TNT: Target-driveN Trajectory Prediction

Hang Zhao[1] * **Jiyang Gao**[1] * **Tian Lan**[1] **Chen Sun**[2] **Benjamin Sapp** [1]
Balakrishnan Varadarajan[1] **Yue Shen**[1] **Yi Shen**[1] **Yuning Chai**[1]
Cordelia Schmid[2] **Congcong Li**[1] **Dragomir Anguelov**[1]

[1]Waymo LLC [2]Google Research

Abstract: Predicting the future behavior of moving agents is essential for real world applications. It is challenging as the intent of the agent and the corresponding behavior is unknown and intrinsically multimodal. Our key insight is that for prediction within a moderate time horizon, the future modes can be effectively captured by a set of target states. This leads to our target-driven trajectory prediction (TNT) framework. TNT has three stages which are trained end-to-end. It first predicts an agent's potential target states T steps into the future, by encoding its interactions with the environment and the other agents. TNT then generates trajectory state sequences conditioned on targets. A final stage estimates trajectory likelihoods and a final compact set of trajectory predictions is selected. This is in contrast to previous work which models agent intents as latent variables, and relies on test-time sampling to generate diverse trajectories. We benchmark TNT on trajectory prediction of vehicles and pedestrians, where we outperform state-of-the-art on Argoverse Forecasting, INTERACTION, Stanford Drone and an in-house Pedestrian-at-Intersection dataset.

Keywords: Trajectory prediction, multimodal prediction.

1 Introduction

Predicting the future states of moving agents in a real-world environment is an important and fundamental problem in robotics. For example, in the setting of autonomous driving on public roads, it is essential to have an accurate understanding of where the other vehicles and pedestrians will likely be in the future, in order for an autonomous vehicle to take safe and effective actions.

A key challenge to future prediction is the high degree of uncertainty, in large part due to not knowing the intents and latent characteristics of the other agents. For example, a vehicle commonly has a multimodal distribution of futures: it could turn, go straight, slow down, speed up, *etc*. Depending on other scene elements, it could pass, yield, change lanes, or pull into a driveway. This challenge has garnered a lot of interest in the past few years. One approach to model the high degree of multimodality is to employ flexible implicit distributions from which samples can be drawn—conditional variational autoencoders (CVAEs) [1], generative adversarial networks (GANs) [2], and single-step policy roll-out methods [3]. Despite their competitive performance, the use of latent variables to model intents prohibits them to be interpreted, and often requires test-time sampling to evaluate probabilistic queries (*e.g*., "how likely is the agent to turn left?"). Furthermore, considerable effort has gone into addressing mode collapse in such models, in the machine learning community at large [4] and specifically for self-driving cars [5, 6].

To address these limitations, we make the observation that for our task (*e.g*. vehicle and pedestrian trajectory prediction), the uncertainties over a moderately long future can be mostly captured by the prediction of possible *targets* of the agents. These targets are not only grounded in physical entities that are interpretable (*e.g*. location), but also correlate well with intent (*e.g*. a lane change or a right turn). We conjecture that the space of *targets* can be discretized in a scene — allowing a deterministic model to generate diverse targets in parallel — and later refined to be more accurate.

*Equal contribution. Correspond to {hangz, jiyanggao}@waymo.com.

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

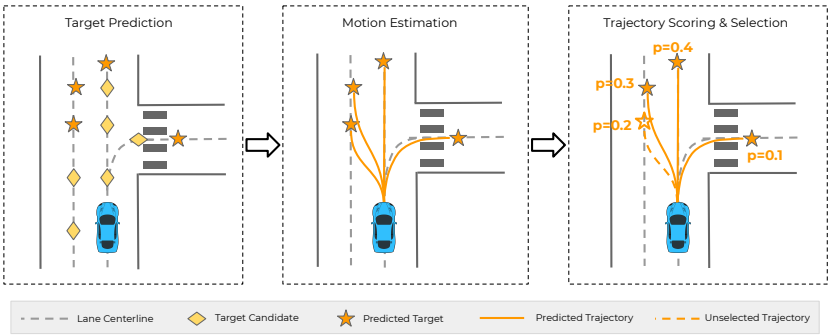

Figure 1: Illustration of the TNT framework when applied to the vehicle future trajectory prediction task. TNT consists of three stages: (a) *target prediction* which proposes a set of plausible targets (stars) among all candidates (diamonds). (b) *target-conditioned motion estimation* which estimates a trajectory (distribution) towards each selected target, (c) *scoring and selection* which ranks trajectory hypotheses and selects a final set of trajectory predictions with likelihood scores.

These observations lead to our proposed *target-driven trajectory prediction* framework, named TNT. We first cast the future prediction problem into predicting a distribution over discretized *target* states, and then formulate a probabilistic model in which trajectory estimation and likelihood are conditioned on such targets. The resulting framework has three stages that are trained end-to-end: (1) *target prediction* estimates a distribution over candidate targets given scene context; (2) *target-conditioned motion estimation* predicts trajectory state sequences per target; and (3) *scoring and selection* estimates the likelihood of each predicted trajectory, taking into account the context of all other predicted trajectories. We obtain a final set of compact diverse predictions by ranking the likelihoods and suppressing redundant trajectories. An illustration of our three-stage model when applied to vehicle trajectory prediction is shown in Figure 1. Although our model is end-to-end trained, its three stage layout, with interpretable outputs at each stage, closely follows the typical processing steps in traditional robotics motion forecasting and planning systems [7, 8], thus making it easy to incorporate domain knowledge during deployment.

We demonstrate the effectiveness of TNT on multiple challenging trajectory prediction benchmarks. In the driving domain we evaluate on the Argoverse Forecasting dataset [9] and INTERACTION dataset [10]; for pedestrians, the Stanford Drone dataset [11] and an in-house Pedestrian-at-Intersection dataset. We achieved the state-of-the-art performance on all benchmarks.

2 Related Work

Trajectory prediction has received much attention recently, especially in autonomous driving [9, 10, 12, 13, 14], in social interaction prediction [15, 16, 17, 18] and sports [19, 20]. One of the key challenges is to model multimodal futures distributions. A popular approach is to model the future modes implicitly as latent variables [1, 3, 2, 21, 22, 23, 24, 25], which aims at capturing the underlying intents of the agents. For example, DESIRE [1] used a conditional VAE [26] while PRECOG [3] used flow-based generative models [27]; SocialGAN [2] proposed an adversarial discriminator to predict realistic futures; Hong *et al*. [22] modeled the motion patterns with a latent Gaussian mixture model. However, the use of non-interpretable, latent variables makes it challenging to incorporate expert knowledge into the system. Furthermore, these models require stochastic sampling from the latent space to obtain implicit distributions at run time. These properties make them less suitable for practical deployment.

Alternatively, some approaches attempted to decompose the trajectory prediction task into subtasks, with the hope that each subtask is more manageable to solve and provides interpretable intermediate results. For example, Ziebart *et al*. [28] proposed planning-based prediction for pedestrians, they first estimated a Bayesian posterior distribution of destinations, and then used inverse reinforcement learning (IRL) to plan the trajectories. Rehder *et al*. [29] introduced the notion of *goals* which are defined as short-term destinations, and decomposed the problem into goal distribution estimation and goal-directed planning. The goals were defined as mixture of Gaussian latent variables. Their followup work [30] then demonstrated that the whole framework can be jointly trained via IRL.

Concurrent to our work, Mangalam *et al*. [31] proposed to generate *endpoints* to guide the full trajectory generation. Unlike TNT, their method still relies on latent variables in CVAE to model the underlying modes of the *endpoints*.

Most related to TNT are works that discretize the output space as *intents* [32] or with *anchors* [33, 34]. IntentNet [32] manually defined several common motion categories for self-driving vehicles, such as left turn and lane changes, and learned a separate motion predictor for each intent. This manual categorization is task and dataset dependent, and may be too coarse to capture intra-category multimodality. More recently, MultiPath [33] and CoverNet [34] chose to quantize the trajectories into *anchors*, where the trajectory prediction task is reformulated into anchor selection and offset regression. The anchors are either pre-clustered into a fixed set *a priori* [33] or obtained dynamically based on kinematic heuristics [34]. Unlike anchor trajectories, the targets in TNT are much lower dimensional and can be easily discretized via uniform sampling or based on expert knowledge (*e.g*. HD maps). Hence, they can be estimated more reliably. Despite their simplicity, we demonstrate that the targets are informative enough to capture most of the uncertainty in predicting future state, and our target-driven framework outperforms the anchor-based methods.

3 Formulation

Given a sequence of observed states for a single agent $\mathbf{s}_P = [s_{-T'+1}, s_{-T'+2}, ..., s_0]$, our goal is to predict its future states $\mathbf{s}_F = [s_1, s_2, ..., s_T]$ up to some fixed time step T. Naturally, the agent interacts with an environment consisting of other agents and scene elements for context: $\mathbf{c}_P = [c_{-T'+1}, c_{-T'+2}, ..., c_0]$. We denote $\mathbf{x} = (\mathbf{s}_P, \mathbf{c}_P)$ for brevity, thus the overall probabilistic distribution we want to capture is $p(\mathbf{s}_F|\mathbf{x})$.

In practice, $p(\mathbf{s}_F|\mathbf{x})$ can be highly multimodal. For example, a vehicle approaching an intersection could turn left, go straight or change lanes. Intuitively, the uncertainty of future states can be decomposed into two parts: the *target or intent uncertainty*, such as the decision between turning left and right; and the *control uncertainty*, such as the fine-grained motion required to perform a turn. We can therefore decompose the probabilistic distribution accordingly by conditioning on targets and then marginalizing over them:

$$p(\mathbf{s}_F|\mathbf{x}) = \int_{\tau \in \mathcal{T}(\mathbf{c}_P)} p(\tau|\mathbf{x}) p(\mathbf{s}_F|\tau, \mathbf{x}) d\tau, \tag{1}$$

where $\mathcal{T}(\mathbf{c}_P)$ represents the space of plausible targets depending on the observed context $\mathbf{c}_P$.

Under this formulation, our main insight is that, for applications such as trajectory prediction, by properly designing the target space $\mathcal{T}(\mathbf{c}_P)$ (*e.g*. target locations), the target distribution $p(\tau|\mathbf{x})$ can well capture the intent uncertainty. Once the target is decided, we further demonstrate that the control uncertainty (*e.g*. trajectories) can be reliably modeled by simple, unimodal distributions. We approximate the target space $\mathcal{T}(\mathbf{c}_P)$ by a set of discrete locations, turning the estimation of $p(\tau|\mathbf{x})$ primarily into a classification task. Compared with latent variational models, our model offers better interpretability in the form of explicit target distributions, and can naturally incorporate expert knowledge (such as road topology), when designing the target space $\mathcal{T}(\mathbf{c}_P)$.

Our overall framework has three conceptual stages. The first stage is **target prediction**, whose goal is to model the intent uncertainty with a discrete set of target states $\mathcal{T}$ based on the observed context $\mathbf{x}$, and outputs the target distribution $p(\tau|\mathbf{x})$. The second stage is **target-conditioned motion estimation**, which models the possible future motions from the initial state to the target with a unimodal distribution. The first two stages give rise to the following probabilistic predictions $p(\mathbf{s}_F|\mathbf{x}) = \sum_{\tau \in \mathcal{T}(\mathbf{c}_P)} p(\tau|\mathbf{x}) p(\mathbf{s}_F|\tau, \mathbf{x})$.

Many downstream applications, such as real-time behavior prediction, require a small set of representative future predictions rather than the full distribution of all possible futures. Our final stage, **scoring and selection**, is tailored for this purpose. We learn a scoring function $\phi(\mathbf{s}_F)$ over all representative predictions, and select a final diversified set of predictions.

4 Target-driveN Trajectory Prediction

This section describes our proposed TNT framework in detail. We focus on the task of future trajectory prediction for moving road agents, where both the states and the targets are represented by

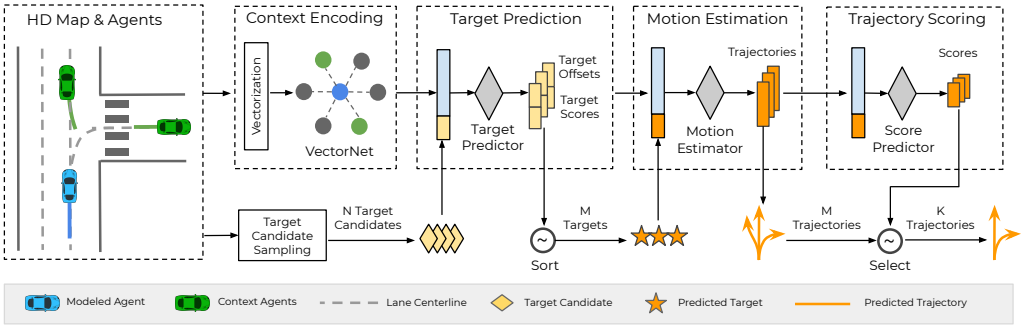

Figure 2: TNT model overview. Scene context is first encoded as the model's inputs. Then follows the core three stages of TNT: (a) *target prediction* which proposes an initial set of M targets; (b) *target-conditioned motion estimation* which estimates a trajectory for each target; (c) *scoring and selection* which ranks trajectory hypotheses and outputs a final set of K predicted trajectories.

their physical locations (x_t, y_t). We begin the section by describing how the context information is encoded efficiently. We then present details on how the proposed three stages are adapted to the task. An overview of the TNT model architecture is shown in Figure 2.

4.1 Scene context encoding

Modeling scene context is a first step in trajectory prediction so as to capture agent-road and agent-agent interactions. TNT can use any suitable context encoder: when the HD map is available, we use a state-of-the-art hierarchical graph neural network VectorNet [35] to encode the context. Specifically, polylines are used to abstract the HD map elements $\mathbf{c}_P$ (lanes, traffic signs) and agent trajectories $\mathbf{s}_P$; a subgraph network is applied to encode each polyline, which contains a variable number of vectors; then a global graph is used to model the interactions between polylines. The output is a global context feature $\mathbf{x}$ for each modeled agent. If scene context is only available in the form of top-down imagery, a ConvNet is used as the context encoder.

4.2 Target prediction

In our formulation, targets τ are defined as the locations (x, y) an agent is likely to be at a fixed time horizon T. In the first *target prediction* stage, we aim to provide a distribution of future targets of an agent $p(\mathcal{T}|\mathbf{x})$. We model the potential future targets via a set of N discrete, quantized locations with continuous offsets: $\mathcal{T} = \{\tau^n\} = \{(x^n, y^n) + (\Delta x^n, \Delta y^n)\}_{n=1}^N$. The distribution over targets can then be modeled via a discrete-continuous factorization:

$$p(\tau^n|\mathbf{x}) = \pi(\tau^n|\mathbf{x}) \cdot \mathcal{N}(\Delta x^n \mid \nu_x^n(\mathbf{x})) \cdot \mathcal{N}(\Delta y^n \mid \nu_y^n(\mathbf{x})), \tag{2}$$

where $\pi(\tau^n|\mathbf{x}) = \exp f(\tau^n, \mathbf{x}) / \sum_{\tau'} \exp f(\tau', \mathbf{x})$ is a discrete distribution over location choices (x^n, y^n). The terms $\mathcal{N}(\cdot|\nu(\cdot))$ denote a generalized normal distribution, where we choose Huber as the distance function. We denote the mean as $\nu(\cdot)$ and assume unit variance.

The trainable functions $f(\cdot)$ and $\nu(\cdot)$ are implemented with a 2-layer multilayer perceptron (MLP), with target coordinates (x^k, y^k) and the scene context feature $\mathbf{x}$ as inputs. They predict a discrete distribution over target locations and their most likely offsets. The loss function for training this stage is given by

$$\mathcal{L}_{\text{S1}} = \mathcal{L}_{\text{cls}}(\pi, u) + \mathcal{L}_{\text{offset}}(\nu_x, \nu_y, \Delta x^u, \Delta y^u), \tag{3}$$

where $\mathcal{L}_{\text{cls}}$ is cross entropy, $\mathcal{L}_{\text{offset}}$ is the Huber loss; u is the target closest to the ground truth location, and $\Delta x^u, \Delta y^u$ are the spatial offsets of u from the ground truth.

The choice of the discrete target space is flexible across different applications. In the vehicle trajectory prediction problem, we sample points on lane centerlines from HD maps and use them as target candidates, with the assumption that vehicles never depart far away from lanes; for pedestrians, we generate a virtual grid around the agent and use the grid points as targets. Comparing to regression, the most prominent advantage of modeling the future as a discrete set of targets is that it does not suffer from mode averaging, which is the major factor that hampers multimodal predictions.

In practice, we over-sample a large number of target candidates as input to this stage, *e.g.* $N = 1000$, to increase the coverage of the potential future locations; and then keep a smaller number of them as output, *e.g.* top $M = 50$, for further processing, as a good choice of M helps to balance between target recall and model efficiency.

4.3 Target-conditioned motion estimation

In the second stage, we model the likelihood of a trajectory given a target as $p(\mathbf{s}_F|\tau, \mathbf{x}) = \prod_{t=1}^{T} p(s_t|\tau, \mathbf{x})$, again with a generalized normal distribution. This makes two assumptions. First, future time steps are conditionally independent, which makes our model computationally efficient by avoiding sequential predictions, as is done in [33, 34, 21, 31]. Second, we are making strong but reasonable assumption that the distribution of the trajectories is unimodal (normal) given the target. This is certainly true for short time horizons; for longer time horizons, one could iterate between (*intermediate*) target prediction and motion estimation so that the assumption still holds.

This stage is implemented with a 2-layer MLP. It takes context feature $\mathbf{x}$ and a target location τ as input, and outputs one most likely future trajectory $[\hat{s}_1, ..., \hat{s}_T]$ per target. Since it is conditioned on the predicted targets from the first stage, to enable a smooth learning process, we apply a teacher forcing technique [36] at training time by feeding the ground truth location (x^u, y^u) as target. The loss term for this stage is the distance between between predicted states $\hat{\mathbf{s}}_t$ and ground truth $\mathbf{s}_t$:

$$\mathcal{L}_{\text{S2}} = \sum_{t=1}^{T} \mathcal{L}_{\text{reg}}(\hat{\mathbf{s}}_t, \mathbf{s}_t), \tag{4}$$

where $\mathcal{L}_{\text{reg}}$ is implemented as Huber loss over per-step coordinate offsets.

4.4 Trajectory scoring and selection

Our final stage estimates the likelihood of full future trajectories $\mathbf{s}_F$. This differs from the second stage, which decomposes over time steps and targets, and from the first stage which only has knowledge of targets, but not full trajectories—*e.g.*, a target might be estimated to have high likelihood, but a full trajectory to reach that target might not.

We use a maximum entropy model to score all the M trajectories from the second stage:

$$\phi(\mathbf{s}_F|\mathbf{x}) = \frac{\exp(g(\mathbf{s}_F, \mathbf{x}))}{\sum_{m=1}^{M} \exp(g(\mathbf{s}_F^m, \mathbf{x}))}, \tag{5}$$

where $g(\cdot)$ is modeled as a 2-layer MLP. The loss term for training this stage is the cross entropy between the predicted scores and ground truth scores,

$$\mathcal{L}_{\text{S3}} = \mathcal{L}_{\text{CE}}(\phi(\mathbf{s}_F|\mathbf{x}), \psi(\mathbf{s}_F)), \tag{6}$$

where the ground truth score of each predicted trajectory is defined by its distance to ground truth trajectory $\psi(\mathbf{s}_F) = \exp(-D(\mathbf{s}, \mathbf{s}_{GT})/\alpha) / \sum_{\mathbf{s}'} \exp(-D(\mathbf{s}', \mathbf{s}_{GT})/\alpha)$, where $D(\cdot)$ is in meters and α is the temperature. The distance metric is defined as $D(\mathbf{s}^i, \mathbf{s}^j) = \max(||s_1^i - s_1^j||_2^2, ..., ||s_t^i - s_t^j||_2^2)$.

To obtain the final small set of K predicted trajectories from the scored M trajectories, we implement a trajectory selection algorithm to reject near-duplicate trajectories. We first sort the trajectories according to their score in descending order, and then pick them greedily; if one trajectory is distant enough from all the selected trajectories, we select it as well, otherwise exclude it. The distance metric used here is the same as for the scoring process. This process is inspired by the non-maximum suppression algorithm commonly used for computer vision problems, such as object detection.

4.5 Training and inference details

The above TNT formulation yields fully supervised end-to-end training, with a total loss function

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{S1}} + \lambda_2 \mathcal{L}_{\text{S2}} + \lambda_3 \mathcal{L}_{\text{S3}}, \tag{7}$$

where $\lambda_1, \lambda_2, \lambda_3$ are chosen to balance the training process.

At inference time, TNT works as follows: (1) encode context; (2) sample N target candidates as input to the target predictor, take the top M targets as estimated by $\pi(\tau|\mathbf{x})$; (3) take the MAP trajectory for each of the M targets from motion estimation model $p(\mathbf{s}_F|\tau, \mathbf{x})$; (4) score the M trajectories by $\phi(\mathbf{s}_F|\tau, \mathbf{x})$, and select a final set of K trajectories.

5 Experiments

5.1 Datasets

Argoverse forecasting dataset [9] provides trajectory histories, context agents and lane centerline for future trajectory prediction. There are 333K 5-second long sequences in the dataset. The trajectories are sampled at 10Hz, with (0, 2] seconds for observation and (2, 5] seconds for future prediction.

INTERACTION dataset [10] focuses on vehicle behavior prediction in highly interactive driving scenarios. It provides 4 different categories of interactive driving scenarios: roundabout (10479 vehicles), un-signalized intersection (14867 vehicles), signalized intersection (10933 vehicles), merging and lane changing (3775 vehicles).

In-house Pedestrian-at-Intersection dataset (PAID) is an in-house pedestrian dataset collected around crosswalks and intersections. There are around 77K unique pedestrians for training and 12k unique pedestrians for test. The trajectories are sampled at 10Hz, 1-sec history trajectory is used to predict 3-sec future. Map features include crosswalks, lane boundaries and stop/yield signs.

Stanford Drone dataset (SDD) [11] is a video dataset with top-down recordings of college campus scenes, collected by drones. The RGB video frames provide context similar to road maps in other datasets. We follow practice of other literature [2, 16, 37], focusing on pedestrian trajectories only: frames are sampled at 2.5 Hz, 2 seconds of history (5 frames) are used as model input, and 4.8 seconds (12 frames) are the future to be predicted.

5.2 Implementation Details

Context encoding. Following VectorNet [35], we convert the map elements and trajectories into a set of polylines and vectors. Each vector is represented as $[p_s, p_e, f, id_p]$, where p_s and p_e are start and end point of the vector, f is a feature vector, which can contain feature type like lane state, and id_p is the polyline index that the vector belongs to. We normalize the vector coordinates to be centered around the location of target agent at the last observed time step. After vectorization, VectorNet is used to encode context of the modeled agent, and its output feature will be consumed by TNT. One exception is the Stanford Drone Dataset, which does not offer map data, we therefore use a standard ResNet-50 [38] ConvNet to encode the birds-eye-view imagery for context encoding.

Target candidate sampling. For vehicle trajectory prediction, we sample points as the target candidates from lane centerlines (Argoverse dataset) or lane boundaries (INTERACTION dataset). At least one point is sampled every meter. For pedestrian trajectory prediction, as pedestrians have much larger moving flexibility, we build a rectangular 2D grid (*e.g.* 10m $\times$ 10m) around the agent, and the center of each cell (*e.g.* 1m $\times$ 1m) is a target candidate.

Model details. The model architectures of all the three stages of TNT are 2-layer MLPs, with the number of hidden units set to 64. We set the temperature α in $\psi(\mathbf{s}_F)$ to be 0.01. The loss weights are $\lambda_1 = 0.1, \lambda_2 = 1.0, \lambda_3 = 0.1$. TNT is trained end-to-end, for approximately 50 epochs with an Adam optimizer [39]. The learning rate is set to be 0.001, and batch size is 128.

Table 1: Performance breakdown after each stage on the Argoverse validation set.

	minFDE		minADE		Miss Rate@2m	
	M=50	K=6	M=50	K=6	M=50	K=6
S1: Target Prediction	0.533	1.629	-	-	0.027	0.216
S1 + S2: Motion Estimation	0.534	1.632	0.488	0.877	0.027	0.216
S1 + S2 + S3: Traj Scoring & Selection	-	1.292	-	0.728	-	0.093

Metrics. We adopt the widely used Average Displacement Error (ADE) and Final Displacement Error (FDE). To evaluate the ADE and FDE for a set of K predicted trajectories, we use minADE_K and minFDE_K. The displacements are all measured in meters, except for the Stanford Drone dataset where it is in pixels. On Argoverse, we also report miss rate (MR) which measures the ratio of scenarios where none of the predictions are within 2 meters of the ground truth according to FDE.

Table 2: Comparison of map target sampling density on Argoverse dataset.

target spacing	minFDE_6	minADE_6
target / 5.0m	1.55	0.79
target / 2.0m	1.31	0.73
target / 1.0m	1.29	0.72
target / 0.5m	1.29	0.72

Table 3: Comparison of grid target sampling density on PAID.

grid size	minFDE_6	minADE_6
2.0m	0.41	0.22
1.0m	0.33	0.19
0.5m	0.32	0.18
0.2m	0.32	0.18

5.3 Ablation study

Performance breakdown by stage. We discuss the efficacy of each stage of TNT by tracing the performance on the Argoverse dataset, shown in Table 1. We can see that S1 achieves good target recall as indicated by minFDE and Miss Rate at $M = 50$; S2 further generates trajectories as evaluated by the minADE metric. The minFDE between S1 and S2 are almost the same, which confirms the fact that the conditional motion estimation is able to generate trajectories ending at the conditioned targets. Finally S3 narrows down the number of predictions to $K = 6$ without much loss compared to $M = 50$.

Target candidate sampling. The target candidate sampling density has an impact on TNT's performance, as shown in Table 2 on Argoverse and Table 3 on PAID respectively. For vehicles in Argoverse, we sample targets from lanes, measured as target spacing along the polyline. For pedestrians in PAID, as they have more freedom of movement, we empirically find that grid targets perform much better than map-based targets, and report only grid target results. We observe that denser targets lead to better performance before the saturating point.

Target regression. The comparison between with and without target offset regression in S1 is shown in Table 4. We can see that with regression the performance improved by 0.16m, which shows the necessity of position refinement from the original target coordinates.

Table 4: Ablation on target offset regression.

	minFDE_{50}@S1
w/ target reg	0.53
w/o target reg	0.69

Table 5: Ablation on motion estimation methods.

	# traj / target	minADE_6@S3
Huber	1	0.73
CVAE	1	0.73
CVAE	10	0.71

Table 6: Comparison with state-of-the-art methods on Argoverse validation and test set. DESIRE and MultiPath are reimplemented with VectorNet context encoder. Our single model result performs on par or better than the Argoverse Challenge 2020 winner on the test set.

	subset	minFDE_6	minADE_6	Miss Rate@2m
DESIRE [1]	validation	1.77	0.92	0.18
MultiPath [33]		1.68	0.80	0.14
TNT (Ours)		**1.29**	**0.73**	**0.09**
Jean (Challenge winner)	test	**1.42**	0.97	**0.13**
TNT (Ours)		1.54	**0.94**	**0.13**

Motion estimation methods. For S2 motion estimation, we compare between our unimodal Huber regressor with a CVAE regressor which generates multimodal predictions. For CVAE, we vary the number of sampled trajectories between 1 and 10. The results are shown in Table 5. As expected, the two perform similar with only 1 trajectory. However, even when we increase the number of CVAE sampled trajectories by $10\times$, it only marginally improves on the minADE metric. This supports our assumption in S2 that the agent motion is unimodal given a target.

Latency analysis. We benchmark the latency of TNT with a NVIDIA v100 GPU: in a light scene of 5 agents, the inference latency is 10.8 ms; in a crowded environment with 50 agents, the latency is 64.4 ms. Our analysis further shows that the VectorNet backbone takes 89% of the processing time, therefore the proposed three-stage trajectory predictor is very efficient.

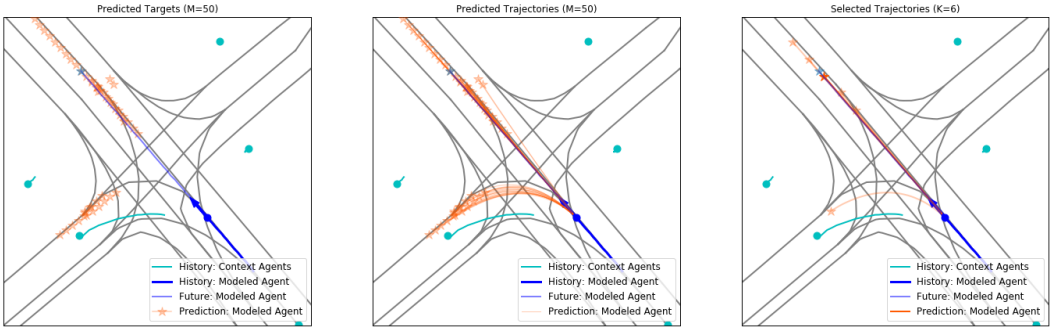

Figure 3: Qualitative results on Argoverse validation set. Lane centerlines are shown in grey, agent's past trajectory in blue, ground truth future trajectory is in light blue. *(Left)* Top predicted targets, where darker color corresponds to higher scores. *(Middle)* Trajectory regression conditioned on targets. *(Right)* Predicted trajectories after scoring and selection. More results in supplemental videos.

Table 7: Model performance on INTERACTION validation set.

	minFDE_6	minADE_6
DESIRE [1]	0.88	0.32
MultiPath [33]	0.99	0.30
TNT (Ours)	**0.67**	**0.21**

Table 8: Comparison with state-of-the-art methods on PAID.

	minFDE_6	minADE_6
DESIRE [1]	0.59	0.29
MultiPath [33]	0.43	0.23
TNT (Ours)	**0.32**	**0.18**

5.4 Comparison with state-of-the-art

Vehicle prediction benchmarks. For trajectory prediction on vehicles, we compare TNT with the state-of-the-art methods on Argoverse and INTERACTION benchmarks. To make fair comparisons, we re-implement MultiPath [33] and DESIRE [1] by replacing their ConvNet context encoders with VectorNet [35]. As shown in Table 6 and Table 7, TNT outperforms all other methods by large margins and achieves the state-of-the-art performance. Visualizations of the TNT predictions on Argoverse can be found in Figure 3. We further submit our single model TNT results to the Argoverse leaderboard. As shown in the bottom rows in Table 6, TNT performs on par or better than the Argoverse Challenge 2020 winner (its details were undisclosed).

Pedestrian prediction benchmarks. For trajectory prediction on pedestrians, we compare TNT with state-of-the-art methods on the in-house Pedestrian-At-Intersection Dataset (PAID) and Stanford Drone Dataset (SDD). On the PAID, we sample targets from a grid of range $20m \times 20m$ with a grid size of $0.5m$. We enhance DESIRE and Multipath with VectorNet for context encoding.

On the SDD, since no map data is provided, we crop an image patch with resolution of 800×800 around the agent of interest, and use a ResNet-50 to extract context features. We use a grid of range 300×300 with a grid size of 6 as targets. As shown in Table 8 and Table 9, TNT outperforms all previous methods and achieves the state-of-the-art performance on both datasets.

Table 9: Comparison with state-of-the-art methods on SDD. Units are pixels.

	minFDE_5	minADE_5
Social LSTM [16]	56.97	31.19
Social GAN [2]	41.44	27.25
DESIRE [1]	34.05	19.25
SoPhie [37]	29.38	16.27
PECNet [31]	25.98	12.79
TNT (Ours)	**21.16**	**12.23**

6 Conclusion

We have presented a novel framework TNT for multimodal trajectory prediction. It consists of three interpretable stages: target prediction, target-conditioned motion estimation, and trajectory scoring. TNT achieves the state-of-the-art performance on four challenging real-world prediction datasets. As future work we plan to extend our framework to long term future prediction by iteratively predicting intermediate targets and trajectories.

Acknowledgments

We would like to thank Anca Dragan for helpful comments on the manuscript.

References

[1] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker. DESIRE: Distant future prediction in dynamic scenes with interacting agents. In *CVPR*, 2017.

[2] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *CVPR*, 2018.

[3] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. PRECOG: Prediction conditioned on goals in visual multi-agent settings. In *ICCV*, 2019.

[4] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *NeurIPS*, 2016.

[5] N. Rhinehart, K. Kitani, and P. Vernaza. R2P2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *ECCV*, 2018.

[6] Y. Yuan and K. Kitani. Diverse trajectory forecasting with determinantal point processes. *ICLR*, 2020.

[7] T. Tsubouchi and S. Arimoto. Behavior of a mobile robot navigated by an" iterated forecast and planning" scheme in the presence of multiple moving obstacles. In *ICRA*, 1994.

[8] A. Broadhurst, S. Baker, and T. Kanade. A prediction and planning framework for road safety analysis, obstacle avoidance and driver information. *CMU-RI-TR-04-11*, 2004.

[9] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *CVPR*, 2019.

[10] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka. INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. *arXiv:1910.03088*, 2019.

[11] A. Robicquet, A. Alahi, A. Sadeghian, B. Anenberg, J. Doherty, E. Wu, and S. Savarese. Forecasting social navigation in crowded complex scenes. *arXiv:1601.00998*, 2016.

[12] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *ITSC*, 2018.

[13] J. Colyar and H. John. Us highway 101 dataset. *FHWA-HRT-07-030*, 2007.

[14] L. Fang, Q. Jiang, J. Shi, and B. Zhou. TPNet: Trajectory proposal network for motion prediction. In *CVPR*, 2020.

[15] K. M. Kitani, D.-A. Huang, and W.-C. Ma. Activity forecasting. In *Group and Crowd Behavior for Computer Vision*. 2017.

[16] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *CVPR*, 2016.

[17] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51 (5):4282, 1995.

[18] W.-C. Ma, D.-A. Huang, N. Lee, and K. M. Kitani. Forecasting interactive dynamics of pedestrians with fictitious play. In *CVPR*, 2017.

[19] S. Zheng, Y. Yue, and J. Hobbs. Generating long-term trajectories using deep hierarchical networks. In *NeurIPS*, 2016.

[20] E. Zhan, S. Zheng, Y. Yue, L. Sha, and P. Lucey. Generative multi-agent behavioral cloning. *arXiv:1803.07612*, 2018.

[21] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *ICRA*, 2019.

[22] J. Hong, B. Sapp, and J. Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *CVPR*, 2019.

[23] R. A. Yeh, A. G. Schwing, J. Huang, and K. Murphy. Diverse generation for multi-agent sports games. In *CVPR*, 2019.

[24] C. Sun, P. Karlsson, J. Wu, J. B. Tenenbaum, and K. Murphy. Stochastic prediction of multi-agent interactions from partial observations. In *ICLR*, 2019.

[25] C. Tang and R. R. Salakhutdinov. Multiple futures prediction. In *NeurIPS*. 2019.

[26] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

[27] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

[28] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa. Planning-based prediction for pedestrians. In *IROS*, 2009.

[29] E. Rehder and H. Kloeden. Goal-directed pedestrian prediction. In *ICCV Workshops*, 2015.

[30] E. Rehder, F. Wirth, M. Lauer, and C. Stiller. Pedestrian prediction by planning using deep neural networks. In *ICRA*, 2018.

[31] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. *arXiv:2004.02025*, 2020.

[32] S. Casas, W. Luo, and R. Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *CoRL*, 2018.

[33] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *CoRL*, 2019.

[34] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff. CoverNet: Multimodal behavior prediction using trajectory sets. *arXiv:1911.10298*, 2019.

[35] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid. VectorNet: Encoding hd maps and agent dynamics from vectorized representation. In *CVPR*, 2020.

[36] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1989.

[37] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *CVPR*, 2019.

[38] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[39] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.