AUXILIARY LOSS WEIGHTING FOR ROBUST MULTI-MODAL SENSOR FUSION WITH DEEP NEURAL NETWORKS

A Thesis

by

YANG LI

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

MASTER OF SCIENCE

May 2019

Major Subject: Computer Engineering

ABSTRACT

The major focus of this research is on sensor fusion. Sensor fusion means to combine multiple sensory input or data from different source such that the performance is better than the best performance would be when those different data were used individually. As we know, in the world of AI, sensors are really important. Traditionally, we treat these data as they are separated. Doing so may deliver good performance when the sensors are exempt from noise and malfunction problems. However, if sensor failure appears, the performance will drop. Sensor fusion is a solution for the above situation.

Recently, deep neural networks have been rigorously studied for sensor fusion applications such as autonomous driving and robot control. Among these studies, various gated neural network architectures were proposed, which have improved the existing classical convolutional neural networks (CNNs).

Several problems existed for those gated neural network architectures. In this research, some of them are described. Then, to solve those problems, a further optimized gated architecture, a gated CNN with auxiliary paths, was proposed. The major focus of this thesis work is on auxiliary loss weighting, a technique to further regulate the gated CNNs with auxiliary paths and improve their performance.

The CAD-60 dataset is utilized as a benchmark to demonstrate the significant performance improvements through the proposed architecture and its robustness in the presence of sensor noise and failures.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Peng Li, and my committee members, Dr. Dezhen Song, and Dr. Gwan S. Choi, for their guidance and support throughout the course of this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, thanks to my mother and father for their encouragement, patience, and love.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis (or) dissertation committee chaired by Professor Peng Li of the Department of Electrical Computer Engineering.

All works conducted for the thesis (or) dissertation was completed by the student independently.

Funding Sources

There are no outside funding contributions to acknowledge related to the research and compilation of this document.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

Nowadays, neural networks and deep learning have gained lots of attention. They can be applied to various fields such as computer vision, natural language processing, and data analytics. Among all of those applications, the application of neural networks in the field of computer vision boosted the development of computer vision significantly. Thanks to CNNs (convolutional neural networks) and several famous CNN architectures like VGG[1], ResNet[2], FCN[3] and so on, now we can extract features for input data more efficiently. The extracted features can be used to detect objects, segment data, analyze data, etc. In this research, I mainly used the CNN deep network.

A technology that using multi-sensor or multi-source data in deep learning is called sensor fusion in deep learning. It is an important technology for systems that are equipped with multiple sensors. For example, autonomous driving system is equipped with cameras, LIDARs and several other sensors. In addition, inertial measurement units (IMUs) and other sensors are utilized in devices such and smartwatches for activity recognition applications.

To achieve autonomous driving, object detection needs to be done first. There are two kinds of object detection:2D object detection and 3D object detection. For 2D object detection, we can deal with the camera and use image recognition technology. Architectures like R-CNN[4], Fast R-CNN[5], Faster R-CNN[6], Mask R-CNN[7] and YOLO[8] are several well recognized 2D object detection architectures that only use the image as the inputs. For 3D object detection, several methods are available. One of such techniques is to use multiple sensors and use sensor fusion, the other is to only use one input sensor's data. Both of the above methods have their well-recognized architecture. For sensor fusion 3D object detection, MV3D [9], AVOD [10] are representative works. PointRCNN[11] presents an approach for 3D object detection t only based on LIDAR sensors.

Specifically, in [9], several sensor fusion methods are studied and discussed under the background of autonomous driving. The dataset used in [9] is KITTI [12]. It contains two kinds of sensors, LIDAR and RGB images. By using this dataset, [9] discussed and compared three fusion methods: early fusion, late fusion, and deep fusion. More specifically, fusion is a way to get use of sensory inputs of multiple sensors. Sensory inputs are processed into features. We can get the extracted feature by using the neural network architecture. One of the most popular architectures in dealing with image data is the CNN architecture which is composed of convolution layers. Early fusion refers to the simple combinations of input features from the outputs of early convolutional layers and then uses the element- wise mean value as the combined features. After that, pass the combined features to the latter parts of the network. Late fusion represents the blending of pre-processed features generated by the networks. As for deep fusion which integrates the pre-processed features with element-wise mean operation in multi-stages, it is applied for joint input sensor fusion due to its flexibility. However, the way how each feature affects the training phase and classification performance is unclear. All the above methods, no matter 2D or 3D, are based on the presumption that all sensors work in good condition. However, things may change when one or more sensors malfunction.

When sensor is noisy or is failed, all the above methods will suffer from relatively big performance drop.

To alleviate the impact of sensor noise and failure [13] proposes a gated information fusion network, in which weight maps are extracted from each sensory input. Raw input sensors are processed with convolutional layers. Furthermore, outputs from the convolutional layers are processed with convolutional layers. Furthermore, outputs from the convolutional layers are fed into weight-extraction architecture to produce weight maps from each sensory input. Then, weight maps are multiplied with each feature map to perform a weighted sum. Note that the weight maps are derived from feature maps. However, the meaning of weight maps and the relationship with sensory input is not fully explored.

Several works have explored gating architecture. In [14], various multi-modal fusion work is reviewed in terms of architecture, and regularization. [15] provides time information fusion schemes in CNNs: single frame, late fusion, early fusion, fusion approaches. The difference and effect of each method are also discussed. In [16] adaptive multi-modal fusion techniques for object detection are proposed and analyzed. In [16], the gating scaler is mentioned and how gating scalars work between different sensors like RGB images and depth images is covered. Nevertheless, that gating scalars parts are not discussed in detail. What is more, only two channels of input are utilized and deep insight of gating scalars is missing. In [17], Patel et al. propose a gated convolutional neural network called NetGated architecture. In this architecture, fusion weight extracted from camera and LIDAR used to combine information in a convolutional neural network (CNN). Compared to traditional CNNs, the gated network (NetGated) is shown to be robust to sensor failures. However, a deep understanding of the relationship between sensory input, fusion weight, network architecture, and the resulting performances is not fully examined.

This research is to propose a further improved for the modified version of the existing baseline gated architecture proposed in [17] that proposed by my team member and investigate how the pre-proposed ARGate [18] by my team member can improve the performance. And how the new architecture works when facing different input situations. Generally speaking, tests for architecture are under different sensor failure setting.

In this research, I am mainly focused on the fusion method and using fusion to boost the robustness of the network. To specify, this work proposed the method called auxiliary loss weighting. This thesis contains several works. It contains 3D object detection by using a multi-sensor fusion method, and human activity detection. Both of them are implemented with sensor fusion architecture. The analyzes are mainly based on the experiment on human activity detection.

The contributions of this work are as follows:

- Proposed a new modified architecture based on ARGate [18] which can fuse all input sensors in a more reasonable way. The new method can alleviate the impact that caused by sensor noise and sensor failure.
- Proposed new loss function named auxiliary loss weighting which weight the loss of auxiliary paths and uses the regularized auxiliary path loss to

further regularize early fusion and deep fusion stage, leading to model optimization and performance improvements.

- The fusion scalers learned by the network is more meaningful. They are the indicator of the quality of the input sensor.
- Well analyzed the characteristics of the proposed architecture

The remainder of this thesis is organized as follows:

Chapter 2 provides some background on deep learning, neural network, object detection, human activity detection and in particular the sensor fusion. Chapter 3 presents problems that currently exist work has and the limitation of them. In this chapter, several improved architectures proposed by my team are also introduced. Chapter 4 presents the solution proposed in this work, the auxiliary loss weighting for robust multi-modal sensor fusion. Chapter 5 shows the experiment results of this work. Those results are evidence that the proposed solution in this work has a better performance. Chapter 6 draws the conclusion and presents a summary of the entire research work.

CHAPTER II

BACKGROUND

This chapter presents the background information required for the contribution and all work in this research. It includes the background of Convolutional Neural Networks, Object Detection, Human Activity Detection, and Gated Architecture.

2.1 Convolutional Neural Networks

2.1.1 Neural Networks

The artificial neural network is inspired by the actual neural network. The key component of the artificial neural network is its connection weight. If we view the artificial neural network as an end-to-end black box, the main operation occurs in the black box. If we treat that black box as a function, we can name it as f, then the goal of the network is to approximate function f. The output is defined as $y,$ the input is defined as x, the parameters of the network defined as θ. By using the above defines we can represent the neural network as follows equation 2.1:

$$y = f(x; \theta) \tag{2.1}$$

That means using f to map input x to output y.

Two kinds of artificial neural networks are existing, the single layer artificial neural network and the multilayer artificial neural network. A single hidden layer neural is made by the input layers, the hidden layer and the output layer. The input layer is responsible gather all input data. The hidden layer is the key point of the single hidden layer neural network. It is responsible for taking the calculations. Each hidden unit use the inputs from the input layer as its input. For each hidden unit, a random initial value is assigned. The computation process is as follow: the value of the hidden unit is multiplied with the inputs and then the results are added to a random noise. After that, the activation function comes. Some activation function like Relu and Sigmoid is applied on the outputs of the hidden layers. The final layer is the output layer. It takes all the previous layer's result as input and multiplies and adds their outputs to initially random values. Then these data get activated by a Sigmoid function. The outputs of the Sigmoid function are numbers between zero and one. Therefore, we can treat the output as the probability that the correspond action could be. Basically, we can increase the accuracy by adding more hidden units. The architecture of a single layer neural network is shown in Figure 1.

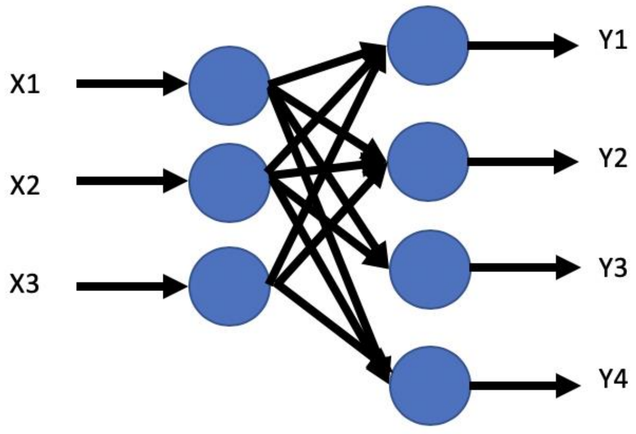

Figure 1: The Single Layer Neural Network

For Multilayer Artificial Neural Network, it means the hidden layer is more than one. Figure 2 shows the architecture of it.

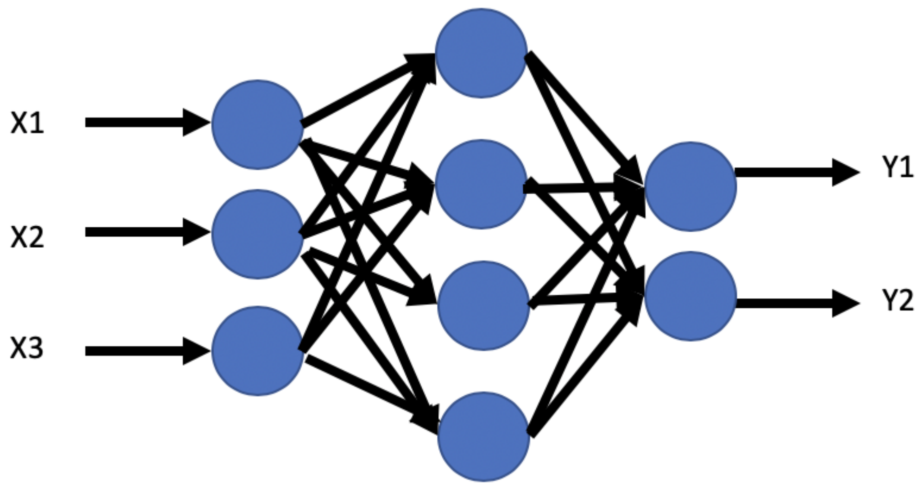

Figure 2: The Multi-Layer Neural Network

2.1.2 Convolutional Networks

Convolutional networks are a kind of neural network designed to process data that has a known grid-like topology[19]. Since the convolution operation focuses more on the adjacent pixel on the input data it has been really successful in image processing. Besides the convolution operation, the convolution neural network is exactly the same as the artificial neural network.

The Convolution Operation

Convolution is a mathematical operation on two functions f and g to produce a third function. Convolution is defined as an integral that expresses the amount of overlap of g as it is shifted over f. It can be described as fellow's equation 2.2:

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau. \tag{2.2}$$

Convolution Networks

The ConvNet is made by several Conv layers and each of those Conv layers has its own activation function. Besides the Conv layers, the pooling layer and the fully-connected layer are also frequently used in a ConvNet. By combining those layers together, we can get a CoveNet.

The convolution layer takes the original parts of the input to do computation at a time. That computation generates one output scalar. Lots of this kind of computation are done during the convolution process. After that, the outputs need to be activated by a activation function. This process is shown in Figure 3. Next, each linear activation is run through a nonlinear activation function. In the next stage, the pooling layer performs a down-sampling operation along the spatial dimensions (width, height). Hence it is common to periodically insert a pooling layer before the successive convolution layer. Finally, the FC layer computes the class score.

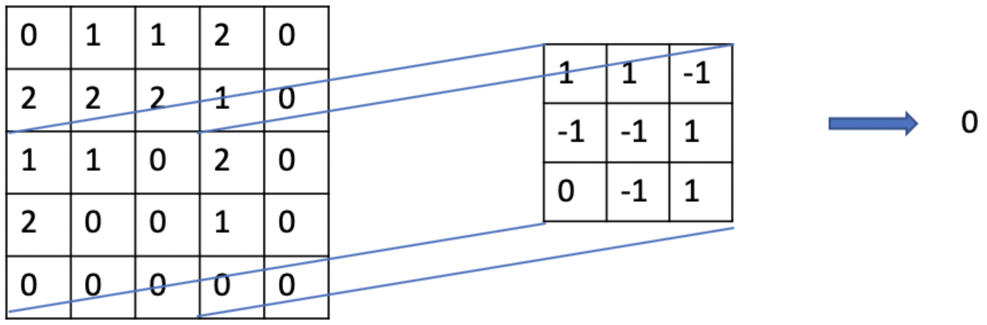

Figure 3. The convolution processes

The spatial arrangement of Conv layer output volume is controlled by three hyperparameters: the depth, stride, and zero-padding. The depth corresponds to the number of filters the network uses to look for different features in the input. Stride specifies how to slide the filter. When the stride is 1, the filter is moved one pixel at a time, and when the stride is 2, the filter jumps two pixels, and so on. This, in turn, will produce a smaller output volume spatially. Padding is used to pad the input volume with zeros, which enables the network to control the spatial size of the output volumes.

2.2 Human Activity Detection

Most human-activity recognition (HAR) research has focused on the recognition of relatively simple activities (e.g., sitting or walking) rather than more complex activities. Very early work in the field using data collected from different sensors placed all around the human body. These data were easy to analyze but inconvenient to collect. Then, video recordings soon became a significant method of HAR research. Scientists have used camera recordings to

recognize different behaviors e.g., hand-gestures. In this research, I mainly used the CAD-60 dataset and my partner mainly focused on the HAR dataset. The detail introduction of the above dataset is presented in section 4.3.

2.3 Gated Architecture

A way to do sensor fusion based on neural network was proposed in [17]. The proposed architecture is a kind of gated architecture, which refers to a gate in the network that can control the data flow. To specify, there are several fusion weight scalars in the network. The scalars participate in the process of fusion. Each feature has its own scalar and then do weighted fusion according to that scalar. The process described above is called weight fusion based on weight scalar and in this research, the architecture that can achieve this is called gating architecture.

CHAPTER III

PROBLEM

3.1 Baseline NetGated Architecture

In the previous research of our team, we use [17] as our baseline NetGate architecture. This NetGate architecture is proposed for unmanned ground vehicle (UGV) control. Input sensors of this application are LIDAR image and camera image. The diagram of this architecture is shown in Figure 4.

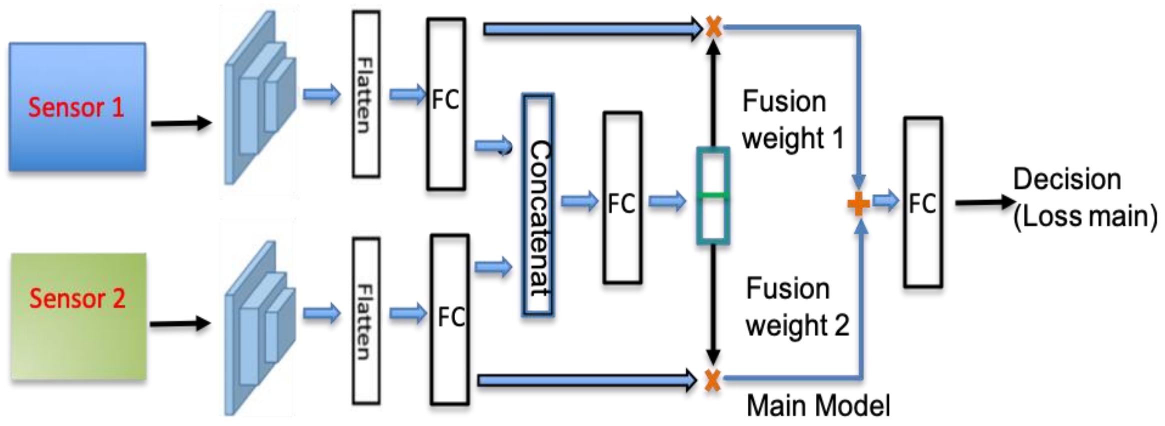

Figure 4 The NetGated CNN architecture [17]

We can see in this architecture, two inputs are processed independently at the first stage and after that, the features are combined together and passed to the later parts for further processing. To extend it to multi-input we have Figure 5. To explain in more detail, the data from three sensors are processed independently. First, they go through convolutional (Conv) layers, pooling layers, and finally fully connected (FC)layers. After that, each sensor's feature maps are generated. Outputs of the FC layers, from “FC-f1” to"FC-f3" in the first dashed box in Fig. 1, are the generated feature maps. After that, these feature maps are concatenated together and then fused by another FC layer

“FC-con”, where three feature-level fusion weight(scalars) is created. Note that each fusion weight is a scalar value, which is the outputs of the “FC-con” layer. At this time, we can use the weight scalars we get to do the further process. The outputs of the feature-level FC layers are multiplied with the corresponding fusion weight. In order to provide a visual aid, the data flow is demonstrated by arrows in Figure 5. Finally, these weighted feature outputs are fused by the last FC layer "FC-out" which produces the final prediction.

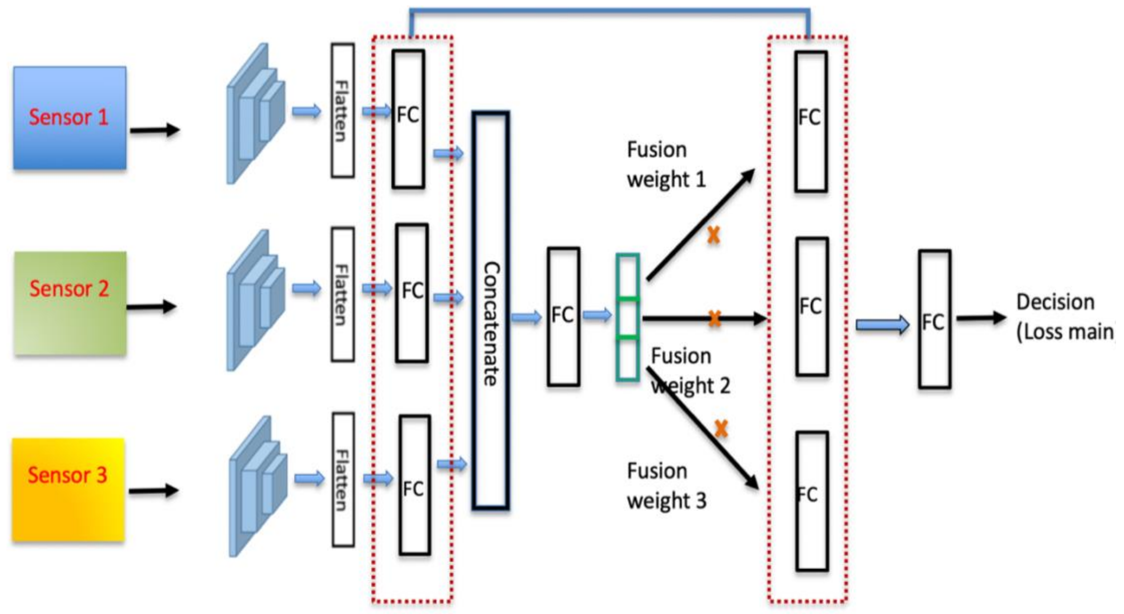

Figure 5: NetGated CNN proposed in[17]. Extend to three sensors

Here it is necessary to point out that the so-called weight scalar or say the extracted feature-level weight may be working as a controller for each input features. They act as a gate. If the scalar value is zero then the gate will be closed which means this input’s feature is turned off. Ideally, when an input sensor is noisy or malfunctioning, that sensor’s input will be closed. Furthermore, we can say that weight scalar can be viewed as an indicator of the quality of the input sensor. Therefore, these gated scalars may represent the ranking of the input features. To some extent, baseline architecture may provide robust sensor fusion capability.

3.2 Limitations of the NetGated Architecture

Through the introduction above we know that the NetGated architecture offers a promising deep learning solution for sensor fusion application. However, it still has several limitations. Since that architecture is an end-to-end all the weight scalars are generated during the network optimization and there is no direct control or regularization to ensure that scalars can truly represent the quality of corresponding sensory inputs. Following section is about several limitations of the NetGated architecture.

Fusion Weight Inconsistency

The first limitation the NetGate architecture suffers is the fusion weight inconsistency problem.

Since our goal is to improve the robustness of the sensor fusion architecture, the architecture should be able to deal with the noise and the sensor failure situation. Therefore, for a well functional neural network architecture, it should have some mechanism to deal with the above situations. NetGate architecture declares that it can use the fusion weight scaler to fuse the data accordingly. However, the true meaning of fusion weight of the NetGate architecture is not explained in detail in [17]. That problem also applies to the function of the fusion weight.

What is more, through experiments we observed that the feature-level fusion weight may not reflect the feature ranking. The meaning of the generated scalar is too chaotic to tell the meaning of it. According to our experiment, there are some cases where the largest extracted fusion weight may correspond to the noise channel, causing fusion weight inconsistency. That means when a channel is noisy the fusion weight scalar of that channel may be relatively large compared with other channels' fusion weight scalars. Which is in contrast with our understanding. According to our understanding, when a channel is noisy it is apparently not an important channel then the corresponding fusion weight scalar should be relatively small comparing with other scalars. This inconsistency negatively affects training and overall prediction performance when it comes to noisy sensory input or corrupted features.

Let us assume we have two input features $f1, f2$ with feature-level fusion weight $fw1, fw2$ where these are extracted from "FC-con" layer based on shared input features. Also, simply assume that input $f1$ is the most important feature in the dataset. The corresponding fusion weight $fw1$ should technically be higher than $fw2$. Furthermore, when the input is noisy, the fusion weight $fw1$should be lower than $fw2$. However, the previously mentioned weight does not fully cast the quality of the input features which lead to non-optimized prediction performance.

Besides above, the inconsistency of the fusion weight scalar also makes the scaler generated by the NetGate architecture cannot represent the quality of the input sensors. Then the whole network will not be a robust network. For the reason that the scalers are not the representation of quality then, to some extent, the fusion according to that scalar can be treated as random fusion. If the fusion is random then we cannot expect the fused features to be a good representation of all the information we can extract from all input sensors' data we have.

Lack of Additional Fusion Mechanisms

Another limitation the NetGate architecture suffers is the lack of additional fusion mechanisms.

From Figure. 5 we can see that the NetGate architecture proposed using “FC-out” to fuse all weighted features together and then the loss function used in the NetGate architecture only treat the entire output as the regulation for optimization. In this way, the physical meaning of each channel’s features may be lost even though all input for “FC-out” layer is weighted features.

This raw input fusion mechanisms can be improved, potentially leading to additional performance enhancement. As we all know, all parameters in the neural network are controlled by the loss function. That is because we use gradient descent as our optimization methods. Furthermore, ADAM and other well-adopted optimization methods are also based on gradient descent. Therefore, the loss function is really important. What is more, if we can add some regulation to the network that can make each channel participated in the process of weight fusion scalar generation. Based on that we generated our first stage solution.

We have resolved the above limitations of the baseline NetGated architecture by gated architecture with auxiliary paths (ARGate-WS). The proposed architecture regularizes the convolution layers and fusion stage which optimizes the baseline NetGated model and improve prediction accuracy.

After the first stage improved we further find that the regulation does have a positive effect on the robustness and performance of the network. That leads to the second stage solution (ARGate-WS-FWR). In [20], we further improved the network's robustness by using the ARGate-WS-FWR.

3.3 Pre-proposed architecture

To alleviate the above problem, we proposed several new gated architectures and those are discussed in this section. Those works are the basement of this work's solution and are quite enlightening.

3.3.1 The Gated CNN with Auxiliary Paths on Selected Channels

Base on the idea of the auxiliary path discussed in [9], [18] came up with an idea that uses the auxiliary path as a competitor to force the main model to get better performance. Named Auxiliary Path on Selected Channels The architecture is shown in Figure 6[20].

We can see that in figure 6, besides the main model, auxiliary paths are also participated in the training process and combined with the main model to become the whole model. For the auxiliary path, it only contains clean sensors. That is the pre-request of using this model. After we know which sensor is clean, we can put it to the auxiliary path. Which is means, the auxiliary path only contains clean sensors. Under the sensor failure situation, the main model's input contains several noisy channels.

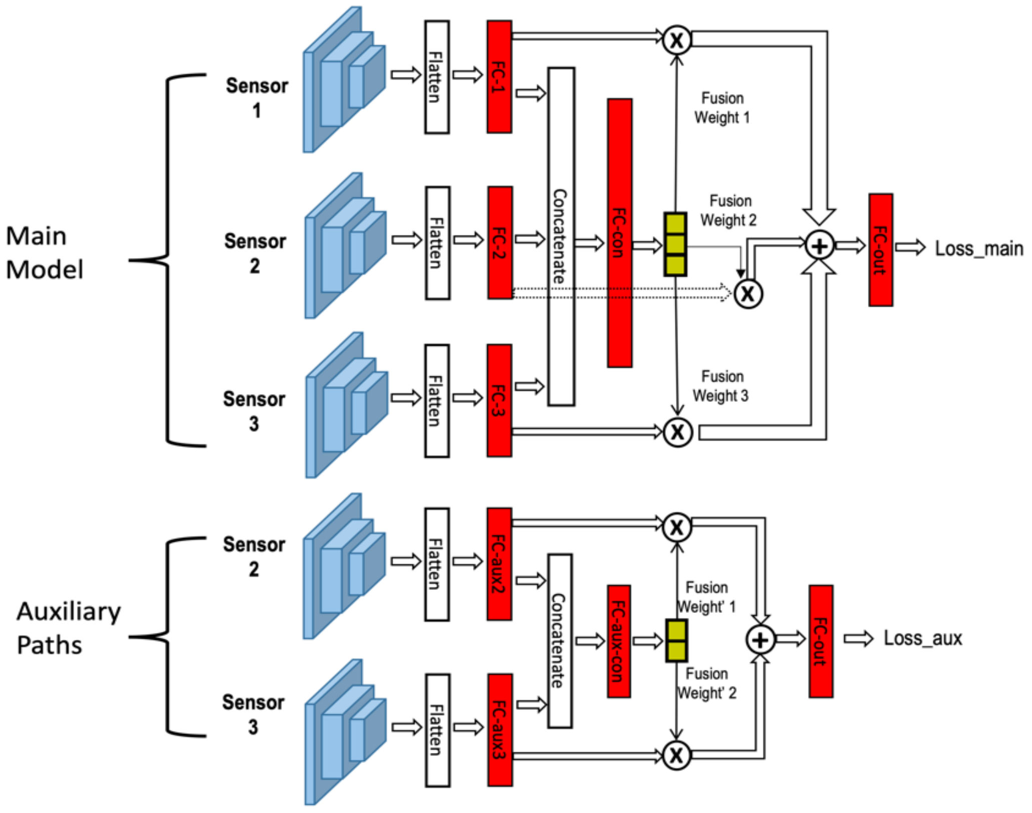

Figure 6: Auxiliary Path on Selected Channels[20]

However, auxiliary paths don't contain noisy input. Based on this truth, we know that if we fusion the sensor's input in the main model by using the element-wise mean fuse, then the noisy channel will affect the clean channel and the performance of the main model can be dropped a lot because of that. On the other side, since the auxiliary path only contains the clean channel, the performance of the auxiliary path is not affected by the noisy sensor. That leads to the idea that uses the auxiliary path as a competitor to regulate the performance of the main model. In this architecture, the performance of the main model should be at least as good as the auxiliary path. In order to achieve that goal, the fusion weight generated by the main model should be as reasonable as possible. This can be achieved by adopting the loss function as follows

$$\text{Loss}_{total} = \begin{cases} \alpha \cdot Loss_{main} + Loss_{aux}, & if\ Loss_{main} \geq Loss_{aux} \\ Loss_{main} + Loss_{aux}\ , & otherwise \end{cases} \tag{3.1}$$

3.3.2 The Gated CNN with Auxiliary Paths

In order to alleviate the limitation for the Auxiliary Paths on Selected Channels architecture. [18] proposed a gated CNN with auxiliary paths as regulation (ARGate-WS). The architecture is shown in Figure 7.

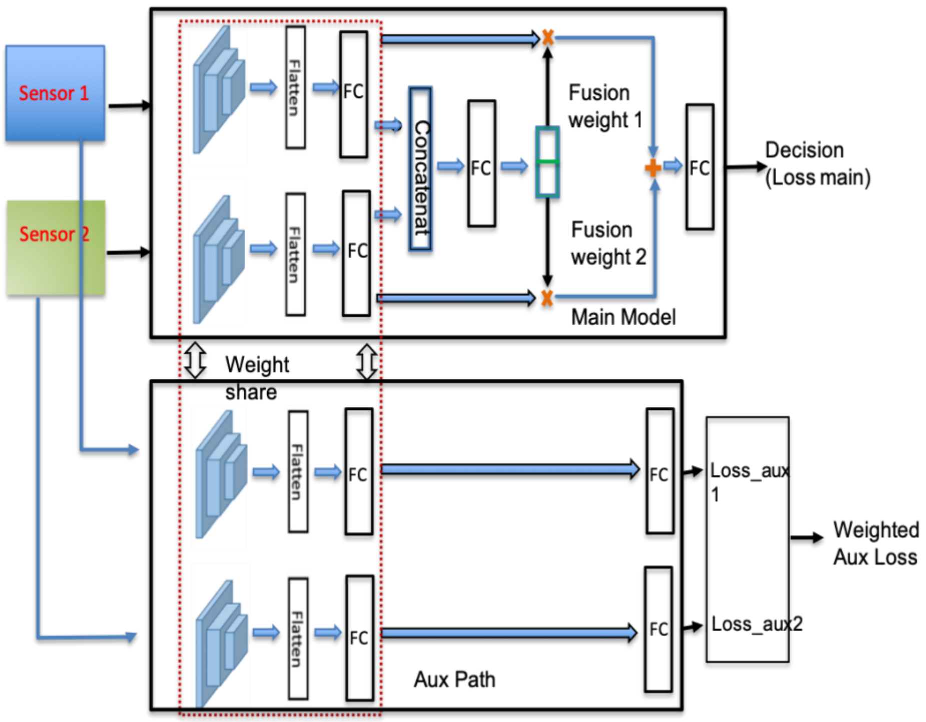

Figure 7: The gated architecture with auxiliary paths (ARGate-WS)

As we can see from Figure 7, the new architecture mainly contains two parts: the main gated model architecture and the auxiliary path for each input feature. The main gated model architecture is similar to the NetGate architecture proposed in[17]. For the auxiliary path, that is the architecture that each input will have its own feature extract network and then proceed the extracted feature to the later FC and prediction part. The feature extract network for each input is the same as the main model. We achieve that by sharing the weight between the main model and auxiliary path.

The loss functions for this architecture is

$$Loss_{final} = \alpha Loss_{main} + \beta Loss_{auxmin} \tag{3.2}$$

To explain it in more detail, we assume that there are three input features in Fig.7: $f1, f2, f3$. In the main gated model, all features are passed through each convolutional layer, pooling layer, and fully connected (FC) layer separately. Then, these three FC layers “FC-f1”, “FC-f2”, and “FC-f3" are concatenated into a FC layer called "FC-con". The output of the FC layer “FC-con” is split into three feature-level fusion weight, which are scalars for three input features individually. Note that when splitting the FC-con output into three fusion weight, L2 normalization and a SoftMax normalization of fusion weight. Then, the fusion weight is multiplied with the corresponding processed feature information “FC-f1”, “FC-f2”, and “FC-f3”. Weighted feature information is finally blended into the FC layer “FC-out”, which produces the classification.

In parallel, the weight of convolutional layers is shared to regularize the convolution layers and are reused individually in auxiliary paths. Weight sharing represents the coupling of input channels with corresponding to the auxiliary paths. Then, reused convolutional layers are connected to the FC layers “FC-f1-aux”, “FC-f2-aux”, and “FC-f3-aux”. These three FC layers are creating their own loss, called $loss_{aux}$. The $loss_{aux}$ are utilized to improve the main gated model with the main model loss function, $loss_{main}$. The detailed loss function of the proposed architecture is demonstrated in the following subsection.

The reasons that add auxiliary path are as follows:

Since our goal is to propose a new architecture which is more robust than the NetGate architecture, our new architecture should alleviate the limitation of NetGate architecture. The main limitation of the NetGate architecture is that the fusion weight scalars are not that reasonable. Nevertheless, the idea that uses fusion weight scalar to do weight fusion is definitely enlightening. Based on that idea, our new method should generate the reasonable fusion weight scalar as exactly as possible, which means we need to let the new architecture generate the fusion weight scalars that truly reflect the quality of the corresponding channel. Then we notice the second limitation of the NetGate architecture, the fusion mechanism, and the whole network is treated as a whole black box. Start from that, the idea that adds an auxiliary path to the main model architecture came up. By adding the auxiliary path, it actually means the black box is opened. Since the weight between the main model and the auxiliary are shared, if we changed the auxiliary path then the main model will be affected by the auxiliary at the same time. That is the meaning of “black box is opened”. Then, by adding the auxiliary path we can actually tune the fusion weight scalars. As a consequence, we can make the fusion weight scalars approach the quality indicator by using the auxiliary path. That is one of the reasons that the auxiliary path is proposed. The other reason is adding the auxiliary can be viewed as a way to do the regulation to the entire network. The robustness of the entire network will be enhanced as well.

The above paragraph discussed the auxiliary path proposed in the ARGate-WS**Error! Reference source not found.** and the reason behind it.

3.4 Problem Summary

To sum it up, our team has proposed several architectures that can alleviate the problem of the NetGate architecture. According to the previous experiments, the proposed architecture outperforms the NetGate architecture. However, there still is space to further improve the performance of the ARGate-WS architecture. In the next section, I introduced the solutions proposed in this work.

CHAPTER IV

SOLUTIONS

In this chapter, a modified version of the ARGate architecture developed as part of this thesis work is discussed. The details are discussed. The relevant experiments are present in the next chapter to demonstrate the performance of the new proposed architecture of this work.

4.1 The Loss Function in the pre-proposed Architecture

In[18], we proposed the ARGate-WS architecture and a Fusion Weight Regularization (FWR) method to regularize the network. The brief summary of FWR is in the next subsection.

4.1.1 The Fusion Weight Regularization (FWR loss function)

The network that uses the FWR is shown in Figure.8. In order to regularize the convolutional layers through the weight sharing mechanism, the main gated model loss ($Loss_{main}$) and losses from auxiliary paths ($Loss_{aux}$) are exploited.

First, only two losses, $Loss_{main}$ and $Loss_{aux}$ are utilized in equation (4.1). Please note that the alpha and beta are user-defined parameters.

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

$$e^{-\widehat{Loss_{aux}^{k}}^{2}} = softmax(l_{norm}^{2} \left(e^{-(Loss_{aux}^{k})^2} + 1 \right) * 2) \qquad (4.1)$$

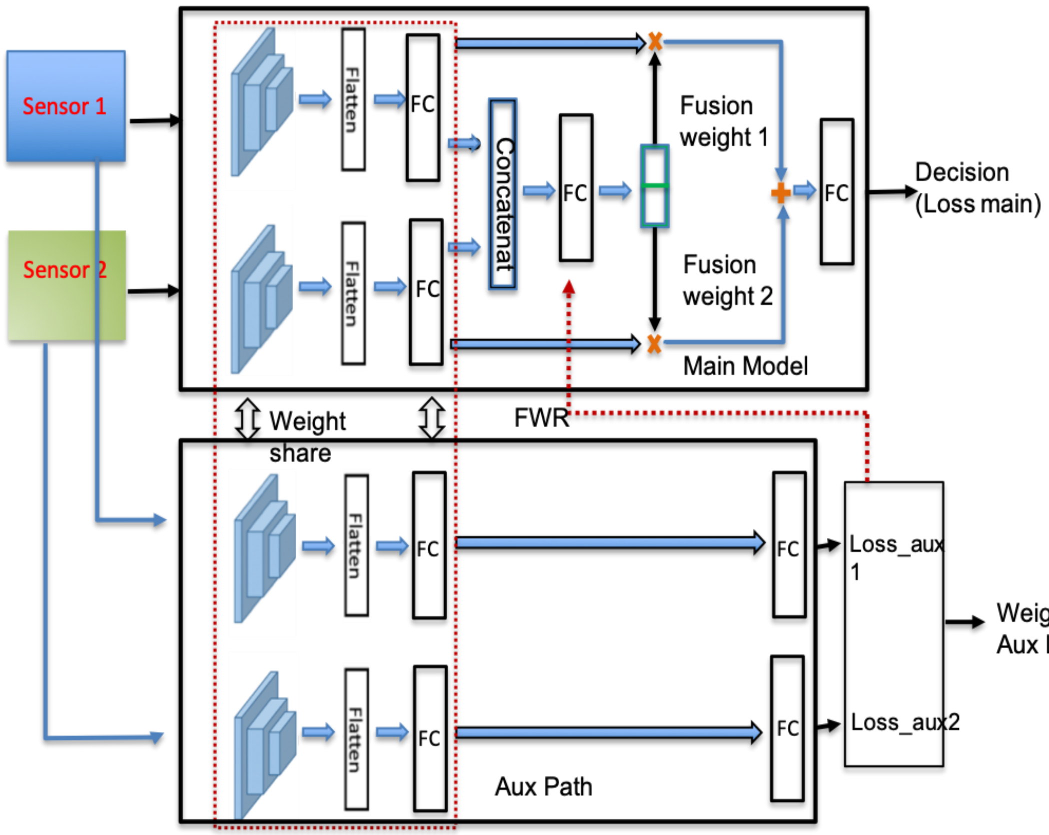

Figure 8: The ARGate architecture with FWR, ARGate-WS-FWR

In the equation, $Loss_{main}$ is the loss of the main model. And $Loss_{aux}^{k}$ is the loss of the kth input channel. w_{fusion}^{k} is the corresponding fusion weight for the kth channel.

Parameter regularization is made in the convolutional layers by sharing weight. However, weight sharing may bring a problem when one or more input channels are noisy or corrupted. Parameter sharing of corrupted input features in the auxiliary paths does not enhance the main gated model. Then add the FWR.

The key idea is to use the loss of the auxiliary path to regulate the fusion weight. In the ARGate-WS, the main is to let the auxiliary path participated into the process of training, and then use the performance of the auxiliary path to force the main model to generate reasonable fusion weight. By this way, the entire model is more capable to deal with the sensor failure. However, the way that an auxiliary path participated in that process is not strong enough. Then the FWR comes. In this method, the loss of the auxiliary path is the indicator of the quality of the corresponding channel. If one has a large loss compared with others, then that channel should have a smaller weight scaler.

4.2 Auxiliary Loss Weighting for Robust Multi-Modal Sensor Fusion

Here is the solution developed in this thesis work. It is aimed to further regulate the fusion weight scalar and improve the performance of the network. The architecture is shown in Figure 9.

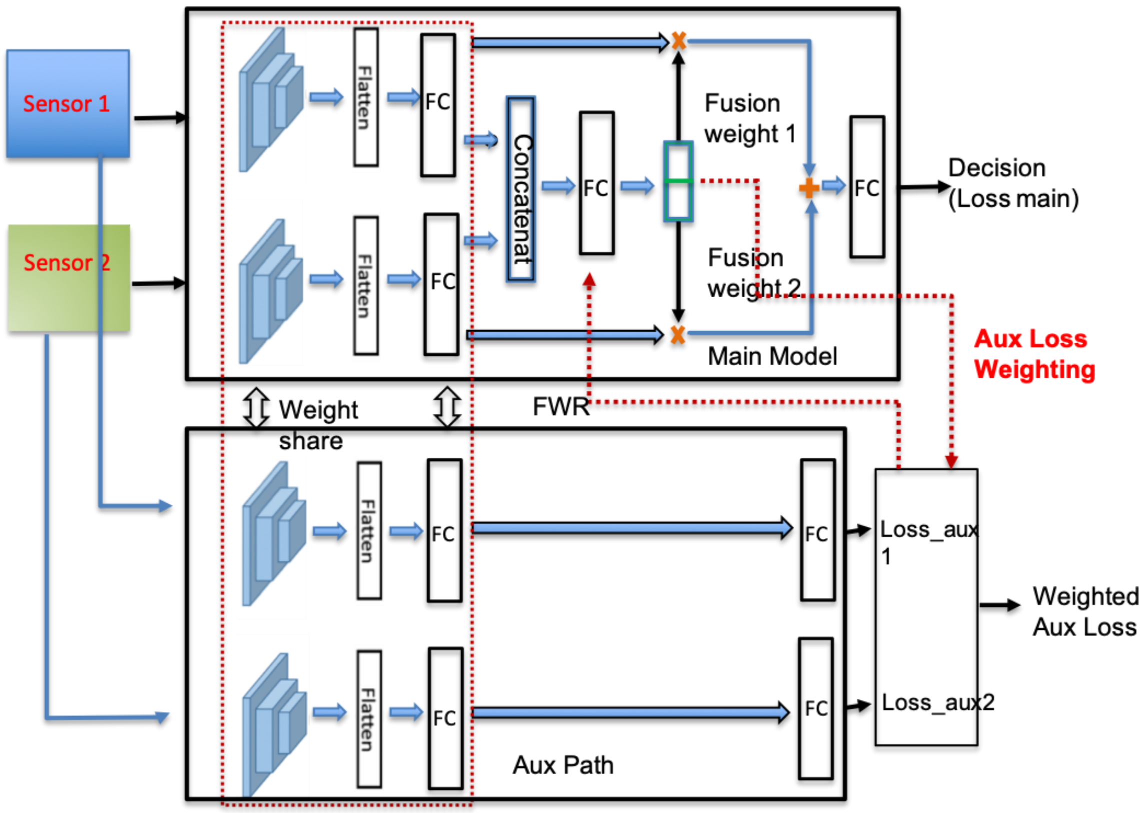

Figure 9. The ARGate with ALW, named ARGate-F

Since loss functions of all auxiliary paths are included in the total loss function of ARGate-WS and ARGate-WS-FWR. However, if one or multiple sensors corrupt, then their large loss functions may dominate other terms in total loss functions, the performances of the network may be degraded because of this. In order to address this issue, these works proposed the auxiliary loss weighting (ALW). Using the fusion weight extracted from the main model to multiply with the corresponding loss of the auxiliary path. It is shown by the red dash arrows in Figure 9. After applied ALW, we have the new loss function:

$$\mathrm{Loss}_{total} = \alpha \cdot Loss_{main} + \beta \cdot \sum_{k=1}^{K} w_{fusion}^{k} \cdot Loss_{aux}^{k} + \sum_{k=1}^{K} (w_{fusion}^{k} - \widehat{e^{-Loss_{aux}^{k}}}^{2})^{2}$$

$$\widehat{e^{-Loss_{aux}^{k}}}^{2} = softmax(l_{norm}^{2}\left(e^{-(Loss_{aux}^{k})^{2}} + 1\right) * 2) \qquad (4.2)$$

$Loss_{main}$ is the loss of the main model. And $Loss_{aux}^{k}$ is the loss of the kth input channel. w_{fusion}^{k} is the corresponding fusion weight for the kth channel.

The key point of this loss function is using the fusion weight of the main model w_{fusion}^{k} are multiplied with auxiliary path loss $Loss_{aux}^{k}$. The idea is that if the k-th sensor fails while the rest sensors are all functioning, the pure noisy inputs would generate a large $Loss_{aux}^{k}$, which will dominate the total loss function Loss_{total}. The quality of sensory input can be represented by fusion weight of the main model, and implementing w_{fusion}^{k} as a weighting term to multiply with $Loss_{aux}^{k}$ would be a solution to this issue.

By adding the ALW, the loss function becomes more complete. This modification can take cover the corner case of the model and as a result, the model that uses ALW has more generalization ability. The result showed in the next section proved this statement.

CHAPTER V

EXPERIMENTS

In this chapter, all experiments this research includes are presented and the related analyzes also shows at this chapter. In this section, several results for the traditional CNN, the baseline NetGate architecture, ARGate[18] and ARGate-WS-FWR [20] are presented and then are compared with the results of ARGare-F that proposed in this work. All data presented in this chapter are demonstrating that the proposed new architecture does have a decent accuracy compared to others.

5.1 Experiment setting

The experiments are based on two datasets: The HAR dataset and the CAD-60 dataset.

This work mainly focuses on the experiment based on the CAD-60 dataset and fellows are several setting this work adopted when test and train the ARGate-F. In the training process, mini-batch training is adopted. For HAR dataset, the batch size is 16 and for CAD-60 dataset, the batch size is 128. The optimization function was the ADAM optimizer and the learning rate was 0.001. Besides that, results for HAR dataset are based on the training for 200 epochs and the results for CAD-60 dataset are based on 100 epochs of training.

HAR dataset [21]

HAR Dataset is utilized which has the accelerometer and gyroscope sensory input with six activities: walking, walking upstairs, walking downstairs, sitting, standing and laying.

Sensors has three-axial total acceleration data ($\mathrm{total\_acc\_x}, \mathrm{total\_acc\_y}, \mathrm{total\_acc\_z}$). three-axial body acceleration data ($\mathrm{body\_acc\_x}, \mathrm{body\_acc\_y}, \mathrm{body\_acc\_z}$), and three-axial angular velocity ($\mathrm{body\_gyro\_x}, \mathrm{body\_gyro\_y}, \mathrm{body\_gyro\_z}$).

All data are sampled in sliding windows of 2.56 seconds with 128 reading per window, distributed between $[-1,1]$. 7352 samples are used for training and 2947 samples for testing the neural networks

CAD-60 Dataset [22]

CAD-60 dataset consists of 60 videos captured by Microsoft Kinect sensor for human activity detection, which contains a RGB video camera and a depth sensor. The dataset records four people's 14 activities, and videos are segmented into RGB and depth images. A feature extraction code is provided to extract features from the raw RGB and depth images, skeletal features. The extracted features are skeletal features, skeletal histogram of oriented gradients (HOG) features on RGB Image, RGB HOG, skeletal HOG features on the depth image sensor, and Depth HOG. Total 5 channels. The same "new person" setting in [22] is applied here, which uses the data from three people in training, and person for testing.

Fusion Weight Normalization

In [17] the concept of fusion weight is proposed and the fusion weight is generated by the FC layer "FC-con" as showed in Figure 4. However, the original NetGated architecture does not have normalization on these fusion weights. If the fusion weight scalars are not normalized then when do the weight fusion the relation between each channel will be chaos. That is based on the criteria that when we want to add several different distribute data together and want the added data has reserved all channels' information we should make all of them added in the same scale. To achieve that, we have applied two-step normalization on original NetGated and our proposed architectures: L2 normalization, then soft-max function on the outputs of the FC layer" FC-con". The two-step normalization is needed for verifying the meaning of fusion weight. Due to the fact that fusion weight is expected to be working as a gated scalar to represent input feature quality, we utilize the normalization for fusion weight comparison.

Neural Network Configuration

For the baseline NetGated and our proposed gated CNN with auxiliary paths (ARGate-F) architecture, 3x3 same padding convolution filters are utilized. That acts as the basement of the convolutional layer. The detailed configuration of ARGate-F for CAD-60 dataset is shown in Table. 1. To describe it in general, the architecture we adopted is based on the CNN and then apply RELU layer after each convolutional layer. After the ReLu layer, the 2*2 max pooling is added. Then the other main part is the Fully Connected layer. For the auxiliary path, the front part of it is the same as the main model, since the weight of the auxiliary path is shared with the main model the architecture needs to be the same. The latter part is a little different. The remaining part of the auxiliary part is the prediction part, the FC layer is applied. To sum it up, the feature extraction part of the main model and the auxiliary path is the same and then to simplify the auxiliary path and the size of the whole network the prediction part of the auxiliary path is slightly different. Table 1 shows the network configuration of the ARate-F on CAD-60.

In HAR dataset, slightly different parameters for convolution layers, pooling layers, and FC layers are applied. Detail settings are shown in appendix A.

Table 1: Network configuration of the ARGate-F on CAD-60 dataset

	Gated Main Model			Auxiliary Paths	
1	Layer Name	Layer Setting		Layer Name	Layer Setting
2	Conv1D ReLU	Kernel size: 3 Stride: 1 Padding: 1 # of filters: 8			
3	Max Pooling	2 * 2 down sample			
4	Conv1D ReLU	Kernel size: 3 Stride: 1Padding: 1 # of filters: 4		Same as the Main Model (share weight)	
5	Max Pooling	2 * 2 downsample			
6	Concatenate	Vertical Concatenation			
7	Fully Connected	# of neurons: 600		_	_
8	Fully Connected	# of neurons: 5		_	_
9	weighted fusion	Multiplication with output5		_	_

Table 1 Continued

	Gated Main Model			Gated Main Model	
10	Addition	Element-wise addition		_	_
11	Fully Connected	# of neurons: 200		_	_
12	Fully Connected	# of neurons: 14		Fully Connected	# of neurons 14

Runtime environment and device setting

This work evaluated the performance of the proposed gated CNN with auxiliary paths (ARGate-F) based on two public datasets: human activity recognition (HAR) dataset and CAD-60dataset. All simulations are done on Ubuntu 16.04 with Python 2.7 and Pytorch 0.4.0. The GPU we used is the NVIDIA TITAN Xp GPUs.

5.2 Sensor Noise and Failure

Sensor Noise

The sensor noise means the sensor work but the output may contain several noises. The noise is an additive noise usually represent by additive Gaussian noise. However, just simply add gaussian to the original clean output of the sensor may not be that appropriate. Because all sensors have their own way to simulate sensor noise. The detail information may be provided by the manufactory of the sensors.

In this research, I mainly focused on sensor failure. The detail information is presented in the follow paragraph.

Sensor Failure

Sensor Failure means the sensor is not working at all. The output of the sensor is several random noises. The simulation we used is based on the uniform distribution. The simulation equation is as equation (5.1)

$$v_n = \xi_n \tag{5.1}$$

Where ξ_n is a random variable for uniform distribution.

To generate noise with the same range of clean data, for CAD-60 dataset, $\xi_n \sim U[data_{min}, data_{max}]$ is exploited and $\xi_n \sim U[-1,1]$ is applied in HAR dataset. Where $data_{min}$, and $data_{max}$ represent the minimum and maximum of the clean data.

Besides the uniformed distribution, the Gaussian distribution also used to simulate the sensor failure situation. The simulation equation is as (5.2)

$$v_g = \xi_g \tag{5.2}$$

The same as the uniform distribution, $\xi_n \sim G[0,1]$ is exploited to both CAD-60 dataset and HAR dataset. Where 0 represent the mean of the data is 0 and 1 means the variance of the data is one.

In the actual runtime, since the γ is the coefficient, for the neural network the linear modification will not affect the result. Therefore, we always ignore the γ in the experiment.

Sensor Failure with Fixed Noisy Channel

To represent the real situation as better as possible, we split both training and testing dataset into the clean set and noisy set. $\frac{1}{3}$ of total training data are randomly selected as “clean set”, where all input channels are pure without noise. The outputs of those channels are just the pure data we get from the dataset. While the rest training data are defined to be “noisy set". In the noisy set, n_{fnoise} out of n channels are selected and fixed as noisy channels. n_{fnoise} is a constant which represents the number of fixed noisy channels. Note that n is the total number of input channels. Same noise scheme is applied to the testing set.

Sensor Failure with Dynamic Noisy Channel

In order to simulate more challenging failure cases, among noisy set defined in previous, n_{dclean} out of n input channels are randomly selected to have clean sensory inputs, while the rest $n - n_{dclean}$ channels have noisy inputs as in equation (5.1), clean channels are dynamically changing in different examples. n_{dclean} is a constant which represents the number of dynamic clean channels.

Mixed Failure Schemes

In practice, sensor failure may be a mixture of different noise schemes. Therefore, we performed a combination of different noise schemes in training and testing processes. In order to verify the robustness of our work, the proposed model is also tested in more realistic sensor failure situations when noise schemes in testing data are more complicated than those in training data.

5.3 Parameters in ALW

To propose the new architecture with the best performance, the parameters in the loss function need to be further tuned. Several experiments are done for that goal. In this section, several experiment results are provided and the final parameters that we used in loss function are presented.

The sensor failure model I used when doing the experiment is the fixed failure scheme. I use this scheme to test the performance of ARGate-F with loss function (5.2). The parameters we need are α and β. To find the best parameters lots experiments were done. From those experiments, we can find for HAR dataset, the best parameters are α = 5 and β=1.

The result is shown in Table 2 and Table 3. (only several main results are displayed here)

Table 2: Performances of ARGate-F with loss functions in HAR dataset

Noise Scheme	$\alpha = 5$ and $\beta = 1$	$\alpha = 1$ and $\beta = 4$	$\alpha = 1$ and $\beta = 5$	$\boldsymbol{\alpha}$ **= 5 and** $\boldsymbol{\beta = 1}$
$n_{dclean} = 1$	62.13%	63.42%	65.34%	**66.42%**

For the CAD-60 dataset. The parameters got by the experiment are $\alpha = 9$ and $\beta = 5$.

Table 3: Performances of ARGate-F in CAD-60 dataset

Noise Scheme	$\alpha = 3$ and $\beta = 5$	$\alpha = 6$ and $\beta = 5$	$\boldsymbol{\alpha = 9}$ **and** $\boldsymbol{\beta = 5}$
$n_{dclean} = 1$	60.31 %	61.49 %	**62.96 %**

5.4 Fusion Weight Distribution

In this part, the results of the distribution of fusion weight for several models are provided. the baseline NetGated and the ARGate with auxiliary paths using FWR and ALW, short named as ARGate-F, under the first dataset.

The goal is to make the fusion weight to become the indicator of the quality of the corresponding channel. Then the scalar should be small when a channel is less important or is noisy. As a consequence, we can judge the quality of our setting by seeing the distribution of the fusion weight scalar. If one architecture can generate the weight scalar that does distribute as our assumption then the corresponding architecture is better than the other.

In the experiment of the network based on ARGate-F, the α is set to 9 and the β is set to 5. Fusion weight normalization is applied to normalize $e^{-Loss^2_{aux_k}}$. After the normalization, the sum of all fusion weight scalar is 1. The results are based on the CAD-60 dataset.

This experiment is under the sensor failure situations, with n_{dclean} set as 1, γ as 0.5, only RGB HOG channel is clean in each training and testing example. When the input feature has pure noise, fusion weight should be distributed around a value which is much smaller than the clean feature. The Fig.10,11,12 shows the fusion weight of channel RGB HOG for clean testing examples based on the NetGated, the ARGate-WS-FWR and ARGate-F respectively

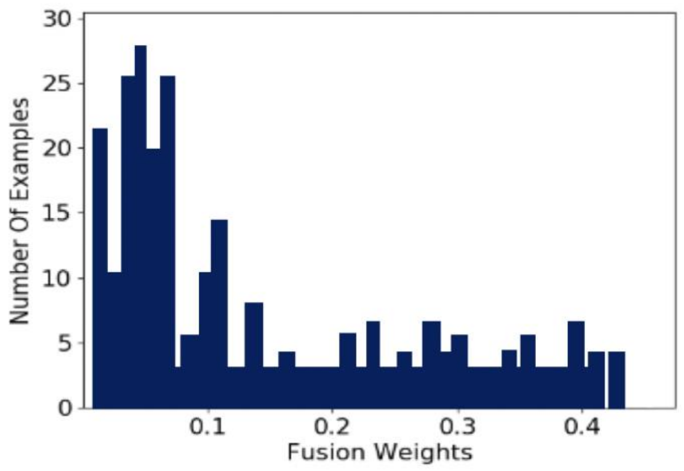

Figure 10: Weight Distributions of clean features for channel RGB HOG using NetGated, when n_{dclean} and γ are set to be 1 and 0.5, respectively

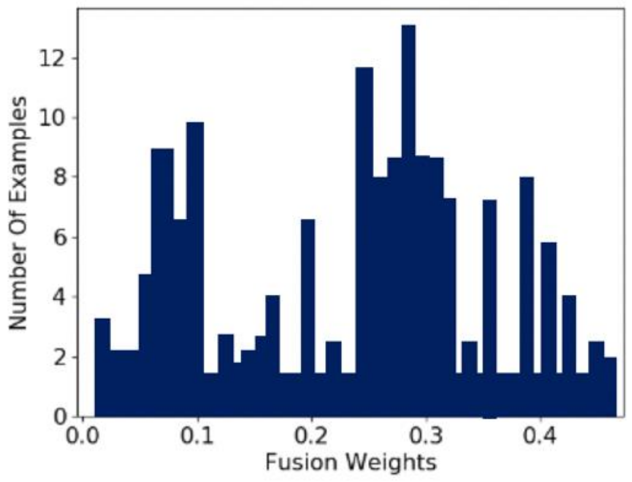

Figure 11: Weight Distributions of clean features for channel RGB HOG using ARGate-WS-FWR, when n_{dclean} and γ are set to be 1 and 0.5, respective

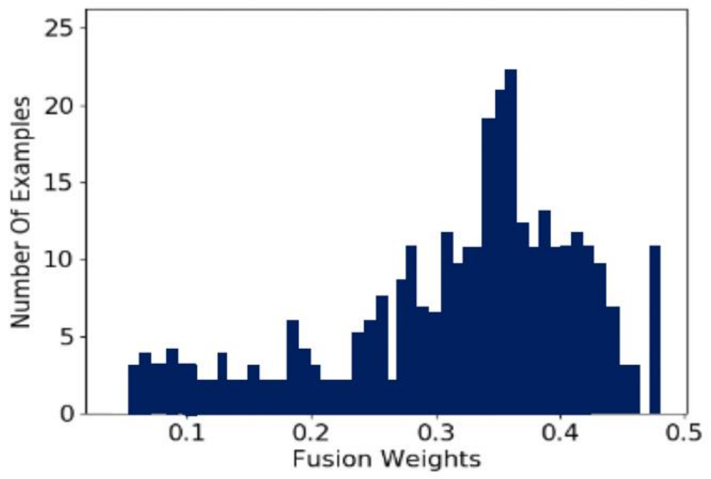

Figure 12: Weight Distributions of clean features for channel RGB HOG using ARGate-F, when n_{dclean} and γ are set to be 1 and 0.5, respectively

As showed in the above pictures, in Figure 10 we see that when using the NetGated architecture, when the corresponding channel is clean the fusion weight is mainly distributed at around 0.1 and relatively fewer fusion weight is distributed are 0.4. In Figure 11, we see that when using the ARGate-WS-FWR[20], the fusion weight does have more in the range around 0.4, however, on the same time, there are also lots of the fusion weight are distributed around 0.1. For the ARGate-F, we can see that most of the fusion weight is distributed at 0.3 ~ 0.4. That is what we want. From Figure 10 to 12 we can get the conclusion that the ARGate-F architecture has the ability to extract reasonable fusion weight and the performance of ARGate-F is the best when compared with the NetGate and the ARGate-WS-FWR. Similarly, when the input sensor RGB HOG is noise, the fusion weight should mostly distribute at smaller values than other clean features.

Together with the weight distributions in Figure.10,11,12, we can see that the Auxiliary loss weighting can further regulated fusion weight and thus is capable of representing sensor quality and thus achieve higher accuracy than NetGate. That supports this work's solution ARGate-F out performs others compared architecture.

5.5 Experimental Findings and Results in CAD-60 Dataset

Failures on Extracted Features

For CAD-60 Dataset, as we introduced in the previous paragraph, it was generated by 3 sensors, the RGB image, the depth image and the Skeleton data. Based on that three sensors, the CAD-60 Dataset also provides a feature extractor code that can extract five features from the original three sensors.

In this part, five extracted features from raw data are treated as five sensor input. Similar noise scheme is implemented to the five features. That means this work assumes we have five sensors instead of three sensors in the experiments presents in this section.

Simulation Results of Sensor Failure with Fixed Noisy Channel

The fellow results are for the experiment based on the simulation of sensor failure in the fixed noisy channel. n_{fnoise} stands for the number of fixed noisy channel. The experiment result is as shown in Table 4. The results of my solution are shown in the last column.

In this experiment, all failing sensors have inputs following uniform distribution between -1 and 1. $n_{fclean} = 1$, skeletal features, RGB HOG, skeletal HOG features on Depth Image, and Depth HOG are simulated to be corrupted sensors. $n_{fclean} = 4$, only skeletal channel is corrupted.

Table 4: Prediction accuracies under fixed failing sensor assignment for CAD-60 dataset

# Clean Channels	Baseline	NetGated	ARGate-WS-FWR	ARGate-F
$n_{fclean} = 1$	60.60%	59.98%	63.22%	65.14%
$n_{fclean} = 4$	64.15%	61.72%	69.83%	72.74%

From the result shows on Table 4 we can see that under the fixed noisy channel situation, the performance of the original CNN network performance is the worst. The NetGate architecture does perform better when compared to the original CNN network. The architecture that we compared with, ARGate-WS-FWR, performs better than baseline CNN and the NetGate. However, the proposed solution in this thesis, the ARGate-F's performance is the best. Compared with the baseline model (ARGate-WS-FWR) that this work improved from, the performance is improved around 2 ~ 3 %.

Theoretical limits for sensor fusion

The theoretical limits are the highest performance we can get when use multi-sensor. In this part, the theoretical limits for fixed channel sensor failure are explored.

To compared to theoretical limits with the result of the ARGate-F that proposed in this work, the noise setting and noise channel are the same as the setting for ARGate-F experiments. The theoretical limits are got as fellows, when one channel is under sensor failure, the corresponding channel will be manually closed. That is means, when do the sensory fusion, that channel's data will not be fused with others clean data.

The results for theoretical limits are shown in Table 5.

Table 5. The theoretical limits for the sensor fusion

# Clean Channels	Theoretical limits	ARGate-F
$n_{fclean} = 1$	66.64%	65.14%
$n_{fclean} = 4$	73.77%	72.74%

From result shows in Table 5 we can see that the result of ARGate-F is almost the same as the Theoretical limits. The difference between these two is around 1%. Further proved ARGate-F does have decent performance. Even though there is still some gap between the ARGate-F and the theoretical limits, the performance of the ARGate-F does perform best when compared with others.

Simulation Results of Sensor Failure with Dynamic Noisy Channel

The fellow's results are for the experiment that simulation of sensor failure with dynamic noisy channel

Simulations are set up with $n_{dclean} \in \{1,2,3,4\}$. For the training epochs, considering the time cost, is set to do 100 epochs training. The detail results of the experiment are shown in Table 6.

We can see from the results in Table. 6 that when all five channels are clean, the proposed gated CNN with auxiliary paths has 0.5% improvement over traditional CNN, and 0.44% over ARGate-WS-FWR. With the increase of the clean channel, all CNN, ARGate-WS-FWR and ARGate-F's performance increased. That is because, with the increase of the clean channel, the total information that input can provide is increased, even though CNN cannot fully use all information the performance of CNN still can improve since the total information is increased.

Table 6: Prediction accuracies under random failing sensor assignment for CAD-60 dataset

Clean Channels	Failure Model	Baseline	NetGated	ARGate-WS-FWR	ARGate-F
All Clean	-	87.01%	86.57%	87.37%	87.51%
$n_{rclean} = 4$	Zero	71.91%	68.87%	78.67%	82.01%
	Uniform	69.76%	65.13%	78.81%	79.78%
	Gaussian	71.55%	73.81%	75.74%	77.93%
$n_{rclean} = 3$	Zero	72.91%	65.48%	71.47%	77.13%
	Uniform	69.38%	67.61%	71.96%	73.86%
	Gaussian	88.41%	89.04%	90.07%	74.75%
$n_{rclean} = 2$	Zero	67.94%	67.98%	64.41%	68.17%
	Uniform	64.98%	62.98%	66.59%	66.70%
	Gaussian	67.41%	66.55%	66.96%	68.51%
$n_{rclean} = 1$	Zero	59.07%	28.18%	59.89%	60.01%
	Uniform	61.42%	57.10%	61.35%	61.82%
	Gaussian	57.44%	57.75%	57.55%	57.96%

However, if we see the performance of CNN, ARGate-WS-FWR, and ARGate-F at the same time we can find that the ARGate-F 's performance keeps being the best one.

To sum it up, from the table we can see that after applied ALW(ARGate-F), the maximum network's performance improved can reach to 3.5 %, on average the improvement is around 2 %.

Combined with the simulation of sensor failure with the fixed noisy channel, we can see that when we treat the five extracted features as five sensors, the new proposed ARGate-F architecture's performance is the best and the improvement is around 3.5 %. The improvement demonstrated the superiority of the ALW and ARGate-F architecture.

Failures on Original Inputs

In the previous paragraph, the input of the network is the five extracted features. Those five extracted features are treated as five sensors. In this part, in order to simulate what actually happens on sensors, noise is added directly to the raw data that sensors have. Threes sensors of CAD-60 dataset are the RGB images, the Depth images and the Skeleton data. The skeleton data are generated from the RGB images.

The noise scheme I adopted for the experiment is the dynamic noise channel scheme. To specify, this experiment is based on the simulation of sensor failure based on dynamic noisy channel and fixed noisy channel.

Since the experiment result shows in Table 4, 5 and 6 already proved the ARGate-F architecture does perform better than the other architecture for the experiment presents in this part, only one setting of noise situation is explored. We try to use the most extreme setting to demonstrate that the ARGate-F architecture does can improve the performance even in such worst situation. Therefore, the number of clean channels is set to be 1 ($n_{dclean} = 1$).

The result is shown in Table 7.

Table 7: Accuracy on CAD-60 when original inputs (RGB and RGB-D images) under dynamic noise schemes

Noise Scheme	Noise Level	CNN	NetGate	ARGate-F
All clean	-	87.01%	86.57%	87.51%
Ndclean = 1	Gaussian	74.33%	74.30%	76.02%
Ndclean = 1	Uniformed	73.16%	72.26%	76.42%
Ndclean = 1	Zero noise	73.53%	72.08%	77.09%

This simulates the mixture of two situations: the two sensors (RGB and RGB-D) are both working, only one of the two sensors are working. The same feature extraction code is implemented to extract five features, which are input to the same network in the experiment of sensor failure with fixed noisy channel.

In Table.7, when RGB and Depth data are both clean, I got the same results as in “All Clean” noise scheme in Table. 6. Under the noise scheme mentioned above, the proposed ARGate-F has an improvement of 1.72% over NetGate, and 1.69% over traditional CNN.

Besides the dynamic failing sensor setting, the fixed noise setting is also explored. The results are shown in Table 8.

Table 8: Accuracy on CAD-60 when original inputs (RGB and DRGB images) under fixed noise schemes

Noise Scheme	Noise Level	CNN	NetGate	ARGate-F
All clean	-	87.01%	86.57%	87.51%
Depth	Gaussian	65.27%	61.05%	67.02%
RGB	Gaussian	70.77%	67.98%	73.71%

Combined the result in Table 7 and Table 8, we can see that the performance of the ARGate-F is also the best one no matter the noise is added on the source of the input or on the extracted features.

Failing Sensor Assignment for Model Generalization

To test the generalization ability of the ARGate-F architecture, fellow experiments are done. The model generalization means when the model is tested in the scenario that has never been seen during the training process. If the model can still have decent performance, then that means the model has generalization ability.

In the fellow experiments, several representations need to be clarified. (1, 4) (2, 4) means the number of failing channels in train dataset in each example is randomly selected from [1, 4], while the range of failing channel numbers in the test set is between [2, 4].

Table 9: Prediction accuracies under failing sensor assignments for testing generalization for CAD-60 dataset

#Failing Channels	Baseline	NetGated	ARGate-F
(1,4)(2,4)	64.08%	63.71%	68.36%
(2,3)(2,8)	55.36%	55.16%	58.04%
(2,4)(1,4)	60.77%	61.05%	62.09%

When the experiment is under setting (1, 4) (2, 4), the improvement is 2.63%. Which is the maximum improvement compared to ARGate-WS-FWR. For other settings, the performance of ARGate-F is also the best.

5.6 Experimental Findings and Results in HAR Dataset

In this section, the result of the CNN Baseline, NetGated, ARGate-WS and ARGate-WS-FWR is cited from [20]. The last column's results are for ARGate-F, which was explored by myself.

Table 10 shows the result of random failing sensor assignment. Table 11 shows the result of fixed noise assignment result.

The results in Table 10 and Table 11 further proves the performance of ARGate-F is the best one and the ALW is proved to be useful.

Table 10: Prediction accuracies under random failing sensor assignment for HAR dataset[20]

# Clean Channels	Failure Model	Baseline[20]	NetGated[20]	ARGate-WS[20]	ARGate-WS-FWR[20]	ARGate-F
All Clean	-	94.06%	94.50%	94.96%	95.09%	95.69%
$n_{rclean} = 8$	Zero	93.02%	93.17%	94.66%	94.04%	94.60%
	Uniform	92.35%	92.20%	92.45%	92.46%	92.57%
	Gaussian	92.94%	93.28%	94.97%	94.35%	94.13%
$n_{rclean} = 5$	Zero	88.36%	87.95%	88.63%	88.83%	89.89%
	Uniform	86.73%	86.80%	88.53%	89.17%	89.51%
	Gaussian	88.41%	89.04%	89.52%	90.07%	90.12%
$n_{rclean} = 1$	Zero	71.56%	71.12%	74.38%	74.44%	74.62%
	Uniform	62.06%	62.90%	65.69%	66.09%	67.09%
	Gaussian	69.67%	70.54%	71.83%	72.58%	73.19%

Table 11: Prediction accuracies under fixed failing sensor assignment for HAR dataset[20]

# Clean Channels	Baseline[20]	NetGated[20]	ARGate-F
$n_{fclean} = 5$	87.68%	89.28%	91.01%
$n_{fclean} = 6$	80.59%	81.94%	84.52%

CHAPTER VI

SUMMARY AND CONCLUSIONS

6.1 Summary

This work introduced a way to further improve the performance of the ARGate-WS-FWR, named Auxiliary Loss Weighting (ALW). The model combined with ALW is called ARGate-F, a sensor fusion neural network architecture. The ARGate-F different from the ARGate-WS-FWR as it has a feedback path from the fusion weight to the loss of the auxiliary path to do regulation for the auxiliary path. This method it the main point of this thesis. ARGate-F, in turn, is more capable to deal with the sensor failure situation. The experiments on the CAD-60 Dataset showed the performance boost of the ARGate-F over the CNN, the NetGate and the ARGate-WS-FWR.

6.2 Conclusions

This thesis proposed the Auxiliary Loss Weighting for Robust Multi-Modal Sensor Fusion with Deep Neural Networks. The proposed method makes the architecture outperform existing ARGate-WS-FWR, NetGated architecture and basic CNN approaches. The proposed model shows robust classification performance on noisy data under sensor failure cases. For future work, the proposed architecture will be exposed to camera and LIDAR images for more complex sensory input.

REFERENCES

[1] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[2] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[3] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[4] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

[5] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

[6] Ren, Shaoqing, et al. "Faster R-CNN: towards real-time object detection with region proposal networks." *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6 (2017): 1137-1149.

[7] He, Kaiming, et al. "Mask r-cnn." *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017.

[8] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[9] Chen, Xiaozhi, et al. "Multi-view 3d object detection network for autonomous driving." *IEEE CVPR*. Vol. 1. No. 2. 2017.

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

[11] Shi, Shaoshuai, Xiaogang Wang, and Hongsheng Li. "PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud." *arXiv preprint arXiv:1812.04244* (2018).

[12] Geiger, Andreas, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous

[13] Kim, Jaekyum, et al. "Robust Deep Multi-modal Learning Based on Gated Information Fusion Network." *arXiv preprint arXiv:1807.06233* (2018).

[14] Ramachandram, Dhanesh, and Graham W. Taylor. "Deep multimodal learning: A survey on recent advances and trends." *IEEE Signal Processing Magazine* 34.6 (2017): 96-108.

[15] Karpathy, Andrej, et al. "Large-scale video classification with convolutional neural networks." *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014.

[16] Mees, Oier, Andreas Eitel, and Wolfram Burgard. "Choosing smartly: Adaptive multimodal fusion for object detection in changing environments." *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016.

[17] Patel, Naman, et al. "Sensor modality fusion with CNNs for UGV autonomous driving in indoor environments." *International Conference on Intelligent Robots and Systems (IROS). IEEE*. 2017.

[18] Zhao, Chenye, Myung Seok Shim, Yang Li, Xuchong Zhang, and Peng Li. "Deep Neural Networks with Auxiliary-Model Regulated Gating for Resilient Multi-Modal Sensor Fusion." arXiv preprint arXiv:1901.10610 (2019).

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[20] Chenye Zhao, Hardware Testbed and Deep Neural Networks for Multi-Modal Sensor Fusion, 2019, Texas A&M University Master of science thesis.

[21] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In ESANN, 2013.

[22] Sung, Jaeyong, Colin Ponce, Bart Selman, and Ashutosh Saxena. "Unstructured human activity detection from rgbd images." In 2012 IEEE international conference on robotics and automation, pp. 842-849. IEEE, 2012.

APPENDIX A

A.1 Neural Network Configuration used for HAR Dataset

	Gated Main Model			Auxiliary Paths	
1	Layer Name	Layer Setting		Layer Name	Layer Setting
2	Conv1D ReLU	Kernel size: 32. Stride: 8 Padding: 0 # of filters: 16		Same as the Main Model (share weight)	
3	Max Pooling	2 * 2 down sample			
4	Fully Connected	# of neurons: 256			
6	Concatenate	Vertical Concatenation			
7	Fully Connected	# of neurons: 256		_	_
8	Fully Connected	# of neurons: 9		_	_
9	weighted fusion	Multiplication with output5		_	_
10	Addition	Element-wise addition		_	_
12	Fully Connected	# of neurons: 6		Fully Connected	# of neurons 6