Learning End-to-end Multimodal Sensor Policies for Autonomous Navigation

Guan-Horng Liu[1], **Avinash Siravuru**[2], **Sai Prabhakar**[1],
Manuela Veloso[1], **and George Kantor**[1]

{guanhorl,asiravur,spandise}@andrew.cmu.edu
mmv@cs.cmu.edu, kantor@ri.cmu.edu
[1]Robotics Institute, Carnegie Mellon University, USA
[2]Department of Mechanical Engineering, Carnegie Mellon University, USA

Abstract: Sensor fusion is indispensable to improve accuracy and robustness in an autonomous navigation setting. However, in the space of end-to-end sensorimotor control, this multimodal outlook has received limited attention. In this work, we propose a novel stochastic regularization technique, called *Sensor Dropout*, to make multimodal sensor policy robust. To further enhance robustness, we introduce an auxiliary loss on policy network in addition to standard DRL loss to reduce variance in actions of the multimodal sensor policy. Through extensive empirical testing, we demonstrate that our proposed policy can 1) operate with minimal performance drop in noisy environments and 2) remain functional even in the face of a sensor subset failure. Finally, through the visualization of gradients, we show that the learned policies are conditioned on the same latent input distribution despite having multiple and diverse observations spaces - a hallmark of true sensor-fusion. This efficacy of a multimodal sensor policy is shown through simulations on TORCS, a popular open-source racing car game. A demo video can be seen here: https://youtu.be/QAK2lcXjNZc.

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 integrated with some fine tuning. More recently, there is an interest in an end-to-end approach that learns complex mappings directly from the input to the output by leveraging the availability of a large volume of task-specific data.

These end-to-end approaches have become more appealing thanks to the extension of deep learning's success to robotics, and has been applied in visuomotor policies for autonomous driving [1]. However, the traditional deep supervised learning-based driving requires a great deal of human annotation and may not be able to deal with the problem of accumulating errors [2]. On the other hand, deep reinforcement learning (DRL) offers a better formulation that allows policy improvement with feedback, and has achieved human-level performance on challenging game environments [3, 4].

In this work, we present an end-to-end controller that uses multi-sensor input to learn an autonomous navigation policy in a physics-based gaming environment called TORCS [5] without needing any pretraining. To show the effectiveness of multimodal perception, we pick two popular continuous action DRL algorithms namely Normalized Advantage Function (NAF) [6] and Deep Deterministic Policy Gradient (DDPG) [7], and augment them to accept multimodal input. We limit our objective to only achieving autonomous navigation without any obstacles or other cars. This problem is kept simpler deliberately to focus our attention more on analyzing the performance of the proposed multimodal configurations using extensive quantitative and qualitative testing. To ensure that the multimodal sensor policy does not heavily rely on all the inputs to the extent that it may fail completely

1st Conference on Robot Learning (CoRL 2017), Mountain View, United States.

even if a single sensor broke down fully or partially, which renders sensor redundancy useless, we apply a novel stochastic regularization method called *Sensor Dropout* during training. Our approach reduces the policy sensitivity to a particular sensor subset, and make it capable of functioning even in the face of partial sensor failure. We further augment the standard DRL loss with an auxiliary loss that helps reduce the action variations of the trained policy.

Recently, promising experimental results were shown combining camera and lidar to build an end-to-end steering controller of a UGV navigation [8]. Similarly, a multimodal DQN was built for a Kuka YouBot [9] by fusing information for homogeneous sensing modalities. However, the fusion stage in [8] is limited to sensors that are spatially redundant with each other, and requires the feature embedding of each sensor to have the same dimensionality. On the other hand, [9] requires a two-stage training by first approximating a Q^* function then refining the policy with DropPath [9] regularization, where each stage may risk to end up optimizing on different distribution in high-dimensional state space. Our proposed method loosens both assumptions. The fusion can be performed on heterogeneous and spatially unrelated sensors, and can be integrated with standard DRL into one single training phase. Through extensive testings in Section 5.2, we highlight its simplicity and efficiency, which make it very general and applicable across a diverse range of problems.

2 Related Work

Multimodal DRL aims to leverage the availability of multiple, potentially imperfect, sensor inputs to improve learned policy. Most autonomous driving vehicles have been equipped with an array of sensors like GPS, Lidar, Camera, and Odometer, etc. While one would offer a long range noisy estimate, the other would offer a shorter range accurate one. When combined, the resulting observer will have a good and reliable estimate of the environment. It is important to note that some of these sensors, like GPS and odometers, are readily available but unfortunately seldom include in these end-to-end learning models [1].

Previous works in DRL predominantly learned policies based on a single input modality, i.e., either low-dimensional physical states, or high-dimensional pixels. For autonomous driving where enhancing safety and accuracy to the maximum possible extent is the top priority, developing policies that operate with multiple inputs is the need of the hour. In fact, multimodal perception was an integral part of autonomous navigation solutions and even played a critical role in their success [10] before the advent of end-to-end deep learning based approaches. Sensor fusion offers several advantages, namely robustness to individual sensor noise/failure, improving object classification and tracking [11], etc. In this light, several recent works in DRL have tried to solve the complex robotics tasks such as human-robot-interaction [12], manipulation [13] and maze navigation [14] with multimodal sensor inputs. In principle, [14] is similar to this work as it uses image, velocity, and depth information to navigate through a maze. However, the robot evolves with simpler dynamics when compared to autonomous road navigation. Additionally, depth is only used as an auxiliary loss, i.e. it is *not* an input to the trained policy but is only used to improve the learning outcomes.

Multimodal deep learning, in general, is an active area of research in other domains like audiovisual systems [15], , text/speech and language models [16], etc. However, Multi-modal learning is conspicuous by its absence in the modern end-to-end autonomous navigation literature. Another challenge in multimodal learning is the specific case of over-fitting where instead of learning the underlying latent object representation using multiple diverse observations, the model instead learns a complex representation in the original space itself, rendering the whole point of multimodal sensing useless and computationally burdensome. An illustrative example for this case is a car navigating when all sensors remain functional but fails to navigate completely even if one fails or is partially corrupted. This kind of behavior is detrimental and suitable regularization measures should be set up during training to avoid it.

Stochastic regularization is an active area of research in deep learning made popular by the success of, *Dropout* [17]. Following this landmark paper, numerous extensions were proposed to further generalize this idea ([18, 19, 20, 21]). In the similar vein, two interesting techniques have been proposed for specialized regularization in the multimodal setting namely ModDrop [22] and ModOut [23]. Given a much wider set of sensors to choose from, ModOut attempts to identify which sensors are actually needed to fully observe the system behavior. This is out of the scope of this work. Here, we focus on improving the latent state representation based on inputs from multiple observers. On the other hand, ModDrop requires pretraining with individual sensor inputs using separate loss functions. In fact, the method is originally designed for multimodal deep learning on a *fixed* dataset. We argue that for DRL where the training dataset is generated during run-time, pretraining for each sensor policy may end up optimizing on *different* distribution especially in high-dimensional state space. In comparison, *Sensor Dropout* is designed to be applicable to the DRL setting. With SD, a network can be directly constructed in an end-to-end fashion and the sensor fusion layer can be added just like Dropout. It is more efficient in training and scales better as the number of sensor modality increases. A similar effort recently [9] used DropPath instead to regularize the sensor fusion problem. However, it only utilized the same sensory inputs, namely lidar data, and pretraining on a Q-network in advance is required before applying the sensor regularization.

3 Multimodal Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) Brief Review: We consider a standard Reinforcement Learning (RL) setup, where an agent operates in an environment E. At each discrete time step t, the agent observes a state $s_t \in \mathcal{S}$, picks an action $a_t \in \mathcal{A}$, and receives a scalar reward $r(s_t, a_t) \in \mathbb{R}$ from the environment. The return $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$ is defined as total discounted future reward at time step t, with γ being a discount factor $\in [0, 1]$. The objective of the agent is to learn a policy that eventually maximizes the expected return. The learned policy, π, can be formulated as either stochastic $\pi(a|s) = \mathbb{P}(a|s)$, or deterministic $a = \mu(s)$. The value function V^π and action-value function Q^π describe the expected return for each state and state-action pair upon following a policy π. Finally, an advantage function $A^\pi(s_t, a_t)$ is defined as the additional reward or advantage that the agent will have for executing some action a_t at state s_t and it is given by $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$.

In high dimensional state/action space, these functions are usually approximated by a suitable parametrization. Accordingly, we define θ^Q, θ^V, θ^A, θ^π, and θ^μ as the parameters for approximating Q, V, A, π, and μ functions, respectively. It was generally believed that using non-linear function approximators would lead to unstable learning in practice. Recently, Mnih et al. [3] applied two novel modifications, namely *replay buffer* and *target network*, to stabilize the learning with deep nets. Later, several variants were introduced that exploited deep architectures and extended to learning tasks with continuous actions [7, 24, 6].

To exhaustively analyze the effect of multi-sensor input and the new stochastic regularization technique, we pick two algorithms in this work namely DDPG and NAF. It is worth noting that the two algorithms are very different, with DDPG being an off-policy actor-critic method and NAF an off-policy value-based one. By augmenting these two algorithms, we highlight that any DRL algorithm, modified appropriately, can benefit from using multi-sensor inputs. Due to space constraint, we list the formulation of the two algorithms in Supplementary Material (Section A).

Multimodal Sensor Policy Architecture: We denote a set of observations composed from M sensors as, $S = [S^{(1)} \; S^{(2)} \; .. \; S^{(M)}]^T$, where $S^{(i)}$ stands for observation from i^{th} sensor. In the multimodal network, each sensory signal is pre-processed along an independent path. Each path has a feature extraction module that can be either pure identity function (modality 1), or convolution-based layer (modality $2 \rightarrow M$). The modularized feature extraction stage naturally allows for independent extraction of salient information that is transferable (with some tuning if needed) to other applications . The outputs of feature extraction modules are eventually flattened and concatenated to form the multimodal state. The schematic illustration of modularized multimodal policy is shown in Fig. 1.

4 Augmenting MDRL

In this section, we propose two methods to improve training of a multi-sensor policy. We first introduce a new stochastic regularization called Sensor Dropout, and explain its advantages over the standard Dropout for this problem. Later, we propose an additional unsupervised auxiliary loss function to reduce the policy variance.

Sensor Dropout (SD) for Robustness: The Sensor Dropout is a variant of the vanilla Dropout [17] that maintains dropping configurations on each sensor module instead of each neuron. Though both methods share a similar motivation on stochastic regularization, SD is better-motivated for training the multimodal sensor policy. By randomly dropping the sensor block during training, the policy network

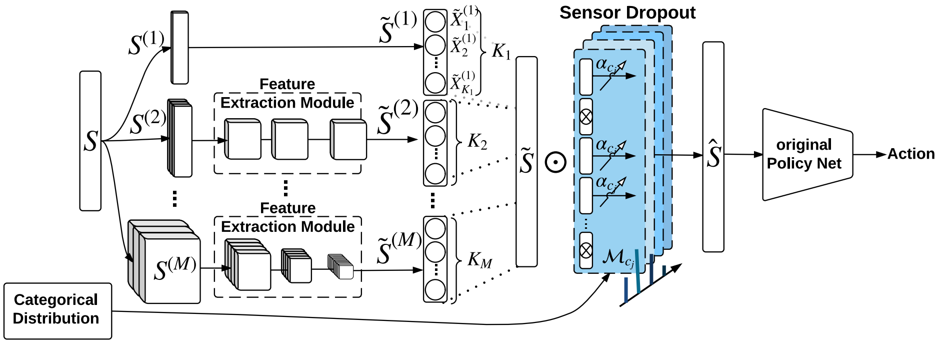


Figure 1: Illustration of multimodal sensor policy augmented with Sensor Dropout. The operation $\odot$ stands for element-wised multiplication. The dropping configuration of Sensor Dropout is sampled from a categorical distribution, which stands as an additional input to the network.

is encouraged to exploit the modularized structure among each sensor stream. In the application to the complex robotics system, SD has advantages on handling imperfect sensing conditions such as latency, noises, and even partial sensor failure. As shown in Fig.1, consider the multimodal state $\tilde{S}$, the dropping configuration is defined as a M-dimensional vector $\boldsymbol{c} = [\delta_c^{(1)}\ \delta_c^{(2)}\ ..\ \delta_c^{(M)}]^T$, where each element $\delta_c^{(i)} \in \{0, 1\}$ represents the on/off indicator for the i^{th} sensor modality. Each sensor modality is represented by a K_i-dimensional vector, denoted as $\tilde{S}^{(i)} = [\tilde{X}_1^{(i)}\ \tilde{X}_2^{(i)}\ ..\ \tilde{X}_{K_i}^{(i)}]^T$. The subscript i indicates that each sensor may have different dimension. We now detail the two main differences between original Dropout and SD along with their interpretations.

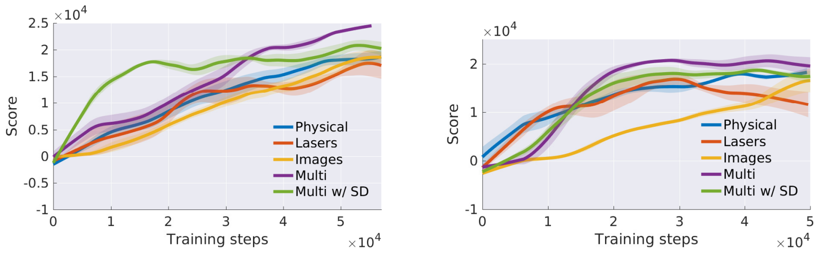
Firstly, note that the dimension of the dropping vector $\boldsymbol{c}$ is much lower than the one in the standard Dropout ($\sum_{i=1}^{M} K_i$). As a consequence, the probability of the event where all sensors are dropped out (i.e. $\boldsymbol{c_0} = [0^{(1)}\ 0^{(2)}\ ..\ 0^{(M)}]^T$) is not negligible in SD. To explicitly remove $\boldsymbol{c_0}$, we slightly depart from [17] in modeling the SD layer. Instead of modeling SD as random process where any sensor block $\tilde{S}^{(i)}$ is switched on/off with a *fixed* probability p, we define the random variable as the dropping configuration $\boldsymbol{c}$ itself. Since there are $N = 2^M - 1$ possible states for $\boldsymbol{c}$, we accordingly sample from an N-state categorical distribution $\mathbb{P}$. We denote the probability of a dropping configuration $\boldsymbol{c_j}$ occurring with p_j, where the subscript j ranges from 1 to N. The corresponding pseudo-Bernoulli [1] distribution for switching on a sensor block $\tilde{S}^{(i)}$ can be calculated as $p^{(i)} = \sum_{j=1}^{N} \delta_{c_j}^{(i)} p_j$. Note that although sampling from standard Bernoulli on sensor blocks with rejection on c_0 will have the same effect with the proposed categorical distribution, the latter is better-motivated in that it offers convenience in analysis and direct interpretation of *how different sensors are fused*, and can be adaptive to the current sensor reliability during run-time.

Another difference from the standard Dropout is the rescaling process. Unlike the standard Dropout which preserves a *fixed* scaling ratio after dropping neurons, the rescaling ratio in SD is formulated as a function of the dropping configuration and sensor dimensions. The intuition is to keep the weighted summations equivalent among different dropping configurations in order to activate the later hidden layers. The scaling ratio is calculated as $\alpha_{c_j} = \frac{\sum_{i=1}^{M} K_i}{\sum_{i=1}^{M} \delta_{c_j}^{(i)} K_i}$.

In summary, the output of SD for the k^{th} feature in i^{th} sensor block (i.e. $\tilde{S}^{(i)}$) given a dropping configuration $\boldsymbol{c_j}$ can be shown as $\hat{S}_{c_j,k}^{(i)} = \mathcal{M}_{c_j}^{(i)} \tilde{X}_k^{(i)}$, where $\mathcal{M}_{c_j}^{(k)} = \alpha_{c_j} \delta_{c_j}^{(i)}$ is an augmented mask encapsulating both dropout and re-scaling.

Auxiliary Loss for Variance Reduction: An alternative interpretation of the SD-augmented policy is that sub-policy induced by each sensor combination are jointly optimized during training. Denote the ultimate SD-augmented policy and sub-policy induced by each sensor combination as $\mu_{\boldsymbol{c} \sim \mathbb{P}}$ and μ_{c_j}, respectively. The final output maintains a geometric mean over N different actions.

[1] We wish to point out that $p^{(i)}$ is pseudo-Bernoulli as we restrict our attention to cases where at least one sensor block is switched on at any given instant in the layer. This implies that, while the switching on of any sensor block $\tilde{S}^{(i)}$ is independent of the other, switching off is not. So the distribution is no longer fully independent.



(a) NAF

(b) DDPG

Figure 2: Training performance comparison of three baseline single sensor policies, and the proposed multi-modal policies, with and without Sensor Dropout.

Despite the expectation of the total policy gradients for each sub-policy is the same, SD provides no guarantees on the consistency of these actions. To encourage the policy network to extract salient features from each sensor that can be embedded with similar representations on the latent space, we further augment an auxiliary loss that penalizes the inconsistency among μ_{c_j}. This additional penalty term provides an alternative gradient that reduces the variation of the ultimate policy, i.e. $Var\left[\mu_{\boldsymbol{c}\sim\mathbb{P}}\right]$. The mechanism is motivated from the recent successes [14, 25] that use the auxiliary tasks to improve both agent's performance and convergence rate. However, unlike most previous works that design the auxiliary tasks carefully from the ground truth environment, we formulate the *target action* from the policy network itself. Under the standard actor-critic architecture, the target action is defined as the output action of the sub-policy in target actor network $\tilde{\mu}_{\boldsymbol{c}\sim\mathbb{P}}$ that maximizes the target critic values $\tilde{Q}$. In other words, we use the currently best-trained sub-policy as a heuristic to guide other sub-policies during training.

$$L_{aux} = \lambda \sum_{i=1}^{N} (\mu_{c_j}(s_i) - \tilde{\mu}_{c^*}(s_i))^2, \quad \text{where } c^* = \underset{c_j \sim \mathbb{P}}{\text{argmax}} \sum_{i=1}^{N} \tilde{Q}(s_i, \tilde{\mu}_{c_j}(s_i)) \tag{1}$$

Here, λ is an additional hyperparameter that indicates the ratio between the two losses, and N is the batch size for off-policy learning.

5 Evaluation Results

5.1 Platform Setup

TORCS Simulator The proposed approach is verified on TORCS [5], a popular open-source car racing simulator that is capable of simulating physically realistic vehicle dynamics as well as multiple sensing modalities [26] to build sophisticated AI agents. In order to make the learning problem representative of the real-world setting, we use the following sensing modalities for our state description: (1) We define *Sensor 1* as a hybrid state containing physical-based information such as odometry and simulated GPS signal. (2) *Sensor 2* consists of 4 consecutive laser scans (i.e., at time t, we input scans from times $t,\ t-1,\ t-2$ & $t-3$). Finally, as *Sensor 3*, we supply 4 consecutive color images capturing the car's front-view. These three representations are used separately to develop our baseline uni-modal sensor policies. The multi-modal state, on the other hand, has access to all sensors at any given point. When Sensor Dropout (SD) is applied, the agent will randomly lose access to a strict subset of sensors. The categorical distribution is initialized with a uniform distribution among total 7 possible combinations of sensor subset, and the best-learned policy is reported here. The action space is a continuous vector in $\mathbb{R}^2$, whose elements represent steering angle, and acceleration. Experiment details such as exploration strategy, network architectures of each model, and sensor dimensionality are shown in the Supplementary Material (Section B).

5.2 Results

Training Summary: The training performances, for all the proposed models and their corresponding baselines, are shown in Fig. 2. For DDPG, using high-dimensional sensory input directly impacts convergence rate of the policy. Note that the *Images* uni-policy (orange line) has a much larger dimensional state space compared with *Multi* policies (purple and green lines). Counter-intuitively, NAF performs a nearly linear improvement over training steps, and is relatively insensitive to the

Table 1: Final Score of Trained Policy (*unit*:$\times 10^4$)

Policy	w/o Noise	w/ Noise	Performance Drop
Multi Uni-modal w/ Meta Controller	1.51 ± 0.57	0.73 ± 0.40	51.7 %
Multimodal w/ SD	2.54 ± 0.08	2.29 ± 0.60	9.8 %

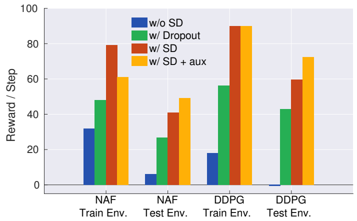


Figure 3: Policy performance when facing random sensor failure.

Table 2: Results of the sensitivity metric.

		Training	Testing
		Env.	Env.
NAF	w/o SD	1.651	1.722
	w/ SD	**1.284**	**1.086**
DDPG	w/o SD	1.458	1.468
	w/ SD	**1.168**	**1.171**

dimensionality of the state space. However, adding Sensor Dropout (SD) dramatically increases the convergence rate. For both algorithms, the final performance for multimodal sensor policies trained with SD is slightly lower than training without SD, indicating that SD has a regularization effect similar to original Dropout.

Comparison with Uni-modal Policies + Meta Controller: One of the intuitive baseline for the multi-sensor problem is to train each uni-modal sensor policy separately. Once individual policies are learned, we can train an additional meta-controller that select which policy to follow given the current state. For this, we follow the setup in [27] by training a meta controller that takes the processed states from each uni-modal policy, and outputs a $3DOF$ softmax layer as the probability of choosing which sub-policy to perform. Note that, we assume perfect sensing during the training. However, to test performance in a more realistic scenario, we simulate mildly imperfect sensing by adding Gaussian noise. Policy performance with and without noise are summarized in Table 1. The performance of the baseline policy drops dramatically once noise is introduced, which implies that the uni-modal policy is prone to over-fitting without any regularization. In fact, the performance drop is sometimes severe in physical-based or laser-based policy. In comparison, the policy trained with SD reaches a higher score in both scenarios, and the drop when noise is introduced is almost negligible.

Policy Robustness Analysis: In this part, we show that SD reduces the learned policy's acute dependence on a subset of sensors in a multimodal sensor setting. First, we consider a scenario when malfunctions of sensors have been detected by the system, and the agent must rely on the remaining sensors to make navigation decisions. To simulate this setting during testing, we randomly block out some sensor modules, and scale the rest using the same rescaling mechanism as proposed in Section 4. Fig. 3 reports the averaging normalized reward of each model. A naive multimodal policy without any stochastic regularization (blue bar) performs poorly in the face of partial sensor failure and transfer tasks. Adding original Dropout makes the policy more generalized, yet the performance is not comparable with SD. Interestingly, by reducing the variance of the multimodal sensor policy with the auxiliary loss, policy tends to have a better generalization among other environments.

Policy Sensitivity Analysis: To monitor the extent to which the learned policy depends on each sensor block, we measure the gradient of the policy output w.r.t a subset block $\tilde{S}^{(i)}$. The technique is motivated from the salient map analysis [28], which has also been applied to DRL study recently [29]. To better analyze the effects of SD, we report on a smaller subset by implementing SD layer to drop either (1) $(physical,\ laser)$ or (2) $vision$. Consequently, the *sensitivity* metric is formulated as the relative sensitivity of the policy on two sensor subsets. If the ratio increases, the agent's dependence shifts toward the sensor block in the numerator and vice versa. Assuming the fusion-of-interest is between the above-mentioned two subsets, we show in Table 2 that, using SD, the metric gets closer to 1.0, indicating nearly equal importance to both the sensing modalities. The *sensitivity metric* is calculated as $\mathcal{T}_2^1 = \frac{1}{M}\sum_i \left(\left| \nabla_{\tilde{S}_i^{(1)}} \mu(\tilde{S}|\theta^\mu) \Big|_{S_i} \right| \right) \left(\left| \nabla_{\tilde{S}_i^{(2)}} \mu(\tilde{S}|\theta^\mu) \Big|_{S_i} \right| \right)^{-1}$.

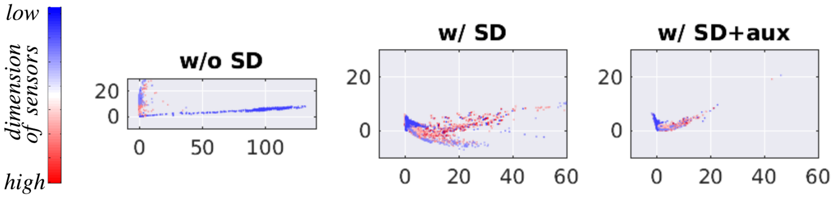


Figure 4: Two-dimensional PCA embedding of the representations in the last hidden layer assigned by the policy networks. The blue dots correspond to the representations induced by the sub-policy that use high dimensional sensor (e.g. vision) as its input. On the other hand, the red dots represent the one with lower sensor stream such as odometry and range finder.

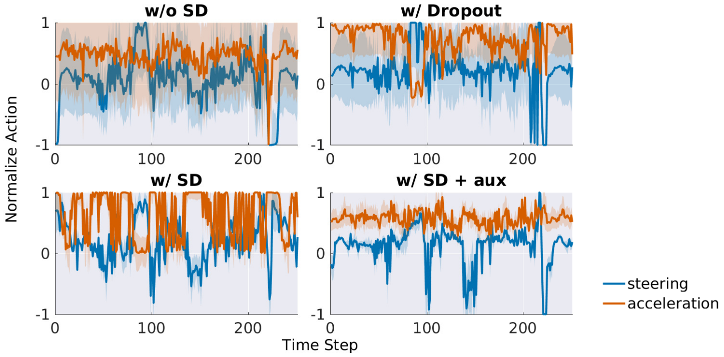


Figure 5: The variance of all the actions induced by sub-policy under each multimodal sensor policy. *Upper-left*: naive policy without any regularization. *Upper-right*: with standard Dropout. *Lower-left*: with Sensor Dropout. *Lower-right*: with Sensor Dropout and auxiliary loss.

Effect of Auxiliary Loss: In this experiment, we verify how the auxiliary loss helps reshape the multimodal sensor policy and reduce the action variance. We extract the representations of the last hidden layer assigned by the policy network throughout a fixed episode. At every time step, the representation induced by each sensor combination is collected. Our intuition is that this latent space represents how the policy network interprets the incoming sensor stream for reaction. Based on this assumption, an ideal multimodal sensor policy should map different sensor streams to a similar distribution as long as the information provided by each combination is representative to lead to the same output action.

As shown in Fig. 4, the naive multimodal sensor policy has a scattered distribution over the latent space, indicating that representative information from each sensor is treated very differently. In comparison, the policy trained with SD has a concentrated distribution, yet it is still distinguishable w.r.t. different sensors. Adding the auxiliary training loss encourages the true sensor fusion as the distribution becomes more integrated. During training, the policy is not only forced to explicitly make decisions under each sensor combination, but also penalized with the disagreements among multimodal sensor policies. In fact, as shown in Fig. 5, the concentration of the latent space directly affect the action variance induced by each sub-policy. We provide the actual covariances for each component and the actual action variance values in the Supplementary Material (Section C).

6 Discussion

Full Sub-Policy Analysis: The performance of each sub-policy is summarized in Fig. 6. As shown in the first and third column, the performances of the naive multimodal sensor policy (red) and the policy trained with standard Dropout (blue) drop dramatically as the policies lose access to image, which shares 87.9% of the total multimodal state. Though Dropout increases the performance of the policy in the testing environment, the generalization is limited to using full multimodel state as input. On the other hand, SD generalizes the policy across *sensor module*, making the sub-policies successfully transfer to the testing environment. It is worth mentioning that the policies trained with SD is capable to operate even when both laser and image sensor are blocked. Interestingly, neither original Dropout or SD show apparent degradation in full policy induced by the regularization. We list more analysis as our future work.

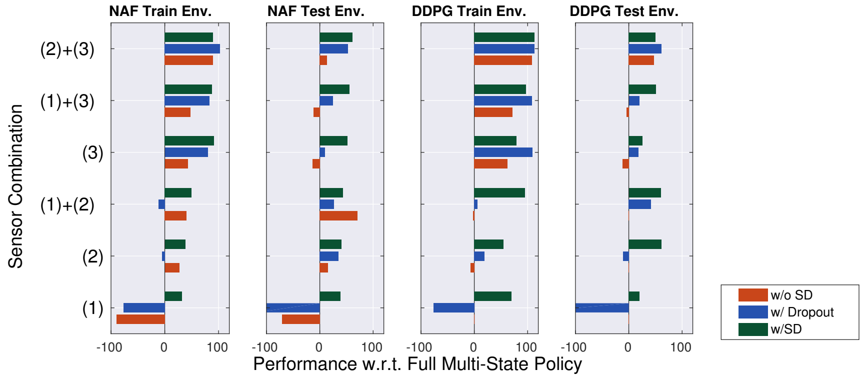


Figure 6: The full analysis of the performance of the total 6 sub-policies. The (1), (2), and (3) labels in y-axis represent physical state, laser, and image, respectively. The x-axis represent the remaining performance w.r.t. the SD policy with all sensor, i.e. (1)+(2)+(3).

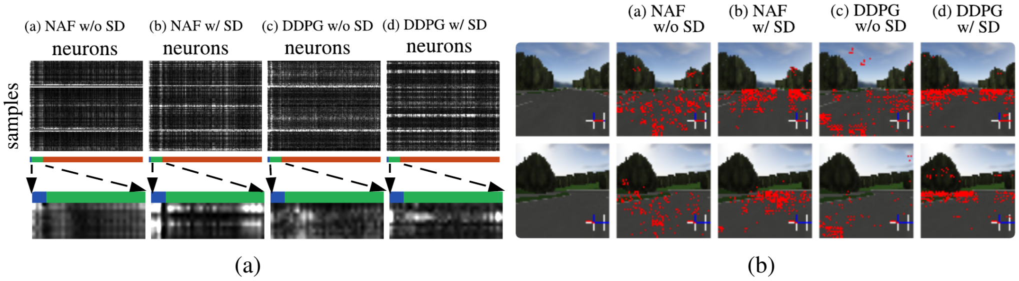


Figure 7: (a)The visualization of the magnitude of gradient for each neuron. The whiter color means the higher gradient. The color bar represents three different sensor modules: physical state(blue), Laser(green), and Image(red). (b) The gradient responses of actions on the image input for each of the multi-modal agents. The top 20% gradients are marked red.

Visualize Policy Attention Region: The average gradient in the policy sensitivity section can also be used to visualize the regions among each sensor where the policy network pays attentions. As shown in Fig. 7(a), we observe that policies trained with SD have higher gradients on neurons corresponding to the corner inputs of the laser sensor, indicating that a more sparse and meaningful policy is learned. These corner inputs corresponded to the laser beams that are oriented perpendicularly to the vehicle's direction of motion, and give an estimate of its relative position on the track. To look for similar patterns in Fig. 7(b), image pixels with higher gradients are marked to interpret the policy's view of the world. We pick two scenarios, 1) straight track and 2) sharp left turn, depicted by the first and second rows in the figure. Note that though policies trained without SD tend to focus more on the road, those areas are in plain color and offer little salient information. In conclusion, policies trained with SD are more sensitive to features such as road boundary, which is crucial for long horizon planning. In comparison, networks trained without SD have relatively low and unclear gradients over both laser and image sensor state space.

7 Conclusions and Future Work

In this work, we introduce a new stochastic regularization technique called Sensor Dropout to promote an effective fusing of information from multiple sensors. The variance of the resulting policy can be further reduced by introducing an auxiliary loss during training. We show that the aid of SD reduces the policy sensitivity to a particular sensor subset, and make it capable of functioning even in the face of partial sensor failure. Moreover, the policy network is able to automatically infer and weight locations providing salient information. For future work, we wish to extend the framework to other environments such as real robotics systems, and other algorithms like TRPO [30], and Q-Prop [31], etc.. Secondly, systematic investigation into the problems such as how to augment the reward function for other important driving tasks like collision avoidance, and lane changing, and how to adaptively adjust the SD distribution during training are also interesting avenues that merit further study.

Acknowledgement

The authors would like to thank Po-Wei Chou, Humphrey Hu, and Ming Hsiao for many helpful discussions, suggestions and comments on the paper. This research was funded under award by Yamaha Motor Corporation.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, volume 1, page 6, 2011.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. In *NIPS'13 Workshop on Deep Learning*, 2013.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[5] B. 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.

[6] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep q-learning with model-based acceleration. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2829–2838, 2016.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[8] N. Patel, A. Choromanska, P. Krishnamurthy, and F. Khorrami. Sensor modality fusion with cnns for ugv autonomous driving in indoor environments. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

[9] S. Bohez, T. Verbelen, E. De Coninck, B. Vankeirsbilck, P. Simoens, and B. Dhoedt. Sensor fusion for robot control through deep reinforcement learning. *preprint arXiv:1703.04550*, 2017.

[10] C. Urmson, J. A. Bagnell, C. R. Baker, M. Hebert, A. Kelly, R. Rajkumar, P. E. Rybski, S. Scherer, R. Simmons, S. Singh, et al. Tartan racing: A multi-modal approach to the darpa urban challenge. 2007.

[11] H. Cho, Y.-W. Seo, B. V. Kumar, and R. R. Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *International Conference on Robotics and Automation (ICRA)*, pages 1836–1843. IEEE, 2014.

[12] A. H. Qureshi, Y. Nakamura, Y. Yoshikawa, and H. Ishiguro. Robot gains social intelligence through multimodal deep reinforcement learning. In *16th International Conference on Humanoid Robots*, pages 745–751. IEEE, 2016.

[13] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[14] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell. Learning to navigate in complex environments. In *International Conference on Learning Representations (ICLR)*, 2017.

[15] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[16] N. Srivastava and R. R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230, 2012.

[17] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[18] C. Murdock, Z. Li, H. Zhou, and T. Duerig. Blockout: Dynamic model selection for hierarchical deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2583–2591, 2016.

[19] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[20] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, H. Larochelle, A. Courville, et al. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*, 2016.

[21] X. Frazão and L. A. Alexandre. Dropall: Generalization of two convolutional neural network regularization methods. In *International Conference Image Analysis and Recognition*, pages 282–289. Springer, 2014.

[22] N. Neverova, C. Wolf, G. Taylor, and F. Nebout. Moddrop: adaptive multi-modal gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1692–1706, 2016.

[23] F. Li, N. Neverova, C. Wolf, and G. Taylor. Modout: Learning to fuse modalities via stochastic regularization. *Journal of Computational Vision and Imaging Systems*, 2(1), 2016.

[24] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

[25] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *CoRR*, abs/1611.05397, 2016. URL `http://arxiv.org/abs/1611.05397`.

[26] N. Yoshida. Gym-torcs. `https://github.com/ugo-nama-kun/gym_torcs`, 2016.

[27] R. Liaw, S. Krishnan, A. Garg, D. Crankshaw, J. E. Gonzalez, and K. Goldberg. Composing meta-policies for autonomous driving using hierarchical deep reinforcement learning.

[28] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[29] Z. Wang, N. de Freitas, and M. Lanctot. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016.

[30] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897, 2015.

[31] S. Gu, T. P. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *International Conference on Learning Representations (ICLR)*, 2017.

[32] R. S. Sutton, D. A. McAllester, S. P. Singh, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063, 1999.

[33] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[34] Y.-P. Lau. Using keras and deep deterministic policy gradient to play torcs. `https://yanpanlau.github.io/2016/10/11/Torcs-Keras.html`, 2016.

[35] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. 2015.

Supplementary Material

A Continuous Action Space Algorithms

A.1 Normalized Advantage Function (NAF)

Q-learning [32] is an off-policy model-free algorithm, where agent learns an approximated Q function, and follows a greedy policy $\mu(s) = \arg\max_a Q(s, a)$ at each step. The objective function $J = \mathbb{E}_{s_i, r_i \sim E\ , a_i \sim \pi}[R_1]$, can be reached by minimizing the square loss Bellman error $L = \frac{1}{N}\sum_i^N (y_i - Q(s_i, a_i|\theta^Q))^2$, where target y_i is defined as $r(s_i, a_i) + \gamma Q(s_{i+1}, \mu(s_{i+1}))$.

Recently, [6] proposed a continuous variant of Deep Q-Learning by a clever network construction. The Q network, which they called Normalized Advantage Function (NAF), parameterized the advantage function quadratically over the action space, and is weighted by non-linear feature of states.

$$Q(s, a|\theta^Q) = A(s, a|\theta^\mu, \theta^L) + V(s|\theta^V) \tag{2}$$

$$A(s, a|\theta^\mu, \theta^L) = -\frac{1}{2}(a - \mu(s|\theta^\mu))^T P(s|\theta^L) \\ (a - \mu(s|\theta^\mu)) \tag{3}$$

$$P(s|\theta^L) = L(s|\theta^L)^T L(s|\theta^L) \tag{4}$$

During run-time, the greedy policy can be performed by simply taking the output of sub-network $a = \mu(s|\theta^\mu)$. The data flow at forward prediction and back-propagation steps are shown in Fig. 8 (a) and (b), respectively.

A.2 Deep Deterministic Policy Gradient (DDPG)

An alternative approach to continuous RL tasks was the use of an actor-critic framework, which maintains an explicit policy function, called *actor*, and an action-value function called as *critic*. In [33], a novel *deterministic* policy gradient (DPG) approach was proposed and it was shown that deterministic policy gradients have a model-free form and follow the gradient of the action-value function.

$$\nabla_{\theta^\mu} J = \mathbb{E}[\nabla_a Q(s, a|\theta^Q) \nabla_a \mu(s)] \tag{5}$$

[33] proved that using the policy gradient calculated in (5) to update model parameters leads to the maximum expected reward.

Building on this result, [7] proposed an extension of DPG with deep architecture to generalize their prior success with discrete action spaces [4] onto continuous spaces. Using the DPG, an off-policy algorithm was developed to estimate the Q function using a differentiable function approximator. Similar techniques as in [4] were utilized for stable learning. In order to explore the full state and action space, an exploration policy was constructed by adding Ornstein-Uhlenbeck noise process. The data flow for prediction and back-propagation steps are shown in Fig. 8 (c) and (d), respectively.

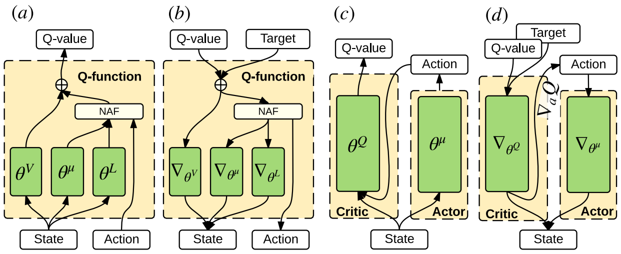


Figure 8: Schematic illustration of (a) forward and (b) back-propagation for NAF, and (c) forward and (d) back-propagation for DDPG. Green modules are functions approximated with Deep Nets.

Table 3: Model Specification

Model ID	State Dimensionality	Description
Physical	10	
Lasers	4 × 19	4 consecutive laser scans
Images	12 × 64 × 64	4 consecutive RGB image
Multi	10+1×19+3×64×64	all sensor streams at current time step

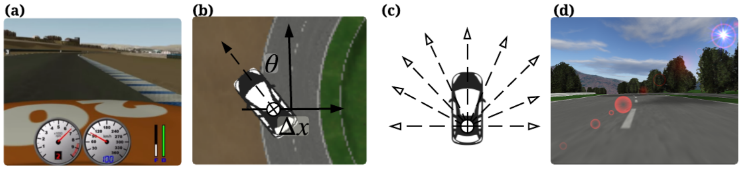


Figure 9: Sensors used in the TORCS racing car simulator: *Sensor 1:* Physical information such as velocity (a), position, and orientation (b), *Sensor 2:* Laser range finder (c), and *Sensor 3:* Front-view camera (d). Sensor dimensionality details listed in Sec. 5.1.

B Experiment Details

B.1 Exploration and Reward

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 [7] and [24] 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 [34].

B.2 Network Architecture

For laser feature extraction module, we use two $1D$ convolution layers with 4 filters of size 4×1, while image feature extraction is composed of three $2D$ convolution layers: one layer of 16 filters of size 4×4 and striding length 4, followed by two layers each with 32 filters of size 2×2 and striding length 2. Batch normalization is followed after every convolution layer. All these extraction modules are fused and are later followed up with two fully-connected layers of 200 hidden units each. All hidden layers have *relu* activations. The final layer of the critic network use *leaner* activation, while the output of the actor network are bounded using *tanh* activation. We use sigmoid activation for the output of L network in NAF. In practice, it leads to a more stable training for high dimensional state space. We trained with minibatch size of 16.

We used Adam [35] for learning the network parameters. For DDPG, the learning rates for actor and critic are 10^{-4} and 10^{-3}, respectively. We allow the actor and critic to maintain its own feature extraction module. In practice, sharing the same extraction module can lead to unstable training. Note that the NAF algorithm maintains three separate networks, which represent the value function ($V(s|\theta^V)$), policy network ($\mu(s|\theta^\mu)$), and the state-dependent covariance matrix in the action space ($P(s|\theta^L)$), respectively. In order to maintain a similar experiment setting and avoid unstable training, we maintain two independent feature extraction modules for θ^μ, and both θ^V and θ^L. In a similar vein, we apply a learning rate of 10^{-4} for θ^μ, and 10^{-3} for both θ^μ and θ^V.

B.3 Simulated Sensor Detail

As shown in Fig. 9, the physical state is a 10 DOF hybrid state, 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). Each laser scan is composed of 19 readings spanning a 180° field-of-view in the the front of car. Finally, camera provides RGB channels with resolution 64×64.

Table 4: Covariance of the first three Principal Component

	NAF			DDPG		
Principal Component	w/oSD	w/SD	w/SD+aux	w/oSD	w/SD	w/SD+aux
First (%)	94.9	82.0	58.9	93.4	59.2	47.4
Second (%)	4.1	12.3	25.2	3.1	20.7	21.9
Third (%)	0.6	3.1	5.3	1.6	6.2	6.1

Table 5: Action Variation w.r.t. multimodal sensor

	NAF			DDPG		
	w/oSD	w/SD	w/SD+aux	w/oSD	w/SD	w/SD+aux
Steering	0.1177	0.0819	**0.0135**	0.3329	0.0302	**0.0290**
Acceleration	0.4559	0.0472	**0.0186**	0.5714	0.0427	**0.0143**

C More Experimental Results

C.1 Effect of Auxiliary Loss

The covariance of PCA and the actual action variance is summarized in Table 4 and 5, respectively.