CoverNet: Multimodal Behavior Prediction using Trajectory Sets

Tung Phan-Minh*
Caltech
tung@caltech.edu

Elena Corina Grigore, Freddy A. Boulton, Oscar Beijbom, and Eric M. Wolff
nuTonomy, an Aptiv company
{elena.corina.grigore, freddy.boulton, oscar, eric}@nutonomy.com

Abstract

We present CoverNet, a new method for multimodal, probabilistic trajectory prediction for urban driving. Previous work has employed a variety of methods, including multimodal regression, occupancy maps, and 1-step stochastic policies. We instead frame the trajectory prediction problem as classification over a diverse set of trajectories. The size of this set remains manageable due to the limited number of distinct actions that can be taken over a reasonable prediction horizon. We structure the trajectory set to a) ensure a desired level of coverage of the state space, and b) eliminate physically impossible trajectories. By dynamically generating trajectory sets based on the agent's current state, we can further improve our method's efficiency. We demonstrate our approach on public, real-world self-driving datasets, and show that it outperforms state-of-the-art methods.

1. Introduction

We are motivated by autonomous systems operating in dynamic, interactive, and uncertain environments. Specifically, we focus on the problem of a self-driving car navigating in an urban environment, where it must share the road with a diverse set of other agents, including vehicles, bicyclists, and pedestrians. In this context, reasoning about the possible future states of agents is critical for safe and confident operation. Effective prediction of future agent states depends on both road context (e.g., lane geometry, crosswalks, traffic lights) and the recent behavior of other agents.

Trajectory prediction is inherently challenging due to a wide distribution of agent preferences (e.g., a cautious vs. aggressive) and intents (e.g., turn right vs. go straight). Useful predictions must represent multiple possibilities and their associated likelihoods. Furthermore, we expect that predicted trajectories are physically realizable.

Multimodal regression models appear naturally suited for this task, but may degenerate during training into a single mode. Avoiding this "mode collapse" requires careful considerations [13, 7, 20]. Additionaly, most state-of-the-art methods predict unconstrained positions [13, 7, 20, 31], resulting in trajectories that may not be physically possible for execution ([12] is a recent exception). Our main insights leverage domain-specific knowledge to effectively structure the output representation and address these concerns.

Our first insight is that there are relatively few *distinct* actions that can be taken over a *reasonable* time horizon. Dynamic constraints considerably limit the set of reachable states over a standard six second prediction horizon, and the inherent uncertainty in agent behavior outweighs small approximation errors. We exploit this insight to formulate multimodal, probabilistic trajectory prediction as classification over a trajectory set. This avoids mode collapse and lets the user design the trajectory set to meet specific requirements (e.g., dynamically feasible, coverage guarantees).

Our second insight is that predicted trajectories should be consistent with the current dynamic state. Thus, we formulate our output as motions relative to our initial state (e.g., turn slightly right, accelerate). When integrated with a dynamics model, the output is converted to an appropriate sequence of positions. Beyond helping ensure physically valid trajectories, this *dynamic* output representation ensures that the outputs are diverse in the control space across a wide range of speeds. While [12] exploit a similar insight for regression, we extend the use of a dynamic representation to classification and anchor-box regression.

We now summarize our main contributions on multimodal, probabilistic trajectory prediction with CoverNet:

- introduce the notion of trajectory sets for multimodal trajectory prediction, and show how to generate them in both a fixed and dynamic manner;
- compare state-of-the-art methods on nuScenes [5], a public, real-world urban driving benchmark;
- empirically show the benefits of classification on trajectory sets over multimodal regression.

2. Related Work

We focus on trajectory prediction approaches based on deep learning, and refer the reader to [26] for a survey of more classical approaches. The approaches below typically

*Work done during an internship at nuTonomy, an Aptiv company.

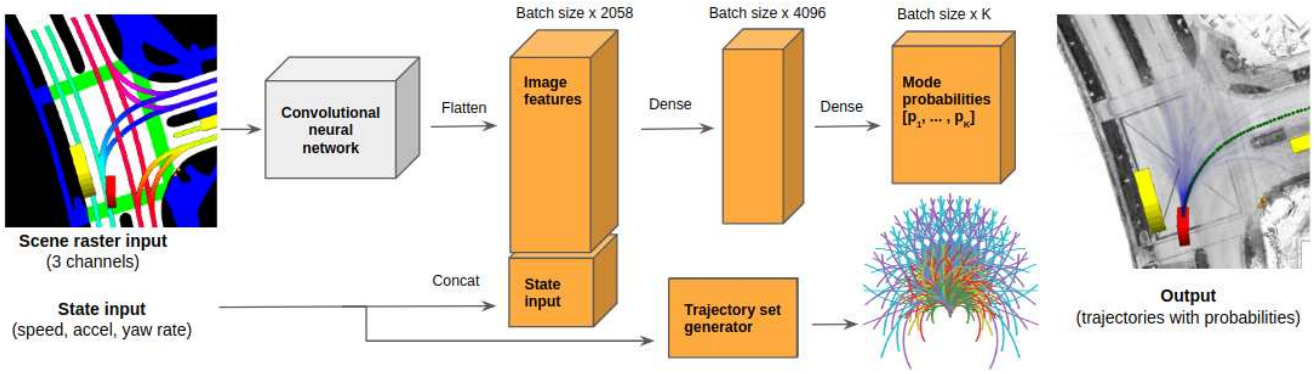


Figure 1: CoverNet overview. We generate a trajectory set (fixed or dynamic based on current state) that we classify over. The input and backbone follow [13].

use CNNs to combine agent history with scene context, and vary significantly in their output representations. Depending on the method, the scene context will include everything from the past states of a single agent, to the past states of all agents along with high-fidelity map information.

Stochastic approaches encode choice over multiple possibilities via sampling random variables. One of the earliest works on motion forecasting frames the problem as learning stochastic 1-step policies [22]. R2P2 [30] improves sample coverage for such policies via a symmetric KL loss. Recent work has considered the multiagent setting [31] and uncertainty in the model itself [19]. Other methods generate samples using CVAEs [20, 25, 2, 21] or GANs [34, 17, 36]. Stochastic approaches can be computationally expensive due to a) repeated 1-step rollouts (in the 1-step policy approach), or b) requiring a large number of samples for acceptable performance (often hard to determine in practice).

Unimodal approaches output a single trajectory per agent [27, 6, 15, 1]. This is often unable to adequately capture possibilities in complex scenarios, even when predicting Gaussian uncertainty. These methods typically average over behaviors, which may result in nonsensical trajectories (e.g., halfway between a right turn and going straight).

Multimodal approaches output either a distribution over multiple trajectories [7, 13, 20, 14] or a spatial-temporal occupancy map [20, 28, 35]. The latter flexibly captures multiple outcomes, but often has large memory requirements for grids at reasonable resolutions. Sampling trajectories from an occupancy map a) is not well defined, and b) adds additional compute during inference. Multimodal regression methods can easily suffer from "mode collapse" to a single mode, leading [7] to use a fixed set of anchor boxes. In contrast, the strength of our contribution lies in framing the problem as classification rather than regression. We also contribute three methods of creating trajectory sets to classify over, and achieve performance improvements over [7].

Most trajectory prediction methods do not explicitly encode motion constraints, predicting trajectories that can be physically infeasible (a recent exception is [12]). By careful choice of our output representation, we exclude all trajectories that would be physically impossible to execute. Although our predictions can result in off-road trajectories at test time, our model learns to assign them a low probability as long as such trajectories are not included during training.

Graph search is a classic approach to motion planning [24], and often used in urban driving applications [4]. A motion planner grows a compact graph (or tree) of possible motions, and computes the best trajectory from this set (e.g., max clearance from obstacles). Since we do not know the other agent's goals or preferences, we cannot directly plan over the trajectory set. Instead, we implicitly estimate these features and directly classify over the set of possible trajectories. There is a fundamental tension between the size of the trajectory set, and the coverage of all potential motions [3]. Since we are only trying to predict the motions of other vehicles well enough to drive, we can easily accept small errors over moderate time horizons (3 to 6 seconds).

Comparing results on trajectory prediction for self-driving cars in urban environments is challenging. Numerous papers are evaluated purely on internal datasets [13, 35, 6, 28], as common public datasets are either relatively small [16], focused on highway driving [9], or are tangentially related to driving [32]. While there are encouraging new developments in public datasets [8, 20], there is no standard. To help provide clear and open results, we evaluate our models on nuScenes [5], a recent public self-driving car dataset focused on urban driving.

3. Method

In this section we outline the main contribution of the paper: a novel method for trajectory set generation, and show how it can be used for behavior prediction.

3.1. Notation

CoverNet computes a multimodal, probabilistic prediction of the future states of a given vehicle using i) the current and past states of all agents (e.g., vehicles, pedestrians, bicyclists), and ii) a high-definition map.

We assume access to the state outputs of an object detection and tracking system of sufficient quality for self-driving. We denote the set of agents that a self-driving car interacts with at time t by $\mathcal{I}_t$ and s_t^i the state of agent $i \in \mathcal{I}_t$ at time t. Let $s_{m:n}^i = \left[s_m^i, \ldots, s_n^i\right]$ where $m < n$ and $i \in \mathcal{I}_t$ denote the discrete-time trajectory of agent i from for times $t = m, \ldots, n$.

Furthermore, we assume access to a high-definition map including lane geometry, crosswalks, drivable area, and other relevant information.

Let $\mathcal{C} = \{\bigcup_i s_{t-m:t}^i; \text{map}\}$ denote the scene context over the past m steps (i.e., map and partial history of all agents).

Figure 1 overviews our model architecture. It largely follows [13], with the key difference in the output representation (see Section 3.2). We use ResNet-50 [18] given its effectiveness in this domain [13, 7].

While our network only computes a prediction for a single agent at a time, our approach can be extended to simultaneously predict for all agents in a similar manner as [7]. We focus on single agent predictions (as in [13]) both to simplify the paper and focus on our main contributions.

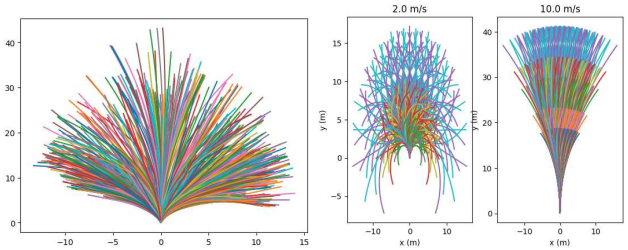
The next sections detail our input and output representations. Our innovations are in our output representations (the *dynamic* encoding of trajectories), and in treating the problem as classification over a diverse set of trajectories.

3.2. Output representation

Due to the relatively short trajectory prediction horizons (up to 6 seconds), and inherent uncertainty in agent behavior, we approximate all possible motions with a set of trajectories that gives sufficient coverage of the space.

Let $\mathcal{R}(s_t)$ be the set of all states that can be reached by an agent with current state s_t in N timesteps (purely based on physical capabilities). We approximate this set by a finite number of trajectories, defining a trajectory set $\mathcal{K} = \{s_{t:t+N}\}$. We define a *dynamic* trajectory set generator as a function $f_N : s_0 \to \mathcal{K}$, which allows the trajectory set to be consistent with the current dynamics. In contrast, a *fixed* generator does not use information about the current state, and thus returns the same trajectories for each instance. We discuss trajectory set construction in Section 3.

We encode multimodal, probabilistic trajectory predictions by classifying over the appropriate trajectory set given an agent of interest and the scene context $\mathcal{C}$. As is common in the classification literature, we use the softmax distribution. Concretely, the probability of the k-th trajectory is given as $p(s_{t:t+N}^k|x) = \frac{\exp f_k(x)}{\sum_i \exp f_i(x)}$, where $f_i(x) \in \mathbb{R}$ is the output of the network's penultimate layer.



(a) fixed (b) dynamic

Figure 2: Overview of trajectory set generation approaches.

In contrast to previous work [13, 7], we choose not to learn an uncertainty distribution over the space. While it is straightforward to add Gaussian uncertainty along each trajectory in a similar manner to [7], the density of our trajectory sets reduces its benefit compared to the case when there are only a handful of modes.

An ideal trajectory set always contains a trajectory that is close to the ground truth. We propose two broad categories of trajectory set generation functions: fixed and dynamic (see Figure 2). In both cases, we normalize the current state to be at the origin, with the heading oriented upwards.

3.3. Fixed trajectory sets

We consider a trajectory set to be *fixed* if the trajectories that it contains do not change as a function of the agent's current dynamic state or environment. Intuitively, this makes it easy to classify over since it allows for a fixed enumeration over the set, but may result in many trajectories that are poor matches for the current situation.

Given a set of representative trajectory data, the problem of finding the smallest fixed approximating trajectory set $\mathcal{K}$ can be cast as an instance of the NP-hard set cover problem. [11]. Approximating a dense trajectory set by a sparse trajectory set that still maintains good coverage and diversity has been studied in the context of robot motion planning [3]. In this work, we use a coverage metric δ defined as the maximum point-wise Euclidean distance between trajectories. Our trajectory set construction procedure starts with subsampling a reasonably large set $\mathcal{K}'$ of trajectories (ours have size 20,000) from the training set. Selecting an acceptable error tolerance ε, we proceed to find the solution to:

$$\begin{aligned} \underset{\mathcal{K}}{\text{argmin}} \quad & |\mathcal{K}| \\ \text{subject to} \quad & \mathcal{K} \subseteq \mathcal{K}', \\ & \forall k \in \mathcal{K}', \ \exists l \in \mathcal{K}, \ \delta(k, l) \leq \varepsilon, \end{aligned} \tag{1}$$

where $\delta(s_{t:t+N}, \hat{s}_{t:t+N}) \coloneqq \max_{\tau=t}^{t+N} \|s_\tau - \hat{s}_\tau\|_2$. We refer to this metric as the maximum point-wise ℓ^2 distance.

We employ a simple greedy approximation algorithm to solve (1), which we refer to as the *bagging* algorithm. We cherry-pick the best among candidate trajectories to place in a bag of trajectories that will be used as the covering set. We repeatedly consider as candidates those trajectories that have not yet been covered and choose the one that covers the most uncovered trajectories (ties are broken arbitrarily).

Standard results (without using the specialized structure of the data) show that our deterministic greedy algorithm is suboptimal by a factor of at most $\log(|\mathcal{K}'|)$ (see Chapter 35.3 [11]). In our experiments, we were able to obtain decent coverage (specifically, under 2 meters in maximum point-wise ℓ^2 distance for 6 second trajectories) with fewer than 2,000 elements in the covering set.

3.4. Dynamic trajectory sets

We consider a trajectory set to be *dynamic* if the trajectories that it contains change as a function of the agent's current dynamic state. This construction guarantees that all trajectories in the set are dynamically feasible.

We now describe a simple approach to constructing such a dynamic trajectory set, focused on predicting vehicle motion. We use a standard vehicle dynamical model [24] as similar models are effective for planning at urban (non-highway) driving speeds [23]. Our approach, however, is not limited to vehicles or any specific model. The dynamical model we use is:

$$\begin{aligned}\dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \frac{v}{b}\tan(u_{steer}) \\ \dot{v} &= u_{accel}\end{aligned}$$

with states: x, y (position), v (speed), θ (yaw); controls: u_{steer} (steering angle), u_{accel} (longitudinal acceleration); and parameter: b (wheelbase).

The dynamics model, controls sequence, and current state determine a trajectory $s_{t:t+N}$ by forward integration. We create a dynamic trajectory set $\mathcal{K}$ based on the current state s_t by integrating forward with our dynamic model over diverse control sequences. Such a dynamic trajectory set has the possibility of being sparser than a fixed set for the same coverage, as each control sequence maps to multiple trajectories (as a function of the current state).

We parameterize the controls (output space) by a diverse set of constant lateral and longitudinal accelerations over the prediction horizon. Using lateral acceleration instead of steering angle is a way of normalizing the output over a range of speeds (a desired lateral acceleration will correspond to different steering angles as a function of speed). We convert the lateral acceleration into a steering angle assuming instantaneous circular motion $a_{lat} = v^2\kappa$ with curvature $\kappa = \tan(u_{steer})/b$. This conversion is ill-defined when the speed is near zero, so we use $\max(v, 1)$ in place of v. Note that it is straightforward to expand the controls (output space) to include multiple lateral and longitudinal accelerations over a non-uniform prediction horizon.

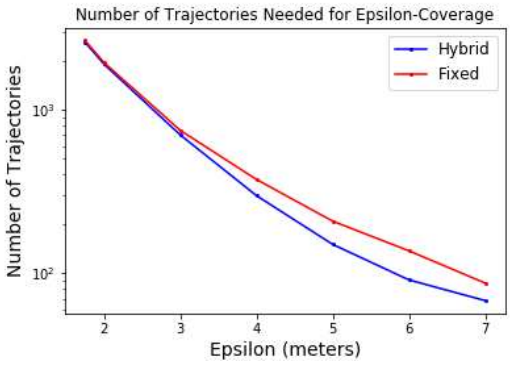


Figure 3: Number of trajectories needed for ε coverage (in meters, see Section 3)

We can further prune the dynamic trajectory set construction in a similar manner to how we handled the fixed trajectory sets in 3.3. The main difference is that the covering set here is constructed from the set of control input profiles as opposed to elements of $\mathcal{K}'$ itself. Namely, we use an analogous greedy procedure to cover the set of sample trajectories with a subset of control profiles (e.g., lateral and longitudinal accelerations as a function of time). Note that unlike the case of fixed trajectories, the synthetic nature of the dynamic profile may not guarantee 100% coverage of $\mathcal{K}'$. To counter this problem we can also create a *hybrid* trajectory set by combining a fixed and dynamic set. Particularly, we find a covering subset for the elements of $\mathcal{K}'$ that cannot be covered by the dynamic choices, and combine this subset with the dynamic choices. When the dynamic set is well-constructed, this can result in a smaller covering set as may be seen from Figure 3.

4. Experiments

We present empirical results on trajectory prediction of vehicles in urban environments. The following sections describe the baselines, metrics, and urban driving datasets that we considered. We used the same input representation and model architecture across our models and baselines.

4.1. Baselines

Physics oracle. We introduce a simple and interpretable model that extends classic physics-based models. We use the track's current velocity, acceleration, and yaw rate to compute the following predictions: i) constant velocity and yaw, ii) constant velocity and yaw rate, iii) constant acceleration and yaw, and iv) constant acceleration and yaw rate.

The *oracle* is the minimum average point-wise Euclidean distance over the four models.

Regression baselines and extensions. We compare our contribution to state-of-the-art methods by implementing two main types of regression models: multimodal regression to coordinates [13] and multimodal regression to residuals from a set of anchors [7] (ordinal regression). We overview these methods for completeness and to provide context for novel variations that we introduce.

Multimodal regression to coordinates. Our implementation follows the details of Multiple-Trajectory Prediction (MTP) [13], adapted for our datasets. This model predicts a fixed number of trajectories (modes) and their associated probabilities. The per-agent loss (agent i at time t) is defined as:

$$\mathcal{L}_{it}^{MTP} = \sum_{k=1}^{|\mathcal{K}|} \mathbb{1}_{k=\hat{k}}[-\log p_{ik} + \alpha L(s_{t:t+N}^{i}, \hat{s}_{t:t+N}^{i})], \tag{2}$$

where $\mathbb{1}(\cdot)$ is the indicator function that equals 1 only for the "best matching" mode, k represents a mode, L is the regression loss, and α is a hyper-parameter used to trade off between classification and regression. With some abuse of notation we use $\mathcal{K}$ to represent the set of trajectories predicted by a model. The original implementation [13] uses a heuristic based on the relative angle between each mode and the ground truth. We select a mode uniformly at random when there are no modes with an angle below the threshold.

Multimodal regression to anchor residuals. Our implementation follows the details of MultiPath (MP) [7]. This model implements ordinal regression by first choosing among a fixed set of anchors (computed a priori) and then regressing to residuals from the chosen anchor. The proposed per-agent loss is (2) where $\alpha = 1$ and the k-th trajectory is the sum of the corresponding anchor and predicted residual. To remain true to the implementation in [7], we choose our best matching anchor by minimizing the average displacement to the ground truth.

We compute the set of fixed anchors by employing the same mechanism described in Section 3.3. Note that this set of trajectories is the same for all agents in our dataset. We then regress to the residuals from the chosen anchor.

4.2. Our models

CoverNet (fixed). Our classification approach where the $\mathcal{K}$ set includes only fixed trajectories.

CoverNet (dynamic). Our classification approach where the $\mathcal{K}$ set is a function of the current agent state.

CoverNet (hybrid). Our classification approach where the $\mathcal{K}$ set is a combination of fixed and dynamic trajectories.

MultiPath with dynamic anchors. The MultiPath approach, extended to use dynamic anchors, described in Section 3.4. The set of anchors is a function of the agent's speed, helping ensure that anchors are dynamically feasible. We then regress to the residuals from the chosen anchor.

4.3. Implementation details

Our implementation setup follows [13] and [7], with key differences highlighted below. See Figure 1 for an overview.

We implemented our models using ResNet-50 [18] as our backbone, with pre-trained ImageNet [33] weights downloaded from [10]. We read the ResNet *conv5* feature map and apply a global pooling layer. We then concatenate the result with an agent state vector (including speed, acceleration, yaw rate), as detailed in [13]. We then add a fully connected layer, with dimension 4096.

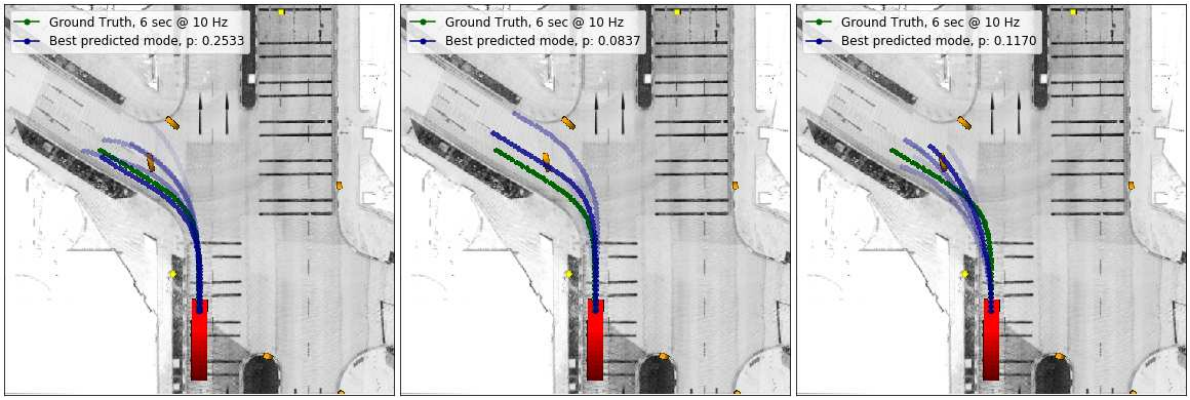
The output dimension of CoverNet is equal to the number of modes, namely $|\mathcal{K}|$. For the hybrid models, the fixed:dynamic trajectory split for the nuScenes dataset is 92:682 and that of the internal dataset is 524:500. We chose these values to maximize coverage at $\epsilon \approx 2\ meters$ and minimize the sum of the total number of categories.

For the regression models, our outputs are of dimension $|\mathcal{K}| \times (|\vec{x}| \times N + 1)$, where $|\mathcal{K}|$ represents the total number of predicted modes, $|\vec{x}|$ represents the number of features we are predicting per point, N represents the number of points in our predictions, and the extra output per mode is the probability associated with each mode. For our implementations, $N = H \times F$, where H represents the length of the prediction horizon in seconds, and F represents the sampling frequency. For each point, we predict (x, y) coordinates, so $|\vec{x}| = 2$.

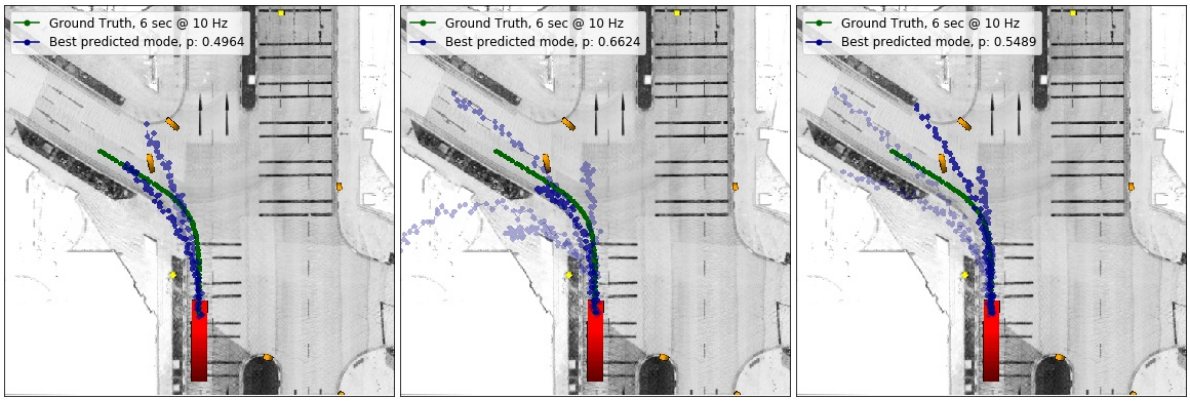
Our internal datasets have $F = 10\ Hz$, while the publicly available nuScenes is sampled at $F = 2\ Hz$. We include results on two different prediction horizon lengths, namely $H = 3$ seconds and $H = 6$ seconds.

The loss functions we use are the same across all of our implementations: for any classification losses, we utilize cross-entropy with positive samples determined by the element in the trajectory set closest to the actual ground truth in minimum average of point-wise Euclidean distances, and for any regression losses, we utilize smooth ℓ^1. For our MTP implementation, we place equal weighting between the classification and regression components of the loss, setting $\alpha = 1$, similar to [13].

For our classification models, we utilize a fixed learning rate of $1\text{e}{-4}$. For our regression models, we use a learning rate of $1\text{e}{-4}$, with a drop by 0.1 as follows: for our internal dataset, we always perform the drop at epoch 6; for nuScenes, we perform the drop at (1) epoch 31 for MTP with 1 and 3 modes and MP dynamic with 16 modes, (2) epoch 12 for MTP with 16 and 64 modes, MP with 16 modes and MP dynamic with 64 modes, and (3) epoch 7 for MP with 64 modes.



(a) CoverNet, fixed, $\varepsilon = 2$, 1937 modes (b) CoverNet, hybrid, 1024 modes (c) CoverNet, dynamic $\varepsilon = 3$, 342 modes



(d) MTP [13], 3 modes (e) MultiPath [13], dynamic, 16 modes (f) MultiPath [13], 64 modes

Figure 4: Examples of predicted trajectories on the same scene. The top row includes our CoverNet models, ranging from fixed to dynamic. The bottom row includes the baselines we compare against, as well as our dynamic templates variation. Objects in the world are rendered up to the current time.

4.4. Metrics

There are multiple ways of evaluating multimodal trajectory prediction. Common measures include log-likelihood [7, 31], average displacement error, and hit rate [20]. We focus on the a) displacement error, and b) hit rate, both computed over a subset of the most likely modes.

For insight into trajectory prediction performance in scenarios where there are multiple plausible actions, we use the minimum average displacement error (ADE). The minADE_k is $\min_{\hat{s} \in \mathcal{P}} \frac{1}{N} \sum_{\tau=t}^{t+N} ||s_\tau - \hat{s}_\tau||$, where $\mathcal{P}$ is the set of k most likely trajectories. We also analyze the final displacement error (FDE), which is $||s_{t+N} - \hat{s}^*_{t+N}||$, where s^* is the most likely mode.

In the context of planning for a self-driving vehicle, the above metrics may be hard to interpret. We use the notion of a *hit rate* (see [20]) to simplify interpretation of whether or not a prediction was "close enough." We define a $\text{Hit}_{k,d}$ for a single instance (agent at a given time) as 1 if $\min_{\hat{s} \in \mathcal{P}} \max_{\tau=t}^{t+N} ||s_\tau - \hat{s}_\tau|| \leq d$, and 0 otherwise. When averaged over all instances, we refer to it as the $\text{HitRate}_{k,d}$.

4.5. Input representation

Similar to [13, 7, 15], we rely on results from an object detection module, and we rasterize the scene for each agent as an RGB image. We start with a blank image of size $(H, W, 3)$ and draw the drivable area, crosswalks, and walk ways using a distinct color for each semantic category.

We rotate the image so that the agent's heading faces up, and place the agent on pixel (l, w), measured from the top-left of the image. We assign a different color to vehicles and pedestrians and choose a different color for the agent so that it is distinguishable. In our experiments, we use a resolution of 0.1 meters per pixel and choose $l = 400$ and $w = 250$. Thus, the model can "see" 40 meters ahead, 10 meters behind, and 25 meters on each side of the agent.

We represent the sequence of past observations for each agent as faded bounding boxes of the same color as the

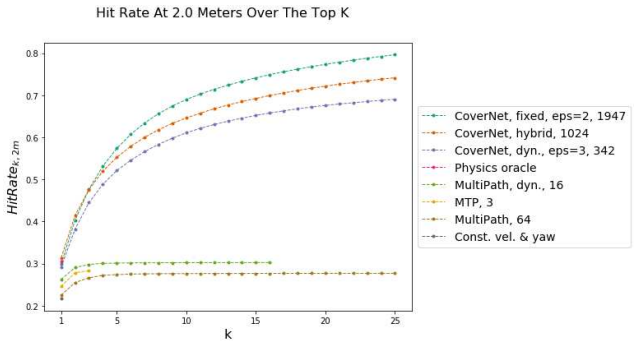


Figure 5: Best models of each type on internal dataset (6 second horizon). CoverNet models significantly outperform others. Legend lists the model name, whether the model is dynamic or fixed (if applicable), and the number of modes.

agent's current bounding box. We fade colors by linearly decreasing saturation (in HSV space) as a function of time.

Although, we have only used one input representation in these experiments, our novel output representation can work with the input representations of [28, 35].

4.6. Datasets

Internal self-driving dataset. We collected 60 hours of real-world, urban driving data in Singapore. Raw sensor data is collected by a car outfitted with cameras, lidars, and radars. A highly-optimized object detection and tracking system filters the raw sensor data to produce tracks at a 10 Hz rate. Each track includes information regarding its type (e.g., car, pedestrian, bicycle, unknown), pose, physical extent, and speed, with quality sufficient for fully-autonomous driving. We also have access to high-definition maps with semantic labels of the road such as the drivable area, lane geometry, and crosswalks.

Each ego vehicle location at a given timestamp is considered a data point. We do not predict on any tracks that are stationary over the entire prediction horizon. Our internal dataset contains around 11 million usable data points but for this analysis we created train, validation, and test sets with 1 million, 300,000, and 300,000 data points, respectively.

nuScenes. We also report results on nuScenes [5], a public self-driving car dataset. nuScenes consists of 1000 scenes, each 20 seconds in length. Scenes are taken from urban driving in Boston, USA and Singapore. Each scene includes hand-annotated tracks and high-definition maps. Tracks have 3D ground truth annotations, and are published at 2 Hz. Since annotations are not public on the test set, we created a set for validation from the train set (called the train-val set) and treated the validation set as the test set. As with our internal dataset, we removed vehicles that are stationary and also removed vehicles that go off the annotated map. This leaves us with 32,186 observations in the train set, 8,560 observations in the train-val set, and 9,041 observations in the validation set. This split publicly available in the nuScenes software development kit [29].

Method	minADE_1	minADE_5	minADE_{10}	minADE_{15}
max ℓ^2	1.0	0.67	0.64	0.64
average ℓ^2	0.96	0.66	0.64	0.64
RMS of ℓ^2	0.96	0.66	0.64	0.63

Table 1: Ground truth matching for fixed trajectory set (150 modes) on internal dataset (3 sec horizon).

5. Results

The main results are summarized in Table 2. Qualitative results are shown in Figure 4.

Quantitative results. Across the six metrics and the two datasets we used, CoverNet outperforms previous methods and baselines in 8 out of 12 cases. However, there are big differences in method ranking depending on the metric.

CoverNet represents a significant improvement on the $\text{HitRate}_{5,\,2\text{m}}$ metric, achieving 33% on nuScenes with the hybrid trajectory set. The next best model is MultiPath, where our dynamic grid extension is a slight improvement over the fixed grid used by the authors (13% vs. 10%). MTP with three modes performs worse, achieving 10%, barely outperforming the constant velocity baseline.

We notice a similar pattern on the internal dataset, where CoverNet outperforms previous methods and baselines. Here, the fixed set with 1,937 modes performs best (57%), closely followed by the hybrid set (55%). Among previous methods, again MultiPath with dynamic set works the best at 30% $\text{HitRate}_{5,\,2\text{m}}$. Figure 5 shows that CoverNet significantly outperforms previous methods as the hit rate is expanded over more modes.

CoverNet also performs well according the Average Displace Error minADE_k metrics, in particular for $k \in \{5, 10, 15\}$, where we see CoverNet outperforming state-of-the-art methods in every category. Most notably, under the minADE_{15} metric for our internal dataset, the hybrid CoverNet with fixed set and 2,206 modes performs best with minADE_{15} of 0.84, 4x better than the constant velocity baseline and 2x better than the MTP and MultiPath. For the minADE_1 metric the regression methods performed the best. This is not surprising since for low k it is more important to have one trajectory very close to the ground truth, a metric paradigm that favors regression over classification.

A notable difference between nuScenes and internal is that the $\text{HitRate}_{5,\,2\text{m}}$ and minADE_k continues to improve for larger sets, while it plateaus, or even decreases at around

Method	Modes	$\text{minADE}_1 \downarrow$	$\text{minADE}_5 \downarrow$	$\text{minADE}_{10} \downarrow$	$\text{minADE}_{15} \downarrow$	FDE $\downarrow$	$\text{HitRate}_{5,\,2\text{m}} \uparrow$
Const. vel. & yaw	N/A	4.61 (3.63)	4.61 (3.63)	4.61 (3.63)	4.61 (3.63)	11.21 (9.86)	0.09 (0.22)
Physics oracle	N/A	**3.70** (**1.88**)	3.70 (1.88)	3.70(1.88)	3.70 (1.88)	**9.09** (5.72)	0.12 (0.31)
MTP [13]	1 (1)	4.17 (**1.88**)	4.17 (1.88)	4.17 (1.88)	4.17 (1.88)	9.37 (**5.22**)	0.05 (0.24)
MTP [13]	3 (3)	4.13 (2.01)	2.93 (1.73)	2.93 (1.73)	2.93 (1.73)	9.23 (5.45)	0.10 (0.28)
MTP [13]	16 (16)	4.55 (3.15)	3.32 (2.48)	3.25 (2.43)	3.23 (2.42)	9.58 (7.79)	0.08 (0.25)
MTP [13]	64 (64)	4.50 (3.21)	3.24 (2.63)	3.15 (2.51)	3.13 (2.47)	9.59 (7.74)	0.09 (0.27)
MultiPath [7]	16 (16)	4.89 (2.34)	2.64 (1.71)	2.47 (1.71)	2.43 (1.70)	10.41 (5.83)	0.08 (0.24)
MultiPath [7]	64 (64)	5.05 (2.30)	2.32 (1.42)	1.96 (1.36)	1.86 (1.34)	10.69 (5.63)	0.10 (0.27)
MultiPath [7], dyn.	16 (16)	3.89 (2.06)	3.34 (1.47)	3.28 (1.46)	3.27 (1.46)	9.19 (5.76)	0.10 (0.30)
MultiPath [7], dyn.	64 (64)	4.05 (2.23)	3.45 (1.53)	3.33 (1.46)	3.28 (1.44)	9.47 (6.17)	0.13 (0.28)
CoverNet, fixed, ε=8	64 (64)	5.16 (2.77)	2.41 (1.98)	2.18 (1.93)	2.13 (1.93)	10.84 (6.65)	0.08 (0.06)
CoverNet, fixed, ε=5	232 (208)	4.73 (2.32)	2.14 (1.35)	1.72 (1.25)	1.60 (1.22)	10.16 (5.67)	0.15 (0.31)
CoverNet, fixed, ε=4	415 (374)	5.07 (2.27)	2.31 (1.29)	1.76 (1.15)	1.57 (1.10)	10.62 (5.85)	0.17 (0.35)
CoverNet, fixed, ε=3	844 (747)	4.74 (2.28)	2.32 (1.32)	1.74 (1.13)	1.51 (1.07)	10.19 (5.92)	0.23 (0.33)
CoverNet, fixed, ε=2	2206 (1937)	5.41 (2.16)	2.62 (**1.16**)	1.92 (**0.93**)	1.63 (**0.84**)	11.36 (5.53)	0.24 (**0.57**)
CoverNet, dyn., ε=3	357 (342)	3.90 (2.06)	2.02 (1.17)	1.57 (0.97)	1.36 (0.88)	9.65 (5.90)	0.33 (0.52)
CoverNet, hybrid	774 (1024)	3.87 (2.18)	**1.96** (1.24)	**1.48** (0.99)	**1.28** (0.88)	9.26 (5.84)	**0.33** (0.55)

Table 2: nuScenes and internal datasets (6 sec horizon). Results listed as nuScenes (internal). Smaller minADE_k and FDE is better. Larger $\text{HitRate}_{5,\,2\text{m}}$ is better. Dyn. = dynamic, vel. = velocity, const. = constant, ε is given in meters.

500-1,000 modes on nuScenes. We hypothesize that this is due to relatively limited size of nuScenes.

Qualititive results. In Figure 4, we show the visualization of a scene overlaid with predictions from our top models compared against our baselines. We note that our prediction horizon for this scene is six seconds. As such, the predictions do not reflect collisions as the pedestrians in the scene will have crossed the road before our vehicle reaches the pedestrian pose reflected in the images.

We emphasize that the CoverNet predictions do not include straight trajectories because the vehicle slows down before the curve. When visualized as a video, we first predict straight trajectories, followed by predicting left turn trajectories when the vehicle starts slowing down. We highlight the smoothness of the trajectories predicted by our model contrasted against the regression baselines. Figure 4 also suggests that the different alternatives for left turns are better captured by CoverNet than by the baseline models.

6. Ablation studies

6.1. Distance function

We analyzed different methods for matching the ground truth to the most suitable trajectory in the trajectory set. Table 1 compares performance using the max, average, and root-mean-square of the point-wise error vector of Euclidean distances for matching ground truth to the "best" trajectory in a fixed trajectory set of size 150. Performance is relatively consistent across all three choices, so we picked the average point-wise ℓ^2 norm to better align with related regression approaches [7].

6.2. Dynamic vs fixed trajectory set coverage

In Figure 3, we compare the number of trajectories needed to achieve 100% coverage of the trajectory set for different levels of ε for the fixed and hybrid trajectory set generation functions, where the latter use a mix of fixed and dynamic trajectories. This figure highlights the advantage of adding dynamic trajectories: they are able to achieve the same level of coverage as the fixed trajectories, but need a smaller number of trajectories to do so.

7. Conclusion

We introduced CoverNet, a novel method for multimodal, probabilistic trajectory prediction in real-world, urban driving scenarios. By framing this problem as classification over a diverse set of trajectories, we were able to a) ensure a desired level of coverage of the state space, b) eliminate dynamically infeasible trajectories, and c) avoid the issue of mode collapse. We showed that the size of our trajectory sets remain manageable over realistic prediction horizons. Dynamically generating trajectory sets based on the agent's current state further improved performance. We compared our results to multiple state-of-the-art methods on real-world self-driving datasets (public and internal), and showed that it outperforms similar methods.

Acknowledgments. We would like to thank Emilio Frazzoli and Sourabh Vora for insightful discussions, and Robert Beaudoin for help on the implementation.

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[2] Apratim Bhattacharyya, Bernt Schiele, and Mario Fritz. Accurate and diverse sampling of sequences based on a "best of many" sample objective. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[3] M. S. Branicky, R. A. Knepper, and J. J. Kuffner. Path and trajectory diversity: Theory and algorithms. In *The IEEE International Conference on Robotics and Automation (ICRA)*, May 2008. 2, 3

[4] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer Publishing Company, 1st edition, 2009. 2

[5] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 1, 2, 7

[6] Sergio Casas, Wenjie Luo, and Raquel Urtasun. IntentNet: Learning to predict intention from raw sensor data. In *Proceedings of The 2nd Conference on Robot Learning*, October 2018. 2

[7] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. MultiPath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *3rd Conference on Robot Learning (CoRL)*, November 2019. 1, 2, 3, 5, 6, 8

[8] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[9] J. Colyar and J. Halkias. US highway 101 dataset, 2007. 2

[10] Torch Contributors. Torchvision.models. https://pytorch.org/docs/stable/torchvision/models.html, 2019. 5

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. 3, 4

[12] Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Jeff Schneider, David Bradley, and Nemanja Djuric. Deep kinematic models for physically realistic prediction of vehicle trajectories, 2019. https://arxiv.org/abs/1908.00219v1. 1, 2

[13] H. Cui, V. Radosavljevic, F. Chou, T. Lin, T. Nguyen, T. Huang, J. Schneider, and N. Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *International Conference on Robotics and Automation (ICRA)*, May 2019. 1, 2, 3, 5, 6, 8

[14] Nachiket Deo and Mohan Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018. 2

[15] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, and Jeff Schneider. Short-term motion prediction of traffic actors for autonomous driving using deep convolutional networks, 2018. https://arxiv.org/abs/1808.05819v2. 2, 6

[16] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2

[17] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 3, 5

[19] Mikael Henaff, Alfredo Canziani, and Yann LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. In *7th International Conference on Learning Representations (ICLR)*, April 2019. 2

[20] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the Road: Predicting driving behavior with a convolutional model of semantic interactions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2, 6

[21] Boris Ivanovic, Edward Schmerling, Karen Leung, and Marco Pavone. Generative modeling of multimodal multi-human behavior. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, October 2018. 2

[22] Kris M. Kitani, Brian D. Ziebart, J. Andrew Bagnell, and Martial Hebert. Activity forecasting. In *The European Conference on Computer Vision (ECCV)*, 2012. 2

[23] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *The IEEE Intelligent Vehicles Symposium (IV)*, June 2015. 4

[24] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, New York, NY, USA, 2006. 2, 4

[25] Namhoon Lee, Wongun Choi, Paul Vernaza, Chris Choy, Philip Torr, and Manmohan Chandraker. DESIRE: Distant future prediction in dynamic scenes with interacting agents. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[26] Stphanie Lefvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH Journal*, 2014. 1

[27] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and Furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[28] Abhijit Ogale Mayank Bansal, Alex Krizhevsky. ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst. In *Robotics: Science and Systems (RSS)*, June 2019. 2, 7

[29] nuScenes Contributors. nuScenes. https://www.nuscenes.org/, 2020. 7

[30] Nicholas Rhinehart, Kris M. Kitani, and Paul Vernaza. R2P2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2

[31] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. PRECOG: Prediction conditioned on goals in visual multi-agent settings. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1, 2, 6

[32] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory prediction in crowded scenes. In *The European Conference on Computer Vision (ECCV)*, 2016. 2

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *The International Journal of Computer Vision*, December 2015. 5

[34] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. SoPhie: An attentive gan for predicting paths compliant to social and physical constraints. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[35] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 7

[36] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2