Ph.D. DISSERTATION

Radar Interference Mitigation Using Deep Learning with Neural Architecture Search

딥러닝을 통한 레이더 간섭제거와 신경망 아키텍처 탐색

BY

MUN JI-WOO

Feb 2023

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

Radar Interference Mitigation Using Deep Learning with Neural Architecture Search

딥러닝을 통한 레이더 간섭제거와 신경망 아키텍처 탐색

BY

MUN JI-WOO

Feb 2023

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Radar Interference Mitigation Using Deep Learning with Neural Architecture Search

딥러닝을 통한 레이더 간섭제거와 신경망 아키텍처 탐색

지도교수 이 정 우
이 논문을 공학박사 학위논문으로 제출함

2022년 10월

서울대학교 대학원

전기 정보 공학부

문 지 우

문지우의 공학박사 학위 논문을 인준함

2022년 11월

위 원 장: _____
부위원장: _____
위　　원: _____
위　　원: _____
위　　원: _____

Abstract

Recently, interest in deep learning has increased, and many studies are being conducted. In this dissertation, I deal with the application part that applies deep learning to radar signal processing and the deep learning theory that deals with neural architecture search.

First, in the automotive system, radar is a key component in autonomous driving. Using transmit radar signal and reflected radar signal by a target, I can capture target range and velocity. However, when interference signals exist, noise floor increases and it severely impacts detectability of target object. Previous studies have been proposed to cancel interference or reconstruct original signals. However, the conventional signal processing methods for canceling the interference or reconstructing the transmit signal are a high-complexity tasks, and also have many restrictions. In this work, I propose a novel concept to mitigate interference using deep learning. The proposed method provides high performance in various interference conditions and has low processing time. Moreover, I show that our proposed method achieves better performance compared to existing signal processing methods.

Second, neural architecture search (NAS) methods automatically find optimal neural networks without human assistance. Numerous algorithms for NAS have been studied to find architectures with gradient-based search. Differentiable architecture search (DARTS), one of the key papers of gradient-based search, dramatically reduced search cost, and shoId outstanding performance through continuous relaxation and meta-learning based approximation. However, one of the issues with DARTS is that the gradient-based search process is biased due to the nested bi-level structure, and the greedy behavior of the gradient descent. As a result, there is a problem that search spaces are limited to a limited set of architectures. To overcome the bias of the gradient-based search, I used dynamic search method. This technique allows the

gradient-based search to have an exploration effect. In this paper, I present a novel approach, namely, Dynamic-Exploration DARTS (DE-DARTS). For effective exploration, I use dynamic attention networks (DANs) in DE-DARTS, which change model architectures based on input data. As our DANs are activated early in the search, various architectures are considered, depending on input data at the beginning of search. Our algorithm is evaluated in multiple image classification datasets including CIFAR-10, CIFAR-100, and ImageNet, and shows improved performance.

Third, based on the method in neural architecture search, I apply it to radar interference cancellation. The model is based on the DARTS paper. As a result of applying the neural architecture search-based model, it was possible to remove radar interference well with the rnn-based model created by AI without designing the model directly.

keywords: radar, signal processing, neural architecture search, dynamic search

student number: 2016-27442

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 Related Work

1.1.1 Radar System

As interest in autonomous vehicles rises, research on radar is also in progress. As an example, in Figure 1.1, 1.2 [2], there are vehicles equipped with radar. As shown in Figure 1.3 the number of vehicles equipped with Advanced Driving Assistance System (ADAS) and automated driving is increasing over time. As time goes by, the number of vehicles for automated driving will increase, and technology for measuring distance, speed and angle will also be needed.

Figure 1.1: Vehicles equipped with radar

Figure 1.2: Vehicles equipped with radar

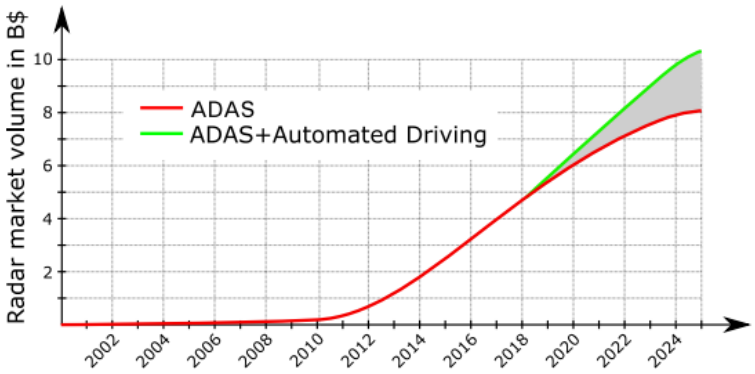


Figure 1.3: Advanced Driving Assistance System and Automated Driving market volume

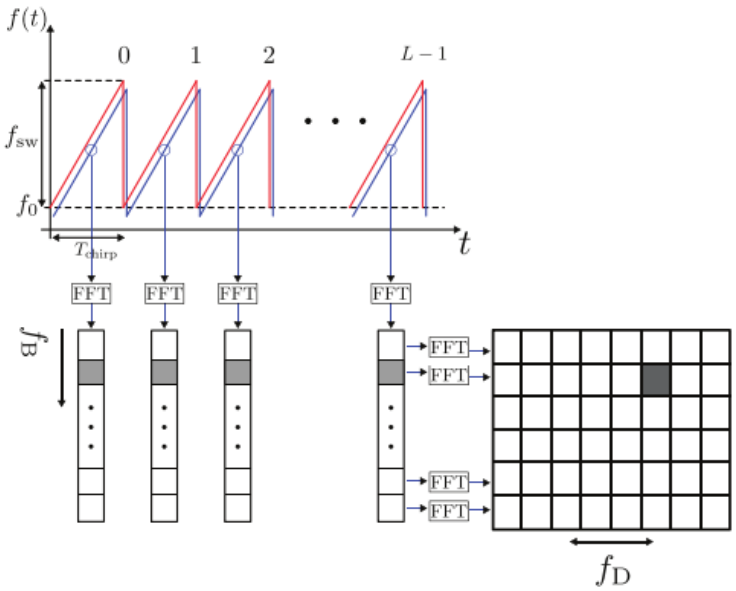


Figure 1.4: Advanced Driving Assistance System and Automated Driving market volume

Radar is mounted on a vehicle and detects the position of another vehicle, mainly by distance or angle. At this time, if another vehicle is present, a measurement error may occur due to interference. Signals mainly used in automotive radar include FMCW and OFDM. Figure 1.4 shows how FMCW works. Measure distance and speed through 2d-fft. There is a limit to removing interference with the existing signal processing method. Because it is difficult to process all interference patterns. However, if you use deep learning, if you put enough data, the model can learn the interference pattern and recover the original signal well.

1.1.2 Neural Architecture Search

In the early study of deep learning, a person directly designed a model to learn deep learning. However, it takes a long time to find the optimal model because you have to try all the parameters. In fact, even when designing a deep learning model used for

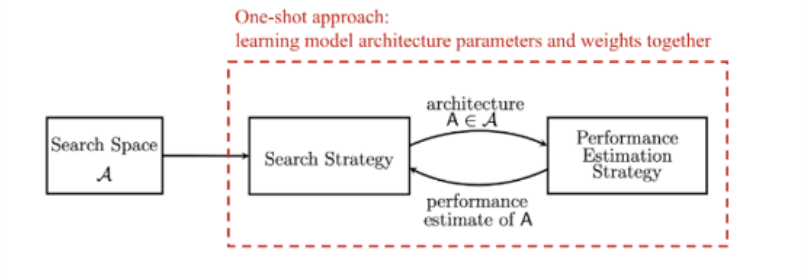


Figure 1.5: search space, search algorithm, evaluation strategy

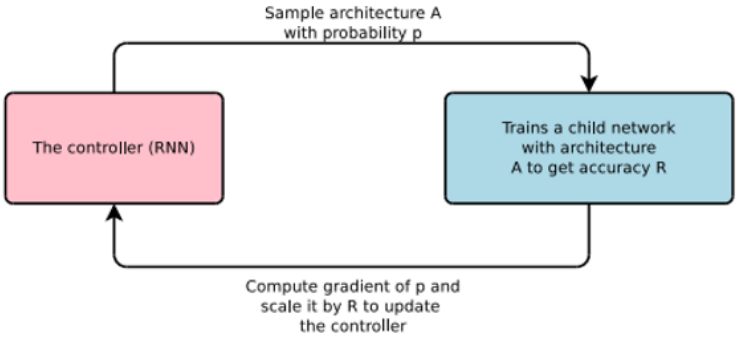


Figure 1.6: reinforcement learning algorithm

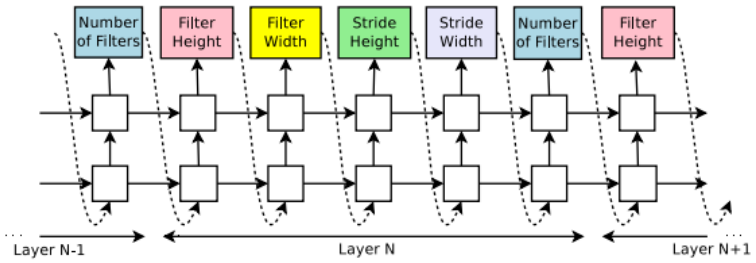


Figure 1.7: reinforcement learning algorithm

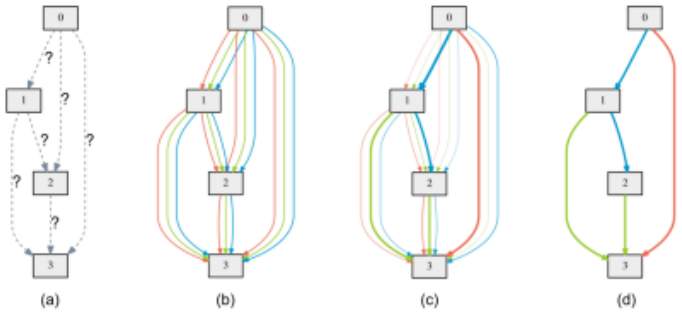


Figure 1.8: darts example

radar interference, it takes a long time to design several models from scratch. In order to optimize the design of deep learning models that take such a long time, AutoML technology Neural architecture Search technology has been proposed. Neural architecture search basically consists of three steps. search space, search algorithm, evaluation strategy. [3]. Search Space : The search space defines which architectures can be represented in principle. Incorporating prior knowledge about typical properties of architectures well-suited for a task can reduce the size of the search space and simplify the search. However, this also introduces a human bias, which may prevent finding novel architectural building blocks that go beyond the current human knowledge. Search Strategy: The search strategy details how to explore the search space(which is often exponentially large or even unbounded). It encompasses the classical exploration-exploitation trade-off since, on the one hand, it is desirable to find well-performing architectures quickly, while on the other hand, premature convergence to a region of suboptimal architectures should be avoided. Performance Estimation Strategy : The objective of NAS is typically to find architectures that achieve high predictive performance on unseen data. Performance Estimation refers to the process of estimating this performance: the simplest option is to perform a standard training and validation of the architecture on data, but this is unfortunately computationally expensive and limits the number of architectures that can be explored. Much recent research therefore focuses

on developing methods that reduce the cost of these performance estimations.

In the early days of neural architecture search technology, it was difficult learning that required a large number of GPUs as the process of learning the model from the beginning using reinforcement learning was involved. Figure 1.6[4] shows the reinforcement learning algorithm. The architecture is sampled through controller rnn and the sampled architecture is learned from the beginning to obtain reward R. The obtained reward is sent back to the controller and learned in a circular structure. Figure 1.7 is the architecture creation process.

However, recently, the learning time is shortened by using the weight sharing technique or the method of learning the model through gradient update. Figure 1.8 shows DARTS [5], one of the weight sharing techniques. The weights of each graph have a structure in which weights are shared, so learning is accelerated. In this study, the study was conducted in a situation where network search was shortened through weight sharing technology, and it is proposed to improve the limitations of the existing method.

1.2 Contributions and Organization

In this paper, I discuss how to apply deep learning to remove radar interference and propose a new automl technique to make a more efficient network.

Chapter 2 covers how to remove interference by applying deep learning to automotive radar for the first time. The existing vehicle radar interference removal method used signal processing technology to remove interference, but in this paper, deep learning was used. When there is interference in the radar signal, the part with the interference basically shows a different pattern from the non-interference signal, and deep learning detects this pattern and deletes it. The deep learning model used was an RNN-based model and showed better performance than the existing signal processing algorithm.

Chapter 3 proposes a deep learning algorithm with improved performance than

Chapter 1 and a method to remove deep learning interference from OFDM modules as well as FMCW. The deep learning model used is a model in which the attention module of transformer is added to the existing RNN model. As a result of the experiment, it showed better performance than the existing algorithm, and it was confirmed that both FMCW and OFDM interference was removed.

In Chapter 4, AutoML technology was researched considering that the existing deep learning model creation method is limited by human creation. I propose the proposed AutoML technology to find the optimal model in a given search space. In this dissertation, in order to solve the problem of increasing the number of skip connections, which is a limitation of DARTS: differentiable architecture search, an additional network is added to explore more diverse models in the model search process. As a result, the tendency to skip connection in the early stages of learning was alleviated, and a balanced model was found. As a result of the experiment, additional performance improvement was achieved compared to the existing papers.

In Chapter 5, I have shown that interference is removed with a non-human neural architecture search-based model. After making a cell with the DARTS technique, it was applied to radar interference cancellation. Attention was applied to the cell created by neural architecture search and compared with the algorithm in the previous paper. As a result, it showed a better result than the existing gru rnn cell.

In, Chapter 6, I summarize my works and propose a future directions

Chapter 2

A Deep Learning Approach for Automotive Radar Interference Mitigation

2.1 Introduction

Radars mounted on advanced vehicles, such as autonomous vehicles, require a variety of functions, including detection of multi-target and long-range sensing. These functions must be performed accurately ensure user safety and solve collision problem between vehicles. Recent popular radar technologies include Frequency Modulated Continuous Wave (FMCW) or Chirp Sequence (CS) radars [6, 7, 8]. However, it is difficult to perform the above functions with interference [9, 10].

Several techniques have been proposed to solve the problems related to interference [11, 12, 13, 14, 15]. [11] used the characteristics of the interference region in the time domain to remove the interference. [13] proposed a method of estimating the amplitude and frequency of the interference signal to recover the original signal as well as the interference elimination with high computational complexity. The paper [15] proposed an algorithm that requires a small computational complexity and showed that it detects targets within small distances without defining an adaptive threshold. The effect of interference still remains, however, because the target is not well detected when

the interference signal source is closer to the radar than the target.

To the best of our knowledge, It is the first to use a deep learning method to mitigate interference in time domain. Recently, the development of deep learning has been remarkable, and in particular, it has made significant achievements in image and language processing. Besides, these deep learning techniques have shown outstanding results in the field of signals, and [16] and [17] showed that deep learning can be useful in signal processing. Especially I apply the Recurrent Neural Network (RNN) model with Gated Recurrent Unit (GRU)[18], which is known to be suitable for processing sequence data, to remove interference and reconstruct transmit signal simultaneously. I can reconstruct transmit signal even in the presence of various interference signals, and the reconstructed signal can be used to detect objects through Fast Fourier Transform (FFT). In particular, through the learned network, signal processing can be done only with the matrix calculation, not with any iteration structure. Also, the algorithm does not require any adaptive threshold. I show that our algorithm outperforms existing algorithms in experiments where noise and interference coexist.

The rest of this paper is organized as follows. In Section 2.2, I introduce the system model considered in the paper. In Section 2.3, I show the deep learning model for our proposed algorithm. In Section 2.4, I show the simulation results for the proposed scheme. Lastly, in Section 2.5, I conclude this paper.

2.2 System Model

2.2.1 CS Radar System

One of the main radar waveforms is the CS waveform [6, 8] as shown in Fig. 2.1. If the transmit signal consists of k linear frequency chirps, frequency and phase of the

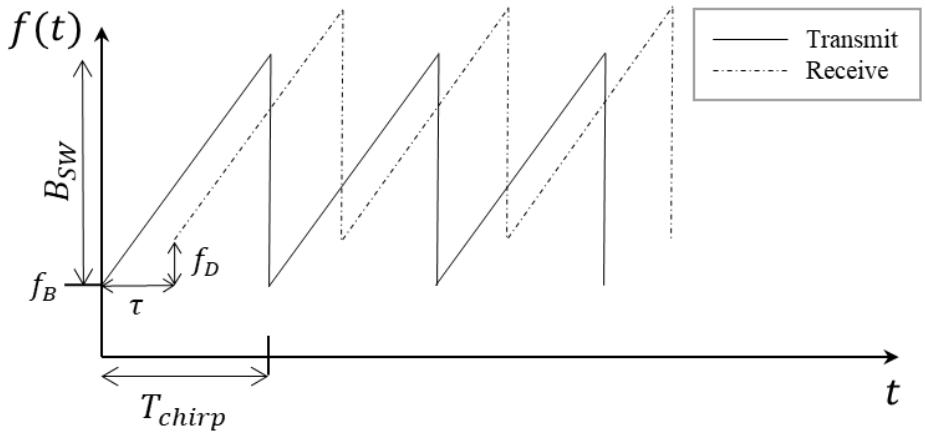


Figure 2.1: CS waveform of transmit and received signal

transmit signal are as follows.

$$\begin{aligned} f(t) &= f_B + \alpha(t - kT_{chirp}) \\ \phi(t) &= 2\pi \int_0^t f(t)dt \\ &= 2\pi(f_B t + \frac{1}{2}\alpha t^2 - \alpha k T_{chirp} t), \end{aligned} \tag{2.1}$$

where B_{SW} is sweep bandwidth, T_{chirp} is chirp duration, $\alpha = B_{SW}/T_{chirp}$ is slope of the CS waveform, and f_B is carrier frequency of the transmit signal. The beat frequency is the difference between the transmit frequency and the received frequency. The beat frequency $f_B(t)$ is represented as Fig. 2.2. A Low Pass Filter (LPF) can remove signals with higher absolute frequency value. So the remaining beat phase through the LPF can be represented as

$$\begin{aligned} \phi_B(t) &= \phi(t) - \phi(t - \tau) \\ &= 2\pi f_B \tau - \pi\alpha(\tau^2 - 2\tau t_k) \\ &\text{if } \tau < t < T_{chirp}. \end{aligned} \tag{2.2}$$

I denote target range, target velocity and speed of light as R, v, c, respectively, and the propagation delay can be represented as τ. Substituting $\tau = \frac{2(R+vt)}{c}$ and $t =$

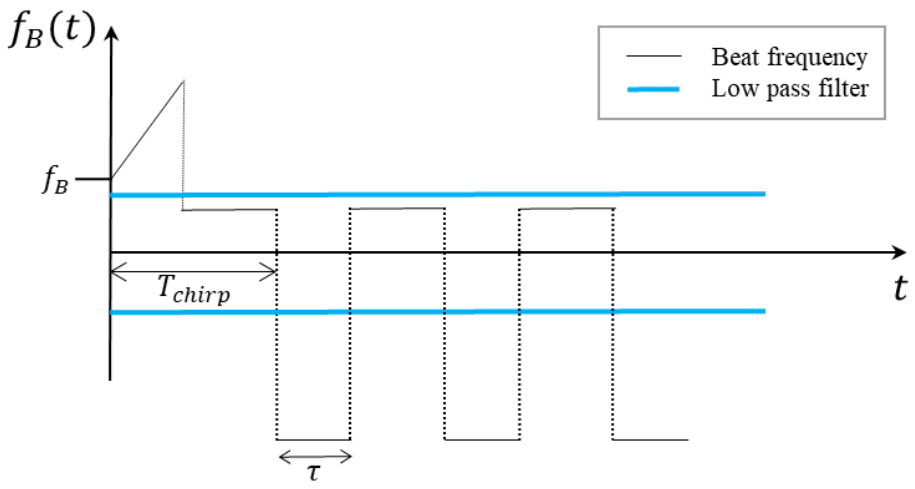


Figure 2.2: Beat frequency

$kT_{chirp}+t_k$ into equation (2.2) (if t is present in k-th chirp), (2.2) can be approximated

$$\begin{aligned}\phi_B(t) &= 2\pi f_B \left(\frac{2R}{c}+\frac{2vt}{c}\right) \\ &\quad - \pi\alpha\left(\left(\frac{2R}{c}+\frac{2vt}{c}\right)^2 - 2\left(\frac{2R}{c}+\frac{2vt}{c}\right)t_k\right) \\ &\approx 2\pi\left(\frac{2R}{c}f_B + \frac{2v}{c}f_B k\, T_{chirp}\right. \\ &\quad \left. + \left(\frac{2\alpha R}{c}+\frac{2v}{c}f_B\right)t_k\right).\end{aligned} \tag{2.3}$$

Applying sampling as $t = nT_s$, phase of the beat signal $\phi_B[n,k]$ is written as

$$\begin{aligned}\phi_B[n,k] = 2\pi\left(\frac{2R}{c}f_B + \frac{2v}{c}f_B k T_{chirp}\right. \\ \left. + \left(\frac{2\alpha R}{c}+\frac{2v}{c}f_B\right)nT_s\right).\end{aligned} \tag{2.4}$$

Using two dimensional Fast Fourier Transform (FFT), I can obtain following two values f_R and f_D,

$$\begin{aligned}f_R &= \frac{2\alpha R}{c} \\ f_D &= \frac{2v}{c}f_B T_{chirp}.\end{aligned} \tag{2.5}$$

Range R and velocity v can be obtain by f_R and f_D.

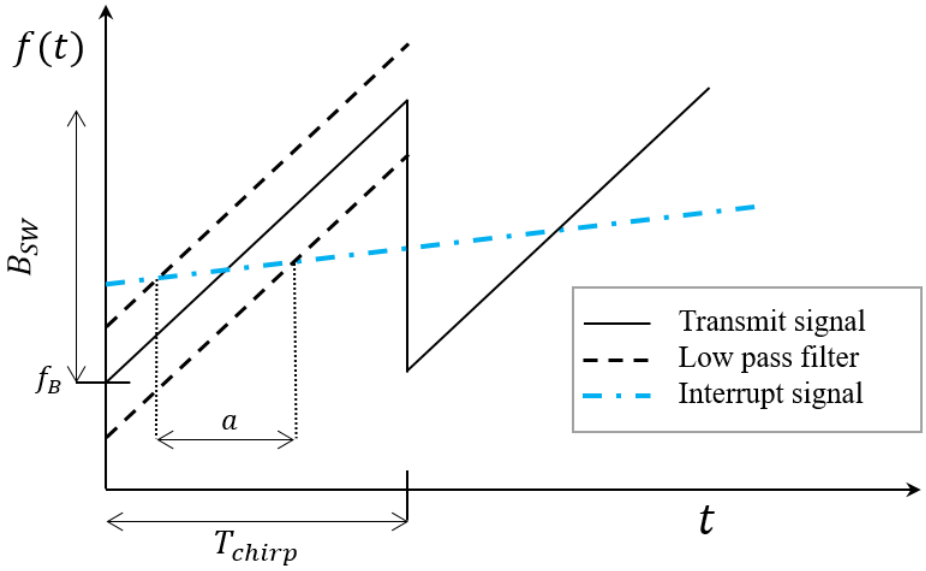


Figure 2.3: Interrupted transmit signal, interference occurs in a.

2.2.2 Interrupted Radar Signal

The equations in the previous subsection are derived in an ideal situation without interference. However, there will be a large error in distance and velocity estimation if interference occurs. In a typical driving situation, I usually encounter CS waveform signals, which have different slopes with the signal being sent, and interference situation would occur as shown in Fig. 2.3. Since the beat frequency passes through the low pass filter, the interference occurs in the section a only, not in the whole section. Fig. 2.4 shows that a large distortion occurs around 0 to 80 time samples, unlike the original beat signal. Conventionally, the interference is removed or the original beat signal is restored by using the characteristics of the time-domain beat signal. However, if noise and interference exist, the cancellation of interference and the restoration of original beat signal are difficult with a traditional method.

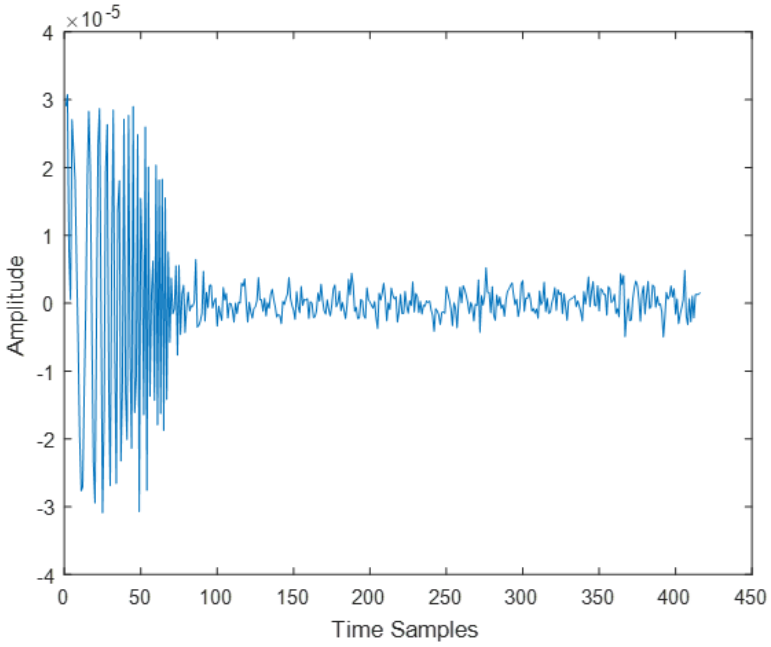


Figure 2.4: Interrupted beat signal, interference occurs around the 0 to 80 samples.

2.3 Interference Mitigation Using Deep Learning

In this section, I propose a deep neural network model which can be used for multi-interference mitigation without relying on adaptive threshold.

2.3.1 Deep Learning Model

As shown in previous studies [19], RNN is known to be suitable for sequence data processing. Since the raw data before preprocessing is consecutive time samples, I apply RNN structure for interference cancellation and restoration in our model. Following equations represents the vanilla RNN elements.

$$\begin{aligned} h_t &= f_W(h_{t-1}, x_t) \\ &= tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ y_t &= W_{hy}h_t. \end{aligned} \tag{2.6}$$

x_t is the input vector, h_t is the hidden state of the RNN network and y_t is the output vector. W_{hh} ,W_{xh} and W_{hy} are weight matrices of the hidden state to another hidden state, the input vector to the hidden state and the hidden state to the output vector, respectively. By using RNN, the network can learn the relation of consecutive samples. The input sequence may consist of hundreds of time samples. It may cause *long-term*

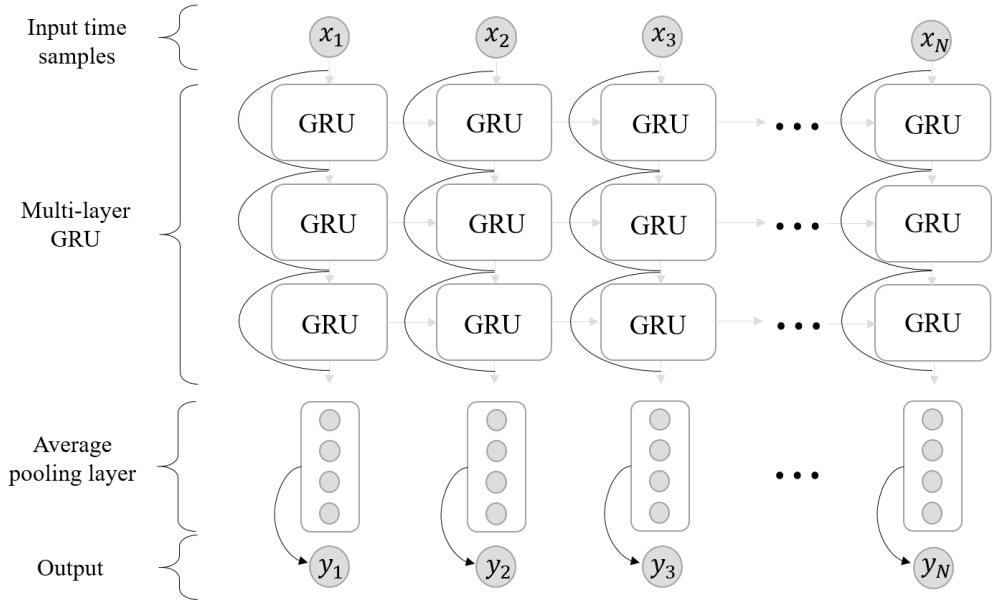


Figure 2.5: Proposed deep learning model

dependency problem in RNN [20]. So I use a GRU cell to solve this problem in RNN. GRU has the same time series structure as RNN, but the contents of the cell are different. In the multi-layer GRU layer, each layer has a bidirectional structure, rather than one direction of the signal[21]. In addition, several GRU layers were piled up to learn various interference cases. The residual network[22] is added between layers for better propagation of gradient flow. The residual connection is written as

$$X^{l+1} = X^l + GRU(X^l), (l = 1, 2, 3, ..., L-1), \tag{2.7}$$

where X^l is l-th layer input vector of GRU cells, and $GRU(X^l)$ is l-th layer output vector of GRU cells. When the total time step is N and the hidden state size is H,

the output value of GRU network is $X^L \in \mathbb{R}^{H \times N}$. If I denote $x_i^L \in \mathbb{R}^H$ as the ith column vector of X^L, X^L can be represented as $[x_1^L, x_2^L, ..., x_N^L]$. To obtain the output dimension identical to the label dimension, I perform average pooling on X^L. The average pooling output $Y \in \mathbb{R}^N$ is written as

$$Y = [average(x_1^L), average(x_2^L), ..., average(x_N^L)]. \tag{2.8}$$

To regularize the network, I applied drop out in each GRU Cells[23]. The proposed RNN model is shown in Fig. 2.5.

2.3.2 Optimizing Model

The inputs is time-sampled interference beat signal, which is represented as $X^0 = X = [x_1, x_2, ..., x_N]$, where $x_i \in \mathbb{R}$ is amplitude of beat signal$(i = 1, ..., N)$. Each input X is normalized and satisfies the following equation.

$$\sum_{i=1}^{N} x_i^2 = 1. \tag{2.9}$$

The output Y is represented as $Y = [y_1, y_2, ..., y_N]$, which has the same length as X. $\hat{Y} = [\hat{y_1}, \hat{y_2}, ..., \hat{y_N}]$ is a beat signal with the same target condition as X but without interference. I called $\hat{Y}$ as label. In order to minimize the difference between the two vectors Y and $\hat{Y}$, the loss L is defined as

$$L = \sum_{i=1}^{N} (\hat{y_i} - y_i)^2. \tag{2.10}$$

The loss L can be minimized by gradient descent. I use a gradient descent algorithm, Adam[24]. As the training progresses, I get output Y which is similar to label $\hat{Y}$. I can then use this value Y to detect target range and velocity.

2.4 Simulation Results

In this section, I introduce radar simulator parameters and deep learning model parameters. The proposed deep learning model is also compared with existing algorithms.

Table 2.1: Radar simulator random parameters

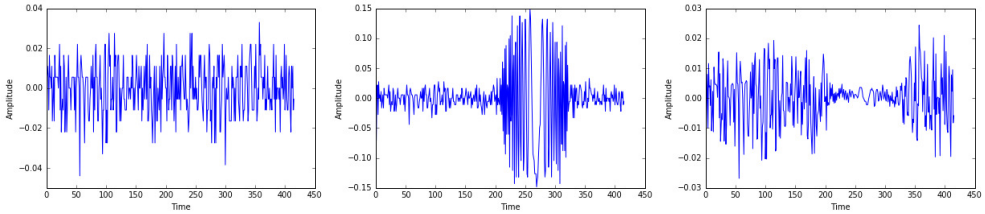
Parameter	Min	Max
Center frequency	76GHz	78GHZ
Distance	1m	130m
Velocity	0km/h	50km/h
Sweep bandwidth	100MHz	200MHz
Chirp duration	20us	40us
Target number	1	2
Interference number	1	4

I have assumed a situation with multi-target, multi-interference, and Gaussian noise in order to reflect the practical situation. I use a randomly generated 150,000 time sampled input sequence (with interference) and 150,000 label sequence (no interference). The range of random parameters for training is shown in Table 2.1. The transmit signal is the CS wave mentioned in Section 2.2 and the interference waveform is the FMCW wave signal with different chirp slope (includes CS waveform, triangle sweep FMCW). The total number of chirps was 75 in both the desired and the interfering signals. The model proposed in Section 2.3 is used and the hyperparameter used in the model is shown in Table 2.2. The deep learning model input and label are beat signals corresponding to one chirp of the transmit signal. To apply RNN, the input and label length must be constant. However, the number of samples of one chirp can vary depending on the sampling period of the signal. So I limit the maximum length of the input and label to 416 and cut the remaining part if the actual length is longer than that, and do zero-padding if it is smaller. In order to solve the *exploding gradients* problem in the GRU structure, the gradient clipping method is used [25].

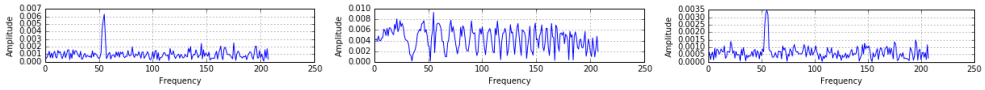
Table 2.2: Deep learning hyperparameter

Hyperparameter	Value
Batch size	128
Learning rate	1e-3
Hidden layer size	100
Number of data	150000
Number of layer	3
Drop out rate	0.3
Optimizer	Adam

I analyzed the interference mitigation performance of the proposed method. The results are shown in Fig. 2.6. I use Fig. 2.6(a) as deep learning label (not interfered), Fig. 2.6(b) as deep learning input (interfered), and the deep learning output is Fig. 2.6(c). I can see that the proposed deep learning algorithm finds out where the interference is. Under the considered situation, the reconstruction of the original signal is not perfect. However, I can see that the result of FFT in Fig. 2.6(f) finds the object more clearly than the interfered input Fig. 2.6(e). To compare result with other methods, I use the average signal to remaining interference noise ratio (SRINR)[15]. The SRINR result is in Table 2.3. Method I is time domain thresholding (TDT) method used in [11]. Method II did not use an adaptive threshold, which was proposed in [15]. The simulation SRINR is average of 50 random scenarios SRINR. Our proposed deep learning algorithm outperforms other methods. Especially, even in situations where the interference signal sources are close and the targets are too far away, our proposed method finds the target properly as shown in Fig. 2.7.



(a) Label (b) Input (c) Output



(d) FFT label (e) FFT input (f) FFT output

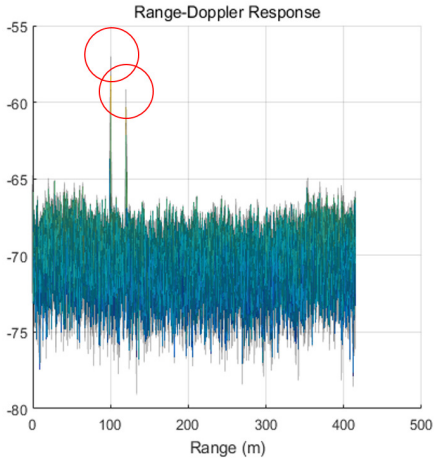
Figure 2.6: Result of deep learning model. (a) to (c) is beat signal, (d) to (f) is FFT result of (a) to (c) signals respectively.

Table 2.3: Simulation results

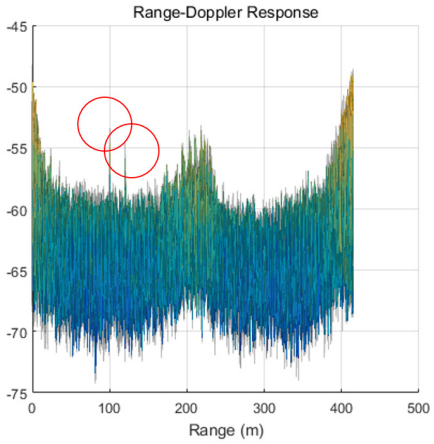
	Method I	Method II	Proposed
SRINR	23.369	22.665	26.091

2.5 Conclusion

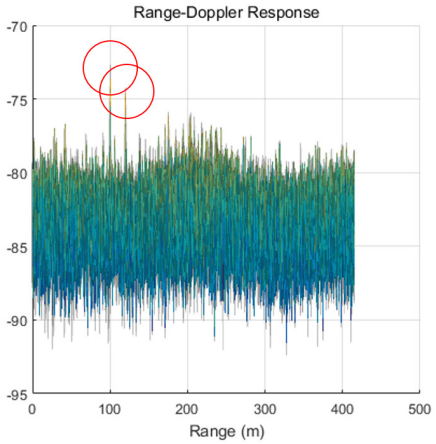
In this paper, I proposed a novel approach to mitigate interference in CS radar system. I used a deep learning approach to mitigate interference. Our method shows better performance compared to other signal processing methods. Our method also shows good performance even when the target is far away. It is believed this method can be applied not only to CS waveforms but also to most situations where frequency changes linearly. This is because interference occurs at the point where the transmit signal crosses the interference signal. The interference patterns of linear frequency signals are similar. Experiments with other waveforms are left as future work.



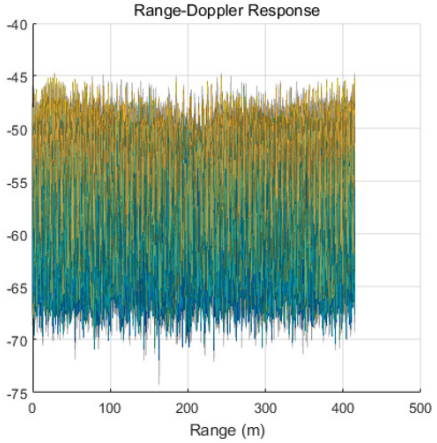
(a) Proposed



(b) Method I



(c) Method II



(d) No processing

Figure 2.7: Simulated power levels with respect to range. Two targets exist in range 100m, 120m. Four interferences exist in range 40m, 50m, 60m, and 70m. Red circles are detected targets.

Chapter 3

Automotive Radar Signal Interference Mitigation using RNN with Self Attention

3.1 Introduction

Recently, as interest in autonomous vehicles has greatly increased, safety concerns in autonomous vehicles is a rising issue. To ensure safety, radar is an key component as a sensor as it is cheap and can cope bad conditions. Radars mounted on advanced vehicles, such as autonomous vehicles, require a variety of functions, including detection of multiple targets and long-range sensing. Using the information, radar can be used in adaptive cruise control or collision mitigation systems. These functions must be performed accurately to ensure user safety and to help prevent collisions between vehicles. Recent popular radar technologies use frequency modulated continuous wave (FMCW), chirp sequence (CS) waveform [6, 7, 8] and OFDM signal [26, 27]. However, it is difficult to perform the above functions in the presence of interference [9, 10]. Several signal processing techniques have been proposed to solve problems related to interference [11, 12, 13, 14, 15, 28, 29]. However, little studies has been done to mitigate FMCW radar interference using deep learning. Also none of studies has been done to mitigate OFDM radar interference.

Recently, the development of deep learning has been remarkable and has facilitated significant achievements in image and language processing. These deep learning techniques have provided outstanding results in the field of signal processing [16, 17] showed that deep learning can be useful in signal processing. Deep learning technology has also been widely used in with radar. For example, [30] used a convolutional neural network(CNN) for gesture recognition and [31] used a CNN for vehicle classification by using FMCW radar.

Our model used a recurrent neural network (RNN) model with a gated recurrent unit (GRU) [18] to process sequence data. However, our previous model could not sufficiently reconstruct the original signal in the presence of interference[32]. In order to restore the original signal perfectly, information on the surrounding signals must be sufficiently learned. Inspired by the self attention model used in recent google deep learning papers[33], I combined a transformer module to our model. After applying the attention mechanism, the restoration of the interfered signal is nearly complete, and the range and velocity of the target can be clearly detected after applying a fast fourier transform (FFT). To validate our model, I experimented with two different environment. First is FMCW radar, and second is OFDM radar.

The main contribution of this paper are

- I presented a novel RNN based deep learning algorithm that can mitigate FMCW radar interference.
- I showed that our algorithm can not only remove the interference but also restore original signal
- I showed that our algorithm has better performance than other existing signal processing method and deep learning method.
- I showed that our algorithm is capable of OFDM radar system interference mitigation

3.2 System Model

3.2.1 FMCW Radar System

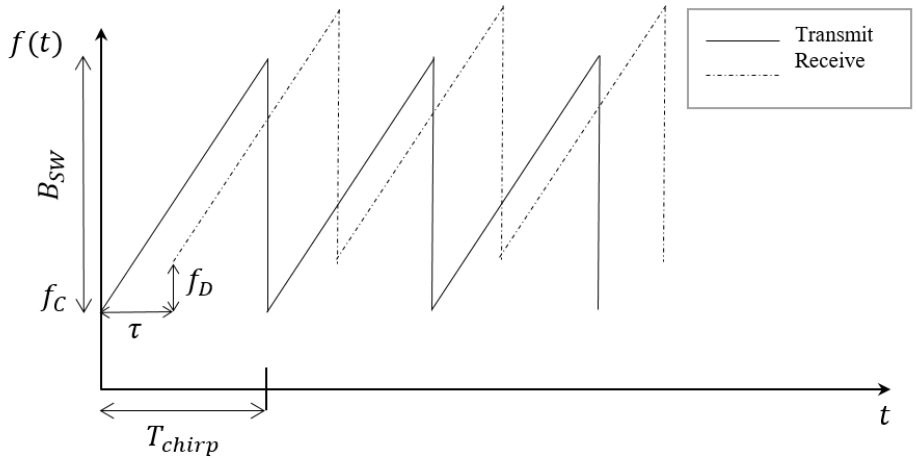


Figure 3.1: FMCW waveform of transmit and received signal

One of the main radar waveforms is the FMCW waveform [6, 8] as shown in Fig. 3.1.

If the transmit signal consists of k linear frequency chirps, frequency and phase of the transmit signal are as follows.

$$\begin{aligned} f(t) &= f_C + \alpha(t - kT_{chirp}) \\ \phi(t) &= 2\pi \int_0^t f(t)dt \\ &= 2\pi(f_C t + \frac{1}{2}\alpha t^2 - \alpha k T_{chirp} t), \end{aligned} \tag{3.1}$$

where B_{SW} is sweep bandwidth, T_{chirp} is chirp duration, $\alpha = B_{SW}/T_{chirp}$ is slope of the FMCW waveform, and f_C is carrier frequency of the transmit signal. The beat frequency is the difference between the transmit frequency and the received frequency. A anti aliasing filter (AAF) can remove higher frequency signals, thus the remaining beat phase through the AAF is

$$\begin{aligned}\phi_B(t) &= \phi(t) - \phi(t-\tau) \\ &= 2\pi f_C \tau - \pi\alpha(\tau^2 - 2\tau t_k) \\ &\text{if } \tau < t < T_{chirp}.\end{aligned} \tag{3.2}$$

I denote target range, target velocity and the speed of light as R, v and c, respectively, and the propagation delay is τ. Using 2d-fft on beat signal, I can get target range and velocity. The equations 3.1, 3.2 are derived in an ideal situation without interference. However, if there are several radar-based autonomous vehicles on the road, radar signals will interfere with each other and cause severe problems in range and velocity measurements. Interference will occur if FMCW such as FMCW with different slopes are encountered. Because AAF removes the high frequency portion, actual interference occurs in the part of the whole signal. Fig. 3.2 shows an example of the remaining interference in the beat signal in the time domain. It is difficult to completely remove the interference because it appears as an irregular waveform.

3.2.2 OFDM Radar System

As described in the previous section, overlapping frequency bands causes performance degradation. One solution to this problem is the orthogonal frequency division multiplexing (OFDM) radar. In communications, OFDM has become a popular transmission technique. Similarly, radar can also use orthogonal signals between users to avoid interference [26, 27]. The possible application areas for these signals are both in the radar networks with spatially distributed sensors as well as in multi-input multi-output (MIMO) radar sensors, both of which employ multiple transmit antennas fed with individual signals in order to achieve high robustness and improved spatial resolution.

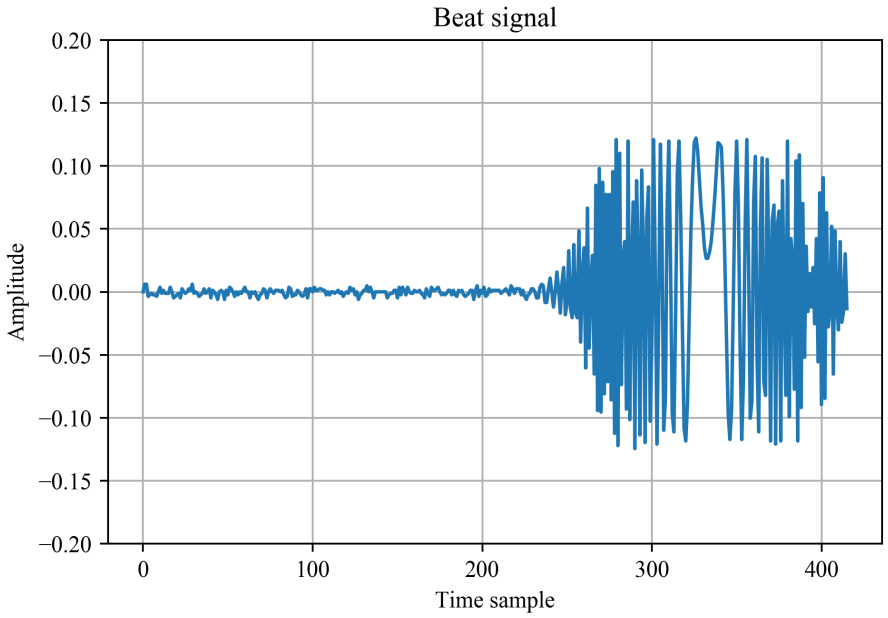


Figure 3.2: Interrupted beat signal. Interference occurs from sample 250 to sample 410.

The received signal of the OFDM radar is expressed as

$$\begin{aligned} d_{R_x}(uN_c + n) = A(u,n) d_{T_x}(uN_c + n) \\ \times exp(-j2\pi n \Delta f \frac{2R}{c}) \\ \times exp(j2\pi u T_{OFDM} \frac{2v_{rel} f_c}{c}). \end{aligned} \tag{3.3}$$

In this expression N_c denotes the number of subcarriers, with n being the index of the individual subcarrier, u is the OFDM symbol index, T_{OFDM} is the total transmitted OFDM symbol duration, and d_{Tx} is a sequence of complex-valued modulation symbols obtained from binary data. To determine the range R and the relative velocity v_{rel} of the reflecting object, the influence of the transmitted modulation symbols d_{Tx} must be removed from this result. This is accomplished by element-wise division of

transmitted and received signals

$$d_{div}(uN_c + n) = \frac{d_{Rx}(uN_c + n)}{d_{Tx}(uN_c + n)}. \tag{3.4}$$

I can get range and velocity by applying 2d-fft on matrix d_{div}. In an ideal situation, interference would not occur due to proper channel assignments between users. However, in practice, interference may occur due to intentional interference or system problems. I consider this situation and assume that $k(k < n)$ out of n channels(n is total number of channels in one user) are corrupted.

3.3 Deep Learning Algorithm

3.3.1 Previous Deep Learning Model

The beat signal we want to mitigate interference is a sequence of longitudinal data, not an image. Therefore, we use the RNN model to learn the relationship between these sequences. RNN has been used in speech recognition, translation, and many other fields, and has been used for processing sequence data [19]. The following equations represents the vanilla RNN elements.

$$\begin{aligned} h_t &= f_W(h_{t-1}, x_t) \\ &= tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ y_t &= W_{hy}h_t. \end{aligned} \tag{3.5}$$

where x_t is the input vector, h_t is the hidden state of the RNN network and y_t is the output vector. W_{hh} ,W_{xh} and W_{hy} are weight matrices relating one hidden state to another hidden state, the input vector to the hidden state and the hidden state to the output vector, respectively. $tanh$ is used as an activation function, which determines whether the output signal is activated in the next step. Information from each time step is stored in h_{t-1} and updated whenever new input information x_t is received.

However, *long-term dependency* or *vanishing gradient* problems arise because the temporal data of the beat signal that we want to use as input data usually contain

hundreds of samples. One way to solve this is to use GRU cells instead of basic RNN cells. The workflow of GRU is expressed as follows:

$$\begin{aligned} r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\ u_t &= \sigma(W_u x_t + U_u h_{t-1} + b_u) \\ c_t &= tanh(W_c x_t + U_c(r_t \cdot h_{t-1}) + b_c) \\ h_t &= u_t \cdot h_{t-1} + (1 - u_t) \cdot c_t, \end{aligned} \tag{3.6}$$

where x_t is the input vector, and h_t is the hidden state of the GRU network. The two vectors are transformed linearly with respect to the matrices W_r and U_u, and a bias vector b_r is subsequently added to obtain the reset gate r_t via a sigmoid function $\sigma(\cdot)$. u_t is the update gate composed of W_u, U_u in a similar way as r_t. c_t is a memory vector composed of W_c, U_c, and the previous hidden state h_{t-1}. The current hidden state h_t is composed of the previous hidden state h_t, reset gate r_t, update gate u_t and memory c_t. The reset gate determines how the new input merges with the previous memory, and the update gate determines how much of the previous memory will be retained. Each layer in the multi-layer GRU layer has a bidirectional structure, rather than one direction. In addition, several GRU layers were piled up to facilitate learning of interference in various cases. The residual network is added between layers for better gradient flow propagation. The residual connection is written as follows:

$$X^{l+1} = X^l + GRU(X^l), (l = 1, 2, 3, ..., L - 1), \tag{3.7}$$

where X^l is l-th layer input vector of the GRU cells and $GRU(X^l)$ is l-th layer output vector from the GRU cells. When the total number of time steps is N and the size of a hidden state is H, the output value from the GRU network is $X^L \in \mathbb{R}^{H \times N}$. We denote $x_i^L \in \mathbb{R}^H$ as the ith column vector of X^L. X^L is represented as $[x_1^L, x_2^L, ..., x_N^L]$. We perform average pooling on X^L to obtain the output dimension identical to the label

dimension. The average pooling output $Y \in \mathbb{R}^N$ is written as follows:

$$Y = [average(x_1^L), average(x_2^L), ..., average(x_N^L)]. \tag{3.8}$$

3.3.2 Model

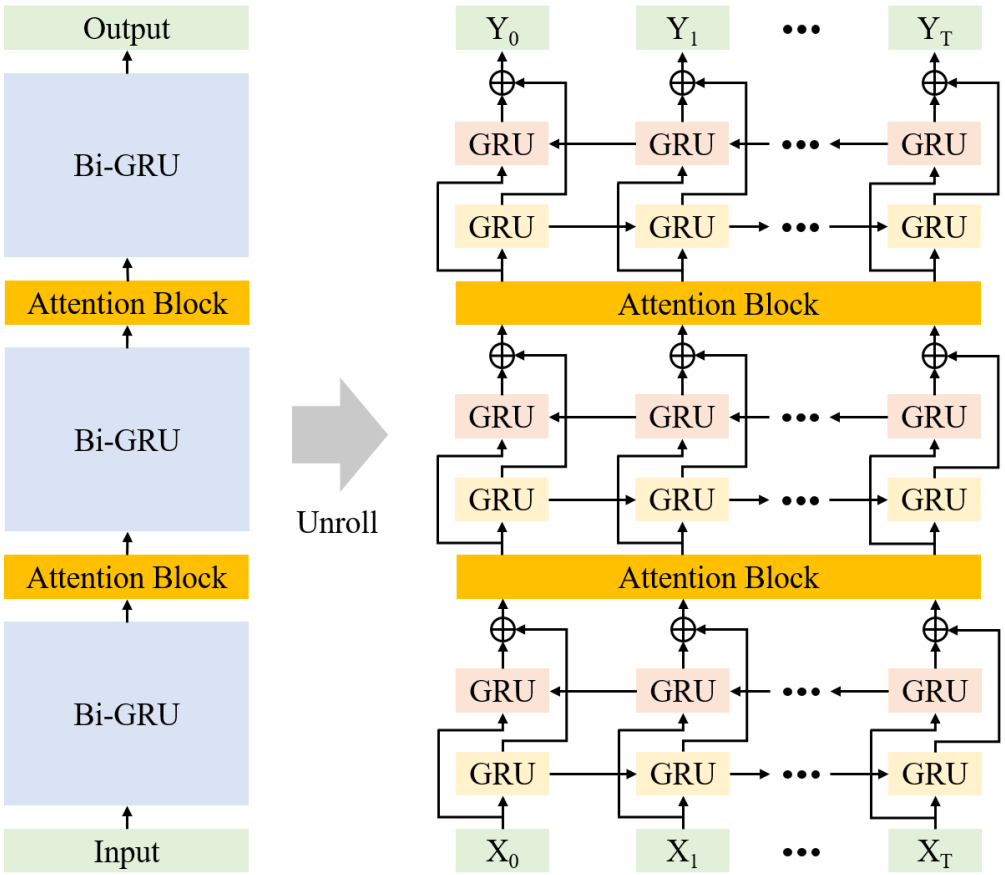


Figure 3.3: Deep learning architecture.

Since the signal I want to recover varies over time, I used an RNN-based model. However, the basic RNN cell has a gradient vanishing problem where the information disappears as the time step increases. One way to solve this problem is to use GRU cells instead of basic RNN cells. I stack multiple GRU layers. In addition, since the interference partially occurs in the entire data, model needs to capture the relationship

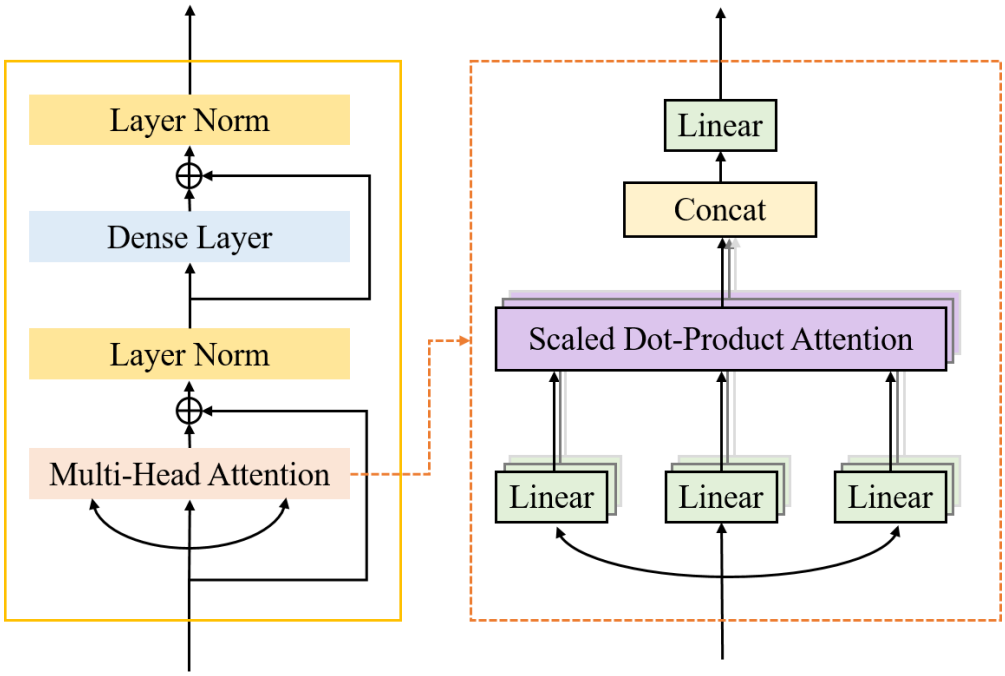


Figure 3.4: Attention Block.

between the entire time steps. Inspired by state of art natural language processing (NLP) paper[33], I add attention blocks to our RNN layer. Our final deep learning architecture is shown in Fig. 3.3 and Fig. 3.4. The key to our attention block is Scaled Dot-Product Attention[33]. The equation of Scaled Dot-Product Attention is expressed as

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \tag{3.9}$$

Q, K, and V are quary, key, and value, respectively. In our model, Q, K, and V are same previous time step vector. d_k is dimension of key vector. By using the attention block, model can better learn the relationship between time step values.

In order to minimize the difference between the deep learning inputs and label, I defined the loss L as

$$L = \sum_{i=1}^{T} (\hat{y}_i - y_i)^2. \tag{3.10}$$

y_i is deep learning output, and $\hat{y}_i$ is label. L can be minimized using a gradient descent algorithm ADAM[24].

3.3.3 Preprocessing

FMCW radar preprocessing

Firstly, I applied a median filter to remove the high power interference. The amplitude of the time samples of the beat signal varies greatly, depending on the power of the transmit signal or signal attenuation due to the distance between the sender and the target. Therefore, I must set the input time sample values to the appropriate size. Algorithm 1 shows the overall flow of preprocessing and applying deep learning model.

OFDM radar preprocessing

FMCW beat signal from the previous section is one-dimensional, whereas OFDM data d_{div} is two-dimensional. In addition, in the OFDM radar, if subcarrier number N_c is 1024 and symbol number N_{sym} is 256, d_{div} has a large data size of $(256, 1024)$, making learning difficult. So instead of putting d_{div} as input, I used row vector of d_{div} as input . In the final step, 256 row vectors were combined into a matrix. The other process is the same as in section 3.3.3.

3.4 Experiment Results

3.4.1 Simulation Setup

A simulation environment for measuring the target distance and speed with a CS radar system is shown in Fig. 3.5. The transmitter generates a CS waveform and the generated signal is reflected from the target. The reflected signals are combined through a mixer and passed through an anti aliasing filter (AAF). The AAF is an 8th order But-

Algorithm 1 Overall interference mitigation algorithm using deep learning

Use simulator, make data

Input : $X = [x_1, x_2, \ldots, x_T]$

Output : $Y = [y_1, y_2, \ldots, y_T]$

Label : $\hat{Y} = [\hat{y_1}, \hat{y_2}, \ldots, \hat{y_T}]$

$\alpha = 100 * median(abs(X))$

$X[X > \alpha] = 0$

Normalize input and label

if $\sum_{i=1}^{T} x_i^2 = K$ **then**

    $x_i \leftarrow \frac{x_i}{\sqrt{K}}$

    $\hat{y_i} \leftarrow \frac{\hat{y_i}}{\sqrt{K}}$

end if

for X, $\hat{Y}$ in data **do**

    Get output Y using deep learning network

    Optimize loss L using gradient descent algorithm ADAM

end for

Restore $y_i \leftarrow \sqrt{K} y_i$

Get target range and velocity using Y

Table 3.1: Random parameters in the radar simulator

Parameter	Min	Max
Center frequency	76GHz	78GHZ
Distance	1m	130m
Velocity	0km/h	50km/h
Sweep bandwidth	100MHz	200MHz
Chirp duration	20us	40us
Train target number	1	2
Train interference number	1	4
Test target number	1	5
Test interference number	1	8

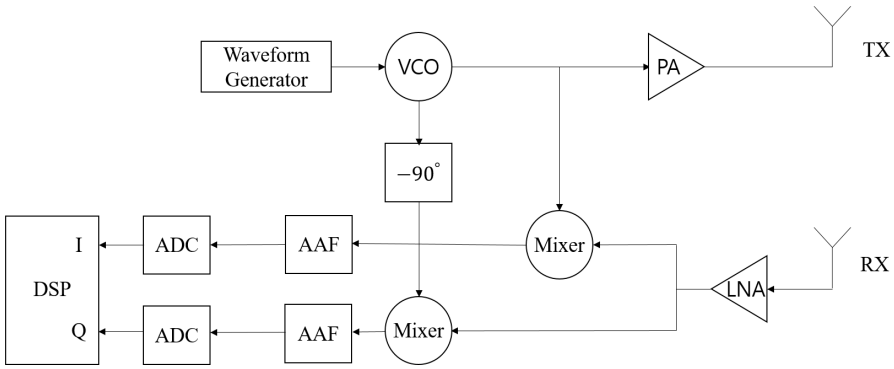


Figure 3.5: FMCW radar system model

terworth low pass filter with impulse invariance. After passing through the AAF, we obtain the target range and velocity using a two dimensional (FFT). To create a situation that accurately represents the actual environment, we added Gaussian noise to the signal and phase noise was added when generating the transmit signal. We assumed multiple targets in multiple interference situations. We selected different numbers of targets and interference sources during training and testing to show that our deep learning algorithm can be used to detect targets in various situations. The maximum number

Table 3.2: Deep learning hyperparameter

Hyperparameter	Value
Batch size	128
Learning rate	1e-3
Hidden layer size	100
Number of data	150000
Number of layer	3
Optimizer	Adam

of targets was set to 2 and the maximum number of interference sources was set to 4 during training. During testing, the maximum number of targets was set to 5 and the maximum number of interference sources was set to 8. Other radar parameters were randomly generated. We gathered random data in Matlab using a phased array system toolbox. Details of the random radar parameters are shown in Table 3.1. The transmit signal is the CS wave mentioned in previous Section and various waveforms with different chirp slopes (includes CS waveform, triangle sweep FMCW, MFSK) were used for the interference waveform. The total number of chirps in the transmitted and interfered signals was set to 75. The model proposed in previosu Section was used for deep learning, and the hyperparameter used in the model is shown in Table 3.2. We created the model using the Google tensorflow api. Because the platform for creating radar data is different from the platform used for deep learning, we make the data from Matlab and train data from tensorflow and then passed the result data back into Matlab. The input data and label data for deep learning are beat signals with length of one chirp and are one-dimensional vectors. However the length of the input and label must be fixed in order to run the RNN. Therefore ,we limited the length of the input and label to an overall average value of 416. The input and label were padded with zeroes if the input and label were short.

3.4.2 Implementation Details

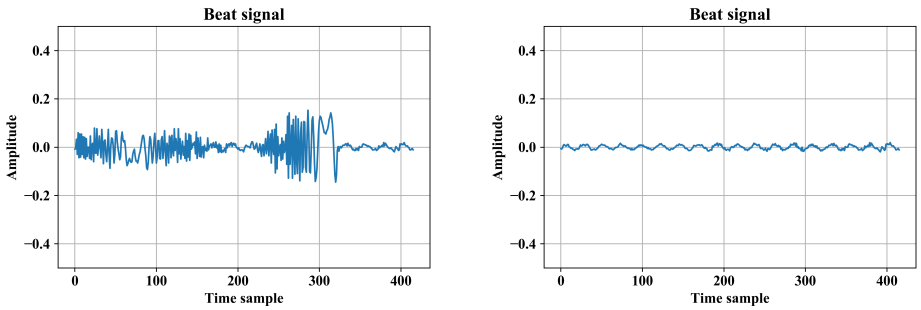
FMCW radar

I assume there are up to five targets and eight interferences. There are three types of interference, Chirp Sequence (CS), triangle FMCW and multiple frequency-shift keying (MFSK) respectively. I gathered random data in MATLAB using a phased array system toolbox.

OFDM radar

Our parameters of OFDM radar are the same as previous OFDM radar paper[27]. In the simulator, a transmitted signal has eight channels, and $k(k < 4)$ channels are corrupted by interference.

3.4.3 FMCW Radar Result



(a) Input (b) Output

Figure 3.6: FMCW beat signal. Input is interrupted signal, and Output is deep learning result.

The original signal is restored after interference cancellation by deep learning. The beat signal result is shown in Fig. 3.6. The Table 3.3 shows average SINR of FMCW radar range and doppler response. Our Attention Bi-RNN algorithm shows better SINR performance than the existing algorithm.

Table 3.3: Average SINR values from 50 trials

	CS	triangle FMCW	MFSK
No algorithm	21.613	21.252	20.751
Method I [11]	25.415	25.038	25.076
Method II [15]	27.429	27.231	28.144
Attention Bi-RNN	**36.700**	**35.147**	**37.425**

Table 3.4: Runtime

	runtime(s)
Method I [11]	0.016
Method II [15]	0.02
Attention Bi-RNN	0.031

3.4.4 Comparison with Other Algorithms

In this section, we tested the robustness of the algorithm by differentiating the training and test environments of the radar system. This consists of four environments. The number of targets and interference sources in each environment are different. The detailed number of targets and interference sources are shown in Table 3.5. In particular, to illustrate the robustness of the deep learning algorithm, the training data contained up to two targets with up to four interference sources, as shown in Table 2.1. We use the average signal to remaining interference noise ratio (SRINR) to compare result with other methods [15], which is equivalent to the ratio of the reflected power from an object to the average remaining interference power. If multiple targets are present, we only consider targets with the highest reflected power. We compared results from the proposed algorithm with those from two other algorithms Method I and Method II. Method I is time domain thresholding (TDT) method used in [11]. Method II does not

Table 3.5: Target and interferences number in training and test time

	training target number	training interference number	test target number	test interference number
Setup1	1 to 2	1 to 4	1 to 2	1 to 4
Setup2	1 to 2	1 to 4	3 to 5	1 to 4
Setup3	1 to 2	1 to 4	1 to 2	5 to 8
Setup4	1 to 2	1 to 4	3 to 5	5 to 8

define an adaptive threshold, which was proposed in [15].

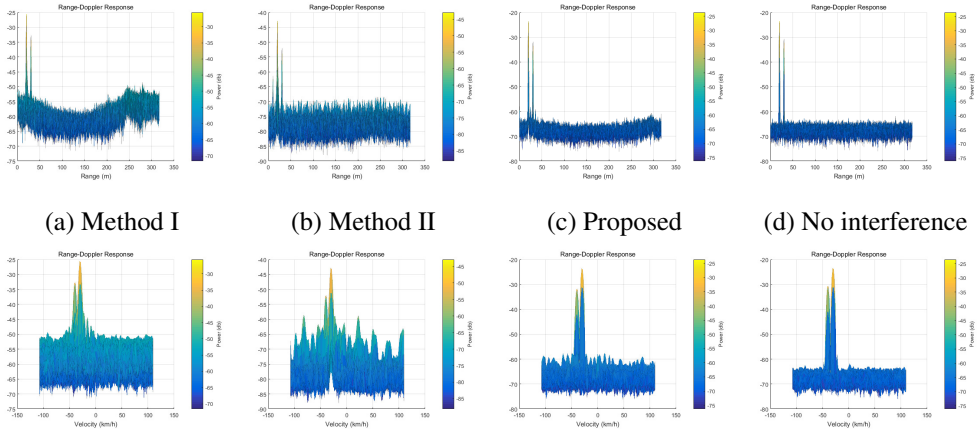
Setup1

We assumed that 1-2 targets, and 1-4 interference sources were present in test case Setup1. ”No interference” assumes a situation where there is no interference with the target alone and it has maximum theoretical SRINR. The simulate SRINR value is an average of SRINR values from 50 random scenarios. We tested each algorithm with three interference waveforms (CS, triangle FMCW, MFSK waveforms). The average SRINR test results are shown in Table 3.6. In Setup1, our proposed algorithm produces average SRINR values of 24.245, 23.424, and 23.494 for CS, Triangle FMCW, and MFSK waveforms, respectively. Fig. 3.7 shows range and velocity measurements of the target. The graphs (a)-(d) on the Fig. 3.7 represent the amplitude of the target beat signal over the target range. Additionally, the graphs (e)-(h) on the Fig. 3.7 show that the relationship between the amplitude of the target beat signal with the target velocity. The unit of the amplitude of the beat signals is dB scale indicated by colored bars located on the right hand side of the each graph. Our proposed algorithm provides better performance than other algorithms, regardless of the type of interference and our proposed algorithm has SRINR value close to the situation without interference

(i.e., ”No interference”).

Table 3.6: Setup1 : average SRINR values from 50 trials(1 to 2 targets and 1 to 4 interference sources)

	CS	triangle FMCW	MFSK
No algorithm [dB]	17.355	17.169	16.958
Method I [dB]	19.735	19.727	19.217
Method II [dB]	19.685	19.663	19.593
Proposed [dB]	**24.245**	**23.424**	**23.494**
No interference [dB]	25.212	25.212	25.212



(a) Method I (b) Method II (c) Proposed (d) No interference

(e) Method I (f) Method II (g) Proposed (h) No interference

Figure 3.7: Setup1. Simulated power levels with respect to range and velocity. Two targets exist at (range : 20m, velocity : -30km/h), (range : 30m, velocity : -40km/h). Four interference sources exist at 30m, 40m, 50m, and 60m.

Setup2

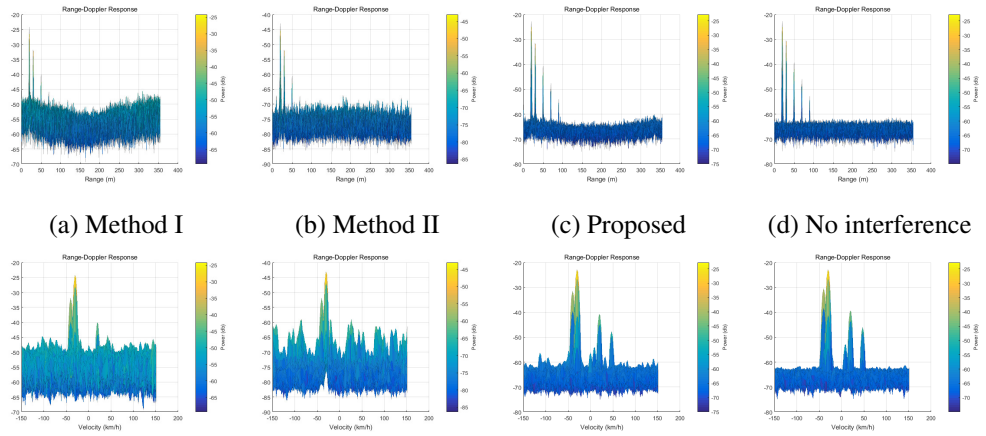
We assumed that 3-5 targets and 1-4 interference sources were present in test case Setup2. The average SRINR test results are shown in Table 3.7. In Setup2, our proposed algorithm produces average SRINR values of 35.159, 34.389, and 34.540 for CS, triangle FMCW, and MFSK waveforms, respectively. Fig. 3.8 shows range and velocity measurements of the target. Compared to Setup1, the number of targets increased, and the likelihood that a target could produce a larger peak also increased. Thus, the overall SRINR increased. Regardless of the number of targets, our proposed algorithm prodecues higher SRINR values than other algorithms.

Table 3.7: Setup2 : average SRINR values from 50 trials(3 to 5 targets and 1 to 4 interference sources)

	CS	triangle FMCW	MFSK
No algorithm [dB]	26.051	25.677	25.015
Method I [dB]	29.207	29.160	28.832
Method II [dB]	28.862	28.855	28.279
Proposed [dB]	**35.159**	**34.389**	**34.540**
No interference [dB]	36.211	36.211	36.211

Setup3

We assumed that 1-2 targets and 5-8 interference sources were present in test case Setup3. The average SRINR test results are shown in Table 3.8. In Setup3, our proposed algorithm produces average SRINR values of 22.334, 20.551, and 21.830 for CS, triangle FMCW, and MFSK waveforms, respectively. Fig. 3.9 shows range and velocity measurements of the target. Compared to Setup1, the overall SRINR decreased because of the number of interference sources increased. However, even if the number of interference sources increases, our proposed algorithm provides higher performance



(a) Method I (b) Method II (c) Proposed (d) No interference

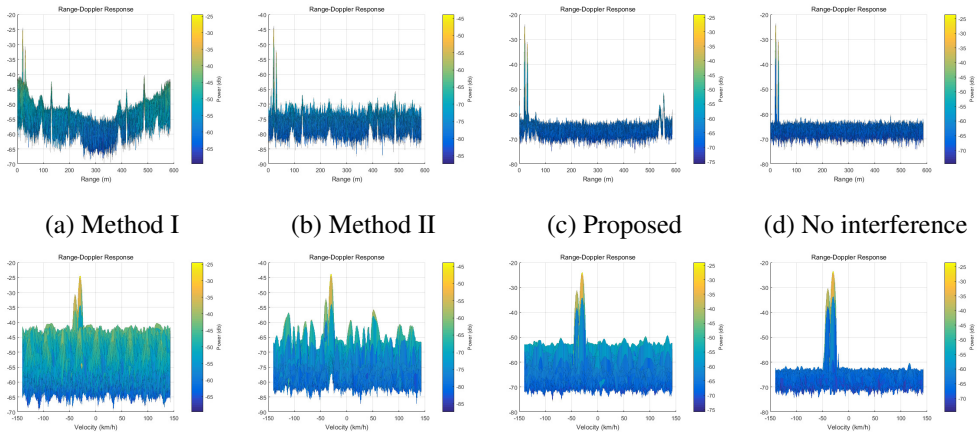
(e) Method I (f) Method II (g) Proposed (h) No interference

Figure 3.8: Setup2. Simulated power levels with respect to range and velocity. Five targets exist at (range : 20m, velocity : -30km/h), (range : 30m, velocity : -40km/h), (range : 50m, velocity : 20km/h), (range : 70m, velocity : 50km/h), and (range : 90m, velocity : 10km/h). Four interference sources exist at 30m, 40m, 50m, and 60m.

than other algorithms.

Table 3.8: Setup3 : average SRINR values from 50 trials(1 to 2 targets, 5 to 8 interference sources)

	CS	triangle FMCW	MFSK
No algorithm [dB]	13.941	13.621	13.379
Method I [dB]	15.942	15.999	16.963
Method II [dB]	16.157	15.692	15.350
Proposed [dB]	**22.334**	**20.551**	**21.830**
No interference [dB]	25.212	25.212	25.212



(a) Method I (b) Method II (c) Proposed (d) No interference

(e) Method I (f) Method II (g) Proposed (h) No interference

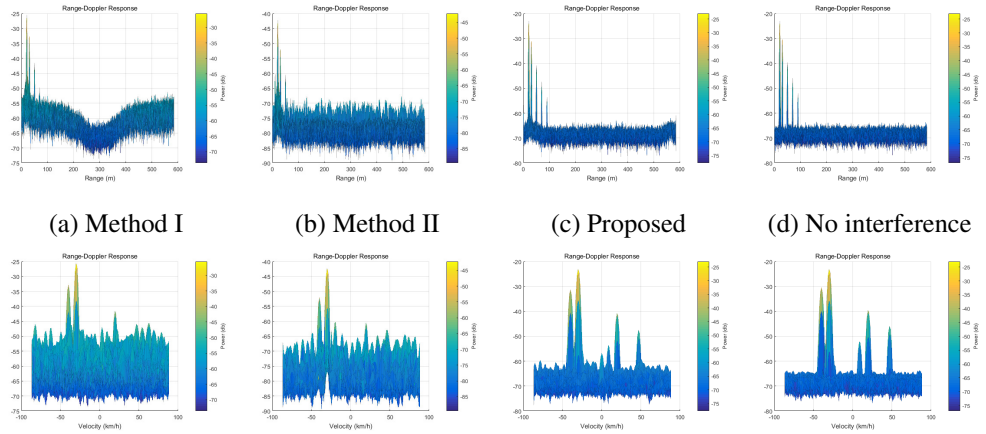
Figure 3.9: Setup3. Simulated power levels with respect to range and velocity. Two targets exist at (range : 20m, velocity : -30km/h), (range : 30m, velocity : -40km/h). Eight interference sources exist at 30m, 40m, 50m, 60m, 55m, 45m, 35m, and 15m.

Setup4

We assumed that 3-5 targets and 5-8 interference sources were present in test case Setup4. The average SRINR test results are shown in Table 3.9. In Setup4, our proposed algorithm produces average SRINR values of 33.361, 31.564, and 33.058 for CS, triangle FMCW, and MFSK waveforms, respectively. Fig. 3.10 shows distance and velocity measurements of the target. Compare to Setup2 and Setup3, the average SRINR value is slightly lower than that in Setup3 because the number of targets and the number of interference sources increases compared to Setup1. Our proposed algorithm showed better performance than the other algorithms even when 1-2 targets and 1-4 interference were used for training, while 3-5 targets and 5-8 interference sources were used for testing. In other words, our proposed deep learning methods provides greater performance, even when the number of training and test data are completely different.

Table 3.9: Setup4 : average SRINR values from 50 trials(3 to 5 targets, 5 to 8 interference sources)

	CS	triangle FMCW	MFSK
No algorithm [dB]	21.373	21.050	20.182
Method I [dB]	24.981	24.911	25.873
Method II [dB]	24.218	23.783	23.620
Proposed [dB]	**33.361**	**31.564**	**33.058**
No interference [dB]	36.211	36.211	36.211



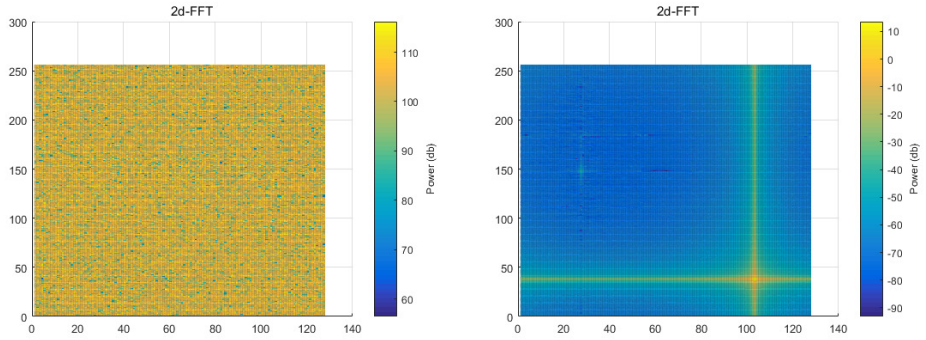
(a) Method I (b) Method II (c) Proposed (d) No interference

(e) Method I (f) Method II (g) Proposed (h) No interference

Figure 3.10: Setup4. Simulated power levels with respect to range and velocity. Five targets exist at (range : 20m, velocity : -30km/h), (range : 30m, velocity : -40km/h), (range : 50m, velocity : 20km/h), (range : 70m, velocity : 50km/h), and (range : 90m, velocity : 10km/h). Eight interference sources exist at 30m, 40m, 50m, 60m, 55m, 45m, 35m, and 15m.

3.4.5 OFDM Radar Result

The Fig. 3.11 is 2d-fft result of interfered channel of d_{div}. After applying the deep learning algorithm, the peak was clearly detected.



(a) Interrupted (b) After deep learning

Figure 3.11: 2d-fft result of d_{div}.

3.5 Conclusion

This paper presents a novel RNN based algorithm to mitigate interference in FMCW and OFDM radar environment. By adding the attention module to the previous GRU model, I can better capture the relationship between time sequences. The proposed model not only removes the interference but also recovers the original signal and shows better performance than the state-of-the-art method. In addition I applied the deep learning algorithm for mimo ofdm interference cancellation for the first time. Our future work will focus on testing our algorithm on real-world data.

Chapter 4

Neural Architecture Search with Dynamic Exploration

4.1 Introduction

The optimal neural architecture is different for each deep learning task, and designing it requires substantial knowledge of experts and a lot of computation time. *Neural architecture search* (NAS) has emerged to solve this problem, and it aims to automate the neural architecture design and weight optimization.

Early NAS approaches set the search problem as *hyperparameter optimization*, where the weights are trained under the sampled architectures and the performance of each architecture is measured by validation error. For effective hyperparameter optimization, reinforcement learning was commonly used [4, 34], but these approaches consume enormous computation time and cost. To reduce the enormous computation, recent NAS approaches [35, 36, 37, 5, 38, 39], called *one-shot NAS*, perform weight training and architecture search jointly by *weight sharing* for all possible architectures. It can successfully save huge amount of computation by not having to train new weights from scratch for different architectures.

Differentiable Architecture Search (DARTS) [5], one of the recent one-shot NAS approaches, has received a lot of attention for its simple methodology and small search cost. Based on *directed acyclic graph* (DAG) for weight sharing [37], it proposed

gradient-based search via continuous relaxation over discrete search space and first-order approximation of bi-level optimization. However, the gradient-based search of DARTS has a bias issue. As DARTS solves a bi-level optimization problem of weight training and architecture search in a nested manner, and the gradient descent has a greedy nature, the search process is inevitably biased.

The issue that the gradient-based search is biased has been addressed in the previous literature. With empirical results, several papers [40, 41] point out that the gradient-based search results are biased to prefer skip-connect operation. As skip-connection helps neural networks to rapidly converge with gradient update, lots of skip connect operations are found in the architectures selected by DARTS.

To tackle the bias issue, I dynamically transform the architecture during the search, so that the search process does not consider only a limited architecture when training the weight of operations. By introducing a dynamic architecture into the search process, I tried to mitigate the bias of the search that bi-level optimization is performed in a nested manner. For dynamic architecture, I propose Dynamic Attention Networks (DANs). Each DAN is implemented on each edge of the DAG, and dynamically changes the neural architecture depending on the input during the early stages of the search process. By considering various architectures during the search, it allows the gradient-based search to have an exploration effect. To reduce exploration at the end of the search and decide network architecture, the attention weight for DAN gradually decreases as the search process proceeds, reaching zero at the end of the search.

The main contributions of our paper are as follows.

- I introduce a dynamic architecture into the search process, which alleviates the drawbacks from the bias of the gradient-based search where bi-level optimization is performed in a nested manner.

- I propose Dynamic Attention Networks (DANs), which change the neural architecture based on inputs so that the gradient-based search has an effective exploration effect.

- Our proposed algorithm achieves state-of-the-art performance in multiple image classification datasets: CIFAR-10, CIFAR-100, and ImageNet.

4.2 Related Work

4.2.1 Neural Architecture Search

Early researches for NAS focused on reinforcement learning. The representative works [4, 34] of reinforcement learning-based search construct a network structure with a RNN controller and validation accuracy is used as rewards for training a RNN controller. In addition, evolutionary algorithm [42] was proposed to combine evolution theory and reinforcement learning. However, it takes too much time for the search. Recently, gradient-based search methods attracted many researchers for simple implementation and powerful performances.

One of the gradient based search methods, DARTS [5], proposed continuous relaxation to make architecture search differentiable. Also, it proposed 2nd order approximation for faster convergence of the search progress. SNAS [38] introduced the stochastic search through a gumbel-softmax [43]. It reduces the inconsistency of subgraphs and a supergraph. DATA [44] proposed ensemble gumbel-softmax for more effective stochastic search. PC-DARTS [45] samples small parts of the super-network to reduce the redundancy in exploring the network space.

4.2.2 Exploration and Exploitation

The gradient update method of the existing DARTS algorithm always learns architecture parameter α in a direction of reducing validation loss. Using such greedy algorithm can result in some of the options not being explored. In this case, architecture learning can fall into local minimum rather than global minimum. Reinforcement learning finds more possibilities by using ϵ-*greedy*, which involves not only $exploitation$ but also $exploration$, to solve this problem. The ϵ-*greedy* of reinforce-

ment learning is expressed as the following equation.

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{m} + (1-\epsilon) & , a^* = \text{argmax}\left\{Q_\pi(s,a)\right\} \\ \frac{\epsilon}{m} & , \text{otherwise} \end{cases} \tag{4.1}$$

Where $\pi(a|s)$ is policy given action a and state s. m is the number of total action and Q function $Q_\pi(s,a)$ is the sum of the reward. This policy ensures that with probability $\frac{e}{m}$, I choose a random action. In reinforcement learning, exploration as well as exploitation is needed to find the optimal. *ϵ-greedy* is one of the methods of exploration, and the following proof shows that epsilon-greedy still result policy improvement.

Theorem 1 *(ϵ-greedy policy iteration) For any ϵ -greedy policy π, the ϵ-greedy policy π' with respect to q_π is an improvement, $v_{\pi'}(s) \geq v_\pi(s)$*

$$\begin{aligned} Q\left(s, \pi'(s)\right) &= \sum_{a \in A} \pi'(a|s) Q_\pi(s,a) \\ &= \frac{\epsilon}{|A|} \sum_{a \in A} Q_\pi(s,a) + (1-\epsilon) \max_a Q_\pi(s,a) \\ &\geq \frac{\epsilon}{|A|} \sum_{a \in A} Q_\pi(s,a) \\ &+ (1-\epsilon) \sum_{a \in A} \frac{\pi(a|s) - \frac{\epsilon}{|A|}}{1-\epsilon} Q_\pi(s,a) \\ &= \sum_{a \in A} \pi(a|s) Q_\pi(s,a) \\ &= \sum_{a \in A} \pi(a|s) Q_\pi(s,a) \\ &= V_\pi(s) \end{aligned} \tag{4.2}$$

Where the inequality holds because the max operation is greater than or equal to an arbitrary weighted sum.

4.2.3 Dynamic Deep Neural Networks

The state-of-the-art performance of deep learning has largely come from increasing the size of neural networks. There are numerous works for efficient memory usage and inference using dynamic deep neural networks.

The dynamic kernels have been proposed for efficient memory usage [46, 47, 48]. In order to avoid applying the same convolutional kernels to every example in a dataset, the convolutional kernels changes based on the input. By transforming the inference path dynamically, the inference time can be efficiently reduced [49, 50, 51, 52, 53]. These approaches aim to use large networks for difficult examples and small networks or sub-network for easy examples. Our proposed method is related to how to dynamically transform the neural architecture for effectively introducing exploration effect into the gradient-based search.

4.3 Proposed Method

4.3.1 Preliminary: DARTS

I first briefly review DARTS, the basis for our proposed architecture search. Following prior works [34, 42, 54], DARTS aims to search for a cell as the building block of the final architecture. Based on the DAG with ordered sequence of N nodes [37], DARTS treats all possible cell architectures as subgraphs of a supergraph using weight sharing. The cell consists of two input nodes $x^{(0)}, x^{(1)}$ and intermediate nodes $x^{(2)}, \cdots, x^{(N-1)}$. The output of each intermediate node $x^{(j)}$, where $2 \leq j \leq N-1$, is computed by sum of its all predecessors $x^{(0)}, \cdots, x^{(j-1)}$ as

$$x^{(j)} = \sum_{i<j} o^{(i,j)}(x^{(i)}) \,, \tag{4.3}$$

where $o^{(i,j)}(\cdot)$ is a mixed operation by continuous relaxation. For each directed edge (i, j), there is a mixed operation $o^{(i,j)}(\cdot)$ as

$$
\begin{aligned}
o^{(i,j)}(x) &= \sum_{o \in \mathcal{O}} f(\alpha^{(i,j)})_o \cdot o(x) \\
&= \sum_{o \in O} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in \mathcal{O}} \exp\left(\alpha_{o'}^{(i,j)}\right)} \cdot o(x)\,,
\end{aligned}
\tag{4.4}
$$

where $\mathcal{O}$ is a set of all possible operations (e.g., convolution, identity connection, etc), and $\alpha_o^{(i,j)}$ is a hyperparameter for weighting operation $o(x)$ on the edge (i, j). The final output of the entire cell is formed by concatenating the output of all intermediate nodes $x^{(2)}, \cdots, x^{(N-1)}$. By relaxing the discrete search space to be continuous, DARTS could simultaneously optimize the architecture $\alpha = \{\alpha^{(i,j)}\}$ and the weights w within all the mixed operations. After the search is finished, operation $o(x)$ with the largest $\alpha_o^{(i,j)}$ value excluding the zero operation is selected on each edge (i, j). Then, two incoming edges with the largest $\alpha_o^{(i,j)}$ value are preserved for each node.

4.3.2 Exploration Strategy for Gradient-Based Search

DARTS alternately optimizes the architecture α and the weights w on the validation set and the training set, respectively. Let α_t and w_t denote the architecture and the weights at training step t, the gradient-based search is performed by alternating gradient descent with a learning rate ξ as

$$
\alpha_{t+1} = \alpha_t - \xi \nabla_{\alpha_t} \mathcal{L}_{val}\left(w_t, \alpha_t\right) \tag{4.5}
$$

$$
w_{t+1} = w_t - \xi \nabla_{w_t} \mathcal{L}_{train}\left(w_t, \alpha_{t+1}\right). \tag{4.6}
$$

Using a gradient descent, α is optimized in direction of reducing the validation loss, and w is updated in direction of reducing the training loss. There is an issue with this bilevel optimization problem. Due to the greedy nature of the gradient-based method and the nested formulation, the search process is inevitably biased, limiting the search for various directions. The optimization of α has a bias due to w at every step of the gradient descent, and vice versa. Thus, α and w affect each other's optimization, and the positions of α and w can greatly affect the direction of the architecture search.

Due to mutually biased optimization, the search proceeds without considering other different architectures. DARTS is a gradient-based search, but in terms of reinforcement learning, it can be viewed as using only exploitation to search the architecture without exploration. To tackle the bias issue of the gradient-based search, I introduce an exploration effect into the gradient-based search.

Let $Z^{(i,j)}$ be a vector where each element represents the value of choosing a particular operation on the edge (i, j). DARTS computes $Z^{(i,j)}$ as a softmax of the architecture parameter $\alpha^{(i,j)}$ as in equation (4.4). In our proposed method, I compute $Z_o^{(i,j)}$ as

$$Z_o^{(i,j)} = \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in O} \exp\left(\alpha_{o'}^{(i,j)}\right)} + \sigma \cdot q_o^{(i,j)}, \tag{4.7}$$

where σ is a weight of how much the exploration effect will be activated during the search, and $q_o^{(i,j)}$ is a random probability independent of the architecture parameter $\alpha^{(i,j)}$. Using an auxiliary noise variable ϵ that is randomly sampled from the univariate Gaussian distribution of mean 0 and variance 1, I compute a random probability $q_o^{(i,j)}$ as

$$q_o^{(i,j)} = \frac{\exp\left(\epsilon_o\right)}{\sum_{o' \in O} \exp\left(\epsilon_{o'}\right)}, \tag{4.8}$$

where $\epsilon_o \sim \mathcal{N}(0, 1)$.

I start the architecture search with $\sigma = 1$, which decreases exponentially with each training epoch, reaching nearly zero at the end. At the beginning of the search, I can estimate less biased $Z^{(i,j)}$ due to the exploration by $q^{(i,j)}$, and weight w is optimized with various cases of the architecture where each edge of the cell has a distribution of $Z^{(i,j)}$. As the search proceeds, the exploration weight σ exponentially decreases, the exploration effect by $q^{(i,j)}$ gradually diminishes. At the end, the exploitation by optimization of $\alpha^{(i,j)}$ becomes dominant. With this simple technique, the architecture gradually changes from the dynamic architecture to the static architecture during the search. After the search, I select a operation $o(x)$ with the largest $Z_o^{(i,j)}$ value excluding

the zero operation on each edge (i, j). Then, two incoming edges with the largest $Z_o^{(i,j)}$ value are preserved for each node, same as DARTS.

4.3.3 Dynamic Attention Networks

Using a random probability vector $q^{(i,j)}$ of equation (4.8) is not an efficient strategy for the exploration effect. For efficient exploration, I propose Dynamic Attention Networks (DANs) that dynamically change the architecture based on the input, rather than randomly changing the architecture. Based on equation (4.7), I replace a random probability $q_o^{(i,j)}$ to compute $Z_o^{(i,j)}$ as

$$Z_o^{(i,j)} = \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in O} \exp\left(\alpha_{o'}^{(i,j)}\right)} + \sigma \cdot F(x^{(i)})_o, \tag{4.9}$$

where $x^{(i)}$ is an input on the edge (i, j), and $F(x^{(i)})_o$ is the output of DAN. DAN computes the attention weights over all possible operations on the edge (i, j). The attention weights make the search to consider more potential architectures than random ones. As shown in Figure 4.1, I uses global average pooling first to squeeze the global information of the edge input $x^{(i)}$. Then, two fully connected layers are followed with a ReLU activation function between them. To generate a normalized attention, final layer is a softmax as

$$F(x^{(i)})_o = \frac{\exp\left(f(x^{(i)})_o\right)}{\sum_{o' \in O} \exp\left(f(x^{(i)})_{o'}\right)}, \tag{4.10}$$

where $f(x^{(i)})$ is the logit of DAN. As in previous Section 4.3.2, I use the attention weight σ to control the exploration effect of DAN during the search. Our proposed method does not require much memory usage. DAN generates a $|\mathcal{O}|$-dimensional vector from the edge input with dimension $H \times W \times C$. The edge input is squeezed to a vector with dimension C by a global average pooling. Then, two fully connected layers in our DAN changes a vector size in order of $C \to C/4 \to |\mathcal{O}|$. I can compute the number of parameters in DAN as $C \times C/4 + C/4 \times |\mathcal{O}|$, and the sum of these param-

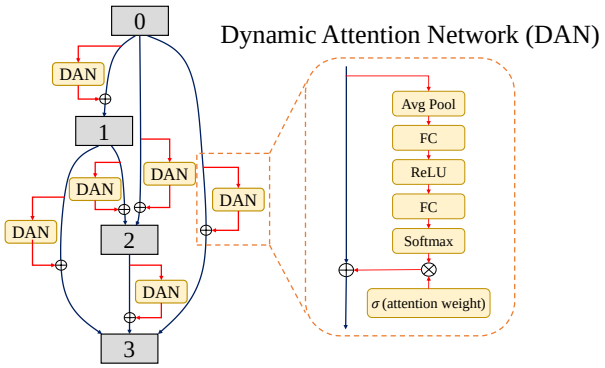


Figure 4.1: Illustration of our proposed method, DE-DARTS. Dynamic Attention Network (DAN) is implemented on each edge of DAG. DAN consists of a global average pooling, two fully connected layers with a ReLU activation between them, and a softmax. I multiply the output of DAN by the attention weight σ, then add it to the architecture parameter α.

eters in all DANs is significantly smaller compared to the sum of the other parameters $\{w, \alpha\}$. Thus, our proposed method, DE-DARTS, is applicable to any gradient-based methods by just implementing DAN on each edge of cells, and has a negligible effect on the search time.

The method of adding noise in equation (4.7) is similar to the gumbel softmax proposed by SNAS[38] et al. However, more efficient architecture search is possible by adding learnable variable to the architecture parameter α through the DAN I use. In addition, the equation (4.7) proposed by us differs from the gumbel softmax in that each noise has different dynamic characteristics for each input in the batch. In the case of gumbel softmax, the noise value added to the α value is the same for all inputs in the batch. The performance comparison of the noise and the DAN is described in Section 4.5.2.

4.4 Experiments

4.4.1 Datasets

I conduct experiments with three image classification datasets including CIFAR-10, CIFAR-100 , and ImageNet. Each of CIFAR-10 and CIFAR-100 [55] contains 50K/10K training/testing RGB images with a fixed size of 32×32. Images in CIFAR-10 are equally distributed over 10 classes, with 6K images per each class. Images in CIFAR-100 are equally distributed over 100 classes, with 600 images per each class. ImageNet [56] contains 1.3M/50K training/validation RGB images with a high resolution. Images in ImageNet are roughly equally distributed over 1,000 object categories. Following the conventions [34, 5], I adopt the mobile setting where the input image size is 224×224 and the number of multi-add operations is restricted to be less than 600M. I conduct the architecture search and evaluation on CIFAR-10, and test the transferability of the architecture on CIFAR-100 and ImageNet.

4.4.2 Architecture Search

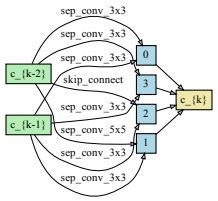
Implementation details

Following DARTS, I set eight candidate operations on each edge: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated convolutions, 3×3 max pooling, 3×3 average pooling, identity, and zero. The architecture is constructed by stacking eight cells (6 normal cells and 2 reduction cells) of 16 initial channels, and two reduction cells are located at the $1/3$ and $2/3$ of the total depth of the network. And each cell consists of $N = 7$ nodes (2 input nodes, 4 intermediate nodes, and 1 output node). For each cell, the two input nodes are from the output of the two previous cells. The output of the cell is computed by concatenating the output of all intermediate nodes. A small network with 8 cells is trained for 50 epochs. For the optimization of weights w, I use a momentum SGD optimizer, with an initial learning rate of 0.025 (annealed down to zero following a cosine schedule without restart [57]), a momentum of 0.9

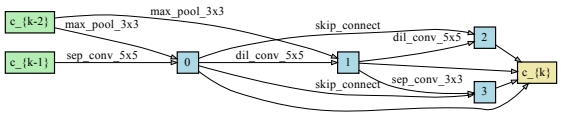
and a weight decay of 3×10^{-4}. And for the optimization of architecture α, I use an Adam [24] optimizer, with an initial learning rate of 3×10^{-4}, a momentum of $(0.5, 0.999)$, and a weight decay of 10^{-3}. As our proposed method can be simply applied to any search method based on DAG, I adopt the second order optimization scheme of DARTS to learn the architecture parameters for high performance.

Dynamic architecture search

Our Dynamic Attention Network (DAN) used for dynamic architecture search consists of a global average pooling, two fully connected layers with a ReLU between them, and a softmax as shown in Figure 4.1. When weights w are optimized, I optimize the parameters in DAN together using a same momentum SGD optimizer. As the number of parameters in DANs is small and parameters are optimized with weights w together, DANs has almost negligible effects on the search time. Thus, the search time is almost same with DARTS (24 hours on a single GPU). The attention weight σ gradually decreases with each epoch, with an initial value of $\sigma = 1$, a decaying rate of $\gamma = 0.9$.



(a) Normal cell.



(b) Reduction cell.

Figure 4.2: Selected cells by DE-DARTS (2nd order) on CIFAR-10.

Search results

After the search, the cell selection is based on the architecture parameter α. After selecting a operation for 14 edges of the cell, two incoming edges per each node are preserved with a largest value α. As in Figure 4.2 (a), the normal cell I found consists of 7 convolutions, and 1 skip connection. And as in Figure 4.2 (b), the reduction cell I found consists of 4 convolutions, 2 max poolings, and 2 skip connections. I provide evaluations of the architecture I found on the next section.

4.4.3 Architecture Evaluation

After the search on CIFAR-10, I evaluate the cell found by DE-DARTS. In addition, I performed transferability test on CIFAR-100 and ImageNet using the cell searched on CIFAR-10. All experiments were conducted using RTX-2080TI GPUs.

Evaluation on CIFAR-10

To evaluate the selected architecture on CIFAR-10, I stack 20 cells of 36 initial channels to make a whole network: 18 normal cells and 2 reduction cells. The reduction cells are located at the $1/3$ and $2/3$ of the total depth of the network. Then, the whole network is trained from scratch for 600 epochs with batch size 80. I use a momentum SGD optimizer, with an initial learning rate of 0.025, a momentum of 0.9, and a weight decay of 3×10^{-4}. Following the existing works [5, 38, 37, 34, 42, 41, 44], I adopt cutout [1], path dropout [58] of probability 0.2, and auxiliary towers [59] with weight 0.4. I repeat training an evaluation network from scratch 3 times, then report the mean and standard deviation of 3 independent runs. Compared to the existing works using the gradient-based search [5, 38, 41, 44, 45], the architecture I searched on CIFAR-10 achieves state-of-the-art performance with a test error of 2.46%. Also, compared to the algorithms using reinforcement learning or evolutionary algorithm which are NASNet [34] and AmoebaNet [42], our algorithm achieves a lower test error. Proxylessnas [39] is the lowest with an accuracy of 2.08, but the search space is different from the pro-

posed paper. The number of parameters in the cell I found is 4.1, which has a relatively larger number of parameters than other DARTS-based papers[5, 38, 41, 44, 45], but shares the same search space with other DARTS-based papers. In other words, our algorithm performs better than other algorithm in the same search space.

Table 4.1: Comparison with state-of-the-art architectures on CIFAR-10. All architectures are constructed by stacking 20 cells of 36 initial channels and trained using cutout [1] enhancement.

Architecture	**Test Error (%)**	**Params (M)**	**Search Cost (GPU days)**	**Search Method**
DenseNet-BC [60]	3.46	25.6	-	manual
NASNet-A + cutout [34]	2.65	3.3	2000	RL
AmoebaNet-A + cutout [42]	3.34 ± 0.06	3.2	3150	evolution
AmoebaNet-B + cutout [42]	2.55 ± 0.05	2.8	3150	evolution
PNAS [54]	3.41 ± 0.09	3.2	225	SMBO
ENAS + cutout [37]	2.89	4.6	0.5	RL
DARTS + cutout [5]	2.76 ± 0.09	3.3	1	gradient-based
SNAS + cutout [38]	2.85 ± 0.02	2.8	1.5	gradient-based
ProxylessNAS + cutout [39]	2.08	5.7	4	gradient-based
P-DARTS + cutout [41]	2.50	3.4	0.3	gradient-based
DATA ($M = 7$) + cutout [44]	2.59	3.4	1	gradient-based
PC-DARTS + cutout [45]	2.57 ± 0.07	3.6	0.1	gradient-based
DE-DARTS + cutout	2.46 ± 0.06	4.1	1	gradient-based

Transferability evaluation on CIFAR-100

I use CIFAR-100 to test the transferability of the architecture searched on CIFAR-10. Experimental settings such as the number of layers and hyperparameters are the

same as those of CIFAR-10. The results are presented in Table 4.2. It shows that DE-DARTS transferability results on CIFAR-100 are highly competitive with a test error of 15.82%.

Table 4.2: Comparison with state-of-the-art architectures on CIFAR-100. † indicates that architecture was searched on CIFAR-100, other wise it was searched on CIFAR-10.

Architecture	**Test Error (%)**	**Params (M)**	**Search Cost (GPU days)**	**Search Method**
DenseNet-BC [60]	17.18	25.6	-	manual
DARTS + cutout (2nd order) [5]	17.54	3.3	1	gradient-based
P-DARTS + cutout (CIFAR-10) [41]	16.55	3.4	0.3	gradient-based
P-DARTS + cutout (CIFAR-100)† [41]	15.92	3.6	0.3	gradient-based
DE-DARTS + cutout	15.82 ± 0.24	4.1	1	gradient-based

Transferability evaluation on ImageNet

In addition to CIFAR-100, I also test the transferability of DE-DARTS on ImageNet. I train a whole network from scratch for 250 epochs with the batch size 128. I use a momentum SGD optimizer with an initial learning rate of 0.1, a momentum of 0.9, and a weight decay of 3×10^{-5}, and use auxiliary towers with weight 0.4 for an additional enhancement. I use learning rate warm up [63] for the first 5 epochs and use label smoothing [64]. Table 4.3 shows that DE-DARTS transferability results on ImageNet are highly competitive with a test error of 24.0%.

Table 4.3: Comparison with state-of-the-art architectures on ImageNet. * indicates that architecture was searched on ImageNet, otherwise it was searched on CIFAR-10.

Architecture	**Test Error (%)**		**Params**	**Search Cost**	**Search Method**
	top-1	top-5	**(M)**	**(GPU days)**	
Inception-v1 [59]	30.2	10.1	6.6	-	manual
MobileNet [61]	29.4	10.5	4.2	-	manual
Shuffle-Net-v2 [62]	25.1	-	5	-	manual
NASNet-A[34]	26.0	8.4	5.3	2000	RL
NASNet-B[34]	27.2	8.7	5.3	2000	RL
NASNet-C[34]	27.5	9.0	4.9	2000	RL
AmoebaNet-A[42]	25.5	8.0	5.1	3150	evolution
AmoebaNet-B[42]	26.0	8.5	5.3	3150	evolution
AmoebaNet- + cutout [42]	24.3	7.6	6.4	3150	evolution
DARTS [5]	26.7	8.7	4.7	4	gradient-based
SNAS [38]	27.3	9.2	4.3	1.5	gradient-based
ProxylessNAS (GPU)* [39]	24.9	7.5	7.1	8.3	gradient-based
PC-DARTS [45]	25.1	7.8	5.3	0.1	gradient-based
P-DARTS [41]	24.4	7.4	4.9	0.3	gradient-based
DATA (M=7) [44]	24.9	8.0	5.0	1	gradient-based
DE-DARTS	24.0	7.2	5.7	1	gradient-based

4.5 Analysis and Ablation Study

4.5.1 Operation Distribution of DARTS and DE-DARTS

I run DARTS and DE-DARTS 5 times with different random seeds and observed the distribution of operations. The number of operations for each search result (S1 $\sim$ S5) is in Table 4.4 and Table 4.5. In Table 4.4 and Table 4.5, I show the total number of

7 operations excluding zero operation. For each search result, there are a total of 16 operations, 8 operations for normal cell and 8 operations for reduction cell respectively. There are many skip connections in each search result by DARTS. On the other hand, DE-DARTS had a relatively higher number of sep-convs and fewer skip-connect than DARTS. Cells searched by DE-DARTS have a lot of convolution operations, so their expressive power is better than cells found with DARTS.

Table 4.4: DARTS

Operation	S1	S2	S3	S4	S5	Mean
max-pool 3x3	3	0	1	1	0	1
avg-pool 3x3	0	4	3	2	4	2.6
skip-connect	7	5	6	8	5	6.2
sep-conv 3x3	3	5	4	3	3	3.6
sep-conv 5x5	1	1	0	0	0	0.4
dil-conv 3x3	2	0	1	1	3	1.4
dil-conv 5x5	0	1	1	1	1	0.8

Table 4.5: DE-DARTS

Operation	S1	S2	S3	S4	S5	Mean
max-pool 3x3	1	5	6	5	2	3.8
avg-pool 3x3	0	0	0	0	0	0
skip-connect	3	3	2	2	3	2.6
sep-conv 3x3	5	4	4	4	7	4.8
sep-conv 5x5	5	2	3	4	2	3.2
dil-conv 3x3	0	0	0	0	0	0
dil-conv 5x5	2	2	1	1	2	1.6

4.5.2 Comparison of Ramdom Noise and DAN

In this section, I discuss the effectiveness of exploration by two different functions of equation (4.7) and equation (4.9). In addition to the evaluation results of the 2nd order DE-DARTS with DAN in CIFAR-10 result, I evaluate the 2nd order DE-DARTS with a random probability vector $q_o^{(i,j)}$ that randomly changes the architecture. To compute a random probability vector $q_o^{(i,j)}$, auxiliary random variable ϵ is first sampled from the univariate Gaussian distribution of zero mean and unit variance. The, random probability $q_o^{(i,j)}$ is computed as a softmax of random variable ϵ.

In Table 4.6, (noise batch static) means the input noise in all batches the same as the gumbel softmax, whereas (noise batch dynamic) means the input noise in each batch different. As a result of the experiment, it was better to make the noise different

for each input in the batch than to make it the same. Also, it turned out that it was better to use a learnable DAN than to use noise.

Table 4.6: Comparison of exploration effects of DAN and noise. Noise is Gaussian distribution of zero mean and unit variance. The experiment was carried out on CIFAR-10.

Architecture	**Test Error (%)**	**Params (M)**
DE-DARTS (noise batch static)	2.73 ± 0.07	3.4
DE-DARTS (noise batch dynamic)	2.67 ± 0.06	4.3
DE-DARTS (DAN)	2.46 ± 0.06	4.1

4.5.3 Decaying Rate γ

We conducted experiments for the effect of decaying rate γ of the attention weight. The search progresses of different decaying rates are shown in the Fig. 4.3. We can see that the smaller decaying rate γ, the higher validation accuracy early in the search. In other words, a lower decaying rate means reducing exploration and increasing exploitation more rapidly so that validation accuracy achieves higher early in the search. The validation accuracy with a decaying rate of $\gamma = 0.5$ increases faster than a decaying rate of $\gamma = 0.9$, but the final validation accuracy is similar, but validation accuracy at the end of the search reaches almost the same value. Except the very early stage of the search, we can see that progresses of different decaying rates are very similar. So, it indicates that DE-DARTS is robust to the value of a decaying rate γ.

Table 4.7: Correlation of operators ranking up to the first 10 epoch. I calculate the correlation of operators between 0-5 epoch and 5-10 epoch.

Edge	DARTS	DE-DARTS
E1	0.77	0.37
E2	0.98	0.05
E3	0.57	-0.6
E4	0.6	-0.84
E5	0.98	-0.08
E6	-0.19	-0.72
E7	0.19	-0.18
E8	1	0.03
E9	0.98	-0.04
E10	0.36	0.75
E11	0.12	0.32
E12	0.5	0.51
E13	1	-0.05
E14	0.9	-0.21

4.5.4 DE-DARTS Learning Curve on CIFAR-10

Fig. 4.4 shows the test accuracy on CIFAR-10 which is evaluated during the training of the architectures found by 2nd order DARTS and 2nd order DE-DARTS. At 600 epochs, DARTS achieves a test accuracy of 97.30% and DE-DARTS achieves a test accuracy of 97.62%. At the beginning of the training procedure, there is no significant difference between DARTS and DE-DARTS test accuracy. After about 200 epochs, the test accuracy of DE-DARTS is gradually higher than that of DARTS.

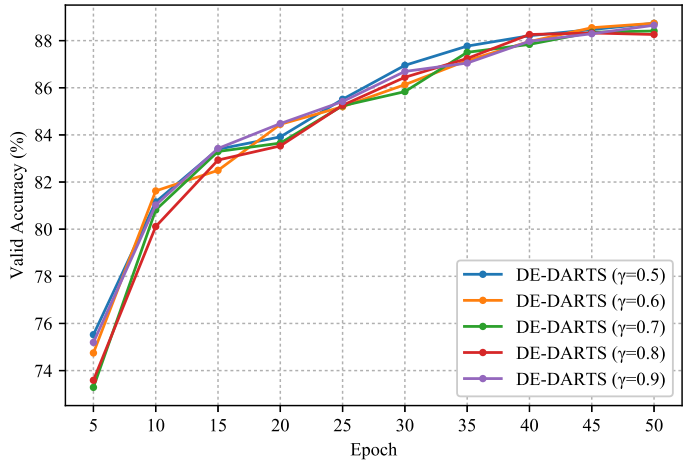


Figure 4.3: Search progresses of DE-DARTS with different decaying rates γ. We measured validation accuracy every 5 epochs with 5 different decaying rates from $\gamma = 0.5$ to $\gamma = 0.9$. The architecture search was conducted for 50 epochs on CIFAR-10.

4.5.5 Exploration Effect in Architecture Search Process

Due to the greedy nature of gradient update and bi-level optimization, the architecture search process of existing DARTS methods is biased. In other words, the search range of the architecture search is too narrow, and the search continues only within similar architectures. This trend is particularly noticeable early in the architecture search. I obtained the ranks of operators for each edge and found how the ranking of operators changes according to epochs through Spearman rank correlation.

I show the correlation between operators of DARTS and DE-DARTS in Table 4.7. I calculated the average of the rankings of operators from 0 to 5 epoch and from 5 to 10 epoch, respectively, and then calculated the correlation between the two. Many edges in DARTS are mostly positive with high correlation, whereas DE-DARTS has a diverse correlation range with a low positive correlation or negative correlation. In other words, DARTS has a similar operator ranking between 0-5 epochs and 5-10 epochs. Therefore the architecture search range of DARTS is narrow at the beginning.

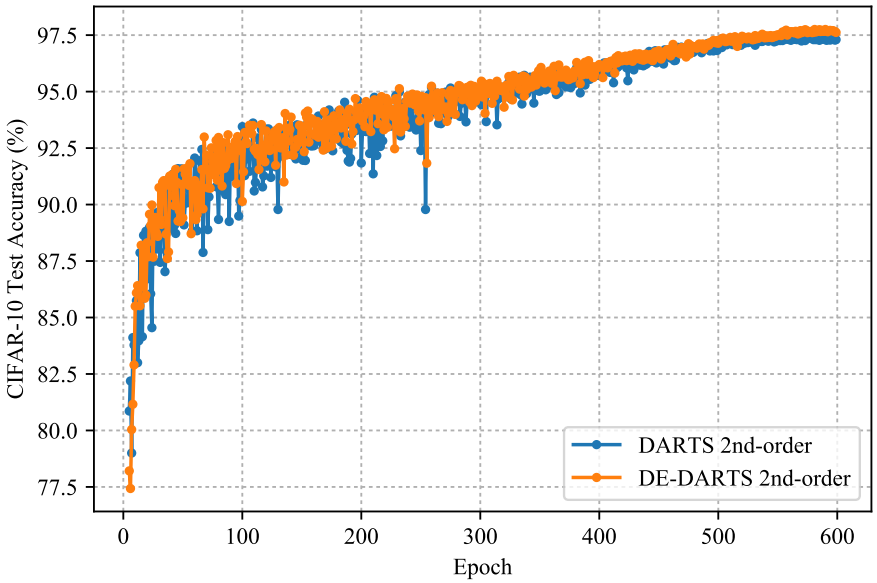


Figure 4.4: Test accuracy on CIFAR-10 during the training procedure. Both DARTS and DE-DARTS were trained for a total of 600 epochs. For a clear comparison of the learning curve, the results from epoch 5 to 600 epoch are plotted on the graph.

However, DE-DARTS has a large change in operator ranking, which indicates a wider range of architecture search is possible.

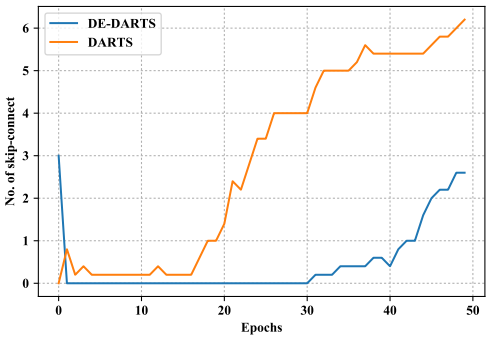


Figure 4.5: This plot shows the number of skip connections of the DARTS and our DE-DARTS as the epoch changes. I averaged a total of 5 runs.

4.5.6 Number of Skip-Connections

In this section, I analyze the skip-connect operation issue, which was also addressed by previous research. It is an issue is that the number of skip-connect operations is too large in the selected architecture by DARTS. As architecture parameters of gradient-based architecture search are updated in the direction that accelerates convergence, the gradient descent begins to prefer skip-connect operation and it becomes more and more dominant. Excessive skip connections do not have parameters compared to other operators, so their expressive power is relatively inferior to convolution operations. As a result, when there are many skip connections, the final accuracy is lower than that of fewer skip connections. As our proposed method aims to reduce the bias in the search process, the number of skip-connect operations is expected to be small in an architecture selected by our proposed method.

Figure 4.5 shows the number of skip connections in search processes as the epoch changes. I select two edges each in the order of the highest architecture parameter based on the input edge coming to the graph node of DARTS from all edges. For the final number of edges, eight edges are selected from the normal cell and the reduce cell. I searched 5 times for both DARTS and DE-DARTS. The number of skip connections in Figure 4.5 is the average sum of the number of skip connections in the normal cell and the reduce cell respectively. Both the DARTS and the DE-DARTS tend to increase the number of skip connections as the epoch increases. In our DE-DARTS, the rate of increase in the number of skip connections was lower than that of the DARTS. Finally, the number of skip connections for DE-DARTS was 2.75 on average compared to the DARTS with an average number of skip connections of 6.5. Compared to DARTS, DE-DARTS has a relatively low number of skip connections, which implies greater expressive power.

Table 4.8: DARTS

Operation	S1	S2	S3	S4	S5	Mean
max-pool 3x3	3	0	1	1	0	1
avg-pool 3x3	0	4	3	2	4	2.6
skip-connect	7	5	6	8	5	6.2
sep-conv 3x3	3	5	4	3	3	3.6
sep-conv 5x5	1	1	0	0	0	0.4
dil-conv 3x3	2	0	1	1	3	1.4
dil-conv 5x5	0	1	1	1	1	0.8

Table 4.9: P-DARTS

Operation	S1	S2	S3	S4	S5	Mean
max-pool 3x3	0	1	1	1	0	0.6
avg-pool 3x3	1	2	4	3	4	2.8
skip-connect	7	6	4	4	4	5
sep-conv 3x3	3	4	3	5	4	3.8
sep-conv 5x5	2	1	1	0	2	1.2
dil-conv 3x3	2	2	3	1	0	1.6
dil-conv 5x5	1	0	0	2	2	1

Table 4.10: PC-DARTS

Operation	S1	S2	S3	S4	S5	Mean
max-pool 3x3	4	2	2	2	2	2.4
avg-pool 3x3	0	1	0	1	1	0.6
skip-connect	1	4	5	2	4	3.2
sep-conv 3x3	7	3	2	6	2	4
sep-conv 5x5	3	2	3	3	4	3
dil-conv 3x3	1	3	3	0	2	1.8
dil-conv 5x5	0	1	1	2	1	1

Table 4.11: DE-DARTS

Operation	S1	S2	S3	S4	S5	Mean
max-pool 3x3	1	5	6	5	2	3.8
avg-pool 3x3	0	0	0	0	0	0
skip-connect	3	3	2	2	3	2.6
sep-conv 3x3	5	4	4	4	7	4.8
sep-conv 5x5	5	2	3	4	2	3.2
dil-conv 3x3	0	0	0	0	0	0
dil-conv 5x5	2	2	1	1	2	1.6

4.5.7 Comparison of Operation Distribution with Other Papers

I added P-DARTS and PC-DARTS to the operation distribution experiments mentioned in section 5.1. I used publicly available code for both P-DARTS and PC-DARTS. In the P-DARTS experiment, there is a part to manually adjust the skip connection in second regularization, so the results of applying operation-level dropout to the first regularization were displayed. As a result of the experiment in Table 4.8, 4.9, 4.10, 4.11, it was found that skip-connection was small in the order of DARTS, P-DARTS, PC-DARTS and DE-DARTS. The operation distribution of PC-DARTS and DE-DARTS with relatively few skip-connections was found to be quite similar. For example, the

number of max-pools is more than the number of avg-pools. All four algorithms have in common that the number of sep-convs is larger than the number of dil-convs.

4.5.8 Changes of Architecture Distribution

To see the changes in the cell architecture during the search process, I visualize the distribution of all edges over seven operations sampled at the end of the epoch. The zero operation is excluded because the operation with the largest α value is selected among seven operations excluding the zero operation. Fig. 4.6 and Fig. 4.7 represent the changes in the distribution of normal and reduction cells during the search process of DARTS. Fig. 4.8 and Fig. 4.9 represent the changes in the distribution of normal and reduction cells during the search process of DARTS. Normal and reduction cells each have 14 edges. The y-axis represents epochs, and the x-axis represents operations with 1 through 7 representing 3×3 max pooling, 3×3 average pooling, identity, 3×3 separable convolution, 5×5 separable convolution, 3×3 dilated convolution, and 5×5 dilated convolution, respectively.

Comparing DARTS and DE-DARTS, DARTS has a similar distribution in most of the epochs, whereas DE-DARTS has a diverse distribution for each epoch. And, especially in the early part of the search, you can see that DE-DARTS considers more diverse cell architectures than DARTS.

4.6 Conclusion

I viewed the gradient-based search in terms of reinforcement learning and proposed DE-DARTS using Dynamic Attention Networks. By adding DAN on the edge, input-dependent dynamic architecture is constructed during the search, and makes exploration effect for the gradient-based search. I experimentally showed that DE-DARTS has fewer skip-connects on average than DARTS. I found the normal cell and the reduction cell that achieve lower test error than other state-of-the-art algorithms on

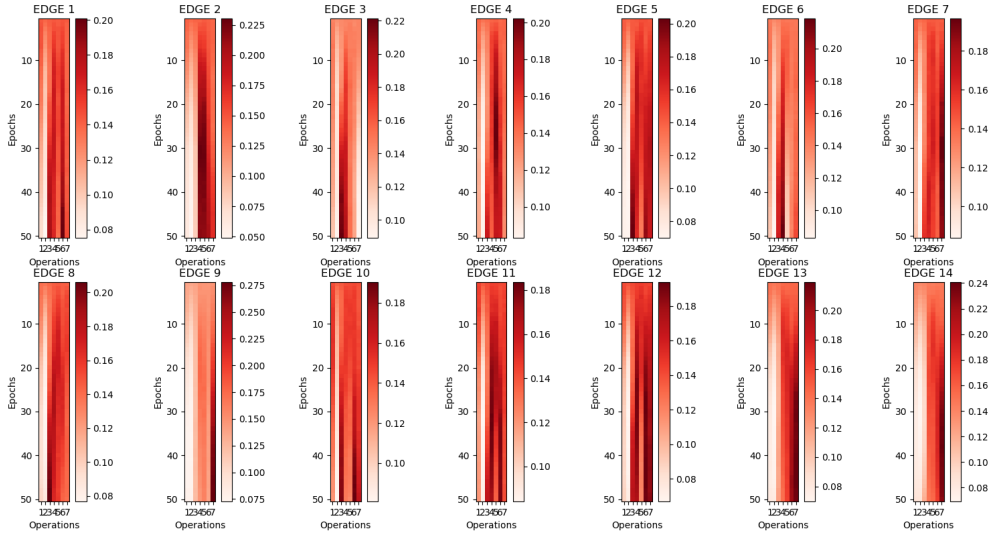


Figure 4.6: Visualization of the changes in the normal cell distribution during the search process (DARTS).

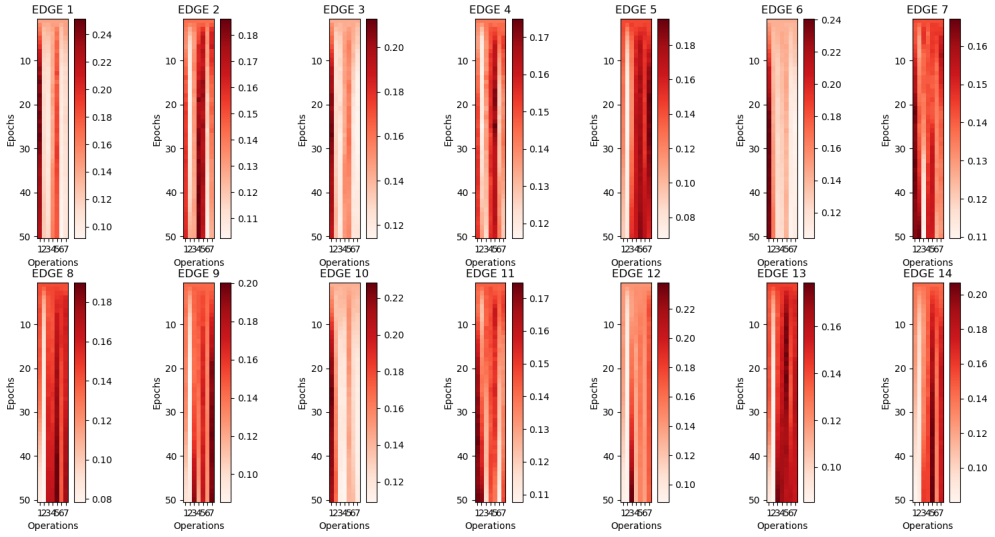


Figure 4.7: Visualization of the changes in the reduction cell distribution during the search process (DARTS).

CIFAR-10, CIFAR-100, and ImageNet. In addition, I believe that our proposed method is easily applicable to any other gradient-based search as well as DARTS algorithm.

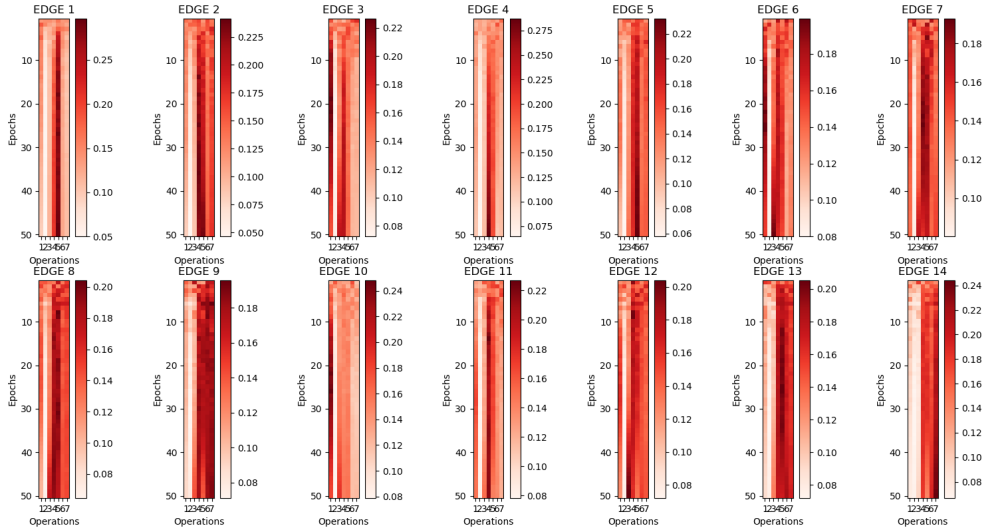


Figure 4.8: Visualization of the changes in the normal cell distribution during the search process (DE-DARTS).

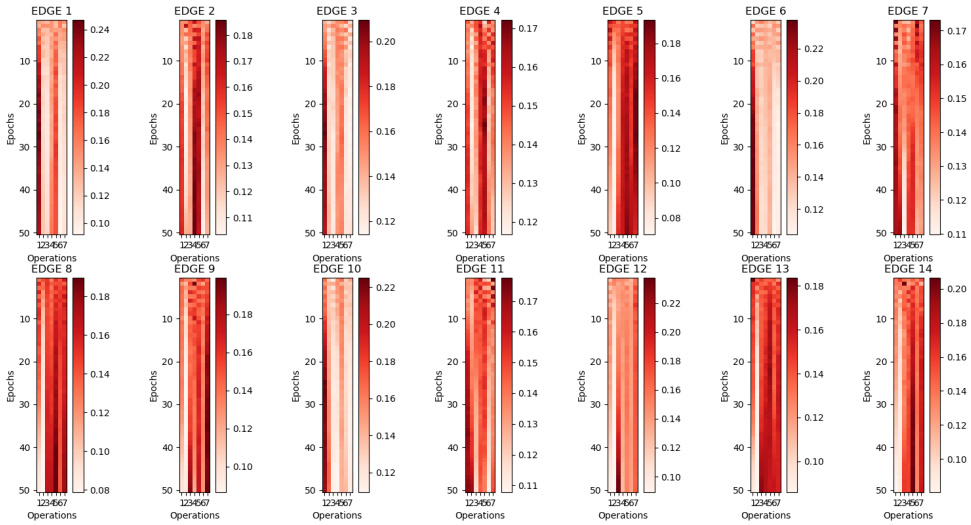


Figure 4.9: Visualization of the changes in the reduction cell distribution during the search process (DE-DARTS).

Chapter 5

Radar Interference Mitigation Using Neural Architecture Search Model

5.1 Introduction

When designing the existing radar interference cancellation model, the RNN-based model was initially conceived, and the attention technique was later introduced. Also, it took a lot of time to adjust the number of layers or hyper parameters of the RNN even after designing the model. Rather than designing a model by humans, I wondered how AI would design a model, so I started to study neural architecture search later. Neural architecture search makes AI design a model, and given data, it provides optimal model results. In this chapter, I apply the neural architecture search-based model we did in chapter 4 to the radar interference removal.

5.2 Model

The basic model was constructed based on the recurrent experiment of the DARTS paper. The word embedding part was removed from the model used in the existing DARTS paper, and the average pooling layer was applied during the last decoding to make the final output a one-dimensional signal.

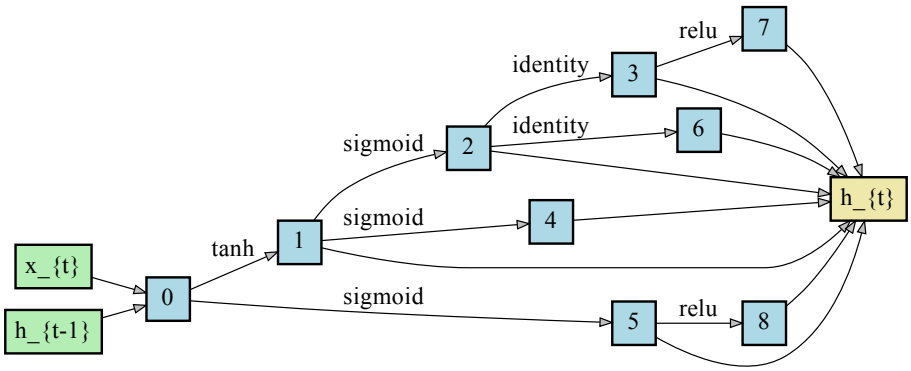


Figure 5.1: recurrent cell

For hyperparameters, the hidden layer size was 300, the batch size was 64, and dropout was not applied. The optimizer used sgd, the initial learning rate was 0.1, and the learning rate decay used cosine decay. The epoch used for training was 100 epochs, and the total number of data was 130000 train data, 10000 valid data, and 10000 test data.

As a result of training the DARTS-based gradient descent model, the result shown in Fig. 5.1 was obtained. Tanh is one, sigmoid is three, identity is two, and relu is two.

5.3 Experiment

The experiment result is shown in Fig. 5.2, Fig. 5.3, Fig. 5.4. Fig. 5.2 is input signal with interference, Fig. 5.3 is output signal using deep learning, Fig. 5.4 is the orignal label without interference. If you look at the output result, you can see that there was a large interference in the middle of the input signal, but it was removed and the original signal was restored well. As a result of this experiment, it has been shown that radar interference is well removed even with a neural architecture search-based model.

Next, we will compare the found cell with other algorithms. The table is in 1. As a result of comparing the tables, when comparing the cells, the performance was

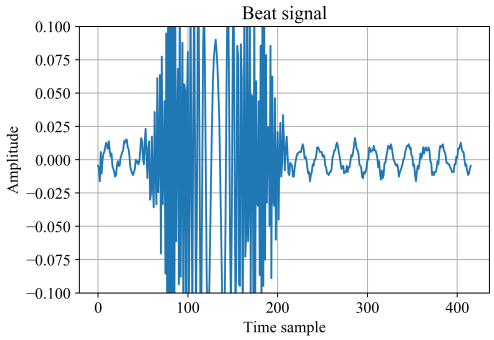


Figure 5.2: input signal

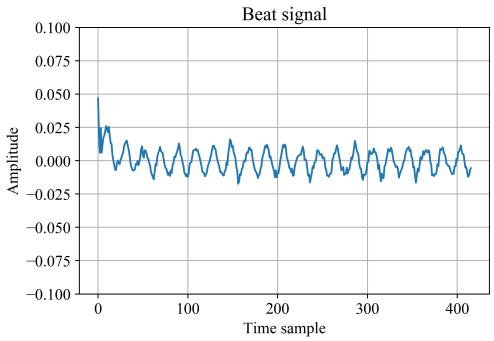


Figure 5.3: deep learning output signal

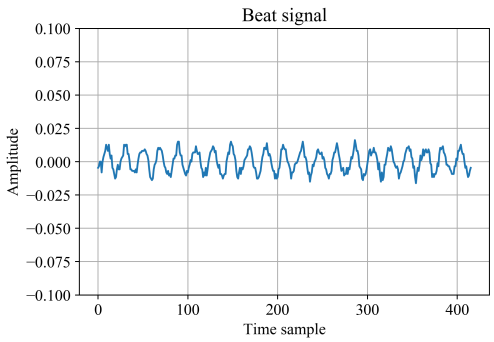


Figure 5.4: label

good in the order of rnn, gru, and seach cell. The performance slightly improved when attention was applied, and the performance when using search cell + attention was

Table 5.1: Comparison of algorithm

cell	val loss(RMSE)
RNN	0.008311
GRU	0.007012
search cell	0.006918
GRU + attention [65]	0.006627
search cell + attention	0.006583

slightly better than when using GRU + attention.

Chapter 6

Conclusion

6.1 Summary

In this dissertation, we discuss radar interference cancellation and new AutoML technology using deep learning.

In chapter 2, how to remove radar interference using deep learning is covered. The most commonly used signal in radar is FMCW. FMCW predicts the distance and angle of the target using the difference between the transmitted signal and the received signal. Existing signal processing methods have limitations in predicting the pattern of this error, but I used deep learning to achieve higher performance.

In chapter 3, the self-attention technique used in AI was added to the deep learning model that removes radar interference to enable more detailed processing, and the interference cancellation technology was applied to OFDM signals as well as FMCW.

In chapter 4, creating an existing deep learning model requires a lot of human-power. To solve this problem, AutoML, in which AI designs models, was introduced. In the early days of AutoML, the model was designed using reinforcement learning, but it takes a long time to learn because the model has to be run from scratch. The weight sharing technique that came out to solve this problem is ENAS and DARTS. In this dissertation, to solve the problem of the existing DARTS I introduce DE-DARTS.

In DE-DARTS, to solve the problem of the increase in the number of skip connections, a dynamic attention network was added at the beginning of learning to give an exploration effect. As a result, the number of skip connections was reduced, and DE-DARTS showed higher performance than existing papers in CIFAR-10, CIFAR-100, Imagnet.

In chapter5, I demonstrate that radar interference is removed with a neural architecture search-based model. As a result, the interference was well removed and the original signal was similarly restored. In addition, when compared with the existing rnn cell and gru cell, performance improvement was achieved with a larger gap than rnn cell and a slightly smaller gap with gru cell.

6.2 Future Direction

In removing radar interference through deep learning, it would be better to experiment directly with real data rather than simulation data. In addition, I plan to update the model through AutoML, which we did in Chapter 4, rather than directly implementing the deep learning model. In DE-DARTS, I plan to experiment with RNN as well as CNN later. Also, I plan to improve the algorithm for further performance improvement later.

Bibliography

[1] Terrance DeVries and Graham W Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[2] Christian Waldschmidt, Juergen Hasch, and Wolfgang Menzel, "Automotive radar—from first efforts to future systems," *IEEE Journal of Microwaves*, vol. 1, no. 1, pp. 135–148, 2021.

[3] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[4] Barret Zoph and Quoc V Le, "Neural architecture search with reinforcement learning," *International Conference on Learning Representations (ICLR)*, 2017.

[5] Hanxiao Liu, Karen Simonyan, and Yiming Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[6] Matthias Kronauge and Hermann Rohling, "New chirp sequence radar waveform," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 4, pp. 2870–2877, 2014.

[7] Volker Winkler, "Range doppler detection for automotive fmcw radars," in *Microwave Conference, 2007. European*. IEEE, 2007, pp. 1445–1448.

[8] Andrew G Stove, "Linear fmcw radar techniques," in *IEE Proceedings F (Radar and Signal Processing)*. IET, 1992, vol. 139, pp. 343–350.

[9] Graham M Brooker, "Mutual interference of millimeter-wave radar systems," *IEEE Transactions on Electromagnetic Compatibility*, vol. 49, no. 1, pp. 170–181, 2007.

[10] Markus Goppelt, H-L Blöcher, and Wolfgang Menzel, "Analytical investigation of mutual interference between automotive fmcw radar sensors," in *Microwave Conference (GeMIC), 2011 German*. IEEE, 2011, pp. 1–4.

[11] Yuu Watanabe and Kazuma Natsume, "Interference determination method and fmcw radar using the same," Mar. 6 2007, US Patent 7,187,321.

[12] Martin Kunert, "The eu project mosarim: A general overview of project objectives and conducted work," in *Radar Conference (EuRAD), 2012 9th European*. IEEE, 2012, pp. 1–5.

[13] Jonathan Bechter and Christian Waldschmidt, "Automotive radar interference mitigation by reconstruction and cancellation of interference component," in *Microwaves for Intelligent Mobility (ICMIM), 2015 IEEE MTT-S International Conference on*. IEEE, 2015, pp. 1–4.

[14] Michael Barjenbruch, Dominik Kellner, Klaus Dietmayer, Jens Klappstein, and Juergen Dickmann, "A method for interference cancellation in automotive radar," in *Microwaves for Intelligent Mobility (ICMIM), 2015 IEEE MTT-S International Conference on*. IEEE, 2015, pp. 1–4.

[15] Matthias Wagner, Fisnik Sulejmani, Alexander Melzer, Paul Meissner, and Mario Huemer, "Threshold-free interference cancellation method for automotive fmcw radar systems," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018, pp. 1–4.

[16] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher, "Deepsense: A unified deep learning framework for time-series mobile sensing data processing," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 351–360.

[17] Dong Yu and Li Deng, "Deep learning and its applications to signal and information processing [exploratory dsp]," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 145–154, 2011.

[18] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[19] Zachary C Lipton, John Berkowitz, and Charles Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[20] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[21] Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[23] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[24] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[26] Christian Sturm and Werner Wiesbeck, "Waveform design and signal processing aspects for fusion of wireless communications and radar sensing," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1236–1259, 2011.

[27] Christian Sturm, Yoke Leen Sit, Martin Braun, and Thomas Zwick, "Spectrally interleaved multi-carrier signals for radar network applications and multi-input multi-output radar," *IET Radar, Sonar & Navigation*, vol. 7, no. 3, pp. 261–269, 2013.

[28] Faruk Uysal and Sasanka Sanka, "Mitigation of automotive radar interference," in *2018 IEEE Radar Conference (RadarConf18)*. IEEE, 2018, pp. 0405–0410.

[29] Faruk Uysal, "Synchronous and asynchronous radar interference mitigation," *IEEE Access*, vol. 7, pp. 5846–5852, 2018.

[30] B Dekker, S Jacobs, AS Kossen, MC Kruithof, AG Huizing, and M Geurts, "Gesture recognition with a low power fmcw radar and a deep convolutional neural network," in *2017 European Radar Conference (EURAD)*. IEEE, 2017, pp. 163–166.

[31] Samuele Capobianco, Luca Facheris, Fabrizio Cuccoli, and Simone Marinai, "Vehicle classification based on convolutional networks applied to fmcw radar signals," in *Italian Conference for the Traffic Police*. Springer, 2017, pp. 115–128.

[32] Jiwoo Mun, Heasung Kim, and Jungwoo Lee, "A deep learning approach for automotive radar interference mitigation," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2018, pp. 1–5.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[34] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[35] Andrew Brock, Theodore Lim, James Millar Ritchie, and Nicholas J Weston, "Smash: One-shot model architecture search through hypernetworks," 2018.

[36] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le, "Understanding and simplifying one-shot architecture search," in *International Conference on Machine Learning*, 2018, pp. 550–559.

[37] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean, "Efficient neural architecture search via parameters sharing," in *International Conference on Machine Learning*, 2018, pp. 4095–4104.

[38] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin, "Snas: stochastic neural architecture search," *arXiv preprint arXiv:1812.09926*, 2018.

[39] Han Cai, Ligeng Zhu, and Song Han, "Proxylessnas: Direct neural architecture search on target task and hardware," *arXiv preprint arXiv:1812.00332*, 2018.

[40] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li, "Darts+: Improved differentiable architecture search with early stopping," *arXiv preprint arXiv:1909.06035*, 2019.

[41] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1294–1303.

[42] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, 2019, vol. 33, pp. 4780–4789.

[43] Eric Jang, Shixiang Gu, and Ben Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[44] Jianlong Chang, Yiwen Guo, GAOFENG MENG, SHIMING XIANG, Chunhong Pan, et al., "Data: Differentiable architecture approximation," in *Advances in Neural Information Processing Systems*, 2019, pp. 874–884.

[45] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong, "Pc-darts: Partial channel connections for memory-efficient differentiable architecture search," *arXiv preprint arXiv:1907.05737*, 2019.

[46] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli, "Pay less attention with lightweight and dynamic convolutions," *arXiv preprint arXiv:1901.10430*, 2019.

[47] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam, "Condconv: Conditionally parameterized convolutions for efficient inference," in *Advances in Neural Information Processing Systems*, 2019, pp. 1305–1316.

[48] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu, "Dynamic convolution: Attention over convolution kernels," *arXiv preprint arXiv:1912.03458*, 2019.

[49] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8817–8826.

[50] Lanlan Liu and Jia Deng, "Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[51] Zhihang Yuan, Bingzhe Wu, Zheng Liang, Shiwan Zhao, Weichen Bi, and Guangyu Sun, "S2dnas: Transforming static cnn model for dynamic inference via neural architecture search," *arXiv preprint arXiv:1911.07033*, 2019.

[52] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama, "Adaptive neural networks for efficient inference," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 527–536.

[53] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu, "Dynamic channel pruning: Feature boosting and suppression," *arXiv preprint arXiv:1810.05331*, 2018.

[54] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.

[55] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," 2009.

[56] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[57] Ilya Loshchilov and Frank Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[58] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," *arXiv preprint arXiv:1605.07648*, 2016.

[59] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[60] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[61] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[62] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 116–131.

[63] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[64] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[65] Jiwoo Mun, Seokhyeon Ha, and Jungwoo Lee, "Automotive radar signal interference mitigation using rnn with self attention," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3802–3806.

초 록

최근 딥러닝에 대한 관심이 높아지고 많은 연구가 진행되고 있습니다. 본 논문에서는 딥러닝을 레이더 신호처리에 적용하는 부분과 신경구조 탐색을 다루는 딥러닝 이론을 다룬다.

첫째, 자동차 시스템에서 레이더는 자율주행의 핵심 부품이다. 표적에 의한 송신 레이더 신호와 반사된 레이더 신호를 이용하여 표적의 범위와 속도를 포착할 수 있습니다. 그러나 간섭 신호가 존재하면 노이즈 플로어가 증가하고 대상 물체의 감지 가능성에 심각한 영향을 미칩니다. 간섭을 제거하거나 원래 신호를 재구성하기 위한 이전 연구들이 제안되었습니다. 그러나, 간섭을 제거하거나 송신 신호를 재구성하는 기존의 신호 처리 방법은 복잡도가 높은 작업이며 많은 제약이 따른다. 이 작업에서는 딥 러닝을 사용하여 간섭을 완화하는 새로운 개념을 제안합니다. 제안하는 방법은 다양한 간섭 조건에서 높은 성능을 제공하며 처리 시간이 짧다. 또한 제안하는 방법이 기존의 신호 처리 방법에 비해 더 나은 성능을 보인다는 것을 보였다.

둘째, 신경망 구조 검색(NAS) 방법은 사람의 도움 없이 자동으로 최적의 신경망을 찾습니다. 그래디언트 기반 검색으로 아키텍처를 찾기 위해 NAS에 대한 수많은 알고리즘이 연구되었습니다. Gradient 기반 검색의 핵심 논문 중 하나인 Differentiable Architecture Search(DARTS)는 검색 비용을 획기적으로 줄였으며, 지속적인 완화와 메타 학습 기반 근사화를 통해 뛰어난 성능을 보여주었습니다. 그러나 DARTS의 문제 중 하나는 그래디언트 기반 검색 프로세스가 중첩된 이중 수준 구조와 그래디언트 디센트의 탐욕스러운 동작으로 인해 편향된다는 것입니다. 결과적으로 검색 공간이 제한된 아키텍처 집합으로 제한되는 문제가 있습니다. 그래디언트

기반 검색의 편향성을 극복하기 위해 동적 검색 방법을 사용했습니다. 이 기술을 사용하면 그래디언트 기반 검색이 탐색 효과를 가질 수 있습니다. 이 논문에서는 새로운 접근 방식인 DE-DARTS(Dynamic-Exploration DARTS)를 제시합니다. 효과적인 탐색을 위해 DE-DARTS에서 입력 데이터를 기반으로 모델 아키텍처를 변경하는 동적 주의 네트워크(DAN)를 사용합니다. 우리의 DAN은 검색 초기에 활성화되기 때문에 검색 초기에 입력 데이터에 따라 다양한 아키텍처가 고려됩니다. 우리의 알고리즘은 CIFAR-10, CIFAR-100 및 ImageNet을 포함한 여러 이미지 분류 데이터 세트에서 평가되었으며 향상된 성능을 보여줍니다.

셋째, 신경구조 탐색 방법을 기반으로 레이더 간섭 제거에 적용한다. 이 모델은 DARTS 논문을 기반으로 합니다. 신경망 구조 검색 기반 모델을 적용한 결과 직접 모델을 설계하지 않고도 AI가 만든 rnn 기반 모델로 레이더 간섭을 잘 제거할 수 있었다.

주요어: 레이더, 신호처리, 신경구조탐색, 동적탐색

학번: 2016-27442

ACKNOWLEGEMENT

석박사 통합 과정을 10년가까이 하면서 학위 논문을 제출하게 되었습니다. 미흡하지만 학업을 지속하는 동안 여러 교수님과 선후배님들, 그리고 친구들의 도움이 있었기에 이 논문을 완성하는 것이 가능하였다고 생각합니다.

우선 지도교수님이신 이정우 교수님께 감사 인사를 드리고 싶습니다. 박사 학위 동안 저를 처음부터 지도해 주셨고 제가 논문 주제에 대해 고민을 할때마다 새로운 아이디어를 제시해주셨습니다. 또한 논문 심사 때도 부족한 점이 있을 때마다 사려 깊은 조언을 해주셔서 제가 성공적으로 박사 학위 심사를 마칠 수 있었습니다. 앞으로 사회에 나가서도 교수님을 본받아 좋은 학자 및 연구자가 되도록 노력하겠습니다.

그리고 부족한 저의 논문에 시간을 할애하여 심사해주신 노종선 교수님, 정교민 교수님, 전세영 교수님, 김건우 교수님께 진심으로 감사드립니다. 연구 주제에 대한 아낌없는 지도와 여러가지 조언들을 해주셨기 때문에 보다 완성도 있는 학위 논문을 작성 할 수 있었습니다.

다음으로 제가 연구실에서 연구하는 동안 많은 도움을 주었던 연구실 동료들에게도 감사의 인사를 하고 싶습니다. 연구실에 있는 동안 항상 열정적인 분위기로 연구를 진행하는 동료들을 보면서 많은 힘이 되었습니다. 제가 논문을 처음 쓸때 작성법 및 여러 어려운 점들을 도와주었고 연구 진행이 막힐 때마다 새로운 아이디어를 제시해주었습니다.

마지막으로 연구 활동을 하는 동안 저를 믿어 주시고 든든히 지원해주신 우리 사랑하는 가족들, 아버지, 어머니, 형에게 정말 감사드립니다. 가족들의 이해와 헌신이 없었다면, 긴 학업기간동안 연구에 전념하지 못했을 겁니다.

이 외에도 많은 분들의 도움과 응원들이 있었기 때문에 이 논문을 완성할 수 있었습니다. 다시 한번 모든 분들께 감사드리며, 앞으로의 삶에 있어서도 항상 감사한 마음을 잊지 않겠습니다.

2023년 1월

문지우 올림