DenseTNT: Waymo Open Dataset Motion Prediction Challenge 1^{st} Place Solution

Junru Gu Qiao Sun Hang Zhao
IIIS, Tsinghua University
{gujunru123,larksq,zhaohang0124}@gmail.com

Abstract

In autonomous driving, goal-based multi-trajectory prediction methods are proved to be effective recently, where they first score goal candidates, then select a final set of goals, and finally complete trajectories based on the selected goals. However, these methods usually involve goal predictions based on sparse predefined anchors. In this work, we propose an anchor-free model, named DenseTNT, which performs dense goal probability estimation for trajectory prediction. Our model achieves state-of-the-art performance, and ranks 1^{st} on the Waymo Open Dataset Motion Prediction Challenge.

1. Introduction

Trajectory prediction is a highly challenging task in autonomous driving due to the inherent stochasticity and multimodality of human behaviors. To model this high degree of uncertainty, some approaches predict multiple future trajectories by sampling from the distribution represented by the latent variables, *e.g*. VAEs [7] and GANs [6]. Other approaches generate a fixed number of trajectories but only perform regression on the closest one during training [6, 8, 2], namely using variety loss. Multipath [1] and CoverNet [9] pose the problem as a classification problem by classifying over template trajectories.

More recently, goal-based methods [14, 10, 11, 13] have gained popularity and achieved state-of-the-art performance. Their key observation is that the endpoint carries most of the uncertainty of a trajectory, therefore they first predicted the goal(s) of an agent, and then further completed the corresponding full trajectory for each goal. They obtained the final goal positions by classifying and regressing predefined *sparse* anchors, as shown in the lower-left part of Figure 1. For example, TNT [14] defined anchors as the points sampled on the lane centerlines; some others [13] took the lane segments as anchors and predicted a goal for each lane segment.

The prediction performance of these goal-based methods heavily depends on the quality of the goal anchors. Since an anchor can only generate one goal, it is impossible for the model to make multiple trajectory predictions around one anchor. Besides, different positions on the same lane segment have different local information, such as the relative distance to the nearest lane boundary. Sparse anchor-based methods cannot make use of such fine-grained information.

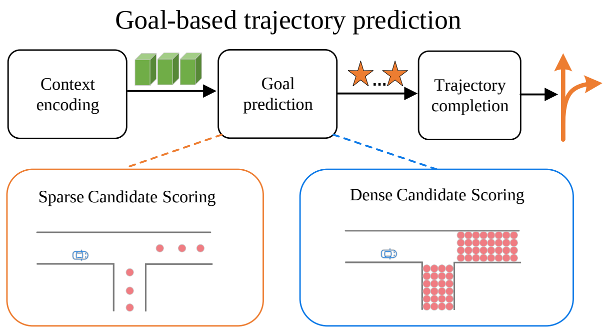

Figure 1. A typical goal-based trajectory prediction pipeline is shown in the upper part of the figure. Existing goal prediction methods (lower left) first define sparse goal anchors heuristically, and then regress and classify these anchors to estimate the goals. In contrast, our method (lower right) estimates the probabilities of the dense goals without relying on the quality of heuristically predefined anchors (anchor-free).

In this work, we propose DenseTNT, an **anchor-free** goal-based trajectory prediction method. It largely improves the performance of goal estimation without relying on the quality of heuristically predefined goal anchors. We first extract sparse scene context features and then employ a *dense* probability estimation to generate the probability distribution of the goal candidates. Finally, a trajectory completion module outputs trajectories based on a set of selected goals.

2. Method

2.1. Scene context encoding

Scene context modeling is the first step in behavior prediction. It extracts the features of the lanes and the agents and captures the interactions among them. Sparse encoding methods [5, 8] (also called vectorized methods) were

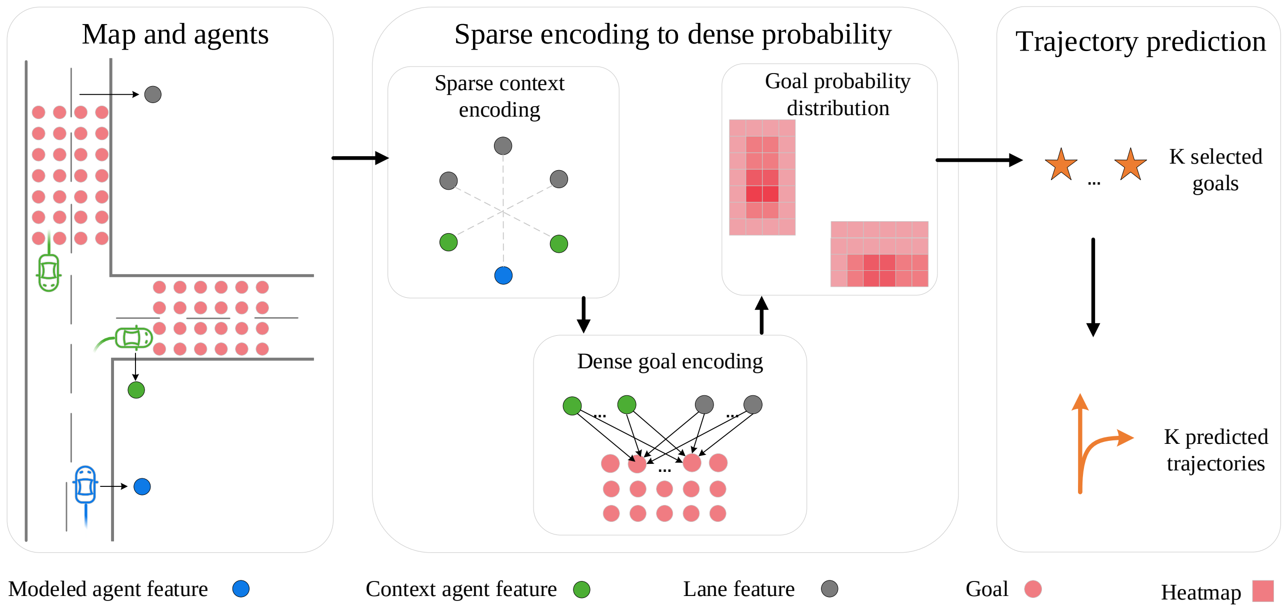

Figure 2. An illustration of DenseTNT. A vectorized encoding method is used to extract features of HD maps and agents; and then a dense goal encoding module is employed to output dense probability distribution.

proposed recently. Compared to dense encoding methods which rasterize the lanes and the agents into images and use CNNs to extract features, sparse encoding methods abstract all the geographic entities (*e.g.* lanes, traffic lights) and vehicles as polylines, and better capture the structural features of high-definition maps.

2.2. Dense goal probability estimation

After scene context encoding, we perform probability estimation for goals on the map. TNT [14] defined discretized sparse anchors on the roads and then assigned probability values upon them. Our key observation is that sparse anchors are not a perfect approximation of real probability distributions on the roads, because (1) one anchor can only generate one goal, we cannot make multi-trajectory predictions around one anchor; (2) there are many ordinary points on the roads (those away from the lane centers or boundaries) that were not well modeled: different ordinary points on the same road have different local information, *i.e.* the relative distance to the nearest lane boundary.

Therefore, we perform dense goal probability estimation on the map instead. Concretely, a dense goal encoding module is used to extract the features of *all* the locations on the road under a certain sampling rate. Then, the probability distribution of the dense goals is predicted.

The dense goal encoding module uses an attention mechanism to extract the local information between the goals and the lanes. We denote the feature of the i_{th} goal as F_i, which is obtained by a 2-layer MLP, and the input of the MLP is 2D coordinates of the i_{th} goal. The local information between the goals and the lanes can be obtained by attention mechanism:

$$Q = FW^Q, K = LW^K, V = LW^V \tag{1}$$

$$A\left(Q, K, V\right) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right) V \tag{2}$$

where $W^Q, W^K, W^V \in \mathbb{R}^{d_h \times d_k}$ are the matrices for linear projection, d_k is the dimension of query / key / value vectors, and F, L are feature matrices of the dense goals and all map elements (*i.e.*, lanes or agents), respectively.

The predicted score of the i_{th} goal can be written as:

$$\phi_i = \frac{\exp(g(F_i))}{\sum_{n=1}^{N} \exp(g(F_n))}, \tag{3}$$

where the trainable function $g(\cdot)$ is also implemented with a 2-layer MLP. The loss term for training the scene context encoding and the dense probability estimation is the binary cross-entropy between the predicted goal scores and the ground truth goal scores:

$$\mathcal{L}_{goal} = \sum_i \mathcal{L}_{CE}(\phi_i, \psi_i), \tag{4}$$

where ψ_i is the ground truth score of the i_{th} goal. The ground truth score of the goal closest to the final position is 1, and the others are 0.

2.3. Goal selection

After the dense probability estimation, we use non-maximum suppression (NMS) algorithm to select goals.

NMS iteratively selects the goal with the highest probability and removes the goals which are close to the selected goal. The first K selected goals are the predicted goals.

2.4. Trajectory completion

Similar to TNT, the last step is to complete each trajectory conditioned on the selected goals. We only have one ground truth trajectory, so we apply a teacher forcing technique [12] by feeding the ground truth goal during training. The loss term is the offset between the predicted trajectory $\hat{s}$ and the ground truth trajectory s:

$$\mathcal{L}_{\text{completion}} = \sum_{t=1}^{T} \mathcal{L}_{\text{reg}}(\hat{s}_t, s_t) \tag{5}$$

where $\mathcal{L}_{\text{reg}}$ is the smooth $l1$ loss between two points.

2.5. Long-term prediction

The previous steps can already achieve good performance in short-term (*e.g.* 3s) motion prediction tasks. However, long-term prediction is still challenging since the probability distribution may diverge into the long future. Inspired from sentence generation in natural language processing, we generate the probability distribution of the goals in an auto-regressive manner, at 3s, 5s and 8s respectively.

Since we aim to roll out dense probability estimation in 3 steps, we develop three branches in our model architecture. The three branches share the same weights for the subgraph module in scene context encoding and have independent weights for other parts, *e.g.* the global graph module in scene context encoding and the dense probability estimation.

With N goal selection at 3s, 5s and 8s auto-regressively, we obtain N^3 goal sets. We sort the top K goal sets according to their probability scores, and then complete them to obtain K trajectories. More concretely, for each goal set, we use the above dense goal encoding module to get the features of the 3 goals. Then the features are passed to the trajectory completion module that is a 2-layer MLP. The output is a full trajectory $[\hat{s}_1, \hat{s}_2, ..., \hat{s}_T]$.

3. Experiments

3.1. Implementation details

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. Since the map in each scenario is quite large, we only encode a submap with a center of (0, 30m) and a radius of 80m.

Following VectorNet [5], the agents are converted into sequences of vectors. Each vector contains the start point, the end point, the timestamps of its start and end point, and the attributes of its corresponding agent. The lanes are converted into sequences of lane segments. Each lane segment contains 10 adjacent lane points and the attributes of its corresponding lane. For example, a lane of 50 lane points is converted into 5 lane segments. Since the sampling distance between two adjacent points is about 1m, a lane segment of 10 lane points is about 10m.

Dense goal sampling. Dense goal sampling aims to sample all possible goals of the target vehicle. Only goal candidates densely located on the roads and the parking lots need to be sampled. The distance between two adjacent goals, *i.e.*, the sampling density, is set to 1m. We don't sample the goals which are located outside of the submap defined above.

Training details. Our model is trained on the training set with a batch size of 64. We use Adam [3] optimizer to train 16 epochs, and the learning rate with an initial value of 0.001 is decayed by a factor of 0.3 every 5 epochs. The hidden size of the feature vectors is set to 128. No data augmentation is used, such as random perturbation or map zooming.

There are three agent types, namely pedestrians, vehicles, and cyclists. We train a model for each of them, since different agent types have different behavioral characteristics.

Table 1. Comparison of sparse and dense goal probability estimation on the Argoverse Forecasting validation set.

Goal Probability Estimation		minFDE	Miss Rate
Sparse	Dense		
✓		1.35	9.5%
	✓	1.28	8.2%

3.2. Results

Sparse and dense goals. We compare and evaluate sparse and dense goal probability estimation on the Argoverse Forecasting dataset. As shown in Table 1, the dense model gives better performance than the sparse model, which is vanilla TNT.

Waymo Open Dataset Motion Prediction Challenge. We evaluate the effectiveness of DenseTNT on the Waymo Open Dataset Motion Prediction Challenge. As shown in Table 2, our method ranks 1^{st} on the leaderboard. The official metric is mAP, which provides a full picture of the model performance [4]. The breakdown performance on each category is shown in Table 3.

Table 2. Top 10 entries of the Waymo Open Dataset Motion Prediction Challenge. mAP is the official ranking metric.

Method	minADE	minFDE	Miss Rate	Overlap Rate	mAP
DenseTNT 1st (Ours)	1.0387	1.5514	0.1573	0.1779	**0.3281**
TVN 2nd	0.7558	1.5859	0.2032	**0.1467**	0.3168
Star Platinum 3rd	0.8102	1.7605	0.2341	0.1774	0.2806
SceneTransformer	**0.6117**	**1.2116**	**0.1564**	0.1473	0.2788
ReCoAt	0.7703	1.6668	0.2437	0.1642	0.2711
AIR	0.8682	1.6691	0.2333	0.1583	0.2596
SimpleCNNOnRaster	0.7400	1.4936	0.2091	0.1560	0.2136
CNN-MultiRegressor	0.8257	1.7101	0.2735	0.1640	0.1944
GOAT	0.7948	1.6838	0.2431	0.1726	0.1930
Waymo LSTM baseline	1.0065	2.3553	0.3750	0.1898	0.1756

Table 3. Performance breakdown of DenseTNT on different categories.

Object Type	minADE	minFDE	Miss Rate	Overlap Rate	mAP
Vehicle	1.3462	1.9120	0.1518	0.1296	0.3698
Pedestrian	0.5013	0.9130	0.1014	0.2725	0.3342
Cyclist	1.2687	1.8292	0.2186	0.1314	0.2802

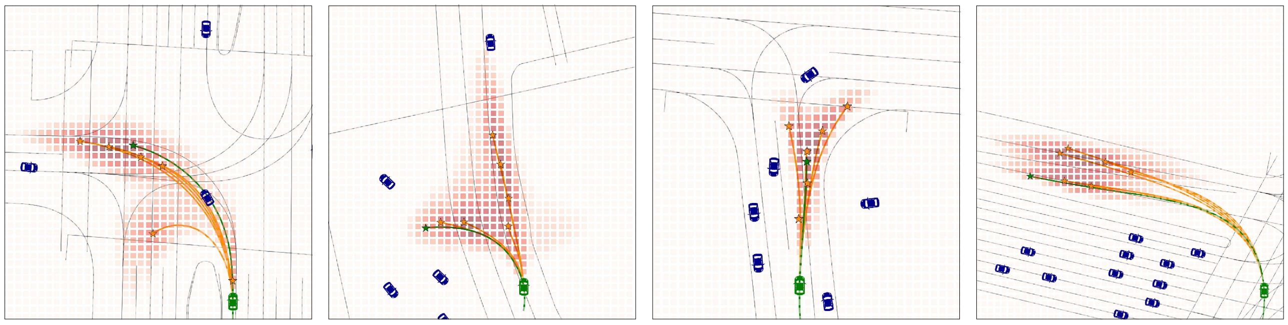

Figure 3. Qualitative results of DenseTNT on the Waymo Open Motion Prediction validation set. The probabilities of the dense goals are shown in red, selected goals and corresponding trajectories are shown in orange, ground truth trajectories are shown in green.

3.3. Qualitative results

In DenseTNT, the goal candidates are densely distributed on the map. We visualize the probabilities of the dense goals and the predicted trajectories based on the selected goals. As shown in Figure 3, DenseTNT gives diverse predictions such as going straight, left/right turns and U-turns.

4. Conclusion

In this report, we propose an anchor-free trajectory prediction model, named DenseTNT. It outperforms previous goal-based methods by removing heuristically predefined goal anchors. DenseTNT achieves state-of-the-art performance and ranks 1st on the Waymo Open Dataset Motion Prediction Challenge.

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. 1

[2] 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

[3] Kingma Da. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[4] 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. 3

[5] 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

[6] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. 1

[7] 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

[8] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision*, pages 541–556. Springer, 2020. 1

[9] 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

[10] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2830, 2019. 1

[11] Hung Tran, Vuong Le, and Truyen Tran. Goal-driven long-term trajectory prediction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 796–805, January 2021. 1

[12] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989. 3

[13] Wenyuan Zeng, Ming Liang, Renjie Liao, and Raquel Urtasun. Lanercnn: Distributed representations for graph-centric motion forecasting. *arXiv preprint arXiv:2101.06653*, 2021. 1

[14] 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, 2