Model-Based Imitation Learning for Urban Driving

Anonymous Author(s)
Affiliation
Address
email

Abstract

An accurate model of the environment and the dynamic agents acting in it offers great potential for improving motion planning. So far, such world models have been shown to be highly effective at solving games, but only in simple visual environments with little interaction among agents. We present MILE: a Model-based Imitation LEarning approach for autonomous driving that scales to the complexity of urban driving scenes. Our approach leverages 3D geometry as an inductive bias and learns a highly compact latent space directly from high resolution videos of expert demonstrations. MILE learns a model of the world and a driving policy from an offline corpus of driving data, without any online interaction with the environment. Our method improves upon prior state-of-the-art by 35% in driving score on the CARLA simulator when deployed in a completely new town and new weather conditions. Further, we qualitatively show that our model can predict diverse and plausible future scenes in bird's-eye view over a long time horizon (> 60s), that are consistent with predicted ego-actions.

1 Introduction

From an early age we start building neurological representations of the world through observation and interaction [1]. Our ability to estimate scene geometry and dynamics is paramount to generating complex and adaptable movements. This accumulated knowledge of the world, part of what we often refer to as common sense, allows us to navigate effectively in unfamiliar situations [2].

In this work, we present MILE, a Model-based Imitation LEarning approach that leverages offline learned knowledge of the world and its dynamics. We demonstrate the effectiveness of our approach in the autonomous driving domain, operating on complex visual inputs labelled only with expert action and semantic segmentation. Unlike prior work on world models [3, 4, 5], our method does not assume access to a ground truth reward, nor does it need any online interaction with the environment. Further, previous environments in OpenAI Gym [3], MuJoCo [4], and Atari [5] were characterised by simplified visual inputs as small as 64×64 single-channel images. In contrast, MILE learns to compress high-resolution input observations to a compact, probabilistic latent space.

Driving inherently requires a geometric understanding of the environment, and MILE exploits 3D geometry as an important inductive bias by first lifting image features to 3D and pooling them into a bird's-eye view (BeV) representation. The compact latent vector representation resulting from encoding this BeV representation is then used by a driving policy network to control the vehicle, and can additionally be decoded back to BeV segmentation for visualisation and as a supervision signal. The evolution of the world is modelled by a latent dynamics model that leverages the compact state representation of the inputs to predict into the future in this learned latent space. MILE is trained to jointly imitate an expert driver (imitation learning) and to predict the future states (world modelling).

Our method also relaxes the assumption made in some recent work [6, 7] that neither the agent nor its actions influence the environment. This assumption rarely holds in urban driving, and therefore

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

MILE is action-conditioned, allowing us to model how other agents respond to ego-actions. We show that our model can predict plausible and diverse futures from latent states and actions over long time horizons.

We showcase the performance of our model on the driving simulator CARLA [8], and demonstrate a new state of the art. MILE achieves a 21% improvement in driving score with respect to previous methods [9, 10] when deployed on similar conditions as the expert training data (train town, train weathers). More interestingly, it achieves a larger improvement of 35% when tested under harder generalisation conditions (unseen town, new weathers). These results demonstrate that MILE not only outperforms previous approaches but it can also generalise better to unseen conditions.

Finally, during inference, because we model time with a recurrent neural network, we can maintain a single state that summarises all the past observations and then efficiently update the state when a new observation is available. We demonstrate that this design decision has important benefits for deployment in terms of latency, with negligible impact on the driving performance.

To summarise the main contributions of this paper:

- We introduce a novel model-based imitation learning architecture that scales to the visual complexity of autonomous driving in urban environments by leveraging 3D geometry as an inductive bias. Our method is trained solely using an offline corpus of expert driving data, and does not require any interaction with an online environment or access to a reward, offering strong potential for real-world application.
- Our camera-only model sets a new state-of-the-art on the CARLA simulator, surpassing other approaches, including those requiring LiDAR inputs.
- Our model predicts a distribution of diverse and plausible futures that can be decoded into BeV over long time horizons (> 60s) and are consistent with predicted ego-actions.

2 Related Work

Our work is at the intersection of imitation learning, 3D scene representation, and world modelling.

Imitation learning. Despite the fact that the first end-to-end method for autonomous driving was envisioned more than 30 years ago [11], early autonomous driving approaches were dominated by modular frameworks, where each module solves a specific task [12, 13, 14]. Recent years have seen the development of several end-to-end self-driving systems that show strong potential to improve driving performance while predicting driving commands from high-dimensional observations alone. Conditional imitation learning has proven to be one successful method to learn end-to-end driving policies that can be deployed in simulation [15] and real-world urban driving scenarios [16]. Nevertheless, difficulties of learning end-to-end policies from high-dimensional visual observations and expert trajectories alone have been highlighted [17].

Several works have attempted to overcome such difficulties by moving past pure imitation learning. DAgger [18] proposes iterative dataset aggregation to collect data from trajectories that are likely to be experienced by the policy during deployment. NEAT [19] additionally supervises the model with BeV semantic segmentation. ChauffeurNet [20] exposes the learner to synthesised perturbations of the expert data in order to produce more robust driving policies. Learning from All Vehicles (LAV) [10] boosts sample efficiency by learning behaviours from not only the ego vehicle, but from all the vehicles in the scene. Roach [9] presents an agent trained with supervision from a reinforcement learning coach that was trained on-policy and with access to privileged information.

3D scene representation. Successful planning for autonomous driving requires being able to understand and reason about the 3D scene, and this can be challenging from monocular cameras. One common solution is to condense the information from multiple cameras to a single BeV representation of the scene. This can be achieved by lifting each image in 3D (by learning a depth distribution of features) and then splatting all frustums into a common rasterised BeV grid [21, 22, 23]. An alternative approach is to rely on transformers to learn the direct mapping from image to BeV [24, 25, 26], without explicitly modelling depth.

World models. Model-based methods have mostly been explored in a reinforcement learning setting and have been shown to be extremely successful [3, 27, 5]. These methods assume access to a reward, and online interaction with the environment, although progress has been made on fully offline

reinforcement learning [28, 29]. Model-based imitation learning has emerged as an alternative to reinforcement learning in robotic manipulation [30] and OpenAI Gym [31]. Even though these methods do not require access to a reward, they still require online interaction with the environment to achieve good performance.

Learning the latent dynamics of a world model from image observations was first introduced in video prediction [32, 33, 34]. Most similar to our approach, [4, 5] additionally modelled the reward function and optimised a policy inside their world model. Contrarily to prior work, our method does not assume access to a reward function, and directly learns a policy from an offline dataset. Additionally, previous methods operate on simple visual inputs, mostly of size 64×64. In contrast, MILE is able to learn the latent dynamics of complex urban driving scenes from high resolution 600×960 input observations, which is important to ensure small details such as traffic lights can be perceived reliably.

Trajectory forecasting. The goal of trajectory forecasting is to estimate the future trajectories of dynamic agents using past physical states (e.g. position, velocity), and scene context (e.g. as an offline HD map) [35, 36, 37, 38]. World models build a latent representation of the environment that explains the observations from the sensory inputs of the ego-agent (e.g. camera images) conditioned on their actions. While trajectory forecasting methods only model the dynamic scene, world models jointly reason on static and dynamic scenes. The future trajectories of moving agents is implicitly encoded in the learned latent representation of the world model, and could be explicitly decoded given we have access to future trajectory labels.

[35, 37, 38] forecast the future trajectory of moving agents, but did not control the ego-agent. They focused on the prediction problem and not on learning expert behaviour from demonstrations (imitation learning). Rhinehart et al. [39] inferred future trajectories of the ego-agent from expert demonstrations, and conditioned on some specified goal to perform new tasks. Rhinehart et al. [36] extended the work of Rhinehart et al. [39] to jointly model the future trajectories of moving agents as well as of the ego-agent. They studied how the goal of the ego-agent changed the predicted behaviour of the other dynamic agents.

Our proposed world model jointly models the motion of other dynamics agents, the behaviour of the ego-agent, as well as the static scene. Contrary to prior work, we do not assume access to ground truth physical states (position, velocity) or to an offline HD map for scene context. Our approach is the first camera-only method that models static scene, dynamic scene, and ego-behaviour in an urban driving environment.

3 MILE: Model-based Imitation LEarning

In this section, we describe MILE: our method that learns to jointly control an autonomous vehicle and model the world and its dynamics. An overview of the architecture in presented in Figure 1 and the full description of the network can be found in Appendix A.2. We first present MILE by describing the full generative model (Section 3.1), and then derive the inference model (Section 3.2). Section 3.3 and Section 3.4 describe the neural networks that model the latent variables of our inference and generative models respectively. Finally, in Section 3.5 we show how our probabilistic model can be used to imagine the future states of the world for arbitrarily long time horizon.

3.1 Probabilistic model

Let $\mathbf{o}_{1:T}$ be a sequence of T video frames with associated expert actions $\mathbf{a}_{1:T}$ and ground truth BeV semantic segmentation labels $\mathbf{y}_{1:T}$. We model their evolution by introducing latent variables $\mathbf{s}_{1:T}$ that govern the temporal dynamics. The initial distribution is parameterised as $\mathbf{s}_1 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and we additionally introduce a variable $\mathbf{h}_1 \sim \delta(\mathbf{0})$ that serves as a deterministic history. The transition consists of (i) a deterministic update $\mathbf{h}_{t+1} = f_\theta(\mathbf{h}_t, \mathbf{s}_t, \mathbf{a}_t)$ that depends on the past state $\mathbf{s}_t$, past history $\mathbf{h}_t$ and past action $\mathbf{a}_t$, followed by (ii) a stochastic update $\mathbf{s}_{t+1} \sim \mathcal{N}(\mu_\theta(\mathbf{h}_{t+1}), \sigma_\theta(\mathbf{h}_{t+1})\boldsymbol{I})$, where we parameterised $\mathbf{s}_t$ as a normal distribution with diagonal covariance. We model these transitions with neural networks: f_θ is a gated recurrent cell, and $(\mu_\theta, \sigma_\theta)$ are multi-layer perceptrons.

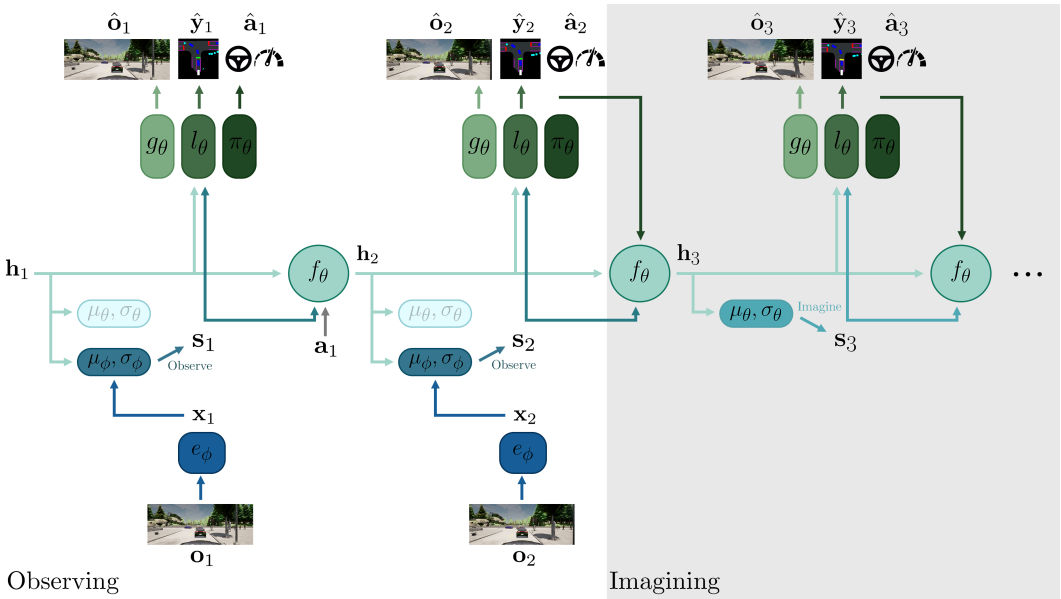


Figure 1: Architecture of MILE.

(i) The goal is to infer the **latent dynamics** $(\mathbf{h}_{1:T}, \mathbf{s}_{1:T})$ that generated the observations $\mathbf{o}_{1:T}$, the expert actions $\mathbf{a}_{1:T}$ and the BeV labels $\mathbf{y}_{1:T}$. The latent dynamics contains a deterministic history $\mathbf{h}_t$ and a stochastic state $\mathbf{s}_t$.

(ii) The **inference model**, with parameters ϕ, estimates the posterior distribution of the stochastic state $q(\mathbf{s}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t}) \sim \mathcal{N}(\mu_\phi(\mathbf{h}_t, \mathbf{x}_t), \sigma_\phi(\mathbf{h}_t, \mathbf{x}_t)\boldsymbol{I})$ with $\mathbf{x}_t = e_\phi(\mathbf{o}_t)$. e_ϕ is the observation encoder that lifts image features to 3D, pools them to BeV, and compresses to a 1D vector.

(iii) The **generative model**, with parameters θ, estimates the prior distribution of the stochastic state $p(\mathbf{s}_t|\mathbf{h}_t) \sim \mathcal{N}(\mu_\theta(\mathbf{h}_t), \sigma_\theta(\mathbf{h}_t)\boldsymbol{I})$, with $\mathbf{h}_t = f_\theta(\mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1})$ the deterministic transition. It additionally estimates the distributions of the observation $p(\mathbf{o}_t|\mathbf{h}_t, \mathbf{s}_t) \sim \mathcal{N}(g_\theta(\mathbf{h}_t, \mathbf{s}_t), \boldsymbol{I})$, the BeV segmentation $p(\mathbf{y}_t|\mathbf{h}_t, \mathbf{s}_t) \sim \text{Categorical}(l_\theta(\mathbf{h}_t, \mathbf{s}_t))$, and the action $p(\mathbf{a}_t|\mathbf{h}_t, \mathbf{s}_t) \sim \text{Laplace}(\pi_\theta(\mathbf{h}_t, \mathbf{s}_t), \mathbf{1})$.

(iv) In the diagram, we represented our model observing inputs for $T = 2$ timesteps, and then imagining future latent states for one step.

The full probabilistic model is given by Equation (1) with its graph represented in Figure 3a.

$$\begin{cases} \mathbf{h}_1 & \sim \delta(\mathbf{0}) \\ \mathbf{s}_1 & \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \\ \mathbf{h}_{t+1} & = f_\theta(\mathbf{h}_t, \mathbf{s}_t, \mathbf{a}_t) \\ \mathbf{s}_{t+1} & \sim \mathcal{N}(\mu_\theta(\mathbf{h}_{t+1}), \sigma_\theta(\mathbf{h}_{t+1})\boldsymbol{I}) \\ \mathbf{o}_t & \sim \mathcal{N}(g_\theta(\mathbf{h}_t, \mathbf{s}_t), \boldsymbol{I}) \\ \mathbf{y}_t & \sim \text{Categorical}(l_\theta(\mathbf{h}_t, \mathbf{s}_t)) \\ \mathbf{a}_t & \sim \text{Laplace}(\pi_\theta(\mathbf{h}_t, \mathbf{s}_t), \mathbf{1}) \end{cases} \tag{1}$$

with δ the Dirac delta function, g_θ the image decoder, l_θ the BeV decoder, and π_θ the policy, which will be described in Section 3.4.

3.2 Variational inference

Following the generative model described in Equation (1), we can factorise the joint probability as:

$$p(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}, \mathbf{h}_{1:T}, \mathbf{s}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{h}_t, \mathbf{s}_t|\mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) p(\mathbf{o}_t, \mathbf{y}_t, \mathbf{a}_t|\mathbf{h}_t, \mathbf{s}_t) \tag{2}$$

with

$$p(\mathbf{h}_t, \mathbf{s}_t | \mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) = p(\mathbf{h}_t | \mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) p(\mathbf{s}_t | \mathbf{h}_t) \tag{3}$$

$$p(\mathbf{o}_t, \mathbf{y}_t, \mathbf{a}_t | \mathbf{h}_t, \mathbf{s}_t) = p(\mathbf{o}_t | \mathbf{h}_t, \mathbf{s}_t) p(\mathbf{y}_t | \mathbf{h}_t, \mathbf{s}_t) p(\mathbf{a}_t | \mathbf{h}_t, \mathbf{s}_t) \tag{4}$$

Given that $\mathbf{h}_{t+1}$ is deterministic according to Equation (1), we have $p(\mathbf{h}_t | \mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) = \delta(\mathbf{h}_t - f_\theta(\mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}))$. Therefore, in order to maximise the marginal likelihood of the observed data $p(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T})$, we need to infer the latent variables $\mathbf{s}_{1:T}$. We do this through deep variational inference by introducing a variational distribution $q_{H,S}$ defined and factorised as follows:

$$q_{H,S} \triangleq q(\mathbf{s}_{1:T}, \mathbf{h}_{1:T} | \mathbf{o}_{1:T}, \mathbf{a}_{1:T-1}) = \prod_{t=1}^{T} q(\mathbf{h}_t | \mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) q(\mathbf{s}_t | \mathbf{o}_{\leq t}, \mathbf{a}_{<t}) \tag{5}$$

with $q(\mathbf{h}_t | \mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) = p(\mathbf{h}_t | \mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1})$, the Delta dirac function defined above, and $q(\mathbf{h}_1) = \delta(\mathbf{0})$. We parameterise this variational distribution with a neural network with weights ϕ, and its inference model is shown in Figure 3b. Hence we can obtain a variational lower bound on the log evidence by applying Jensen's inequality:

$$\begin{aligned}
\log p(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}) \geq\ & \mathcal{L}(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}; \theta, \phi) \\
\triangleq\ & \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{h}_{1:t}, \mathbf{s}_{1:t} | \mathbf{o}_{\leq t}, \mathbf{a}_{<t})} \Bigg[\underbrace{\log p(\mathbf{o}_t | \mathbf{h}_t, \mathbf{s}_t)}_{\text{image reconstruction}} + \underbrace{\log p(\mathbf{y}_t | \mathbf{h}_t, \mathbf{s}_t)}_{\text{BeV segmentation}} + \underbrace{\log p(\mathbf{a}_t | \mathbf{h}_t, \mathbf{s}_t)}_{\text{action}} \Bigg] \\
& - \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{h}_{1:t-1}, \mathbf{s}_{1:t-1} | \mathbf{o}_{\leq t-1}, \mathbf{a}_{<t-1})} \Bigg[\underbrace{D_{\mathrm{KL}}\Big(q(\mathbf{s}_t | \mathbf{o}_{\leq t}, \mathbf{a}_{<t}) \,||\, p(\mathbf{s}_t | \mathbf{h}_t) \Big)}_{\text{posterior and prior matching}} \Bigg]
\end{aligned} \tag{6}$$

Please refer to Appendix A.1 for the full derivation. We model $q(\mathbf{s}_t | \mathbf{o}_{\leq t}, \mathbf{a}_{<t})$ as a Gaussian distribution so that the Kullback-Leibler (KL) divergence can be computed in closed-form. Given the image observations $\mathbf{o}_t$ are modelled as Gaussian distributions with unit variance, the resulting loss is the mean-squared error. Similarly, the action being modelled as a Laplace distribution and the BeV labels as a categorical distribution, the resulting losses are respectively, L_1 and cross-entropy. The expectations over the variational distribution can be efficiently approximated with a single sequence sample from $q_{H,S}$, and backpropagating gradients with the reparametrisation trick [40].

3.3 Inference model ϕ

The inference network, parameterised by ϕ, models $q(\mathbf{s}_t | \mathbf{o}_{\leq t}, \mathbf{a}_{<t})$, which approximates the true (unobserved) posterior $p(\mathbf{s}_t | \mathbf{o}_{\leq t}, \mathbf{a}_{<t})$. It is formed of two elements: the observation encoder e_ϕ, that embeds input images, route map and vehicle control sensor data to a low-dimensional vector, and the posterior network (μ_ϕ, σ_ϕ), that estimates the probability distribution of the Gaussian posterior.

3.3.1 Observation encoder

The state of our model should be compact and low-dimensional in order to effectively learn dynamics. Therefore, we need to embed the high resolution input images to a low-dimensional vector. Naively encoding this image to a 1D vector similarly to an image classification task results in poor performance as shown in Section 5.2. Instead, we explicitly encode 3D geometric inductive biases in the model.

Lifting image features to 3D. Since autonomous driving is a geometric problem where it is necessary to reason on the environment and dynamic agents in 3D, we first lift the image features to 3D. More precisely, we encode the image inputs $\mathbf{o}_t \in \mathbb{R}^{3 \times H \times W}$ with an image encoder to extract features $\mathbf{u}_t \in \mathbb{R}^{C_e \times H_e \times W_e}$. Then similarly to Philion and Fidler [21], we predict a depth probability distribution for each image feature along a predefined grid of depth bins $\mathbf{d}_t \in \mathbb{R}^{D \times H_e, \times W_e}$. Using the depth probability distribution, the camera intrinsics K and extrinsics M, we can lift the image features to 3D: $\text{Lift}(\mathbf{u}_t, \mathbf{d}_t, K^{-1}, M) \in \mathbb{R}^{C_e \times D \times H_e \times W_e \times 3}$.

Pooling to BeV. The 3D feature voxels are then sum-pooled to BeV space using a predefined grid with spatial extent $H_b \times W_b$ and spatial resolution $\mathbf{b}_{\text{res}}$. The resulting feature is $\mathbf{b}_t \in \mathbb{R}^{C_e \times H_b \times W_b}$.

Mapping to a 1D vector. In traditional computer vision tasks (e.g. semantic segmentation [41], depth prediction [42]), the bottleneck feature is usually a spatial tensor, in the order of $10^5 - 10^6$ features. Such high dimensionality is prohibitive for a world model that has to match the distribution of the priors (what it thinks will happen given the executed action) to the posteriors (what actually happened by observing the image input). Therefore, using a convolutional backbone, we compress the BeV feature $\mathbf{b}_t$ to a single vector $\mathbf{x}'_t \in \mathbb{R}^{C'}$, with $C' = 512$. As shown in Section 5.2, we found it critical to compress in BeV space rather than directly in image space.

Route map and speed. We provide the agent with a goal in the form of a route map [9], which is a small grayscale image indicating to the agent where to navigate at intersections. The route map is encoded using a convolutional module resulting in a 1D feature $\mathbf{r}_t$. The current speed is encoded with fully connected layers as $\mathbf{m}_t$. At each timestep t, the embedding $\mathbf{x}_t$ is the concatenation of the image feature, route map feature and speed feature: $\mathbf{x}_t = [\mathbf{x}'_t, \mathbf{r}_t, \mathbf{m}_t] \in \mathbb{R}^C$. Please refer to Appendix A.2 for a full description of the neural networks.

3.3.2 Posterior network

The posterior network (μ_ϕ, σ_ϕ) estimates the parameters of the variational distribution $q(\mathbf{s}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t}) \sim \mathcal{N}\left(\mu_\phi(\mathbf{h}_t, e_\phi(\mathbf{o}_t)), \sigma_\phi(\mathbf{h}_t, e_\phi(\mathbf{o}_t))\right)$ with $\mathbf{h}_t = f_\theta(\mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1})$. Note that $\mathbf{h}_t$ was inferred using f_θ because we have assumed that $\mathbf{h}_t$ is deterministic so $q(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) = p(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) = \delta(\mathbf{h}_t - f_\theta(\mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}))$.

3.4 Generative model θ

The generative network, parameterised by θ, models the latent dynamics $(\mathbf{h}_{1:T}, \mathbf{s}_{1:T})$ as well as the generative process of $(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T})$. It comprises a gated recurrent cell f_θ, a prior network $(\mu_\theta, \sigma_\theta)$, an image decoder g_θ, a BeV decoder l_θ, and a policy π_θ.

The prior network estimates the parameters of the Gaussian distribution $p(\mathbf{s}_t|\mathbf{h}_t) \sim \mathcal{N}\left(\mu_\theta(\mathbf{h}_t), \sigma_\theta(\mathbf{h}_t)\right)$. The image and BeV decoders have an architecture similar to StyleGAN [43]. The prediction starts as a learned constant tensor, and is progressively upsampled to the final resolution. At each resolution, the latent state is injected in the network with adaptive instance normalisation. This allows the latent states to modulate the predictions at different resolutions. The policy is a multi-layer perceptron. Please refer to Appendix A.2 for more details.

3.5 Imagining the future and observation dropout

Our model can imagine future latent states by using the learned policy to infer actions $\hat{\mathbf{a}}_{T+i} = \pi_\theta(\mathbf{h}_{T+i}, \mathbf{s}_{T+i})$, predicting the next deterministic state $\mathbf{h}_{T+i+1} = f_\theta(\mathbf{h}_{T+i}, \mathbf{s}_{T+i}, \hat{\mathbf{a}}_{T+i})$ and sampling from the prior distribution $\mathbf{s}_{T+i+1} \sim \mathcal{N}(\mu_\theta(\mathbf{h}_{T+i+1}), \sigma_\theta(\mathbf{h}_{T+i+1}))$, for $i \geq 0$. This process can be iteratively applied to generate sequences of longer futures in latent space, and the predicted futures can be visualised through the decoders.

At training time the priors are trained to match posteriors through the KL divergence, however they are not necessarily optimised for robust long term future prediction. Hafner et al. [4] optimised states for robust multi-step predictions by iteratively applying the transition model and integrating out intermediate states. In our case, we supervise priors unrolled with random temporal horizons (i.e. predict states at $t + k$ with $k \geq 1$). More precisely, during training, with probability p_{s} we sample the stochastic state $\mathbf{s}_t$ from the prior instead of the posterior. We call this observation dropout. If we denote X the random variable representing the k number of times a prior is unrolled, X follows a geometric distribution with probability of success $(1 - p_{\text{s}})$. Observation dropout resembles z-dropout from Henaff et al. [44], where the posterior distribution is modelled as a mixture of two Gaussians, one of which comes from the prior. During training, some posterior variables are randomly dropped out, forcing other posterior variables to maximise their information extraction from input images. Observation dropout can be seen as a global variant of z-dropout since it drops out all posterior variables together.

Table 1: Driving performance on a new town and new weather conditions in CARLA. Metrics are averaged across three runs. We include reward signals from past work where available.

	Driving Score	**Route**	**Infraction**	**Reward**	**Norm. Reward**
CILRS [17]	3.7 ± 2.2	7.2 ± 3.0	-	-	-
LBC [47]	7.1 ± 2.1	32.1 ± 7.4	-	-	-
TransFuser [48]	33.2 ± 4.0	56.4 ± 7.1	-	-	-
Roach [9]	41.6 ± 1.8	96.4 ± 2.1	43.3 ± 2.8	4236 ± 468	0.34 ± 0.05
LAV [10]	45.2 ± 6.4	91.6 ± 5.6	49.0 ± 6.0	-	-
MILE	$\mathbf{61.1 \pm 3.2}$	$\mathbf{97.4 \pm 0.8}$	$\mathbf{63.0 \pm 3.0}$	$\mathbf{7621 \pm 460}$	$\mathbf{0.67 \pm 0.02}$
Expert	88.4 ± 0.9	97.6 ± 1.2	90.5 ± 1.2	8694 ± 88	0.70 ± 0.01

4 Experimental Setting

Dataset. The training data was collected in the CARLA simulator with an expert reinforcement learning (RL) agent [9] that was trained using privileged information as input (BeV semantic segmentations and vehicle measurements). This RL agent generates more diverse runs and has greater driving performance than CARLA's in-built autopilot [9].

We collect data at 25Hz in four different training towns (Town01, Town03, Town04, Town06) and four weather conditions (ClearNoon, WetNoon, HardRainNoon, ClearSunset) for a total of 2.9M frames, or 32 hours of driving data. At each timestep, we save a tuple $(\mathbf{o}_t, \mathbf{route}_t, \mathbf{speed}_t, \mathbf{a}_t, \mathbf{y}_t)$, with $\mathbf{o}_t \in \mathbb{R}^{3\times 600\times 960}$ the forward camera RGB image, $\mathbf{route}_t \in \mathbb{R}^{1\times 64\times 64}$ the route map (visualized as an inset on the top right of the RGB images in Figure 2), $\mathbf{speed}_t \in \mathbb{R}$ the current velocity of the vehicle, $\mathbf{a}_t \in \mathbb{R}^2$ the action executed by the expert (acceleration and steering), and $\mathbf{y}_t \in \mathbb{R}^{C_b\times 192\times 192}$ the BeV semantic segmentation. There are $C_b = 8$ semantic classes: background, road, line markings, vehicles, pedestrians, and traffic light states (red, yellow, green). In urban driving environments, the dynamics of the scene do not contain high frequency components, which allows us to subsample frames at 5Hz in our sequence model.

Training. Our model was trained for $50,000$ iterations on a batch size of 64 on 8 V100 GPUs, with training sequence length $T = 12$. We use the AdamW optimiser with learning rate 10^{-4} and weight decay 0.01.

Metrics. We report metrics from the CARLA challenge [45] to measure on-road performance: driving score, route completion, and infraction penalty. These metrics are however very coarse, as they only give a sense of how well the agent performs with hard penalties (such as hitting virtual pedestrians). Core driving competencies such as lane keeping and driving at an appropriate speed are obscured. Therefore we also report the cumulative reward of the agent. At each timestep the reward [46] penalises the agent for deviating from the lane centre, for driving too slowly/fast, or for causing infractions. It measures how well the agent drives at the timestep level. In order to account for the length of the simulation (due to various stochastic events, it can be longer or shorter), we also report the normalised cumulative reward. Please refer to Appendix A.3 for more details.

5 Results

5.1 Driving performance

We evaluate our model inside the CARLA simulator on a town and weather conditions never seen during training. We picked Town05 as it is the most complex testing town, and use the 10 routes of Town05 as specified in the CARLA challenge [45], in four different weather conditions (see Appendix A.3). Table 1 shows the comparison against prior state-of-the-art methods. MILE outperforms previous works on all metrics, with a 35% relative improvement in driving score with respect to LAV. Even though some methods have access to additional sensor information such as LiDAR (TransFuser [48], LAV [10]), our approach demonstates superior performance while only using RGB images from the front camera. Moreover, we observe that our method almost doubles the cumulative reward of Roach (which was trained on the same dataset) and approaches the performance of the privileged expert. For MILE, Roach and Expert we report all the metrics defined in Section 4. For all the other methods we report the results from the original papers.

Table 2: Ablation studies. We report driving performance on a new town and new weather conditions in CARLA. Results are averaged across three runs.

	Driving Score	**Route**	**Infraction**	**Reward**	**Norm. Reward**
Single frame	47.9 ± 2.7	73.7 ± 2.4	64.8 ± 1.7	1259 ± 1057	0.14 ± 0.11
Single frame, seg, no 3D	51.8 ± 3.0	78.3 ± 3.0	68.3 ± 2.8	1878 ± 296	0.20 ± 0.04
Single frame, seg	59.6 ± 3.6	94.5 ± 0.6	64.7 ± 3.3	6630 ± 168	0.60 ± 0.01
Deterministic temporal	$\mathbf{63.3 \pm 2.2}$	91.5 ± 5.0	68.7 ± 1.8	6084 ± 1429	0.55 ± 0.07
No 3D	62.0 ± 0.0	87.2 ± 0.0	$\mathbf{71.9 \pm 0.0}$	4522 ± 0.0	0.42 ± 0.0
No segmentation	53.6 ± 0.0	90.0 ± 0.0	60.7 ± 0.0	7085 ± 0.0	0.66 ± 0.0
No obs. dropout	61.1 ± 0.0	89.9 ± 0.0	68.9 ± 0.0	5429 ± 0.0	0.57 ± 0.0
MILE	61.1 ± 3.2	$\mathbf{97.4 \pm 0.8}$	63.0 ± 3.0	$\mathbf{7621 \pm 460}$	$\mathbf{0.67 \pm 0.02}$
Expert	88.4 ± 0.9	97.6 ± 1.2	90.5 ± 1.2	8694 ± 88	0.70 ± 0.01

We also evaluate our method on towns and weather conditions seen during training. As reported in Appendix B, MILE again demonstrates state-of-the-art performance with a 21% relative improvement in driving score with respect to Roach. With these results we demonstrate that MILE outperforms previous approaches on two settings: deployment in familiar conditions (train town, train weathers) and generalisation to unfamiliar conditions (unseen town, new weathers).

5.2 Ablation studies

We next examine the effect of various design decisions in our approach.

Single frame. We remove the world model and directly predict the action from the current image observation. As shown in Table 2, this results in a drastic decrease in cumulative reward. However, supervising this single frame with BeV semantic segmentation ('Single frame, seg') results in a large performance gain: from 1259 to 6630. Importantly, we observe that if we remove the 3D lifting and BeV projection step ('Single frame, seg, no 3D'), the performance gain is much more negligible (only 1259 -> 1878). This highlights the importance of the 3D geometry inductive bias to learn the latent state.

Probabilistic model. At any given time while driving, there exist multiple possible valid behaviours. For example, the driver can slightly adjust its speed, decide to change lane, or decide what is a safe distance to follow behind a vehicle. A deterministic driving policy cannot model these subtleties. In ambiguous situations where multiple choices are possible, it will often learn the mean behaviour, which is valid in certain situations (e.g. the mean safety distance and mean cruising speed are reasonable choices), but unsafe in others (e.g. in lane changing: the expert can change lane early, or late; the mean behaviour is to drive on the line marking). We compare MILE with a deterministic temporal model that simply aggregates past context over time with a recurrent network. We see in Table 2 that MILE results in the highest cumulative reward.

5.3 Low-latency inference

In Appendix B, we compare the closed-loop performance of our model with two different strategies:

(i) **Reset state**: for every new observation, we re-initialise the latent state and recompute the new state $[h_T, s_T]$, with T matching the training sequence length.

(ii) **Fully recurrent**: the latent state is initialised at the beginning of the evaluation, and is recurrently updated with new observations. It is never reset, and instead, the model must have learned a representation that generalises to integrating information for orders of magnitude more steps than the T used during training.

Table 3 shows that our model can be deployed with recurrent updates, matching the performance of the *Reset state* approach, while being much more computationally efficient ($7\times$ faster from 6.2fps with $T = 12$ of fixed context to 43.0fps with a fully recurrent approach). A hypothesis that could explain why the *Fully recurrent* deployment method works well is because the world model has learned to always discard all past information and rely solely on the present input. To test this hypothesis, we add Gaussian noise to the past latent state during deployment. If the recurrent network is simply discarding all past information, its performance shouldn't be affected. However, we see

that the cumulative reward drops dramatically, showing our model does not simply discard all past context, but actively makes use of it.

Table 3: Comparison of two deployment methods: (i) *Reset state*: for each new observation a fresh state is computed from a zero-initialised latent state using T observations, and (ii) *Fully recurrent*: the latent state is recurrently updated with new observations. We report driving performance on an unseen town and unseen weather conditions in CARLA.

	Driving Score	**Route**	**Infraction**	**Reward**	**Norm. Reward**	**Freq.**
Reset state	61.1 ± 3.2	$\mathbf{97.4 \pm 0.8}$	63.0 ± 3.0	$\mathbf{7621 \pm 460}$	$\mathbf{0.67 \pm 0.02}$	6.2
Fully recurrent	$\mathbf{62.1 \pm 0.5}$	93.5 ± 4.8	$\mathbf{66.6 \pm 3.4}$	7532 ± 1122	0.67 ± 0.04	**43.0**
Recurrent+noise	48.8 ± 1.8	81.1 ± 7.0	61.5 ± 6.4	3603 ± 780	0.35 ± 0.07	**43.0**

5.4 Long horizon, diverse future predictions

Our model can imagine diverse futures in the latent space, which can be decoded to BeV semantic segmentation for interpretability. Figure 2 shows examples of multi-modal futures predicted by MILE. Appendix B.5 contains more qualitative examples: prediction 60 seconds in the future, as well as interpolating two vectors in the latent space to obtain smooth transitions between the two scenarios. Please refer to the supplementary material to see videos of predicted futures.

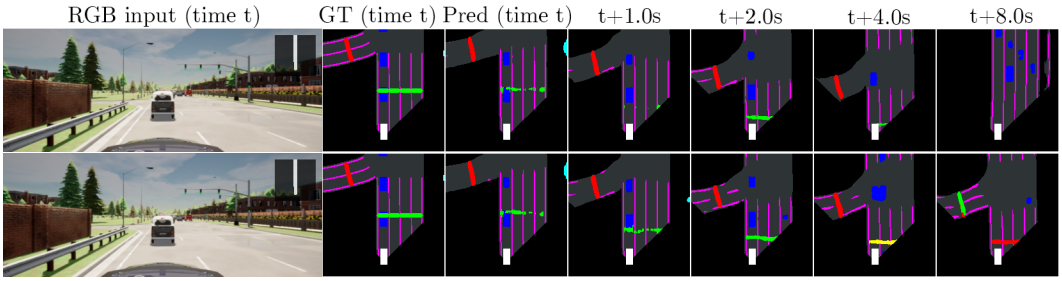


Figure 2: Qualitative examples of multi-modal predictions, for 8 seconds in the future. BeV segmentation legend: white = ego-vehicle, black = background, gray = road, purple = line marking, blue = vehicles, cyan = pedestrians, green/yellow/red = traffic lights. Ground truth labels (GT) outside the field-of-view of the front camera are masked out. In this example, we visualise two distinct futures predicted by the model: 1) (top row) driving through the green light, 2) (bottom row) stopping because the model imagines the traffic light turning red. Note the transition from green, to yellow, to red, and also at the last frame $t + 8.0\text{s}$ how the traffic light in the left lane turns green.

6 Conclusion

We presented MILE: a Model-based Imitation LEarning approach for urban driving, that jointly learns a driving policy and a world model from offline expert demonstrations alone. MILE exploits geometric inductive biases and handles high-dimensional visual input. We demonstrated state-of-the-art performance on two settings that are highly representative of the complexity of deploying driving policies in the real world: familiar towns and weather conditions, and generalisation to new towns and weather conditions. MILE's internal world model is able to predict stable, long-horizon futures. By exploiting the recurrent ability of our model, we showcased MILE can be efficiently deployed by recurrently updating the state with incoming image observations, resulting in low-latency inference.

An open problem is how to infer the driving reward function from expert trajectories with inverse reinforcement learning, as this would enable planning in the learned world model. Another exciting avenue is self-supervision in order to relax the dependency on the BeV segmentation labels. Although such labels can be obtained for real-world data, they are expensive and subject to noise and inaccuracy. Full self-supervision, such as depth or scene flow, could fully unlock the potential of world models for real-world driving and other robotics tasks.

References

[1] H. B. Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

[2] D. M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural networks*, 11(7-8):1317–1329, 1998.

[3] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[4] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

[5] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering Atari with discrete world models. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[6] D. Chen, V. Koltun, and P. Krähenbühl. Learning to drive from a world on rails. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15590–15599, 2021.

[7] V. Sobal, A. Canziani, N. Carion, K. Cho, and Y. LeCun. Separating the world and ego models for self-driving. *Proceedings of the International Conference on Learning Representations, workshop (ICLRw)*, 2022.

[8] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 1–16, 2017.

[9] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15222–15232, 2021.

[10] D. Chen and P. Krähenbühl. Learning from all vehicles. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[11] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems (NeurIPS)*, 1988.

[12] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, et al. Odin: Team VictorTango's entry in the DARPA Urban Challenge. *Journal of field Robotics*, 25(8):467–492, 2008.

[13] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Practical search techniques in path planning for autonomous driving. *AAAI Workshop - Technical Report*, 01 2008.

[14] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008.

[15] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 4693–4700, 2018.

[16] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklethwaite, N. Griffiths, A. Shah, et al. Urban driving with conditional imitation learning. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 251–257, 2020.

[17] F. Codevilla, E. Santana, A. M. López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9329–9338, 2019.

[18] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

[19] K. Chitta, A. Prakash, and A. Geiger. NEAT: Neural attention fields for end-to-end autonomous driving. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[20] M. Bansal, A. Krizhevsky, and A. Ogale. ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst. In *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

[21] J. Philion and S. Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 194–210, 2020.

[22] A. Saha, O. Mendez, C. Russell, and R. Bowden. Enabling spatio-temporal aggregation in birds-eye-view vehicle estimation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2021.

[23] A. Hu, Z. Murez, N. Mohan, S. Dudas, J. Hawke, V. Badrinarayanan, R. Cipolla, and A. Kendall. FIERY: Future instance prediction in bird's-eye view from surround monocular cameras. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 15273–15282, 2021.

[24] L. Peng, Z. Chen, Z. Fu, P. Liang, and E. Cheng. BEVSegFormer: Bird's eye view semantic segmentation from arbitrary camera rigs. *arXiv preprint arXiv:2203.04050*, 2022.

[25] N. Gosala and A. Valada. Bird's-eye-view panoptic segmentation using monocular frontal view images. *IEEE Robotics and Automation Letters*, 2022.

[26] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Q. Yu, and J. Dai. BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[27] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 2020.

[28] W. Zhou, S. Bajracharya, and D. Held. PLAS: Latent action space for offline reinforcement learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2020.

[29] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[30] P. Englert, A. Paraschos, M. P. Deisenroth, and J. Peters. Probabilistic model-based imitation learning. *Adaptive Behavior*, 21(5):388–403, 2013.

[31] R. Kidambi, J. Chang, and W. Sun. MobILE: Model-based imitation learning from observation alone. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

[32] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine. Stochastic variational video prediction. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[33] E. Denton and R. Fergus. Stochastic video generation with a learned prior. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[34] J.-Y. Franceschi, E. Delasalles, M. Chen, S. Lamprier, and P. Gallinari. Stochastic latent residual video prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[35] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker. DESIRE: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[36] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. PRECOG: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[37] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, C. Li, and D. Anguelov. TNT: Target-driven trajectory prediction. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2020.

[38] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[39] N. Rhinehart, R. McAllister, and S. Levine. Deep imitative models for flexible inference, planning, and control. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[40] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[41] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[42] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow. Digging into self-supervised monocular depth prediction. *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[43] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[44] M. Henaff, A. Canziani, and Y. LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[45] CARLA Autonomous Driving Leaderboard. https://leaderboard.carla.org/get_started/, 2019.

[46] M. Toromanoff, E. Wirbel, and F. Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[47] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 66–75, 2020.

[48] A. Prakash, K. Chitta, and A. Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7077–7087, 2021.

[49] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L. Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[50] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[51] CARLA Maps. https://carla.readthedocs.io/en/latest/core_map/, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Table 1, Section 5.4 and Table 2.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.1.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Table 1 and Table 2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Appendix A.2.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Technical section

A.1 Lower bound derivation

Proof. Let $q_{H,S} \triangleq q(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}|\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}) = q(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T-1})$ be the variational distribution (where we have assumed independence of $(\mathbf{y}_{1:T}, \mathbf{a}_T)$ given $(\mathbf{o}_{1:T}, \mathbf{a}_{1:T-1})$), and $p(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}|\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T})$ be the posterior distribution. The Kullback-Leibler divergence between these two distributions writes as:

$$
\begin{aligned}
&D_{\mathrm{KL}}\left(q(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}|\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}) \,||\, p(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}|\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T})\right) \\
&= \mathbb{E}_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T} \sim q_{H,S}} \left[\log \frac{q(\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}|\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T})}{p(\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}|\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T})}\right] \\
&= \mathbb{E}_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T} \sim q_{H,S}} \left[\log \frac{q(\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}|\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}) p(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T})}{p(\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}) p(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}|\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T})}\right] \\
&= \log p(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}) - \mathbb{E}_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T} \sim q_{H,S}} \left[\log p(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}|\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T})\right] \\
&\quad + D_{\mathrm{KL}}(q(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}|\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}) \,||\, p(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}))
\end{aligned}
$$

Since $D_{\mathrm{KL}}\left(q(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}|\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}) \,||\, p(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}|\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T})\right) \geq 0$, we obtain the following evidence lower bound:

$$
\begin{aligned}
\log p(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}) \geq &\mathbb{E}_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T} \sim q_{H,S}} \left[\log p(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}|\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T})\right] \\
&- D_{\mathrm{KL}}(q(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T-1}) \,||\, p(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}))
\end{aligned} \tag{7}
$$

Let us now calculate the two terms of this lower bound separately. On the one hand:

$$
\begin{aligned}
&\mathbb{E}_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T} \sim q_{H,S}} \left[\log p(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}|\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T})\right] \\
&= \mathbb{E}_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T} \sim q_{H,S}} \left[\log \prod_{t=1}^{T} p(\mathbf{o}_t|\widetilde{\mathbf{h}}_t, \widetilde{\mathbf{s}}_t) p(\mathbf{y}_t|\widetilde{\mathbf{h}}_t, \widetilde{\mathbf{s}}_t) p(\mathbf{a}_t|\widetilde{\mathbf{h}}_t, \widetilde{\mathbf{s}}_t)\right] && (8) \\
&= \sum_{t=1}^{T} \mathbb{E}_{\widetilde{\mathbf{h}}_{1:t}, \widetilde{\mathbf{s}}_{1:t} \sim q(\widetilde{\mathbf{h}}_{1:t}, \widetilde{\mathbf{s}}_{1:t}|o_{\leq t}, a_{<t})} \left[\log p(\mathbf{o}_t|\widetilde{\mathbf{h}}_t, \widetilde{\mathbf{s}}_t) + \log p(\mathbf{y}_t|\widetilde{\mathbf{h}}_t, \widetilde{\mathbf{s}}_t) + \log p(\mathbf{a}_t|\widetilde{\mathbf{h}}_t, \widetilde{\mathbf{s}}_t)\right] && (9)
\end{aligned}
$$

where Equation (8) follows from Equation (2), and Equation (9) was obtained by integrating over remaining latent variables $(\widetilde{\mathbf{h}}_{t+1:T}, \widetilde{\mathbf{s}}_{t+1:T})$.

On the other hand:

$$
\begin{aligned}
&D_{\mathrm{KL}}(q(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T-1}) \,||\, p(\mathbf{h}_{1:T}, \mathbf{s}_{1:T})) \\
&= \mathbb{E}_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T} \sim q_{H,S}} \left[\log \frac{q(\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T-1})}{p(\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T})}\right] \\
&= \int_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}} q(\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T-1}) \log \frac{q(\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T-1})}{p(\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T})} \,\mathrm{d}\widetilde{\mathbf{h}}_{1:T} \,\mathrm{d}\widetilde{\mathbf{s}}_{1:T} \\
&= \int_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}} q(\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T-1}) \log \left[\prod_{t=1}^{T} \frac{q(\widetilde{\mathbf{h}}_t|\widetilde{\mathbf{h}}_{t-1}, \widetilde{\mathbf{s}}_{t-1}, \mathbf{a}_{t-1}) q(\widetilde{\mathbf{s}}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t})}{p(\widetilde{\mathbf{h}}_t|\widetilde{\mathbf{h}}_{t-1}, \widetilde{\mathbf{s}}_{t-1}, \mathbf{a}_{t-1}) p(\widetilde{\mathbf{s}}_t|\widetilde{\mathbf{h}}_t)}\right] \,\mathrm{d}\widetilde{\mathbf{h}}_{1:T} \,\mathrm{d}\widetilde{\mathbf{s}}_{1:T} && (10) \\
&= \int_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}} q(\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T-1}) \log \left[\prod_{t=1}^{T} \frac{q(\widetilde{\mathbf{s}}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t})}{p(\widetilde{\mathbf{s}}_t|\widetilde{\mathbf{h}}_t)}\right] \,\mathrm{d}\widetilde{\mathbf{h}}_{1:T} \,\mathrm{d}\widetilde{\mathbf{s}}_{1:T} && (11) \\
& && (12)
\end{aligned}
$$

where:

- Equation (10) follows from the factorisations defined in Equation (2) and Equation (5).
- The simplification in Equation (11) results of $q(\widetilde{\mathbf{h}}_t|\widetilde{\mathbf{h}}_{t-1}, \widetilde{\mathbf{s}}_{t-1}, \mathbf{a}_{t-1}) = p(\widetilde{\mathbf{h}}_t|\widetilde{\mathbf{h}}_{t-1}, \widetilde{\mathbf{s}}_{t-1}, \mathbf{a}_{t-1})$.

Thus:

$$
\begin{aligned}
& D_{\mathrm{KL}}(q(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T-1}) \,||\, p(\mathbf{h}_{1:T}, \mathbf{s}_{1:T})) \\
&= \int_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}} \left(\prod_{t=1}^{T} q(\widetilde{\mathbf{h}}_t|\widetilde{\mathbf{h}}_{t-1}, \widetilde{\mathbf{s}}_{t-1}, \mathbf{a}_{t-1}) q(\widetilde{\mathbf{s}}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t}) \right) \left(\sum_{t=1}^{T} \log \frac{q(\widetilde{\mathbf{s}}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t})}{p(\widetilde{\mathbf{s}}_t|\widetilde{\mathbf{h}}_t)} \right) \mathrm{d}\widetilde{\mathbf{h}}_{1:T} \, \mathrm{d}\widetilde{\mathbf{s}}_{1:T} \\
&= \int_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}} \left(\prod_{t=1}^{T} q(\widetilde{\mathbf{h}}_t|\widetilde{\mathbf{h}}_{t-1}, \widetilde{\mathbf{s}}_{t-1}, \mathbf{a}_{t-1}) q(\widetilde{\mathbf{s}}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t}) \right) \left(\log \frac{q(\widetilde{\mathbf{s}}_1|\mathbf{o}_1)}{p(\widetilde{\mathbf{s}}_1)} + \sum_{t=2}^{T} \log \frac{q(\widetilde{\mathbf{s}}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t})}{p(\widetilde{\mathbf{s}}_t|\widetilde{\mathbf{h}}_t)} \right) \mathrm{d}\widetilde{\mathbf{h}}_{1:T} \, \mathrm{d}\widetilde{\mathbf{s}}_{1:T} \\
&= \mathbb{E}_{\widetilde{\mathbf{s}}_1 \sim q(\widetilde{\mathbf{s}}_1|\mathbf{o}_1)} \left[\log \frac{q(\widetilde{\mathbf{s}}_1|\mathbf{o}_1)}{p(\widetilde{\mathbf{s}}_1)} \right] \\
&\quad + \int_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}} \left(\prod_{t=1}^{T} q(\widetilde{\mathbf{h}}_t|\widetilde{\mathbf{h}}_{t-1}, \widetilde{\mathbf{s}}_{t-1}, \mathbf{a}_{t-1}) q(\widetilde{\mathbf{s}}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t}) \right) \left(\sum_{t=2}^{T} \log \frac{q(\widetilde{\mathbf{s}}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t})}{p(\widetilde{\mathbf{s}}_t|\widetilde{\mathbf{h}}_t)} \right) \mathrm{d}\widetilde{\mathbf{h}}_{1:T} \, \mathrm{d}\widetilde{\mathbf{s}}_{1:T} \qquad (13) \\
&= D_{\mathrm{KL}}(q(\mathbf{s}_1|\mathbf{o}_1) \,||\, p(\mathbf{s}_1)) \\
&\quad + \int_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}} \left(\prod_{t=1}^{T} q(\widetilde{\mathbf{h}}_t|\widetilde{\mathbf{h}}_{t-1}, \widetilde{\mathbf{s}}_{t-1}, \mathbf{a}_{t-1}) q(\widetilde{\mathbf{s}}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t}) \right) \left(\log \frac{q(\widetilde{\mathbf{s}}_2|\mathbf{o}_{1:2}, \mathbf{a}_1)}{p(\widetilde{\mathbf{s}}_2|\widetilde{\mathbf{h}}_2)} + \sum_{t=3}^{T} \log \frac{q(\widetilde{\mathbf{s}}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t})}{p(\widetilde{\mathbf{s}}_t|\widetilde{\mathbf{h}}_t)} \right) \mathrm{d}\widetilde{\mathbf{h}}_{1:T} \, \mathrm{d}\widetilde{\mathbf{s}}_{1:T} \\
&= D_{\mathrm{KL}}(q(\mathbf{s}_1|\mathbf{o}_1) \,||\, p(\mathbf{s}_1)) + \mathbb{E}_{\widetilde{\mathbf{h}}_1, \widetilde{\mathbf{s}}_1 \sim q(\widetilde{\mathbf{h}}_1, \widetilde{\mathbf{s}}_1|\mathbf{o}_1)} \left[D_{\mathrm{KL}}(q(\mathbf{s}_2|\mathbf{o}_{1:2}, \mathbf{a}_1) \,||\, p(\mathbf{s}_2|\widetilde{\mathbf{h}}_2)) \right] \\
&\quad + \int_{\widetilde{\mathbf{h}}_{1:T}, \widetilde{\mathbf{s}}_{1:T}} \left(\prod_{t=1}^{T} q(\widetilde{\mathbf{h}}_t|\widetilde{\mathbf{h}}_{t-1}, \widetilde{\mathbf{s}}_{t-1}, \mathbf{a}_{t-1}) q(\widetilde{\mathbf{s}}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t}) \right) \left(\sum_{t=3}^{T} \log \frac{q(\widetilde{\mathbf{s}}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t})}{p(\widetilde{\mathbf{s}}_t|\widetilde{\mathbf{h}}_t)} \right) \mathrm{d}\widetilde{\mathbf{h}}_{1:T} \, \mathrm{d}\widetilde{\mathbf{s}}_{1:T} \qquad (14)
\end{aligned}
$$

where Equation (13) and Equation (14) were obtained by splitting the integral in two and integrating over remaining latent variables. By recursively applying this process on the sum of logarithms indexed by t, we get:

$$
\begin{aligned}
& D_{\mathrm{KL}}(q(\mathbf{h}_{1:T}, \mathbf{s}_{1:T}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T-1}) \,||\, p(\mathbf{h}_{1:T}, \mathbf{s}_{1:T})) \\
&= \sum_{t=1}^{T} \mathbb{E}_{\widetilde{\mathbf{h}}_{1:t-1}, \widetilde{\mathbf{s}}_{1:t-1} \sim q(\widetilde{\mathbf{h}}_{1:t-1}, \widetilde{\mathbf{s}}_{1:t-1}|\mathbf{o}_{<t}, \mathbf{a}_{<t-1})} \left[D_{\mathrm{KL}}(q(\mathbf{s}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t}) \,||\, p(\mathbf{s}_t|\widetilde{\mathbf{h}}_t)) \right] \qquad (15)
\end{aligned}
$$

Finally, we inject Equation (9) and Equation (15) in Equation (7) to obtain the desired lower bound:

$$
\begin{aligned}
& \log p(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}, \mathbf{a}_{1:T}) \\
&\geq \sum_{t=1}^{T} \mathbb{E}_{\widetilde{\mathbf{h}}_{1:t}, \widetilde{\mathbf{s}}_{1:t} \sim q(\widetilde{\mathbf{h}}_{1:t}, \widetilde{\mathbf{s}}_{1:t}|\mathbf{o}_{\leq}, \mathbf{a}_{<t})} \left[\log p(\mathbf{o}_t|\widetilde{\mathbf{h}}_t, \widetilde{\mathbf{s}}_t) + \log p(\mathbf{y}_t|\widetilde{\mathbf{h}}_t, \widetilde{\mathbf{s}}_t) + \log p(\mathbf{a}_t|\widetilde{\mathbf{h}}_t, \widetilde{\mathbf{s}}_t) \right] \\
&\quad - \sum_{t=1}^{T} \mathbb{E}_{\widetilde{\mathbf{h}}_{1:t-1}, \widetilde{\mathbf{s}}_{1:t-1} \sim q(\widetilde{\mathbf{h}}_{1:t-1}, \widetilde{\mathbf{s}}_{1:t-1}|\mathbf{o}_{<t}, \mathbf{a}_{<t-1})} \left[D_{\mathrm{KL}}(q(\mathbf{s}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t}) \,||\, p(\mathbf{s}_t|\widetilde{\mathbf{h}}_t)) \right] \qquad \square
\end{aligned}
$$

A.2 Model description

We give a full description of our model. Table A.2.1 shows the number of parameters of each component, and Table A.2.2 contains all the hyperparameters used during training. Table A.2.3 describes the inference model, and Table A.2.4 the generative model. Figure 3b represents the generative and inference graphical models.

Table A.2.1: Parameters of the model.

	Name	Parameters
Inference model ϕ	Observation encoder e_ϕ	34.9M
	Posterior network (μ_ϕ, σ_ϕ)	3.9M
Generative model θ	Prior network $(\mu_\theta, \sigma_\theta)$	2.1M
	Recurrent cell f_θ	6.9M
	BeV decoder l_θ	34.2M
	Policy π_θ	5.9M

Table A.2.2: Hyperparameters.

Category	Name	Value
Training	GPUs	8 Tesla V100
	batch size	64
	precision	Mixed precision (16-bit)
	iterations	5×10^4
Optimiser	name	AdamW
	learning rate	1×10^{-4}
	weight decay	1×10^{-2}
	β_1	0.9
	β_2	0.999
	ε	1×10^{-8}
	scheduler	OneCycleLR
	pct start	0.2
Input image	size	600×960
	crop	$[64, 138, 896, 458]$ (left, top, right, bottom)
	field of view	$100°$
	camera position	$[-1.5, 0.0, 2.0]$ (forward, right, up)
	camera rotation	$[0.0, 0.0, 0.0]$ (pitch, yaw, roll)
BeV label	size	192×192
	resolution	0.2m/pixel
Sequence	length	12
	frequency	5 Hz
Loss	action weight	1.0
	segmentation weight	0.1
	segmentation top-k	0.25
	instance weight	0.1
	instance center weight	200.0
	instance offset weight	0.1
	image weight	0.0
	kl weight	1×10^{-3}
	kl balancing	0.75

Lifting features to 3D. Given image features $\mathbf{u}_t \in \mathbb{R}^{C_e \times H_e \times W_e}$ and depth probabilities $\mathbf{d}_t \in \mathbb{R}^{D \times H_e, \times W_e}$, we can lift the features to 3D using the camera intrinsics K and extrinsics

Table A.2.3: Inference model ϕ.

Category	Layer	Output size	Parameters
Image encoder e_ϕ	Input	$3 \times 320 \times 832 = \mathbf{o}_t$	0
	ResNet18 [50]	$[128 \times 40 \times 104, 256 \times 20 \times 52, 512 \times 10 \times 26]$	11.2M
	Feature aggregation	$64 \times 40 \times 104 = \mathbf{u}_t$	0.5M
	Depth	$37 \times 40 \times 104 = \mathbf{d}_t$	0.5M
	Lifting to 3D	$64 \times 37 \times 40 \times 104$	0
	Pooling to BeV	$64 \times 48 \times 48 = \mathbf{b}_t$	0
Route map encoder e_ϕ	Input	$3 \times 64 \times 64 = \mathbf{route}_t$	0
	ResNet18 [50]	$16 = \mathbf{r}_t$	11.2M
Speed encoder e_ϕ	Input	$1 = \mathbf{speed}_t$	0
	Dense layers	$16 = \mathbf{m}_t$	304
Compressing to 1D e_ϕ	Input	$[64 \times 48 \times 48, 16, 16] = [\mathbf{b}_t, \mathbf{r}_t, \mathbf{m}_t]$	0
	ResNet18 [50]	$512 = \mathbf{x}_t$	11.5M
Posterior network (μ_ϕ, σ_ϕ)	Input	$[1024, 512] = [\mathbf{h}_t, \mathbf{x}_t]$	0
	Dense layers	$[512, 512]$	3.9M

Table A.2.4: Generative model θ.

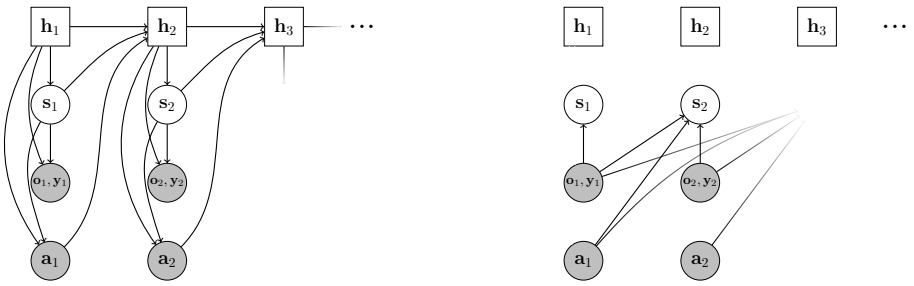
Category	Layer	Output size	Parameters
Prior network $(\mu_\theta, \sigma_\theta)$	Input	$1024 = \mathbf{h}_t$	0
	Dense layers	$[512, 512]$	2.1M
Recurrent cell f_θ	Input	$[1024, 512, 2] = [\mathbf{h}_t, \mathbf{s}_t, \mathbf{a}_t]$	0
	Action layer	64	192
	Pre GRU layer	1024	0.6M
	GRU cell	$1024 = \mathbf{h}_{t+1}$	6.3M
BeV decoder l_θ	Input	$[512 \times 3 \times 3, 1024, 512] = [\text{constant}, \mathbf{h}_t, \mathbf{s}_t]$	0
	Adaptive instance norm	$512 \times 3 \times 3$	1.6M
	Conv. instance norm	$512 \times 3 \times 3$	3.9M
	Upsample conv. instance norm	$512 \times 6 \times 6$	7.9M
	Upsample conv. instance norm	$512 \times 12 \times 12$	7.9M
	Upsample conv. instance norm	$512 \times 24 \times 24$	7.9M
	Upsample conv. instance norm	$256 \times 48 \times 48$	3.3M
	Upsample conv. instance norm	$128 \times 96 \times 96$	1.2M
	Upsample conv. instance norm	$64 \times 192 \times 192$	0.5M
	Output layer	$[8 \times 192 \times 192, 1 \times 192 \times 192, 2 \times 192 \times 192]$	715
Policy π_θ	Input	$[1024, 512] = [\mathbf{h}_t, \mathbf{s}_t]$	0
	Dense layers	2	5.9M

M: $\text{Lift}(\mathbf{u}_t, \mathbf{d}_t, K^{-1}, M) \in \mathbb{R}^{C_e \times D \times H_e \times W_e \times 3}$. The Lift operation can be detailed as follows: (i) Using the inverse intrinsics K^{-1} and predicted depth, the features in the pixel image space are lifted to 3D in camera coordinates with a pinhole camera model, (ii) the rigid body motion M transforms the 3D camera coordinates to 3D vehicle coordinates (center of inertia of the ego-vehicle).

Additional details. The action space is in $\mathbb{R}^2$ with the first component being the acceleration in $[-1, 1]$. Negative values correspond to braking, and positive values to throttle. The second component is steering in $[-1, 1]$, with negative values corresponding to turning left, and positive values to turning right. For simplicity, we have set the weight parameter of the image reconstruction to 0. In order to improve reconstruction of the BeV vehicles and pedestrians, we also include an instance segmentation loss [49]. Finally, we use the KL balancing technique from [5].

A.3 Data collection and evaluation

Data collection. The data was collected in Town01, Town03, Town04 and Town06 with the following weather conditions: ClearNoon, WetNoon, HardRainNoon, ClearSunset. Each run was



(a) Generative model. (b) Inference model.

Figure 3: Graphical models representing the conditional dependence between states. Deterministic and stochastic states are represented by, respectively, squares and circles. Observed states are in gray.

randomised with a different start and end position, as well as with traffic agents [9]. A random number of vehicles and pedestrians were spawned in the environment as specified in Table A.3.1.

Table A.3.1: Uniform sampling intervals of spawned vehicles and pedestrians in each town during training.

Town	**Number of vehicles**	**Number of pedestrians**
Town01	$[80, 160]$	$[80, 160]$
Town03	$[40, 100]$	$[40, 100]$
Town04	$[100, 200]$	$[40, 120]$
Town06	$[80, 160]$	$[40, 120]$

Metrics. We also wanted to highlight the limitations of the driving score as it is obtained by multiplying the route completion with the infraction penalty. The route completion (in $[0, 1]$) can be understood as the recall: how far the agent has travelled along the specified route. The infraction penalty (also in $[0, 1]$) starts at 1.0 and decreases with each infraction with multiplicative penalties. It can be understood as the precision: how many infractions has the agent successfully avoided. Therefore, two models are only comparable at a given recall (or route completion), as the more miles are driven, the more likely the agent risks causing infractions. We instead suggest reporting the cumulative reward in future, that overcomes the limitations of the driving score by being measured at the timestep level. The more route is driven, the more rewards are accumulated along the way. This reward is however modulated by the driving abilities of the model (and can be negative when encountering hard penalties. More formally:

- **Route completion** $R_{\text{completion}} \in [0, 1]$: for a given simulation scenario, the percentage of route completed by the driving agent. The simulation can end early if the agent deviates from the desired route by more than 30m, or does not take any action for 180s.
- **Infraction penalty** I_{penalty}: multiplicative penalty due to various infractions from the agent (collision with pedestrians/vehicles/static objects, running red lights etc.). $I_{\text{penalty}} \in [0, 1]$, with $I_{\text{penalty}} = 1$ meaning no infraction was observed.
- **Driving score** D: measures both how far the agent drives on the given route, but also how well it drives. D is defined as $D = R_{\text{completion}} \times I_{\text{penalty}} \in [0, 1]$, with $D = 1$ corresponding perfect driving. For a full description of these metrics, please refer to [45].

At every timestep, the environment computes a reward $r \in [R_{\text{min}}, 1]$ [46] for the driving agent. If N is the number of timesteps the agent was deployed for without hitting a termination criteria, then the **cumulative reward** $R \in [N \times R_{\text{min}}, N]$. In order to account for the length of the simulation (due to various stochastic events, it can be longer or shorter), we also report the **normalised cumulative reward** $\overline{R} = R/N$.

Evaluation settings. We measure the performance of our model on two settings. Each evaluation is repeated three times.

- **New town, new weathers**: the 10 test scenarios in Town05 [45], on 4 unseen weather conditions: SoftRainSunset, WetSunset, CloudyNoon, MidRainSunset.
- **Train town, train weathers**: the 20 train scenarios in Town03 [45], on 4 train weather conditions: ClearNoon, WetNoon, HardRainNoon, ClearSunset.

B Extended results

B.1 Dreaming in the simulator

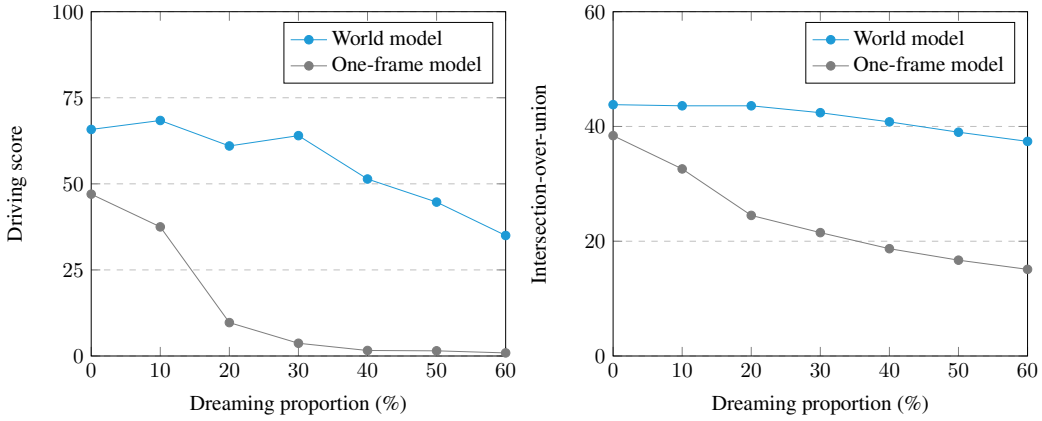
Humans build a mental model of the world while driving. Even if no observations are available for a brief moment (e.g. if they close their eyes), they can still reliably predict their next states and actions to exhibit reasonable driving behaviour.

We test whether our learned world model could display such abilities by making our model dream during closed-loop driving. We deploy the model in the fully recurrent setting, and at fixed intervals: (i) we let the model imagine future states and actions, without observing new images, and execute those actions in the simulator. (ii) We then let the model update its knowledge of the world by observing new image frames. More precisely, we set the fixed interval to a two-second window, and set a ratio of dreaming vs. observing. If for example that ratio is set to 0.5, we make the model dream by sampling from the prior distribution for 1.0s, then sample from the posterior distribution for 1.0s, and alternate between these two settings during the whole evaluation run.

We make the ratio of dreaming vs. observing vary from 0 (always observing each image frame, which is the default behaviour) to 0.6 (dreaming for 60% of the time). We report both the driving performance and perception accuracy in Figure 4. The driving performance is measured with the driving score, and the perception accuracy using the intersection-over-union with the ground truth BeV semantic segmentation. We compare MILE with a one-frame baseline which has no memory and only uses a single image frame for inference.

Figure 4a shows that our model can dream for up to 30% of the time without any significant drop in driving performance. After this point, the driving score starts decreasing but remains much higher than its one-frame counterpart. In Figure 4b, we see that the predicted states remain fairly accurate (by decoding to BeV segmentation), even with an important amount of dreaming. These results demonstrate that our model can predict plausible future states and actions, accurate enough to control a vehicle in closed-loop.

Figure 5 and Figure 6 illustrate examples of the model dreaming in the simulator and successfully stopping for traffic and making a right turn at an intersection.



(a) Driving performance.

(b) Perception accuracy.

Figure 4: Dreaming in the simulator. We report the closed-loop driving performance and perception accuracy in CARLA when the model dreams and does not observe a proportion of the images.

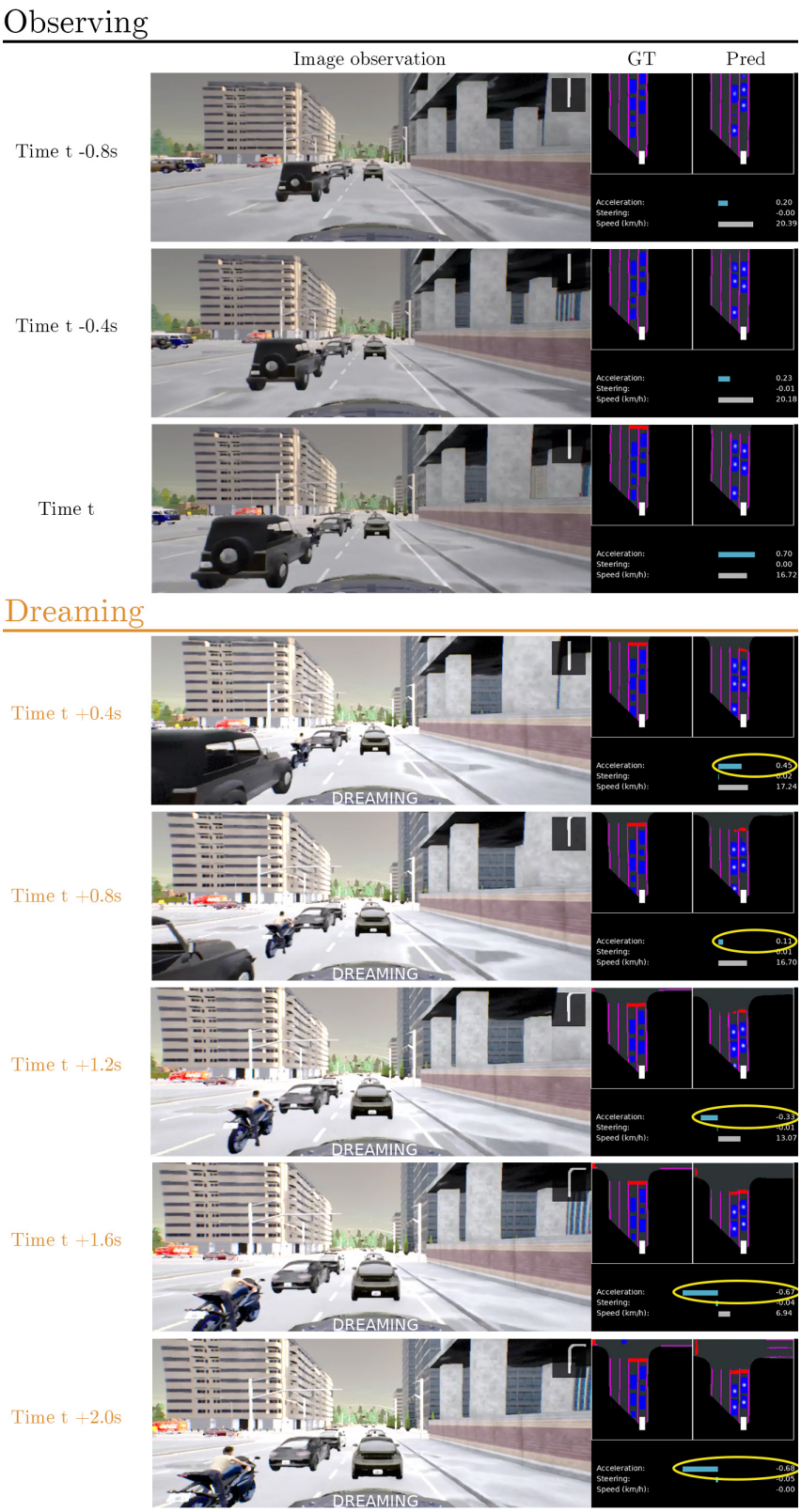


Figure 5: An example of the model dreaming and accurately predicting future states and actions to slow down for traffic. When dreaming, the model does not observe the image frames, but predicts the future states and actions from its latent state. Notice how the model predicts deceleration as it approaches the vehicle in front.

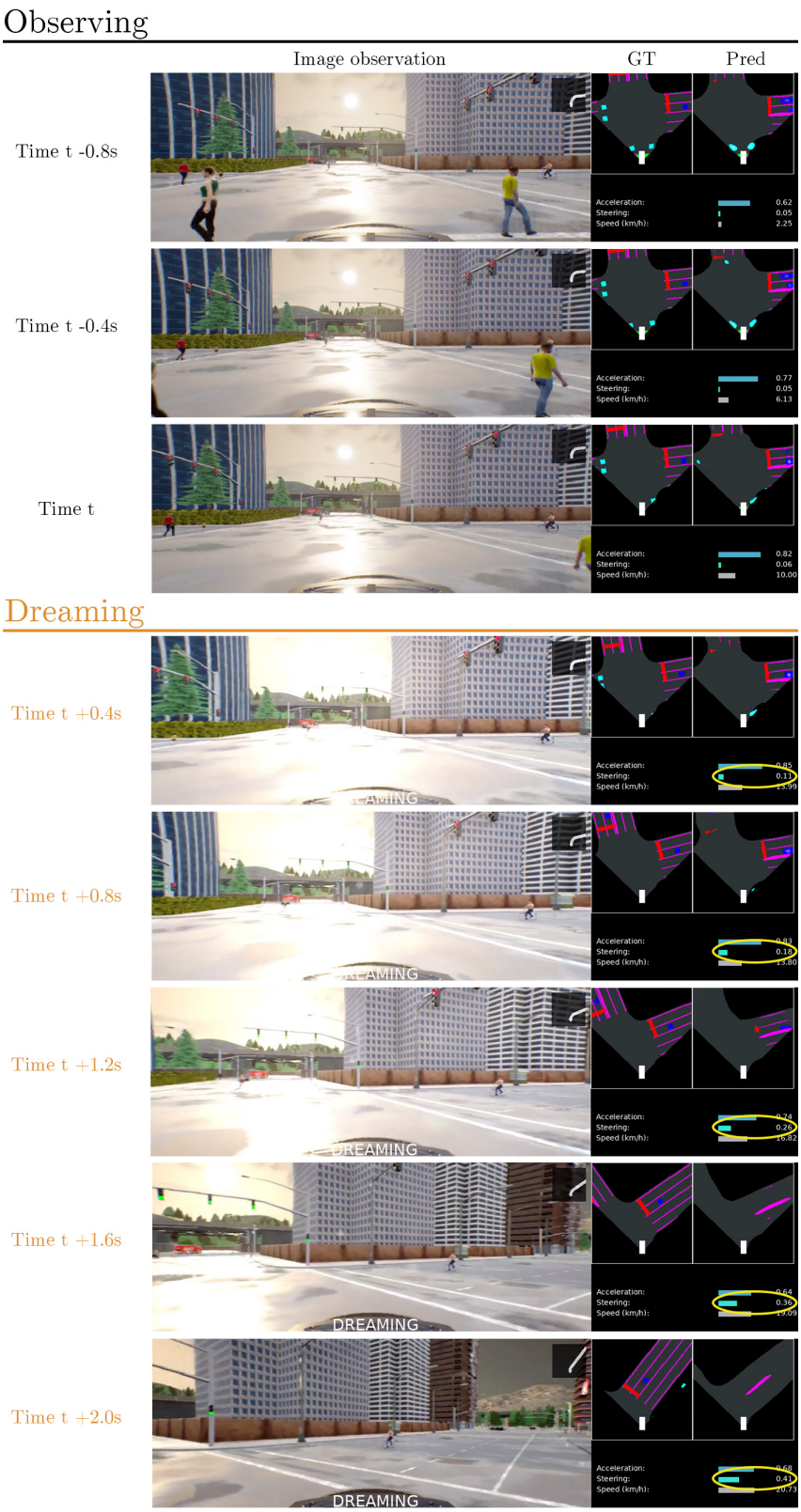


Figure 6: An example of the model dreaming and accurately predicting future states and actions to make a right turn at an intersection. When dreaming, the model does not observe the image frames, but predicts the future states and actions from its latent state. Notice how the model predicts steering towards the right.

B.2 Image resolution

In urban driving, small elements in the scene can have an important role in decision making. One typical example is traffic lights, which only occupy a small portion of the image, but dictate whether a vehicle can continue driving forward or need to stop at a red light. Figure 7 and Figure 8 illustrate how traffic lights and pedestrians become much harder to distinguish in lower image resolutions.

We evaluate the importance of image resolution by training MILE at different resolutions: 75×120, 150×240, 300×480, and 600×960 (our proposed resolution). We report the results in Table 4 and observe a significant decrease in both driving score and cumulative reward. The performance drop is most severe in the infraction penalty metric. To get a better understanding of what is happening, we detail in Table 5 the breakdown of the infractions. We report the number of red lights run, the number of vehicle collisions, and the number of pedestrian collisions, all per kilometre driven. As the resolution of the image lowers, the number of infractions increases across all modalities (red lights, vehicles, and pedestrians). These results highlight the importance of high resolution images to reliably detect traffic lights, vehicles, and pedestrians.

Table 4: Ablation studies on the image resolution. We report driving performance on a new town and new weather conditions in CARLA.

Image resolution	Driving Score	Route	Infraction	Reward	Norm. Reward
75×120	20.9 ± 0.0	87.5 ± 0.0	25.3 ± 0.0	5674 ± 0.0	0.65 ± 0.0
150×240	27.9 ± 0.0	81.8 ± 0.0	40.4 ± 0.0	5017 ± 0.0	0.65 ± 0.0
300×480	43.3 ± 0.0	96.1 ± 0.0	44.4 ± 0.0	5814 ± 0.0	0.55 ± 0.0
600×960	$\mathbf{61.1 \pm 3.2}$	$\mathbf{97.4 \pm 0.8}$	$\mathbf{63.0 \pm 3.0}$	$\mathbf{7621 \pm 460}$	$\mathbf{0.67 \pm 0.02}$
Expert	88.4 ± 0.9	97.6 ± 1.2	90.5 ± 1.2	8694 ± 88	0.70 ± 0.01

Table 5: Ablation studies on the image resolution. We report the breakdown of infraction penalties on a new town and new weather conditions in CARLA. The metrics are: number of red lights run, number of vehicle collisions, and number of pedestrian collisions. They are normalised per kilometre driven. Lower is better.

Image resolution	Red lights (↓)	Vehicles (↓)	Pedestrians (↓)
75×120	3.07 ± 0.0	0.77 ± 0.0	0.07 ± 0.0
150×240	2.39 ± 0.0	0.35 ± 0.0	0.03 ± 0.0
300×480	0.99 ± 0.0	0.31 ± 0.0	0.05 ± 0.0
600×960	$\mathbf{0.13 \pm 0.04}$	$\mathbf{0.24 \pm 0.05}$	$\mathbf{0.01 \pm 0.01}$
Expert	0.04 ± 0.01	0.15 ± 0.01	0.02 ± 0.00

(a) Resolution 600×960.

(b) Resolution 300×480.

(c) Resolution 150×240.

(d) Resolution 75×120.

Figure 7: Input image observation at different resolutions. The red traffic light becomes almost indistinguishable in lower resolutions.

(a) Resolution 600×960.

(b) Resolution 300×480.

(c) Resolution 150×240.

(d) Resolution 75×120.

Figure 8: Input image observation at different resolutions. It becomes increasingly harder to see the pedestrian as the resolution decreases.

B.3 Performance scaling with data

Table 6 details the dataset size of prior works on CARLA. Most recent methods have a similar dataset size of around 20-30 hours.

We investigate how performance scales with data. We train our proposed model with a fraction of the full dataset ($1/2$, $1/4$, $1/8$, $1/16$ and $1/32$) and report the results in Table 7. We observe similar performance across all metrics for a dataset size of 8 to 32 hours. Performance however degrades monotonically starting from 4 hours of training data. We conclude that on the CARLA simulator, around 8 hours of data collected by our expert driver is sufficient.

Table 6: Dataset size of prior works.

	Hours of data	**Frames**	**Frame rate (Hz)**
CILRS [17]	100	-	10
LBC [47]	4	0.2M	10
TransFuser [48]	31	0.2M	2
Roach [9]	18	-	10
LAV [10]	28	0.4M	4
MILE	32	2.9M	25

Table 7: Ablation studies on the dataset size. We report driving performance on a new town and new weather conditions in CARLA.

Dataset size	**Driving Score**	**Route**	**Infraction**	**Reward**	**Norm. Reward**
1h (3.125%)	52.9 ± 0.0	96.6 ± 0.0	54.7 ± 0.0	6816 ± 0.0	0.62 ± 0.0
2h (6.25%)	55.8 ± 0.0	95.7 ± 0.0	57.6 ± 0.0	7079 ± 0.0	0.61 ± 0.0
4h (12.5%)	57.6 ± 0.0	95.5 ± 0.0	60.6 ± 0.0	7042 ± 0.0	0.62 ± 0.0
8h (25%)	$\mathbf{61.6 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	61.6 ± 0.0	$\mathbf{7954 \pm 0.0}$	0.66 ± 0.0
16h (50%)	59.6 ± 0.0	97.6 ± 0.0	62.0 ± 0.0	7345 ± 0.0	0.65 ± 0.0
32h (100%)	61.1 ± 3.2	97.4 ± 0.8	$\mathbf{63.0 \pm 3.0}$	7621 ± 460	$\mathbf{0.67 \pm 0.02}$
Expert	88.4 ± 0.9	97.6 ± 1.2	90.5 ± 1.2	8694 ± 88	0.70 ± 0.01

B.4 Training town performance

In Table 8, we report the driving performance in a train town (Town03) and training weather conditions (ClearNoon, WetNoon, HardRainNoon, ClearSunset). Note that the RL expert has a lower performance than in test town Town05, because Town03 was designed as the most complex town [51].

Table 8: Driving performance in CARLA on a town and weather conditions seen during training. Metrics are averaged across three runs. The other methods did not report results in this setting.

	Driving Score	**Route**	**Infraction**	**Reward**	**Norm. Reward**
Roach [9]	50.6 ± 1.9	$\mathbf{91.0 \pm 0.7}$	56.9 ± 1.2	4419 ± 487	0.38 ± 0.04
MILE	$\mathbf{61.4 \pm 0.3}$	89.3 ± 2.5	$\mathbf{69.4 \pm 1.3}$	$\mathbf{7627 \pm 190}$	$\mathbf{0.71 \pm 0.01}$
Expert	81.5 ± 2.8	95.1 ± 1.2	85.6 ± 1.7	7740 ± 220	0.69 ± 0.03

B.5 Qualitative examples

Latent space interpolation. We investigate the structure of the learned latent space. To do so, we first calculate posterior latent states from two random scenarios from the evaluation set. We then interpolate between the two states, unroll their future states, and decode them to BeV segmentation. We use spherical interpolation (slerp) with equal step sizes in the latent space.

In Figure 9, the two scenes correspond to a multi-lane intersection and a T-junction. As we interpolate with a coefficient α the two scenes, we observe the multi-lane intersection ($\alpha = 0$) smoothly transitioning to the T-junction ($\alpha = 1$): the two horizontal lanes become one at $\alpha = 0.5$, and the vertical lane ahead disappears at $\alpha = 0.75$.

In Figure 10, we interpolate between an empty multi-lane straight road ($\alpha = 0$) and a complex intersection ($\alpha = 2$). Again, we observe a smooth transition between the two scenarios with increasingly more complex lane structure and traffic density. These interpolations show evidence that the learned latent space has a structure that is coherent with road complexity, and that our model is capable of representing key differences between two scenarios.

Long-horizon prediction. Figure 11 shows interesting behaviour predicted by the world model: making turns and imagining the traffic pulling away at a traffic light. Figure 12 shows a prediction one minute in the future. Please refer to the supplementary material to visualise the predictions as a video.

Figure 9: Latent space interpolation. From top to bottom, we interpolate the latent states of two scenarios with α controlling the degree of interpolation from 0 to 1. From left to right, we use the interpolated states to imagine future states. We observe the first scenario (multi-lane intersection, top row) smoothly transitions to the second scenario (T-junction, bottom row): the two horizontal lanes become one at $\alpha = 0.5$, and the vertical lane ahead disappears at $\alpha = 0.75$.

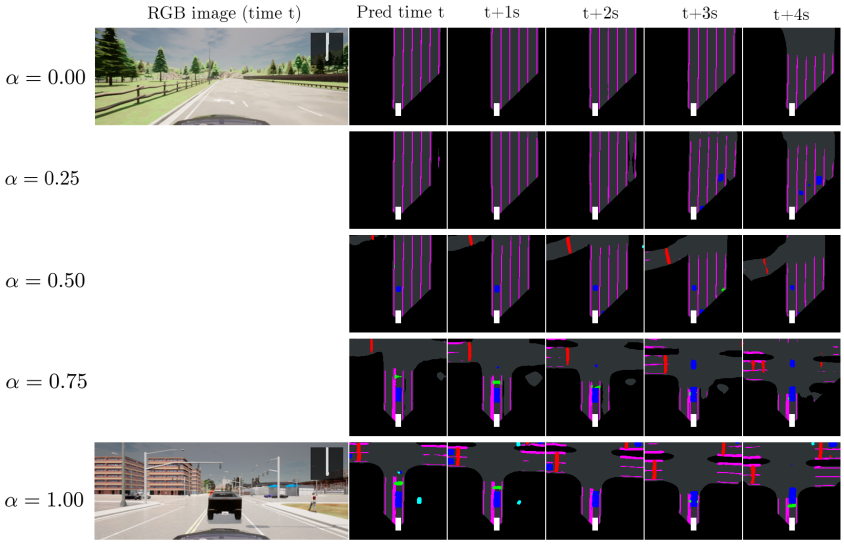
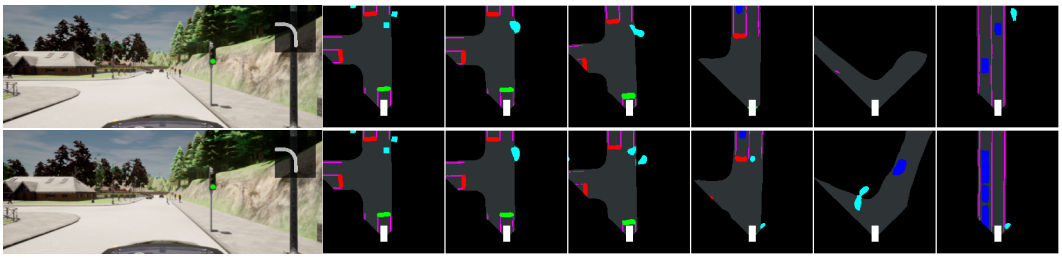
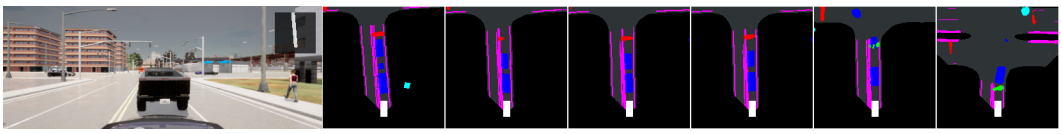


Figure 10: Latent space interpolation. From top to bottom, we interpolate the latent states of two scenarios with α controlling the degree of interpolation from 0 to 1. From left to right, we use the interpolated states to imagine future states. We observe the first scenario (empty multi-lane straight road, top row) smoothly transitions to the second scenario (complex intersection, bottom row): the road structure and traffic become increasingly more complex.



(a) The model imagines two left turns at this intersection, with different traffic density in the left lane (which is not visible yet from the RGB images).



(b) Stuck as this red light, the model imagines the traffic light turning green, resulting in the traffic pulling away.

Figure 11: Qualitative examples of multi-modal predictions, for 8 seconds in the future. BeV segmentation legend: white = ego-vehicle, black = background, gray = road, purple = line marking, blue = vehicles, cyan = pedestrians, green/yellow/red = traffic lights. Ground truth labels (GT) outside the field-of-view of the front camera are masked out.

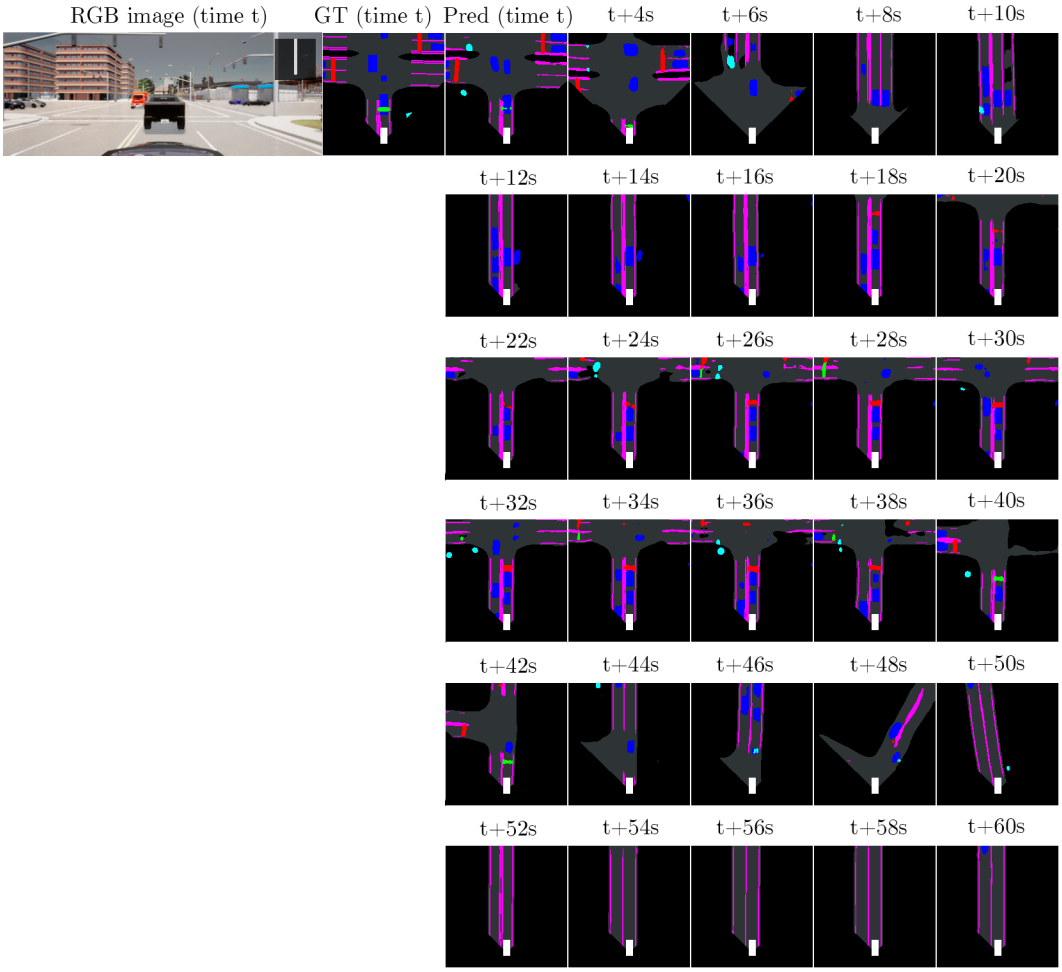


Figure 12: Our model can predict plausible long-horizon futures. In this example, we predict one minute in the future.