LEARNING ARCHITECTURES AND ALGORITHMS FOR COLLISION AVOIDANCE AND SENSOR FUSION

A Dissertation

by

MYUNG SEOK SHIM

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Peng Li
Committee Members,	P. R. Kumar
	Byung-Jun Yoon
	Dezhen Song
Head of Department,	Miroslav Begovic

December 2020

Major Subject: Computer Engineering

ABSTRACT

The deep learning, which is a machine learning method based on artificial neural networks, enables cutting edge technologies such as autonomous driving, collision avoidance, activity recognition, etc. While various machine learning algorithms are proposed for these applications, these algorithms suffer from limitations in terms of multi-modal sensor fusion, feature-level importance interpretability, feature reduction and fusion architecture modeling. In this dissertation, the aforementioned problems are addressed through spiking neural networks (SNNs) and convolutional neural networks (CNNs). Spiking neural network based algorithms and convolutional neural network based architectures are proposed to incorporate interpretable sensor fusion, and robust learning algorithms for noise resiliency and high inference performance.

An SNN is an brain-inspired computing neural network which mimics brain activity more closely. Although various spike-timing-dependent plasticity (STDP) based SNNs which are unsupervised learning implementations were proposed for mobile robot collision avoidance, these algorithms only focused on simple environments and feature-fusion in the networks was not well considered. Along with the exploration of reinforcement learning implementation in SNN via additive reward-modulated STDP (A-RM-STDP), we propose a new multiplicative RM-STDP rule (M-RM-STDP) for mobile robot target tracking and collision avoidance. Furthermore, a more biologically plausible feed-forward SNN architecture including coarser grained global rewards is examined.

Sensor fusion technologies are getting attention due to their enormous potential for health care and autonomous systems. CNNs empower sensor fusion architectures with improved performance but are limited in terms of feature-importance analysis, fusion weights inconsistency, and noise resiliency. Deep multi-model sensor fusion architectures with robustness under sensor failure circumstances are proposed. The proposed fusion weight regularized gating network exceeds the baseline deep learning architectures with and without gating factors.

Furthermore, beyond the proposed gating architecture, we explore feature dimension reduction

for rare failure detection by learning from high dimensional analog/mixed-signal (AMS) circuit data. Convolution layers in CNNs effectively extract principal features from input data, but results in loss of information. A reversible residual network (RevNet) is adopted in our proposed gating architecture and explored to mitigate loss of feature information in the process of dimension reduction and to improve the performance of data-driven AMS circuit rare failure detection.

DEDICATION

To my mother, my father, my brother, and my love Swanie.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Professor Peng Li, who is my advisor. He has guided, and supported me over the last 6 years. Without his help, I could not become an independent researcher who define and resolve scientific problems. It was a great honor working with Professor. Li for my Ph.D program.

I would love to show my sincere appreciation to my committee members: Dr. P. R. Kumar, Dr. Byung-Jun Yoon, and Dr. Dezhen Song. They offered valuable ideas, suggestions and comments during the prelim and final exam. Dr. Kumar and Dr. Song provided guidance for the research problem.

I had great pleasure working with my colleagues in Computer Engineering & System Group. Dr. Yingyezhe Jin introduced and helped me to get some ideas on the spiking neural networks. Ph.D student Hanbin Hu gave advice on resversible neural networks and guided me to find better feature dimension reduction algorithm. We enjoyed fruitful discussion for our research. High Performance Research Computing (HPRC) at Texas A&M University always give me a hand for using their computing power.

Last, I would like to thank my parents and brother. My mom, Junghee Lee, dad, Deokjin Shim, and brother, Myungjoon Shim, always love me and encourage me. My love, Swanie Kang, love in me. Thanks for her being with me, sharing all of my feeling.

Special thanks to my awesome friends: Hyun-Myung Woo, Woorim Hong, Stella Kang, Serena Kang, Doowon Choi, Jaehyung Son, and Zhixing Li.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Peng Li, Professor P. R. Kumar, Professor Byung-Jun Yoon of the Department of Electrical and Computer Engineering and Professor Dezhen Song of the Department of Computer Science and Engineering.

All other work conducted for dissertation was completed by the student independently.

Funding Sources

This dissertation is based upon work supported by the National Science Foundation under Grants No. 1940761 and No. 1956313, and the Semiconductor Research Corporation (SRC) under Task 2692.001.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

With the advance progress of deep learning technologies, many applications are implemented and getting improved. Now we are using facial recognition as shown in Fig. 1.1 on our phone to unlock it or use the recognition for payment. Also, regarding autonomous driving technology in Fig. 1.2, which is placed in level3 currently, it is widely studied with many researchers and utilized in some automobiles. For health care products such as a smart watch in Fig. 1.3, many sensors including GPS, accelerometer, gyroscope, even ECG and blood oxygen sensors are used for checking our health, recognizing posture and predicting dangerous heart attack in our bodies. And, a deep learning algorithm recommends television shows based on our history of watching programs.

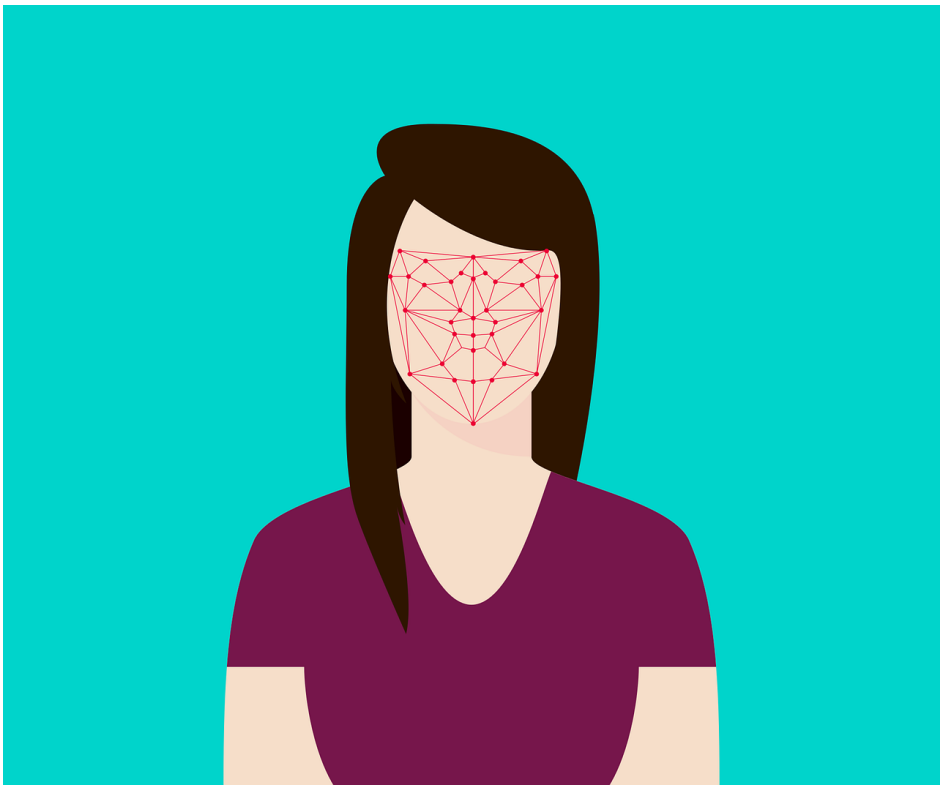

Figure 1.1: Facial recognition. Reprinted with permission from Pixabay

Figure 1.2: An autonomous vehicle. Reprinted with permission from Pixabay

For these cutting edge technologies, data is important and its size is expanding. For example, BDD100k [1] driving dataset, which is provided by UC Berkeley, composed of more than 100,000 HD video data over 1,100 hour driving experience. Stereo camera, lidar, and radar data are included in the dataset. Additionally, GPS, IMU sensor, and OBD2 from a vehicle are utilized for autonomous driving research. As an example of biological datasets with Dorothea [2] for RNA sequence analysis, there are 100,000 features.

In this dissertation, autonomous driving technology is focused in terms of safety. First, collision avoidance is examined. The collision avoidance is pre-crash or forward collision warning system with brake control. Nowadays, with machine learning algorithms, it is a system to make an agent have learning capabilities to avoid collisions in a specific environment. Many machine learning techniques are applied to implement collision avoidance. Since testing these algorithms on actual cars is difficult and dangerous, a mobile robot is often used as a test bed.

The second challenge is neural network architectures for data fusion. Since the number of sensory inputs and features is increasing, the way how to treat these inputs is important. For au-

Figure 1.3: A smart watch. Reprinted with permission from Pixabay

tonomous driving, many sensors are needed such as stereo camera, lidar, radar, GPS, IMU, OBD2, etc. Furthermore, regarding feature importance, there are some essential features and redundant features in datasets for target prediction. The inference performance may be boosted by weighting the important features in neural networks. Therefore, the method of fusing all sensory inputs is one of essential topic for robust autonomous driving.

The last challenge is data itself. Due to growing complexity in high dimensional data from automotive, medical and analog and mixed-signal (AMS) area, it is important to find active input features. If the dimension is too large, and a large neural network is needed accordingly, which needs more computational power. In the second challenge, we will explore how to fuse the input data with fusion weights, which can tell us important and non-important features. Now, another point of view on the active input features on the AMS datasets are demonstrated. Since the AMS data is high-dimensional but has low amount of samples, the neural network for these datasets can be stuck at overfitting easily due to the curse of dimensionality.

Furthermore, very high level of reliability is required in the circuit industry such as failure rate less than 1 DPPM (Defective parts per million). To get circuit data, it is expensive running circuit

simulations. Hence, with a large number of input features but only with small amount of samples, it is challenging to find rare failure points. To solve the issue, more data collection is required, but it is not the ideal solution. Consequently, dimensionality reduction is a key approach for resolving the problem.

In chapter 2, a spiking neural network (SNN) based reinforcement learning for collision avoidance is proposed in the mobile robot setup. The main idea of the approach is using a multiplicative weight update scheme in SNNs to more closely mimic the human brain. Also, more biological synaptic connections are implemented.

In the next chapter, a convolutional neural network (CNN) is utilized for sensor fusion. A gating scheme is proposed with fusion weight regularization and its target learning on the proposed architecture for resiliency and feature interpretability. Some sensory failure situations are covered with various experiments are made on three datasets.

In chapter 4, a reversible gating architecture for rare failure detection problem is covered with high-dimensional circuit data. Through the proposed reversible architecture, dimensional reduction is successfully made without the loss of information and the performance.

And we finish this dissertation in chapter 5 with concluding remarks and possible future works.

2. BIO-INSPIRED LEARNING ALGORITHM FOR AUTONOMOUS VEHICLE COLLISION AVOIDANCE *

In this chapter, collision avoidance application is covered for autonomous driving technology and a mobile robot is used as a test bed.

The collision avoidance is essential technology to autonomous systems such as mobile robots and self-driving cars. This collision avoidance system is deeply studied and developed in the auto industry [3]. In terms of a mobile robot, various sensors are used for an autonomous agent such as cameras, lasers, ultrasonic sensors. The goal of the collision avoidance is to make certain forms of learning capabilities to avoid collisions in a given environment. Towards this end, numerous types of reinforcement learning techniques such as adaptive dynamic programming and temporal difference are studied [4, 5]. Also, collision avoidance techniques which are based on these reinforcement learning approaches have been studied in the literature [6–11].

Among these algorithms, Q-learning [5], a popular implementation of reinforcement learning, has been adopted for collision avoidance where an optimal state-action policy is computed based on the Markov Decision Process (MDP). [7] employs Q-learning to train multiple robots to achieve a certain shaped formation and move in the formation while avoiding obstacles. [6] uses an artificial neural network to more efficiently represent the Q-values as part of Q-learning for robot collision avoidance without a target. The considered actions are discrete and correspond to moving and rotating in different directions. However, due to the discrete control actions, the movement of the robot does not look natural. Also, it is shown through simulation that the Q-learning algorithm requires longer training for for the convergence as around 500 epochs for environments with a few obstacles in [6]. In addition, a simplistic reward function which depends only on the actions taken and occurrences of collisions is used.

In recent years, spiking neural networks (SNNs), the third generation model of neural networks,

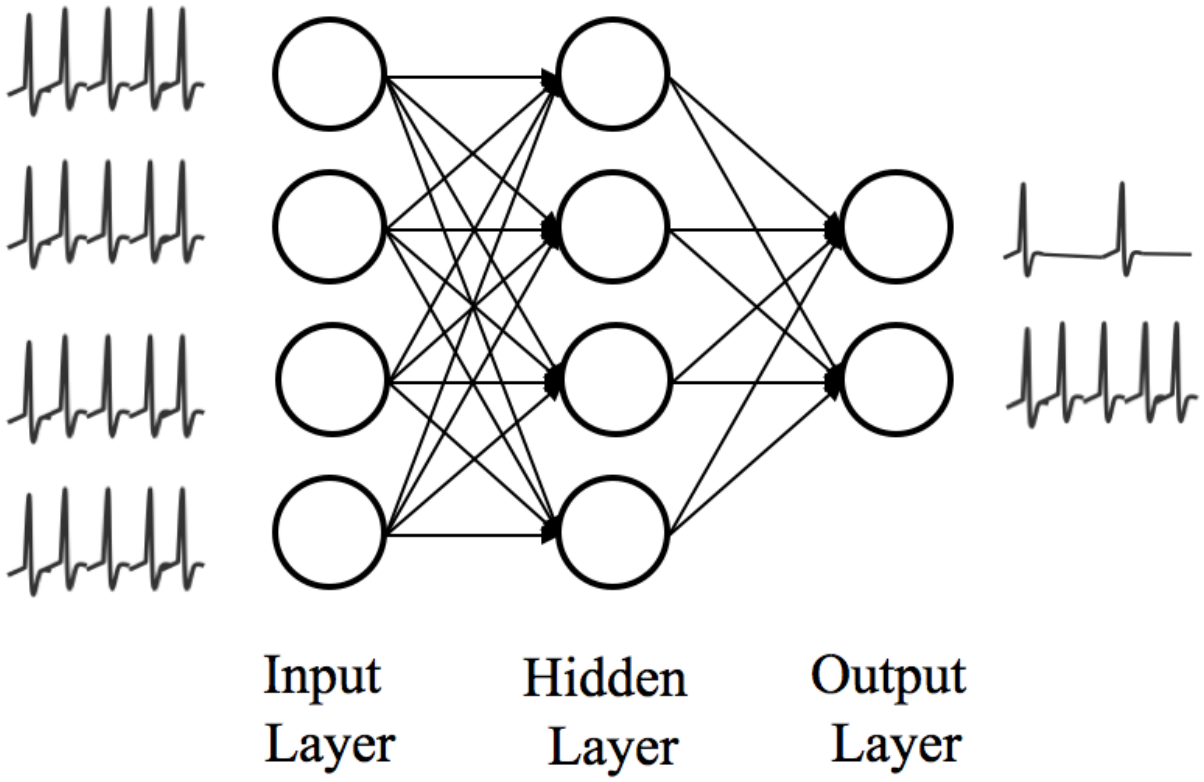

Figure 2.1: A feed-forward spiking neural network. © 2017 IEEE

have attracted increased attention due to their closer resemblance to neural circuits in biological brains. As an example, Fig. 2.1 shows a feed-forward spiking neural network.

Ideas on exploring reinforcement learning under the context of SNNs, particularly the concept of Reward-Modulated STDP (RM-STDP), have been suggested by the neuroscience community [12–14]. Up to now, an SNN may learn an optimal policy for decision making under a properly designed global reward signal with eligibility trace for delayed rewards. Note that RM-STDP is modulated by the reward signal and is different from the standard Spike-Timing-Dependent Plasticity (STDP), which is a form of unsupervised Hebbian learning in the SNN.

While SNN based reinforcement learning and similar techniques are still a relatively new territory of research, there have been some initial attempts towards applying such techniques for the robot collision avoidance. [15] presents a SNN-based target tracking robot controller, where a charge coupled device(CCD) camera is used to detect the target and 16 ultrasonic sensors are used to detect obstacles around the robot. The camera inputs are transformed into spike trains and drive directly the output neurons which control the motors. On the other hand, ultrasonic sensor inputs are projected to the hidden layer to learn collision avoidance, and the outputs of the hidden layers are fed to the output layer. The overall learning approach is not based on reinforcement learning where an unsupervised spike-based Hebbian learning mechanism is adopted to update synaptic weights. In [11], a relatively simple conditionally reward-modulated Hebbian plasticity mechanism is used for learning simple goal-directed behaviors in open environments without any obstacle. [10] employs short-term plasticity and long-term plastiticy realized using the temporal difference (TD) rule for wall following. [16] uses standard STDP for collision avoidance and target approaching in simple environments.

While the application of spiking neural networks has not been well studied for mobile robot navigation, the key objective of this chapter is to investigate the potential of biologically-motivated spiking neural networks for goal-directed collision avoidance in reasonably complex environments under the context of reinforcement learning. Our main new contributions are:

- In addition to the existing standard additive RM-STDP scheme (A-RM-STDP), for the first

time, we explore a multiplicative RM-STDP scheme (M-RM-STDP) under the context of reinforcement learning and demonstrate its potential.

- We explore a more biologically plausible feed-forward SNN architecture where in the hidden layer, both excitatory and inhibitory neurons are employed. We further develop a fine-grained reward scheme for the new more biological SNN architecture.

- Finally, we demonstrate that by combining the above two techniques leading to a further optimized solution with improved success rates and navigation trajectories.

To demonstrate the proposed ideas, we design several feed-forward SNNs with one input, one hidden and one output layer. Our proposed approaches completely outperform Q-learning in terms of both the success rate and the quality of trajectories to the target.

With respect to our first contribution, multiplicative reward-modulated STDP schemes have not been explored for reinforcement learning. In this chapter, for the first time, we present a multiplicative reward-modulated STDP scheme (M-RM-STDP). We demonstrate that for the application of mobile robot navigation, our M-RM-STDP outperforms the conventional additive manner of RM-STDP, i.e. A-RM-STDP, significantly, for example by increasing the success rate by 40% in the testing phase of robot navigation. In addition, M-RM-STDP reduces the number of steering movements taken for reaching the target by 10%, which shows the optimized path to a target location in simulation environments.

Our second contribution is motivated by the fact that excitatory and inhibitory mechanisms produce rich behaviors in biological brains. For this, we introduce both excitatory and inhibitory neurons not only in the input layer but also in the hidden layer to improve learning performance. Furthermore, we propose optimized reward functions which specify the amount of reward as a function of the distances to nearby obstacles, the distance to the target, and the angle between the moving direction of the robot and the vector that points to the target from the robot. Very importantly, instead of utilizing a single global reward for all synapses as typically done in the prior work, we take a finer-grained approach that is based on the fact that the inclusion of both

exictatory and inhibitory neurons in the first two layers produces feed-forward paths that have rather different effects on each output neuron. As a result, the net effects of two synapses on the same output neuron can be opposite of each other, with one being excitatory and the other inhibitory, depending the nature of the signal paths which these synapses are on. Recognizing this disparity, we conduct additional sign adjustment of the reward for different synapses in the network, which significantly improves performance.

Based on the first two contributions, our combined approach, i.e. the third contribution, produces the best results among studied approaches. It significantly outperforms the standard RM-STDP scheme, for example by increasing the success rate in the testing phase by up to 45% while reducing the number of steering movements taken to get to the target.

2.1 Background

We briefly discuss the leaky integrate-and-fire (LIF) spiking neuron model, STDP, and additive reward-modulated STDP.

2.1.1 The Leaky Integrate-and-Fire Neuron Model

A number of spiking neural models such as the (leaky) integrate-and-fire, Hodgkin-Huxley [17], and Izhikevich [18] models have been adopted for modeling SNNs. In this work, the leaky integrate-and-fire(LIF) model is utilized due to its simplicity:

$$\tau_m \frac{dv(t)}{dt} = -\left(v(t) - v_{rest}\right) + RI(t), \tag{2.1}$$

where $v(t)$ is the membrane potential, τ_m the membrane time constant, v_{rest} the resting potential, R the membrane resistance, and $I(t)$ the synaptic input current. Under some injected $I(t)$, the membrane potential may go beyond the threshold voltage starting from which an action potential, or a spike, would be generated [19].

2.1.2 Spike-Timing-Dependent Plasticity

STDP is an unsupervised Hebbian learning mechanism, which adjusts a synaptic weight based upon the spike timing difference between the corresponding pre-synaptic and post-synaptic spikes [20]. The weight w_{ij} is strengthened if the pre-synaptic neuron fires before the post-synaptic neuron, otherwise it is weakened. The temporal difference between the firing times of each pair of the pre-synaptic and post-synaptic spikes $\Delta t = t_{post} - t_{pre}$ determines the amount of weight change:

$$\begin{aligned} \Delta w_+ &= A_+ \cdot e^{-\frac{\Delta t}{\tau_+}} \quad if \quad \Delta t > 0 \\ \Delta w_- &= A_- \cdot e^{-\frac{\Delta t}{\tau_-}} \quad if \quad \Delta t < 0, \end{aligned} \tag{2.2}$$

where Δw_+ and Δw_- are the weight modifications induced by long-term potentiation (LTP)and long-term depression (LTD), respectively, A_+ and A_- are some positive and negative constant parameter and determine the strength of LTP and LTD, respectively, and τ_+ and τ_- set the temporal windows over which STDP is active. A typical STDP curve is shown in Fig. 2.2.

2.1.3 Additive Reward-Modulated STDP

Additive reward-modulated STDP (A-RM-STDP) is an implementation of reinforcement learning mechanism, which updates the synaptic efficacy in an additive manner. While STDP operates based upon the correlation between the spike timings of the pre- and postsynaptic neurons, a reward is introduced to modulate STDP in A-RM-STDP. If the reward is positive, the corresponding synapse is potentiated. Otherwise, it is depressed.

For a synapse projecting from the j-th neuron to the i-th neuron, A-RM-STDP may be realized according to [12]:

$$\frac{dw_{ij}(t)}{dt} = \gamma r(t) z_{ij}(t), \tag{2.3}$$

where γ is the learning rate, $r(t)$ the reward signal, and $z_{ij}(t)$ the eligibility trace which decays the

amount of weight update over time. The eligibility trace can be defined as:

$$\tau_z \frac{dz_{ij}(t)}{dt} = -z_{ij}(t) + \xi_{ij}(t), \tag{2.4}$$

where additional dynamic variables ${P_{ij}}^+$, ${P_{ij}}^-$ and ξ_{ij} are utilized to track the effect of pre-synaptic and post-synaptic spikes [12]:

$$\xi_{ij}(t) = {P_{ij}}^+ \Phi_i(t) + {P_{ij}}^- \Phi_j(t), \tag{2.5}$$

$$\frac{d{P_{ij}}^+(t)}{dt} = -\frac{{P_{ij}}^+(t)}{\tau_+} + A_+ \Phi_j(t), \tag{2.6}$$

$$\frac{d{P_{ij}}^-(t)}{dt} = -\frac{{P_{ij}}^-(t)}{\tau_-} + A_- \Phi_i(t), \tag{2.7}$$

where Φ_i is the Dirac delta function which is non-zero only at times t when the neuron i fires, and A_+ and A_- are certain positive and negative constant, respectively. A typical RM-STDP characteristics is shown in Fig. 2.3.

2.2 Reinforcement Learning with the Proposed Multiplicative RM-STDP

While multiplicative reward-modulated STDP has not been studied for reinforcement learning, for the first time we present a multiplicative reward-modulated STDP scheme (M-RM-STDP) which shows good performance for mobile robot navigation. We then present an optimized reward function which specifies the amount of reward as a function of the distance to obstacles, the distance to the target, and the angle between the moving direction of the robot and the vector that points to the target from the robot.

2.2.1 Multiplicative Reward-Modulated STDP

In the past, only additive reward-modulated STDP has been considered for reinforcement learning under the context of spiking neural networks. In this chapter, in contrast to (2.3), we explore a new multiplicative scheme as follows:

$$\frac{dw_{ij}(t)}{dt} = \gamma w_{ij}(t) r(t) z_{ij}(t). \tag{2.8}$$

As can been seen, in this new M-RM-STDP scheme, the synaptic weight is updated based on the product of four components: the current weight, learning rate, reward, and eligibility trace. As a result, the instantaneous rate of weight change is proportional to the current weight value $w_{ij}(t)$.

Under different contexts, earlier study on multiplicative STDP demonstrates that it may behave differently from additive STDP, for example, by producing stable unimodal distributions of synaptic weights, which makes synapses less sensitive to input perturbations [21]. Under the context of SNN-based reinforcement learning for mobile robot navigation, we have experimentally observed that in addition to possible performance boost, the maximum value of weight change resulted from the proposed M-RM-STDP increases rather noticeably, for example, by five times in some cases over A-RM-STDP. As a result, it is also observed that the multiplicative nature of the proposed RM-STDP scheme leads to faster learning.

2.2.2 Proposed Reward Functions

In this work, we employ feed-forward spiking neural networks with an input layer, hidden layer, and output layer. The output layer consists of two output neurons controlling the left and right motors of the robot, i.e. the output neuron that fires with a higher frequency makes the robot turn based on the corresponding motor. More details about the network setting are provided in Section 2.4.

Reward functions play an important role for reinforcement learning as they provide critical feedback from the environment to the agent. In conjunction with the existing A-RM-STDP and proposed M-RM-STDP schemes, we make use of reward functions optimized for the targeted application. Unlike the simple reward function used in [6], which only depends on actions and occurrences of collision, the proposed reward functions takes the distance to obstacles, the distance to the target, and the angle between the moving direction of the robot and the vector that points to the target from the robot into consideration. In this section, we describe how rewards are computed.

The specific ways in which the rewards are applied to the network will be discussed in the next section.

2.2.2.1 Rewards for Collisions and Arrival

During each training trial, when the robot collides with an obstacle, the entire network gets a negative reward of R_{col}. Otherwise, if the robot arrives at the target, the network gets a positive reward of R_{arr}. In our experiments, we set: $R_{col} = -2.0$ and $R_{arr} = 2.0$. In both cases, the training trial ends after the application of the reward.

2.2.2.2 Rewards for Collision Avoidance

In the absence of collisions and arrival at the target, we compute the following two rewards to promote collision avoidance when the robot gets close to obstacles, which may be detected, for example, by ultrasonic sensors:

$$\begin{aligned} r_1(d) &= +\frac{d_{safe} - d}{d_{safe}} + k, \quad for \quad d < d_{safe} \\ r_2(d) &= -\frac{d_{safe} - d}{d_{safe}} - k, \quad for \quad d < d_{safe} \end{aligned} \tag{2.9}$$

where d is the distance to the nearest obstacle. In our simulation environment, we set d_{safe} to 100 pixels: $d_{safe} = 100$, and constant k to 0.3 for A-RM-STDP and 1.3 for M-RM-STDP, respectively. The reward values are earned experimentally for avoiding deadlock situations.

More specifically, if the nearest obstacle is within d_{safe} from the robot on the right, the negative reward r_2 is applied to the subset of synapses that influence the firing of the left motor (output) neuron while the positive reward of r_1 is applied to the remaining synapses of the network, which influence the right motor (output) neuron. In this case, there is a tendency for the right motor to rotate faster than the left motor, steering the robot to the left. We swap the roles of r_1 and r_2 if the nearest obstacle is within d_{safe} from the robot on the left.

2.2.2.3 Reward in the Vicinity of the Target

After conditionally applying the rewards specified in (2.9), we further check if the robot is in the vicinity of the target, i.e. if the distance d between the target and robot is less than a specified threshold d_{tar}. If so, the following additional award is computed:

$$r_3(d) = \frac{1}{d_{tar}}(d_{tar} - d) + 1.0, \quad for \quad d < d_{tar}, \tag{2.10}$$

where d_{tar} is set to 200 pixels in our experiments.

We further consider the angle between the moving direction of the robot and the vector that points to the target from the robot. As shown in Fig. 2.4, we split the environment into four regions around the robot. If the target falls into the regions "2" and "4", the reward r_3 is then applied to the synapses influencing the left and right motor neuron, respectively. No reward is applied when the target falls in the regions "1" and "3".

2.3 Proposed Feed-forward SNNs and Fine-grained Rewards

2.3.1 Feed-forward SNNs

Cortical circuits in biological brains operate based upon both excitatory and inhibitory mechanisms. A proper balancing between excitation and inhibition in feed-forward networks has been shown to be beneficial [22]. While earlier works have employed both inhibitory and excitatory neurons only in the input layer [12, 23], we extend by doing the same for the hidden layer as well as shown in Fig. 2.5. Here, the synapses from each excitatory (inhibitory) neuron are considered to be excitatory (inhibitory) in nature.

While the above approach improves learning performance as demonstrated by our experimental results, such improvements can only be achieved if and only if each reward signal is properly applied to the network. In particular, the inclusion of both excitatory and inhibitory neurons in the network nevertheless introduces certain complications in the application of rewards. Our experiments have shown that treating each reward as a "global" signal and applying it uni-formally to all

synapses in the network, as what is typically done in the literature, can lead to very poor learning performance. To address this problem, we propose a fine-grained approach for applying a reward to the network as discussed next.

2.3.2 Fine-Grained Rewards

We first recognize that there exist four types of feed-forward paths from the input layer, to the hidden layer, and finally to the output layer in the network of Fig. 2.5: Inhibitory-Inhibitory (I-I), Inhibitory-Excitatory (I-E), Excitatory-Inhibitory (E-I), and Excitatory-Excitatory (E-E), where the first designation specifies the synapse type from the input to the hidden layer and the second the synapse type from the hidden to the output layer on the path.

Recall that Section 2.2.2 describes several different types of reward. Each reward may be applied to all synapses or just a subset of them in the network. Once obtaining the value of a reward, we do not immediately apply the reward to the targeted synapses. Instead, additional sign adjustment may be performed to properly deal with each of the four types of feed-forward signal paths.

To explain our idea, let us consider the following illustrative example for which a reward value of r is computed based on one of the reward functions described in Section 2.2.2. Let us further assume that this reward is intended for the synapses influencing the right motor (output) neuron to incentivize the right motor to rotate faster than the left motor to make the robot turn left. In this case, we consider all four different type paths ending at the right motor neuron.

To achieve our goal, we potentiate or depress each synapse on a given signal path as follows (Fig. 2.5):

- I-I: potentiate the first with a reward of r; depress the second with a reward of $-r$;
- I-E: depress the first with a reward of $-r$; potentiate the second with a reward of r;
- E-I: depress the first with a reward of $-r$; depress the second with a reward of $-r$;
- E-E: potentiate the first with a reward of r; potentiate the second with a reward of r;

Note that here depressing an inhibitory synapse with a negative reward means reducing the absolute value of the synaptic weight. The basic idea behind the above fine-grained reward approach is to consider the excitatory or inhibitory nature of each synapse with respect to the targeted output neuron. For example, for the E-I type signal paths, while the first synapse is excitatory, its effect on the right motor neuron is in fact *inhibitory*. As a result, we depress this synapse with a negative reward of $-r$.

2.4 Experimental Settings

To demonstrate the proposed reinforcement learning approach, we consider the problem of autonomous mobile robot navigation towards a fixed target in environments with stationary obstacles. As shown in Fig. 2.6, we assume that the targeted robot has five ultrasonic sensors to measure distances to nearby obstacles. In addition, we also assume that the distance from the current location to the target and the angle between the moving direction of the robot and the vector that points to the target from the robot are also available to the robot through a "distance" and "angle" sensor, respectively. Note that the robot is modeled simplistically without internal delay and motor control dynamics. The adopted simulation environment is based on Pygame 1.9.2, a game library in Python, and Brian 1.41 [24], an SNN simulator in Python.

2.4.1 Spiking Neural Networks

We employ feed-forward spiking neural networks of three layers: an input, hidden, and output layer. The input layer is composed of seven groups of Poisson neurons for generating spike trains encoding the inputs from the five ultrasonic sensors, distance sensor and angle sensor. Each group has 15 excitatory and 15 inhibitory neurons.

To demonstrate the SNN architecture proposed in Section 2.3.1, we consider two compositions for the hidden layer: 210 excitatory neurons (Fig. 2.7) vs. 104 excitatory and 104 inhibitory neurons (Fig. 2.8), with the former being a reference and the latter representing the proposed architecture. In both networks, the input layer is fully connected to the hidden layer. The hidden layer is split into two equal halves, with each projecting to one of the two output neurons with fully

connectivity. The two output neurons control the left and right motors of the robot and can steer the robot to right and left, respectively.

2.4.2 State Space and Discretization

The sensory inputs from the five ultrasonic (measuring the distances to nearby obstacles), one distance (measuring the distance to the target) and one angle sensor form the seven-dimensional state of the robot. Each of the five ultrasonic sensors can measure up to 300 pixels. For the reference SNN, every ultrasonic sensor input is discretized into three intervals with each encoded using a different firing frequency: 0 to 99 pixels (0 Hz), 100 to 199 pixels (20 Hz), and 200 to 300 pixels (40 Hz). The encoding of the ultrasonic sensor inputs is done somewhat differently for the proposed SNN: 0 to 99 pixels (0 Hz), 100 to 199 pixels (50 Hz), and 200 to 300 pixels (100 Hz). Each frequency value is used to set the firing frequency of the Poisson neurons in the corresponding input neuron group.

The distance sensor can measure up to 800 pixels for the distance to the target. Its input range (0 to 800 pixels) is divided evenly into four intervals. For the reference SNN, the encoded Poisson firing frequency is 10 Hz, 20 Hz, 30 Hz and 40 Hz, respectively for these intervals. For the proposed SNN, the encoded frequencies are 25 Hz, 50 Hz, 75 Hz, and 100 Hz.

Finally, the input from the angle sensor is discretized into four intervals: north, south, west and east direction, as shown in Fig. 2.4. The frequency encoding schemes used for the distance sensor are adopted for the angle sensor.

2.4.3 Motor Control

The speed of the robot is fixed as 15 pixels/sec such that the robot always moves forward. At each simulation time step, the instantaneous firing rates of the two output (motor) neurons are used to determine the steering angle, that is, the steering angle θ is set to $\theta = freq_l - freq_r$, where $freq_l$ and $freq_r$ are the firing frequencies of the left and right motor neuron, respectively. As such, θ is continuous-valued and the robot is steered to the left if θ is negative. Note that, in Q-learning, ϵ-greedy with ϵ set to 0.3 is used for selecting from three actions: move forward, turn left by 30°

and turn right by 30°.

2.4.4 STDP parameter settings

The parameter settings of the A-RM-STDP and M-RM-STDP schemes are given in Table 2.1. The initial weights of excitatory synapses are set randomly between 0 - W_{max} mV while those of inhibitory synapses are set randomly between W_{min} - 0 mV. For the proposed network architecture in Fig. 2.8, the maximum synaptic weight is set to 5mV for the half of the synapses between input and hidden layer, and 2.5mV for the remaining half.

Table 2.1: Parameter settings of A-RM-STDP and M-RM-STDP. © 2017 IEEE

Parameter	Value
A_+	1mV
A_-	-1mV
τ_+	20ms
τ_-	20ms
W_{max}	5mV
W_{min}	-5mV

2.5 Experimental Results

By using the setups described previously, we compare four spiking neural networks and Q-learning as summarized in Table 2.2. Two spiking neural networks have both excitatory and inhibitory neurons only in the input layer while the other two have excitatory and inhibitory neurons in both the input and hidden layers. All SNNs have the same numbers of neurons in three layers.

We examine these five approaches by the *success rate*, SR, defined as as the number of successful trials, i.e. ones in which the robot gets to the target without any collision, divided by the total number of trials. We evaluate the success rate for both the learning and testing phase. The average number of steering movements per trial, N_{mv} (the smaller the better), is another indicator of the quality of learning. We consider N_{mv} in the testing phase for the four SNNs, and N_{mv} in the learning phase for Q-learning as it does not perform well in testing.

Table 2.2: Simulation results on validation set. © 2017 IEEE

Network	Algorithm	Network Structure
Q-learning	Q-learning	Look-up table
S1	A-RM-STDP	E & I in Input layer; E in Hidden layer
S2	M-RM-STDP	E & I in Input layer; E in Hidden layer
S3	A-RM-STDP	E & I in Input and Hidden layers
S4	M-RM-STDP	E & I in Input and Hidden layers

2.5.1 Scenario 1

As shown in Fig. 2.9 and Fig. 2.10, the target (red box) is located in the center of a closed environment. The robot navigates from a randomly chosen point within one of the blue circles on the left and right sides. The starting blue circle is also chosen at random. This setup forces the robot to explore the unknown environment widely. 2,000 training and 100 testing trials are used in the learning and testing phase, respectively, for Q-learning. The learning with the SNNs converges much faster so we use only 20 training trials and 30 testing trials. The results are shown in Table 2.3.

Table 2.3: Result of Scenario 1 © 2017 IEEE

Network	SR-Learning	SR-Testing	N_{mv}
Q-learning	4.05%	10%	62
S1:A-RM-STDP	45%	70%	50
S2:M-RM-STDP	65%	95%	49
S3:A-RM-STDP (E+I in hidden)	55%	90%	45
S4:M-RM-STDP (E+I in hidden)	70%	100%	46

Despite of the large number of training trials used, Q-learning performs poorly with an extremely low success rate (SR) in both the learning and testing phase. In addition, Q-learning also leads to a large number of steering movements in the testing phase, indicating a poor qual-

ity of trajectories learnt. Note again that in this chapter, for Q-learning, N_{mv} is reported based on the training data. The four SNNs demonstrate a much better performance than Q-learning. Among them, the proposed multiplicative RM-STDP scheme (M-RM-STDP) outperforms its additive counterpart (A-RM-STDP) rather noticeably in terms of success rate for both learning and testing.

The proposed neural network architecture of Fig. 2.8, where the hidden layer is composed of an equal number of excitatory and inhibitory neurons, performs fairly well. In particular, the network S4, which combines the proposed M-RM-STDP and network architecture has the highest success rate for both learning and testing, and dramatically outperforms S1, which is the baseline SNN. S4 also produces almost the lowest number of steering movements in testing. Fig. 2.9 and Fig. 2.10 show two representative trajectories produced by S4 in the learning and testing phase.

It is interesting to note that S2 somewhat outperforms S3 in terms of success rate while S3 produces a smaller N_{mv}. S2 employs only the proposed M-RM-STDP while S3 only employs the proposed neural network architecture.

2.5.2 Scenario 2

The second scenario has obstacles of varying shapes (Fig. 2.11 and Fig. 2.12). Again, the robot starts to navigate randomly from two blue circles. The same numbers of trials are used for Q-learning: 2,000 in the learning phase and 100 in the testing phase. For the SNNs, 20 training trials are used, 10 of which start from the left circle and the remaining 10 start from the right circle. 20 trials are used for testing. Table 2.4 summarizes the performances of the five approaches.

For this more challenging test case, Q-learning performs even worse, and completely fails the testing. Among the four SNNs, adopting the proposed M-RM-STDP leads to a success rate lower than A-RM-STDP in the learning phase, but a significantly boosted success rate in the testing phase. For example, the testing SR is 55% for S3, which is boosted to 90% by S4. The proposed neural network architecture appears to have a negative impact on the SR in the learning phase, as observed by comparing S3 to S1 and S4 to S2. It, however, noticeably improves the SR during testing. S4, which combines the two proposed techniques, has the highest success rate (90%) and

Table 2.4: Result of Scenario 2 © 2017 IEEE

Network	SR-Learning	SR-Testing	N_{mv}
Q-learning	4.05%	0%	75
S1: A-RM-STDP	75%	45%	67
S2: M-RM-STDP	50%	85%	65
S3: A-RM-STDP (E + I in hidden)	55%	55%	64
S4: M-RM-STDP (E + I in hidden)	40%	90%	62

the lowest N_{mv} (62) for testing, showing the best overall performance. We show two simulated trials of S4 in Fig. 2.11 and Fig. 2.12.

2.5.3 Scenario 3

In this last test case, the robot navigates from randomly sampled points in the blue circle at the bottom-right corner of the enclosed environment with more obstacles (Fig. 2.13 and Fig. 2.14). 2,000 trials are used for training the Q-learning based robot and 100 trials for testing. The detailed simulation results are described in Table 2.5. Q-learning again performs poorly and is not able to successfully reach the target at all during testing phase. In comparison, the SNNs deliver a much better performance with the network S4 producing the highest success rate (90%) and the lowest N_{mv} (54) during testing. Two simulated trajectories of S4 are shown in Fig. 2.13 and Fig. 2.14.

Table 2.5: Results of Scenario 3 © 2017 IEEE

Network	SR-Learning	SR-Testing	N_{mv}
Q-learning	2.05%	0%	72
S1: A-RM-STDP	20%	75%	61
S2: M-RM-STDP	40%	90%	57
S3: A-RM-STDP (E + I in hidden)	60%	65%	56
S4: M-RM-STDP (E + I in hidden)	45%	90%	54

2.6 Summary and Discussions

This chapter demonstrates two techniques for spiking neural network based reinforcement learning for mobile robot navigation: a new multiplicative RM-STDP (M-RM-STDP) scheme and feed-forward spiking neural network architecture with fine-grained rewards. It has been shown that the proposed techniques significantly outperform Q-learning and a baseline SNN approach. Especially, combining the two proposed techniques leads to a fairly robust solution with significantly improved success rates and quality of navigation trajectories measured by the number of steering movements. In our future work, the two proposed techniques will be demonstrated on a real robot.

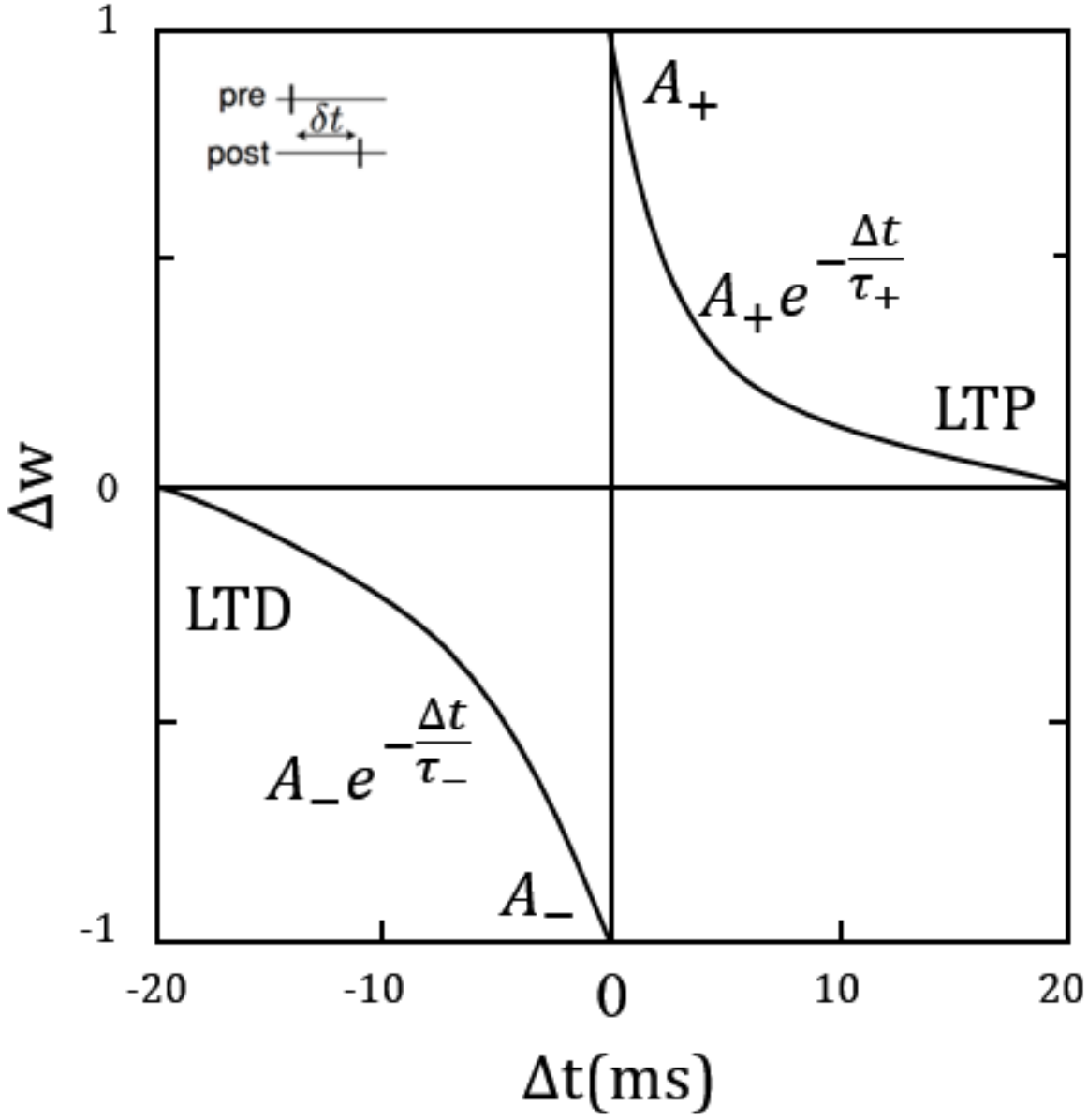

Figure 2.2: A typical STDP curve. © 2017 IEEE

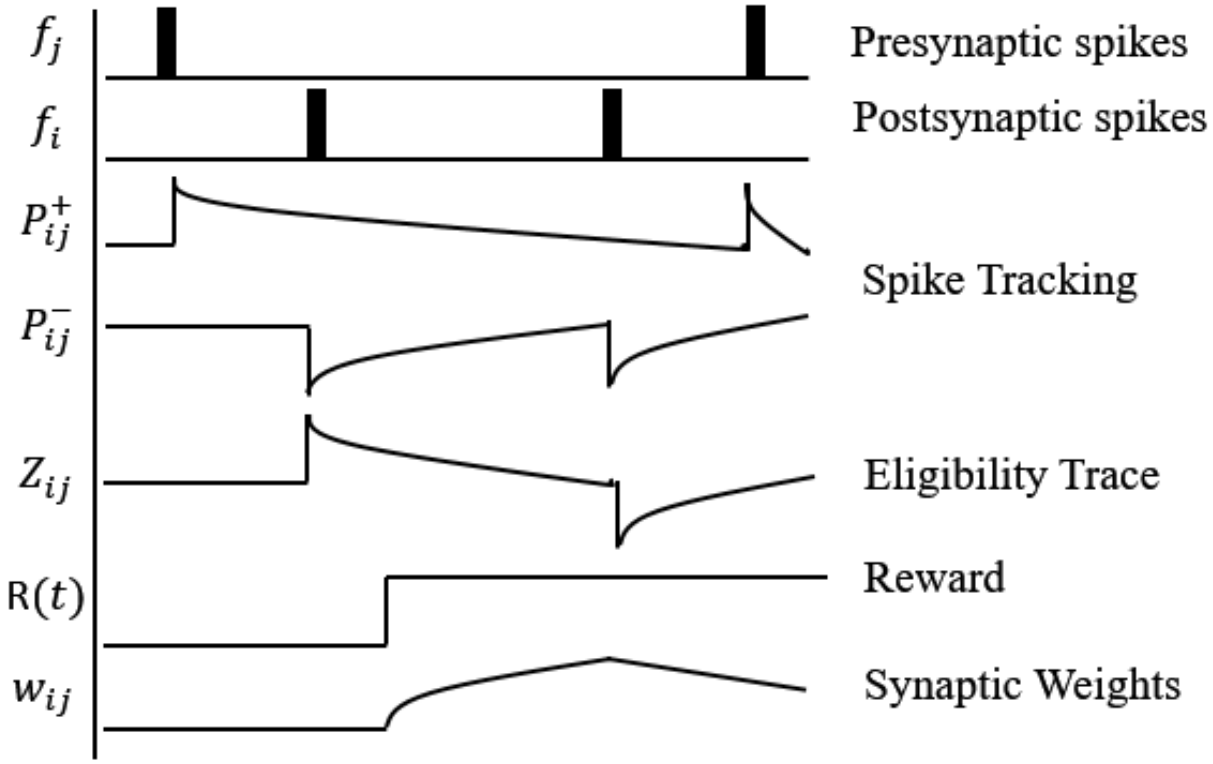

Figure 2.3: A typical additive RM-STDP characteristics. © 2017 IEEE

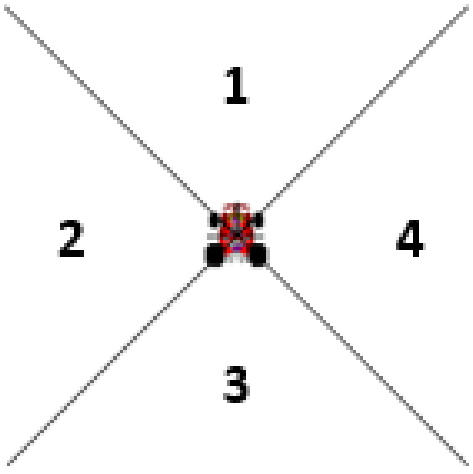

Figure 2.4: Four intervals for the angle between the moving direction of the robot and the vector that points to the target from the robot. © 2017 IEEE

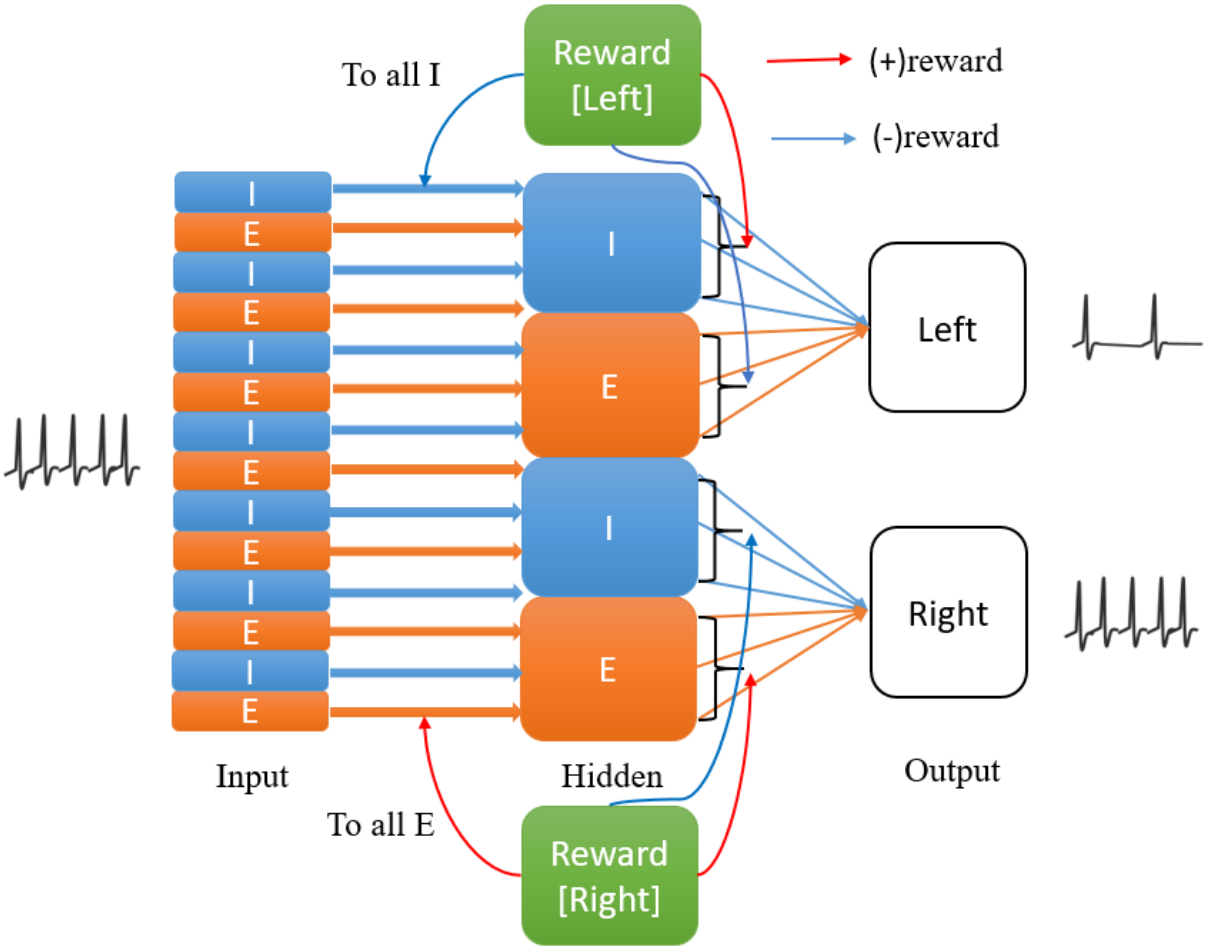

Figure 2.5: The proposed spiking neural network with fine-grained rewards. © 2017 IEEE

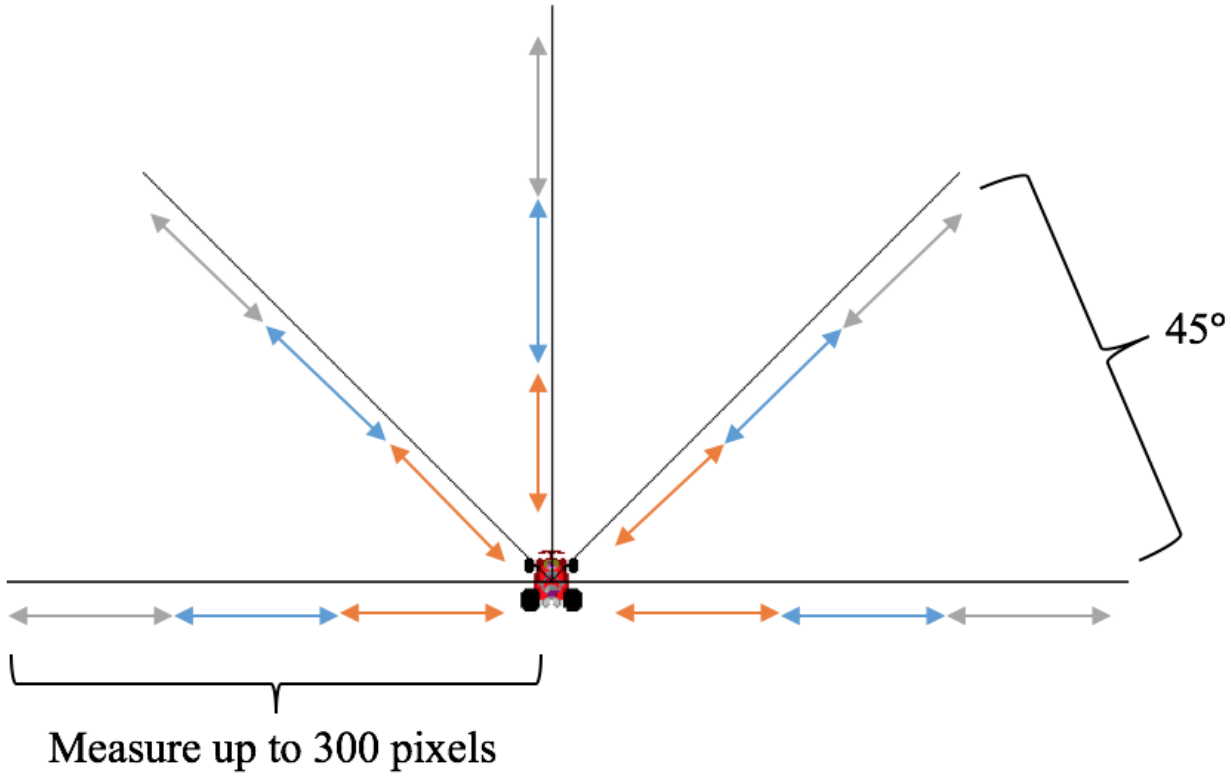

Figure 2.6: The adopted car model. © 2017 IEEE

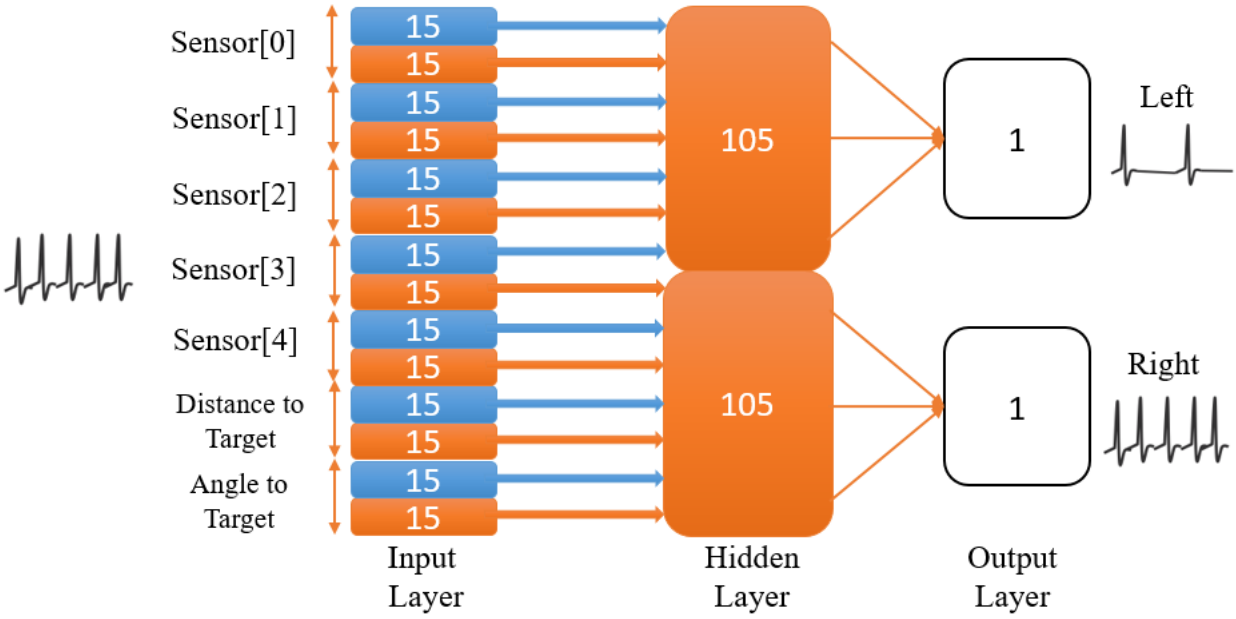

Figure 2.7: The reference SNN with only excitatory neurons in the hidden layer. © 2017 IEEE

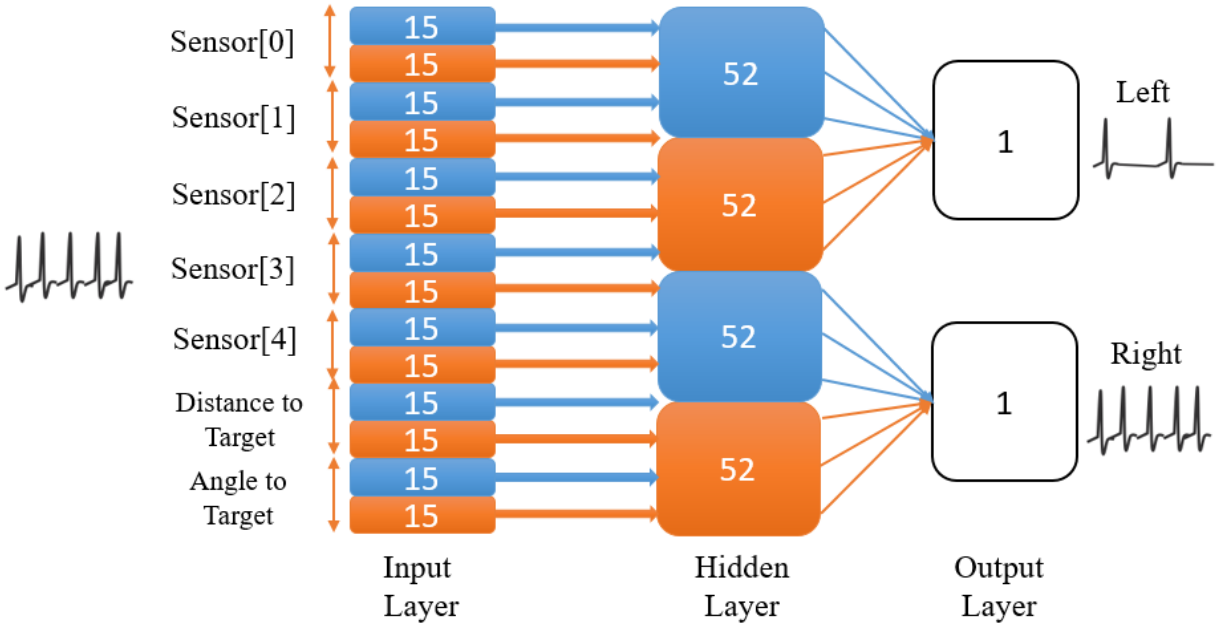

Figure 2.8: The proposed SNN with both excitatory and inhibitory-neurons in the hidden layer. © 2017 IEEE

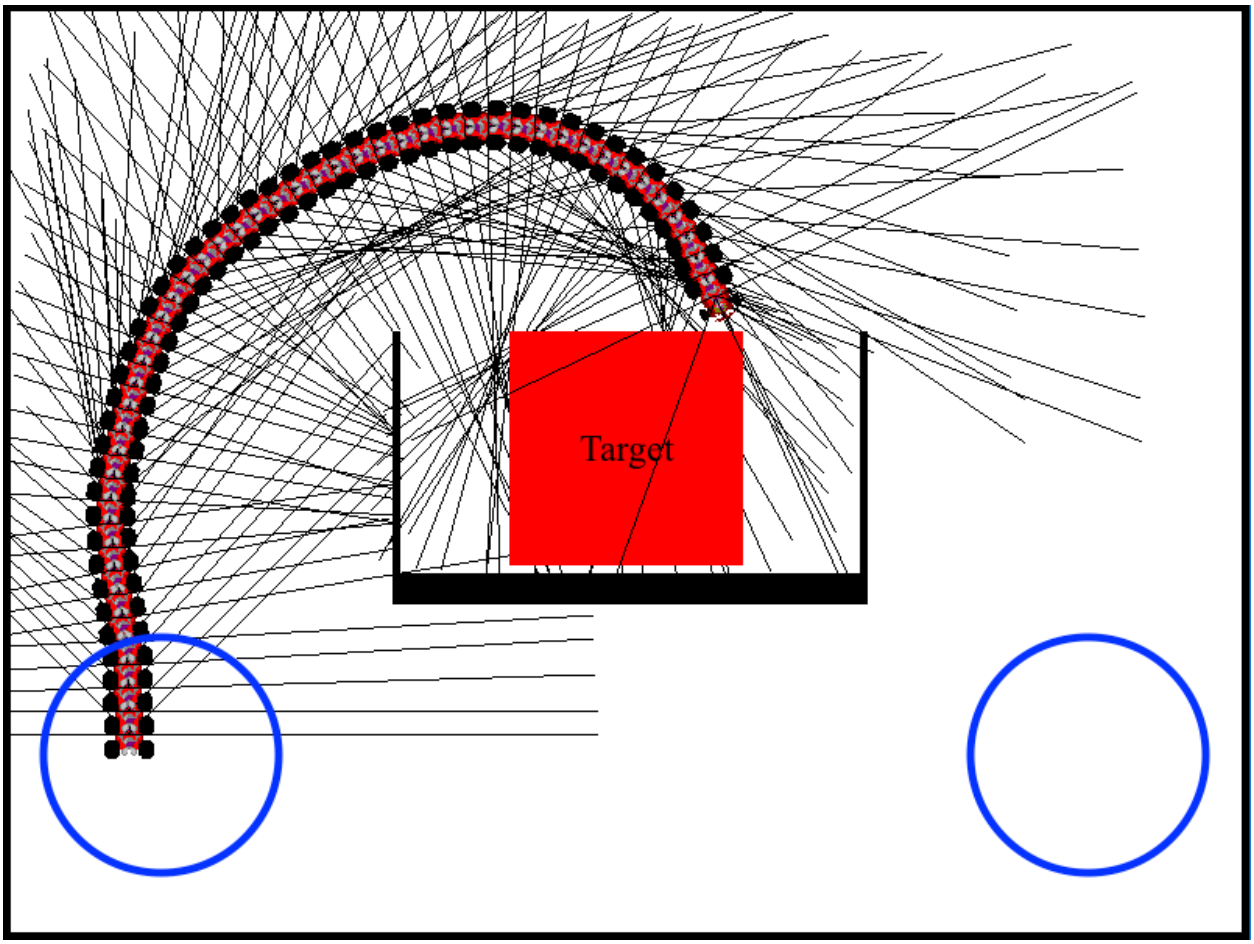

Figure 2.9: A simulated trial in the learning phase of S4 for Scenario 1. © 2017 IEEE

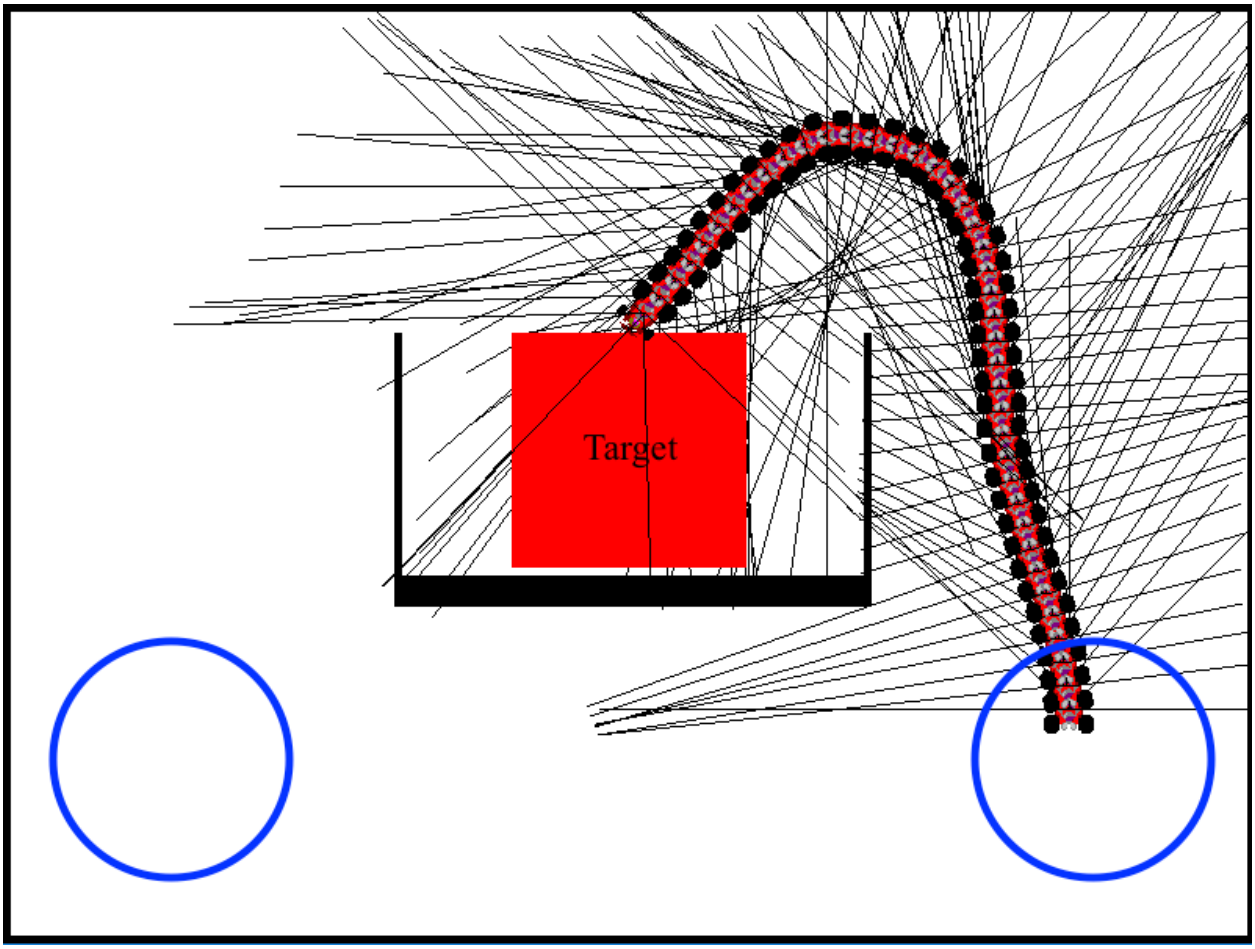

Figure 2.10: A simulated trial in the testing phase of S4 for Scenario 1. © 2017 IEEE

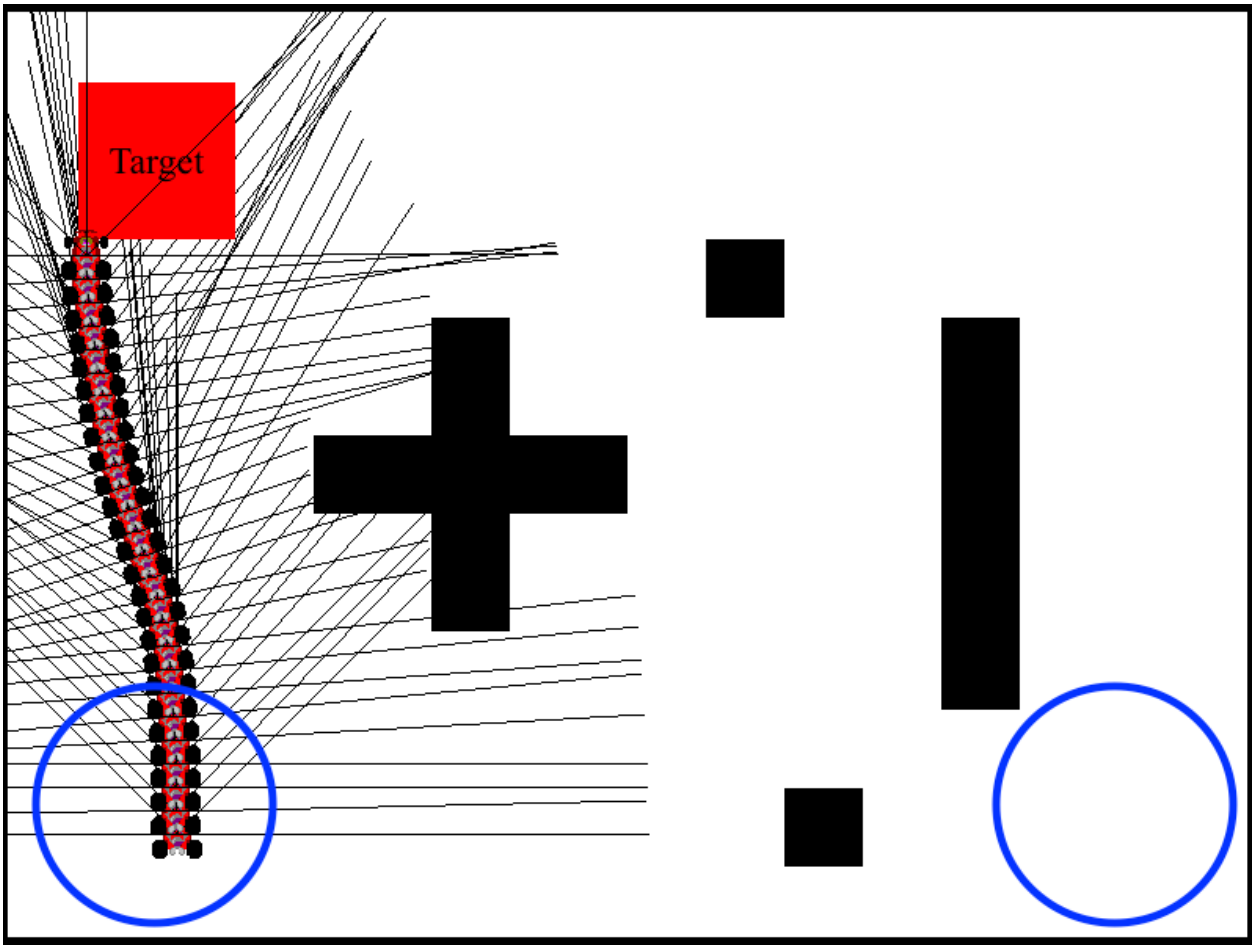

Figure 2.11: A simulated trial in the learning phase of S4 for Scenario 2. © 2017 IEEE

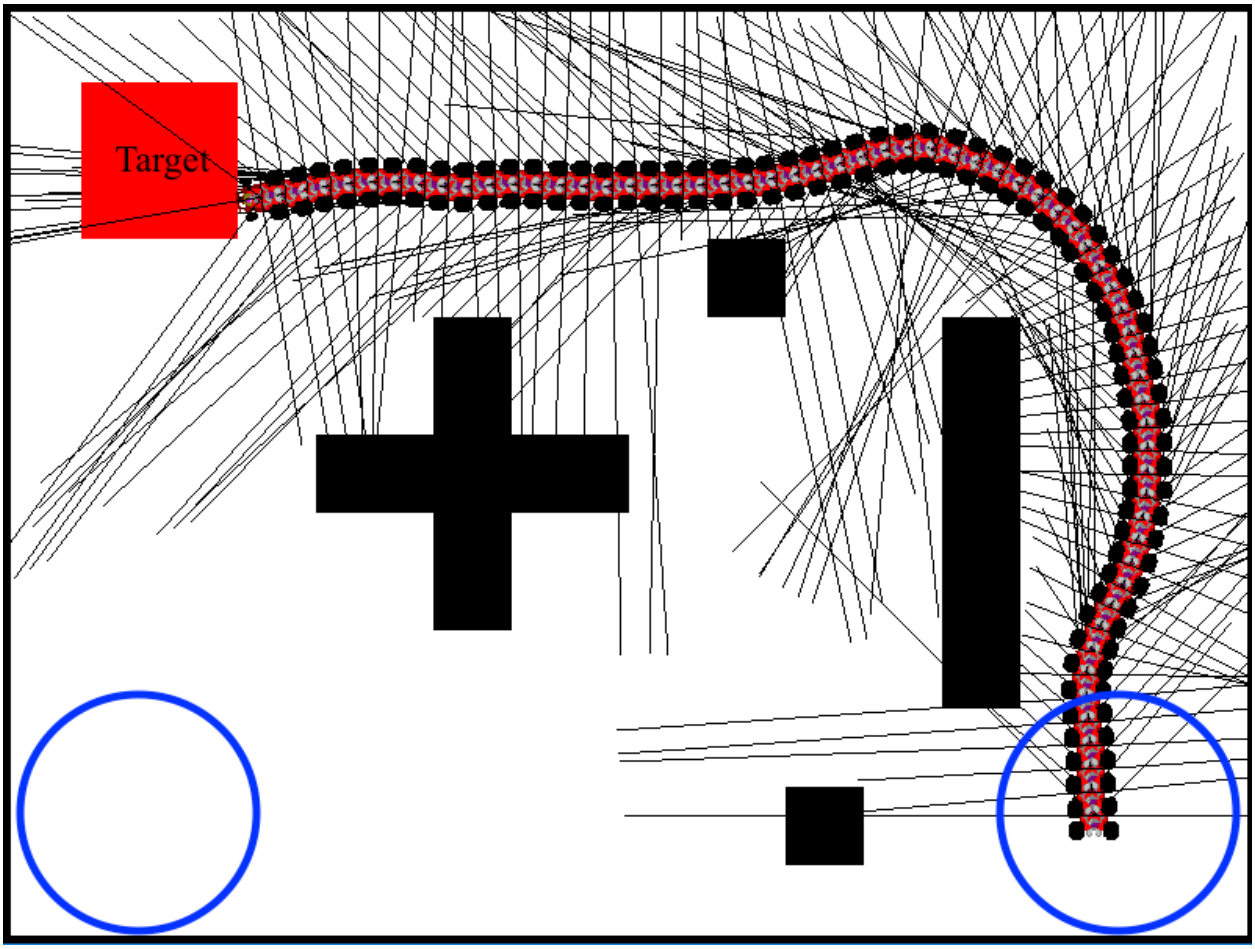

Figure 2.12: A simulated trial in the testing phase of S4 for Scenario 2. © 2017 IEEE

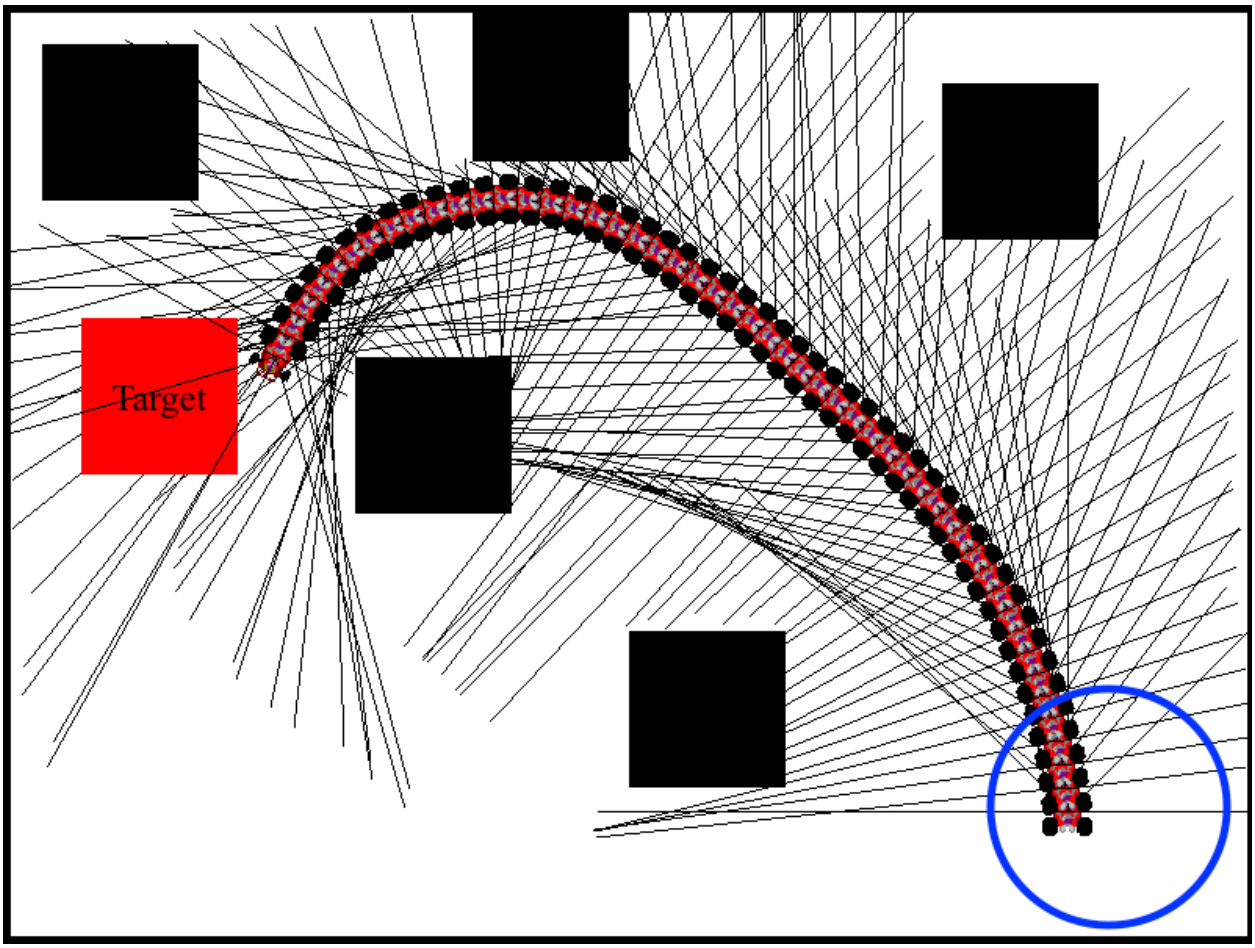

Figure 2.13: A simulated trial in the learning phase of S4 for Scenario 3. © 2017 IEEE

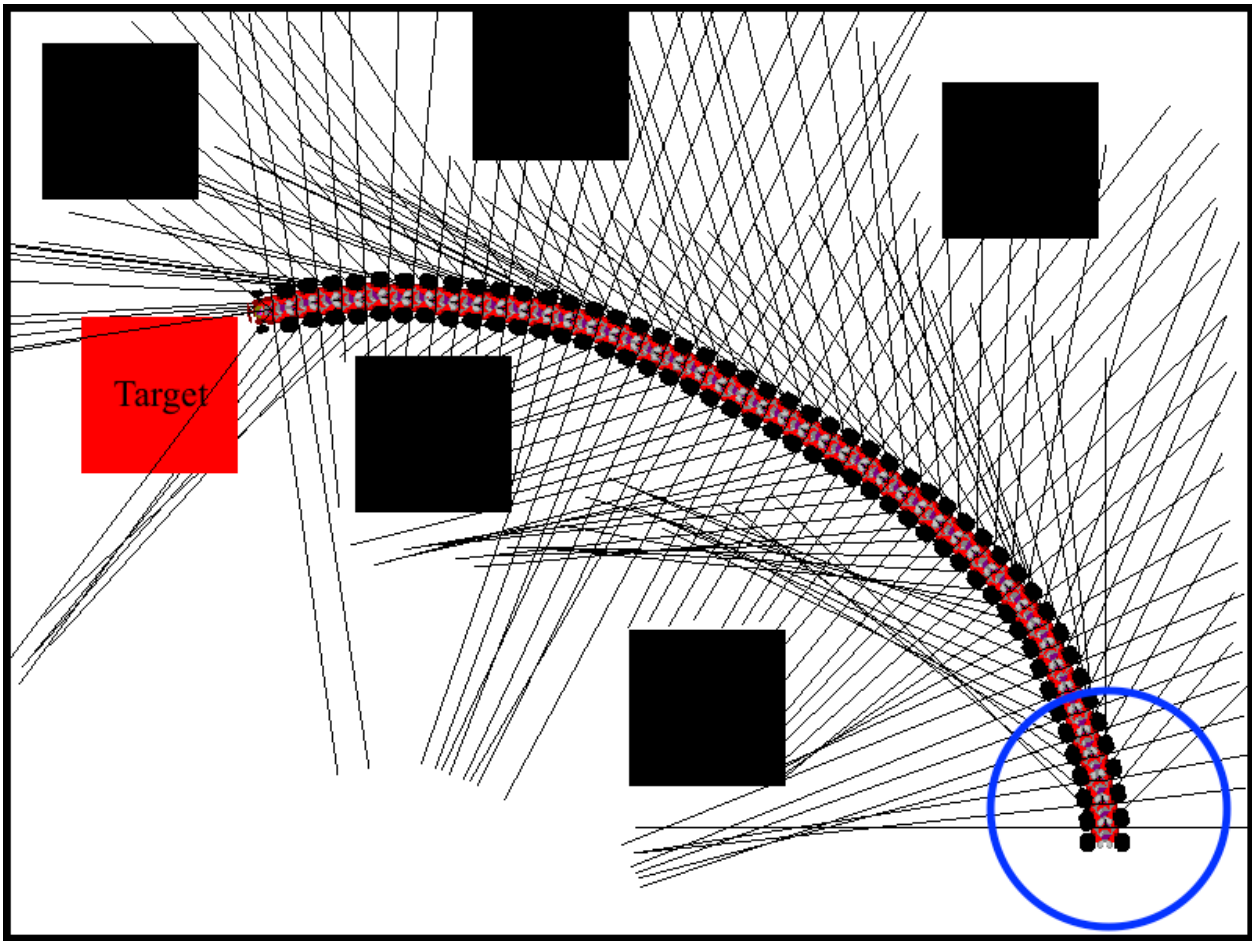

Figure 2.14: A simulated trial in the testing phase of S4 for Scenario 3. © 2017 IEEE

3. ROBUST FUSION ARCHITECTURE FOR AUTOMOBILE

In this chapter, a sensor fusion architecture with a fusion weight regularization and a target learning scheme is proposed for resiliency and feature interpretability.

Sensor fusion is a key technique for autonomous driving system with multiple sensors such as camera, lidar, radar, GPS, IMU, OBD2, etc [25–30]. Its objective is to make an accurate ground truth with a combination of multiple sensory inputs and create an environment where the fusion is greater than simple sum of the sensory inputs.

[27] presents a convolutional neural network (CNN) approach to sensor fusion in autonomous driving and compares three fusion schemes: early, late, and deep fusion. However, this work does not cover sensor failures and provides no deep insight on computational principles of sensor fusion, rather fuse the two sensory inputs with simple element wise mean operation. The typical late fusion DNN architecture is shown in Fig. 3.1 (a) where each modality is first processed, e.g. by a number of convolutional/pooling/activation layers. And then all extracted feature maps are fused together, e.g. by element-wise mean at the layer marked by "+", and processed further to make the final decision.

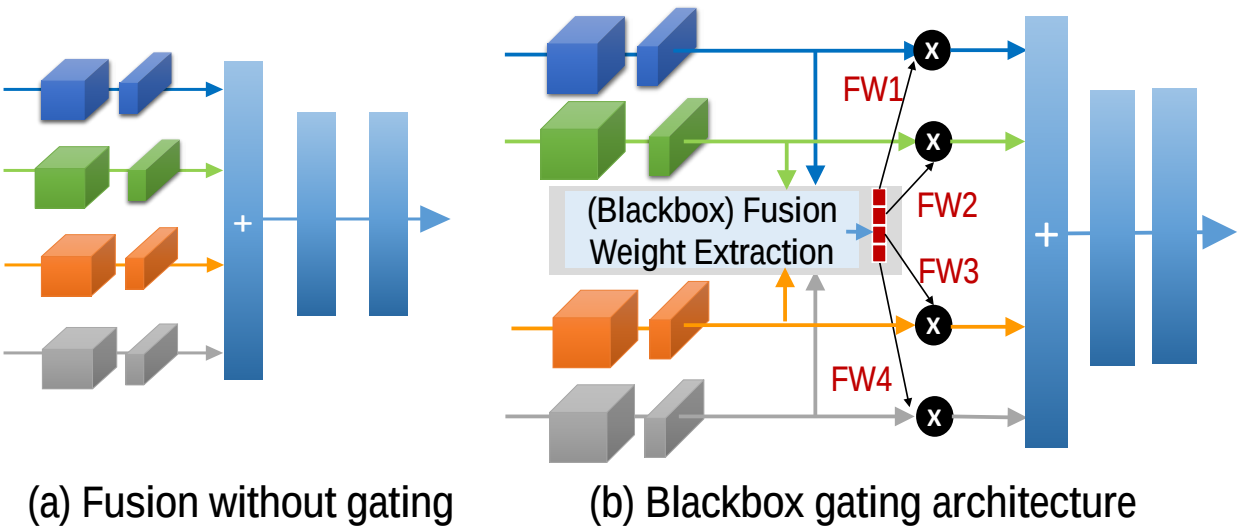

Figure 3.1: Two DNN sensor fusion architectures.

A key objective and challenge in the sensor fusion is securing *resilience*, i.e., the fusion task shall be conducted robustly not only under clean sensory inputs but also in the presence of sensor failures. [31], [32], and [33] display CNN based object tracking with input weighting (gating) fusion scheme and auxiliary losses. By weighting more important input channel with fusion weights (gating factors) which are processed with a sigmoid and softmax function, the weighted important inputs have more effect on outputs, and the classification accuracy is further improved. Furthermore, through the softmax function which makes the range of the fusion weights between [0, 1], the feature importance is easily interpreted.

We present a new gating architecture and its variants with significantly improved performance and resilience under both clean sensory inputs and sensor failures. Our main contributions are: **1)** propose a new auxiliary-model regulated gating architecture, called *ARGate*, to robustly learn gating fusion weights of different modalities using auxiliary unisensory processing paths during training; **2)** As part of the ARGate architecture, propose two regularization techniques, namely, *fusion weight regularization with auxiliary losses*, and *monotonic fusion target learning* to regularize fusion weights with corresponding auxiliary losses and target for consistent fusion weights in the training process and significantly improve the performance and robustness; and **3)** shed light on the fusion and regularization mechanisms that are responsible for the observed performance improvements.

We perform comprehensive evaluation of the proposed ARGate architectures while comparing with a baseline non-gating CNN architecture and the NetGated architecture [29] utilizing human activity recognition (HAR) dataset [34], driver identification dataset [35], and KITTI dataset [36] under various sensory input conditions. It is demonstrated that the proposed architectures consistently outperform the baseline and NetGated architectures, and improve classification accuracy by up to 8.49% and 13.39% over the baseline and NetGated architectures, respectively. For the KITTI dataset, the proposed architecture outperforms a reference architecture [26] by up to 4.81%.

3.1 Overview of the ARGate Architectures

The ARGate architecture with the two most essential regularization techniques, i.e. *fusion weight regularization with auxiliary losses* and *monotonic fusion weight target learning*, is depicted in Fig. 3.2.

Multisensory processing has been under extensive study in neuroscience and recent research has shed increasing light on the computational mechanisms underlying the multi-sensory processing of superior colliculus (SC) neurons in the midbrain [37, 38], providing a relevant reference for the proposed ARGate architectures. Animals must promptly select the most pertinent modalities while suppressing spurious noise from other senses for their survival, where *multisensory competition* plays an essential role. This is analogous to the targeted robust sensor fusion problem under catastrophic sensor failures. On the other hand, mature SC neurons are able to enhance their performance via *multisensory cooperation* where congruent stimuli across modalities are integrated. This corresponds to performance benefits in the targeted sensor fusion brought by fusing co-variant and complementary sensory inputs.

As such, multisensory competition and cooperation may be considered as two intertwined operating modes, transitions between which may be orchestrated based on sensory conditions: operate in competition when inputs are non-congruent (e.g. due to strong sensory noise or catastrophic failures) to ensure robustness while switching to cooperation to maximize performance with clean/covariant inputs.

Loosely speaking, the proposed architecture in Fig. 3.2 (a detailed full implementation is in Fig. 3.4) bears high-level resemblance to mode transitions in SC neurons suggested by the electrophysiological recording and computational studies [37]. However, it takes a rather different strategy to balance between multisensory competition and cooperation in the end-to-end deep learning architecture during training. The bottom block of Fig. 3.2 outputs the fusion weight targets (FW_T1 to FW_T4), which are used to regularize the fusion weight extraction block such that each fusion weight FW_i is constrained to be near the corresponding target FW_Ti. This acts as a solution to the fusion weight inconsistency problem of the typical gating architectures.

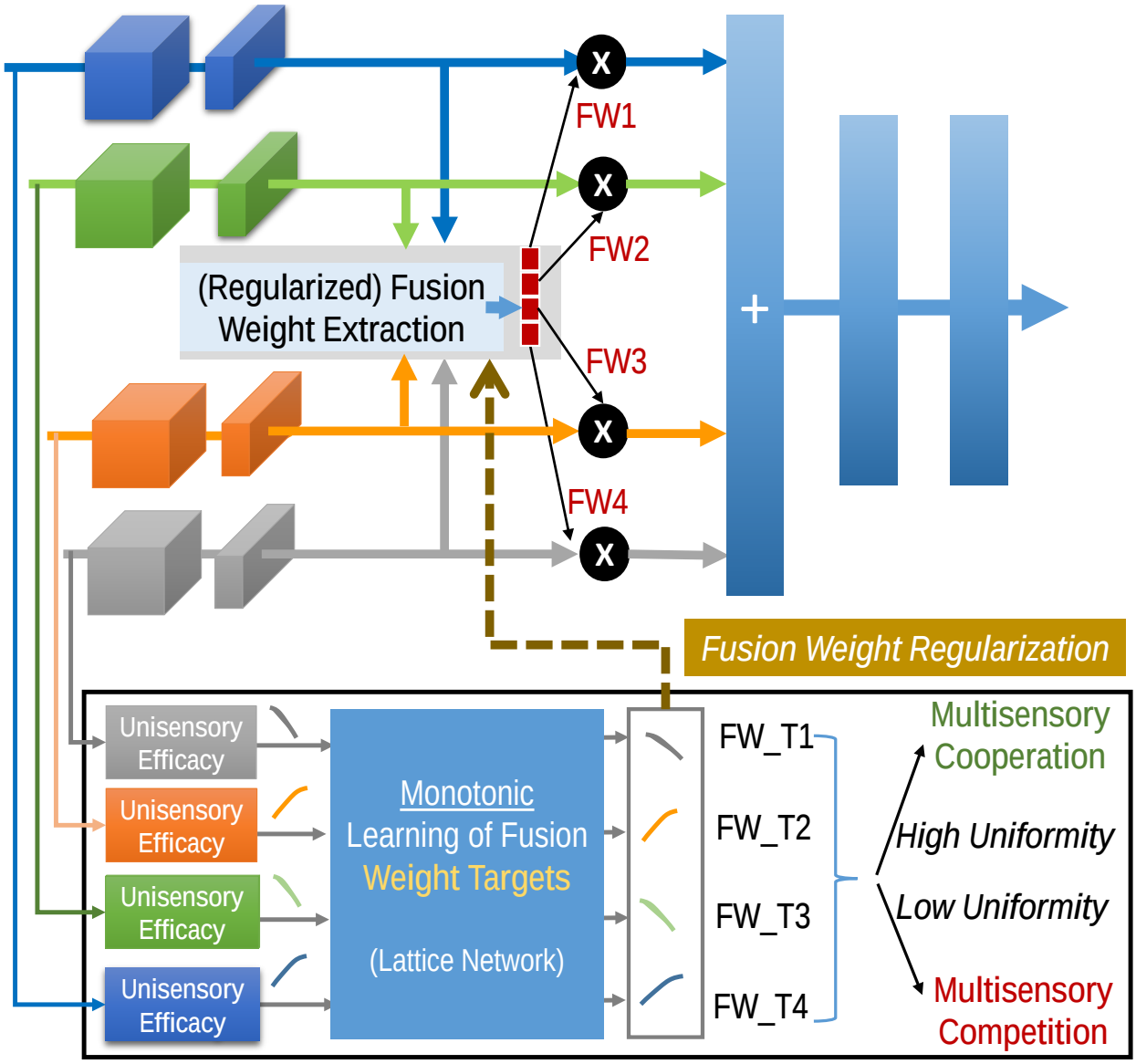

Figure 3.2: ARGate architecture with fusion weight regularization and monotonic fusion weight target learning. The bottom box offers fusion weight regulation for training the main model (upper portion) and is removed for inference.

The key idea here is to transparently balance between multisensory competition and cooperation by detecting the efficacy (i.e. conditioning or importance) of each unisensory input for the same end prediction task. The efficacy is evaluated by the training loss of a “unisensory efficacy" block, which is essentially a unisensory model for the same task with the corresponding modality being the sole input. Intuitively, a high unisensory efficacy shall be mapped to a high fusion weight target value. To allow for a flexible end-to-end architecture, the efficacies are mapped to the fusion weight targets (FW_T1 to FW_T4) by a trainable monotonic lattice network ensuring the monotonic relationship between the two. Since fusion weight targets sum up to 1.0, low uniformity among fusion weight targets immediately leads to multisensory competition where the inputs with smaller fusion weight targets tend to be depressed. High uniformity in the fusion weight targets would give rise to multisensory cooperation where modalities with similar fusion target values are integrated in a balanced way. The design of the ARGate architectures is detailed as follows.

3.2 The Proposed ARGate Architecture

3.2.1 Basic structure of ARGate

As a first step, we present a primitive ARGate architecture with weight sharing (ARGate-WS), which is enclosed in a more complete version of the ARGate architecture called ARGate+ in Fig. 3.3. The ARGate+ architecture will be built upon ARGate-WS and explained in the next subsection.

For ease of discussion, we assume that there exist two sensory inputs as in Fig. 3.3. To assist training of a main model, which is employed for inference, an auxiliary (aux) model is included. The main model is architecturally similar to the NetGated architecture. When splitting outputs of “FC-con" layer into fusion weights, we further introduce a sigmoid and a softmax function to normalize the fusion weights between [0,1], so that importance of input features are expressed with magnitude of the fusion weights.

In general, the aux model consists of multiple independent auxiliary paths, one for each modality without fusion. Weight parameters in convolutional layers and early FC layers are shared

between the corresponding modalities across the main and aux models. Total training loss is a weighted sum of the losses of the main model and all auxiliary paths. The adopted weight sharing (WS) is illustrated by the dashed purple box in Fig. 3.3. Weight sharing is commonly used in the literature [27, 31]. Here, it acts as a way of regularizing the main model.

Based on this primitive ARGate-WS architecture only with the weight sharing, we explore the more complete ARGate+ architecture with two fusion weight regularization techniques next.

3.2.2 Fusion Weight Regularization with Auxiliary Losses: ARGate+ Architecture

When one or more sensory channels are completely corrupted, weight sharing in ARGate-WS fails to properly regularize the corresponding convolutional/FC layers for the corrupted input channels in the main model. This is because that these auxiliary paths can no longer be trained to deliver a good performance based on the single corrupted modality.

Our key observation is that the losses of different auxiliary paths reflect the relevance of these modalities w.r.t. to the classification task, and hence, can be used for *fusion weight regularization* (FWR), which is depicted using a purple dashed line pointing from the aux model to the main model in Fig. 3.3. When a particular sensory modality is corrupted under an input example, a high loss of its auxiliary path will constrain the training of the “FC-con" layer in the main model to produce a low fusion weight for that modality.

Since the loss of each auxiliary path is included in the total training loss, the large loss of any corrupted sensory input can be dominant, potentially lowering performance. To resolve this problem, we use the extracted fusion weights from the main model as the corresponding weights of the auxiliary losses in the total training loss. We call this scheme auxiliary loss weighting (ALW) as shown by the dashed purple line pointing from the main model to the aux model in Fig. 3.3. This leads to a more complete ARGate+ architecture with a new loss function:

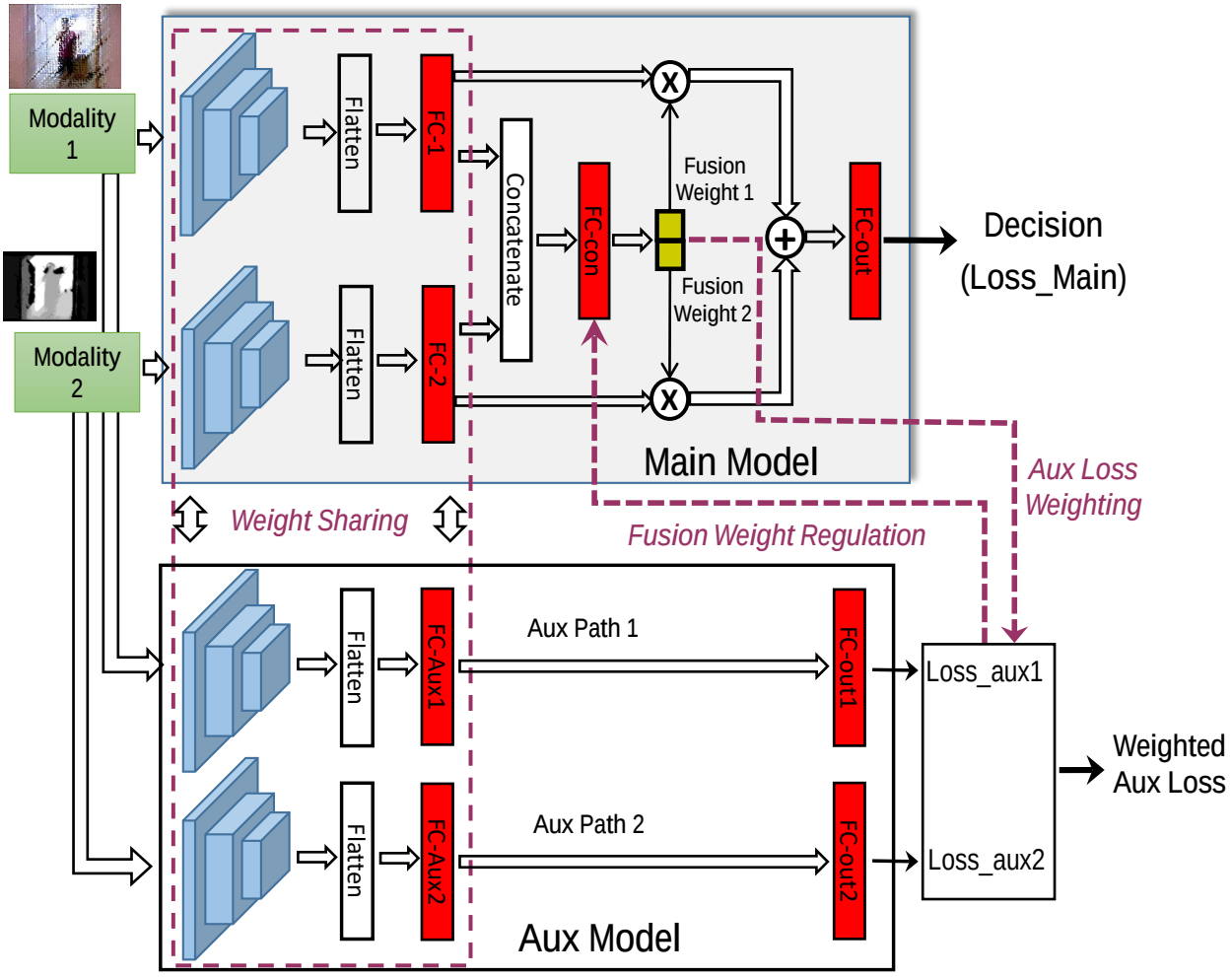

Figure 3.3: The proposed ARGate+ architecture

$$Loss = \quad Loss_{main} + \alpha \cdot \sum_{k=1}^{K} w_{fusion}^{k} Loss_{aux}^{k} + \beta \cdot \sum_{k=1}^{K} \left(w_{fusion}^{k} - \widehat{e^{-Loss_{aux}^{k}}^{2}} \right)^{2} \tag{3.1}$$

where the auxiliary path loss $Loss_{aux}^{k}$ is weighted by the fusion weight w_{fusion}^{k}, hence the dominance of any large auxiliary path loss resulted from sensor failures is diminished.

Each normalized auxiliary-path loss $\widehat{e^{-Loss_{aux}^{k}}^{2}}$ is employed to regularize w_{fusion}^{k}. Here, $\widehat{e^{-Loss_{aux}^{k}}^{2}}$ is obtained by plugging $Loss_{aux}^{k}$ into the exponential function and then normalizing $e^{-Loss_{aux}^{k}}^{2}$ using sigmoid and then softmax normalization so that $\widehat{e^{-Loss_{aux}^{k}}^{2}}$ is between [0,1].

With above-mentioned explanation, the last term of the right hand side of (3.1) is directly used as fusion weight targets for regularizing the fusion weights of the main model. The ARGate+ architecture integrates weight sharing (WS), fusion weight regularization with auxiliary losses (FWR), and auxiliary loss weighting (ALW). Next, we discuss the other key proposed technique, namely, monotonic fusion target learning.

3.2.3 Monotonic Fusion Target Learning

Regularizing fusion weight through auxiliary losses is a key step towards improving overall performance. Notice that the last term in (3.1) implements fusion weight regularization in which the exponential of each normalized auxiliary path is called the fusion weight target for the corresponding fusion weight. While this specific form of fusion weight targets pushes the training of the network towards producing a low fusion weight value for modalities with a large auxiliary path loss, the optimal mapping from the auxiliary path loss to the corresponding fusion weight target is not known *a priori*. Our key idea is to optimize this mapping end-to-end as part of the overall network architecture based on the available training data.

Since the fusion weight target of an auxiliary path shall be a *monotonically* non-increasing function of the corresponding auxiliary path loss, this motivates us to introduce a dedicated small

network to learn the mapping from the set of auxiliary path losses as following expression $\vec{Loss}_{aux} = [Loss^1_{aux}, Loss^2_{aux}, \cdots, Loss^K_{aux}]^T$ to each fusion weight target $w^k_{fusion,t}$: $w^k_{fusion,t} = f_k(\vec{Loss}_{aux})$. Corresponding, the loss is modified from (3.1) to:

$$\begin{aligned} Loss = \quad & Loss_{main} + \alpha \cdot \sum_{k=1}^{K} w^k_{fusion} Loss^k_{aux} \\ & + \beta \cdot \sum_{k=1}^{K} \left(w^k_{fusion} - f_k(\vec{Loss}_{aux}) \right)^2 . \end{aligned} \tag{3.2}$$

The mappings from the auxiliary losses to the fusion weight targets may be learned with great flexibility by a generic feed-forward multi-layer neural network. However, this approach is not robust due to lack of regularization. We propose to embed a regularized deep lattice network (DLN) [39] to the ARGate+ architecture for more robust learning of fusion targets. As illustrated in Fig. 3.5, a lattice network can learn input-to-output mappings while guaranteeing the user-specified full or partial monotonicity between the a set of the inputs and outputs.

Integrating a DLN network into ARGate+ leads to a new architecture called ARGate-L in Fig. 3.4. In general, DLNs consist of three types of layers: calibrators, linear embeddings, and lattices, all of which can be constrained to obey partial or full monotonicity between selected inputs and outputs. Linear embedding layers map the inputs linearly to the outputs. Monotonicity between a subsest of inputs/outputs can be forced by choosing non-negative coefficients between them. A lattice is a linearly interpolated multi-dimensional look-up table; each output of the lattice can be constrained to be monotonic in a subset of the inputs. Calibrators are 1-d lattices and can nonlinearly transform a single input and may be used for pre-processing and normalization between layers in the DLN.

The proposed lattice network is illustrated in Fig. 3.4 for the general case of mapping K auxiliary path losses to the corresponding K fusion targets. In its most general form, this can be done by having K independent subnetworks, one for each fusion target. While a particular architecture is chosen for each subnetwork in Fig. 3.4, in practice, it can be simplified/optimized to suit a given

application. The key idea in the proposed lattice network design is to ensure the partial monotonicity between each pair of auxiliary loss and fusion target; there are K such constraints. In the lattice network of Fig. 3.4, monotonic inputs/outputs are processed by a calibrator in dashed green while non-monotonic ones are processed by a calibrator in solid gray. The output of a component is monotonic if any of its inputs is monotonic to ensure the end-to-end monotonicity between a pair of auxiliary loss and fusion targets. Specifically, there exist a green dashed path from each auxiliary loss k to its corresponding fusion target k. Imposing these monotonicity constraints acts as a regularization mechanism, making the robust end-to-end learning of fusion weight targets possible and leading to improved performance as we demonstrate later.

3.3 Experimental Settings

We perform comprehensive comparison of a non-gating fusion CNN baseline [27], the NetGated architecture [29], and variants of the proposed ARGate architecture on the HAR [34], driver identification [35], and KITTI [36] datasets.

Mini-batch scheme size is 16 for the HAR dataset, and 128 for the driver identification dataset. We train these networks with 200 epochs for the HAR dataset and 100 epochs for the driver identification dataset. For the HAR and the driver identification dataset, an ADAM optimizer is utilized with a learning rate of 0.001. Cross-entropy loss is chosen as the loss for the CNN baseline, the NetGated architecture, and the main model and auxiliary paths of ARGates. For the KITTI dataset, an ADAM optimizer is adopted with an initial learning rate of 0.0001, which decays every 30K iterations with 0.8 as a decay factor. All simulations are done with Python 3.6, Pytorch 0.4.0 [40] for the HAR and driver identification datasets, and Tensorflow 1.3.0 [41] for the KITTI dataset using a NVIDIA Tesla K80 GPU.

3.3.1 Datasets

3.3.1.1 The Human Activity Recognition Dataset

The human activity recognition (HAR) dataset [34] includes six activities to be recognized: walking, walking upstairs, walking downstairs, sitting, standing, and laying. From accelerome-

ter and gyroscope, nine sensory inputs are utilized for experiments, where each sensory input is distributed between $[-1, 1]$. There are 7,352 examples for training and 2,947 examples for testing.

3.3.1.2 The Driver Identification Dataset

The driver identification dataset [35] consists of 10 drivers' cruising data collected by a CarbigsP OBD-II scanner. There are 51 features in total, but 15 features are used in our experiments which is same experimental setup from the original paper. 75,501 training and 18,879 testing examples are used.

3.3.1.3 KITTI Dataset

In the KITTI dataset [36], there are two sensory inputs: a RGB image and a velodyne laser scanned bird eye view (BEV) image. The provided 7481 training frames are split into a training and a validation set. For evaluation with detecting the *car* class in images, we adopt easy, moderate and hard difficulty-level settings provided by KITTI.

3.3.2 Neural Network Configurations

3.3.2.1 Configurations for the HAR and Driver Identification Datasets

In the HAR and driver identification datasets, the non-gating CNN baseline, the NetGated, and the proposed ARGate variants are compared. The late fusion scheme is utilized in the CNN baseline. For fair comparison, tunable parameters of all neural networks are closely matched.

3.3.2.2 Configurations for the KITTI Dataset

AVOD Baseline. We use the Aggregate View Object Detection (AVOD) approach [26] as a baseline.

AVOD-ARGate. We embed our ARGate-L into the AVOD, specifically in a region proposal network(RPN) as shown in Fig. 3.6. Feature maps from each input are passed into small ARGate block to produce fusion weights which are regularized by a lattice network. Outputs from these FC layers are added and passed onto the *fusion* block (a FC layer) to create the fusion weights. These fusion weights are multiplied with the corresponding feature maps from the outputs of the

1x1 feature extracting layers. Furthermore, the fusion weights from the RPN is utilized again as gating factors for each modality in the second stage fusion(AVOD) to fortify network robustness. For target fusion weight learning, the lattice architecture (Cal-Lin-Cal-Lat) is used with the linear embedding layers processing two input channels and one lattice. The total number of tunable parameters of the AVOD baseline and AVOD-ARGate are closely matched for fair performance comparisons. Moreover, since AVOD based algorithms need plane data for training and test, we utilize plane data provided by AVOD for training set, and custom plane data for test set for fair comparison.

3.3.3 Sensor Failures

Apart from using the clean data in the HAR and driver identification datasets, we introduce sensor failures and set up various training/testing sets to comprehensively compare the robustness/generalization of different architectures.

3.3.3.1 Modeling of failing sensors

All clean scalar inputs are normalized to be within $[-1, 1]$. We use two schemes: *uniform*, and *Gaussian*, to model a failing sensor by setting its input values respectively to pure noise following a uniform distribution between $[-1, 1]$, and pure noise following the Gaussian distribution $\mathcal{N}(0, 1)$.

3.3.3.2 Corrupted examples for training/testing

For HAR and driver identification datasets, we use clean and corrupted examples in both the training and test sets, where in each set $\frac{1}{3}$ of the examples are randomly chosen and kept clean while the remaining ones are corrupted by one or more failing sensors using one of the approaches described below.

Fixed Failing Sensor Assignment. This mimics the situation in which a number of sensors have failed permanently. We select n_{fclean} channels out of a total of n sensors to be clean and assume that all remaining sensors have permanent failure when setting up the corrupted examples of for both the training and test sets in each experiment.

Random Failing Sensor Assignment. To more closely mimic random nature of sensor fail-

# Clean Channels	Failure Model	Baseline	NetGated	ARGate-WS	AR Gate+	AR Gate-L
All Clean	-	94.06	94.50	94.96	95.69	**96.71**
n_{rclean}=8	Uniform	92.35	92.20	92.45	92.57	**94.06**
	Gaussian	92.94	93.28	94.97	94.13	**94.81**
n_{rclean}=5	Uniform	86.73	86.80	88.53	87.68	**88.77**
	Gaussian	88.41	89.04	89.52	90.12	**91.35**
n_{rclean}=1	Uniform	62.06	62.90	65.69	67.09	**69.63**
	Gaussian	69.67	70.54	71.83	**73.19**	72.78

Table 3.1: Prediction accuracies(in %) under clean data and random failing sensor assignment for the HAR dataset.

ures, for each corrupted example in the training/test set, we randomly select n_{rclean} channels out of all n sensors to be clean and corrupt the remaining channels. As such, sensors that have failed may vary from one example to another. n_{fclean} and n_{rclean} are varied to for different severity levels of failures.

Failing Sensor Generation Test. This tests model generation by using a test set containing corrupted examples which have a larger or different number of failing sensors from the corrupted examples used to train the model, i.e. the test set has examples with a severe level of sensor failure.

3.4 Evaluation

3.4.1 Quality of Fusion Weight Extraction

To shed some light on the fusion mechanisms of NetGated, ARGate-WS and the proposed ARGate+ architectures, we examine distributions of the trained fusion weight of a sensory input $total\_acc\_y$ under the HAR dataset. If the $total\_acc\_y$ is corrupted, then the fusion weight of itself should be distributed lower than the fusion weight of clean $total\_acc\_y$. Please note that ARGate-WS only utilizes weight sharing so that we compare our proposed ARGate+ with ARGate-WS in terms of fusion weight regularization. The sensor failures are modeled using the random failing sensor assignment with uniform distribution and with $n_{rclean} = 1$ so that 8 out of 9 sensory inputs are corrupted in each example. To show effects of sensor failures on the fusion weights, we split

the examples into two subsets: one in which $total\_acc\_y$ is corrupted with 8 other inputs and the other subset where $total\_acc\_y$ in the only clean input. Fig. 7(a, b, c) displays the fusion weight distributions of the first subset for the three architectures while Fig. 7(d, e, f) shows those of the second subset.

One may expect that, in a properly trained model, the fusion weight value for a corrupted sensory modality shall be much smaller than when the modality is clean. However even when $total\_acc\_y$ is corrupted, Fig. 7(a, b, c) show that the $total\_acc\_y$ fusion weight distribution of the NetGated architecture has a peak around the large value of 0.4 which is not presented in the case of ARGate-WS. The distribution of ARGate-WS has a much reduced mass on large fusion weight values compared to that of the NetGated. Furthermore, the fusion weight value can go beyond 0.4 in NetGated and ARGate-WS while it is pretty much constrained between 0.06 and 0.1 in ARGate+. One also expects that the fusion weight of $total\_acc\_y$ shall be large for the second subset since $total\_acc\_y$ is the only clean input. However, as seen in Fig. 7d, the distribution of NetGated has a large population mass on low fusion weight values. The population mass on low fusion weight values gets reduced significantly in the case of ARGate-WS Fig. 7e, which is further reduced in ARGate+ for which most fusion weights are distributed between 0.33 and 0.45 as shown in Fig. 7f.

# Clean Channels	Baseline	NetGated	AR Gate+	AR Gate-L
n_{fclean}=6	87.68	89.28	91.01	**92.97**
n_{fclean}=5	80.59	81.94	84.52	**89.01**

Table 3.2: Prediction accuracies(in %) under fixed failing sensor assignment for the HAR dataset.

3.4.2 Results on the HAR Dataset

[Fixed Failing Sensor Assignment] We consider two cases where the number of clean input channels n_{fclean} is set to 5 and 6, respectively. When $n_{fclean} = 5$, $body\_total\_acc\_x$ $body\_acc\_x$

and $body\_gyro\_x$ are corrupted and set to uniform-ally distributed noise while $body\_acc\_z$ and $body\_gyro\_x$ are corrupted by uniform-ally distributed noise when $n_{fclean} = 6$. Table 3.2 shows that our proposed ARGate architectures outperform the CNN and NetGated significantly while adopting the lattice network in ARGate-L further improves over ARGate+.

[Random Failing Sensor Assignment] In Table 3.1 compares the baseline CNN, NetGated, ARGate-WS, proposed ARGate+, and ARGate-L architectures with the number of randomly chosen clean sensors $n_{rclean} \in \{1, 5, 8\}$. When all channels are clean, NetGated has 0.44% prediction accuracy improvement over the baseline while ARGate+, and ARGate-L outperform the baseline by 1.63%, and 2.65%, respectively. ARGate architectures always have better accuracy than the baseline and NetGated, and in general ARGate+ further improves over ARGate-WS which only employs weight sharing (WS) between the main and auxiliary model, demonstrating the effect of fusion weight regularization with auxiliary losses (FWR). ARGate-L is the best-performing model, which incorporates WS, FWR, and the lattice network for monotonic fusion weight target learning. With $n_{rclean} = 1$ and sensor failures which are modeled using uniformly distributed noise, ARGate-L outperforms the baseline, NetGated, ARGate-WS, and ARGate by 7.57%, 6.73%, 3.94%, and 2.54%, respectively. We expect that the performance improvements of the ARGate architectures may be attributed to their improved quality in the fusion weight target learning.

# Clean Channels	Baseline	NetGated	AR Gate+	AR Gate-L
(1,2)(3,8)	72.91	72.75	77.10	**77.16**
(1,3)(4,8)	70.98	70.78	75.43	**76.15**
(1,4)(5,8)	69.38	69.53	73.06	**73.64**

Table 3.3: Prediction accuracies(in %) under failing sensor generation test for the HAR dataset.

[Failing Sensor Generation Test] In Table 3.3, the first column specifies the numbers of randomly chosen failing channels used in the training and test sets. For example, (1,2)(3,8) means

# Clean Channels	Baseline	NetGated	AR Gate+	AR Gate-L
n_{fclean}=5	79.48	81.46	82.29	**87.73**
n_{fclean}=7	90.39	92.81	94.56	**95.34**

Table 3.4: Prediction accuracies(in %) under fixed failing sensor assignment for the driver identification dataset.

# Clean Channels	Failure Model	Baseline	NetGated	AR Gate+	AR Gate-L
All Clean	-	95.49	95.79	96.65	**96.78**
n_{rclean}=12	Uniform	93.43	92.19	93.50	**93.58**
	Gaussian	92.19	91.40	92.31	**93.16**
n_{rclean}=8	Uniform	83.50	80.61	85.08	**86.86**
	Gaussian	80.82	79.19	82.23	**85.48**
n_{rclean}=5	Uniform	65.36	62.21	68.96	**73.85**
	Gaussian	65.84	62.69	68.20	**76.08**

Table 3.5: Accuracies(in %) under clean data and random failing sensor assignment for the driver identification dataset.

that the number of failing sensors for training are randomly picked from [1,2] while the range of the failing sensors for testing is [3,8]. Essentially, we evaluate the generalization of the models by including corrupted examples with more failing channels in the test set than in the training set. Failing sensors are modeled using uniformly distributed noise. NetGated can underperform the baseline while ARGate-L always outperforms the baseline and NetGated by upto 5.17% and 5.37%, respectively.

3.4.3 Results on the Driver Identification Dataset

[Fixed Failing Sensor Assignment] In Table 3.4, the corrupted inputs are set to uniformly distributed noise. When $n_{fclean} = 5$, *Long Term Fuel Trim Bank1, Maximum indicated engine torque, Calculated LOAD value, Activation of Air compressor*, and *Engine coolant temperature* are corrupted. When $n_{fclean} = 7$, two more input channels, *Intake air pressure* and *Fuel consumption*

are corrupted. ARGate-L significantly outperforms the other two models. When $n_{fclean} = 5$, ARGate-L improves over the baseline and NetGated by 8.25% and 6.27%, respectively.

[Random Failing Sensor Assignment] In Table 3.5, different models with the number of randomly selected clean channels $n_{rclean} \in \{5, 8, 12\}$ are compared. When all 15 input channels are clean, the proposed ARGate-L improves the baseline and NetGated by 1.29% and 0.99%, respectively. In many cases, NetGated is worse than the baseline. ARGate-L always has the best performance among all models. For example, with $n_{rclean} = 5$ and the uniform noise sensor failure model, ARGate-L significantly outperforms the baseline and NetGated by 8.49% and 13.39%, respectively.

[Failing Sensor Generation Test] We compare the generalization of two models in Table 3.6 with the same notation of Table 3.3. ARGate-L demonstrates noticeable improvements over all other models.

# Failing Channels	Baseline	NetGated	AR Gate+	AR Gate-L
(1,2)(3,15)	60.38	60.39	59.42	**64.16**

Table 3.6: Prediction accuracies(in %) under the failing sensor generation test for the driver identification dataset.

3.4.4 Results on the KITTI Dataset

We compare our ARGate-L architecture to the AVOD baseline [26] on car detection in KITTI validation and test set. Average precision (AP) in 2D image frame, oriented overlap on image, AP in BEV, and AP in 3D metrics are used for performance comparison.

In terms of validation results in Table. 3.7, 1.17% improvement is made on 3D car detection benchmark in moderate difficulty. Furthermore, about 0.4% of improvements are found from BEV AP. With fusion weights utilized on both RPN and AVOD network, our ARGate-L architecture improves the detection performance in the validation set.

For test results evaluated by official KITTI online server, since the AVOD plane data is not provided for test set, we use our custom plane data for AVOD and ARGate-L for fair comparison. Based on this setup, in the moderate difficulty, our ARGate architecture improves 4.63% in 2D car detection benchmark. For orientation, 4.32% improvement is observed. Lastly, for 3D detection and BEV, the proposed architecture outperforms AVOD by 4.81% and 2.22%, respectively. Furthermore, similar range of improvements are shown on the hard difficulty, which shows strength of the proposed architecture, especially on touch situation. Overall, our proposed techniques outperform the baseline AVOD rather noticeably.

Network	Benchmark	Easy	Moderate	Hard
AVOD	Car (3D Detection)	84.41	74.44	68.65
	Car (BEV)	89.72	86.85	79.69
AVOD-ARGate	Car (3D Detection)	**84.61**	**75.61**	68.65
	Car (BEV)	**89.95**	**87.23**	**79.89**

Table 3.7: Average Precision (in %) comparison of car detection on the KITTI *validation* set.

Network	Benchmark	Easy	Moderate	Hard
AVOD	Car (Detection)	90.17	79.77	74.84
	Car (Orientation)	89.96	79.19	74.16
	Car (3D Detection)	73.32	59.74	55.08
	Car (BEV)	88.06	77.80	71.16
AVOD-ARGate	Car (Detection)	**92.36**	**84.40**	**79.59**
	Car (Orientation)	**92.06**	**83.51**	**78.63**
	Car (3D Detection)	**76.52**	**64.55**	**60.01**
	Car (BEV)	87.85	**80.02**	**75.39**

Table 3.8: Average Precision (in %) comparison of car detection on the KITTI *test* set.

3.5 Summary and Discussions

In this chapter, We have proposed the ARGate architectures for resilient sensor fusion by addressing the limitations of the conventional fusion schemes including the existing gating architectures. Leveraging the two proposed regularization techniques, namely, fusion weight regularization with auxiliary losses weighting, and monotonic fusion target learning, the proposed gating architectures incorporate an auxiliary model to regularize the main model to robustly learn the fusion weight for each modality. Our architectures have demonstrated significant performance improvements over other models particularly in the presence of sensor failures.

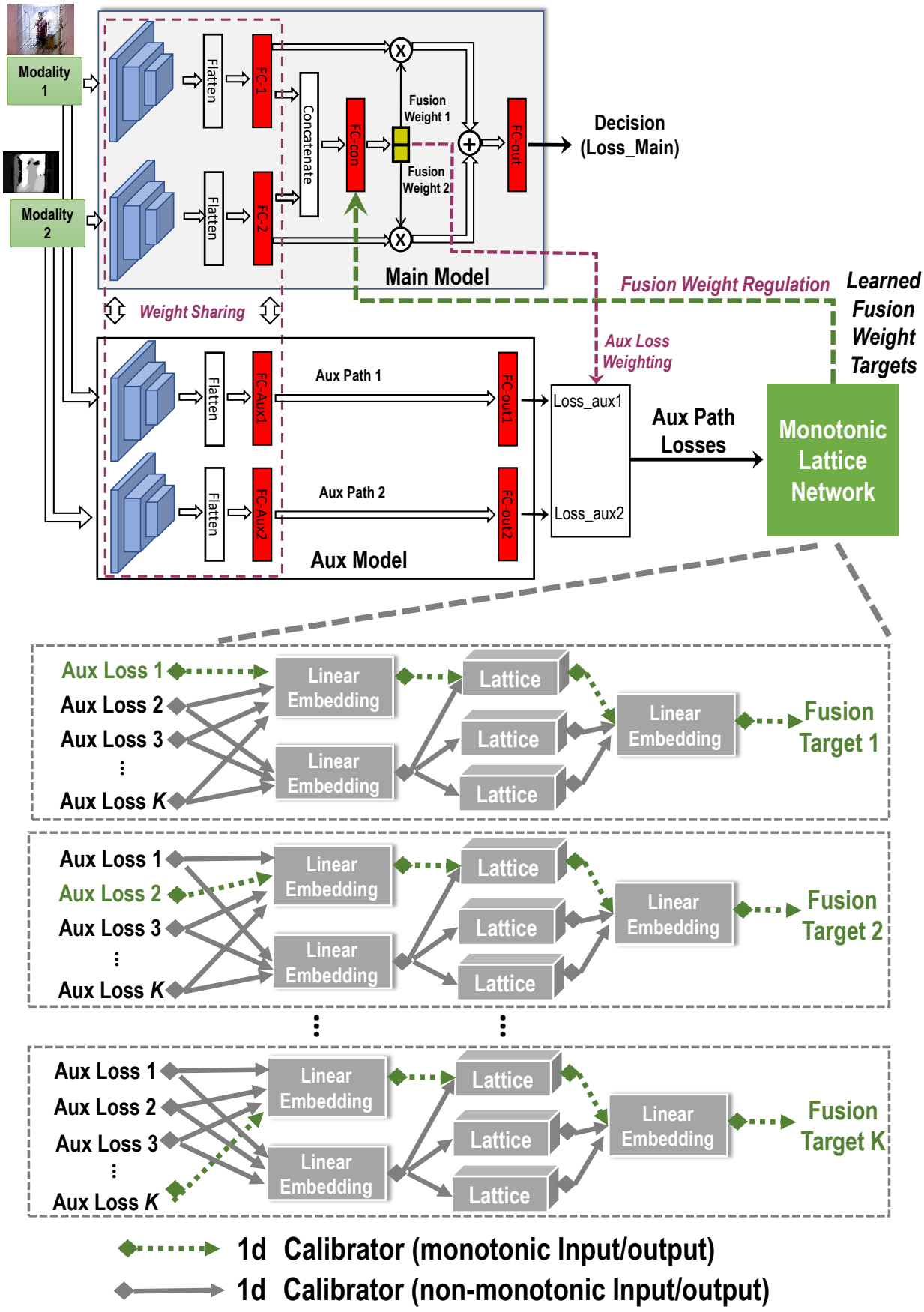

Figure 3.4: The proposed ARGate-L architecture with end-to-end monotonic learning of fusion targets using a lattice network.

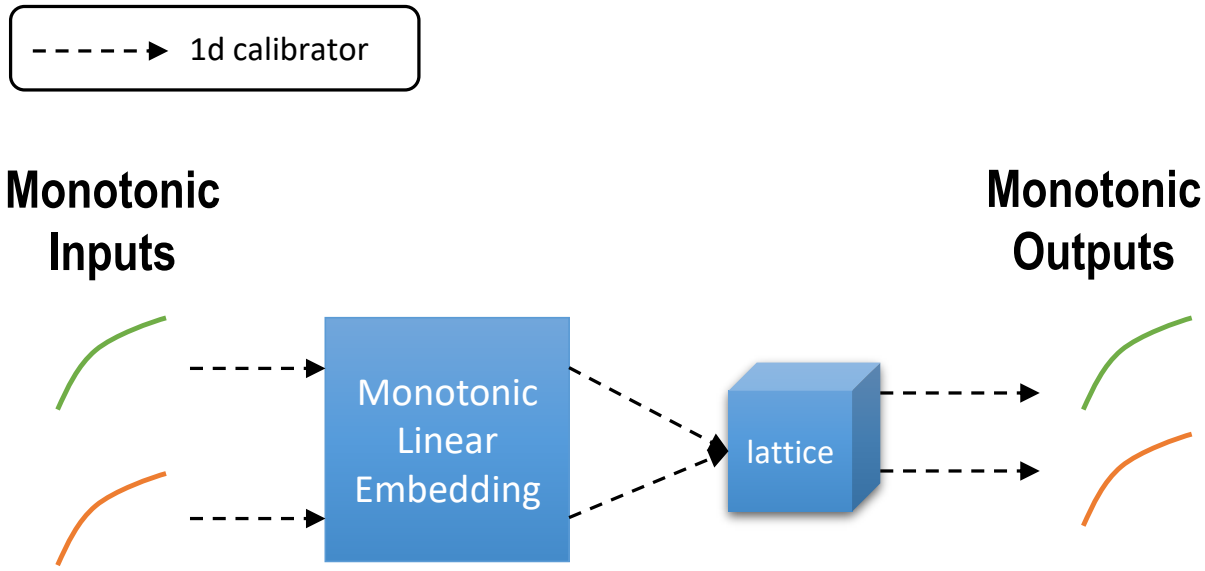

Figure 3.5: Learning of monotonic input-output mappings.

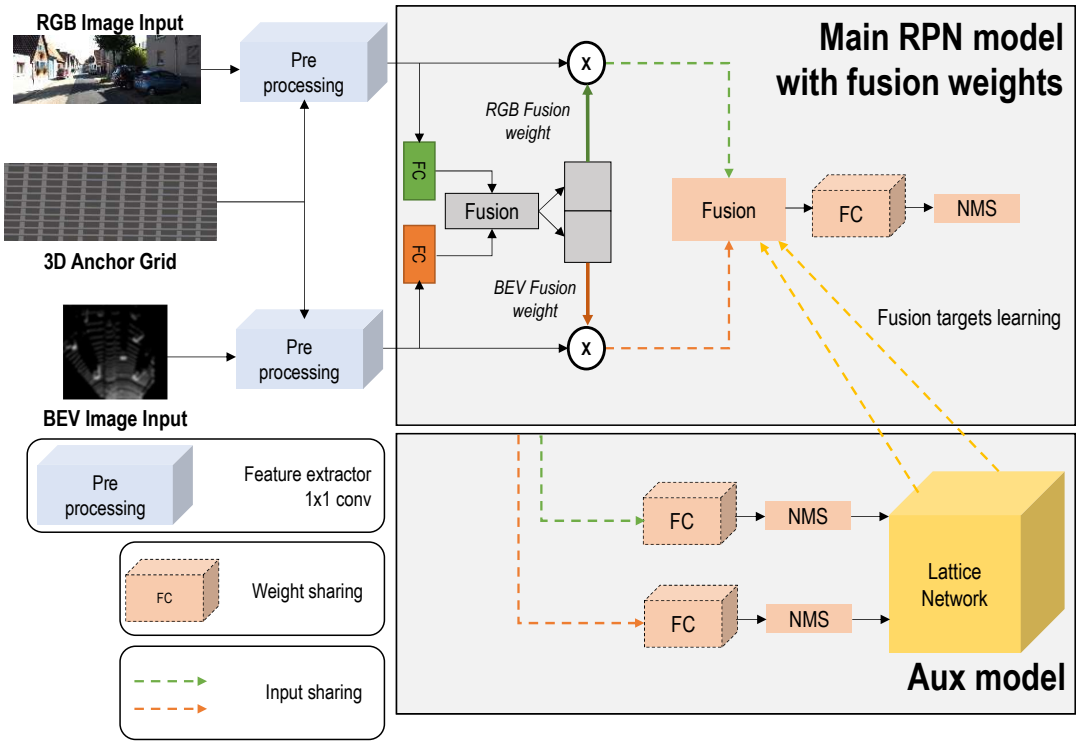

Figure 3.6: The proposed ARGate architecture with RPN model for training.

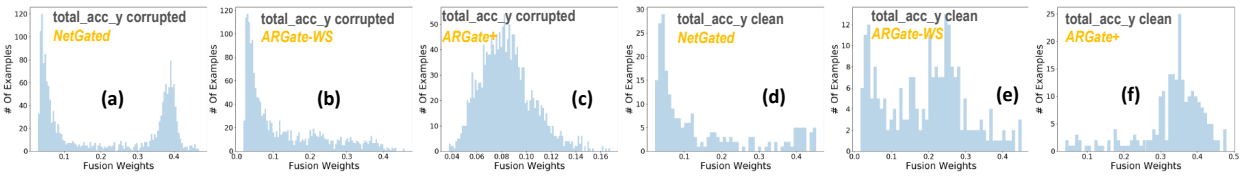

Figure 3.7: Fusion weight distributions of the clean or corrupted channel total_acc_y extracted by NetGated, ARGate-WS and ARGate+ under random failing sensor assignment with $n_{rclean} = 1$. (a),(b) and (c) show the fusion weights distributions of the NetGated, ARGate-WS and ARGate+ models, respectively, when total_acc_y is corrupted. (d),(e) and (f) are the distributions of the NetGated, ARGate-WS and ARGate+ models, respectively, when total_acc_y is clean.

4. DATA EFFICIENT LEARNING TECHNIQUES FOR RARE FAILURE DETECTION

Analog and mixed-signal (AMS) systems have complex and strict requirement from design to tape-out. For example, in biomedical device and circuit industry, stringent reliability is required e.g. 1 DPPM (defective parts per million) or less. This severe condition is very challenging for circuit design, failure detection and verification of a design.

Most common practical verification method is to use Monte Carlo (MC) simulations to detect rare failures. However, MC methods computationally expensive in terms of simulation time so many sampling methods are proposed [42–47]. [42, 47] present importance sampling method with Bayesian optimization (BO) to find the global extrema and high trustworthy estimated failure detection with few circuit simulations. In [46], a correlated Bayesian inference (CBI) based algorithm is implemented for system-level failure estimation in large-scale circuit systems. This work show around 10x runtime reduction without the loss of accuracy. However, for the extremely rare failure in a bounded parameter space is not considered. Furthermore, the results from these works are based on mid-level amount of dimensionality in their experimental data.

On the other hand, in terms of the same high dimensionality problem from AMS data, through rapid advance of the deep learning, and high performance of computing power, large neural networks for big data are developed and further expanding its limits. However, as the number of data sample increases, the amount of redundant and noisy input features also increases significantly, which affects both inference performance and robustness of neural network models. Therefore, extracting essential and informative features from the datasets is crucial for building robust networks.

To resolve the aforementioned issues, a dimension reduction scheme is widely used for mapping high-dimensional data into a low-dimensional input space. While reversible residual network (RevNets) are getting attention due to its reducing memory usage in back propagation and competitive performance on unsupervised learning problems [45, 48, 49], we focus on the factor that loss of information on the input can be avoided through the RevNets. [45] proposed a RevNet based

dimension reduction method, which is called nonlinear level-set learning (NLL) for transforming of input space to low-dimensional structures of the target functions. The NLL can extract the nonlinear information from the level sets so complex rocket inter-stage manufacture dataset results are demonstrated as an example of nonlinear dimension reduction problems.

In this chapter, we propose a reversible gating neural network and its variant with improved performance for the rare failure detection problem. Our main contributions are: **1)** propose a new RevNet based auxiliary-model regulated gating architecture, called *Rev-ARGate*, to utilize gating fusion weights of each pre-processed features from the RevNet in the input stage as feature importance factor; **2)** After checking with feature importance, propose two backward restoration methods, *essential input restoration with zero value* and *all input restoration* to restore from pre-processed feature space to the original input space through Bayesian neural network (BNN); and **3)** investigate restoration mechanisms which are responsible for the rare failure detection performance improvements.

As an extension to our work in [50], the original ARGate is deployed as the main fusion model and the RevNet is adopted for processing the input data. Our proposed Rev-ARGate shows the better performance to find the worst case failure in the high-dimensional datasets.

4.1 Failure Detection Problem Formulation

Under a given D-dimensional parameter space $\Omega \subseteq \mathbb{R}^D$, the goal of failure detection is to find a failure point x to meet the following requirement:

$$y(x) < T, \exists x \in \Omega \tag{4.1}$$

where T is the threshold target for specification requirement, and the $y(x)$ represents the performance of circuit at the parameter variation vector x. When the value of $y(x)$ is smaller than the threshold T, the performance is considered as the failure with the specific point x. Due to the nature of $y(x)$ which is severely nonlinear in the high dimensional space, it is hard and costly to get this value in terms of simulation time and computational resources.

Based on the problems above, we slightly modify the failure detection problem into the optimization issue below.

$$\min_{x \in \Omega} y(x) < T \tag{4.2}$$

[43] proposes a high-dimensional Bayesian optimization (HDBO) to optimize acquisition function by exploring random embedding to dimensionality reduction. The key idea of this solution is to find the lowest number of d_e-dimensional effective linear subspace $\mathcal{V}$ in the original D-dimensional space, which does not affect the overall performance. In [51], it is shown that $z \in \mathbb{R}^d$ exists with a random matrix $A \in \mathbb{R}^d$ following $N(0, 1)$ where embedding dimension d is equal or larger than d_e. Therefore, the original high-dimensional input space can be ported into a low dimensional space through random embedding with a random matrix. Furthermore, the failure search region is defined as bounded hyper-cube $[-\sqrt{d}, \sqrt{d}]^d$.

We adopt the HDBO with the proposed Rev-ARGate architecture to optimize the objective function $y(x)$. Note that the function $y(x)$ is considered as a black-box objective function due to the fact that the mapping from input x to y is challenging.

4.2 Overview of RevNet

Reversible residual network (RevNet) in [48, 49] is one of the variants from Residual neural network (ResNet) [52]. Most neural networks are trained with back propagation scheme. However, activation values need to be saved in computer memory and the number of activation is proportional to the size of neural network, causing bottleneck issue. [49] provides a solution to this problem without saving activation to the memory. In this work, we focus on bijective nonlinear transformations in [45] by exploiting the reversibility of the RevNet for the dimensionality reduction as follows.

$$z = g(x) \in \mathbb{R}^d \quad and \quad x = g^{-1}(z) \tag{4.3}$$

where $z = (z_1, .., z_d)^T$. Although $z \in \mathbb{R}^d$ defined in $\mathbb{R}^d$, the z can be partitioned into two

channels: $z = (z_{act}, z_{inact})$ with $dim(z_{act})$ which is much smaller than d so that $f \circ g^{-1}(z)$ is only affected by z_{act}.

Based on the bijective nature of the RevNet, we utilize the RevNet model from [53, 54] as follows.

$$\begin{aligned} u_{n+1} &= u_n + hK_{n,1}^T \sigma(K_{n,1} v_n + b_{n,1}), \\ v_{n+1} &= v_n - hK_{n,2}^T \sigma(K_{n,2} u_{n+1} + b_{n,2}) \end{aligned} \tag{4.4}$$

for $n = 0, 1, ..., N-1$, where u_n and v_n are state partitions, the time step scalar h, weight matrices $K_{n,1}$, $K_{n,2}$, biases $b_{n,1}$, $b_{n,2}$ and the activation function σ.

To define nonlinear transformation $g : x \mapsto z$, the input x is partitioned equally into two groups u_0, and v_0 and the output of RevNet z with u_N and v_N as well. So, the nonlinear transformation g is define as follows.

$$x = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \underset{g^{-1}}{\overset{g}{\rightleftarrows}} \begin{bmatrix} u_N \\ v_N \end{bmatrix} = z \tag{4.5}$$

where $u_0 := (x_1, ..., x_{[d/2]})^T, v_0 := (x_{[d/2]+1}, ..., x_d)^T$ and $u_N := (z_1, ..., z_{[d/2]})^T$, $v_N := (z_{[d/2]+1}, ..., z_d)^T$.

4.3 Overview of ARGate Architecture

Our ARGate architecture in [50] has the two most essential regularization techniques: *fusion weight regularization with auxiliary losses* and *monotonic fusion weight target learning*, as displayed in Fig. 4.1.

In terms of network structure, the ARGate architecture is composed of two networks : a main model and an auxiliary model. The fusion weights are generated from mild blue box in Fig. 4.1, where fusion happens with pre-procesed features by using a fully connected (FC) layer. Then, the fusion weights are multiplied with each FC layer for input features and blended with the multiplied values for classification/regression.

An auxiliary (aux) model assists the main model, which is deployed for inference, during train-

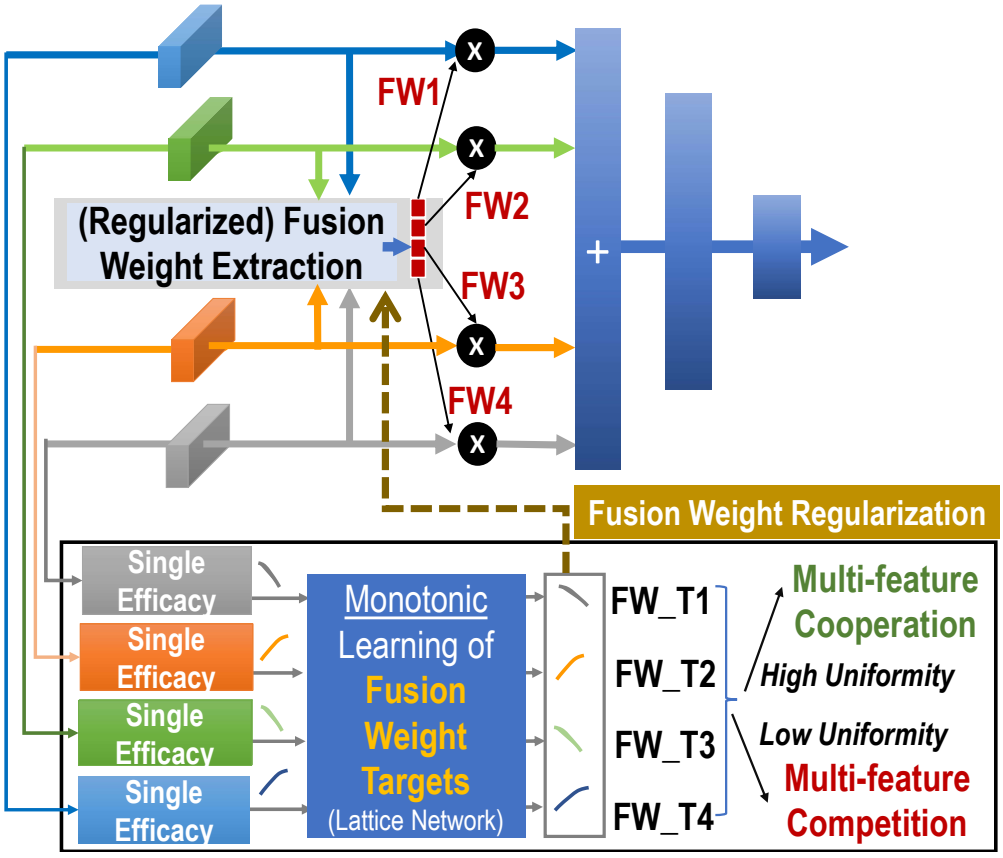

Figure 4.1: The ARGate architecture overview.

ing process. When partitioning output of the fully connected (FC) layer into the fusion weights, a sigmoid and a softmax function is adopted to normalize the fusion weights between [0,1] for the feature importance interpretability. So, the importance of input features are represented with magnitude of the fusion weights. Furthermore, the fusion weights are regularized with losses of different auxiliary paths, which reflects the relevance of inputs and can be used for *fusion weight regularization* (FWR). When a particular input is corrupted under noise, the high loss of its auxiliary path will constrain the training of the fusion layer in the main model to produce a low fusion weight for that input feature.

For fusion weight target learning, the lattice network is adopted in our auxiliary paths. Since the fusion weight target of the auxiliary paths shall be a decreasing function of the corresponding auxiliary path loss, there is a decreasing monotonicity between the fusion weight targets and the auxiliary losses.

With these two regularization techniques above, we customize loss function for training phase as follows:

$$\begin{aligned} Loss = \quad & Loss_{main} + \alpha \cdot \sum_{k=1}^{K} w_{fusion}^{k} Loss_{aux}^{k} \\ & + \beta \cdot \sum_{k=1}^{K} \left(w_{fusion}^{k} - f_k(\vec{Loss}_{aux}) \right)^2 . \end{aligned} \tag{4.6}$$

4.4 Proposed Rev-ARGate based Bayesian Optimization

To tackle the current BO based algorithms for the failure detection, we propose a Rev-ARgate based BO, which effectively reduce the dimension through the bijective characteristic of the RevNet.

4.4.1 Restoration of the input x with zero values

Let assume that original input x to the RevNet has D-dimensional parameter space i.e. $x \in \mathbb{R}^d$. After a forward computation of the RevNet, output z has the same dimension $z \in \mathbb{R}^d$ as the input. Therefore, through the RevNet which satisfy the equation 4.3, active and non-active components

of z is extracted. Then, the output z from the RevNet pass through the ARGate, which is utilized for robust fusion with the regularized fusion weights. From the fusion weights in the ARGate architecture, active and non-active components of z can be found as (z_{act}, z_{inact}), which is also considered as important and non-important features for the target. Based on the (z_{act}, z_{inact}), since z_{inact} is not important for the target, all z_{inact} values are replace with 0's. The overview of the Rev-ARGate architecture with BO using the zero scheme is presented in Fig. 4.2.

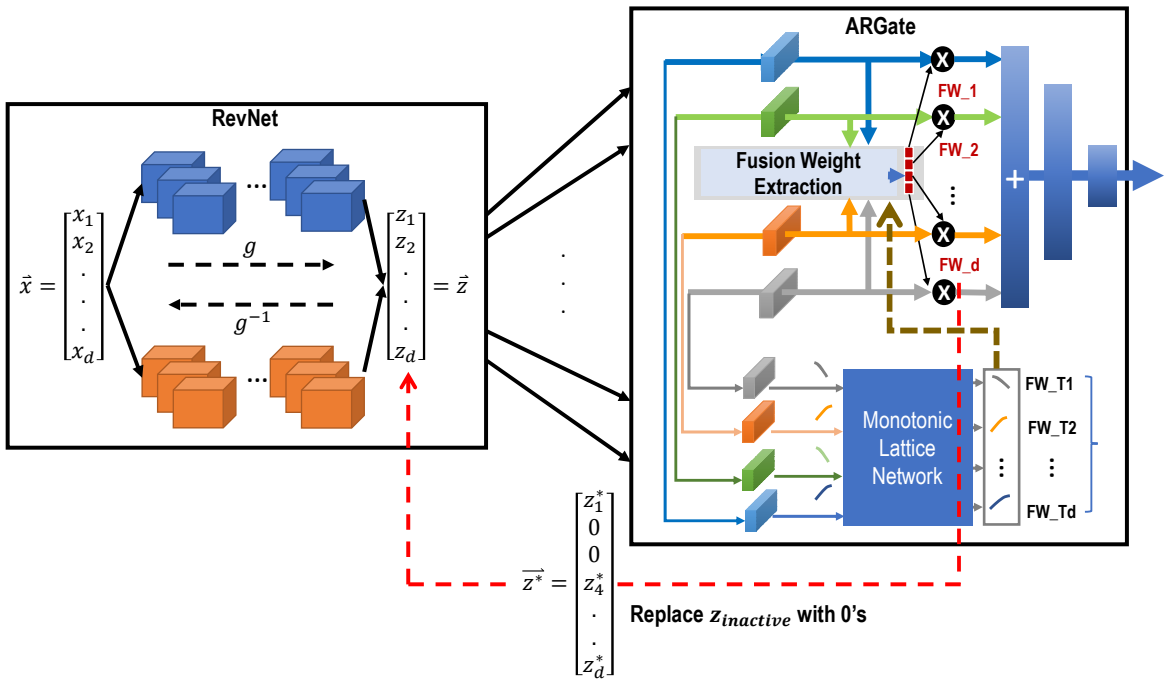

Figure 4.2: The Rev-ARGate architecture with the restoration with 0's.

4.4.2 Restoration of the input x with Bayesian Neural Network

While the restoration of the input x from the output z through appending zero values shows some improvements for inference in the RevNet, we are focusing on the probability distribution between z_{act} and z_{inact}. In a neural network (NN), weights in the neural network are determined through training process and training data X. After training phase, the trained weight values are fixed and the prediction y is deterministic, which means that the training data is highly trusted and then the most suitable weights are chosen for inference. On the other hand, the weight distribution of Bayesian neural network (BNN) are determined with the training process, not like the NN. Based on the BNN, we have trained the relationship between z_{act} and z_{inact} to get the distribution between these pre-processed features from the RevNet. After the training, the fixed BNN is applied for z_{inact} estimation and the estimated values are replaced in the z vector for the input restoration through the backward computation of the RevNet as shown in Fig. 4.3.

4.5 Experimental Results

4.5.1 Experimental Setups

We demonstrate our proposed Rev-ARGate architecture with BO approach with two circuits : a dc-dc converter [55] (44 dimensions) and a low-dropout regulator [56] (60 dimensions), as shown in Fig. 4.5 and 4.6.

The proposed Rev-ARGate is implemented with Pytorch 1.2 [57]. To be specific with the training process, first, the Rev-ARGate is trained with the simulation circuit data. After the training, fusion weight values were examined and top-N important features z_{act} are extracted. Second, with the important features z_{act}, we train a simple FC-layer based network for the performance verification i.e. if the simple FC-layer with z_{act} inputs show the same or similar performance with the trained Rev-ARGate, then z_{act} shows its importance for the target. Third, with the trained RevNet and the simple FC-layer network, we run BO so that the z^* vector is computed with backward process on the RevNet and x^* is generated. The generated x^* is used as input sample for the trained FC-layer network and the FC network bring the output y^* to pass this output to the BO aquisition

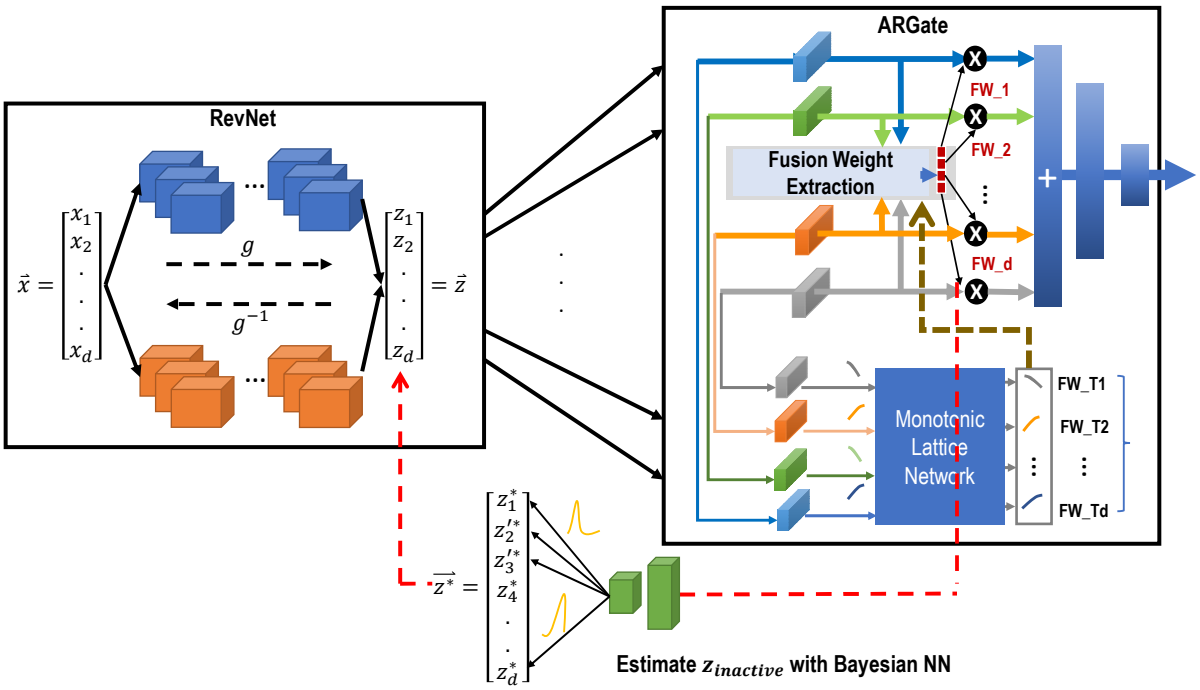

Figure 4.3: The Rev-ARGate architecture with the restoration through Bayesian NN.

function for producing z^*. This whole simulation process is working as a loop. The overall process of Rev-ARGate with BO is shown in Fig. 4.4.

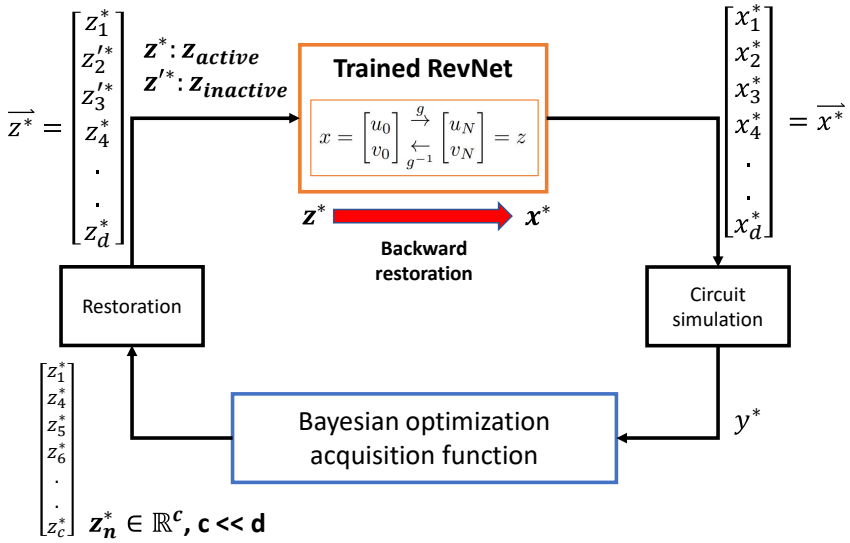

Figure 4.4: The overall Rev-ARGate with Bayesian optimization.

For the rare failure detection performance comparison, our proposed architectures are examined with pBO in [58] and parallelizable Bayesian optimization with random embedding (HDBO) in [43]. The BayesOpt [59] is utilized for implementing BO methods. All the simulations were run on a workstation with a 3.70GHz AMD Ryzen 2700x CPU.

4.5.2 Low-dropout Regulator

Three specifications, quiescent current, undershoot and load regulation, are chosen as verification targets of the LDO regulator. As shown in Fig. 4.6, three kinds of transistor-level variations are considered for all 20 transistors: channel length, threshold voltage and gate oxide thickness. Total 60 features are used as inputs for the simulation. All experiments of Bayesian optimization use 50 samples for the first gaussian process (GP) model training and total 300 examples for the sequential experiment. From [58], we verify the dimension reduction performance of our proposed Rev-ARGate architecture. Since HDBO can discover the number of effective features during the simulation, we compare the number of active features between HDBO and the proposed algorithm. The number of active features from the proposed architecture is 22 out of 60, which is roughly one third of the number of input features and the same as the result from HDBO. Furthermore, in the light of circuit aspects, the Rev-ARGate identify the actual important feature from the inputs while HDBO cannot. We observe that most important parameters are located on the output stage in the LDO regulator, which is similar to the circuit designer's view.

4.5.3 DC-DC converter

In Fig. 4.5, total 22 transistors are included in the DC-DC converter with two characteristics: channel length and width, resulting in 44 input features for the simulation. Two specifications are considered: output accuracy and overshoot. All other simulation settings are the same used in the LDO regulator simulation. Through our proposed Rev-ARGate, the number of input dimension of the converter into 18 from 44, which is less than half of the total number of input features.

4.5.4 Failure Detection Results

Based on the effective number of dimension discussed above, 22 is picked for the LDO regulator and 18 for the DC-DC converter. With the effective parameters, similar regression performance is achieved from the proposed Rev-ARGate architecture.

From Table 4.1 and 4.2, the parallelizable Bayesian optimization (pBO) method cannot detect a failure case due to the challenging high-dimensional parameter space in the LDO regulator. On

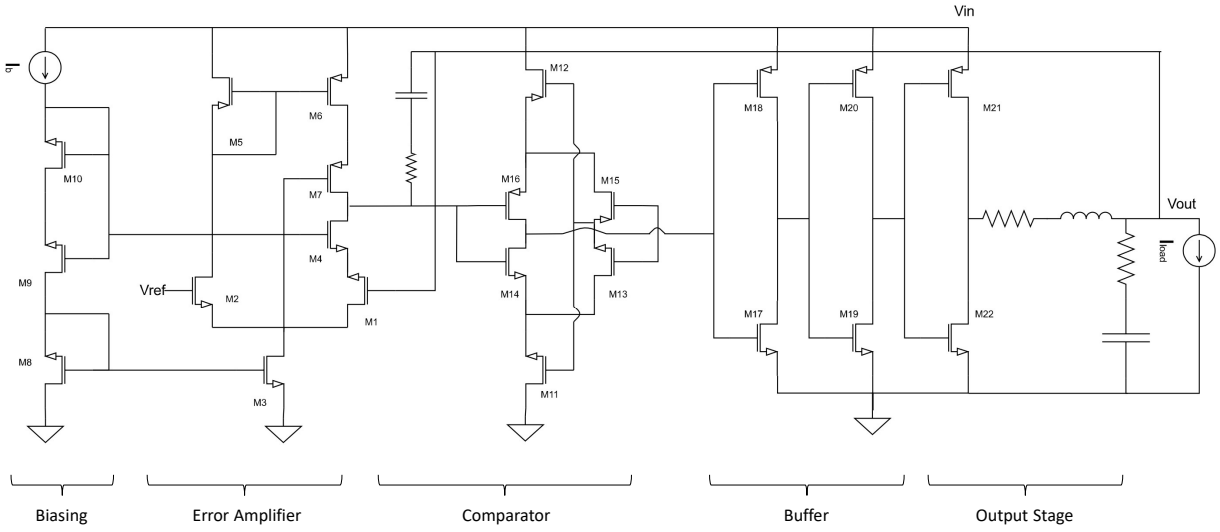

Figure 4.5: A pwm/pfm dc-dc converter.

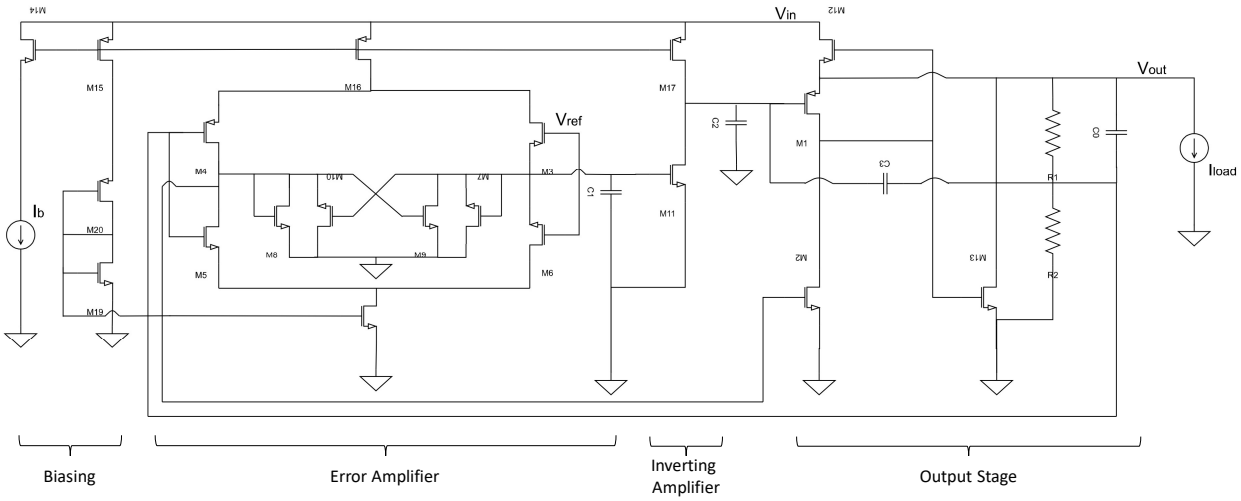

Figure 4.6: A low-dropout regulator.

Table 4.1: Failure detection result comparison for the LDO regulator (60 dimension).

Spec	Target	Method	Worst Case	1st Failure Hit
Quiescent Current	10.5mA	pBO	9.4mA	-
		HDBO	10.35mA	92
		Rev-ARGate with 0's	11.0mA	298
		Rev-ARGate with BNN	10.5mA	206
Undershoot	0.35V	pBO	0.31V	-
		HDBO	0.36V	221
		Rev-ARGate with 0's	0.36V	262
		Rev-ARGate with BNN	0.37V	242
Load regulation	0.27%	pBO	26.1%	-
		HDBO	29.1%	287
		Rev-ARGate with 0's	27.2%	191
		Rev-ARGate with BNN	30.6%	261

the other hand, our proposed failure detection approaches the Rev-ARGate based BOs successfully find the worst case for all specifications. The Rev-ARGate with the zero restoration scheme takes only 191 epochs for load regulation specification in the LDO regulator. In terms of the magnitude of the failure case, the proposed architecture demonstrate much worse case than the given target, while the baseline sampling methods are failed to find the failure.

In terms of the worst case trend demonstrated in Fig. 4.7, the Rev-ARGate architecture with BNN begins from lower point than HDBO. While HDBO shows higher worst case in the first 50 epochs, the improvement from starting point is smaller than the Rev-ARGate. On the other hand, in the Rev-ARGate architecture shows that the worst case of failure keeps increasing and better performance on the rare failure detection than HDBO. As another example of this trend in Fig. 4.8, HDBO stuck at the first worst case while our proposed algorithm keeps finding the worst case during the simulation.

4.6 Conclusion

In this work, we present a RevNet based gating architecture with Bayesian optimization for rare failure detection of analog mixed-signal circuits. The RevNet is adopted for the dimensionality reduction and we utilized Bayesian neural network for estimation of inactive parameters for

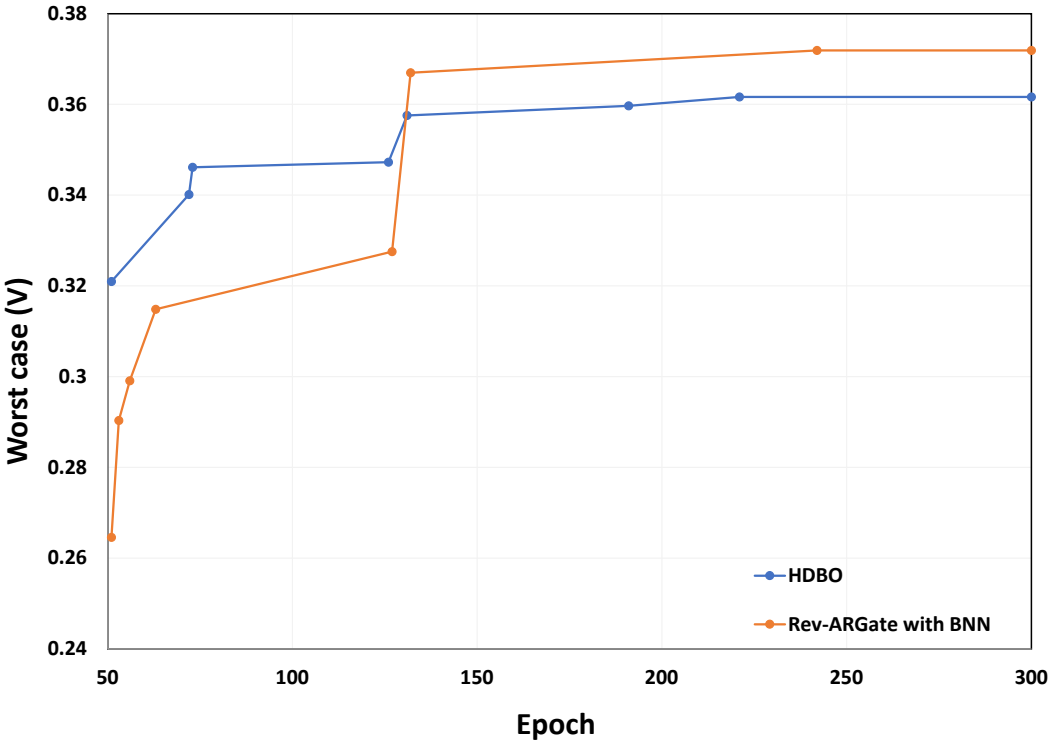

Figure 4.7: The worst case trend versus epoch of the LDO regulator with undershoot specification.

Table 4.2: Failure detection result comparison for the DC-DC converter (44 dimension).

Spec	Target	Method	Worst Case	1st Failure Hit
Output accuracy	38mV	pBO	34.25mV	-
		HDBO	35.61mV	-
		Rev-ARGate with 0's	38.68mV	112
		Rev-ARGate with BNN	38.99mV	112
Overshoot	11.5mV	pBO	10.31mV	-
		HDBO	11.11mV	-
		Rev-ARGate with 0's	9.5mV	-
		Rev-ARGate with BNN	11.95mV	210

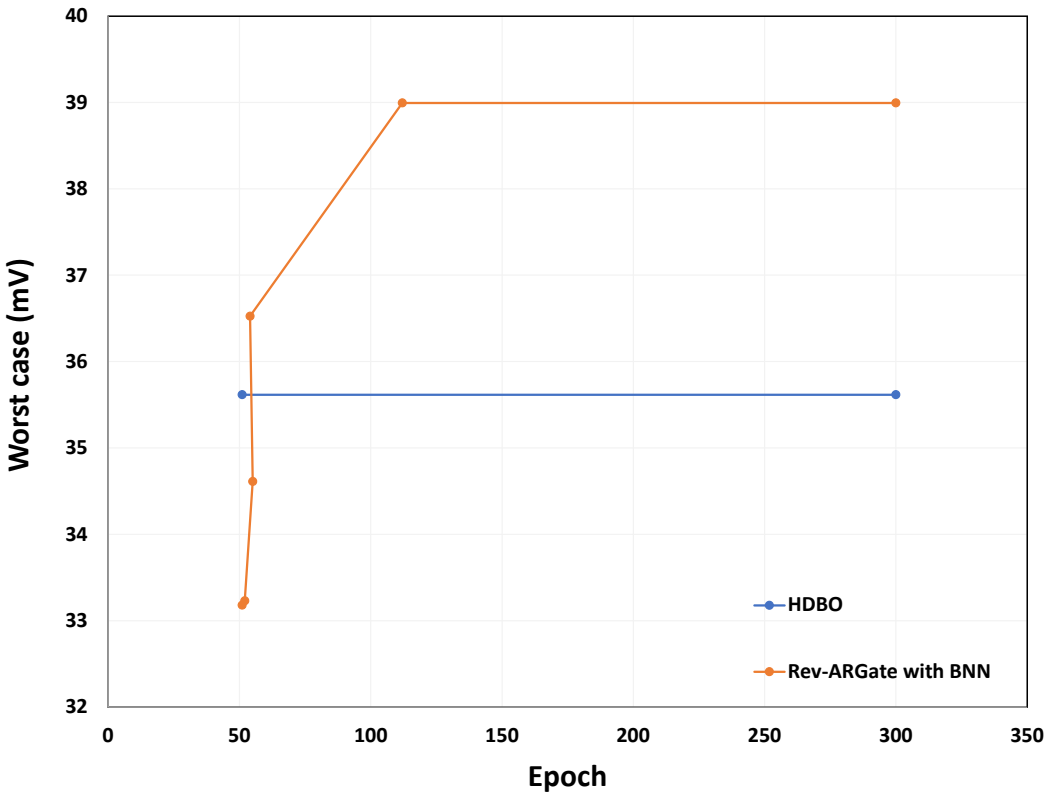

Figure 4.8: The worst case trend versus epoch of the DC-DC converter with output accuracy specification.

input restoration in the original parameter space. Our experimental results show that our proposed algorithm detects rare failure cases in high dimensional space, while Bayesian optimization with traditional and improved acquisition function does not find anomaly during the simulation.

5. SUMMARY AND CONCLUSIONS

In this dissertation, machine learning based learning architectures and algorithms for collision avoidance and sensor fusion are presented.

First, bio-inspied learning algorithm is demonstrated for autonomous vehicle collision avoidance in a mobile robot as test bed. Two techniques for a spiking neural network based reinforcement learning for mobile robot navigation are proposed: a multiplicative RM-STDP (M-RM-STDP) scheme and more biological feed-forward spiking neural network architecture with fine-grained rewards. It has been shown that the proposed techniques significantly outperform Q-learning and a baseline SNN approach. Especially, combining the two proposed techniques leads to a fairly robust solution with significantly improved success rates and quality of navigation trajectories measured by the number of steering movements.

Second, robust fusion architectures ARGate+ and ARGate-L for sensor fusion in a car are proposed. The ARGate architectures are proving its ability for resilient sensor fusion by addressing the limitations of the conventional fusion schemes including the existing gating architectures. The normalized fusion weights give interpretability for feature importance. By leveraging the two regularization techniques, fusion weight regularization with auxiliary losses weighting, and monotonic fusion target learning, our proposed gating architectures incorporate an auxiliary model to regularize the main model to robustly learn the fusion weight for each modality. Our architectures have demonstrated significant performance improvements over other models particularly in the presence of sensor failures.

Last, in terms of data efficient learning with less amount of data, reversible gating architecture for rare failure detection problem is displayed through Bayesian optimization. The proposed Rev-ARGate is demonstrated for its effective dimensionality reduction targeting rare failure detection of analog mixed-signal circuits. The RevNet is deployed for the dimensionality reduction and we utilized our proposed zero-replacement scheme or Bayesian neural network for estimation of inactive parameters for input restoration in the original parameter space. Our experimental results

show that the proposed algorithm detects rare failure cases in the high dimensional space, while Bayesian optimization techniques with traditional and improved acquisition function does not.

For future research problem, extension of the SNN with auxiliary paths for sensor fusion is good approach. Since the SNN is not well studied for the sensor fusion application, the auxiliary paths with various regularization scheme could be deeply explored.

Furthermore, reversible ARGate can be another interesting topic for the sensor fusion. Due to the fact that loss of information is avoided through the RevNet, the reversible ARGate which is composed of reversible convolutional layers can extend the prediction performance further to its limit. Also, this reversible layer is beneficial for performance of the auxiliary paths, which improves the main model in the ARGate and overall performance with better regularized fusion weights.

Utilizing the BNN in the ARGate for probability distribution of the fusion weights is also good area for future research.

REFERENCES

[1] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2633–2642, 2020.

[2] L. Garcia-Alonso, C. H. Holland, M. M. Ibrahim, D. Turei, and J. Saez-Rodriguez, "Benchmark and integration of resources for the estimation of human transcription factor activities," *Genome research*, vol. 29, no. 8, pp. 1363–1375, 2019.

[3] J. Gorzelany, "The safest cars and crossovers for 2016," *Forbes*, 2016.

[4] T. Dierks, B. T. Thumati, and S. Jagannathan, "Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence," *Neural Networks*, vol. 22, no. 5, pp. 851–860, 2009.

[5] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[6] B.-Q. Huang, G.-Y. Cao, and M. Guo, "Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 1, pp. 85–89, IEEE, 2005.

[7] O. Azouaoui, A. Cherifi, R. Bensalem, A. Farah, and K. Achour, "Reinforcement learning-based group navigation approach for multiple autonomous robotic systems," *Advanced Robotics*, vol. 20, no. 5, pp. 519–542, 2006.

[8] J. Qiao, Z. Hou, and X. Ruan, "Application of reinforcement learning based on neural network to dynamic obstacle avoidance," in *Information and Automation, 2008. ICIA 2008. International Conference on*, pp. 784–788, IEEE, 2008.

[9] K. Macek, I. PetroviC, and N. Peric, "A reinforcement learning approach to obstacle avoidance of mobile robots," in *Advanced Motion Control, 2002. 7th International Workshop on*,

pp. 462–466, IEEE, 2002.

[10] E. Nichols, L. J. McDaid, and N. Siddique, "Biologically inspired snn for robot control," *IEEE transactions on cybernetics*, vol. 43, no. 1, pp. 115–128, 2013.

[11] L. I. Helgadóttir, J. Haenicke, T. Landgraf, R. Rojas, and M. P. Nawrot, "Conditioned behavior in a robot controlled by a spiking neural network," in *Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on*, pp. 891–894, IEEE, 2013.

[12] R. V. Florian, "Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity," *Neural Computation*, vol. 19, no. 6, pp. 1468–1502, 2007.

[13] E. M. Izhikevich, "Solving the distal reward problem through linkage of stdp and dopamine signaling," *Cerebral cortex*, vol. 17, no. 10, pp. 2443–2452, 2007.

[14] R. Legenstein, D. Pecevski, and W. Maass, "A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback," *PLoS Comput Biol*, vol. 4, no. 10, p. e1000180, 2008.

[15] Z. Cao, L. Cheng, C. Zhou, N. Gu, X. Wang, and M. Tan, "Spiking neural network-based target tracking control for autonomous mobile robots," *Neural Computing and Applications*, vol. 26, no. 8, pp. 1839–1847, 2015.

[16] P. Arena, L. Fortuna, M. Frasca, and L. Patané, "Learning anticipation via spiking networks: application to navigation control," *IEEE transactions on neural networks*, vol. 20, no. 2, pp. 202–216, 2009.

[17] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.

[18] E. M. Izhikevich *et al.*, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[19] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[20] J. Sjöström and W. Gerstner, "Spike-timing dependent plasticity," *Spike-timing dependent plasticity*, p. 35, 2010.

[21] M. C. Van Rossum, G. Q. Bi, and G. G. Turrigiano, "Stable hebbian learning from spike timing-dependent plasticity," *The Journal of Neuroscience*, vol. 20, no. 23, pp. 8812–8821, 2000.

[22] Y. Luz and M. Shamir, "Balancing feed-forward excitation and inhibition via hebbian inhibitory synaptic plasticity," *PLoS Comput Biol*, vol. 8, no. 1, p. e1002334, 2012.

[23] R. Evans, "Reinforcement learning in a neurally controlled robot using dopamine modulated stdp," *arXiv preprint arXiv:1502.06096*, 2015.

[24] D. F. Goodman and R. Brette, "The brian simulator," *Frontiers in neuroscience*, vol. 3, p. 26, 2009.

[25] Ramachandram *et al.*, "Deep multimodal learning: A survey on recent advances and trends," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 96–108, 2017.

[26] Ku *et al.*, "Joint 3d proposal generation and object detection from view aggregation," *arXiv preprint arXiv:1712.02294*, 2017.

[27] Chen *et al.*, "Multi-view 3d object detection network for autonomous driving," in *IEEE CVPR*, 2017.

[28] Wei *et al.*, "Lidar and camera detection fusion in a real-time industrial multi-sensor collision avoidance system," *Electronics*, vol. 7, no. 6, p. 84, 2018.

[29] Patel *et al.*, "Sensor modality fusion with cnns for ugv autonomous driving in indoor environments," in *IROS, IEEE*, 2017.

[30] Mees *et al.*, "Choosing smartly: Adaptive multimodal fusion for object detection in changing environments," in *IEEE IROS*, pp. 151–156, IEEE, 2016.

[31] W. Zhang, H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy, "Robust multi-modality multi-object tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2365–2374, 2019.

[32] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018.

[33] P. Gao, H. You, Z. Zhang, X. Wang, and H. Li, "Multi-modality latent interaction network for visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5825–5835, 2019.

[34] Anguita *et al.*, "A public domain dataset for human activity recognition using smartphones.," in *ESANN*, 2013.

[35] Kwak *et al.*, "Know your master: Driver profiling-based anti-theft method," in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pp. 211–218, IEEE, 2016.

[36] A. Geiger *et al.*, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE CVPR*, 2012.

[37] L. Yu, C. Cuppini, J. Xu, B. A. Rowland, and B. E. Stein, "Cross-modal competition: The default computation for multisensory processing," *Journal of Neuroscience*, vol. 39, no. 8, pp. 1374–1385, 2019.

[38] B. E. Stein *et al.*, "Development of multisensory integration from the perspective of the individual neuron," *Nature reviews. Neuroscience*, vol. 15, no. 8, pp. 520–535, 2014.

[39] S. You *et al.*, "Deep lattice networks and partial monotonic functions," in *NIPS*, 2017.

[40] Paszke *et al.*, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[41] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[42] D. D. Weller, M. Hefenbrock, M. S. Golanbari, M. Beigl, and M. B. Tahoori, "Bayesian optimized importance sampling for high sigma failure rate estimation," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1667–1672, 2019.

[43] H. Hu, P. Li, and J. Z. Huang, "Enabling high-dimensional bayesian optimization for efficient failure detection of analog and mixed-signal circuits," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2019.

[44] X. Shi, H. Yan, J. Wang, X. Xu, F. Liu, L. Shi, and L. He, "Adaptive clustering and sampling for high-dimensional and multi-failure-region sram yield analysis," in *Proceedings of the 2019 International Symposium on Physical Design*, pp. 139–146, 2019.

[45] G. Zhang, J. Zhang, and J. Hinkle, "Learning nonlinear level sets for dimensionality reduction in function approximation," in *Advances in Neural Information Processing Systems*, pp. 13220–13229, 2019.

[46] Z. Gao, J. Tao, Y. Su, D. Zhou, X. Zeng, and X. Li, "Efficient rare failure analysis over multiple corners via correlated bayesian inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2029–2041, 2020.

[47] M. Hefenbrock, D. D. Weller, M. Beigl, and M. B. Tahoori, "Fast and accurate high-sigma failure rate estimation through extended bayesian optimized importance sampling," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 103–108, 2020.

[48] J.-H. Jacobsen, A. Smeulders, and E. Oyallon, "i-revnet: Deep invertible networks," *arXiv preprint arXiv:1802.07088*, 2018.

[49] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, "The reversible residual network: Backpropagation without storing activations," in *Advances in neural information processing systems*, pp. 2214–2224, 2017.

[50] M. S. Shim, C. Zhao, Y. Li, X. Zhang, and P. Li, "Robust deep multi-modal sensor fusion using fusion weight regularization and target learning," *CoRR*, vol. abs/1901.10610, 2019.

[51] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, N. De Freitas, *et al.*, "Bayesian optimization in high dimensions via random embeddings.," in *IJCAI*, pp. 1778–1784, 2013.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[53] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham, "Reversible architectures for arbitrarily deep residual neural networks," *arXiv preprint arXiv:1709.03698*, 2017.

[54] E. Haber and L. Ruthotto, "Stable architectures for deep neural networks," *Inverse Problems*, vol. 34, no. 1, p. 014004, 2017.

[55] Y. Wang, P. Li, and S. Lai, "A unifying and robust method for efficient envelope-following simulation of pwm/pfm dc-dc converters," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 618–625, IEEE, 2014.

[56] S. Lai and P. Li, "A fully on-chip area-efficient cmos low-dropout regulator with fast load regulation," *Analog Integrated Circuits and Signal Processing*, vol. 72, no. 2, pp. 433–450, 2012.

[57] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[58] H. Hu, P. Li, and J. Z. Huang, "Parallelizable bayesian optimization for analog and mixed-signal rare failure detection with high coverage," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 1–8, 2018.

[59] R. Martinez-Cantin, “Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3735–3739, 2014.