Motion Prediction using Trajectory Sets and Self-Driving Domain Knowledge

Freddy A. Boulton **Elena Corina Grigore**

Eric M. Wolff
nuTonomy, Hyundai-Aptiv Joint Venture
{freddy.boulton, elena.corina.grigore, eric}@nutonomy.com

Abstract

Predicting the future motion of vehicles has been studied using various techniques, including stochastic policies, generative models, and regression. Recent work has shown that classification over a trajectory set, which approximates possible motions, achieves state-of-the-art performance and avoids issues like mode collapse. However, map information and the physical relationships between nearby trajectories is not fully exploited in this formulation. We build on classification-based approaches to motion prediction by adding an auxiliary loss that penalizes off-road predictions. This auxiliary loss can easily be *pretrained* using only map information (e.g., off-road area), which significantly improves performance on small datasets. We also investigate weighted cross-entropy losses to capture spatial-temporal relationships among trajectories. Our final contribution is a detailed comparison of classification and ordinal regression on two public self-driving datasets.

1 Introduction

Self-driving cars must share the road with other vehicles, bicyclists, and pedestrians. To safely operate in this dynamic and uncertain environment, it is important to reason about the likely future motions of other road users. Each road user has different goals (e.g., turn left, speed up, stop) and preferences (e.g., desired speed, braking profile), which influence his or her actions. Thus, useful predictions of future motion must represent multiple possibilities and their associated likelihoods.

Motion prediction is fundamentally challenging due to the uncertainty in road user behavior. Likely behaviors are constrained by both the road (e.g., following lanes, stopping for traffic lights) and the actions of others (e.g., slowing for a car ahead, stopping for a pedestrian crossing the road).

State-of-the-art motion prediction models learn features over a combined representation of the map and the recent history of other road users [6, 12, 19]. These methods use a variety of multimodal output representations, including stochastic policies [28], occupancy maps [19], and regression [6, 12]. Recent work [27] has shown that classification over a trajectory set, which approximates possible motions, achieves state-of-the-art performance and avoids issues like mode collapse. However, that work did not use map data and trajectory set geometry in the loss, and had a limited comparison with ordinal regression. This paper addresses those limitations.

The classification loss used in [27] does not explicitly penalize predictions that go off the road. To use the prior knowledge that cars typically drive on the road, we simplify and adapt the off-road loss from [25] to classification over trajectory sets. Our formulation allows *pretraining* a model using only map data, which significantly improves performance for small datasets.

Preprint. Under review.

The trajectory sets introduced for motion prediction in [27] have rich spatial-temporal structure. However, the cross-entropy loss they used does not exploit the fact that classes correspond to physical trajectories. We investigate weighted cross-entropy losses that determine how much "near-misses" in the trajectory set are penalized. Surprisingly, we find that a variety of weighted losses do not improve on the original formulation. We analyze these results in the context of prediction diversity.

The classification-based approach of [27] is closely related to the ordinal regression approach of [6], which computes residuals from a set of "anchor" trajectories that are analogous to a trajectory set. We perform a detailed comparison of these approaches on two public datasets. Interestingly, we find that performance of ordinal regression can benefit from using an order of magnitude more "anchor" trajectories than previously reported [6].

Our main contributions on multimodal, probabilistic motion prediction are summarized as follows:

- use an off-road loss with pretraining to help learn domain knowledge;
- explore weighted cross-entropy losses to capture spatial relationships within a trajectory set;
- carefully compare classification vs ordinal regression on nuScenes [3] and Argoverse [7].

2 Related Work

State-of-the-art motion prediction algorithms now typically use CNNs to learn appropriate features from a birds-eye-view rendering of the scene (map and road users). Other road users are represented as sensor data [5, 23] or the output of a tracking and fusion system [12, 19]. Recent work [4, 15] has also explored using graph neural networks (GNNs) to encode interactions.

Complementing the input representations described above, various approaches have been used to represent the possible future motions. Generative models encode choice over multiple actions via sampling latent variables. Examples include stochastic policies [21, 28, 29, 32], CVAEs [2, 19, 20, 22] and GANs [17, 31, 34]. These approaches require multiple samples or policy rollouts at inference.

Regression models either predict a single future trajectory [1, 5, 14, 23], or a distribution over multiple trajectories [12, 13, 19]. The former unrealistically average over behaviors in many driving situations, while the latter can suffer from mode collapse. In [6], the authors propose a hybrid approach using ordinal regression. This method regresses to residuals from a set of pre-defined anchors, much like in object detection. This technique helps mitigate mode collapse.

CoverNet [27] frames the problem as classification over a trajectory set, which approximates all possible motions. Our work further explores this formulation through losses that use the map and trajectory set geometry, as well as detailed experimental comparison to ordinal regression.

2.1 Use of Domain Knowledge

Both dynamic constraints and "rules-of-the-road" place strong priors on likely motions. Dynamic constraints were explicitly enforced via trajectory sets in [27] and a kinematic layer in [11].

Prior work using map-based losses includes the use of an approximate prior that captures off-road areas as part of a symmetric KL loss [28], and an off-road loss on future occupancy maps [24]. However, these loss formulations are not directly compatible with most trajectory prediction approaches that output point coordinates. The most closely related work is [25], which applies an off-road loss to multimodal regression. Our formulation is simpler and allows pretraining using just the map.

We leverage domain knowledge by encoding the relationships among trajectories via a weighted cross-entropy loss, where the weight is a function of distance to the ground truth. To our knowledge, this loss has not been explored in motion forecasting, likely due to the prior focus on generative and regression models. This loss is typically used to mitigate class imbalance problems [8].

2.2 Public datasets

Until recently, public self-driving datasets suitable for motion prediction were either relatively small [16] or for highway driving [9]. Accordingly, many publications are evaluated only on private datasets [5, 12, 24, 33]. We report results on two recently released self-driving datasets, nuScenes [3]

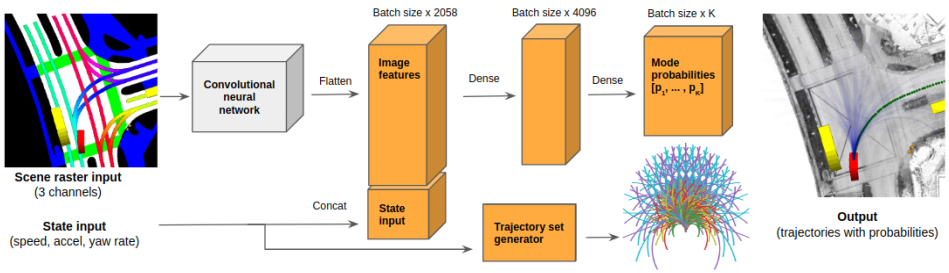


Figure 1: CoverNet overview from [27].

and Argoverse [7], to help establish clear comparisons between state-of-the-art methods for motion forecasting on city roads.

3 Preliminaries

We now set notation and give a brief, self-contained overview of CoverNet [27], which we will extend with modified loss functions in Section 4. CoverNet uses the past states of all road users and a high-definition map to compute a distribution over a vehicle's possible future states.

3.1 Notation

We use the state outputs of an object detection and tracking system, and a high-definition map that includes lanes, drivable area, and other relevant information. Both are typically used by self-driving cars operating in urban environments and are also assumed in [6, 12, 27].

We denote the set of road users at time t by $\mathcal{I}_t$ and the state of object $i \in \mathcal{I}_t$ at time t by s_t^i. 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 object i at times $t = m, \ldots, n$. Let $\mathcal{C} = \{\bigcup_i s_{t-m:t}^i; \text{map}\}$ denote the scene context over the past m steps (i.e., partial history of all objects and the map). We are interested in predicting $s_{t:t+T}^i$ given $\mathcal{C}$, where T is the prediction horizon.

3.2 CoverNet overview

We briefly summarize CoverNet [27], which makes multimodal, probabilistic trajectory predictions for a vehicle of interest by classifying over a *trajectory set*. Figure 1 overviews the model architecture.

Input The input is a birds-eye-view raster image centered around the agent of interest that combines both map data and the past states of all objects. The raster image is an RGB image that is aligned so that the agent's heading points up. Map layers are drawn first, followed by vehicles and pedestrians. The agent of interest, each map layer, and each object type are assigned different colors in the raster. The sequence of past observations for each object is represented through fading object colors using linearly decreasing saturation (in HSV space) as a function of time.

Output The output is a distribution over a *trajectory set*, which approximates the vehicle's possible motions. Using a trajectory set is a reasonable approximation given the relatively short prediction horizons (3 to 6 seconds) and inherent uncertainty in agent behavior. Let $\mathcal{K}$ be a trajectory set. Then, the output dimension is equal to the number of modes, namely $|\mathcal{K}|$.

The output layer applies the softmax function to convert network activations to a probability distribution. The probability of the k-th trajectory is $p(s_{t:t+T}^k|x) = \frac{\exp f_k(x)}{\sum_i \exp f_i(x)}$, where $f_i(x) \in \mathbb{R}$ is the output of the network's penultimate layer.

Training The model is trained as a multi-class classification problem using a standard cross-entropy loss. For each example, the positive label is the element in the trajectory set closest to the ground truth, as measured by the minimum average of point-wise Euclidean distances.

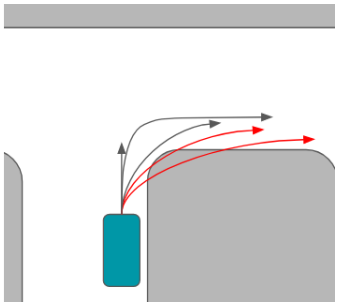


Figure 2: Visualization of on-road (black) and off-road (red) trajectories.

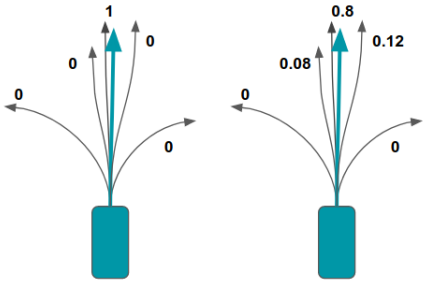


Figure 3: Visualization of the target distribution in the standard cross-entropy formulation (left), and the weighted cross-entropy loss (right).

4 Losses

We now introduce an off-road loss to partially encode "rules-of-the-road," and a weighted cross-entropy classification loss to capture spatial relationships in a trajectory set.

4.1 Off-road loss

Let $x_t^i \in \mathbb{R}^{|\mathcal{K}|}$ be the activations of CoverNet's penultimate layer for agent i at time t and let $r_t^i \in \mathbb{R}^{|\mathcal{K}|}$ be a binary vector whose value is 1 at entry k if trajectory k in the set is entirely contained in the drivable area. Let $\sigma(x)$ be the sigmoid function and let $\text{bce}(\hat{y}, y) = (1-y)\log(1-\hat{y}) + y\log(\hat{y})$ be the binary cross-entropy loss. We define the off-road loss as

$$\Omega_t^i = \sum_{k=1}^{|\mathcal{K}|} \text{bce}(\sigma(x_i^t), r_i^t). \tag{1}$$

4.2 Weighted cross-entropy loss

The approach proposed in [27] uses a cross-entropy loss where positive samples are determined by the element in the trajectory set closest to the ground truth. This loss penalizes the second-closest trajectory just as much as the furthest, since it ignores the geometric structure of the trajectory set.

Our proposed modification is to create a probability distribution over all modes that are "close enough" to the ground truth. So, instead of a delta distribution over the closest mode, there is also probability assigned to near misses (see Figure 3).

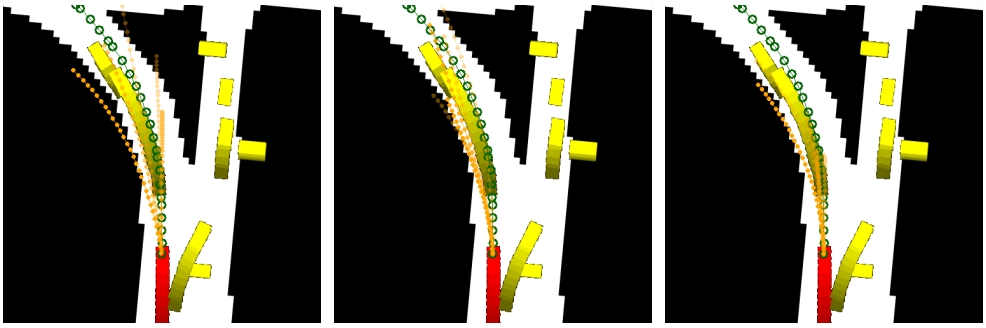
Let D be a real-valued function that measures the distance between trajectories in trajectory set $\mathcal{K}$ and the ground truth for agent i at time t and let $d_t^i = D(\mathcal{K}, s_{t:t+T}^i) \in \mathbb{R}^{|\mathcal{K}|}$. We set a threshold that defines which trajectories are "close enough" to the ground truth and let $\mathcal{K}_{\text{close}} \subset \mathcal{K}$ be the set of these trajectories. We experiment with D as both the max and mean of element-wise Euclidean distances, denoted by Max ℓ^2 and Mean ℓ^2, respectively.

We take the element-wise inverse of d_t^i and set the value of trajectories not in $\mathcal{K}_{\text{close}}$ to 0. We linearly normalize the entries of this vector so they sum to 1 and use this vector as the target probability distribution with the standard cross-entropy loss and we denote it by w_t^i. Finally, let S be the softmax function and let $S(x_t^i) \in \mathbb{R}^{|\mathcal{K}|}$ be the result of applying softmax to the model's penultimate layer. Our weighted cross entropy loss is defined as:

$$\mathcal{H}_t^i = \sum_{k=1}^{|\mathcal{K}|} -w_{tk}^i \log S(x_t^i)_k. \tag{2}$$

We introduce hyperparameter $\lambda \in [0, \infty)$ to encode the relative weight of the off-road loss versus the classification loss. The total loss for agent i at time t is then

$$\mathcal{L}_t^i = \mathcal{H}_t^i + \lambda \Omega_t^i. \tag{3}$$



(a) CoverNet, $\varepsilon = 2$, $\lambda = 0$ (b) CoverNet, $\varepsilon = 2$, $\lambda = 1$ (c) CoverNet, $\varepsilon = 2$, $\lambda = 10$

(d) Unimodal (e) MultiPath [12], 64 modes (f) MultiPath [12], $\varepsilon = 2$

Figure 4: Examples of predicted trajectories on the same Argoverse scenario. The top row includes a CoverNet model with $\varepsilon = 2m$, in increasing order of off-road penalty. The bottom row includes the baselines we compare against. Objects in the world are rendered up to the current time. Note how the CoverNet trajectories comply with the drivable area as the penalty increases.

5 Experiments

For our motion prediction experiments, the input representation and model architecture were fixed across all models and baselines. We varied both the loss functions and output representations.

5.1 Baselines

Physics oracle. The *physics oracle* [27] is the minimum average point-wise Euclidean distance over the following physics-based models: i) constant velocity and yaw, ii) constant velocity and yaw rate, iii) constant acceleration and yaw, and iv) constant acceleration and yaw rate.

CoverNet [27]. As described in Section 3.2.

Regression to anchor residuals. We compare our contributions to MultiPath (MP) [6], a state-of-the-art multimodal regression model. 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. 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:

$$\mathcal{L}_{it}^{MP} = \sum_{k=1}^{|\mathcal{K}|} \mathbb{1}_{k=\hat{k}}[-\log p_{ik} + \alpha L(s_{t:t+N}^{i}, \hat{s}_{t:t+N}^{i})], \tag{4}$$

where $\mathbb{1}(\cdot)$ is the indicator function that equals 1 only for the "closest" mode, k is the mode index, L is the regression loss, and α is a hyper-parameter used to trade off between classification and regression. The k-th trajectory is the sum of the corresponding anchor and predicted residual. We use $\alpha = 1$ and use CoverNet trajectory sets as the fixed anchors.

5.2 Implementation details

Our implementation largely follows [12] and [27]. See Figure 1 for an overview.

The input raster is an RGB image of size $(H, W, 3)$. The image is aligned so that the agent's heading faces up, and the agent is placed on pixel (l, w) as measured from the top-left of the image. In our experiments, we use a resolution of 0.25 meters per pixel and choose $l = 320$ and $w = 200$. Thus, the model can "see" 80 meters ahead, 20 meters behind, and 25 meters on each side of the agent. Objects are rendered based on their oriented bounding box, which are imputed for Argoverse.

All models use a ResNet-50 [18] backbone with pretrained ImageNet [30] weights [10]. We apply a global pooling layer to the ResNet *conv5* feature map and concatenate the result with the agent's speed, acceleration, and yaw rate. We then add a 4096 dimensional fully connected layer.

Trajectory sets for each dataset were constructed as described in [27] and our supplementary material. The variable ε specifies the maximum distance between the ground truth and the trajectory set.

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 coordinates 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 is the length of the prediction horizon, and F is the sampling frequency. We predict (x, y) coordinates, so $|\vec{x}| = 2$.

We use a smooth ℓ^1 loss for all regression baselines.

Models were trained for 20 epochs on a 4 GPU machine, using a batch size of 32, and a learning rate schedule that starts at $1\mathrm{e}{-3}$ and applies a multiplicative factor of 0.9 every epoch. We found that these parameters gave good performance across all models. The per-batch core time was ~300 ms.

5.3 Metrics

We use commonly reported metrics to give insight into various facets of multimodal trajectory prediction. In our experiments, we average the metrics described below over all instances.

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 use the notion of a *miss rate* to simplify interpretation of whether or not a prediction was "close enough." We define the $\text{MissRate}_{k,d}$ 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.

To measure how well predicted trajectories satisfy the domain knowledge that cars drive on the road, we use Drivable Area Compliance (DAC) [7]. DAC is computed as $(n - n_{\text{offroad}})/n$, where n is the total number of predictions and n_{offroad} is the number that are off the road.

5.4 The impact of domain knowledge

We investigate the impact of the domain knowledge that we encoded with the losses in Section 4. All results in this section are evaluated on the Argoverse [7] validation set.

Argoverse. Argoverse [7] is a large-scale motion prediction dataset recorded in Pittsburgh and Miami. Object centroids (imperfectly estimated by a tracking system) are published at 10 Hz. The task is to predict the last 3 seconds of each scene given its first 2 seconds and the map. We use the split for version 1.1 of the dataset, which includes $205,942$ training scenarios.

5.4.1 Off-road loss and pretraining

As all vehicles in Argoverse are on the drivable area, a perfect model should have drivable area compliance (DAC) of 1. Figure 5 explores how model performance varies with the amount of training data available. Using an off-road penalty improves DAC in all cases. Pretraining on the drivable area also gives a significant performance boost in data-limited regimes.

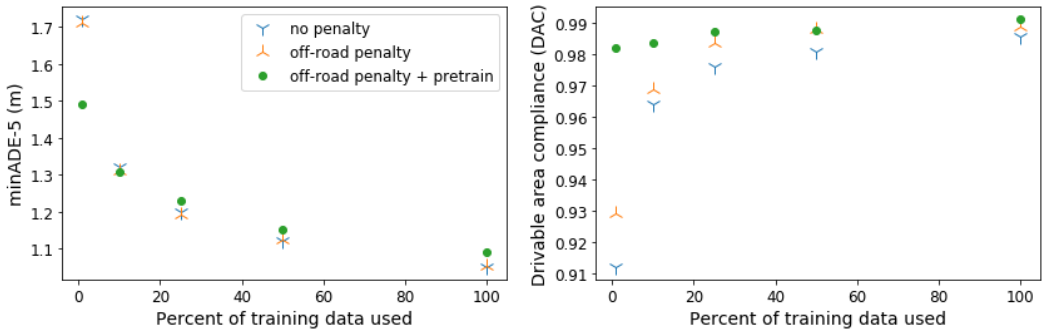


Figure 5: Effect of limited training data. The off-road penalty is $\lambda = 1$.

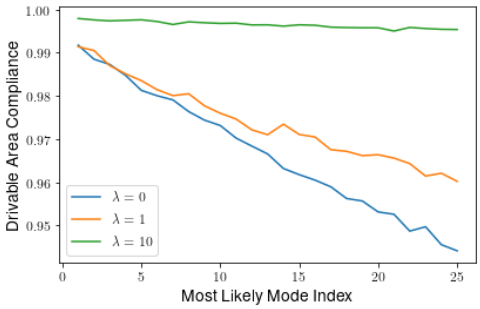


Figure 6: DAC ordered by mode probability.

Anchors	ℓ^1	ℓ^{inf}
16	0.70	63.67
64	0.49	74.34
682	0.29	5.76
1571	0.18	2.93

Figure 7: Ordinal regression residuals (m) on the Argoverse val set.

Figure 6 further explores the impact of off-road losses on DAC of less likely modes. We see small, but consistent improvements in DAC as we increase the off-road loss penalty. Furthermore, the relative improvements increase with less likely modes.

5.4.2 Weighted cross-entropy loss

Loss	CE	Max ℓ^2 WCE			Mean ℓ^2 WCE			Avoid Nearby WCE
Threshold (m)	N/A	2	3	1000	2	3	1000	2
$\text{minADE}_1 \downarrow$	**1.75**	1.79	1.84	2.01	2.05	1.91	1.94	**1.75**
$\text{minADE}_5 \downarrow$	**1.05**	1.19	1.26	1.42	1.30	1.32	1.41	**1.05**
$\text{MissRate}_{5,\,2\text{m}} \downarrow$	**0.38**	0.44	0.49	0.55	0.48	0.49	0.53	**0.38**
Mean dist. between modes	1.34	1.05	1.13	1.01	1.17	1.23	0.93	1.35

Table 1: Effect of various weighted cross-entropy (WCE) loss functions.

Table 1 compares performance of a dense trajectory set ($\varepsilon = 1m$) when we consider all trajectories within various distances of the ground truth as a match. Surprisingly, the cross-entropy loss performs better than weighted max ℓ^2 and mean ℓ^2 cross-entropy losses. We hypothesize that this is due to the weighted cross-entropy loss causing more "clumping" of modes. We explored this hypothesis by computing the mean distance between predicted trajectories, and results in Table 1 support this interpretation. To reduce clumping, we tried an "Avoid Nearby" weighted cross entropy loss that assigns weight of 1 to the closest match, 0 to all other trajectories within 2 meters of ground truth, and $1/|\mathcal{K}|$ to the rest. We see that we are able to increase mode diversity and recover the performance of the baseline loss. Our results indicate that losses that are better able to enforce mode diversity may lead to improved performance.

5.5 Classification vs. ordinal regression

We perform a detailed comparison of classification (CoverNet) and ordinal regression (MultiPath). All models use the same input representation and backbone to focus on the output representation.

nuScenes. nuScenes [3] is a self-driving car dataset that consists of $1,000$ scenes recorded in Boston and Singapore. Objects are hand-annotated in 3D and published at 2 Hz. The task is to predict the next 6 seconds given the prior 1 second and the map. We use the split publicly available in the nuScenes software development kit [26], which only includes moving vehicles. This split includes $32,186$ observations in the train set.

5.5.1 Discussion

For CoverNet models, we find that $\epsilon = 2m$ offers the best or second best performance for all metrics on both datasets. Additionally, smaller ϵ models tend to overfit, especially on the smaller nuScenes dataset. For MultiPath, we find that the $\text{MissRate}_{5,\,2\text{m}}$ monotonically decreases as the trajectory set size increases but using a modest number of modes, such as 64, can achieve the best minADE_5.

We emphasize the strong performance of the MultiPath approach with a large number of modes ($\epsilon = 3$ and $\epsilon = 2$). The MultiPath paper [6] noted decreasing performance with more than 64 anchors, whereas we see benefits from using an order of magnitude more. Our analysis suggests that the increase in performance associated with the higher number of modes happens due to better coverage of space via anchors, leaving the network to learn smaller residuals. Table 7 displays the average ℓ^1 and ℓ^{inf} norms of the residuals learned by the model using different numbers of modes.

Method	Modes	$\text{minADE}_1 \downarrow$	$\text{minADE}_5 \downarrow$	$\text{minADE}_{10} \downarrow$	$\text{MissRate}_{5,\,2\text{m}} \downarrow$
Constant velocity	N/A	2.33 | 4.11	2.33 | 4.11	2.33 | 4.11	0.77 | 0.90
Physics oracle	N/A	2.26 | **2.97**	2.26 | 2.97	2.26 | 2.97	0.76 | 0.85
MTP [12]	all 1	1.80 | 3.41	1.80 | 3.41	1.80 | 3.41	0.71 | 0.90
MultiPath [6]	all 16	1.81 | 4.30	1.14 | 2.22	1.10 | 2.16	0.56 | 0.89
MultiPath [6]	all 64	1.90 | 4.40	0.98 | **1.94**	0.89 | **1.68**	0.41 | 0.84
MultiPath [6], ε=3	682 | 946	1.77 | 4.70	**0.91** | 2.27	**0.75** | 1.73	**0.32** | 0.74
MultiPath [6], ε=2	1571 | 2339	1.76 | 4.80	0.94 | 2.50	0.76 | 1.85	**0.32** | 0.72
CoverNet [27], ε=3	682 | 946	**1.75** | 4.60	1.05 | 2.22	0.91 | **1.68**	0.38 | 0.78
CoverNet [27], ε=2	1571 | 2339	1.79 | 4.60	1.00 | 2.30	0.82 | 1.70	0.36 | **0.71**
CoverNet [27], ε=1	5077 | 9909	1.94 | 6.10	1.03 | 2.90	0.87 | 2.12	0.41 | 0.80

Table 2: Results listed as Argoverse | nuScenes. The respective prediction horizons are 3 and 6 sec. Smaller minADE_k and $\text{MissRate}_{5,\,2\text{m}}$ is better. CoverNet and MultiPath share trajectory sets. ε is in meters.

5.5.2 Argoverse test set

We further compared our implementation and modifications of MultiPath and CoverNet on the Argoverse test set. Table 3 compares our results with notable published methods. Our multimodal results are similar to MFP [32], which explicitly models interactions. Our poor unimodal performance may be due to the simple input representation we used, which encodes history via fading colors.

Model	CV [7]	LSTM ED [7]	VectorNet [15]	MFP [32]	MultiPath* [6]	CoverNet* [27]
$\text{minADE}_1 \downarrow$	3.55	2.15	1.81	-	2.34	2.38
$\text{minADE}_6 \downarrow$	3.55	2.15	1.18	1.40	1.28	1.42

Table 3: Performance against selected published results on Argoverse test set. CV = constant velocity. MultiPath and CoverNet models (*) use our implementation with a $\varepsilon = 2m$ trajectory set.

6 Conclusion

We extended a state-of-the-art classification-based motion prediction algorithm to utilize domain knowledge. By adding an auxiliary loss that penalizes off-road predictions, the model can better

learn that likely future motions stay on the road. Additionally, our formulation makes it easy to *pretrain* using only map information (e.g., off-road area), which significantly improves performance on small datasets. In an attempt to better encode spatial-temporal relationships among trajectories, we also investigated various weighted cross-entropy losses. Our results here did not improve on the baseline, although pointed towards the need for losses that promote mode diversity. Finally, our detailed analysis of classification and ordinal regression (on public self-driving datasets) showed that best performance can be achieved with an order of magnitude more modes than previously reported.

Acknowledgments. We would like to thank Oscar Beijbom, Caglayan Dicle, Emilio Frazzoli, and Sourabh Vora for helping improve the presentation of our work.

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] 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.

[3] 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.

[4] Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. Spatially-aware graph neural networks for relational behavior forecasting from sensor data, 2019. https://arxiv.org/abs/1910.08233.

[5] 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.

[6] 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.

[7] 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.

[8] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems*, pages 742–751, 2017.

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

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. VectorNet: Encoding HD maps and agent dynamics from vectorized representation, 2020. https://arxiv.org/abs/2005.04259.

[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.

[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.

[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.

[19] 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.

[20] 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.

[21] Kris M. Kitani, Brian D. Ziebart, J. Andrew Bagnell, and Martial Hebert. Activity forecasting. In *The European Conference on Computer Vision (ECCV)*, 2012.
[22] 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.
[23] 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.
[24] 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.
[25] Matthew Niedoba, Henggang Cui, Kevin Luo, Darshan Hegde, Fang-Chieh Chou, and Nemanja Djuric. Improving movement prediction of traffic actors using off-road loss and bias mitigation. In *Advances in Neural Information Processing Systems*, 2019.
[26] nuScenes Contributors. nuScenes. `https://www.nuscenes.org/`, 2020.
[27] Tung Phan-Minh, Elena Corina Grigore, Freddy A. Boulton, Oscar Beijbom, and Eric M. Wolff. CoverNet: Multimodal behavior prediction using trajectory sets, 2019. https://arxiv.org/abs/1911.10298.
[28] 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.
[29] 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.
[30] 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.
[31] 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.
[32] Yichuan Charlie Tang and Ruslan Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems*, 2019.
[33] 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.
[34] 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.

7 Supplementary Material

We created fixed trajectory sets as described in Section 3 of [27]. We used $60,000$ ground truth trajectories from the Argoverse training set and all trajectories from the nuScenes training set to compute trajectory sets. The parameter ε controls the furthest a ground truth trajectory in our training set can be from a trajectory in our trajectory set. We visualize the trajectory sets used by CoverNet and MultiPath on the Argoverse data set and observe that for all ε, the trajectory sets include fast-moving lane keeping trajectories and some very wide turns. As we decrease ε, the diversity of lateral maneuvers increases.

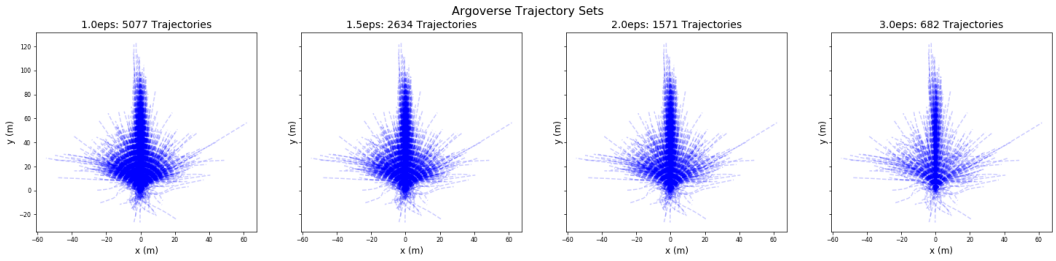


Figure 8: The trajectory sets used by CoverNet and Multipath ($\varepsilon = 2, 3$ m) models for Argoverse.