# Efficient Context-Aware Graph Transformer for Vehicle Motion Prediction

Carlos Gómez-Huélamo[1], Marcos V. Conde[2], Rodrigo Gutiérrez-Moreno[1], Rafael Barea[1], Ángel Llamazares[1], Miguel Antunes[1], Luis M. Bergasa[1]

***Abstract*— Motion Prediction (MP) of multiple surrounding agents, and accurate trajectory forecasting, is a crucial task for self-driving vehicles and robots. Current techniques tackle this problem using end-to-end pipelines, where the input data is usually a Bird Eye View (BEV) HD map and the past trajectories of the most relevant agents; leveraging this information is a must to obtain optimal performance. In that sense, a reliable Autonomous Driving Stack (ADS) must produce fast predictions. However, despite many approaches use simple ConvNets and LSTMs to obtain the social latent features, State-Of-The-Art (SOTA) models might be too complex for real-time applications when using both sources of information (map and past trajectories) as well as little interpretable, specially considering the physical information. Moreover, the performance of such models highly depends on the number of available inputs for each particular traffic scenario, which are expensive to obtain, particularly, annotated High-Definition (HD) maps.**

**In this work, we propose a transformer-based model that does not rely on HD maps, but on minimal interpretable map information. The proposed model combines the powerful attention mechanisms with GNNs to model agent interactions, it has less parameters than other methods, and it is faster than most previous methods. We achieve near-SOTA results on the Argoverse Motion Forecasting Benchmark. Our code is publicly available at https://github.com/Cram3r95/mapfe4mp.**

***Keywords:*** Autonomous Driving, Motion Prediction

## I. INTRODUCTION

Autonomous Driving (AD) is a trendy research topic in academia and industry due to its real-world impact. Predicting fast and accurately the future behavior of traffic agents around the ego-vehicle is one of the key unsolved challenges in reaching full self-driving autonomy. In that sense, an ADS can be hierarchically split in the following tasks: (i) perception, identify what is around the ego-agent, then track and predict the behavious of the other agents in the scene. (ii) planning and decision-making. (iii) control and command -brake, throttle and steering angle- the agent.

Assuming the surrounding agents have been detected and tracked *i.e.* we have their past trajectories, the core task of the perception layer is Motion Prediction (MP), that is, predicting the future trajectories [1] of the surrounding traffic agents given the past on-board sensor and map information, and taking into account the corresponding traffic rules and social interaction among the agents. These predictions are typically **multi-modal** *i.e.* given the past motion of a particular vehicle and its surrounding scene, there may exist more than one possible future behaviour (also known as modes). Therefore, MP models need to cover the different choices a driver could make (e.g. going straight, turning, accelerate) as a possible trajectory in the immediate future, or in a probabilistic manner [2].

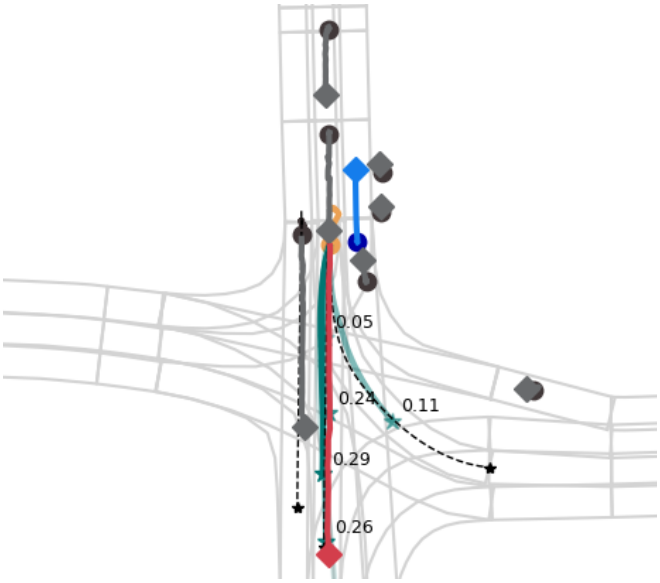


Fig. 1: Motion Prediction Scenario in **Argoverse 1** [3]. We represent: our vehicle (**ego**), the **target agent**, and **other agents**. We can also see the **ground-truth** trajectory of the target agent, our **multimodal predictions** (with the corresponding confidences) and **plausible centerlines**. Circles represent last observations and diamonds last future positions.

Traditional methods for motion forecasting [4], [5] are based on physical kinematic constraints and road map information with handcrafted rules. These approaches fail to capture the rich behavior strategies and interaction in complex scenarios, in such a way they are only suitable for simple prediction scenes and short-time prediction tasks.

The advances in Deep Learning (DL) allows us to understand and capture the complexity of a driving scenario using data-driven methods [6], [7] and achieve the most

This work has been supported from the Spanish PID2021-126623OB-I00 project, funded by MICIN/AEI and FEDER, TED2021-130131A-I00, PDC2022-133470-I00 projects, funded by MICIN/AEI and the European Union NextGenerationEU/PRTR, and ELLIS Unit Madrid funded by Autonomous Community of Madrid.

[1] Carlos Gómez-Huélamo, Rodrigo Gutiérrez-Moreno, Rafael Barea, Ángel Llamazares, Miguel Antunes and Luis M. Bergasa are with the Electronics Department, University of Alcalá (UAH), Spain. {carlos.gomezh, rodrigo.gutierrez, rafael.barea, angel.llamazares, miguel.antunes, luism.bergasa}@uah.es

[2] Marcos Conde (marcos.conde@uni-wuerzburg.de) is with the University of Würzburg, CAIDAS, Computer Vision Lab

promising *state-of-the-art* results by learning such intrinsic rules, and agent interactions. In this work we focus on attention **transformer-based** approaches such as [8], [9], [10] for context encoding and Graph Neural Networks (GNNs) to compute the most interesting social interactions.

In these models, an encoder usually takes into account: (ii) multiple-agents **history** states (position, velocity, etc.), and (ii) a High Definition **(HD) Map** [4] that includes: intersections, traffic lights and signals, multi-channel codification or complex vectorization [2], [11]. Note that obtaining and fusing this information (*e.g.* actor-to-actor, map-to-actor) is a research topic by itself [12], [11] and a core part in the AD pipeline. Here we identify a bottleneck for efficient real-time applications [4], as usually, more (complex) data-inputs implies higher model complexity and inference time [13].
Most *state-of-the-art* methods require an overwhelmed amount of information as input, specially in terms of the physical context -HD maps-, this can be inefficient in terms of latency and computation [13], [14].

In this paper, we aim to achieve accurate trajectory forecasting, yet, using light-weight transformer-based models that take as input the past trajectories of each agent, and integrate prior-knowledge about the map easily. We make the following **contributions**:

1) We propose an Efficient Context-Aware Graph Transformer model that does not rely on heavily annotated HD maps, but on minimal interpretable geometric map information.
2) Compared to other methods that use LSTM/RNN for temporal encoding and agents interaction, these networks are non-parallelizable, therefore, despite having more parameters, transformers are faster [15]. Then, our model has less parameters than other methods, and it is faster than most previous methods.
3) We achieve near-SOTA results on the Argoverse Motion Forecasting Benchmark.
4) We provide an open-source framework for MP.

## II. RELATED WORKS

One of the crucial tasks that Autonomous Vehicles (AV) must face during navigation, specially in arbitrarily complex urban scenarios, is to predict the behaviour of dynamic obstacles. In a similar way to humans that pay more attention to close obstacles and upcoming turns, rather than considering the obstacles far away, the perception layer of an AD stack must focus more on the salient regions of the scene, and the more relevant agents to predict the future behaviour of each traffic participant.

**Traditional methods** [5], [16] usually consider only physics-related factors (*e.g.* the velocity and acceleration of the target vehicle) and road-related factors (predictions must be in the proper lane), and are only suitable for **short-time** prediction tasks [5] and simple traffic scenarios, *e.g.* constant velocity in a highway (Constant Turn Rate Velocity, CTRV) where only a single path is allowed.

Recently, MP **learning-based methods** [1], [6], [17], [12], [18] have become increasingly popular since they are able not only to take into account these above-mentioned factors but also consider interaction-related factors (like agent-agent [19], agent-map [6] and map-map [11]) in such a way the algorithm can adapt to more complex traffic scenarios (intersections, sudden breaks and accelerations, etc).

Methods based on Graph Neural Networks (GNNs) [20], [11], [21] have shown very promising results.

MultiPath by Chai et al. [7] uses ConvNets as encoder and adopts pre-defined trajectory anchors to regress multiple possible future trajectories.

HOME by Gilles et al. [2], [20] presented a novel representation for multi-modal trajectory prediction, where the model takes as input the context (HD map) and history of past trajectories, and generates a 2D heatmap of the agent's possible future trajectories.

Most SOTA methods use attention mechanisms [22], [23]. In this work we focus on transformer-based approaches [9], [10] such as Liu et al. mmTransformer [8].

## III. OUR APPROACH

Considering the trade-off between curated input data and complexity, we aim to achieve competitive results on the Argoverse Benchmark [3] using (i) the agents past observations and their corresponding interactions and (ii) minimal interpretable map information in the form of most plausible centerlines around the target agent. Therefore, our model does not require full-annotated (including, topological, geometric and semantic information) HD Maps or BEV representations of the scene to compute the physical context.

We use a simple-yet-powerful map preprocessing algorithm where the target agents trajectory is initially filtered. Next, we compute the **feasible area** where the target agent can interact taking into account only the geometric information of the HD Map (lane, roads). In Fig. 2 we show an overview of our final approach.

### A. Problem Definition

We tackle the task of predicting the future positions of certain agents. Each position of the whole sequence is expressed via $x$ and $y$ coordinates in the 2D ground plane. We observe the trajectories $X_i = \{(x_i^t, y_i^t) \in \mathbb{R}^2 | t = 1, \ldots, t_{obs}\}$ of $N$ agents in the scene and the corresponding physical information of the scene (2D HD Map), observed at the timestep $t_{obs}$. Regarding the Argoverse 1 Motion Forecasting dataset, our goal is to predict the future positions $Y_i = \{(x_i^t, y_i^t) \in \mathbb{R}^2 | t = t_{obs}+1, \ldots, t_{pred}\}$ of a particular agent, also referred as the target agent. These future trajectories should be compliant with the social (i.e. traffic rules, such as right-a-way, crosswalk, left/right turning) and physical (ensuring the presence of the vehicle in the driveable area) constraints of the scene.

### B. Preprocessing

Multiple methods [11] [23] consider only the vehicles that are observable at *t=0*, handling those agents that are not observed over the full sequence spectrum (observation length = $obs_{len}$ + prediction length = $pred_{len}$) by concatenating a

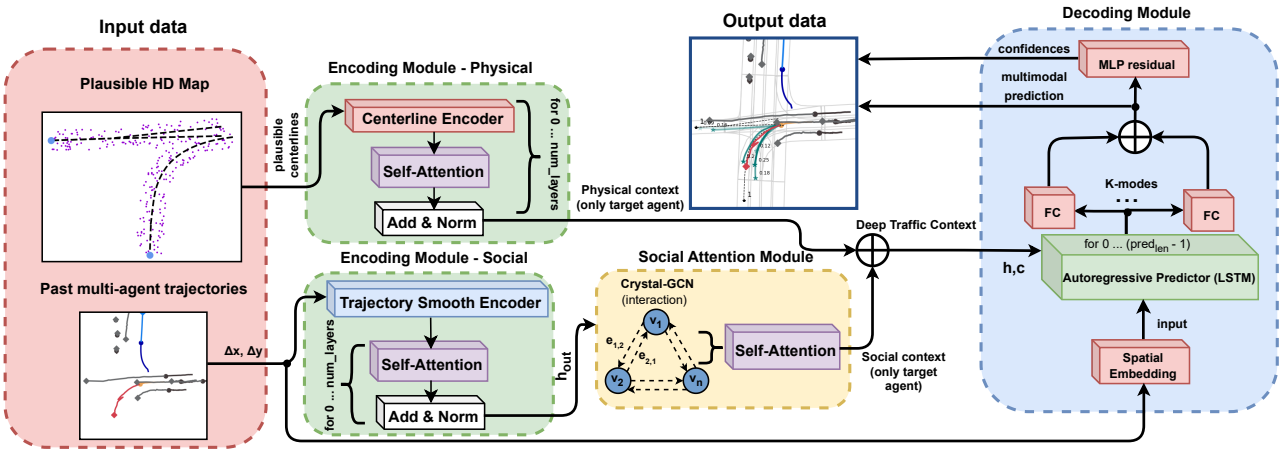


Fig. 2: Overview of our Motion Prediction pipeline. We distinguish: 1) **Encoding module**, which uses minimal HD map information (most plausible centerlines around the vehicle) and agents past observations (in terms of relative displacements) to compute the motion and physical latent features, 2) **Social Attention module**, which computes the social relationships among the different agents and returns the most relevant social features regarding the target agent, 3) Finally, the **Decoding module** calculates the multimodal predictions with their corresponding confidences using the autoregressive strategy using the past observation of the target agent as input and the concatenation of the social and physical context as hidden vector.

binary flag $b_i^t$ that indicates if the agent is padded or not. In our case, we consider the agents that have information over the full history horizon $full_{len} = obs_{len} + pred_{len}$ (*e.g.* 5s timeframe for Argoverse), reducing the number of agents to be considered in complex traffic scenarios. Instead of using 2D-BEV ($xy$ plane) local coordinates, the input for the agent $i$ is a series of relative displacements:

$$\Delta v_i^t = v_i^t - v_i^{t-1} \tag{1}$$

Where $v_i^t$ represents the state vector (in this case, $xy$ position) of the agent $i$ at timestamp $t$.

In terms of map information, we carry out the following heuristic to obtain the most relevant centerlines around a particular agent:

1) Filter the agent trajectory as a polynomial curve fitting problem by means of the Least Squares ($2^{nd}$ order) per axis and Savitzky-Golais algorithms to obtain a smooth representation of the position vector.
2) Assuming the agent is moving with a constant acceleration, we are able to calculate the subsequent derivatives (velocity and acceleration) of the target agent in $t_{obs_{len}}$.
3) In order to compute the velocity, acceleration and yaw angle in the last observation frame, we compute a weighted mean by assigning less importance (weight) to the first positions of the corresponding vector and higher importance to the latter states, in such a way immediate past observations are the key states to determine the current spatio-temporal variables of the agent.
4) We compute the future travelled distance by means of the well-known Constant Acceleration (CA) model:

$$d(t) = x_0 + vt + \frac{1}{2}at^2 \tag{2}$$

where $t$ corresponds to the prediction horizon $t_pred$, $x_0$ is equal to 0 since we want to determine the travelled distance from the current position and $v$ and $a$ are the velocity and acceleration in the last observation frame previously calculated.

5) Get all lane candidates within a bubble, given the agent last observation and Manhattan distance.
6) Expand the bubble until at least 1 lane is found.
7) Once some preliminary proposals are found, we employ the Depth First Search (DFS) algorithm to get all successor and predecessor candidates, merging the past and future candidates and removing the overlapping ones.
8) Given these raw lanes, we compute closest candidates to our current position. Then, the above-mentioned travelled distance is evaluated along the raw centerlines. We determine the end-point index $p$ of the centerline $m$ as the waypoint (each discrete node of the centerline) where the accumulated distance (considering the $\mathscr{L}_2$ distance between each waypoint) is greater or equal than the above-computed $d(obs_{len})$:

$$p \quad : \quad d(obs_{len}) \leq \sum_{p=start_{point}}^{centerline_{length}} \mathscr{L}_2(w(p+1), w(p)) \tag{3}$$

9) Finally, in order to have the same points per centerline (particularly, the prediction horizon $pred_len$), we interpolate them using a $1^{st}$ spline order, considering as start point the last agent observation and as end or goal point the aforementioned travelled distance along

the corresponding centerline, and compute the relative displacements among the different points, in a similar way to the social past trajectories. Note that if the number of proposed centerlines is lower than a predefined number $M$, a virtual centerline is created and padded with zeros.

Note that in order to enhance the generalization of the model and prevent overfitting, we follow the principles of translation and rotation invariant, since the coordinate system in our model is BEV centered of a given target agent at $t = 0$, and we use the orientation from the target location given in the same timestamp as the positive $x$-axis.

### C. Social and Map encoding

Once the map and social information are processed, the model needs to encode the spatial and temporal information of inputs by encoding them into feature vectors. In this work we focus of designing an effective encoder transformer while keeping its structure as simple and efficient as possible. In a similar way to [24], we adopt the combination of CNN/MLP, attention block and normalization, as observed in Fig. 2

In order to encode the centerlines, we first use an MLP-based encoder to transform the input vector at each time stamp ($d_i^t$, which actually represents a plausible position of the target agent) into deep features:

$$f_i^t = MLP_{\text{map}}\left(d_i^t; W_{\text{map}}\right) \tag{4}$$

where $MLP_{\text{map}}$ is a Multi-Layer (3) Perceptron with a ReLU asnon-linear layer and $W_{\text{map}}$ as the weight matrix that is learnable. However, to predict the future trajectory, the separate feature of each vector is insufficient. For example, even if two road segments in the first half have the same structure, the difference in the last half can result in a total difference in geometric meaning. Therefore, we make use of the well-established Multi-Head Self-Attention (MHSA) [15] mechanism to encode the overall set of physical features per agent as a single vector.

To be more specific, we first calculate the query, key and value matrix:

$$q_i^t = W^q f_i^t, k_i^t = W^k f_i^t, v_i^t = W^q f_i^t,$$

where $W^q, W^k, W^v$ are the learnable weight matrices. Then, we take these three matrices as the inputs of the weighting block based on softmax:

$$h_i^t = \text{softmax}\left(\frac{q_i^t \cdot k_i^{tT}}{\sqrt{d_k}}\right) v_i^t,$$

where $d_k$ is the length of matrix $k$. Finally, we adopt a 2-layer MLP to aggregate the features of vectors within a road segment:

$$h_i = MLP_{agg}\left(h_i^t; W_{agg}\right),$$

where $MLP_{agg}$ is a 2-layer MLP with a ReLU non-linear layer and $W_{agg}$ is the weight matrix that is learnable. Now, we have the feature vector for each target agent, stored as a 2D matrix $(M, H)$, where $M$ is the number of road segments and H is the length of hidden features.

For agents, we use similar techniques to encode and aggregate the information. In particular, we use a trajectory encoder block to encode each vector into the form of a feature vector. Then, similar to roads, even two vehicles have the same movement in the first half of their trajectory, and the differences in the last half of trajectories can lead to a totally different future trajectory. Therefore, we use a MHSA block to encode the overall feature of one trajectory in the observed time period and form a single feature vector for each agent. Finally, a 2-layer MLP-based aggregator is used to construct a single feature vector for each trajectory.

One aspect worth mentioning is the agent encoder. While trajectory data are, unlike roads (well structured) usually non-smooth, as expected from real-world datasets. Then, while we make use of MLP to compute the deep physical features, we use a 1D-CNN based motion encoder in the first stage due to its wider receptive field compared with MLP in such a way the convolutional encoder can smooth the trajectories and reduce the influence of noisy input trajectories.

### D. Social Attention Module

After encoding the past history of each vehicle in the sequence, we compute the agent-agent interactions to obtain the most relevant social information of the scene. For this purpose, we construct an interaction graph using Crystal-GCN [23]. Then, MHSA [15] is applied to enhance the learning of agent-agent interactions. One of the advantages of using this powerful combination (GCN + Attention) is that, unlike other methods, we do not limit nor fix the number of agents per sequence, in such a way the agent of interest only pays attention to the actual social context around it.

Before creating the **interaction mechanism**, we split the temporal information in the corresponding scenes, taking into account that each traffic scenario may have a different number of agents. The interaction mechanism is defined in [23] as a bidirectional fully-connected graph, where the initial node features $\mathbf{v}_i^{(0)}$ are represented by the latent temporal information for each vehicle $\mathbf{h}_{i,out}$ computed by the motion history encoder. On the other hand, the edges from node $k$ to node $l$ is represented as the vector distance ($\mathbf{e}_{k,l}$) between the corresponding agents at t = $obs_{len}$ in absolute coordinates, where the origin of the sequence ($x = 0, y = 0$) is represented by the position of the target at t = $obs_{len}$:

$$\mathbf{e}_{k,l} = \mathbf{v}_k^{obs_{len}} - \mathbf{v}_l^{obs_{len}}, \tag{5}$$

Given the interaction graph (nodes and edges), the Crystal-GCN, proposed by [25], is defined as:

$$\mathbf{v}_i^{(g+1)} = \mathbf{v}_i^{(g)} + \sum_{j=0: j\neq i}^{N} \sigma\left(\mathbf{z}_{i,j}^{(g)}\mathbf{W}_{\text{f}}^{(g)} + \mathbf{b}_{\text{f}}^{(g)}\right) \odot \mu\left(\mathbf{z}_{i,j}^{(g)}\mathbf{W}_{\text{s}}^{(g)} + \mathbf{b}_{\text{s}}^{(g)}\right). \tag{6}$$

This operator, in contrast to many other graph convolution operators [21], [11], allows the incorporation of edge

features in order to update the node features based on the distance among vehicles (the closer a vehicle is, the more is going to affect to a particular node). As stated by [23], we use $L_g = 2$ layers of the GNN ($g \in 0, \ldots, L_g$ denotes the corresponding Crystal-GCN layer) with ReLU and batch normalization as non-linearities between the layers. $\sigma$ and $\mu$ are the sigmoid and softplus activation functions respectively. Moreover, $\mathbf{z}_{i,j}^{(g)} = (\mathbf{v}_i^{(g)} || \mathbf{v}_j^{(g)} || \mathbf{e}_{i,j})$ corresponds to the concatenation of two node features in the $g_{th}$ GNN layer and the corresponding edge feature (distance between agents), N represents the total number of agents in the scene and $\mathbf{W}$ and $\mathbf{b}$ the weights and bias of the corresponding layers respectively.

After the interaction graph, each updated node feature $\mathbf{v}_i^{(L_g)}$ contains information about the temporal and social context of the agent $i$. Nevertheless, depending on their current position and past trajectory, an agent may require to pay attention to specific social information. To model this, we make use of a scaled dot-product Multi-Head Self-Attention mechanism [15] which is applied to the updated node feature matrix $\mathbf{V}^{(L_g)}$ that contains the node features $\mathbf{v}_i^{(L_g)}$ as rows.

Then, after computing each head separately, we combine the information of the different attention heads in a single matrix, which will represent the Social Attention Matrix **SATT** (output of the Social Attention Module, after the Crystal-GCN and MHSA mechanisms), where each row illustrates the interaction-aware feature of the agent $i$ with surroundings agents, considering the temporal information under the hood, being $\mathbf{W}_o$ / $\mathbf{b}_o$ the corresponding weight and bias of the layer that merges the different attention heads.

Regarding the Argoverse Motion Forecasting benchmark, we **only consider** the row of the final matrix that takes into account the interactions of the target agent with surrounding obstacles.

### E. Decoding Module

The decoding module is the third component of our model, as observed in Fig. 2. It consists of an LSTM network, which recursively estimate the relative displacements for the future timesteps, in the same way we represent the past observations as the difference between two consecutive timesteps in the x and y-axis. We concatenate the social context provided by the Social Interaction Module -only paying attention to the target agent row- and the latent features provided by the physical encoder, which will finally represent the deep traffic context around the target agent (its encoded trajectory, social interactions and encoded plausible future trajectories on the road, that is, the encoded centerlines).

Regarding the LSTM input, it is represented by the encoded past $n$ relative displacements of the target agent after a spatial embedding. We process the output of the LSTM using a standard Fully-Connected (FC) layer (one per mode). Once we have the relative prediction in the timestep $t$, we shift the initial past observation data in such a way we introduce our last-computed relative displacement at the end of the vector, removing the first data. We identify this technique as a *temporal decoder*, where a window of size $n$ is analized by the autoregressive decoder in contrast to other techniques [26] [19] where only the last data is considered.

Finally, after performing relative displacements to absolute coordinates operation, we obtain our multimodal predictions $\hat{Y} \in \mathbb{R}^{k \times pred_{len} \times data_{dim}}$, where $k = 6$ represents the number of modes, $pred_{len} = 30$ represents the prediction horizon and $data_{dim}$ represents the data dimensionality, in this case 2 ($xy$, predictions from the BEV perspective). Once the multimodal predictions are computed, they are concatenated and processed by a residual MLP to obtain the confidences for each trajectory.

## IV. EXPERIMENTAL RESULTS

*a) Dataset:* Large-scale annotated **datasets** have been proposed to impulse the research on the MP task. Focusing on self-driving cars, we find several *state-of-the-art* datasets with their corresponding benchmarks.

The Waymo Open Motion Prediction [27] and NuScenes Prediction [28] datasets offer thousands of real driving scenarios and exhaustive annotations. In this work, we focus on the **Argoverse** Motion Forecasting Dataset [3], which is the most frequently used dataset for MP development in the field of AD. It contains more than 300K scenarios, each traffic scenario contains a 2D BEV centroid of unique objects (so, multi-object tracked) at 10 Hz. The task is to predict the future trajectories of a particular target agent in the next 3s, given the past 2s observations in addition to the HD map features.

For training (205,942 samples) and validation (39,472 samples), full 5-second trajectories are provided, while for testing (78,143 samples), only the first 2 seconds trajectories are given.

*b) Metrics:* We evaluate the performance of our models using the standard metrics for multimodal MP [3]: (i) Average Displacement Error (**ADE**), which averages the $L_2$ distances between the ground truth and predicted output across all timesteps, (ii) Final Displacement Error (**FDE**), which computes the $L_2$ distance between the final points of the ground truth and the predicted final position. When the output is multimodal, we generate $k$ outputs (also known as modes) per prediction step and report the metrics for the best out of $k$ outputs, regarding the agent $i$.

We report results for $k = 1$ (unimodal case, only the mode with the best confidence is considered) and $k = 6$ as this is the standard in the Argoverse Motion Forecasting dataset in order to compare with other models.

### A. Implementation Details

We train our models to convergence using a single NVIDIA RTX 3090, and validate our results on the official Argoverse validation set [35]. We use Adam optimizer with learning rate 0.001, batch size 128 and linear LR Scheduler with 0.5 decay factor on plateaus. The hidden dimension in both encoding modules is 128, whilst the hidden dimension for the autoregressive prediction is 256. The social encoder first smooths the trajectories of the agents and then presents 2 layers of self-attention and normalization mechanisms, whilst

TABLE I: Results on the Argoverse 1 Motion Forecasting Leaderboard. We borrow some numbers from [3], [2], [20]. We specify the map info for each model: Raster, GNN or polyline. We indicate the error difference of our best method w.r.t top-25 SOTA methods, in centimeters. Our predictions differ w.r.t top-25 SOTA only 6cm and 10cm for the uni-modal and multi-modal minADE metric respectively, yet our model is much more efficient. We bold the best results in **black** and the second best in **blue** for each metric. Our methods are indicated with †. TP = Target Points, CB = Class Balance.

| Model | Map info | $K$=1 | | $K$=6 | |
|---|---|---|---|---|---|
| | | minADE ↓ | minFDE ↓ | minADE ↓ | minFDE ↓ |
| Constant Velocity [3] | - | 3.53 | 7.89 | | |
| Argoverse Baseline (Nearest Neighbour) [3] | - | 3.45 | 7.88 | 1.71 | 3.29 |
| Argoverse Baseline (LSTM) [3] | Polyline | 2.96 | 6.81 | 2.34 | 5.44 |
| TPNet-map-mm [29] | Raster | 2.23 | 4.70 | 1.61 | 3.70 |
| Challenge Winner: uulm-mrm (2nd) [3] | Polyline | 1.90 | 4.19 | 0.94 | 1.55 |
| Challenge Winner: Jean (1st) [22], [3] | Polyline | 1.74 | 4.24 | 0.98 | 1.42 |
| TNT [18] | GNN | 1.77 | 3.91 | 0.94 | 1.54 |
| mmTransformer [8] | Polyline | 1.77 | 4.00 | 0.84 | 1.33 |
| HOME [2] | Raster | 1.72 | 3.73 | 0.92 | 1.36 |
| LaneConv [30] | Raster | 1.71 | 3.78 | 0.87 | 1.36 |
| LaneGCN [11] | GNN | 1.70 | 3.77 | 0.87 | 1.36 |
| LaneRCNN [21] | GNN | 1.70 | 3.70 | 0.90 | 1.45 |
| GOHOME [20] | GNN | 1.69 | 3.65 | 0.94 | 1.45 |
| † Attention-based GAN [31], including CB and TP | Polyline | 1.73 | 4.07 | - | - |
| † MAPFE4MP (Social baseline) [32] | - | 1.89 | 4.19 | 1.26 | 2.67 |
| † MAPFE4MP (Map baseline) [32] | Polyline | 1.72 | 3.89 | 0.96 | 1.63 |
| **SOTA (top-10)** [20], [8], [33], [34] | | **1.57**±0.06 | **3.44**±0.15 | **0.79**±0.02 | **1.17**±0.04 |
| **SOTA (top-25)** [20], [8], [33], [34] | | **1.63**±0.08 | **3.59**±0.20 | **0.81**±0.03 | **1.22**±0.06 |
| † Ours | Polyline | 1.71 (8cm) | 3.75 (26cm) | 0.91 (10cm) | 1.49 (27cm) |

the physical encoder employs an MLP encoder in every layer (again 2) in addition to self-attention and normalization, performing feature aggregation along the hidden dimension.

In terms of the Social Interaction module, the latent vector of the Crystal-GCN layers is 128 and the number of heads in the MHSA module is $L_h = 4$. Regarding the Autoregressive predictor, we set the *window size* to 20 to match the observation length. We set the number of plausible centerlines $M$ as 3, which cover most cases (padding with zeros if needed).

The regression head is represented by $k$=6 FC layers that map the output latent vector returned by the LSTM to the final output relative displacements (dim = 2, xy). Multimodal predictions are processed by a residual MLP to obtain the corresponding confidences (similar to [11]). We refer to the aforementioned **code** for more details.

*a) Loss:* We use the Hinge (a.k.a. max-margin) and Winner-Takes-All (WTA) losses between the predicted trajectories and the ground-truth, optimizing for confidences and regressions [11]:

$$\mathscr{L} = \beta \mathscr{L}_{Hinge} + \gamma \mathscr{L}_{WTA} \tag{7}$$

Where $\beta = 0.5$ and $\gamma = 1$ initially, and can be manually adjusted during training (especially $\gamma$).

### B. Results

As we state in Section I and III, our main goal is to achieve competitive results while not using complex HD maps, and being efficient in terms of model complexity (FLOPs - Floating-Point Operations per seconds- and parameters). For this reason, we have proposed a light-weight transformer model, whose main input is the history of past trajectories of the agents, complemented by interpretable map-based features. In this section we aim to analyze our results and ablation studies, and prove the benefits of our approach for self-driving motion prediction.

The Argoverse Benchmark [35] has over 300 submitted methods, in our opinion, the top-100 submissions achieve essentially the same performance *i.e.* the standard deviation (in meters) of the ADE errors is 0.05m, meaning that there is no significant performance difference. As observed in Table **??**, best metrics are obtained by methods that employ complex graphs-based mechanisms to encode the physical information and agent-map interactions, though we achieve up-to-pair results with other *state-of-the-art* methods, specially in terms of multimodal prediction ($k = 6$).

Furthermore, in terms of **efficiency**, we find very few methods that reports efficiency-related information [20], [2], [8], [13]. Furthermore, comparing runtimes is difficult, as only a few methods provide code, and this information is missing at the Argoverse Benchmark [35]. As studied by [13], CNN-based models for processing the HD map information are able to capture social and map interactions, but most of them are **computationally too expensive** - See Table II -. We show the efficiency comparison with other relevant methods in Table II. We calculate FLOPs and parameters using a third-party library [1]. Some minor operations were not supported, yet, their contributions to the number of FLOPs were residual and ignored. The information for the other

[1]https://github.com/facebookresearch/fvcore

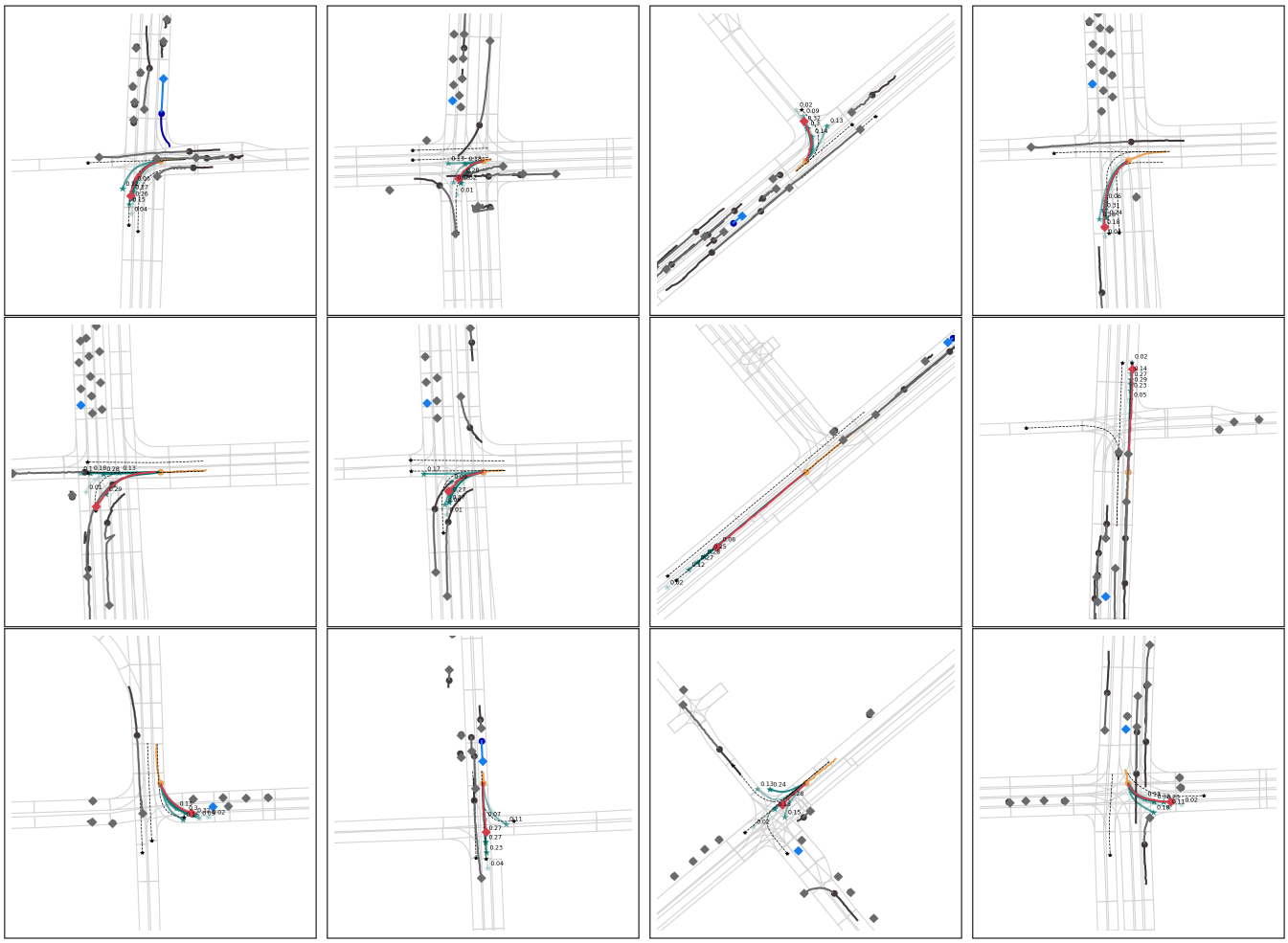

Fig. 3: Qualitative Results on challenging scenarios (intersections, sudden accelerations and breaks, etc.) using our best model. We represent: our vehicle (**ego**), the **target agent**, and **other agents**. We can also see the **ground-truth** trajectory of the target agent, our **multimodal predictions** (with the corresponding confidences) and **plausible centerlines**. Circles represent last observations and diamonds last future positions. Circles represent last observations and diamonds last future positions.

methods is consulted from [2] [20] [13] [39] [8]. To calculate the FLOPs, we follow the common practice [13] [38] [20] of fixing the number of lanes, which in our case, is limited to 3. Gao et al. [13] compares its GNN method with CNNs of different kernel sizes and map resolution to compute deep map features (decoder operations and parameters are excluded, min), demonstrating how CNN-based methods notably increase the amount of parameters and operations per second. **We do not require CNNs** to extract features from the complex HD map - see Sec. III-B. Moreover, our map prior-features are interpretable in comparison with CNNs high-dimensional outputs.

We use MHSA with a dynamic number of input agents, this typically implies a quadratic growth in complexity with the number of agents in the scene [15].

Even though our method do not obtain the best regression metrics, we achieve comparable results (Table II) against other SOTA approaches whilst our number of FLOPs is several orders of magnitude smaller than other approaches [38] [11], obtaining a good trade-off between model complexity and error (minADE, k=6).

Moreover, as it is well known in machine learning, the number of parameters is not always proportional to the inference speed. In that sense, our transformer approach also has certain benefits in comparison to LSTM/RNN temporal encoding, since these are non-parallelizable, therefore, despite having more parameters, transformers are faster [15].

We provide **qualitative** results in Fig. 3, where we show challenging scenarios with multiple agents and complex topology. It can be observed that the target agent is able to illustrate different modality predictions in terms of different directions (when facing an intersection) or in terms of different profiles (sudden break, constant velocity or sudden acceleration in a highway).

TABLE II: Efficiency comparison among SOTA methods. We show the number of parameters for each model, FLOPs, minADE (k=6) in the Argoverse test set, and runtime. Works from [13] focus on unimodal predictions (k=1). *N/A* stands for *Not Available*. Time measured on a RTX 2080 Ti (using batch 32). Some numbers from [36], [37]. We bold the best results in **black** and the second best in **blue** for each metric. Our methods are indicated with †.

| Model | # Par. (M) | FLOPs (G) ↓ | minADE (m) ↓ | Run (ms) ↓ |
|---|---|---|---|---|
| CtsConv [14] | 1.08 | 0.34 | 1.85 | 684 |
| R18-k3-c1-r100 [13] | **0.25** | 0.66 | 2.21 | *N/A* |
| R18-k3-c1-r400 [13] | **0.25** | 10.56 | 2.16 | *N/A* |
| VectorNet [13] | **0.072** | 0.41 | 1.66 | 1103 |
| DenseTNT (w/ 100ms opt.) [38] | 1.1 | 0.763 | 0.88 | 2644 |
| DenseTNT (w/ goal set pred.) [38] | 1.1 | 0.763 | 0.85 | 531 |
| LaneGCN [11] | 3.7 | 1.071 | 0.87 | 173 |
| mmTransformer [8] | 2.607 | 0.177 | **0.84** | *N/A* |
| MF-Transformer [39] | 2.469 | 0.408 | **0.82** | *N/A* |
| HOME+GOHOME [20] | 0.40 | 0.09 | 0.94 | 32 |
| MAPFE4MP (Map baseline) [32] | 0.621 | **0.047** | 0.96 | **31** |
| Ours | 1.235 | **0.038** | 0.91 | **16** |

## V. CONCLUSIONS

In this work, we propose a transformer-based model that does not rely on heavily annotated HD maps, yet it uses past trajectories and minimal map priors. The proposed method combines the transformer attention mechanisms with GNNs to model agent interactions. We show that it has less parameters than other methods, and it is faster than most previous methods. We achieve near-SOTA results on the Argoverse Motion Forecasting Benchmark while having a low computational cost compared to other state-of-the-art proposals. In future works, we plan to extend our work for multi-agent modal prediction in the Argoverse 2 dataset, taking into account more complex features and interactions in an efficienct and powerful way.Our framework is open-sourced.

## REFERENCES


[1] R. Mahjourian, J. Kim, Y. Chai, M. Tan, B. Sapp, and D. Anguelov, "Occupancy flow fields for motion forecasting in autonomous driving," *IEEE Robotics and Automation Letters*, 2022.

[2] 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)*, pp. 500–507, IEEE, 2021.

[3] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8748–8757, 2019.

[4] C. Gómez-Huélamo, L. M. Bergasa, R. Gutiérrez, J. F. Arango, and A. Díaz, "Smartmot: Exploiting the fusion of hdmaps and multi-object tracking for real-time scene understanding in intelligent vehicles applications," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 710–715, IEEE, 2021.

[5] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A survey on trajectory-prediction methods for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, 2022.

[6] S. Casas, W. Luo, and R. Urtasun, "Intentnet: Learning to predict intention from raw sensor data," in *Conference on Robot Learning*, pp. 947–956, PMLR, 2018.

[7] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," in *Proceedings of the Conference on Robot Learning* (L. P. Kaelbling, D. Kragic, and K. Sugiura, eds.), vol. 100 of *Proceedings of Machine Learning Research*, pp. 86–99, PMLR, 30 Oct–01 Nov 2020.

[8] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal motion prediction with stacked transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7577–7586, 2021.

[9] Z. Huang, X. Mo, and C. Lv, "Multi-modal motion prediction with transformer-based neural network for autonomous driving," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 2605–2611, IEEE, 2022.

[10] J. Ngiam, V. Vasudevan, B. Caine, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, *et al.*, "Scene transformer: A unified architecture for predicting future trajectories of multiple agents," in *International Conference on Learning Representations*, 2021.

[11] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *European Conference on Computer Vision*, pp. 541–556, Springer, 2020.

[12] 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," *arXiv preprint arXiv:2111.14973*, 2021.

[13] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11525–11533, 2020.

[14] R. Walters, J. Li, and R. Yu, "Trajectory prediction using equivariant continuous convolution," *arXiv preprint arXiv:2010.11344*, 2020.

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

[16] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1929–1934, IEEE, 2019.

[17] B. Ivanovic, K.-H. Lee, P. Tokmakov, B. Wulfe, R. McAllister, A. Gaidon, and M. Pavone, "Heterogeneous-agent trajectory forecasting incorporating class uncertainty," *arXiv preprint arXiv:2104.12446*, 2021.

[18] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, *et al.*, "Tnt: Target-driven trajectory prediction," *arXiv preprint arXiv:2008.08294*, 2020.

[19] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social

gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2255–2264, 2018.

[20] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Gohome: Graph-oriented heatmap output for future motion estimation," *arXiv preprint arXiv:2109.01827*, 2021.

[21] 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)*, pp. 532–539, IEEE, 2021.

[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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9638–9644, IEEE, 2020.

[23] 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," *arXiv preprint arXiv:2202.04488*, 2022.

[24] Z. Wang, J. Guo, Z. Hu, H. Zhang, J. Zhang, and J. Pu, "Lane transformer: A high efficiency trajectory prediction model," *IEEE Open Journal of Intelligent Transportation Systems*, 2023.

[25] T. Xie and J. C. Grossman, "Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties," *Physical review letters*, vol. 120, no. 14, p. 145301, 2018.

[26] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "Sophie: An attentive gan for predicting paths compliant to social and physical constraints," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1349–1358, 2019.

[27] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, *et al.*, "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9710–9719, 2021.

[28] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

[29] L. Fang, Q. Jiang, J. Shi, and B. Zhou, "Tpnet: Trajectory proposal network for motion prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6797–6806, 2020.

[30] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1468–1476, 2018.

[31] C. Gómez-Huélamo, M. V. Conde, M. Ortiz, S. Montiel, R. Barea, and L. M. Bergasa, "Exploring attention gan for vehicle motion prediction," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 4011–4016, IEEE, 2022.

[32] C. Gómez-Huélamo, M. V. Conde, and M. Ortiz, "Exploring map-based features for efficient attention-based vehicle motion prediction," *arXiv preprint arXiv:2205.13071*, 2022.

[33] 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 *2022 International Conference on Robotics and Automation (ICRA)*, pp. 7814–7821, IEEE, 2022.

[34] M. Ye, T. Cao, and Q. Chen, "Tpcn: Temporal point cloud networks for motion forecasting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11318–11327, 2021.

[35] "Argoverse benchmark." https://eval.ai/web/challenges/challenge-page/454/evaluation. Accessed: 2023-01-13.

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

[37] C. H. Prarthana Bhattacharyya and K. Czarnecki, "Ssl-lanes: Self-supervised learning for motion forecasting in autonomous driving," 2022.

[38] J. Gu, Q. Sun, and H. Zhao, "Densetnt: Waymo open dataset motion prediction challenge 1st place solution," *arXiv preprint arXiv:2106.14160*, 2021.

[39] B. He and Y. Li, "Multi-future transformer: Learning diverse interaction modes for behaviour prediction in autonomous driving," *IET Intelligent Transport Systems*, 2022.