Supplementary Material for Learning Situational Driving

Eshed Ohn-Bar[1,3] Aditya Prakash[1] Aseem Behl[1,2] Kashyap Chitta[1,2] Andreas Geiger[1,2]
[1]Max Planck Institute for Intelligent Systems, Tübingen [2]University of Tübingen [3]Boston University
{firstname.lastname}@tue.mpg.de

Abstract

In this supplementary document, we discuss (1) ***network architecture and training details****, including reward function definition, and (2)* ***additional details regarding the experiments and evaluation*** *in the main paper. Please also see the provided supplementary video for a summary of the work.*

1. Training and Implementation Details

Image Input: The model observes a 256×256 pixels image, the speed of the ego-vehicle, and a high-level command. Starting from a 800×600 image, we follow [4] and crop 125 pixels from the top and 90 at the bottom of the image.

Command Input: We encode the high-level command c as a one-hot vector and input it to the network, unlike the hand-defined control gating for the policy heads employed in most previous studies [3, 4, 8]. While the two architecture design choices are orthogonal, there are two main practical reasons for this modification. This architecture modification allows us to isolate the benefits of data-driven prediction heads, i.e., in the mixture model formulation. As we vary the number of components, the number of prediction heads can be set higher than the number of high-level commands. Second, with c as an input the network can potentially learn to leverage similarities among the commands, as opposed to learning a control output for each command separately (indeed, there are control similarities between right and left turn, as well as forward and stay in lane). Hence, our approach can potentially better scale for real-world applications, i.e., beyond CARLA, where there could be a variety of high-level commands.

Network Architecture: Table 1 and Table 2 show the architectures for the MoE and VAE models, respectively. To ensure sufficient network capacity throughout the experiments we employ a ResNet-50 backbone, but keep the VAE and reconstruction branch shallow [9]. We collect a large driving demonstration dataset during training of about 50 hours in total. Visualizing the dataset in Fig. 1 shows clear multi-modal patterns. Therefore, an MoE model is well-suited for modeling the multiple modalities in the training data.

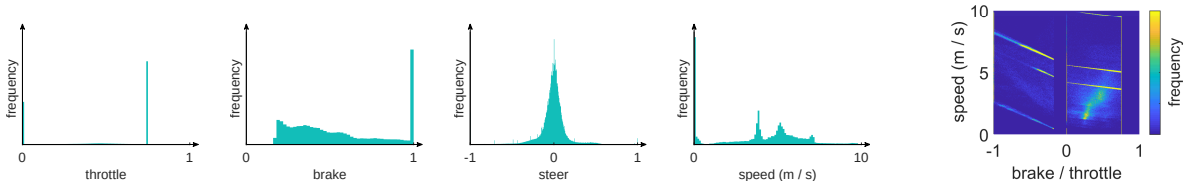

Figure 1: **Expert Demonstrations Statistics.** The figure summarizes the training data used in the experiments. As described in the main paper, the data is indeed multi-modal.

Context Embedding: The goal of the context embedding is to provide a general purpose context feature of the ego-vehicle's surroundings. This standalone module trained with a separate objective can also provide additionally diversity during the

Table 1: **MoE Architecture.**

Module	Input	Output
Image Encoder	ResNet [7]	512
Speed	1	512
	512	512
Command	4	512
	512	512
Speed Prediction	512	512
	512	512
	512	1
Expert Prediction	512×3	512
	512	512
	512	6

Table 2: **VAE Encoder Architecture.** We train a Convolutional VAE. The decoder for up-sampling the latent code back to a 256×256 image is symmetric wrt. the encoder.

Module	Input	Output
ConvVAE Encoder	256×256	3
	127×127	16
	62×62	32
	30×30	64
	14×14	128
	6×6	256
	2×2	512
μ_z	2048	128
σ_z	2048	128

policy learning process, and can be easily trained over all experienced visual observations. Moreover, the purpose of this learned image embedding is to provide additional model capacity (i.e., over the already strong experts) during the task-driven optimization step. Although beyond the scope of our work, more sophisticated image embedding architectures can also be investigated in the future, e.g., [1]. The encoder/decoder utilizes 4×4 convolutional/deconvolution filters at a stride of 2. We also find that enforcing a Gaussian prior over the latent vector z impacts the learned policy performance significantly, and set $\beta = 1$ as in [6]. In contrast, setting $\beta = 0$ deteriorates driving performance. Hence, we can explain this impact on policy learning by the latent code normalization that the β value provides.

Task-Driven Policy Refinement: The driving policy is further optimized using CMA-ES for 1000 generations and a population of 8 agents. To optimize with respect to the task, we define a reward function as a combination of several task-based metrics, computed over measurements from the simulator. Following an action in each time step, the immediate reward is computed as,

$$r_t(\mathbf{o}_t, c_t, \mathbf{a}_t, \mathbf{p}_t, \mathbf{e}_t) = r_t^{goal} + r_t^{speed} + r_t^{steer} + r_t^{lane} + r_t^{light} + r_t^{collision} + r_t^{safedist} \tag{1}$$

While some of the reward terms can be computed using the velocity measurements, the high-level command, and the actions of the ego-vehicle, others employ the current ego-vehicle position $\mathbf{p}_t$ and additional properties of the environment, $\mathbf{e}_t$, i.e., for the purpose of infraction detection, as described next.

The overall task is captured with a distance to goal reward. Following [5], we compute the distance d traveled towards to the goal g, in meters, given the ego-vehicle position $\mathbf{p}$,

$$r_t^{goal}(\mathbf{p}_{t-1}, \mathbf{p}_t, \mathbf{g}) = d(\mathbf{p}_{t-1}, \mathbf{g}) - d(\mathbf{p}_t, \mathbf{g}) \tag{2}$$

In practice, it is common to add a small constant value to the goal reward term in order to encourage timely arrival. In this work, we add a penalty based on the current speed to ensure timely completion of the episode,

$$r_t^{speed}(v_t) = \begin{cases} -0.01 & \text{if } v_t < 10 \text{ km/h} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

When training the task-optimized policy, we found that the agent can still learn to complete the task with the above reward terms but with unacceptable driving behavior. For instance, the agent may develop a side-to-side wobbling strategy when driving in the ego-lane. To avoid learning such strategies, following [8], we define a steering term,

$$r_t^{steer}(\mathbf{a}_t, c_t) = \begin{cases} -0.05 & \text{if } \mathbf{a}_t[2] > 0 \text{ and } c_t = left \text{ or } \mathbf{a}_t[2] < 0 \text{ and } c_t = right \\ -0.05 & \text{if } |\mathbf{a}_t[2]| > 0.05 \text{ and } c_t = straight \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Where $\mathbf{a}_t[2]$ is the second component of the regressed action vector, i.e., the steering control. The steering term also helps ensure correct turning behavior. Similarly, the agent is penalized for not staying in lane or for driving over the sidewalk,

$$r_t^{lane}(\mathbf{e}_t) = \begin{cases} -0.05 & \text{if overlap with other lane or sidewalk} > 0.2 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

While we terminate an episode due to a time-out or collision, to be consistent with the evaluation on CARLA we allow running red lights but at a high cost,

$$r_t^{light}(\mathbf{e}_t) = \begin{cases} -10 & \text{if ran red traffic light} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

A collision leads to a high negative reward and result in episode termination,

$$r_t^{collision}(\mathbf{e}_t) = \begin{cases} -100 & \text{if any collision} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Finally, we add a reward term that encourages the agent to maintain a safe distance from dynamic obstacles. The safe distance term aims to reduce the likelihood of stopping too late and colliding with an object. Moreover, encouraging the agent to maintain a safe distance results in a more desirable driving behavior. We compute the distance in meters to any pedestrians $\mathbf{p}_{pedestrians}$ or vehicles $\mathbf{p}_{vehicles}$ ahead of the ego-vehicle and penalize instances where it is less than one,

$$r_t^{safedist}(\mathbf{e}_t) = \begin{cases} -0.05 & \text{if } d(\mathbf{p}_t, \mathbf{p}_{pedestrians}) < 1 \text{ or } d(\mathbf{p}_t, \mathbf{p}_{vehicles}) < 1 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Although the reward is computed at each time step, it is summed over the entire episode before updating the model. Hence, this form of reward-based supervision is still somewhat weak, i.e., compared to methods that assume precise knowledge of the entire 3D state of the surroundings for action prediction supervision at every time step, e.g., as in Chen et al. [2].

2. Experimental Details

Example Model Predictions: Fig. 2 and Fig. 3 show qualitative results for example turn and traffic light scenarios. Inspecting the individual components (Eq. 1 of the main paper) for these specific cases demonstrates the benefits of the proposed approach. Specifically, we can see how the expert models produce diverse control actions. Also, as expected, the actions produced by the mixture of experts model are shown to account for a large portion of the final action. Nonetheless, the context embedding (after multiplication by $\mathbf{\Psi}$) also produces a significant contribution that is representative of the event, i.e., steering left during a left turn and braking during a red traffic light.

Visual Observation

Model Components

π	$\pi_{\boldsymbol{\theta}}^{MoE}$	$\pi_{\mathbf{\Psi},\boldsymbol{\phi}}^{CE}$
$[0.436, -0.511]$	$[0.385, -0.414]$	$[0.051, -0.097]$

$\pi_{\boldsymbol{\theta}}^{1}$	$\pi_{\boldsymbol{\theta}}^{2}$	$\pi_{\boldsymbol{\theta}}^{3}$	$\pi_{\boldsymbol{\theta}}^{4}$	$\pi_{\boldsymbol{\theta}}^{5}$
$[-1.000, -0.455]$	$[0.237, -0.152]$	$[-0.855, -0.364]$	$[0.748, -0.376]$	$[0.969, -0.469]$

$\alpha_{\boldsymbol{\theta}}^{1}$	$\alpha_{\boldsymbol{\theta}}^{2}$	$\alpha_{\boldsymbol{\theta}}^{3}$	$\alpha_{\boldsymbol{\theta}}^{4}$	$\alpha_{\boldsymbol{\theta}}^{5}$
$[0.148, 0.002]$	$[0.119, 0.173]$	$[0.027, 0.001]$	$[0.703, 0.000]$	$[0.003, 0.824]$

Figure 2: **Example Model Components During a Turn.** To understand the situational policy components, we inspect the model predictions (see Eq. 1 in the main paper) given example input observations. We show predictions for: final action [throttle/brake, steering] output from policy π, the output of the mixture of experts policy $\pi_{\boldsymbol{\theta}}^{MoE}$, the output of the context embedding policy $\pi_{\mathbf{\Psi},\boldsymbol{\phi}}^{CE}$ (i.e., $\pi = \pi_{\boldsymbol{\theta}}^{MoE} + \pi_{\mathbf{\Psi},\boldsymbol{\phi}}^{CE}$), the individual expert models predictions provided by $\pi_{\boldsymbol{\theta}}^{k}$, and the mixture weights $\alpha_{\boldsymbol{\theta}}^{k}$.

Visual Observation

Model Components

π	$\pi^{MoE}_{\boldsymbol{\theta}}$	$\pi^{CE}_{\boldsymbol{\Psi},\boldsymbol{\phi}}$
$[-0.064, 0.082]$	$[-0.046, 0.089]$	$[-0.018, -0.007]$

$\pi^1_{\boldsymbol{\theta}}$	$\pi^2_{\boldsymbol{\theta}}$	$\pi^3_{\boldsymbol{\theta}}$	$\pi^4_{\boldsymbol{\theta}}$	$\pi^5_{\boldsymbol{\theta}}$
$[-0.615, 0.097]$	$[-0.295, 0.093]$	$[-1.000, 0.047]$	$[0.780, 0.100]$	$[0.666, 0.101]$

$\alpha^1_{\boldsymbol{\theta}}$	$\alpha^2_{\boldsymbol{\theta}}$	$\alpha^3_{\boldsymbol{\theta}}$	$\alpha^4_{\boldsymbol{\theta}}$	$\alpha^5_{\boldsymbol{\theta}}$
$[0.134, 0.001]$	$[0.246, 0.895]$	$[0.210, 0.092]$	$[0.402, 0.001]$	$[0.008, 0.010]$

Figure 3: **Example Model Components During a Red Traffic Light.** We show predictions for: final action [throttle/brake, steering] output from policy π, the output of the mixture of experts policy $\pi^{MoE}_{\boldsymbol{\theta}}$, the output of the context embedding policy $\pi^{CE}_{\boldsymbol{\Psi},\boldsymbol{\phi}}$ (i.e., $\pi = \pi^{MoE}_{\boldsymbol{\theta}} + \pi^{CE}_{\boldsymbol{\Psi},\boldsymbol{\phi}}$), the individual expert models predictions provided by $\pi^k_{\boldsymbol{\theta}}$, and the mixture weights $\alpha^k_{\boldsymbol{\theta}}$.

Model Limitations in Harsh Environments and the *AnyWeather* Benchmark: In this section, we provide additional details about the generalization experiment in the main paper. For clarity, we illustrate all the weathers used in the experiments in Fig. 4. As can be seen, some weathers are significantly more challenging to drive in than others due to limited visibility and reflections.

Figure 4: **Visualization of All Weathers in the *AnyWeather* Benchmark.** We evaluate driving models on a larger number of weathers that were not seen during training. The experiment enables us to better understand the limitations of current state-of-the-art driving models. A "*" indicates the two most challenging weathers, not commonly used in evaluation in related literature.

Environment Limitations: In this section, we discuss limitations in the CARLA $0.8.4$ environment as these could be taken into account in future studies. As shown in Table 5 in the main paper, the expert agent does not achieve 100% success rate. In particular, there are a couple of issues under dense evaluation settings due to the high number of vehicles and pedestrians. While designing an expert driver for such conditions is more challenging, upon inspection we can also see that the dense settings may lead to issues unrelated to the ego-vehicle's behavior. For instance, intersections may become congested to the point of no movement by any surrounding agent for lengthy periods of time Fig. 5. As a result, the episode then times-out while our agent waits to enter the intersection. Moreover, as the collision checks with pedestrians are not accurate and pedestrian behavior is somewhat random, unrealistic collisions can occur.

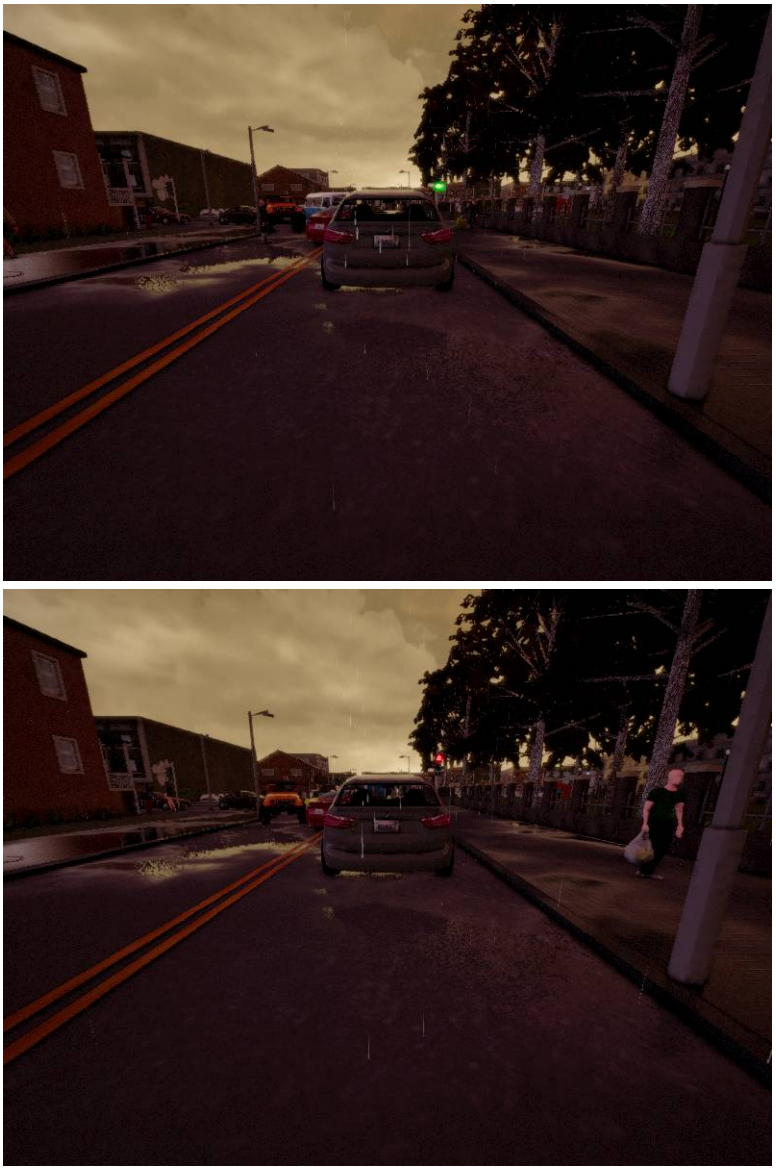

Figure 5: **Environment Limitations.** The intersection ahead becomes congested as the agent waits for it to be cleared. Consequently, the episode times-out. There is a 30 second time difference between the top image and bottom image.

References

[1] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv*, 1901.11390, 2019. 2

[2] D. Chen, B. Zhou, and V. Koltun. Learning by cheating. In *CoRL*, 2019. 3

[3] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *ICRA*. 1

[4] F. Codevilla, E. Santana, A. M. López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. *ICCV*, 2019. 1

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *CoRL*, 2017. 3

[6] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, 2018. 2

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[8] X. Liang, T. Wang, L. Yang, and E. Xing. CIRL: Controllable imitative reinforcement learning for vision-based self-driving. In *ECCV*, 2018. 1, 3

[9] X. Wang, F. Yu, L. Dunlap, Y.-A. Ma, R. Wang, A. Mirhoseini, T. Darrell, and J. E. Gonzalez. Deep mixture of experts via shallow embedding. *ICML*, 2018. 1