Improving Autonomous Driving Policy Generalization via Auxiliary Tasks and Latent Modeling

Hemanth Manjunatha
School of Aerospace Engineering
Institute for Robotics and Intelligent Machines
Georgia Institute of Technology
Atlanta, Georgia
hmanjunatha6@gatech.edu

Mahdi Ghanei
School of Aerospace Engineering
Institute for Robotics and Intelligent Machines
Georgia Institute of Technology
Atlanta, Georgia
mahdi.ghanei@gatech.edu

Andrey Pak
School of Aerospace Engineering
Institute for Robotics and Intelligent Machines
Georgia Institute of Technology
Atlanta, Georgia
andrey.pak@gatech.edu

Panagiotis Tsiotras
School of Aerospace Engineering
Institute for Robotics and Intelligent Machines
Georgia Institute of Technology
Atlanta, Georgia
tsiotras@gatech.edu

Abstract

With the development of deep representation learning, reinforcement learning (RL) has become a powerful framework for automated driving tasks capable of learning complex policies in high-dimensional environments. However, one of the most critical criteria to deploy learned policies in real-world tasks is generalizing to unseen situations at deployment time, avoiding over-fitting to the training environments. Studying this is vital if we use the RL algorithms for decision-making in real-world scenarios, where the environment will be diverse, dynamic, and unpredictable, like autonomous driving. In this work, we propose a novel architecture that combines different information about the driving conditions and the environment to inform the RL agent in the form of latent vectors. For instance, while the host vehicle is near a traffic sign, it is desirable that the RL agent knows about the traffic, distance to the intersection, etc., to take appropriate actions. This will allow us to develop a robust model by overparameterizing the policy network in a structured manner. Although the use of latent representations for RL is quite common, the use of different latent representations and selectively combining them using self-attention as proposed in this work is novel. The basic premise here is that different latent representations provide parallel, complementary pathways that parameterize the actor/critic RL network. In essence, while the value (critic) network tries to estimate $Q_\theta(x, a)$ where (x, a) is the state-action pair, our network will try to estimate $Q_\theta(\ell, \eta, \alpha, a)$ where ℓ, η and α, are the additional latent variables representing different aspects of driving. Preliminary results show the efficacy of using different latent vectors and combining them in a structured manner to derive a driving policy with improved generalization.

Keywords: Reinforcement Learning, Autonomous Driving, Generalisation, Latent Modeling, Auxiliary Tasks, Self Attention

1 Motivation and Objective

Reinforcement Learning (RL) has been widely used for decision-making with great success over the past 20 years and recently it is making its way to the domain of autonomous vehicles. State-of-the-art algorithms such as Deep Deterministic Policy Gradient (DDPG) and Soft Actor-Critic (SAC) are being applied to complex tasks such as robot control and autonomous driving. The greatest success of RL algorithms has been mainly in game environments where the failure to make good decisions does not usually incur a high cost. However, the same cannot be said for autonomous vehicles where the decision's generalizability and robustness are paramount in anomalous situations. By the latter, we mean the ability of the system/agent to perform reasonably well even in cases of failures or unforeseeable situations that have never been encountered before. For example, the performance should not drop dramatically when the car encounters a new map (with possibly different weather conditions) where the RL agent has not been trained. In this direction, a significant body of work has accumulated [11, 9] in the domain of supervised learning (e.g., MNIST, CIFAR10) studying generalization in deep learning networks. However, the generalizability is less explored [12, 4] in reinforcement learning, particularly for autonomous driving. In this regard, this work explores a popular empirical observation that suggests that overparameterized neural networks improve both optimization and generalization [1], in the context of autonomous driving using RL.

Deep learning (DL) has seen tremendous success in computer vision, speech recognition, natural language processing, audio recognition, and bioinformatics [16]. Surprisingly, even though the number of parameters in Deep Learning (DL) models is significantly more than the number of training examples, the models exhibit good generalization and lower error rate in test cases [15, 10]. This empirical observation appears to contradict the traditional learning theory [2]. Consequently, a significant amount of effort has been devoted to investigating theoretical properties of the deep learning models [2, 8]. Recently, Casper et al. [3] explored the emergence of prunable and redundant units in relation to the generalization ability of the deep neural networks. The authors observed that the prunable and redundant units grow at a rate outpacing the model size.

Inspired by these works that relate overparameterization and deep learning, we explore whether overparameterization and redundancy can also improve the robustness in a reinforcement learning setting. This is particularly interesting because most of the analysis on generalization is explored in the supervised domain. At the same time, there is no direct way to convert RL problems into a supervised problem. However, there is a crucial difference in the proposed work. While previous works [15, 2, 3] focus on the overparameterization by increasing the size of the learnable parameters within the network, here the overparameterization arises in the form of redundancy in a structured manner. We also use a self-attention mechanism to combine different information effectively.

2 Method

To study the effect of overparameterization in a structured way, we introduce a novel architecture with self-attention that explicitly combines redundant information as shown in Figure 1 leading to overparameterization. The architecture consists of five sub-networks: *base network, future latent vector prediction network, network trained using auxiliary tasks, action prediction network*, and *value network*. The implementation details of each network are discussed in the subsequent text. The future latent vector prediction network and network trained using auxiliary tasks are trained offline, and the weights are frozen for RL training. Only the base network along with the action prediction network and value network (critique network) are trained in the RL setting. All the three network receive a stack of 4 images ($x_{t-3} \dots x_t$) at time instance t. The base network compresses the input images to an embedding η_t through a series of convolution operations. The future latent vector prediction network predicts the embedding ℓ_{t+1}, which signifies the future information. The structure of the future latent vector prediction network is inspired by world models [6]. The rationale for incorporating future latent vectors is to inform the policy network about the influence of present action and serve as a predictive model. The network trained on the auxiliary tasks embeds the input images to α_t through convolution operation. Additionally, the network also uses auxiliary task information along with input images. The reasoning for using auxiliary tasks is to provide rich information (e.g., traffic lights, crossings, obstacles) to learn the actions better. A self-attention mechanism is used to combine the three embeddings ($\ell_{t+1},\ \eta_t,$ and α_t) together to form a feature vector which is used for by the action network and value network.

Further, to support goal-oriented actions, we introduce a gating unit that selects different branches of the action network depending on the global routing commands [7]. The sub-networks in the action network are regular multi-layer networks whose output are continuous throttle, steer, and brake commands. Similarly, the value network receives the augmented feature vector, action, and the global planner command to provide the value approximation.

2.1 Base Network

The base network consists of general convolution neural network architecture without any fully-connected layers. Given the expert driving action h_t at the state x_t and the routing command c_t, we can pre-train the policy network shown in

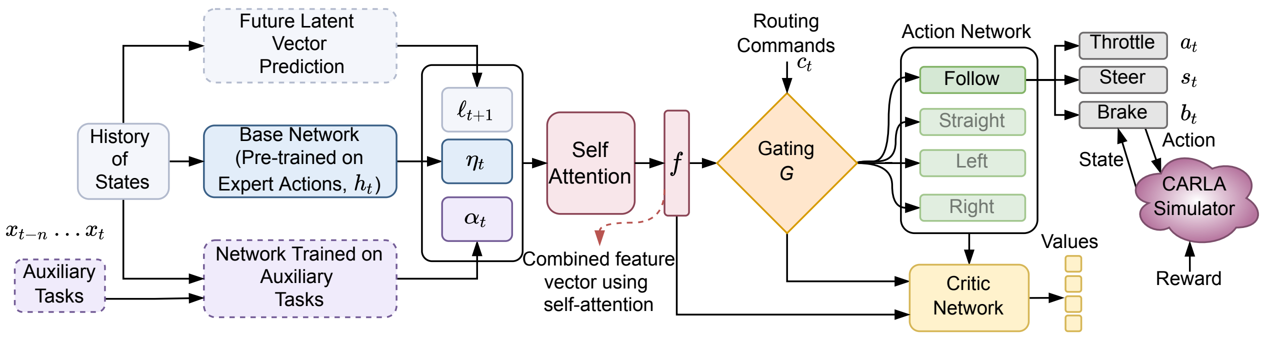

Figure 1: The architecture consists of five sub-networks: base network, future latent vector prediction network, network trained using auxiliary tasks, action prediction network, and value network. The future latent vector prediction network and network trained using auxiliary tasks are trained offline, and the parameters are frozen during RL training. The embeddings from the base network, future latent vector prediction network, and network trained using auxiliary tasks are augmented together with self-attention to form the input for the action prediction network. The network uses the routing commands from a global path-planner to select a sub-network in action prediction. The output of these sub-networks is the throttle, steer, and brake.

Figure 1 via basic imitation learning [7] to mimic an expert driver. Note that, for imitation learning, only the *base network* in Figure 1 will be trained. The future latent vector prediction network and the auxiliary tasks network will be trained separately and augmented with the base network for reinforcement learning. The control command c_t is introduced to handle goal-oriented behavior starting from an initial location to reach the final destination (similar to [7]). The command c_t is a categorical variable that controls the selective branch activation via the gating function $G(c_t)$, where c_t can be one of four different commands, i.e., follow the lane (Follow), drive straight at the next intersection (Straight), turn left at the next intersection (TurnLeft), and turn right at the next intersection (TurnRight). Four policy branches are specifically learned to encode the distinct hidden knowledge for each case and thus selectively used for action prediction. For loss function, we can directly use mean squared error between predicted and expert actions given as:

$$\mathcal{L}(\hat{h}_t, h_t) = \|\hat{s}_t - s_t\|^2 + \|\hat{a}_t - a_t\|^2 + \|\hat{b}_t - b_t\|^2, \tag{1}$$

where the loss function $\mathcal{L}$ is the weighted (here for simplicity taken as 1) summations of L2 losses for three predicted actions $\hat{h}_t$. The notion of re-training the pre-trained model in a reinforcement learning setting can be considered analogous to residual policy learning [13]. Here, we start with a fixed policy, and then learn a residual policy to modify the fixed policy for a more complex situation.

2.2 Future Latent Vector Prediction Network

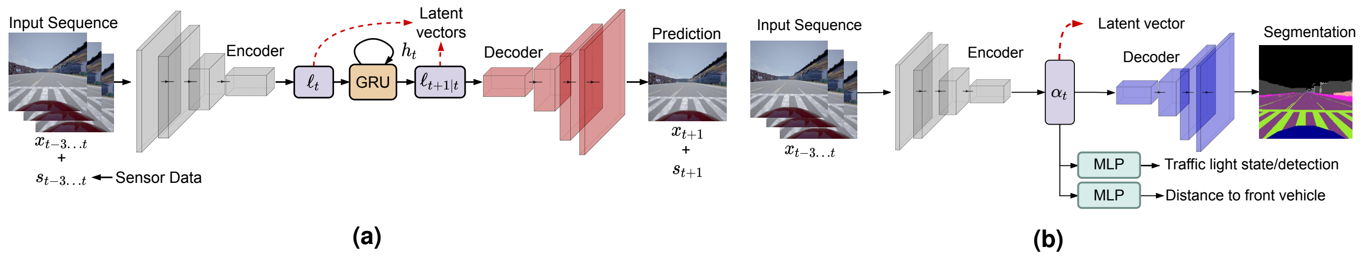

Figure 2: (a) Future latent vector prediction network. A sequence of traffic states is first encoded into latent space, propagated through the recurrent network and outputs the estimate of the latent space at time step t_{n+1}. (b) A Network trained to predict auxiliary tasks such as traffic light state, distance to the front vehicle.

We propose a novel neural architecture to learn the environment dynamics/world model from expert demonstrations (Figure 2a). By "environment" here we mean a succinct description of the interaction between the autonomous (host) vehicle and the surrounding vehicles in traffic. The proposed combined architecture for future latent vector prediction is shown in Figure 2. Given a history of states $x_0, \dots, x_n$, their latent representations $\ell_0, \dots, \ell_n$, and the corresponding sensor data $s_0, \dots, s_n$ (e.g., radar or Lidar scans, images from on-board cameras, IMU measurements, etc.), we wish to predict x_{n+1}, ℓ_{n+1} and s_{n+1}. In other words, the network learns the probability distribution $\mathbb{P}\left(x_{n+1}, \ell_{n+1}, s_{n+1} \mid x_{0:n}, \ell_{0:n}, s_{0:n}\right)$. Here, instead of the sensor readings $s_{0:n}$, we can also use the human-driver actions $a_{0:n}$. As shown in Figure 2, the state x_i is encoded into the latent space $\ell_i = E(x_i)$, where E is the encoder. Note that the encoder/decoder can be any architecture of a typical convolutional neural network or a recurrent neural network. The encoded vector ℓ_i is used as an input for the gated recurrent unit (GRU) block, whose output is the next time step latent space vector $\ell_{i+1|i}$. The predicted latent state $\ell_{i+1|i}$ is used as a hidden state for the next time step, as well as an input for reconstructing the next time step state $x_{i+1} = D(\ell_{i+1|i})$, where D is a decoder. Thus, the predicted latent vector ℓ_{i+1} at time step $i + 1$ not only depends on ℓ_i, but also on ℓ_{i-1}. The overall loss function for the single-step prediction can be broken down into three terms given by equation (2) below

$$\mathcal{L}_{\text{total}} = \underbrace{\mathcal{L}_2(x_{0:n+1}, \hat{x}_{0:n+1})}_{\text{State Prediction}} + \underbrace{\mathcal{L}_2(\ell_{n+1|n}, \ell_n)}_{\text{Latent prediction}} + \underbrace{\mathcal{L}_2(s_{n+1|n}, s_n)}_{\text{Sensor prediction}}, \tag{2}$$

where ℓ_n, $\ell_{n+1|n}$ denote the encoded latent variable at step n and the conditioned latent prediction one step further, x_n, $\hat{x}_n$ are the source and reconstructed states, respectively, and s_n, $s_{n+1|n}$ are the current and predicted sensor measurements. Note that if the states include images, we can use image-relevant loss functions [14]. The intuition behind combining and training the Autoencoder (AE) and Recurrent Neural Network (RNN) simultaneously stems from the ability to incorporate additional continuity constraints in the latent vector during learning, which enables a smooth transition between latent vectors. Additionally, sharing reduces the number of trainable parameters, thus facilitating faster training times while using less computational resources.

2.3 Auxiliary Task Prediction Network

In a sophisticated task such as autonomous driving, end-to-end learning often fails to train good policies. Clearly, humans instinctively use a much richer and hierarchical task decomposition when driving. Rather than directly thinking about the final throttle and steering values, humans try first to keep the car within the lanes, maintain the same distance from the front vehicle, and gradually reduce their speed when they see a red light. We attempt to employ the same pipeline for learning an autonomous driving policy. Using auxiliary tasks, which are human-designed intermediate objectives for achieving the end-goal, we obtain an immensely feature-rich latent vector used to train the policy.

Our model is shown in Figure 2b, which is trained to do semantic segmentation of the driving scene and predict auxiliary tasks such as the estimated distance to the front vehicle and the status of a traffic light. The loss function used for training is a weighted summation of all the cross-entropy losses used for the individual tasks. We train the driving policy consisting of the pre-trained backbone Convolutional Neural Network (CNN) with frozen weights and a small subsequent Multilayer perceptron (MLP) to map the latent vector to discretized driving actions.

3 Preliminary Results and Discussion

This section provides preliminary results two sub-networks: *future latent vector prediction network and network trained using auxiliary tasks*. We validate the efficacy of the proposed models through imitation learning. In the experiments, we use CARLA [5] simulation platform. For imitation learning, given N expert driving sequences $ED_i, i \in (1, \ldots, N)$ with corresponding observation frames $I_{i,t}$, we learn a deterministic policy using a network parameterized by θ to mimic the expert actions. The action space $A_{i,t}$ contains two continuous actions: steering angle and acceleration. However, for imitation learning, we discretized the continuous values into three levels. The steering angle assumes the values $[-0.2, 0.0, 0.2]$ rad and acceleration assumes the values $[-3, 0.0, 3]$ $\mathrm{m/s^2}$ values. Imitation learning using future latent vectors is set up as a classification problem with 9 classes resulting from different combinations of the previous discrete steering and acceleration values. However, for imitation learning using latent vectors trained on the auxiliary tasks, we discretized the steering and acceleration more finely resulting in 20 classes. For training, we have used mean cross-entropy loss between the predicted class and the autopilot class. Training was performed on a single machine equipped with a GeForce RTX3090 GPU, Ryzen 5950x CPU, and 16GB of RAM. A total of 500K time steps were generated using random roll-outs of the internal CARLA vehicle autopilot module. Also, the 500K time-steps are split into 70% training, 15% validation, and 15% testing data.

The imitation classification results with future latent vector and the baseline model (World Models (WM) [6]) are shown in Table 1. The results show that the proposed model performs better than the baseline model while having significantly less number of trainable parameters. Table 2 shows the preliminary results of imitation learning validated on

Table 1: Imitation learning results using predicted future latent vectors.

Model	Classification Accuracy%
World Models	71.37 ± 0.52
Proposed Model	$\mathbf{77.08 \pm 0.51}$

Table 2: Imitation learning results using latent vector trained on auxiliary tasks.

Auxiliary Tasks	Classification Accuracy%
Not Included	62.6
Included	**68.6**

unseen conditions (different weather condition) using the network trained on auxiliary tasks. The results show a ≈6%

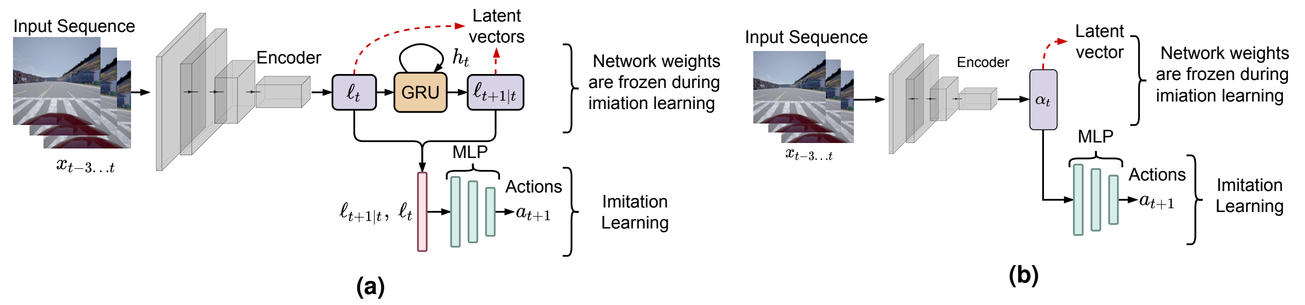

Figure 3: (a) Imitation learning using the latent vector predicted by future latent vector prediction model. (b) Imitation learning using the latent vector trained on auxiliary tasks. Note for both imitation learning experiments only a small MLP is trained while freezing the weights of latent vector extraction networks.

improvement when using auxiliary tasks. Note that this is a significant gain, as most of the performance improvement corresponds to hard driving scenarios where auxiliary tasks become relevant (i.e., intersections and close-proximity to the front vehicle). Finally, the training of overall architecture with all the combined latent vector with self-attention is still a work in progress. However, the initial results provides an efficacy of using latent vector capturing different aspects of driving.

References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in neural information processing systems*, 2019.

[2] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pages 254–263. PMLR, 2018.

[3] Stephen Casper, Xavier Boix, Vanessa D'Amario, Ling Guo, Martin Schrimpf, Kasper Vinken, and Gabriel Kreiman. Frivolous units: Wider networks are not really that wide. *arXiv preprint arXiv:1912.04783*, 2019.

[4] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.

[5] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[6] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[7] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric Xing. Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 584–599, 2018.

[8] Vaishnavh Nagarajan. Explaining generalization in deep learning: progress and fundamental limits. *arXiv preprint arXiv:2110.08922*, 2021.

[9] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

[10] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

[11] Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

[12] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

[13] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[14] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[15] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[16] Xiang Zhang, Lina Yao, Xianzhi Wang, Jessica Monaghan, David Mcalpine, and Yu Zhang. A survey on deep learning based brain computer interface: Recent advances and new frontiers. *arXiv preprint arXiv:1905.04149*, 2019.