MotionCNN: A Strong Baseline for Motion Prediction in Autonomous Driving

Stepan Konev
Skoltech
stevenkonev@gmail.com

Kirill Brodt
Novosibirsk State University
cyrill.brodt@gmail.com

Artsiom Sanakoyeu
Heidelberg University
a.sanakoyeu@gmail.com

Abstract

To plan a safe and efficient route, an autonomous vehicle should anticipate future motions of other agents around it. Motion prediction is an extremely challenging task which recently gained significant attention of the research community. In this work, we present a simple and yet very strong baseline for multimodal motion prediction based purely on Convolutional Neural Networks. While being easy-to-implement, the proposed approach achieves competitive performance compared to the state-of-the-art methods and ranks 3rd on the 2021 Waymo Open Dataset Motion Prediction Challenge. Our source code is publicly available at GitHub[*].

1. Introduction

One of the key components of a self-driving system is motion prediction [23, 3]. It is crucial for an autonomous vehicle (AV) to reliably predict future trajectories of other traffic agents, such as cars, cyclist and pedestrians. However, future motion prediction along with the AV's route planning are still very challenging problems and are yet to be solved for an arbitrary environment scenario. In this paper we tackle the motion prediction task. The most prominent approaches include image-based models which leverage birds-eye-view rasterised scene representations [15, 6, 4, 11, 18, 14] and methods incarnated using graph neural networks [2, 8, 25].

We establish a simple and yet efficient motion prediction baseline based purely on Convolutional Neural Networks (CNNs). Our model takes a raster image centered around a target agent as input and directly predicts a set of possible trajectories along with their confidences. The raster image is obtained by rasterisation of a scene and history of the all the agents. We evaluate our model on the 2021 Waymo Open Dataset Motion Prediction Challenge [7] where it achieves very competitive performance: Ranks 1st using minimum average displacement error and 3rd using mAP score. We open-source our code* and hope that our baseline will provide a reference for future research.

*https://github.com/kbrodt/waymo-motion-prediction-2021

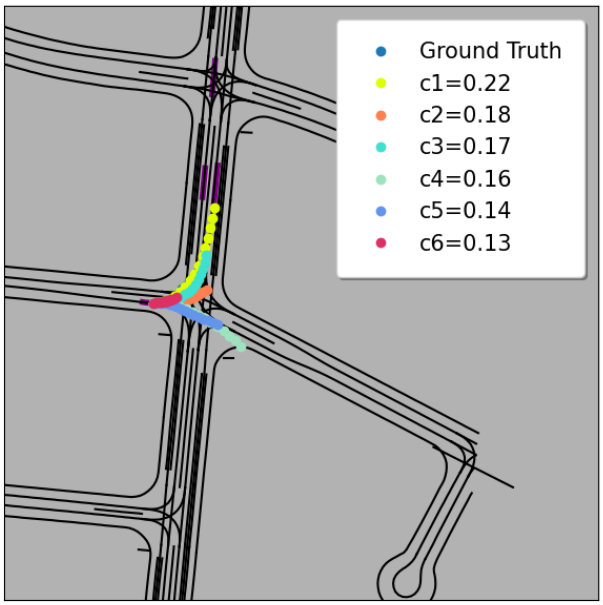

Figure 1: Six trajectories predicted by our model for a target agent. We visualize the trajectories using different colors and show their confidences $c_1, \ldots, c_6$ in the legend.

2. Method

We assume that object tracks are provided by some perception system [19, 24] and focus only on the motion prediction. Our task is to predict the trajectory of an agent for the next T seconds in the future. In this section, first, we describe how we rasterise the data and produce multi-channel images. After that we describe the architecture of our model and the loss function used for training.

Rasterisation To generate training images from raw data, we rasterise historical trajectories of the agents along with the corresponding map providing a context for the road environment. To standardise the input, we rotate and shift the frame in such a way that the target agent at the time of

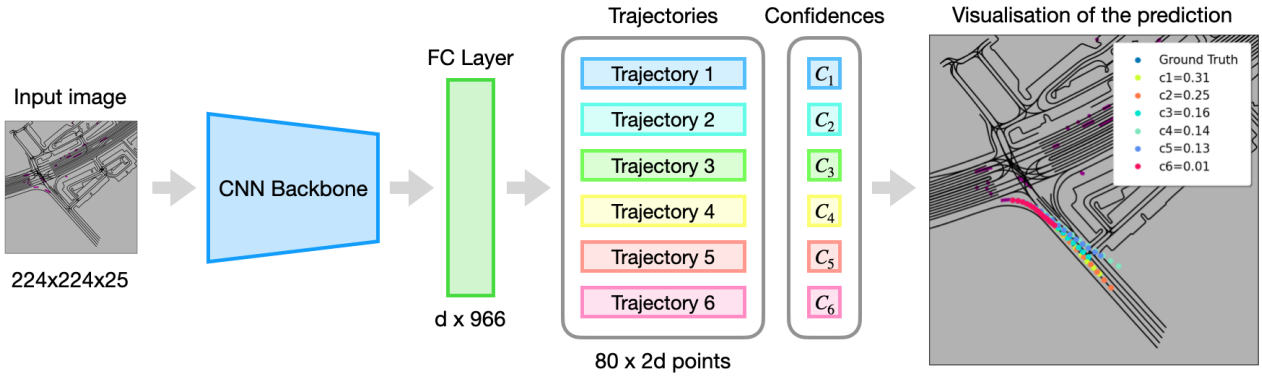

Figure 2: Overview of the architecture of our model.

prediction is always located at a fixed position on the raster image and its velocity is aligned with the X-axis.

Model The future is ambiguous, so we aim to produce K different hypothesis (proposals) for the future trajectory which will be evaluated against the ground truth trajectory. We incarnate our model as an image-based regression. Our model consists of CNN backbone pretrained on ImageNet [20] with one fully-connected layer attached on top (see Fig. 2). The model takes a multi-channel raster image as input and predicts K trajectories along with the corresponding confidence values $c_1, \ldots, c_K$, which are normalized using softmax operator such that $\sum_k c_k = 1$.

Loss function The straightforward solution would be to use a Mean Squared Error (MSE) loss. However, this loss does not allow a probabilistic modelling of multiple hypotheses and it showed poor performance in our preliminary experiments. Instead, we propose to model possible future trajectories as the mixture of K Gaussian distributions. In this case our network outputs the means of the Gaussians while we fix the covariance of every Gaussian in the mixture to be equal to the identity matrix I.

Then for the loss we can use *negative log-likelihood (NLL)* of this mixture of Gaussians defined by the predicted proposals given the ground truth coordinates. In other words, given a ground truth trajectory

$$X^{gt} = [(x_1, y_1), \ldots, (x_T, y_T)]$$

and K predicted trajectory hypotheses

$$X_k = [(x_{k,1}, y_{k,1}), \ldots, (x_{k,T}, y_{k,T})], \; k = 1, \ldots, K,$$

we compute negative log probability of the ground truth trajectory under the predicted mixture of Gaussians with the means equal to the predicted trajectories and the identity matrix I as covariance:

$$L = -logP(X^{gt}) = -log \sum_k c_k \mathcal{N}(X^{gt}; \mu = X_k, \Sigma = I)$$

where $\mathcal{N}(\cdot; \mu, \Sigma)$ is the probability density function for the multivariate Gaussian distribution with mean μ and covariance matrix Σ. The loss can be further decomposed into the product of 1-dimensional Gaussians, and we get just a logarithm of the sum of the exponents:

$$\begin{aligned} L &= -log \sum_k c_k \prod_{t=1}^{T} \mathcal{N}(x_t^{gt}; x_{k,t}, 1) \mathcal{N}(y_t^{gt}; y_{k,t}, 1), \\ &= -log \sum_k e^{log(c_k) - \frac{1}{2} \sum_{t=1}^{T} (x_t^{gt} - x_{k,t})^2 + (y_t^{gt} - yk,t)^2} \end{aligned}$$

Inference We select the number of components in the mixture K equal to the desired number of predicted hypotheses. For example, during evaluation on Waymo Open Motion Dataset [7] we are allowed to provide up to 6 hypotheses of future trajectory for a target agent, so we select $K = 6$. Since we model the possible space of solutions using the probability distribution, it is beneficial to produce the most diverse set of hypotheses from our distribution. One of the ways to achieve this is to simply select means of the components comprising the predicted mixture of Gaussians along with the coefficients c_k as their confidences as final hypotheses for evaluation. While we admit that this might not be the optimal solution, we leave the exploration of other ways to sample trajectories from the predicted distribution for future work.

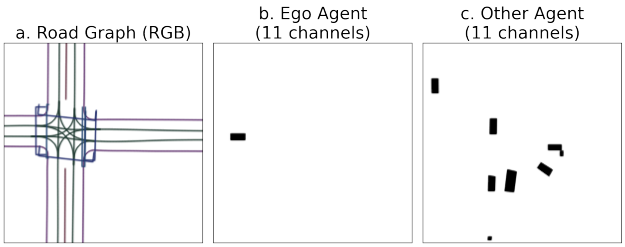

Figure 3: A channel-wise illustration of the input raster of size 224×224 pixels with 25 channels produced for Waymo Open Motion Dataset [7]. The raster consists of three main blocks: (a) the first 3 channels represent the map, (b) the next 11 channels encode the history of target object (one-channel mask per history snapshot), and, similarly, (c) another 11 channels encode the history of all others objects.

3. Experiments

3.1. Dataset

We evaluate our approach on Waymo Open Motion Dataset [22, 7] by submitting our predictions to the Waymo motion prediction challenge [1]. This dataset contains object trajectories and corresponding 3D maps for $103,354$ segments. Each segment is a 20 seconds recording of an object trajectory at 10Hz and map data for the area covered by the segment. A single sample comprises 1 second of history and 8 seconds of future data obtained by breaking the segments into 9-second windows with 5 second overlap. Every such sample contains up to 8 agents marked as "valid" for which the model needs to predict their positions for 8 seconds into the future.

Rasterisation details We create the preprocessing pipeline for Waymo Open Motion Dataset [7] which converts the raw data in TFRecord format to multi-channel raster images for each target object. For every agent we have 1 second of history which is provided as 10 snapshots taken at 10Hz and a snapshot at the time of prediction (current). So, in total we have $T = 11$ snapshots for every dynamic object. We use the raster size $224 \times 224 \times (3 + 2T)$, where the first 3 channels is the RGB map (road lines, crosswalks, traffic lights, etc), and every history snapshot is represented by two extra channels: (1) The mask representing the location of the target agent, and (2) the mask representing all other agents nearby (see Fig. 3). To eliminate the redundant degrees of freedom we shift and rotate the local coordinate system in such a way that the center of the target agent is located at pixel coordinate $(61, 112)$ and its velocity is aligned with the X-axis of the image.

One of the major bottlenecks in the data pipeline is the speed of image rasterisation. To make training faster we cache the rasterised images to disk as compressed npz files.

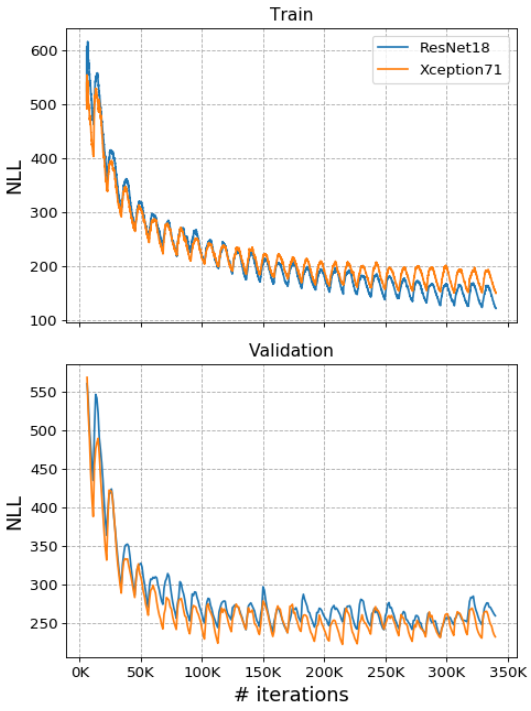

Figure 4: Negative multivariate log-likelihood loss (NLL) during training iterations for MotionCNN with two different backbones.

And during training we just load them from disk instead of costly online rasterisation. This results in significant speedup enabling us to read more than a hundred images per second using a single process.

We create raster images only for the agents with the flag $state/tracks\_to\_predict$ equal to 1 (meaning that they are "valid"). Therefore, in total, we obtain $N \approx 2.2M$ training, $192,181$ validation and $196,056$ test images.

3.2. Metrics

Following the evaluation protocol in [7], we predict 6 hypotheses for every target agent, but only trajectory points subsampled at 2Hz (which results in the subset of 16 2-dimensional coordinates from the predicted 80 points) are used for computing test and validation metrics. Average Displacement Error, Final Displacement Error (FDE) are the commonly used metrics for evaluation:

$$\text{ADE} = \frac{1}{T}\|X^{gt} - X\|_2, \ \text{FDE} = \|x_T^{gt} - x_T\|_2,$$

where X^{gt} is the ground truth trajectory and X is a predicted one. To evaluate multiple hypotheses we use minADE and

	Method	mAP	Min ADE	Min FDE	Miss Rate	Overlap Rate
Test	Waymo LSTM baseline [1]	0.1756	1.0065	2.3553	0.3750	0.1898
	ReCoAt (2^{nd} place) [12]	0.2711	0.7703	1.6668	0.2437	0.1642
	DenseTNT (1^{st} place) [9]	**0.3281**	1.0387	1.5514	**0.1573**	0.1779
	MotionCNN-Xception71 (Ours)	0.2136	**0.7400**	**1.4936**	0.2091	**0.1560**
Val	MotionCNN-ResNet18 (Ours)	0.1920	0.8154	1.6396	0.2552	0.1605
	MotionCNN-Xception71 (Ours)	0.2123	0.7383	1.4957	0.2072	0.1576

Table 1: Quantitative evaluation on test and validation sets of Waymo Open Motion Dataset [22, 7].

	Object Type	mAP	Min ADE	Min FDE	Miss Rate	Overlap Rate
Test	Vehicle	0.2357	0.8946	1.8175	0.2138	0.0886
	Pedestrian	0.2175	0.4449	0.9131	0.1276	0.2725
	Cyclist	0.1875	0.8803	1.7501	0.2860	0.1071
	Avg	0.2136	0.7400	1.4936	0.2091	0.1560
Val	Vehicle	0.2371	0.8919	1.8154	0.2128	0.0877
	Pedestrian	0.2092	0.4387	0.9010	0.1254	0.2684
	Cyclist	0.1905	0.8843	1.7707	0.2835	0.1168
	Avg	0.2123	0.7383	1.4957	0.2072	0.1576

Table 2: Detailed evaluation of our MotionCNN-Xception71 model on test and validation sets of Waymo Open Motion Dataset [22, 7].

minFDE:

$$\text{minADE} = \min_k \frac{1}{T}\|X^{gt} - X_k\|_2,$$
$$\text{minFDE} = \min_k \|x_T^{gt} - x_{k,T}\|_2.$$

Additionally, following [7], we use a few other metrics such as Miss Rate(MR) and mean average precision (mAP). For the detailed explanation of these metrics we refer the reader to the work [7].

3.3. Implementation details

Our implementation is partially based on the winning solution in Lyft Motion Prediction Challenge [13] by Sanakoyeu et al. [21]. We use Xception71 [5] (up to the global averaged pooling) pretrained on Imagenet as a backbone. The output of our model is $K = 6$ trajectories, each containing $T = 80$ 2-dimensional coordinates. We train our model using AdamW [17] optimizing for $340,500$ iterations, thus using early stopping as a regularization. We use a learning rate of 10^{-3}, weight decay 10^{-2} and a batch size of 48. We also use cosine annealing scheduler with warm restarts [16] every $T_0 = 11,350$ iterations, and with $T_{mult} = 1$, $\eta_{min} = 10^{-5}$. Training our model with Xception71 [5] backbone took around 3 days on a single NVIDIA V100 GPU with 32Gb VRAM.

3.4. Results

Results from the final leaderboard of the Waymo open dataset motion prediction challenge [1] are presented in Tab. 1. Despite the simplicity of the proposed approach we secured the 3rd place according to the mAP metric. Moreover, our model is superior to the other competing methods according to Min ADE, Min FDE, and Overlap Rate metrics. Note that in contrast to methods [12, 9], our simple model achieves such impressive results without any use of advanced deep learning techniques or complex architectures.

To test a more lightweight architecture, we also trained our model using ResNet18 [10] as the backbone and evaluated it on the validation set (see Tab. 1). This architecture is 3x times faster to train than the one with Xception71 backbone, but it does not reach the same high performance showing that a sufficiently deep model is necessary for attaining good results. In Fig. 4 we show plots with train and validation loss values during training.

In Tab. 2 we also provide more detailed evaluation results for different object types separately.

4. Conclusion

We presented a simple yet strong baseline – MotionCNN which is based on CNNs and produces a distribution of the hypothetical trajectories for a target agent. The proposed model is straightforward to implement and easy to train. It utilizes a birds-eye-view rasterised scene representation, which we cache as multi-channel images for faster training. We evaluated our approach on Waymo Motion Prediction Challenge [1] where it ranked 3rd, despite being more simple then other competitors. We hope that our work will become a solid reference point for the future advancements in motion prediction.

References

[1] Waymo open dataset motion prediction challenge: Leaderboard. https://waymo.com/open/challenges/2021/motion-prediction, 2021. 3, 4

[2] Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. Spatially-aware graph neural networks for relational behavior forecasting from sensor data. *International Conference on Robotics and Automation (ICRA)*, 2020. 1

[3] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956. PMLR, 2018. 1

[4] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conference on Robot Learning*, 2019. 1

[5] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 4

[6] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019. 1

[7] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. *arXiv preprint arXiv:2104.10133*, 2021. 1, 2, 3, 4

[8] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020. 1

[9] Junru Gu, Qiao Sun, and Hang Zhao. Densetnt: Waymo open dataset motion prediction challenge 1st place solution. https://drive.google.com/file/d/1Cp7jhosYBuVl-w1gIU8z1OSdMKJDgk17/view, 2021. 4

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[11] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8454–8462, 2019. 1

[12] Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Recoat: A deep learning framework with attention mechanism for multi-modal motion prediction. https://drive.google.com/file/d/1Ksq7X5dzouMV2jG1QYcgWzpUl2dKWUDW/view, 2021. 4

[13] Kaggle.com. Lyft motion prediction for autonomous vehicles. https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/overview. 4

[14] Atsushi Kawasaki and Akihito Seki. Multimodal trajectory predictions for autonomous driving without a detailed prior map. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3723–3732, 2021. 1

[15] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 1

[16] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *Int. Conf. Learn. Represent. (ICLR)*, 2017. 4

[17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 4

[18] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14074–14083, 2020. 1

[19] Charles R Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3d object detection from point cloud sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6134–6144, 2021. 1

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2

[21] Artsiom Sanakoyeu, Dmytro Poplavskiy, and Artsem Zhyvalkouski. Winning solution for kaggle challenge: Lyft motion prediction for autonomous vehicles. https://gdude.de/blog/2021-02-05/Kaggle-Lyft-solution, 2021. 4

[22] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset, 2020. 3, 4

[23] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993. IEEE, 2010. 1

[24] Bin Yang, Min Bai, Ming Liang, Wenyuan Zeng, and Raquel Urtasun. Auto4d: Learning to label 4d objects from sequential point clouds. *arXiv preprint arXiv:2101.06586*, 2021. 1

[25] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294*, 2020. 1