FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

LTE
Lehrstuhl für Technische Elektronik

Lehrstuhl für Technische Elektronik
Prof. Dr.-Ing. Dr.-Ing. habil. Robert Weigel
Prof. Dr.-Ing. Georg Fischer

Masterarbeit

im Studiengang
"Elektrotechnik, Elektronik und Informationstechnik (EEI)"

von

Anqi Dai

zum Thema

Radar Data Denoising using GAN

Betreuer:	Anand Dubey
	Jonas Fuchs
Beginn:	01.07.2018
Abgabe:	03.02.2019

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde.

Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 27.02.2020

Anqi Dai

Kurzfassung

Das Entrauschen von Bildern ist ein traditionelles Problem in der digitalen Bildverarbeitung, obwohl es erst ein halbes Jahrhundert alt ist. Trotz seines Alters bleibt es ein aktives Thema, insbesondere im Bereich der Radardaten. Bei einem Radarsystem sind die von einem Ziel reflektierten Signale mehreren Rauschfaktoren wie Unordnung, Mehrwegereflexion und gegenseitiger Interferenz ausgesetzt. Ferner hängt die Leistung des empfangenen Signals von ac RCS und der Entfernung des Ziels ab. Der kombinierte Effekt kann zu einer Verringerung der Signalleistung unter dem Grundrauschen führen. Infolgedessen sind Aufgaben wie Klassifizierung oder Nachverfolgung sehr anfällig für falsche Erkennung. In dieser Arbeit werden einige Methoden des Standes der Technik wie ac PCA, ac PPCA und Gammakorrektur implementiert. Unter Verwendung von konvoltionsbasierten neuronalen Netzen werden verschiedene ac AE - und GAN-basierte Architekturen untersucht. Zu diesem Zweck wurden synthetische gepaarte verrauschte und rauschfreie Radar-Entfernungs-Doppler-Karten für das Training verwendet, während die gemessenen Daten validiert und getestet wurden. Schließlich hat ein qualitativer und quantitativer Vergleich gezeigt, dass die Ergebnisse des GAN-Frameworks im Bereich der Entrauschung besser sind.

Schlüsselwörter: AE, Denoise, GANs, Gamma correction, PCA, PPCA

Abstract

Image denoising is a traditional issue in digital image processing, although only half a century old. Despite its age, it remains an active theme, especially in the field of radar data. For a radar system, the reflected signals from a target are prone to multiple noise factors like clutter, multipath-reflection and mutual interferance. Further, power of received signal depends on radar cross section (RCS) and distance of target. The combined effect may result in reduction of signal power below noise floor. As a result, tasks like classification or tracking are highly prone to false detection. In this thesis, some state-of-the-art methods such as Principal component analysis (PCA), Probability principal component analysis (PPCA) and Gamma correction are implemented. Further, taking advantage of convoltional based neural networks, different Autoencoder (AE) and GAN based architecture are explored. For this, synthetic paired noisy and noise-free radar range-Doppler maps were used for training, while validation and test was carried out on the measured data. Finally, qualitative and quantitive comparision demonstrated the results of the GAN framework is better in denoising area.

key words: AE, Denoise, GANs, Gamma correction, PCA, PPCA

Contents

List of Abbreviations

Adam	Adaptive Moment Estimation
AE	Autoencoder
cGAN	Conditional generative adversarial neural network
CNN	Convolutional neural network
CRT	Cathode ray tube
CW	continuous wave
FFT	Fast Fourier Transformation
FM	Frequency-modulated
FMCW	Frequency-modulated continuous wave
GAN	Generative adversarial neural network
HRRP	High resolution range profile
ISAR	Inverse synthetic aperture radar
MSE	Mean Square Error
PCA	Principal component analysis
PPCA	Probability principal component analysis
PSNR	Peak Signal to Noise Ratio
RATR	Radar automatic target recognition
RCS	radar cross section
RD	Range-Doppler
RDM	Range-Doppler map
ReLu	Rectified linear unit
SAR	Synthetic aperture radar
SSIM	Structural Similarity Index
VAE	Variational autoencoder
VCO	Voltage controlled oscillator

1 Introduction

1.1 Motivation

The term radar was originally coined by the US Navy in 1940 [1]. It was first used as a radio detection and ranging device. Later, due to the needs of military during the Second World War, radar was used by many countries, mainly for military purposes such as search, interception, and identification of targets. The alignment radar technology has entered a stage of rapid development. Nowadays radar is no longer just for military purposes, its uses have become more diverse. Including air and ground traffic control, landmark positioning, marine resource detection, meteorological monitoring, and geological observations such as volcanoes [2].
The main purpose of traditional radar is detection and ranging. Frequency-modulated (FM)-frequency radars are particularly suitable for these applications. Frequency-modulated continuous wave (FMCW) [3] radars use low intercept probability waveforms, which are very suitable in using with simple fixed transmitters and are used in naval tactical navigation, automotive application and other fields like industrial automation. But in modern military warfare, traditional radars no longer meet operational requirements. Therefore, Radar Automatic Target Recognition (RATR) technology was born [4]. With the continuous development of high-resolution radar technology, synthetic aperture radar and inverse synthetic aperture radar, more structural information from targets can be obtained, thereby promoting the development of automatic target recognition technology. In literature, the most common approach for an automatic target recognition are done using High Resolution Range Profile (HRRP) target recognition [5], Synthetic Aperture Radar (SAR) image target recognition [6] and Inverse SAR (ISAR) image target recognition [7].
Due to the influence of environmental noise during detection and noise during signal transmission, radar images are highly affected by noise such as coherent speckle noise in SAR images [8]. The noise in the radar image is not only detrimental to the extraction of image information and interpretation of the image, but also reduces the signal-to-noise ratio of the image and the sharpness of the texture of the radar image, especially for some high-resolution radar images. In the noise suppression process, it is desirable to minimize the interference caused by noise, and to keep the details such as edge contours and textures in the radar image intact. Therefore, radar data denoising is also one of the research hotspots and difficulties in the field of radar data application.

1.2 Objective

After AlphaGo defeated Lee Sedol, the term deep learning became a very hot topic. At present, many intelligent products are based on deep learning as the core technology. Deep learning was first applied to image recognition. In the following years, with the advent of convolutional neu-

ral networks(CNN) [9] and generative adversarial neural networks(GAN) [10], deep learning was promoted to various fields, including speech recognition, audio processing, natural language processing, smart furniture, automatic diagnosis of medical CT films, game production, search engines and advertising pushing. It can be seen that deep learning has begun to affect all aspects of people's lives. In the problem of image classification, deep learning has successfully reduced the error rate from 26% to 3.5% [11]. In speech recognition, deep learning has reduced the error rate by 25%, far exceeding the performance of the past few years [12]. Deep learning can extract the deep features of the target, and has good performance in target recognition. Therefore, applying the deep learning method to radar data denoising is the focus of this thesis.

1.3 Construction of the Thesis

In this thesis, noise reduction is performed based on FMCW radar signals. Different conventional and deep learning based methods are utilized for performance. As a result, Gamma correction, PCA and PPCA, AE and GAN are explored.
This thesis is arranged as follows:
Chapter 1 summarizes the research background of the thesis topic and the importance of noise reduction in the radar field, combines the development of deep learning, and finally gives the main content of this paper.
Chapter 2 briefly introduces basic concepts of radar system and signal processing used in during this thesis, a overview of convolutional neural networks and generative adversarial neural networks are introduced. Finally, data processing and the software on which this thesis are based are described.
Chapter 3 describes about different approach used for denoising in depth. In the starting, math behind Gamma correction is explained followed by working principle of PCA and PPCA.
Chapter 4 introduces the principle of the autoencoder and the generative adversarial networks, and then describes the implementation of these two neural networks in detail.
Chapter 5 describes the criteria for performance evaluation and compares the results of previous applications.
Chapter 6 summarizes the main work of this thesis and points out the content and direction of sustainable research in this thesis.

2 Fundamentals

2.1 Radar

2.1.1 Radar System

Radar combines radio detection with ranging systems. It's actually an electromagnetic system that detects the position and distance between the object and the radar. It radiates signal energy into space and monitors echoes or reflected signals from objects. Radars can be divided into pulse radars and continuous wave (CW) radars according to the types of transmitted signals. Conventional pulse radars emit periodic modulated pulse signals, while continuous wave radars emit continuous wave signals. Generally, pulse radars have higher peak power and smaller duty cycle, while continuous wave radars have 100% duty cycle and lower power. The FMCW radar used in this thesis is very useful in achieving high-resolution scenes with low transmit power, including automotive radar, close-range imaging and many other application scenarios.
Radar systems typically consist of transmitters that produce electromagnetic signals that are radiated into space by antennas. When an electromagnetic signal hits an object, it bounces off the surface of the object based on the reflection effect. The reflected or echo signal is then picked up by the radar antenna and transmitted to the receiver, where the information is processed. The range is determined by calculating the time it takes for the signal to reach the target from the radar and return.

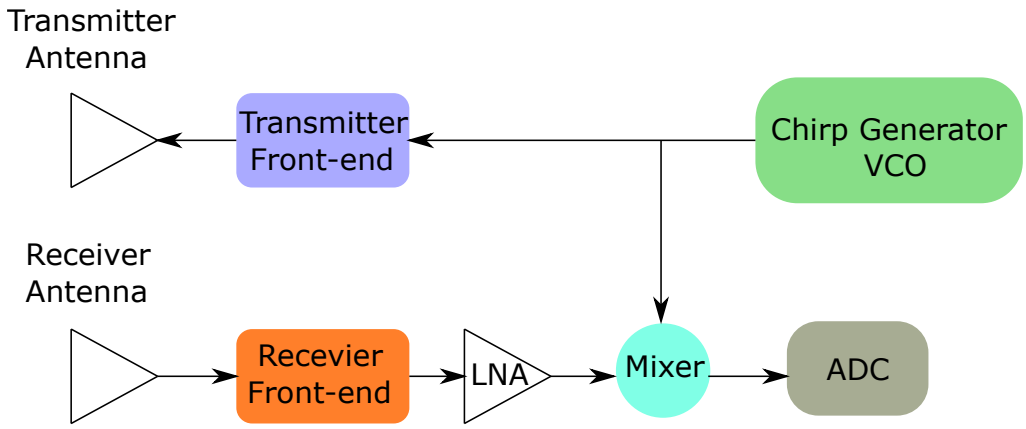

Figure 2.1: FMCW structure diagram

The basic structure of the FMCW radar is generally composed of voltage controlled oscillator (VCO), duplexer, antenna and receiver. The structure diagram is shown in Figure 2.1. The radar uses a duplexer, so that the transceiver can share the antenna and ensure the continuous wave operating mode of the radar. The echo generated by the transmitted linear frequency sweep signal after being reflected by the target passes through the antenna and duplexer, enters a mixer,

and performs self-beat mixing with the transmitted signal to obtain a lower frequency beat frequency signal. The frequency information in the target has distance and speed information of the target, and then the signal processor can obtain relevant information about the target.

2.1.2 Signal Processing

The task of signal processing is to extract the beat frequency of the echo signal [13]. First, the beat frequency signal is amplified for signal acquisition, and then the acquisited signal will be sent to a signal processor for Fast Fourier Transform (FFT). The signal in the time domain is converted into the frequency domain, and the power spectrum distribution of the echo beat frequency signal is obtained, so as to obtain the one-dimensional range spectrum of the radar on the range axis.

Range Estimation

There are several different waveforms designed in the literature that are used in FMCW such as sawtooth, triangle and sinusoidal [14]. In this thesis, we will consider a sawtooth model of the FMCW signal, seen in Figure 2.2.

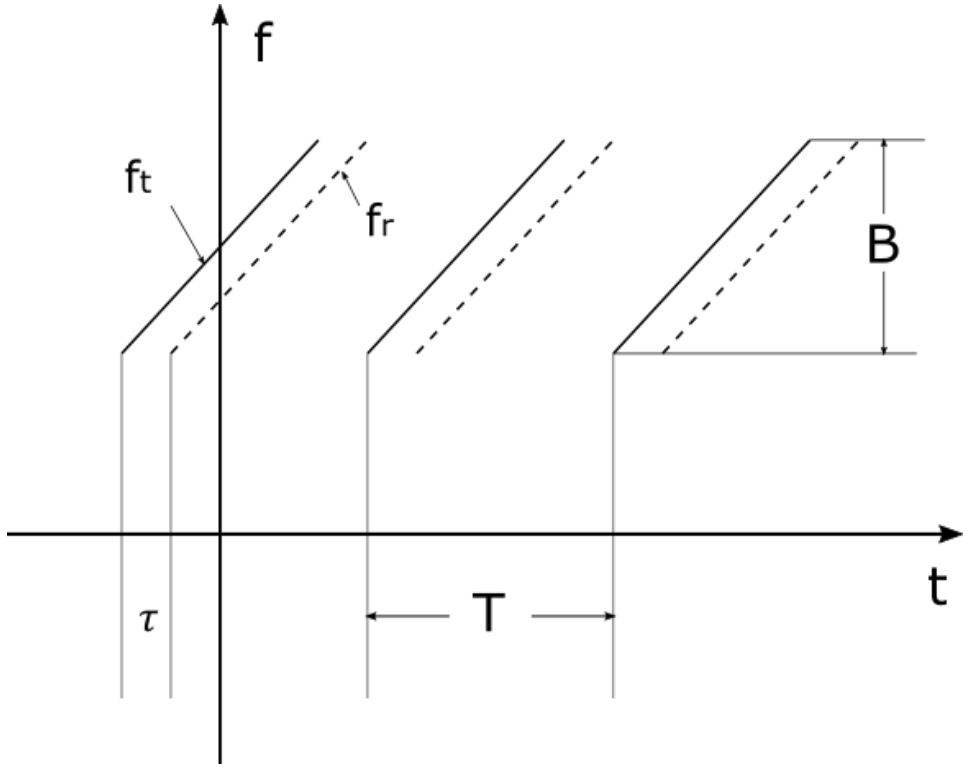

Figure 2.2: time frequency diagram

Where T is the chirp duration of the sawtooth wave signal, τ is the time delay of the echo signal, and B is the bandwidth. In Figure 2.2, the solid line is the curve of the frequency of the transmitted signal over time, and the dashed line is the curve of the frequency of the echo signal

over time. The frequency of the signal changes with time. As shown in Figure 2.2, according to the form of the signal, the frequency of the transmitted signal can be expressed in:

$$f_t = f_0 + \frac{\mathrm{d}f_t}{\mathrm{d}t}t \tag{2.1}$$

The frequency of the echo signal is:

$$f_r = f_t(t-\tau) = f_0 + \frac{\mathrm{d}f_t}{\mathrm{d}t}(t-\tau) \tag{2.2}$$

The frequency of the beat frequency signal obtained by the mixer after mixing the transmission with the echo signal is:

$$f_b = |f_t - f_r| = \frac{\mathrm{d}f_t}{\mathrm{d}t}\tau \tag{2.3}$$

As can be seen from Figure 2.2:

$$\frac{\mathrm{d}f_t}{\mathrm{d}t} = \frac{B}{T} \tag{2.4}$$

$$\tau = 2R/C \tag{2.5}$$

Where τ is the time required for the signal to transmit and receive, R is the range from the FMCW radar to the target, and C is the velocity of the electromagnetic wave in the air.
So we get:

$$R = \frac{TC}{2B}f_b \tag{2.6}$$

According to Equation (2.6), it can be known that under the condition that the modulation parameters bandwidth B and chirp duration T are constant, the beat frequency f_b is proportional to the Range R, and the Range R can be calculated by measuring the beat frequency f_b.

Range Doppler Matrix

Range-Dopple Matrix is an effective method for multi-target information extraction of FMCW radar. It performs fast-time and slow-time dimensions on multiple periodic chirp sequences and echo signal sent by the radar. The range-Doppler (RD) heat map can be obtained, and the range and velocity information of multiple targets can be extracted.
According to the beat signal of FMCW, we know that the frequency of the beat signal is

$$f_{movingBeat} = f_{staticBeat} \pm f_d = \frac{2BR}{CT} \pm \frac{2fv}{C} \tag{2.7}$$

where $f_{movingBeat}$ and $f_{staticBeat}$ are the frequencies of the beat signal in the moving and stationary states of the target, f_d is the Doppler frequency, B is the sweep bandwidth, R is the target range, C is the velocity of waves, T is the chirp duration, f is the chirp signal center frequency, v is the velocity of the target.

Range-FFT

In fast time dimension processing, the chirp sequence of a single cycle has a short cycle time, which is short enough to neglect the influence of the Doppler frequency, so it is relatively easy to determine the relative range R in each chirp. At this time, it can be considered that the moving target beat frequency $f_{movingBeat}$ of the target in the range dimension is approximately equal to the static target beat frequency $f_{staticBeat}$.

$$f_{movingBeat} \approx f_{staticBeat} = \frac{2BR}{CT} \tag{2.8}$$

In order to obtain the relative range R of each chirp, the beat frequency in each chirp must first be obtained. Therefore, each chirp is sampled, then a FFT is used to obtain the beat frequency in each Chirp to obtain the relative distance R. Since the range is obtained after performing an FFT on each chirp, this process is also called Range-FFT.

Doppler-FFT

In the processing of fast time dimension, the impact of speed is considered to be negligible. By stacking multiple Chirp sequences over multiple frames of data in the slow time dimension, the frequency impact of speed cannot be ignored. The frequency obtained in the slow time dimension is the Doppler frequency.

$$f_d = \frac{2fv}{C} \tag{2.9}$$

The processing of the slow time dimension is the result of performing FFT on the same range unit after accumulation of multiple chirp sequences. Since the relative velocity is obtained after the FFT, this process is also called Doppler-FFT.
After processing, the following range-Doppler Map (RDM) can be obtained.

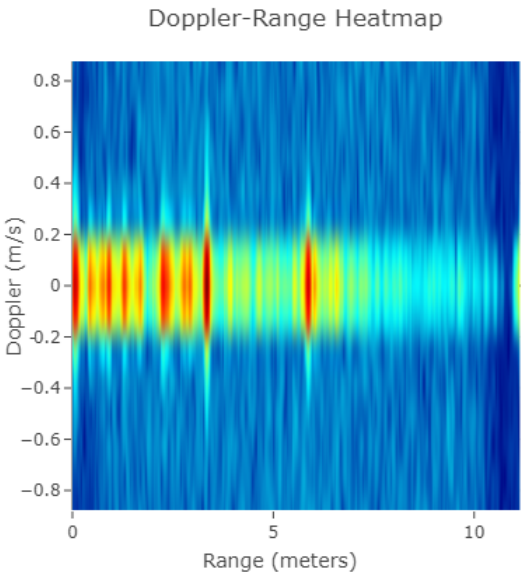

Figure 2.3: Range-Doppler Map

2.2 Neural Network

2.2.1 CNN

In traditional machine learning, the selection and extraction of features need to rely on a manual method, which is too professional and has limited effects. Therefore, how to let the computer automatically select and extract features has attracted researchers' attention. Artificial neural networks have emerged as the times require. The basic idea of the neural network is to imitate the human brain nerves, information is transmitted between neurons. By inputting some samples to be learned into the network and letting the network train according to certain criteria, a network with learning ability can be obtained. With the development of science and technology and the improvement of computer computing capabilities, more and more complex networks have been proposed, the learning capabilities of neural networks have also become stronger.
A neural network is composed of many neurons, and each neuron is connected to each other. When the signal sent by a certain neuron is greater than the threshold between the two neurons, the neuron is activated to achieve information transmission. Classic neural networks include perceptron networks [15], Hopfield networks [16] and deep learning networks.
Convolutional neural network (CNN) is a very efficient neural network. In CNN, researchers use part of the connection and weight sharing strategy to greatly reduce the number of parameters that need to be learned. CNNs do not require manual selection of image features. As the number of layers in the network continues to deepen, CNNs can gradually learn deeper features of the image, the network becomes less and less sensitive to the scale, translation and rotation of the image, which has a great effect on image classification.

Basic Architecture of CNN

As shown in Figure 2.4, the basic network structure of a CNN are consist of convolutional layer, fully connected layer, activation layer, pooling layer and fully connected layer.

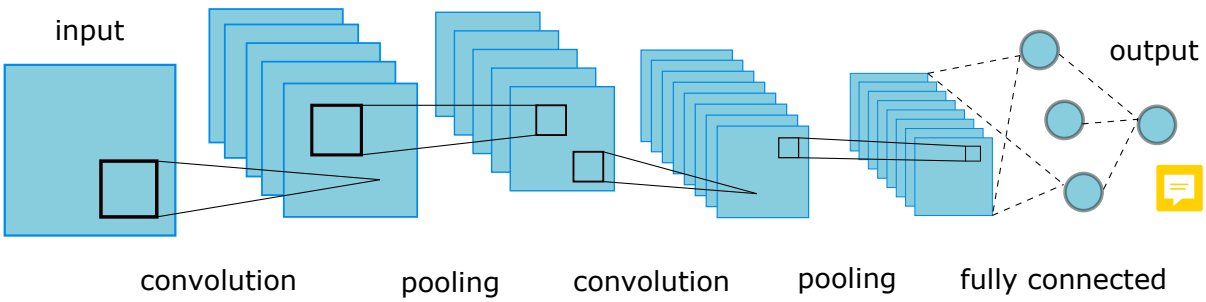

Figure 2.4: basic architecture of CNN

From input to output, the layers of the CNN establish relationships with each other through different computational neural nodes and pass input information layer by layer; the continuous convolution-pooling structure decodes the characteristic informations of the original data and maps them to the feature space of the hidden layer, and the subsequent fully connected layers are classified and output according to the extracted features.

Convolution Operation

Mathematically, convolution is an important analytical operation. It is a mathematical operator that generates a third function from two functions f and g. It represents the area of the overlapping part of the function f and the function g that has been flipped or translated. The calculation formula is usually defined by Equation (2.10).

$$z(t) = f(t) * g(t) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(t-\tau) \tag{2.10}$$

In image processing, a digital image can be regarded as a discrete function in a two-dimensional space, denoted as $f(x,y)$. Assuming a two-dimensional convolution function $g(x,y)$, the output image $z(x,y)$ will be generated, which can be expressed by Equation (2.11):

$$z(x,y) = f(x,y) * g(x,y) \tag{2.11}$$

Convolutional Layer

The convolutional layer in CNN uses local connection and convolution kernel weight sharing. The input and output of the convolutional layer are multi-dimensional feature maps. The convolution kernel passes a sliding window, and the sum of the products of the weights corresponding to the input feature maps in the window serves as the output. The output of the convolutional layer is usually a 4-dimensional feature map, which is the number of samples, the number of channels, the height and the width.
The way of local connection is inspired by the way of human visual perception. Human visual perception is from local to global, and local connection makes the CNN's extraction of image features also locally relevant. Local connection is a connection relationship between neurons in the current layer and neurons in the local area in the output feature map of the previous layer. Neurons in the current layer can only perceive local areas of the features in the previous layer. In a convolutional layer, the feature maps connected between layers are extracted from pixel-level image features by one or more convolution kernels (also known as filters) through convolution operations, and the result of the convolution operation passes an activation function. The mapping transformation constitutes the feature mapping relationship from input to output. Each convolution kernel uses a sliding window to traverse the entire feature map. The convolution kernel gathers and fuses the feature information of each small area to complete the characterization of a small local area of the image. There is usually more than one convolution kernel in a convolutional layer. Each convolution kernel is responsible for extracting a convolution feature. The convolution process of the convolutional layer is shown in Figure 2.5.

Activation Layer

Starting from early artificial neural networks, the interconnected neural nodes established a mapping relationship from input to output through activation functions. The activation layer

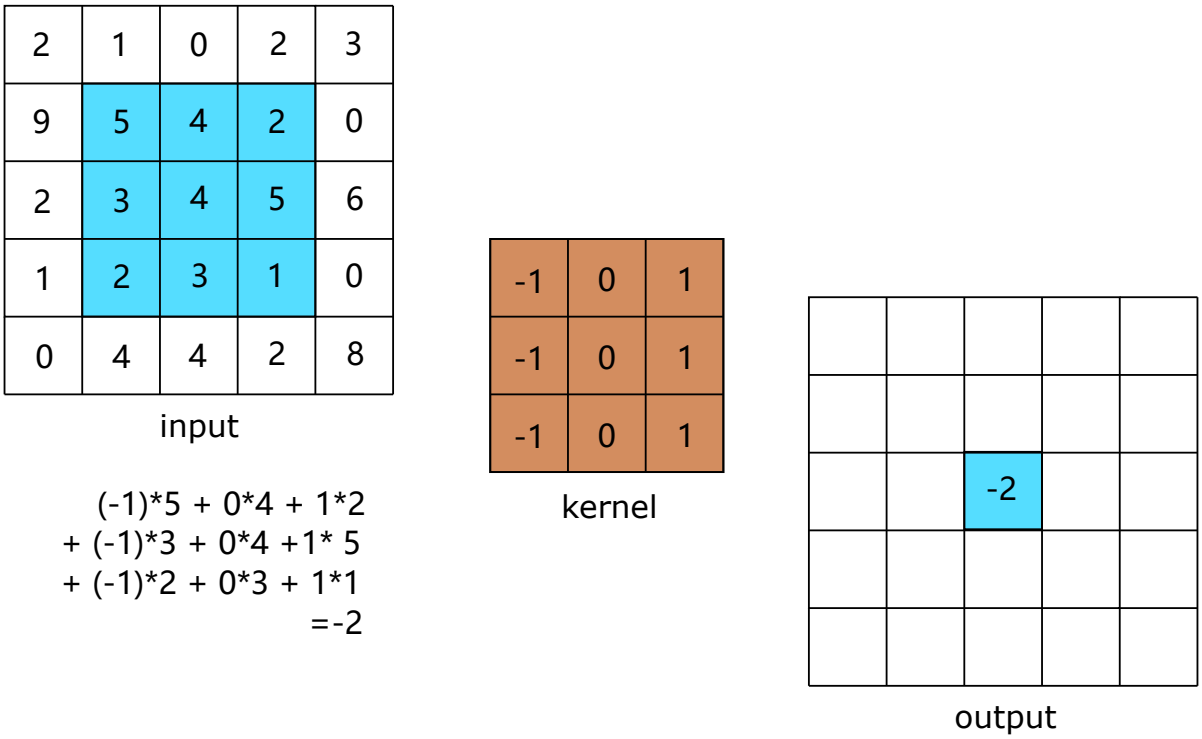

Figure 2.5: Calculation of convolutional layers

mainly sets the activation function after the convolutional layer. Its essence is a function mapping, which maps the input data and provides the network's non-linear modeling capabilities. During the calculation, the calculation is performed element by element without changing the size of the original data, which means the input and output data sizes are equal.
In the actual environment, deep learning models are used to establish a fitting relationship with actual data, and the data itself usually exhibits a non-linear distribution. These non-linear activation functions give deep neural networks the ability to learn hierarchically non-linear maps. Therefore, the activation function in the network is an important part of the deep neural network. The commonly used activation functions in neural networks are sigmoid function [17], ReLu function [18] and Leaky ReLu function [19], which also used in this thesis.

sigmoid function

The sigmoid function is also called Logistic function, which is usually used for the output of hidden neurons. The Equation (2.12) of the sigmoid function is given by

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.12}$$

Figure 2.6 shows the sigmoid function in the coordinate system. As can be seen from the figure 2.6, the sigmoid function maps the variable with a value of $(-\infty, +\infty)$ to $(0, 1)$, which is suitable for a binary classification problem. However, the sigmoid function is not often used as the activation function in actual neural networks. It can be seen from the figure that as x increases, the gradient of the function decreases rapidly or even approaches zero. This

will cause the gradient of the weight W to be close to 0, so that the gradient update is very slow, which cause the gradient vanishing. And the mean of the function is not 0, which is not conducive to the calculation of the subsequent layers.

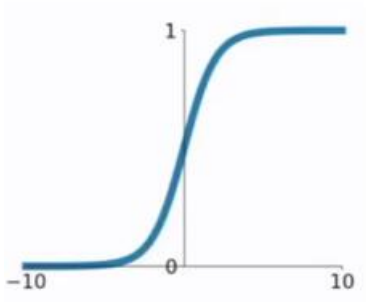

Figure 2.6: Sigmoid Function

ReLu Function

The ReLU function is also called the Rectified Linear Unit, which is considered the most common activation function in neural networks. It is a piecewise linear function that solves the problem of vanishing gradients in the sigmoid function. The Equation of the ReLu function is as follows:

$$f(x) = \begin{cases} x & \text{x} > 0 \\ 0 & \text{x} \leq 0 \end{cases} \tag{2.13}$$

Figure 2.7 shows the ReLu function in the coordinate system. It can be seen from the figure 2.7 that when the value of x is positive, the derivative of ReLu is a constant, which will not cause the problem of gradient vanishing. Regardless of forward or reverse propagation, ReLu's calculation speed is much faster than sigmoid because of the linear relationship of ReLU function. However, when x is negative and the gradient is 0, there will still be a problem of the gradient vanishing, so the ReLu function is improved .

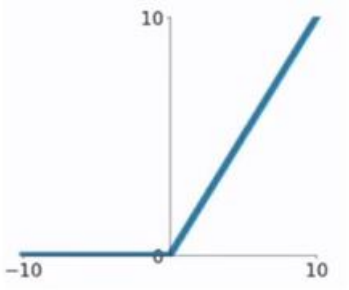

Figure 2.7: Relu Function

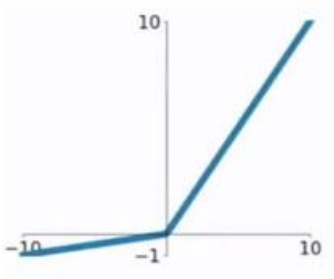

Figure 2.8: Leaky Relu Function

Leaky Relu Function

The Leaky ReLu function is an improvement over ReLu. It gives the function a small gradient when x is negative, which solves the problem of gradient vanishing in the case of $x < 0$. The Equation of Leaky Relu is given as follows:

$$f(x) = \begin{cases} x & \text{x > 0} \\ 0.01x & \text{x } \leq 0 \end{cases} \tag{2.14}$$

The diagram of Leaky Relu is shown in the figure 2.8.

Pooling Layer

The pooling layer is also called the downsampling layer. Its main operation is to select a value in the pooling window as the value after downsampling. In CNNs, a pooling layer is usually added after the activation layer. There are two main pooling methods: Max-Pooling and Average-Pooling. The use of pooling layers in CNNs has the following advantages:

- The feature map in a CNN usually has a large size. A extremely large size of the feature map will cause a large amount of calculation of the network, and the network will cause excessive memory consumption during the calculation. By pooling the layer and downsampling the features, the feature map scale can be reduced, thus reducing the amount of computation and memory usage.
- Through the pooling process, the robustness of network translation and deformation is enhanced.

The maximum pooling method is to select the maximum value from the pooling window as the pooled value, the Equation (2.15) is given by:

$$f(x) = max(x_{ij}) \tag{2.15}$$

where x_{ij} belongs to the value in the corresponding input feature map in the pooling window. The average pooling method is to take the average of all values in the pooling window as the pooled value, the formula is shown in (2.16).

$$f(x) = \frac{1}{n} \sum_i \sum_j x_{ij} \tag{2.16}$$

where n is the number of neurons in the pooling window.

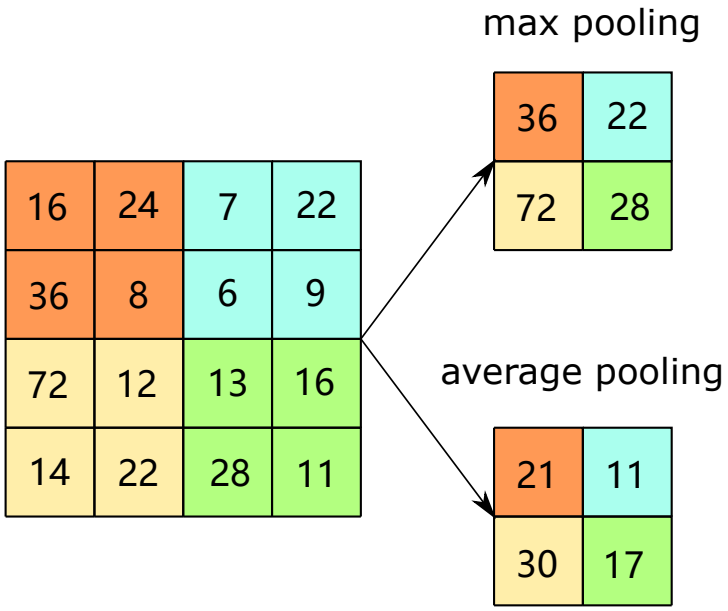

Figure 2.9: the operation of pooling

Fully Connected Layer

The fully connected layer appears at the end of the network structure and is a traditional multi-layer perceptron network. Every neuron in the fully connected layer is fully connected to every neuron in the previous layer, which is also the source of the name of the fully connected layer. The final output of the network is the high-level features of the input image, and after statistical calculation by the classifier, the probability of the category label corresponding to the input image is output. Softmax regression classification model [20] is often used as the last layer of the fully connected layer, and the output value is the probability of each category between $0-1$.

Convolutional network training method

In deep learning theory, most CNNs now build neurons layer by layer by building a multilayer structure. Training a single layer network at a time makes the error propagate layer by layer. The training process mainly involves two stages of forward propagation and back propagation. Forward propagation is mainly used for the transmission of feature information, and back propagation reflects the correction and fine-tuning of model parameters based on error information.

Forward propagation

The process of forward propagation is to feed the input value to the neural network and get the output of the predicted value. When the input value is provided to the first layer of the neural network, it does nothing. The second layer takes values from the first layer and performs multiplication, activation operations, then passes the obtained values to the next layer. The

same operation is performed in the subsequent layers, and finally we get an output value in the last layer.

Back propagation

After the forward propagation, an output value called a predicted value is obtained. To calculate the error, the predicted value is compared with the actual output value. Use the loss function to calculate the error value, then calculate the derivative of each error value and each weight in the neural network. Backpropagation uses the chain rule of differentials. In the chain rule, the derivative of the error value corresponding to the weight of the last layer is first calculated. These derivatives are gradients, and these gradient values are then used to calculate the gradient of the penultimate layer. Repeat this process until get the gradient of each weight in the neural network. This gradient value is then subtracted from the weights to reduce the error value. In this way, it is closer to the local minimum.

Loss Function

Deep learning neural networks are trained using the stochastic gradient descent optimization algorithm. As part of the optimization algorithm, the error for the current state of the model must be estimated repeatedly. This requires the choice of an error function, conventionally called the loss function, that can be used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation.
The loss function is used to optimize the parameter values in a neural network model to reduce the loss of the neural network. In the field of deep learning, there are many different loss functions. In this thesis, the cross-entropy Loss [21], SSIM loss [22] and perceptual loss [23] are introduced.

Optimizer

Optimizers are algorithms or methods used to change the parameters of the neural network such as weights and learning rate in order to reduce the losses and speed up the neural network training. The Optimization algorithms are responsible for reducing the losses and to provide the most accurate results possible. The most common used optimization algorithm is gradient descent. The optimizer that used in this thesis is Adaptive Moment Estimation (Adam) [24], which is an adaptive learning rate adjustment algorithm.

2.2.2 GAN

background of GAN

In the field of deep learning, generative adversarial networks (GAN) is another major breakthrough after CNNs. Known as the father of CNNs, Professor Yann Lecun called it ”the coolest idea in machine learning in the last twenty years”. GANs have amazing effects when dealing with difficult data generation tasks, such as image generation, text generation and speech generation, but what really makes the adversarial network shine is the image field.

In the image field, GANs are often used for tasks such as image generation and image translation which have achieved amazing results on such tasks [25]. The latest research has even begun to use it to generate video and three-dimensional object models [26][27]. Since this thesis uses the idea of GANs to solve the problem of radar image denoising, this part will introduce the basic idea of the GANs above all.

GAN Introduction

GANs were first proposed by Goodfellow et al. in 2014 [10]. Its core idea is same like the idea of Two-player Zero-sum Game. In a two-player Zero-sum Game, the two playes compete with each other, and the gain of one player necessarily means the loss of the other player. The sum of the gain and loss of the two player is always ”zero”.

In GANs, these two players are two neural networks: a generative network (also called a generator) and a discriminative network (also called a discriminator). Usually, they all use CNN structures. The role of the generator is to generate an image. It takes random noise z as input and outputs a generated fake image. On the other hand, the role of the discriminator is to learn to distinguish between true and false. It takes real images and fake images as input, and outputs the probability that the image is a real image. In the course of this game, the generator G needs to try to generate a more realistic image to deceive the discriminator; The discriminator D plays the role of punishing the generator G. It improves the discrimination ability by reading a large number of positive and negative samples, recognizes the false samples generated by G, and punishes G through the loss function. Then, the generator G further adjusts the network parameters according to the feedback result of D, thereby generating more realistic samples. The generator and discriminator are both trained at the same time until the Nash equilibrium [10] is reached. The data distribution generated by the generator completely coincides with the real data distribution, and the discriminator can no longer correctly distinguish the generated sample and the real sample, which means the classification accuracy is 50%. At this time, both the generator and the discriminator are optimal. The flowchart 2.10 of GAN is shown below:

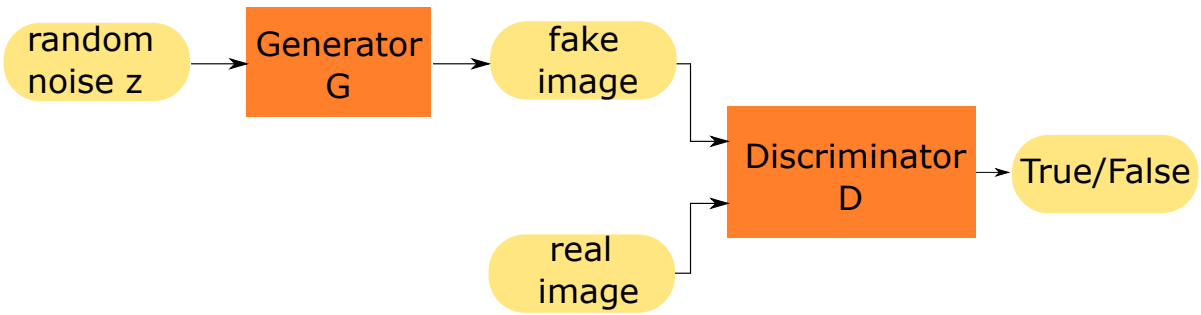

Figure 2.10: flow chart of the the adversarial generative network

2.3 Software Framework

2.3.1 Data Processing

In thesis thesis, two parts of the data sets are evaluated. The first data set is simulated by the software MATLAB. The other data set is measured in the real word, which is also combined with simulated interference. The objects which we selected are cars and humans, whether in the simulator or in the real-world measurements. In this way, we can obtain more training data sets under limited measurement conditions. At the same time, we can use real measured data as validation and test data to make it to be the basis for training and evaluation.

Simulation

The implemented radar is FMCW Radar, which is used for the data collection. The used radar operates with the start sweep frequency at 76 GHz and the measurable maxium doppler is $v_{max} = 3.9m/s$. In fact, the generated FMCW signal represents as the form of Time-Freuqnece signal. During the Short-Time Fourier Transform (STFT) (2.17), the ranger-doppler map (RDM) can be visualized.

$$S(m,f) = 20\log\left\|\sum_{n=-\infty}^{\infty} w(t-m)x(t)exp(-j2\pi)\right\| \tag{2.17}$$

where $x(t)$ is the input continuous time signal, $w(t-m)$ represents a smooth window to sample the signal at $t-m$. In this thesis, the window size implemented is 256. The details of FMCW Radar parameters for automotive scenario simulation is shown in the table 2.1 as follow:

Table 2.1: FMCW Radar Parameters for measurement

Radar Parameters	Values
Sweep start frequency, f_0	76 GHz
Sweep Bandwidth, B	5 GHz
Chip duration, T_C	41.6 μs
Chirp repetition time, T_{PRT}	83.45 μs
Sampling frequency, f_s	40 MHz
Number of fast-time samples, N_C	256
Maximum range, R_{max}	24m
Maximum velocity, V_{max}	5.2 m/s

After measurement, we get 199 snapshots of RDMs as the original radar data. Later, as post processing interference noise was added. Thus, another 199 radar data with noise are obtained, which are also called artificial radar range-Doppler (RD) images.

Real-world measurement

The measurements were recorded in typical inner-city traffic scenarios. All the measurements were done on measurement data by using the FMCW configuration. Therefore, another 199 snapshots are obtained as the real measured radar data.

In the end, 199 original RDMs, 199 artificial RDMs and 199 measurement RDMs were obtained. Generally speaking, the most popular neural network data sets are tens of thousands of image data. At the same time, a large data set is a guarantee of good results. Therefore, we need to perform data augmentation on existing data sets. By vertically flipping the three kinds of image data, we finally get 398 original RDMs, artificial RDMs, and measurement RMDs respectively, and use these images as the total data set. These range-Doppler images were resized to the same fix size of 249x256x1 unsigned 16-bit images.

2.3.2 Software

The softwares which utilized in this thesis are MATLAB and Python. We use MATLAB to generate the training datasets. On the other side, Python is the main programming language for this thesis. In addition, Keras and Tensorflow are implemented to built the neural network model, and matplotlib is also used for graphic visualization.

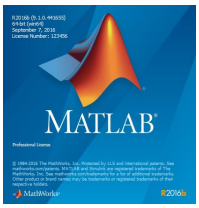

Figure 2.11: MATLAB [28]

Figure 2.12: Keras Tensorflow [29]

MATLAB

MATLAB is an abbreviation of Matrix Lab. It is a powerful, operating language that integrates three basic functions of numerical calculation, symbolic calculation and graphic visualization developed by the American mathworks company. It has a main package of hundreds of internal functions and over thirty toolbox. Among them, this thesis uses the Automated Driving Toolbox in MATLAB [28].

Keras and Tensorflow

Keras is a high-level neural network API, which is written in pure Python and is based on Tensorflow backend. With the help of Keras we can easily built the neural network which we need. Tensorflow is an open source deep learning algorithm framework with code efficiency and clear network architecture, which provides a good learning implementation tool for the research of deep learning algorithms [29].

3 Classic Approach of radar data denoinising

3.1 non-machine learning approach–Gamma Correction

3.1.1 Introduction

In the process of digitizing an image, the color and brightness effects of the image are distorted due to various reasons, so the image must be corrected. Image correction is a series of technologies to improve the visual effect of the image, improve the sharpness of the image or convert the image into a form more suitable for human eye observation and automatic machine analysis. Under normal circumstances, image correction processing will highlight certain useful information of the image and increase the use value of the image. A common image correction processing operation is Gamma Correction.
Generally, gamma has the following three meanings:

- Display Gamma: It is fixed and uncorrectable due to the physical properties of the display. A higher display gamma results in a darker image with greater contrast.
- Image Gamma: This is applied either by the camera or RAW development software whenever a captured image is converted into a standard JPEG or TIFF file.
- System Gamma: Adjust the relationship between input data and output data through parameter control driver.

In display science, photography, video images, and computer graphics, Gamma Correction is an important process to realize that the image reflects the original object or the visual information of the original image as realistically as possible. Gamma represents a data parameter that describes the non-linearity of intensity or brightness reproduction, it involves four disciplines of conceptual physics, perception, photography, and video imaging. For the correct processing of Gamma Correction, the process of reproducing, processing and displaying images can get good results.

3.1.2 Basic Theory of Gamma Correction

Gamma correction was first used to express the non-linear characteristics of Cathode ray tube (CRT) displays [30]. Due to the physical properties of the CRT electron gun, a layer of relationship is imposed between the voltage input and output. This layer of relationship is called five-halves power law. Therefore, in the traditional cathode ray tube (CRT), the image reproduction

is essentially non-linear and the light intensity generated on the screen surface is proportional to the 5/2 power of the input voltage, as shown in the Figure 3.1 below:

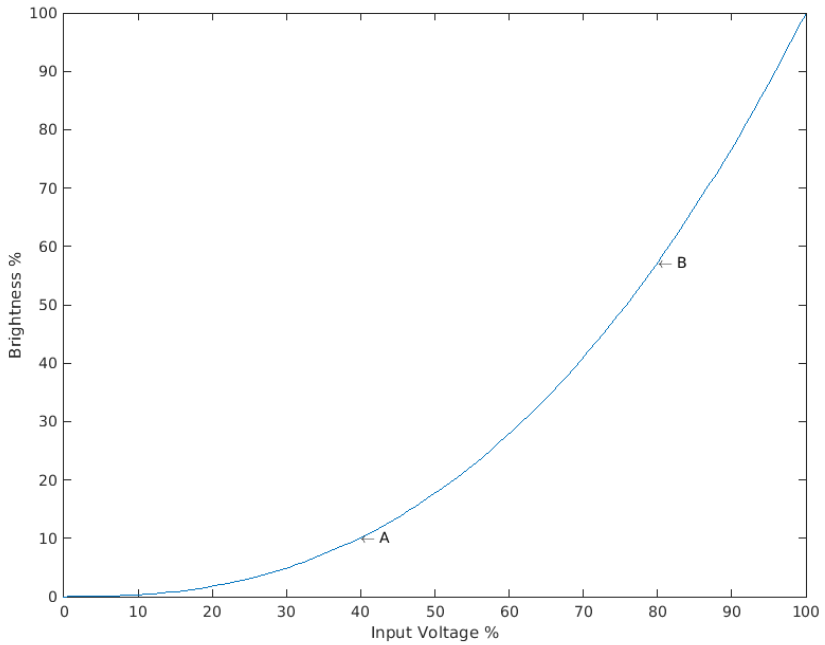

Figure 3.1: CRT Gamma Curve

The exponential value of this power function is defined as γ. Thus, the nonlinear relationship between the input and output can be roughly expressed by the intensity function (3.1):

$$I = (V)^{\gamma} \tag{3.1}$$

Where I is Intensity, V represents the voltage. Considering the human visual response curve, the intensity generally refers to luminance and the voltage refers to the input voltage. The relationship curve between the input voltage and the output brightness in Figure 3.1 is the gamma curve of the CRT display device. When the gamma value is equal to 1, the curve is a straight line at 45 degrees to the coordinate axis. At this time, the input and output are free of distortion. CRT displays have inherent gamma values greater than one. As can be seen from Figure 3.1, if the input is a gray image signal, when the input voltage is doubled, the brightness output is less than half, close to half at point A and close to $2/3$ at point B. Obviously this distortion will cause that the displayed image is dark. If a color signal image is input, in addition to making the displayed image darker, this distortion will also shift the tone of the image. In fact, for CRT, the value of gamma is close to the theoretical value of 2.5, so the input image is darker on the screen than the original image. Due to the display distortion, the displayed image cannot reproduce the input original image well, so this distortion needs to be corrected. From the above, it can be known that for the display, the gamma value is constant and cannot be changed. This non-linearity is pre-compensated by calculating the voltage signal from the light intensity value, which means the input voltage signal is adjusted. This process is called Gamma Correction, which is a compensation process to obtain the correct reproduction light

intensity. The normal image voltage signal is adjusted in the direction opposite to the distortion of the display and then input to the display to offset the distortion of the display. Therefore, the input signal of the display should be compensated according to the curve shown in the Figure 3.2 below, so that the ideal output result can be obtained on the display.

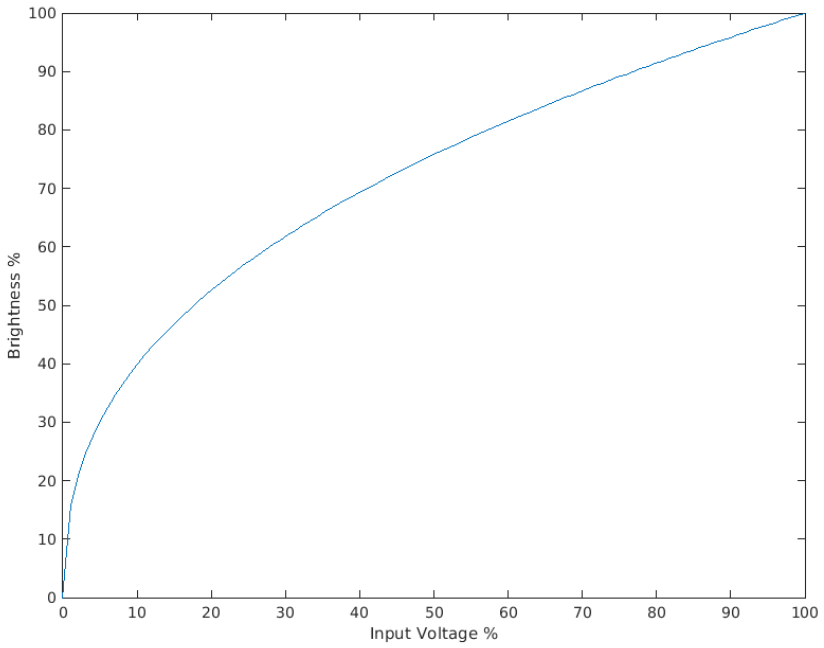

Figure 3.2: Correct Curve of CRT

The Equation (3.2) of the curve is as follows:

$$L = (V + \varepsilon)^{\gamma} \tag{3.2}$$

Where is ε the compensation coefficient, L is the luminance, V is the voltage and γ is the value of the gamma.

Figure 3.3: Image with different Gamma

From the figure 3.3 we can notice that, when the gamma value is bigger than 1, the adjusted image is much darker than the original image; when the gamma value is less than 1, the output is much lighter than the original image.

3.1.3 Range-Doppler image denoising using Gamma Correction

When gamma correction is performed on an image, the noise value of the darker part of the image can be suppressed and the contrast of the image can be improved.
Therefore, in this thesis, we use this traditional method to reduce the noise of range-Doppler images. The flow chart 3.4 of Gamma Correction denoising is as follows:

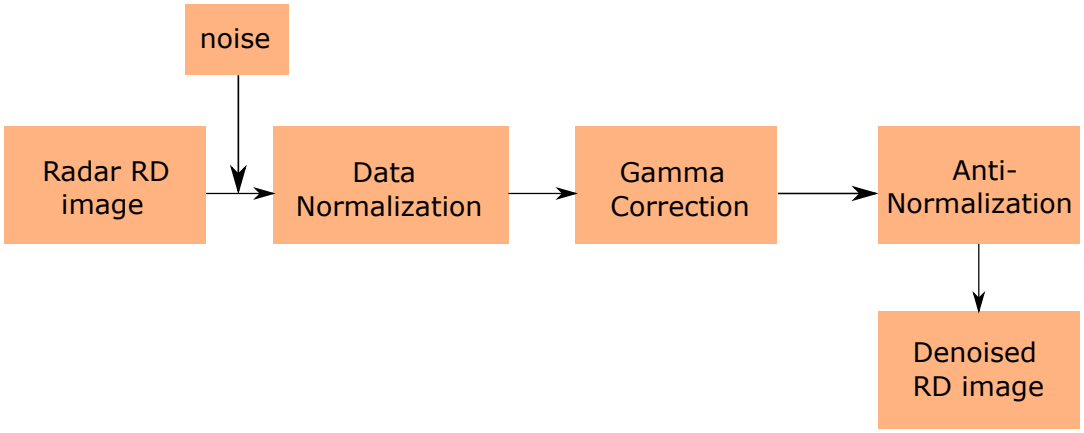

Figure 3.4: Flow Chart of Gamma Correction Denoising

The implementation steps of the Gamma Correction are given:

- Add Gaussians noise on the original Radar Range Doppler maps so that we can get the noisy radar data.
- Normalize the input noisy data.
- Manually select the value of to perform gamma correction on the data to suppress noise.
- De-normalize the processed data to obtain a noise-reduced radar image.
- Repeat this process until the noise reduction effect is the best based on the selected value.

3.2 machine learning approach–PCA and PPCA

3.2.1 Introduction

Principal component analysis (PCA), as a multivariate statistical technique, is also a kind of multidimensional orthogonal linear transformation based on statistical characteristics. It is often used for feature extraction of signals and dimensionality reduction of data [31]. Pearson first proposed the concept of PCA in 1901, then it was developed by scholars such as Hotelling and J.E. Jackson [32]. Later, researchers described the PCA algorithm again in the form of probability theory, which made the PCA method develop further. Nowadays, PCA is widely used in various fields such as chemistry, pattern recognition, image processing, etc. PCA is given different names in different application fields, such as Karhunen-Loeve Transform [33], Hotelling Transform [34] and Subspace Approach [35].

PCA is one of the main methods for feature extraction. The idea is to map n-dimensional features to k-dimension ($k < n$). Then seek the direction of the maximum variance after data transformation, thereby reducing the data dimension and retaining most of the information. Generally, the signal energy is gathered in a small subset of the PCA transformed data set, but the noise energy is evenly distributed over the entire data set. Therefore, signals and noise can be better distinguished in the PCA domain.

PCA is accomplished by linear transformation of variables. It obtains the principal component and selects the direction of the principal component to establish a new coordinate system, and the original sample is projected onto the reconstructed coordinate axis to obtain a new sample with the largest variance. Essentially, the direction with the largest data variance is taken as the main feature of the new sample, so that the data has no correlation in different orthogonal directions. Therefore, PCA has strong anti-interference ability, and can obtain higher signal-to-noise ratio.

Although PCA retains a lot of information, it does not consider the probability distribution of the data, which means that an appropriate probability model is not defined, which makes it inherently limited: No probabilistic model has been established for the observation data; The case of missing data cannot be handled well, which makes the analysis result unsatisfactory; When the data dimension is high dimension, it takes a lot of time and space to calculate the covariance matrix. For these reasons, the application of PCA is limited. Probabilistic principal component analysis (PPCA) was proposed to solve the problem of the limitations of ordinary principal component analysis.

3.2.2 Basic Theory of PCA and PPCA

3.2.2.1 Gaussian Process Latent Variable Model

Probabilistic PCA is generated through the evolution of Gaussian process latent variable models [36]. In this section, the latent variable model is first introduced. Latent variables are usually associated with a set of observed variables. It is not directly observable, so it is necessary to use indirect methods to obtain such comprehensive variables, generally need the help of mathematical models.

A latent variable model is a statistical model that includes one or more unobservable latent variables, which can provide a low-dimensional representation for the data and the dependencies between the data. The purpose of the latent variable model is to describe the relationship between the p-dimensional latent variable vector y and the m-dimensional observed variable vector x, where $p < m$. The latent variable model can be expressed as:

$$x = f(y; w) + \varepsilon \tag{3.3}$$

f is the function of the latent variable y with respect to the regression coefficient w, ε represents noise independent of the latent variables. The parameters of the model can be estimated by the method of parameter estimation, then the distribution of the observed data can be derived according to the above equation. Both factor analysis and probabilistic PCA are derived from this latent variable model.

3.2.2.2 Principal Component Analysis and Factor Analysis

Principal component analysis is a classic statistical method for dimensionality reduction. It extracts information from observable variables and forms the latent variables that cannot be directly observed. It is essentially a dimensionality reduction process for variables, by replacing the original variables with fewer variables. The new variable is a combination of the original variables.
The method of finding the principal components of the m-dimensional observation vector x_i is: First, calculate the covariance matrix S of the m-dimensional observation vector x_i, and μ is the mean of the observation vector:

$$S = \sum_{i=1}^{n} (x_i - \mu) (x_i - \mu)^T / n \tag{3.4}$$

$$\mu = \sum_{i=1}^{n} x_i / n \tag{3.5}$$

Then, eigenvalue decomposition is performed on S, and the first q large eigenvalues are retained to maximize the variance. The eigenvectors corresponding to the eigenvalues are used to form the eigensubspace, which is the main component of the observed data.
The mathematical model of factor analysis is derived from principal component analysis and can be used as a generalized form of principal component analysis. The main idea used is to find common factors among the variables to simplify the relationship between variables. There are many differences between factor analysis and PCA. Firstly, PCA decomposes variance into orthogonal components, while factor analysis divides variance into different factors, and the factors are not necessarily orthogonal. Secondly, PCA is obtained from the data projection. There is no mathematical model. On the other side, factor analysis needs to obtain the common factors of the variables according to the mathematical model. Thirdly, the main work of principal component analysis is to project the observed variables onto the principal components, while factor analysis focuses on transforming from the basic factors to the observed variables.
It can be known from the above analysis that factor analysis uses mathematical models to obtain a low-dimensional factor, which can fully represent the original variables on the premise that the dimension is low. According to the factor analysis model, if its assumptions are changed so that it can be combined with PCA, the probability PCA is obtained.

3.2.2.3 Probability Principal Component Analysis

Proposition of Probability PCA

PCA and Probability PCA are both second-order statistical methods that use only covariance information. The PPCA model is equivalent to PCA's data reconstruction with noise. Since the PCA method does not have a probability model to describe it, in order to define an appropriate probability model for principal component analysis, Tipping et al. proposed a so-called probabilistic PCA based on a Gaussian process latent variable model closely related to factor analysis [37]. In traditional PCA, information outside the main subspace is simply discarded. However, in PPCA, this information will be estimated as Gaussian noise. The column of W in

PPCA is the eigenvector of the covariance matrix of the sample after PCA, which is obtained after scaling and rotation. For probabilistic PCA model parameters, the maximum likelihood function [38] or expectation-maximization iterative algorithm
citemclachlan2007algorithm can be used for estimation. Compared with traditional PCA, the probabilistic PCA defines a proper probability model, so that the probabilistic PCA has the same processing capabilities as PCA, such as dimensionality reduction, at the same time it can directly use its probability model for classification; In addition, the feature space of the PCA algorithm is global linear, instead the feature space of the probabilistic PCA algorithm is modeled in a locally linear manner.
Suppose $X = \{x_1, x_2, \dots, x_n\} \in R^{mxn}$ represents all observation data, where m is the dimension of the sample and n is the number of samples. The generated model can be represented as follows:

$$X = WY + \mu + \epsilon \tag{3.6}$$

Where W is the loadings matrix, the dimension is mxn. In addition, $q < m$ is the numbers of the principal components. In general, $Y \sim N(0, I)$ are latent variables and the parameter vector μ follows the non-zero mean distribution. Special attention should be paid to noise. In the model, noise ϵ is assumed to obey the Gaussian distribution of the mean value of 0 and covariance of $\sigma^2 I$. σ^2 represents noise variable parameter and I is the variance. Thus, the probability distribution of the latent variables Y is given by:

$$P(y) = (2\pi)^{-q/2} \exp\left\{-\frac{1}{2\sigma^2} y^T y\right\} \tag{3.7}$$

This can get the probability distribution of x in space when y is given.

$$P(x|y) = \left(2\pi\sigma^2\right)^{-m/2} \exp\left\{-\frac{1}{2\sigma^2}\|x - Wy - \mu\|^2\right\} \tag{3.8}$$

According to formula $P(x) = \int P(x|y)P(y)dx$, the probability distribution of the observed data $P(x)$ can be calculated as:

$$P(x) = (2\pi)^{-m/2}|C|^{-1/2} exp\left\{-(x-\mu)^T C^{-1}(x-\mu)/2\right\} \tag{3.9}$$

Where $C = \sigma^2 I + WW^T$, the observation covariance model. For the given observed variable x, we can derive the posterior probability distribution $P(y|x)$ of the latent variable:

$$\begin{aligned} P(y|x) =& (2\pi)^{-q/2}|C|^{-1/2} \exp\left(-\left[\left(y - M^{-1}W(x-\mu)^T\right)\right]\left(\sigma^2 M\right)\left[y - M^{-1}W(x)\right.\right. \\ & \left.\left. - \mu)\right]/2\right\} \end{aligned} \tag{3.10}$$

Where $M = \sigma^2 I + W^T W$. It can be inferred from the above equation (3.10) that the conditional probability distribution of the latent variables under the observed variables is:

$$y|x \sim N\left(M^{-1}M^T(x-\mu), \sigma^2 M^{-1}\right) \tag{3.11}$$

Parameter Estimation

From the probability distribution of the observed data, we can get the log function as:

$$L = \sum_{i=0}^{n} \ln\left\{P\left(x_i\right)\right\} = -\frac{n}{2}\left\{m\ln(2\pi) + \ln|C| + tr\left(C^{-1}S\right)\right\} \tag{3.12}$$

The main parameters are noise variable parameter σ and loadings matrix W.

Parameter estimation using maximum likelihood

$$\frac{\partial L}{\partial W} = n\left(C^{-1}SC^{-1}W - C^{-1}W\right) = 0 \tag{3.13}$$

$$SC^{-1}W = W \tag{3.14}$$

Singular value decomposition of the above formula, we can get:

$$W = U_m\left(K_m - \sigma^2 I\right)^{1/2} R \tag{3.15}$$

Where U_m is a matrix, its m columns are the eigenvectors of the covariance matrix$S$, K_m is a diagonal Matrix and R is the orthogonal matrix.
Bring Equation (3.15) into the log function (3.12):

$$L = -\frac{n}{2} * \left\{ mln(2\pi) + \sum_{i=1}^{p} \ln\left(\lambda_i\right) + \frac{1}{\sigma^2}\sum_{j=p+1}^{m} \lambda_j + (m-p)\ \ln\left(\sigma^2\right) + m \right\} \tag{3.16}$$

Calculate partial derivatives for the above Equation (3.16).

$$\sigma^2 = \frac{1}{m-p}\sum_{j=p+1}^{m} \lambda_i \tag{3.17}$$

Where λ_i are the eigenvalues, p is the number of non-zero.

Use EM to estimate parameters

In statistics, an expectation-maximization (EM) algorithm is an iterative method to find maximum likelihood or maximum a posteriori estimates of parameters in statistical models. In the probabilistic PCA's EM algorithm, latent variables can be considered as incomplete data. If the value of the implicit variable is known, the parameters can be directly estimated by using the Least square error for Equation (3.6). However, in practice, for a given vector, we often do not know its corresponding latent variable, but know the joint distribution of the observed variable and the latent variable, so we can calculate the expected value of the log function of the complete data. In step E of the EM algorithm, calculate the log likelihood function expectation of the complete data. In step M, the parameters to be estimated are obtained by maximizing the expectation of the log function of the complete data.
The log function for complete data is:

$$L_{EM} = \sum_{i=1}^{n} \ln\left\{p\left(x_i, y_i\right)\right\} = \sum_{i=1}^{n} \ln p\left(x_i|y_i\right) p\left(y_i\right) \tag{3.18}$$

$$p\left(x_i, y_i\right) = \left(2\pi\sigma^2\right)^{-2/p} \exp\left\{-\frac{\|x_i - Wy_i - \mu\|}{2\sigma^2}\right\} * (2\pi)^{-2/m} exp\left\{-y_i^T y_i\right\} \tag{3.19}$$

In step E, use Equation (3.10) to get the expectation of the above formula:

$$\begin{aligned} L = -\sum_{i=1}^{n}\Bigg\{ \frac{p}{2}\ln\left(\sigma^2\right) + \frac{1}{2}\operatorname{tr}\left(\left\langle yy^T\right\rangle\right) + \frac{1}{2\sigma^2}\|x-\mu\|^2 - \frac{1}{\sigma^2}(y)^T W^T(x-\mu) \\ + \frac{1}{2\sigma^2}\operatorname{tr}\left(W^T W\left\langle yy^T\right\rangle\right)\Bigg\} \end{aligned} \tag{3.20}$$

The posterior mean $\langle y \rangle$ is as follows:

$$\langle y \rangle = M^{-1} W^T (x - \mu) \tag{3.21}$$

$$\left\langle y y^T \right\rangle = \sigma^2 M^{-1} + \langle y \rangle \langle y \rangle^T \tag{3.22}$$

In step M, find the partial derivative of L with respect to the parameters W and σ^2 by Equation (3.20), and make it 0 to obtain the new parameter $\bar{W}$ and $\overline{\sigma^2}$.

$$\bar{W} = \left[\sum_{i=1}^{n} (x_i - \mu) \left\langle y_i^T \right\rangle \right] \left[\sum_{i=1}^{n} \langle y_i y_i \rangle \right]^{-1} \tag{3.23}$$

$$\overline{\sigma^2} = \frac{1}{np} \sum_{i=1}^{n} \left\{ \|x_i - \mu\|^2 - 2 \left\langle y_i^T \right\rangle \bar{W} (x_i - \mu) + tr \left(\left\langle y_i y_i^T \right\rangle \overline{W^T W} \right) \right\} \tag{3.24}$$

Use the above four formulas to iterate until the algorithm converges. If formulas (3.21) and (3.22) are substituted into (3.23) and (3.24), then the E and M steps in the EM algorithm are combined to obtain the parameter model:

$$\widetilde{W} = TW \left(\sigma^2 I + M^{-1} W^T T W \right)^{-1} \tag{3.25}$$

$$\widetilde{\sigma^2} = \frac{1}{M} \operatorname{tr} \left(T - TWM^{-1} \widetilde{W} \right)^{-1} \tag{3.26}$$

It can be seen from the above two formulas that the PPCA EM algorithm is only related to the covariance matrix of the observed data. Where $T = \sum_{i=1}^{n} x_i x_i^T / n$ is the covariance matrix, $tr(.)$ represents the trace of the matrix. Repeat the iterative formula until convergence, so that the parameters σ^2 and W in the PPCA model are obtained [37].

Application of Probability PCA

As a generalized extension of principal component analysis, the main purpose of probabilistic PCA is to define an appropriate probability model for PCA. In traditional PCA, information outside the subspace is simply discarded, whereas in probabilistic PCA, this information is estimated as Gaussian noise. The main application of probabilistic PCA is to generate specific mixed probabilistic models and deal with related problems in missing data. Recently, typical applications are using PPCA in the detection process, traffic flow calculation and analysis of 3D data with missing data. For noisy images, it can be regarded as a matrix processing with missing data, so that the problem of image denoising can be solved with probability PCA.

3.2.3 Range-Doppler image denoising using PCA and PPCA

The principles of PCA probability and PCA are given in the previous sections 3.2.2, and the model is analyzed to derive the estimation method of model parameters. In this section, PCA and probabilistic PCA will be used for range-Doppler image denoising. In order to understand the entire algorithm flow clearly and intuitively, the flow chart 3.5 is as follows:
The implementation steps of PCA denoising are as follows:

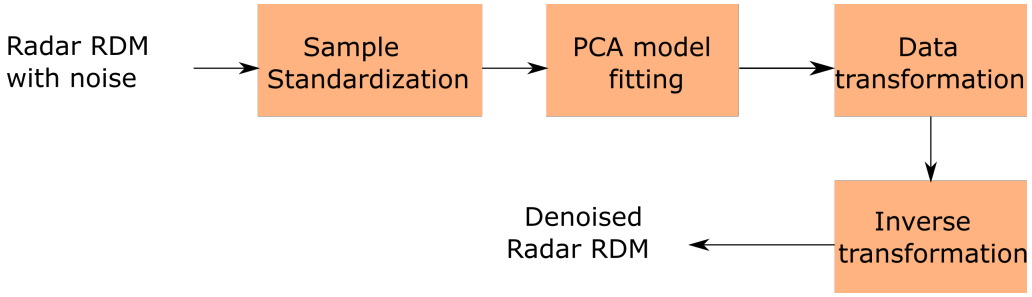

Figure 3.5: PCA flow chart

- Standardize the input RD image which covered by noise to make the data zero mean and unit variance.
- Fit the data to the PCA model. The number of the components of the latent subspace is determined in this step. The number of latent variables is determined by the sum of the observed variance of the data.
- Transform the data to the latent subspace to get the latent variable.
- Transform the latent variables back to the original subspace to get the reconstructed RD image which is denoised.

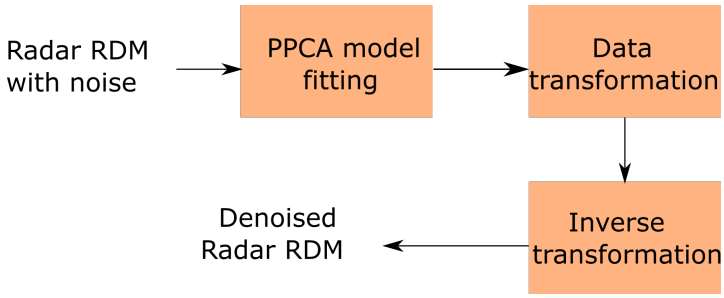

Figure 3.6: PPCA flow chart

Compared to PCA, the denoising step by PPCA is similar to that by PCA. The standardization of the input data is not necessary to perform PPCA due to the probability model. In addition to determining the number of principal components, the PPCA model also needs to calculate the parameters W and noise σ^2 by the EM algorithm.

4 Deep Learning Approach of radar data denoinising

4.1 Radar data denoinising using Autoencoder

4.1.1 Introduction of Autoencoder

Autoencoders (AE) are an unsupervised learning technique in the field of machine learning, in which we use neural networks to represent the set of the data [39]. Actually, autoencoders are a specific type of feedforward neural networks that learn to copy its inputs to its outputs, which means the outputs are same as the inputs. They have an internal hidden layer that can describe a code, also called encoded data, which is used to present the inputs. The code is similar to the compact "compression" of the inputs, and also referred to as a latent-space representation. The inputs are compressed into the lower-dimensional code and then the outputs are reconstructed from this presentation.

An Autoencoder consists of three main components: encoder, code(the encoded data in latent space) and decoder. The basic architecture is shown in the Figure 4.1 as follows:

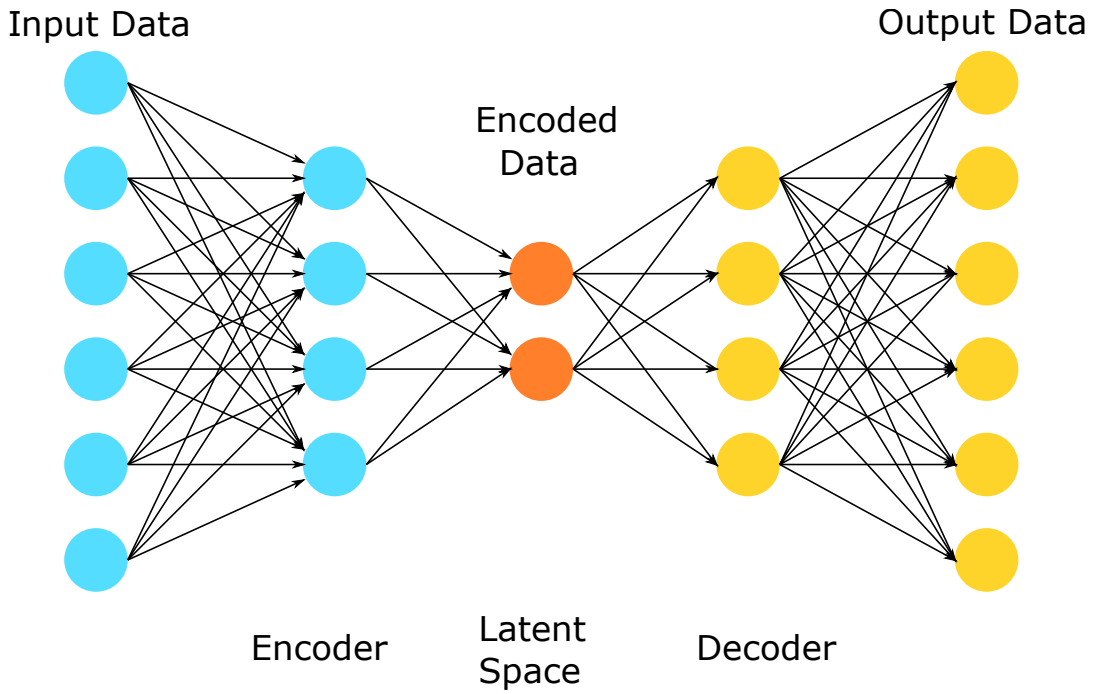

Figure 4.1: Architecture of Autoencoder

This is the simplest architecture of an AE, which is a feedforward, non-recurrent neural network. It has an input layer , an output layer and one or one more hidden layers to connect them, where the output layer has the same number of neurons as the input layers, and with the aim of the reconstructing its inputs. The output of the autoencoder will not be exactly the same as the input, it will be a close but degraded representation. So AE is considered as an unsupervised learning technique since they don't need explicit labels to train on. But to be more precise it is self-supervised because they generate their own labels from the training data.

4.1.2 Principle of Autoencoder

AE is closely to related to principal component analysis PCA . In fact, if the activation function which used within the AE is linear within each layer, the latent variables present at the smallest layer in the network (code) directly correspond to the principal components from PCA. Generally, the activation function used in AE is non-linear, typical activation functions are ReLu(Rectified Linear Unit) and sigmoid.
The AE encodes the input to obtain the new features, and hopes that the original input can be reconstructed from the new features. Essentially, we spilt the network into two segments, the encoder and the decoder, which can be defined as transition Φ and ψ.

$$\phi : X \to F \tag{4.1}$$

$$\psi : F \to X \tag{4.2}$$

$$\phi, \psi = \underset{\phi,\psi}{\arg\min} \|X - (\psi \circ \phi)X\|^2 \tag{4.3}$$

The encoder function, which is denoted by Φ, maps the original input X, to a latent space F, which is presented at the code. The decoder function, which is denoted by ψ, maps the latent space F at the code to the output. Therefore, we are basically trying to recreate the original image after some generalized non-linear compression. The encoding network can be represented by the standard neural network function which is passed through an activation function, where h is the latent dimension.

$$h = \sigma(Wx + b) \tag{4.4}$$

Here, σ is an element-wise activation function such as a sigmoid function or a ReLu function. W is a weight matrix and b is a bias vector. Weights and biases are usually initialized randomly, and then updated iteratively during training. Similarly, the decoding network can be represented in the same way, but with different weight, bias, and the activation functions will be used as the same time.

$$x' = \sigma'\left(Wh + b'\right) \tag{4.5}$$

AE is trained to minimize reconstruction errors, which is usually referred to as the "loss". The loss function can be written in terms of these network functions:

$$L(x, x) = \|x - x'\|^2 = \|x - \sigma'\left(W\sigma(Wx + b) + b'\right)\|^2 \tag{4.6}$$

Thus, the aim of the AE is to select encoder and decoder functions in such a way that we require the minimal information to encode the image such that it can be regenerated on the other side.

4.1.3 Implementation of Denoising Autoencoder

There are several other types of autoencoders. One of the most commonly autoencoders which is used in image denoising area is the denoising autoencoder (DAE). In fact, DAE is a stochastic extension of the basic AE, which try to achieve a good representation by changing the reconstruction criterior. This method is to force the AE to learn useful features, which is adding random noise to its inputs and making it recover the original noise-free data. Thus, the objective of DAE is that of cleaning the corrupted input which means denoising. The basic principle of a denoising autoencoder is shown in the Figure 4.2.

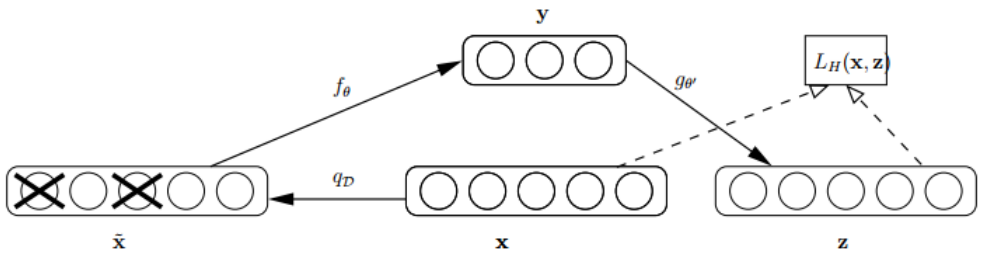

Figure 4.2: Denoising Autoencoder [40]

Some of the inputs are randomly setting to missing by the stochastic corruption process. Denoising autoencoder is forced to predict the corrupted values for randomly selected subsets of the missing pattern [40]. The training process of a DAE works as follow:

- The initial inputs x is corrupted into $\tilde{x}$ through stochastic mapping $\tilde{x} \sim q_D(\tilde{x}|x)$.
- The corrupted inputs $\tilde{x}$ is then mapped to a hidden representation with the same process of the standard AE.
$$h = f_\theta(\tilde{x}) = \sigma(W\tilde{x} + b) \tag{4.7}$$
- From the hidden representation, the model reconstructs $z = g_{\theta'}(h)$.

The model's parameters θ and θ' are trained to minimize the average reconstruction error over the training data and minimize the difference between z and the original uncorrupted input x [40].

Architecture of DAE model

In this thesis, the architecture of DAE model implemented is shown in Figure 4.3. The encoder part consists of several convolutional layers. On the other side, the decoder is composed of the same numbers deconvolutional layers as encoder part. Therefore, the DAE model we implemented can also be called convolutional DAE. The input image is the clean radar range-Doppler image which is covered by artificial noise. The output image is the reconstructed RD image which is denoised. Table 4.1 shows the detailed parameters of DAE model built in this thesis, where Conv represents convolution layer, ConT represents deconvolution layer, Flatten represents flatten layers and Dense represents Dense layers.

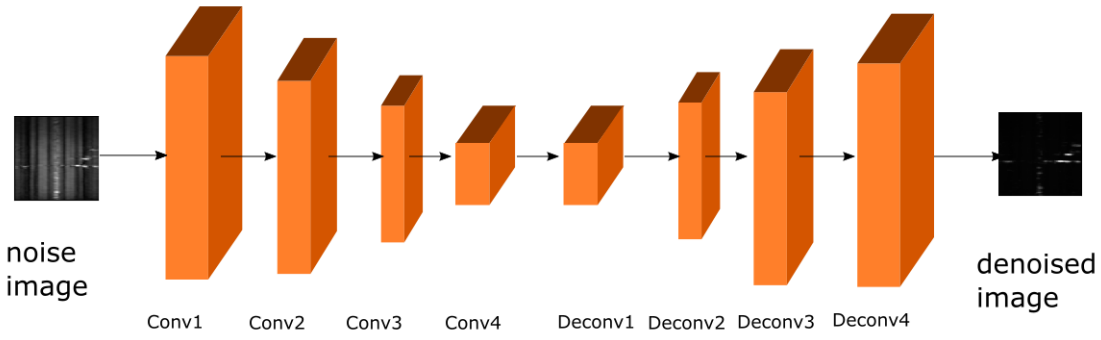

Figure 4.3: Architecture of DAE implemented

Table 4.1: specific parameters of DAE

	layer	Kernel size	stride	padding	Activation function	Output size
Input	input					256x256x1
encoder	Cov1	3x3	2	same	Relu	128x128x64
	Cov2	3x3	2	same	Relu	64x64x64
	Cov3	3x3	2	same	Relu	32x32x32
	Cov4	3x3	2	same	Relu	16x16x32
	Cov5	3x3	2	same	Relu	8x8x32
	Flatten					2048
	Dense					512
decoder	Dense					2048
	ConvT1	3x3	2	same	Relu	16x16x64
	ConvT2	3x3	2	same	Relu	32x32x64
	ConvT3	3x3	2	same	Relu	64x64x32
	ConvT4	3x3	2	same	Relu	128x128x32
	ConvT5/output	3x3	2	same	Relu	256x256x1

Loss

The loss function used is made of two parts, the Cross-entropy loss and the structural similarity (SSIM) loss.

Cross-entropy loss

$$L_c(y, y') = -(y \log(y') + (1 - y) \log(1 - y') \tag{4.8}$$

We calculate the cross-entropy between the output and the ground truth.

SSIM loss

The structural similarity index is a method for predicting the perceived quality of digital television and cinematic pictures, as well as other kinds of digital images and videos. In image processing area, SSIM is always used for measuring the similarity between two images. It also can be used as a loss function instead of L1 loss. SSIM can be defined as :

$$\text{SSIM}(y, y') = \left[l(y, y')^{\alpha} \cdot c(y, y')^{\beta} \cdot s(y, y')^{\gamma} \right] \tag{4.9}$$

where y is the input image, y' is the output image, $l(.)$ represents the luminance, $c(.)$ represents the contrast, $s(.)$ represents the structure. α, β and γ is the weight, which is chosen to be 1 here. Thus, the loss function can be then written as follows:

$$L^{SSIM}(y,y') = 1 - SSIM(y,y') \tag{4.10}$$

Finally, the overall network loss is the sum of all mentioned losses and can be expressed as follow:

$$L = L_c + L^{SSIM} \tag{4.11}$$

4.2 Radar data denoinising using GANs

4.2.1 Basic Theory of GANs

GANs learn the probability distribution of real data implicitly. Assume that the parameter of the generator G is θ, the parameter of the discriminator D is $\emptyset$, the probability distribution of the random variable z is q_z. The data generated by the generator is $G_\theta(z)$, the probability distribution is p_G. The real data is x, and the probability distribution is p_x. $D_\emptyset(x)$ represents the output of the discriminator to the real data, and $D_\emptyset\left(G_\theta(z)\right)$ represents the output of the discriminator to the generated data. The discriminator is optimized by maximizing the correct classification probability of the real data x and the generated data $G_\theta(z)$. On the other side, the generator is optimized by minimizing the probability that the generated data is false: The optimization problem of G_θ and $D_\emptyset$ is a mini-max problem, the mathematical formula of the loss function $V(D_\emptyset, G_\theta)$ is as follows:

$$\min_{\theta} \max_{\emptyset} V\left(D_\emptyset, G_\theta\right) = E_{x \sim p_x}\left[\log\left(D_\emptyset(x)\right)\right] + E_{z \sim q_z}\left[\log\left(\left(1 - D_\emptyset\left(G_\theta(z)\right)\right)\right)\right] \tag{4.12}$$

During the training process of the network, the generator G_θ and the discriminator $D_\emptyset$ are trained alternately, which means the other network is trained while fixing one network parameter. When the generator G_θ is fixed, the loss function of the $D_\emptyset$ network needs to be maximized, so that the weight parameters of the $D_\emptyset$ network can be updated. At this time, the $D_\emptyset$ network has only two tasks:

- to judge the real data x as true;
- to judge the fake data $G_\theta(z)$ as false.

Thus, for discriminator $D_\emptyset$, the loss function can be written as:

$$\max_{\emptyset} V\left(D_\emptyset, G_\theta\right) = E_{x\sim p_x}\left[\log\left(D_\emptyset(x)\right)\right] + E_{z\sim q_z}\left[\log\left(\left(1 - D_\emptyset\left(G_\theta(z)\right)\right)\right)\right] \tag{4.13}$$

When the discriminator $D_\emptyset$ is fixed, it is necessary to minimize the loss function of the G_θ network, so that the G_θ network can generate fake data to fool the $D_\emptyset$ network. Therefore, for the G_θ network, the loss function can be written as:

$$\min_{\theta} V\left(D_\emptyset, G_\theta\right) = E_{z\sim q_z}\left[\log\left(\left(1 - D_\emptyset\left(G_\theta(z)\right)\right)\right)\right] \tag{4.14}$$

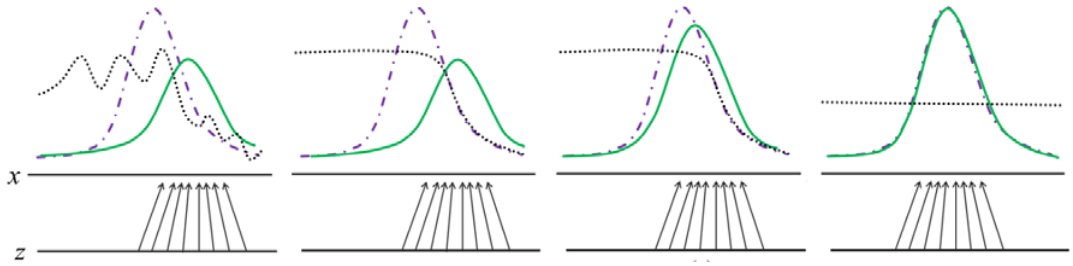

Figure 4.4: Explanation of GAN learning process [10]

Figure 4.4 explains the learning process of GAN intuitively and describes the process of generator at learning data distribution informally. In the figure 4.4, the dotted purple line indicates the training data distribution p_x, the green curve represents the generated data probability distribution p_G, the black dotted line represents the output distribution of the discriminator $D_\emptyset$. The lower black horizontal line represents the domain space of the random variable z samples, the higher black horizontal line represents the domain space of the data x. The upward clipping indicates that the generator G_θ maps the random variable z to the domain space of x, thereby generating the data distribution p_G. G_θ shrinks in the high-density region of p_G but expands in the low-density region.The first Figure shows that the opposing sides near converge, p_G and p_x partially overlap, the classification of is partially accurate. The second Figure represents that the discriminator $D_\emptyset$ is optimized until it converges to the optimal. The third Figure shows that after updating G_θ, the gradient of $D_\emptyset$ has guided $G_\theta(z)$ to approach the area that is more likely to be classified as data. The fourth Figure represents that after training, if the discriminator and generator have sufficient capabilities, the optimization will stop at $p_x = p_G$. At this time, the discriminator cannot distinguish between these two distributions [10].
GANs have comprehensive theoretical support. In the literature, Goodfellow et al. proved the existence of Nash equilibrium solution of GANs [10]. According to the theory, when it reaches

the Nash equilibrium state, it should have very powerful data generation capabilities, which can completely simulate the distribution of input data. However, in practical applications, when using the gradient descent method to solve the objective function, it will be found that GANs are very difficult to train and may not necessarily find the Nash equilibrium solution. The trainer needs to carefully balance the training levels of the generator and the discriminator. For example, the discriminator is updated k times in one iteration, but the generator is updated only once. Otherwise, problems such as non-convergence of the model, unstable training, mode collapse, and gradients vanishing will occur. Otherwise, problems such as non-convergence of the model, unstable training, mode collapse, and disappearance of gradients will occur. In addition, the loss functions of the generator and discriminator do not have a clear physical meaning. We cannot determine the training state of the model through the loss function, but can only determine whether the model is optimizing in the correct direction by constantly reviewing the generated samples.

4.2.2 Improvement of GAN Model

conditional generative adversarial network

The reason why GANs are very difficult to train is that it learns a specific probability distri-

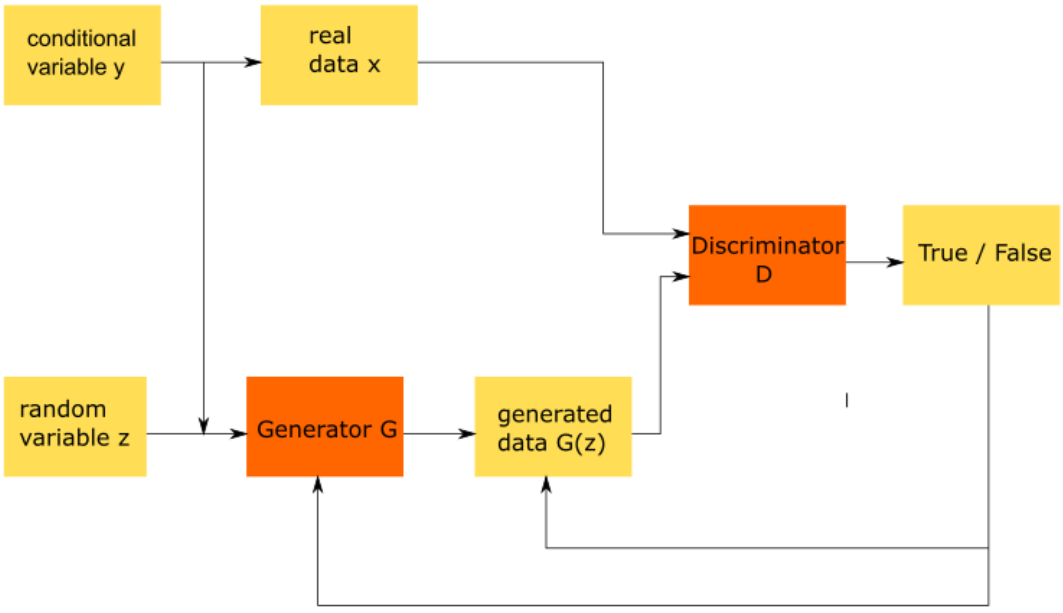

Figure 4.5: The basic architecture of conditional GANs

bution from a large, complex and unlabeled data set, then expresses it. This process is very slow and unconstrained. There is a certain deviation between the machine's understanding of data and human's understanding, which will also cause the final result to be different from expected. The simplest and effective improvement is to add constraints to the training process

of the adversarial network to guide the network training direction. Therefore, Mirza Mehdi al. proposed a conditional generative adversarial network (cGAN) [41]. In cGAN algorithm, while the random variable z and the real data x are input, the conditional variable y is also input into the network. The network structure is shown in the figure. The input condition y will constrain the model training and guide the generation of data. The form of condition y is various. It can be the category label of the data. In this way, unsupervised learning can be transformed into supervised learning. It can also be the corresponding target image, so that the network can learn with targets. This improvement is relatively simple but very effective and is widely used in subsequent research. Figure 4.5 shows the basic architecture of the cGANs.
Therefore, if y is the constraint given to the network, then the mathematical model of cGAN can be expressed as:

$$\min_{\theta}\max_{\emptyset} V\left(D_{\emptyset}, G_{\theta}\right) = E_{x\sim p_x}\left[\log\left(D_{\emptyset}(x|y)\right)\right] + E_{z\sim q_z}\left[\log\left(\left(1 - D_{\emptyset}\left(G_{\theta}(z|y)\right)\right)\right]\right. \tag{4.15}$$

Compared with Equation (4.12), the generator and discriminator of Equation (4.15) have an additional condition y when input, and the others are exactly the same.

Deep convolution generative adversarial network

Radford et al. proposed a deep convolutional generative adversarial network in 2016, which combines a CNN with a generative adversarial network [42]. Due to the powerful performance of the CNN, Radford built both the generator and the discriminator with the topology of the CNN. By setting a series of restrictions in the topology of the CNN, the entire network can be trained stably. Deep convolutional generative adversarial networks were originally applied in the field of image generation. The pooling layer in CNNs will ignore a lot of information, which will reduce the quality of the generated images. Therefore, stride convolution and micro-stride convolution are used instead of the pooling layer in the deep convolution generative adversarial network. Stride convolution and micro-step convolution can retain most of the information in the image and ensure the quality of the generated image. In addition, the author also introduced a batch normalization operation to solve the problem of gradient disappearance during training. At the same time, the fully connected layer of the CNN is removed, and the activation function is replaced. The ReLu activation function is used in the generator, while the Leaky ReLu activation function is used in the discriminator. Therefore, the deep convolution generating adversarial network has good performance and can produce realistic images.

4.2.3 cGAN based denoising method for radar data

This section will focus on the implementation of cGAN to denoise from the aspects of network model training and use, network model design, and loss function design.

4.2.3.1 Model training and usage process

The cGAN-based radar data denoising method in this thesis is to train a large number of images so that the model can obtain the ability to process high-noise radar images. The network model in this thesis is divided into a generative model and a discriminative model. The generative

model is used to process the noise radar image, and the discriminative model is used to discriminate the true and false of the processed image. We use two CNN models with different structures as the generative model and discriminative model.

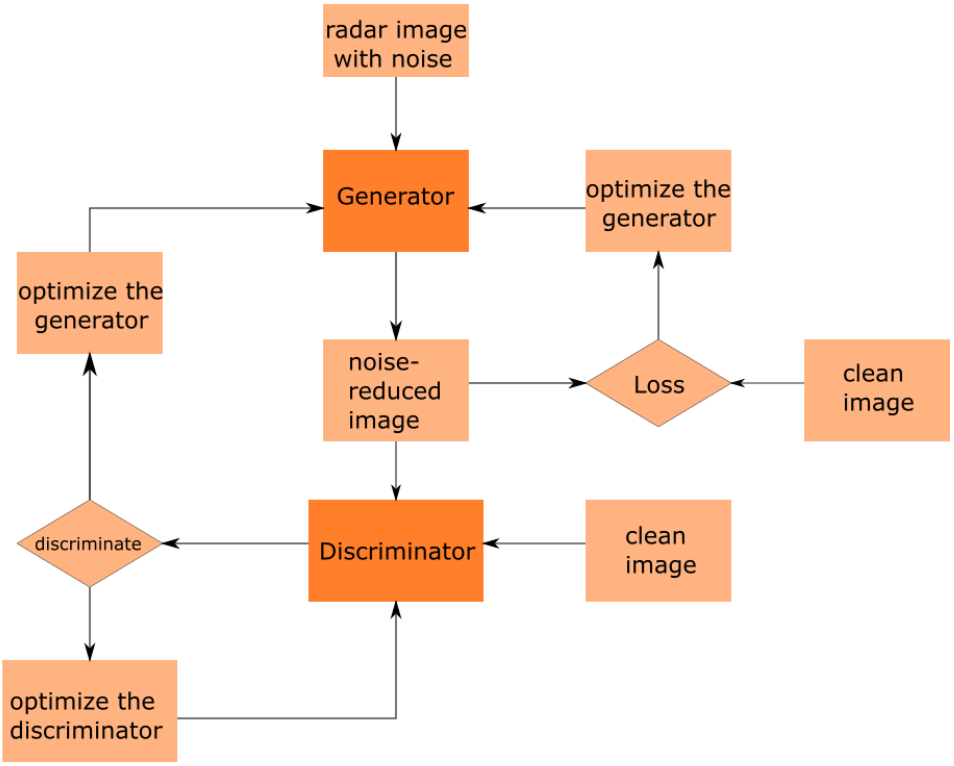

Figure 4.6: training process of cGAN model

The entire training process is shown in the figure 4.6. The figure integrates the optimization process of the generative model and discriminative model. The training process is as follows: First select a group of images, including a noisy artificial RD image and its corresponding clean RD image. During training, the noisy RD image is input to the generative model, and a generated denoised image is output after a series of transformation operation in the model. Then, the gap between the generated image and the corresponding clean RD image is calculated according to the loss function, so that the optimizer will adjust the parameters of the generative model. At the same time, the generated noise-reduced image and its corresponding clean RD image will be input to the discriminative model. The discriminative model extracts image features, determines the authenticity from the feature differences between the real image and the noise-reduced image, then feeds back to the generative model, so that the generative model is further improved. The noise-reduced image generated by the improved generative model is more realistic and less noisy than the previous one, so that the discriminative model can further adjust the discrimination ability. This process is repeated, the generative model and the discriminative model confront each other, and finally the model reaches a balance and completes the training. In the process of using the model for denoising, the discriminative model does not participate in the processing. only the noisy artificial RD image is input into the generative model, and the

corresponding clean image is not provided. The generator processes the noisy image end-to-end and outputs the noise-reduced image. Due to the supervision of the discriminative model during the training process and the adversarial relationship between the two sets of models, the cGAN method is easier to adjust the model parameters to the best than the method using only a set of CNN model training processes, thereby obtaining better processing results.

4.2.3.2 Loss Function Design

The loss function determines the learning goal of the deep network. Reasonably designing the loss function so that the two models can learn what they should learn is the core of cGANs. An overview of the proposed cGAN architecture is depicted in Figure 4.7. In this model, the loss function of the entire network is divided into two parts: adversarial loss and non-adversarial loss. The two types of loss functions perform their respective functions and guide the generator to produce clear and realistic images.

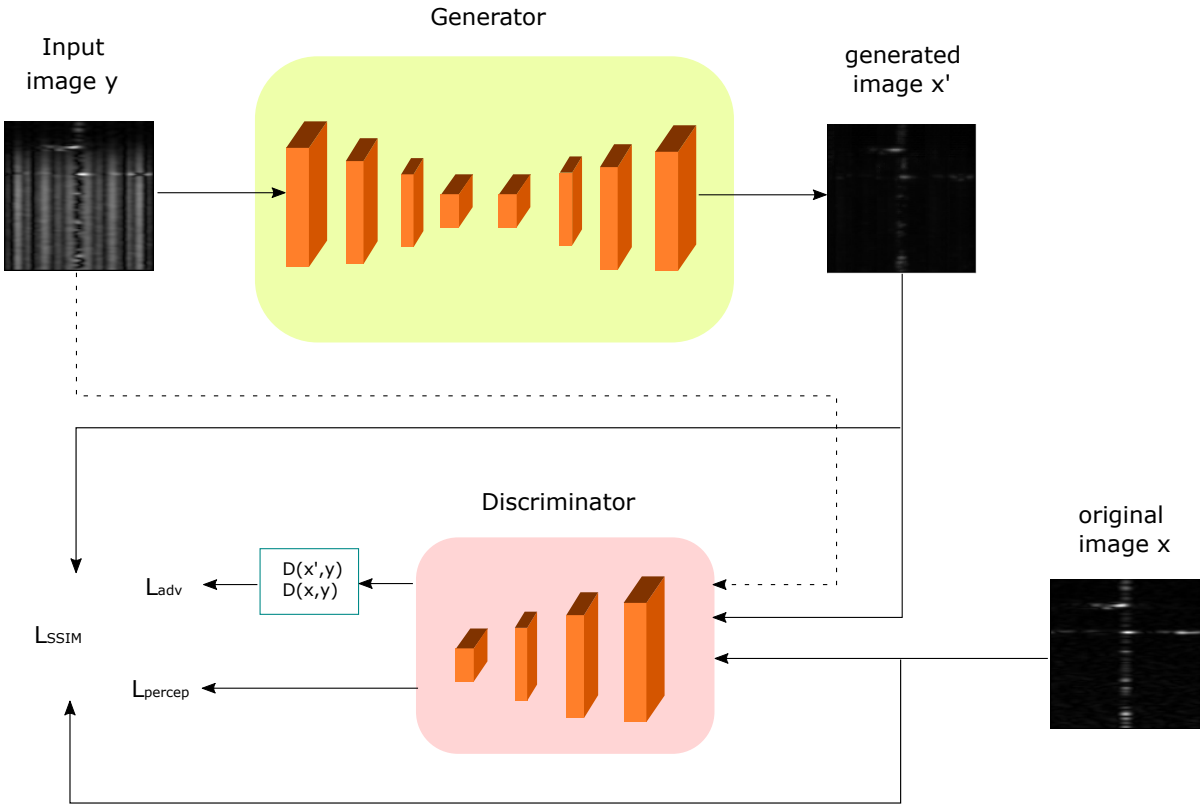

Figure 4.7: cGAN model implemented

adversarial loss

Adversarial loss is one of the loss functions of these two neural network models. The original intention of using GAN was to pursue more realistic image generation and denoising effects than CNNs. Specifically, for a generator, its role is to make the generated image as real and clear as possible; for a discriminator, it is to try to determine that the generated image is false.

This effect is achieved by adversarial losses. In the short years of development of generating adversarial networks, various forms of adversarial loss functions have emerged. Among the most representative loss functions are cross entropy loss function, square loss function [43], Wasserstein distance loss function [44] and so on. This thesis uses the most common cross-entropy loss function as the adversarial loss function. In this case, y is the input radar range-Doppler image with noise. The generator is trained to map y to the generated noise-reduced RD image $x' = G(y)$. The generated image x and the corresponding paired clean RD image x' are then passed through the discriminator as the inputs. The discriminator is trained as a binary classifier, such like $D(x',y) = 0$ and $D(x,y) = 1$, to distinguish which image is generated noise-reduced image and which image is the real image [45]. Thus, the network adversarial loss is expressed as follows:

$$\min_G \max_D L_{adv} = E_{x,y}[\log(D(x,y))] + E_{x,y}[(1 - \log\mathrm{D}(x',y))] \tag{4.16}$$

non-adversarial loss

Using only adversarial losses may produce a RD image that do not match the requirement. Therefore, in order to solve this problem, non-adversarial losses need to be added as additional losses to constrain the generated image content to ensure that the accurate and realistic noise-reduced images are obtained. Compared to adversarial loss, its purpose is to guide the generator to generate images with appropriate content.
The non-adversarial losses introduced generally calculates the loss value of the sample compared with the original sample after forward or backward propagation. It is the key to affecting the quality of image generation, so it should be chosen carefully. A good loss function should be close to the visual characteristics of the human eye, not sensitive to noise, but sensitive to the changes in structure, local details, lighting and color.

SSIM loss

The first non-adversarial loss introduced in this thesis is also the SSIM loss which already implemented in DAE part. SSIM loss compares the local area of the target value between the reconstructed image and the original image. When testing the real measurement RD image, there is no clean image matching it, so it is more appropriate to choose SSIM loss.

$$L_{SSIM}(x,x') = 1 - SSIM(x,x') \tag{4.17}$$

perceptual loss

Another non-adversarial loss that is utilized is the perceptual loss. Compared with SSIM loss, it has a better visual perception ability closer to the human eye, which can make the generated image more natural and realistic. In the other words, perceptual loss can minimize the perceptual difference between the generated image x' and the corresponding clean image x. The perceptual loss was originally proposed by Johnson et al. [23], and obtained surprising results on many tasks such as neural style transfer [46], image super-resolution reconstruction [47], and image deblurring [48].
Usually calculating the perceptual loss requires an additional neural network–a pre-trained VGG network (Visual Geometry Group Network) [49] on the ImageNet dataset [50]. The

specific calculation steps are as follows: input two test images into a pre-trained VGG network, then extract the feature map output from one or several convolutional layers from the VGG, finally calculate the L1 distance on the feature maps corresponding to the two images which are tested respectively. Instead in this thesis, we calculate the perceptual loss directly in the discriminative model, which means we capture the output features of the generated image x' and corresponding ground truth x' on each intermediate convolutional layer of the discriminator. Thus, the mathematical formula of the perceptual loss can be expressed as follows:

$$W_i\left(x, x'\right) = \left\| D_i(x, y) - D_i\left(x', y\right) \right\|_1 \tag{4.18}$$

Where W_i is the intermediate perceptual loss between the features which are extracted separately from the generated image and corresponding clean image in the layer i of the discriminator. Thus, the total perceptual loss can be expressed as follows:

$$L_{perc} = \sum_{i=0}^{L} W_i\left(x, x'\right) \tag{4.19}$$

where L is the total number of the hidden layers in the discriminator [45].

Total loss

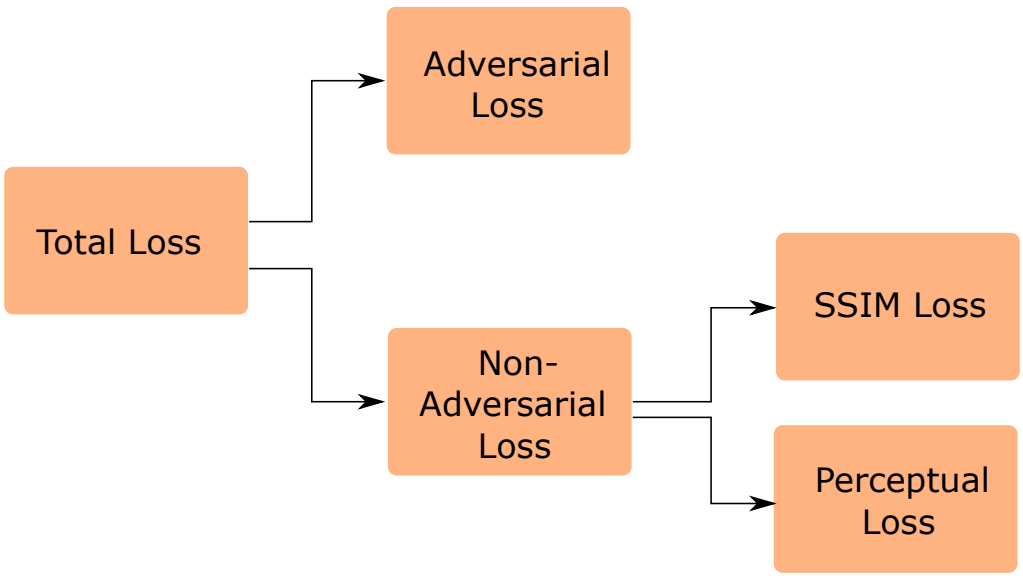

Figure 4.8: Total loss of the cGAN

In summary, the overall loss function of the network is shown in the figure 4.8. Therefore, the entire loss function of this cGAN is the sum of all the loss functions mentioned above.

$$L = L_{adv} + L_{SSIM} + L_{perc} \tag{4.20}$$

4.2.3.3 Architecture of the cGAN

The network model structure proposed in this thesis includes generative model and discriminative model. These two parts of the model are shown in Figure 4.7.
In terms of the structural design of deep networks, the predecessors have carried out quite a lot of research. Among them, some classic network designs have achieved outstanding results in many tasks and are widely used by scholars, greatly simplifying the work of network design. In image denoising tasks, commonly used CNNs mainly include encoder-decoder structure [51], U-Net structure [52] and deep residual network structure [53].

Architecture of the Generator

In this thesis, the U-Net structure is implemented as a generator.

Principle of U-Net

The U-Net model is a CNN, which was proposed by Olaf Ronneberger et al. in 2015 [52]. U-Net is improved on the basis of a fully connected neural network, which uses an encoding-decoding architecture model. The encoding part extracts the feature information of the image layer by layer through continuous convolution operation. On the other side, the decoding part can gradually restore the position and characteristic information of the image through continuous deconvolution operation. The overall structure of the model is relatively simple, it looks like a capital English letter ”U”, so it is called U-Net. The specific structure of the model is shown in the figure 4.9 below.

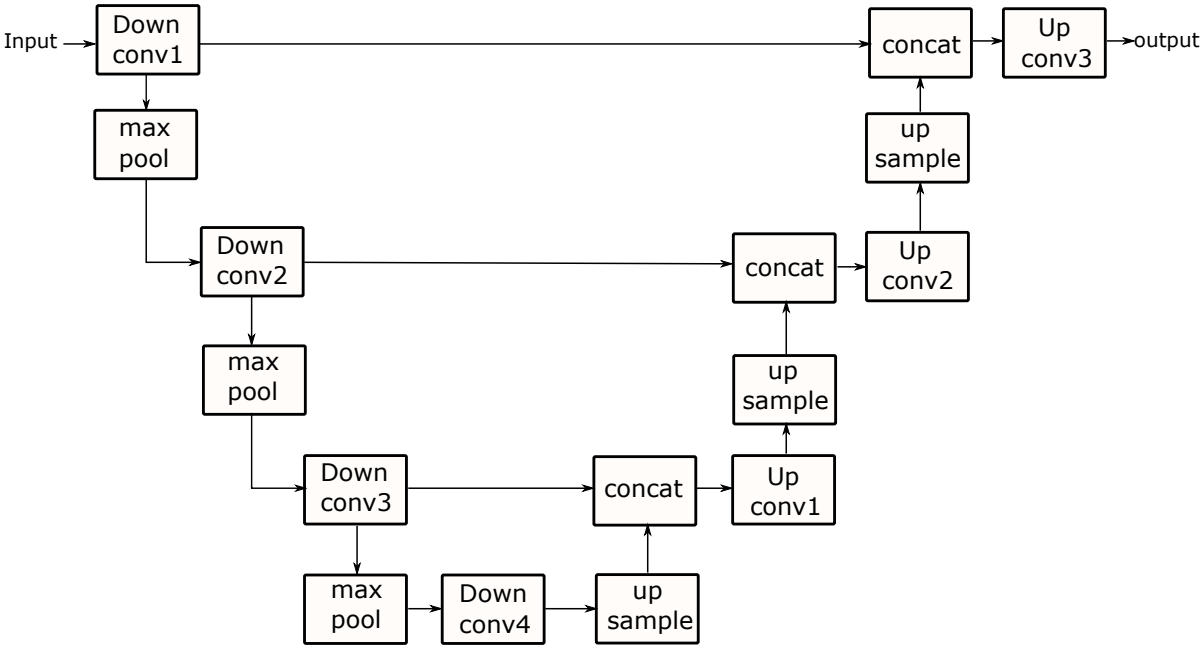

Figure 4.9: Basic Architecture of U-Net

In the encoding part, the network continuously uses convolution layers with the max pooling layers to extract the feature map. The size of the convolution kernel is 3x3 pixels, and the

pooling window is 2x2 pixels. In the decoding part of the network, each layer is composed of an upsampling layer with a convolution layer. When performing upsampling, the model performs a deconvolution operation on the feature map to make it larger in size but halved in dimension, then merge the feature map in corresponding encoding layer with it. Then the convolution operation is performed on the merged feature map. The operation process is repeated until the prediction result map is finally output.
In summary, the U-Net model has the following major features: First, it is a deep network structure based on encoding and decoding, which can achieve end-to-end result output which means the input is an image, the output is also an image; Second, it has a special U-shaped network structure, which reduces the image by 16 times through four times downsampling operations, and extracts the image features deeply. The image restoration operation was performed by four times upsampling operations to restore the extracted advanced feature map to the original image size. The upsampling also ensured that the edges of the restored image were not too rough. Third, the network also has the characteristics of skip connections. Combining the semantic feature maps of the upper layer and the visual feature maps of the lower layer at the same level is actually equivalent to combining feature maps of different scales. Therefore, multi-scale prediction can be effectively realized and the image can be better restored.

U-Net implementation

Table 4.2 shows the specific parameters of the U-Net model built in the thesis, where Conv represents convolution layer, ConvT represents deconvolution layer, Act represents activation, Drop represents random Dropout, Cat represents Concatenate, BN represents Batch Normalization and Crop represents Cropping layer.
This U-Net has two parts, which consists of encoder part and decoder part. There are 4 encoder blocks and 4 decoder blocks. For example, the encoder block 2 has one convolutional layer Conv2. The result is passed to Batch Normalization bn2 and followed by a LeakyRelu activation function. Then, the output of the encoder block 2 is passed to the next encoder block 3 and to the Concatenate layer in decoder block 3. Similarly, the decoder block has one deconvolutional layer. But the random Dropout is introduced into the decoder block. The output of the decoder block is concatenated with the corresponding output of the encoder block. Then, the merged information is passed to the next block.

Architecture of the Discriminator

The discriminative model implemented in this thesis is a simple binary CNN model for discriminative authenticity, which consists of the convolutional layers and a fully connected layer. After the image is input into the discriminative model, the feature extraction and convolution operations are performed continuously, and then the features extracted by the convolution layer are summarized and output through the fully connected layer. Based on the size of the final output value, the discriminator determines whether the input image is true or false.
Table 4.3 shows the detailed parameters of the discriminative model built in the thesis, where Conv represents convolution layer, Act represents activation, drop represents random Dropout, BN represents Batch Normalization, Flatten represents Flatten layer and Dense represents the Dense layer.

Table 4.2: specific parameters of U-Net Model

	layer	Kernel size	stride	padding	Activation function	Output size
Input	input					256x256x1
	ZeroPadding					256x256x1
encoder	Cov1	3x3	2	same		128x128x16
	Act1				LeakyRelu	128x128x16
	Cov2	3x3	2	same		64x64x32
	BN2					64x64x32
	Act2				LeakyRelu	64x64x32
	Cov3	3x3	2	same		32x32x64
	BN3					32x32x64
	Act3				LeakyRelu	32x32x64
	Cov4	3x3	2	same		16x16x128
	BN4					16x16x128
	Act4				LeakyRelu	16x16x128
decoder	ConvT1	3x3	2	same		32x32x64
	BN1					32x32x64
	Drop1					32x32x64
	Cat1					32x32x128
	Act1				LeakyRelu	32x32x128
	ConvT2	3x3	2	same		64x64x32
	BN2					64x64x32
	Drop2					64x64x32
	Cat2					64x64x64
	Act2				LeakyRelu	64x64x64
	ConvT3	3x3	2	same		128x128x16
	BN3					128x128x16
	Drop3					128x128x16
	Cat3					128x128x32
	Act3				LeakyRelu	128x128x32
	ConvT4	3x3	2	same		256x256x1
output	Crop					256x256x1

Table 4.3: specific parameters of discirminative Model

layer	Kernel size	stride	padding	Activation function	Output size
input					256x256x1
Cov1	3x3	2	same		128x128x16
Act1				LeakyRelu	128x128x16
Cov2	3x3	2	same		64x64x32
BN2					64x64x32
Act2				LeakyRelu	64x64x32
Cov3	3x3	2	same		32x32x64
BN3					32x32x64
Act3				LeakyRelu	32x32x64
Cov4	3x3	2	same		16x16x128
BN4					16x16x128
Act4				LeakyRelu	16x16x128
Flatten					32768
Dense					1

5 Experiment and Results

5.1 Performance Metrics Evaluation

In the past few decades, quality assessment (QA) has been widely used in many fields, such as image compression, video encoding and decoding, and video surveillance. Therefore, QA has become a research area of interest. Every year, a large number of new QA algorithms appear. Some are extensions of existing algorithms, and some are applications of QA algorithms.
Quality assessment can be divided into Image Quality Assessment (IQA) and Video Quality Assessment (VQA). This thesis focus the image quality assessment. IQA can be divided into subjective assessment and objective assessment. The subjective evaluation is to evaluate the quality of the image from the subjective perception of the person. First, the original reference image and the distorted image are given, and the annotator is allowed to score the distorted image. Generally, the average subjective score (Mean Opinion Score, MOS) or the average subjective score difference (Differential Mean Opinion Score, DMOS) is used. The objective assessment is to establish a mathematical model based on the subjective visual system of the human eye, and calculate the image quality through a specific formula. Compared with subjective assessment, objective assessment has the characteristics of batch processing and reproducible results. This thesis uses PSNR and SSIM as the evaluation standard [54].

PSNR

Peak Signal to Noise Ratio (PSNR) is usually used to evaluate the quality of an image after compression compared with the original image. The higher the PSNR, the smaller the distortion after image compression. It can be calculated with the help of Mean Square Error (MSE). The calculation Equation is give by:

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [I(i,j) - K(i,j)]^2 \tag{5.1}$$

$$PSNR = 10 \log_{10}(\frac{(MAX)^2}{MSE}) \tag{5.2}$$

where I and K represents two monochrome images with the size of MxN, MAX represents the maximum value of the color for the image color. If each sampling point is represented by 8 bits, the value is 255.
A larger PSNR value means less distortion.

SSIM

PSNR is an image quality evaluation based on errors between corresponding pixels. But the

human visual system is not taken into account, it often happens that the evaluation results are inconsistent with human subjective feelings. It is possible that an image with a higher PSNR may look worse than a lower PSNR image. Therefore, the evaluation method of Structural Similarity Index (SSIM) was proposed.
SSIM is used for measuring the similarity between two images. It is based on three comparison measurements between the sample of x and y: luminace (l), contrast (c) and structure (s). The Equation of SSIM (5.3) is given as follows:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5.3}$$

where μ_x and μ_y are the average of x and y, σ_x^2 and σ_y^2 are the variance of x and y, σ_{xy} is the correlation coefficient of x and y, c_1 and c_2 are the variable to stablize the division with weak denominator. The value range of SSIM is $[0, 1]$, the larger the value, the smaller the image distortion.

5.2 results of Gamma Correction

The results of radar data noise reduction by using Gamma Correction are generated from Python with the program library Matplotlib. All the results are presented in grayscale images. After manually adjusting the value of gamma, and then comparing the corresponding output image, it is determined that when the value of gamma is 2, the noise reduction effect is the best. Thus, in this section, γ value is chosen to be 2.
First, we perform gamma correction on the Range Doppler (RD) images after artificially adding noise. The results are shown in the figure 5.1. From the figure 5.1 we can see that the image on the far left is original RDMs. We have selected 3 samples from the database. The image in the middle column is the artificially processed RDMs corresponding to the original RDMs. The image on the far right is the corresponding RDMs after Gamma Correction. Analyzing the image quality, it can be found that after gamma correction, the noise on the artificial RDMs is reduced compared to before the correction, but there is still some noise on the image.

Table 5.1: Quantitative Evaluation Results of Artificial RDMs using Gamma Correction

	SSIM	PSNR
Input A	0.252	19.70
Output A	0.462	28.51
Input B	0.392	25.82
Output B	0.664	36.26
Input C	0.184	18.66
Output C	0.486	27.15

Table 5.1 describes the SSIM and PSNR values of three randomly selected samples before input and output. From the table 5.1, it can be clearly found that after Gamma correction, the SSIM and PSNR values of the three samples have significantly improved, which means the objective image has a good noise reduction effect, and the image distortion is smaller.

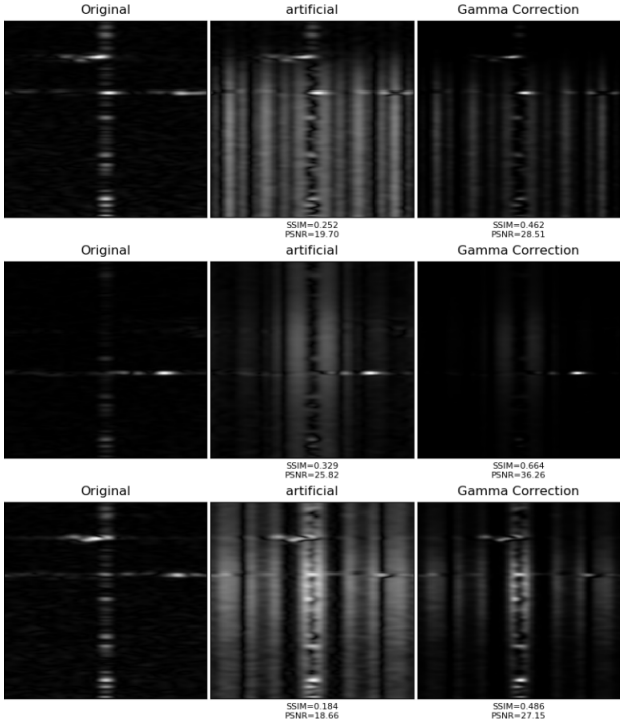

Figure 5.1: artificial RDMs using Gamma Correction

Table 5.2: Quantitative Evaluation Results of Measured RDMs using Gamma Correction

	SSIM	PSNR
Input A	0.014	15.53
Output A	0.353	19.75
Input B	0.127	16.98
Output B	0.524	21.57
Input C	0.101	15.47
Output C	0.328	21.13

Then, we take real measured RDMs as input. Then after gamma correction, the experimental results are shown in Figure 5.2. The same sampled original RDMs are still used for comparison. According to the figure 5.2, it can be found that the small noise on the image is reduced a lot after the measured RDMs are corrected. However, compared with the artificial RDMs after noise reduction, there is still a lot of large noise. Because the noise in the measured RDMs is more classified than the artificially added noise, the performance of gamma correction on the measurement data set is not so good.
Table 5.2 describes the quantitative evaluation results of selected samples of measured RMDs. It can be found that the value of SSIM has been greatly improved, but the value of PSNR has not improved much. But overall the distortion of the RDMs becomes smaller.

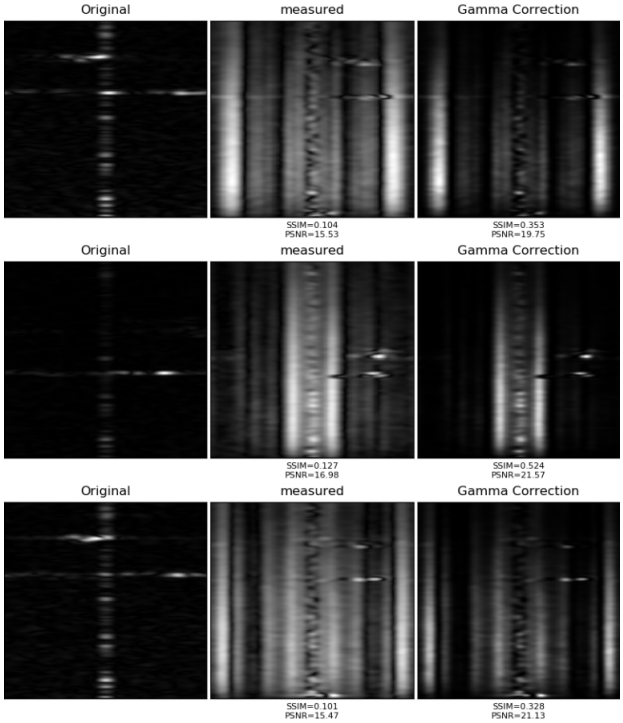

Figure 5.2: measured RDMs using Gamma Correction

5.3 results of PCA and PPCA

We also use Python to implement PCA and PPCA for noise reduction on RDMs. For the parameter selection part of PCA and PPCA, we choose to retain 90% of the image features after image reconstruction. The same steps as Gamma Correction, we choose three artificial RDMs samples that are the same as the gamma correction part, implement PCA and PPCA respectively, and finally get the results, as shown in Figure 5.3 and Figure 5.4.

According to the figure 5.3 and 5.4, it can be found that after simply implementing PCA and PPCA to reconstruct RDMs, their performance is not good, or even worse. On the RDMs reconstructed by PCA and PPCA, the two methods of PCA and PPCA do not remove the artificial noise originally added, and even the maps are partially distorted. The original detection points on RDMs were even removed as noise reconstruction. Compared with PCA, although PPCA fails to fully identify the noise and the target part, the image is not distorted like PCA.

Table 5.3: Quantitative Evaluation Results of Artificial RDMs using PCA and PPCA

	SSIM(PCA)	PSNR(PCA)	SSIM(PPCA)	PSNR(PPCA)
Input A	0.252	19.70	0.252	19.70
Output A	0.237	19.63	0.237	19.72
Input B	0.329	25.82	0.329	25.82
Output B	0.319	25.60	0.327	25.83
Input C	0.184	18.66	0.184	18.66
Output C	0.176	18.67	0.187	18.66

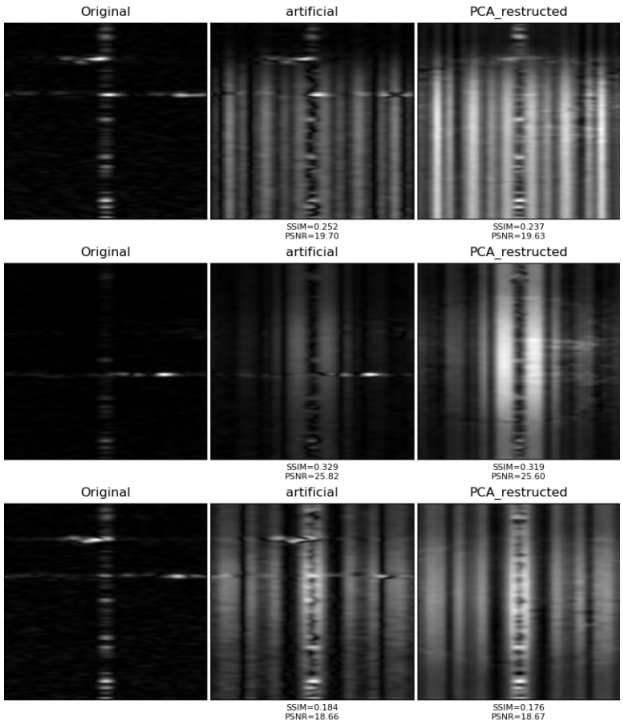

Figure 5.3: artificial RDMs using PCA

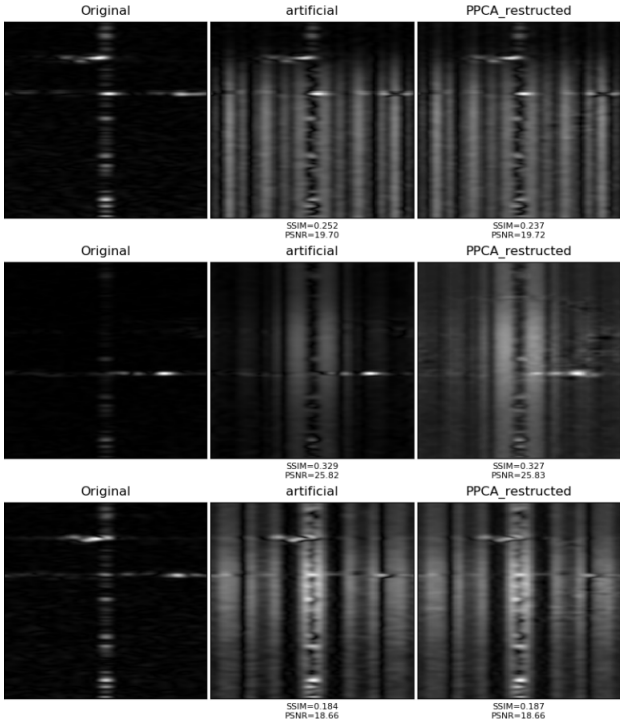

Figure 5.4: artificial RDMs using PPCA

From the Table 5.3, we can see that the SSIM values and PSNR values of the RDMs after PCA reconstruction have basically not changed significantly, and have even declined slightly. This shows that the effect of noise reduction by PCA is not good and the maps is distorted. The SSIM and PSNR values of the RDMs after PPCA reconstruction increased slightly. This also means that PPCA has a better performance than PCA because of the existence of probabilistic models.
After implementing the PCA and PPCA on the same selected measurement RDMs, the noise reduction results are obtained, as shown in Figure 5.5 and Figure 5.6. We can find that the reconstruction results are similar to the previous processing results of artificial RDMs, and the image distortion phenomenon also appears and the noise cannot be stripped. As can be seen from the Table 5.4, the effect of measuring RDMs after reconstruction by PCA and PPCA is worse.
The reason for this phenomenon may be that the patterns of noise data and target data are distributed uniformly and mixed together in the RDMs, and the data cannot be easily distinguished after the dimension reduction. It may even be that the target data has too few patterns, and the noise data far exceeds the target data, causing the PCA and PPCA algorithms to discard the target data after dimensionality reduction. This may be the cause of the poor noise reduction effect of PCA and PPCA.

Table 5.4: Quantitative Evaluation Results of Measured RDMs using PCA and PPCA

	SSIM(PCA)	PSNR(PCA)	SSIM(PPCA)	PSNR(PPCA)
Input A	0.104	15.53	0.104	15.53
Output A	0.237	19.63	0.105	15.53
Input B	0.127	16.98	0.127	16.98
Output B	0.319	25.60	0.124	17.00
Input C	0.101	15.47	0.106	15.57
Output C	0.176	18.67	0.102	15.47

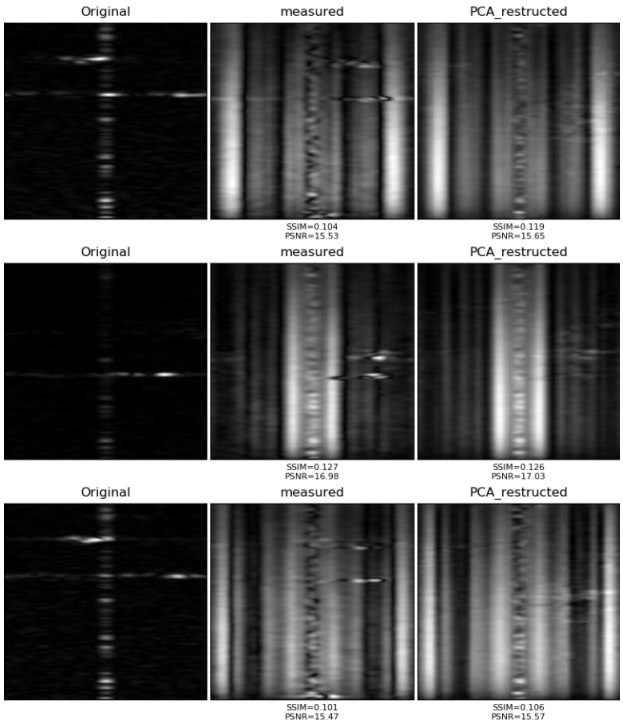

Figure 5.5: measured RDMs using PCA

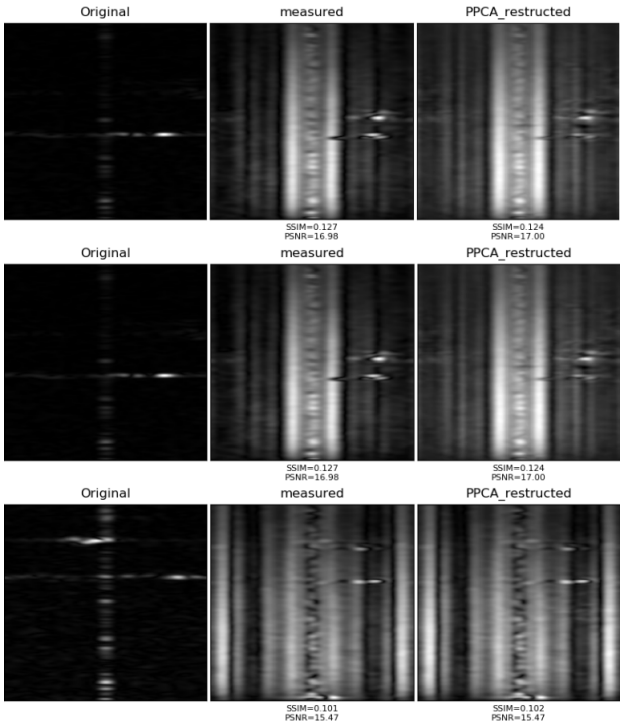

Figure 5.6: measured RDMs using PPCA

5.4 results of Denoised Autoencoder

To train the Denoised Autoencoder model, the pre-processed radar data in Chapter 2 as the entire data set is used. The data set is divided into a training set and a test set at a ratio of 4 to 1. During the training process, the artificial RDMs in the training set and the corresponding original RDMs are used as the training set.
First, the artificial RDMs in the test set and the corresponding original RDMs are used as the test set. Since the noise reduction is not completely completed after training for 100 epochs, 200 epochs are used in this part. For easier comparison, we also sampled the same RDMs as the previous noise reduction method. Figure 5.7 illustrates the noise reduction effect of artificial RDMs after 200 epoch training.

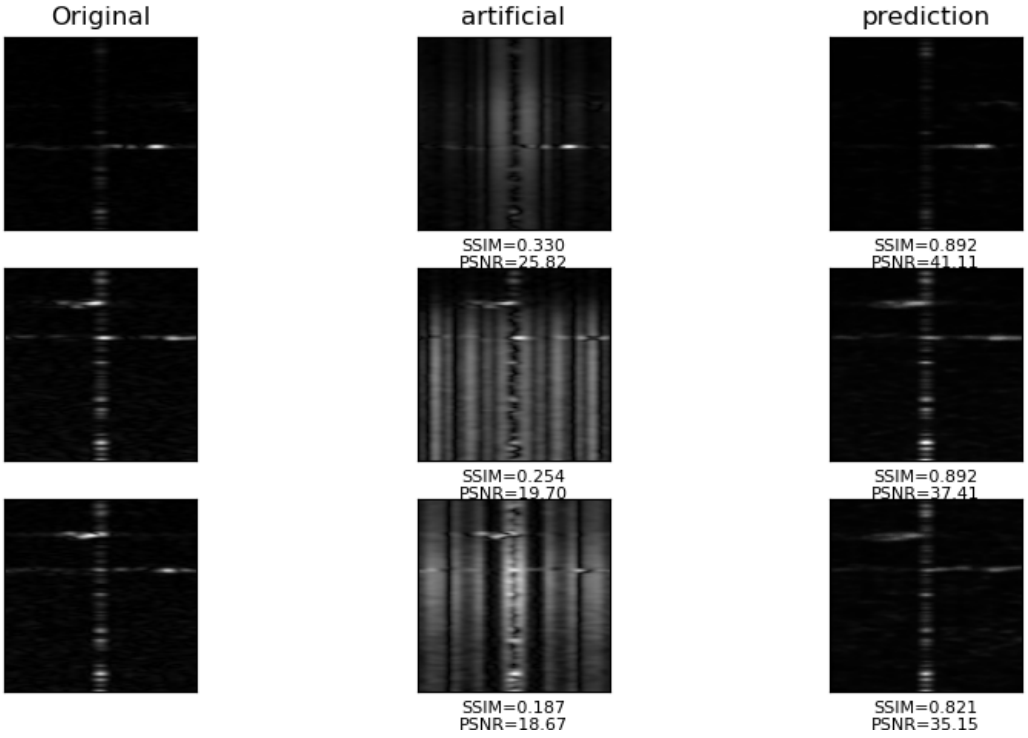

Figure 5.7: artificial RDMs using DAE

It can be found from the figure 5.7 that the noise on the artificial RDMs after DAE noise reduction is well removed, and the image is very clear. The targets in the generated image are also well described. According to the table 5.5, it can be clearly seen that after DAE noise reduction, the SSIM value and PSNR value of the generated RDMs have been greatly improved. Especially the SSIM value, which indicates that the generated image is very similar to the original RDMs.
Then, the measured RDMs and the original RDMs in the test set are used as the test set. After training for 200 epochs, the noise reduction results of the measured RDMs are obtained, as shown in Figure 5.8. After DAE noise reduction, the RDMs generated are clear and most of the noise is gone. However, the targets in the generated images are not well described, there is a certain deviation between the original RDMs and the generated measured RDMs. Because

Table 5.5: Quantitative Evaluation Results of Artificial RDMs using DAE

	SSIM	PSNR
Input A	0.330	25.82
Output A	0.892	41.11
Input B	0.254	19.70
Output B	0.892	37.41
Input C	0.187	18.67
Output C	0.821	35.14

there are fewer pattern of the target, the DAE model may not fully capture the features it needs. At the same time, the measured RDMs in the test set do not match the original RDMs, and the training set is relatively small. The above reasons may cause some deviations in the features learned by the DAE model.

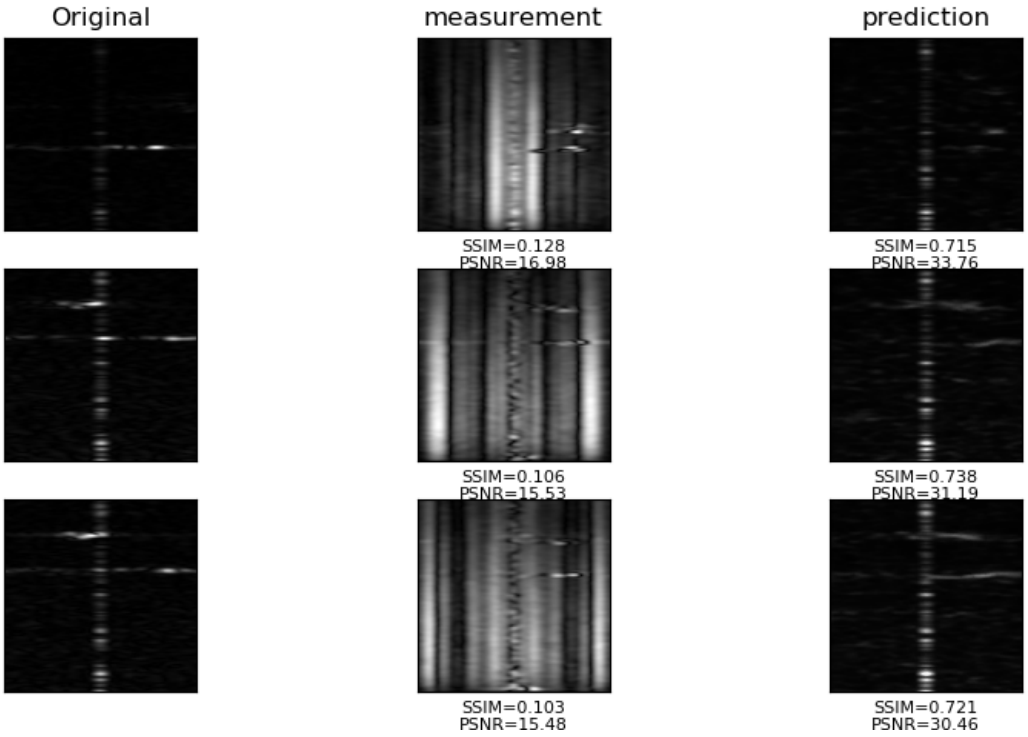

Figure 5.8: measured RDMs using DAE

From the table 5.6, we can see that the DAE noise reduction effect is quiet well. The SSIM value and PSNR value of measured RDMs have been greatly improved after training, especially the SSIM value, which further shows that most of the noise is removed already.

Table 5.6: Quantitative Evaluation Results of Measured RDMs using DAE

	SSIM	PSNR
Input A	0.128	16.98
Output A	0.715	33.76
Input B	0.106	15.53
Output B	0.738	31.19
Input C	0.103	15.48
Output C	0.721	30.46

5.5 results of U-Net and cGANs

In this section, we first train and test the generative model U-Net separately, and then cooperate with the discriminative model to train the entire conditional GAN model. The data set structure used is the same as when training the DAE. Since U-Net and cGAN model are faster to train, we choose 100 times as the training epoch.

First, we still choose the training part of the splitted artificial RDMs and the original RDMs as the training set of U-Net model, the testing part of the artificial RMDs and the original RDMs as its test set. After 100epochs training, the three new RDMs generated are shown in Figure 5.9. It can be found that the new RDMs generated by U-Net as a generative model of the adversarial network is very similar to the original image, no noise is seen on the map, and the patterns of most targets are identified and generated.

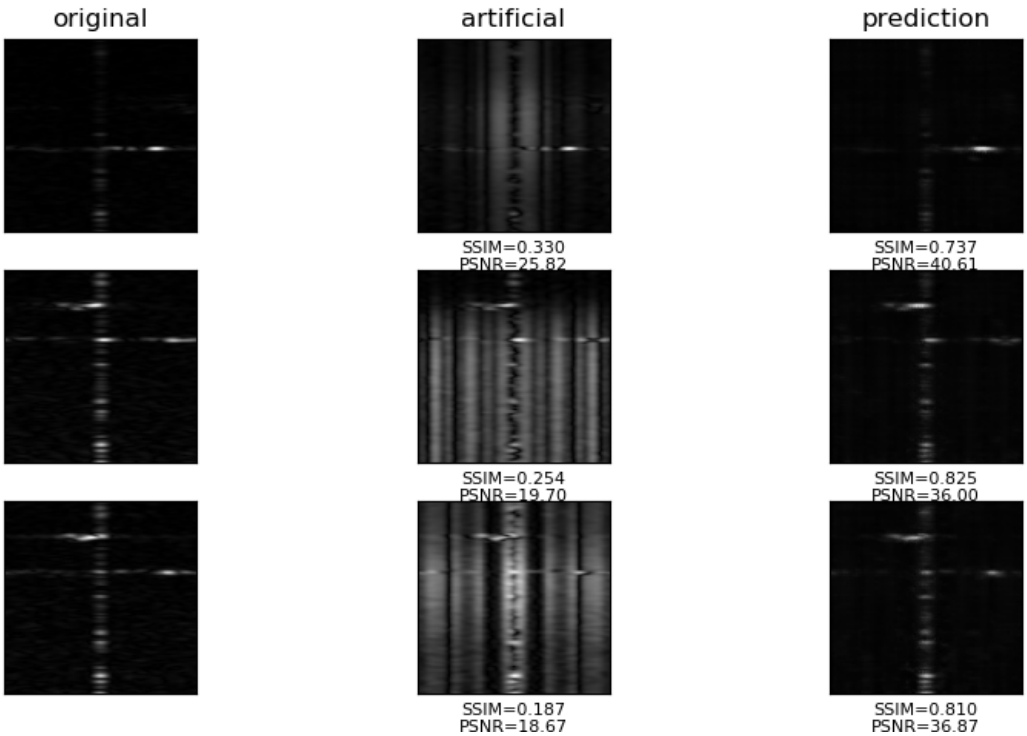

Figure 5.9: artificial RDMs using U-Net

According to the table 5.7, the generated RDMs have greatly improved both in SSIM value and PSNR value. This shows that the noise reduction performance of U-Net is excellent on the simulation test set.

Table 5.7: Quantitative Evaluation Results of Artificial RDMs using U-Net

	SSIM	PSNR
Input A	0.330	25.82
Output A	0.737	40.61
Input B	0.254	19.70
Output B	0.825	36.00
Input C	0.187	18.67
Output C	0.810	36.87

For more convenient analysis, we introduce the training results of conditional GAN here. Conditional GAN requires one more training step than U-Net. The RDMs generated by the U-Net model and the original RDMs are used as the training set of the discriminative model. They are the input part of the discriminative model.
Because the training speed of the conditional GAN model is a little faster than that of U-Net, and its Loss will fluctuate due to the confrontation between the two models, the prediction results obtained by each epoch training will also fluctuate. Therefore, we choose the one with the best prediction performance among these 100 epochs as its result.

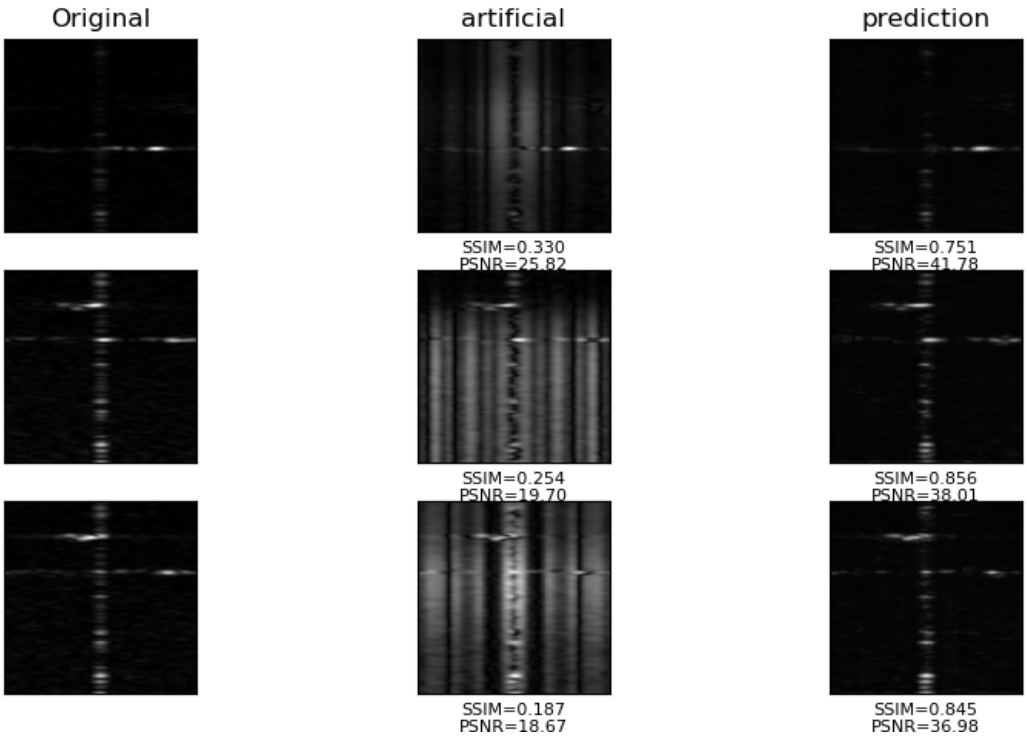

Figure 5.10: artificial RDMs using cGAN

Observing the figure 5.10 with the naked eye, it can be seen that the new RDMs generated are basically no different from the original RDMs. Visually, there is no noise, and the generated target pattern is very similar to the target pattern on the original RDMs.

Table 5.8: Quantitative Evaluation Results of Artificial RDMs using cGAN

	SSIM	PSNR
Input A	0.330	25.82
Output A	0.751	41.78
Input B	0.254	19.70
Output B	0.856	38.01
Input C	0.187	18.67
Output C	0.845	36.98

Comparing Table 5.7 and Table 5.8, the RDMs generated by the cGAN model are higher in both the SSIM value and the PSNR value, which indicates that the RDMs generated by the cGAN model are more realistic and better on the simulated data set.
Next, we test the noise reduction effect of the U-Net model and the cGAN model on the real measured RDMs in turn. It is still the original artificial RDMs and the measured RDMs that replace the artificial RDMs in the test part as the test set of the entire model. After training 100epochs, the result of U-Net model is shown in Figure 5.11. On the real measurement test data set, the noise reduction effect of the new RDMs generated by U-Net is still good. But there is no corresponding original image to match it, a small part of the target pattern is not recognized.
According to the table 5.9, the RDMs generated by the U-Net model on the real measurement data set have some declines in the SSIM value and the PSNR value, compared to the simulation data set.

Table 5.9: Quantitative Evaluation Results of Measured RDMs using U-Net

	SSIM	PSNR
Input A	0.128	16.98
Output A	0.618	33.66
Input B	0.106	15.53
Output B	0.679	30.06
Input C	0.103	15.48
Output C	0.681	29.58

However, Figure 5.12 illustrates that, the RDMs generated by the cGAN model on the measurement dataset are clearer and better at identifying target patterns than U-Net. This means that with the existence of a discriminative network, even if there is no corresponding original image, the cGAN model can still restore the measured RDMs without noise. From the table 5.10, we can find that on SSIM and PSNR, the performance of cGAN on the real measurement data set is somewhat lower than that on the simulation set. But compared to U-Net, cGAN has a better noise reduction effect, and the RDMs generated are more realistic.

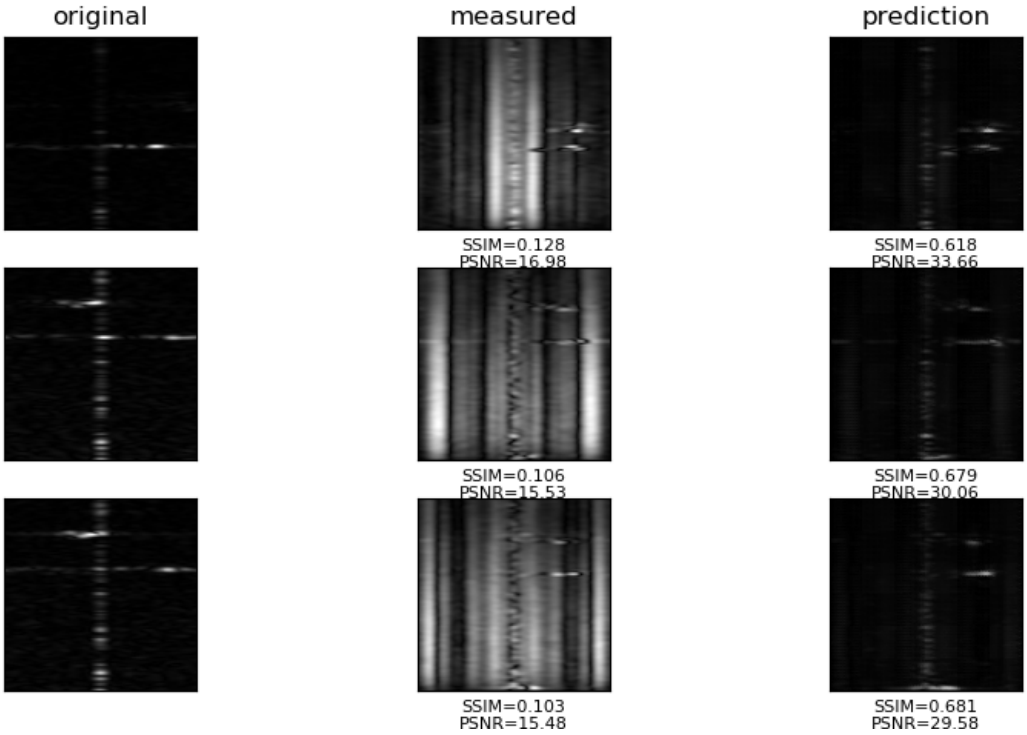

Figure 5.11: measured RDMs using U-Net

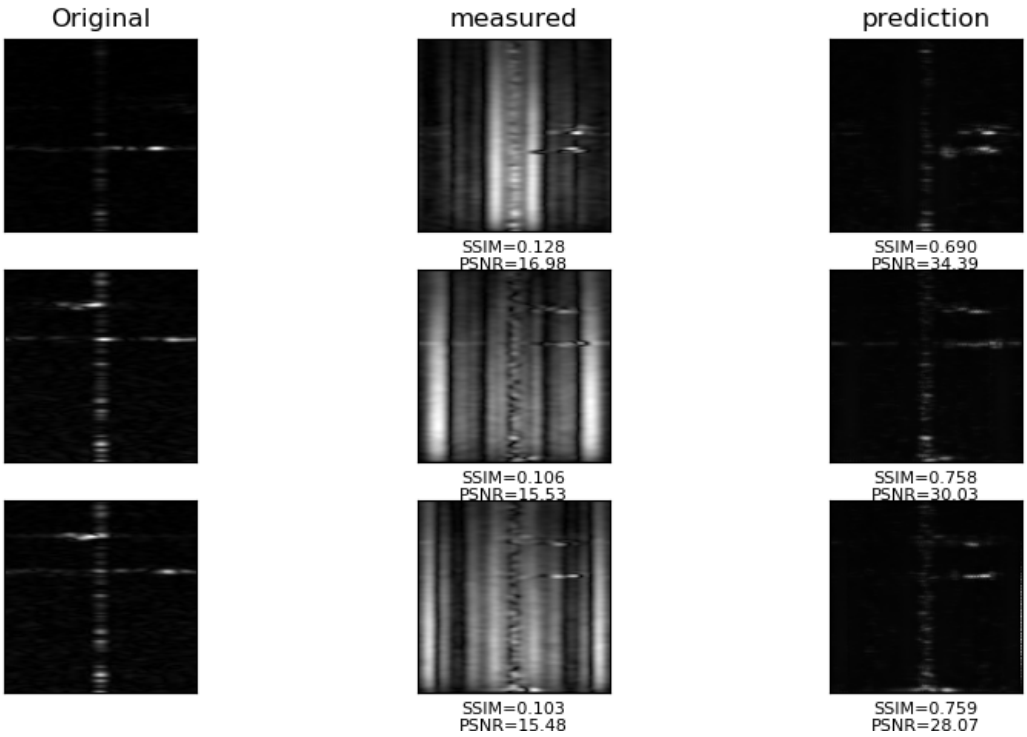

Figure 5.12: measured RDMs using cGAN

Table 5.10: Quantitative Evaluation Results of Measured RDMs using cGAN

	SSIM	PSNR
Input A	0.128	16.98
Output A	0.690	34.39
Input B	0.106	15.53
Output B	0.758	30.03
Input C	0.103	15.48
Output C	0.759	28.07

5.6 Comparision

In this section, we put together all the noise reduction techniques involved in this paper such as gamma correction, PCA, PPCA, DAE, U-NET, and cGAN to compare the results.
First, start with the comparison of image quality. The overall result is shown in Figure 5.13. We selected one sample from the three samples. The image on the left is the input, which is artificial RDM and measured RDM. These images on the right are their corresponding output RDMs. The top row on the right is the result of traditional noise reduction methods, namely gamma correction, PCA and PPCA. The lower row of images on the right is the result of deep learning noise reduction methods, which are DAE, U-Net and cGAN denoising.
In the traditional method, it can be clearly seen that the PCA and PPCA noise reduction methods are not good, but gamma correction can suppress noise and partially reduce noise. According to the figure 5.13, compared with the deep learning method of noise reduction technology, the traditional method of noise reduction has many defects and cannot completely remove the noise, especially in the recognition of the target pattern, the traditional method cannot completely distinguish the noise and the target. Comparing the three deep learning methods, it can be found that the noise reduction effect of each method is very good. There is almost no noise on the generated RDM, and the rendering of the target on the RDM is better restored. But in terms of training speed, cGAN has the fastest training speed, followed by U-Net, and the slowest is DAE.
On the comparison between the noise reduction results of artificial RDM and measured RDM, we can clearly see that both the traditional method and the deep learning method have a performance decline. The traditional method has the most degradation, and the deep learning method has relatively little performance degradation, especially the cGAN noise reduction method.

Table 5.11: Comparison of SSIM values with different denoising techniques

	Gamma Correction	PCA	PPCA	DAE	U-Net	cGAN
Artificial	0.664	0.319	0.327	0.892	0.737	0.751
Measurement	0.524	0.126	0.124	0.715	0.618	0.690

Table 5.11 shows the comparison of SSIM values with different denoising techniques. It is clearly given that DAE gives best SSIM value among all on both the artificial and measurement RDM. Ranked second is the cGAN noise reduction method. On the comparison between simulation and real measurement results, we can find that the performance degradation of cGAN is the smallest.

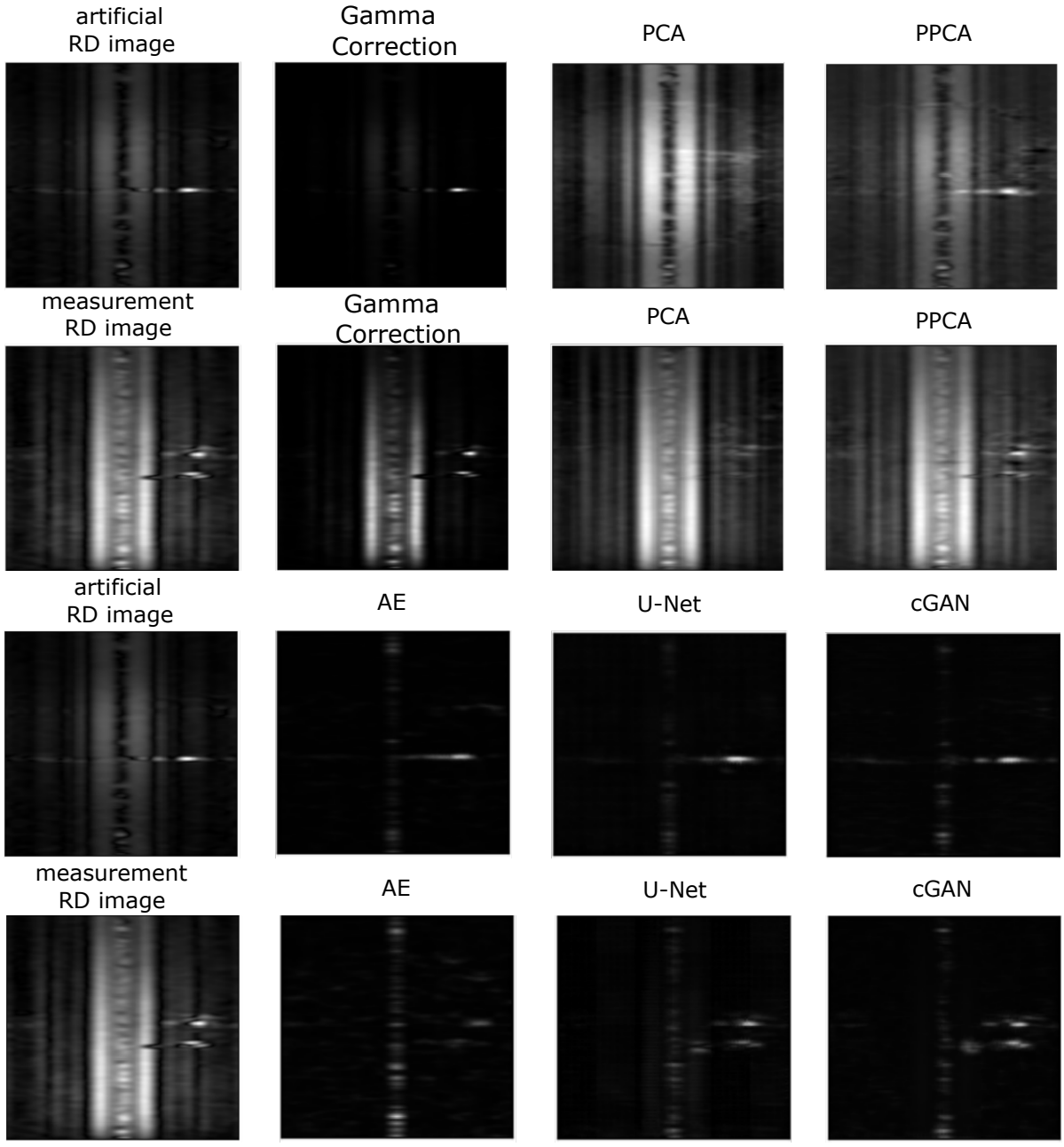

Figure 5.13: Denoising Output Comparison

Table 5.12: Comparison of PSNR values with different denoising techniques

	Gamma Correction	PCA	PPCA	DAE	U-Net	cGAN
Artificial	36.26	25.60	25.83	41.11	40.61	41.78
Measurement	21.57	17.03	17.00	33.76	33.66	34.39

Table 5.12 shows the comparison of PSNR values with the denoising techniques above mentioned. It is obvious shown that cGAN gives the best PSNR vale among all both on the artificial and measurement RDMs. As for the performance degradation, there is no doubt that cGAN has the smallest performance decline between simulation and measurement results. Since the SSIM value represents the structure similarity between the output image and the original image, in the actual real measurement environment, we cannot get a clean original image. Therefore, on the measurement data set, we need to evaluate the combination of SSIM and PSNR.
Combining quantitative and qualitative comparisons, it can finally be determined that cGAN has the best performance, the strongest noise reduction effect, the fastest training speed, and the highest similarity in the radar range Doppler denoising techniques.

6 Summary and Outlook

6.1 Summary

With the continuous development of the field of autonomous driving, the requirements for radar detection, radar tracking, and radar classification are becoming higher and more accurate. Therefore, the application value of Radar data denoising is constantly increasing. Due to all the deficiencies of traditional denoising technology, this thesis utilizes deep learning method to denoise the radar Range Doppler maps based on traditional denoising technology.

The specific work content of this thesis can be summarized as follows:

Firstly, this article briefly introduces the basic background of radar technology and its signal processing methods. On this basis, we use the automatic driving toolbox in Matlab to perform a series of simulation processing to obtain the original and artificially noise added radar Range Doppler maps. Combined with the radar Range Doppler maps measured in the real world, the original data set required for training is finally formed, thereby solving the problem of data set acquisition.

Secondly, in order to facilitate the comparison between traditional methods and deep learning methods, traditional denoising techniques, gamma correction, principal component analysis, and probabilistic principal component analysis are applied.

Thirdly, this thesis implements several methods of deep learning to denoise the radar Range Doppler maps. Based on Autoencoder, a radar data denoising method based on Denoising Autoencoder is proposed. This paper improves the structure of the traditional denoising autoencoder. It uses a convolutional neural network architecture to build the Denoising Autoencoder. The cross-entropy loss is used as the reconstruction loss and the SSIM loss is introduced. The two losses are combined to constrain the learning process of Denoising Autoencoder. After that, based on generative adversarial network, a radar data denoising algorithm model based on conditional generative adversarial learning is established. In this thesis, the traditional U-Net is improved as a generative model, and a two-class CNN is designed as a discriminant model. The three loss functions of adversarial loss, SSIM loss, and perceptual loss are used to control the learning process of two network models. Depending on the mutual game between the two models during training, the models can obtain better denoising capabilities.

Finally, a series of experiments prove the effectiveness of the method in this thesis and verify the performance of the above denoising technology on the simulation data set and the measurement data set. Then, the denoising effects of cGAN denoising technology and other technologies on radar Range Doppler maps were compared subjectively and objectively. The results show that the denoising method designed in this thesis is significantly better than several other denoising methods, the generated images are much more natural and realistic.

6.2 Outlook and Future Scope

The ability to train directly on the real measured noise radar range-Doppler maps makes the denoising method using cGAN in this thesis expected to be widely used in practical scenarios. On the subject of radar data denoising, the following points are worth exploring and trying.

More Data Set Collection

Deep learning networks usually require large data sets so that they can learn the features they need. Compared to pre-trained deep learning frameworks, the dataset used in this thesis is too small. In future work, more data sets can be collected, which can make the cGAN network used in this thesis have better denoising effect.

Proposition of more objective evaluation indicators

The image quality evaluation standards used in this article are PSNR and SSIM, but these two indicators are still far from human subjective perception. Therefore, the field of radar data denoising needs to propose more effective and objective evaluation standards, and it is closer to the subjective perception of the human eye.

Improvement of network structure

The network structure in this thesis is not explored deeply, only using a U-Net-type network as a generator and a two-class convolutional neural network as a discriminator. In order to pursue higher generation quality and better denoising effect, future work can try to introduce a multi-scale network structure into the model, while deepening the number of network layers, and improving the model's capacity.

Research on Perceptual Loss

In this thesis, the data features captured by the discriminator are used as image perceptual losses, and the perceptual losses provided by the pre-trained model VGG are not used. In the future, we can further research what is the difference between these two losses and what are their advantages.

Choose another generative model

There are several powerful generative models such Variational Autoencoder(VAE) [55]. In future work, we can replace the U-Net-type Generator used in this thesis with VAE model and combine it with the GAN network to make it have better image generation capabilities.

Bibliography

[1] J. Duda, "A History of Radar Meteorology: People, Technology, and Theory", *Iowa State University. URL*, 2015.

[2] Z. Lu, D. Mann, J. T. Freymueller, and D. J. Meyer, "Synthetic aperture radar interferometry of Okmok volcano, Alaska: Radar observations", *Journal of Geophysical Research: Solid Earth*, Wiley Online Library, vol. 105 (B5), pp. 10 791–10 806, 2000.

[3] A. G. Stove, "Linear FMCW radar techniques", in *IEE Proceedings F*, IET, vol. 139, 1992, pp. 343–350.

[4] D. E. Dudgeon and R. T. Lacoss, "An overview ofautomatic target recognition", Citeseer, 1993.

[5] S. P. Jacobs and J. A. O'Sullivan, "Automatic target recognition using sequences of high resolution radar range-profiles", *IEEE Transactions on Aerospace and Electronic Systems*, IEEE, vol. 36 (2), pp. 364–381, 2000.

[6] K. El-Darymli, E. W. Gill, P. McGuire, D. Power, and C. Moloney, "Automatic target recognition in synthetic aperture radar imagery: A state-of-the-art review", *IEEE access*, IEEE, vol. 4, pp. 6014–6058, 2016.

[7] M. Martorella, E. Giusti, A. Capria, F. Berizzi, and B. Bates, "Automatic target recognition by means of polarimetric ISAR images and neural networks", *IEEE transactions on geoscience and remote sensing*, IEEE, vol. 47 (11), pp. 3786–3794, 2009.

[8] P. Singh and R. Shree, "Analysis and effects of speckle noise in SAR images", in *2016 2nd International Conference on Advances in Computing, Communication, & Automation*, IEEE, 2016, pp. 1–5.

[9] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach", *IEEE transactions on neural networks*, IEEE, vol. 8 (1), pp. 98–113, 1997.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, and B. Xu, "Generative adversarial nets", in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[11] M. Z. Alom, T. M. Taha, C. Yakopcic, and S. Westberg, "A state-of-the-art survey on deep learning theory and architectures", *Electronics*, Multidisciplinary Digital Publishing Institute, vol. 8 (3), p. 292, 2019.

[12] J. Li, L. Deng, R. Haeb-Umbach, and Y. Gong, "Robust automatic speech recognition: a bridge to practical applications". Academic Press, 2015.

[13] J. J. Wit, A. Meta, and P. Hoogeboom, "Modified range-Doppler processing for FM-CW synthetic aperture radar", *IEEE Geoscience and Remote Sensing Letters*, IEEE, vol. 3 (1), pp. 83–87, 2006.

[14] A. Delitto and S. J. Rose, "Comparative comfort of three waveforms used in electrically eliciting quadriceps femoris muscle contractions", *Physical therapy*, Oxford University Press, vol. 66 (11), pp. 1704–1707, 1986.

[15] A. G. Parlos, B. Fernandez, A. F. Atiya, J. Muthusami, and W. K. Tsai, "An accelerated learning algorithm for multilayer perceptron networks", *IEEE Transactions on Neural Networks*, IEEE, vol. 5 (3), pp. 493–497, 1994.

[16] A. Coolen and T. W. Ruijgrok, "Image evolution in Hopfield networks", *Physical Review A*, APS, vol. 38 (8), p. 4253, 1988.

[17] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning", in *International Workshop on Artificial Neural Networks*, Springer, 1995, pp. 195–201.

[18] A. F. Agarap, "Deep learning using rectified linear units (relu)", *arXiv preprint arXiv:1803.08375*, 2018.

[19] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks", *arXiv preprint arXiv:1412.6830*, 2014.

[20] S. Tao, T. Zhang, J. Yang, X. Wang, and W. Lu, "Bearing fault diagnosis method based on stacked autoencoder and softmax regression", in *2015 34th Chinese Control Conference*, IEEE, 2015, pp. 6331–6335.

[21] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels", in *Advances in neural information processing systems*, 2018, pp. 8778–8788.

[22] K. Hammernik, F. Knoll, D. K. Sodickson, and T. Pock, "L2 or not L2: impact of loss function design for deep learning MRI reconstruction", in *Proceedings of the 25th Annual Meeting of ISMRM, Honolulu, HI*, 2017.

[23] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution", in *European conference on computer vision*, Springer, 2016, pp. 694–711.

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

[25] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks", in *Advances in neural information processing systems*, 2017, pp. 700–708.

[26] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling", in *Advances in neural information processing systems*, 2016, pp. 82–90.

[27] J. Walker, K. Marino, A. Gupta, and M. Hebert, "The pose knows: Video forecasting by generating pose futures", in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3332–3341.

[28] T. Mathworks, "Matlab logo", 2019.

[29] Martín Abadi, Ashish Agarwal, Paul Barham, and Eugene Brevdo, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems", Software available from tensorflow.org, 2015.

[30] C. A. Poynton, "Rehabilitation of gamma", in *Human Vision and Electronic Imaging III*, International Society for Optics and Photonics, vol. 3299, 1998, pp. 232–249.

[31] L. Cao, K. S. Chua, W. Chong, H. Lee, and Q. Gu, "A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine", *Neurocomputing*, Elsevier, vol. 55 (1-2), pp. 321–336, 2003.

[32] K. Pearson, "Principal components analysis", *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 6 (2), p. 559, 1901.

[33] M. Gastpar, P. L. Dragotti, and M. Vetterli, "The distributed karhunen–loeve transform", *IEEE Transactions on Information Theory*, IEEE, vol. 52 (12), pp. 5177–5196, 2006.

[34] F. J. Sanchez-Marin, "Image registration of gray-scale images using the Hotelling transform", in *Proceedings EC-VIP-MC 2003. 4th EURASIP Conference focused on Video/Image Processing and Multimedia Communications*, IEEE, vol. 1, 2003, pp. 119–123.

[35] X. Wang and H. V. Poor, "Blind multiuser detection: A subspace approach", *IEEE Transactions on Information Theory*, IEEE, vol. 44 (2), pp. 677–690, 1998.

[36] N. Lawrence, "Probabilistic non-linear principal component analysis with Gaussian process latent variable models", *Journal of machine learning research*, vol. 6 (Nov), pp. 1783–1816, 2005.

[37] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis", *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Wiley Online Library, vol. 61 (3), pp. 611–622, 1999.

[38] P. D. Wentzell, D. T. Andrews, D. C. Hamilton, K. Faber, and B. R. Kowalski, "Maximum likelihood principal component analysis", *Journal of Chemometrics: A Journal of the Chemometrics Society*, Wiley Online Library, vol. 11 (4), pp. 339–366, 1997.

[39] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures", in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.

[40] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders", in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[41] M. Mirza and S. Osindero, "Conditional generative adversarial nets", *arXiv preprint arXiv:1411.1784*, 2014.

[42] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks", *arXiv preprint arXiv:1511.06434*, 2015.

[43] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error", *arXiv preprint arXiv:1511.05440*, 2015.

[44] C. Frogner, C. Zhang, H. Mobahi, M. Araya, and T. A. Poggio, "Learning with a Wasserstein loss", in *Advances in Neural Information Processing Systems*, 2015, pp. 2053–2061.

[45] K. Armanious, S. Abdulatif, F. Aziz, B. Kleiner, and B. Yang, "Towards Adversarial Denoising of Radar Micro-Doppler Signatures", *arXiv preprint arXiv:1811.04678*, 2018.

[46] Y. Li, N. Wang, J. Liu, and X. Hou, "Demystifying neural style transfer", *arXiv preprint arXiv:1701.01036*, 2017.

[47] H. Li, Y.-Y. Chen, L.-W. Chu, and J. He, "Image super-resolution reconstruction system and method", US Patent 9,240,033, Jan. 2016.

[48] W. Dong, L. Zhang, G. Shi, and X. Wu, "Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization", *IEEE Transactions on image processing*, IEEE, vol. 20 (7), pp. 1838–1857, 2011.

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

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[51] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation", in *Proceedings of the European conference on computer vision*, 2018, pp. 801–818.

[52] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation", in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.

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

[54] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM", in *2010 20th International Conference on Pattern Recognition*, IEEE, 2010, pp. 2366–2369.

[55] Y. Pu, Z. Gan, R. Henao, and X. Yuan, "Variational autoencoder for deep learning of images, labels and captions", in *Advances in neural information processing systems*, 2016, pp. 2352–2360.

List of Figures

List of Tables