MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction

Yuning Chai[*]   **Benjamin Sapp**[*]   **Mayank Bansal**   **Dragomir Anguelov**

Waymo LLC
{chaiy,bensapp}@waymo.com

Abstract: Predicting human behavior is a difficult and crucial task required for motion planning. It is challenging in large part due to the highly uncertain and multi-modal set of possible outcomes in real-world domains such as autonomous driving. Beyond single MAP trajectory prediction [1, 2], obtaining an accurate probability distribution of the future is an area of active interest [3, 4]. We present MultiPath, which leverages a fixed set of future state-sequence anchors that correspond to modes of the trajectory distribution. At inference, our model predicts a discrete distribution over the anchors and, for each anchor, regresses offsets from anchor waypoints along with uncertainties, yielding a Gaussian mixture at each time step. Our model is efficient, requiring only one forward inference pass to obtain multi-modal future distributions, and the output is parametric, allowing compact communication and analytical probabilistic queries. We show on several datasets that our model achieves more accurate predictions, and compared to sampling baselines, does so with an order of magnitude fewer trajectories.

1 Introduction

We focus on the problem of predicting future agent states, which is a crucial task for robot planning in real-world environments. We are particularly interested in addressing this problem for self-driving vehicles, an application with a potentially enormous societal impact. Importantly, predicting the future of other agents in this domain is vital for safe, comfortable and efficient operation. For example, it is important to know whether to yield to a vehicle if they are going to cut in front of our robot or when would be the best time to merge into traffic. Such future prediction requires an understanding of the static and dynamic world context: road semantics (*e.g.*, lane connectivity, stop lines), traffic light information, and past observations of other agents, as depicted in Fig. 1.

A fundamental aspect of future state prediction is that it is inherently *stochastic*, as agents cannot know each other's motivations. When driving, we can never really be sure what other drivers will do next, and it is important to consider multiple outcomes and their likelihoods.

We seek a model of the future that can provide both (1) a weighted, parsimonious set of discrete trajectories that covers the space of likely outcomes and (2) a closed-form evaluation of the likelihood of any trajectory. These two attributes enable efficient reasoning in crucial planning use-cases, for example, human-like reactions to discrete trajectory hypotheses (*e.g.*, yielding, following), and probabilistic queries such as the expected risk of collision in a space-time region.

Both of these attributes present modeling challenges. Models which try to achieve diversity and coverage often suffer from mode collapse during training [4, 5, 6], while tractable probabilistic inference is difficult due to the space of possible trajectories growing exponentially over time.

Our MultiPath model addresses these issues with a key insight: it employs a fixed set of *trajectory anchors* as the basis of our modeling. This lets us factor stochastic uncertainty hierarchically: First, *intent uncertainty* captures the uncertainty of *what* an agent intends to do and is encoded as a distribution over the set of anchor trajectories. Second, given an intent, *control uncertainty* represents our uncertainty over *how* they might achieve it. We assume control uncertainty is normally distributed
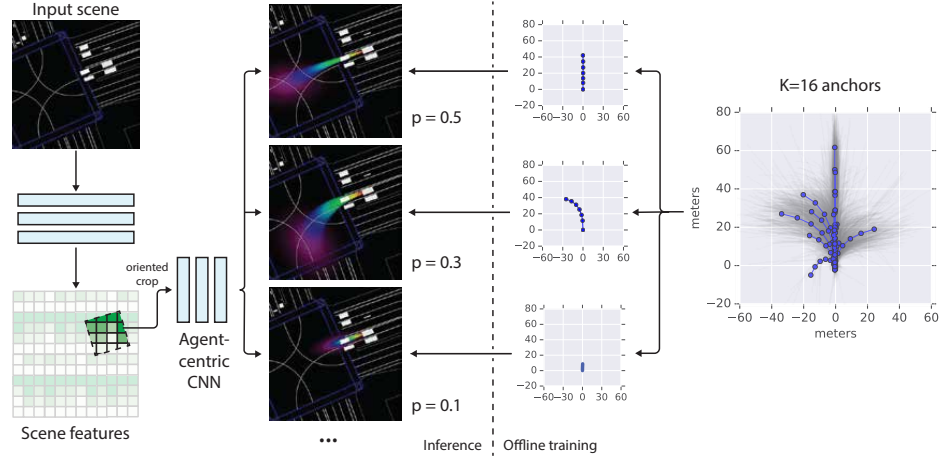
---

[*]Equal contribution

Figure 1: MultiPath estimates the distribution over future trajectories per agent in a scene, as follows: 1) Based on a top-down scene representation, the Scene CNN extracts mid-level features that encode the state of individual agents and their interactions. 2) For each agent in the scene, we crop an agent-centric view of the mid-level feature representation and predict the probabilities over the fixed set of K predefined anchor trajectories. 3) For each anchor, the model regresses offsets from the anchor states and uncertainty distributions for each future time step.

at each future time step [7], parameterized such that the mean corresponds to a context-specific offset from the anchor state, with the associated covariance capturing the unimodal aleatoric uncertainty [8]. Fig. 1 illustrates a typical scenario where there are 3 likely intents given the scene context, with control mean offset refinements respecting the road geometry, and control uncertainty intuitively growing over time.

Our trajectory anchors are modes found in our training data in state-sequence space via unsupervised learning. These anchors provide templates for coarse-granularity futures for an agent and might correspond to semantic concepts like "change lanes", or "slow down" (although to be clear, we don't use any semantic concepts in our modeling).

Our complete model predicts a Gaussian mixture model (GMM) at each time step, with the mixture weights (intent distribution) fixed over time. Given such a parametric distribution model, we can directly evaluate the likelihood of any future trajectory and also have a simple way to obtain a compact, diverse weighted set of trajectory samples: the MAP sample from each anchor-intent.

Our model contrasts with popular past approaches which either provide only a single MAP trajectory [1, 2, 9, 10, 11] or an unweighted set of samples via a generative model [3, 4, 6, 12, 13, 14, 15]. There are a number of downsides to sample-based methods when it comes to real-world applications such as self-driving vehicles: (1) non-determinism in a safety critical system, (2) a poor handle on approximation error (e.g,. "how many samples must I draw to know the chance the pedestrian will jaywalk?"), (3) no easy way to perform probabilistic inference for relevant queries, such as computing expectations over a spacetime region.

We demonstrate empirically that our model emits distributions which predict the observed outcomes better on synthetic and real-world prediction datasets: we achieve higher likelihood than a model which emits unimodal parametric distributions, showing the importance of multiple anchors in real-world data. We also compare to sampling-based methods by using our weighted set of MAP trajectories per anchor, which describe the future better with far fewer samples on sample-set metrics.

2   Related work

We broadly categorize previous approaches to predicting future trajectory distributions into two classes of models: deterministic and stochastic. Deterministic models predict a single most-likely trajectory per agent, usually via supervised regression [1, 2, 9, 10, 11, 16].

Stochastic models incorporate random sampling during training and inference to capture future non-determinism. The seminal motion forecasting work of Kitani *et al.* [14] cast this as a Markov

decision process and learns a 1-step policy, as does follow on work focusing on egocentric video and pedestrians [15, 17]. To encourage sample diversity and coverage, R2P2 [4] proposes a symmetric KL loss between the predicted and data distributions. Several works explore the use of conditional variational autoencoders (CVAEs) and GANs to generate samples [3, 6, 13, 18, 19]. One drawback of such non-deterministic approaches is that they can make reproducing and analyzing results in a larger system difficult.

Like us, a few previous works directly model probability distributions, either parametric [6, 12, 20] or in the form of probabilistic state-space occupancy grids (POGs) [6, 11]. While extremely flexible, POGs require state-space-dense storage to describe the distribution rather than just a few parameters, and it's not obvious how best to extract trajectory samples from POG space-time volumes.

Our method is influenced heavily by the concept of predefined anchors, which have a rich history in machine learning applications to handle multi-modal problems, starting with classic semi-parametric methods such as locally-weighted logistic regression, radial basis SVM and Gaussian Mixture Models [5]. In the computer vision literature, they have been used effectively for detection [21] and human-pose estimation [22]. Like ours, these effective approaches predict the likelihood of anchors and also predict continuous refinements of state conditioned on these anchors (*e.g.* box corners, joint locations or vehicle positions).

3 Method

Given observations $\mathbf{x}$ in the form of past trajectories of all agents in a scene and possibly additional contextual information (*e.g.*, lane semantics, traffic light states) , MultiPath seeks to provide (1) a parametric distribution over future trajectories $\mathbf{s}$: $p(\mathbf{s}|\mathbf{x})$, and (2) a compact weighted set of explicit trajectories which summarizes this distribution well.

Let t denote a discrete time step, and let s_t denote the state of an agent at time t, the future trajectory $\mathbf{s} = [s_1, \ldots, s_T]$ is a sequence of states from $t = 1$ to a fixed time horizon T. We also refer to a state in a trajectory as a *waypoint*.

We factorize the notion of uncertainty into independent quantities. *Intent uncertainty* models uncertainty about the agents' latent coarse-scale intent or desired goal. For example, in a driving context, uncertainty about which lane the agent is attempting to reach. Conditioned on intent, there is still *control uncertainty*, which describes the uncertainty over the sequence of states the agent will follow to satisfy its intent. Both intent and control uncertainty depend on the past observations of static and dynamic world context $\mathbf{x}$.

We model a discrete set of *intents* as a set of K anchor trajectories $\mathcal{A} = \{\mathbf{a}^k\}_{k=1}^K$, where each anchor trajectory is a sequence of states: $\mathbf{a}^k = [a_1^k, \ldots, a_T^k]$, assumed given for now. We model uncertainty over this discrete set of intents with a softmax distribution: $\pi(\mathbf{a}^k|\mathbf{x}) = \frac{\exp f_k(\mathbf{x})}{\sum_i \exp f_i(\mathbf{x}))}$, where $f_k(\mathbf{x}) : \mathbb{R}^{d(\mathbf{x})} \mapsto \mathbb{R}$ is the output of a deep neural network.

We make the simplifying assumption that uncertainty is unimodal given intent, and model *control uncertainty* as a Gaussian distribution dependent on each waypoint state of an anchor trajectory:

$$\phi(s_t^k|\mathbf{a}^k, \mathbf{x}) = \mathcal{N}(s_t^k|a_t^k + \mu_t^k(\mathbf{x}), \Sigma_t^k(\mathbf{x})) \tag{1}$$

The Gaussian parameters μ_t^k and Σ_t^k are directly predicted by our model as a function of $\mathbf{x}$ for each time-step of each anchor trajectory $\mathbf{a}_t^k$. Note in the Gaussian distribution mean, $a_t^k + \mu_t^k$, the μ_t^k represents a scene-specific offset from the anchor state a_t^k; it can be thought of as modeling a scene-specific residual or error term on top of the prior anchor distribution. This allows the model to refine the static anchor trajectories to the current context, with variations coming from, *e.g.* specific road geometry, traffic light state, or interactions with other agents.

The time-step distributions are assumed to be conditionally independent given an anchor, *i.e.*, we write $\phi(s_t|\cdot)$ instead of $\phi(s_t|\cdot, s_{1:t-1})$. This modeling assumption allows us to predict for all time steps jointly with a single inference pass, making our model simple to train and efficient to evaluate. If desired, it is straightforward to add a conditional next-time-step dependency to our model, using a recurrent structure (RNN).

To obtain a distribution over the entire state space, we marginalize over agent intent:

$$p(\mathbf{s}|\mathbf{x}) = \sum_{k=1}^{K} \pi(\mathbf{a}^k|\mathbf{x}) \prod_{t=1}^{T} \phi(s_t|\mathbf{a}^k, \mathbf{x}) \tag{2}$$

Note that this yields a Gaussian Mixture Model distribution, with mixture weights fixed over all time steps. This is a natural choice to model both types of uncertainty: it has rich representational power, a closed-form partition function, and is also compact. It is easy to evaluate this distribution on a discretely sampled grid to obtain a probabilistic occupancy grid, more cheaply and with fewer parameters than a native occupancy grid formulation [6, 11].

Obtaining anchor trajectories. Our distribution is parameterized by anchor trajectories $\mathcal{A}$. As noted by [6, 5], directly learning a mixture suffers from issues of mode collapse. As is common practice in other domains such as object detection [23] and human pose estimation [22], we estimate our anchors a-priori before fixing them to learn the rest of our parameters. In practice, we used the k-means algorithm as a simple approximation to obtain $\mathcal{A}$ with the following squared distance between trajectories: $d(\mathbf{u}, \mathbf{v}) = \sum_{t}^{T} ||M_u \mathbf{u}_t - M_v \mathbf{v}_t||_2^2$, where M_u, M_v are affine transformation matrices which put trajectories into a canonical rotation- and translation-invariant agent-centric coordinate frame. In Sec. 4, on some datasets, k-means leads to highly redundant clusters due to prior distributions that are heavily skewed to a few common modes. To address this, we employ a simpler approach to obtain $\mathcal{A}$ by uniformly sampling trajectory space.

Learning. We train our model via *imitation learning* by fitting our parameters to maximize the log-likelihood of recorded driving trajectories. Let our data be of the form $\{(\mathbf{x}^m, \hat{\mathbf{s}}^m)\}_{m=1}^{M}$. We learn to predict distribution parameters $\pi(\mathbf{a}^k|\mathbf{x})$, $\mu(\mathbf{x})_t^k$ and $\Sigma(\mathbf{x})_t^k$ as outputs of a deep neural network parameterized by weights θ with the following negative log-likelihood loss built upon Equation 2:

$$\ell(\theta) = - \sum_{m=1}^{M} \sum_{k=1}^{K} \mathbb{1}(k = \hat{k}^m) \Big[\log \pi(\mathbf{a}^k|\mathbf{x}^m; \theta) + \sum_{t=1}^{T} \log \mathcal{N}(s_t^k | a_t^k + \mu_t^k, \Sigma_t^k; \mathbf{x}^m; \theta) \Big]. \tag{3}$$

This is a time-sequence extension of standard GMM likelihood fitting [5]. The notation $\mathbb{1}(\cdot)$ is the indicator function, and $\hat{k}^m$ is the index of the anchor most closely matching the groundtruth trajectory $\hat{\mathbf{s}}^m$, measured as ℓ^2-norm distance in state-sequence space. This hard-assignment of groundtruth anchors sidesteps the intractability of direct GMM likelihood fitting, avoids resorting to an expectation-maximization procedure, and gives practitioners control over the design of the anchors as they wish (see our choice below). One could also employ a soft-assignment to anchors (*e.g.*, proportional to the distance of the anchor to the groundtruth trajectory), just as easily.

Inferring a diverse weighted set of test-time trajectories. Our model allows us to eschew standard sampling techniques at test time, and obtain a weighted set of K trajectories without any additional computation: we take the MAP trajectory estimates from each of our K anchor modes, and consider the distribution over anchors $\pi(\mathbf{a}_k|\mathbf{x})$ the sample weights (*i.e.*, importance sampling). When metrics and applications call for a set of top $\kappa < K$ trajectories for evaluation, we return the top κ according to these sample weights.

Input representation. We follow other recent approaches [2, 6, 11] and represent a history of dynamic and static scene context as a 3-dimensional array of data rendered from a top-down orthographic perspective. The first two dimensions represent spatial locations in the top-down image. The channels in the depth dimension hold static and time-varying (dynamic) content of a fixed number of previous time steps. Agent observations are rendered as oriented bounding box binary images, one channel for each time step. Other dynamic context such as traffic light state and static context of the road (lane connectivity and type, stop lines, speed limit, *etc.*) form additional channels. See Sec. 4 for further details, as the input content differs from dataset to dataset. An important benefit of using such a top-down representation is the simplicity of representing contextual information like the agents' spatial relationships to each other and semantic road information. In Sec. B.4, we empirically highlight its benefit towards behavior prediction.

Neural network details. As shown in Fig. 1, we designed a jointly-trained, two-stage architecture that first extracts a feature representation for the whole scene and then attends to each agent in the scene to make agent-specific trajectory predictions.

4

The first stage is fully convolutional to preserve spatial structure; it takes the 3D input representation described above and outputs a 3D feature map of the entire top-down scene. We opt to use ResNet-based architectures [24] for this scene-level feature extractor. We employ depth-wise thinned-out networks for all experiments, and a different number of residual layers depending on the dataset. See Sec. B.2 for a speed-accuracy analysis of different ResNet setups.

The second phase extracts patches of size 11×11 centered on agents locations in this feature map. To be orientation invariant, the extracted features are also rotated to an agent-centric coordinate system via a differentiable bilinear warping. The efficacy of this type of heading-normalization is shown in Sec. B.3. The second agent-centric network then operates on a per-agent basis. It contains 4 convolutional layers with kernel size 3 and 8 or 16 depth channels. It produces $K \times T \times 5$ parameters describing bivariate Gaussian's per time step per anchor (parameterized by $\mu_x, \mu_y, \log \sigma_x, \log \sigma_y$ and ρ; the last 3 parameters define the 2×2 covariance matrix Σ_{xy} in the agent-centric x, y-coordinate space), as well as K softmax logits to represent $\pi(\mathbf{a}|\mathbf{x})$.

4 Experiments

This section presents empirical results on a number of prediction tasks. We consider the following methods in order to contrast to different aspects of MultiPath.

MultiPath μ [, Σ]. Our proposed method with multiple anchors, modeling offsets μ and control uncertainty covariances Σ. For some experiments, we keep Σ frozen, which reduces the maximum-likelihood loss to simple ℓ^2-loss. However, we can no longer estimate likelihood $p(\mathbf{s}|\mathbf{x})$ without Σ and only report distance-based metrics.

Regression μ [, Σ]. To verify our hypothesis that modeling multiple intents is important, we modified the MultiPath architecture to regress a single output trajectory. This is similar to [1]'s output (but extended to include uncertainty).

Min-of-K [20]. This method predicts K trajectories directly, without pre-defined anchors. The authors define an ℓ^2-loss on the single trajectory (out of K) with minimum distance to the groundtruth trajectory. This is similar to our method, but with implicit anchors and evolving hard-assignment of anchors to groundtruth as training progresses. This representation has inherent ambiguity problems and can suffer from mode collapse. In our experiments below, we extend this method to also predict μ, Σ values at each waypoint to evaluate likelihood.

CVAE. The Conditional Variational Auto-Encoder is a standard implicit generative sampling model and has been successfully adapted to predict trajectory for autonomous driving in [3]. We are interested in comparing its ability to generate a diverse set of samples compared to MultiPath's MAP trajectory per anchor—we hypothesize that MultiPath will have better coverage with the same number of trajectories due to its choice of anchors. For this baseline, we add a CVAE at the end of the second stage agent-centric feature extractor. The decoder and encoder have the same architecture: 4 fully-connected layers of 32 units each.

Linear. Following [25], we use a linear model on past states to establish how well a simple constant velocity model can perform. We fit past observed positions as a linear function of time: $\mathbf{x}^t = [\alpha t + \beta, \gamma t + \delta]$ for $t \leq 0$, and use these models to evaluate future positions $\mathbf{x}^1, \ldots, \mathbf{x}^T$. We investigated using higher-order polynomials with worse results.

We implemented the single-trajectory regression, Min-of-K, and CVAE using the same input representation and a comparable model architecture in order to achieve a fair comparison. For benchmark datasets, we also report numbers taken from recent publications.

4.1 Metrics

Different approaches use a variety of output representations; primary examples are single trajectory prediction [1], an unweighted set of trajectory samples [3], a distribution over trajectories (ours), or probabilistic occupancy grids [11]. Each representation comes with its own salient metrics, making it difficult to compare across all methods. Let $\hat{\mathbf{s}} = \hat{s}_{t=1...T}$ be a groundtruth trajectory. We consider the following metrics:

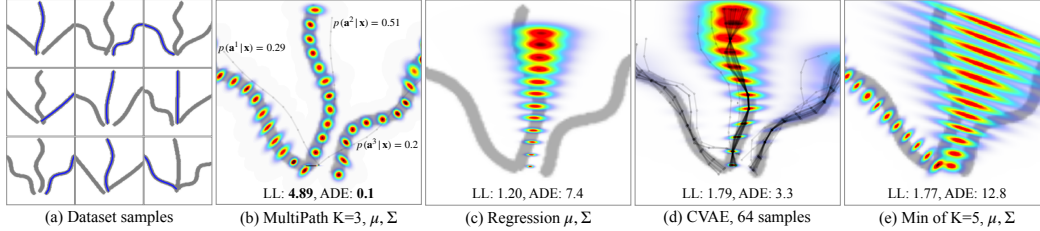| (a) Dataset samples	(b) MultiPath K=3, μ, Σ	(c) Regression μ, Σ	(d) CVAE, 64 samples	(e) Min of K=5, μ, Σ
	LL: **4.89**, ADE: **0.1**	LL: 1.20, ADE: 7.4	LL: 1.79, ADE: 3.3	LL: 1.77, ADE: 12.8

Figure 2: Results on the 3-way intersection toy example. Due to large dynamic range, uncertainties shown with jet colormap are scaled per-plot per-timestep, and not directly comparable. (a) Samples drawn from the data generation procedure, with groundtruth paths shown as blue lines. (b) MultiPath with $K = 3$ anchors correctly learns the intent and uncertainty distributions, achieving high likelihood. The anchors (light gray) were estimated by averaging 10^5 samples. (c) Single-trajectory modeling predicts a mean over all paths with corresponding location uncertainty which grows as paths diverge. (d) CVAE samples with a Kernel Density Estimate distribution fit [5]. (e) Min-of $K = 5$ trajectories. This model is very sensitive to initial weights, and on 5 trials with 4 learning rates, collapsed to only 1 or 2 active modes only (2 are shown here). We initialized starting regression weights to be uniform random in a small region surrounding the $t = 0$ position, for a better chance of learning multiple unique modes.

Log-likelihood (LL). We report $\log p(\hat{\mathbf{s}}|\mathbf{x})$ if the model admits evaluation of likelihood, as does MultiPath when all parameters are learned (see Eq. (2)). The metric is scaled down by a factor of $2 \times T$, where T is the number of time steps and 2 for the two spatial dimensions.

Distance-based. In this category are the commonly-used average displacement error (**ADE**) $\frac{1}{T} \sum_{t=1}^{T} \|\hat{s}_t - s_t^*\|_2$ and final displacement error (**FDE**) $\|\hat{s}_T - s_T^*\|_2$, where $\mathbf{s}^*$ is the most-likely within a weighted set. For evaluating a set of trajectories, $\textbf{minADE}_M \min_{s_m} \frac{1}{T} \sum_{t=1}^{T} \|\hat{s}_t - s_{m,t}\|_2$ measures the displacement error against the closest trajectory in the set of size M, so that reasonable predictions that simply do not happen to be the logged groundtruth are not penalized. Note that there is also the $\textbf{minMSD}_M$ [4], which is similar but the average is calculated on squared distances instead.

4.2 Toy experiment: 3-way intersection

We first explore a simple proof-of-concept dataset generated based on our modeling assumptions. We generate synthetic 3-way intersections, with the probability of choosing the left, the middle or the right path set a priori to be the *intent uncertainty* distribution $\{0.3, 0.5, 0.2\}$. To emphasize the flexibility of our single-trajectory *control uncertainty* modeling, each path is generated by sampling parameterized sine waves: $y = \sin(\omega t + \phi)$, where the frequency $\omega \sim \mathcal{U}(0, 2)$ and phase shift $\phi \sim \mathcal{U}(-\pi, \pi)$. As shown in Figure 2, MultiPath is able to fit the underlying distribution correctly, recovering the intent uncertainty, and reaching approximately Bayes-optimal likelihood, while other methods fare worse.

4.3 Behavior prediction for autonomous driving.

To verify the performance of the proposed system, we collected a large dataset of real-world driving scenes from several cities in North America. Data is captured by a vehicle equipped with cameras, lidar and radar. As in [2, 6], we assume that an industry-grade perception system provides sufficiently accurate poses and tracks for all nearby agents, including vehicles, pedestrians, and cyclists. In our experiments, we treat the sensing vehicle as an additional agent, indistinguishable from any other agent in the scene. Most of the collected vehicle trajectories are either stationary or moving straight at a constant speed. Neither case is particularly interesting from a behavior prediction point of view. To address this and other dataset skew, we partitioned the space of future trajectories via a uniform, 2D grid over constant curvatures and distances, and performed stratified sampling such that the number of examples in each partition was capped to be at most 5% of the resulting dataset. The balanced dataset totals 3.85 million examples, contains 5.75 million agent trajectories and constitutes approximately 200 hours of driving.

The top-down rendered input tensor for this data has a resolution of 400 px $\times$ 400 px and corresponds to 80 m $\times$ 80 m in real-world coordinates. We sample time steps every 0.2 s (5 Hz). The following features are stacked in the depth dimension: 3 channels of color-coded road semantics, 1 channel

Table 1: Comparison of MultiPath on a large autonomous driving dataset. Brackets appear for cells where a metric expects a set of future trajectories but the method only produces one. All experiments (except for *linear*) were repeated 5 times with random initialization to produce the mean and standard deviation values. **LL** is the log-likelihood. **ADE** is the average distance error. **minADE**$_M$ is the top-M average distance error given M trajectory predictions. One out of five CVAE runs degenerated. **CVAE** includes all 5 runs, while **CVAE select** excludes the degenerated run. A more detailed analysis is in Sec. A.

| Method	LL ↑	ADE ↓	minADE$_5$ ↓	minADE$_{10}$ ↓	minADE$_{15}$ ↓
Linear	-	3.26	(3.26)	(3.26)	(3.26)
Regression μ	-	**1.17**$_{\pm 0.01}$	(1.17$_{\pm 0.01}$)	(1.17$_{\pm 0.01}$)	(1.17$_{\pm 0.01}$)
Regression μ, Σ	3.64$_{\pm 0.01}$	1.41$_{\pm 0.02}$	(1.41$_{\pm 0.02}$)	(1.41$_{\pm 0.02}$)	(1.41$_{\pm 0.02}$)
CVAE	-	2.16$_{\pm 2.15}$	1.82$_{\pm 2.35}$	1.74$_{\pm 2.39}$	1.71$_{\pm 2.41}$
CVAE select	-	1.20$_{\pm 0.03}$	0.77$_{\pm 0.03}$	0.67$_{\pm 0.03}$	0.63$_{\pm 0.03}$
Min-Of-K μ [20]	-	1.37$_{\pm 0.02}$	0.87$_{\pm 0.02}$	0.86$_{\pm 0.02}$	0.86$_{\pm 0.02}$
Min-Of-K μ, Σ [20]	4.26$_{\pm 0.04}$	1.23$_{\pm 0.02}$	0.70$_{\pm 0.01}$	0.70$_{\pm 0.01}$	0.70$_{\pm 0.01}$
MultiPath μ (ours)	-	**1.17**$_{\pm 0.00}$	**0.58**$_{\pm 0.00}$	**0.48**$_{\pm 0.00}$	**0.46**$_{\pm 0.00}$
MultiPath μ, Σ (ours)	**4.37**$_{\pm 0.00}$	1.25$_{\pm 0.01}$	0.63$_{\pm 0.00}$	0.61$_{\pm 0.00}$	0.61$_{\pm 0.00}$
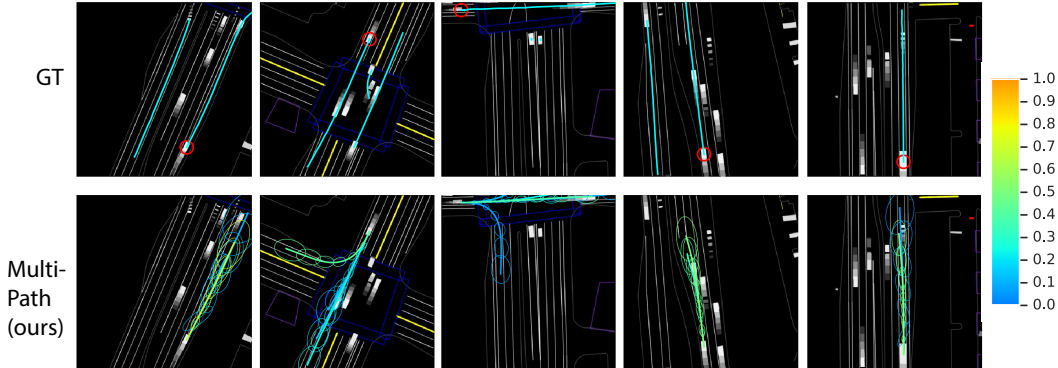
Figure 3: **MultiPath example results.** We show in one column the result for one agent (focused agent) in a scene. **Top**: Logged trajectories of all agents are displayed in cyan. The focused agent is highlighted by a red circle. **Bottom**: MultiPath showing up to 5 trajectories with uncertainty ellipses. Trajectory probabilities (softmax outputs) are encoded in a color map shown to the right. MultiPath can predict uncertain future trajectories for various speed (1st column), different intent at intersections (2nd and 3rd columns) and lane changes (4th and 5th columns), where the regression baseline only predicts a single intent.

of distance-to-road-edge map, 1 channel encoding the speed limit, 5 channels encoding the traffic light states over the past 5 time steps (=1 second), and 5 channels each showing vehicles' top-down orthographic projection for each of the past 5 time steps. This results in 15 input channels in total. We predict trajectories up to 30 frames / 6 seconds into the future. The number of anchors K is set to 16 for MultiPath μ, Σ and 64 for MultiPath μ. The scene-level network is a ResNet50 [24] with a depth multiplier of 25%, followed by a depth-to-space operation that restores some of the lost spatial resolution in the ResNet back to 200×200. Finally, we train the model end-to-end for 500k steps at a batch size of 32, with a learning rate warm-up phase and a cosine learning rate decay

Experimental results are shown in Tab. 1. MultiPath outperforms the baselines in all metrics. With respect to the log-likelihood, we have observed the most log-likelihood measurements for this task to fall between 3 to 4.2 nats, so the gain of roughly 0.2 nat by MultiPath compared to the regression baseline is quite significant. See Sec. A for in-depth analyses of these results. 16 anchors are used for MultiPath μ, Σ, while 64 was the best K for MultiPath μ. An analysis of the effect of the number of anchors K is in Sec. B.1, while the figures in Sec. C visualize the anchors.

4.4 Stanford Drone

The Stanford Drone Dataset [28] consists of top-down, near-orthographic videos of college campus scenes, collected by drones, containing interacting pedestrians, cyclists and vehicles. The RGB camera frames provide context similar to a rendered road semantics in the driving vehicle environment, and

Table 2: Comparison of MultiPath and baselines on the Stanford Drone Dataset (SDD). Distance measures are in terms of pixels in the original video resolution.

Method	LL $\uparrow$	ADE $\downarrow$	FDE $\downarrow$	minADE$_5$ $\downarrow$
Linear	–	26.14	53.24	–
CVAE	–	30.91	61.40	26.29
Regression μ	–	27.44	56.44	–
Regression μ, Σ	3.06	26.67	54.34	–
MultiPath μ, Σ	**3.52**	28.32	58.38	**17.51**
DESIRE-SI-IT0 [3]	–	36.48	61.35	30.78
CAR-Net [9]	–	**25.72**	**51.80**	–
Social Forces [26]	–	36.48	58.14	–
Social LSTM [27]	–	31.19	56.97	–

Table 3: Comparison of MultiPath and baselines on the CARLA Dataset. **minMSD$_{12}$** is the minimum mean-squared-distance computed on the top 12 predicted trajectories.

Town01 (5 agents)	minMSD$_{12}$ $\downarrow$	Town02 (5 agents)	minMSD$_{12}$ $\downarrow$
DESIRE [3]	2.60	DESIRE [3]	2.42
SocialGAN [19]	1.46	SocialGAN [19]	1.14
R2P2-MA [4]	0.84	R2P2-MA [4]	0.77
ESP [18]	0.72	ESP [18]	**0.68**
MultiPath μ, Σ	**0.68**	MultiPath μ, Σ	0.69

we treat it as such. We use the most common settings in the literature: sampling at 2.5 Hz, and predicting 4.8 seconds (12 frames) into the future, using 2 seconds of history (5 frames). Additional experimental details are in Sec. D.

As shown in Tab. 2, we perform at or better than state-of-the-art in best single-trajectory distance metrics. Notably, CAR-Net [9] outperforms our comparable single-trajectory model; their method focuses on a sophisticated attention and sequential architecture tuned to get the best single-trajectory distance metric performance. Interestingly, our single-trajectory model performs better when trained to predict uncertainty as well, a potential benefit of modeling uncertainty discussed in [8].

4.5 CARLA

We evaluate MultiPath on the publicly available multi-agent trajectory forecasting and planning dataset generated using the CARLA [29] simulator by [18]. Experimental details are in Sec. E. Tab. 3 reproduces results reported by [18] for the DESIRE [3], SocialGAN [19], R2P2-MA [4], and the PRECOG-ESP [18] methods and compares the performance of MultiPath against them. We report the minMSD metric with the top $K = 12$ predictions as defined in [18] to report our evaluation results.

5 Conclusion

We have introduced MultiPath, a model which predicts parametric distributions of future trajectories for agents in real-world settings. Through synthetic and real-world datasets, we have shown the benefits of MultiPath over previous single-trajectory and stochastic models in achieving likelihood and trajectory-set metrics and needing only 1 feed-forward inference pass.

References

[1] 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. In *CVPR*, 2018.

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

[3] 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. In *CVPR*, 2017.

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

[5] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

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

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

[8] A. Kendall and Y. Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *NeurIPS*, 2017.

[9] A. Sadeghian, F. Legros, M. Voisin, R. Vesel, A. Alahi, and S. Savarese. Car-net: Clairvoyant attentive recurrent network. In *ECCV*, 2018.

[10] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV*, 2009.

[11] M. Bansal, A. Krizhevsky, and A. Ogale. ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst. *RSS*, 2019.

[12] B. Ivanovic, E. Schmerling, K. Leung, and M. Pavone. Generative modeling of multimodal multi-human behavior. In *IROS*, 2018.

[13] A. Bhattacharyya, B. Schiele, and M. Fritz. Accurate and diverse sampling of sequences based on a "best of many" sample objective. In *CVPR*, 2018.

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

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

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

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

[18] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. PRECOG: Prediction conditioned on goals in visual multi-agent settings. *arXiv preprint arXiv:1905.01296*, 2019.

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

[20] 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.

[21] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *CVPR*, 2014.

[22] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *PAMI*, 2012.

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot MultiBbox detector. In *ECCV*, 2016.

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

[25] S. Becker, R. Hug, W. Hubner, and M. Arens. RED: A simple but effective baseline predictor for the TrajNet benchmark. In *ECCV*, 2018.

[26] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg. Who are you with and where are you going? In *CVPR 2011*, 2011.

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

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

[29] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

A   In-depth analysis

We provide a more detailed analysis of Tab. 1. The results are strictly different representations of the same experiments, all evaluated on the autonomous driving dataset, as described in Sec. 4.3.

A.1   Group by groundtruth final waypoint

Statistically, most vehicles we observe are parked (stationary) or going straight. Neither case is particularly interesting since a trivial solution such as the linear fitting is probably good enough to solve them. In this exercise, we split the evaluation samples into seven disjoint categories according to the location of the final waypoint at 6 seconds: 1) **Stationary**, agents that have moved less than 4 meters; 2) **Slow**, agents that moved farther than 4 meters but less than 8 meters; 3) **Straight**, agents that moved farther than 8 meters and the final waypoint is within $\pm5°$ with respect of the initial heading; 4) Slight left (**SLeft**), farther than 8 meters and between $-5°$ and $-30°$; 5) **Left**, farther than 8 meters and larger than $-30°$; 6) Slight right (**SRight**), same as SLeft, but in the other direction; 7) **Right**, same as Left, but in the other direction.

Results are shown in Fig. 4. In Fig. 4a, we compare MultiPath μ, Σ against the regression baseline μ, Σ on the log-likelihood metric. As expected, the baseline performs reasonably well in the Stationary and Slow categories but is unable to model faster trajectories, where MultiPath outperforms the baseline decisively.
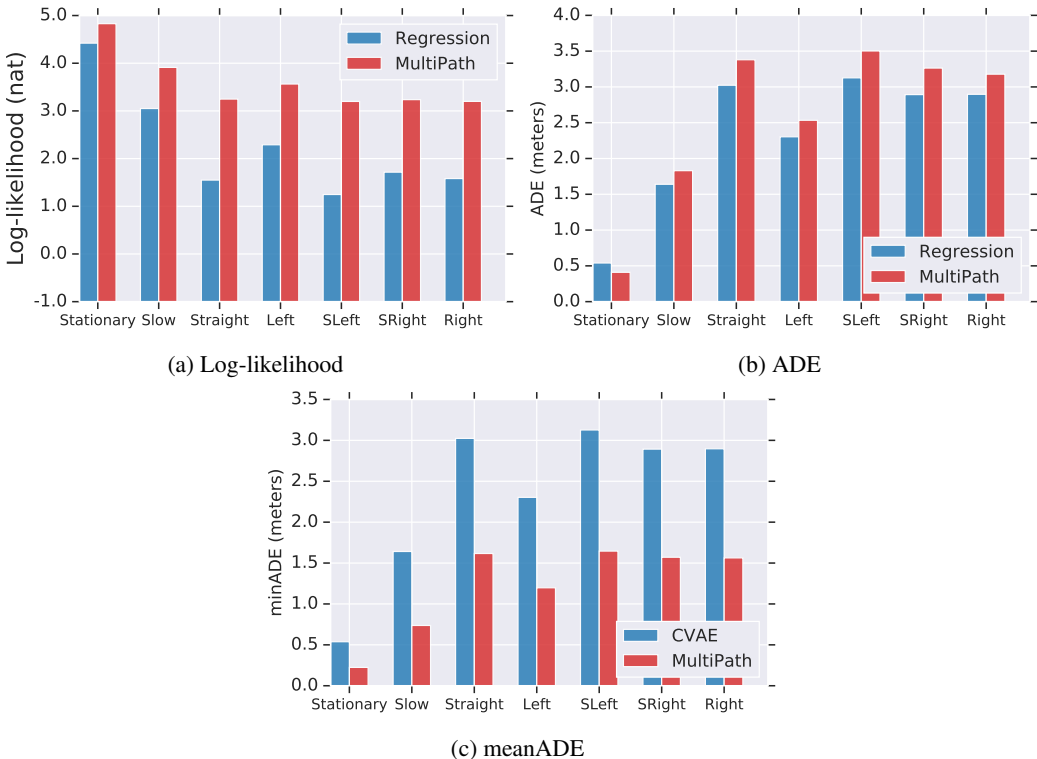

(a) Log-likelihood   (b) ADE

(c) meanADE

Figure 4: Detailed results grouped by the groundtruth final waypoint. (a): Detailed comparison between MultiPath μ, Σ and the regression μ, Σ baseline on the log-likelihood. (b): Detailed comparison between MultiPath μ and the regression μ baseline on the ADE. Detailed comparison between MultiPath μ and the CVAE baseline on the meanADE. The top 5 trajectories are kept for MultiPath, and 100000 trajectories are sampled from the CVAE.

A.2   Group by time step

We look at the distance-based errors as the function of the predicted time step up to 6 seconds in Fig. 5. This error is usually expected to grow linearly in time. However, our curves are slightly super-linear, most likely caused by the natural distribution of heading-normalized trajectories, where the acceleration in both heading and speed is smooth.
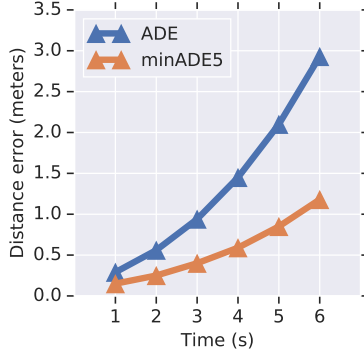
Figure 5: Detailed results from MultiPath μ grouped by time step on the ADE and minADE$_5$ metrics.

B Hyperparameters

There are numerous moving parts of the proposed model. In this section, we conducted extensive experiments on the choice of key hyperparameters, discuss the sensitivity of the model with respect to these parameters.

B.1 Number of anchors K

One of the key hyperparameters of MultiPath is the choice of the number of anchors K, as it is always the case with methods that rely on unsupervised clustering. To this end, we trained several MultiPath μ, Σ and MultiPath μ models with different Ks, the results are shown in Fig. 6.

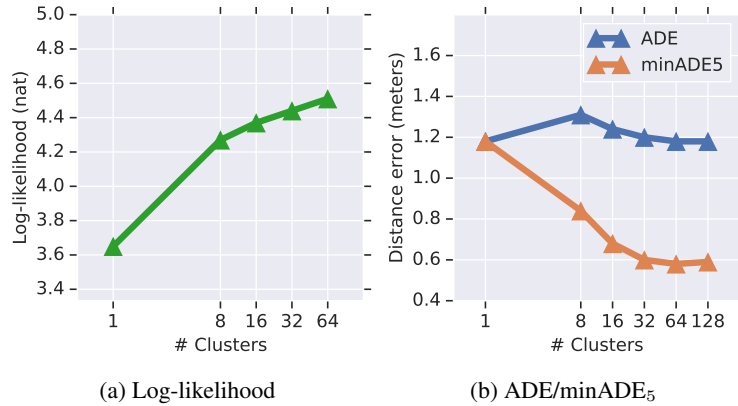(a) Log-likelihood   (b) ADE/minADE$_5$

Figure 6: Results as function of the number of clusters K. (a): MultiPath μ, Σ on the log-likelihood metric with K ranging from 1 to 64. (b): MultiPath μ on the ADE and minADE$_5$ metrics, with K ranging from 1 to 128.

B.2 Backbone network sensitivity

We run the MultiPath μ, Σ model using 4 ResNet [24] backbones of various complexity: ResNet8_thin, ResNet18_thin, ResNet50_thin, ResNet50. The backbones denoted with em thin are using a depth multiplier of 25%. The results are shown in Fig. 7.

B.3 Importance of heading normalization

The heading of a vehicle is a valuable signal, especially since cars cannot move in arbitrary directions. In Sec. 4.3, we assumed that the heading of the agents is given, alongside the positions and sizes of them over the past 1 second. We then crop the agent-specific feature map while compensating for the heading, so that the heading of the agent points to the same direction after the crop. In Fig. 8, we show the importance of this heading normalization.
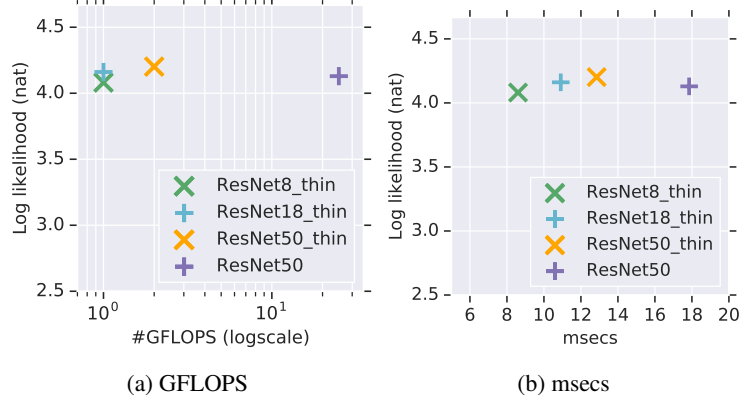
(a) GFLOPS

(b) msecs

Figure 7: Analysis of different backbone networks. a) Theoretical complexity in GFLOPS (1 billion FLOPS). b) Empirical latency on a TITAN GPU, averaged over 100 runs. The ResNet50_thin backbone is used in all other experiments. Complexity and latency are measured assuming exactly 10 agents per scene.

As expected, the sweet spot for the number of clusters K larger when the heading is not compensated for. It appears that empirically, the heading is of great importance, and the heading normalized models outperform their counterparts on all three metrics.

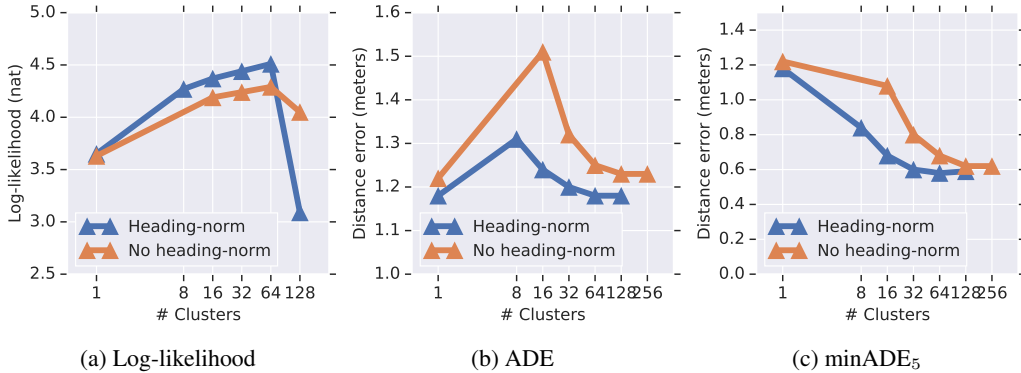(a) Log-likelihood

(b) ADE

(c) minADE$_5$

Figure 8: Impact of heading normalization on three metrics. Having the heading compensated during agent-centric feature crop is beneficial throughout.

B.4   Road semantic information and traffic lights

One main benefit of using a rasterized top-down input representation is the simplicity to encode spatial information such as semantic road information and traffic lights, which are crucial for human driving. To validate that MultiPath also considers this contextual information, we trained ablation models by stripping the road information and traffic lights from the input. Results are shown in Tab. 4. The gain of including the contextual information is significant in all distance-based metrics. Interestingly, the gain is less apparent in the log-likelihood. We believe this is due to the fact that the context is mostly useful to prevent false positive trajectories, which has a smaller impact on the log-likelihood.

Table 4: Impact of including the semantic road information and traffic light context.

Method	Log-likelihood ↑	ADE ↓	FDE ↓	minADE$_5$ ↓
With context μ, Σ	4.37	1.25	3.17	0.63
No context μ, Σ	4.32	1.45	3.83	0.68
With context μ	-	1.18	2.93	0.58
No context μ	-	1.44	3.71	0.73

C Visualization of all anchors

The anchors for the autonomous driving dataset for the various number of anchors K are shown in Fig. 9. Since the data is captured in North America, left turns are expected to have a larger curvature than right turns, and u-turns mostly are on the left side.
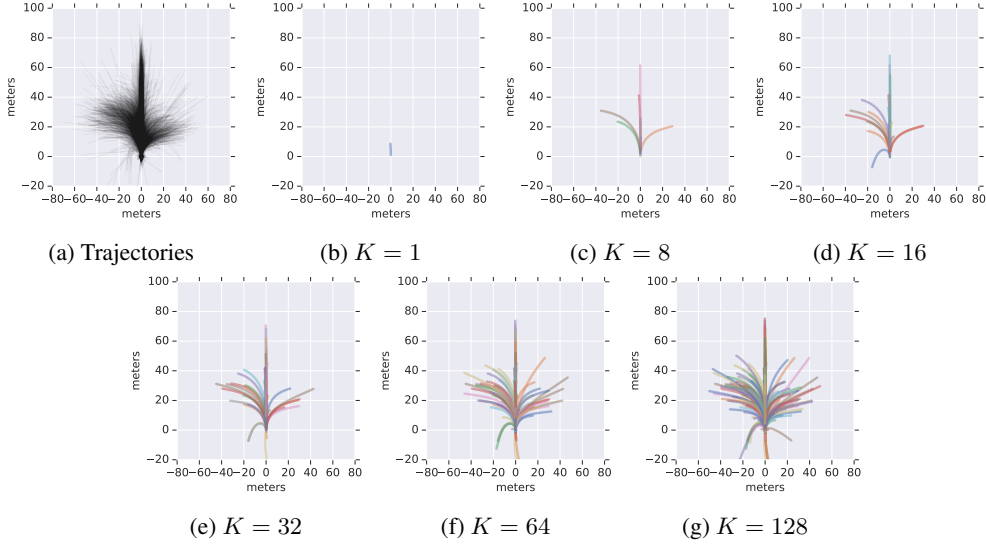

(a) Trajectories (b) $K = 1$ (c) $K = 8$ (d) $K = 16$

(e) $K = 32$ (f) $K = 64$ (g) $K = 128$

Figure 9: Visualization of trajectories and their clusters. **(a)**: 20k randomly sampled trajectories. **(b) - (g)**: Anchors for a variety of K values.

D Stanford Drone Dataset experimental details

Unlike much of the past literature, we do not encode the raw location information directly. In keeping with our top-down rendered representation as detailed in Sec. 3, we render a history of oriented boxes as input channels concatenated with the current RGB video frame. Oriented boxes are estimated via past positions plus fixed-size extents. For consistent processing, we resize and pad videos to have a resolution of 800 px $\times$ 800 px, maintaining the original aspect ratio, but report results at the original resolution.

Due to the small size of the dataset and lack of heading information, we created our anchor set for this dataset by enumeration rather than K-means: we chose 64 straight-trajectory anchors via 16 evenly-spaced orientations at 4 different final-waypoint distances (roughly 5%, 10%, 15% and 20% of the max image dimension), and 1 stationary anchor. Due to the complexity of natural image (RGB) input, we chose a higher-capacity feature processing backbone than other datasets: 28-layer ResNet, each layer of depth 32. This proved beneficial over our default choice in Sec. 4.3, and we did no other architecture search.

E CARLA experimental details

This dataset contains $60,701$ training, $7,586$ validation and $7,567$ test scenes from Town01 and $16,960$ test scenes from Town02. Each scene contains the autopilot's LIDAR observation together with 2 seconds of past and 4 seconds of future position information for the autopilot and 4 other agents at 5Hz. To allow comparison to the numbers reported in [18], we follow their training data setup: we use a 100×100 pixel crop of the top-down images, and only use the two LIDAR channels that represent a 2-bin histogram of points above and at ground-level in 0.5 m $\times$ 0.5 m cells.