arXiv:2108.09640v2 [cs.CV] 24 Nov 2021

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

Junru Gu[1] Chen Sun[2] Hang Zhao[1*]
[1]IIIS, Tsinghua University [2]Brown University

Abstract

Due to the stochasticity of human behaviors, predicting the future trajectories of road agents is challenging for autonomous driving. Recently, goal-based multi-trajectory prediction methods are proved to be effective, where they first score over-sampled goal candidates and then select a final set from them. However, these methods usually involve goal predictions based on sparse pre-defined anchors and heuristic goal selection algorithms. In this work, we propose an anchor-free and end-to-end trajectory prediction model, named DenseTNT, that directly outputs a set of trajectories from dense goal candidates. In addition, we introduce an offline optimization-based technique to provide multi-future pseudo-labels for our final online model. Experiments show that DenseTNT achieves state-of-the-art performance, ranking 1^{st} on the Argoverse motion forecasting benchmark and being the 1^{st} place winner of the 2021 Waymo Open Dataset Motion Prediction Challenge.[1]

1. Introduction

For a safe and smooth autonomous driving system, an essential technology is to predict the future behaviors of road participants. For example, knowing whether other vehicles intend to cut in better helps us to make brake decisions. However, motion prediction is a highly challenging task 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 [19, 37] and GANs [14]. Other approaches generate a set of trajectories but only perform regression on the closest one during training [14, 21, 8], namely using variety loss. However, sampling-based methods cannot output the likelihood of the predicted futures and the variety loss lacks interpretability on the outputs.

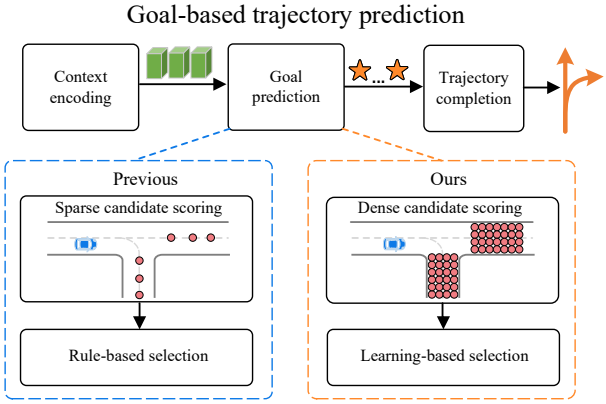


Figure 1: A typical goal-based trajectory prediction pipeline is shown in the upper part of the figure. Existing goal prediction methods (lower left) first define sparse goal anchors heuristically, and regress and classify these anchors to estimate the goals; then rules like non-maximum suppression (NMS) are used for goal selection. In contrast, our method (lower right) estimates the probabilities of dense goal candidates without relying on the heuristic anchors (anchor-free). And it gets rid of rule-based post-processing by generating a set of goals in an end-to-end manner.

More recently, goal-based methods [40, 32, 38] have gained popularity and achieved state-of-the-art performance. Their key observation is that the goal (endpoint) carries most of the uncertainty of a trajectory, therefore they first predict the goal(s) of an agent, and then further complete the corresponding full trajectory for each goal. The final goal positions are obtained by classifying and regressing predefined *sparse anchors*, as shown in the lower-left part of Figure 1. For example, TNT [40] defines anchors as the points sampled on the lane centerlines; some others [38] take the lane segments as anchors and predicted a goal for each lane segment. Another technique commonly adopted by these methods is to apply a rule-based algorithm to select a final small number of goals. The most notable algorithm is non-maximum suppression (NMS) [40], where only locally high-scored goals are selected.

The limitations of these methods are two-folds. First, the prediction performance of these methods heavily depends on the quality of the goal anchors. Since an anchor

*Corresponding to: hangzhao@mail.tsinghua.edu.cn
[1]Project page: https://tsinghua-mars-lab.github.io/DenseTNT

can only generate one goal, a model cannot make multiple trajectory predictions around one anchor. Besides, sparse anchor-based methods cannot capture fine-grained information, *i.e*. different positions on the same lane segment contain different local information, such as the relative distance to the nearest lane boundary. Moreover, after estimating the probability of the sparse goals, NMS is used to heuristically select the goal set, which is a greedy algorithm and is not guaranteed to find the optimal solution given the multimodal nature of the problem.

To address these issues, we propose DenseTNT, an anchor-free and end-to-end multi-trajectory prediction method. DenseTNT first generates *dense* goal candidates with their probabilities from the scene context; from the goal probabilities, it further employs a *goal set predictor* to produce a final set of trajectory goals. Compared to the previous methods, DenseTNT better models goal candidates and gets rid of post-processing.

Goal set prediction in DenseTNT is a multi-label prediction problem and requires multiple labels as training targets. However, unlike object detection, which innately has multiple label boxes as supervision [2], in trajectory prediction we only observe *one* ground truth future out of many possible futures in each training sample, making it extremely challenging to supervise the model. To tackle this problem, we devise an *offline model* to provide multi-future pseudo-labels for our *online model*. Compared with the above online model, the offline model uses an optimization algorithm instead of the goal set predictor for goal set prediction. The optimization algorithm finds an optimal goal set from the probability distribution of the goals; and then the set of goals are used as pseudo-labels for the training of the online model.

DenseTNT achieves state-of-the-art performance in autonomous driving trajectory prediction tasks, ranking 1^{st} on the Argoverse motion forecasting benchmark and 1^{st} in the 2021 Waymo Open Dataset Motion Prediction Challenge.

2. Related Work

Future predictions are highly uncertain because of the unknown intents and behaviors of agents [15, 27, 34, 18, 22, 30, 39] and the complicated interactions among agents [13, 23]. In the field of autonomous driving, to model the high degree of multimodality, implicitly using latent variables is a popular approach [16, 36, 29, 31]. DESIRE [19] used conditional variational autoencoders (CVAEs), and some approaches aimed to address mode collapse [26, 37, 24, 10, 3]. More recently, goal-based multi-trajectory prediction methods have gained popularity due to their superior performance. We will discuss their details later in this section.

Map encoding. The map encoding methods can be divided into two categories: rasterized encoding and vectorized encoding. Rasterized encoding methods rasterize the HD map elements together with agents into an image and use CNNs to encode the image. Based on rasterized encoding, Cui *et al*. [8] went beyond a single trajectory and predicted multiple trajectories as well as their probabilities. IntentNet [4] developed a detector composed of CNNs to extract features from not only raster images but also LiDAR points. Multipath [5] used CNNs to extracts features from raster images, then predicted the probabilities over K predefined anchor trajectories and regressed offsets from the anchor states. Liang *et al*. [20] designed multi-scale location encodings and convolutional RNNs over graphs for map encoding. To capture the uncertainty in long-range human trajectory prediction, Jain *et al*. [17] predicted and updated discretized distribution over spatial locations. These rasterized methods cannot capture the structural information of high-definition maps and do not allow non-grid sampling of goal points due to the shape of convolutions.

Recently, sparse (vectorized) encoding methods, which can better capture the structural information of high-definition maps, have developed rapidly. They treat each entity (a lane or an agent) as a sparse set of elements and use graph neural networks to extract both the features of the entities and the interactions among different entities. VectorNet [11] is the first to directly incorporate vectorized information of both the lanes and the agents. LaneGCN [21] constructed a lane graph and used graph convolutions with adjacency matrices to capture the complex topology of the lane graph. Instead of representing each agent by a feature vector, LaneRCNN [38] proposed a graph-based representation for each agent and captured the interactions among agents by modeling graph-to-graph interactions. Besides, TPCN [35] employed point cloud learning strategies to model. Unlike these vectorized methods which only consider the lane centerlines or lane boundaries of the HD maps, we model dense spatial locations on the roads.

Goal-based trajectory prediction. Rehder *et al*. [25] introduced the pedestrian's goal as a latent variable and thus converted the prediction problem into a planning problem. TNT [40] first sampled anchors from the road maps and generated trajectories conditioned on these anchors. Trajectories were then scored and non-maximum suppression (NMS) was used to select a final set of trajectories. Similar to TNT, the decoding pipeline of LaneRCNN [38] treated a lane segment as an anchor and output each anchor's probability, then used NMS to remove duplicate goals if two predictions are too close. DROGON [7] focused on a different task that the intentional destinations of individual agents are given. They created a trajectory prediction dataset to investigate the goal-oriented behavior and used a conditional VAE framework to forecast multiple possible trajectories. The goal-based idea has also been used in finding the opti-

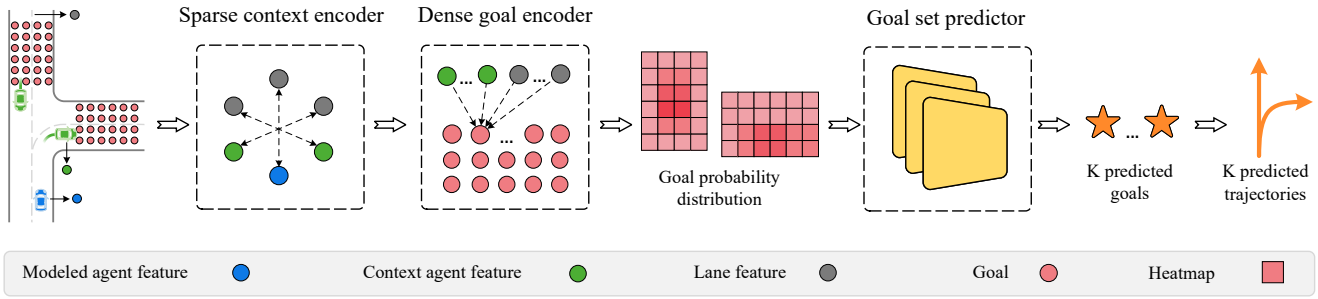


Figure 2: An overview of DenseTNT. A sparse context encoder is used to extract the features of HD maps and agents; and then a dense goal encoder is employed to output dense goal probability distribution; finally a goal set predictor takes the probability distribution of the goals as input and generates a set of predicted goals.

mal planning policy for autonomous driving [1] and in human trajectory prediction [32]. Compared to the previous works, DenseTNT is an anchor-free goal-based model that can be learned in an end-to-end manner. A concurrent work HOME [12] that used CNNs to generate a heatmap and designed greedy algorithms for goal sampling, is very similar to our dense probability estimation.

3. Method

DenseTNT is an anchor-free and end-to-end trajectory prediction method that directly outputs a set of trajectories from dense goal candidates. We first utilize a sparse (vectorized) encoding method to extract features, which captures the structural features of high-definition maps (Section 3.1). Then we employ a dense goal encoder to generate the probability distribution of the goals (Section 3.2). Finally, a goal set predictor takes the probability distribution of the goals as input and generates a set of goals directly (Section 3.3). To train our model, more specifically the goal set predictor, we devise an optimization-based offline model which produces pseudo-labels for supervision.

3.1. Sparse 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 [11, 21] (also called vectorized methods) were proposed recently. Compared to rasterized 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.

We adopt VectorNet [11] in this work for its outstanding performance. VectorNet is a hierarchical graph neural network composed of a subgraph module and a global graph module. The subgraph module is used to encode the features of the lanes and the agents, and the global graph module uses the attention mechanism to capture the interactions among the lanes and the agents. After context encoding, we obtain a 2D feature matrix $\mathbf{L}$, where each row $\mathbf{L}_i$ indicates the feature of the i^{th} map element (*i.e.*, a lane or an agent).

3.2. Dense goal probability estimation

After the sparse context encoding, we perform probability estimation for goals on the map. TNT [40] 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 multiple trajectory predictions around one anchor; (2) sparse anchor-based methods cannot capture fine-grained information, *i.e.* different positions on the same lane segment contain different local information, such as the relative distance to the nearest lane boundary.

Therefore, we perform dense goal probability estimation on the map instead, so that the goal prediction is anchor-free. Concretely, a dense goal encoder is used to extract the features of the locations on the road under a certain sampling rate. Then, the probability distribution of the dense goal candidates is predicted.

Lane scoring. Before goal probability estimation, we adopt a lane scoring module to predict the lane the goal will land on to reduce the number of goal candidates. As a higher level of abstraction, there are tens of goals on each lane. By scoring lanes, we can filter away the goal candidates which are not located on the candidate lanes, reducing computation in the later stage.

The scoring of lanes is modeled as a classification problem, and a binary cross-entropy loss $\mathcal{L}_{\text{lane}}$ is used for training. The ground truth score of the lane closest to the ground truth goal is 1, and the others are 0. The distance between a lane $\mathbf{l}$ and the ground truth goal y_{gt} is defined as $d(\mathbf{l}, y_{\text{gt}}) = \min(||l_1 - y_{\text{gt}}||^2, ||l_2 - y_{\text{gt}}||^2, \ldots, ||l_t - y_{\text{gt}}||^2)$.

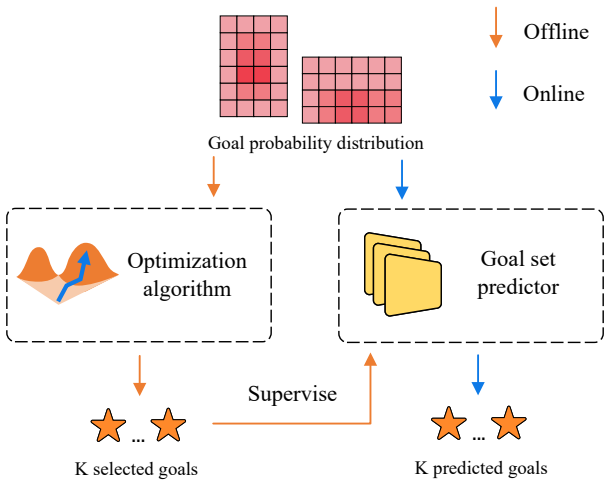
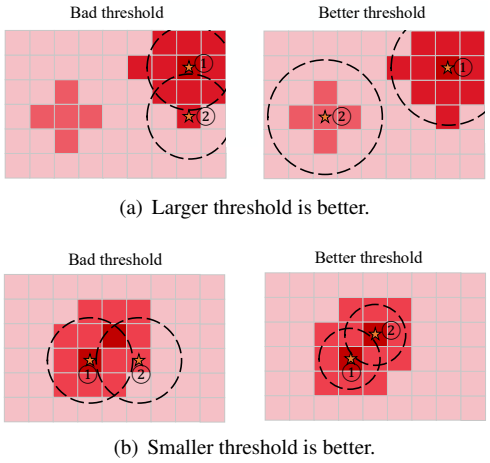


Figure 3: Two-stage training of goal set predictor. In the first stage, we use the ground truth goals to train all the modules except for the goal set predictor. In the second stage, we only train the goal set predictor, using pseudo-labels generated by the optimization algorithm.



(a) Larger threshold is better.

(b) Smaller threshold is better.

Figure 4: NMS results in suboptimal goal selection. The upper example requires a larger threshold while the lower one needs a smaller threshold. Orange stars denote selected goals ($K = 2$) of different NMS thresholds for different heatmaps, NMS threshold is depicted as the radius of the circle.

Probability estimation. The dense goal encoder uses an attention mechanism to extract the local information between the goals and the lanes. We first get the initial feature matrix $\mathbf{F}$ of the goals by encoding their 2D coordinates using MLP. The local information between the goals and the lanes can be obtained by attention mechanism:

$$\mathbf{Q} = \mathbf{F}\mathbf{W}^Q, \mathbf{K} = \mathbf{L}\mathbf{W}^K, \mathbf{V} = \mathbf{L}\mathbf{W}^V, \tag{1}$$

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

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

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

$$\phi_i = \frac{\exp(g(\mathbf{F}_i))}{\sum_{n=1}^{N}\exp(g(\mathbf{F}_n))}, \tag{3}$$

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

$$\mathcal{L}_{\text{goal}} = \mathcal{L}_{\text{CE}}(\boldsymbol{\phi}, \boldsymbol{\psi}). \tag{4}$$

The ground truth score of the goal closest to the final position is 1, and the others are 0.

3.3. Goal set prediction

With the dense probability estimation above, we obtain a heatmap indicating the probability distribution of the final positions of the trajectories. We aim to pick the most likely goals across different modalities, *i.e.* some distinctive peaks in the heatmap. Typical goal-based trajectory prediction pipeline adopts non-maximum suppression (NMS) for goal selection. However, NMS cannot handle various situations flexibly because different heatmaps have different optimal NMS thresholds, as shown in Figure 4.

Our finding is that goal selection can be modeled as a set prediction task, so we design a goal set predictor that takes this heatmap as input and generates the goal set in an end-to-end manner. However, different from object detection, which has multiple label boxes [2], in the trajectory prediction problem, we can only observe one ground truth future out of many possible futures. To tackle this problem, we devise an offline model to provide multi-future pseudo-labels for our online model (more specifically, the goal set predictor). The offline model is composed of the same encoding modules as the online model, but with an optimization algorithm in place of the goal set predictor. In the following, we first introduce the optimization algorithm, and then detail our goal set predictor. The training procedure of the goal set predictor is shown in Figure 3.

Optimization (offline). The heatmap obtained from the above steps is denoted by a mapping h from $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ to $[0, 1] \subset \mathbb{R}$, where $c_i \in \mathbb{R}^2$ is the i^{th} goal on the map. Let Y be random variable of the coordinates of the final position, and its probability distribution satisfies $\mathbb{P}(Y = c_i) = h(c_i)$. Given a predicted goal set

Algorithm 1 Optimization for Offline Goal Set Prediction

Input: a heatmap h that is a mapping from the dense goal candidates $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ to $[0, 1] \subset \mathbb{R}$, indicating the probability distribution of the final position.
Objective: to find a goal set that minimizes $f(\mathbf{y}) = \mathbb{E}[d(\mathbf{y}, Y)] = \sum_{i=1}^{m} h(c_i) d(\mathbf{y}, c_i)$.
The current goal set $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_K\}$ is randomly sampled from the dense goal candidates $\mathcal{C}$, and the current expected error is $f(\hat{\mathbf{y}})$.
for $step \in \{1..\infty\}$ **do**
  if time exceeded **then**
    break
  end if
  for each goal $\hat{y}_i$ **do**
    $\hat{y}_i^{'} = \mathrm{RandomPerturbation}(\hat{y}_i)$
  end for
  $e = f(\hat{\mathbf{y}})$
  $e^{'} = f(\hat{\mathbf{y}}^{'})$
  $r = \mathrm{Random}(0, 1)$
  if $e < e^{'}$ or $r < 0.01$ **then**
    $\hat{\mathbf{y}} = \hat{\mathbf{y}}^{'}$
  end if
end for

$\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_K\}$ and the ground truth goal y_{gt}, the error of $\hat{\mathbf{y}}$ is $d(\hat{\mathbf{y}}, y_{\mathrm{gt}})$, *e.g.*, the minimal final displacement error (FDE) is:

$$d_{\mathrm{FDE}}(\hat{\mathbf{y}}, y_{\mathrm{gt}}) = \min_{y_i \in \hat{\mathbf{y}}} ||y_i - y_{\mathrm{gt}}||. \tag{5}$$

Since we do not know the exact value of y_{gt}, it is difficult to obtain the error $d(\hat{\mathbf{y}}, y_{\mathrm{gt}})$. However, we can use the probability distribution of Y to obtain the expected error of $\hat{\mathbf{y}}$:

$$\mathbb{E}[d(\hat{\mathbf{y}}, Y)] = \sum_{i=1}^{m} h(c_i) d(\hat{\mathbf{y}}, c_i). \tag{6}$$

We define our objective function as $f(\mathbf{y}) = \mathbb{E}[d(\mathbf{y}, Y)]$. Our objective is to find the global optimal solution $\tilde{\mathbf{y}}$ that minimizes $f(\mathbf{y})$. An optimization algorithm is a procedure that is executed iteratively by comparing various solutions till an optimum or a satisfactory solution is found. We adopt a hill climbing algorithm in this paper, which is an iterative algorithm that attempts to make an incremental change to the current solution every step. The detail of this algorithm is described in Algorithm 1. Then we can obtain $\hat{\mathbf{y}}$ that is very close to the global optimal solution $\tilde{\mathbf{y}}$:

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmin}}\, \mathbb{E}[d(\mathbf{y}, Y)], \tag{7}$$

where $\mathcal{Y}$ is the search space of the optimization procedure. Now for each x in the training set, we can use the above steps to generate a heatmap h, and then use the optimization algorithm to get $\hat{\mathbf{y}}$.

Goal set predictor (online). Set predictor was introduced by DETR [2], which views object detection as a set prediction problem and designs a loss based on the Hungarian matching. In this multi-future prediction problem, we treat it as a set prediction problem as well and use the output of the offline model as pseudo-labels to train the goal set predictor of the online model. Instead of performing Hungarian matching between the set of predicted goals and pseudo-labels, we perform offline optimization during training, using each optimized pseudo-label to supervise its corresponding predicted goal.

Let us denote $\dot{\mathbf{y}} = \{\dot{y}_i\}_{i=1}^{K}$ as the set of K predicted goals generated by the goal set predictor at the current training step. We use the above optimization algorithm to generate the pseudo-labels $\hat{\mathbf{y}}$ for this training step. The initial goal set of the optimization algorithm is set to the predicted goal set $\dot{\mathbf{y}}$. The optimization algorithm only searches for the neighbors of $\dot{\mathbf{y}}$ instead of searching for the optimal solution. Specifically, we run the random perturbation for $L(L = 100)$ times to get L goal sets. The pseudo-labels $\hat{\mathbf{y}}$ for the goal set predictor at the current training step is the goal set with the lowest expected error.

The loss term is the offset between the predicted goal set $\dot{\mathbf{y}}$ and the pseudo-labels $\hat{\mathbf{y}}$:

$$\mathcal{L}_{\mathrm{set}}(\dot{\mathbf{y}}, \hat{\mathbf{y}}) = \sum_{i=1}^{K} \mathcal{L}_{\mathrm{reg}}(\dot{y}_i, \hat{y}_i), \tag{8}$$

where $\mathcal{L}_{\mathrm{reg}}$ is the standard ℓ_1 loss between two goals.

Since the probability distribution indicated by the heatmap is diverse, it is difficult for a single regressor to handle. The goal set predictor has multiple heads to predict N goal sets simultaneously. Specifically, every head will predict $2K + 1$ values, including the 2D coordinates of K goals and the confidence of this head. Every head is composed of a heatmap encoder and a decoder. The heatmap encoder is a one-layer self-attention followed by a max-pooling, and the decoder is a two-layer MLP that outputs $2K + 1$ values. The parameters of the heatmap encoders of all heads are shared to reduce computation.

During training, the optimization algorithm only generates pseudo-labels for the head with the lowest expected error and the goal set predictor only performs regression on this head. To predict the confidences of the multiple heads, we use a binary cross-entropy loss:

$$\mathcal{L}_{\mathrm{head}} = \mathcal{L}_{\mathrm{CE}}(\boldsymbol{\mu}, \boldsymbol{\nu}), \tag{9}$$

where $\boldsymbol{\mu}$ is the predicted confidence of the heads, and $\boldsymbol{\nu}$ is the confidence label. $\nu_i = 1$ for the head with the lowest expected error and $\nu_i = 0$ for other heads. During inference,

	Method	minADE	minFDE	Miss Rate
Validation Set	DESIRE [19]	0.92	1.77	18%
	MultiPath [5]	0.80	1.68	14%
	TNT [40]	0.73	1.29	9.3%
	LaneRCNN [38]	0.77	1.19	8.2%
	DenseTNT w/ 100ms optimization	0.80	1.27	**7.0%**
	DenseTNT w/ 100ms optimization (minFDE)	**0.73**	**1.05**	9.8%
	DenseTNT w/ goal set predictor (online)	0.82	1.37	**7.0%**

	Method	minADE	minFDE	Miss Rate
Leaderboard	TNT [40] 8^{th}	0.94	1.54	13.3%
	LaneRCNN [38] 5^{th}	0.90	1.45	12.3%
	SenseTime_AP 4^{th}	0.87	1.36	12.0%
	poly 3^{rd}	**0.87**	1.47	12.0%
	PRIME [28] 2^{nd}	1.22	1.56	11.5%
	Huawei_IOV_France 1^{st}	0.91	1.36	11.2%
	DenseTNT w/ 100ms optimization	0.94	1.49	**10.5%**
	DenseTNT w/ 100ms optimization (minFDE)	0.88	**1.28**	12.6%
	DenseTNT w/ goal set predictor (online)	0.93	1.45	10.7%

Table 1: Model performance on the Argoverse validation set and leaderboard (as of March 16, 2021). Miss Rate is the official ranking metric.

we take the head with the highest confidence as the output of the goal set predictor.

3.4. Trajectory completion

Similar to TNT, the last step is to complete each trajectory conditioned on the predicted goals. We first calculate the feature of each goal similar to the above dense goal encoding, then pass it to the decoder that is a 2-layer MLP. The output of the decoder is the whole trajectory $[\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_T]$.

We only have one ground truth trajectory, so we apply a teacher forcing technique [33] by feeding the ground truth goal during training. The loss term is the offset between the predicted trajectory $\hat{\mathbf{s}}$ and the ground truth trajectory $\mathbf{s}$:

$$\mathcal{L}_{\text{completion}} = \sum_{t=1}^{T} \mathcal{L}_{\text{reg}}(\hat{s}_t, s_t), \tag{10}$$

where $\mathcal{L}_{\text{reg}}$ is the smooth ℓ_1 loss between two points. During inference, this trajectory completion module is used to generate K trajectories of the K goals simultaneously.

3.5. Learning

The training procedure of our method has two stages. In the first stage, we use the ground truth trajectories to train all the modules except for the goal set predictor:

$$\mathcal{L}_{\text{S1}} = \mathcal{L}_{\text{lane}} + \mathcal{L}_{\text{goal}} + \mathcal{L}_{\text{completion}}. \tag{11}$$

In the second stage, we train the goal set predictor on the training set, which is supervised by the pseudo-labels generated by the offline model (encoding + optimization algorithm):

$$\mathcal{L}_{\text{S2}} = \mathcal{L}_{\text{head}} + \mathcal{L}_{\text{set}}. \tag{12}$$

4. Experiments

4.1. Datasets

Argoverse forecasting dataset. Argoverse forecasting dataset [6] is a dataset with agent trajectories and high-definition maps. Given the trajectory of the target vehicle in the past two seconds, which are sampled at 10Hz, we need to predict the future trajectory in the next 3 seconds. There are 333K real-world driving sequences that are at intersections or in dense traffic, and each sequence contains one target vehicle for prediction. The training, validation, and test sets contain 205942, 39472, and 78143 sequences, respectively.

Waymo open motion dataset. Waymo open motion dataset [9] is by far the most diverse interactive motion dataset. It contains more than 570 hours of unique data over 1750km of roadways with over 100,000 scenes, each 20 seconds long. There are three types of agents in the dataset, namely vehicles, pedestrians and cyclists. Given a 1-second history trajectory of the target agent, an 8-second future trajectory needs to be predicted.

Metrics. We follow the Argoverse benchmark and use minimum average displacement error (minADE), minimum final displacement error (minFDE), and miss rate (MR).

End-to-end	Probability estimation		Goal selection	Optimization Objective					
	Sparse	Dense		minADE	minFDE	MR	minADE	minFDE	MR
✓			variety loss	0.78	1.25	13.3%	0.78	1.25	13.3%
	✓		NMS	0.81	1.33	10.1%	0.82	1.35	9.5%
		✓	NMS	0.79	1.26	8.6%	0.79	1.28	8.2%
		✓	optimization	0.73	**1.05**	9.8%	0.80	1.28	**7.0%**
✓		✓	goal set predictor	0.75	**1.05**	9.7%	0.82	1.37	**7.0%**

Table 2: Ablation studies on the main components in our method: dense probability estimation, goal selection, *etc*. We also tested the effectiveness of our method under different optimization objectives, minFDE and MR.

Each trajectory is represented by a sequence of points over time. ADE is the average displacement between each point of the predicted trajectory and its corresponding ground-truth point. minADE is the minimum ADE of the predicted K trajectories, and minFDE is the minimum displacement between K final positions and the ground truth final position. Miss rate is the ratio of scenarios where none of the predicted trajectories are within 2.0 meters of ground truth according to the final displacement error.

4.2. Implementation Details

Goal candidate sampling. We first sample the lanes within 50m (Manhattan distance) from the target vehicle. Then we sample goal candidates which are densely distributed on these lanes. Therefore, the number of the sampled goal candidates depends on the lanes around the target vehicle. For the lanes represented by the lane centerlines, the goal candidates within 3m from the centerlines are sampled, while for the lanes represented by lane boundaries, the goal candidates within the boundaries are sampled. The distance between two adjacent goals, *i.e*., the sampling density, is set to 1m.

Training details. Our model is trained on the training set with a batch size of 64. In the first stage, we train all the modules except for the goal set predictor for 16 epochs, and the learning rate with an initial value of 0.001 decays to 30% every 5 epochs. In the second stage, we train the goal set predictor for 6 epochs, and the learning rate with an initial value of 0.001 decays to 30% every epoch. The hidden size of the feature vectors is set to 128. The head number of our goal set predictor is 12. No data augmentation is used.

4.3. Results on benchmarks

Argoverse motion forecasting benchmark. We evaluate DenseTNT on the Argoverse validation set, and report results in Table 1. As can be seen, DenseTNT greatly outperforms popular models in the literature. It is also worth noticing that our online model (DenseTNT w/ goal set predictor), though trained from the pseudo-labels provided by the offline model (DenseTNT w/ optimization), achieves comparable results as the offline model. We further compare DenseTNT with the top performers on the Argoverse leaderboard in Table 1. Since the details of the 1st, 3rd and 4th methods were undisclosed, we could not compare them qualitatively. Our method can generate the trajectories in an end-to-end manner during real-time usage, in contrast with PRIME (2nd) and LaneRCNN (5th) that use NMS to perform post-processing. We achieve superior performance on the official ranking metric MR, which verifies the effectiveness of our method. For another popular metric minFDE, we can also achieve state-of-the-art performance by using it as the optimization objective.

Figure 5 shows the qualitative results generated by our online model. The probability distribution of the goals in some cases is quite multimodal, making it difficult for NMS to handle in the post-processing stage. Our model makes diverse trajectory predictions with high coverage of the heatmaps.

Method	mADE	mFDE	MR	mAP
DenseTNT 1^{st} (Ours)	1.0387	1.5514	0.1779	**0.3281**
TVN 2^{nd}	0.7558	1.5859	0.2032	0.3168
Star Platinum 3^{rd}	0.8102	1.7605	0.2341	0.2806
SceneTransformer[23]	0.6117	1.2116	0.1564	0.2788
ReCoAt	0.7703	1.6668	0.2437	0.2711

Table 3: Top 5 entries of the 2021 Waymo Open Dataset Motion Prediction Challenge. mAP is the official ranking metric.

Waymo Open Dataset Motion Prediction Challenge. We developed a variant of DenseTNT for the 2021 Waymo Open Dataset Motion Prediction Challenge, and got the 1^{st} place. The challenge leaderboard is shown in Table 3. Details of this variant are discussed in our technical report[2].

4.4. Ablation Study

Model architecture. We conduct ablation studies on the main components of our model. These components are the

[2]Available on https://waymo.com/open/challenges.

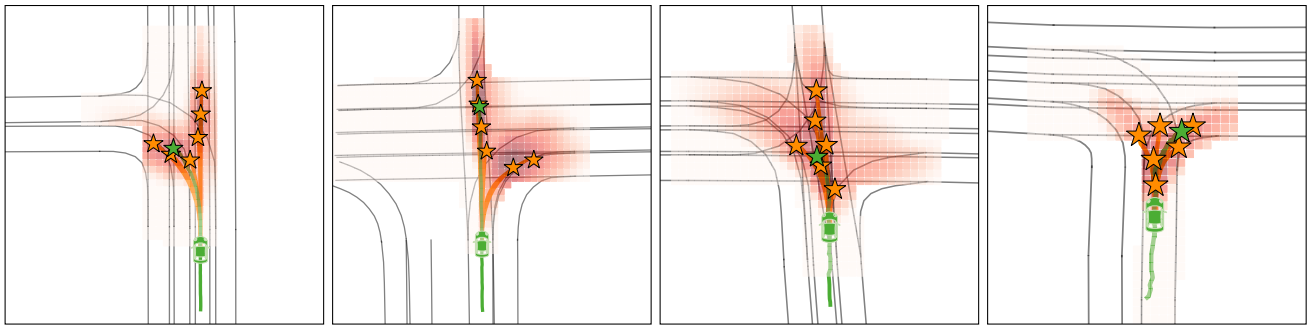


Figure 5: Qualitative results of DenseTNT (online). Dense predicted heatmaps are shown in red, predicted goal sets and corresponding trajectories are shown in orange, ground truth trajectories are shown in green.

dense probability estimation, the optimization algorithm for generating pseudo-labels, and the goal set predictor. There are different metrics to measure the performance of generating the most likely trajectories. We tested the effectiveness of our method under different optimization objectives, as shown in Table 2.

Each component plays an important role in our method. The dense probability estimation performs much better than the sparse probability estimation, since the dense probability estimation provides more fine-grained local information. Moreover, the sparse probability estimation can only be combined with NMS which is a heuristic rule-based algorithm. The hyper-parameter of NMS is the threshold of removing neighboring points, *i.e.*, two points with a distance less than the threshold are regarded as the same point. For a fair comparison, we show the best result of NMS under different metrics. The result of the online model is almost the same as the offline model, proving the effectiveness of the goal set predictor. Variety loss is a traditional end-to-end trajectory prediction method, which generates a fixed number of trajectories but only performs regression on the closest one during training. Our end-to-end method outperforms it by a large margin.

Goal density. To denote the probability distribution of the final position, we densely sample goal candidates on the lanes. The sampling density of the goals has an impact on our method's performance, and we show it in Table 4. It indicates that a higher density leads to better performance until the saturation point is reached.

Sampling density	minFDE	MR
3.0m	1.42	12.5%
2.0m	1.34	9.0%
1.0m	1.27	7.0%
0.5m	1.27	7.0%

Table 4: Comparison of different goal sampling densities on the Argoverse validation set.

Optimization time	minFDE	MR
20ms	1.29	7.6%
50ms	1.28	7.2%
100ms	1.27	7.0%
200ms	1.27	6.9%
500ms	1.27	6.9%

Table 5: Performance under different optimization time per sample on the Argoverse validation set.

Optimization. Given a heatmap indicating the probability distribution of the vehicle's final position, the optimization algorithm is used to find the global optimal solution. The maximum running time of the optimization algorithm per instance has an impact on the performance. Table 5 shows the optimization performance over time. The performance increases drastically before $t = 100\text{ms}$ and remains almost unchanged after $t = 200\text{ms}$.

5. Conclusion

In this paper, we propose an anchor-free and end-to-end trajectory prediction model, named DenseTNT, that directly outputs a set of trajectories from dense goal candidates. In addition, we introduce an optimization-based offline model to provide multi-future pseudo-labels to train the online model. DenseTNT not only runs online, but also has similar performance as the offline model, demonstrating the effectiveness of the goal set predictor design and our training paradigm. Comprehensive experiments show that DenseTNT achieves state-of-the-art performance, ranking 1^{st} on the Argoverse motion forecasting benchmark and being the 1^{st} place winner of the 2021 Waymo Open Dataset Motion Prediction Challenge.

Acknowledgments. We would like to thank Dr. Jiyang Gao and Chenzhuang Du for helpful discussions on the manuscript.

References

[1] Stefano V Albrecht, Cillian Brewitt, John Wilhelm, Balint Gyevnar, Francisco Eiras, Mihai Dobre, and Subramanian Ramamoorthy. Interpretable goal-based prediction and planning for autonomous driving. *arXiv preprint arXiv:2002.02277*, 2020. 3

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020. 2, 4, 5

[3] S. Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and R. Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *ECCV*, 2020. 2

[4] 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. 2

[5] 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, 6, 11

[6] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 6

[7] Chiho Choi, Abhishek Patil, and Srikanth Malla. Drogon: A causal reasoning framework for future trajectory forecast. *arXiv preprint arXiv:1908.00024*, 2019. 2

[8] 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, 2

[9] 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. 6

[10] 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. 2

[11] 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. 2, 3, 11

[12] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. *arXiv preprint arXiv:2105.10968*, 2021. 3

[13] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Thomas: Trajectory heatmap output with learned multi-agent sampling. *arXiv preprint arXiv:2110.06607*, 2021. 2

[14] 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

[15] Irtiza Hasan, Francesco Setti, Theodore Tsesmelis, Alessio Del Bue, Fabio Galasso, and Marco Cristani. Mx-lstm: mixing tracklets and vislets to jointly forecast trajectories and head poses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6067–6076, 2018. 2

[16] 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. 2

[17] Ajay Jain, S. Casas, Renjie Liao, Y. Xiong, Song Feng, Sean Segal, and R. Urtasun. Discrete residual flow for probabilistic pedestrian behavior prediction. In *CoRL*, 2019. 2

[18] Siddhesh Khandelwal, William Qi, Jagjeet Singh, Andrew Hartnett, and Deva Ramanan. What-if motion prediction for autonomous driving. *arXiv preprint arXiv:2008.10587*, 2020. 2

[19] 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, 2, 6, 11

[20] Junwei Liang, Lu Jiang, Kevin Murphy, Ting Yu, and Alexander Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10508–10518, 2020. 2

[21] 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, 2, 3

[22] Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. Mantra: Memory augmented networks for multiple trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7143–7152, 2020. 2

[23] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified multi-task model for behavior prediction and planning. *arXiv preprint arXiv:2106.08417*, 2021. 2, 7

[24] 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. 2

[25] Eike Rehder and Horst Kloeden. Goal-directed pedestrian prediction. In *Proceedings of the IEEE International Con-*

ference on Computer Vision Workshops, pages 50–58, 2015. 2

[26] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018. 2

[27] 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. 2

[28] Haoran Song, Di Luan, Wenchao Ding, Michael Yu Wang, and Qifeng Chen. Learning to predict vehicle trajectories with model-based planning. *arXiv preprint arXiv:2103.04027*, 2021. 6

[29] Chen Sun, Per Karlsson, Jiajun Wu, Joshua B Tenenbaum, and Kevin Murphy. Stochastic prediction of multi-agent interactions from partial observations. *arXiv preprint arXiv:1902.09641*, 2019. 2

[30] Jianhua Sun, Qinhong Jiang, and Cewu Lu. Recursive social behavior graph for trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 660–669, 2020. 2

[31] Yichuan Charlie Tang and Ruslan Salakhutdinov. Multiple futures prediction. *arXiv preprint arXiv:1911.00997*, 2019. 2

[32] 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*, pages 796–805, 2021. 1, 3

[33] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989. 6

[34] Yu Yao, Mingze Xu, Chiho Choi, David J Crandall, Ella M Atkins, and Behzad Dariush. Egocentric vision-based future vehicle localization for intelligent driving assistance systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9711–9717. IEEE, 2019. 2

[35] Maosheng Ye, Tongyi Cao, and Qifeng Chen. Tpcn: Temporal point cloud networks for motion forecasting. *arXiv preprint arXiv:2103.03067*, 2021. 2

[36] Raymond A Yeh, Alexander G Schwing, Jonathan Huang, and Kevin Murphy. Diverse generation for multi-agent sports games. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4610–4619, 2019. 2

[37] Ye Yuan and Kris M Kitani. Diverse trajectory forecasting with determinantal point processes. In *International Conference on Learning Representations*, 2019. 1, 2

[38] Wenyuan Zeng, Ming Liang, Renjie Liao, and Raquel Urtasun. Lanercnn: Distributed representations for graph-centric motion forecasting. *arXiv preprint arXiv:2101.06653*, 2021. 1, 2, 6, 11

[39] Zhishuai Zhang, Jiyang Gao, Junhua Mao, Yukai Liu, Dragomir Anguelov, and Congcong Li. Stinet: Spatio-temporal-interactive network for pedestrian detection and trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11346–11355, 2020. 2

[40] 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, 3, 6, 11

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 6.

B. 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. Following VectorNet [11], 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.

Optimization algorithm. The optimization algorithm aims to find a goal set that minimizes the expected 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 expected 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 expected 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 expected error, we divide each heatmap goal to 9 heatmap goals with a probability of $\frac{1}{9}h(c_i)$, and each of them represents a space of $\frac{1}{3}m \times \frac{1}{3}m$.

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. Qualitative Results

Figure 6 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 set in Figure 7. The probability distribution of the final position in some cases is pretty diverse, and it is difficult for NMS to handle well.

	Method			minADE	minFDE	Miss Rate
Validation Set	DESIRE [19]			0.92	1.77	18%
	MultiPath [5]			0.80	1.68	14%
	TNT [40]			0.73	1.29	9.3%
	LaneRCNN [38]			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 6: Performance of the optimization algorithm under different combinations of optimization objectives.

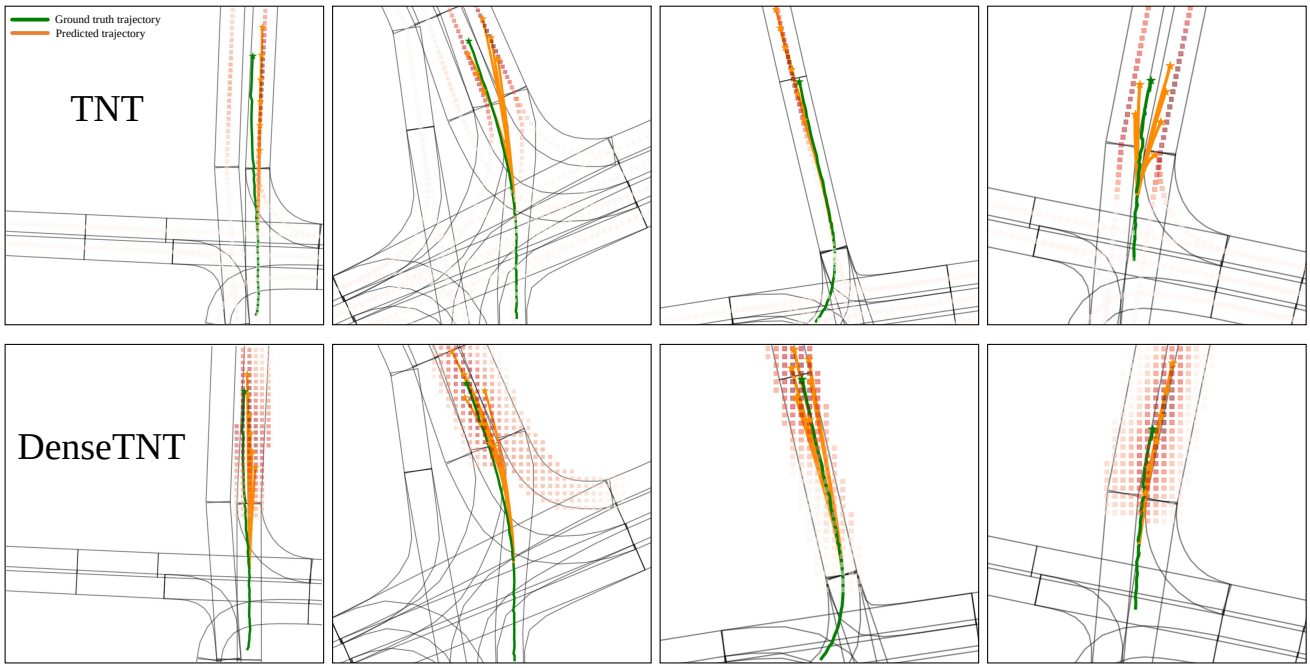


Figure 6: Qualitative comparisons between TNT (the upper row) and our DenseTNT (online, the lower row) on the Argoverse validation set. Here, we choose TNT as a representative for typical goal-based trajectory prediction methods, of which performance heavily depends on the quality of heuristically predefined anchors. Both the anchors of TNT and the heatmaps of DenseTNT are shown in red. Predicted trajectories are shown in orange, and ground truth trajectories are shown in green.

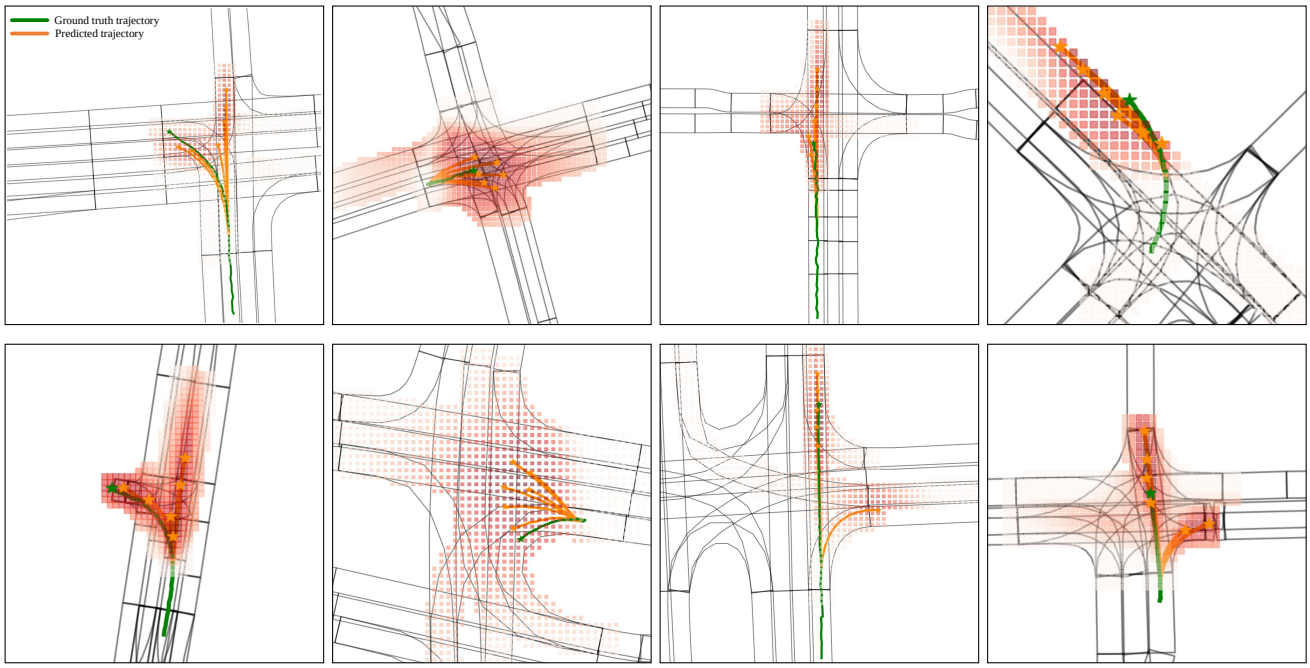


Figure 7: Qualitative results of DenseTNT (online) in diverse traffic scenarios on the Argoverse validation set. Dense predicted heatmaps are shown in red, predicted goal sets and corresponding trajectories are shown in orange, ground truth trajectories are shown in green.