Rules of the Road: Predicting Driving Behavior with a Convolutional Model of Semantic Interactions

Joey Hong*
Caltech
jhhong@caltech.edu

Benjamin Sapp*
benjamin.sapp@gmail.com

James Philbin
Zoox
james@zoox.com

Abstract

We focus on the problem of predicting future states of entities in complex, real-world driving scenarios. Previous research has used low-level signals to predict short time horizons, and has not addressed how to leverage key assets relied upon heavily by industry self-driving systems: (1) large 3D perception efforts which provide highly accurate 3D states of agents with rich attributes, and (2) detailed and accurate semantic maps of the environment (lanes, traffic lights, crosswalks, etc). We present a unified representation which encodes such high-level semantic information in a spatial grid, allowing the use of deep convolutional models to fuse complex scene context. This enables learning entity-entity and entity-environment interactions with simple, feed-forward computations in each timestep within an overall temporal model of an agent's behavior. We propose different ways of modelling the future as a distribution *over future states using standard supervised learning. We introduce a novel dataset providing industry-grade rich perception and semantic inputs, and empirically show we can effectively learn fundamentals of driving behavior.*

1. Introduction

A crucial component of real-world robotic systems is predicting future states of other actors in the environment. In the general setting, other actors' intents are unobserved, thus the challenge is to produce a likely distribution over possible futures, given current and past observations. A motivating application for this work is a self-driving robot operating in an unconstrained urban environment; one of the most impactful yet challenging real-world robotics applications today. It requires a deep understanding of the semantics of the static and dynamic environment, including understanding traffic laws, unspecified driving conventions, and interactions between human and robot actors.

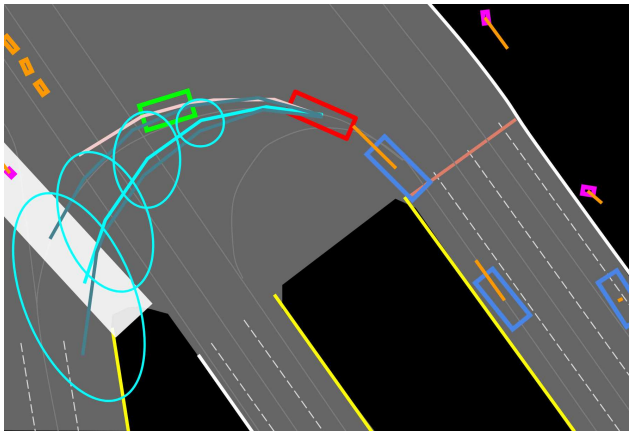


Figure 1: Entity future state prediction task on a top-down scene: A target entity of interest is shown in red, with a real future trajectory shown in pink. The most likely predicted trajectory is shown in cyan, with alternate trajectories shown in green. Uncertainty ellipses showing 1 standard deviation of uncertainty are displayed for the most likely trajectory only. Other entities are rendered in magenta (pedestrians), blue (vehicles) and orange (bicycles). The ego vehicle which captured the scene is shown in green. Velocities are shown as orange lines scaled proportional to $1m/s$. Examples of underlying semantic map information shown are lane lines, crosswalks and stop lines.

While there is a large amount of research dedicated to real-world perception in this domain [15, 25, 26, 12, 35, 29, 13, 9], there is a surprising lack of work on entity state prediction in the same domain (see Section 2), which we attribute to two main causes:

One, most previous research [25, 22] takes as input only raw sensor information (a combination of camera, lidar or

*Work done while at Zoox. The authors would also like to thank and acknowledge Kai Wang (kai@zoox.com) for his work on this project. Kai has been instrumental in coordinating the final version of the paper and preparing the dataset for release.

radar). Thus the research effort by necessity requires a heavy emphasis on extracting high-level representations of entities. In contrast, in real-world systems in industry, low-level perception systems provide entity states and attributes from sensor data via detection and tracking in 2d and 3d. These systems are have matured over the past few years and have high fidelity output in the common case.

Two, publicly available datasets for learning and evaluating state prediction models are inadequately small and/or unrealistic. A good prediction dataset should include a diverse set of real-world locations and a large number of unique agent 3d tracks over meaningful time intervals. These criteria are necessary to develop models which can generalize to new scenarios and leverage past behavior to make meaningful future predictions out to 5 seconds or more. A final omission in previous prediction research is a key asset relied upon heavily by industry self-driving robots: semantic maps of the driving environment, as shown in Fig. 1.

In this paper, we introduce a vehicle prediction dataset which is significantly richer and larger than existing datasets—9,659 unique vehicles in 83,880 prediction scenarios (173 hours), in 88 physically-distinct locations—and includes semantic map information (see Figure 1).

We propose a model which encodes a history of world state (both static and dynamic) and semantic map information in a unified, top-down spatial grid. This allows us to use a deep convolutional architecture to model entity dynamics, entity interactions, and scene context jointly.

An additional important contribution of this work is directly predicting *distributions* of future states, rather than a single point estimate at each future timestep. Representing multimodal uncertainty is crucial in real-world planning for driving, which must consider vehicles taking different possible trajectories, or assessing the expected risk of collision within some spatial extent.

We explore a variety of parametric and non-parametric output distribution representations, and show strong performance on predicting vehicle behavior up to 5 seconds in the future. We demonstrate that our model leverages road information and other agents' state to improve predicted behavior performance.

2. Related Work

This paper focuses on predicting future distributions of entity state. This requires implicitly or explicitly modeling entity intent, dynamics, and interactions, as well as incorporating semantic environmental context.

Activity/motion forecasting Early work by Kitani *et al*. [19] formulated this setting as a Partially-Observable Markov Decision Process, and cast the solution as recovering a policy via Inverse Optimal Control (IOC). More recently, Rhinehart *et al*. [31] also learn one-step control policy distributions within a reinforcement learning framework, specifically trained (via symmetric KL-divergence loss) to generate a set of trajectories with a balance between the notions of diversity and precision. DESIRE [22] also generates trajectories out to $4s$ by iteratively rolling out a one-step policy, via sampling in a Conditional Variational Auto Encoder / Decoder module trained end-to-end in conjunction with a trajectory-ranking module. Other work also concentrates on multimodal distribution modeling [16, 34], but do so outside the self-driving domain.

Fast and Furious [25] and follow-on work IntentNet [10] take a standard supervised regression approach; so-called *behavior cloning* in the RL literature. These works emphasize joint detection (from lidar), tracking and motion forecasting in one model. The recently published IntentNet uses a semantic road map information as an input. It proposes single trajectories per entity out to $3s$.

Relatedly, ChaufferNet [4] fuses high-level agent information (as we do) with a road map. This work focuses on the robot planning setting where the intent is known and fed as input. The model employs behavior cloning to regress the best motion plan, and employs a number of domain-specific losses to encourage road rule-following and collision avoidance. Other notable behavior-cloning approaches in the autonomous vehicle industry are [28, 8, 11].

Another line of work attempts forecasting from egocentric viewpoints (moving camera frame), either of the ego-entity [30] or of other entities [6]. This introduces the additional challenge of needing to recover the ego-position and/or velocity, a problem these works address. There is a wealth of other forecasting work which we don't review here, limiting this discussion to the state prediction in multi-agent environments, although there are connections to unsupervised video prediction (*e.g*. [24]) and activity prediction (*e.g*. [20]).

Modeling entity interactions Much of the above research models interactions implicitly by encoding them as surrounding dynamic context. Other works explicitly model interactions: SocialLSTM [1] pools hidden temporal state between entity models, [2] is one example modeling explicit graph structure over entities to infer semantic actions, and [21] is an earlier work posing the structure as a Bayesian network.

Vehicle Forecasting Datasets We evaluate on vehicles in this work, and propose a new dataset of roughly 80K examples / 170 hours of data. Related datasets are Kitti [15], primarily a detection and tracking benchmark, which has about 50 examples / 10 minutes of data; IntentNet's dataset [10] (unpublished) is roughly 5000 examples / 35 hours, and CaliForecasting [31] (as yet unreleased) is 10K examples / 1.5 hours.

Contrast with our method Our method is, to our knowledge, the only end-to-end method that both (1) encodes se-

mantic scene context and entity interactions from a mature perception stack, as well as (2) predicts multimodal future state distributions.

Most similar to our work, the recent IntentNet [10] and ChaufferNet [4] papers propose encoding static and dynamic scene context in a top-down rasterized grid. A main difference in our work is that we explicitly model multimodal distributions for other entities, rather than regress single trajectories. This is an important and challenging task.

DESIRE [22] and R2P2 [31] address multimodality, but both do so via 1-step stochastic policies, in contrast to ours which directly predicts a time sequence of multimodal distributions. Such policy-based methods require both future roll-out and sampling to obtain a set of possible trajectories, which has computational trade-offs to our one-shot feed-forward approach. Also, knowing how many samples are required to be confident in your empirical distribution is a hard problem, and depends on the scenario.

3. Method

Our method consists of (1) a novel input representation of an entity and it's surrounding world context, (2) a neural network mapping such past and present world representation to future behaviors, and (3) one of several possible output representations of future behavior amenable to integration into a robot planning system. Note our model is an *entity-centric* model, meaning it models a single "target entity", but takes all other world context, including other entities, into account[1].

3.1. Input representation: modeling the world as a convolutional grid

A complete model of trajectory prediction requires not only past history of the target entity, but also dynamics of other entities and semantic scene context.

Road network representation We have access to road network data which includes lane and junction extent and connectivity, as well as other relevant features necessary for driving: crosswalks, traffic light-lane permissibility, and stop and yield lines[2]. We map this information to geometric primitives, and render it in a top-down grid representation as an RGB image with unique colors corresponding to each element type.

This top-down grid establishes the common coordinate space with which we register all additional features. Note that through rendering, we lose the true graph structure of the road network, leaving it as a modeling challenge to learn valid road rules like legal traffic direction, and valid paths through a junction. We denote the rendered tensor of static road information R of size $W \times H \times 3$. Traffic light information is added to a tensor of perception information per timestep described below.

Entity representation We assume access to a black-box perception module which maps from low-level sensor information to 3D tracked entities[3]. For each timestep t, we have measured quantities for each tracked entity i including 2D position x_i^t, velocity v_i^t, and acceleration a_i^t. Our perception module also gives us state estimation uncertainty in the form of covariance matrices, and we include this information in our representation via covariance norms $||\Sigma^t_{i\{x,v,a\}}||_F$. All feature dimensions are scaled by an estimate of their 99^{th} percentile magnitude to have comparable dynamic ranges near $[-1, 1]$.

We form a tensor for the target entity i at any timestep t denoted E_i^t, that has a channel for each state dimension above, encoding the scalar at the center of the entity position, which is in spatial correspondence with road graph tensor R. To model entity interactions, we aggregate all other entities in a tensor encoded in a the same way: $E^t_{-i} = \sum_{j \neq i} E^t_j$. These tensors are $W \times H \times 7$.

Additional dynamic context We encode additional scene context as an RGB image D^t of size $W \times H \times 3$. It contains oriented bounding boxes for all entities in the scene colored by class type (one of cyclist, vehicle, pedestrian), to encode extents and orientations of objects. It also contains a rendering of traffic light permissibility in junctions: we render permitted (green light), yield (unprotected), or prohibited (red light) by masking road connections in a junction that exhibit each permissibility.

Temporal modeling All inputs at timestep t and target entity i are concatenated (in the third channel dimension) into a tensor:

$$C_i^t = \left[E_i^t, E^t_{-i}, D^t, R\right]$$

which is $W \times H \times 20$. See Fig. 2 for an illustration. We concatenate all C_i^t over past history along a temporal dimension. We fix the coordinate system for a static R for all timestamps by centering the reference frame at the position of the target entity at the time of prediction.

This top-down representation is simple to augment with additional entity features in the future. For example: vehicle brake lights and turn signals, person pose and gestures, and even audio cues could all be integrated as additional state channel dimensions.

[1] At run time, straightforward extensions can enable inference for all entities in the scene efficiently, with most of the computation re-used for each agent (since each differs only in a translation of the input).

[2] This type of content is freely available as open source data, *e.g.* at www.openstreetmap.org, but we use a proprietary source in this work.

[3] This is a reasonable assumption for large companies where such modules are relatively mature.

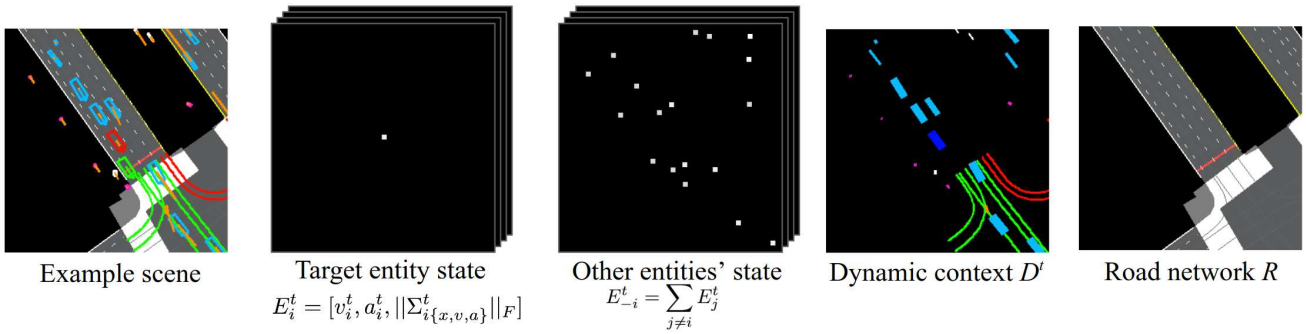
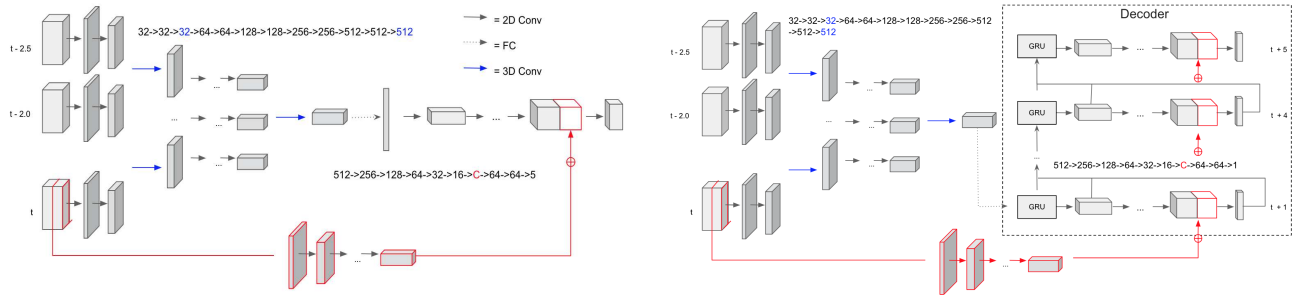


Figure 2: Entity and world context representation. For an example scene (visualized left-most), the world is represented with the tensors shown, as described in the text.



(a) One-shot

(b) with RNN decoder

Figure 3: Two different network architectures for occupancy grid maps (predicting Gaussian trajectories instead is done by simply replacing the convolutional-transpose network with a fully-connected layer).

3.2. Output representation: modeling uncertainty and multiple modes

We seek to model an entity's future states. We believe a good output representation must have the following characteristics, which differs from some previous work [10, 31]. It should be: (1) *A probability distribution* over the entity state space at each timestep. The future is inherently uncertain, and a single most-likely point estimate isn't sufficient for a safety-critical system. (2) *Multimodal*, as it is important to cover a diversity of possible implicit actions an entity might take (*e.g.*, which way through a junction). (3) *One-shot*: For efficiency reasons, it is desirable to predict full trajectories (more specifically: time sequences of state distributions) without iteratively applying a recurrence step.

We explore these desiderata by proposing a variety of output distribution representations. The problem can be naturally formulated as a sequence-to-sequence generation problem. For a time instant t, we observe $X = \{C_{t-\ell+1}, C_{t-\ell+2}, \ldots, C_t\}$ of past scenes, and predict future xy-displacements from time t, denoted $Y = \{(x_{t+1}, y_{t+1}), (x_{t+2}, y_{t+2}), \ldots, (x_{t+m}, y_{t+m})\}$, for a target entity[4], where ℓ and m are the time range limits for past observations and future time horizon we consider. We discuss a variety of approaches to model $P(Y|X)$ next.

3.3. Parametric regression output with uncertainty

In this representation, we predict sufficient statistics for a bivariate Gaussian for each future position (x_t, y_t): the mean $\mu_t = (\mu_{x,t}, \mu_{y,t})$, standard deviation $\sigma_t = (\sigma_{x,t}, \sigma_{y,t})$, and spatial correlation scalar ρ_t. This has benefits of being a richer and more useful output than vanilla regression. In addition, during learning, the model can attenuate the effect of outlier trajectories in the data to have a smaller effect on the loss, an observation made in other problem domains [18].

We train the model to predict log standard deviation $s_t = \log \sigma_t$, as it has a larger numerically-stable range for optimization. We learn parameters via via maximum likeli-

[4]We ignore modeling future entity orientation, but we believe it is straightforward to extend the ideas here to 3-dof or 6-dof state estimation in future work.

hood estimation

$$\log P(Y \mid X) = \sum_{t'=t+1}^{t+m} \log p(x_{t'}, y_{t'} \mid \mu_{t'}, s_{t'}, \rho_{t'}), \quad (1)$$

where p is the density function of a bivariate Gaussian parameterized by $\mathcal{N}(\mu_t, \sigma_t, \rho_t)$.

Multi-modal regression with uncertainty We can extend this to predict up to k different Gaussian trajectories, indexed by $(\mu^i, \hat{s}^i, \rho^i)$, and a weight w^i associated with the probability of a trajectory,

$$P(Y \mid X) = \sum_{i=1}^{k} w^i P(Y \mid \mu^i, \hat{s}^i, \rho^i).$$

However, we run into two major problems with this naive method: (1) exchangeability, and (2) collapse of modes.

For (1), we see that the output is invariant to permutations, namely $\{(\mu^i, s^i, \rho^i)\} = \{(\mu^{\pi(i)}, s^{\pi(i)}, \rho^{\pi(i)})\}$ for some permutation π of $\{1, 2, \ldots, k\}$. To illustrate (2), we consider a mixture of two distributions with equal covariance, or $Y \sim \frac{1}{2}\mathcal{N}(y^1, \sigma^2) + \frac{1}{2}\mathcal{N}(y^2, \sigma^2)$. It can be shown that if y^i are not far enough, specifically $\mid y^1 - y^2 \mid \leq 2\sigma$ [5], then the mixture has a single mode at $\frac{1}{2}(y^1 + y^2)$, and thus can be approximated by a single Gaussian parameterized by $\mathcal{N}(\frac{1}{2}(y^1 + y^2), \sigma^2 + \frac{1}{2}(y^1 - y^2)^2)$. This allows the learning to center the k Gaussians on a single state with large variance ("mode collapse"), with nothing to encourage the modes to be well-separated.

Inspired by Latent Dirichlet Allocation (LDA) [7], we introduce a latent variable z such that (μ^i, s^i, ρ^i) are identically and independently distributed conditioned on z,

$$P(Y \mid X) = \sum_{i=1}^{K} P(z_i \mid X) P(Y \mid \mu^i, s^i, \rho^i, z_i), \quad (2)$$

where (μ^i, s^i, ρ^i) is some fixed function of both the input X and latent variable z, resolving the mode exchangeability problem. To address mode collapse, we choose z to be k-dimensional with small k, and the model is encouraged through the learning procedure to output a diverse set of modes.

To train such a model, we use a conditional variational autoencoder (CVAE) approach, and model $P(z|X)$ as a categorical distribution, using a Gumbell-Softmax distribution with the reparameterization trick [17] to sample and backpropagate gradients through z. In our experiments, we use $P(z|X)$ as our discrete, k-Gaussians mixture distribution over state at future timesteps, and refer to this method in our experiments (loosely) as "GMM-CVAE".

3.4. Non-parametric output distributions

As an alternative to parametric forms, we consider occupancy grid maps [33] as an output representation, where there is an output grid for each future modeled timestep, and each grid location holds the probability of the corresponding output state.

This representation trades off the compactness of the parametric distributions discussed above with arbitrary expressiveness: non-parametric distributions can capture non-elliptical uncertainty and a variable number of modes at each timestep. Furthermore, they offer alternative ways of integrating into robot planning systems (simple to combine with other non-parametric distributions; fast approximate integration for, *e.g.*, collision risk calculation).

We discretize any future state to 2D grid coordinates (i_t, j_t). We train a model to maximize log-likelihood on the predicted grid maps $g_t[i, j] \equiv P(Y = (i_t, j_t)|X)$, which are discrete distributions over the discretized state space at each timestep. Thus the training loss is a sum of cross-entropy losses for $t' \in [t + 1, t + m]$.

Diverse Trajectory Sampling: While occupancy maps have intrinsic representational benefits discussed above, it is still useful in many planning applications to extract a discrete set of trajectories. We discuss an optimization framework to obtain a variable number of trajectories derived from a future occupancy map, with the ability to impose hard and soft constraints on geometric plausibility and diversity / coverage of the trajectory set.

Let $\xi_t = (i_t, j_t)$ be a sampled discretized state at time t, and $\xi = \{\xi_{t+1}, \ldots, \xi_{t+m}\}$ be a sampled trajectory. We define a pairwise-structured score for a sampled trajectory as:

$$s(\xi) = \sum_{t'=t+1}^{t+m} \log P(Y = \xi_{t'}|X) - \lambda \cdot \phi(\xi_{t'}, \xi_{t'-1}). \quad (3)$$

where $\phi(\cdot, \cdot)$ can be an arbitrary score function of the compatibility of consecutive states. We designed a hard-soft constraint cost

$$\phi(\xi_t, \xi_{t-1}) = \begin{cases} \|\xi_t - v(\xi_{t-1})\|_2^2 & \text{if } \|\cdot\|_\infty \leq 5 \\ \infty & \text{otherwise,} \end{cases}$$

where $\lambda = 0.1$ was set by hand in our experiments and $v(\xi_t)$ is the state transitioned to under a constant velocity motion from time t to $t + 1$. This serves as a Gaussian next-state prior centered on a constant velocity motion model, with a cut-off disallowing unreasonable deviations.

This is an instance of a standard chain graphical model [14], and we can solve for the best trajectory $\xi^\star = \arg\max_\xi s(\xi)$ efficiently using a max-sum message passing dynamic program.

Beyond $\xi^\star$, we are interested in extracting a *set* of trajec-

tories which satisfy:

$$\{\xi^{\star_1}, \ldots, \xi^{\star_k}\} = \underset{\{\xi^1,\ldots,\xi^k\}}{\arg\max} \sum_{i=1}^{k} s(\xi^i) \quad (4)$$

$$\text{subject to: } ||\xi^i - \xi^j|| > 1. \quad (5)$$

This seeks to find a set of k trajectories which maximizes $s(\cdot)$ but are sufficiently far from each other, for some norm $||\cdot||$. Following [27], we solve this by iteratively extracting trajectories by solving $s(\xi)$ and then masking regions of g_t to guarantee the distance constraint on the next optimization of $s(\xi)$ for the next trajectory.

Note this framework is employed only during inference. Incorporating it into a learned end-to-end system is an interesting avenue for future work.

3.5. Model

We employ an encoder-decoder architecture for modelling, where the encoder maps the 4D input tensor (time $\times$ space $\times$ channels) into some internal latent representation, and the decoder uses that representation to model the output distribution over states at a pre-determined set of future time offsets.

Encoder: We use a convolutional network (CNN) backbone of 2D convolutions similar to VGG16 [32], on each 3D tensor of the input sequence. Following [3], we found that temporal convolutions acheived better performance and significantly faster training than a recurrent neural network (RNN) structure. To incorporate the temporal dimension, we add in two 3D convolutions – one towards the beginning of the backbone, and one at the end – of kernel size $3\times3\times3$ and $4 \times 3 \times 3$ without padding, respectively.

Decoder: We experiment with two different decoding architectures: (1) "one-shot" prediction of the entire output sequence, and (2) an RNN-decoder which emits a distribution at each inference recurrence step.

One-shot prediction simply requires a two-layer network to regress all the distribution parameters at once, or a 2D convolutional-transpose network with channels equal to the sequence length. For our RNN-decoder, we use only a single GRU cell, whose hidden output is used to regress the true output, which is then fed in as next input.

For the occupancy grid map output representation, the semantic road map R is fed through a separate, shallow CNN tower ($16 \rightarrow 16 \rightarrow 1$ filters), yielding a spatial grid. This grid intuitively acts as a prior heatmap of static information that is appended to the decoder before applying softmax. This should allow the model to easily penalize positions corresponding to obstacles and non-drivable surfaces. See Fig. 3 for a depiction of the architecture.

Dataset	Methods	Mode type	Road info	# tracks $> 3s$	# scenes
ETH+UCY	SocialLSTM* [1]	Peds	no	1,536	4
Stanford Drone	DESIRE [22]	Peds	no	19,564	100
KITTI	DESIRE [22]	Cars	no	309	20
CaliForecasting†	R2P2 [31], C3PO†	Cars	no	10,000	—
Ours	Ours	Cars	yes	72,878	79
private	FaF [25]	Cars	no	—	—
private	IntentNet [10]	Cars	yes	—	—
private	ChauffeurNet†	Cars	yes	—	—

*: Code is publicly available.
†: Method or dataset not yet published.

Table 1: Overview of related public datasets' *training set* statistics.

3.6. Implementation Details

Spatio-temporal dimensions: We chose $\ell = 2.5$s of past history, and predict up to $m = 5$s in the future. To reduce the problem space complexity, we also subsample the past at 2 Hz and predict future control points at 1s intervals. We chose the input frames to be 128×128 pixels, and chose the corresponding real-world extents to be $50 \times 50m^2$. The output grid size was set to 64×64 pixels, so that each pixel covers $0.78m^2$. For sampling a diverse trajectory set, we choose the norm in Equation 5 to be $||\text{diag}([0, 1/3, 1/3, 1/5, 1/5])x||_\infty$, in pixel coordinates, which forces greater spatial diversity in later time steps between trajectories.

Model: To conserve memory in our CNN backbone, we use separable 2D convolutions for most of the backbone with stride 2 (see Fig. 3). We employ batch-normalization layers to deal with the different dynamic ranges of our input channels (*e.g.*, binary masks, RGB pixel values, m/s, m/s^2). We use CoordConv [23] for all 2D convolutions in the encoder, which aids in mapping spatial outputs to regressed coordinates. We chose a 512-dimensional latent representation as the final layer of the encoder, and for all GRU units.

Training: For training, we started with a learning rate of 10^{-4} and Adam optimizer, and switched to SGD with a learning rate of 10^{-5} after convergence with Adam. For training Gaussian parameters we found that pretraining the model on mean-squared error produced better trajectories. All models were trained on a single NVIDIA Maxwell Titan X GPU with Tensorflow Keras. On the dataset described next, training models took approximately 14 hours[5].

4. Experiments

4.1. Dataset

We collected a substantial dataset to evaluate our methods. It is over an order of magnitude larger than [10] and

[5]Training time was heavily dominated by reading data from disk, which could be significantly optimized with more efficient I/O strategies.

Method	Target	Other	Road	RNN	RMSE	mean L2 / hit-rate $< 1m$ 1 sec	2 sec	5 sec
Gaussian Regression	✓				2.55	0.50 / 0.79	1.04 / 0.59	3.91 / 0.34
	✓	✓			2.12	0.48 / 0.80	0.96 / 0.60	3.36 / 0.35
	✓	✓	✓		1.90	0.47 / 0.84	0.94 / 0.64	3.03 / 0.35
	✓	✓	✓	✓	**1.82**	**0.44 / 0.88**	**0.86** / 0.66	**2.99** / 0.34
Grid map Top-1	✓				3.53	0.87 / 0.63	1.31 / 0.52	5.04 / 0.42
	✓	✓			2.71	0.77 / 0.64	1.17 / 0.52	4.40 / 0.42
	✓	✓	✓		2.05	0.61 / 0.86	1.00 / **0.69**	3.37 / 0.40
	✓	✓	✓	✓	1.99	0.55 / 0.87	0.97 / 0.67	3.23 / **0.42**

Table 2: Ablation study on Gaussian Regression trajectories and Grid Map methods.

Method	RMSE	1 sec	2 sec	5 sec
Linear	3.53	**0.37** / 0.89	1.08 / 0.62	5.87 / 0.26
Industry	2.31	**0.37** / **0.90**	0.92 / **0.67**	4.18 / 0.40
Gauss. Reg.	**1.82**	0.44 / 0.88	**0.86** / 0.66	**2.99** / 0.34
GMM	2.33	0.48 / 0.82	0.95 / 0.65	4.17 / 0.19
Grid map	1.99	0.55 / 0.87	0.97 / **0.67**	3.23 / **0.42**
GMM top-5	1.58	0.43 / 0.89	0.79 / 0.70	2.54 / 0.32
Grid top-5	1.25	0.47 / 0.87	0.82 / 0.69	1.39 / 0.56

Table 3: Results for multimodal prediction methods.

many orders of magnitude larger than KITTI [15]. Our dataset consists of tracked vehicles in view of an ego data collection vehicle driving in a dense urban area of San Francisco. A state-of-the-art perception and localization stack processes multiple sensor modalities to produce top-down projected 2D bounding boxes in world coordinates, as well as v_i^t, a_i^t and $\Sigma_{i\{x,v,a\}}^t$. We also include the high-definition rendering of the road network, with detailed annotations described in Section 3.1.

The dataset has more than 6.25 million frames from more than 173 hours of driving in the period of June–July 2018. We split the data into non-overlapping events of 7.5 seconds, and extract $72,878$ train and $10,473$ test events. Events are collected near intersections to skew the dataset distribution towards non-trivial and non-straight driving behavior. To measure our methods' generalization power to new intersections, the test/train partition ensures no intersections appear in both; there are 79 and 9 unique intersections in train and test, respectively. To reduce the bias of any one geographic location, we capped the number of samples per intersection at $5,000$. Note that intersections can be much more complex than simple 4-way junctions.

For an overview of related datasets' attributes and sizes, please refer to Table 1. Note that other vehicle datasets are either quite small (*e.g.*, KITTI [15]), or are not available for comparison. The Stanford Drone dataset is comparable in size and provides a natively top-down representation, which allows us to extend our model to pedestrian prediction in the future.

4.2. Results

We measure performance of all methods on root mean-squared error (RMSE) over all future timesteps $\Delta t \in \{1, 2, 3, 4, 5\}$ seconds, as well as the following metrics on 1, 2, 5s futures: (1) L2 distance between the mean predictions and ground truth, (2) hit-rate of L2 distance under a $1m$ threshold, and (3) oracle error, defined as the minimum L2 distance (*i.e.* $\min_i \left\|Y - \xi^i\right\|_2$) out of top-$k$ predictions.

Models: We compare the following methods:

· *Linear*: Baseline that extrapolates trajectories using the most recent velocity and acceleration, held constant.

· *Industry*: Closed-source industry method used in real-world autonomous vehicle driving. It consists of a hybrid of physics, hand-designed rules, and targeted machine learning classification models. Specific problem modes are modeled in a custom fashion, requiring multiple person-years of modeling effort.

· *Regression with Gaussian Uncertainty*: Our method to regress a Gaussian distribution per future timestep.

· *Multi-modal Gaussian Regression (GMM-CVAE)*: Our described method to predict a set of Gaussians, sampling from a categorial latent variable.

· *Grid map*: Our method to predict occupancy grid maps. To compare to the other methods, we extract trajectories using the described trajectory sampling procedure.

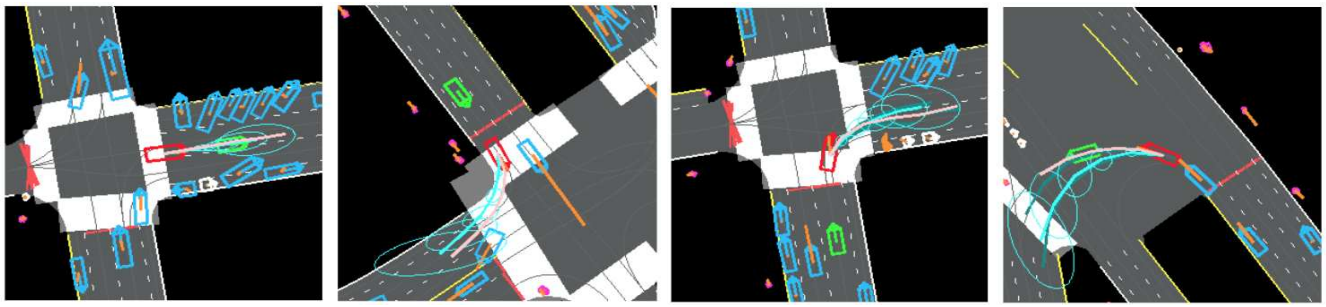
Ablation study: To determine the efficacy of different input channels, we group them and evaluate their performance as follows:

· *Target state*: State of the entity of interest including its rendered past and present bounding boxes.

· *Other state*: Features for other dynamic entities including their bounding boxes rendered.

· *Road map*: Road map and traffic light rendering.

As shown in Table 2, each feature type contributes to improved model performance: Adding the dynamic context of other entities improves 5s prediction by 0.55m in average



(a) Gaussian Regression (b) GMM-CVAE

Figure 4: Examples of Gaussian Regression and GMM-CVAE methods. Ellipses represent a standard deviation of uncertainty, and are only drawn for the top trajectory; only trajectories with probability > 0.05 are shown, with cyan the most probable.We see that uncertainty ellipses are larger when turning than straight, and often follow the direction of velocity. In the GMM-CVAE example, different samples result in turning into different lanes in a junction.

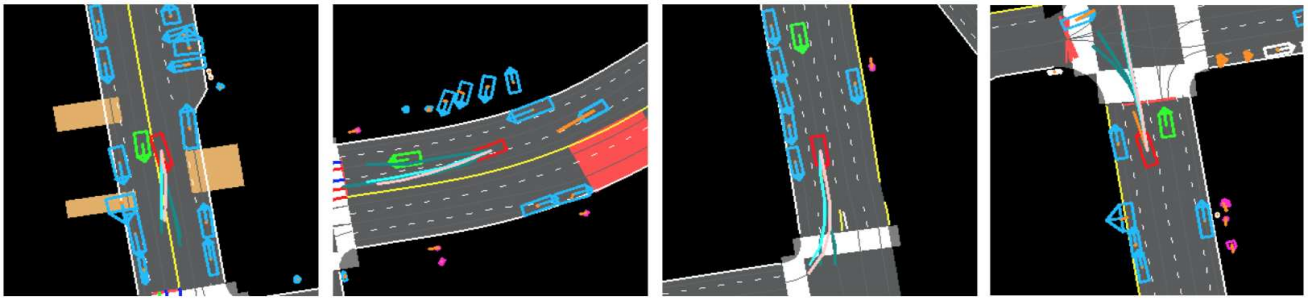


Figure 5: Examples of trajectories sampled from the Grid Map method. The rightmost example is a failure case, as the method predicts a mode that turns into oncoming traffic; however, such traffic rules may be hard to discern from only a road map. The method predicts sophisticated behavior such as maneuvering around vehicles and changing lanes.

L2-error, and including the road map adds another 0.33m improvement. See the supplementary material for qualitative visualizations of how ablated features contribute to performance.

Quantitative method comparison: In Table 3, we compare all methods using all features. Evaluating the best-in-top-5 trajectory performed better than the single MAP trajectory in all metrics, indicating some value of probabilistic prediction over multiple modes.

In general our mixture of sampled Gaussian trajectories underperformed our other proposed methods; we observed that some samples were implausible. We leave it to future work to determine better techniques for obtaining a diverse set of trajectory samples directly from a learned model.

Interestingly, both Linear and Industry baselines performed worse relative to our methods at larger time offsets, but better better at smaller offsets. This can be attributed to the fact that predicting near futures can be accurately achieved with classical physics (which both baselines leverage)—more distant future predictions, however, require more challenging semantic understanding.

Note that while all models are evaluated here in terms of L2-error, none of the models directly optimize this quantity but instead optimize the likelihood of distributions over the future state space, which has other benefits over regression—this is demonstrated in the top-5 metric, as well as in the qualitative results below.

Qualitative Analysis: We show examples of the Gaussian Regression and GMM-CVAE trajectories in Fig. 4, and sampled Grid Map trajectories in Fig. 5. Please see the supplementary material for more examples and visualizations. Overall, trajectories have plausibly learned traffic rules: lane-keeping, traffic light obeyance, following behavior, and even the illegal ones are specious.

5. Conclusion

We present a unified framework for multimodal future state prediction. Our novel input encoding encapsulates static and dynamic scene context, taking advantage of measurements from sensor modalities and a high-definition road map. We experiment with continuous and discrete output representations, and arrive at solutions that address the uncertainty and multi-modality of future prediction. Empirical and qualitative evaluation show that our methods improve on baselines that do not encode scene context, and successfully create diverse samples in complex driving scenarios.

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. SocialLSTM: Human trajectory prediction in crowded spaces. *CVPR*, 2016. 2, 6

[2] T. M. Bagautdinov, A. Alahi, F. Fleuret, P. Fua, and S. Savarese. Social scene understanding: End-to-end multi-person action localization and collective activity recognition. In *CVPR*, 2017. 2

[3] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018. 6

[4] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*. 2, 3

[5] J. Behboodian. On the modes of a mixture of two normal distributions. *Technometrics*, pages 131–139, 1970. 5

[6] A. Bhattacharyya, M. Fritz, and B. Schiele. Long-term on-board prediction of people in traffic scenes under uncertainty. In *CVPR*, 2018. 2

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 2003. 5

[8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 2

[9] S. Bullinger, C. Bodensteiner, M. Arens, and R. Stiefelhagen. 3d vehicle trajectory reconstruction in monocular video data using environment structure constraints. In *ECCV*, 2018. 1

[10] S. Casas, W. Luo, and R. Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *CoRL*, 2018. 2, 3, 4, 6

[11] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *ICCV*, 2015. 2

[12] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017. 1

[13] N. Dinesh Reddy, M. Vo, and S. G. Narasimhan. Carfusion: Combining point tracking and part detection for dynamic 3d reconstruction of vehicles. In *CVPR*, 2018. 1

[14] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005. 5

[15] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. *CVPR*, 2012. 1, 2, 7

[16] B. Ivanovic, E. Schmerling, K. Leung, and M. Pavone. Generative modeling of multimodal multi-human behavior. 2018. 2

[17] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *ICLR*, 2017. 5

[18] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? *NIPS*, 2017. 4

[19] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. *ECCV*, 2012. 2

[20] Y. Kong and Y. Fu. Human action recognition and prediction: A survey. *arXiv preprint arXiv:1806.11230*, 2018. 2

[21] J. F. P. Kooij, N. Schneider, F. Flohr, and D. Gavrila. Context-based pedestrian path prediction. In *ECCV*, 2014. 2

[22] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. K. Chandraker. DESIRE: distant future prediction in dynamic scenes with interacting agents. *CVPR*, 2017. 1, 2, 3, 6

[23] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the CoordConv solution. *arXiv preprint arXiv:1807.03247*, 2018. 6

[24] W. Lotter, G. Kreiman, and D. D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *CoRR*, 2016. 2

[25] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. *CVPR*, 2018. 1, 2, 6

[26] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017. 1

[27] D. Park and D. Ramanan. N-best maximal decoders for part models. *ICCV*, 2011. 6

[28] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NIPS*, 1989. 2

[29] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, 2016. 1

[30] N. Rhinehart and K. M. Kitani. First-person activity forecasting with online inverse reinforcement learning. In *ICCV*, 2017. 2

[31] N. Rhinehart, K. M. Kitani, and P. Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. *ECCV*, 2018. 2, 3, 4, 6

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014. 6

[33] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005. 5

[34] J. Wiest, M. Hoffken, U. Kresel, and K. Dietmayer. Probabilistic trajectory prediction with gaussian mixture models. *Intelligent Vehicles Symposium*, 2012. 2

[35] Y. Zhou and O. Tuzel. VoxelNet: End-to-end learning for point cloud based 3d object detection. *CoRR*, 2017. 1