Multi-modal Deep Reinforcement Learning with a Novel Sensor-based Dropout

Guan-Horng Liu
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
guanhorl@andrew.cmu.edu

Avinash Siravuru
Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
asiravur@andrew.cmu.edu

Sai Prabhakar
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
spandise@andrew.cmu.edu

Manuela Veloso
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
mmv@cs.cmu.edu

George Kantor
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
kantor@ri.cmu.edu

Abstract

Sensor fusion is a key driver in the success of autonomous driving, given how instrumental it is to improve accuracy and robustness in the vehicle's algorithmic decision making. However, in the space of end-to-end sensorimotor control, this multi-modal outlook has not received much attention. In the interest of enhancing safety and accuracy in control, a multi-modal approach to end-to-end autonomous navigation is need of the hour. Here, we introduce Multi-modal Deep Reinforcement Learning, and demonstrate how the use of multiple sensors improves the reward for an agent. For this purpose, we augment using both DDPG and NAF algorithms to admit multiple sensor input. The efficacy of a multi-modal policy is shown through extensive simulations experiments in TORCS, a popular open-source racing car game. Additionally, we introduce a new stochastic regularization technique, called Sensor Dropout to reduces the network's sensitivity to any one sensor. Suitable metrics have been devised to study this behavior and highlight its applicability to other domains that operate in multi-modal settings.

Keywords: Multi-modal Learning, Deep Reinforcement Learning, Stochastic Regularization

1 Introduction

One of the key challenges to building robust autonomous navigation systems is to develop a strong intelligence pipeline that is able to efficiently gather incoming sensor data and take suitable control actions with good repeatability and fault-tolerance. In the past, this was addressed in a modular fashion, where specialized algorithms were developed for each sub-system and later integrated with some fine tuning. Recently, there is a great interest in applying an end-to-end approach [1] to autonomous driving wherein one can learn a complex mapping that goes directly from the input to the output by leveraging the availability to a large volume of task specific data. This end-to-end approach can be formulated better in Deep Reinforcement Learning (DRL) that allows policy improvement with feedback, and has been shown to achieve human-level performance on many gaming environments [2].

Previous work in DRL predominantly learned policies based on a single input modality, i.e., either low-dimensional physical states, or high-dimensional pixels. We have to however keep in mind the importance of enhancing safety and accuracy to the maximum extent possible in autonomous driving. In this light, we believe that it is worth exploring tangible ways to develop policies that operate with multiple inputs. Multi-modal Perception was an integral part of autonomous navigation solutions and even played a critical role in their success before the advent of end-of-end deep learning based approaches. Sensor fusion offers several advantages namely robustness to individual sensor noise/failure, improved object classification and tracking, robustness to varying weather and environmental conditions, etc. Multi-modal Deep Learning, in general, is an active area of research in other domains like audiovisual systems , gesture recognition , text/speech and language models , etc. However, Multi-modal Learning is conspicuous by its absence in the modern end-to-end autonomous navigation literature.

In this work, we present a end-to-end controller that uses multi-sensor input to learn an autonomous navigation policy in a physics-based gaming environment. To show the effectiveness of Multi-modal Perception, we picked two popular continuous action DRL algorithms namely Normalized Advantage Function (NAF) [3] and Deep Deterministic Policy Gradient (DDPG) [4], and augmented them to accept multi-modal input. We show through extensive empirical testing that a multi-modal deep reinforcement learning architecture achieves higher average reward. Moreover, we propose a novel stochastic regularization method called *Sensor Dropout* during training to prevent a multi-modal policy from rely heavily on only sensor subset. *Sensor Dropout* can be directly applied at the sensor fusion layer just like Dropout [5]. Its appeal lies in its simplicity during implementation and is designed to be applicable even to the DRL setting. As far as we know, this is the first attempt at applying stochastic regularization in a DRL setting. We show extensive simulation results to validate the net improvement in performance and robustness with Sensor Dropout.

2 Background

Deep Reinforcement Learning (DRL) has become a thriving research branch after Mnich et al. [2] proposed Deep Q-Network (DQN) that uses a deep architecture as a non-linear function to approximate action-value function Q. Several novel modifications, namely *replay buffer* and *target network*, has been proposed to stabilize the learning with deep nets.

Later, Gu et al.[3] proposed a continuous variant of DQN by a clever network construction. The Q function, which they called Normalized Advantage Function (NAF), parameterized the advantage function A quadratically over the action space $a \in \mathbf{A}$, and is weighted by non-linear feature of states $s \in \mathbf{S}$ with matrix $P(s|\theta^P)$. During run-time, the greedy policy can be performed by simply taking the output of policy network $a = \mu(s|\theta^\mu)$.

$$Q(s,a|\theta^Q) = A(s,a|\theta^A) + V(s|\theta^V) \tag{1}$$

$$A(s,a|\theta^A) = -\frac{1}{2}(a-\mu(s|\theta^\mu))^T P(s|\theta^P)(a-\mu(s|\theta^\mu)) \tag{2}$$

An alternative approach to continuous RL tasks is the use of an actor-critic framework. In [6], a novel *deterministic* policy gradient was proposed and it was shown that deterministic policy gradients have a model-free form and follow the gradient of the action-value function.

$$\nabla_{\theta^\mu} J = \mathbb{E}[\nabla_a Q(s,a|\theta^Q)\nabla_a \mu(s)] \tag{3}$$

Building on this result, Lillicrap et al. proposed an extension of DPG with deep architecture to generalize their prior success[2] with discrete action spaces onto continuous spaces. Using the DPG, an off-policy algorithm was developed to estimate the Q function using similar techniques as in [2] for stable learning.

3 Multimodal Deep Reinforcement Learning

3.1 Multimodal Network Architecture

We denote a set of observations composed from K sensors as, $S = [S^{(1)}\ S^{(2)}\ ..\ S^{(K)}]^T$, where $S^{(i)}$ stands for observation from i^{th} sensor. In the multi-modal network, each sensory signal is pre-processed along independent paths. Each path has a feature extraction module with an appropriate network architecture, using randomly initialized or pre-trained weights. The modularized feature extraction stages for multiple inputs naturally allows for independent extraction of salient information that is transferable (with some tuning

if needed) to other applications like collision avoidance, pedestrian detection and tracking, etc. The outputs of feature extraction modules are eventually flattened and concatenated to form the multi-modal state $\tilde{S} = [\tilde{S}^{(1)}\ \tilde{S}^{(2)}\ ..\ \tilde{S}^{(K)}]^T$.

3.2 Sensor Dropout (SD)

As shown in Fig. 1, consider the multi-modal state given by $\tilde{S} = [\tilde{S}^{(1)}\ \tilde{S}^{(2)}\ ..\ \tilde{S}^{(K)}]^T$,where $\tilde{S}^{(i)} = [\tilde{X}_1^{(i)}\ \tilde{X}_2^{(i)}\ ..\ \tilde{X}_{F_i}^{(i)}]^T$, the Sensor Dropout randomly drops (set to zero) at most $K-1$ modality at each training step, and scales the rest *inversely proportional* to the summation over number of remained hidden units. Denote the dropping configuration as a K-element vector $\delta_c = [\delta_c^{(1)}\ \delta_c^{(2)}\ ..\ \delta_c^{(K)}]^T$, where $\delta_c^{(i)} \in \{1,0\}$ represents the switching indicator of i^{th} sensor block, the rescaling ratio is defined as:

$$\alpha_c = \frac{\sum_{i=1}^{K} F_i}{\sum_{i=1}^{K} \delta_c^{(i)} F_i}. \tag{4}$$

The output of Sensor Dropout for the k^{th} node in i^{th} sensor block for some layer configuration c can be shown as,

$$\hat{S}_{k,c}^{(i)} = \mathbf{M}_c^{(i)} \tilde{X}_k^{(i)}, \qquad \text{where } \mathbf{M}_c^{(i)} = \alpha_c \delta_c^{(i)}. \tag{5}$$

$\mathbf{M}_c^{(k)}$ is an augmented mask encapsulating both dropout and re-scaling.

Experience Replay is a commonly used technique in many off-policy DRL algorithms to break the correlation among the agent's experience and stabilize the training process. However, extending a dropout-like regularization technique to DRL setting to operate on the experience replay is non-trivial and can cause instability . Moreover, the experience generated at each training step is now *sensor-dropout's configuration dependent*. In order to correctly update the network with experience generated under specific dropping configuration, one possible way is to maintain multiple replay buffers for each of the sensor drop configuration. During batch updates, a specific replay buffer is queried given the current layer configuration, followed by the standard batch sampling and update network. This computational advantage makes the principle of sensor-based dropout more attractive and ideally suited in the DRL setting.

4 Evaluation and Analysis

4.1 Platform Setup

The proposed approach was verified on TORCS [7], a popular open-source car racing simulator that is capable of simulating sophisticated vehicle dynamics. In order to make the learning problem representative of the real-world setting, we picked the following sensors from the TORCS package [8]: the 2D laser range finder, front-view camera with RGB channel, vehicle state - position and speed. The action space is a continuous vector in $\mathbb{R}^3$, whose elements represent acceleration, steering angle, and braking, respectively. For ease of analysis, we normalize each action variable.

An exploration strategy is injected adding an Ornstein-Uhlenbeck process noise to the output of the policy network. The choice of reward function is slightly different from m Lillicrap et al.[4] as an additional penalty term to penalize side-ways drifting along the track was added. In practice, this modification leads to more stable policies during training [9].

We use the following sensing modalities for our state description: (1) We define *Sensor 1* as a 10 DOF hybrid state containing all physical information, including 3D velocity (3 DOF), position and orientation with respect to track center-line (2 DOF), and finally rotational speed of 4 wheels (4 DOF) and engine (1 DOF). (2) *Sensor 2* consists of 4 consecutive laser scans . Here, each laser scan is composed of 19 readings spanning a 180° field-of-view in the the front of car. Finally, as *Sensor 3*, we supply 4 consecutive color images capturing the car's front-view and each one has a resolution 64×64.

These three representations are used separately to develop our baseline single sensor based policies. The multi-modal state on the other hand has access to all sensors at any given point. When Sensor Dropout (SD) is applied, agent will randomly lose access to a strict subset of sensors. To this end, three different categorical distributions on dropping configurations have been experimented with, and the best learned policy among three configurations is reported here.

4.2 Analysis of Results

4.2.1 Convergence

The training performance, for all the proposed models and their corresponding baselines, is shown in Fig. 2. As expected, using high dimensional sensory input directly impacts convergence rate of the policy. However, despite the high dimensionality, with the addition Sensor Dropout, the convergence rate improves, sometime even drastically, as shown in Fig. 2(b) for DDPG.

4.2.2 Robustness

To analyze the generalization of the polices, we pick up the the best learned policies of each model and tested its performance in another similar racing configuration. As summarized in Fig. 3, we found out that the DDPG agent was capable of finding a reasonable policy regardless of the representation of states. The performance of NAF, on the other hand, was greatly affected by the dimensionality and informative level carried on its input. In fact, the agent failed to achieve even an reasonable positive score in testing when using visual information. Using multi-modal in this case alleviated the issue by providing alternative informative states alone with abstract visual input.

We also measured the the average reductions of performance by injecting Gaussian noise into the sensor modules. The relative drop in performance compared with a noiseless environment is summarized in Fig. 4. The performance of the NAF agent drops dramatically when the noise is introduced. From Fig. 2(a) we also observed that NAF in the multi-modal is sensitive to states from sensors which are easily interpretable such as laser scanners. This effect shows that using an over-complete state representation holds a risk of the agent learning an undesired policy where the influence of different features gets unbalanced. The regularization introduced by Sensor Dropout alleviates this issue and learns a stable policy on both algorithms, with only slight decrease of the performance compared with multi-modal agents trained without SD as shown in Fig. 3.

4.2.3 Sensitivity

In this part, we introduce an intuitive but effective way of monitoring how much policy depends on each sensor subset by measuring the gradient of policy network with respect to each subset. Under our implementation of Sensor Dropout, the agent will observe the sensor data from one of the following three combinations during training: (1) hybrid state composed of physical information and laser, (2) image, or (3) full multi-modal state. Therefore, the policy sensitivity of the first sensor subset over the second one in Table 1. The Metric can be calculated with:

$$\mathbf{T}_2^1 = \frac{1}{M}\sum_{i=1}^{M} \frac{\left|\nabla_{\tilde{S}_i^{(1)}} \mu(\tilde{S}|\theta^\mu)\Big|_{S_i}\right|}{\left|\nabla_{\tilde{S}_i^{(2)}} \mu(\tilde{S}|\theta^\mu)\Big|_{S_i}\right|} \tag{6}$$

The dataset $\mathbf{S} = \{S_i\}_{i=1\ldots M}$ is collected from a stable policy running on both training and testing environments. With the aid of SD, the over-dependence of the policy toward the first group of sensor subset is significantly reduced.

Figure and Table

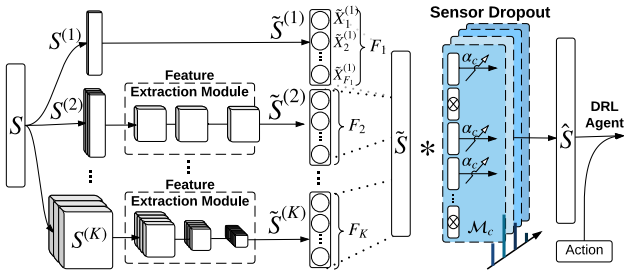
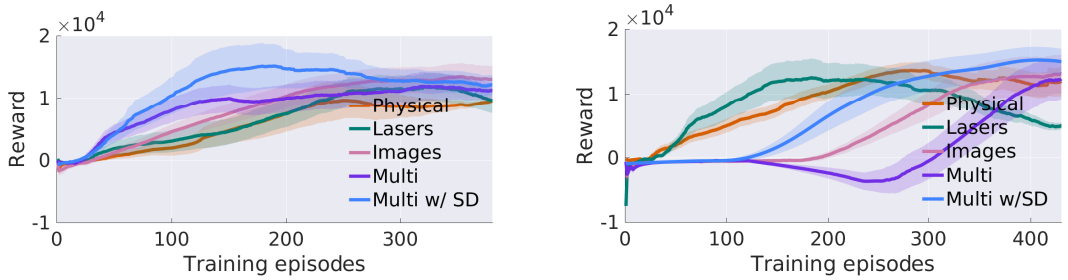
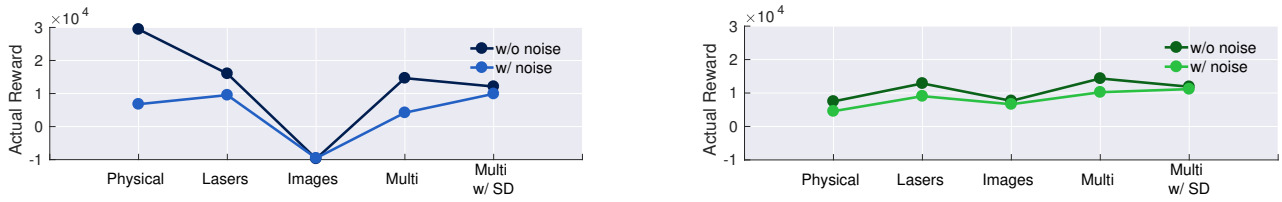


Figure 1: Illustration of Multimodal Architecture and Sensor Dropout. The feature extraction module can be either pure identity function (modality 1), or convolution-based layer (modality $2 \rightarrow K$). The operation $*$ stands for element-wised multiplication.



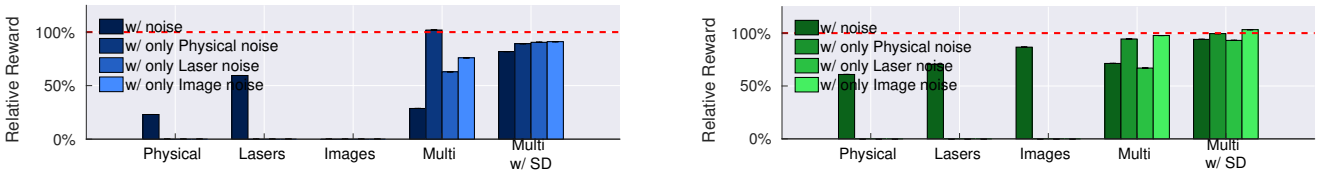
(a) NAF (b) DDPG

Figure 2: Training Summary of three baseline Uni-modal agents, and multi-modal agents with and without Sensor Dropout regularization.



(a) NAF (b) DDPG

Figure 3: Policy Robustness Analysis: darker line connects average rewards of leaner policies with accurate sensing while the lighter lines connects the corresponding policies in the face of sensor noise.



(a) NAF (b) DDPG

Figure 4: Relative reward of robustness analysis experiment. The bar box measures the relative scale among each of the models when noise is introduced. The red dotted lines show the performance without noise.

Table 1: Result of Policy Sensitivity using Eq. (6). Closer to one is better.

		Training Env.		Testing Env.	
		Mean	Median	Mean	Median
NAF	w/o SD	1.651	1.871	1.722	1.940
	w/ SD	**1.284**	**1.379**	**1.086**	**1.112**
DDPG	w/o SD	1.458	1.449	1.468	1.437
	w/ SD	**1.168**	**1.072**	**1.171**	**1.120**

References

[1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[3] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. *arXiv preprint arXiv:1603.00748*, 2016.

[4] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[5] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[6] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[7] Bernhard Wymann, E Espié, C Guionneau, C Dimitrakakis, R Coulom, and A Sumner. Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net*, 2000.

[8] Naoto Yoshida. Gym-torcs, 2016.

[9] Yan-Pan Lau. Using keras and deep deterministic policy gradient to play torcs, 2016.