Increasing the Efficiency of Policy Learning for Autonomous Vehicles by Multi-Task Representation Learning

Eshagh Kargar and Ville Kyrki, *Senior Member, IEEE*

***Abstract*—Driving in a dynamic, multi-agent, and complex urban environment is a difficult task requiring a complex decision-making policy. The learning of such a policy requires a state representation that can encode the entire environment. Mid-level representations that encode a vehicle's environment as images have become a popular choice. Still, they are quite high-dimensional, limiting their use in data-hungry approaches such as reinforcement learning. In this article, we propose to learn a low-dimensional and rich latent representation of the environment by leveraging the knowledge of relevant semantic factors. To do this, we train an encoder-decoder deep neural network to predict multiple application-relevant factors such as the trajectories of other agents and the ego car. Furthermore, we propose a hazard signal based on other vehicles' future trajectories and the planned route which is used in conjunction with the learned latent representation as input to a down-stream policy. We demonstrate that using the multi-head encoder-decoder neural network results in a more informative representation than a standard single-head model. In particular, the proposed representation learning and the hazard signal help reinforcement learning to learn faster, with increased performance and less data than baseline methods.**

***Index Terms*—Autonomous vehicles, representation learning, policy learning, multi-task learning.**

I. Introduction

DRIVING in unstructured and dynamic urban environments is an arduous task. Many moving agents such as cars, bicycles, and pedestrians affect driver's behavior and decisions. To drive a car, a driver, whether a human being or an artificial agent, needs to perceive and understand other agents' behaviors, plans, and interactions in the environment, to react accordingly. The number of factors in the scene makes the state space of this problem very large, and safe autonomous driving in this complexity is an open challenge for the research community and industry.

A central challenge related to the high complexity of the traffic environment is representing the environment state.

In end-to-end methods, immediate sensor measurements are used directly as the state for the decision-making policy [1], [2]. However, the high dimensionality of the sensor data makes its direct use challenging, as a vast number of data are needed to constrain the learning problem [3]. For that reason, mid-level representations, that render all perceptual inputs together with static information such as HD-map and route as bird's eye view (BEV) images, have recently received increasing attention (see e.g. [4]–[6]). However, even the mid-level representations may be so high-dimensional that their use with data-hungry methods such as Reinforcement Learning (RL) is limited.

To alleviate this, we propose a new multi-task learning approach to learn low-dimensional and rich representations. In particular, we combine the recently proposed idea to learn a low-dimensional latent space of the mid-level image [5] with the prediction of multiple auxiliary application-relevant tasks. A single latent representation is extracted from the mid-level input representations such that the latent representation is enforced to predict the trajectories of both the ego vehicle and other vehicles in addition to a bird's eye view of the scene, using a multi-head network structure depicted in Fig. 1. All heads of the network represent the information as images, which allows their straightforward interpretation. Also, we use the motion prediction head result and the route to calculate a hazard signal as an additional input to the policy. This will give the policy information about hazardous situations and collision chance.

Experiments demonstrate that the auxiliary tasks allow the latent vector to learn more representative information from the scene. Therefore, a policy network can be trained faster and performs better, even with less data than a representation based on a single-head network trained for the optimal reconstruction of the scene.

The primary contributions of this work are: (a) a multi-task network with auxiliary heads to improve the quality of low-dimensional representations, (b) a hazard signal calculated by the likelihood between route and predicted trajectories of dynamic agents, and (c) an experimental study of an RL policy learning, showing that the learned latent-vector by using auxiliary tasks and also the hazard signal, can help the policy to be (i) trained faster, (ii) perform better, (iii) learn to solve the task using less data, (iv) and generalize better to new scenarios.

The rest of this paper is organized as follows. The review of related works in Section II demonstrates that while Reinforcement Learning has been used in autonomous driving,

Manuscript received 28 June 2021; revised 30 September 2021 and 16 December 2021; accepted 31 January 2022. Date of publication 9 February 2022; date of current version 24 October 2022. The work of Eshagh Kargar was supported by the Academy of Finland under Grant 326346. *(Corresponding author: Eshagh Kargar.)*

The authors are with the Department of Electrical Engineering and Automation, School of Electrical Engineering, Aalto University, 02150 Espoo, Finland (e-mail: eshagh.kargar@aalto.fi; ville.kyrki@aalto.fi).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIV.2022.3149891.

Digital Object Identifier 10.1109/TIV.2022.3149891

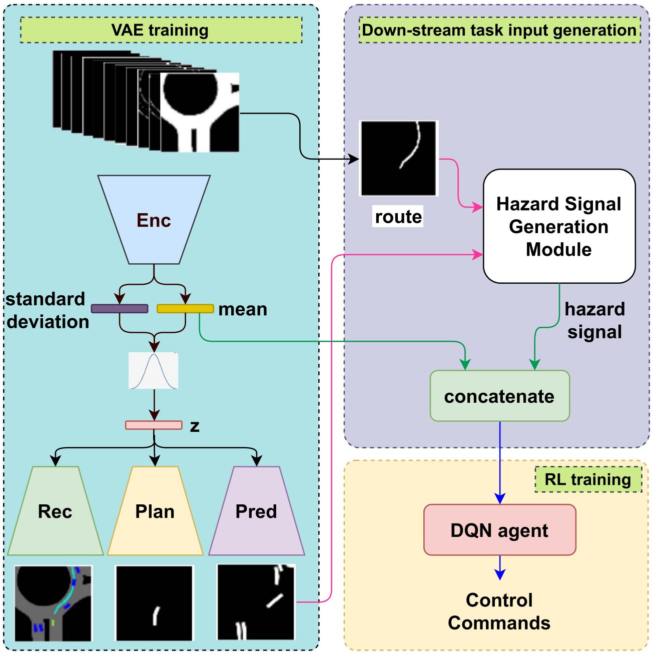


Fig. 1. Proposed framework. A multi-head Variational Auto-Encoder is trained using supervised learning to reconstruct a bird's eye view image of the scene, plan for ego car, and predict future trajectories of other vehicles. Route and motion prediction masks are used to calculate a hazard signal. Low-dimensional encoding of the environment and the hazard signals are used as state for reinforcement learning.

learning a low-dimensional state space for policy learning needs to be explored more. In Section III, we provide the required background in Variational Auto-Encoder (VAE), Reinforcement Learning, and Deep Q-Network. The problem definition and the proposed method are described in Section IV, with emphasis on the two innovations, multi-head VAE for latent representation learning and hazard signal. Then, Section V presents empirical evaluation in two driving scenarios showing superior performance of the proposed approach compared to state-of-the-art. Finally, in Section VI we conclude that using task-relevant heads in VAE and also the generated hazard signal can help to learn the down-stream driving policy more efficiently.

II. Related Work

Reinforcement learning is a popular approach to learn a decision-making policy in autonomous driving [2], [5], [7]–[13]. Despite the typically data-hungry nature of RL, Kendall *et al.* [1] recently demonstrated learning of a lane following task using real-world data gathered in a single day. However, the complexity of complex traffic environments requires great amounts of data since the state space of the decision making problem is vast.

A solution to the high-dimensional state space is to first learn a low dimensional representation of the scene from raw sensor data. The low dimensional representation can then be used as input to a decision making policy. This can be achieved e.g. by training a Variational Auto-Encoder (VAE) to form a latent representation of sensor data such as camera images [14].

Another approach to address the problem of high-dimensional state space is to use a mid-level representation, such as a bird's eye view of the current scene, as the state space. The mid-level representations can be constructed by engineered perception modules or by learning the mapping from sensor measurements to the mid-level representation [15]. Such representations are useful because they can capture the entire traffic environment around the vehicle in an interpretable fashion, but their dimensionality will still be high. Thus, techniques that increase the amount of available data are needed to use the mid-level representations directly. For example, Bansal *et al.* [4] proposed to use data augmentation to learn to imitate a driving policy using bird's eye view images as input. Similar mid-level representations can also be used for motion forecasting [16]–[20].

It is also possible to combine mid-level input with learning a low-dimensional representation. Chen *et al.* [5] trained a VAE to reconstruct a bird's eye view image of the scene with rendered mid-level information, and then used the latent representation as input to an RL agent.

To further constrain the learned latent representation to be useful, auxiliary tasks that are semantically relevant to the driving problem can be used. Hawke *et al.* [21] proposed to use auxiliary tasks in the sensor space of camera images that output segmentation, monocular depth, and optical flow to learn a better representation. The learned mid-level feature map was then used to train a policy to output control commands using imitation learning.

Our work combines the learning of low-dimensional representations from mid-level representations with the use of auxiliary tasks to further constrain the learned representation. The proposed approach uses motion prediction of other agents as one of the auxiliary tasks, which allows us to propose a novel hazard signal that can be further used in reinforcement learning to inform of potential collisions during the learning process.

III. Background

Next we will outline the required theoretical concepts behind the proposed method, including Variational Auto-Encoders (VAEs), Reinforcement learning, and Deep Q-Networks.

A. Variational Auto-Encoder

Auto-Encoder (AE) is a type of artificial neural network used in data embedding and reconstruction with an encoder-decoder structure aiming to learn a compact and low-dimensional representation z for high-dimensional input data x. Variational Auto-Encoder (VAE) extends AE by mapping input data to a distribution instead of a value. To learn compact representations, the latent distribution is subjected to a prior $p(z)$, typically a normalized Gaussian distribution.

The latent representation is then learned by minimizing the loss function [22]

$$L_{vae}(\phi,\theta) = -E_{q_\theta(z|x)} log(p_\phi(x|z)) + D_{kl}(q_\theta(z|x)||p(z)) \tag{1}$$

where $q_\theta(z|x)$ is the encoder network mapping inputs to latent space, $p_\phi(x|z)$ is the decoder network, and D_{kl} term is the KL-divergence between the encoder output and the prior. The first term of (1) corresponds then to the reconstruction error. The second term enforces the representation to be compact.

In order to improve disentanglement of the representation, β-VAE was introduced in [23] that instead of considering the KL-divergence between encoder and prior directly as a cost term, constrains the KL-divergence by an upper bound. Using Karush-Kuhn-Tucker conditions, the constrained optimization can be written using a Lagrangian factor β as an unconstrained optimization problem as

$$L_{\beta-vae}(\phi,\theta) = -E_{q_\theta(z|x)} log(p_\phi(x|z)) + \beta D_{kl}(q_\theta(z|x)||p(z)). \tag{2}$$

When $\beta = 1$, this corresponds to the regular VAE. Increasing β encourages more disentanglement with the cost of reconstruction quality.

B. Reinforcement Learning

Reinforcement learning aims to find an optimal control policy $\pi : S \rightarrow A$ from states to actions that maximizes total expected future rewards

$$R(\pi) = E_\pi\left[\sum_t \gamma^t r(s_t, a_t)\right] \tag{3}$$

where s_t, a_t, r, and γ are state at time t, action at time t, reward, and discount factor, respectively.

Value-function based RL solves the RL problem by determining an optimal value function $Q : S, A \rightarrow \mathbb{R}$ that describes the expected cumulative rewards when starting from a particular state and choosing a particular action

$$Q^*(s, a) = \max_\pi E\left[\sum_{t=0} \gamma^t r(s_t, a_t)|s_0 = s, a_0 = a\right]. \tag{4}$$

Knowing the optimal value function, an optimal policy π^* can then be determined as $\pi^*(s) = \text{argmax}_a Q^*(s, a)$.

C. Deep Q-Network

In continuous state spaces with unknown dynamics, it is usually impossible to determine the value function exactly. Deep Q-Networks (DQNs) [26] are a successful RL algorithm that use a deep neural network $Q(s, a; \psi)$ to approximate the value function where ψ are the parameters of the neural network. DQN also uses a replay buffer $D = \{(s, a, r, s')\}$ to store experiences from past to expedite and stabilize learning. To further stabilize the learning, DQN defines a target Q-network with parameters ψ' which are updated only every τ steps to the current ψ. To optimize ψ, the Q-learning loss

$$L_{DQN}(\psi) = E_U(D)\left[\left(r + \gamma \max_{a'} Q(s,' a'; \psi') - Q(s, a, \psi)\right)^2\right] \tag{5}$$

is minimized for a uniform sample of transitions sampled from D.

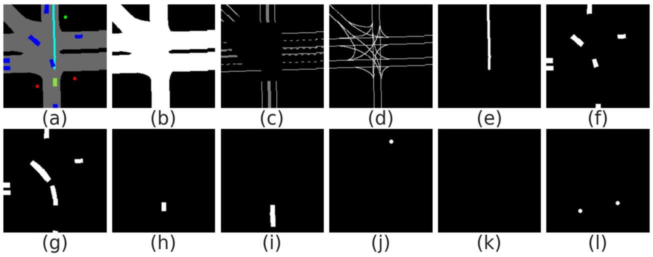


Fig. 2. Illustration of input channels: (a) the bird's eye view rendered RGB image, (b) road area, (c) lane lines, (d) lane centers, (e) route, (f) dynamic object's pose at the current time-step, (g) motion history of dynamic objects, (h) ego car's pose at the current time-step, (i) motion history of the ego car, (j) green traffic light, (k) yellow traffic light, and (l) red traffic light.

IV. Method

The proposed method consists of two parts as illustrated in Fig. 1. First, a low-dimensional latent representation for bird's eye view images is constructed (left). Second, a driving policy is learned using the low-dimensional representation and a hazard signal that is formed from predicted future trajectories of other vehicles (right).

In the following, we first describe the mid-level representation used as the input (Section IV-A). We then explain how the latent representation is learned using a multi-head VAE (Section IV-B). We continue by defining the hazard signal in Section IV-C. Finally, in Section IV-D we describe the policy learning.

A. Mid-Level Input Representation

Information in the mid-level input representation includes road relevant structures, planned route, current and past poses of ego vehicle and other vehicles, and traffic light state. These mid-level inputs are selected based on BEV input representation used in the previous works [4]. Each of these is described using one or more channels of a 11-channel image, as illustrated in Fig. 2.

In simulation experiments, the perception and route information is provided by the CARLA simulator [24].

The information is represented as follows:

1) *Map:* The HD-Map information includes drivable areas, lane lines, center of lane lines, and curbs, as shown in Fig. 2(b–d).
2) *Route*: The route from a starting point to the current destination is assumed to originate from an external planning system such as a standard vehicle navigator. An example is shown in Fig. 2 (e).
3) *Current and Past Poses of Other Vehicles:* The current and past position, orientation, and size of other cars are rendered on two different channels, presented in Fig. 2(f–g). We consider $1.5s$ of motion history for dynamic targets in the scene. This information can come from the perception module with object detection and tracking algorithms and can produce other cars' current and past poses input channels by rendering these data on a single channel image. Note that the data from previous time-steps needs to be

transferred to the coordinate frame of the ego car in the current time step.

4) *Current and Past Ego Vehicle Poses:* The current and past position, orientation, and size of the ego vehicle are rendered on two different channels, described in Fig. 2(h–i). We consider $1.5s$ of motion history for ego vehicle. This information can come from the Localization module.
5) *Traffic Lights:* The information for different traffic light colors are rendered on three different channels for green, red, and yellow as shown in Fig. 2(j–l). Traffic lights' locations are usually available in the HD-Map, and their color can be detected using the perception module.

BEV dimensions are 25 m on the left and right side, 37.5 m in front, and 12.5 m behind the ego car.

B. Learning Latent Representation Using Multi-Head VAE

To constrain the learning of the latent representation, it is enforced to learn three task-relevant factors: scene reconstruction, ego vehicle plan, and predicted motion of other vehicles. To do this, we use an encoder-decoder VAE which takes the 11-channel scene as the input to the encoder $q_\theta(z|x)$. The encoder transforms its input into a low-dimensional latent distribution z. The latent distribution is decoded into the three factors such that each factor is a separate decoder head of the neural network. The architecture for the multi-head VAE is illustrated in Fig. 1 in the "VAE training" box.

The three decoder networks are:

1) *Scene Reconstruction Head:* gets the latent vector z and reconstructs a three-channel bird's eye view RGB image of the scene rgb, rendered based on the 11-channel input similar to e.g. [5]. The neural network for this head is expressed by $p_{\phi_1}(rgb|z)$.
2) *Planning Head:* maps the latent vector to a single-channel bird's eye view planning mask for the ego vehicle for the next 2 s. This head is represented by $p_{\phi_2}(plan|z)$.
3) *Motion Prediction Head:* forecasts the next $2s$ pose of dynamic agents in the scene. It gets the latent-vector z and outputs a bird's eye view mask of other agents' future motion. It is expressed by $p_{\phi_3}(pred|z)$.

The network architecture for each of the decoder heads is the same, but the network weights ϕ_1, ϕ_2, ϕ_3 are different. The multi-head VAE is optimized by minimizing the loss function

$$\begin{aligned} L(\theta,\phi_1,\phi_2,\phi_3) = & -w_1 E_{q_\theta(z|x)} log(p_{\phi_1}(rgb|z)) \\ & - w_2 E_{q_\theta(z|x)} log(p_{\phi_2}(plan|z)) \\ & - w_3 E_{q_\theta(z|x)} log(p_{\phi_3}(pred|z)) \\ & + w_4 D_{kl}(q_\theta(z|x)||p(z)) \end{aligned} \tag{6}$$

with $p(z)$ a normal distribution prior. The optimization is performed by gradient descent using a dataset of known trajectories in order to be able to train the planning and prediction heads.

C. Generation of Hazard Signal

To quantify the degree of conflict between ego car's and other vehicles' trajectories, we calculate a hazard signal h as the log-likelihood that the predicted motion of other vehicles and known planned route are equal under Gaussian noise.

$$h = \log P(route - pred = 0) = \sum_i \log P(route_i - pred_i = 0) \tag{7}$$

where the difference has Gaussian distribution, $(route_i - pred_i) \sim N(0,1)$ for each pixel i independently. By substituting the normal distribution density function to (7), we get

$$h = -\sum_i \frac{(route_i - pred_i)^2}{2} + c. \tag{8}$$

Thus, in the end the hazard signal is the sum of squared difference between pixel values between own route and predicted routes of other agents represented as images.

The calculated hazard signal is used in addition to the latent encoding as input to the RL policy as shown in Fig. 1.

D. Policy Learning Using DQN

In order to evaluate the learned latent space, a DQN policy learning is considered as a down stream task. The DQN policy can be replaced with any other policy learning method. The latent encoding of the current state and the current hazard signal value form the input space to a DQN agent such that $s = (\mu_z, h)$, with μ_z representing the mean of the current latent encoding (see Fig. 1). The action space is a vector of three values for throttle, brake, and steering angle, $a = (a_t, a_b, a_s)$, such that each dimension is discretized into discrete choices.

The reward function is a sum of terms related to collisions, performance, obeying traffic rules (staying in lane, obeying traffic lights), and comfort:

$$r = r_c + r_v + r_o + r_\alpha + r_w + r_{tl} + c \tag{9}$$

where r_c is a collision penalty, r_v is a speed reward term to match desired speed, r_o is an out-of-lane penalty, r_α is the steering angle penalty to improve driving comfort, r_w is penalty for high lateral acceleration, r_{tl} is the penalty term for passing red traffic lights, and c is a constant time penalty [5]. Section V-B reveals more details about the weight for each reward term. The policy is then learned using standard DQN.

V. Experiments

We performed simulation experiments to study the following questions:

1) How much is the effect of auxiliary heads and the generated hazard signal?
2) How our method with auxiliary heads and the hazard signal performs compared to baselines?
3) Can we decrease the dataset size by using auxiliary heads?
4) Can latent space learn about the scene structure, future potential trajectories for the ego agent and other agents?
5) How is the generalization capability of the proposed method compared to other methods?

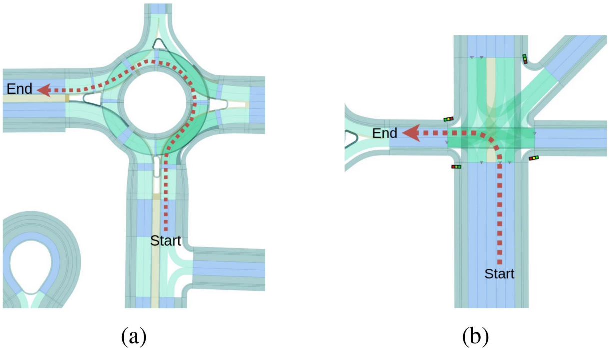


Fig. 3. Scenarios: (a) roundabout and (b) 5-way intersection.

In the following sections, first, the simulation environment and data collection have been described. Then, the implementation details of the multi-head VAE and policy network architectures and also the reward function are detailed. After that, the effect of different heads in the multi-head VAE is evaluated to answer question 1. The proposed method is then compared with state-of-the-art methods to answer question 2. Dataset size effect is evaluated next to answer question 3. Then, a qualitative analysis of the learned latent-vector is done to answer question 4. Finally, to answer question 5, the generalization capability of the proposed approach is evaluated. All experiments were repeated for a set of five identical random seeds for each case.

A. Simulation Environment and Data Collection

To collect the dataset to train the multi-head VAE, we used the CARLA simulator [24], an open-source simulator for autonomous driving, to collect the dataset. We designed two scenarios to collect the dataset: a roundabout and a complex intersection shown in Fig. 3. There are no traffic lights at the roundabout, but there are several at the intersection. For data collection phase for VAE training, we start from a random position around the start positions shown in the figure and select the destination randomly in the city. But for RL training phase, to add stochasticity, start and end positions are considered randomly around the positions shown in the figure.

For the data collection phase, 100 vehicles were spawned randomly in Town number three in CARLA and near the designed scenarios. We used CARLA autopilot mode to drive and collect a dataset.

By recording all agents' poses in each time-step during driving, we had the required information to create the dataset without any manual labeling. To create the ground truth data for motion prediction and planning heads, we used the ego car's pose and other agents in the next time-steps, transformed them into the current ego vehicle's coordinate frame, and rendered them on a binary image. So it did not require manual labeling and was done in a self-supervised manner. The dataset size is around 200 k frames.

For the RL policy learning phase, we used curriculum learning and increased number of agents from zero to 100 to increase the difficulty level step-by-step. We also set 20 percent of cars to ignore traffic lights and have aggressive driving behavior in both data collection for multi-head VAE training and RL policy training. Therefore, there is a need for multi-agent interaction and attention to other cars' movements in the intersection scenario.

B. Implementation Details

Fig. 1 shows the multi-head VAE architecture in the "VAE training" box. The encoder was a ResNet-18 [25] in which the first Convolutional layer had changed to get a BEV $64 \times 64 \times 11$ input tensor. The output feature vector of the ResNet network had the size of 512. Then we used two fully connected layers with 20 neurons to output μ and σ vectors for a Gaussian distribution on latent-space. Then the latent-vector was sampled from this distribution. There were three decoder heads with similar architecture but different weights. We used a network similar to ResNet-18 but in inverse order for decoder heads. The output of planning and prediction heads had one channel, but the reconstruction head had three channels for the output RGB image.

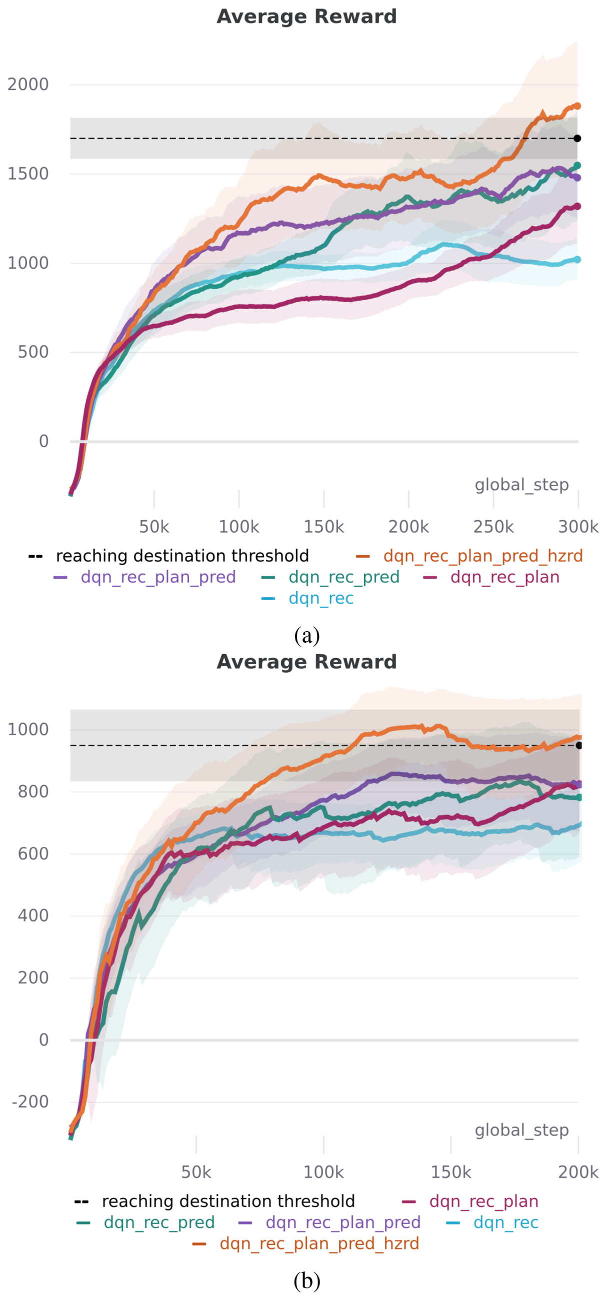
The weights used for the loss terms were $w_1 = 1, w_2 = 1, w_3 = 50$, and $w_4 = 50$ and were found using grid search.

The DQN policy, which is shown in the "RL training" box in Fig. 1, is a network with three fully-connected layers with 128, 64, and nine (output) neurons that gets the input vector with the size of 21 and generate the control commands. Output control commands of the policy network are: (1) acceleration $\in \{-0.3, 0, 0.3\}$ which positive value is for throttle, negative value for brake, and zero for braking and throttle, and (2) steering $\in \{-0.15, 0, 0.15\}$.

The terms of the reward function (9) were set as follows: The collision penalty r_c was set to $r_c = -200$ if there is a collision, otherwise $r_c = 0$. The speed reward r_v was set to the ego vehicle's speed, 10 m/s, and if $r_v > 10\ m/s$ then a penalty of -10 will be added to the reward function. The going out of lane penalty r_o was set to $r_o = -1$ if the distance between the ego vehicle's position and the planned route is more than 2.5 m, otherwise $r_o = 0$. The steering angle penalty r_α was set to $r_\alpha = -0.5 \times \alpha^2$. The high lateral acceleration penalty r_w was set to $r_w = -0.2 \times w$, where w is the lateral acceleration. The passing red traffic lights penalty term r_{tl} (only in the intersection scenario) was set to $r_{tl} = -10$ if the ego vehicle passes a red traffic light, otherwise $r_{tl} = 0$. Finally, the constant time penalty c was set to -0.1 to prevent the car from stopping.

C. Effect of Different Heads and the Hazard Signal

To answer question 1 and see the effect of each head on the learned latent-vector and the proposed hazard signal, a downstream RL task, DQN agent, was considered. The VAE with a reconstruction head proposed in [5] was considered as the baseline (dqn_rec). Four other models are trained to analyze the effect of each head and the hazard signal: (a) VAE with reconstruction and planning heads (dqn_rec_plan), (b) VAE with reconstruction and motion prediction heads (dqn_rec_pred), (c) VAE with reconstruction, planning, and motion prediction heads (dqn_rec_plan_pred), and (d) VAE with auxiliary hazard signal in addition to three heads in case c (dqn_rec_plan_pred_hzrd).



(a)

(b)

Fig. 4. Analysis of the effect of different VAE decoder heads and the hazard signal on the down-stream RL task's performance in two scenarios: (a) roundabout and (b) 5-way intersection.

All networks are trained using the full dataset. Then the encoder of each case was used as the feature extractor for the DQN agent. The encoder weights were fixed, and we only trained DQN weights in this phase. For case d, the DQN agent also used the hazard signal. Fig. 1 shows the full model used in case d.

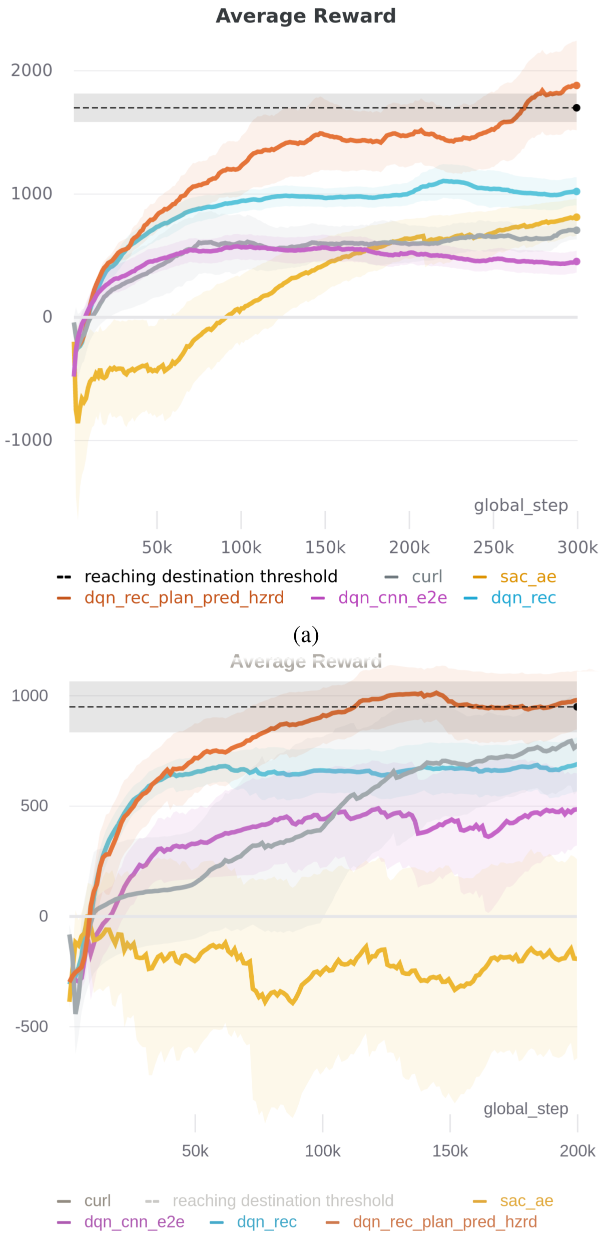
Fig. 4 shows the comparison. Our proposed method (dqn_rec_plan_pred_hzrd) outperformed all other cases, achieved a higher mean reward, and solved the task in both roundabout and intersection scenarios. The baseline method (dqn_rec) cannot solve the task, but by adding the planning head, the agent can finally solve the task in some episodes and reach the destination. We can also see the effect of planning and prediction heads by comparing the VAE with reconstruction and prediction heads (dqn_rec_pred) against the VAE with reconstruction and planning heads (dqn_rec_plan). It shows that the motion prediction head that has information about other dynamic agents' future motion can be more beneficial than the planning head and improve the down-stream task's performance by a large margin. We can also see that using both planning and prediction heads in addition to the reconstruction head (dqn_rec_plan_pred) can outperform using planning and prediction heads separately. Note that reaching the destination threshold shown in the charts isn't an exact value, but rather an area resulting from several runs of the full method (dqn_rec_plan_pred_hzrd) with different random seeds.

D. Comparison to Baselines

This section compares our method with four baseline approaches in order to answer question 2. The proposed multi-head VAE with three decoder heads and also the calculated auxiliary hazard signal as input to the DQN agent (dqn_rec_plan_pred_hzrd), shown in Fig. 1, was compared with: (a) an encoder and a policy network which are trained end-to-end using RL, similar to the DQN work on Atari games [26], to get the BEV input tensor and generate control commands (dqn_cnn_e2e), (b) a single-head VAE with a reconstruction head (dqn_rec) [5] which was pre-trained using supervised learning and then the weights were fixed, and the encoder was used as the feature extractor for BEV input tensor, (c) SAC-AE (sac_ae) [27], a variant of the standard SAC, which is more sample efficient than standard SAC and performs better in environments with image inputs, and (d) CURL (curl) [28] which uses a contrastive unsupervised representation learning approach for RL algorithms. Then the DQN agent used the encoded latent-vector as input to generate control commands.

Fig. 5 shows the comparison in two scenarios in terms of average reward. As can be seen, our proposed method (dqn_rec_plan_pred_hzrd) could solve both tasks and exceeded other methods by a considerable margin. Following our method, none of the baselines could solve the roundabout scenario, and VAE with reconstruction head (dqn_rec) performed better than other baselines. In the intersection environment, CURL (curl) worked very well and was able to accomplish the task in some runs and outperformed other methods. The next best was the VAE with reconstruction head (dqn_rec). The SAC-AE (sac_ae) also performed poorly in this environment.

We ran the trained models in two different scenarios to compare different methods in terms of crash percentage and success rate. The reported results in [5] for DDQN [29] algorithm show 0% success rate of reaching the goal position in the roundabout scenario and we found the same results for the dqn_rec method. However, our method performed better and reached the success rate of $5 \pm 2\%$ in three different runs, each consisting of 100 episodes. But it was not sufficient to get a good understanding of the performance differences. Therefore, instead of using DQN for both lateral and longitudinal control, we decided to use DQN for speed control only and use a PID controller for steering



(a)

(b)

Fig. 5. Comparison of the proposed method against baselines in two scenarios: (a) roundabout and (b) 5-way intersection.

TABLE I
THE CRASH PERCENTAGE IN 3 RUNS, 100 EPISODES EACH, FOR TWO SCENARIOS: ROUNDABOUT AND 5-WAY INTERSECTION

Scenario	*Method*	*Number of Cars* *10*	*50*	*100*
Roundabout	dqn_cnn_e2e	33 ± 3.5	48 ± 4	50 ± 3.2
	sac_ae	25 ± 2.16	28 ± 3.3	31.7 ± 2.5
	curl	29.4 ± 1.7	42 ± 1.7	47 ± 3
	dqn_rec	25 ± 4.2	38 ± 3.8	41 ± 4.2
	ours	**6 ± 3.4**	**15.6 ± 4.1**	**18.6 ± 5.4**
5-way Intersection	dqn_cnn_e2e	33.7 ± 2.5	39.7 ± 2.5	47 ± 1.7
	sac_ae	27.7 ± 1.3	35.7 ± 0.5	39.7 ± 1.3
	curl	25.7 ± 1.3	35.4 ± 1	41.7 ± 1.3
	dqn_rec	26.7 ± 0.5	33.7 ± 1.7	40 ± 1
	ours	**13 ± 2.2**	**20.7 ± 1.7**	**22.4 ± 2.1**

TABLE II
THE SUCCESS RATE IN 3 RUNS, 100 EPISODES EACH, FOR TWO SCENARIOS: ROUNDABOUT AND 5-WAY INTERSECTION

Scenario	*Method*	*Number of Cars* *10*	*50*	*100*
Roundabout	dqn_cnn_e2e	59 ± 4	49 ± 3.1	47 ± 2.4
	sac_ae	73.7 ± 1.7	71 ± 3	66.7 ± 1.7
	curl	60.4 ± 2.9	45 ± 1.7	48 ± 3
	dqn_rec	65 ± 4.4	54.7 ± 3.2	52 ± 3.6
	ours	**91 ± 3.3**	**82.4 ± 3.5**	**79.7 ± 4.8**
5-way Intersection	dqn_cnn_e2e	66.7 ± 1.3	59 ± 2.5	52.7 ± 2.1
	sac_ae	70.7 ± 1.3	64.7 ± 1.3	58.7 ± 1
	curl	72.3 ± 1	63 ± 0.9	57.6 ± 1.7
	dqn_rec	71.3 ± 1.3	65.4 ± 0.5	59.4 ± 1.3
	ours	**84.7 ± 3.4**	**78 ± 1.7**	**75.7 ± 3.1**

TABLE III
THE RATE OF PASSING TRAFFIC LIGHT IN 3 RUNS, 100 EPISODES EACH, FOR THE 5-WAY INTERSECTION SCENARIO WHEN 100 CARS ARE SPAWNED IN THE CITY

Method				
dqn_cnn_e2e	sac_ae	curl	dqn_rec	ours
50 ± 5	31.4 ± 2	34.4 ± 1.3	38.4 ± 1.7	**21 ± 2.5**

control. We only did this for the results reported in the Tables I, II, and III, not for the charts. Also, note that the reported numbers for crash and success in each case do not necessarily add up to 100% and the ego car also can reach the maximum episode time. Additionally, we set 50% of cars to ignore traffic lights in the 5-way intersection scenario to make a more complex interactive scenario.

Tables Iand II show the crash percentage and success rate for different methods in two scenarios for different numbers of spawned cars in the city, respectively. When we reduced the number of spawned cars, all methods performed better. In both scenarios, our method had a lower crash percentage and a higher success rate. Following that, in the roundabout scenario, SAC-AE (sac_ae) and then VAE with reconstruction head (dqn_rec) performed better than the others.

After our method, SAC-AE (sac_ae), CURL (curl), and VAE with reconstruction head (dqn_rec) all performed almost identically in the 5-way intersection scenario. This emphasizes the importance of motion prediction and hazard signals that assist ego cars to handle interactions with aggressive vehicles at an intersection. Note that the episode was not terminated when the car passed a red traffic light in the intersection scenario, and the reported success rate numbers indicate the car reached the goal position even when it passed a red traffic light.

To compare the performance of methods in terms of respecting traffic lights, we evaluated them in the 5-way intersection scenario. The results are detailed in Table III. As we didn't see much difference in the performance of different algorithms with a lower number of spawned cars, we only reported the case for 100 spawned cars in the city.

Our method outperformed others by a large margin, as shown in Table III. This is because of the motion planning head, which has to imitate the driving experience of expert drivers in order to learn the correct driving response in this situation, red traffic light.

E. Effect of the Dataset Size

In this part we try to investigate question 3. Several models were trained using different dataset sizes to show the benefit of using multiple decoder heads and the hazard signal on the learned latent-vector and the down-stream task's performance. The full model, shown in Fig. 1, was trained using the full dataset (dqn_rec_plan_pred_hzrd_full), half of the dataset (dqn_rec_plan_pred_hzrd_half), and a quarter of the dataset (dqn_rec_plan_pred_hzrd_quart). We then compared them against the baseline VAE [5] with just a reconstruction decoder (dqn_rec) trained on the full dataset. For our proposed method, a DQN agent was trained to get the concatenation of the latent-vector and the hazard signal and output control commands. Note that for the baseline, there is no hazard signal, as there is no prediction head.

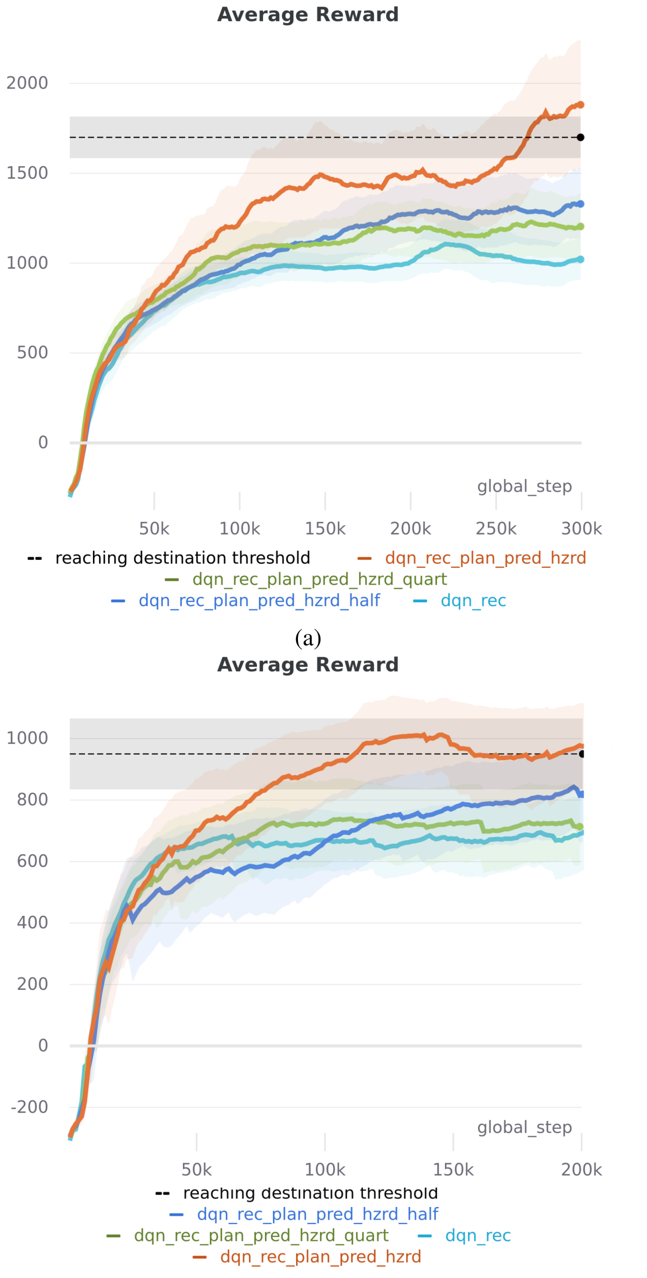
The results, shown in Fig. 6 for two scenarios, present better performance of our proposed method even when it was trained with a quarter of the dataset. The case trained with half of the dataset can solve the intersection scenario for some runs, but the model trained with a quarter of the dataset cannot; however, it outperformed the baseline.

F. Qualitative Analysis

In this section, we explore the latent-space to answer question 4. The learned latent-vector has 20 elements some of which have almost zero standard deviation and changing their value do not cause any difference in outputs. We tested several latent space sizes. Low latent sizes push the network to mix information in latent elements. So, the disentanglement of latent elements will be lower, and by changing the latent element's value, multiple elements will change in the reconstruction heads. On the other hand, by considering higher latent space sizes, most of the elements will be zero, and it doesn't guarantee to have more disentanglement in latent space. We selected 20 as a number in between, but we have some latent elements close to zero with a low standard deviation. Note that we do not try to solve the disentanglement problem here. Fig. 7 shows the results for three latent elements. We used a sample input from dataset and encoded it to get the latent-vector. Then for each latent element i, we changed its value from $\mu_i - \sigma_i$ to $\mu_i + \sigma_i$ while other latent elements were fixed. By exploring latent-space, we see changes in different factors in the scene such as road structure, traffic light color, route, dynamic objects pose, planning, and motion prediction. This shows the meaningfulness of the learned latent-space. We can also see the harmony between different heads' outputs when changing latent values.

G. Generalization Analysis

In order to answer question 5, we evaluated the generalization capability of the proposed method by running the trained model on the 5-way intersection scenario in a new 4-way intersection scenario, shown in Fig 8, without any further training and fine-tuning.



(a)

(b)

Fig. 6. Analysis of the effect of different dataset sizes on the down-stream RL task in two scenarios: (a) roundabout and (b) 5-way intersection.

Due to no further training on the new scenario, the performance of all algorithms was lower than the 5-way intersection scenario in terms of crash percentage and success rate metrics. Nevertheless, our method outperformed other methods, as shown in Table IV. Furthermore, we realized that sometimes the car sticks and doesn't move at all, which can be due to changing the scene and out of distribution input image. The results also reveal that the passing traffic light metric does not change that much in the new 4-way intersection, which illustrates that the network has learned about the traffic light colors and rules and

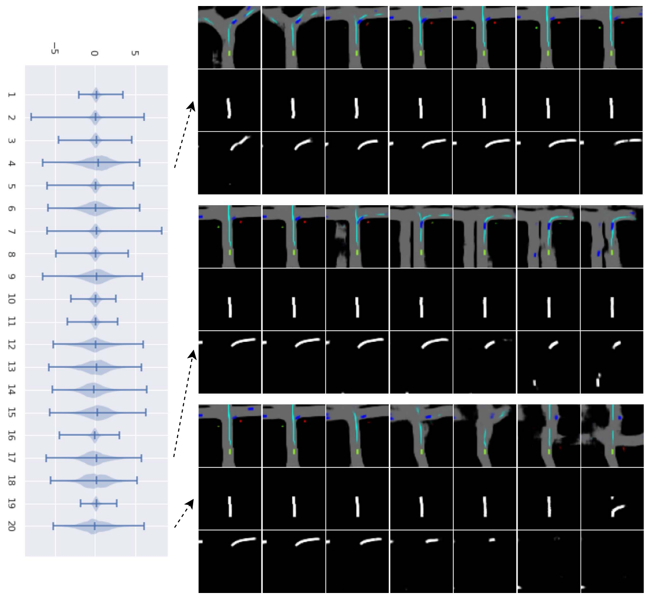


Fig. 7. Latent-space exploration. The left plot shows the violin-plot of the learned latent-space – mean, standard deviation, min, and max – for training dataset. The right hand side figures show outputs of the multi-head VAE when the ith latent element is changed from $\mu_i - \sigma_i$ to $\mu_i + \sigma_i$. μ_i, σ_i are mean and standard deviation of the ith element for the whole training dataset. We show only the results of exploring three latent elements: 4th, 17th, and 20th. A video with more examples of latent space exploration can be found at: https: //youtu.be/5Tk8j6LXBmA*https://youtu.be/5Tk8j6LXBmA*.

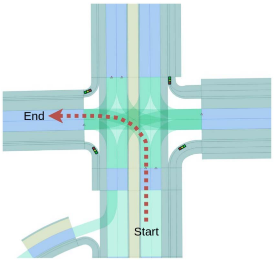


Fig. 8. The 4-way intersection for generalization analysis.

TABLE IV

THE CRASH PERCENTAGE, SUCCESS RATE, AND RATE OF TRAFFIC LIGHT PASSING IN 3 RUNS, 100 EPISODES EACH, FOR THE TRAINED MODEL ON THE 5-WAY INTERSECTION SCENARIO AND EVALUATED IN THE 4-WAY INTERSECTION SCENARIO

Metric	***Method***	***Number of Cars***		
		10	*50*	*100*
Crash Percentage	dqn_cnn_e2e	40 ± 2.8	45 ± 2.8	55 ± 2.5
	sac_ae	31 ± 3.2	42 ± 2.1	53 ± 3.1
	curl	32 ± 3.5	39 ± 2.2	51 ± 2.6
	dqn_rec	30 ± 2.6	38 ± 2.5	49 ± 3.9
	ours	**20 ± 2.5**	**27 ± 2.1**	**33 ± 3.6**
Success Rate	dqn_cnn_e2e	45 ± 2.3	40 ± 3.2	31 ± 3.1
	sac_ae	50 ± 2.8	42 ± 2.6	34 ± 2.8
	curl	52 ± 3.2	43 ± 2.7	36 ± 3.4
	dqn_rec	54 ± 2.9	48 ± 3.2	45 ± 3.1
	ours	**72 ± 2.2**	**63 ± 2.1**	**56 ± 2.4**
Passing Traffic Light	dqn_cnn_e2e	-	-	54 ± 2.5
	sac_ae	-	-	38 ± 3.6
	curl	-	-	42 ± 2.9
	dqn_rec	-	-	42 ± 3.2
	ours	-	-	**26 ± 2.3**

can generalize to new scenarios. But the new road and scene structure causes weaker performance in the crash percentage and success rate.

VI. CONCLUSION

Machine learning presents an important avenue to handle the complexity of situations in autonomous driving but its applicability is significantly hindered by the huge amounts of data needed for learning. In this paper, we proposed a multi-head VAE network with a bird's eye view input and task-relevant heads to learn efficient latent-space representations. These representations can then be used to train a driving policy more efficiently. Experimental comparison against baselines in two scenarios showed that the use of task-relevant heads in representation learning improves policy learning in four ways: the policy quality is better, the learning converges faster, policy learning requires less data, and the learned policy generalizes better to new scenarios. The proposed approach can be easily extended to other task-relevant factors if they can be encoded as images.

In real traffic environments, multiple vehicles with different driving policies interact with each other. In this case, generalization capability of a representation would likely benefit from disentanglement of that representation with respect to the different actors. The use of multi-task learning similar to this paper thus seems to provide a valuable avenue towards autonomous driving in complex traffic conditions.

ACKNOWLEDGMENT

The authors would like to thank CSC-IT Center for Science, Finland, for generous computational resources and also would like to thank the computational resources provided by the Aalto Science-IT project.

REFERENCES

[1] A. Kendall *et al.*, "Learning to drive in a day," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 8248–8254.

[2] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, to be published, doi: 10.1109/TITS.2020.3046646.

[3] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," in *Proc. 2nd Annu. Conf. Robot Learn., (CoRL), Zürich, Switzerland, 29–31*, ser. Proc. Mach. Learning Res. (PMLR), vol. 87, 2018, pp. 1–15. [Online]. Available: http://proceedings.mlr.press/v87/mueller18a.html

[4] M. Bansal, A. Krizhevsky, and A. S. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," in *Robot, Sci. Syst. XV*, Univ. Freiburg, Freiburg, Germany, vol. 15, 2019.

[5] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2019, pp. 2765–2771.

[6] E. Kargar and V. Kyrki, "Vision transformer for learning driving policies in complex multi-agent environments," 2021, *arXiv:2109.06514*.

[7] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *Proc. IEEE 21st Int. Conf. Intell. Transp. Syst.*, 2018, pp. 2156–2162.

[8] P. Wang and C.-Y. Chan, "Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst.*, 2017, pp. 1–6.

[9] T. Shi, P. Wang, X. Cheng, C.-Y. Chan, and D. Huang, "Driving decision and control for automated lane change behavior based on deep reinforcement learning," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2019, pp. 2895–2900.

[10] S. Kuutti, R. Bowden, H. Joshi, R. de Temple, and S. Fallah, "End-to-end reinforcement learning for autonomous longitudinal control using advantage actor critic with temporal context," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2019, pp. 2456–2462.

[11] N. Deshpande and A. Spalanzani, "Deep reinforcement learning based vehicle navigation amongst pedestrians using a grid-based state representation," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2019, pp. 2081–2086.

[12] S.-H. Kong, I. M. A. Nahrendra, and D.-H. Paek, "Enhanced off-policy reinforcement learning with focused experience replay," *IEEE Access*, vol. 9, pp. 93152–93164, 2021.

[13] B. R. Kiran *et al.*, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, to be published, doi: 10.1109/TITS.2021.3054625.

[14] R. Bonatti, R. Madaan, V. Vineet, S. Scherer, and A. Kapoor, "Learning visuomotor policies for aerial navigation using cross-modal representations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 1637–1644.

[15] M. Itkina, K. Driggs-Campbell, and M. J. Kochenderfer, "Dynamic environment prediction in urban scenes using recurrent representation learning," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2019, pp. 2052–2059.

[16] H. Cui *et al.*, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 2090–2096.

[17] H. Cui *et al.*, "Deep kinematic models for kinematically feasible vehicle trajectory predictions," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, IEEE, 2020, pp. 10 563–10 569.

[18] F.-C. Chou *et al.*, "Predicting motion of vulnerable road users using high-definition maps and efficient convnets," in *Proc. IEEE Intell. Veh. Symp. 4th*, 2020, pp. 1655–1662.

[19] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 14 074–14 083.

[20] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," in *Proc. Conf. Robot Learn.*, 2019, pp. 86–99.

[21] J. Hawke *et al.*, "Urban driving with conditional imitation learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 251–257.

[22] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. 2nd Int. Conf. Learn. Representations, ICLR*, Banff, AB, Canada, Apr. 2014.

[23] I. Higgins *et al.*, "Beta-VAE: Learning basic visual concepts with a constrained variational framework," in *ICLR*, 2017.

[24] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. López, and V. Koltun, "Carla: An open urban driving simulator," in *Proc. Conf. Robot Learn.*, 2017, pp. 1–16.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[26] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[27] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, pp. 10 674–10 681, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17276

[28] M. Laskin, A. Srinivas, and P. Abbeel, "CURL: Contrastive unsupervised representations for reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5639–5650.

[29] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, ser. AAAI'16. AAAI Press, 2016, 2016, pp. 2094–2100.

Eshagh Kargar received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Tehran, Tehran, Iran, in 2013 and 2016, respectively. In 2019, he joined the Intelligent Robotics Group, Aalto University, Helsinki, Finland, where he is currently working toward the Doctoral degree. His primary research interests include robotic perception, decision making, and learning.

Ville Kyrki (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from the Lappeenranta University of Technology, Lappeenranta, Finland, in 1999 and 2002, respectively.

From 2003 to 2004, he was a Postdoctoral Fellow with the Royal Institute of Technology, Stockholm, Sweden, after which he returned to the Lappeenranta University of Technology, holding various positions from 2003 to 2009. During 2009–2012, he was a Professor of computer science with the Lappeenranta University of Technology. Since 2012, he has been an Associate Professor of intelligent mobile machines with Aalto University, Helsinki, Finland. His primary research interests include robotic perception, decision making, and learning.

Dr. Kyrki is a Fellow of the Academy of Engineering Sciences, Finland, and a Member of Finnish Robotics Society and the Finnish Society of Automation. He was the Chair and Vice Chair of the IEEE Finland Section Jt. Chapter of CS, RA, and SMC Societies during 2012–2015 and 2015–2016, respectively, a Treasurer of IEEE Finland Section during 2012–2013, and the Co-Chair of IEEE RAS TC in Computer and Robot Vision during 2009–2013. From 2014 to 2017, he was an Associate Editor for the IEEE TRANSACTIONS ON ROBOTICS.