FIERY: Future Instance Prediction in Bird’s-Eye View from Surround Monocular Cameras

Anthony Hu\textsuperscript{1,2} Zak Murez\textsuperscript{1} Nikhil Mohan\textsuperscript{1} Sofía Dudas\textsuperscript{1} Jeff Hawke\textsuperscript{1} Vijay Badrinarayanan\textsuperscript{1} Roberto Cipolla\textsuperscript{2} Alex Kendall\textsuperscript{1}

\textsuperscript{1}Wayve, UK. \textsuperscript{2}University of Cambridge, UK.

Abstract

Driving requires interacting with road agents and predicting their future behaviour in order to navigate safely. We present FIERY: a probabilistic future prediction model in bird’s-eye view from monocular cameras. Our model predicts future instance segmentation and motion of dynamic agents that can be transformed into non-parametric future trajectories. Our approach combines the perception, sensor fusion and prediction components of a traditional autonomous driving stack by estimating bird’s-eye-view prediction directly from surround RGB monocular camera inputs. FIERY learns to model the inherent stochastic nature of the future directly from camera driving data in an end-to-end manner, without relying on HD maps, and predicts multimodal future trajectories. We show that our model outperforms previous prediction baselines on the NuScenes and Lyft datasets.\textsuperscript{1}

1. Introduction

Prediction of future states is a key challenge in many autonomous decision making systems. This is particularly true for motion planning in highly dynamic environments: for example in autonomous driving where the motion of other road users and pedestrians has a substantial influence on the success of motion planning [10]. Estimating the motion and future poses of these road users enables motion planning algorithms to better resolve multimodal outcomes where the optimal action may be ambiguous knowing only the current state of the world.

Autonomous driving is inherently a geometric problem, where the goal is to navigate a vehicle safely and correctly through 3D space. As such, an orthographic bird’s-eye view (BEV) perspective is commonly used for motion planning and prediction based on LiDAR sensing [38, 49]. Recent advances in camera-based perception have rivalled LiDAR-based perception [48], and we anticipate that this will also be possible for wider monocular vision tasks, including prediction. Building a perception and prediction system based on cameras would enable a leaner, cheaper and higher resolution visual recognition system over LiDAR sensing.

Most of the work in camera-based prediction to date has either been performed directly in the perspective view coordinate frame [1, 23], or using simplified BEV raster representations of the scene [28, 12, 10] generated by HD-mapping systems such as [29, 16]. We wish to build predictive models that operate in an orthographic bird’s-eye view frame (due to the benefits for planning and control [34]), though without relying on auxiliary systems to generate a BEV raster representation of the scene.

A key theme in robust perception systems for autonomous vehicles has been the concept of early sensor fusion, generating 3D object detections directly from image and LiDAR data rather than seeking to merge the predicted outputs of independent object detectors on each sensor input. Learning a task jointly from multiple sources of sensory data as in [50], rather than a staged pipeline, has been demonstrated to offer improvement to perception performance in tasks such as object detection. We seek similar benefits in joining perception and sensor fusion to prediction by estimating bird’s-eye-view prediction directly from surround RGB monocular camera inputs, rather than a multi-stage discrete pipeline of tasks.

Further, traditional autonomous driving stacks [13] tackle future prediction by extrapolating the current behaviour of dynamic agents, without taking into account possible interactions. They rely on HD maps and use road connectivity to generate a set of future trajectories. FIERY learns to predict future motion of road agents directly from camera driving data in an end-to-end manner, without relying on HD maps. It can reason about the probabilistic nature of the future, and predicts multimodal future trajectories (see blog post and Figure 1).

To summarise the main contributions of this paper:
1. We present the first future prediction model in bird’s-eye view from monocular camera videos. Our framework explicitly reasons about multi-agent dynamics by predicting temporally consistent future instance segmentation and motion in bird’s-eye view.

2. Our probabilistic model predicts plausible and multimodal futures of the dynamic environment.

3. We demonstrate quantitative benchmarks for future dynamic scene segmentation, and show that our learned prediction outperforms previous prediction baselines for autonomous driving on the NuScenes [5] and Lyft [25] datasets.
2. Related Work

Bird’s-eye view representation from cameras. Many prior works [51, 47] have tackled the inherently ill-posed problem [17] of lifting 2D perspective images into a bird’s-eye view representation. [35, 33] dealt specifically with the problem of generating semantic BEV maps directly from images and used a simulator to obtain the ground truth.

Recent multi-sensor datasets, such as NuScenes [5] or Lyft [25], made it possible to directly supervise models on real-world data by generating bird’s-eye view semantic segmentation labels from 3D object detections. [39] proposed a Bayesian occupancy network to predict road elements and dynamic agents in BEV directly from monocular RGB images. Most similar to our approach, Lift-Splat [37] learned a dynamic agents in BEV directly from monocular RGB images and used a simulator to obtain the ground truth.


The depth probabilities act as a form of self-attention, modulating the features according to which depth plane they are predicted to belong to. Using the known camera intrinsics and extrinsics (position of the cameras with respect to the center of gravity of the vehicle), these tensors from each camera (u\textsubscript{\text{t,1}},...,u\textsubscript{\text{t,t}}) are lifted to 3D in a common reference frame (the inertial center of the ego-vehicle at time t).

3.2. Projecting to bird’s-eye view

In our experiments, to obtain a bird’s-eye view feature, we discretise the space in 0.50m × 0.50m columns in a 100m × 100m capture size around the ego-vehicle. The 3D features are sum pooled along the vertical dimension to form bird’s-eye view feature maps x\textsubscript{t} ∈ R\textsuperscript{C×H×W}, with (H, W) = (200, 200) the spatial extent of the BEV feature.

3.3. Learning a temporal representation

The past bird’s-eye view features (x\textsubscript{t},...,x\textsubscript{t}) are transformed to the present’s reference frame (time t) using known past ego-motion (a\textsubscript{t-1},...,a\textsubscript{t-1}) a\textsubscript{t-1} ∈ S\textsubscript{E}(3) corresponds to the ego-motion from t – 1 to t, i.e. the translation and rotation of the ego-vehicle. Using a Spatial Transformer [22] module S, we warp past features x\textsubscript{t} to the present for i ∈ {1,...,t – 1}:

x\textsubscript{t,i} = S(x\textsubscript{t},a\textsubscript{t-1} · a\textsubscript{t-2} · ... · a\textsubscript{t-1}) (2)

Since we lose the past ego-motion information with this operation, we concatenate spatially-broadcast actions to the warped past features x\textsubscript{t,i}.

These features are then the input to a temporal model T which outputs a spatio-temporal state s\textsubscript{t}:

s\textsubscript{t} = T(x\textsubscript{t,1},x\textsubscript{t,2},...,x\textsubscript{t,t}) (3)

with x\textsubscript{t,1} = x\textsubscript{t}. T is a 3D convolutional network with local spatio-temporal convolutions, global 3D pooling lay-
1. At each past timestep \( \{1, \ldots, t\} \), we lift camera inputs \((O_1, \ldots, O_t)\) to 3D by predicting a depth probability distribution over pixels and using known camera intrinsics and extrinsics.

2. These features are projected to bird’s-eye view \((x_1, \ldots, x_t)\). Using past ego-motion \((a_1, \ldots, a_{t-1})\), we transform the bird’s-eye view features into the present reference frame (time \(t\)) with a Spatial Transformer module \(S\).

3. A 3D convolutional temporal model learns a spatio-temporal state \(s_t\).

4. We parametrise two probability distributions: the present and the future distribution. The present distribution is conditioned on the current state \(s_t\), and the future distribution is conditioned on both the current state \(s_t\) and future labels \((y_{t+1}, \ldots, y_{t+H})\), with \(H\) the future prediction horizon. The labels correspond to future centerness, offset, segmentation, and flow (see Section 3.6).

5. We sample a latent code \(\eta_t\) from the future distribution during training, and from the present distribution during inference. The current state \(s_t\) and the latent code \(\eta_t\) are the inputs to the future prediction model that recursively predicts future states \((\hat{s}_{t+1}, \ldots, \hat{s}_{t+H})\).

6. The states are decoded into future instance segmentation and future motion in bird’s-eye view \((\hat{y}_{t}, \ldots, \hat{y}_{t+H})\).

### 3.4. Present and future distributions

Following [21] we adopt a conditional variational approach to model the inherent stochasticity of future prediction. We introduce two distributions: a present distribution \(P\) which only has access to the current spatio-temporal state \(s_t\), and a future distribution \(F\) that additionally has access to the observed future labels \((y_{t+1}, \ldots, y_{t+H})\), with \(H\) the future prediction horizon. The labels correspond to future centerness, offset, segmentation, and flow (see Section 3.6).

We parametrise both distributions as diagonal Gaussians with mean \(\mu \in \mathbb{R}^L\) and variance \(\sigma^2 \in \mathbb{R}^L\), \(L\) being the latent dimension. During training, we use samples \(\eta_t \sim \mathcal{N}(\mu_{\text{future}}, \sigma^2_{\text{future}})\) from the future distribution to enforce predictions consistent with the observed future, and a mode covering Kullback-Leibler divergence loss to encourage the present distribution to cover the observed futures:

\[
L_{\text{probabilistic}} = D_{\text{KL}}(F(\cdot|s_t, y_{t+1}, \ldots, y_{t+H}) \parallel P(\cdot|s_t)) \quad (4)
\]

During inference, we sample \(\eta_t \sim \mathcal{N}(\mu_{t,\text{present}}, \sigma^2_{t,\text{present}})\) from the present distribution where each sample encodes a possible future.

### 3.5. Future prediction in bird’s-eye view

The future prediction model is a convolutional gated recurrent unit network taking as input the current state \(s_t\) and the latent code \(\eta_t\) sampled from the future distribution \(F\) during training, or the present distribution \(P\) for inference. It recursively predicts future states \((\hat{s}_{t+1}, \ldots, \hat{s}_{t+H})\).
3.6. Future instance segmentation and motion

The resulting features are the inputs to a bird’s-eye view decoder $D$ which has multiple output heads: semantic segmentation, instance centerness and instance offset (similarly to [9]), and future instance flow. For $j \in \{0, ..., H\}$:

$$\hat{y}_{t+j} = D(s_{t+j})$$

with $s_t = s_{t}$.  

For each future timestep $j$, the instance centerness indicates the probability of finding an instance center (see Figure 3b). By running non-maximum suppression, we get a set of instance centers. The offset is a vector pointing to the center of the instance (Figure 3d), and can be used jointly with the segmentation map (Figure 3c) to assign neighbouring pixels to its nearest instance center and form the bird’s-eye view instance segmentation (Figure 3f). The future flow (Figure 3e) is a displacement vector field of the dynamic agents. It is used to consistently track instances over time by comparing the flow-warped instance centers at time $t+j$ and the detected instance centers at time $t+j+1$ and running a Hungarian matching algorithm [27].

A full description of our model is given in Appendix B.

3.7. Losses

For semantic segmentation, we use a top-$k$ cross-entropy loss [48]. As the bird’s-eye view image is largely dominated by the background, we only backpropagate the top-$k$ hardest pixels. In our experiments, we set $k = 25\%$. The centerness loss is an $L_2$ distance, and both offset and flow losses are $L_1$ distances. We exponentially discount future timesteps with a parameter $\gamma = 0.95$.

4. Experimental Setting

4.1. Dataset

We evaluate our approach on the NuScenes [5] and Lyft [25] datasets. NuScenes contains 1000 scenes, each 20 seconds in length, annotated at 2Hz. The Lyft dataset contains 180 scenes, each 25 – 45 seconds in length, annotated at 5Hz. In both datasets, the camera rig covers the full $360^\circ$ field of view around the ego-vehicle, and is comprised of 6 cameras with a small overlap in field of view. Camera intrinsics and extrinsics are available for each camera in every scene.

The labels $(y_t, ..., y_{t+H})$ are generated by projecting the provided 3D bounding boxes of vehicles into the bird’s-eye view plane to create a bird’s-eye view occupancy grid. See
Appendix B.2 for more detail. All the labels \((y_t, ..., y_{t+H})\) are in the present’s reference frame and are obtained by transforming the labels with the ground truth future ego-motion.

4.2. Metrics

Future Video Panoptic Quality. We want to measure the performance of our system in both:

(i) Recognition quality: how consistently the instances are detected over time.
(ii) Segmentation quality: how accurate the instance segmentations are.

We use the Video Panoptic Quality (VQP) [26] metric defined as:

\[
VPQ = \frac{\sum_{t=0}^{H} (TP_t \cap TP_t')}{\sum_{t=0}^{H} \frac{1}{2} (TP_t + FP_t) + \frac{1}{2} (FN_t)}
\]

where \(TP_t\), the set of true positives at timestep \(t\) (correctly detected ground truth instances), \(FP_t\), the set of false positives at timestep \(t\) (predicted instances that do not match any ground truth instance), and \(FN_t\), the set of false negatives at timestep \(t\) (ground truth instances that were not detected). A true positive corresponds to a predicted instance segmentation that has: (i) an intersection-over-union (IoU) over 0.5 with the ground truth, and (ii) an instance id that is consistent with the ground truth over time (correctly tracked).

Generalised Energy Distance. To measure the ability of our model to predict multi-modal futures, we report the Generalised Energy Distance (\(D_{GED}\)) [44]. Let \(\hat{Y}, \hat{Y}'\) be samples of predicted futures from our model, \((Y, Y')\) be samples of ground truth futures and \(d\) be a distance metric. \(D_{GED}\) is defined as:

\[
D_{GED} = 2\mathbb{E}[d(\hat{Y}, Y)] - \mathbb{E}[d(\hat{Y}, \hat{Y}')] - \mathbb{E}[d(Y, Y')] \tag{7}
\]

We set our distance metric \(d\) to \(d(x, y) = 1 - VPQ(x,y)\). Since we only have access to a unique ground truth future \(Y\), the Generalised Energy Distance simplifies to:

\[
D_{GED} = 2\mathbb{E}[d(\hat{Y}, Y)] - \mathbb{E}[d(\hat{Y}, \hat{Y}')] \tag{8}
\]

4.3. Training

Our model takes 1.0s of past context and predicts 2.0s in the future. In NuScenes, this corresponds to 3 frames of past temporal context and 4 frames into the future at 2Hz. In the Lyft dataset, this corresponds to 6 frames of past context and 10 frames in the future at 5Hz.

For each past timestep, our model processes 6 camera images at resolution 224 x 480. It outputs a sequence of 100m x 100m BEV predictions at 50cm pixel resolution in both the \(x\) and \(y\) directions resulting in a bird’s-eye view video with spatial dimension 200 x 200. We use the Adam optimiser with a constant learning rate of \(3 \times 10^{-4}\). We train our model on 4 Tesla V100 GPUs with a batch size of 12 for 20 epochs at mixed precision.

5. Results

5.1. Comparison to the literature

Since predicting future instance segmentation in bird’s-eye view is a new task, we begin by comparing our model to previous published methods on bird’s-eye view semantic segmentation from monocular cameras.

Many previous works [30, 35, 39, 37, 42] have proposed a model to output the dynamic scene bird’s-eye view segmentation from multiview camera images of a single time-frame. For comparison, we adapt our model so that the past context is reduced to a single observation, and we set the future horizon \(H = 0\) (to only predict the present’s segmentation). We call this model FIERY Static and report the results in Table 1. We observe that FIERY Static outperforms all previous baselines.

<table>
<thead>
<tr>
<th>Method</th>
<th>Intersection-over-Union (IoU)</th>
<th>Setting 1</th>
<th>Setting 2</th>
<th>Setting 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>VED [30]</td>
<td>8.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PON [39]</td>
<td>24.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VPN [35]</td>
<td>25.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STA [42]</td>
<td>36.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Lift-Splat [37]</td>
<td>-</td>
<td>32.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Fishing Camera [19]</td>
<td>-</td>
<td>-</td>
<td>30.0</td>
<td>-</td>
</tr>
<tr>
<td>Fishing Lidar [19]</td>
<td>-</td>
<td>-</td>
<td>44.3</td>
<td>-</td>
</tr>
<tr>
<td>FIERY Static</td>
<td>39.9</td>
<td>36.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>FIERY</strong></td>
<td><strong>41.1</strong></td>
<td><strong>38.2</strong></td>
<td><strong>58.5</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Bird’s-eye view semantic segmentation on NuScenes in the settings of the respective published methods.

Setting 1: 100m x 50m at 25cm resolution. Prediction of the present timeframe.
Setting 2: 100m x 100m at 50cm resolution. Prediction of the present timeframe.
Setting 3: 32.0m x 19.2m at 10cm resolution. Prediction 2.0s in the future. In this last setting we compare our model to two variants of Fishing Net [19]: one using camera inputs, and one using LiDAR inputs.

We also train a model that takes 1.0s of past observations as context (FIERY) and note that it achieves an even higher intersection-over-union over its single-timeframe counterpart that has no past context. We hypothesise this is due to our model’s ability to accumulate information over time and
better handle partial observability and occlusions. Qualitatively, as shown in Figure 4, our predictions are much sharper and more accurate.

Finally, we compare our model to Fishing Net [19], where the authors predict bird’s-eye view semantic segmentation 2.0s in the future. Fishing Net proposes two variants of their model: one using camera as inputs, and one using LiDAR as inputs. FIERY performs much better than both the camera and LiDAR models, hinting that computer vision networks are starting to become competitive with LiDAR sensing for the prediction task.

5.2. Future instance prediction

In order to compare the performance of our model for future instance segmentation and motion prediction, we introduce the following baselines:

**Static model.** The most simple approach to model dynamic obstacles is to assume that they will not move and remain static. We use FIERY Static to predict the instance segmentation of the present timestep (time $t$), and repeat this prediction in the future. We call this baseline the *Static model* as it should correctly detect all static vehicles, since the future labels are in the present’s reference frame.

**Extrapolation model.** Classical prediction methods [14, 15] extrapolate the current behaviour of dynamic agents in the future. We run FIERY Static on every past timesteps to obtain a sequence of past instance segmentations. We re-identify past instances by comparing the instance centers and running a Hungarian matching algorithm. We then obtain past trajectories of detected vehicles, which we extrap-
Table 2: Future instance segmentation in bird’s-eye view for 2.0s in the future on NuScenes. We report future Intersection-over-Union (IoU) and Video Panoptic Quality (VPQ), evaluated at different ranges: 30m × 30m (Short) and 100m × 100m (Long) around the ego-vehicle. Results are reported as percentages.

We also report the results of various ablations of our proposed architecture:

- **No temporal context.** This model only uses the features $x_t$ from the present timestep to predict the future (i.e. we set the 3D convolutional temporal model to the identity function).
- **No transformation.** Past bird’s-eye view features $(x_1, ..., x_t)$ are not warped to the present’s reference frame.
- **No future flow.** This model does not predict future flow.
- **No unrolling.** Instead of recursively predicting the next states $\hat{s}_{t+j}$ and decoding the corresponding instance information $\hat{y}_{t+j} = D(\hat{s}_{t+j})$, this variant directly predicts all future instance centerness, offset, segmentation and flow from $s_t$.
- **Uniform depth.** We lift the features from the encoder $(e_1^t, ..., e_n^t)$ with the Orthographic Feature Transform [40] module. This corresponds to setting the depth probability distribution to a uniform distribution.
- **Deterministic.** We remove the probabilistic modeling.

We report the results in Table 2 (on NuScenes) and Table 3 (on Lyft) of the mean prediction of our probabilistic model (i.e. we set the latent code $\eta_t$ to the mean of the present distribution: $\eta_t = \mu_{t,present}$).

5.3. Analysis

FIERY largely outperforms the Static and Extrapolation baselines for the task of future prediction. Figure 5 shows the performance boost our model gains from different parts of the model.

Figure 5: Performance comparison of various ablations of our model. We measure future Video Panoptic Quality 2.0s in the future on NuScenes.

**Temporal model.** The No temporal context variant performs similarly to the static model. That is to be expected as this model does not have any information from the past, and cannot infer much about the motion of road agents.

**Transformation to the present’s reference frame.** There is a large performance drop when we do not transform past features to the present’s reference frame. This can be explained by how much easier it is for the temporal model to learn correspondences between dynamic vehicles when the ego-motion is factored out.

Past prediction models either naively fed past images to a temporal model [4, 21], or did not use a temporal model altogether and simply concatenated past features [31, 19]. We believe that in order to learn temporal correspondences, past features have to be mapped to a common reference frame and fed into a high capacity temporal model, such as our proposed 3D convolutional architecture.
### Table 3: Future instance prediction in bird’s-eye view for 2.0s in the future on the Lyft dataset. We report future Intersection-over-Union and Video Panoptic Quality.

<table>
<thead>
<tr>
<th></th>
<th>IoU</th>
<th>VPQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static model</td>
<td>35.3</td>
<td>36.4</td>
</tr>
<tr>
<td>Extrapolation model</td>
<td>37.4</td>
<td>37.5</td>
</tr>
<tr>
<td>FIERY</td>
<td>58.1</td>
<td>49.9</td>
</tr>
<tr>
<td>Long</td>
<td>24.1</td>
<td>20.7</td>
</tr>
<tr>
<td></td>
<td>24.8</td>
<td>21.2</td>
</tr>
<tr>
<td></td>
<td>36.6</td>
<td>29.5</td>
</tr>
</tbody>
</table>

Predicting future states. When predicting the future, it is important to model its sequential nature, i.e. the prediction at time \( t + j + 1 \) should be conditioned on the prediction at time \( t + j \).

The No unrolling variant which directly predicts all future instance segmentations and motions from the current state \( s_t \), results in a large performance drop. This is because the sequential constraint is no longer enforced, contrarily to our approach that predicts future states in a recursive way.

Future motion. Learning to predict future motion allows our model to re-identify instances using the predicted flow and comparing instance centers. Our model is the first to produce temporally consistent future instance segmentation in bird’s-eye view of dynamic agents. Without future flow, the predictions are no longer temporally consistent explaining the sharp decrease in performance.

Lifting the features to 3D Using a perfect depth model we could directly lift each pixel to its correct location in 3D space. Since our depth prediction is uncertain, we instead lift the features at different possible depth locations and assign a probability mass at each location, similar to [37]. The Uniform depth baseline uses the Orthographic Feature Transform to lift features in 3D, by setting a uniform distribution on all depth positions. We observe that such a naive lifting performs worse compared to a learned weighting over depth.

Present and future distributions. A deterministic model has a hard task at hand. It has to output with full confidence which future will happen, even though the said future is uncertain. In our probabilistic setting, the model is guided during training with the future distribution that outputs a latent code that indicates the correct future. It also encourages the present distribution to cover the modes of the future distribution. This paradigm allows FIERY to predict both accurate and diverse futures as we will see in section Section 5.4.

Further analyses on understanding the structure of the learned space and on the temporal horizon of future prediction is available in Appendix A.

### 5.4. Probabilistic modelling

We compare our probabilistic future prediction model to the following baselines:

- **M-Head.** The M-head model inspired by [41] outputs \( M \) different futures. During training, the best performing head backpropagates its loss with weight \( (1 - \epsilon) \) while the other heads are weighted by \( \epsilon/(M-1) \). We set \( \epsilon = 0.05 \).
- **Bayesian Dropout.** We insert a dropout layer after every 3D temporal convolution in the temporal model. We also insert a dropout layer in the first 3 layers of the decoder, similarly to [2]. We set the dropout parameter to \( p = 0.25 \).
- **Classical VAE.** We use a Centered Unit Gaussian to constrain our probability distribution similarly to the technique used in [3]. Different latent codes are sampled from \( \mathcal{N}(0, I_L) \) during inference.

We report the results in Table 4.

<table>
<thead>
<tr>
<th></th>
<th>Generalised Energy Distance (( \downarrow ))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Short</td>
</tr>
<tr>
<td>M-Head</td>
<td>96.6</td>
</tr>
<tr>
<td>Bayesian Dropout</td>
<td>92.5</td>
</tr>
<tr>
<td>Classical VAE</td>
<td>93.2</td>
</tr>
<tr>
<td>FIERY</td>
<td>90.5</td>
</tr>
</tbody>
</table>

Table 4: Generalised Energy Distance on NuScenes, for 2.0s future prediction and \( M = 10 \) samples, showing that our model is able to predict the most accurate and diverse futures.

### 6. Conclusion

Autonomous driving requires decision making in multimodal scenarios, where the present state of the world is not always sufficient to reason correctly alone. Predictive models estimating the future state of the world – particularly other dynamic agents – are therefore a key component to robust driving. We presented the first prediction model of dynamic agents for autonomous driving in bird’s-eye view from surround RGB videos. We posed this as an end-to-end learning problem in which our network models future stochasticity with a variational distribution. We demonstrated that FIERY predicts temporally consistent future instance segmentations and motion and is able to model diverse futures accurately. In future work, we would like to jointly train a driving policy to condition the future prediction model on future actions. Such a framework would enable effective motion planning in a model-based reinforcement learning setting.
A. Additional Results

A.1. Visualisation of the learned states

We run a Principal Component Analysis on the states $s_t$ and a Gaussian Mixture algorithm on the projected features in order to obtain clusters. We then visualise the inputs and predictions of the clusters in Figures 6, 8 and 9. We observe that examples in a given cluster correspond to similar scenarios. Therefore, we better understand why our model is able to learn diverse and multimodal futures from a deterministic training dataset. Since similar scenes are mapped to the same state $s_t$, our model will effectively observe different futures starting from the same initial state. The present distribution will thus learn to capture the different modes in the future.

A.2. Temporal horizon of future prediction

Figure 7 shows the performance of our model for different temporal horizon: from 1.0s to 8.0s in the future. The performance seems to plateau beyond 6.0s in the future. In such a large future horizon, the prediction task becomes increasingly difficult as (i) uncertainty in the future grows further in time, and (ii) dynamic agents might not even be visible from past frames.

Figure 7: Future prediction performance for different temporal horizons. We report future Video Panoptic Quality on NuScenes at different capture sizes around the ego-car: $30m \times 30m$ (Short) and $100m \times 100m$ (Long).

![Figure 7: Future prediction performance for different temporal horizons.](image)

(a) Approaching an intersection.

Figure 6: An example of cluster obtained from the spatio-temporal states $s_t$ by running a Gaussian Mixture algorithm on the NuScenes validation set. Our model learns to map similar situations to similar states. Even though the training dataset is deterministic, after mapping the RGB inputs to the state $s_t$, different futures can be observed from the same starting state. This explains why our probabilistic paradigm can learn to predict diverse and plausible futures.
Figure 8: More example of clusters.
Figure 9: More example of clusters.

(a) Stuck in traffic.

(b) Turning right at an intersection.
B. Model and Dataset

B.1. Model description

Our model processes \( t = 3 \) past observations each with \( n = 6 \) cameras images at resolution \( (H_{in}, W_{in}) = (224 \times 480) \), i.e. 18 images. The minimum depth value we consider is \( D_{\text{min}} = 2.0 \text{m} \), which corresponds to the spatial extent of the ego-car. The maximum depth value is \( D_{\text{max}} = 50.0 \text{m} \) and the size of each depth slice was set to \( D_{\text{size}} = 1.0 \text{m} \). We use uncertainty [24] to weight the segmentation, centerness, offset and flow losses. The probabilistic loss is weighted by \( \lambda_{\text{probabilistic}} = 100 \).

Our model contains a total of 8.1M parameters and trains in a day on 4 Tesla V100 GPUs with 32GB of memory. All our layers use batch normalisation and a ReLU activation function.

Bird’s-eye view encoder. For every past timestep, each image in the observation \( O_t = \{ I_1^t, ..., I_n^t \} \) is encoded \( e^t_k = E(I_k^t) \in \mathbb{R}^{(C+D) \times H \times W} \). We use the EfficientNet-B4 [45] backbone with an output stride of 8 in our implementation, so \( (H_e, W_e) = (\frac{H_{in}}{8}, \frac{W_{in}}{8}) = (28, 60) \). The number of channel is \( C = 64 \) and the number of depth slices is \( D = D_{\text{max}} - D_{\text{min}} = 48 \).

These features are then lifted and projected to bird’s-eye view to obtain a tensor \( x_t \in \mathbb{R}^{C \times H \times W} \) with \( (H, W) = (200, 200) \). Using past ego-motion and a spatial transformer module, we transform the bird’s-eye view features to the present’s reference frame.

Temporal model. The 3D convolutional temporal model is composed of Temporal Blocks. Let \( C_{in} \) be the number of input channels and \( C_{out} \) the number of output channels. A single Temporal block is composed of:

- a 3D convolution, with kernel size \( (k_t, k_s, k_s) = (2, 3, 3) \). \( k_t \) is the temporal kernel size, and \( k_s \) the spatial kernel size.
- a 3D convolution with kernel size \( (1, 3, 3) \).
- a 3D global average pooling layer with kernel size \( (2, H, W) \).

Each of these operations are preceded by a feature compression layer, which is a \( (1, 1, 1) \) 3D convolution with output channels \( C_{in} \). All the resulting features are concatenated and fed through a \( (1, 1, 1) \) 3D convolution with output channel \( C_{out} \). The temporal block module also has a skip connection. The final feature \( s_t \in \mathbb{R}^{64 \times 200 \times 200} \).

Present and future distributions. The architecture of the present and future distributions are identical, except for the number of input channels. The present distribution takes as input \( s_t \), and the future distribution takes as input the concatenation of \( (s_t, y_{t+1}, ..., y_{t+H}) \). Let \( C_p = 64 \) be the number of input channel of the present distribution and \( C_f = 64 + C_y \cdot H = 88 \) the number of input channels of the future distribution (since \( C_y = 6 \) and \( H = 4 \)). The module contains four residual block layers [18] each with spatial downsampling 2. These four layers divide the number of input channels by 2. A spatial average pooling layer then decimates the spatial dimension, and a final \( (1,1) \) 2D convolution regresses the mean and log standard deviation of the distribution in \( \mathbb{R}^L \times \mathbb{R}^L \) with \( L = 32 \).

Future prediction. The future prediction module is made of the following structure repeated three times: a convolutional Gated Recurrent Unit [4] followed by 3 residual blocks with kernel size \( (3, 3) \).

Future instance segmentation and motion decoder. The decoder has a shared backbone and multiple output heads to predict centerness, offset, segmentation and flow. The shared backbone contains:

- a 2D convolution with output channel 64 and stride 2.
- the following block repeated three times: four 2D residual convolutions with kernel size \( (3, 3) \). The respective output channels are \( [64, 128, 256] \) and strides \( [1, 2, 2] \).
- three upsampling layers of factor 2, with skip connections and output channel 64.

Each head is then the success two 2D convolutions outputting the required number of channels.

B.2. Labels generation

We compute instance center labels as a 2D Gaussian centered at each instance center of mass with standard deviation \( \sigma_x = \sigma_y = 3 \). The centerness label indicates the likelihood of a pixel to be the center of an instance and is a \( \mathbb{R}^{1 \times H \times W} \) tensor. For all pixels belonging to a given instance, we calculate the offset labels as the vector pointing to the instance center of mass (a \( \mathbb{R}^{2 \times H \times W} \) tensor). Finally, we obtain future flow labels (a \( \mathbb{R}^{2 \times H \times W} \) tensor) by comparing the position of the instance centers of gravity between two consecutive timesteps.

We use the vehicles category to obtain 3D bounding boxes of road agents on both the NuScenes and Lyft datasets.

We report results on the official NuScenes validation split. Since the Lyft dataset does not provide a validation set, we create one by selecting random scenes from the dataset so that it contains roughly the same number of samples (6,174) as NuScenes (6,019).
References


