May 2, 2017
DRAFT

Some fancy title

Guan-Horng Liu

May 2, 2017

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
George A. Kantor
Manuela Veloso
Devin Schwab

*Submitted in partial fulfillment of the requirements
for the degree of Masters of Computer Science.*

Copyright © 2017 Guan-Horng Liu

May 2, 2017
DRAFT

Abstract

TODO

May 2, 2017
DRAFT

May 2, 2017
DRAFT

Acknowledgments

TODO

May 2, 2017
DRAFT

May 2, 2017
DRAFT

Contents

1 Introduction **1**

2 Kinodynamic Planning for All-Terrain Vehicle **3**

3 Deep Inverse Reinforcement Learning **5**

4 Multimodal Deep Reinforcement Learning **7**
4.1 Introduction 7
4.2 Background 9
4.2.1 Deep Reinforcement Learning (DRL) 9
4.2.2 Normalized Advantage Function (NAF) 9
4.2.3 Deep Deterministic Policy Gradient (DDPG) 10
4.3 Multimodal Deep Reinforcement Learning 11
4.3.1 Multimodal Network Architecture 11
4.3.2 Sensor Dropout (SD) 11
4.3.3 Platform Setup 14
4.3.4 Results 15
4.3.5 Probing through the gradients 16
4.4 Discussion 18
4.4.1 Sensor Dropout v/s *traditional* Dropout 18
4.4.2 Fusion Layer Configurations 19
4.4.3 Implementation of NAF with Multi-modal 19
4.5 Conclusions and Future Work 20

May 2, 2017
DRAFT

List of Figures

4.1 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. 10
4.2 Illustration of Multimodal Architecture and Sensor Dropout. The feature extraction module can be either pure identity function (modality 1), or convolution-based layer (modality $2 \rightarrow M$). The operation $*$ stands for element-wised multiplication. 12
4.3 Training performance comparison of three baseline single sensor policies, and the proposed multi-modal policies, with and without Sensor Dropout. 15
4.4 Policy Robustness Analysis: Darker lines connects average rewards of leaned policies with accurate sensing while the lighter lines connects the corresponding policies in the face of sensor noise. 16
4.5 Policy Sensitivity to Sensor Failure: The blue, green, and red bars denote full multi-modal policy, multi-modal with no image input, and multi-modal with no laser and physical state input, respectively. 17
4.6 The visualization of the magnitude of gradient for each neuron on training environment. The whiter color means the higher gradient. The color bar represents three different sensor modules: physical state(blue), Laser(green), and Image(red). 18
4.7 The gradient responses of actions on the image input for each of the multi-modal agents. The top 20% gradients are marked red. 19

May 2, 2017
DRAFT

May 2, 2017
DRAFT

List of Tables

4.1 Results of the $\mathcal{T}_2^1$ dependency metric. 16

May 2, 2017
DRAFT

Chapter 1

Introduction

One of the key challenges to building accurate and 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 subsystem and later integrated with some fine tuning. More recently however, there is great interest in applying a more end-to-end approach 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 approach is purported to perform better since we are optimizing the whole system.

This end-to-end approach has now become a viable alternative thanks to the extension of deep learning's success to robotics and has been applied successfully to autonomous driving [**?** **?** **?**]. However, the traditional deep supervised learning-based driving requires labeling and may not be able to deal with the problem of accumulating errors [**?**]. On the other hand, deep reinforcement learning (DRL) provides a much better formulation that allows policy improvement with feedback, and has been shown to achieve human-level performance on many gaming environments [**?** **?** **?**].

Previous work 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 task where enhancing safety and accuracy to the maximum extent possible is emphasized, developing policies that operate with multiple inputs is crucial. Indeed, 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-to-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.

Recently, there is also an great progress on extending DRL approaches to multi-input fashion in order to tackle complex robotics tasks such as human-robot-interaction [**?**] and manipulation [**?**]. Recently, **?**] proposed an novel approach namely *NAV A3C* and embedded information such as vision, depth, and agent velocity for maze navigation. However, it is worth mentioned that the system dynamic is relatively simple compared with autonomous driving.

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 [**?**]. To show the effectiveness of Multi-modal Perception, we picked two popular continuous action DRL algorithms namely Normalized Advantage Function (NAF) [?] and Deep Deterministic Policy Gradient (DDPG) [?], and augmented them to accept multi-modal 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 multi-modal configurations using extensive quantitative and qualitative testing. We believe however that it is representative of the larger urban navigation problem, and we ensure that our proposed network architecture is task-agnostic and transferable. We show through extensive empirical testing that a multi-modal deep reinforcement learning architecture achieves higher average reward.

Moreover, we also observe that while a multi-modal policy greatly improves the reward, it might rely heavily on all the inputs to the extent that it may fail completely even if a single sensor broke down fully or partially. This undesirable consequence renders sensor redundancy useless. To ensure that the learned policy does not succumb to such over-fitting, we apply a novel stochastic regularization method called *Sensor Dropout* during training.

Stochastic regularization is an active area of research in deep learning made popular by the succes of, *Dropout* [?]. Following this landmark paper, numerous extensions were proposed to further generalize this idea such as *Blockout* [?], *DropConnect* [?], *Zoneout* [?], etc. In the similar vein, two interesting techniques have been proposed for specialized regularization in the multi-modal setting namely ModDrop [?] and ModOut [?]. 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 assume that all the sensors are critical and we only focus on improving the state information based on inputs from multiple observers. ModDrop is much closer in spirit to the proposed *Sensor Dropout (SD)*. However, unlike ModDrop, pretraining with individual sensor inputs using separate loss functions is not required. A network can be directly constructed in an end-to-end fashion and *Sensor Dropout* can be directly applied at the sensor fusion layer just like Dropout. 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.

Through extensive empirical testing we show the following exciting results in this paper:

1. Multimodal-DRL with Sensor Dropout (SD) reduces performance drop in a noisy environment from $\approx 30\%$ to just 5%, when compared to a baseline single sensor system.

2. Policies learned using SD best leverage the multi-modal setting by greatly reducing over-dependence on any one sensing modality. Additionally, for each sensor it was observed that SD enforces sparsity and promotes each sensor to base the policy primarily on intuitive and salient features.

3. A multi-modal policy with SD guarantees functionality even in a face a sensor failure. This is a huge plus and the best use-case for the need for redundancy in safety-critical application like autonomous navigation.

May 2, 2017
DRAFT

Chapter 2

Kinodynamic Planning for All-Terrain Vehicle

TODO

May 2, 2017
DRAFT

May 2, 2017
DRAFT

Chapter 3

Deep Inverse Reinforcement Learning

TODO

Chapter 4

Multimodal Deep Reinforcement Learning

4.1 Introduction

One of the key challenges to building accurate and 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 subsystem and later integrated with some fine tuning. More recently however, there is great interest in applying a more end-to-end approach 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 approach is purported to perform better since we are optimizing the whole system.

This end-to-end approach has now become a viable alternative thanks to the extension of deep learning's success to robotics and has been applied successfully to autonomous driving [**?** **?** **?**]. However, the traditional deep supervised learning-based driving requires labeling and may not be able to deal with the problem of accumulating errors [**?**]. On the other hand, deep reinforcement learning (DRL) provides a much better formulation that allows policy improvement with feedback, and has been shown to achieve human-level performance on many gaming environments [**?** **?** **?**].

Previous work 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 task where enhancing safety and accuracy to the maximum extent possible is emphasized, developing policies that operate with multiple inputs is crucial. Indeed, 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-to-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.

Recently, there is also an great progress on extending DRL approaches to multi-input fashion in order to tackle complex robotics tasks such as human-robot-interaction [**?**] and manipulation [**?**]. Recently, **?**] proposed an novel approach namely *NAV A3C* and embedded information such as vision, depth, and agent velocity for maze navigation. However, it is worth mentioned that the system dynamic is relatively simple compared with autonomous driving.

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 [**?**]. To show the effectiveness of Multi-modal Perception, we picked two popular continuous action DRL algorithms namely Normalized Advantage Function (NAF) [**?**] and Deep Deterministic Policy Gradient (DDPG) [**?**], and augmented them to accept multi-modal 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 multi-modal configurations using extensive quantitative and qualitative testing. We believe however that it is representative of the larger urban navigation problem, and we ensure that our proposed network architecture is task-agnostic and transferable. We show through extensive empirical testing that a multi-modal deep reinforcement learning architecture achieves higher average reward.

Moreover, we also observe that while a multi-modal policy greatly improves the reward, it might rely heavily on all the inputs to the extent that it may fail completely even if a single sensor broke down fully or partially. This undesirable consequence renders sensor redundancy useless. To ensure that the learned policy does not succumb to such over-fitting, we apply a novel stochastic regularization method called *Sensor Dropout* during training.

Stochastic regularization is an active area of research in deep learning made popular by the succes of, *Dropout* [**?**]. Following this landmark paper, numerous extensions were proposed to further generalize this idea such as *Blockout* [**?**], *DropConnect* [**?**], *Zoneout* [**?**], etc. In the similar vein, two interesting techniques have been proposed for specialized regularization in the multi-modal setting namely ModDrop [**?**] and ModOut [**?**]. 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 assume that all the sensors are critical and we only focus on improving the state information based on inputs from multiple observers. ModDrop is much closer in spirit to the proposed *Sensor Dropout (SD)*. However, unlike ModDrop, pretraining with individual sensor inputs using separate loss functions is not required. A network can be directly constructed in an end-to-end fashion and *Sensor Dropout* can be directly applied at the sensor fusion layer just like Dropout. 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.

Through extensive empirical testing we show the following exciting results in this paper:

1. Multimodal-DRL with Sensor Dropout (SD) reduces performance drop in a noisy environment from $\approx 30\%$ to just 5%, when compared to a baseline single sensor system.
2. Policies learned using SD best leverage the multi-modal setting by greatly reducing over-dependence on any one sensing modality. Additionally, for each sensor it was observed that SD enforces sparsity and promotes each sensor to base the policy primarily on intuitive and salient features.
3. A multi-modal policy with SD guarantees functionality even in a face a sensor failure. This is a huge plus and the best use-case for the need for redundancy in safety-critical

May 2, 2017
DRAFT

application like autonomous navigation.

4.2 Background

4.2.1 Deep Reinforcement Learning (DRL)

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.

$$V^\pi(s_t) = \mathbb{E}_{r_{i \geq t}, s_{i>t} \sim E, a_{i \geq t} \sim \pi}[R_t | a_t, s_t] \tag{4.1}$$

$$\begin{aligned} Q^\pi(s_t, a_t) = \mathbb{E}_{r_{i \geq t}, s_{i>t} \sim E}[&r(s_t, a_t) \\ &+ \gamma \mathbb{E}_{a_{i>t} \sim \pi}[Q^\pi(s_{t+1}, a_{t+1})]] \end{aligned} \tag{4.2}$$

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 its 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 for both Q and V functions would lead to unstable learning in practice. Recently, **?**] 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 [**? ? ? ?**].

To exhaustively analyze the effect of multi-sensor input and the new stochastic regularization technique, we picked 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 multiple inputs. Before introducing the multi-modal architecture, we briefly summarize the two algorithms below.

4.2.2 Normalized Advantage Function (NAF)

Q-learning [**?**] 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 $r(s_i, a_i) + \gamma Q(s_{i+1}, \mu(s_{i+1}))$.

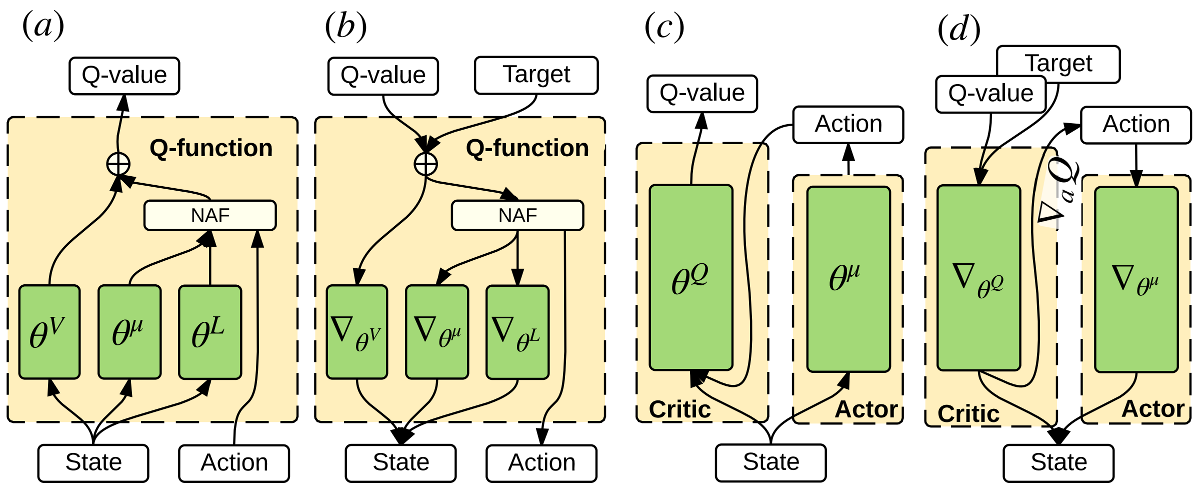


Figure 4.1: Schematic illustration of (a) forward and (b) back-propagation for NAF, and (c) forward and (d) back-propagation for DDPG. Green modules are functions approximated with Deep Nets.

Recently, **?**] 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.

$$Q(s, a|\theta^Q) = A(s, a|\theta^\mu, \theta^L) + V(s|\theta^V) \tag{4.3}$$

$$A(s, a|\theta^\mu, \theta^L) = -\frac{1}{2}(a - \mu(s|\theta^\mu))^T P(s|\theta^L) (a - \mu(s|\theta^\mu)) \tag{4.4}$$

$$P(s|\theta^L) = L(s|\theta^L)^T L(s|\theta^L) \tag{4.5}$$

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. 4.1 (a) and (b), respectively.

4.2.3 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 **?**], 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.

$$\nabla_{\theta^\mu} J = \mathbb{E}[\nabla_a Q(s, a|\theta^Q) \nabla_a \mu(s)] \tag{4.6}$$

?] proved that using the policy gradient calculated in (4.6) to update model parameters leads to the maximum expected reward.

Building on this result, **?**] proposed an extension of DPG with deep architecture to generalize their prior success with discrete action spaces [**?**] 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 [**?**] 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. 4.1 (c) and (d), respectively.

4.3 Multimodal Deep Reinforcement Learning

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 though, the resulting observer will have a good and reliable estimate of the environment.

4.3.1 Multimodal Network 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 independent paths. Each path has a feature extraction module with an appropriate network architecture, using randomly initialized or pre-trained weights. In this work, we use three different inputs namely image, laser scan and physical parameters (like wheel speed, position, odometry, etc. The details of each of the feature extraction module are listed in the Appendix. 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 schematic illustration of modularized Multi-modal architecture is shown in Fig. 4.2. The outputs of feature extraction modules are eventually flattened and concatenated to form the multi-modal state.

4.3.2 Sensor Dropout (SD)

As shown in Fig.4.2, consider the multi-modal state $\tilde{S}$, obtained from feature extraction and given by $\tilde{S} = [\tilde{S}^{(1)}\ \tilde{S}^{(2)}\ ..\ \tilde{S}^{(M)}]^T$, where $\tilde{S}^{(i)} = [\tilde{X}_1^{(i)}\ \tilde{X}_2^{(i)}\ ..\ \tilde{X}_{K_i}^{(i)}]^T$. The objective is to generate a random mask $\mathbf{c} = [\delta_c^{(1)}\ \delta_c^{(2)}\ ..\ \delta_c^{(M)}]^T$, where $\delta_c^{(i)} \in \{0, 1\}$ represents the on/off indicator for the i^{th} modality. However, we need that at least one sensor remains *on* at any given instant in order to maintain training stability. Therefore, we cannot naively extend original dropout idea to sensor blocks $\tilde{S}^{(i)}$.

Let each block's on/off status be sampled from a Bernoulli distribution. We can generate 2^M random layer configurations masks and one of them is all the blocks switched off, i.e., $\mathbf{c_0} = [0^{(1)}\ 0^{(2)}\ ..\ 0^{(M)}]^T$. The probability of this event occurring decreases exponentially in the size of M, therefore not having much impact as formulated in **?**], where its applied to individual nodes of a layer. However, when M is small as in the case of Sensor Dropout, the probability of

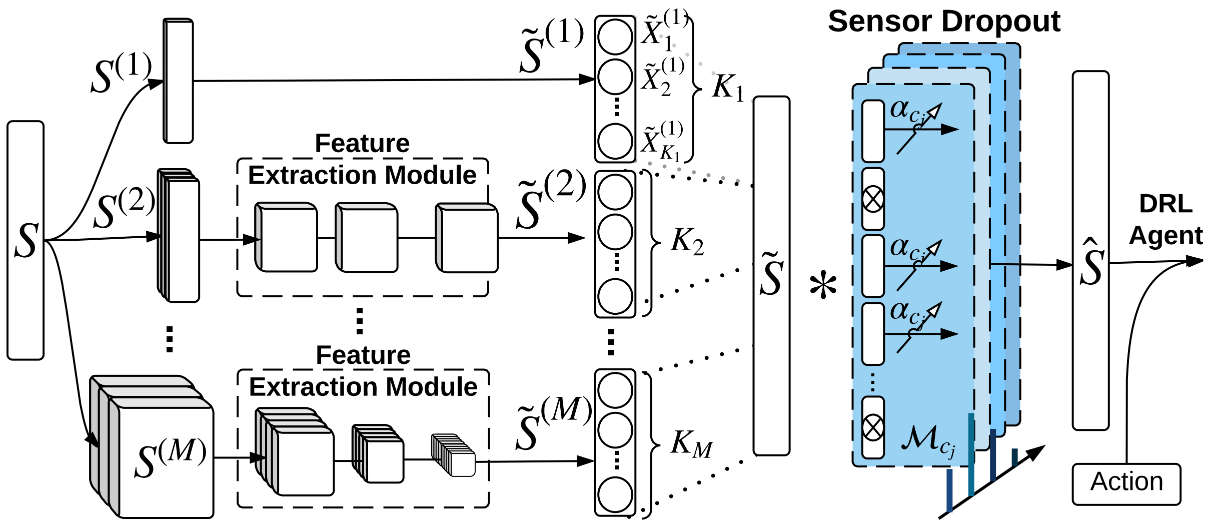


Figure 4.2: Illustration of Multimodal Architecture and Sensor Dropout. The feature extraction module can be either pure identity function (modality 1), or convolution-based layer (modality $2 \rightarrow M$). The operation $*$ stands for element-wised multiplication.

$\mathbf{c_0}$ occurring is high enough to severely impact training performance. Therefore, we explicitly remove this rogue case and only consider $2^M - 1$ layer configuration masks. Another motivation for Sensor Dropout is to allow for each block $\tilde{S}^{(i)}$ to have a unique dropout probability $p^{(i)}$, if required, with the added cost of more hyper-parameter tuning. However, this tuning is tractable and meaningful in this setup.

Finally, for this work, we slightly depart from **?**] in modeling our dropout layer. Instead of modeling Dropout as random process where any sensor block $\tilde{S}^{(i)}$ of the layer is switched on/off with a *fix* probability p, we propose an alternative view where the random variable is the layer configuration $\mathbf{c}$ itself and not individual block state $\tilde{S}^{(i)}$. Since there are $N = 2^M - 1$ possible states for $\mathbf{c}$, we accordingly sample from an N-state categorical distribution. We denote the probability of a layer configuration $\mathbf{c_j}$ occurring with p_j, where the subscript j ranges from 1 to N. It is easy to derive the corresponding pseudo-Bernoulli[1] distribution for switching on a sensor block $\tilde{S}^{(i)}$. Let us denote that as $p^{(i)}$.

$$p^{(i)} = \sum_{j=1}^{N} \delta_{c_j}^{(i)} p_j \tag{4.7}$$

Once the sensor dropout mask is applied, we have to deal with the issue of rescaling the non-zero weights. For this, we follow the same convention as in **?**] but extend it to sensor blocks as

[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.

Algorithm 1 M-DRL with Sensor Dropout

```
Input: Initialize DRL Algorithm AGENT,
       State pre-processed network $f(\cdot|\theta)$,
       $N$ network configurations $\mathbb{C} = \{c_j\}_{j=1\,..\,N}$,
       $N$-state categorical distribution $\mathbb{P}(X = c_j) = p_j$,
Initialize experience replay buffers $\mathcal{B} = \{\mathcal{B}_j\}_{j=1\,..\,N}$
for $episode = 1$ to $Eps$ do
    Receive initial state $s_1$
    for $t = 1$ to $T$ do
        $\tilde{s}_t \leftarrow f(s_t|\theta)$
        $\hat{s}_t \leftarrow$ SENSORDROPOUT$(\tilde{s}_t, c_j)$, where $c_j \sim \mathbb{P}$
        $a_t \leftarrow$AGENT.ACT$(\hat{s}_t)$
        $r_t, s_{t+1}, terminate \leftarrow$ ENV.OBSERVE$(s_t, a_t)$
        $\mathcal{B}_j \leftarrow$ QUERY$(\mathcal{B}, c_j)$
        $\mathcal{B}_j$.APPEND$(s_t, a_t, r_t, s_{t+1})$
        AGENT.UPDATE$(\mathcal{B}_j)$
    end for
end for=0
```

opposed to individual nodes.

$$\alpha_{c_j} = \frac{\sum_{i=1}^{M} K_i}{\sum_{i=1}^{M} \delta_{c_j}^{(i)} K_i}. \tag{4.8}$$

Finally, the output of Sensor Dropout for the k^{th} node in i^{th} sensor block $\tilde{S}^{(i)}$ for some layer configuration $\mathbf{c_j}$ can be shown as,

$$\hat{S}_{c_j,k}^{(i)} = \mathcal{M}_{c_j}^{(i)} \tilde{X}_k^{(i)}, \qquad \text{where } \mathcal{M}_{c_j}^{(i)} = \alpha_{c_j} \delta_{c_j}^{(i)}. \tag{4.9}$$

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

Augmenting Experience Replay

Experience Replay is a commonly used technique in many off-policy DRL algorithms to break the time-based correlation in the agent's experience and thereby stabilize the training process. It differs from the standard deep supervised learning setting, in that batches are sampled from a memory buffer containing all the experience history. However, extending a dropout-like regularization technique to DRL setting to operate on the experience replay is non-trivial and can cause instability because the target values during training are usually non-stationary. In the following paragraph we investigate the instability issue of dropout-like regularization and argue that it is only combinatorially tractable if blocks are dropped out instead of individual nodes. The Sensor Dropout is not only principled but also well-motivated.

First, let us denote an instance of the experience as $[s_t, a_t, r_t, s_{t+1}]$, where $a_t = \mu(s_t, \tilde{\mathcal{M}_t})$ is generated from the policy network, with a random mask $\tilde{\mathcal{M}_t}$ sampled from an arbitrary distribution at the time step t. Note that under the dropout setting, the experience is now *mask* *configuration dependent*. During Bellman's updates, we calculate the one-step look-ahead target value with $Q^{'}(s_{t+1}, \mu^{'}(s_{t+1}, \tilde{\mathcal{M}}_{t+1}))$. If two masks, i.e. $\tilde{\mathcal{M}}_t$ and $\tilde{\mathcal{M}}_{t+1}$ are not the same, the behavior of μ and $\mu^{'}$ can be dramatircally different at least in the *beginning of training* as they use different sensor modalities as input and thus causing instability during training. However, under the implementation of Sensor Dropout, this issue can be mitigated by simply maintaining multiple replay buffers for each of the sensor drop configuration $\mathbf{c_j}$. This is not possible with original Dropout as the number of replay buffers will grow exponentially with layer size. Sensor Dropout, on the other hand, makes this tractable as it only grows in the number of sensors. During batch updates, a specific replay buffer is queried given the current layer configuration $\mathbf{c_j}$, 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.

The overall Multimodal framework with Sensor Dropout regularization is summarized in Algorithm 4.3.2. It can be seen from the above discussion that Sensor Dropout approach is general enough to work with any existing off-policy DRL algorithm in a multi-modal setting. For on-policy algorithms however, Sensor Dropout can be implemented by fixing layer configuration $\mathbf{c_j}$ for every episode. In the next section, we validate the above-mentioned techniques in an open-source racing game called TORCS [**?**] and analyze their efficacy.

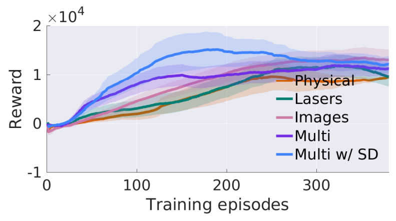
4.3.3 Platform Setup

TORCS Simulator:

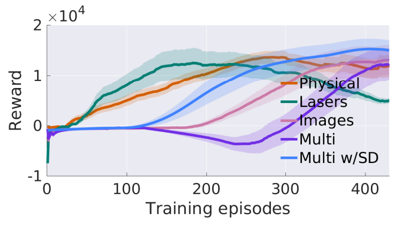
The proposed approach was verified on TORCS [**?**], a popular open-source car racing simulator that is capable of simulating physically realistic vehicle dynamics as well as multiple sensing modalities [**?**] to build sophisticated AI agents. In order to make the learning problem representative of the real-world setting, we picked the following sensors from the TORCS package: 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. Moreover, all the actions are bounded and for this learning problem they are also normalized.

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 **?**] and **?**] 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 [**?**].

State Information: Most of the related works use either low-dimensional physical state such as joint angles, or high-dimensional pixels as input. Here, 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 (i.e., at time t, we input scans from times $t,\ t-1,\ t-2\ \&\ t-3$). 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.



(a) NAF



(b) DDPG

Figure 4.3: Training performance comparison of three baseline single sensor policies, and the proposed multi-modal policies, with and without Sensor Dropout.

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. A more detailed commentary on sensor-fusion configurations follows in Section 4.4.2, with network details listed in the Appendix.

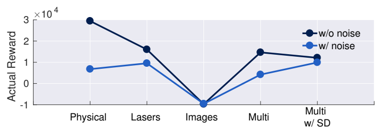
4.3.4 Results

Convergence: The training performance, for all the proposed models and their corresponding baselines, is shown in Fig. 4.3. As expected, using high dimensional sensory input directly impacts convergence rate of the policy. However, despite the curse of dimensionality, the convergence rate improves with the addition of Sensor Dropout, and quite drastically so for DDPG (see Fig. 4.3b).

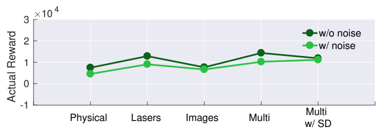
To assess the degree of generalization of the learned policies, we pick the the best learned policies for each model and test its performance on other racing tracks. As summarized in Fig. **??**, we find that the DDPG agent is capable of finding a reasonable policy regardless of the representation of states. The performance of NAF on the other hand is greatly affected by the dimensionality of the agent's input. In fact, the agent fails to achieve even a non-zero reward during testing, when trained with image as input. However, using multi-sensor information and thereby reducing the fused input dimension aids in learning a good policy. Network architecture details are described in Section 4.4.3.

Robustness: Note that, we assumed 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 is plotted for comparison in Fig. **??**. With the addition of noise the performance drop is sometimes severe in a single input policy, as seen for NAF with physical state input. In comparison, the drop is ore contained for the multi-modal policy and almost negligible when Sensor Dropout is used.

Sensitivity Analysis: In this part, we further validate our hypothesis Sensor Dropout (SD) reduces the learned policy's acute dependence on a subset of sensors in a multi-modal setting. First, we considered a scenario when malfunction of a sensor has been detected by the system, and the agent need to rely on the remaining sensors to make the decision. During testing, we



(a) NAF



(b) DDPG

Figure 4.4: Policy Robustness Analysis: Darker lines connects average rewards of leaned policies with accurate sensing while the lighter lines connects the corresponding policies in the face of sensor noise.

Table 4.1: Results of the $\mathcal{T}_2^1$ dependency metric.

		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**

manually blocked off part of the sensor modules, and scaled the rest of observation using the same rescaling mechanism as proposed in Section 4.3.2. Fig. 4.5 reports the average of the normalized reward, i.e. total reward divided by total steps, of each model. For both the algorithms, models trained with SD performed comparatively better when decisions were forced to depend on a subset of sensors than the ones trained without SD. We observed that the policy could either be highly correlated, which is the case of NAF w/o SD in Fig. 4.5, or depend on sensor subset, as shown in DDPG w/o SD in Fig. 4.5. We therefore emphasize that introducing Sensor Dropout during training has significant practical benefits, in that the policies induced by different configurations of sensors are learned at the same time.

4.3.5 Probing through the gradients

Dependency metric:

To further examine the impact of Sensor Dropout on effective sensor fusion, we introduce an intuitive method for monitoring the extent to which the learned policy depends on each sensor block by measuring the gradient of the policy output w.r.t a subset block $\tilde{S}^{(i)}$. In a multi-input-to-output mapping, this denotes the relative weights/importance of each input Since in this work, SD was implemented to drop either (1) $(physical\ state,\ laser)$ or (2) $vision$, we define the *dependency* metric for this problem as the ratio between configurations (1) and (2), as shown

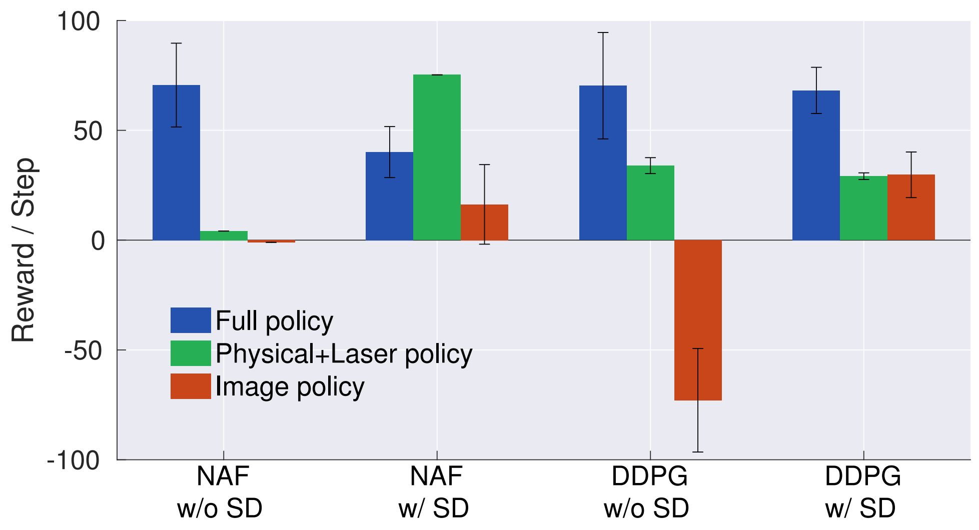


Figure 4.5: Policy Sensitivity to Sensor Failure: The blue, green, and red bars denote full multi-modal policy, multi-modal with no image input, and multi-modal with no laser and physical state input, respectively.

below.

$$\mathcal{T}_2^1 = \frac{1}{M}\sum_{i=1}^{M} \frac{\left| \nabla_{\tilde{S}_i^{(1)}} \mu(\tilde{S}|\theta^\mu)\Big|_{S_i} \right|}{\left| \nabla_{\tilde{S}_i^{(2)}} \mu(\tilde{S}|\theta^\mu)\Big|_{S_i} \right|} \tag{4.10}$$

Assuming the fusion-of-interest is between the above-mentioned two subsets, we intend to show that, using SD, the metric should get closer to one, indicating nearly equal importance to both the sensing modalities. This metric is evaluated for policies learned with and without SD for NAF and DDPG and the mean values are reported in Table 4.1. For this, the data was collected from a stable policy by evaluating it on both training and testing environments. We can verify that, using SD, NAF policies become dramatically more dependent all sensors. Additionally, we visualize the dependence on each neuron by estimating the average gradient. As shown in Fig. 4.6, models trained with SD tend to make better use of visual information (location is indicated by the red bar.). Without SD however, agents tend to rely more heavily on physical state and laser inputs.

Visualizing weights:

As shown in Fig. 4.6, we observe that models 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. Clearly, the network is able to automatically infer and weight locations providing salient information. This behavior is consistent in both DDPG and NAF.

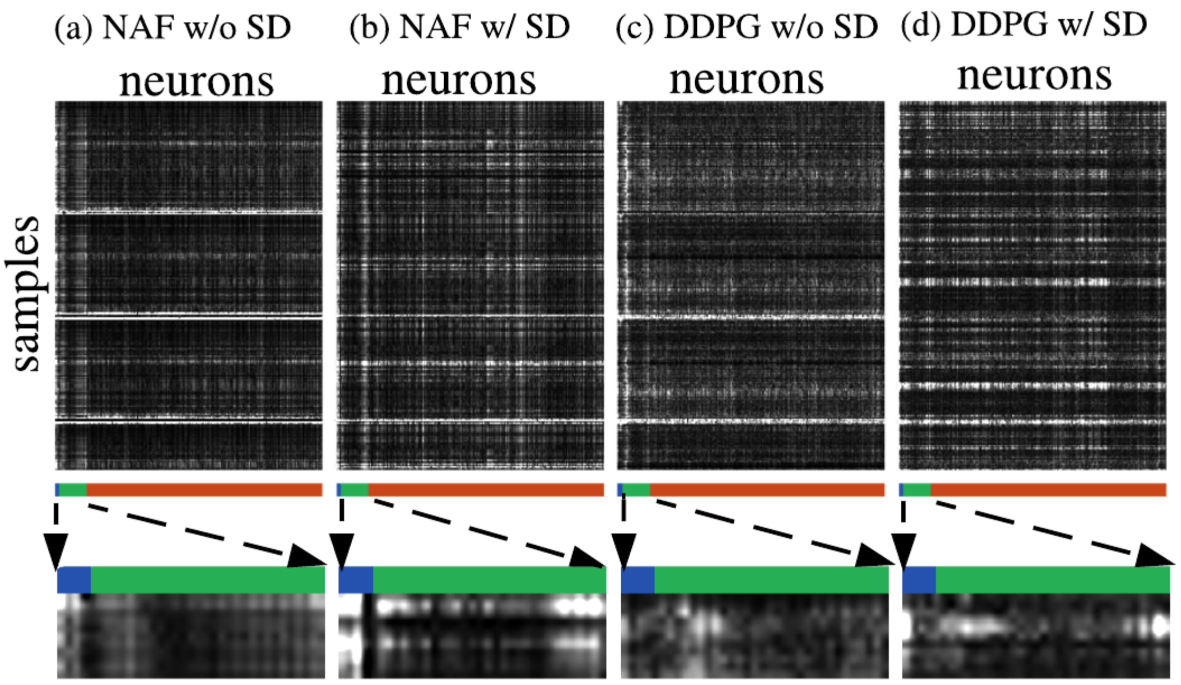


Figure 4.6: The visualization of the magnitude of gradient for each neuron on training environment. The whiter color means the higher gradient. The color bar represents three different sensor modules: physical state(blue), Laser(green), and Image(red).

To look for similar patterns, in Fig. 4.7, image pixels with higher gradients are marked to visualize and 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. Models trained with SD tend to capture relevant visual information such as road boundary. In comparisons, models trained without SD have a relatively low and unclear gradients over both laser and image sensor state space.

4.4 Discussion

4.4.1 Sensor Dropout v/s *traditional* Dropout

In addition to demonstrating the value of using Sensor Dropout, it also critical to verify whether similar performance can be extracted with Dropout or not. Therefore, we implemented the Dropout on our multi-modal agent with dropping probabilities of 0.5 and 0.25 as baselines. Both models performed poorly in the testing environment, even with perfect sensing. For DDPG, mean reward using dropout was only 2.2×10^3, compared with 1.1×10^4 and 1.4×10^4, obtained on vanilla-multi-modal policy and multi-modal + SD, respectively. Similar results were observed with NAF too. This furthers our claim that traditional implementation of dropout increases the

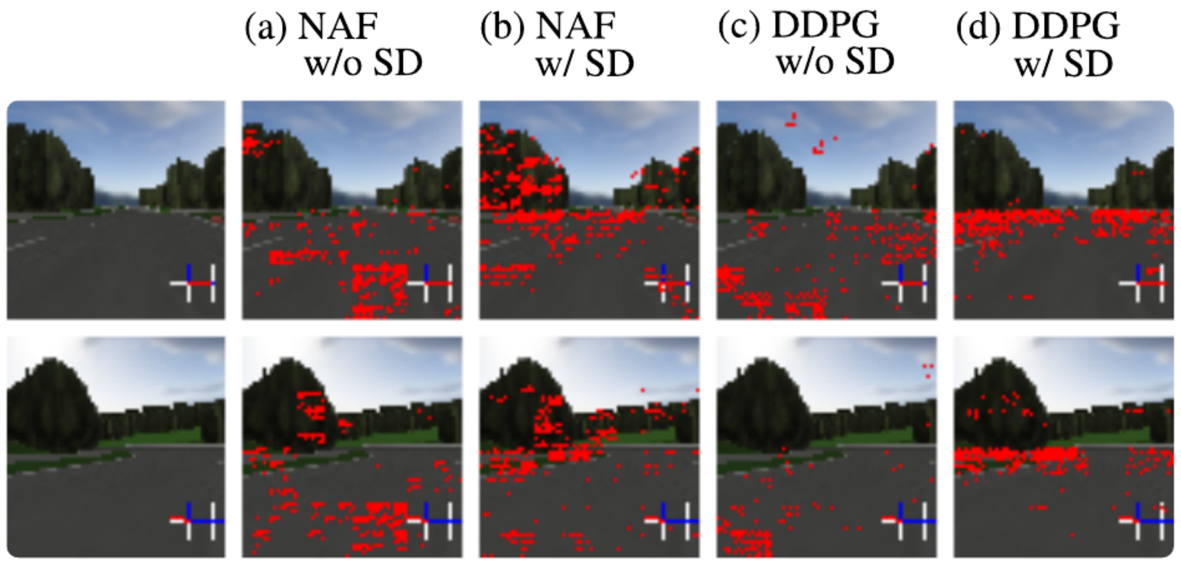


Figure 4.7: The gradient responses of actions on the image input for each of the multi-modal agents. The top 20% gradients are marked red.

risk of destabilizing the DRL policy.

4.4.2 Fusion Layer Configurations

Under the current implementation, the agent will observe the sensor data from one of the following three combinations during training: (1)$(physical\ state,\ laser)$, (2) $(image)$, or (3) $(physical\ state,\ laser,\ image)$. We have experimented with following three different settings:(a) $[p_1\ p_2] = [0.25\ 0.25]$, (b) $[p_1\ p_2] = [0.4\ 0.4]$, or (c) $[p_1\ p_2] = [0.5\ 0.5]$. Here, p_1 and p_2 represent the probability of using the first and second combinations, respectively, while the probability of the third combination p_3 is given by $p_3 = 1 - p_1 - p_2$. For the first two configurations, the agent has access to all sensors occasionally, but in the last case, the agent only has access to a subset of sensors at any given point.

We note that as the probabilities $p_1,\ p_2$ increase, the policy convergence rate and the normalized average rewards induced by each sensor subset also increase, as described in Section 4.3.4. However, the performance of the full policy decreases in both training and testing environment. This empirical result suggests a more principled way to apply Sensor Dropout. For instance, we can start with a high value for $(p_1,\ p_2)$ and benefit from faster policy convergence for each sensor subset and then gradually decrease the probability to promote fusion.

4.4.3 Implementation of NAF with Multi-modal

Since NAF is not designed for high-dimensional state space, unstable policies result if extended naively to operate with image as input. To mitigate this problem, we add two additional fullyconnected layers to reduce the state dimensionality to apply the NAF algorithm. We consider this slight modification as a useful and practical method to embed a low-dimensional representation of visual information. However, purely image-based policy learned with the NAF agent still results in an unstable behavior with large negative rewards *during testing*. This could be a case of over-fitting where the experience from training environment does not generalize well to fully operate in the state space, and the agent may have just memorized specific but irrelevant visual cues. Surprisingly, the multi-modal representation alleviated the issue by fusing visual state with other sensory information. The results specific to NAF and the above-mentioned findings alone offer great scope for greater probing and is saved for future work.

4.5 Conclusions and Future Work

In this work, we extend popular DRL algorithms like DDPG and NAF to the multi-modal setting. Additionally, we introduce a new stochastic regularization technique called Sensor Dropout to promote an effective fusing of information from multiple sensors.

For future work, we wish to extend Multi-modal DRL to other algorithms like ICNN [**?**], TRPO [**?**], GPS [**?**], and Q-Prop [**?**], etc. Secondly, we wish to augment the reward function to also perform other important driving tasks like collision avoidance, and lane changing, etc. Moreover, systematic investigation into the failures of image-based NAF policies and the choices for $p_1 and p_2$ (hyperparameter optimization) are also interesting avenues that merit further study.