GOHOME: Graph-Oriented Heatmap Output for future Motion Estimation

Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, Fabien Moutarde

GOHOME: Graph-Oriented Heatmap Output for future Motion Estimation

Thomas Gilles[1,2], Stefano Sabatini[1], Dzmitry Tsishkou[1], Bogdan Stanciulescu[2], Fabien Moutarde[2]

***Abstract*— In this paper, we propose GOHOME, a method leveraging graph representations of the High Definition Map and sparse projections to generate a heatmap output representing the future position probability distribution for a given agent in a traffic scene. This heatmap output yields an unconstrained 2D grid representation of agent future possible locations, allowing inherent multimodality and a measure of the uncertainty of the prediction. Our graph-oriented model avoids the high computation burden of representing the surrounding context as squared images and processing it with classical CNNs, but focuses instead only on the most probable lanes where the agent could end up in the immediate future. GOHOME reaches $2nd$ on Argoverse Motion Forecasting Benchmark on the MissRate_6 metric while achieving significant speed-up and memory burden diminution compared to Argoverse 1^{st} place method HOME. We also highlight that heatmap output enables multimodal ensembling and improve 1^{st} place MissRate_6 by more than 15% with our best ensemble on Argoverse. Finally, we evaluate and reach state-of-the-art performance on the other trajectory prediction datasets nuScenes and Interaction, demonstrating the generalizability of our method.**

I. INTRODUCTION

Trajectory prediction inherently faces many uncertainties. These uncertainties can be split in two categories: aleatoric and epistemic. Aleatoric uncertainty is the natural randomness of a process: it is the consequence of control noise and will lead to variations in acceleration, curvature, etc ... It translates into a spread of the possible future position and is often tackled by the use of Gaussian predictions in motion estimation [1], [2]. Epistemic uncertainty outlines some knowledge that can't be known by the observer at prediction time: what is the car destination, will it choose to overtake the car by the left or stay behind ? Recent methods use multimodal outputs based on anchors [3], [4], predefined learning heads [5] or the available HD-Map [6]–[9] to cover the span of these possible manoeuvres.

However, existing methods trying to deal with the aforementioned uncertainties in trajectory prediction have limitations. Gaussian predictions are constrained to a 2D predefined shape, that cannot adapt easily to the specific road context, for example at high speed on a curvy road the distribution of the future agent position should resemble the center lane curvature. Regressed sets of coordinates may encounter mode collapse and converge to the same solution, as they are trained with only one ground truth per sample. On the other hand, anchor-based and map-based predictions are restricted to a predefined set of possibilities, depending on preprocessed trajectory clustering or a fixed sampling of the centerlines.

Motion forecasting can also be tackled through the use of a heatmap output representing the final trajectory point location distribution [10], [11]. The intermediary waypoints can then be reconstructed from the history and end point. This brings many advantages for uncertainty modelization and multimodal prediction. First of all it does not restrict the representation of the prediction uncertainty to a parametric form (like a gaussian). Moreover, it conveys a richer information regarding the future probability distribution compared to a predefined number of predicted trajectories, enabling predictions with a much better coverage . This is usually achieved rasterizing an image to represent the context around the car, and processing it through an encoder-decoder CNN. However, the distance a car can travel in a given time horizon can exceed this image boundary, and extending the reach

[1]IoV team, Paris Research Center, Huawei Technologies France
[2]MINES ParisTech, PSL University, Center for robotics
Contact: thomas.gilles@mines-paristech.fr

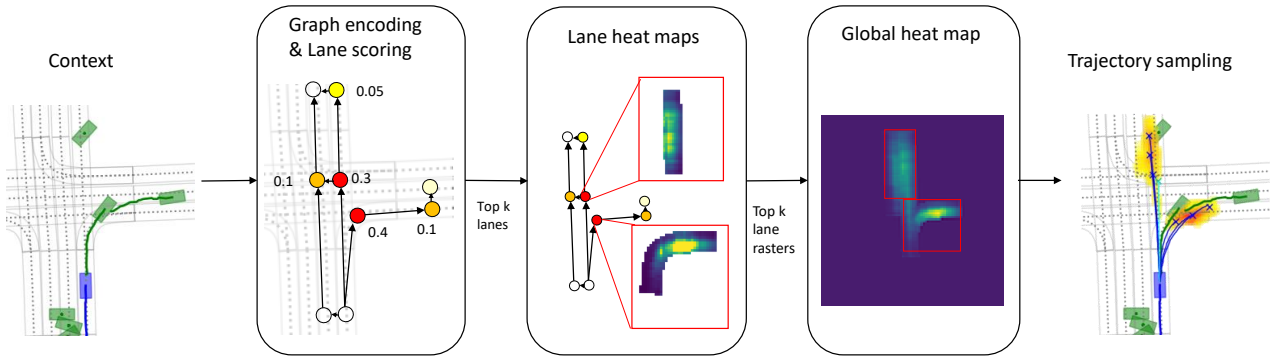

Fig. 1: GOHOME pipeline. The lane graph extracted from the HD-Map is processed through a graph encoder. Each lane then generates a local curvilinear raster that is combined into a predicted probability distribution heatmap.

results in quadratic complexity increase. Moreover, convolutional networks commonly used for the task of generating images have to operate on the full square image while the actual road and drivable area take a much sparser space.

HOME [10] introduces the use of a probability heatmap as model output for car trajectory prediction, but uses a fully-convolutional model and is limited to a restricted image size . We iterate on this work and present an optimized motion forecasting framework solely based on graph operations to provide efficiently an heatmap with uncertainty measure exploiting the vectorized form of HD map. We also highlight that heatmap output is suitable for model ensembling without any risk of mode collapse and bring significant improvement to the state-of-the-art using this ensembling. Our GOHOME pipeline is illustrated in Fig. 1.

II. RELATED WORK

The sequential nature of temporal trajectories makes them a straightforward application of recurrent neural networks [2], [12]. However, the need for local map and context information leads them to be often combined with Convolution Neural Networks (CNNs) applied on top-view images [5], [13]–[15].

Lately, Graph Neural Networks (GNNs) have been increasingly applied in order to process compact map encodings with a deepened connectivity understanding. VectorNet [16] treats indifferently trajectory and map lanes as sets of points (polylines), and encodes them into a global interaction graph. LaneGCN [17] uses graph convolutions onto the connected lane graph before fusing this lane information with actor information.

While most previous works tackle multimodality through learned regression heads, recent work brought different output representations in order to avoid mode collapse and sample inefficiency. CoverNet [4] and MultiPath [3] use anchor trajectory priors to have identified modes without risk of averaging them. PRIME [18] generates model-based trajectories and then ranks them with a learned model. TNT [6] samples target candidates along lane priors and scores them using a VectorNet backbone, while LaneRCNN [7] generates a lane graph for each actor and uses the lane nodes as a classification output for future position. GoalNet [19] identifies possible long-term goals proposals with the map, and runs a GNN where the features for each possible goal are a path-relative raster.

Generative methods can also be used to obtain multimodal predictions, through either Variational Auto Encoders [13], [14], [20], [21] or Generative Adversarial Networks [22], [23], but they require multiple forward passes for each prediction, do not guarantee diversity in the samples obtained and their inherent randomness in not advisable in production systems..

Other methods focus on a heatmap output to represent the future distribution, as it possesses natural multimodality and is therefore not subject to mode collapse. Mangalam *et al.* [11] models both long-term goals and intermediary waypoints in the two-dimensional space as an image for pedestrian trajectory forecasting, combined with random sampling and Kmeans clustering. HOME [10] predicts a future probability heatmap for car trajectories, and devises deterministic sampling algorithms for various metric optimization. However, most of these heatmap-based methods use a full CNN architecture. To our knowledge, we are the first work to combine a GNN architecture and a heatmap output without the use of any CNNs.

III. METHOD

The goal is to output a heatmap that represents the position of an agent at a given time in the future. The trajectory is then regressed conditioned to the final end point. To achieve this, our GOHOME system focuses on lane-level operations as illustrated in Fig. 1. The local HD-Map is provided in the dataset as a graph of L lanelets. A lanelet represents a macro section of the road (10 to 20 meters on average), as our goal is to encode connectivity at a macro level (lane segments), and not micro level (every meter). Each lanelet is defined as a sequence of centerline points, and is connected to its predecessor, successor, left and right neighbors if they exist. We encode each lanelet into a road graph, where geometric and connectivity information are represented. Our model yields a score for each of these lanelets, that is used to identify most probables lanes. A partial heatmap is then generated for the top ranking lanelets, and projected onto a global heatmap. Afterwards, we sample a set of endpoints from the heatmap and recreate a trajectory for each.

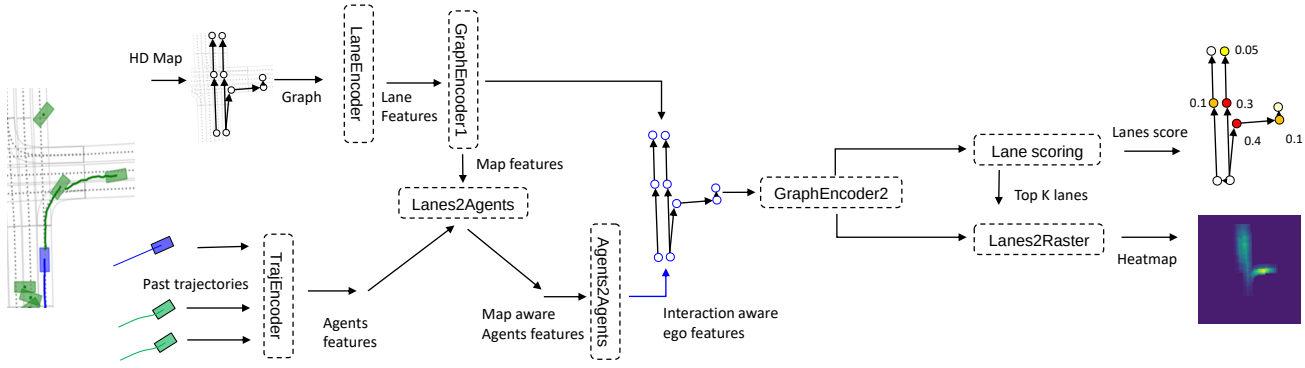

Fig. 2: GOHOME model architecture

A. Graph neural network for HD-Map input

The model architecture is illustrated in Fig. 2. We encode each lanelet through a shared 1D convolution and UGRU [24], [25] recurrent $LaneEncoder$ into features F of C channels. The lanelet features F are then updated through a $GraphEncoder1$ made of a sequence of four graph convolution operations similar to [17] in order to spread connectivity information:

$$F \leftarrow FW + \sum_r A_r F W_r \tag{1}$$

where F is the (L, C) lane feature matrix , W is the learned (C, C) weight for ego features encoding. A_r and W_r are the respective adjacency matrix (L, L) and learned weight (C, C) for the relation $r \in \{predecessor, successor, left, right\}$ derived from the lane graph. A_r is fixed as it comes from the HD Map, while W_r enables to learn different operations for each possible relation.

Parallely, each agent trajectory, defined as a sequence of position, speed and yaw, is encoded with a shared $TrajEncoder$ also made of a shared 1D convolution and a UGRU layer. Each agent feature is then updated with map information through a cross-attention $Lanes2Agents$ layer on the lanelet features. Interactions between agents are then taken into account through a self-attention $Agents2Agents$ layer between agents. Finally the target agent feature is concatenated to all the lanelet features by $Ego2Agents$ and then treated through a final $GraphEncoder2$ layer, also made of 4 graph convolutions to obtain the final graph encoding that will be used to generate the different predictions.

Compared to other methods using graph neural networks, our method uses graph convolutions like LaneGCN [17] and LaneRCNN [7], but applies them to lanelets instead of lane nodes (a lane node is a single point in the sequence of a lanelet). VectorNet [16] and TNT [6] also use lanelets, called polylines, but connect them through global attention instead of using graph connectivity. We chose to use a GNN on the lanelets since we wanted an efficient and high-level representation allowing to spread information easily through the graph, while still leveraging connectivity.

B. Heatmap generation through Lane-level rasters

For the heatmap output, we wish to have a dense image in cartesian coordinates of dimensions (H, W). To do so without using any convolution on the full image, we create a raster for every lanelet in curvilinear coordinates. We use lane ranking to generate these lane rasters only for the top k lanelets and not all of them.

a) Lane raster generation: Each of the small lane rasters of size (h, w) has a longitudinal lenght of 20m and a transversal width of 4m. These lane rasters are created as a discretization of the Frenet-Serret referential along the lane , as illustrated in Fig. 3. We decompose the probability distribution along a lanelet in a longitudinal component $(h, 1, 8)$ and a lateral component $(1, w, 8)$ predicted from the lanelet encoding. These components are summed together with broadcast to create a $(h, w, 8)$ $R_{features}$ volume. This way the complexity to create the volume is $(h+w)\times 8$ instead of $h \times w \times 8$. The obtained volume is then concatenated with pixelwise cartesian coordinates, heading, lane occupancy and curvature informations before a final linear layer on which is applied a sigmoid to get the R_{proba} output.

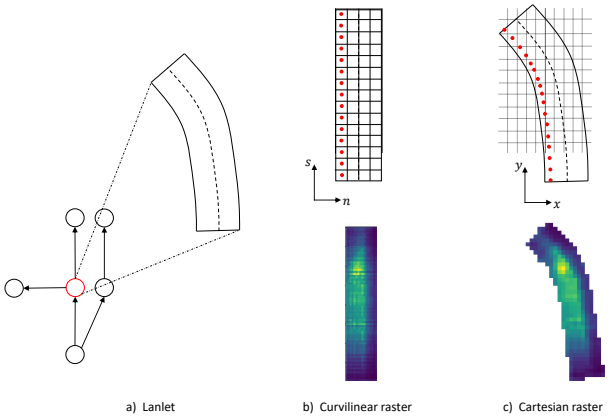

Fig. 3: Lane raster grid projection onto cartesian coordinates. a) A single node of the graph is a lanelet and describes a road segment. b) A rectangular raster is generated along the curvilinear coordinates of the lanelet. c) The lanelet coordinates are then used to project the predicted raster back into cartesian coordinates to complete the final heatmap output.

The resulting lane-level R_{proba} heatmaps are then projected onto the full cartesian heatmap $\hat{Y}$ using each pixel cartesian coordinates as illustrated in Figure 3. If multiple lane-level pixels fall into the same cartesian pixel, their values are averaged. The lane raster widths are set such that adjacent lane rasters overlap and can cover lane change behaviors. The target Y for this final prediction $\hat{Y}$ is a Gaussian centered around the target agent ground truth position. We use the same pixel-wise focal loss as [10] inspired from CenterNet [26]:

$$L = -\frac{1}{P}\sum_p (Y_p - \hat{Y}_p)^2 f(Y_p, \hat{Y}_p)$$
$$\text{with } f(Y_p, \hat{Y}_p) = \begin{cases} \log(\hat{Y}_p) & \text{if } Y_p{=}1 \\ (1-Y_p)^4 \log(1-\hat{Y}_p) & \text{otherwise} \end{cases} \tag{2}$$

b) Lane ranking: The lanelet classification is obtained with a linear layer on the graph encoding, followed by a sigmoid activation. The ground-truth is defined by a 1 for all lanelets where the future car position is inside the lanelet polygon and 0 otherwise. The loss is a binary cross-entropy added to the pixel-wise loss of Eq. 2 with a $1e^{-2}$ coefficient. Since only a fraction of the lanelets will actually be useful to represent the future car location, we can compute the lane-level rasters only for a subselection of lanelets, saving more computation. We use the classification score c_{lane} predicted by the network to select only the top k ranking lanelets, and

only compute and project the lane raster for these. Since the raster predictions for the other unselected lanelets would have been very close to zero anyway, this does not decrease performance at all, as demonstrated in Sec. IV-C.2.

c) Cartesian image connection: Some lane rasters may be projected onto the same pixels and overlap, which is very difficult for the model to know of beforehand. To help the model know of overlaps and propagate location information through the lane rasters, we do a first projection of the lane rasters onto the cartesian coordinates before the final probability estimation. The $(h, w, 8)$ lane raster features $R_{features}$ are projected onto a $(H, W, 8)$ cartesian image $I_{features}$ through the same operation previously described for the probability heatmaps, with the overlaps averaged. The occupancy (number of raster pixels aggregated in each cartesian pixel) information is also concatenated to the volume $I_{features}$. A linear layer is then applied on the last dimension of $I_{features}$, which is then reprojected onto curvilinear coordinates and concatenated to the initial lane rasters $R_{features}$ before final probability estimation and projection. This way, the features of overlapping lanes are shared between them so that they can propagate information and homogenise probability.

C. Sparse sampling for Miss Rate Optimization and Full trajectory generation

To derive final trajectory points from the heatmap, we use the same MissRate optimization sampling algorithm as HOME [10]: we iteratively select the grid point that maximizes surrounding probability in a local neighborhood of radius r, then set this local neighborhood probabilities to zero so the next iteration doesn't select the same location. The radius r is a simple hyper-parameter that can be tuned according to the spread of uncertainty present in the dataset, as we will demonstrate in Sec. IV-B.

Full trajectories are then inferred from the sampled end points with a simple fully connected model of 2 hidden layers with 64 features each, taking the history and end point as inputs and trained with the ground truth end points.

D. Model ensembling

Because of the multimodal nature of the predictions, model ensembling is usually tedious in trajectory prediction, as it is not possible to determine which modality should be averaged with which, and even shortest distance matching doesn't guarantee that two predictions highlight the same decision and would make sense averaged together. On the other hand, probability heatmap are a great way of representing information coming from different sources or models in a common system of reference and can be averaged together without any assumption nor risk of mode collapsing.

IV. Experimental Results

A. Experimental settings

a) Datasets: The Argoverse Motion Forecasting Dataset [41] is made of 205942 training samples, 39472 validation samples and 78143 test samples, with 2 seconds history, and 3 seconds future sampled at 10Hz. The NuScenes Prediction dataset [42] is made of 32186 training samples and 9041 validation samples, with 2 seconds history and 6 seconds future sampled at 2Hz. The Interaction dataset is made of 447626 training samples and 130403, with 1 seconds history and 3 seconds future trajectory sampled at 10Hz. All datasets provide the local HD-Map as a lanelet graph.

b) Metrics: The metrics commonly used by both datasets are the MR_k and minFDE_k, which are respectively the Miss Rate and the minimum Final Displacement Error for the top k predictions, as well as the minimum Average Displacement Error minADE_k. Following their respective leaderboards, we report these metrics for k=1,5 for Argoverse, k=1,5,10 for NuScenes and k=1,6 for Interaction. All metrics are the lower the better.

c) Implementation details: All models are trained with batchsize 32 and Adam optimizer. Trainings last 16 epochs for validation evaluation and 32 epochs for test evaluation. The initial learning rate is $1e^{-3}$ and is divided by 2 at epochs 3, 6, 9 and 13. All layers have 64 channels, graph convolution and attention layers are followed by Layer Normalization [43]. ReLU activation is used after every layer. We upsample the HD-Maps lanelets to obtain an average lenght of 10m per lanelet.

The architecture is the same for all datasets, with the exception of the sampling radius r that we can tune according to the uncertainty spread of the dataset and the metrics we want to optimize. The default sampling radius we use

TABLE I: Argoverse Leaderboard [27]

	K=1		K=6		
	minFDE	MR	minADE	minFDE	MR
LaneGCN [17]	3.78	59.1	0.87	1.36	16.3
TPCN [28]	3.64	58.6	**0.85**	1.35	15.9
Jean [2]	4.24	68.6	1.00	1.42	13.1
SceneTrans [29]	4.06	59.2	**0.80**	**1.23**	12.6
LaneRCNN [7]	3.69	56.9	0.90	1.45	12.3
PRIME [18]	3.82	58.7	1.22	1.56	11.5
DenseTNT [30]	3.70	59.9	0.94	1.49	10.5
HOME [10]	3.65	57.1	0.93	1.44	9.8
GOHOME	3.65	57.2	0.94	1.45	10.5
HO+HO	3.57	56	0.92	1.41	9.4
HO+GO	**3.53**	**55.8**	0.92	1.40	9.1
Best ensemble	3.68	57.2	0.89	1.29	**8.5**

TABLE II: NuScenes Leaderboard [31]

	K=5		K=10		k=1
	minADE	MR	minADE	MR	minFDE
CoverNet [4]	1.96	67	1.48	_	_
Trajectron++ [32]	1.88	70	1.51	57	9.52
ALAN [8]	1.87	60	1.22	49	9.98
SG-Net [33]	1.86	67	1.40	52	9.25
WIMP [34]	1.84	55	**1.11**	43	8.49
MHA-JAM [35]	1.81	59	1.24	46	8.57
CXX [36]	1.63	69	1.29	60	8.86
LaPred [9]	1.53	_	1.12	_	8.12
P2T [37]	1.45	64	1.16	46	10.50
GOHOME (r=2.6m)	**1.42**	57	1.15	47	**6.99**
GOHOME (r=1.8m)	1.59	**46**	1.15	**34**	7.01

TABLE III: Interaction Validation

	minFDE_1	minFDE_6
TNT [6]	–	0.67
HEAT-I-R [38]	0.66	–
ReCoG [39]	0.65	–
ITRA [40]	–	0.49
GOHOME	**0.61**	**0.45**

TABLE IV: Performance/Complexity comparison

Model	K=6		#Param	FLOPs
	minFDE	MR		
HOME	1.28	6.8	5.1M	4.8G
GNN-HOME	1.28	7.2	0.43M	0.81G
GOHOME	1.26	7.1	0.40M	0.09G

TABLE V: Lane ranking speed-up

# lanes	K=6		FPS
	minFDE	MR	
All	1.28	7.5	17
20	1.26	7.1	34
10	1.26	7.3	45

for Argoverse is 1.8m, slightly less than the 2m threshold defining the MissRate. Since the nuScenes dataset has a longer prediction horizon, it is more uncertain and therefore a larger radius of 2.6m is required to optimiwe the minADE_5, however MissRate remains better for a lower radius of 1.8m. Finally, since Interaction perception data comes from drones, it is much less noisy and therefore generates much more focused probability predictions which we sample with a radius of 1.4m.

B. Comparison with State-of-the-art

We report our results on the online Argoverse test set in Tab. I, on the online NuScenes leaderboard in Tab. II, and on the Interaction validation set in Tab. III. We compare it to the published methods on each of these benchmarks. On Argoverse, our GOHOME method reaches 2^{nd} place in MR_6, with the use of a lighter and faster model than 1^{st} HOME as will be showed in Sec. IV-C. On NuScenes and Interaction, GOHOME ranks first in multiple metrics as well.

a) Ensembling for increased performance and highlight of model differences: We also report the results of our ensembled models on the test set. We first highlight that the ensemble of two similar HOME models (HO+HO) brings significant improvement compared to HOME alone. As a general rule, the more different and complementary two models are, the greater the performance increase will be. We notice that the combination of HOME and GOHOME models (HO+GO) brings a greater improvement than HO+HO, despite each HOME model being better in single performance than the GOHOME model. Our best ensembling, a weighted combination of 9 HOME and GOHOME models, allows us to improve on the existing state-of-the-art by a significant margin, with a more than 15% MR_6 decrease.

C. Ablation studies

We highlight the gains made by replacing convolution operations with graph operations. To measure inference time, we use a batchsize of 16, which can be considered as an average number of agents to be predicted at a given time. We report only the model forward pass, omitting preprocessing and postprocessing, but notice that image preprocessing is sensibly slower, particularly because of the rasterization of the different semantic layers. All times are measured with a Nvidia 2080 TI. While we report inference time, we also notice that training times record an even greater difference. We mostly consider three different architectures: HOME, GNN-HOME which is a modified HOME model with a GNN encoder but the usual CNN decoder, and our new method GOHOME. All numbers are reported on the Argoverse validation set.

1) Graph operation speed-up: We evaluate the speed-up gained by using graph encoding and lane rasters instead of full convolutions in Tab. IV. The CNN encoding and decoding corresponds to the full HOME model, and the GNN with Lane Rasters (LR) to the GOHOME model. We also estimate the impact from the encoding separately by testing a model with GNN encoding and CNN decoding. We report FLOPs, number of parameters and Frame per Seconds. We measure an average number of 140 lanelets and 10 agents per sample to compute FLOPs.

2) Trade-off from lane ranking: We show in Tab. V the effects of only selecting the top k lanes to extract rasters. We

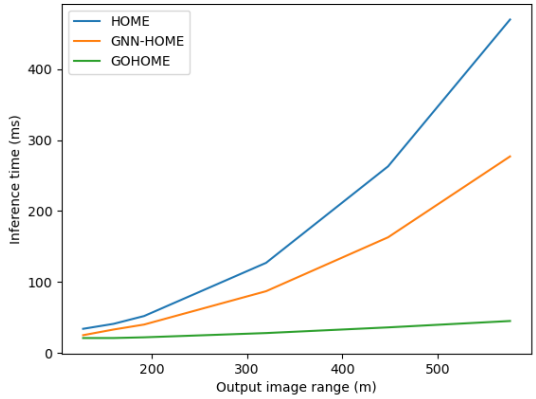

Fig. 4: Inference time with regard to output range

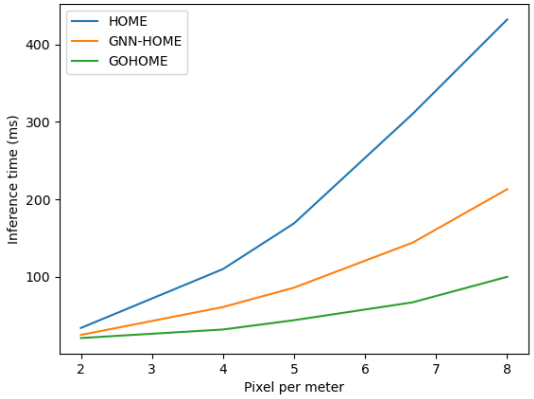

Fig. 5: Inference time with regard to pixels per meters

fix an input range of 128 and output range of 192 with 0.5m x 0.5m pixel resolution. We observe that this ranked selection doesn't decrease performance, as limiting the number of projected lanes seems to actually improve the metrics, and brings effective speed-up.

3) Image size and resolution scaling: While a 192m image range, which amount to a 88m reach in each direction, may be sufficient in most urban driving predictions with a time horizon of 3s, other datasets can require predictions up to 6 or 8 seconds [42], [44]. There is therefore a need to increase this output range, which can be done without necessarily increasing the input size, as far distances are reached with long straight trajectories that can be easily extrapolated on highways.

We compare the scaling of our graph-based GOHOME model to the one of an image-based HOME model with regard to output range and resolution. Fig. 4 highlights the output range scaling, where we use a fixed input range of 128 meters, and a resolution of 0.5m per pixel. Whereas the CNN decoders of HOME and GNN-HOME lead to a quadratic scaling, the lane rasters combined with the top 20 lane ranking enable a scaling that is even less than linear.

We show in Fig. 5 the inference time with regard to the number of pixels per meter, which is the inverse of the resolution. While efficient optimization of convolution and constant costs lead to close inference times for the initial 2 pixels per meter, the more efficient scaling shows clearly for GNN inputs and especially Lane Rasters outputs. As the resolution of the lane rasters is also scaled with the total resolution, the quadratic complexity is still applied, but with a much lesser coefficient that allows for realistic training and inference times for finer resolutions.

D. Qualitative results

We show in Fig. 6 some qualitative results of our GOHOME model. The lane prediction displayed on top can be assimilated to the representation of epistemic uncertainty, as the choice of where the driver will decide to go, whereas the spread of the final heatmap modes models aleatoric uncertainty in the trajectory controls. We observe that the model assign different modes for each lane possibility, and that each of these modes is well aligned with the corresponding lane with a spread along the curvilinear direction.

V. Conclusion

In this paper, we propose GOHOME, a trajectory prediction framework generating a global heatmap probability distribution without the use of any image based convolution. Through the use of graph operations, ranking and projections, our model reaches state-of-the-art performance on three datasets with great scaling with regards to the predicted range and resolution.

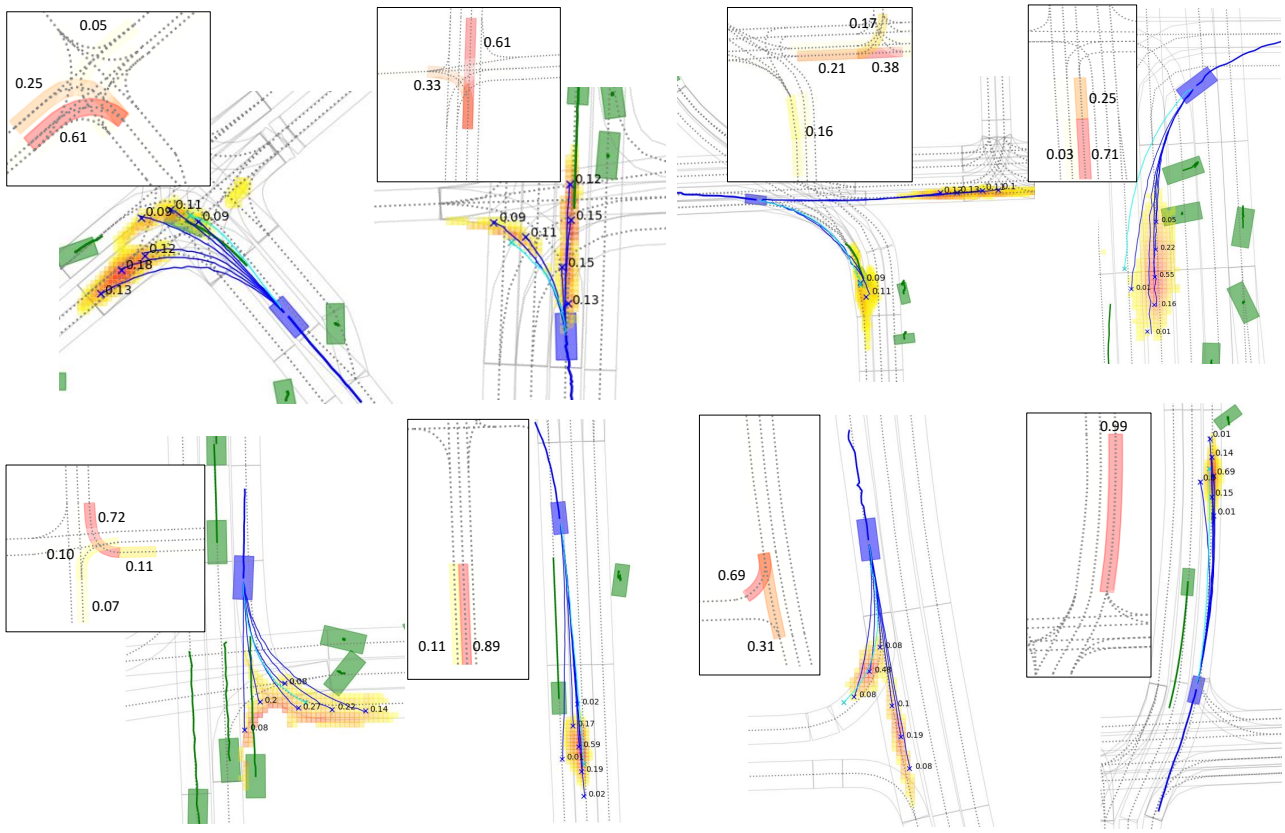

Fig. 6: Qualitative examples of GOHOME output. Graph lane classification is shown in framed inserts

References

[1] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," in *IV*, 2018.

[2] 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 *ICRA*, 2020.

[3] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," in *CoRL*, 2020.

[4] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," in *CVPR*, 2020.

[5] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *ICRA*, 2019.

[6] 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," *CoRL*, 2020.

[7] W. Zeng, M. Liang, R. Liao, and R. Urtasun, "Lanercnn: Distributed representations for graph-centric motion forecasting," *arXiv:2101.06653*, 2021.

[8] S. Narayanan, R. Moslemi, F. Pittaluga, B. Liu, and M. Chandraker, "Divide-and-conquer for lane-aware diverse trajectory prediction," in *CVPR*, 2021.

[9] B. Kim, S. H. Park, S. Lee, E. Khoshimjonov, D. Kum, J. Kim, J. S. Kim, and J. W. Choi, "Lapred: Lane-aware prediction of multi-modal future trajectories of dynamic agents," in *CVPR*, 2021.

[10] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Home: Heatmap output for future motion estimation," *arXiv preprint arXiv:2105.10968*, 2021.

[11] K. Mangalam, Y. An, H. Girase, and J. Malik, "From goals, waypoints & paths to long term human trajectory forecasting," *arXiv:2012.01526*, 2020.

[12] F. Altché and A. de La Fortelle, "An lstm network for highway trajectory prediction," in *ITSC*, 2017.

[13] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *CVPR*, 2017.

[14] Y. C. Tang and R. Salakhutdinov, "Multiple futures prediction," in *NeurIPS*, 2019.

[15] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal motion prediction with stacked transformers," *arXiv preprint arXiv:2103.11624*, 2021.

[16] 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 *CVPR*, 2020.

[17] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *ECCV*, 2020.

[18] H. Song, D. Luan, W. Ding, M. Y. Wang, and Q. Chen, "Learning to predict vehicle trajectories with model-based planning," *arXiv:2103.04027*, 2021.

[19] L. Zhang, P.-H. Su, J. Hoang, G. C. Haynes, and M. Marchetti-Bowick, "Map-adaptive goal-based trajectory prediction," in *CoRL*, 2020.

[20] N. Rhinehart, K. M. Kitani, and P. Vernaza, "R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *ECCV*, 2018.

[21] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It is not the journey but the destination: Endpoint conditioned trajectory prediction," in *ECCV*, 2020.

[22] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *CVPR*, 2016.

[23] 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 *CVPR*, 2019.

[24] A. Erdem, *6th place solution: Very custom gru*, `www.kaggle.com/c/riiid-test-answer-prediction/discussion/209581`.

[25] R. Rozenberg, J. Gesnouin, and F. Moutarde, "Asymmetrical bi-rnn for pedestrian trajectory encoding," *arXiv:2106.04419*, 2021.

[26] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv:1904.07850*, 2019.

[27] *Argoverse motion forecasting competition*, `https://eval.ai/web/challenges/challenge-page/454/leaderboard/1279`, Accessed: 2021-06-15.

[28] M. Ye, T. Cao, and Q. Chen, "Tpcn: Temporal point cloud networks for motion forecasting," in *CVPR*, 2021.

[29] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, *et al.*, "Scene transformer: A unified multi-task model for behavior prediction and planning," *arXiv:2106.08417*, 2021.

[30] J. Gu, C. Sun, and H. Zhao, "Densetnt: End-to-end trajectory prediction from dense goal sets," in *ICCV*, 2021.

[31] *Nuscenes prediction competition*, `https://eval.ai/web/challenges/challenge-page/591/leaderboard/1659`, Accessed: 2021-09-14.

[32] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *ECCV*, 2020.

[33] C. Wang, Y. Wang, M. Xu, and D. J. Crandall, "Stepwise goal-driven networks for trajectory prediction," *arXiv:2103.14107*, 2021.

[34] S. Khandelwal, W. Qi, J. Singh, A. Hartnett, and D. Ramanan, "What-if motion prediction for autonomous driving," *arXiv:2008.10587*, 2020.

[35] K. Messaoud, N. Deo, M. M. Trivedi, and F. Nashashibi, "Multi-head attention with joint agent-map representation for trajectory prediction in autonomous driving," *arXiv:2005.02545*, 2020.

[36] C. Luo, L. Sun, D. Dabiri, and A. Yuille, "Probabilistic multi-modal trajectory prediction with lane attention for autonomous vehicles," *arXiv:2007.02574*, 2020.

[37] N. Deo and M. M. Trivedi, "Trajectory forecasts in unknown environments conditioned on grid-based plans," *arXiv:2001.00735*, 2020.

[38] X. Mo, Y. Xing, and C. Lv, "Heterogeneous edge-enhanced graph attention network for multi-agent trajectory prediction," *arXiv:2106.07161*, 2021.

[39] ——, "Recog: A deep learning framework with heterogeneous graph for interaction-aware trajectory prediction," *arXiv:2012.05032*, 2020.

[40] A. Scibior, V. Lioutas, D. Reda, P. Bateni, and F. Wood, "Imagining the road ahead: Multi-agent trajectory prediction via differentiable simulation," *arXiv:2104.11212*, 2021.

[41] 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 *CVPR*, 2019.

[42] 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 *CVPR*, 2020.

[43] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," in *arXiv:1607.06450*, 2016.

[44] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou, *et al.*, "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," *arXiv:2104.10133*, 2021.