MODEL OPTIMIZATION AND APPLICATIONS IN DEEP LEARNING

Chengchen Mao

MODEL OPTIMIZATION AND APPLICATIONS IN DEEP LEARNING

by

CHENGCHEN MAO

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2023

To my dear family.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Qilian Liang for constantly motivating and encouraging me, and also for his invaluable advice during the course of my doctoral studies. Without his training, I would not be able have the chances to explore the broad scope of signal processing and deep learning. I wish to thank my academic advisors Dr. Chenyun Pan, Dr. Ioannis D. Schizas, for their interest in my research and for taking time to serve in my dissertation committee.

I am grateful to all the teachers who taught me during the years I spent in school, first in China, then in the United States. I would like to thank my supervisor of master degree Dr. Jing Liang, who used to be the outstanding graduate from the University of Texas at Arlington, for encouraging and inspiring me to pursue graduate studies in the United States.

Great thanks are expressed to our group members from the Wireless Communications and Networking (WCN) Laboratory, Dr. Zikai Wang, Dr. Zhangliang Chen, Dr. Dheeral Bhole, and Swarada, for their interest in my research, helpful discussions, and invaluable comments. I would also like to extend my appreciation to the people I met during these years. I want to thank Dr. Yuxin Zhao, Dr. Yangyang Qian, and Dr. Guangda Zang for their emotional support.

Finally, I would like to express my deep gratitude to my parents who have encouraged and inspired me. I am extremely fortunate to be so blessed to have such loving and supportive parents. Without their constant encouragement and guidance, I could not have made it through the difficult times of the COVID-19 pandemic.

During the pandemic, when I was feeling anxious and stressed, my parents provided me with emotional support that helped me get through those tough moments.

This work was supported by U.S. National Science Foundation (NSF) under Grant CCF-2219753.

August 9, 2023

ABSTRACT

MODEL OPTIMIZATION AND APPLICATIONS IN DEEP LEARNING

Chengchen Mao, Ph.D.

The University of Texas at Arlington, 2023

Supervising Professor: Qilian Liang

Machine learning refers to a machine or an algorithm that draws experience from data. A certain pattern is found to build a model, which is used to solve real problems.

Deep learning, an important branch and extension of machine learning, employs a neural network structure containing multiple hidden layers. It learns critical features of the data by combining lower-level features to form more abstract higher-level representations of attribute categories or features.

In this dissertation, deep learning network models were applied to sense-through-foliage target detection and extended with Rake structure. The deep learning network models had a large number of redundant parameters from the convolutional layer to the fully-connected layer, and a large number of neuron activation values converged to zero. The challenging task was to reduce parameter redundancy while maintaining model accuracy.

In Chapter 2, an approach based on stacked autoencoders (SAE) was proposed for ultra wide band radar for sense-through-foliage target detection. SAE, as one of

the widely used deep learning structures, could learn representations of data with multiple levels of abstraction automatically. The SAE-based target detection approach performed well in processing poor signal collections in some positions. In other positions, a single radar target detection performed under satisfaction. Rake structure was applied in radar sensor networks with maximum ratio combining and equal combining to combine radar echoes from different radar cluster-members.

In Chapter 3, pruning in deep learning network models was investigated. Pruning presented significant opportunities for compression and acceleration in deep neural networks by eliminating redundant parameters. Structured pruning gained popularity in the edge computing research area, especially with more terminal chips integrated with AI accelerators for Internet of Things (IoT) devices. Stripe-wise pruning (SWP), which conducted pruning at the level of stripes in each filter, was different from filter pruning and group-wise pruning. The existing SWP method introduced filter skeleton (FS) to each stripe, setting an absolute threshold for the values in FS, and removing stripes whose corresponding values in FS could not meet the threshold. The research involved investigating the process of stripe-wise convolution and using the statistical properties of the weights located on each stripe to learn the importance between those stripes in a filter and remove stripes with low importance.

In Chapter 4, the conception of a deep energy autoencoder (EA) for a noncoherent multicarrier single-input and multiple-output (SIMO) system operating amidst multipath channels was explored. The multicarrier SIMO structure involved a single-antenna sender and a multi-antenna receiver, both depicted via neural networks. The encoder generated a real-valued vector for each subcarrier, while the decoder received the combination of energy from all the receiving antennas. To address the major challenge of mitigating intersymbol interference (ISI) caused by multipath channels without relying on delicate designs common in traditional communication systems,

two different types of neural networks, namely DNN (Deep Neural Network) and RNN (Recurrent Neural Network), were adopted for the demodulation rule at the receiver. Simulation results demonstrated that, with adequate training, RNN efficiently recovered the transmitted data even in the absence of channel state information, which was often required in traditional communication systems.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Deep learning (DL) has demonstrated its remarkable power across various domains, transforming the way we understand and interact with technology. In the field of computer vision, deep learning models facilitate deep analysis of images, making tasks like facial recognition, object detection, and semantic segmentation possible [1]. These advancements have revolutionized industries, ranging from security to entertainment. Meanwhile, in the realm of speech recognition, deep learning has taken the lead, enabling applications such as speech-to-text and voice assistants [2]. These applications have become integral in our daily lives, enhancing accessibility and convenience in communication and information retrieval. The capabilities of deep learning extend beyond visual and auditory processing. Large language models, like GPT, leverage the power of deep learning to deeply understand natural language [3], providing powerful tools for tasks like chatbots, machine translation, and automated text generation. This has brought significant strides in bridging language barriers and personalizing user experiences. Additionally, deep learning's application in data analysis, including signal propagation in complex networks [4], enables the extraction of valuable insights and patterns from large datasets. These insights have been pivotal in understanding how signals propagate and interact within various networks. Building on these successes, the applications of deep learning continue to expand into more specialized fields, such as the analysis of criminal networks [5] and decoding complex brain signals in EEG Signal Analysis [6]. These explorations demonstrate deep learning's potential to address intricate challenges, contributing to fields like law enforcement and healthcare, and further emphasizing its pivotal role in technological innovation.

Model Optimization is a critical aspect of deep learning that involves the fine-tuning and alteration of algorithms to enhance efficiency and performance. This may include reducing the complexity of the model, the number of computations required, or the memory needed to execute the program. Optimization techniques such as pruning, quantization, and the use of efficient architectures like MobileNets are commonly implemented [7]. These methods enable deep learning models to run on a broader range of platforms, including low-power devices like smartphones and embedded systems. By reducing resource requirements, model optimization opens doors to real-time applications and makes the deployment of advanced AI technologies more accessible and cost-effective.

In this dissertation, we will delve into the subject of model optimization and applications in deep learning. The focus of our discussion will be on pivotal topics that illustrate the range and capability of optimized models within complex environments. First, we will explore the sense-through-foliage target detection based on stacked autoencoder and UWB radar sensor networks. Next, we will examine a statistical spproach for neural network pruning with application to the Internet of Things (IoT). Lastly, we will discuss the implementation of deep energy autoencoder for noncoherent multicarrier SIMO systems in multipath scenarios. Together, these subjects encapsulate the cutting-edge developments in model optimization, shedding light on the applicability of these methods across a variety of fields and applications. Through the exploration of these intricate topics, we aim to provide a comprehensive understanding of how model optimization shapes the landscape of deep learning and empowers new technological frontiers.

1.2 Fundamentals of UWB ranging

Ultra-wideband (UWB) is a low-energy, short-range, high-bandwidth wireless protocol using 3.1 to 10.6 GHz frequencies. Initially defined by DARPA with a fractional bandwidth (B_f) over 0.25, it was later reduced to 0.2, based on specific frequency parameters [8]. The FCC considers a signal as UWB if it's 500MHz or more and under 0.5mW, limiting applications but allowing co-existence with other signals.

The fractional bandwidth, as specified in (1.1), is determined by two frequencies: f_L, representing the lower frequency of the -10dB emission input, and fH, representing the upper frequency of the -10dB emission point.

$$B_f = 2\frac{f_H - f_L}{f_H + f_L} \tag{1.1}$$

The transmission center frequency, denoted as f_c, is computed by taking the average of the indicated cut-off frequencies in (1.2):

$$f_c = \frac{f_H - f_L}{f_H + f_L} \tag{1.2}$$

The IEEE 802.15.4a standard provides support for the UWB PHY option, which grants devices a remarkable ranging capability [9]. UWB technology utilizes the time it takes for radio waves to travel between devices, enabling precise tracking of objects. Apple's AirTag makes the most of this UWB capability, empowering users to locate lost or misplaced items with unparalleled accuracy via the Find My network [10].

In Chapter 2, we will utilize the ranging capabilities of UWB in conjunction with the stacked autoencoder to detect sense-through-foliage targets.

1.3 Essentials of Deep Learning

Activation functions are crucial to Deep Neural Networks (DNNs). They transform the weighted input into the neuron's output, adding non-linearity to the network which helps it learn complex patterns. Common activation functions include ReLU, tanh, sigmoid, and softmax. Each has specific benefits and is suitable for different scenarios. Choosing the right function depends on the problem and data characteristics. Table 1.1 summarizes the commonly used activation functions.

Table 1.1: List of activation functions

Name	$[\sigma(\mathbf{u})]_{\mathbf{i}}$	Range
linear	u_i	$(-\infty, \infty)$
ReLU	$\max(0, u_i)$	$[0, \infty)$
tanh	$\tanh(u_i)$	$(-1, 1)$
sigmoid	$\frac{1}{1+e^{-u_i}}$	$(0, 1)$
softmax	$\frac{e^{u_i}}{\sum_j e^{u_j}}$	$(0, 1)$

Convolutional Neural Networks (CNNs) are a class of deep learning algorithms, mainly utilized in image processing. They consist of convolutional layers that automatically detect patterns like edges, colors, and textures. By stacking these layers, CNNs can recognize complex features. In Chapter 3, we will delve into the optimization of two prominent CNN architectures, namely VGG (Visual Geometry Group) and ResNet (Residual Networks). These optimizations are targeted at enhancing their performance and efficiency, adapting them for various applications.

VGG network is a convolutional neural network model known for its simplicity and effectiveness. Created by the VGG team at Oxford, VGG has become a popular choice for image recognition tasks. It consists of multiple convolution layers with small 3×3 filters, followed by max-pooling layers. The architecture is followed by fully connected layers leading to the final classification output. As shown in Figure

1.1, the VGG network consists of several convolution layers and max-pooling layers. Despite its performance, VGG is often criticized for being computationally intensive, which may lead to challenges in deployment on devices with limited resources.

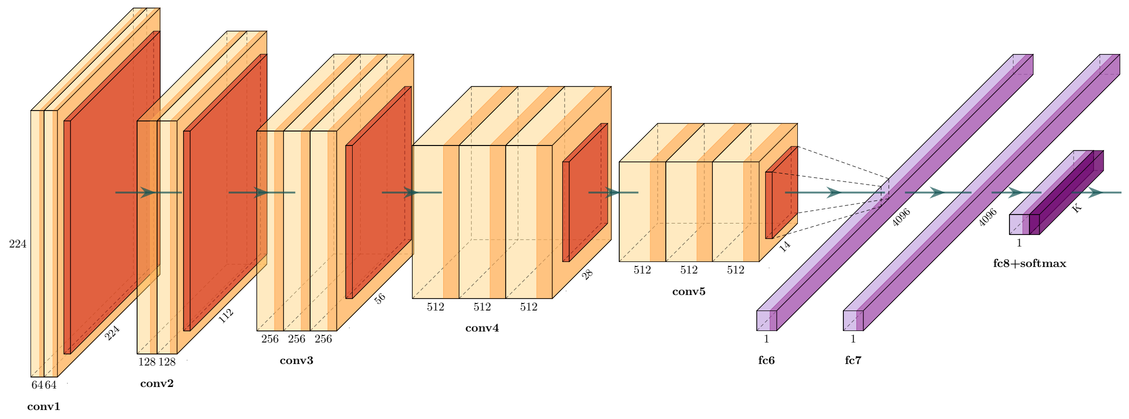


Figure 1.1: A VGG network

ResNet, is a pioneering architecture in deep learning that leverages residual connections or ”skip connections”, as shown in Figure. 1.2. These connections allow the gradient to flow directly through the network, making it easier to train deeper models [11]. By having layers learn residual functions and performing identity mappings merged with layer outputs, ResNet can significantly mitigate the vanishing gradient problem. This unique mechanism has enabled the training of networks with hundreds of layers, pushing the boundaries of depth and complexity. ResNet has become foundational in various applications including image recognition, and its principles are found in other advanced models and systems like Transformer models [12] and AlphaGo.

Recurrent Neural Networks (RNNs) have been designed to imbue neural networks with the capability of memory. This property is key for handling sequential data such as in Natural Language Processing (NLP) where the context plays a signifi-

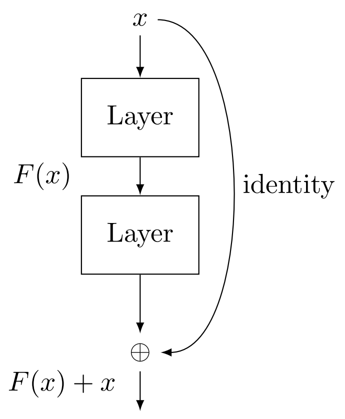


Figure 1.2: The Residual Connection skips two layers

cant role. In more traditional, memoryless neural networks, the neurons in each layer are connected only to those in the preceding and subsequent layers, with no intra-layer connections. However, this architecture does not provide the network with the ability to maintain and utilize any contextual or sequential information from prior states, which could limit its performance on tasks that inherently require knowledge about prior inputs. RNNs address this limitation by incorporating feedback connections in the hidden layers (Figure. 1.3). This means that the neurons in a given layer receive not just inputs from the preceding layer, but also the outputs of their own layer from previous steps. In essence, a recurrent neuron maintains a kind of memory by using its output from the previous step as part of its input for the current step. This allows the network to 'remember' and use information from the past, effectively enabling it to handle data where temporal dynamics and dependencies matter. In chapter 4, we will leverage the memory capabilities of RNN to mitigate the signal detection problem in SIMO systems caused by different delays in multipath channels.

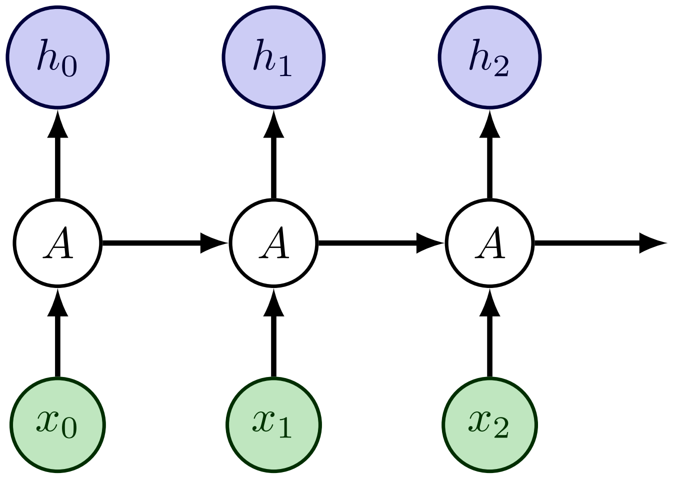


Figure 1.3: Parts of an RNN network

1.4 Overview of Dissertation

Chapter 2 introduces a method for sense-through-foliage target detection utilizing a Stacked Autoencoder (SAE) based approach. The SAE is capable of extracting essential information and deep features from radar echoes, even in cases of poor signal quality. Experimental results demonstrate its high detection accuracy, although there are positions where performance is suboptimal. To enhance the system, a RAKE structure in RSN has been implemented, which preprocesses input by combining echoes from different cluster-member radars. Simulations confirm that this integration significantly improves detection accuracy."

Chapter 3 presents an exploration of pruning's remarkable potential for compressing and accelerating deep neural networks, with a focus on eliminating redundant parameters. In conjunction with the integration of terminal chips with AI accelerators in Internet of Things (IoT) devices, structured pruning, specifically stripe-wise pruning (SWP), is emerging as a significant trend in edge computing. Unlike traditional methods, SWP targets the stripes in each filter, introducing a filter skeleton (FS) to set a threshold for value removal. The chapter delves into the statistical examination of the weights on each stripe, learning their importance and removing those with low significance. This novel approach has led to substantial results in our pruned VGG-16 model, achieving a 4-fold reduction in parameters with only a minor decrease in accuracy.

Chapter 4 explores the potential of a deep energy autoencoder (EA) for non-coherent multicarrier SIMO systems operating in multipath channel environments. The heart of our methodology lies in a distinctive multicarrier SIMO architecture, employing a single-antenna sender and multi-antenna receiver, both modeled through neural networks. The encoder uniquely generates real-valued vectors for individual subcarriers, while the decoder skillfully compiles energy from all receiving antennas. Challenging conventional methods, we addressed the issue of ISI caused by multipath channels not through traditional complex designs, but by capitalizing on DNNs and RNNs during the demodulation process. Our simulations incorporated the Jakes' model, considering variable Doppler frequencies, thus enhancing the design's practicality and robustness.

Chapter 5 concludes the dissertation by emphasizing its main accomplishments and outlining potential avenues for future research.

CHAPTER 2

Sense-Through-Foliage Target Detection Based on Stacked Autoencoder and UWB Radar Sensor Networks

2.1 Introduction

Radar target detection is a significant topic in obstacle avoidance as well as homeland security. Especially in a strong background clutter, the non-stationary nature of the complex environment like forest, provides a good cover for hostile forces and sabotage activities, where the doppler shift caused by the wind blowing through the leaves and the branches makes the target detection difficult. Meanwhile, in the rich scattering environment the multipath fading that impulsively corrupts received echoes including target as well as clutter information can degrade detection performance [13].

In this work, we focus on sense-through-foliage target detection problems using ultra wide band (UWB) radar. There are a wide range of signals on target detection in foliage related applications, like signals at VHF and UHF bands in near ground path loss modeling in a foliage [14], and millimeter-wave frequencies in wave propagation through foliage and forest ground reflectivity [15]. Good penetration capability and high range resolutions make UWB radar ideal for target detection behind clutter.

There are several works conducted on sense-through-foliage target detection with UWB radar. Approaches with discrete-cosine-transform (DCT)-based and short-time Fourier transform (STFT)-based approach were proposed for target detection in [16] and [17], respectively. Methods based on information theory using mutual information [18] and relative entropy [19, 20] were also appeared in this area. Some feature-extraction based detection methods were also put forward [21]. However, these features are mostly designated by the researchers and would be incomplete without sufficient prior knowledge. The deep learning theory could automatically extract the deep abstract features. It constructs a neural network containing multiple processing layers to learn representations of data with multiple levels of abstraction [22]. As a widely utilized deep learning structure, stacked autoencoder (SAE) [23], has attracted considerable attention in radar sensing society, including hyperspectral remote sensing [24], SAR image classification [25] and HRRP target recognition [26].

Inspired by the prior researches, with UWB radar, we apply the SAE to sense-through-foliage target detection. In addition, to overcome the poor quality echo signal collected from radar sensors [27–29], we combine RAKE structure and SAE based sense-through-foliage target detection, which brings a significant improvement in performance.

In this chapter, we applied stacked autoencoder and UWB radar sensor networks sense-through-foliage target detection. The reminder of this chapter is organized as follows: Section 2.2 briefly introduces measurement and data collection used in this chapter. In Section 2.3, we present the basic knowledge of SAE. In Section 2.4, the SAE based sense-through-foliage target detection approach is present. Section 2.5 adds RAKE structure and shows the improvement. In Section 2.6, this chapter is summarized.

2.2 Data Measurement and collection

The sense through foliage UWB data were collected by Virtual Machine Company in Massachusetts in late summer and early fall. As show in Figure. 2.1, the target is a trihedral shape metal reflector.

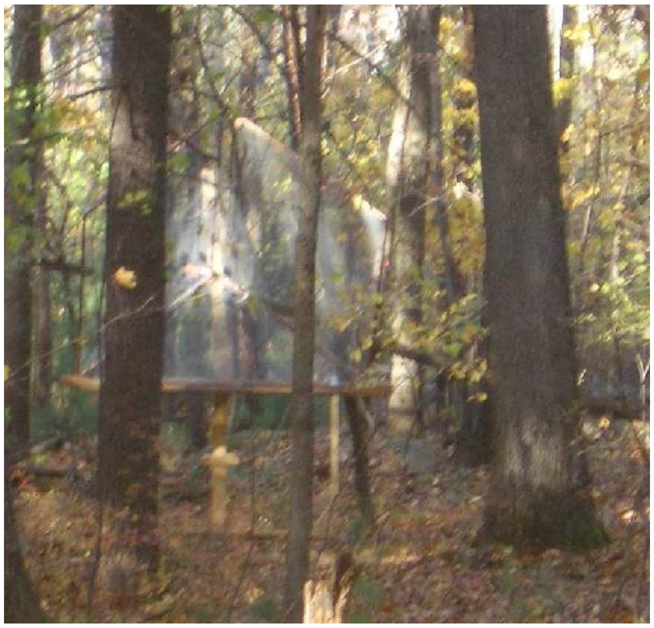


Figure 2.1: The target (a trihedral reflector)

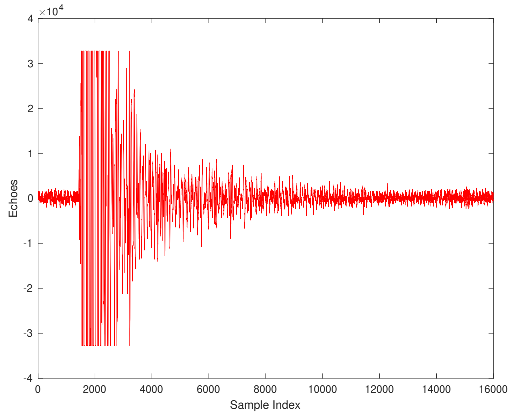
For the data utilized in this work, 16,000 samples are included in each data collection. In Figure. 2.2a and Figure. 2.2b we plot the received echoes without target and the received echoes with target on range respectively. The target's presence around 14,000 sample.

For clarity, we enlarge the relevant area in Figure. 2.2 from sample 13,000 to 14,999 as shown in the first two subfigures in Figure. 2.3. The difference is shown in the third one, from which we can't tell if there is a target. These collections as poor signal quality.

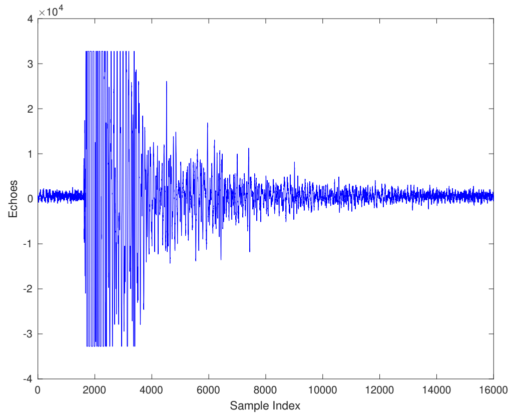
2.3 Stacked Autoencoder

An autoencoder (AE) is an unsupervised learning algorithm [30]. It is a neural network that learns to generate its output which is almost close to its input.

As shown in Figure. 2.4, an AE consists of two parts, the encoder, which computes a latent representation of the input, and the decoder, which reproduces the



(a)



(b)

Figure 2.2: Received echoes used in this chapter. (a) Without target. (b) Target on range.

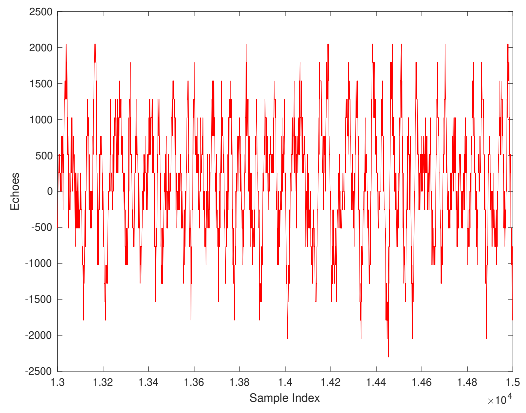
original input from the latent representation. Define the encoder parameters as ϕ and the decoder parameters as ψ, i.e.,

$$\phi : \mathcal{X} \rightarrow \mathcal{F} \tag{2.1}$$

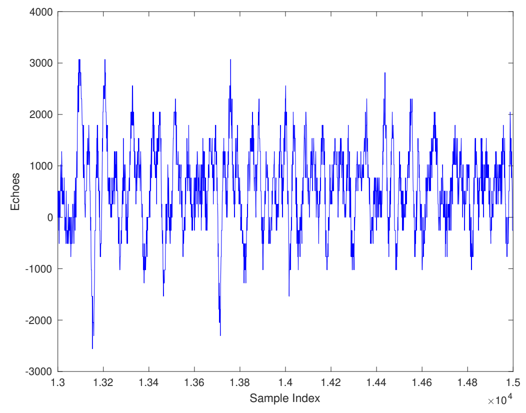
$$\psi : \mathcal{F} \rightarrow \mathcal{X} \tag{2.2}$$

$$\phi, \psi = \underset{\phi,\psi}{\arg\min} \, \|X - (\psi \circ \phi)X\|^2 \tag{2.3}$$

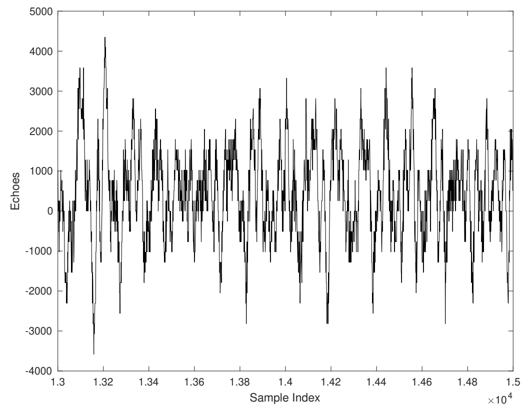
where $\mathcal{X}$ is the data space, $\mathcal{F}$ is the latent space and $X \in \mathcal{X}$.



(a)



(b)



(c)

Figure 2.3: Expanded view of received echoes used in this chapter from samples 13,000 to 14,999. (a) Without target. (b)Target on range. (c) Differences between echoes.

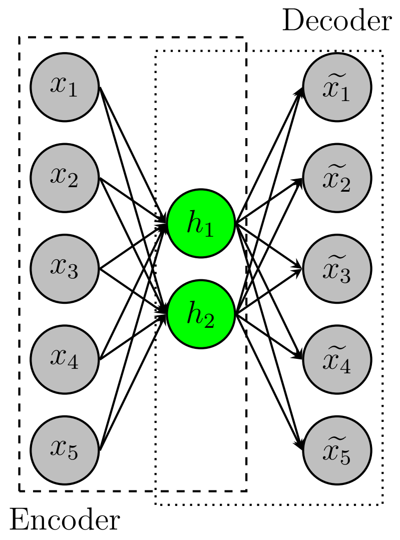


Figure 2.4: An illustration of an autoencoder (AE).

Without loss of generality, we assume that the data and the latent spaces are real-valued with dimension d and p, respectively. The encoder takes the input $\mathbf{x} \in \mathbb{R}^d = \mathcal{X}$ and maps it to $\mathbf{h} \in \mathbb{R}^p = \mathcal{F}$:

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.4}$$

where σ is an activation function, $\mathbf{W}$ is a weight matrix and $\mathbf{b}$ is a bias vector. The decoder maps $\mathbf{h}$ to the reconstruction $\mathbf{x}'$:

$$\mathbf{x}' = \sigma'(\mathbf{W}'\mathbf{h} + \mathbf{b}') \tag{2.5}$$

where σ' is an activation function, $\mathbf{W}'$ is a weight matrix and $\mathbf{b}'$ is a bias vector.

When $\mathbf{x} \approx \mathbf{x}'$, it is considered that the trained AE reconstructs the input. The cost function could be defined as follows:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 + \sum_{i,j,k} (\omega_{jk}^{(i)})^2 \tag{2.6}$$

where $\omega_{jk}^{(i)}$ is the connection weights between the j-th neurons of layer i and the k-th neurons of layer $i+1$. The first term (mean squared errors) is the reconstruction error, and the second term (weight decay) is a regularizing penalty which is typically included to prevent overfitting.

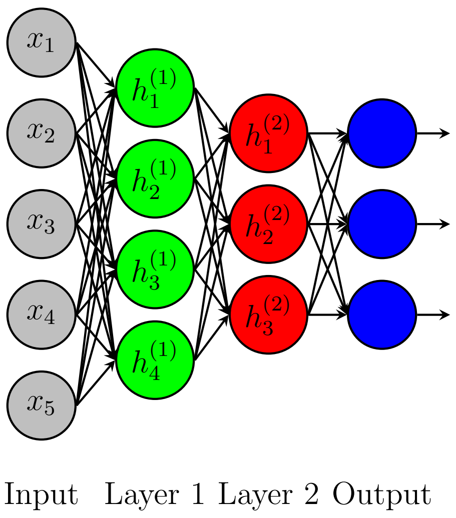


Figure 2.5: An illustration of a stacked autoencoder (SAE).

As shown in Figure. 2.5, SAE is a neural network structure stacking multiple layers of autoencoders in which the output of each layer is connected to the inputs of the successive layer. By using greedy layer-wise training [31], we could obtain the optimal parameters for a SAE. The training process could be divided into following steps:

1. Train the first autoencoder using the raw input data and acquire a vector consisting of the latent feature.
2. Use the latent vector from the previous layer as the input of the next layer, and repeat this step until the training is completed.
3. Use the backpropagation (BP) algorithm to minimize the cost function and update the weights with the training set to achieve fine tuning.

2.4 Sense-Through-Foliage Target Detection Using Signal Radar and SAE

In this section, we present the results of Sense-Through-Foliage target detection using signal radar and SAE. The data used here is present in Figure 2.3 from 2 positions. Each position has 2 sample ranges, i.e. 13500 to 14499, and 13000 to 14999. The number of hidden layers is two and the number of nodes in the hidden layers are 1000-100, and 2000-100, for different sample ranges. Before network training, echoes are pre-processed to fit the need of input.

Table 2.1 and 2.2 show the confusion matrix of sense-through-foliage target detection results in two different positions for different sample ranges.

From both tables, we could find the SAE scheme could finish the target detection mission and the detection performance in position 1 is better than position 2. Compare table 2.1 with 2.2, we might draw a conclusion that the large input size could increase the detection rate.

Table 2.1: Sample range from 13500 to 14499

position 1			position 2		
	NO	YES		NO	YES
no target	0.9360	0.0640	no target	0.9040	0.0960
target	0.0293	0.9707	target	0.2053	0.7947

Table 2.2: Sample range from 13000 to 14999

position 1			position 2		
	NO	YES		NO	YES
no target	0.9707	0.0293	no target	0.9107	0.0893
target	0.0133	0.9867	target	0.1560	0.8440

2.5 Sense-Through-Foliage Target Detection Using RSN and SAE

From Table 2.1 and 2.2, we found that the detection performance in position 2 was under satisfaction. Assuming that the measurements are independent, we use some diverse combination techniques in the radar sensor network (RSN) to improve the quality of the received signal [32, 33]. In the RSN, each radar can provide their pulse parameters such as timing to their clusterhead radar, and the clusterhead radar can combine the echos (RF returns) from the target and clutter.

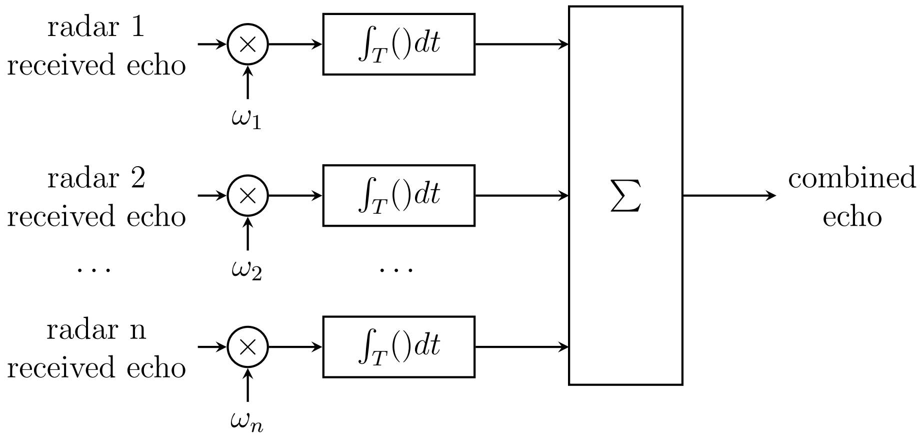


Figure 2.6: RAKE structure.

Therefore, we propose to use a RAKE structure to control poor signal quality target detection problem, as illustrated by Figure. 2.6 . Because uncorrelated radar measurements may experience different attenuation levels, the RAKE structure is an effective diversity combining method. The echoes from the radars of different cluster members are combined by the cluster leader.

In RAKE structure used in this chapter, two different diversity combing schemes are under consideration: one is equal gain combing; the other one is maximum ratio combining. Suppose the number of radar echoes is R.

The equal gain combined signal has following formula:

$$x_{eq}(n) = \frac{1}{N}\sum_{i=1}^{R} x_i(n) \tag{2.7}$$

The power of each echo $x_i(n)$ is $E_i = \text{var}(x_i(n)) + [\text{mean}(x_i(n))]^2$. Construct weight coefficient ω_i

$$\omega_i = \frac{E_i}{\sum_{i=1}^{R} E_i} \tag{2.8}$$

The combined signal $x_{\text{MRC}}(n)$ through the method of maximum ratio:

$$x_{\text{MRC}}(n) = \sum_{i=1}^{R} \omega_i x_i(n) \tag{2.9}$$

In our database, totally 70 radar echoes can be used to construct the Rake structure receiver. We choose $R = 3$, radar echoes in the database and 50 Monte Carlo simulations are performed at each combing level. Table 2.3 and 2.4 show the confusion matrix of sense-through-foliage target detection results in position 2 using two different combination schemes.

Table 2.3: Sample range from 13500 to 14499 in position 2

single radar		
	NO	YES
no target	0.9040	0.0960
target	0.2053	0.7947
equal combine		
	NO	YES
no target	0.9916	0.0084
target	0.0428	0.9672
MRC		
	NO	YES
no target	0.9976	0.0024
target	0.0172	0.9828

From Table 2.3 and 2.4, we find the usage of RSN and the RAKE structure could improve the detection level drastically. And maximum ratio combing method performs better than the method of equal gain combing in both sample ranges. However, when the sample range is 13000 to 14999, the target detection accuracy (0.9788) under maximum ratio combing method is not larger than when the sample range is 13500 to 14499 (0.9828), which is different from the single radar case. The main reason is that when the RSN is using maximum ratio combing, the clutters existing in different sample ranges are all considered into the combined echoes, which compromises the final detection performance.

Table 2.4: Sample range from 13000 to 14999 in position 2

single radar		
	NO	YES
no target	0.9107	0.0993
target	0.1560	0.8440
equal combine		
	NO	YES
no target	0.9984	0.0016
target	0.0252	0.9748
MRC		
	NO	YES
no target	1	0
target	0.0212	0.9788

2.6 Conclusion

This chapter presented a SAE based approach for sense-through-foliage target detection. SAE could uncover the essential information and extract deep features of sense-through-foliage radar echoes, especially when the received echoes are in the poor signal quality. The experimental results showed that SAE could achieve a high detection accuracy. However, in some positions the performance was under satisfaction. A RAKE structure using in RSN by combining echoes from different cluster-member radars is utilized for preprocessing before inputting the neural network. The results of simulation indicate that integrating different radar echoes could improve the detection accuracy significantly.

CHAPTER 3

A Statistical Approach for Neural Network Pruning with Application to Internet of Things

3.1 Introduction

In the internet of Things (IoT) realm, sensors and actuators seamlessly integrate with the environment [34], enabling cross-platform information flow for environmental metrics, while numerous connected devices generate massive data, offering convenience but also high latency [35]. However, applications such as vehicle-to-vehicle (V2V) communication which enhances the traffic safety by automobile collaboration, are highly latency-sensitive and security-sensitive [36]. Edge computing offers vast potential for consumers and entrepreneurs by bringing data processing closer to end users, enhancing response times, bandwidth availability, privacy, and alleviating information security threats [37, 38].

Even though chip giants are integrating more and more AI accelerators into their design for the IoT devices [39, 40], the massive number of parameters and the huge amount of computation would bring horrible experience to the consumers when Deep Neural Networks (DNNs) are employed in their devices [41]. To alleviate such kind of problems, researchers have made efforts in many directions, which could be mainly categorized into two types: unstructured ones and structured ones.

Pruning the individual weights whose values are close to 0 is one way to downsize the number of parameters in DNNs [42, 43]. This kind of unstructured method could wind up as a sparse structure and maintain the original performance. However, the random and unpredictable positions of the remaining weights bring the burden of extra records of themselves and make this method unable to utilize AI accelerators effectively [44].

By contrast, as shown in Figure. 3.1, structured methods remove the weights at higher levels and avoid the problem brought by unstructured ones. Filter (channel) pruning (FP) based methods prunes weights at the level of filters or channels [45–47]. Usually, a traditional FP-based method needs to follow the "Train, Prune, Fine-tune" pipeline. Group-wise pruning based methods delete the weights at the identical position among all the filters in a certain layer [48]. However, these approaches ignore the assumption of filters' independence. Stripe-wise pruning (SWP) based methods trim all the weights laid in some stripes of certain filters [49]. The proposed method introduced the concept of filter skeleton (FS). During the training, when some values on FS are under a certain threshold, the corresponding stripes can be pruned.

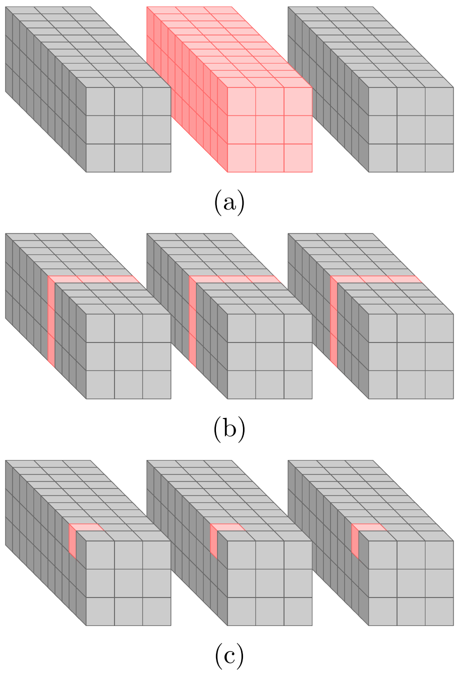


Figure 3.1: Different types of pruning. (Red parts were pruned.) (a) Filter-wise. (b) Channel-wise. (c) Group-wise.

However setting an absolute threshold sometimes could not express the relative importance of each stripe in a filter. To resolve this problem, in this work, we put forward a new method, using the statistical properties of the weights located on each stripe, to learn the importance between those stripes in a filter. The intuition of this method is triggered by the process during stripe wise convolution and the properties of normal distributions. Our principal contributions in this paper could be summarized as follows:

- New threshold determination approach for SWP: The research proposes a new method for determining which weights in a neural network can be pruned without sacrificing accuracy. Our pruned VGG16 achieves results comparable to the existing model, with a 4-fold reduction in parameters and only a 0.4% decrease in accuracy.
- Stable theoretical basis: The proposed method is based on sound theoretical principles, making it more trustworthy and easier to understand and apply.
- Deployment of different deep layers on edge devices: The effectiveness of the proposed approach is tested on different neural network architectures (VGG11, VGG13, VGG16, and ResNet56) and evaluated on edge devices with limited computational resources.

In this chapter, we investigated pruning in deep learning network models. As a reminder, this chapter is organized as follows. In Section 3.2, we briefly review related work. In Section 3.3, we present our method as well as the theoretical framework behind it. Section 3.4 explains the experimental details and data processing. In Section 3.5, we demonstrate comparisons between our method and the original method, as well as exhibit the performance of our method deployed on edge devices. In Section 3.6, we discuss the implications of our findings. Finally, concluding remarks are provided in Section 5.

3.2 Related work

Neural network pruning algorithms have undergone decades of research [50]. As mentioned in Section 3.1, these algorithms could be mainly categorized into two types, i.e., unstructured ones and structured ones.

Unstructured pruning methods prune individual weights based on the importance of themselves. For example, by using the second-order derivatives of the error function, Optimal Brain Damage and Optimal Brain Surgery proposed to remove unimportance weights from a trained network [42,43]. Deep Compression compressed neural networks by pruning the unimportant connections, quantizing the network, and applying Huffman coding [51]. With Taylor expansion that approximates the change in the cost function, [52] pruned convolutional kernels to enable efficient inference and could handle the transfer learning tasks effectively. Lookahead pruning scheme, a magnitude-based method, minimized the Frobenius distortion of multi-layer operation and avoids tuning hyper-parameters [53]. A major downside of the unstructured methods is the sparse matrix and the relative indices after pruning, which leads to the complexity and inefficiency on hardware [44].

Structured methods prune weights in a predictable way. [45] pruned unimportant filters with L_1 norm. [54] pruned filters based on statistics information computed from its next layer, not the current layer. [46] pruned channels by LASSO regression. By using scaling factors from batch normalization layers, [47] removed unimportant channels. [48] revisited the idea of brain damage and extended it to group wise, obtaining the sparsities in new neural network. [55] put forward a structured sparsity learning (SSL) approach. With group Lasso regularization, SSL could learn a compressed structure, including filters, channels and filter shapes. To the best of our knowledge, one recent study [49] proposed a stripe-wise pruning based methods by introducing filter skeleton to learn the shape of filters and then performed pruning on the stripes according to the corresponding values of the filter skeleton. However setting an absolute threshold sometimes is unable to distinguish the importance of the convolution result for one stripe from the other result for corresponding stripes.

The comparisons of the two methods are summarized in Table. 3.1.

Table 3.1: Unstructured and structured pruning methods.

Methods	**Advantages**	**Disadvantages**
Unstructured [42–44]	Sparse structure original performance	High complexity and inefficiency on hardware
Structured [45–48, 55]	Prune weights in a predictable way	May not maintain unpruned performance

3.3 The proposed method

In this section, we begin with introducing stripe wise convolution (SWC), then analyze our threshold determination with stripe weight combination based on the properties of normal distributions. Furthermore, we discuss our approach for stripe wise pruning (SWP).

3.3.1 Stripe Wise Convolution

In l-th convolution layer, suppose the weight 4-D matrix W is of size $\mathbb{R}^{N\times C\times K\times K}$, where N, C and K are the numbers of filters, the channel dimension and the kernel size, respectively.

Let $x^{l}_{c,h,w}$ be one point of feature map in the l-th layer and $x^{l+1}_{n,h,w}$ be the convolution result in the $l+1$-th layer. Mathematically, the standard convolution in a neural network could be written as (3.1). We modify the calculation order as (3.2) to stripe wise convolution. These two types of convolution are illustrated in Figure. 3.2 and Figure. 3.3, respectively.

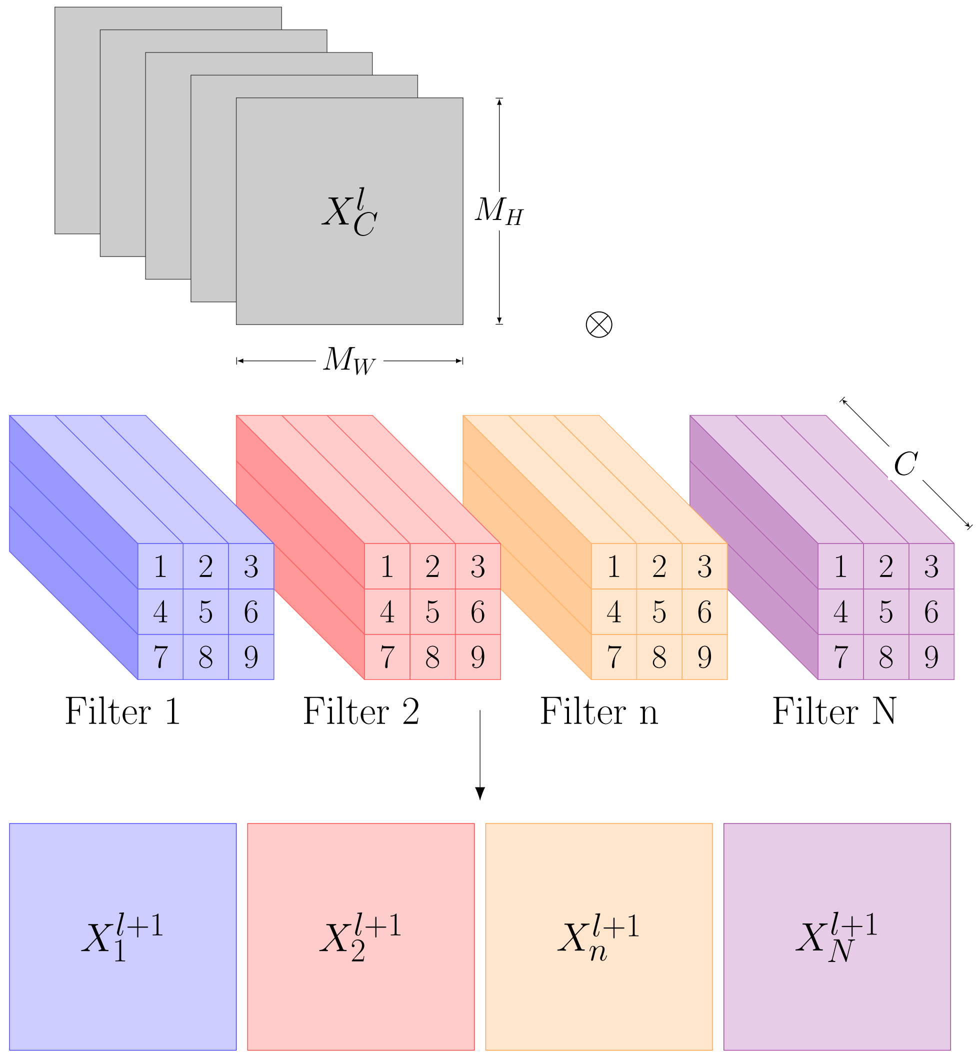


Figure 3.2: Standard convolution (The kernel size of the filter is 3).

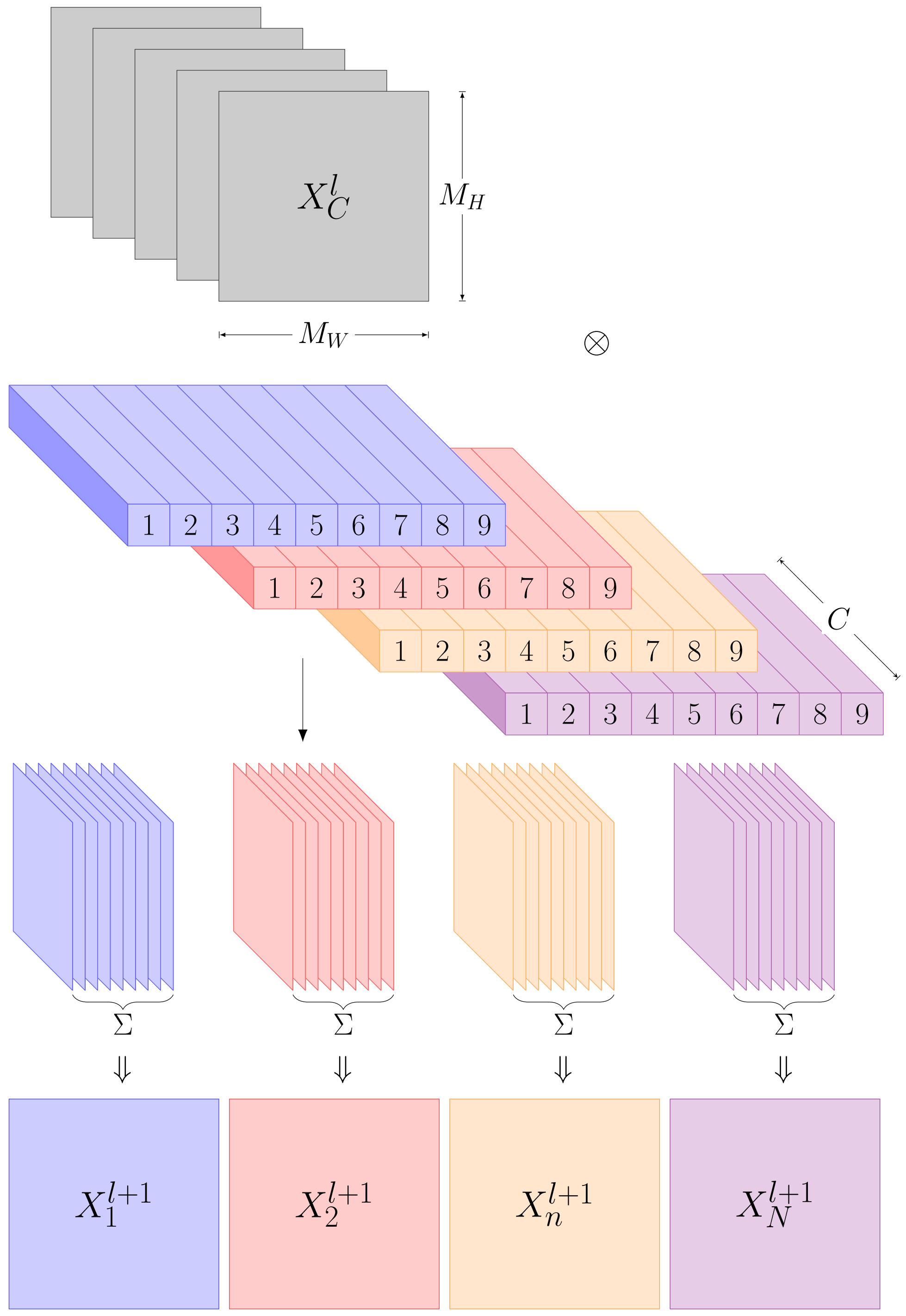


Figure 3.3: Stripe wise convolution (The kernel size of the filter is 3).

$$x^{l+1}_{n,h,w} = \sum_{c}^{C}\sum_{i}^{K}\sum_{j}^{K}(w^{l}_{n,c,i,j} \times x^{l}_{c,h+i-\frac{K+1}{2},w+j-\frac{K+1}{2}}) \tag{3.1}$$

$$= \sum_{i}^{K}\sum_{j}^{K}(\sum_{c}^{C} w^{l}_{n,c,i,j} \times x^{l}_{c,h+i-\frac{K+1}{2},w+j-\frac{K+1}{2}}) \tag{3.2}$$

$$= \sum_{i}^{K}\sum_{j}^{K}(x^{l+1}_{n,h,w,i,j}) \tag{3.3}$$

$x^{l}_{c,p,q} = 0$, when $p < 1$ or $p > M_H$ or $q < 1$ or $q > M_W$. M_H is the height of the feature map, while M_W represents the width.

From Figure. 3.3, we could find that in stripe wise convolution, the convolution result of individual filter is the summation of the convolution result of the stripes which belongs to this filter. One intuition is that if the convolution result of the stripe 1 is much smaller than the convolution result of the stripe 2, Stripe 1 could be pruned and Stripe 2 could be kept as shown in Figure. 3.4. The following part will prove it in a theoretical manner.

3.3.2 Theoretical Analysis

Batch normalization (BN) is widely used in a neural network. This method could make DNN faster and more stable [56]. In one filter, suppose B is a mini-batch of size m, i.e., $B = \{a_1, ...a_m\}$. BN layer processes these following transformation steps:

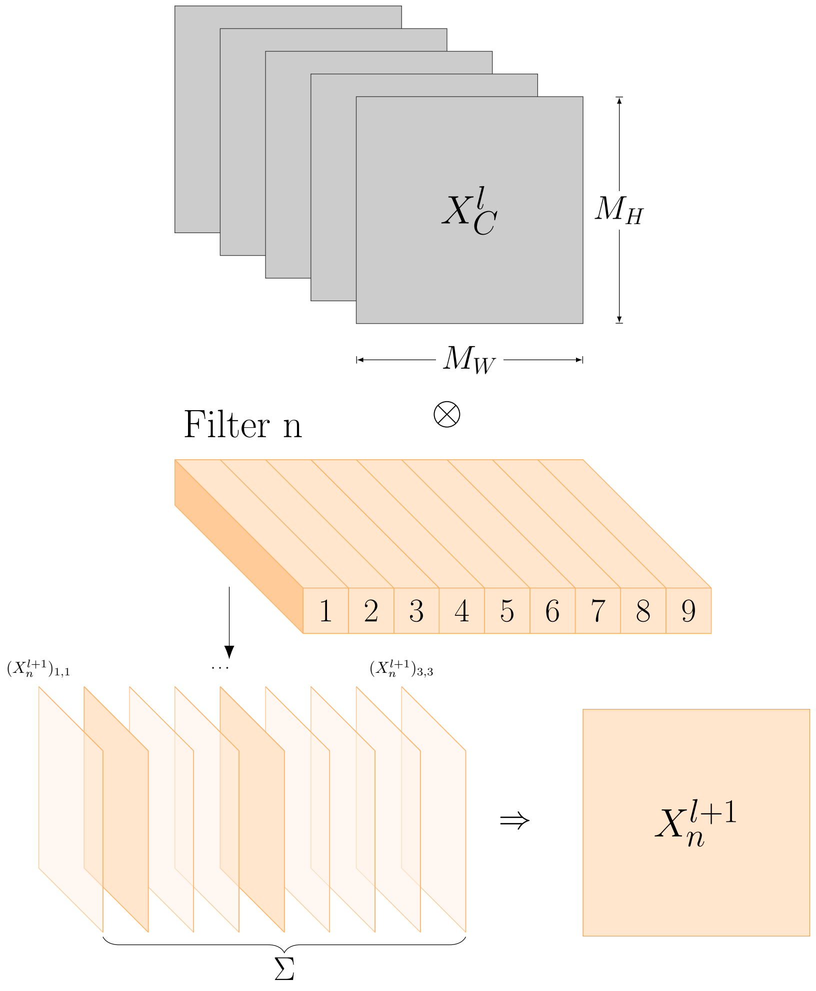


Figure 3.4: Stripe wise convolution (Single filter case). The squares with dark orange indicate they have larger stripe convolution results than the light orange ones, which means the corresponding stripes could be remained during pruning.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} a_i \tag{3.4}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(a_i - \mu_B)^2 \tag{3.5}$$

$$\hat{a}_i = \frac{a_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{3.6}$$

$$x_i = \gamma \hat{a}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(a_i) \tag{3.7}$$

where μ_B and σ_B are the empirical mean and standard deviation of B. To resume the representation ability of the network, scale γ and shift β are learned during the whole process.

After transformation in the BN layer, in c-th channel of l-th layer, the input feature map could be

$$X_c^l \sim \mathcal{N}(\beta_c^l, (\gamma_c^l)^2). \tag{3.8}$$

When M_H is large, $(X_c^l)_{i,j} \sim \mathcal{N}(\beta_c^l, (\gamma_c^l)^2)$. From (3.2), we could get

$$X_n^{l+1} = \sum_i^K \sum_j^K (\sum_c^C w_{n,c,i,j}^l \times (X_c^l)_{i,j}) \tag{3.9}$$

Assuming all data is independently identically distribution, with the properties of normal distribution [57], we have

$$X_n^{l+1} \sim \mathcal{N}(\mu_n^{l+1}, (\sigma_n^{l+1})^2) \tag{3.10}$$

where

$$\mu_n^{l+1} = \sum_i^K \sum_j^K (\sum_c^C w_{n,c,i,j}^l \beta_c^l) \tag{3.11}$$

$$(\sigma_n^{l+1})^2 = \sum_i^K \sum_j^K (\sum_c^C (w_{n,c,i,j}^l)^2 (\gamma_c^l)^2) \tag{3.12}$$

To reduce the number of parameters $w_{n,c,i,j}^l$ and avoid the value of μ_n^{l+1} in (3.11) change, we introducing an importance indicator $Q_{n,i,j}^l$ to the output of convolution of each stripe and have the following loss function.

$$L_n = \mathrm{loss}(\mu_\mathrm{n}^\mathrm{l+1}, \sum_\mathrm{i}^\mathrm{K} \sum_\mathrm{j}^\mathrm{K} \mathrm{Q}_\mathrm{n,i,j}^\mathrm{l} (\sum_\mathrm{c}^\mathrm{C} \mathrm{w}_\mathrm{n,c,i,j}^\mathrm{l} \beta_\mathrm{c}^\mathrm{l})) + \alpha \mathrm{g}_\mathrm{n}(\mathrm{Q}) \tag{3.13}$$

where $g_n(Q) = \sum_i^K \sum_j^K \left| Q_{n,i,j}^l \right|, Q_{n,i,j}^l = 1$ or 0.

Let

$$s^l_{n,i,j} \triangleq \sum_c^C w^l_{n,c,i,j}. \tag{3.14}$$

If we assume $\beta^l_1 = \beta^l_2 \cdots = \beta^l_c = \beta^l$, combing with (3.11), (3.13) could be written as

$$L_n = \mathrm{loss}(\beta^l \sum_a^K \sum_b^K \mathrm{s}^l_{n,a,b}, \beta^l \sum_i^K \sum_j^K \mathrm{Q}^l_{n,i,j} \mathrm{s}^l_{n,i,j}) + \alpha \mathrm{g_n}(\mathrm{Q}) \tag{3.15}$$

which can be further written as

$$\begin{aligned} L_n &= \mathrm{loss}(1, \frac{\sum_i^K \sum_j^K \mathrm{Q}^l_{n,i,j} \mathrm{s}^l_{n,i,j}}{\sum_a^K \sum_b^K \mathrm{s}^l_{n,a,b}}) + \alpha' \mathrm{g_n}(\mathrm{Q}) \\ &= \mathrm{loss}(1, \sum_i^K \sum_j^K \mathrm{Q}^l_{n,i,j} \mathrm{T}^l_{n,i,j}) + \alpha' \mathrm{g_n}(\mathrm{Q}) \end{aligned} \tag{3.16}$$

where

$$T^l_{n,i,j} = \frac{s^l_{n,i,j}}{\sum_a^K \sum_b^K s^l_{n,a,b}} \tag{3.17}$$

Obviously,

$$\sum_i^K \sum_j^K T^l_{n,i,j} = 1, 0 \le T^l_{n,i,j} < 1 \tag{3.18}$$

To minimize (3.16), we could set $Q^l_{n,i,j} = 0$ to those $T^l_{n,i,j}$ close to 0, which means the corresponding stripes will be pruned.

$T^l_{n,i,j}$ could be used to describe the relative importance of stripe$_{i,j}$ in filter$_n$. When $T^l_{n,i,j} \to 1$, stripe$_{i,j}$ contributes more than other stripes. When $T^l_{n,i,j} \to 0$, stripe$_{i,j}$ contributes less than other stripes and could be pruned.

3.3.3 Method Description

Before setting a threshold for $T^l_{n,i,j}$ to prune stripes, we need to impose regularization on the whole neural network to achieve sparsity. This method could avoid so-called "Train, Prune, Fine-tune" pipeline. The regularization on the FS will be

$$L = \sum_{(x,y)} \mathrm{loss}(\mathrm{f}(\mathrm{x}, \mathrm{W}), \mathrm{y}) + \alpha \mathrm{g}(\mathrm{W}) \tag{3.19}$$

where α adjusts the degree of regularization. $g(W)$ is L_1 norm penalty on W and could be written as:

$$g(W) = \sum_{l=1}^{L}(\sum_{n=1}^{N}\sum_{c=1}^{C}\sum_{i=1}^{K}\sum_{j=1}^{K}|W_{n,c,i,j}^{l}|) \tag{3.20}$$

To avoid using sub-gradient at non-smooth point, instead of the L_1 penalty, we deploy the smooth-L_1 penalty [58].

Summarize the proposed algorithm below.

Algorithm 1 The Proposed Algorithm

```
1: The training dataset D, the unpruned neural network model, the regularization
   factor $\alpha$, stripe pruning threshold $T$, the total iteration number $i_{\text{total}}$.
2: Initialize the FS with all-one matrix.
3: Initialize the iteration number $i = 0$.
4: repeat
5:   Have the stripe wise convolution by (3.2);
6:   Update weights $W$ by (3.19) with the smooth-$L_1$ penalty;
7:   Calculate $T_{n,i,j}^{l}$ by (3.17);
8:   Prune stripes with $T_{n,i,j}^{l}$ smaller than $T$;
9:   $i = i + 1$;
10: until $i = i_{\text{total}}$;
11: Output the pruned model;
```

3.3.4 Computational Complexity

Assuming the number of weight parameters to be P, the computational cost for implementing the smooth-L_1 penalty is $O(P)$. Assuming that the computational cost of each epoch of neural network training is $O(Q)$, the process of pruning costs $O(e \cdot Q)$

computations, where e denotes the percentage of pruning in the neural network of an epoch. Therefore, the computational cost for each epoch is $O(P + e \cdot Q)$. If K denotes the total number of iterations, the computational complexity of our method is $O(K(P + e \cdot Q))$.

3.4 Experiments

In order to assess the performance of the proposed model and confirm its effectiveness, we carry out experiments on two datasets including CIFAR-10 and bearings dataset from the Case Western Reserve University.

3.4.1 Experiments on CIFAR-10

Our method is implemented using the publicly available Torch [59].

Dataset and Model: CIFAR-10 [60] is one of the most popular image collection data sets. This dataset contains 60K color images from 10 different classes. 50K and 10K images are included in the training and testing sets respectively. By adopting CIFAR-10, we evaluated the proposed method mainly on VGG [61] and ResNet56 [11]. VGG16 and ResNet56 are the networks used to demonstrate the performance before and after network pruning. VGG11 and VGG13, which have sizes more compact than VGG16, are then deployed to make comparisons with VGG16 in terms of the total time which is required for classifying 3270 image patches of size 224×224, i.e. inference time.

Baseline Setting: We train the model using mini-batch size of 64 for 100 epochs. The initial learning rate is set to 0.1, and is divided by 10 at the epoch 50. Random crop and random horizontal flip are used as data augmentation for training images. Image is scaled to 256×256. Then a 224×224 part is randomly cropped from the scaled image for training. The testing is the center crop with 224×224.

Experiment environment: NVidia 1080-TI and Intel Core i5-8500B are selected as two different computing platforms representatives of the server and the edge device, respectively. The first is a GPU which has high computation ability, however needs communication with sensors and actuators. The second is a CPU to represent the restricted computer power of an edge device.

3.4.2 Experiments on Bearings Dataset

Rotating element bearings (REBs) are among the most common parts in rotating equipment, and their malfunction is a leading cause of machinery failure. The Case Western Reserve University (CWRU) Bearing Data Center's dataset has emerged as a benchmark in the domain of bearing diagnostics [62]. We will use the data from this center for the following part of the experiment. The fundamental configuration of the testing apparatus is displayed in the Figure. 3.5.

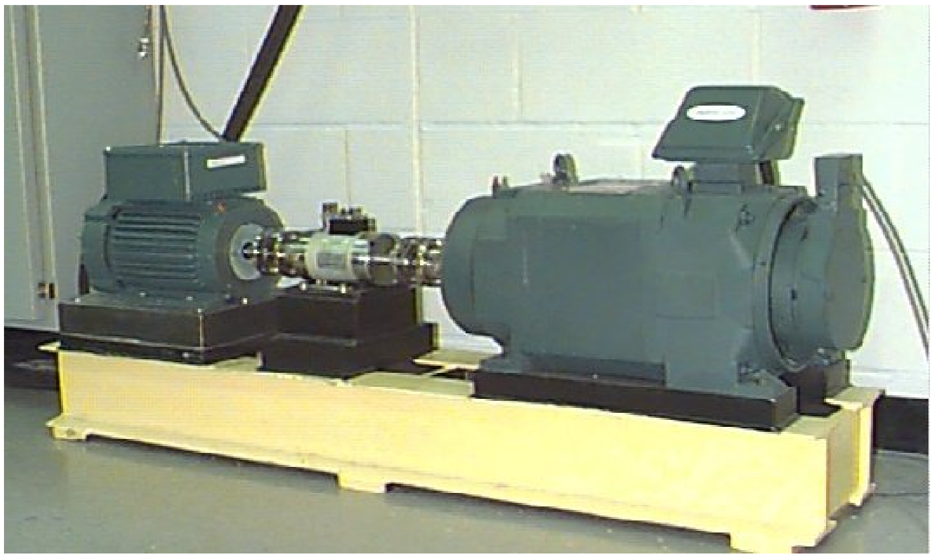


Figure 3.5: CWRU bearing test rig. (The components of the test stand include a 2 hp motor on the left, a torque transducer/encoder in the center, and a dynamometer on the right.)

The testing setup includes a 2 hp reliance electric motor that powers a shaft equipped with a torque transducer and encoder. Torque is exerted on the shaft via a dynamometer and electronic control system.

3.4.2.1 Symmetrized Dot Pattern

Symmetrized Dot Pattern (SDP) is a technique used for visual representation of acoustic and vibration signals to quickly identify any faulty condition of the system [63]. This technique transforms the time-domain signal into a scatter plot with sextuple symmetry.

The time-domain signal is $S = \{s_1, s_2..., s_i..., s_D\}$, s_i is the ith sampling point. Then, given a specific index i, the value s_i could be transformed into its corresponding polar coordinate space, represented by the expression $P(r(i), \theta(i), \phi(i))$.

$P(r(i))$ is the radius, which could be expressed as follows:

$$r(i) = \frac{s_i - s_{\min}}{s_{\max} - s_{\min}} \tag{3.21}$$

where $s_{\max}$ and $s_{\min}$ are the maximum and minimum amplitudes values of the time-domain signal sequence, respectively.

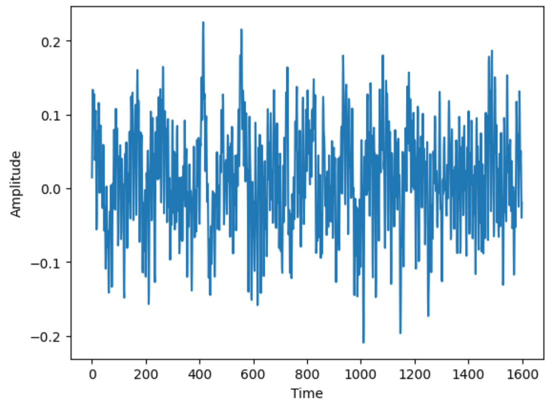
$\theta(i)$ is the clockwise rotation angle of the initial line, while $\phi(i)$ represents the counterclockwise one. These angles could be expressed as follows:

$$\theta(i) = \phi - \frac{s_i - s_{\min}}{s_{\max} - s_{\min}}\zeta \tag{3.22}$$

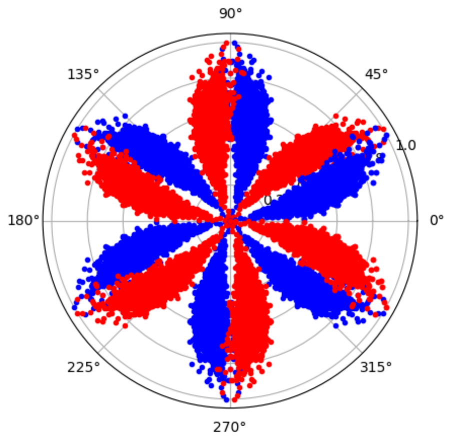
$$\phi(i) = \phi + \frac{s_i - s_{\min}}{s_{\max} - s_{\min}}\zeta \tag{3.23}$$

where ϕ is the initial rotation angle ($\phi = 60m + \phi_0, m = 1, ..., 6$, ϕ_0 is a starting term that can rotate the plot). ζ is the amplification coefficient.

The time-domain signal S can be transformed into its corresponding polar plot by marking all θ points as red and all ϕ points as blue, as shown in Figure. 3.6.



(a)



(b)

Figure 3.6: Typical time-domain signal for SDP technique (a) and corresponding SDP plot (b).

3.4.2.2 Data categories and preprocess

Four different operational conditions were tested at varying bearing loads (0-3 hp) to collect vibration signals. Each operational condition contains datasets representing rolling element faults and inner and outer race faults. In addition to the normal condition, the SDP for these three operating faults are shown in Figure. 3.7. The outer race faults are further classified into three categories based on their location relative to the load zone: ‘centered’ (fault at 6 o’clock position), ‘orthogonal’ (fault at 3 o’clock position), and ‘opposite’ (fault at 12 o’clock position). Combining with fault size (0.007 to 0.028 in.), we choose twelve fault categories as shown in Table. 3.2. Adding one healthy state resulted in a total of thirteen different bearing categories.

In order to test the performance of the proposed framework under various working environments, several sub-datasets are created as follows.

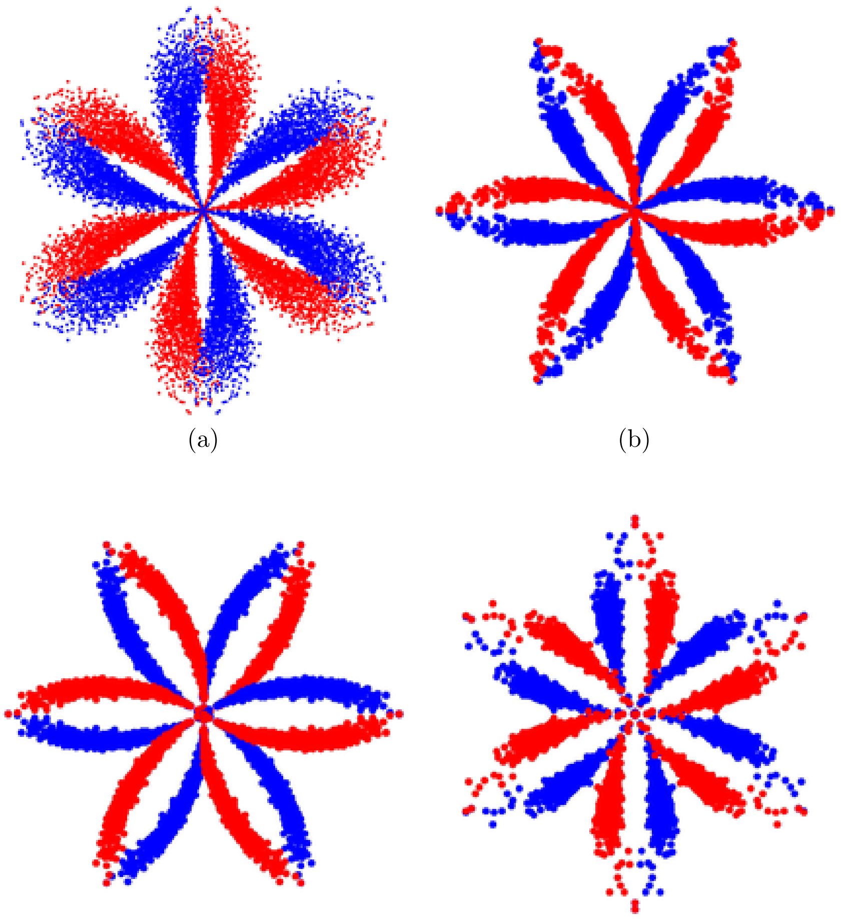


Figure 3.7: SDP (a) Normal case. (b) Inner race fault. (c) Outer race fault. (d) Rolling element fault.

- Training data and testing data are both from vibration signals under the same working load.
  - e.g. Training data and testing data are both from vibration signals under working load of 0 hp.
- Training data come from vibration signals under a certain working load while testing data are from different working loads.

Table 3.2: Twelve fault categories.

Fault Diameter	Inner Race	Ball	Outer Race Position Relative to Load Zone centered @6	orthogonal @3	opposite @12
0.007”	IR007	B007	OR007@6	OR007@3	OR007@12
0.014”	IR014	B014	-	-	-
0.021”	IR021	B021	OR021@6	OR021@3	OR021@12

– e.g. Training data come from vibration signals under working load 0 hp while testing data are from working load of 3 hp.

Therefore, there are 16 sub-datasets, with each bearing sub-dataset treated as a 13-class classification task for fault diagnosis. To facilitate the demonstration, we use Dij to represent a sub-dataset, where the training set is derived from the workload of i hp, and the testing set is derived from the workload of j hp. Each sub-dataset consists of 500 samples for each machine state, resulting in a total of 6,500 samples for the 13 classes. Each sample is derived from a randomly cropped 1600-length time-domain signal, and augmented by rotating the image using a random value of ϕ_0 in the SDP transformation to expand the dataset.

The following steps are the same as those used with CIFAR-10.

3.5 Results

3.5.1 Results on on CIFAR-10

3.5.1.1 Comparing with the original SWP

To compare our method with the original SWP, we revisit the concept of filter skeleton (FS) from [49]. As mentioned before, in l-th layer, the weight W is of size $\mathbb{R}^{N\times C\times K\times K}$. Then the size of FS in this layer is $\mathbb{R}^{N\times K\times K}$. Each value in FS corresponds to a stripe in the filter. During training, the filters' weights are multiplied with FS. With I representing the FS, the stripe wise convolution could be written as

$$x^{l+1}_{n,h,w} = \sum_{i}^{K}\sum_{j}^{K} I^{l}_{n,i,j}(\sum_{c}^{C} w^{l}_{n,c,i,j} \times x^{l}_{c,h+i-\frac{K+1}{2},w+j-\frac{K+1}{2}}) \tag{3.24}$$

where $I^{l}_{n,i,j}$ is initialized with 1.

The regularization on the FS will be

$$L = \sum_{(x,y)} \mathrm{loss}(\mathrm{f}(\mathrm{x}, \mathrm{W} \odot \mathrm{I}), \mathrm{y}) + \alpha \mathrm{g}(\mathrm{I}) \tag{3.25}$$

where $\odot$ denotes dot product and α adjusts the degree of regularization. $g(I)$ is written as:

$$g(I) = \sum_{l=1}^{L}(\sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{K} |I^{l}_{n,i,j}|) \tag{3.26}$$

For convenience, in Table. 3.3 for the comparison on CIFAR-10, both the original method and our method use FS to train and prune the whole neural network. Both of them use the coefficient α of regularization, which is set to $1e-5$ and $5e-5$. The difference is that for the original method, pruning is based on the value in FS which corresponds to a stripe and for our method, pruning is based on $T^{l}_{n,i,j}$ which combines the weights located in a stripe. Regarding the choice of T, we used the value corresponding to the highest accuracy.

From the table, we could find both methods could reduce the number of parameters and the amount of computation (FLOPs) in a considerable volume without losing network performance. For the backbone is VGG16 situation, when $\alpha = 1e-5$, the number of parameters and the amount of computation of our method are larger than the original approach. This is because our method will keep at least one stripe in a filter, while the original approach might prune a whole filter. However, when $\alpha = 5e-5$, the original approach could not converge and our method could reach a

Table 3.3: Comparison with the original SWP on CIFAR-10.

Backbone	Metrics	Params	FLOPS	Accuracy
VGG16	baseline	14.76M	627.37M	93.76 %
	Original($\alpha = 1e-5$)	3.62M	350.28M	93.46 %
	Original($\alpha = 5e-5$)	could not converge		
	Ours ($\alpha = 1e-5$, $T = 0.0001$)	4.63M	385.49M	93.43 %
	Ours ($\alpha = 5e-5$, $T = 0.005$)	0.84M	126.03M	93.06 %
ResNet56	baseline	0.87M	251.50M	93.11 %
	Original($\alpha = 1e-5$)	0.60M	150.63M	93.41 %
	Ours ($\alpha = 5e-5$, $T = 0.001$)	0.23M	60.76M	92.96 %

high compression rate both in the number of parameters and the amount of computation. Our pruned VGG16 could achieve 95% reduction in memory demands.

For the backbone is Resnet56 situation, we present our result of $\alpha = 5e - 5$. To compare with the original approach's result of $\alpha = 1e - 5$, our method could see a large reduction in the number of parameters and the amount of computation while sacrificing a bit of accuracy. Our pruned Resnet56 could achieve 75% reduction in memory demands.

3.5.1.2 Ablation study

In our method, there are two decisive hyper-parameters in the neural network, the coefficient α of regularization in (3.25) and the weight combination threshold T in (3.17). As the outcomes of the experiment demonstrated in Table. 3.4, we display the effects of the hyper-parameters in pruning consequences. It could be noticed that $\alpha = 5e - 5$ and $T = 0.005$ holds an acceptable pruning ratio as well as test accuracy.

In Figure. 3.8, we show how many stripes in VGG16 on CIFAR-10 are kept after SWP. We could find that most stripes (around 85%) in the first and the last

Table 3.4: Different coefficient α and weight combination threshold.

α	1e-5			5e-5			
T	0.0001	0.001	0.01	0.0001	0.0005	0.001	0.005
Params (M)	4.63	4,17	2.89	0.78	0.80	0.79	0.84
FLOPS (M)	385.49	327.17	200.09	130.96	135.56	134.68	126.03
Accuracy (%)	93.43	93.28	92.99	92.79	92.86	92.96	93.06

layers are pruned. There are higher pruning ratios in the second half layers than the first half ones. The pruning ratio for this VGG16 neural network is 26.25%.

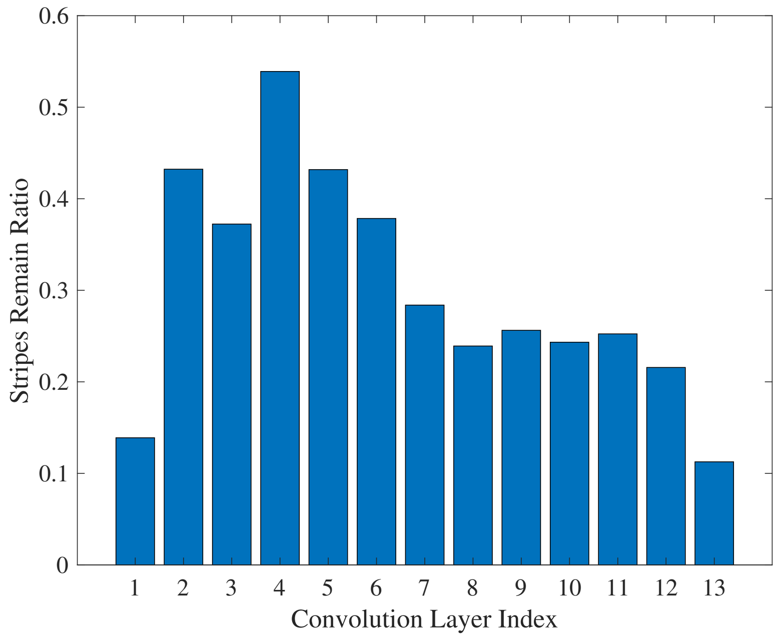


Figure 3.8: The ratio of remaining stripes in each layer.

3.5.1.3 Edge device performance

We further verify our approach in an edge device. As shown in Figure. 3.9, pruning is executed on the server as training consumes computing resources on learning the importance between the stripes and serval complete passes of the training dataset through the whole neural network. The pruned networks are then deployed on these two computing platforms to test results and get the inference time. The comparison is shown in Figure. 3.10-3.13. It should be noted that stripe wise convolution is not yet optimized in CUDA. Along with the increase in percentage of parameters pruned, the decline in inference time in servers is not quite clear. However, the inference time in edge device drops by half when $75 - 95\%$ of parameters are pruned.

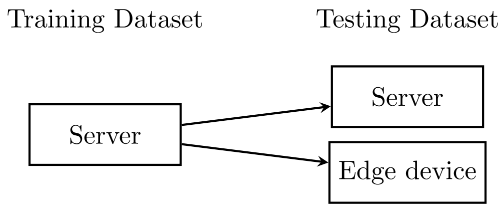


Figure 3.9: Experiment setup of edge device performance.

Separately speaking, Figure 3.10 shows the variation of the inference time when using pruned VGG16. On edge device, the inference time decreases from 195.1 (sec) to 77.9 (sec). For the server setup, the inference time decreases from 24.01 (sec) to 19.53 (sec). Figure 3.11 and Figure 3.12 show the variation of the inference time with pruned VGG13 and VGG11, respectively. Similar results of decrease in inference time could be observed on both VGG structures.

Figure 3.13 reports the results of using pruned ResNet56. On edge device, the inference time decreases from 277.7 (sec) to 100.5 (sec). For the server setup, the inference time decreases from 61.9 (sec) to 54.9 (sec).

Figure 3.14 compares the inference time of 3 types of VGG as well as ResNet56 when deployed on the edge device. Due to the reduction of the number of layers,

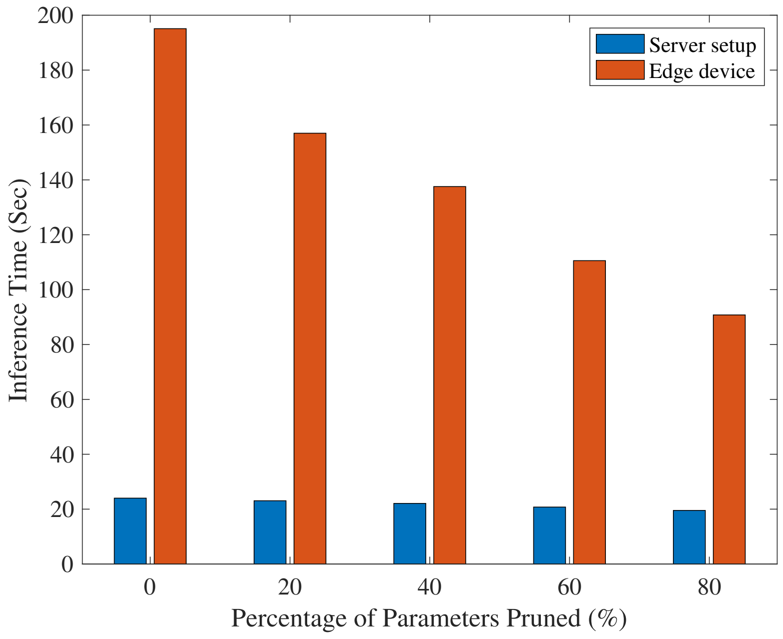


Figure 3.10: Inference Time Required for Pruned VGG16

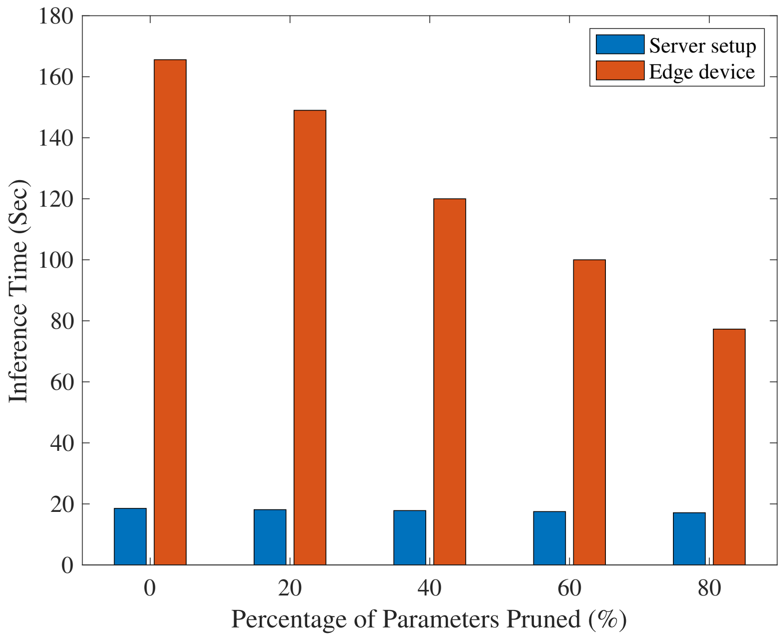


Figure 3.11: Inference Time Required for Pruned VGG13

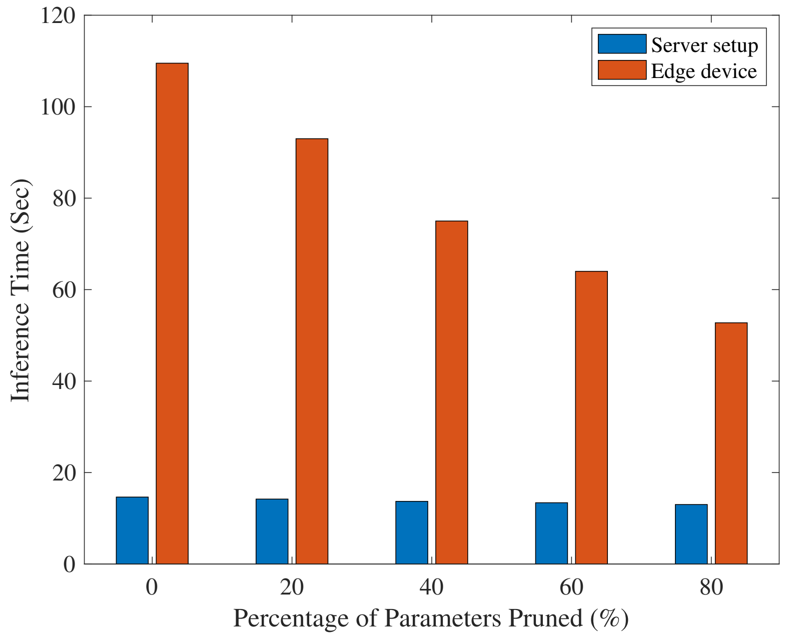


Figure 3.12: Inference Time Required for Pruned VGG11

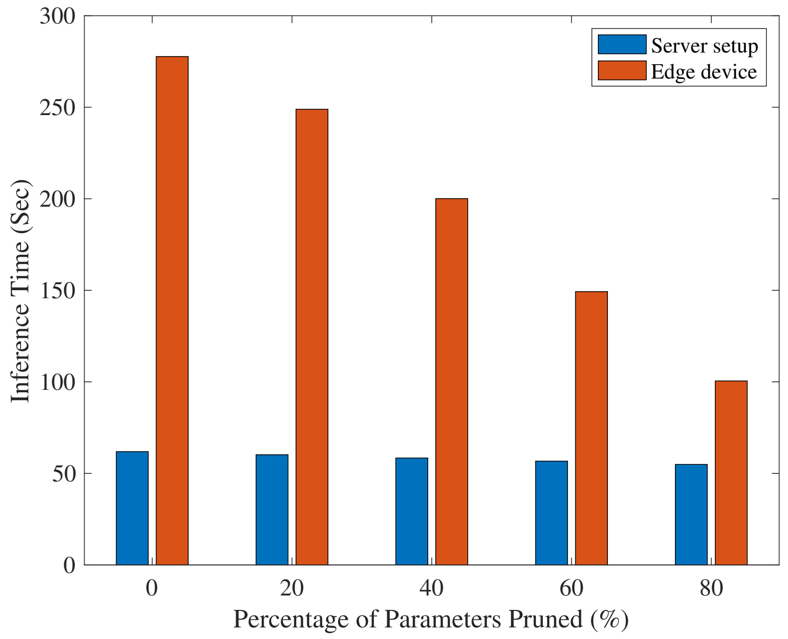


Figure 3.13: Inference Time Required for Pruned ResNet56

regardless of parameters pruned percentage, VGG11 could classify the most images at the same time in all pruned models, while ResNet56 classifies the least.

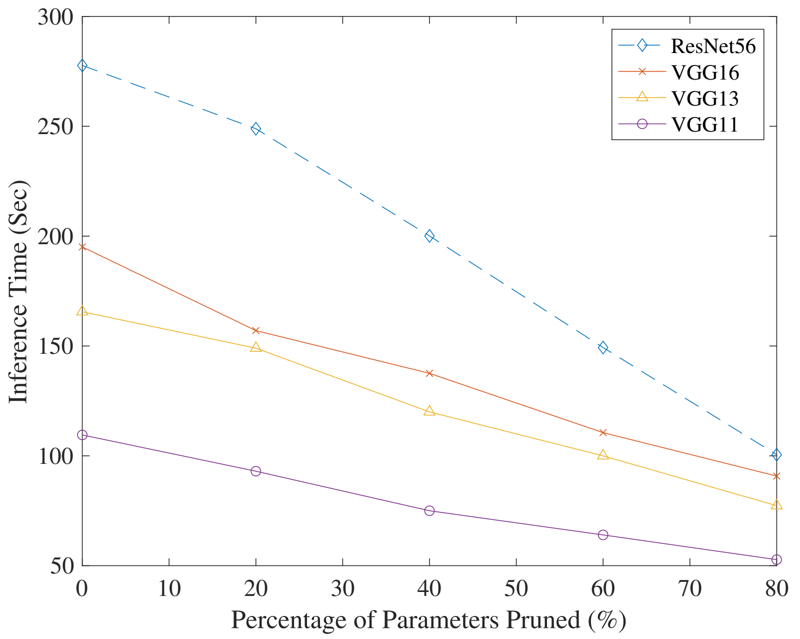


Figure 3.14: Inference Time Required for Different Backbone Models

It could also be found in Table. 3.3, in terms of memory requirement, the percentage reduction in parameters for our pruned ResNet56 is also not great as pruned VGG16.

3.5.2 Results on Bearings Dataset

3.5.2.1 Comparison of the original model and the pruned Model

Without loss of generality, we choose VGG16 as the backbone and set the coefficient α to $5e-5$ and the weight combination threshold T to 0.005 in the pruned model. The accuracy results of the classification are shown in Table. 3.5. It can be observed that both the original and pruned models achieved good accuracy rates in each of the sub-datasets.

Table 3.5: Comparison of Accuracy Under Different Sub-datasets

(a) Original Model

Sub-dataset	**Accuracy**
D00	0.94385
D01	0.99277
D02	0.92292
D03	0.97400
D10	0.98923
D11	0.96615
D12	0.97908
D13	0.98846
D20	0.98031
D21	0.98308
D22	0.97077
D23	0.98892
D30	0.98600
D31	0.98092
D32	0.98877
D33	0.96462

(b) Pruned Model

Sub-dataset	**Accuracy**
D00	0.97846
D01	0.99323
D02	0.98800
D03	0.97246
D10	0.98785
D11	0.95077
D12	0.98554
D13	0.99031
D20	0.97477
D21	0.98600
D22	0.97000
D23	0.99031
D30	0.98923
D31	0.98062
D32	0.97754
D33	0.97615

In Table. 3.6, we compare the performance on the original method and our method. The sub-dataset used is D03 and the backbone is VGG16. It is noticeable that pruned models are capable of reducing the number of parameters and FLOPs significantly, without compromising network performance.

Table 3.6: Comparison with the original SWP on D03.

Backbone	Metrics	Params	FLOPS	Accuracy
VGG16	baseline	14.76M	627.37M	97.40 %
	Original($\alpha = 1e-5$)	3.32M	341.65M	97.35 %
	Ours ($\alpha = 1e-5$, $T = 0.0001$)	4.43M	375.68M	97.43 %

3.5.2.2 Edge device performance

We also verify our approach in the edge device. The experimental setup and procedures are identical to that used for the CIFAR-10 dataset in the previous part. The sub-dataset used is D12 and the backbone is VGG16. The results are presented in Figure. 3.15, which shows that the inference time of the pruned model did not significantly change in terms of edge device performance.

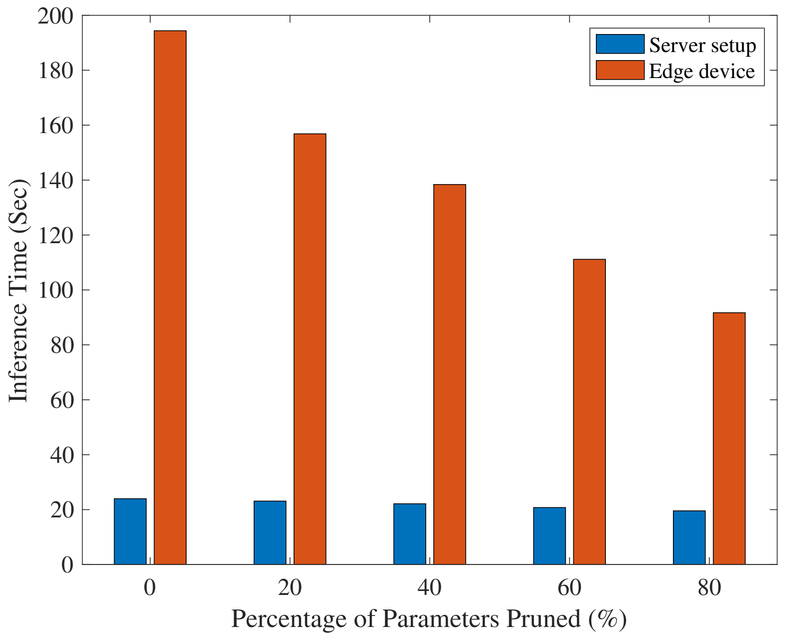


Figure 3.15: Inference Time Required for Pruned VGG16

3.6 Discussion

Our pruned VGG16 achieves results comparable to the existing model, with a 4-fold reduction in parameters and only a 0.4% decrease in accuracy. When deployed on the edge device, the inference time in the pruned network could drop by half. In addition to validating our pruned model on CIFAR-10, we also tested our model on the widely used bearing dataset from the Case Western Reserve University (CWRU) Bearing Data Center. The accuracy and the inference time are similar to those when using CIFAR-10. To the best of our knowledge, this is the first time that a stripe-wise pruning algorithm has been applied to the edge devices and Bearing datasets. However, due to the relatively small size of the CIFAR-10 and Bearing Data Center datasets, the pruned model may have limitations. We need to validate our theories on larger datasets.

3.7 Conclusion

In this work, we avoid using an absolute threshold in existing stripe-wise pruning by combining the weights located on each stripe. This allows us to learn the importance between stripes in a filter and remove those with low importance. Our pruned method effectively reduces the parameters and inference time of our VGG16 model without significantly impacting accuracy. In future work, we will explore the introduction of regularizers to prune filters with single stripes, which may further compress deep neural networks and improve performance.

CHAPTER 4

Enhancing Multipath Mitigation in Noncoherent Multicarrier SIMO Systems with Deep Energy Autoencoder

4.1 Introduction

In the realm of wireless communication, the precision and efficiency of data transfer are considered paramount. To ensure these, conventional systems rely heavily on a well-structured division of processing modules [64]. Each of these modules has been expertly designed and assigned with a unique responsibility to handle a specific aspect of the communication process.

Deep learning (DL) has significantly impacted various fields with its extraordinary capabilities. In computer vision, DL models enable tasks like facial recognition, object detection, and semantic segmentation [1]. In speech recognition, it powers applications like speech-to-text and voice assistants [2], becoming crucial in our daily lives. Large language models, such as GPT [3], use deep learning to understand natural language, offering powerful tools for chatbots, machine translation, and automated text generation.

Building on the remarkable influence DL has demonstrated across diverse domains, its potential has also been increasingly acknowledged in the field of communication, especially in recent years. This can be attributed to deep learning's capacity to decipher and analyze intricate non-linear relationships between inputs and outputs. Such ability has led to substantial enhancements in several processing modules within the communication system [65]. Moreover, DL's automatic feature extraction capability can reduce the reliance on handcrafted features [66], thereby simplifying the communication system's design. For instance, DL can be employed to enhance the performance of processing modules such as signal detection [67], channel estimation [68], and power allocationsn [69].

To address the challenges posed by the interdependence among different modules in a communication system, a comprehensive design approach is necessary. As stated earlier, alterations made to one module can have repercussions on the operation of other modules, potentially leading to a decline in overall efficiency. However, there have been emerging DL applications that involve the joint design of certain modules [70, 71], offering potential solutions to these challenges. In recent years, an alternative approach called end-to-end learning-based systems has gained attention. In this approach, deep neural networks are used to represent both the transmitter and the receiver, allowing the entire communication system to be trained end-to-end as a single entity. This approach offers the potential for improved efficiency and seamless integration of the system components. Incorporating autoencoder (AE), by considering the interplay between system modules and leveraging the power of neural networks, end-to-end learning-based systems present a promising direction for advancing communication technology [72].

Noncoherent transmissions refer to wireless communication systems where the receiver does not have a phase reference for the received signal, and thus cannot perform coherent demodulation. In these systems, various energy-based detection (ED) schemes have been proposed. In ED, the receiver calculates the energy of the received signal, and compares it to a threshold to determine the transmitted symbol. In the domain of noncoherent single-carrier transmissions, extensive research has been conducted into a variety of ED mechanisms under the scope of nonnegative pulse amplitude modulation (PAM). These studies have been particularly prevalent within the context of single-input multiple-output (SIMO) systems. In [73], an analysis was conducted on the performance of a massive SIMO system based on ED, leading to the derivation of an optimal power allocation design. A proposal was made in [74] for noncoherent SIMO systems, introducing a distinctively factorable hexagonal constellation. The assumption is that the channels remain constant within consecutive pairs of time slots.

Inspired by articles such as [72], the concept of AE was also applied to noncoherent SIMO systems [75]. For ED applications, a deep energy autoencoder (EA) was proposed for noncoherent multicarrier multiuser single-input multiple-output (MU-SIMO) systems under fading channels in [76]. This proposed system employed a single-user noncoherent EA-based (NC-EA) approach within the multicarrier SIMO framework. In this system, deep neural networks (DNNs) were used to represent both the transmitter and receiver, serving as the encoder and decoder of an EA. However, one noticeable shortcoming in [76] is the insufficient discussion of the impact of multipath fading as shown in Figure. 4.1. Fading channels, particularly the Doppler frequency shifts caused by relative motion between the transmitter and receiver, may significantly impact the performance of wireless communications. Furthermore, a noncoherent massive SIMO system over a multipath channel was discussed in [77], yet this research similarly lacks an in-depth exploration of fading channel conditions and the implications of Doppler frequency shifts.

This chapter presents an extension of the deep energy autoencoder (EA) technique to address multipath scenes in multicarrier SIMO systems. Our primary contributions can be summarized as follows:

- We extended the use of the deep energy autoencoder (EA) technique in multicarrier SIMO systems from handling only a single message to being capable of processing message sequences.

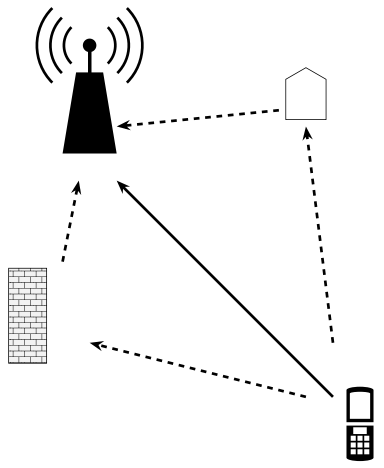


Figure 4.1: Over multipath fading channels

- Under the influence of the multipath fading channel, we explored two receiver architectures for the decoder: the DNN (Deep Neural Network) from the original design and the RNN (Recurrent Neural Network), which leverages its memory properties.
- By integrating the Jakes' model to account for varying Doppler frequencies, we enhanced the practicality and robustness of our design. This contribution enriches the accuracy and reliability of our approach in real-world scenarios.

The remainder of this chapter is organized as follows. In Section 4.2, we describe the system model of deep energy autoencoder (EA) for multicarrier SIMO systems. Section 4.3 presents our scheme for multipath scenes. In Section 4.4, we present the simulation results of the proposed schemes and system performances. Section 4.5 concludes this paper.

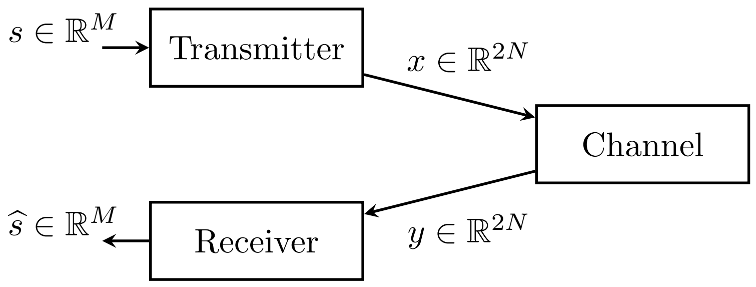


Figure 4.2: A basic autoencoder-based end-to-end communication system.

4.2 System Model

As illustrated in Figure. 4.2, a transmitter, a channel, and a receiver together form a communication system in its simplest way, which could be interpreted as an autoencoder [72].

The transmitter intends to transmit one of M possible messages, denoted as $s \in \mathcal{M} = \{1, 2, ..., M\}$, to the receiver using $2N$ discrete channel uses. In order to achieve this, the transmitter applies the function $f : \mathcal{M} \rightarrow \mathbb{R}^{2N}$ to the message s, resulting in the transmitted signal $x = f(s) \in \mathbb{R}^{2N}$. The channel can be characterized by the conditional probability density function $p(y|x)$, where $y \in \mathbb{R}^{2N}$represents the received signal. After receiving y the receiver applies a transformation function $g : \mathbb{R}^{2N} \rightarrow \mathcal{M}$. This transformation is used to generate an estimate $\hat{s}$ of the transmitted message s.

Drawing inspiration from the architecture and its variations presented in Figure. 4.2, [76] introduces an efficient autoencoder-based SIMO system with noncoherent multicarrier energy-based detection, as illustrated in Figure. 4.3. This unique system consists of transmitter and receiver components that are intricately designed using Deep Neural Network (DNN) layers. Furthermore, these layers are jointly optimized to achieve optimal performance, as indicated by the design methodology employed. It assumes that the communication system uses N sub-carriers without any CSI

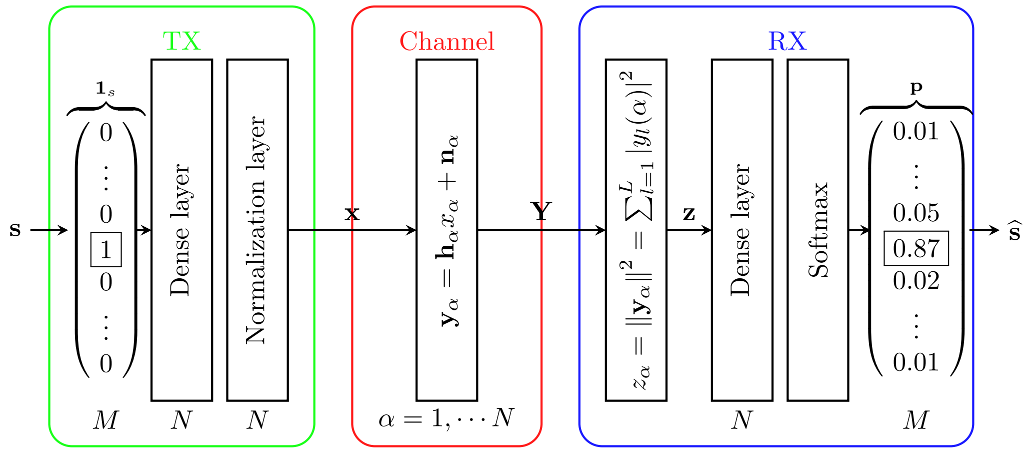


Figure 4.3: Structure of the noncoherent energy-based autoencoder system.

estimation. There is only one antenna for the transmitter, but the receiver has L antennas.

To be more precise, at the transmitter, the message, denoted as s, is mapped to a distinct one-hot vector of dimensions $M \times 1$. This one-hot vector is then utilized as the input for the encoder. The set of all possible messages is denoted by $s \in \mathcal{M} = \{1, 2, ..., M\}$, where there are $M = 2^m$ messages in total, each containing m data bits. The encoder comprises a fully-connected (FC) layer, leveraging the hyperbolic tangent (Tanh) activation function σ_{Tanh}. This FC layer produces an output $\mathbf{u}$.

Subsequently, $\mathbf{u}$ is normalized to ensure the mean transmission power across each sub-carrier is restricted to a fixed value, as demonstrated below

$$\mathbf{x} = \frac{\sqrt{NSE_s}\mathbf{u}}{\sqrt{\sum_{i=1}^{S} \|\mathbf{u}_i\|^2}} \tag{4.1}$$

In above equation, E_s represents the average transmit power per sub-carrier. The output corresponding to the i-th batch from the set $\Omega = \{s_1, \cdots, s_T\}$, where T is the batch size and the set consists of S training samples, is represented by $\mathbf{u}_i$.

The signal vector received by L antennas is represented as follows for each frequency sub-carrier α ranging from 1 to N, with an additive noise $\mathbf{n}_\alpha$:

$$\mathbf{y}_\alpha = \mathbf{h}_\alpha x_\alpha + \mathbf{n}_\alpha \tag{4.2}$$

where $\mathbf{y}_\alpha = [y_{1\alpha}, \cdots, y_{L\alpha}]^\top$, x_α is the α-th entry of $\mathbf{x}$. $\mathbf{h}_\alpha = [h_{1\alpha}, \cdots, h_{L\alpha}]^\top$ denotes the fading channel response between the transmitter to L receive antennas with $h_{l\alpha} \sim \mathcal{CN}(0, 1)$.

The total energy captured from L receiving antennas for each sub-carrier is initially computed as the decoder's input:

$$z_\alpha = \|\mathbf{y}_\alpha\|^2 = \sum_{l=1}^{L} |y_l(\alpha)|^2 \tag{4.3}$$

This leads to the formation of the collective energy vector $\mathbf{z} = [z_1, ..., z_N]^T$, having dimensions of $N \times 1$, applicable to all sub-carriers. In the architecture of the decoder, it includes two non-linear fully connected (FC) layers. The initial FC layer is comprised of Q nodes, and it employs the Tanh activation function, denoted as σ_{Tanh}. On the other hand, the latter FC layer, which functions as the output layer, contains M nodes and uses the softmax activation function, symbolized as σ_{Softmax}.

The calculation of the estimated value $\widehat{\mathbf{s}}$ is dependent on the highest value of $\widehat{\mathbf{s}}$.

4.3 Multipath Environments

Multipath in wireless communication is the phenomenon of radio signals traveling multiple paths from the transmitter to the receiver. This results in multiple copies of the signal arriving at the receiver at different times and with varying signal strengths.

Given this challenge of multipath propagation, we extend the NC-EA system to handle the multipath case, enabling robust communication even when signals take

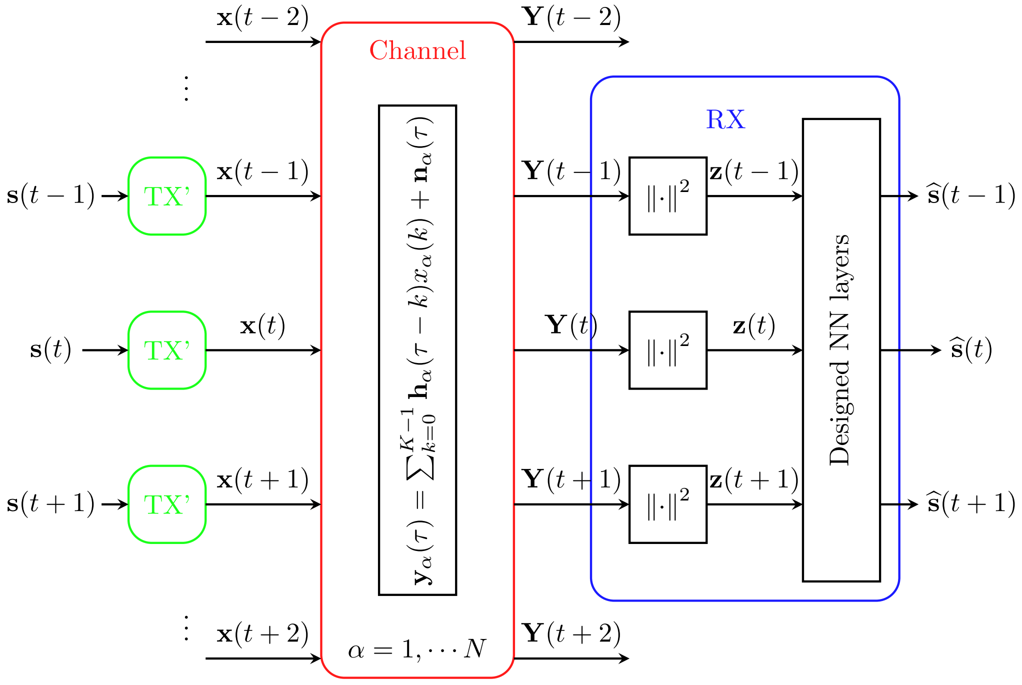


Figure 4.4: Structure of the noncoherent energy-based autoencoder system under multipath.

multipaths between the transmitter and receiver, as shown in Figure. 4.4. Each message in the message sequences s_m is randomly sampled from the training dataset $\mathcal{M}=\{1,2,...,M\}$. The number of the messages in the message sequences is D. In each one-time slot t, only one message $s(t)$ enters the transmitter. After passing through the dense layer and normalized layer, the allocation to each subcarrier α is denoted as $x_\alpha(t)$. The number of multipath channels is represented by N. In each sub-carrier α, for $\alpha=1,\cdots,N$, the received signal is at time slot τ represented by

$$\mathbf{y}_\alpha(\tau)=\sum_{k=0}^{K-1}\mathbf{h}_\alpha(\tau-k)x_\alpha(k)+\mathbf{n}_\alpha(\tau) \tag{4.4}$$

where $\mathbf{y}_\alpha(\tau)=[y_{1\alpha}(\tau),\cdots,y_{L\alpha}(\tau)]^\top$, $x_\alpha(n)$ is the α-th entry of $\mathbf{x}(n)$. $\mathbf{h}_\alpha(\tau)=[h_{1\alpha}(\tau),\cdots,h_{L\alpha}(\tau)]^\top$ denotes the fading channel response between the transmitter to L receive antennas with $h_{l\alpha}(\tau)\sim\mathcal{CN}(0,1)$, and $\mathbf{n}_\alpha(\tau)$ is the additive noise vector.

Likewise, when $t = \tau$, the decoder's input is determined by calculating the total energy received from L antennas for each sub-carrier:

$$z_\alpha(\tau) = \|\mathbf{y}_\alpha(\tau)\|^2 = \sum_{l=1}^{L} |y_{l\alpha}(\tau)|^2 \tag{4.5}$$

This results in the formation of the collective energy vector, $\mathbf{z}(\tau) = [z_1(\tau), ..., z_N(\tau)]^T$.

Composing a matrix using the collective energy vectors received at fixed intervals $\mathbf{z} = [\mathbf{z}(1), ...\mathbf{z}(\tau)..., \mathbf{z}(t)]^T$, the next part implements an NN receiver as described in the right-hand side of Figure. 4.4.

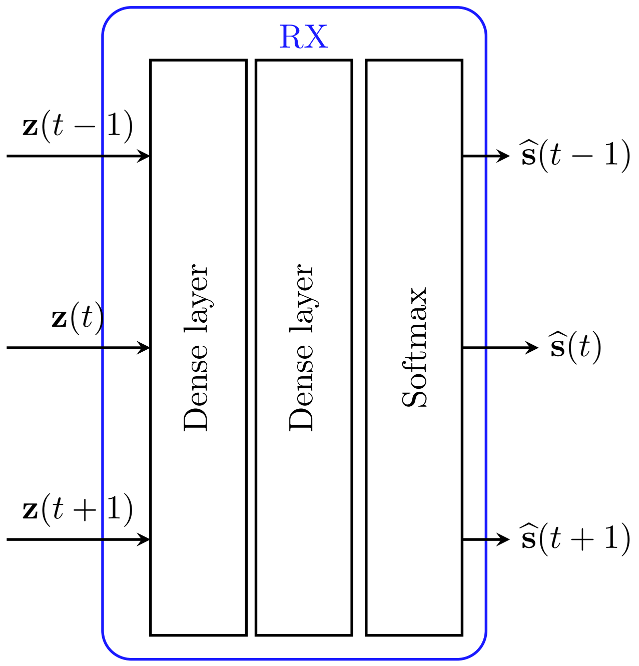


Figure 4.5: Structure of CNN receiver under multipath.

As shown in Figure. 4.5, the first approach continues to use the Dense layer combined with the softmax function, except that multiple layers of Dense layers are required here. As shown in Figure. 4.6, the second approach considers the effects of multipath delay, taking advantage of the memory properties of RNN (Recurrent

Neural Network). This allows the use of previous channel information to help decode information at later moments.

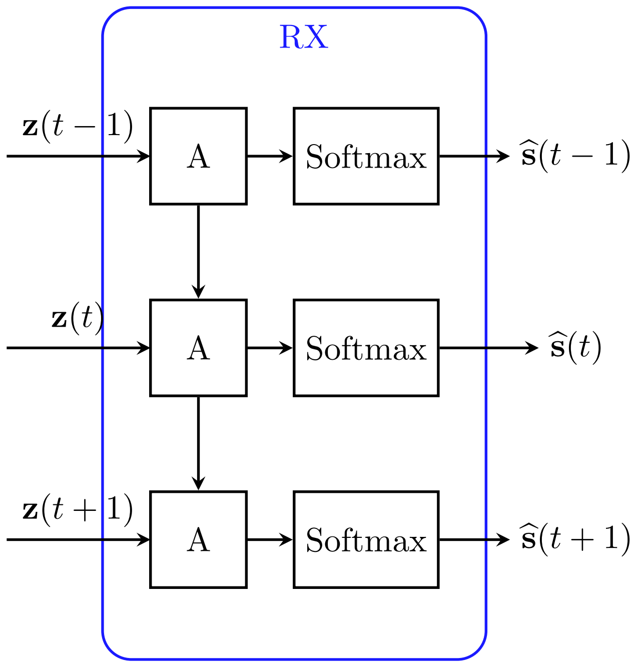

RX
$\mathbf{z}(t-1)$
$\mathbf{z}(t)$
$\mathbf{z}(t+1)$
A
A
A
Softmax
Softmax
Softmax
$\widehat{\mathbf{s}}(t-1)$
$\widehat{\mathbf{s}}(t)$
$\widehat{\mathbf{s}}(t+1)$

Figure 4.6: Structure of RNN receiver under multipath.

At each moment t, the calculation of the estimated value $\widehat{\mathbf{s}}(t)$ depends on the highest value of $\widehat{\mathbf{s}}(t)$.

4.3.1 Loss function

During each training phase, both the transmitters and the receiver are trained in parallel. For a message sequences of length D, the loss function is

$$\mathcal{L}(\theta) = \sum_{d=1}^{D} \varepsilon_d + \sum_{d=1}^{D} (\varepsilon_d - \bar{\varepsilon})^2 \tag{4.6}$$

where $\varepsilon_d = \|\mathbf{s}(d) - \widehat{\mathbf{s}}(d)\|^2$ is the least squared error (LSE) at time slot d and $\bar{\varepsilon} = \frac{1}{D}\sum_{d=1}^{D} \varepsilon_d$. The first term of 4.6 represents the reconstruction loss, which is the total LSE of the whole message sequences. The second term is introduced to minimize the standard deviation of individual LSEs (ε_d), aiming to make them as similar as possible.

Utilizing (4.6), the parameters of the whole model undergo updates through the stochastic gradient descent (SGD) algorithm according to the following procedure:

$$\theta := \theta - \eta \nabla \mathcal{L}(\theta) \tag{4.7}$$

where η represents the learning rate, governing the extent to which the parameters are adjusted.

4.4 Simulation

We conduct thorough simulations to validate the error performance of the proposed methods.

4.4.1 Jakes' model

Specifically, we integrate the Jakes's model, which accounts for varying Doppler frequencies, to simulate the Rayleigh fading channel.

A Rayleigh fading channel can be characterized by generating the real and imaginary parts of a complex number using independent normal Gaussian variables. However, this method overlooks the Doppler effect induced by the relative movement between the transmitter and the receiver. In this paper, we consider Jakes' model, which takes into consideration this effect. Jakes' model is based on the concept that the overall received signal at a mobile unit is the summation of numerous plane waves arriving from various directions.

Assumes R rays of identical intensity reach the moving receiver, distributed evenly across all arrival angles. This premise allows the kth fading waveform to be conceptualized as follows [78]:

$$R_k(t) = 2\sqrt{2}\left[\sum_{n=1}^{R} (\cos\beta_n + j\sin\beta_n)\cos(2\pi f_n t + \theta_{n,k}) + \frac{1}{\sqrt{2}}\cos(2\pi f_d t + \theta_{0,k})\right] \tag{4.8}$$

where f_d is the Doppler shift while $f_n = f_d \cos\alpha_n$ is the Doppler shift on ray n. $\beta_n = \frac{\pi n}{R+1}$. To generate the multiple waveform,

$$\theta_{n,k} = \beta_n + \frac{2\pi(k-1)}{R+1} \tag{4.9}$$

There are two crucial parameters in Jakes' model, namely the number of scatterers R and the Doppler shift f_d. Without loss of generality, in the simulation, let R be equal to 50. Set f_d to 20Hz and 200Hz, representing the relative motion of low speed and high speed, respectively.

4.4.2 Simulation setting

We utilize the block error ratio (BLER) as a metric to demonstrate the system's performance. BLER is defined as the probability of error in the integer z, which represents the message in the message sequence. The training set comprises 20,000 samples, and the testing set comprises 50,000 samples. All simulation results are conducted within the Keras framework.

Due to the EA's reliance on received energy for signal detection, its decoding performance is highly sensitive to the Signal-to-Noise Ratio (SNR) level $\bar{\gamma}$ used during training. Consequently, the model trained with a specific training SNR ($\bar{\gamma}_{\text{tr}}$) excels only when tested at SNRs close to $\bar{\gamma}_{\text{tr}}$ (designated as $\bar{\gamma}_{\text{te}}$). Conversely, it performs poorly when tested at other SNRs significantly different from $\bar{\gamma}_{\text{tr}}$.

To address this issue of overfitting, we employ a solution proposed in [76] where the model is trained with multiple SNRs. Subsequently, we test the trained models using the same SNR value for testing ($\bar{\gamma}_{\text{te}} = \bar{\gamma}_{\text{tr}}$). This approach helps mitigate the overfitting problem and enhances the generalization capability of the model across various SNR levels during testing. Therefore, when facing different channel variances, it is necessary to either retrain the model whenever $\bar{\gamma}$ undergoes a change or maintain several pre-trained models with distinct values of $\bar{\gamma}_{\text{tr}}$.

4.4.3 Results

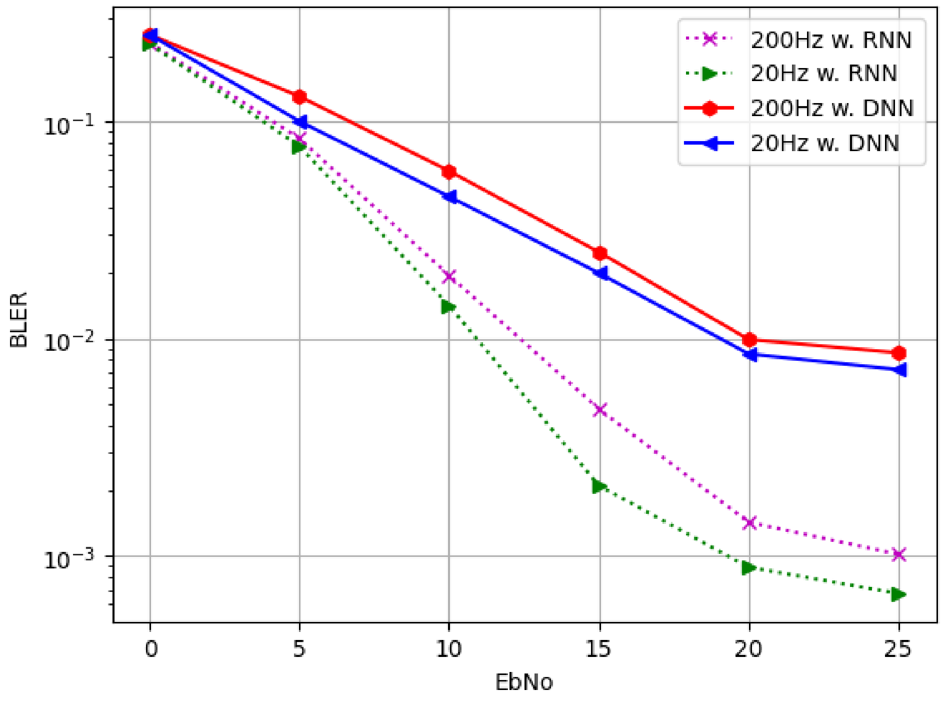


Figure 4.7: BLER comparison between two types of receivers under different SNRs, when $(N, M, L, D) = (4, 4, 2, 3)$.

Figure. 4.7 plots the block error probability (BLER) as a function of the different SNRs in comparison with the RNN and DNN as receivers. It also presents different curves at Doppler frequencies (f_d) of 20Hz and 200Hz. It is noticeable that DNNs are not an optimal design for every SNR level, while the RNN can obtain a better performance curve for any SNR levels via training. In addition, because the fluctuation of the channel at 20Hz is smaller than at 200Hz, the performance curve at 20Hz is superior to that under the same conditions at 200Hz.

Figure. 4.8 plots the BLER as a function of different SNRs for various performance curves at different numbers of receiver antennas. It can be observed that as the number of receiver antennas increases, the BLER performance improves accordingly.

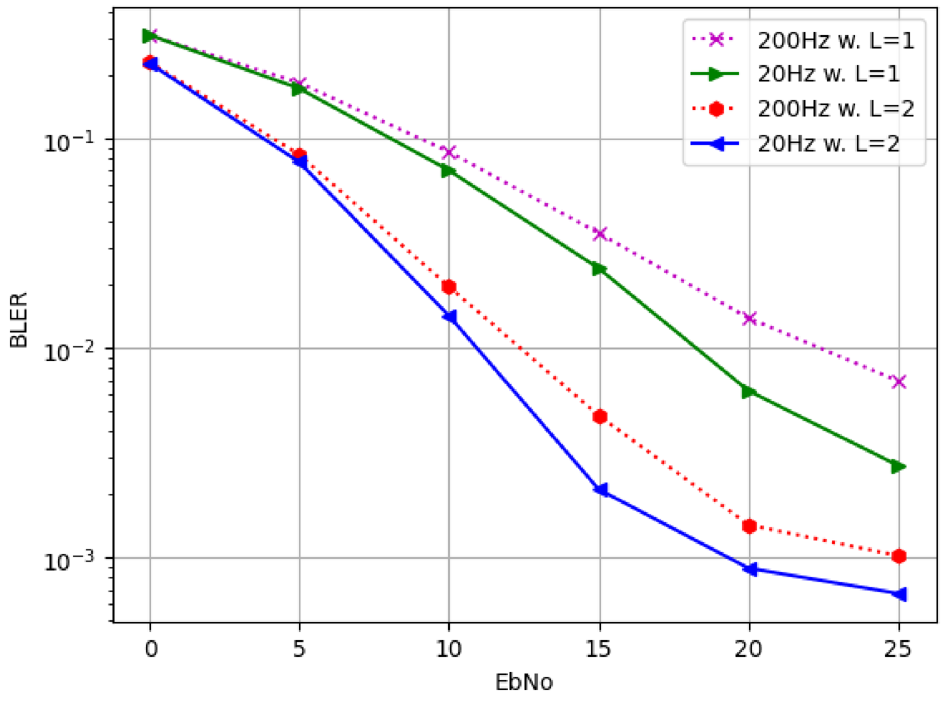


Figure 4.8: BLER comparison between various numbers of antennas at different SNRs, when $(N, M, D) = (4, 4, 3)$.

Figure. 4.9 plots the BLER as a function of different SNRs for performance curves with varying message sequence lengths. It can be observed that as the message sequence length increases, the BLER performance weakens. This is because, in longer sequence lengths, RNN cannot fully memorize the channel information, leading to a decrease in detection performance.

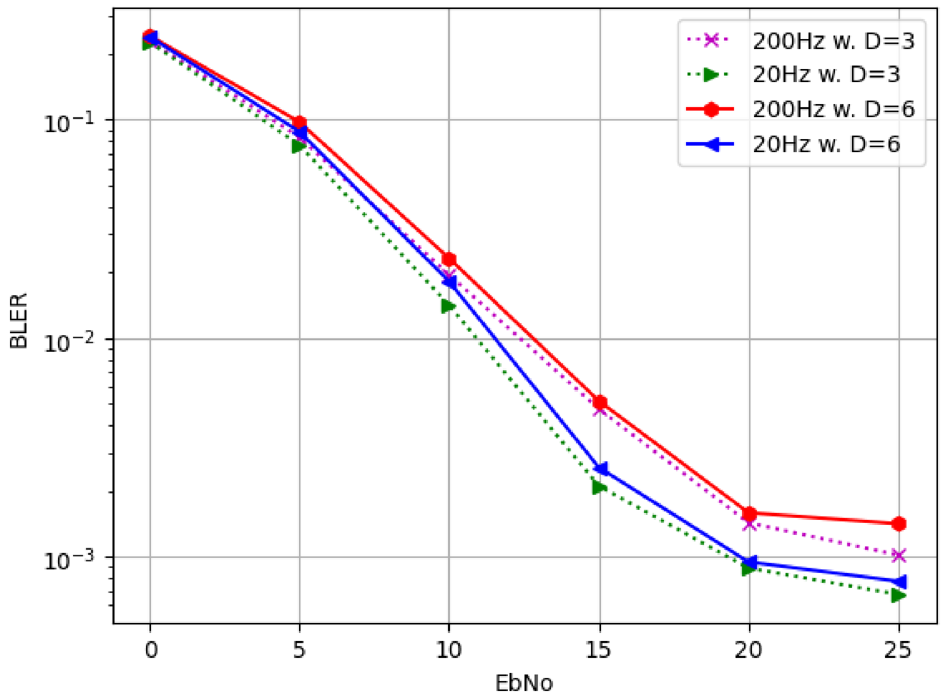


Figure 4.9: BLER comparison between various lengths of message sequences at different SNRs, when $(N, M, L) = (4, 4, 2)$.

Figure. 4.10 plots the BLER as a function of different SNRs for various curves without using the Jakes' model and with the Jakes model at different f_d of 20Hz and 200Hz in a Rayleigh fading channel. Without using the Jakes' model, there is significant fluctuation, and the performance curve is worse compared to the curves at the same conditions with f_d at 20Hz and 200Hz.

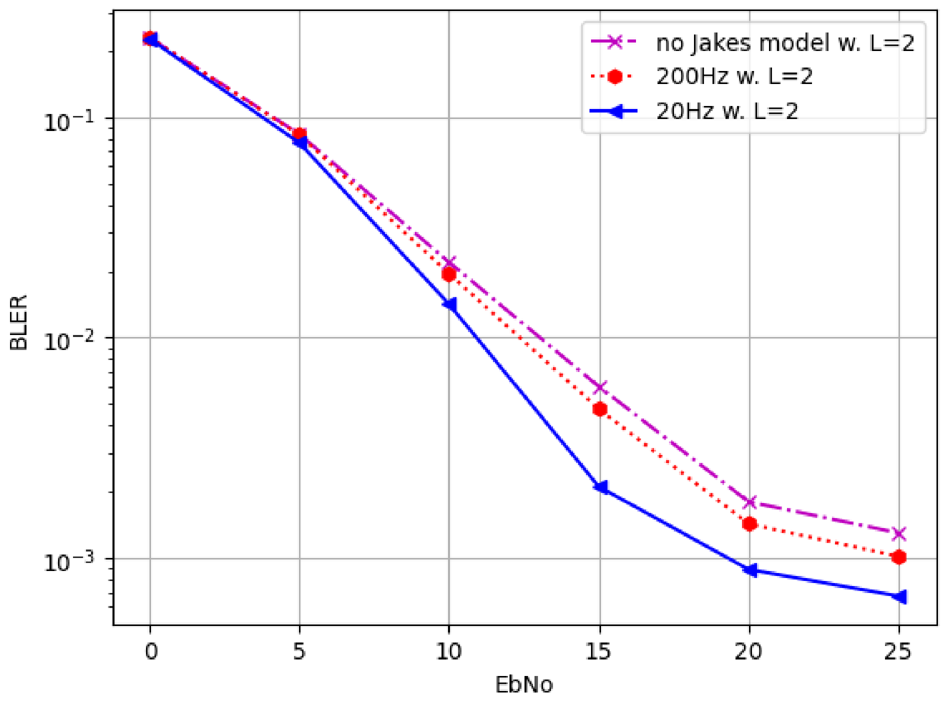


Figure 4.10: BLER comparison between using Jakes model or not under different SNRs, when $(N, M, L, D) = (4, 4, 2, 3)$.

Figure. 4.11 plots the BLER as a function of different SNRs for performance curves with varying numbers of carriers N and optional signal numbers M. It can be noticed that under the same conditions $(N, M) = (4, 4)$, the performance is slightly better than $(N, M) = (8, 8)$. As with traditional communications, increasing the number of symbols allows the system to transmit more data, but it may also lead to a higher occurrence of bit errors.

4.5 Conclusion

In this work, we delved into the potential of a deep energy autoencoder (EA) tailored for a noncoherent multicarrier SIMO system that operates in multipath channel environments. Our approach pivots on a unique multicarrier SIMO architecture characterized by a single-antenna sender and a multi-antenna receiver, both encapsu-

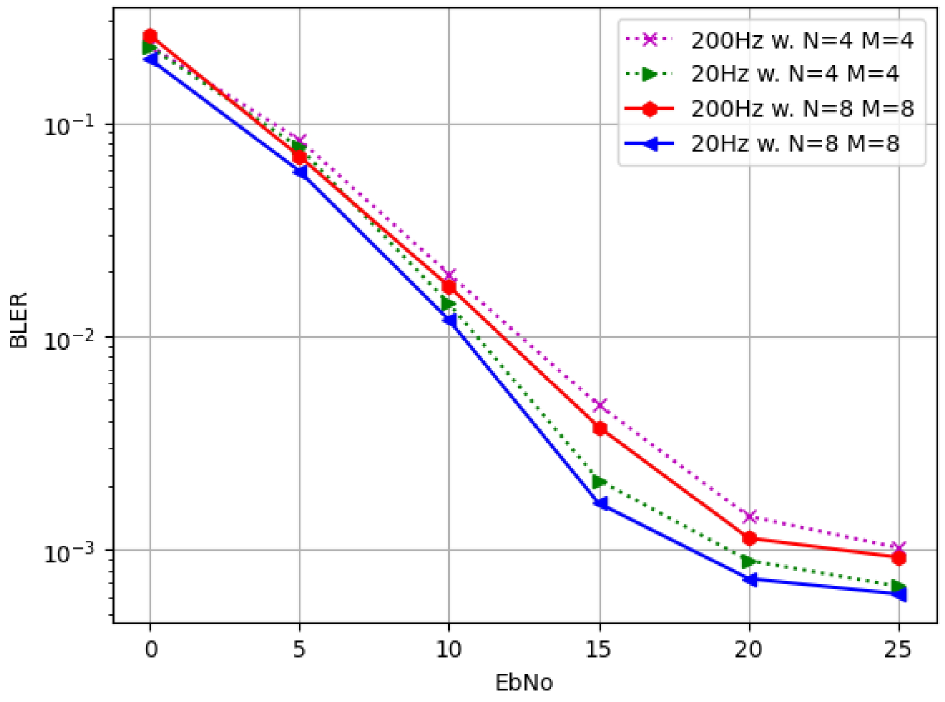


Figure 4.11: BLER comparison between various numbers of carriers at different SNRs, when $(L, D) = (2, 3)$.

lated through neural networks. A pivotal highlight of our method lies in the encoder's ability to generate a real-valued vector for individual subcarriers, while the decoder proficiently aggregates energy from all the receiving antennas. To counteract the significant concern of ISI caused by multipath channels, we sidestepped the intricate designs traditionally employed and instead leveraged the capabilities of DNNs and RNNs in the demodulation procedure. Notably, during our simulations, we also incorporated the Jakes' model to consider varying Doppler frequencies, enriching the practicality and robustness of our design. Our results affirm the proficiency of RNNs in restoring the transmitted data, even without the typically mandated channel state information inherent to conventional communication methodologies. As we look ahead, our research trajectory is set to embrace advancements such as integrating Generative Adversarial Networks (GAN) and accommodating multi-user scenarios.

CHAPTER 5

Conclusion and Future Works

This chapter represents the culmination of the entire dissertation. It commences with a recapitulation of the dissertation's findings and contributions, and subsequently delves into exploring potential avenues for future research concerning the optimization and applications in deep learning.

5.1 Conclusions

This thesis has centered on the optimization and applications in deep learning. The key accomplishments of this research are:

1. *Sense-Through-Foliage Target Detection Based on Stacked Autoencoder and UWB Radar Sensor Networks:*An approach for sense-through-foliage target detection using a Sparse Autoencoder (SAE) technique was presented. SAE proved effective in extracting essential information and deep features from sense-through-foliage radar echoes, particularly in scenarios where the received echoes had poor signal quality. The experimental results demonstrated that SAE achieved high detection accuracy, although there were some instances where the performance was not entirely satisfactory. To enhance the detection accuracy further, a preprocessing step was introduced before feeding the data into the neural network. This step involved utilizing a RAKE structure in the Radar Sensor Network (RSN), which combined echoes from different cluster-member radars. The simulation results indicated that integrating different radar echoes significantly improved the detection accuracy.

2. *A Statistical Approach for Neural Network Pruning with Application to Internet of Things:*We combine the weights located on each stripe in existing stripe-wise pruning, allowing us to learn the importance between stripes in a filter and remove those with low importance. Our pruned method effectively reduces the parameters and inference time of our VGG16 model with only a 0.4% decrease in accuracy, achieving results comparable to the existing model and a 4-fold reduction in parameters. When deployed on an edge device, the inference time in the pruned network drops by half. Our method has been validated on both the CIFAR-10 and the widely used bearing dataset from the Case Western Reserve University (CWRU) Bearing Data Center, with similar accuracy and inference time. To the best of our knowledge, this is the first application of a stripe-wise pruning algorithm to edge devices and Bearing datasets.
3. *Enhancing Multipath Mitigation in Noncoherent Multicarrier SIMO Systems with Deep Energy Autoencoder:* Our contribution is the extension and enhancement of the deep energy autoencoder (EA) technique for multicarrier SIMO systems to enable sequence-based message processing under multipath channel conditions. Specifically, we explore the integration of recurrent neural network (RNN) architectures in the decoder to leverage memory properties for sequence learning, as an alternative to the original deep neural network (DNN) design. Further, by incorporating the Jakes' model to simulate Doppler frequency shifts induced by mobility, we significantly improve the practicality and robustness of the EA approach.

5.2 Future Direction

5.2.1 Introduction regularizers to prune filters and applications in other areas

In future work, we will explore the introduction of regularizers to prune filters with single stripes, which may further compress deep neural networks and improve performance. This approach is particularly relevant in the context of large language models like GPT-4 [79], where the number of parameters is vast. The need for pruning in these situations becomes increasingly crucial as it can contribute to making the models more efficient and manageable, without sacrificing much in terms of predictive power or accuracy. By actively employing pruning within these complex models, we may uncover innovative ways to enhance their functionality and scalability, catering to the ever-growing demands of modern machine learning applications.

Similarly, we will extend the use of my pruning model to other domains, exploring its potential to optimize various systems and contribute to a wide range of applications. This decision has arisen from the recognition of the model's inherent flexibility and its proven success in previous projects. By adapting it to new contexts, we aim to further leverage its capabilities for broad technological advancements. In particular, we are planning to apply the pruning model to areas such as network optimization, where it could streamline data flow and reduce unnecessary redundancy. Additionally, the model's utility in energy efficiency could lead to more sustainable operations in industrial settings, minimizing waste and potentially reducing costs.

5.2.2 Advancing Research with GAN Integration and Multi-User Adaptation

As for SIMO's part, we are gearing up to integrate GANs into our research. GANs comprise two neural networks – a generator and a discriminator. As these two networks engage in competition, they gradually refine themselves, eventually producing highly realistic data. We believe incorporating GANs can furnish our models with superior quality data, enhancing accuracy and robustness. Moreover, we are also setting our sights on accommodating multi-user scenarios.

CHAPTER 6

Publication List

1. Chengchen Mao and Qilian Liang. “ Sense-through-foliage target detection based on stacked autoencoder and uwb radar sensor networks,” In *Lecture Notes in Electrical Engineering*, pages 390–397. Springer, 2022.
2. Chengchen Mao, Qilian Liang, Chenyun Pan, and Ioannis Schizas “ A Statistical Approach for Neural Network Pruning with Application to Internet of Things,” In *EURASIP Journal on Wireless Communications and Networking*, pages 1–21. SpringerOpen, 2022.
3. Chengchen Mao, Qilian Liang, Chenyun Pan, and Ioannis Schizas. “ Advancing Internet of Things Through Statistical Pruning of Neural Networks,” In *Lecture Notes in Electrical Engineering*, 2023.
4. Chengchen Mao, Zongwen Mu, Qilian Liang, Ioannis Schizas, and Chenyun Pan. “ Deep Learning in Physical Layer Communications: Evolution and Prospects in 5G and 6G Networks,” In *IET Communications*, 2023.

REFERENCES

[1] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," 2023.

[2] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *International Conference on Machine Learning.* PMLR, 2023, pp. 28 492–28 518.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[4] P. Ji, J. Ye, Y. Mu, W. Lin, Y. Tian, C. Hens, M. Perc, Y. Tang, J. Sun, and J. Kurths, "Signal propagation in complex networks," *Physics Reports*, vol. 1017, pp. 1–96, 2023.

[5] H. V. Ribeiro, D. D. Lopes, A. A. Pessa, A. F. Martins, B. R. da Cunha, S. Gonçalves, E. K. Lenzi, Q. S. Hanley, and M. Perc, "Deep learning criminal networks," *Chaos, Solitons & Fractals*, vol. 172, p. 113579, 2023.

[6] Z. Gao, W. Dang, X. Wang, X. Hong, L. Hou, K. Ma, and M. Perc, "Complex networks and deep learning for eeg signal analysis," *Cognitive Neurodynamics*, vol. 15, pp. 369–388, 2021.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[8] I. Oppermann, M. Hämäläinen, and J. Iinatti, *UWB: theory and applications.* John Wiley & Sons, 2004.

[9] E. Karapistoli, F.-N. Pavlidou, I. Gragopoulos, and I. Tsetsinas, "An overview of the ieee 802.15. 4a standard," *IEEE Communications Magazine*, vol. 48, no. 1, pp. 47–53, 2010.

[10] A. Heinrich, N. Bittner, and M. Hollick, "Airguard-protecting android users from stalking attacks by apple find my devices," in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2022, pp. 26–38.

[11] 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*, 2016, pp. 770–778.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[13] J. Liang, Q. Liang, and S. W. Samn, "Foliage clutter modeling using the uwb radar," in *2008 IEEE International Conference on Communications.* IEEE, 2008, pp. 1937–1941.

[14] Y. S. Meng, Y. H. Lee, and B. C. Ng, "Empirical near ground path loss modeling in a forest at vhf and uhf bands," *IEEE transactions on antennas and propagation*, vol. 57, no. 5, pp. 1461–1468, 2009.

[15] A. Y. Nashashibi, K. Sarabandi, S. Oveisgharan, M. C. Dobson, W. S. Walker, and E. Burke, "Millimeter-wave measurements of foliage attenuation and ground reflectivity of tree stands at nadir incidence," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 5, pp. 1211–1222, 2004.

[16] Q. Liang, S. W. Samn, and X. Cheng, "Uwb radar sensor networks for sense-through-foliage target detection," in *2008 IEEE International Conference on Communications*. IEEE, 2008, pp. 2228–2232.

[17] J. Liang and Q. Liang, "Sense-through-foliage target detection using uwb radar sensor networks," *Pattern Recognition Letters*, vol. 31, no. 11, pp. 1412–1421, 2010.

[18] I. Maherin and Q. Liang, "A mutual information based approach for target detection through foliage using uwb radar," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 6406–6410.

[19] ——, "Radar sensor network for target detection using chernoff information and relative entropy," *Physical Communication*, vol. 13, pp. 244–252, 2014.

[20] ——, "Multistep information fusion for target detection using uwb radar sensor network," *IEEE Sensors Journal*, vol. 15, no. 10, pp. 5927–5937, 2015.

[21] G. Zhao, Q. Liang, and T. S. Durrani, "Uwb radar target detection based on hidden markov models," *IEEE Access*, vol. 6, pp. 28 702–28 711, 2018.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[23] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[24] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected topics in applied earth observations and remote sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.

[25] J. Geng, J. Fan, H. Wang, X. Ma, B. Li, and F. Chen, "High-resolution sar image classification via deep convolutional autoencoders," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 11, pp. 2351–2355, 2015.

[26] B. Feng, B. Chen, and H. Liu, "Radar hrrp target recognition with deep networks," *Pattern Recognition*, vol. 61, pp. 379–393, 2017.

[27] Q. Ren and Q. Liang, "Fuzzy logic-optimized secure media access control (fs-mac) protocol wireless sensor networks," in *CIHSPS 2005. Proceedings of the 2005 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety, 2005.* IEEE, 2005, pp. 37–43.

[28] ——, "Energy and quality aware query processing in wireless sensor database systems," *Information Sciences*, vol. 177, no. 10, pp. 2188–2205, 2007.

[29] ——, "Throughput and energy-efficiency-aware protocol for ultrawideband communication in wireless sensor networks: a cross-layer approach," *IEEE Transactions on Mobile Computing*, vol. 7, no. 6, pp. 805–816, 2008.

[30] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[31] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[32] Q. Liang, L. Wang, and Q. Ren, "Fault-tolerant and energy efficient cross-layer design for wireless sensor networks," *International Journal of Sensor Networks*, vol. 2, no. 3-4, pp. 248–257, 2007.

[33] L. Xu and Q. Liang, "Zero correlation zone sequence pair sets for mimo radar," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 3, pp. 2100–2113, 2012.

[34] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[35] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, *et al.*, "Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 70–78, 2017.

[36] J. Mei, K. Zheng, L. Zhao, Y. Teng, and X. Wang, "A latency and reliability guaranteed resource allocation scheme for lte v2v communication systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 3850–3860, 2018.

[37] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[38] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[39] J. Tang, D. Sun, S. Liu, and J.-L. Gaudiot, "Enabling deep learning on iot devices," *Computer*, vol. 50, no. 10, pp. 92–96, 2017.

[40] S. Hooker, "The hardware lottery," *Communications of the ACM*, vol. 64, pp. 58 – 65, 2021.

[41] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.

[42] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[43] C. M. Bishop *et al.*, *Neural networks for pattern recognition.* Oxford university press, 1995.

[44] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[45] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[46] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1389–1397.

[47] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2736–2744.

[48] V. Lebedev and V. Lempitsky, "Fast convnets using group-wise brain damage," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2554–2564.

[49] F. Meng, H. Cheng, K. Li, H. Luo, X. Guo, G. Lu, and X. Sun, "Pruning filter in filter," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 629–17 640, 2020.

[50] R. Reed, "Pruning algorithms-a survey," *IEEE transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[51] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[52] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[53] S. Park, J. Lee, S. Mo, and J. Shin, "Lookahead: A far-sighted alternative of magnitude-based pruning," *arXiv preprint arXiv:2002.04809*, 2020.

[54] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[55] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *Advances in neural information processing systems*, vol. 29, pp. 2074–2082, 2016.

[56] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[57] W. Feller, *An introduction to probability theory and its applications, vol 2*. John Wiley & Sons, 2008.

[58] M. Schmidt, G. Fung, and R. Rosales, "Fast optimization methods for l1 regularization: A comparative study and two new approaches," in *European Conference on Machine Learning*. Springer, 2007, pp. 286–297.

[59] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS workshop*, no. CONF, 2021.

[60] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep., 2009.

[61] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[62] K. Loparo, "Case western reserve university bearing data center," *Bearings Vibration Data Sets, Case Western Reserve University*, pp. 22–28, 2012.

[63] C. A. Pickover, "On the use of symmetrized dot patterns for the visual characterization of speech waveforms and other sampled data," *The Journal of the Acoustical Society of America*, vol. 80, no. 3, pp. 955–960, 1986.

[64] A. Goldsmith, *Wireless communications*. Cambridge university press, 2005.

[65] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2595–2621, 2018.

[66] L. Nanni, S. Ghidoni, and S. Brahnam, "Handcrafted vs. non-handcrafted features for computer vision classification," *Pattern Recognition*, vol. 71, pp. 158–172, 2017.

[67] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "Model-driven deep learning for mimo detection," *IEEE Transactions on Signal Processing*, vol. 68, pp. 1702–1715, 2020.

[68] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep learning-based channel estimation," *IEEE Communications Letters*, vol. 23, no. 4, pp. 652–655, 2019.

[69] X. Chen, G. Liu, Z. Ma, X. Zhang, W. Xu, and P. Fan, "Optimal power allocations for non-orthogonal multiple access over 5g full/half-duplex relaying mobile wireless networks," *IEEE transactions on Wireless communications*, vol. 18, no. 1, pp. 77–92, 2018.

[70] X. Wang, H. Hua, and Y. Xu, "Pilot-assisted channel estimation and signal detection in uplink multi-user mimo systems with deep learning," *IEEE Access*, vol. 8, pp. 44 936–44 946, 2020.

[71] Y. Zhang, J. Sun, J. Xue, G. Y. Li, and Z. Xu, "Deep expectation-maximization for joint mimo channel estimation and signal detection," *IEEE Transactions on Signal Processing*, vol. 70, pp. 4483–4497, 2022.

[72] T. O'shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[73] M. Hammouda, S. Akin, and J. Peissig, "Performance analysis of energy-detection-based massive simo," in *2015 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom).* IEEE, 2015, pp. 152–156.

[74] E. Leung, Z. Dong, and J.-K. Zhang, "Uniquely factorable hexagonal constellation designs for noncoherent simo systems," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 5495–5501, 2016.

[75] S. Xue, Y. Ma, N. Yi, and R. Tafazolli, "Unsupervised deep learning for mu-simo joint transmitter and noncoherent receiver design," *ieee wireless communications letters*, vol. 8, no. 1, pp. 177–180, 2018.

[76] T. Van Luong, Y. Ko, N. A. Vien, M. Matthaiou, and H. Q. Ngo, "Deep energy autoencoder for noncoherent multicarrier mu-simo systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 3952–3962, 2020.

[77] H. Zhang, M. Lan, J. Huang, C. Huang, and S. Cui, "Noncoherent energy-modulated massive simo in multipath channels: A machine learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8263–8270, 2020.

[78] W. C. Jakes and D. C. Cox, *Microwave mobile communications.* Wiley-IEEE press, 1994.

[79] OpenAI, "Gpt-4 technical report," 2023.

BIOGRAPHICAL STATEMENT

Chengchen Mao was born in Yiwu, China in 1992. He received his B.S. and M.Sc degree from University of Electronic Science and Technology of China, China, in 2014 and 2017 respectively, both in Electrical Engineering. From 2015 to 2016, he was with the School of Communication and Information Engineering, University of Electronic Science and Technology of China as a Research Assistant in the Radar and Localization Lab. His current research interest is in the area of Deep Learning, Signal Processing, and Wireless Communication.