a simple multi-layer perceptron-based decoder with a sophisticated transformer-based method.

In summary, our main contributions include:

- We propose a new, simplistic but highly effective, transformer-based trajectory prediction model achieving results competitive to the state-of-the-art without incorporating prior knowledge or time intensive pre-training.
- We put an explicit emphasis on limiting the required resources for training our model, making the model easily adoptable by practitioners.
- We provide preliminary results on the AV1 dataset and execute a detailed model study on the complex AV2 dataset, where we report training resources, inference speed and prediction accuracy.

II. RELATED WORK

A. Transformer-based Motion Prediction Models

In recent years, the use of transformer methods for motion prediction has been widely studied. Early works use attention mechanisms to model the interaction between different agents in traffic scenes, *e.g.,* [20], [21], [22]. LaneGCN [12] uses an attention-based network to fuse actor and map features, which are represented in a graph structure. The fusion network executes actor-to-lane, actor-to-actor, lane-to-lane and lane-to-actor attention to model the relationship between the target agent, road topology and other agents.

mmTransformer [5] uses a stacked transformer backbone. Its transformer blocks aggregate different input modalities (*i.e.,* map information, target agent history and agent interactions) using cross-attention. They employ a custom training strategy and predict future trajectories based on fixed proposals. Wayformer [7] studies different attention types and multiple strategies to fuse multi-modal input data. Implementing a transformer-based encoder and decoder structure, they achieved state-of-the-art results in 2022 on the Argoverse 1 (AV1) and Waymo Open Motion Dataset (WOMD) using a modality-agnostic early fusion method.

MTR [6] introduces a trajectory decoder based on the detection transformer (DETR) [23]. It combines a global intent localization module with a local movement refinement by iteratively updating the predictions based on a set of motion queries. To this end, they initially cluster the training data endpoints into a fixed set of intent points, which are then used to create the motion query. Hence, each motion query is used to model a specific agent intent.

QCNet [15] also utilizes a DETR-like decoder structure. To overcome the limitations of anchor-based approaches (dependence on prior knowledge) and anchor-free approaches (mode collapse), it utilizes a two-stage decoding pipeline. First, trajectory proposals are generated using anchor-free, learnable queries. Second, the proposals are used as anchors for generating the final predictions. Both stages recurrently apply cross-attention to scene and agent tokens, as well as, self-attention across mode queries.

In recent work, Forecast-MAE [8] shows that a simplistic network architecture without additional prior knowledge is also capable of yielding competitive results on the AV2 dataset. Furthermore, they boost their prediction accuracy by implementing a pre-training scheme for masked agents and lane tracks. Forecast-MAE utilizes a neighborhood attention [24] to encode agent histories and a mini-PointNet [25] based approach to encode lane information. Then, they concatenate actor and lane tokens, add positional information and encode it using self-attention via standard transformer blocks. They use a simple multi-layer perceptron (MLP)-based decoder for generating the trajectories and scores. SEPT [9] also follows the idea of using a simple network structure and pre-training. They utilize self-attention and max-pooling on agent histories to compute the agent tokens. Following, they use a spatial encoder to learn the spatio-temporal information of agents and the road network. As a decoder, they also propose the use of a DETR-like module that does cross-attention of learnable queries to the scene tokens. Additionally, the pre-training tasks differ from Forecast-MAE, and the token granularity is finer (waypoints instead of whole history/future and short road vectors compared to whole segments). Also, the bigger model size is a likely reason for the performance advantage over Forecast-MAE. At the moment, SEPT achieves state-of-the-art results on AV1 and AV2. A major shortcoming of these approaches, however, is that training/inference is resource intensive – we address this with our more lightweight prediction model.

B. Lightweight Motion Prediction Models

HiVT [26] hierarchically applies transformer blocks for efficient learning of spatio-temporal information. The scene is split into different regions, which are then processed by local encoders. Following this, a global interaction module is used to model the interaction between different regions. They provide a detailed study on the influence of the local region size, which yields an information vs. complexity trade-off and, furthermore, an accuracy vs. inference speed trade-off.

ADAPT [19], ProphNet [16] and MACFormer [18] are three recent motion prediction methods with an emphasis on inference speed. ADAPT utilizes a LaneGCN [12] inspired transformer-based encoder with modality-specific relation modeling. Following this, they use a two-stage MLP-based decoder. Initially, they predict endpoint candidates based on the scene features. Next, the scene features and endpoints are fed into an MLP to generate refined endpoints. They only evaluate on the AV1 dataset and do not provide results, *e.g.,* on the challenging AV2 dataset.

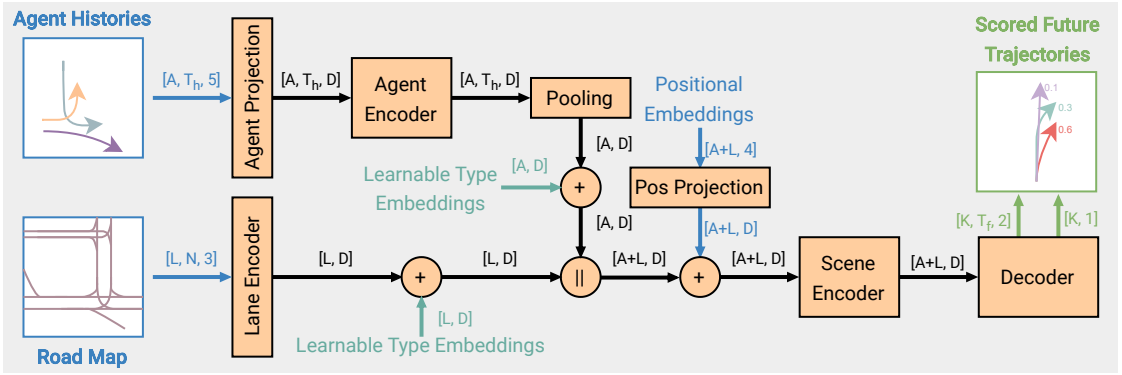

Fig. 2. General overview of our EMP model architecture. || denotes concatenation.

ProphNet [16] also utilizes a multi-stage decoding approach after encoding the agent and road inputs in an agent-centric approach using gated MLPs [27]. Then, they generate proposals by cross-attending learnable queries to the agent histories and anchors by self-attention of scene features. Subsequently, the proposals and anchors are combined to predict the trajectories. However, ProphNet requires significant training resources, *i.e.,* 16 NVIDIA V100 GPUs with a batch size of 64, whereas our model can be trained on a single V100 with batch size 96.

MacFormer [18] uses a multitask optimization strategy which explicitly takes map constraints into account. The single MacFormer models show low latency and good accuracy on the AV 1 dataset. On the difficult AV2 dataset, they propose a model ensemble to achieve state-of-the-art results. Compared to our approach, their ensemble method yields slightly better accuracy, but also has a much higher latency. Furthermore, training a model ensemble opposes our goal of using only limited training resources.

HPTR [17] provides a hierarchical framework based on a new attention mechanism using K-nearest neighbor attention and relative pose encoding. In their work, they also address the required training resources. They train for 5–10 days on 4 NVIDIA RTX 2080 Ti GPUs (batch size 12), which is significantly less training effort than, *e.g.,* Wayformer [7] and ProphNet [16]. Even with fewer training resources (our models require less than 15 hours to train on a single RTX 2080 Ti with batch size 64), our model exceeds the performance of HPTR on the AV2 dataset.

III. EFFICIENT TRAJECTORY PREDICTION

We propose a new efficient motion prediction model (EMP) based on standard transformer blocks. Our model follows recent ideas to use a simplistic architecture and avoid the introduction of inductive bias via prior knowledge [8], [9]. Figure 2 gives an overview of our method architecture.

Our main focus is to build a model without processing bottlenecks, which allows fast and efficient training of the model, as well as rapid inference speed. To this end, we also disregard the idea of pre-training proposed by [8], [9]. While pre-training allows to achieve performance gains compared to training from scratch, it heavily contradicts with our emphasis on low training resource consumption. For reference, pre-training Forecast-MAE on the AV2 dataset would take around 60 hours on a single V100 GPU.

A. Agent Encoding

We use the 2D position of agents, agent velocity magnitude, a step counter and a mask flag (indicates if agent was observed at a given timestamp) to encode the state of each agent a at time t into a 5-element tensor. The step counter is used to preserve the state sequence order during attention operations. The agent positions are normalized with respect to the center pose of each track. Next, we project the state tensor $\mathbb{R}^{A\times T_h\times 5}$, where A corresponds to the number of agents and T_h corresponds to the number of historic time steps, to our embedding space $\mathbb{R}^{A\times T_h\times D}$. To this end, we utilize a simple linear layer.

Subsequently, we apply self-attention along the temporal dimension using standard multi-head attention transformer blocks to efficiently learn information about each agent track [9]. Our experiments show that this architecture yields significant speed advantages compared to other agent encoder structures, *e.g.,* [8]. Then, we reduce our tokens to $\mathbb{R}^{A\times D}$ via max pooling along the temporal dimension. Finally, we add learnable type embeddings $\mathbb{R}^{A\times D}$ to get our our final agent tokens. The type embeddings are used to learn categorical information about the input agents.

B. Lane Encoding

To efficiently process lane information, we encode the data of each lane segment to a lane token. Each lane segment is initially represented by a set of N points in 2D space and a masking flag ($\mathbb{R}^{L\times N\times 3}$). The points describe the shape of each lane center line w.r.t. to the origin of each lane object. We apply a mini PointNet-like [25] lane encoder, as proposed by [8], to get $\mathbb{R}^{L\times D}$ lane embeddings. Analog to agent encoding, we also add learnable type embeddings to get our final lane tokens.

C. Scene Encoding

We concatenate the agent tokens ($\mathbb{R}^{A\times D}$) and lane tokens ($\mathbb{R}^{L\times D}$) to form scene tokens ($\mathbb{R}^{A+L\times D}$). Positional embeddings are added to the scene tokens to describe the spatial relationships between different tokens in the scene. To compute the positional encoding, we project the centers

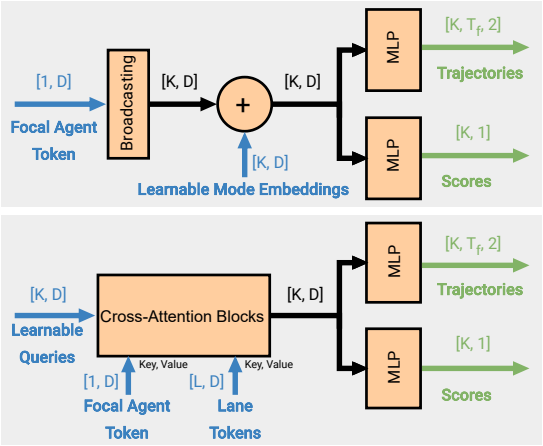

Fig. 3. Decode module architectures: simple MLP-based decoder (for EMP-M, top) and DETR-like decoder (for EMP-D, bottom).

of all tracks and lanes ($\mathbb{R}^{A+L\times 4}$) to our embedding space $\mathbb{R}^{A+L\times D}$ using two linear layers (GeLU activation function on the first and D units each). We define the center using $[x, y, \cos\alpha, \sin\alpha]$ where α is the orientation of the corresponding object. Subsequently, we encode the scene tokens using self-attention blocks (scene encoder).

D. Decoding

We experiment with two decoder architectures, shown in Figure 3. First, we use a pure multi-layer perception (MLP)-based approach to achieve a very lightweight model (EMP-M). Second, we utilize a more sophisticated transformer-based decoder (EMP-D), similar to [6], [7], [15], [9], who also use DETR-like decoders for trajectory prediction.

1) MLP-based Decoder (EMP-M): We use learnable mode embeddings $\mathbb{R}^{K\times D}$ to efficiently generate K multi-modal output trajectories for each focal agent token $\mathbb{R}^{1\times D}$. First, we broadcast the token to $\mathbb{R}^{6\times D}$, then we add our learnable embeddings. Afterwards, we use two MLPs to predict the multi-modal trajectory positions and confidence scores. Both MLPs have a single hidden layer with $2\cdot D$ units and *ReLU* activation function. For predicting the trajectories we use a linear output layer with $2\cdot T_f$ units, where T_f corresponds to the number of future time steps. For the confidence scores we use a linear output layer with K units.

2) DETR-like Decoder (EMP-D): To model multi-modality, we adopt learnable query embeddings, *i.e.,* [6], [9]. We initialize queries $\mathbb{R}^{K\times D}$ using K learnable embeddings. Then, we process the queries using multiple cross-attention blocks. In each block, we first perform cross-attention on the focal agent embeddings. Second, we execute cross-attention to the lane tokens to improve alignment with the road network. We do not apply additional cross-attention to other agent tokens [6], [15] or mode self-attention [15], thus accepting a marginally lower accuracy in favor of a lightweight architecture. Lastly, we use two MLPs to map the updated queries to the trajectories and scores. The MLPs have the same structure as in our pure MLP-based decoder.

IV. EXPERIMENTAL SETUP

A. Dataset

We conduct extensive evaluations on the motion forecasting datasets Argoverse 1 (AV1) [1] and Argoverse 2 (AV2) [2]. Both dataset include map data represented as centerline-based polylines and agent trajectory data. The sampling rate for the trajectory data is 10 Hz and for each scene a focal agent is defined. AV1 contains short sequences (5 seconds overall) which are rather simple (most vehicles are going straight-forward). The first 2 seconds are considered as historic context and the goal is to predict the following 3 seconds of the focal agent. AV2 focuses on long-term prediction: sequences are 11 seconds long, where 5 seconds are context and 6 seconds should be predicted. The AV2 dataset consists of sequences recorded in six American cities. Overall, it includes 250,000 non-overlapping scenarios, which corresponds to 763 hours of data. The dataset is randomly split into a train (199,908 sequences), validation (24,988 sequences) and test (24,984 sequences) set. AV1 is only collected in two cities, but has a slightly larger number of sequences (324,557 overall).

We follow the standard pre-processing setups: For AV1, we set the radius of interest to 65 meters around the focal agent, following [5]. For AV2, we follow Forecast-MAE [8], which uses a radius of 150 meters for data aggregation.

B. Metrics

We utilize the standard evaluation metrics of the AV 2 benchmark: miss rate (MR_K), minimum average displacement error (minADE_K), minimum final displacement error (minFDE_K) and brier minimum final displacement error (brier-minFDE_K). Each metric is computed based on a set of K trajectories predicted by the model. For $k > 1$ the best fitting trajectory (minimum L^2 distance) is used for computing the errors. The miss rate denotes the percentage of scenarios where no predicted trajectory endpoint is within a threshold distance to the ground truth endpoint (2 m for the Argoverse benchmarks). The brier-minFDE_K adds the penalty term $(1-p)^2$ to minFDE_K, where p is the confidence score of the best matching trajectory. Following the AV 2 benchmark, we report minADE_1, minFDE_1, minADE_6, minFDE_6, MR_6 and brier-minFDE_6.

C. Implementation Details

In our model, we utilize an embedding size $D = 128$. We stack four transformer blocks for our agent and scene encoders and use three attention blocks in our query decoder. In each transformer block, we use eight heads in multi-head-attention and apply normalization before the attention operation and before our feed-forward network pass.

We train our model on a single NVIDIA V100 GPU with 16 GB VRAM for 60 epochs using a AdamW [34] optimizer. We utilize a batch size of 96, which is the largest feasible value given our GPU. For the learning rate, we use 10 warm-up epochs to increase the learning rate to 0.001, before decreasing it to 0.0001 using a cosine schedule. We apply norm-based gradient clipping and execute weight decay. No

TABLE I

RESULTS ON THE ARGOVERSE 2 SINGLE AGENT FORECASTING CHALLENGE SORTED IN DESCENDING ORDER BY BRIER-MINFDE$_6$.

Method	Validation Set			Test Set					
	MR_6	$minADE_6$	$minFDE_6$	MR_6	$minADE_1$	$minFDE_1$	$minADE_6$	$minFDE_6$	brier-$minFDE_6$
FRM [28]	-	-	-	0.29	2.37	5.93	0.89	1.81	2.47
THOMAS [29]	-	-	-	0.20	1.95	4.71	0.88	1.51	2.16
EMP-M (Ours)	0.19	0.73	1.46	0.19	1.80	4.53	0.72	1.43	2.07
Forecast-MAE Scratch [8]	0.19	0.81	1.44	0.19	1.85	4.60	0.73	1.43	2.06
SIMPL [30]	-	-	-	0.19	2.03	5.50	0.72	1.43	2.05
HPTR [17]	-	-	-	0.19	1.84	4.61	0.73	1.43	2.03
BANet (Singe Model) [31]	-	-	-	0.18	1.84	4.70	0.73	1.39	2.03
Forecast-MAE Finetuned [8]	0.18	0.80	1.41	0.17	1.74	4.36	0.71	1.39	2.03
GoRela [32]	-	-	-	0.22	1.82	4.62	0.76	1.48	2.01
HeteroGCN (Single Model) [33]	-	-	-	0.18	1.79	4.53	0.73	1.37	2.00
MTR [6]	-	-	-	0.15	1.74	4.39	0.73	1.44	1.98
EMP-D (Ours)	0.18	0.71	1.39	0.17	1.75	4.35	0.71	1.37	1.98
QCNet (Single Model) [15]	0.16	0.73	1.27	0.16	1.69	4.30	0.65	1.29	1.91
MacFormer-E (Ensemble) [18]	-	-	-	0.19	-	-	0.70	1.38	1.90
ProphNet [16]	-	-	-	0.18	1.80	4.74	0.68	1.33	1.88
SEPT [9]	-	-	-	**0.14**	**1.49**	**3.70**	**0.61**	**1.15**	**1.74**

augmentations are used in training. As loss function, we sum a regression loss (Huber loss [35]) and a classification loss (cross-entropy loss). We compute the regression loss only between the ground truth trajectory and the best fitting trajectory (smallest average displacement error). Additionally, we utilize an auxiliary loss, where we predict a single trajectory for each agent in the scene [8]. We apply a linear layer $[D, 2{\cdot}T_f]$ to predict a single trajectory from each agent token and compute the Huber loss.

Additionally, we provide further timing analysis for training on a NVIDIA RTX 4090. We also compare inference latency on the AV2 dataset using NVIDIA RTX 2080 Ti and RTX 4090 GPUs. Hence, we can highlight our model effectiveness on different GPU architectures and provide a broader comparison to reference methods.

V. RESULTS AND DISCUSSION

We present detailed evaluations on the challenging AV2 single-agent prediction benchmark and the AV1 benchmark. We report results from official publications and the leaderboard[1]. To study the accuracy/resource trade-off, we compare the training times and inference latency based on own experiments and reported results.

A. Evaluation on AV2

Table I summarizes the results on the AV2 single-agent prediction task. Due to our focus on efficiency, we report scores for single models rather than ensemble results when available. Our simple model (EMP-M) using a MLP-based decoder achieves decent results on the validation and test set. It yields a brier-$minFDE_6$, which is significantly lower than, *e.g.,* THOMAS [29], and only slightly higher than Forecast-MAE [8] (without pre-training).

Using a DETR-like decoder (EMP-D), we get competitive results on the AV2 test set. EMP-D achieves a brier-$minFDE_6$ of 1.98, which is the same as the sophisticated MTR [6] and better than recent methods like GoRela [32] and SIMPL [30]. In comparison to Forecast-MAE, [8] which also shares the idea of a compact architecture, our model improves the brier-$minFDE_6$ by 0.08 (compared to Forecast-MAE without pre-training) and by 0.05 (finetuned Forecast-MAE).

A few models, *i.e.,* QCNet [15], MacFormer-E [18], ProphNet [16] and SEPT [9] outperform our approach at the expense of higher training resources and slower inference. At the time of submission, SEPT [9] ranks third on the public leaderboard, whereas the top methods have not yet been published. In Figure 4, we show some prediction results of our EMP-D model on the AV2 dataset.

B. Comparison of Training and Inference Resources

Following, we compare the required training resources and inference latency of the methods listed in Table I. For experiments on QCNet[2] and Forecast-MAE[3] we utilize the official code implementations. To date, no code implementations have been provided for the high accurate methods MacFormer, ProphNet and SEPT.

The required GPU infrastructure and training duration listed in Table II show that our model requires very small resources. In terms of the prediction accuracy, our EMP even outperforms several state-of-the-art models which require substantially more training resources. The more complex DETR-like decoder variant, EMP-D, only requires slightly more resources than EMP-M. This shows that a sophisticated decoder design enables much better scores while only slightly increasing resource demands.

Generally, better performing models utilize significantly larger training infrastructure. Only SEPT states that it also supports training on a single RTX 3090 (which is more powerful and has more VRAM than our V100). But SEPT has three times the parameters as our model and its two phase training process indicates a much longer training process overall. Unfortunately, SEPT does not report training times and no code implementation is available at the moment.

[1] https://eval.ai/web/challenges/challenge-page/1719/leaderboard/4098

[2] https://github.com/ZikangZhou/QCNet

[3] https://github.com/jchengai/forecast-mae

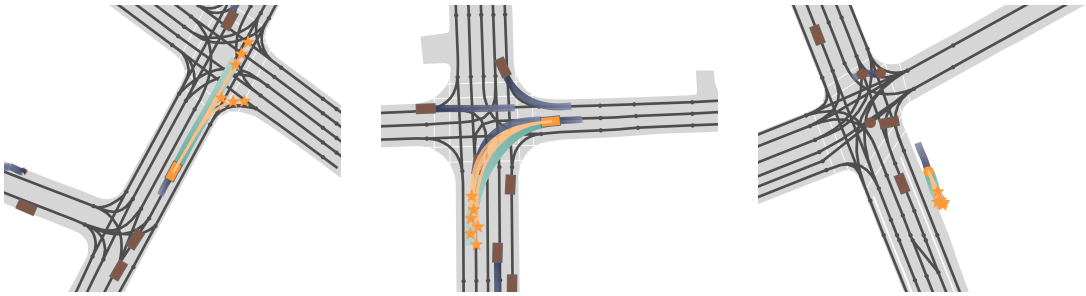

Fig. 4. Exemplary EMP-D results on AV2 for 2 vehicles (left, middle) and 1 pedestrian (right), showing the focal agent (orange), predictions (orange), ground truth (turquoise), history (blue), other agents (brown) and lane centerlines (black).

TABLE II

OVERVIEW OF MODEL SIZE AND REQUIRED TRAINING INFRASTRUCTURE FOR METHODS LISTED IN TABLE I. GPU COLUMN INDICATES TYPE OF NVIDIA GPUS USED FOR TRAINING. TRAINING TIMES ARE REPORTED ON THE AV2 DATASET. * DENOTES MEASUREMENTS DONE BY US.

Method	Params	Training Time	GPUs
ProphNet [16]	-	-	16x V100
GoRela [32]	-	-	16x T4
QCNet [15]	7.7M	-	8x RTX 3090
SIMPL [30]	1.9M	-	8x RTX 3090
MTR [6]	-	-	8x RTX 8000
MacFormer-E [18]	5x 2.5M	-	8x RTX 2080 Ti
HPTR [17]	∼15M	5-10 days	4x RTX 2080 Ti
SEPT [9]	9.6M	-	1x RTX 3090 Ti
THOMAS [29]	-	-	1x RTX 2080
Forecast-MAE [8] (Finetuned)	1.9M	$>$ 80h*	1x V100
		$>$ 40h*	1x RTX 4090
Forecast-MAE [8] (Scratch - w/o pre-train)	1.9M	$\sim$ 22.7h*	1x V100
		$\sim$ 11.5h*	1x RTX 4090
EMP-D (Ours)	3.2M	$\sim$ 12.7h*	1x V100
		$\sim$ 6.3h*	1x RTX 4090
EMP-M (Ours)	2.0M	$\sim$ 11.6h*	1x V100
		$\sim$ 6h*	1x RTX 4090

TABLE III

INFERENCE SPEED ON AV2 (LATENCY FOR BATCH PREDICTIONS OF 32 SCENARIOS). * DENOTES MEASUREMENTS DONE BY US.

GPU	Method	Latency
V100	QCNet [15]	679 ms*
	Forecast-MAE [8]	69 ms*
	EMP-D (Ours)	37 ms*
	EMP-M (Ours)	**28 ms***
RTX 2080 Ti	MacFormer-E (ensemble) [18]	94 ms
	Forecast-MAE [8]	55 ms*
	EMP-D (Ours)	33 ms*
	EMP-M (Ours)	**30 ms***
RTX 4090	QCNet [15]	318 ms*
	Forecast-MAE [8]	23 ms*
	EMP-D (Ours)	13 ms*
	EMP-M (Ours)	**11 ms***

TABLE IV

PRELIMINARY RESULTS ON THE ARGOVERSE 1 VALIDATION SPLIT.

Method	Validation Set		
	MR_6	$minADE_6$	$minFDE_6$
Forecast-MAE Scratch [8]	0.103	0.74	1.10
Forecast-MAE Finetuned [8]	0.095	0.71	1.05
EMP-M (Ours)	0.096	0.63	1.04
EMP-D (Ours)	**0.090**	**0.63**	**1.02**

We experimentally compare the train speed of our models to Forecast-MAE, which is a recent method also supporting training on a single GPU. Our EMP-D model requires only nearly half of the training time, but still outperforms the Forecast-MAE model from scratch by a large margin. Additionally, our EMP-D model even exceeds the performance of the pre-trained Forecast-MAE. Forecast-MAE would require over 60 hour for pretraining on our GPU.

Table III shows the inference speed of different methods on the AV2 dataset. We measure the latency for predicting a batch of 32 scenarios using three different GPU types: NVIDIA V100, NVIDIA RTX 2080 Ti and NVIDIA RTX 4090. On all GPUs our EMP-M ranks first and EMP-D second. Both models are almost twice as fast as Forecast-MAE. On a RTX 2080 Ti our models require less than 35 ms whereas a latency of 94 ms is reported for MacFormer-E [18]. Our approach is more than 20 times faster than QCNet [15] on V100 and RTX 4090 GPUs. SEPT [9] reports that its inference is twice as fast as QCNet, from which we can derive that our model is substantially faster than SEPT. ProphNet achieves an inference latency of 27.4 ms for predicting one scenario on a V100 GPU, but no measurements for multi-scenario settings have been provided. HeteroGCN [33] reports an inference time of 57.03 ms for batch size 1 on an RTX 2080, which also indicates a much higher latency than our model.

C. Evaluation on AV1

Finally, we present preliminary results on the AV1 dataset in Table IV. Achieving an optimal latency/accuracy trade-off requires adapting the data collection to the scenario complexity (AV1 sequences are shorter and most vehicles go straight). As we focus on AV2, which better reflects the challenges of real-world driving, we did not optimize the preprocessing or hyperparameters for AV1. Our models still outperform the most closely related Forecast-MAE despite needing less training and inference resources.

VI. CONCLUSIONS

We propose a new **e**fficient **m**otion **p**rediction (EMP) architecture based on standard transformer blocks. Our approach yields highly competitive results on the challenging AV2 dataset without incorporating prior-knowledge or using pre-training. Even with restricted training time our effective model architecture outperforms recent architectures requiring

vast resources. Additionally, we highlight the efficiency of our model by comparing inference latency on different GPU architectures. As a result, our model allows easier finetuning on new data and is also suitable for deployment to autonomous vehicles with restricted compute resources. However, a few models with significantly higher resource demands achieve better accuracy than ours. Therefore, it is important to consider the balance between required accuracy and feasibility of a model for each specific use case.

REFERENCES

[1] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, *et al.*, "Argoverse: 3D Tracking and Forecasting with Rich Maps," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[2] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays, "Argoverse 2: Next Generation Datasets for Self-driving Perception and Forecasting," in *Proc. of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[4] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, *et al.*, "Large Scale Interactive Motion Forecasting for Autonomous Driving : The WAYMO OPEN MOTION DATASET," in *Proc. of the IEEE/CVF Conference on Computer Vision (ICCV)*, 2021.

[5] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal Motion Prediction with Stacked Transformers," *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7577–7586, 2021.

[6] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Motion Transformer with Global Intention Localization and Local Movement Refinement," in *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[7] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp, "Wayformer: Motion Forecasting via Simple & Efficient Attention Networks," in *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2023.

[8] J. Cheng, X. Mei, and M. Liu, "Forecast-MAE: Self-supervised Pre-training for Motion Forecasting with Masked Autoencoders," in *Proc. of the IEEE/CVF Conference on Computer Vision (ICCV)*, 2023.

[9] Z. Lan, Y. Jiang, Y. Mu, C. Chen, S. E. Li, H. Zhao, and K. Li, "SEPT: Towards Efficient Scene Representation Learning for Motion Prediction," *arXiv preprint arXiv:2309.15289*, 2023.

[10] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "CoverNet: Multimodal Behavior Prediction Using Trajectory Sets," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[11] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction," in *Proc. of the Conference on Robot Learning (CoRL)*, 2020.

[12] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning Lane Graph Representations for Motion Forecasting," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2020.

[13] W. Zeng, M. Liang, R. Liao, and R. Urtasun, "LaneRCNN: Distributed Representations for Graph-Centric Motion Forecasting," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[14] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov, *et al.*, "MultiPath++: Efficient Information Fusion and Trajectory Aggregation for Behavior Prediction," in *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2022.

[15] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang, "Query-Centric Trajectory Prediction," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[16] X. Wang, T. Su, F. Da, and X. Yang, "ProphNet: Efficient Agent-Centric Motion Forecasting with Anchor-Informed Proposals," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[17] Z. Zhang, A. Liniger, C. Sakaridis, F. Yu, and L. Van Gool, "Real-Time Motion Prediction via Heterogeneous Polyline Transformer with Relative Pose Encoding," in *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[18] C. Feng, H. Zhou, H. Lin, Z. Zhang, Z. Xu, C. Zhang, B. Zhou, and S. Shen, "MacFormer: Map-Agent Coupled Transformer for Real-Time and Robust Trajectory Prediction," *IEEE Robotics and Automation Letters (RAL)*, 2023.

[19] G. Aydemir, A. K. Akan, and F. Güney, "ADAPT: Efficient Multi-Agent Trajectory Prediction with Adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[20] A. Vemula, K. Muelling, and J. Oh, "Social Attention: Modeling Attention in Human Crowds," in *Proc. of the International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[21] L. L. Li, B. Yang, M. Liang, W. Zeng, M. Ren, S. Segal, and R. Urtasun, "End-to-end Contextual Perception and Prediction with Interaction Transformer," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[22] J. Mercat, T. Gilles, N. El Zoghby, G. Sandou, D. Beauvois, and G. P. Gil, "Multi-Head Attention for Multi-Modal Joint Vehicle Motion Forecasting," in *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2020.

[23] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2020.

[24] A. Hassani, S. Walton, J. Li, S. Li, and H. Shi, "Neighborhood Attention Transformer," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 652–660.

[26] Z. Zhou, L. Ye, J. Wang, K. Wu, and K. Lu, "HiVT: Hierarchical Vector Transformer for Multi-Agent Motion Prediction," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[27] H. Liu, Z. Dai, D. So, and Q. V. Le, "Pay Attention to MLPs," in *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[28] D. Park, H. Ryu, Y. Yang, J. Cho, J. Kim, and K.-J. Yoon, "Leveraging Future Relationship Reasoning for Vehicle Trajectory Prediction," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2023.

[29] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "THOMAS: Trajectory Heatmap Output with learned Multi-Agent Sampling," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2022.

[30] L. Zhang, P. Li, S. Liu, and S. Shen, "SIMPL: A Simple and Efficient Multi-agent Motion Prediction Baseline for Autonomous Driving," *arXiv preprint arXiv:2402.02519*, 2024.

[31] C. Zhang, H. Sun, C. Chen, and Y. Guo, "BANet: Motion Forecasting with Boundary Aware Network," *arXiv preprint arXiv:2206.07934*, vol. 2, p. 7, 2022.

[32] A. Cui, S. Casas, K. Wong, S. Suo, and R. Urtasun, "GoRela: Go Relative for Viewpoint-Invariant Motion Forecasting," in *Proc. of the International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.

[33] X. Gao, X. Jia, Y. Li, and H. Xiong, "Dynamic Scenario Representation Learning for Motion Forecasting With Heterogeneous Graph Convolutional Recurrent Networks," *IEEE Robotics and Automation Letters (RAL)*, vol. 8, no. 5, pp. 2946–2953, 2023.

[34] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[35] P. J. Huber, "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.