Object Centric Approach to Motion Prediction for Autonomous Vehicles

Prakhar Singh*
Dept. of Computer Science
University of Texas at Austin
prakharsingh95@gmail.com

Shivam Garg*
Dept. of Computer Science
University of Texas at Austin
shivgarg@live.com

***Abstract*—In this work, we explore whether an object centric approach can help improve performance for the motion prediction task. Our approach extracts features from dense feature maps of ResNet-34 using RoIAlign followed by a suitable aggregation strategy. Our results demonstrate a 20% improvement over Lyft's baseline. We also conduct thorough ablation studies of various hyper-parameters specifc to our approach and find that bounding box size, object count, multi-trajectory prediction have substantial impact on performance.**

I. Introduction

Autonomous driving systems involve three major components: Perception, Prediction and Planning. Perception involves utilizing multiple modalities to understand the environment and the various objects in it. Prediction involves estimating the future behaviour of these objects and finally planning involves utilizing the information gleaned through prediction and planning to make intelligent decisions.

In this project, our focus is on the second component, prediction. Given data collected by an autonomous vehicle, the goal is to predict the behaviour of nearby traffic participants for 50 future timesteps (0.1 seconds/step). This project is inspired by the recent Kaggle competition[1] by Lyft. As such, we utilize their data and problem formulation of the motion prediction task. Our key contribution is an object centric model architecture to solve this problem.

More specifically, the problem asks us, given 99 historical frames containing motion data of various agents like cars, motorcycles, pedestrians, etc along with a rasterized Bird's Eye View (BEV) map of the environment, to predict the future positions of these agents 50 frames into the future. The prediction can include upto 3 trajectories for each agent along with accompanying confidence scores $\mathbf{c}$ (with $\sum_{i=1}^{3} c_i = 1$). The evaluation metric is the negative log likelihood of the true trajectory under a Gaussian distribution centered at the predicted trajectory (we describe it in detail in subsection IV-A).

Our approach to this problem is motivated by the idea that immediate future motion of any agent is highly influenced by the motion of nearby objects in the scene. To exploit this idea, we modify Lyft's baseline approach (see subsection III-A) to extract feature maps of objects nearby the object of interest using RoIAlign [3]. We then evaluate various architectures to aggregate information from all extracted feature maps followed by two heads that predict the trajectory and confidence scores.

We consider various aggregation strategies like averaging, LSTMs, GraphCNNs, etc. Surprisingly, we found that averaging the feature maps works the best rather than the more involved approaches. Our best result improves upon the baseline by more than 20% given the same number of gradient updates. In addition, we conduct extensive ablation studies to understand the effect of various architecture and hyper-parameter choices (like region sizes, pooled sizes, etc). We find that increasing the number of agents considered is very helpful and filtering out certain agents, as in Lyft's baseline, harms performance.

The rest of this report is organized as follows. In section II we describe the data in detail. We also highlight some key optimizations we made to achieve a improve our dataloader performance which is what each our our experiments is bottlenecked by. In section III we describe our methodology in detail and in particular focus on how we process the extracted object features. We report our main results, our ablation studies in section IV. We briefly list some releavnt prior work in section V and finally conclude in section VI.

We have attached our code along with this report. Please see `README.md` in the attached code for details.

II. Data

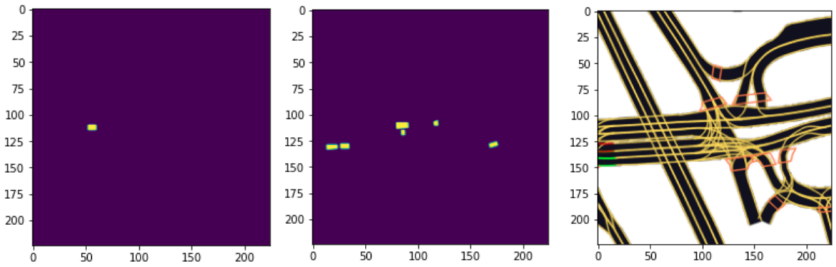
A. Data Format

We use Lyft Motion Prediction Dataset[4] in this work. The dataset has 170k scenes of 25 seconds duration each, in total having approx. 1,118 hours of driving logs. There are on average 79 traffic agents in each frame with a total of about 320M agents. The data has been captured during day time and post processed to provide road geometry and aerial map information.

The dataset is organized in three arrays:

- The first array contains information about scenes. Each scene is composed of a sequence of frames.
- The second array contains information about frames, where each frame contains information of the environment at any particular step. It contains information about all the agents present in the frame alongwith all the

[1] https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles



(a) Object to track (1 channel). (b) Nearby objects (1 channel). (c) BEV Raster (3 channels).

Fig. 1: Rasterized agent, object and BEV channels.

recognised traffic light faces. The frame also contains the AV's translation and rotation matrices to transfer between various coordinate spaces.
- The third array contains information about each agent: its centroid, extent, yaw, velocity and agent type.

B. Data Extraction Process

In this work, we rely on agent centered views of the environment to predict the agent's future trajectory (similar to Lyft's baseline). We leverage Lyft's l5kit[2] to extract the following information for each agent:

- Agent centered rasterized BEV map
- Immediate preceding 10 positions of the focus agent. This information is plotted on a 2D map, one for each time step.
- Immediate preceding 10 positions of other agents in the scene, again for each time step.
- A mask denoting the class the traffic the agent belongs to. This is derived from a probability distribution over all the classes the agent could be belong to available as part of the dataset. The threshold we use to binarize this distribution is 0.5.
- The ground truth trajectory for upto 50 steps in the future.

A sample instance is shown in Figure 1. The BEV map is concatenated with all the 2D maps for each of the historical timesteps for both the focus and surrounding agents.

C. Dataloader Optimization

One of our biggest implementation challenges in this project relates to the dataloader. Specifically, in all our experiments the training was CPU bottlenecked due to the dataloader.

This problem was further exacerbated when we implemented object bounding box extraction. We found that training slowed down by $2.4\times$ which was significant as we were already looking at long training times even with the standard dataloader.

We spent significant time investigating Lyft's code underlying their API and identified some key Lyft APIs that were causing the additional slowdown. Specifically we identified that a binary search operation on all $320M$ agents and a method call `AgentDataset.get_frame_indices()` were the worst offenders.

We refactored those out and utilized the raw ZARR data to extract object information. This resulted in significant implementation effort but resulted in us getting rid of the $2.4\times$ slowdown. This optimization was key to us being able to test significantly more hypotheses.

III. Methodology

A. Baseline

We consider Lyft's approach as the baseline. In this approach, the rasterized BEV map and and spatial rendering of objects (based on their current and historical positions) on a $(224, 244)$ image is input to ResNet-34. Then, they replace the final classification layer with a FC layer for feature extraction which is then used to predict the agent's future trajectory. Specifically, the output is a $(50, 2)$ vector corresponding to its predicted (x, y) positions (in its own coordinate space) 50 steps in future. This model is optimised using MSE loss between the predicted and the ground truth trajectories.

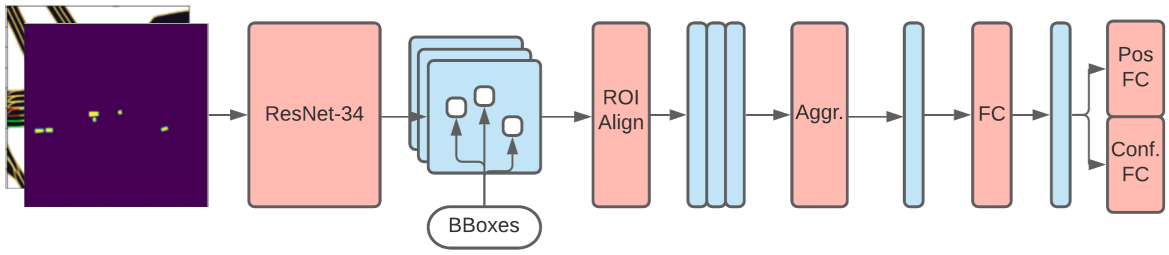
B. Preliminaries

The focus of this project is to explore object centric features to improve upon the baseline. Our approach is motivated by the hypothesis that an agent's motion is significantly informed by the motion of its nearby agents. We treat each agent in the sence as an "object".

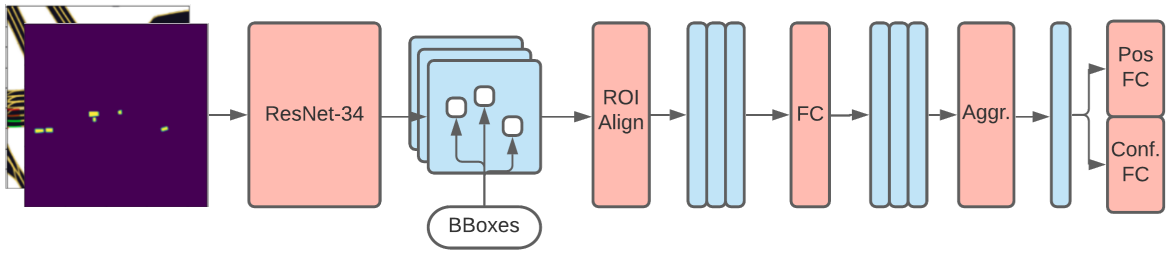
Before we describe our approach in detail in subsection III-C, we describe some key details upon which we build our approach.

1) Agents Representation: For each object, we extract a representative feature map using RoIAlign from stage 4 of ResNet-34 using fixed bounding boxes of size (w, w) centered on the objects. The output of the RoIAlign layer for each object is determined by the ResNet backbone. For ResNet-34, it $(p, p, 512)$ where p is the pooled size for RoIAlign. We study the effect of bounding box size w and pooled size p in our ablation studies as discussed in subsubsection IV-D1 and subsubsection IV-D2.

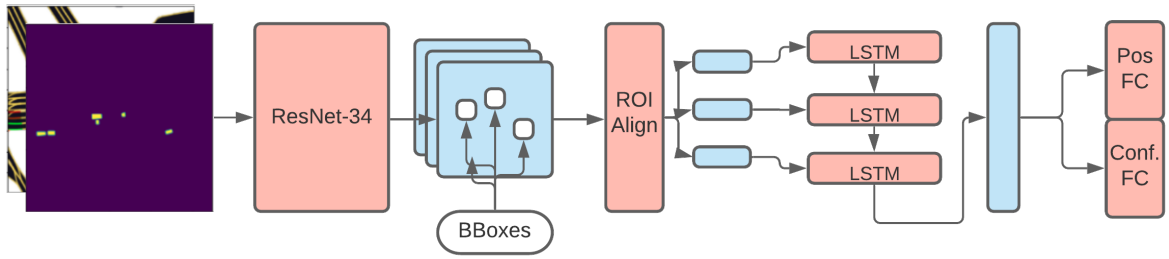
[2]https://github.com/lyft/l5kit



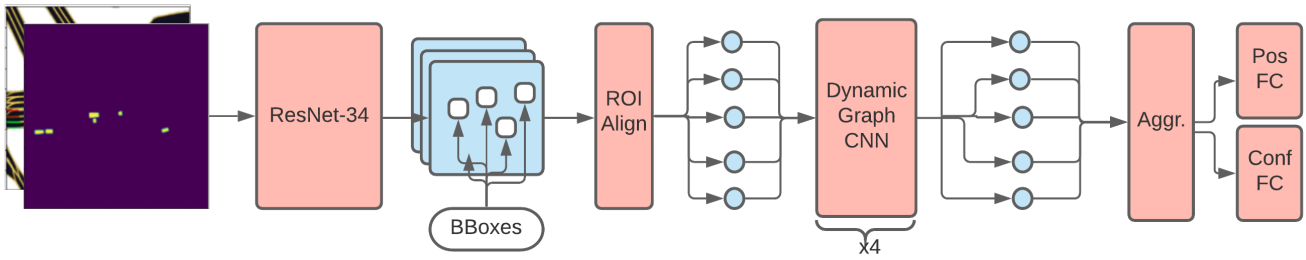
(a) Averaging aggregation.



(b) Averaging aggregation with transformation first.



(c) LSTM Aggregation



(d) Dynamic Graph CNN Aggregation

Fig. 2: Overview of the key architectures that we consider in this project. “ResNet-34” denotes the stage 4 output from Resnet-34.

2) Agent Selection: Considering that there are multiple objects ("agents") in any given scene, we try two methods of selecting agents to extract features for and process further based on two criteria:

- Agent Masks: For this criteria we explore two choices, one where we take all the agents present in the scene and another when we filter agents based on the agents masks (see subsection II-B). See subsubsection IV-D3 for experimental results.
- Distance from the focus agent: We choose N agents which are closer to the focus agent where N is a hyper-parameter. We explore the impact of various choices of N in subsubsection IV-D4 .

3) Multi-Trajectory Prediction: The task setup allows predicting upto $K = 3$ trajectories. We employ two setups in our experiments, one in which we predict a single trajectory and another where we predict three candidate trajectory along with confidence scores for each trajectory.

In the multiple-trajectory setting, we have three heads, one for each trajectory and another head to predict the confidence of each trajectory We refer to these as `Pos. FC` and `Conf. FC` respectively. The effect of predicting multiple trajectories is discussed in subsection IV-E.

C. Our Approach

1) Feature Aggregation: This is our first line of experiments where we aggregate agent's features by averaging RoIAlign features. Specifically, we bisect ResNet-34 at the end of stage 4 and apply an RoI Align layer to extract object ("agent") specific feature maps. These feature maps are aggregated into a single feature map using average pooling. This aggregated representation is passed through a FC and finally into the prediction heads. Refer to Figure 2a for block diagram of the approach.

2) Averaging Feature Aggregation: We also experiment with a slight variation to this approach where we extract feature maps from ResNet-34 stage 4 using RoI Align but instead of aggregating feature maps directly, we instead apply a transformation (a FC layer) first and then aggregate. Refer to Figure 2b for block diagram of the approach.

3) Integrating Label Information: The dataset provides the type label for each agent, e.g. car, bicycle, pedestrian etc. Apart from using this for filtering purposes (as in the baseline), we also experiment by appending information to the RoI features maps. The is motivated by the observation that different traffic agents behave differently, e.g. bicycles usually have lower speeds than cars, pedestrians have lower speeds than bicycles and usually stay off-road, etc. Refer to subsection IV-F for results.

4) LSTM Feature Aggregation: In this experiment, instead of average pooling the RoI feature maps, we instead aggregate the feature maps using LSTM. Specifically, LSTM processes feature representations from all agents in the scene and, ideally, learns relevant information about how the motion of nearby agents affects the motion of the focus agent.

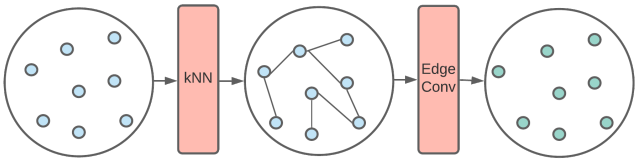


Fig. 3: Our implementation of Dynamic Graph CNN.

We pass the RoI features directly into a stacked LSTM and use the hidden state representation at the last LSTM step to predict trajectories and confidence values.

The schematic diagram of the approach is shown in Figure 2c. This approach is inspired from Zhang et al. [17] where they also pass features from regions of interest directly to an LSTM for image classification.

5) Dynamic GraphCNN Based Aggregation: In this approach, we utilize Dynamic Graph CNN[13] for aggregating RoIAlign features. This is motivated by the observation that nearby agents should be assigned more weight considering that the BEV map covers a large area (112×112 sqm), as opposed to assigning uniform weight to all agents in the scene like the previous approaches would do.

Specifically, we again extract features from ResNet-34 stage 4 using RoIAlign and apply four layers of Dynamic Graph CNN. Inside each layer, we onstruct an undirected graph using a k nearest neighbour approach. We apply EdgeConv operation on all the nodes to generate neighbour contextualised representation for each node and average the edge features at each node to compute node features.

We describe the Dynamic Graph CNN operation visually in Figure 3. Following the Dynamic Graph CNN layers, average pool the representations for all nodes and predict trajectories and confidence scores. The overall approach is shown schematically in Figure 2d.

6) Recurrent Prediction: In all the previously described approaches, we predict the next 50 steps in one go, i.e. there is no dependence of position at $t + 1$ step on the positions from 1 to t^{th} time steps. We also try a recurrent motion prediction approach which predicts motion for only a few steps into the future repeatedly. This approach performed poorly by a large margin despite our various attempts. As a result, we discarded this approach altogether.

IV. Experiments

A. Metric

The evaluation metric is the negative log likelihood of the true trajectory under a Gaussian distribution centered at the predicted trajectory. Specifically, the density model assumes all predictions are i.i.d and the overall distribution is expressed as a product of unit Gaussians centered at the predicted positions (note that centering the Gaussians at the target positions and computing the likelihood of the predicted trajectory will yield identical results).

	150k	180k	210k
Baseline	55.0	47.5	47.5
Averaging	**43.4**	**38.2**	**38.2**
LSTM	86.5	74.3	64.7
Dynamic Graph CNN	61.7	58.9	54.2

TABLE I: Comparison of different object feature aggregation strategies.

Mathematically, given the ground truth trajectory $x_1, ..., x_T, y_1, ...y_T$ and k predicted trajectories ($1 \leq k \leq 3$), $\bar{x}_1^k, ...\bar{x}_T^k, \bar{y}_1^k, ...\bar{y}_T^k$, the density model, which assumes spatial and temporal independence, is defined as:

$$\begin{aligned} & p(x_{1...T}, y_{1,,T} | c^{1....K}, \bar{x}_{1,...T}^{1,...K}, \bar{y}_{1,...T}^{1,...K}) \\ & = \sum_k c^k \mathcal{N}(x_{1,....T} | \bar{x}_{1,...T}^k, \Sigma = 1) \mathcal{N}(y_{1,....T} | \bar{y}_{1,...T}^k, \Sigma = 1) \\ & = \sum_k c^k \prod_t \mathcal{N}(x_t | \bar{x}_t^k, \sigma = 1) \mathcal{N}(y_t | \bar{y}_t^k, \sigma = 1) \end{aligned} \tag{1}$$

The final score then is the negative log of the true trajectory given the above density model:

$$\begin{aligned} L &= -\log p(x_{1...T}, y_{1,,T} | c^{1....K}, \bar{x}_{1,...T}^{1,...K}, \bar{y}_{1,...T}^{1,...K}) \\ &= -\log \sum_k e^{\log(c^k) - \frac{1}{2} \sum_t ((\bar{x}_t^k - x_t)^2 + (\bar{y}_t^k - y_t)^2)} \end{aligned} \tag{2}$$

Here c^k refers to the predicted condidence for the k-th trajectory. Only predictions where $0 \leq c^k \leq 1$ for all k and $\sum_k c^k = 1$ are considered valid.

B. Training Details

We train each model for up to $210,000$ gradient steps. For optimization, we use Adam [7] with a learning rate of $1e-3$ and linearly decay the learning rate in such a fashion that the learning rate would reach zero in two epochs (in practice the dataset is too big to finish even one epoch). We use a warmup of 500 steps. We train with a batch size of 16.

We use the NLL metric described in subsection IV-A as our loss as this metric is fully differentiable. During the NLL implementation in PyTorch, we faced numerical stability issues due to the various `softmax`, `log` and `exp` operations where we observed that the loss would `NaN` out after some number of gradient updates. We addressed this by switching to `log_softmax` and `logsumexp` where we always work with log probabilities.

C. Comparing Aggregation Methods

Our overall results comparing the different aggregation methods are available in Table I. We find that the averaging the RoIAlign features works best and significantly outperforms all the other aggregation approaches we considered. In particular, we notice that it takes a large number of gradient updates before the loss starts decreasing when LSTM aggregation is used. We hypothesize that this is a consequence of permutation invariant features being hard to learn. Surprisingly, averaging also outperforms Dynamic Graph CNN based aggregation.

	150k	150k	210k
RoI Region: (7, 7)	47.7	**46.1**	46.1
RoI Region: (13, 13)	**47.2**	46.3	**45.8**
*RoI Region: (24, 24)**	54.9	50.8	46.6

TABLE II: Comparison of different RoI region sizes with averaging aggregation.

	150k	150k	210k
*RoI Pool Size: (1, 1)**	**54.9**	**50.8**	**46.6**
RoI Pool Size: (3, 3)	65.0	62.4	62.4

TABLE III: Comparison of different RoI pool sizes with averaging aggregation.

D. Ablation Studies

In this subsection, we take the overall best performing architecture - averaging aggregation - and present results for various ablation studies through which we ascertain the impact of different hyper-parameters. We limit our ablation studies to hyper-parameters specific to our proposed architecture.

Specifically, we use the following settings as reference for all the following ablation studies:

- RoI Region Size: $(24, 24)$ (w.r.t. input image size)
- RoI Pooled (Region) Size: $(1, 1)$
- RoI sampling ratio: 2
- Num of objects considered: 12
- Object Masking: Yes (this is the same setting as Lyft's baseline, see subsection III-A)

Please note that while this reference setting outperforms Lyft's baseline, this is not the one that yields the best results in Table I. This is because we conducted these ablation studies towards the end of the project and one of these studies ended up improving our best result.

1) Effect of RoI Region Size: Table II shows the results for RoI Region size ablation. By RoI region size, we refer to the bounding box used in the RoIAlign operation. We find that the best region size is $(13, 13)$ and both smaller $(7, 7)$ and bigger $(24, 24)$ region sizes yield poorer results. We hypothesize that the loss in performance due to the bigger sizes is because of noisy information from the feature map being averaged in by RoIAlign while we attribute the loss in performance due to the smaller size to useful agent features being missed out.

2) Effect of RoI Pooling Size: Table III shows the results for RoI Pooling size ablation. By RoI pooling size, we refer to the size of the pooled features post the RoIAlign operation. For $(1, 1)$ pooling the object feature size is 512. For $(3, 3)$ pooling, we flatten the features along the channel dimension resulting in object features of size 4608. We find that $(1, 1)$ works better despite the lower feature size.

3) Effect of Object Masking: As we discussed in subsection II-B, the dataset has mask representing objects' classes, e.g. cars. In this ablation study, we study the effect of keeping

	150k	150k	210k
*With mask**	54.9	50.8	46.6
No mask	**46.4**	**41.5**	**39.8**

TABLE IV: Comparison of object masking vs no masking with averaging aggregation.

	150k	150k	210k
Upto 6 objects	54.1	54.1	44.3
Upto 12 objects	54.9	50.8	46.6
Upto 18 objects	**43.4**	**38.2**	**38.2**

TABLE V: Comparing the effect of using different neighbour counts with averaging aggregation.

this mask on vs removing it. Surprisingly, (Table IV) we find that the mask isn't useful and removing the mask significantly improves the results.

4) Effect of Object Count: In Table V, we present the results for different neighbour counts. We find that with smaller number of neighbours the results are comparable but with a larger number of neighbours (18+) the results improve significantly.

E. Effect of Multi-Trajectory Predictions

In Table VI we study the impact of predicting single ($k = 1$) vs multiple ($k = 3$) trajectories. We find that the results for multiple trajectories are significantly better. Similar to Lyft's reasoning, we suspect that predicting multiple trajectories allows the different prediction heads to specialize to different trajectory types and sizes like curved vs straight, short vs long etc. We show various examples from the validation set depicting this phenomenon in Figure 4.

F. Effect of Averaging after Feature Transformation

In Table VII we study the impact of switching the order of the fully connected feature transformation layer and the RoI averaging operation. We find that the results degrade if we do feature transformation first. However, if we pass on the extra label information (for which we require feature transformation first) we find that the results improve.

V. Related Work

Latest approaches in field of motion prediction, like most recent approaches in Computer Vision, utilize deep learning models based on CNN, GNN and LSTMs. We discuss some of these recent approaches below.

Wu et al. [15] predict future video frames given a sequence of continuous video frames. They separate the problem in two parts by separating the background and moving objects in the image. They predict the background scene by calculating the optical flow and warp the last scene background. They also use a spatial transformer network [6] to predict the motion of dynamic objects. Their approach can be used to predict future BEV maps and thus infer, future locations of each agent in our context.

	150k	150k	210k
1 trajectory	94.4	94.4	93.9
3 trajectories	**54.9**	**38.2**	**38.2**

TABLE VI: Comparing single vs multi trajectory predictions with averaging aggregation.

	150k	150k	210k
Avg Before	54.9	50.8	46.6
Avg After	58.0	54.2	54.2
Avg After w/ Object Label	**50**	**44.2**	**44.2**

TABLE VII: Comparing two setups for averaging aggregation: averaging before vs after the FC layer with and without object label information.

Wu et al. [14] propose a deep model, MotionNet, to perform perception and motion prediction from 3D point clouds. They take in LIDAR scans as input and output a BEV map which is passed to their proposed spatio-temporal pyramid network to predict the BEV for the next time step.

Hu et al. [5] do motion prediction by modelling interaction between agents via a graph neural network. They propose a novel Neural Motion Message Passing (NMMP) framework to model the interactions between different objects in the scene. NMMP module to learn individual and interactive embeddings per agent in the environment based on the history of positions. The NNMP module uses LSTM to learn individual embedding for each agent. For interactive embeddings, the NMMP uses a directed Graph Network to learn the embeddings. This directed graph models the asymmetric relations between nodes in the traffic, where the follower traffic is dependent on the leader but not vice-versa. They use a GAN framework to train the NMMP. The generator uses representations from the NMMP module and generates trajectory for each agent. The discriminator also uses a NMMP module and the representation of the trajectories are passed to a MLP to classify it as a fake or real trajectory.

Fang et al. [2] propose a two stage network to predict motion of agents. In the first stage, the CNN encoder-decoder takes as input the history of the agent locations along with surrounding road information and outputs a rough estimate of the end point of the agent path. This estimate is then input to a proposal generation module which outputs a set of trajectory proposals using CNN encoder-decoder. These generated proposals are classified for feasibility and regressed for fine tuning the predictions. A drawback of their approach is that they do not model the interaction with other agents explicitly, instead relying on the CNN to learn such dependencies implicitly.

Aliakbarian et al. [1] propose a framework to predict human poses given a history of poses. Their framework uses a VAE [8] to control for the diversity of the generated future trajectories generated by adding a noise to the RNN hidden state input to the VAE. This helps in generating diverse motion sequences.

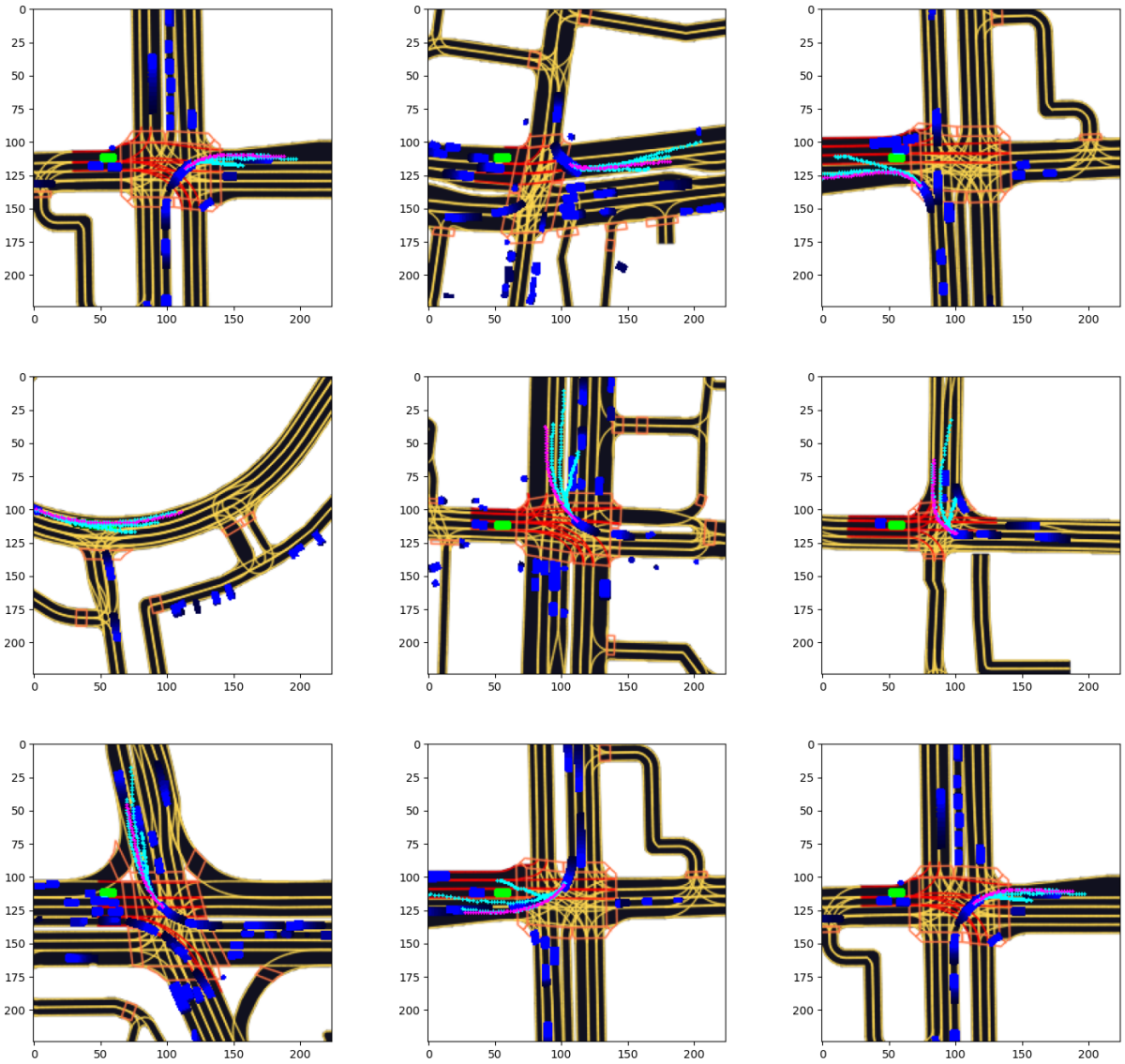


Fig. 4: Examples of multi-trajectory prediction from the validation set from the perspective of the AV (in green). In each example we show the prediction and the ground truth for exactly one agent to avoid clutter. Ground truth is purple while the predicted trajectories are blue. Predicting multiple trajectories allows each trajectory head to specialize as we see in examples above.

Sadeghian et al. [10] propose a GAN framework for motion prediction. In the generator, they generate the trajectories for each agent using the BEV maps alongwith the agents trajectories. The BEV map is processed using a CNN and the trajectories are modelled using LSTM. Then for each agent, two different attention modules, Physical and Social Attention, learn relevant portions of the BEV image and the neighbouring agents trajectories respectively. These learnt representations are concatenated and sent to the decoder LSTM to output the position of the agent in the future. These trajectories are then modelled using a LSTM in discriminator to predict whether the trajectory is fake or not.

Kosaraju et al. [9] is another notable work where the authors use Bicycle-GAN[18] to model the motion prediction task. In the generator, encode agent's trajectory using a LSTM and the global scene image using VGG[11] network. These representations are then used with Graph Attention Networks[12] to learn a global embedding common for every agent. Also, for each agent, attention is performed over the image features by using the hidden state of LSTM. The outputs of the graph attention network, LSTM hidden state and contextualised image feature are concatenated and given to LSTM as input to generate future positions. They have two discriminators, one for classifying the trajectories based on local context and

one by using global image context. They incorporate a latent encoder to enforce a bijection constraint between the generated trajectory and input noise to the generator.

Another work similar to ours but in a differnt domain is that of Ye et al. [16]. In this paper, they approach the problem of robotic manipulation by modelling the as a forward model. The forward model takes in the spatial position explicitly alongwith with RoI image features extracted from the vicinity of the object which are both then concatenated. A graph network is then used to model interactions between various objects and actions for each object are predicted. Our approach differs in two key ways: (1) we consider various aggregation strategies, and (2) our domain of application is very different.

A. Future Work

In this section, we identify some sailent directions for future work:

1) The current approach does not encode spatial context of agents explicitly, rather we rely on it being learned via Resnet, esp. since stage 4 has large receptive fields. More explicit encodings of spatial context may improve results.
2) Our graph CNN approach makes the assumption that the relation between various traffic agents is undirected, which is not necessarily true, e.g. in traffic the trailing cars are likely to mimic the motion of leading cars. Therefore, it will be worth while to explore directed graph CNN architectures and graph construction algorithms.
3) Another promising direction to explore would be recursive prediction based approaches, where instead of predicting for all 50 steps at one we predict for a smaller number of steps for *all* agents and repeat. This has the benefit of temporal mixing of predictions across agents. In our experiments such approaches didn't yield good results but it's possible that there's a better setting of hyperparameters that would improve results.
4) Our approach only makes use of the rendered BEV map. In the dataset, satellite imagery is also available. Approaches that utilize information from both of these sources may yield better results.

VI. Conclusion

In this report, we explored the hypothesis of whether object centric RoI features are helpful for agent motion prediction task. We found that modelling the problem using these features is significantly beneficial and our approach improves upon the baseline by nearly 20%. To optimally utilize agent RoI features, we experimented with different architectures motivated by different inductive biases and found, surprisingly, that vanilla averaging outperforms LSTM and Graph CNN based methods. We conducted further experiments to evaluate various hyperparameters that control feature extraction. Finally, we cite some related work that has also addressed this task and present ideas for extending our approach in the future.

References

[1] Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Lars Petersson, and Stephen Gould. A stochastic conditioning scheme for diverse human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[2] Liangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. Tpnet: Trajectory proposal network for motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6797–6806, 2020.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:386–397, 2020.

[4] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. One thousand and one hours: Self-driving motion prediction dataset. https://level5.lyft.com/dataset/, 2020.

[5] Yue Hu, Siheng Chen, Ya Zhang, and Xiao Gu. Collaborative motion prediction via neural motion message passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6319–6328, 2020.

[6] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[8] Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[9] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, Hamid Rezatofighi, and Silvio Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In *Advances in Neural Information Processing Systems*, pages 137–146, 2019.

[10] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1349–1358, 2019.

[11] K. Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[12] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[13] Yue Wang, Yongbin Sun, Z. Liu, S. Sarma, M. Bronstein, and J. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38:1 – 12, 2019.

[14] Pengxiang Wu, Siheng Chen, and Dimitris N. Metaxas.

Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[15] Yue Wu, Rongrong Gao, Jaesik Park, and Qifeng Chen. Future video synthesis with object motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5539–5548, 2020.

[16] Yufei Ye, Dhiraj Gandhi, Abhinav Gupta, and Shubham Tulsiani. Object-centric forward modeling for model predictive control. In *CoRL*, 2019.

[17] J. Zhang, Qi Wu, Chunhua Shen, and Jianfeng Lu. Multilabel image classification with regional latent semantic dependencies. *IEEE Transactions on Multimedia*, 20: 2801–2813, 2018.

[18] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. *ArXiv*, abs/1711.11586, 2017.