DenseTNT: End-to-end Trajectory Prediction from Dense Goal Sets

Supplementary Material

A. Offline Optimization

To boost the training of DenseTNT, we devise an offline model which is composed of a context encoding module and an optimization algorithm. There are different metrics to measure the performance of multi-trajectory prediction methods. For a comprehensive evaluation, we tested the effectiveness of the optimization algorithm under different combinations of optimization objectives, as shown in Table 1.

B. Implementation Details

B.1. Agent and map encoding.

To normalize the map, we take the last position of the target vehicle as the origin and the direction of the target vehicle as the y-axis. Following VectorNet [2], the lanes and the agents are converted into sequences of vectors. Each vector contains the start point, the endpoint, and the attributes of its corresponding lane or agent. A vector that belongs to a lane also contains its index in this lane, and a vector that belongs to an agent contains the timestamps of its start point and end point. After the sparse context encoding, we obtain the features of the lanes and the agents.

B.2. Optimization algorithm

The optimization algorithm aims to find a goal set which minimizes the expectation error. It is implemented by a statically-typed language to achieve the fastest speed and search for hundreds of goal sets in 100ms. We run the optimization algorithm on 8 CPUs in parallel with different initializations and pick the best result. The main cost is on the calculation of the expectation error of each searched goal set.

The probability distribution of the final position is indicated by heatmap goals $\mathcal{C} = \{c_1, c_2, ..., c_m\}$ and their corresponding probabilities $H(c_i)$. When calculating the expectation error of a given goal set, we only consider c_i which satisfies $H(c_i) \geq 10^{-3}$. Since the sample density is 1m, each heatmap goal c_i represents a space of $1m \times 1m$. To obtain a more precise expectation error, we divide each heatmap goal to 9 heatmap goals with the same probability, and each of them represents a space of $\frac{1}{3}m \times \frac{1}{3}m$.

B.3. Goal set predictor

The goal set predictor aims to learn a mapping from the heatmap to the goal set. We only encode heatmap goals which satisfies $H(c_i) \geq 10^{-5}$. First, we normalize both the 2D coordinates of heatmap goals and the pseudo labels by taking the heatmap goal with the highest probability as the origin. Then, a 2-layer MLP is used to encode the heatmap goals, of which input is the 2D coordinates of each goal and its corresponding log probability. The features of heatmap goals are passed to the predictor heads. A softmax function is employed to normalize the predicted confidence of all heads. The head number of the goal set predictor is set to 12.

C. Planning

For a safe and smooth autonomous driving system, predicting the future behaviors of road participants helps us find a safe planning policy for autonomous vehicles. DenseTNT outputs a goal heatmap and it can be used by a downstream planning algorithm. We go back to trajectories because many existing planning methods are based on the trajectories. Motion planning based on heatmaps is a direction worth exploring for the community, and our dense probability map is compatible with this setting. For autonomous vehicles, prediction and planning are performed every moment. The predicted trajectories may jitter over time, causing the planned trajectory to be not smooth enough. Planning based on the goal heatmap can alleviate this problem since the heatmap changes more smoothly over time.

D. Qualitative Results

Figure 1 shows some representative comparisons with typical goal-based trajectory prediction methods, of which performance heavily depends on the quality of heuristically predefined anchors. We also provide more qualitative results in diverse traffic scenarios on the Argoverse validation

	Method			minADE	minFDE	Miss Rate
Validation Set	DESIRE [3]			0.92	1.77	18%
	MultiPath [1]			0.80	1.68	14%
	TNT [5]			0.73	1.29	9.3%
	LaneRCNN [4]			0.77	1.19	8.2%
	DenseTNT w/ optimization (100ms)	minFDE	Miss Rate			
		0%	100%	0.80	1.27	**7.0**%
		30%	70%	0.74	1.12	7.3%
		50%	50%	0.74	1.09	7.5%
		70%	30%	0.73	1.08	8.0%
		100%	0%	**0.73**	**1.05**	9.8%

Table 1. Performance of the optimization algorithm under different combinations of optimization objectives.

set in Figure 2. The probability distribution of the final position in some cases is pretty diverse, and it is difficult for NMS to handle well.

References

[1] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 2

[2] 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

[3] 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. 2

[4] Wenyuan Zeng, Ming Liang, Renjie Liao, and Raquel Urtasun. Lanercnn: Distributed representations for graph-centric motion forecasting. *arXiv preprint arXiv:2101.06653*, 2021. 2

[5] 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. 2