

# Application of Deep Learning Method in analyzing Electron Diffraction Data

Master's Thesis

for the Degree of  
Master of Science  
(M. Sc.)  
in the Subject of Physics

the Faculty of Natural Sciences  
Humboldt University of Berlin

by  
Jie Chen  
*January 2022*

Supervisors:

1. *Prof. Christoph T. Koch*
2. *Prof. Dr.-Ing. Peter Eisert*

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Theoretical Background</b>	<b>4</b>
2.1	Electron Diffraction	4
2.1.1	LARBED Pattern	4
2.1.2	Fast Electrons Scattering	5
2.1.3	The Bloch-wave Method	6
2.2	Basics of Neural Networks	10
2.2.1	Artificial Neural Networks	10
2.2.2	Convolutional Neural Networks	14
<b>3</b>	<b>Methodology</b>	<b>20</b>
3.1	Generation of Dataset	20
3.1.1	pyQEDA	20
3.1.2	Data	20
3.1.3	Data Preprocessing	23
3.2	Neural Networks Design	23
3.2.1	CNN	23
3.2.2	ANN	25
3.3	The Training	26
3.3.1	Task 1: $U_g$ Deviation	26
3.3.2	Task 2: $U_g$ Determination	28
<b>4</b>	<b>Results</b>	<b>30</b>
4.1	Results of Task 1	30
4.1.1	Initial Training	30
4.1.2	Iterative Training	32
4.1.3	Overall Results	34
4.2	Results of Task 2	37
4.2.1	Initial Training	37
4.2.2	Iterative Training	38
4.2.3	Overall results	42
<b>5</b>	<b>Discussion</b>	<b>52</b>
5.1	Discussion of the Training Performance	52
5.2	Discussion of the Prediction	54
5.2.1	Factors influencing Prediction Performance	54
5.2.2	Diffraction Parameters	55
5.2.3	$ZnO$ v.s. $SrTiO_3$	56
5.3	Discussion for the Fundamental Idea	58



# Chapter 1

## Introduction

A wealth of basic crystallographic information, such as the thickness and structure factors of a crystalline sample is contained in the diffraction pattern [1][2]. To obtain an accurate crystal structure, sufficient data are required to uniquely solve the equations provided by dynamical theory [3]. The Large-Angle Rocking-Beam Electron Diffraction (LARBED) technique provides a lot of different dynamical diffraction patterns for the same set of structure factors. By acquiring diffraction data at different beam tilts over a large angular range, the LARBED pattern provides rich and detailed information for crystals as small as a few nanometers [4][5][6].

A recent research paper by Feng et al. [4] used the conjugate gradient descent method to find a set of structure factors that best describes the experimental LARBED pattern. However, this approach tends to get stuck in local minima in the case of more complex crystal structures, large specimen thickness and small tilt range. Recently, more and more applications of neural network methods are emerging in material science. Convolutional Neural Network (CNN) has had some success in predicting material properties from diffraction data [7][8][9]. To find a more general and robust method to determine the structure factors, we explore the capabilities of Artificial Neural Networks (ANN) in analyzing LARBED data. A CNN is applied to analyze 2-dimensional representations of LARBED data. In addition, a simple fully connected ANN is applied to a 1-dimensional list of intensities extracted from the LARBED data for comparison. According to the information of diffraction intensities, CNN determines the structure factors by matching the distribution of diffraction intensities within the diffraction discs, while the ANN aims to fit the structure factors to a list of intensities directly, without having to learn the shape of the discs in addition.

In this project, the CNN and ANN are used to determine the structure factors and the specimen thickness of a set of LARBED patterns. The LARBED patterns are generated by a quantitative diffraction simulation software pyQEDA [10] based on different parameters, including the number of reflections, tilt range, sample thickness etc.. How the diffraction factors influence the prediction performance is investigated. Before conducting prediction tasks of LARBED data, we first explore the capabilities of the networks.

# Chapter 2

## Theoretical Background

### 2.1 Electron Diffraction

This section provides an introduction to the physical basics of electron diffraction and the Bloch-wave method.

#### 2.1.1 LARBED Pattern

Many properties of materials depend on their structures and the diffraction method is an essential technique for determining the structure. Several methods – X-ray, neutron, and electron diffraction, as well as the high resolution TEM (usually in combination with X-ray diffraction) are used in crystal structure determination. [11] [12]. Comparing electrons with X-ray and neutrons, the interaction strength is on the order of 4-5 orders of magnitude stronger [4] [12]. As the amount of scattering increases, the diffracted intensities will increase, becoming comparable to the intensity of the incident beam. This process is described by the dynamical scattering theory which considers multiple scattering effects, while a simple kinematic theory is typically sufficient to interpret X-ray and neutron diffraction. This advantage allows much smaller crystals, down to the  $nm$  scale, to be analyzed by electron diffraction [11] [12].

The dynamical diffraction theory shows that the electron diffraction intensities have a highly non-linear relationship with the structure factors with increasing thickness [4] [13] [12]. This will be discussed in the section 2.1.3.

Various types of electron diffraction techniques are available for the determination of structure factors. The Select-Area Electron Diffraction (SAED) and Convergent-Beam Electron Diffraction (CBED) are two main types of them. While the incident beam in SAED is a parallel beam, the CBED is performed with a convergent beam [14] [12].

In CBED, the convergent beam produces a disc pattern instead of a spot pattern in SAED. Since the size of the diffracted area is defined by the size of aperture on the specimen, the minimal size of the diffracted area in SAED is limited to  $500\text{ nm}$  [14] [12], while the diffracted area in CBED can reach a scale of less than a nanometer by converging the incident beam onto a small spot on the specimen. However, the convergence angle of the incident beam in CBED is limited by the smallest Bragg angle for electron diffraction with high intensity, because the fast electron has a

very small wavelength. A half-convergence angle larger than the smallest Bragg angle will result in overlapping in diffraction discs, because the radius of the discs is proportional to the convergence angle [5][3].

This overlap-problem can be solved by the Large-Angle Rocking Beam Electron Diffraction (LARBED) technique, in which the diffraction data are collected at different incident beam tilts allowing a 2-dimensional intensity rocking curve to be recorded [4][5][3]. Since the LARBED pattern provides a large amount of diffraction data over an extensive angular range, it contains a wealth of information of crystals as small as a few nanometers [4][3]. Thus an accurate measurement of structure factors is, in principle, possible by analyzing LARBED data.

### 2.1.2 Fast Electrons Scattering

A relativistic treatment is necessary for electrons accelerated by a potential of 100 KeV, because the classical expression  $E = mv^2/2$  becomes inaccurate for the object having a speed of a significant fraction of the speed of light in vacuum [15].

According to the Special Theory of Relativity, the relativistic mass of a fast electron is given by

$$m = \gamma m_0, \quad (2.1)$$

where  $\gamma = 1/(1 - v^2/c^2)^{1/2}$  is the relativistic factor,  $v$  is the speed of the electron accelerated in a potential of  $V_{acc}$ .

The de Broglie wavelength defined through the Planck constant  $h$  and the relativistic momentum of the electron is then given by [16]

$$\lambda = \frac{h}{p} = \frac{h}{mv} = \frac{h}{\sqrt{2meV_{acc}}} = \frac{h}{\sqrt{2m_0eV_{acc}}} \frac{1}{\sqrt{1 + \frac{eV_{acc}}{2m_0c^2}}}. \quad (2.2)$$

As electrons pass through the crystal, they are further accelerated by the positive mean inner potential of the material. Therefore the de Broglie wavelength of the electron inside the crystal should take the mean crystal potential  $V_0$  into account. The relativistic wavelength now is

$$\lambda = \frac{h}{\sqrt{2me(V_{acc} + V_0)}} = \frac{h}{\sqrt{2m_0e(V_{acc} + V_0)}} \frac{1}{\sqrt{1 + \frac{e(V_{acc} + V_0)}{2m_0c^2}}}, \quad (2.3)$$

and the relativistic wavenumber is defined as

$$\vec{k} = \frac{1}{\lambda}. \quad (2.4)$$

As the mathematical description of electrons is given by a wave function, the problem of the scattering between a fast electron and an atom can be described by the following Schrödinger equation [17]:

$$\nabla^2 \Psi(\vec{r}) + \frac{8\pi^2 me}{h^2} [V_{acc} + V(\vec{r})] \Psi(\vec{r}) = 0, \quad (2.5)$$

where  $\Psi(\vec{r})$  is the wave function and  $e$  is the magnitude of the electron charge. From equation (2.3) and (2.4),  $|\vec{k}|^2$  is given by

$$|\vec{k}|^2 = \frac{2me}{h^2} (V_{acc} + V_0). \quad (2.6)$$

Substituting both equation (2.6) and the modified crystal potential

$$U(\vec{r}) = \frac{2me}{h^2} V(\vec{r}) \quad (2.7)$$

in equation (2.5), the Schrödinger equation can be written as

$$\nabla^2 \Psi(\vec{r}) + 4\pi^2 [|\vec{k}|^2 + U(\vec{r})] \Psi(\vec{r}) = 0. \quad (2.8)$$

For solving this Schrödinger equation, the Bloch-wave method will be applied.

### 2.1.3 The Bloch-wave Method

The atoms in a perfect crystal are arranged in a regular periodic array. The Bloch's theorem states that in a periodic potential, the solutions to the Schrödinger equation must have the form of a plane wave modulated by a periodic function. The wave functions satisfy the periodicity are called the Bloch waves [4][17][18].

#### Crystal Potential

Because of the periodicity, the crystal potential can be expanded as a Fourier series on crystal lattice [17][18]:

$$V(\vec{r}) = \sum_l V_{\vec{g}_l} \exp(2\pi i \vec{g}_l \cdot \vec{r}), \quad (2.9)$$

where  $\vec{g}_l$  is a reciprocal lattice vector (having three integer components),  $V_{\vec{g}_l}$ s are the Fourier amplitudes of the expansion and  $\vec{r}$  is the real space coordinates. From equation (2.9), it can be deduced that the condition  $V_{\vec{g}_l} = V_{-\vec{g}_l}^*$  (asterix denotes complex conjugate) should be satisfied for all  $\vec{g}_l$ s. Through inverse Fourier transformation, the Fourier coefficients of crystal potential  $V_{\vec{g}_l}$  can be written as

$$V_{\vec{g}_l} = \frac{1}{V_{cell}} \int_{cell} V(\vec{r}) \exp(-2\pi i \vec{g}_l \cdot \vec{r}) d\vec{r}, \quad (2.10)$$

where  $V_{cell}$  is the volume of unit cell. Substituting the structure amplitude

$$F_{\vec{g}_l} = \sum_i f_i^B(\vec{g}_l) \exp(-2\pi i \vec{g}_l \cdot \vec{r}_i), \quad (2.11)$$

the Fourier coefficients of crystal potential become

$$V_{\vec{g}_l} = \frac{1}{V_{cell}} \frac{h^2}{2\pi m e} F_{\vec{g}_l} = \frac{1}{V_{cell}} \frac{h^2}{2\pi m e} \sum_i f_i^B(\vec{g}_l) \exp(-2\pi i \vec{g}_l \cdot \vec{r}_i), \quad (2.12)$$

where  $f_i^B(\vec{g}_l)$  is the Born approximation atomic scattering amplitude of  $i$ th atom in the unit cell,  $\vec{r}_i$  is the position of the  $i$ th atom in the unit cell. From equation (2.7) and (2.9), a constant  $U_{\vec{g}_l}$  can be defined as

$$U_{\vec{g}_l} = \frac{2me}{h^2} V_{\vec{g}_l}, \quad (2.13)$$

which is called the structure factor. Thus the periodic potential  $U(r)$  can be represented as

$$U(\vec{r}) = \sum_l U_{\vec{g}_l} \exp(2\pi i \vec{g}_l \cdot \vec{r}), \quad (2.14)$$

for each reflection  $\vec{g}_l$  it must satisfy  $U_{\vec{g}_l} = U_{-\vec{g}_l}^*$ . Since the Fourier series describes the periodic arrangements of the atomic composition in the crystal, it also describes the crystal structure. The structure factors  $U_{\vec{g}_l}$ s are defined as the coefficients of the Fourier series, therefore they also represent the crystal structure [18].

## Wave Function

According to the Bloch's theorem, the general solution of the Schrödinger equation is a linear combination of Bloch waves. The Bloch wave is defined as

$$\psi^{(j)}(\vec{r}) = C^{(j)} \exp[2\pi i \vec{k}^{(j)} \cdot \vec{r}], \quad (2.15)$$

where  $C^{(j)}$  is the amplitude of the Bloch wave. The total wave function of the fast electron within the crystal is therefore represented as

$$\begin{aligned} \Psi(\vec{r}) &= \sum_j \alpha^{(j)} \psi^{(j)}(\vec{r}) \\ &= \sum_j \alpha^{(j)} C^{(j)} \exp[2\pi i \vec{k}^{(j)} \cdot \vec{r}], \end{aligned} \quad (2.16)$$

where  $\alpha^{(j)}$  is the excitation amplitude of  $j$ th Bloch wave  $\psi^{(j)}(\vec{r})$ . Since  $C^{(j)}$  has the periodicity of the lattice, it can be expanded as Fourier series taking a form of

$$C^{(j)} = \sum_n C_n^{(j)} \exp[2\pi i \vec{g}_n \cdot \vec{r}], \quad (2.17)$$

where  $C_n^{(j)}$  is the coefficient of  $j$ th Bloch wave. The further expanded total wave function becomes

$$\Psi(\vec{r}) = \sum_j \alpha^{(j)} \sum_n C_n^{(j)} \exp[2\pi i (\vec{k}^{(j)} + \vec{g}_n) \cdot \vec{r}], \quad (2.18)$$

thus

$$\nabla^2 \Psi(\vec{r}) = -4\pi^2 \sum_j \alpha^{(j)} \sum_n C_n^{(j)} (\vec{k}^{(j)} + \vec{g}_n)^2 \exp[2\pi i (\vec{k}^{(j)} + \vec{g}_n) \cdot \vec{r}]. \quad (2.19)$$

Substituting equation (2.14), (2.18) and (2.19) in the Schrödinger equation (2.8) yields

$$\begin{aligned} &\sum_j \alpha^{(j)} \sum_n \left[ |\vec{k}|^2 C_n^{(j)} + \sum_l U_{\vec{g}_n - \vec{g}_l} C_l^{(j)} \right] \exp[2\pi i (\vec{k}^{(j)} + \vec{g}_n) \cdot \vec{r}] \\ &= \sum_j \alpha^{(j)} \sum_n C_n^{(j)} \left[ \vec{k}^{(j)} + \vec{g}_n \right]^2 \exp[2\pi i (\vec{k}^{(j)} + \vec{g}_n) \cdot \vec{r}]. \end{aligned} \quad (2.20)$$

For each reflection  $\vec{g}_n$  and  $\vec{g}_l$ :

$$\left( |\vec{k}|^2 - \left[ \vec{k}^{(j)} + \vec{g}_n \right]^2 \right) C_n^{(j)} + \sum_l U_{\vec{g}_n - \vec{g}_l} C_l^{(j)} = 0. \quad (2.21)$$



## High Energy Approximation

For fast electrons, the mean crystal potential  $V_0$  is much smaller than the acceleration potential  $V_{acc}$  [17], thus the high energy approximation should be applied:

$$\begin{aligned} |\vec{k}| &\approx |\vec{k}^{(j)}| \\ |\vec{k}| &\gg \vec{g}_n \text{ (for fast electrons)} \\ \Rightarrow |\vec{k}| &\approx |\vec{k}^{(j)} + \vec{g}_n|. \end{aligned} \quad (2.22)$$

Since  $\vec{g}_n$  is parallel to the crystal surface, the component of  $\vec{g}_n$  in  $z$  direction is vanishing. Besides, the wavevector  $\vec{k}^{(j)}$  can be separated into a component in  $z$  direction and a component in the plane of the crystal surface:

$$\vec{k}^{(j)} = \begin{cases} \vec{k}_z^{(j)} : & \text{component in } z \text{ direction} \\ \vec{k}_t^{(j)} : & \text{component in tangential direction} \end{cases} \quad (2.23)$$

Due to the conservation of transverse momentum, each tangential component  $\vec{k}_t^{(j)}$  must equal to the tangential component of the incident wavevector  $\vec{k}_t$ :

$$\Rightarrow \vec{k}^{(j)2} = \vec{k}_z^{(j)2} + \vec{k}_t^2. \quad (2.24)$$

Putting equation (2.22) and (2.24) together with  $\vec{g}_{nz} = 0$ , we have

$$\begin{aligned} |\vec{k}|^2 - [\vec{k}^{(j)} + \vec{g}_n]^2 &= |\vec{k}|^2 - [\vec{k}^{(j)2} + \vec{g}_n^2 + 2\vec{g}_n \vec{k}_t] \\ &= (|\vec{k}|^2 - \vec{k}^{(j)2}) - (\vec{g}_n^2 + 2\vec{g}_n \vec{k}_t) \\ &\approx 2|\vec{k}|(|\vec{k}| - \vec{k}_z^{(j)}) - \vec{k}_t^2 - (\vec{g}_n^2 + 2\vec{g}_n \vec{k}_t) \\ &\approx 2|\vec{k}|(|\vec{k}| - \vec{k}_z^{(j)}) - (\vec{k}_t + \vec{g}_n)^2. \end{aligned} \quad (2.25)$$

## Diffracted Intensities

Combining equation (2.25) and (2.21), we get:

$$\left[ 2|\vec{k}|(|\vec{k}| - \vec{k}_z^{(j)}) - (\vec{k}_t + \vec{g}_n)^2 \right] C_n^{(j)} + \sum_l U_{\vec{g}_n - \vec{g}_l} C_l^{(j)} = 0. \quad (2.26)$$

For a problem of  $N$  beams, there are  $N$  such equations, which can be represented in a matrix form:

$$\frac{1}{2|\vec{k}|} \begin{pmatrix} -[\vec{k}_t + \vec{g}_0]^2 + U_0 & U_{\vec{g}_0 - \vec{g}_1} & \cdots & U_{\vec{g}_0 - \vec{g}_N} \\ U_{\vec{g}_1 - \vec{g}_0} & -[\vec{k}_t + \vec{g}_1]^2 + U_0 & \cdots & U_{\vec{g}_1 - \vec{g}_N} \\ \vdots & \vdots & \ddots & \vdots \\ U_{\vec{g}_N - \vec{g}_0} & U_{\vec{g}_N - \vec{g}_1} & \cdots & -[\vec{k}_t + \vec{g}_N]^2 + U_0 \end{pmatrix} \begin{pmatrix} C_0^{(j)} \\ C_1^{(j)} \\ \vdots \\ C_N^{(j)} \end{pmatrix} = (\vec{k}_z^{(j)} - |\vec{k}|) \begin{pmatrix} C_0^{(j)} \\ C_1^{(j)} \\ \vdots \\ C_N^{(j)} \end{pmatrix} \quad (2.27)$$

or

$$A(\vec{k}_t, U_g) C(\vec{k}_t) = \Gamma(\vec{k}_t) C(\vec{k}_t), \quad (2.28)$$

where  $A$  is a  $(N \times N)$  structure matrix at the beam tilt  $\vec{k}_t$ .  $U_g$  are all the structure factors appearing in the structure matrix  $A$ . Since the structure factors are the Fourier coefficients of the crystal potential, they are normally complex numbers.

Due to  $U_{\vec{g}_l} = U_{-\vec{g}_l}^*$ , the matrix  $A$  is Hermitian. It can be real if the crystal structure is centrosymmetric [19].  $C$  and  $\Gamma$  are the eigenvectors and eigenvalues of  $A$ , respectively. According to equation (2.18), the wave function of the electron leaving the exit surface of the crystal of thickness  $t$  is given by

$$\begin{aligned}\Psi(t, \vec{k}_t) &= \sum_j \alpha^{(j)} \sum_n C_n^{(j)} \exp [2\pi i (\vec{k}^{(j)} + \vec{g}_n)_z \cdot t] \\ &= \sum_j \alpha^{(j)} \sum_n C_n^{(j)} \exp [2\pi i \vec{k}_z^{(j)} \cdot t].\end{aligned}\quad (2.29)$$

Considering the boundary condition at the entrance surface of the crystal, we have  $\alpha^{(j)} = C_0^{(j)}$ . Hence, the wave function along reflection (lattice direction)  $\vec{g}_n$  is:

$$\begin{aligned}\Psi_{\vec{g}_n}(t, \vec{k}_t) &= \sum_j \alpha^{(j)} C_n^{(j)} \exp [2\pi i \vec{k}_z^{(j)} \cdot t] \\ &= \sum_j C_0^{(j)*} C_n^{(j)} \exp [2\pi i \vec{k}_z^{(j)} \cdot t]\end{aligned}\quad (2.30)$$

Since  $C$  is a unitary matrix, the equation (2.30) can be rewritten as

$$\begin{aligned}\Psi_{\vec{g}_n}(t, \vec{k}_t, U_g) &= \left( C^{(-1)} \exp (2\pi i \Gamma(\vec{k}_t) \cdot t) C \right)_{0,n} \\ &= \exp(2\pi i A(\vec{k}_t, U_g)t)_{0,n} \\ &= S(t, \vec{k}_t, U_g)_{0,n}.\end{aligned}\quad (2.31)$$

As a scattering matrix describing the propagation of an electron wave through the crystal of thickness  $t$  is defined as

$$S(t, \vec{k}_t, U_g) = \exp \left( 2\pi i A(\vec{k}_t, U_g)t \right) \quad (2.32)$$

[19], the diffracted intensity at the reflection  $\vec{g}_n$  is

$$I_{\vec{g}_n}(t, \vec{k}_t, U_g) = |\Psi_{\vec{g}_n}(t, \vec{k}_t, U_g)|^2 = |S(t, \vec{k}_t, U_g)_{0,n}|^2, \quad (2.33)$$

where the 0 in the index indicates that the coefficients are zero except for  $\vec{g}_0$ . The non-linearity between the diffracted intensities and the structure factors described in dynamical theory is therefore proved. The collected diffracted intensities are the diffraction pattern, which contains the information of the crystal structure. The aim of the project is to determine the structure factors  $U_g$  and the thickness  $t$  of the crystal from a given LARBED pattern.

## Structure Factors

The structure factor  $U_{g\_hkl}$  describes the amplitude and phase of a wave diffracted from crystal lattice planes characterised by Miller indices  $h, k, l$ , which is the result of all waves scattered in the direction of the  $hkl$  reflection by the  $n$  atoms contained in the unit cell:

$$\begin{aligned}U_{g\_hkl} &= |U_{g\_hkl}| \exp(i\phi_{hkl}) \\ &= \sum_i^n f_i e^{2\pi i [hx_i + ky_i + lz_i]}\end{aligned}\quad (2.34)$$

where  $x_i, y_i, z_i$  are the positional coordinates of the  $i$ th atom,  $f_i$  is the scattering factor of the  $i$ th atom, and  $\phi_{hkl}$  is the phase of the diffracted beam. The intensity of a diffracted beam is directly related to the amplitude of the structure factor. A set of lattice planes of a crystal with structure factor of 0 is called a "forbidden reflection". The intensity of the reflection vanishes [15][16].

## 2.2 Basics of Neural Networks

This section introduces the deep learning approach used in this project and its principles and algorithms.

### 2.2.1 Artificial Neural Networks

The Neural Network is one of the most powerful learning algorithms in recent years. The idea of artificial neural networks is to have computers artificially mimic the biological neural network of human brain. Artificial neural networks are structured by three types of layers, each layer is a group with a different amount of neurons connected to each other. The first layer in the neural network called the input layer, collects input patterns containing multiple features. The final layer called output layer represents the output signals. The layers between the input layer and the output layer are the hidden layers, the number of hidden layers defines the depth of a neural network. The way the neural networks are connected is called the architecture, a neural network architecture with a depth of three is shown in the Figure 2.1.

#### Perceptron

Each neuron is a simple model called the perceptron, it receives a vector of features  $x_1, \dots, x_n$  including a bias  $x_0$  (with a typical value of +1) as input and provides an output  $y$  according to its own model type defined by an activation function  $\sigma(z)$ . This process for a single perceptron is represented in Figure 2.2. First, an activation value  $z$  is computed as the sum of the linear combination of the inputs with corresponding weights  $w_0, \dots, w_n$ :

$$z = \sum_{i=0}^n x_i \cdot w_i. \quad (2.35)$$

Afterwards, by passing  $z$  to a nonlinear function  $\sigma(z)$  (called the activation function), the value of the output  $a = \sigma(z)$  will be properly limited [20].

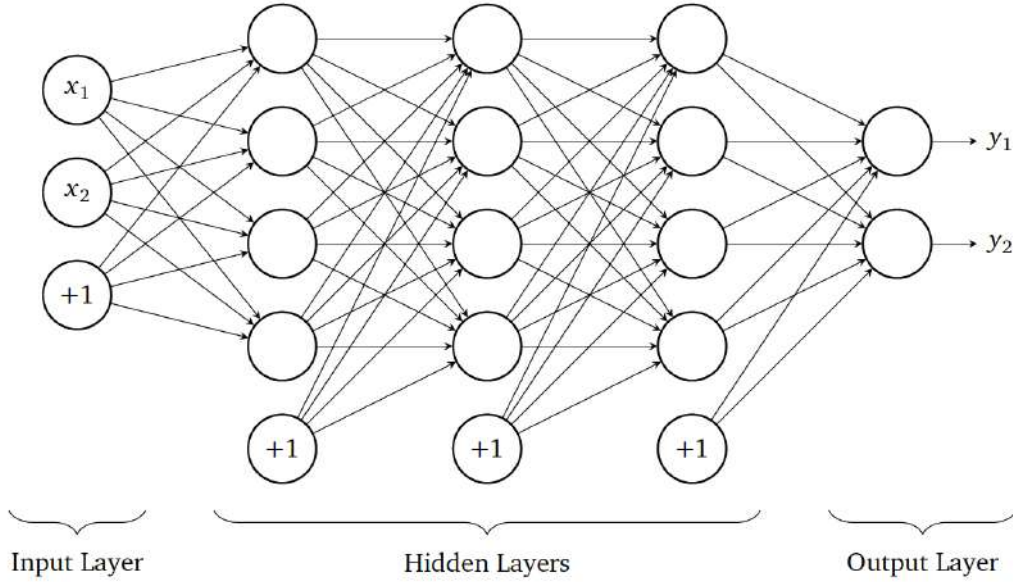


Figure 2.1: An ANN example with three hidden layers, 2 input neurons and 2 output neurons.  $+1$  denotes the bias units.

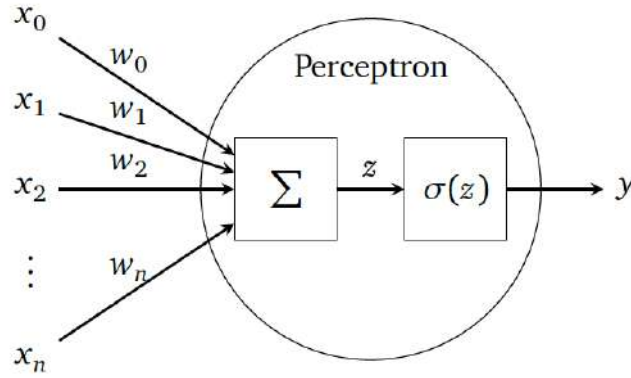


Figure 2.2: One perceptron with inputs  $x_0, x_1, \dots, x_n$  and an output  $y$ .

## Activation Functions

There are numerous activation functions to solve complex nonlinear problems. Different parts of neural networks typically use different types of activation functions. The choice for an activation function in the output layer depends on the prediction, while the choice in hidden layers is more dependent upon the architecture of neural network and the type of input values. Rectified Linear Unit (*ReLU*) function (Figure 2.3 (a)) is the most commonly used activation function in neural networks, it is actually a simple max function:  $\max(0, z)$ . Any negative output value will be treated as zero through *ReLU* activation function. Hyperbolic Tangent (*Tanh*) function has a "S-shape" shown in the Figure 2.3 (b), it takes any real value as input and maps to  $[-1, 1]$ . In this project, a *ReLU* function is chosen as the activation function in the hidden layers of the CNN and the *Tanh* activation function is used in the ANN and in the output layer in the CNN.

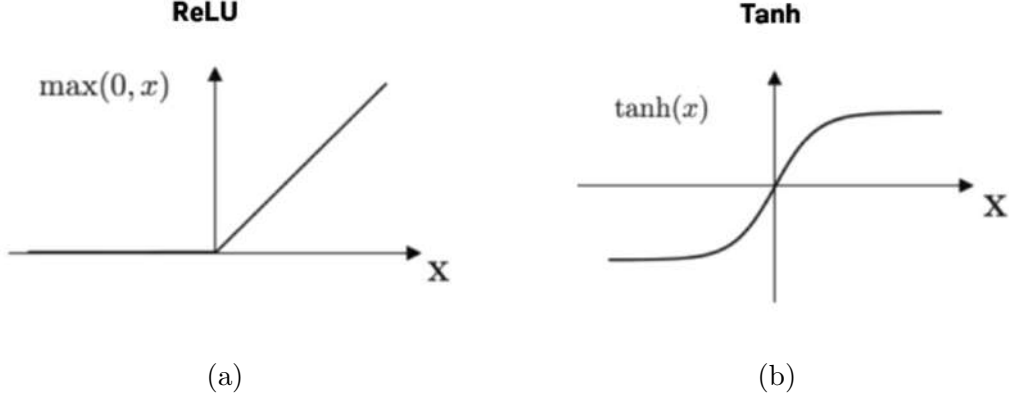


Figure 2.3: Activation functions: a) *ReLU* b) *Tanh*.

### Forward Propagation

After giving inputs with initial weights in the input layer, each neuron at each layer is able to give an output limited by an activation function. The outputs at the previous layer become the inputs of the current hidden layer. The final output at the last layer is the neural network's estimation. This process is called the forward propagation.

### Cost Function

Since the neural networks employed in this work implement a supervised learning process, the training sample should be a pair consisting of an input and a desired output. Suppose there are  $m$  pairs of training samples  $(x^{(i)}, y^{(i)})$ , where the input vector  $x^{(i)}$  is  $L$  dimensional and its corresponding label  $y^{(i)}$  ( $1 \leq i \leq m$ ) is a  $K$  dimensional vector, which also means that the number of neurons at the input and output layer should be  $L$  and  $K$  respectively. For given training set  $x$  ( $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ ), the neural network will give vectors  $h_{\theta}(x)$  that are also  $K$  dimensional. A cost function is needed to compute the difference of the network's output  $h_{\theta}(x)$  from the true value  $y$ .  $\theta$  denotes the parameters or the weights in the neural network. One very common cost function is the Mean Square Error (MSE), which calculates the averaged squared difference between the true values against the predicted values:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \|h_{\theta}(x^{(i)}) - y^{(i)}\|^2. \quad (2.36)$$

Where  $m$  is the number of training samples. The cost function  $J(\theta)$  depends on the current weights  $\theta$  in the neural network. By minimizing  $J(\theta)$ , it's possible to determine the optimal parameters  $\theta$ .

### Backpropagation

The backpropagation algorithm is an algorithm to find the parameters  $\theta$  which minimize the cost function  $J(\theta)$ . For each node  $j$  in layer  $l$ , there is an error term  $\delta_j^{(l)}$  to measure the attribution of node  $j$  to the error in the output. In contrast to

the forward propagation, backpropagation starts by computing the error term for the output layer, and then calculates the error of each layer backward. The error term in an output node  $j$  can be directly calculated by the difference between the network's final output  $h_\theta(x)_j$  and the true target value  $y_j$ . For nodes in a hidden layer  $l$ , the error term should be calculated by the weighted average of the error terms of the nodes in layer  $l + 1$  [21]:

$$\begin{cases} \delta_j^{(l)} = a_j^{(l)} - y_j = h_\theta(x)_j - y_j & \text{for node } j \text{ in output layer } l \\ \delta_j^{(l)} = \left( \sum_{k=1}^q \delta_k^{(l+1)} \cdot \theta_{jk}^{(l)} \right) \cdot \sigma'_l(z_j^{(l)}) & \text{for node } j \text{ in hidden layer } l \end{cases} \quad (2.37)$$

Where  $a_j^{(l)}$  is the output of node  $j$  in the layer  $l$ .  $\theta_{jk}^{(l)}$  is a weight acting on the layer  $l$ , where  $j$  describes the unit in the layer  $l$  and  $k$  describes the unit in the layer  $l + 1$ .  $q$  is the number of units in layer  $l + 1$ .  $z_j^{(l)}$  is the activation value of node  $j$  in layer  $l$  calculated by equation (2.35).  $\sigma_l$  is an activation function defined in layer  $l$ .

In order to use the gradient descend method or one of the advanced optimization algorithm, we need to compute the gradient of  $J(\theta)$  with respect to all the parameters in  $\theta$ . It can be mathematically proved that the gradient terms are given by:

$$\frac{\partial}{\partial \theta_{jk}^{(l)}} J(\theta) = a_j^{(l)} \cdot \delta_k^{(l+1)}. \quad (2.38)$$

These partial derivatives can be now used to update all weights in the neural networks:

$$\theta_{jk}^{(l)} = \theta_{jk}^{(l)} - \eta \frac{\partial}{\partial \theta_{jk}^{(l)}} J(\theta), \quad (2.39)$$

where  $\eta$  is the learning rate. Take a simple convex cost function with only one parameter  $\theta$  as example (Figure 2.4), what the gradient descent method does is: it starts at one point on the cost function, calculates the slope of this point and then moves a constant step in the downward direction of the slope and repeats until the slope converges to zero. It takes longer for a smaller learning rate to find the optimal parameter which minimizes the cost function, while a larger one is faster but might miss the minimum. On the other hand, an adaptive gradient descent method is more efficient to find the minimum by taking larger steps at the beginning and then taking smaller steps as the gradient gets closer to zero. One of the most popular adaptive gradient descent algorithms is Adam, which is used in this project to train both the ANN and CNN model. Adam method computes individual adaptive learning rates for different parameters by estimating both first and second moments of the gradients [22].

Another advantage of Adam algorithm is that it is an extension of stochastic gradient descent. In contrast to the batch gradient descent, stochastic gradient descent doesn't need to scan all the training samples in every iteration of weights updating, it scans only one single training sample or a mini-batch training samples. Furthermore, there can be multiple local minimums for a more complex problem. As the batch gradient descent tends to find the most reasonably trajectory to get to the minimum by looking at all the training data in every iteration, it can get stuck at a local minimum. In contrast, by Adam, through shuffling the mini-batches of training data, the weights updating starts at a random point as shown in Figure 2.5. By repeating this by several epochs, we find trajectories starting at different

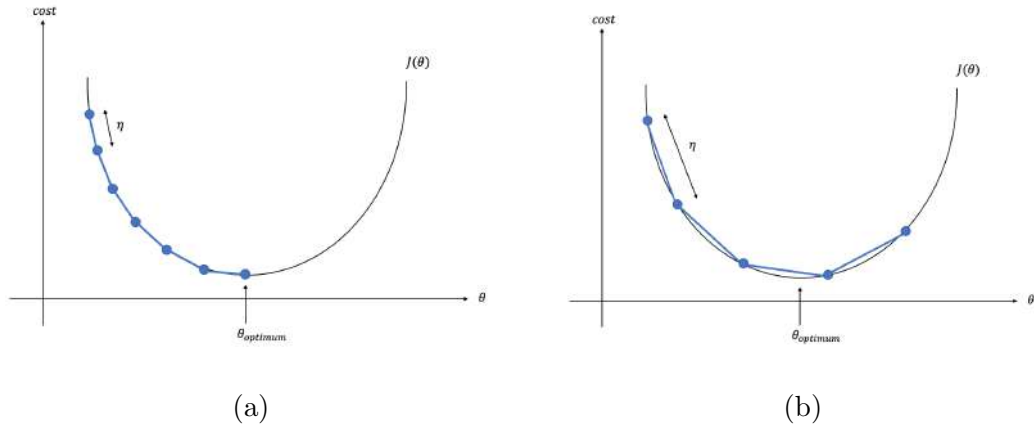


Figure 2.4: A simple cost function with a single parameter  $\theta$ : a) with a small learning rate b) with a large learning rate.

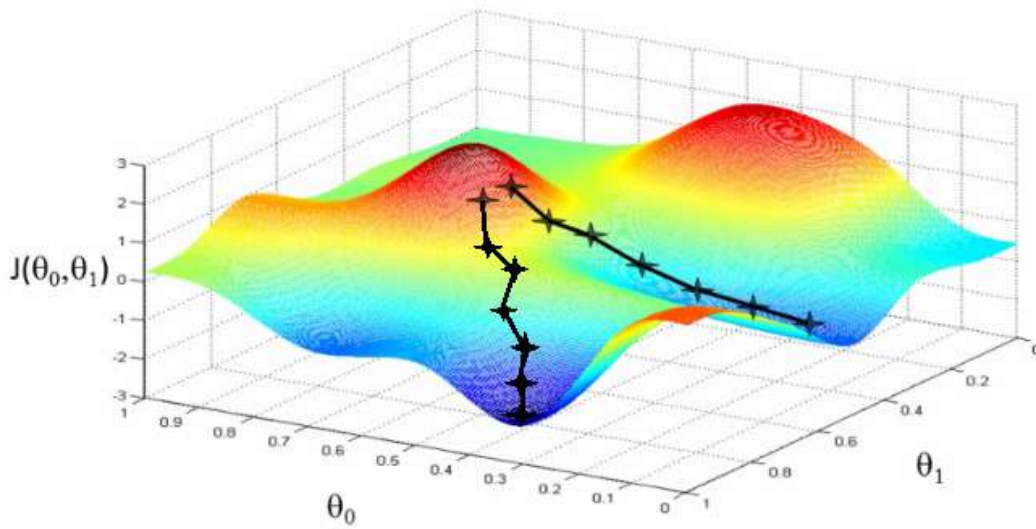


Figure 2.5: Gradient descent with two parameters  $\theta_0$  and  $\theta_1$ : converging at different minimum from different starting points [23].

random points. It is therefore more likely to find the parameters describing the most efficient direction towards the global minimum [24].

## 2.2.2 Convolutional Neural Networks

The Convolutional Neural Network (CNN) is a specific type of artificial neural network generally used for image recognition. A CNN contains three types of layers known as convolution layers, pooling layers and fully connected layers. An image is coded as a matrix of pixels. The CNN extracts multiple important features of the image by applying functions to the matrix through different layers.

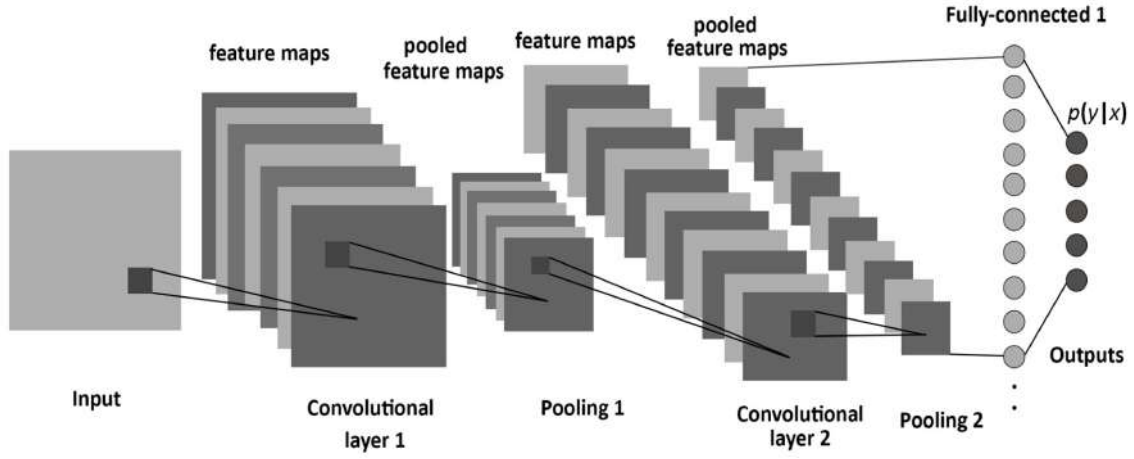


Figure 2.6: Convolutional neural network architecture. A CNN consists of several convolution and pooling layers, a pooling layer is always applied after the convolution layer; the fully connected layer is found at the end of the CNN architecture [25].

### Convolution Layer

The convolution operation is one of the fundamental blocks of a CNN. In a convolution layer, a number of matrices called kernels or filters are constructed. A convolution operation is performed between the input image and each filter, a new image is produced as output. The size of the output feature map is determined by the hyperparameters: kernel size, stride and padding [26]:

- Kernel Size ( $F$ ): The width of the kernel.
- Stride ( $S$ ): The step the filter jumps at a time.
- Padding ( $P$ ): Padding, or most commonly zero-padding is the process of adding  $P$  zeros to each side of the boundaries of the input image. The size of zero-padding controls the spatial size of the output.

Given  $I$  as the width of input, the output size  $O$  of the feature map is then given by

$$O = (I - F + 2P)/S + 1. \quad (2.40)$$

Other important filter hyperparameters are:

- Depth of Filter ( $C$ ): The depth of a filter must match the depth of the input image, which means a filter with width  $F$  applied to an input image containing  $C$  color channels must have a size of  $F \times F \times C$ .
- Number of Filters ( $K$ ): the number of filters  $K$  determines the number of feature maps to be generated. Given an input image of size  $I \times I \times C$  and  $K$  filters of size  $F \times F \times C$ , the output feature map should have a size of  $O \times O \times K$ .

In the backpropagation of a CNN, the trainable parameters are exactly the numbers in the filters. Since each filter is a  $F \times F \times C$  matrix of trainable weights, the number of parameters to learn in a convolution layer that contains  $K$  filters is given by

$$\text{number of parameters} = (F \times F \times C + 1) \cdot K, \quad (2.41)$$



where the 1 is to include the biases. The number of trainable parameters can be used to evaluate the complexity of the model.

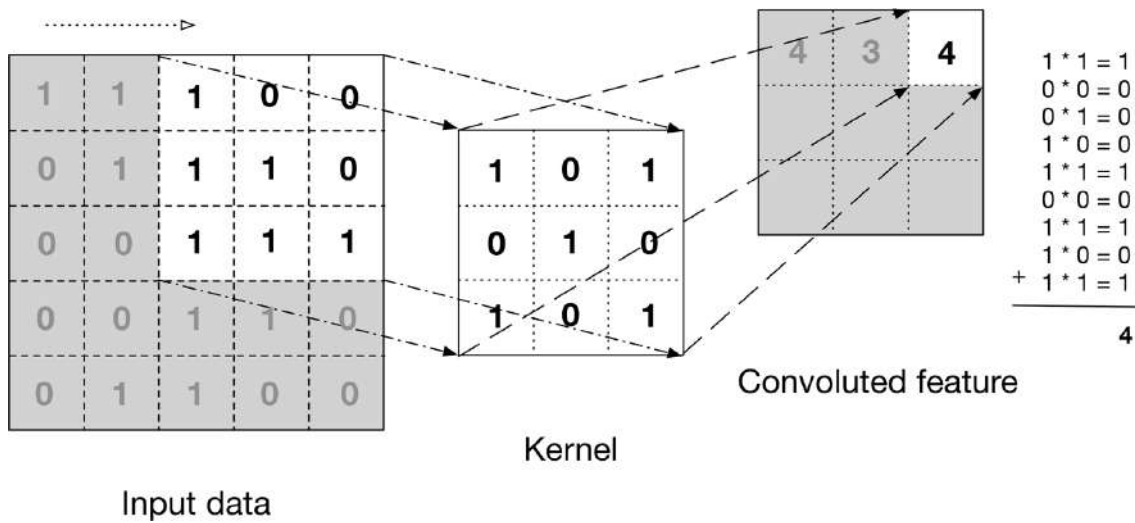


Figure 2.7: Convolution operation with stride 1 without padding, a  $5 \times 5$  input  $I$  will lead to a  $3 \times 3$  output  $O$  [27].

Figure 2.7 illustrates this process. The left matrix of size  $5 \times 5$  is the input data, which represents a simplified two dimensional grayscale image. The 1s in the matrix give brighter pixel intensity values and the 0s give darker pixel intensity values. A filter of size  $3 \times 3$  is applied to the input image, with a stride of 1 we move the kernel one pixel each time. The right matrix is the resulting output which is a  $3 \times 3$  matrix. By using a default activation function of *ReLU*, the non-negative values remain, while the negative values become 0. There are  $(3 \times 3 \times 1 + 1) \cdot 1 = 10$  parameters used in this process.

## Pooling Layer

A pooling layer is often applied after a convolution layer. It can be used to reduce the size of the output feature map while keeping the important information. A pooling layer accelerates the computation while establishing spatial invariance. Max- and average-pooling are two most common kinds of pooling, which compute the maximum or average value of a current field of the feature map. In contrast to convolution layer, the pooling layer has no activation function or weights to learn. A simple example is given by the Figure 2.8: a  $4 \times 4$  feature map separated into 4 different regions is taken as input. For both max- and average pooling, a filter of size  $2 \times 2$  with a stride of size  $2 \times 2$  is used to downsample the feature map. Each output is exactly the max or the average value of the corresponding colored region.

## Fully Connected Layer

Finally, towards the end of the CNN architecture, the feature map will be flattened into a string, which becomes the input of the fully connected layers, as shown

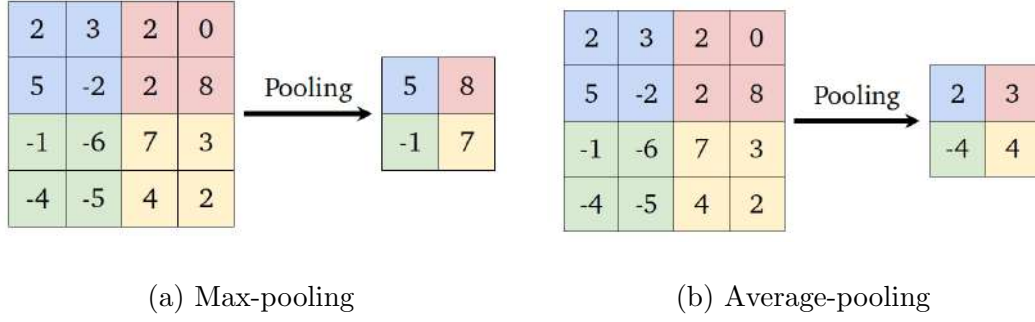


Figure 2.8: Pooling: a) max-pooling with a filter size of  $2 \times 2$  and a stride of  $2 \times 2$ ; b) average-pooling with a filter size of  $2 \times 2$  and a stride of  $2 \times 2$ .

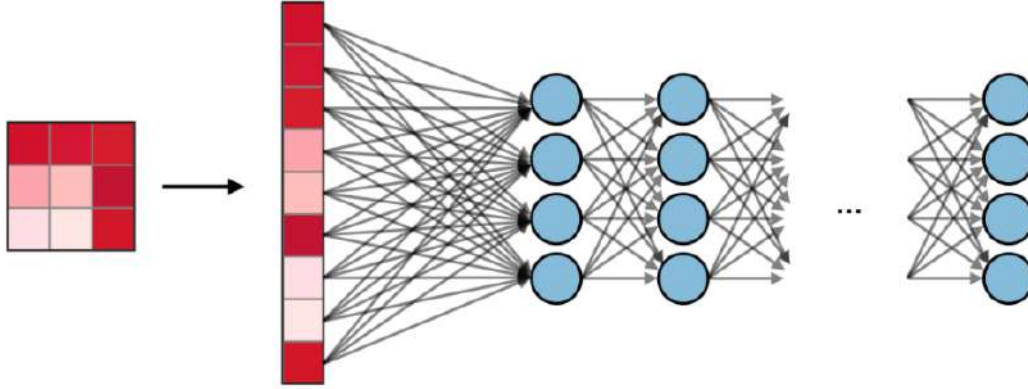


Figure 2.9: Fully connected layer. A  $3 \times 3$  feature map has been flattened inputted into the fully connected layers [28].

in Figure 2.9. The size of the final output of a CNN is the number of the output neurons.

## Residual Network

Theoretically, a deeper neural network can solve more complex task with a higher accuracy. But in reality, as the number of layers increases, the training error will no longer decrease but instead increase (shown in Figure 2.10). This accuracy degradation is due to the vanishing gradient problem. As the gradient is updated through backpropagation, the gradient in first layers may become extremely small due to repeated multiplication operation. Residual Networks or ResNets, are developed by He et al. [29] to address this degradation problem in a deep network. The training performance in a ResNet improves as the layer number increases.

Assuming that a shallow network has reached the saturation accuracy, adding identity mapping layer (the output equals input) will increase the depth of the network without increasing the training error. This is the main idea of the ResNet. The problem now is to learn the identity mapping function. It is difficult to fit a potential identity mapping function  $H(x) = x$  directly. However, if the network is designed as  $H(x) = F(x) + x$  (Figure 2.11), we can convert this problem into learning a residual function  $F(x) = H(x) - x$ . As long as  $F(x) = 0$ , it constitutes an identity mapping  $H(x) = x$  [29].

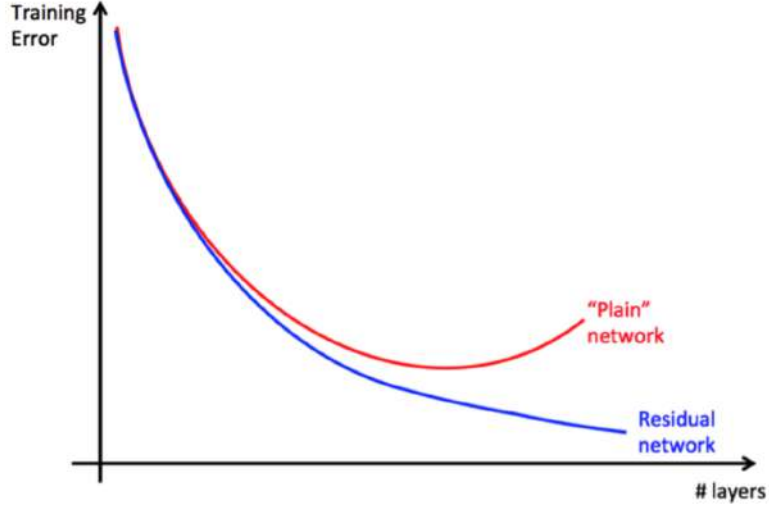


Figure 2.10: Training error with respect to the number of layers in a plain network (red) and in a residual network (blue).

Residual block with shortcut connections in ResNet realizes the residual learning. An example of residual block is shown in the Figure 2.11. Assuming that the input of this residual block is  $x$ . The output result of the block is  $H(x) = F(x) + x$ , where  $F(x)$  is the output of the stacked layers. When the accuracy has reached the saturation (which means the input  $x$  is approximate to the output  $H(x)$ ), then the identity mapping is optimal and it will push the residual  $F(x)$  to zero. In this way, ResNet no longer learns a complex unreferenced function, but a residual function with respect to the layer input, which is much easier to optimize. Through shortcut connections,  $x$  is able to skip one or more layers and reaches a deeper layer. It solves the degradation problem in the case of extremely deep neural networks [30].

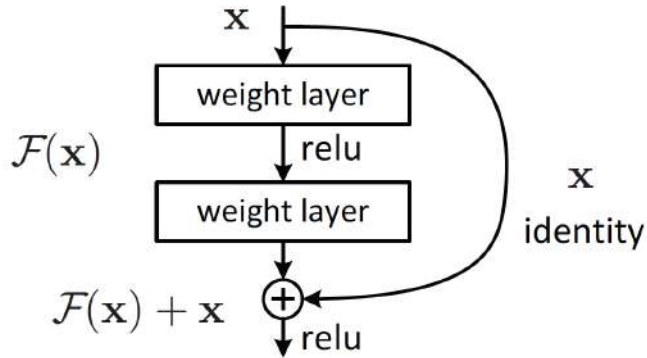


Figure 2.11: Residual learning in a block [29]

ResNet-50 is a 50-layer deep ResNet. It contains 48 convolution layers along with 1 max-pooling layer located at the front and 1 average-pooling layer located at the end of the network. In a ResNet-50, a stack of three convolution layers having the size of  $1 \times 1$ ,  $3 \times 3$ , and  $1 \times 1$  are used in a residual block (shown in Figure 2.12).

The architecture of a ResNet-50 can be described as  $CB^1 - MP - CB^2 - CB^3 - CB^4 - CB^5 - AP - SM$ , where  $CB^i$  denotes the  $i$ th convolution block,  $MP$  and  $AP$  denote the max-pooling layer and the average-pooling layer respectively. A softmax

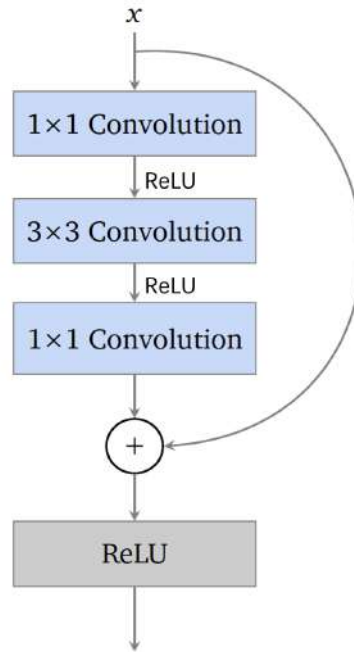


Figure 2.12: A residual block in ResNet-50. The first and second  $1 \times 1$  convolution are used to reduce and increase the dimension of feature maps respectively [29].

layer is denoted as  $SM$  [31]. Each convolution block  $CB^i$  contains several residual blocks.

# Chapter 3

## Methodology

### 3.1 Generation of Dataset

For supervised learning, a set of labeled data containing both input(features) and output(labels) is necessary for training. Datasets of LARBED patterns with structure factors were generated through pyQEDA. For our regression problem, the label should be a combination of structure factor vector and the sample thickness; the features should be the diffraction intensities. The details of the datasets are described in this section.

#### 3.1.1 pyQEDA

The pyQEDA program realizes the simulation of dynamical scattering by using the Bloch-wave method. The code was developed by Prof. Christoph T. Koch and wrapped for python by Sam Fairman and Sherjeel Shabih [10]. It has 2 working modes, the simulation mode has been used in this project.

How a diffraction pattern actually looks like depends on both the diffraction geometric conditions and the crystal sample (which is related to the crystal structure and sample thickness). Under simulation mode, the geometrical parameters of the experimental set-up including the accelerating energy, zone axis, and tilt range etc., will be given from an input parameter file. The pyQEDA can be set to

- **RandomUg Channel:** produce the LARBED pattern with random structure factors and thickness;
- **ComputeUg Channel:** calculate the LARBED pattern with the correct structure factors calculated according to the crystal file;
- **InputUg Channel:** simulate a LARBED pattern by an input structure factor vector and thickness.

#### 3.1.2 Data

$SrTiO_3$  (Strontium Titanate) crystals and  $ZnO$  (Zinc Oxide) crystals are tested throughout this project, which are centrosymmetric and non-centrosymmetric crys-

tals respectively. Several different parameter files are used to figure out the suitability and robustness of the method under different situations. The parameter files are given in the Table 3.1. Diffraction patterns of  $SrTiO_3$  and  $ZnO$  with the thickness

File Name	Number of Beams	Number of Tilts	Thickness ( $nm$ )	Tilt Range ( $mrad$ )	Voltage (KV)
$SrTiO_3 - 1$	13	441	30	70	120
$SrTiO_3 - 2$	13	441	30	100	120
$SrTiO_3 - 3$	13	441	15	100	120
$SrTiO_3 - 4$	13	797	30	70	120
$SrTiO_3 - 5$	13	797	30	40	120
$SrTiO_3 - 6$	13	1009	30	100	120
$SrTiO_3 - 7$	13	1009	15	100	120
$SrTiO_3 - 8$	25	441	30	70	120
$SrTiO_3 - 9$	25	441	15	100	120
$SrTiO_3 - 10$	69	441	30	70	120
$ZnO - 1$	95	197	50	30	100
$ZnO - 2$	97	253	50	80	60

Table 3.1: parameter files

of 15, 30 and 50  $nm$  are simulated under an accelerating voltage of 120, 100 and 60 KV. The numbers of beams (diffraction spots) tested are  $\{13, 25, 69, 95, 97\}$ . Different numbers of beam tilts are tested, which are  $\{197, 253, 441, 797, 1009\}$ . Five maximum tilt range are tested including  $\{30 \text{ } mrad, 40 \text{ } mrad, 70 \text{ } mrad, 80 \text{ } mrad, 100 \text{ } mrad\}$ .

For each parameter file, a pseudo-experimental pattern with its correct structure factors will be computed. In addition, a dataset of random LARBED patterns with their structure factors will be generated. All pseudo-experimental patterns and random patterns are oriented near the (001) zone axis.

## Pseudo-Experimental Data

The experimental data is simulated by pyQEDA (all "experimental data" mentioned below are "pseudo-experimental data"). A 2D LARBED pattern  $I^{exp}$  and its corresponding structure factor vector  $U_g^{exp} = [U_{g_1}^{exp}, U_{g_2}^{exp}, \dots, U_{g_n}^{exp}]$  with the crystal thickness  $t^{exp}$  are computed through pyQEDA by setting the channel to **ComputeUg**. This pair of  $\{I^{exp}, (U_g^{exp}, t^{exp})\}$  is the experimental data, where  $(U_g^{exp}, t^{exp})$  denotes the combination of  $U_g^{exp}$  and  $t^{exp}$ .

The structure factors of  $U_g^{exp}$  are automatically calculated as Hermitian. The intensities in the experimental pattern are normalized during the generation. The thickness of the experimental pattern is read directly from the parameter file, with a unit of Angstrom ( $\text{\AA}$ ).

## Random Datasets

A number of structure factor vectors  $\{U_g^{(1)}, U_g^{(2)}, \dots, U_g^{(n)}\}$  and their corresponding LARBED patterns  $\{I^{(1)}, I^{(2)}, \dots, I^{(n)}\}$  could be randomly generated through pyQEDA

in **RandomUg** channel. The thickness  $t^{(i)}$  of each pattern is randomly selected from a range of 5-100 nm.  $\{I^{(i)}, (U_g^{(i)}, t^{(i)})\}$  is a pair of data. There are totally 140 random data samples in each random dataset.

To normalize the intensities of the random LARBED patterns (to achieve same scale of the experimental pattern), the randomly generated structure factors are ensured to be Hermitian for each reflection. This is achieved by a **normalization function**, which finds the opposite reflection of each reflection and sets their corresponding structure factors ensuring  $U_g = U_{-g}^*$ .

It should be noted that there will be several combinations of structure factors showing exactly the same LARBED pattern, because the diffraction intensities are insensitive to the choice of unit cell origin [7]. To ensure a correct prediction, all random patterns need to be set to the same origin, therefore it is necessary to apply a function to make the possibility unique. A simple way is to define a unit cell origin by making the phases of two selected reference beams equal to a same value. This is realized by a **makeUniqueUgVector function** written by Prof. Christoph T. Koch. The function works in the following steps:

1. Select two reference structure factors  $U_{g1}$  and  $U_{g2}$  whose reflections  $\vec{g}_1$  and  $\vec{g}_2$  are non-collinear. Shift the origin of the unit cell such that the phases of the  $U_{g1}$  and  $U_{g2}$  are set to the same value.
2. Calculate the shift  $r_0$  by solving the linear equations:

$$\begin{aligned} 2\pi(g_{1x}r_{0,1} + g_{1y}r_{0,1}) &= \Delta\phi_1 \\ 2\pi(g_{2x}r_{0,2} + g_{2y}r_{0,2}) &= \Delta\phi_2 \end{aligned} \quad (3.1)$$

where  $\vec{g}_1 = [g_{1x}, g_{1y}]$  and  $\vec{g}_2 = [g_{2x}, g_{2y}]$  are the non-collinear 2D Laue zone reflections for  $U_{g1}$  and  $U_{g2}$ . The difference between phases  $\Delta\phi_g = \phi_{g,target} - \phi_g$  is computed from the equation

$$\frac{\exp[i\phi_{g,target}]}{U_g} = \frac{\exp[i\phi_{g,target}]}{|U_g| \exp[i\phi_g]} = \frac{1}{|U_g|} \exp[i\Delta\phi_g]. \quad (3.2)$$

where  $\phi_{1,target}$  and  $\phi_{2,target}$  are the target values of the phases of  $U_{g1}$  and  $U_{g2}$  (default by 0).

3. Apply the same shift  $r_0$  that has been applied to unit cell origin to all structure factors:

$$U_{g,unique} = U_g \exp[2\pi i(g_x r_{0,1} + g_y r_{0,2})]. \quad (3.3)$$

This set of structure factors  $U_{g,unique}$  is unique.

In this process, the phases of all other beams are changed according to the change in unit cell origin. The diffraction intensities that pyQEDA computes from a modified  $U_{g,unique}$  is identical to the one computed from the origin  $U_g$ . This **makeUniqueUgVector function** is applied to each random structure factor vector processed by the **normalization function**.

### 3.1.3 Data Preprocessing

Before feeding data into a network, a preprocessing step is necessary. The diffraction intensities were normalized to 0 – 1 by ensuring all random structure factors generated by pyQEDA Hermitian. All thicknesses were scaled to 0 – 1 by dividing 1000 Å, because they were randomly distributed in the range of 50 - 1000 Å. Both imaginary part and real part of the structure factors were limited to a maximum value of  $\pm 0.2$  during the generation. Since the structure factors were generally complex numbers, to properly use the MSE function to minimize the loss during training, each structure factor vector  $U_g^{(i)}$  was represented by two real numbers (real and imaginary part), and further joined with the normalized thickness  $t^{(i)}$ . The final form of the input is a one-dimensional real vector denoted as  $U_g t^{(i)}$ . For a LARBED pattern produced by  $N_b$  electron beams, there are totally  $N_b$  complex structure factors reformed into a  $U_g t$  vector, which has the size of  $2N_b + 1$ .

The pre-processed experimental data and the random dataset are denoted as  $\{I^{exp}, U_g t^{exp}\}$  and  $\{I^{(i)}, U_g t^{(i)}\}$ . They are now in a form that the neural networks can directly accept.

## 3.2 Neural Networks Design

This section presents two versions of model architectures for the prediction tasks. The models are realized by TensorFlow and developed in Python. All training and testing are run by a 6-core 3.2GHz Intel Core i7 CPU.

### 3.2.1 CNN

Some other CNN architectures, such as VGG-16 and simple CNN were tested before concluding that the ResNet-50 produces the highest accuracy. The ResNet-50 architecture was applied with the method of transfer learning. The transfer learning transfers the weights from a previously trained model to achieve better learning results faster. Since the pre-trained model is learned on large datasets with millions of images, the weights are well initialized for training a new smaller dataset [32].

Since the ResNet-50 is initialized for classification, for solving the regression problem, the last soft-max layer used to conduct classification was removed and replaced with a fully connected layer followed by 4 dense layers with *Tanh* activation functions. Figure 3.1 demonstrates the final ResNet-50 architecture, which can be simply denoted as  $CB^1 - MP - CB^2 - CB^3 - CB^4 - CB^5 - AP - FC$ . Each convolution Block  $CB^i$  contains different numbers of residual blocks (which was introduced in Section 2.2).  $CB^1$  contains only conv1, which is a convolution layer with 64 different  $7 \times 7$  kernels and a stride of size 2;  $MP$  is a max pooling of 64



kernels with stride size of 2; In  $CB^2$ , there are 3 residual blocks denoted as conv2-1, conv2-2 and conv2-3, there is a  $1 \times 1$ , 64 kernels convolution layer, followed by a  $3 \times 3$ , 64 kernels convolution layer, and a convolution layer with  $1 \times 1$ , 256 kernels; etc.. After extensive tests, we determine the optimal number of fully connected layers to achieve the highest accuracy to be 4. Therefore 4 layers are applied in the last  $FC$  block before regression. The number of neurons are selected as  $16 \times \#targets$ ,  $4 \times \#targets$ ,  $\#targets$  and  $\#targets$  in each dense layer, where  $\#targets$  is the number of targets in the final output layer, i.e., the size of  $U_g-t$ .

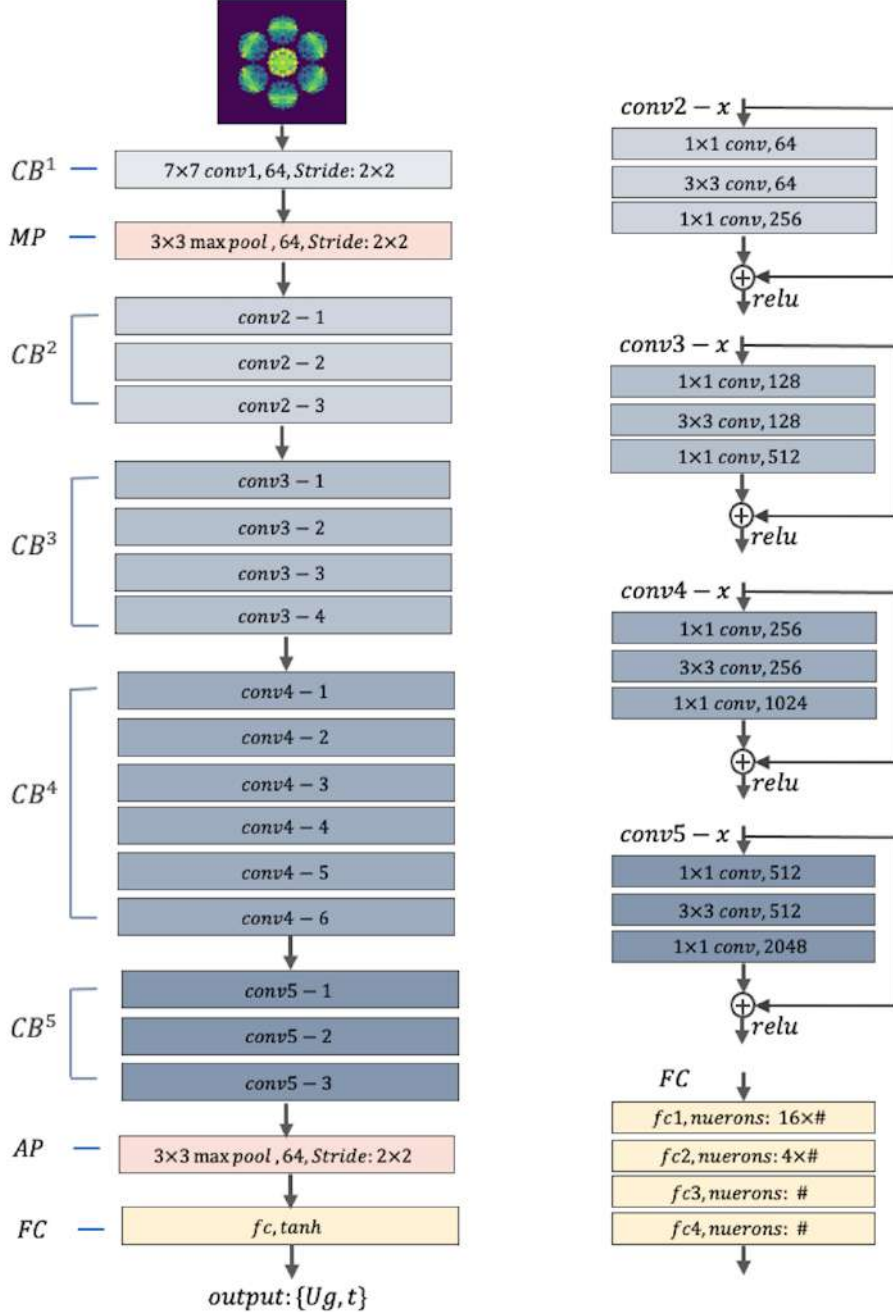


Figure 3.1: ResNet-50 architecture used for regression.

The first 49 layers of the basic ResNet-50 model were frozen to keep the previous trained weights unchanged during the training process. Only the dense layers added on top of the frozen layers are trainable. They will further train and update the

weights based on this project's dataset.

Since the ResNet is initialized to accept 3-channel input, the grayscale LARBED patterns will be converted to 3 Channel images to meet the input requirements. The last fully connected layer for regression was defined in terms of the number of targets specific to each LARBED pattern. Take a *ZnO* pattern with  $70 \times 70$  pixels as example, which was produced by 95 beams. A  $70 \times 70 \times 3$  -pixel input will be fed into the network and a vector with a size of 191 will be given in the final output layer. The CNN network has totally 58,849,583 parameters (weights) to train.

### 3.2.2 ANN

CNN determines the structure factors by learning the round shape and edge of the diffraction discs. On the other hand, instead of feeding the diffraction patterns in a CNN, a more straightforward way is to feed the intensities in a fully connected ANN.

In the ANN, a five-layer model was used (as shown in Figure 3.2), which contains three hidden layers. The size of input and output layers were determined by the number of intensities and structure factors regarding to each LARBED pattern. The number of hidden neurons should be between the size of the input layer and the size of the output layer. We set  $2/3$  of the input sample size as the number of neurons in the first layer, and  $1/3$  of the input sample size for the second layer. The last hidden layer has a same number of neurons as the output layer. All hidden layers and the output layer use *Tanh* function as activation function.

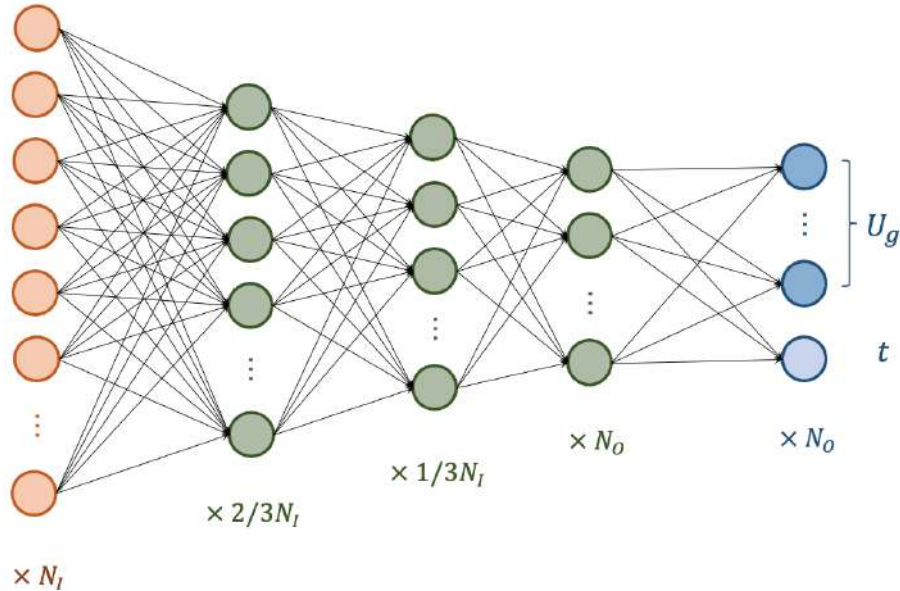


Figure 3.2: ANN architecture for the regression. A five-layer model was used, it contains an input layer of  $N_I$  neurons; 3 hidden layers having  $2/3 N_I$ ,  $1/3 N_I$  and  $N_O$  neurons with *Tanh* activation function; a final output layer of  $N_O$  neurons with a *Tanh* activation function.

A LARBED pattern is constructed by collecting a set of diffraction patterns

recorded at a discrete set of beam tilts on a disc. For a LARBED pattern produced by  $N_b$  electron beams and  $N_g$  beam tilts,  $N_b \times N_g$  intensities are collected. These intensities are integrated based on the corresponding diffraction peak at the incident beam tilt [13][6]. A 2D-table of intensities can be created with each tilt and its corresponding structure factor. The input of the ANN could be either the intensities in the LARBED pattern or the 2D-table of intensities.

### 3.3 The Training

The neural network is trained by comparing the output of the neural network with the true label. The MSE loss function is applied to measure the difference between the prediction and true value. The weights are updated through error backpropagation via Adam algorithm. The choice of hyperparameters including the learning rate, epochs and batch size are essential for a good training performance [33]. After testing different parameter settings, a default setup used for the training is shown in Table 3.2. The initial learning rate is set to be  $10^{-3}$ , Adam will compute the adaptive learning rates for each weight during training. For both CNN and

Model	Learning rate	Batch size	Epochs	Epochs in loop
CNN	$10^{-3}$	20	20	5
ANN	$10^{-3}$	20	30	5

Table 3.2: Default hyperparameter for training and testing.

ANN, the number of epochs for the initial training is determined as the slope of the training loss reaching a plateau. A smaller number of epochs 5 is applied in the loop for further iterative training. The batch size of ANN is fine tuned to 20. As the performance of ResNet-50 was tested to be less dependent on the batch size, the batch size of 20 is used in CNN.

In both tasks, 100 pairs of random samples are used as training data. For testing the learned model, another 40 samples are used.

#### 3.3.1 Task 1: $U_g$ Deviation

For a given experimental pattern, we want to determine a set of accurate structure factors through randomly generated diffraction data. However, after testing, our network model isn't able to give a correct prediction from the random diffraction data which is far deviated from the true value. Therefore, we want to explore the capability of the network, i.e. measuring the degree of deviation of the diffraction data from the correct value, by which the network is no longer able to give a correct prediction.

In this task, the pseudo-experimental data is used as target data. Instead of starting with random guess of the structure factors, we start with some random numbers which are close to the correct solution. By adding an increasing small random offset to the correct  $U_g$ , the diffraction data are gradually deviated from the target data. By calculating the difference between the predicted result of the network and the correct value, we can estimate how sensitive the prediction accuracy is to the random offsets of the  $U_g$ .

The following steps are executed:

1. **Experimental data:** Get the pseudo-experimental data  $\{I^{exp}, U_g t^{exp}\}$  as target data.
2. **Deviation:** Through adding a small random offset to every element of the correct structure factor vector  $U_g^{exp}$ , we get a deviated structure factor vector  $U_{gdev}^{exp}$  close to the correct answer  $U_g^{exp}$ . In order to ensure that the forbidden reflections can still be found, each random offset is set to be proportional to the amplitude of the structure factor to keep the structure factors of the forbidden reflections at zero. The variable of thickness is also deviated randomly by a few *nms*.
3. **Normalization:** After adding the small random offset, the structure factors are symmetrized by applying the **Normalization function**. The **Normalization function** ensures that the reflections of opposite *hkl* Miller indicies have Hermitian structure factors, i.e.  $U_g = U_{-g}^*$ .
4. **Generate training data:** A corresponding pattern  $I_{dev}^{exp}$  is simulated by inputting this  $U_{gdev}^{exp}$  into pyQEDA (in **InputUg Channel**). Such a pair of  $\{I_{dev}^{exp}, U_g t_{dev}^{exp}\}$  is a new sample. 140 data including 100 training data and 40 testing data are generated. It should be noted that the selection of the offset for each data is randomly selected under the same scale, that ensures no two patterns in the training set are same, while their degree of deviation are same.
5. **Training:** The CNN and ANN trained and tested on these 140 data can now be used to predict a set of structure factors and thickness vector  $U_g t_{pred}^{exp}$  of the experimental pattern  $I^{exp}$ . By comparing the pattern  $I_{pred}^{exp}$  simulated from the predicted  $U_g t_{pred}^{exp}$  with the experimental pattern  $I^{exp}$ , we know whether a deviation scale is able to return a correct solution of structure factors.

The above process will be repeated for each different deviation scale. The deviation scale is initialized at a very small value, it gets larger and larger until the network won't return to the correct solution anymore (until the R-factor between the simulation pattern and the experimental pattern reaches 0.5). The complete procedure of this task are listed in Figure 3.3. The deviation scale is initially 0.01, and will increase exponentially by two. Through this task, we can:

- determine an acceptable range of randomness of structure factors
- get an insight of the random offset values to be added in the task of determining  $U_g$ .

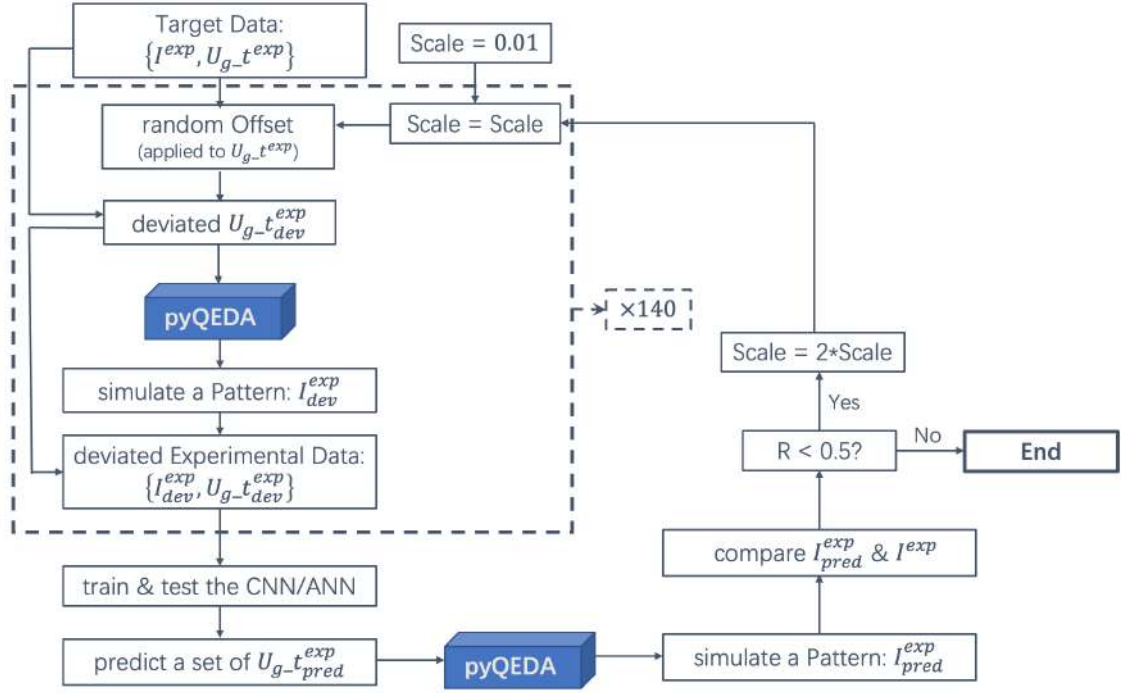


Figure 3.3: Complete routines of task  $U_g$  deviation.

### 3.3.2 Task 2: $U_g$ Determination

In this task, we are going to determine the structure factors and thickness of the experimental pattern. 100 random data are used as initial training data to train the neural network. Another 40 random data are used for testing. The aim of this task is to find the correct structure factors and thickness, from which a similar pattern can be simulated by pyQEDA.

Since the initial 100 training data are generated randomly and largely different from the experimental pattern, it is hard for the network to make an accurate prediction of the structure factor vector. To solve this problem, we expect the training data to have some similarities with the experimental data. One way is to keep feeding new training data to the network. The new training data are generated based on the set of structure factors that the network predicts from the experimental pattern.

Based on this idea, a loop (as shown in Figure 3.4) is applied to replace the original random training data with new training data which is similar to the experimental pattern. The loop works in the following steps:

1. **Initial training:** Get 140 pairs of random samples  $\{I^{(i)}, U_g-t^{(i)}\}$ . 100 of them are used to train the network and another 40 are used for testing. Once the network has been pre-trained, it can be used to produce a set of structure factors and thickness for the experimental pattern  $I^{exp}$ . The predicted structure factors and thickness vector is noted as  $U_g-t_{pred}^{exp}$ .
2. **Deviation:** Although we aim to add some similarity to the training data, we don't want to train the network over the same datasets. To keep the randomness of training data, a small random offset will be added to every element of the predicted structure factor vector. This slightly deviated  $U_g-t_{pred}^{exp}$  will be used to replace the initial training data later. By adding slightly

deviated results we increase the scope of the neural network model. The initial scale of the random offset is determined by the result of Task 1 and will decrease with iterations at a rate of 0.99 to keep the new training data more and more similar to the experimental one. The predicted thickness is also slightly deviated by a few *nms*.

3. **Normalization:** After adding the small random offset, this deviated prediction  $U_{g-t_{pred}}^{exp}$  needs to be symmetrized by performing the **Normalization function**. A new LARBED pattern  $I_{pred}^{exp}$  is simulated by pyQEDA in the **InputUg channel**.
4. **Make unique  $U_g$ :** Apply the **MakeUniqueUgVector function** to the result of the structure factor vector of the previous step to prevent the new training data from having a different unit cell origin.
5. **Replace training data:** Through the procedure above, the pair of  $\{I_{pred}^{exp}, U_{g-t_{pred}}^{exp}\}$  can now be used as a new data to replace the first item in the initial training set. Then the network will be trained again. The network trained next will generate another set of structure factors and simulation pattern, which again will be added to the training data. This procedure will be repeated until all the 100 initial training data are replaced. To retrain a network at iteration  $t$  of the loop, the starting weights will be the weights from the trained network at iteration  $t - 1$ .

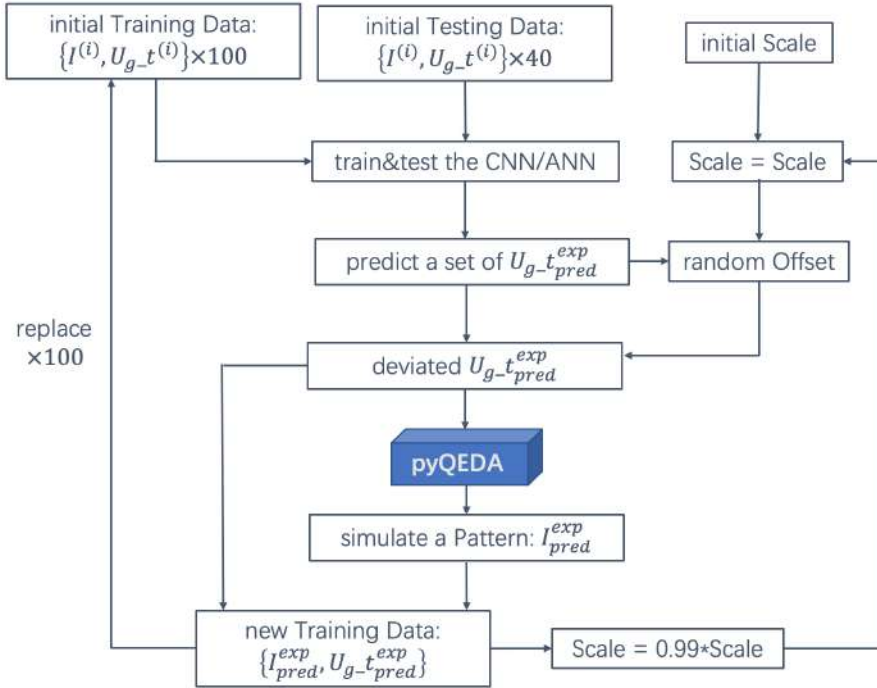


Figure 3.4: Complete routines of task  $U_g$  determination.

Finally, by calculating the intensity difference between the experimental pattern and simulation pattern, we are able to determine the ability of the neural network to make accurate prediction. The expectation is that the simulation pattern will become more and more similar to the experimental pattern during the iteration.

# Chapter 4

## Results

### 4.1 Results of Task 1

This section demonstrates the training performance and the results of CNN and ANN in the task of  $U_g$  Deviation. In this task, the neural networks are trained several times. In each training iteration, 140 pairs of data samples ( $\{I_{dev}^{exp}, U_{g-t_{dev}}^{exp}\}$ ), which are similar to the target data ( $\{I^{exp}, U_{g-t}^{exp}\}$ ), are used to train and test the network. The "similarity" of the data decreases gradually through increasing the deviation scale (from the initial value of 0.01 to a value that no longer leads to a correct prediction).

#### 4.1.1 Initial Training

The initial training starts with a training set very close to the correct solution (deviated by 1%). Figure 4.1 shows the training history of neural networks at the initial deviation scale of 0.01 for the example file of  $SrTiO_3 - 1$ , which has 13 beams. The data are generated from the target data of  $SrTiO_3 - 1$ . Both the training and testing loss gradually converge to  $e^{-6}$  within 20 epochs in CNN. ANN reaches the minimum within more epochs.

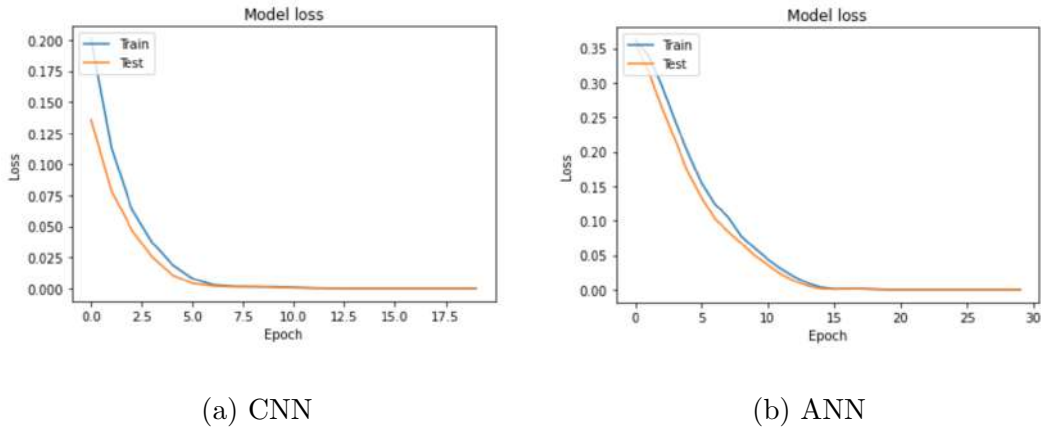
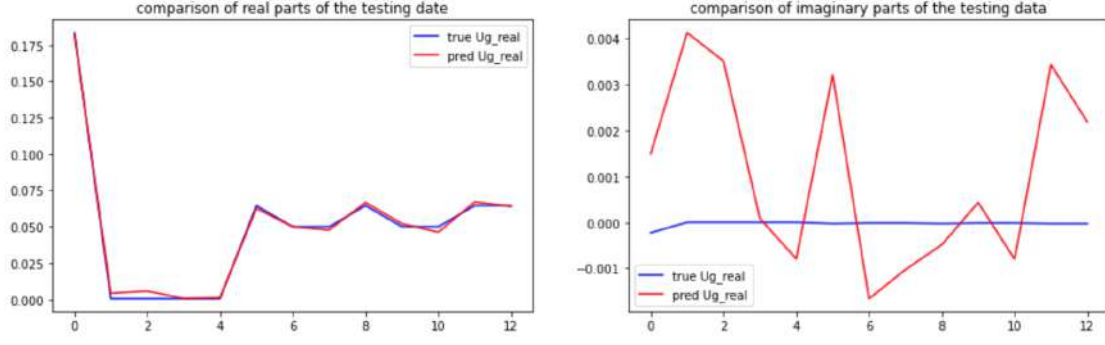


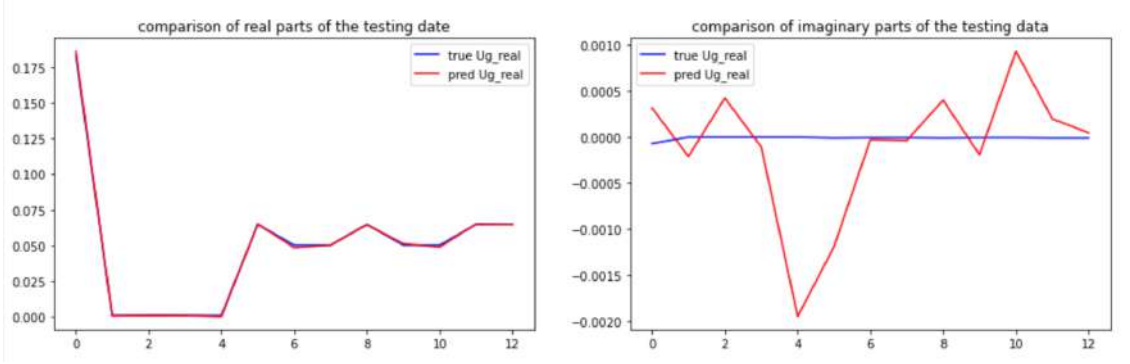
Figure 4.1: Training and testing loss of (a) CNN and (b) ANN at the initial round in Task 1. The training and testing data are generated based on the target data of  $SrTiO_3 - 1$ .



A detailed testing performance of both networks is shown in Figure 4.2, the left figures show a comparison between the prediction of the real part of  $U_g$  and the true value, and the right figures are the comparison of imaginary part  $U_g$ . After training with the 100 data that are deviated from the target data at a small scale of 0.01, the neural networks had a good performance on the testing data. It can be observed that the prediction of the real part of the structure factors is basically consistent with the true values. The prediction of the imaginary part is not vanishing as the true value, it fluctuates within a small range around 0 instead.



(a) CNN



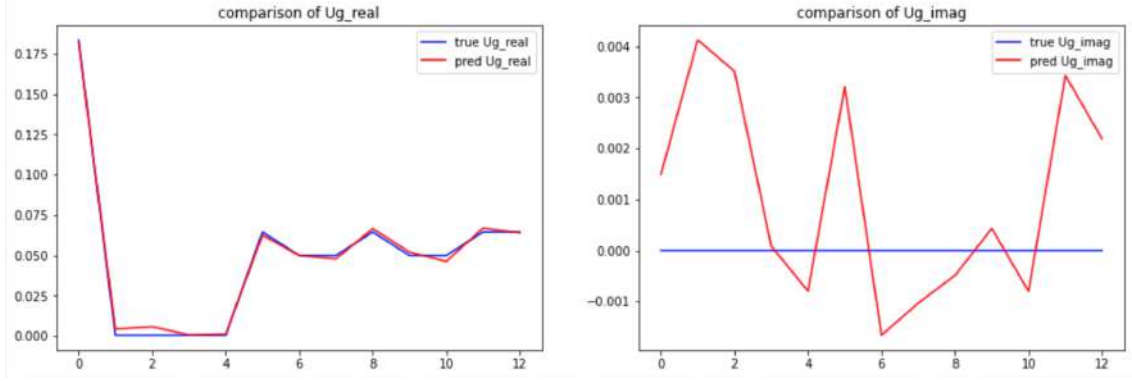
(b) ANN

Figure 4.2: The testing results of (a) CNN and (b) ANN. The figures on the left show the results of the real part of the structure factors, and the right figures the imaginary part. The x-label denotes the index of the beams, there are totally 13 beams used for  $SrTiO_3 - 1$ ; the y-label is the value of structure factors. The blue lines present the true label of the testing data, and the red lines are the prediction of the network.

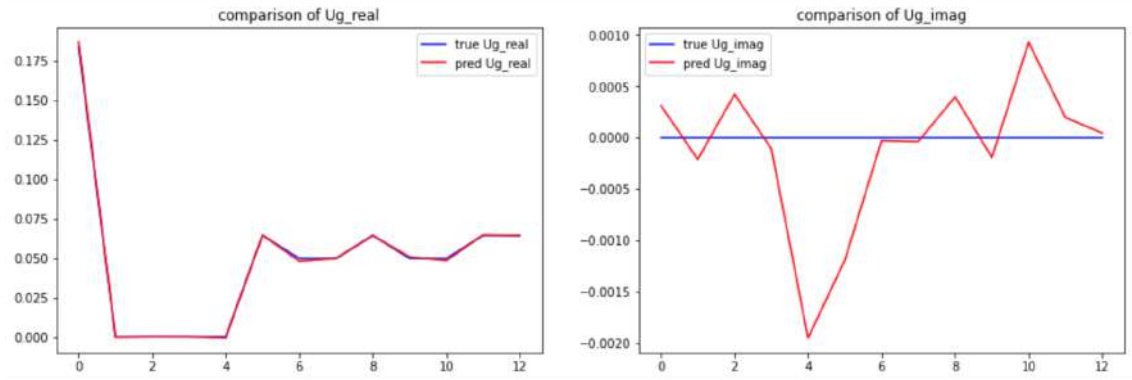
### Initial Prediction

Then, the well trained models are used to predict the structure factors of the experimental pattern  $I^{exp}$ . Figure 4.4 (a) shows the experimental pattern of  $SrTiO_3 - 1$  oriented near the (001) zone axis with 13 beams and 441 tilts covering a disc of maximum radius of 70  $mrad$ . Figure 4.3 shows the predicted results  $U_g \cdot t_{pred}^{exp}$  compared with the true structure factors  $U_g \cdot t^{exp}$ . Since the training and testing data of the initial round are the most similar to the target data, the prediction of the experimental





(a) CNN



(b) ANN

Figure 4.3: The initial prediction results of (a) CNN and (b) ANN.

pattern is almost the same as the prediction on the testing data. The corresponding simulation patterns are generated by inserting the predicted structure factors into pyQEDA, the simulation results are shown in Figure 4.4 with the experimental pattern. Since the prediction of the real part  $U_g$  matches the true value better than the imaginary part (Figure 4.3), and the real part of  $U_g$  has a larger magnitude than the imaginary part of  $U_g$ , the high intensity part of the experimental pattern (yellow circle in the pictures) are better learned by the network.

To quantitatively evaluate the difference between the simulation pattern and the experimental pattern, the R-factor was employed, which is calculated as

$$R = \frac{\sum |I^{exp} - I^{sim}|}{\sum I^{exp}}. \quad (4.1)$$

$I^{exp}$  and  $I^{sim}(=I_{pred}^{exp})$  are the intensities of the experimental and simulation pattern. The R-factor of the two simulation patterns is 0.0133 for the CNN and 0.0001 for the ANN. They are small enough to conclude that the simulation pattern is similar to the experimental pattern.

### 4.1.2 Iterative Training

After the initial training, the deviation scale is gradually increased by multiplying a factor of 2. Therefore, the new dataset deviates gradually from the target data.

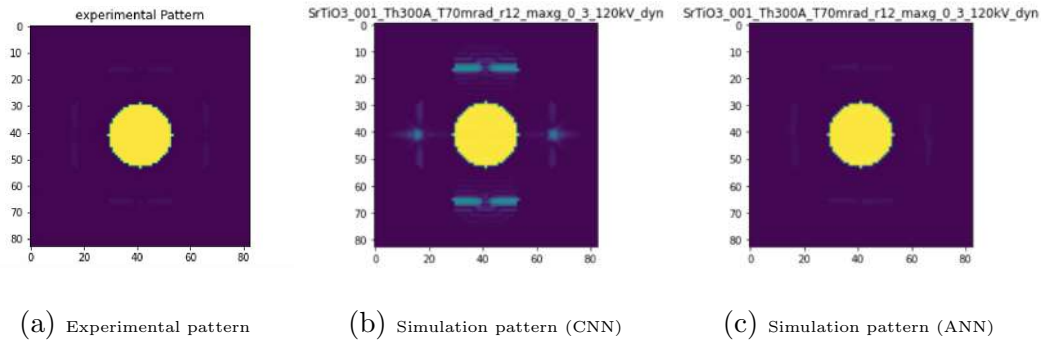


Figure 4.4: (a) Experimental pattern of  $SrTiO_3 - 1$ ; (b) the simulation pattern retrieved from the prediction of structure factors of CNN after the initial training, the R-factor between this simulation pattern and the experimental pattern is 0.0133; (c) the simulation pattern retrieved from the prediction of structure factors of ANN after the initial training, with an R-factor of 0.0001.

Each time the network has been trained on a new data set, it would be used to make a new prediction. An R-factor between the simulation pattern and the experimental pattern will be computed after each prediction. This process will be repeated until the R-factor reaches the value of 0.5. Some of the simulation patterns produced from the predicted structure factors are listed in the Table 4.1. The simulation pattern has not changed much within a deviation scale of 5, the R value remains small until the deviation scale reaches 5.

Deviation scale	0.01	0.16	1.28	5.12	10.24
Simulation pattern (CNN)					
R-factor value	0.0133	0.0078	0.0023	0.0622	0.0240
Simulation pattern (ANN)					
R-factor value	0.0001	0.0001	0.0004	0.0274	0.657

Table 4.1: Some simulation patterns computed from the predicted structure factors at different deviation scales with their R values.

Figure 4.5 shows a comparison of the predicted structure factors with the true value at the deviation scale of 5.12 and 10.24. The left figures show that the prediction of the real parts of structure factors are sufficiently accurate at both deviation scales for CNN and ANN. For the deviation scale of 5.12, the predicted imaginary part of  $U_g$  peaks 0.05 in CNN and 0.1 in ANN. Although these two values are many times larger than the peak of the predicted imaginary part at the deviation scale of 0.01, they are still within the acceptable range (The diffraction pattern admits a

R-value smaller than 0.1). When the deviation scale increases to 10.24, both CNN and ANN get an imaginary part of  $U_g$  with a peak of 0.4, which is large enough to have an fundamental impact on the intensity of the diffraction pattern. The R value of the simulation pattern of ANN even reaches 0.657.

The change of R value with different deviation scales is shown in Figure 4.6. For  $SrTiO_3 - 1$ , both CNN and ANN are able to obtain a structure factor vector with R value of lower than 0.1, from a training set deviated from the experimental pattern at a scale smaller than 5.

From the R-deviation scale curve (Figure 4.6), the comparison between the predicted and the true  $U_g$  (Figure 4.5) and the comparison between simulation pattern and the experimental pattern (Table 4.1), we can draw the following conclusions for the file  $SrTiO_3 - 1$ :

1. The prediction of the neural network gets worse as the training data deviates from the experimental pattern.
2. Random diffraction data deviated from the experimental data at a scale of less than 5 trains the neural network well, and the network would make an accurate prediction.

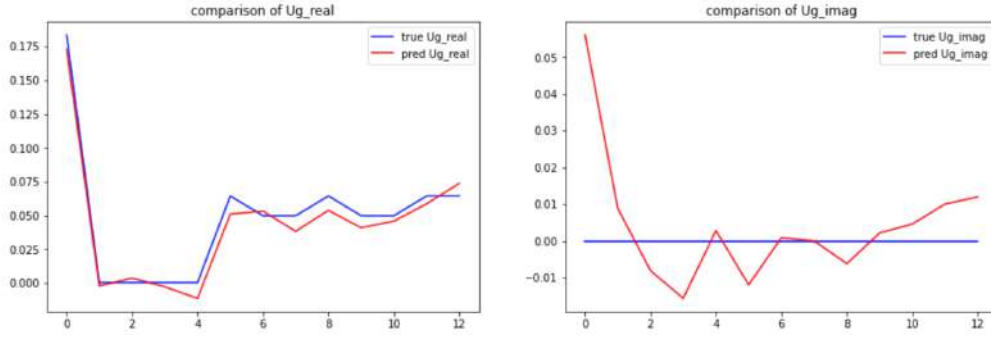
### 4.1.3 Overall Results

The overall results of all parameter files are listed in Table 4.2. The threshold of the deviation scale is determined by the condition that the R value of the simulation pattern should be smaller than 0.3. Besides, the magnitude of predicted  $U_g$  must be smaller than 0.2.

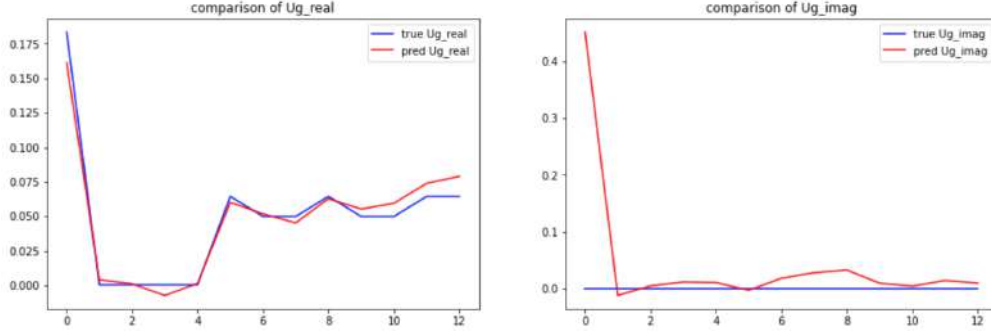
Parameter file	CNN		ANN	
	Threshold of deviation	Running time in s	Threshold of deviation	Running time in s
$SrTiO_3 - 1$	5.12	4031	5.12	3077
$SrTiO_3 - 2$	5.12	4655	5.12	3820
$SrTiO_3 - 3$	5.12	4386	5.12	4071
$SrTiO_3 - 4$	5.12	9295	5.12	12243
$SrTiO_3 - 5$	5.12	7504	5.12	8291
$SrTiO_3 - 6$	5.12	14697	1.28	6269
$SrTiO_3 - 7$	5.12	9152	5.12	13773
$SrTiO_3 - 8$	5.12	3657	5.12	11794
$SrTiO_3 - 9$	5.12	5135	5.12	4328
$SrTiO_3 - 10$	5.12	3986	5.12	5216
$ZnO - 1$	10.24	2104	10.24	7777
$ZnO - 2$	10.24	3060	2.56	6420

Table 4.2: Results of Task 1.

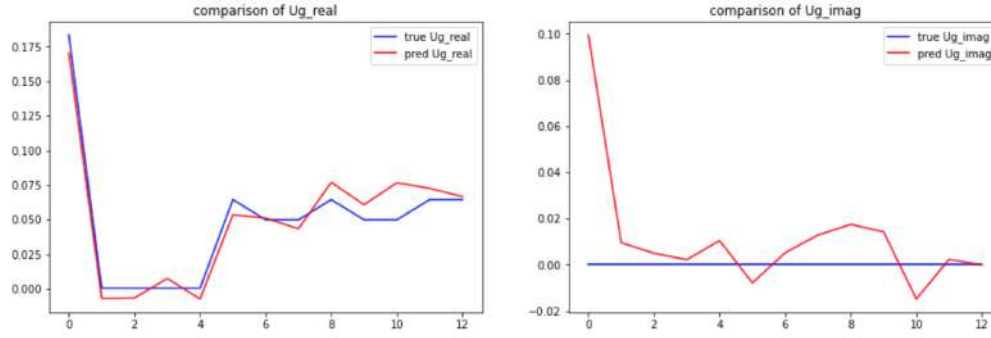
Generally speaking, the threshold of the deviation scale of all  $SrTiO_3$  parameter files is determined to be 5.12, while the threshold of  $ZnO$  is 10.24. The CNN network is more stable than ANN. Their predictions on the same parameter file are basically



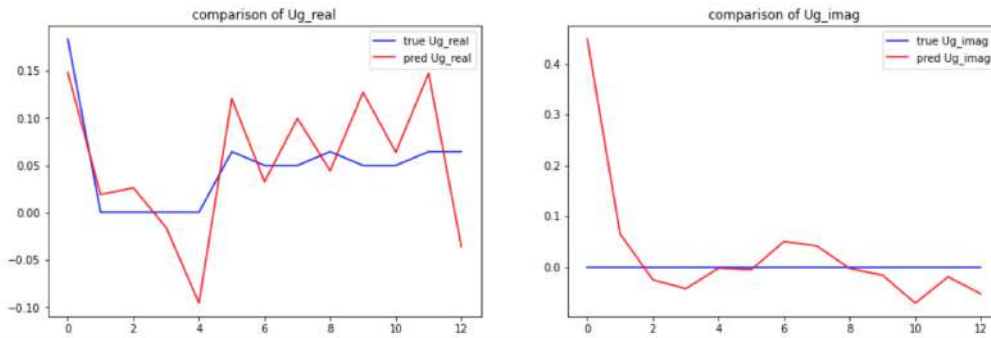
(a) CNN, deviation scale = 5.12



(b) CNN, deviation scale = 10.24

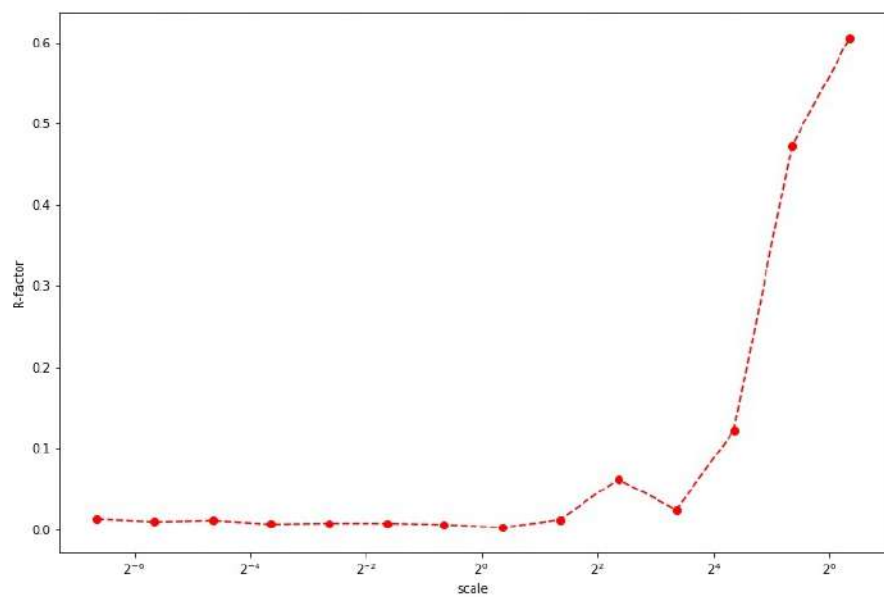


(c) ANN, deviation scale = 5.12

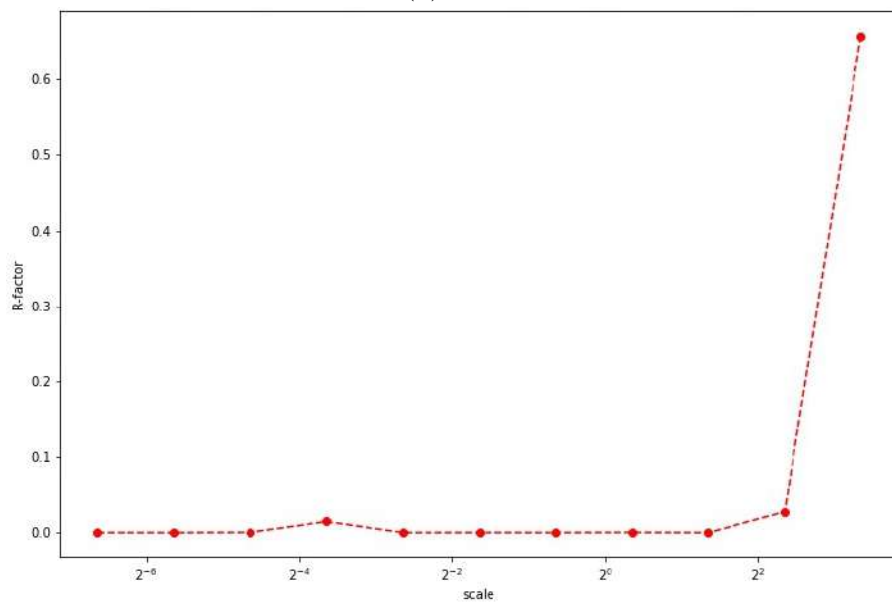


(d) ANN, deviation scale = 10.24

Figure 4.5: Comparison between the predicted structure factors and the true value in the case of (a) CNN with a deviation scale of 5.12; (b) CNN with a deviation scale of 10.24; (c) ANN with a deviation scale of 5.12; (d) ANN with a deviation scale of 10.24. The blue and red lines represent the prediction of neural network and true value respectively.



(a) CNN



(b) ANN

Figure 4.6: Change of the R-factor value calculated from the simulation and the experimental pattern with increasing deviation scale.

the same, while the thresholds obtained under ANN have occasionally outliers. The final result for ANN in the table 4.2 is the result with the most occurrences. As to the time required for processing, CNN is more time efficient than ANN on diffraction data with more beams and tilts.

To summarize, both CNN and ANN trained on the training data with the deviation scale less than 5 are able to predict the correct structure factors for the experimental pattern. This also means that adding a random offset within a deviation scale of 5 has little influence on the simulation pattern retrieved from the predicted structure factors of the network. For this reason, an initial scale is determined to be 5 in the next task for both neural networks.

## 4.2 Results of Task 2

The training performance and the prediction of CNN and ANN in the task of  $U_g$  Determination are presented in this section. In this task, both CNN and ANN are trained iteratively in a loop with the random dataset. The results from the predictions replace the initial training set iteratively.

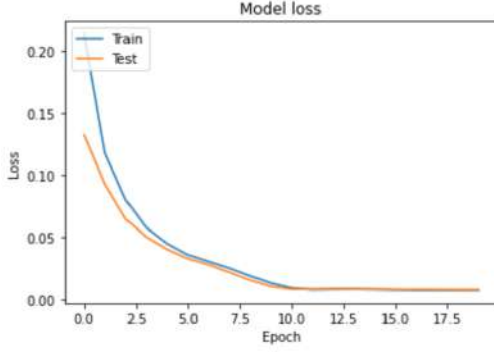
### 4.2.1 Initial Training

In the initial training, 140 pairs of data samples ( $\{I^{(i)}, U_g t^{(i)}\}$ ) randomly generated by pyQEDA are used to train and test the network. Figure 4.7 shows the training and testing error of neural networks in the initial training for the example file of  $SrTiO_3 - 1$ . Within the default 20 and 30 epochs in CNN and ANN, both the training error and testing error calculated by MSE are smoothly reduced to a minimum value of 0.007 without overfitting or underfitting problems.

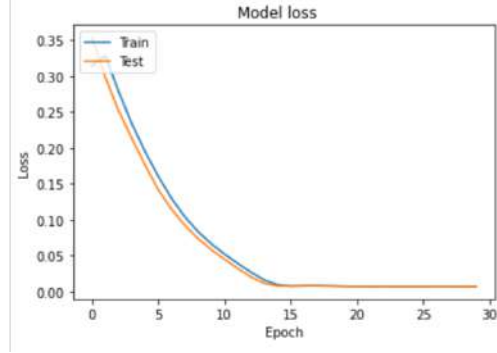
The prediction of the testing set compared with the true label is shown in Figure 4.8. During the initial training, the prediction of CNN and ANN on the testing set are very similar. For the real part of the structure factors, although the predicted value (red) does not perfectly match the true value (blue), the general relationship between the structure factors and different reflections are reflected by our model. The imaginary part disappears for most of the beams with the predicted result is close to 0.

### Initial Prediction

The pre-trained neural network is then used to make a prediction of the structure factors-thickness vector  $U_g t_{pred}^{exp}$  of the experimental pattern of  $SrTiO_3 - 1$ . The diffraction parameters of this example are the followings:



(a) CNN



(b) ANN

Figure 4.7: Training and testing loss of (a) CNN and (b) ANN at the initial training in Task 2. The training and testing data are the random dataset of  $SrTiO_3 - 1$ .

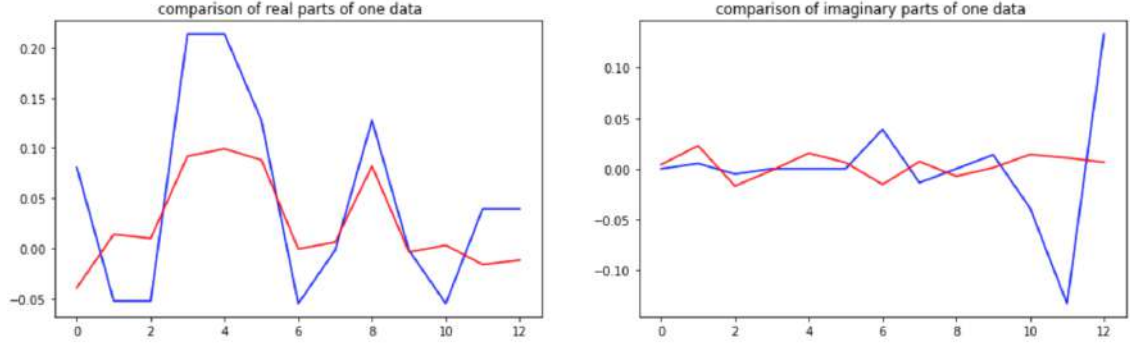
File Name	Number of Beams	Number of Tilts	Thickness (nm)	Tilt Range (mrad)	Voltage (KV)
$SrTiO_3 - 1$	13	441	30	70	120

Table 4.3: diffraction parameters of file  $SrTiO_3 - 1$

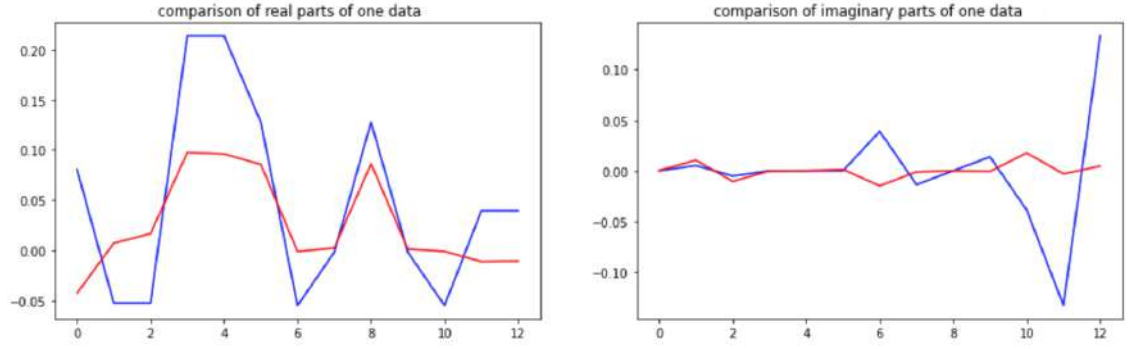
The predicted thickness is 51.15 nm for CNN and 52.60 nm for ANN. The simulation patterns  $I_{pred}^{exp}$  generated by the predicted structure factors  $U_g t_{pred}^{exp}$  are shown in Figure 4.9. The simulation patterns show a large deviation from the experimental pattern. The R-factors are 0.6564 in CNN and 0.6687 in ANN. The main reason is that the neural networks are trained with completely random diffraction data. From Task 1 we know that it is hard for the networks to make an accurate prediction if the training set is very different from the experimental data. In order to obtain a more accurate prediction, the networks will be further trained with newly added training data more similar to the experimental data. A loop will be used to add new training data. The pair of predicted structure factors and its simulation pattern  $\{I_{pred}^{exp}, U_g t_{pred}^{exp}\}$  of the initial training will be the first pair of new training data.

## 4.2.2 Iterative Training

In each further training iteration, the prediction will be used to replace one of the initial training data. The initial weights of each new training are set to the weights trained from the neural network at the previous iteration. In this way, the network keeps the information obtained from the previous training and fine-tunes the weights through the newly added training data. To adjust the weights, 5 epochs are applied in both CNN and ANN. Figure 4.10 represents the training and testing loss of the first and the last iteration in both CNN and ANN. In the first iteration, only the first initial training data is replaced by the new training data. The training and testing errors are not significantly reduced in 5 epochs. Although the loss does not decrease significantly in each iteration, it does slowly decrease by iterations. The training and testing loss in the last iteration, where the entire initial training



(a) CNN



(b) ANN

Figure 4.8: The initial prediction results on one testing data from a completely random training set. The predicted value (red lines) are compared with the true value (blue lines) for (a) CNN and (b) ANN. The figures on the left correspond to the real part of the structure factors, and the right figures the imaginary part.

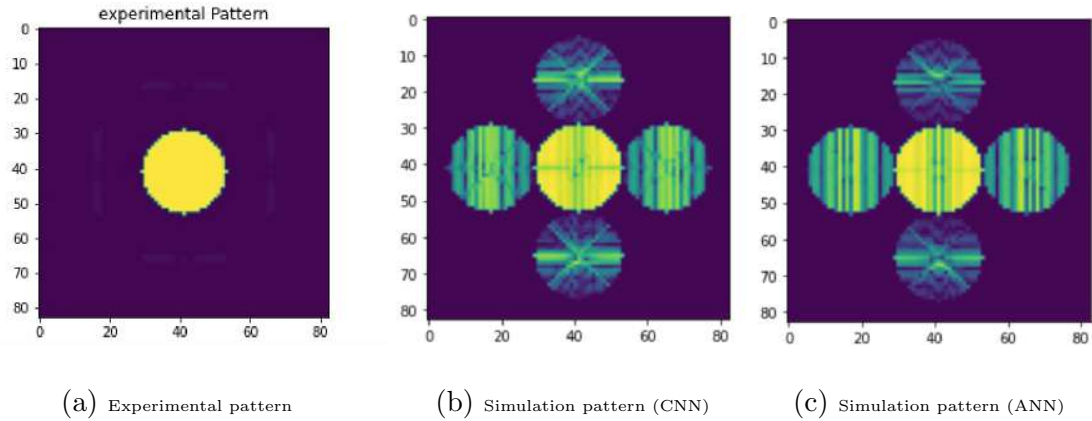
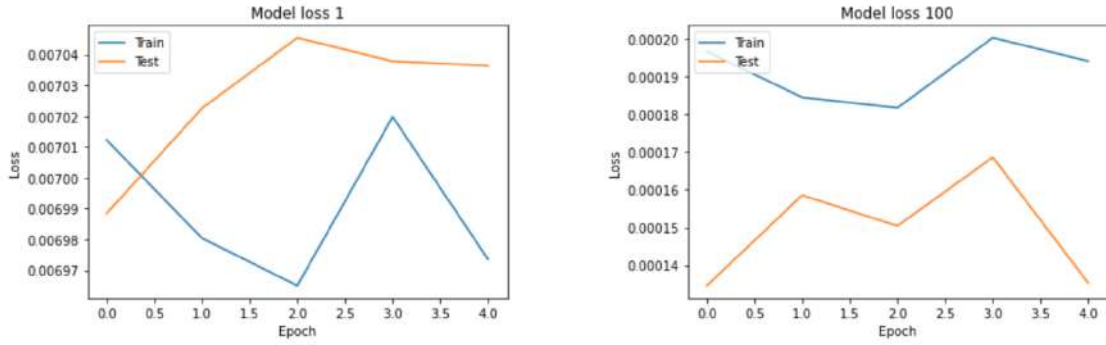


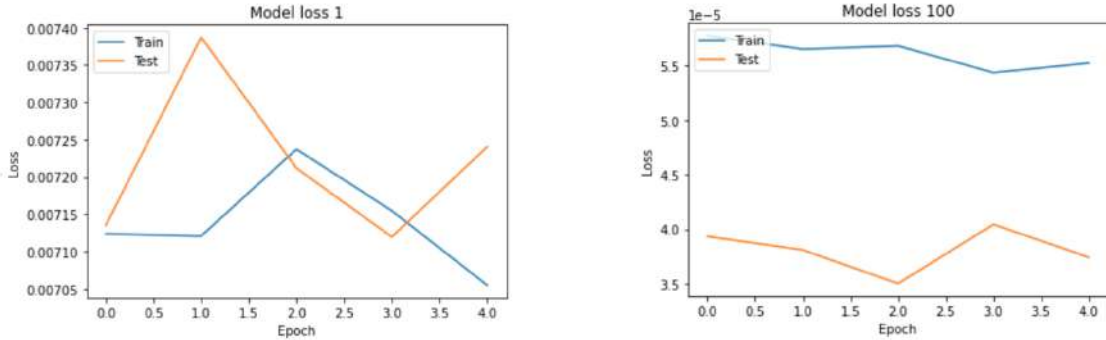
Figure 4.9: (a) Experimental pattern of  $SrTiO_3 - 1$ ; (b) the simulation pattern retrieved from the predicted structure factors of CNN after the initial training, the R-factor is 0.6564; (c) the simulation pattern retrieved from the predicted structure factors of ANN after the initial training with an R-factor of 0.6687. The neural networks are trained on the random dataset (with random structure factors in the range  $\pm 0.2$  and thickness in range 50-100 nm).



set is replaced by new training data, are reduced to  $e^{-4}$  and  $e^{-5}$ .



(a) CNN



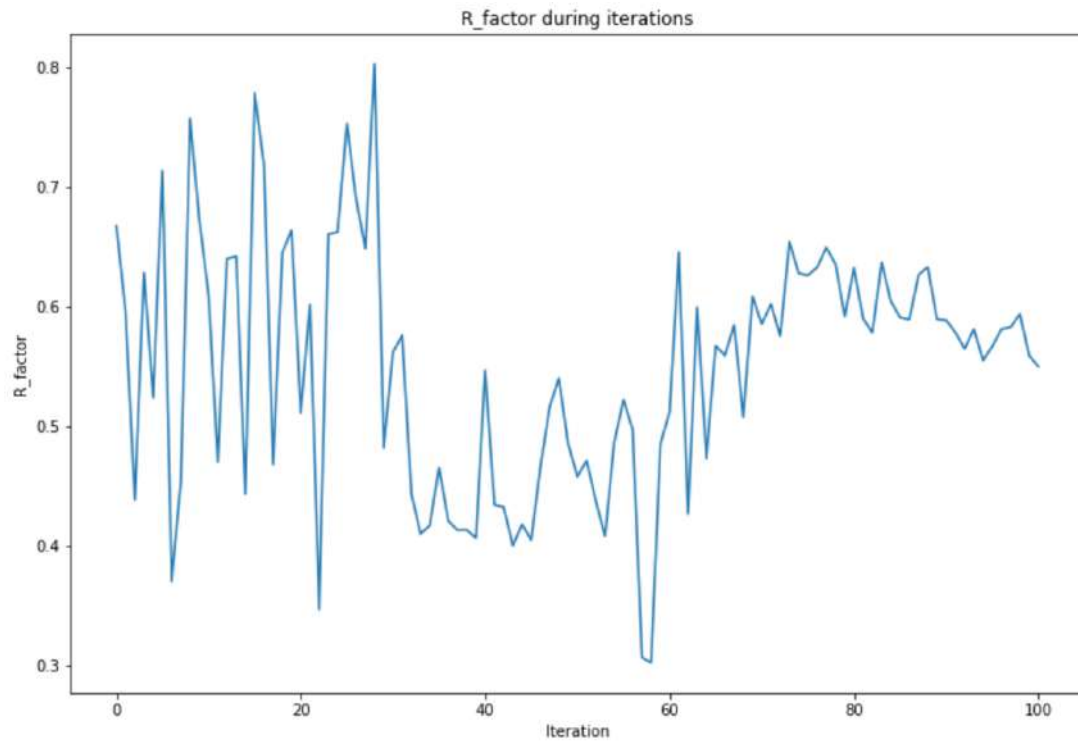
(b) ANN

Figure 4.10: Training and testing loss at the 1st iteration and 100th iteration in (a) CNN and (b) ANN. In the 1st iteration, the pair of predicted  $U_g$  and simulation pattern from the initial trained network replaced the first data in the initial training set. In the 100th iteration, the whole initial training data has been replaced with new training data.

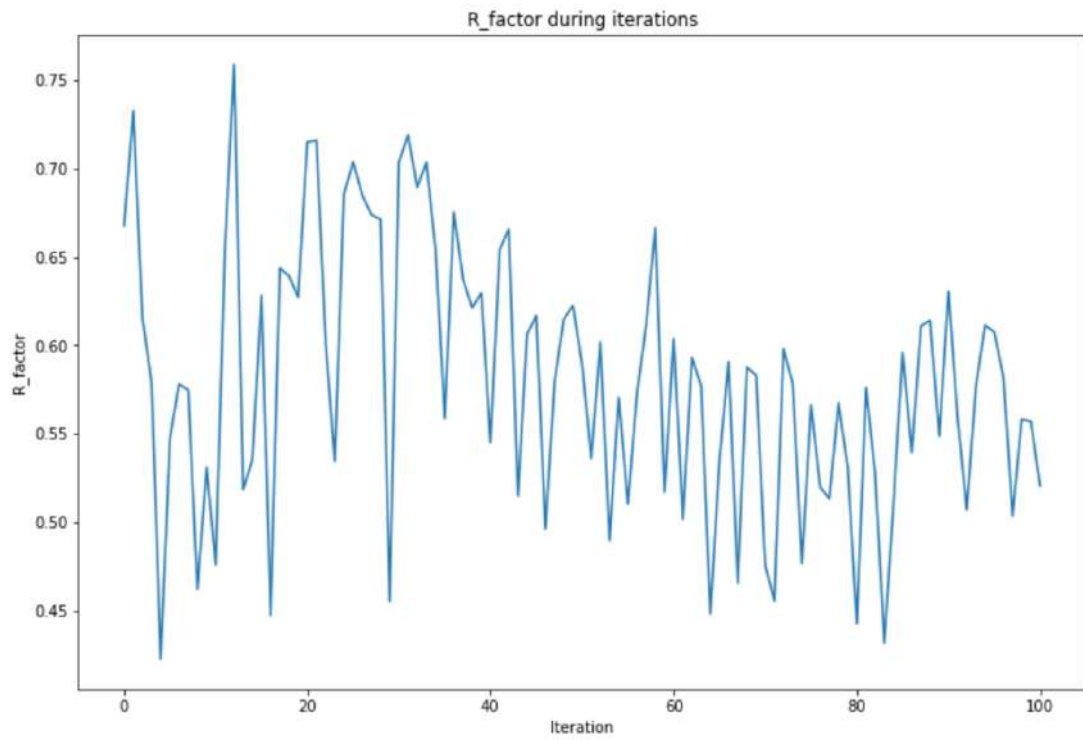
Although the neural network is trained by minimizing the difference between the output structure factors and the true label, the ultimate goal of this task is to obtain a set of structure factors that minimizes the R value of the simulation pattern. In the training process of 100 iterations, the network made a total number of 100 predictions. An R value was calculated for each pattern simulated by the predicted structure factors. The change of the R value is shown in Figure 4.11. The R-iteration curve does not decrease step by step like the training loss, multiple peaks appear randomly on the curve. Although the R value of the last iteration is smaller than the R value obtained in the initial prediction, the minimal R is obtained during the iteration process.

In the case of  $SrTiO_3 - 1$ , the minimal R values obtained by CNN and ANN are 0.3027 and 0.4225. Figure 4.12 shows the "best" simulation patterns with the smallest R value for two networks. Although the "best" simulation patterns obtained from the predicted structure factors are still quite different from the experimental pattern, we at least get a better result than the initial prediction.

The predicted thickness corresponding to the "best" simulation patterns are 55.45 nm and 52.38 nm, which are far different from the true value of 30 nm given by the parameter file. All predictions of the thickness during the iteration process



(a) CNN



(b) ANN

Figure 4.11: R-iteration curve for (a) CNN and (b) ANN.

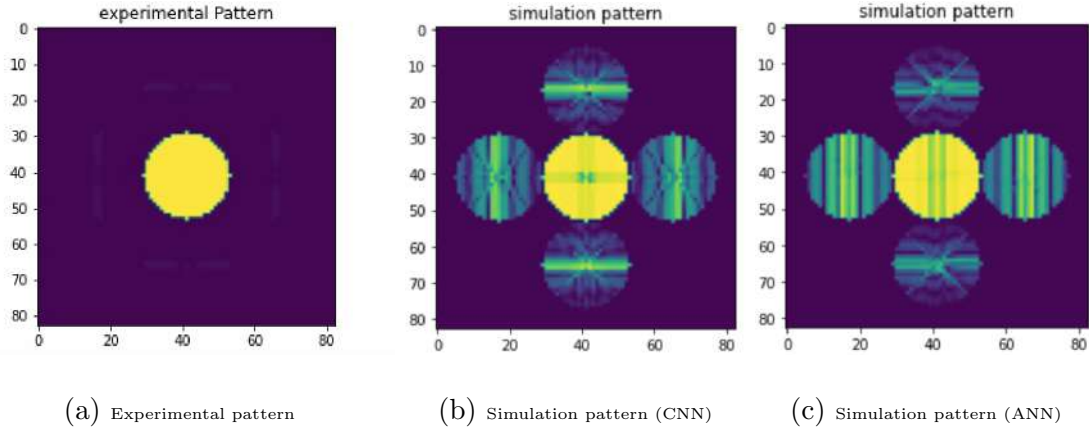


Figure 4.12: The experimental pattern of  $SrTiO_3 - 1$  (a); the best simulation patterns obtained by (b) CNN and (c) ANN, which have a R value of 0.3027 and 0.4225 respectively.

are shown in Figure 4.13. The average thickness of sample  $SrTiO_3 - 1$  predicted by the CNN is about 0.5 (50 nm), while the prediction of the ANN is even larger. None of the two networks gives an accurate prediction of the thickness.

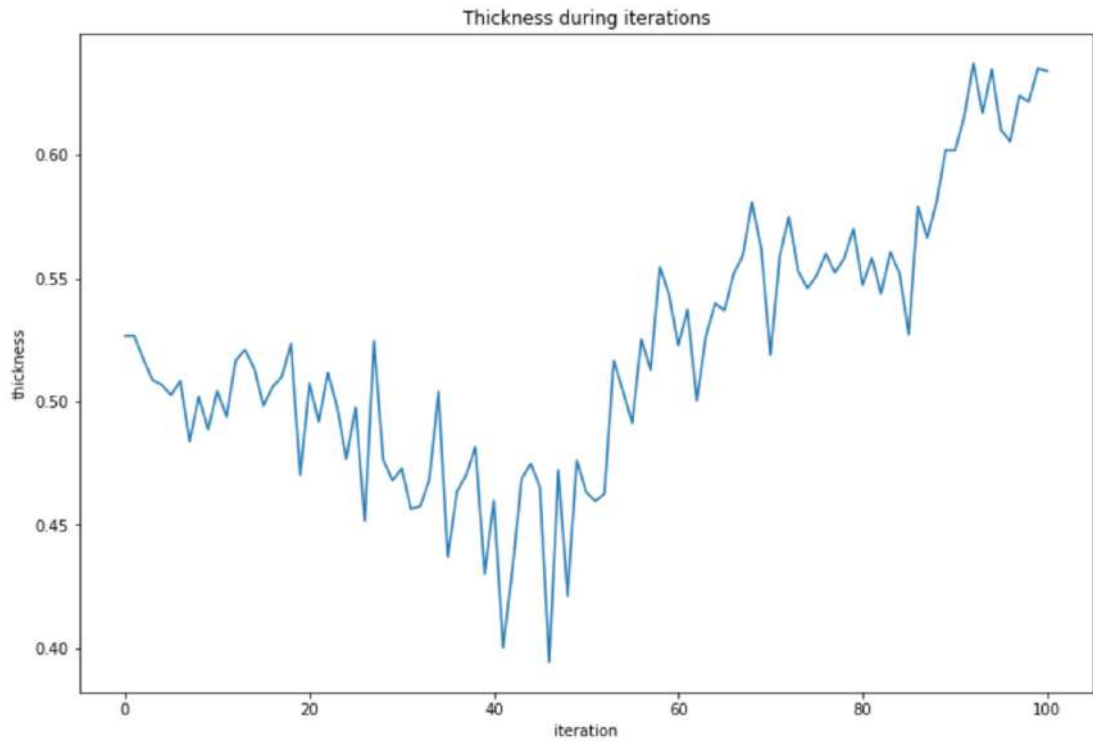
### 4.2.3 Overall results

Parameter file	CNN			ANN		
	$R_{min}$	Thickness in nm	Running time in s	$R_{min}$	Thickness in nm	Running time in s
$SrTiO_3 - 1$	0.303	55.5	1127	0.423	52.4	952
$SrTiO_3 - 2$	0.197	36.8	1174	0.219	52.9	1064
$SrTiO_3 - 3$	0.215	55.9	1223	0.176	51.3	1729
$SrTiO_3 - 4$	0.397	55.6	1891	0.353	55.4	2271
$SrTiO_3 - 5$	0.515	53.0	1737	0.510	51.7	3411
$SrTiO_3 - 6$	0.221	48.6	2071	0.177	47.5	4881
$SrTiO_3 - 7$	0.199	48.3	2027	0.169	49.3	4234
$SrTiO_3 - 8$	0.505	47.7	1554	0.479	48.6	1658
$SrTiO_3 - 9$	0.301	55.5	1630	0.262	55.1	2561
$SrTiO_3 - 10$	0.630	51.6	2771	0.711	55.6	10459
$ZnO - 1$	0.495	47.2	1424	0.531	46.8	3317
$ZnO - 2$	0.666	52.3	3122	0.713	54.2	6293

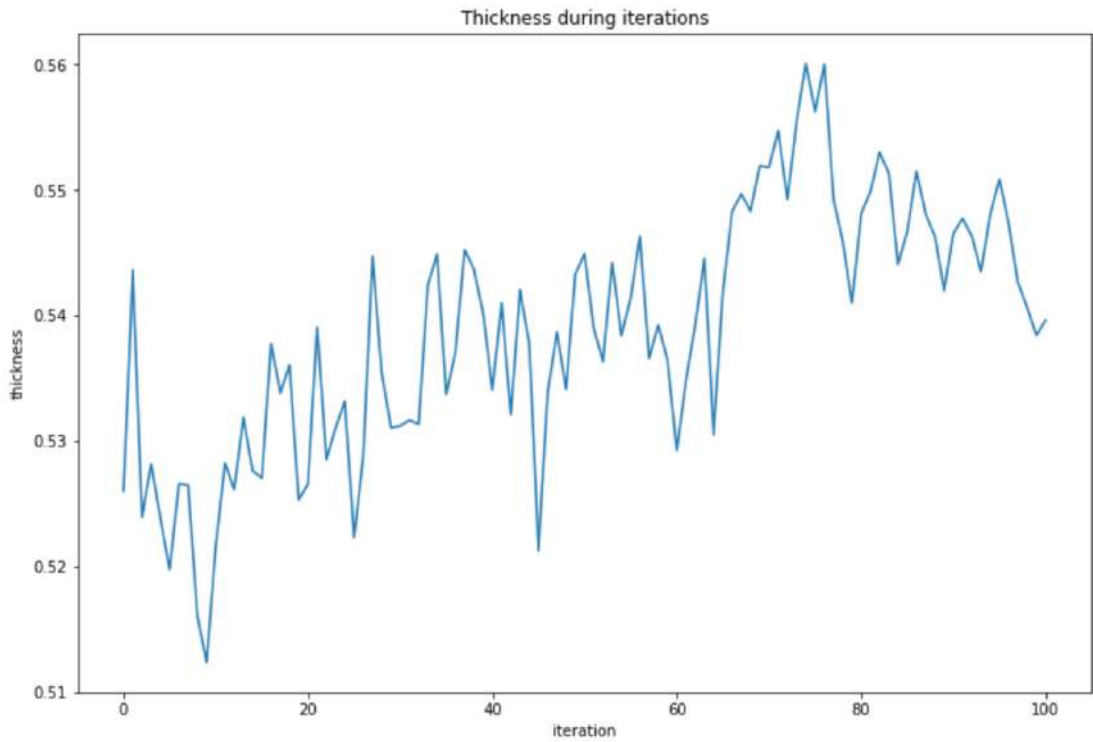
Table 4.4: Results of Task 2.

The predicted results of all the parameter files are summarized in the Table 4.4. For each parameter file, the "best" simulation pattern is obtained following the process. The R value and the predicted thickness of the "best" simulation pattern are recorded in the table. In addition, the running time of the entire training process is also calculated.

By observing the R values, both CNN and ANN exhibit a better prediction performance for the LARBED patterns of the parameter files of  $SrTiO_3 - 2$ ,  $SrTiO_3 - 3$ ,



(a) CNN



(b) ANN

Figure 4.13: Predicted thickness - iteration curve of (a) CNN and (b) ANN. The thicknesses are scaled to 0 – 1 by dividing 1000 Å.

$SrTiO_3-6$ ,  $SrTiO_3-7$  and  $SrTiO_3-9$ . Table 4.5 shows the diffraction parameters of these patterns. Their "best" simulation patterns have the R value below or equal to 0.3. All the LARBED patterns have a tilt range of 100 *mrad*, which is larger

File Name	Number of Beams	Number of Tilts	Thickness ( <i>nm</i> )	Tilt Range ( <i>mrad</i> )	Voltage (KV)
$SrTiO_3-2$	13	441	30	100	120
$SrTiO_3-3$	13	441	15	100	120
$SrTiO_3-6$	13	1009	30	100	120
$SrTiO_3-7$	13	1009	15	100	120
$SrTiO_3-9$	25	441	15	100	120

Table 4.5: some diffraction parameters

than other patterns. In addition, these LARBED patterns are mostly composed of 13 beams, only  $SrTiO_3-9$  is composed of 25 beams. The "best" simulation patterns and the experimental patterns of these parameter files are shown in Table 4.11. The results of all other files are collected in Table 4.12 and Table 4.13.

To further confirm the positive influence of the large tilt range on the prediction results, the results of the  $SrTiO_3-1$  and  $SrTiO_3-2$ , as well as the  $SrTiO_3-4$  and  $SrTiO_3-5$  are compared in Table 4.6. They have different tilt ranges while all other parameters are the same. Both CNN and ANN get a better prediction result for the LARBED pattern with a larger tilt range. The R value of the simulation pattern of  $SrTiO_3-5$ , which has a relatively small tilt range of 40 *mrad*, reaches 0.5.

File Name	Tilt Range ( <i>mrad</i> )	$R_{min}$ (CNN)	$R_{min}$ (ANN)
$SrTiO_3-1$	70	0.303	0.423
$SrTiO_3-2$	100	0.197	0.219
$SrTiO_3-4$	70	0.397	0.353
$SrTiO_3-5$	40	0.515	0.510

Table 4.6: Comparison of different tilt ranges

Similarly, by comparing the prediction results of  $SrTiO_3-1$ ,  $SrTiO_3-8$  and  $SrTiO_3-10$ ; as well as  $SrTiO_3-3$  and  $SrTiO_3-9$ , we can observe the influence of the number of beams constructing the LARBED patterns on the performance of the model prediction. It can be concluded from Table 4.7 that while other parameters are the same, the R value of the simulation pattern is larger for patterns with more beams. The prediction performance of the networks for a LARBED pattern with 69 beams becomes highly inaccurate. The difference between the simulation pattern and the experimental pattern is more than 50%.

It can be observed that the number of beams and the tilt range have a major effect on the ability of the neural network to determine structure factors from LARBED patterns. However, the prediction performances of both models are not significantly related to the crystal thickness and the number of tilts (see Table 4.8 and Table 4.9).

The predicted thickness is found to be independent from the true thickness of the experimental pattern. No matter how thick the crystal given by the experimental pattern is, the prediction of the network is always around 50 *nm*.

File Name	Number of beams	$R_{min}$ (CNN)	$R_{min}$ (ANN)
$SrTiO_3 - 1$	13	0.303	0.423
$SrTiO_3 - 8$	25	0.505	0.479
$SrTiO_3 - 10$	69	0.630	0.711
$SrTiO_3 - 3$	13	0.215	0.176
$SrTiO_3 - 9$	25	0.301	0.262

Table 4.7: Comparison of different number of beams

File Name	Number of tilts	$R_{min}$ (CNN)	$R_{min}$ (ANN)
$SrTiO_3 - 1$	441	0.303	0.423
$SrTiO_3 - 4$	797	0.397	0.353
$SrTiO_3 - 2$	441	0.197	0.219
$SrTiO_3 - 6$	1009	0.221	0.177
$SrTiO_3 - 3$	441	0.215	0.176
$SrTiO_3 - 7$	1009	0.199	0.169

Table 4.8: Comparison of different number of tilts

File Name	Thickness ( $nm$ )	$R_{min}$ (CNN)	$R_{min}$ (ANN)
$SrTiO_3 - 2$	30	0.197	0.219
$SrTiO_3 - 3$	15	0.215	0.176
$SrTiO_3 - 6$	30	0.221	0.177
$SrTiO_3 - 7$	15	0.199	0.169

Table 4.9: Comparison of different thicknesses

To compare with  $SrTiO_3$ , models also try to predict the structure factors of the  $ZnO$  LARBED pattern, which has a non-vanishing imaginary part of the structure factors. Figure 4.14 shows the experimental pattern as well as the "best" simulation patterns of  $ZnO-1$  with 95 beams, 197 tilts with the tilt range of  $30\text{ mrad}$ . Although the "best" simulation patterns of  $ZnO-1$  look similar to the experimental pattern, the R value reaches 0.5. At the same R value, the simulation pattern of  $ZnO$  looks more like the experimental pattern than  $SrTiO_3$  because the pattern of  $ZnO$  has more non-zero intensities, i.e. more strong reflections.

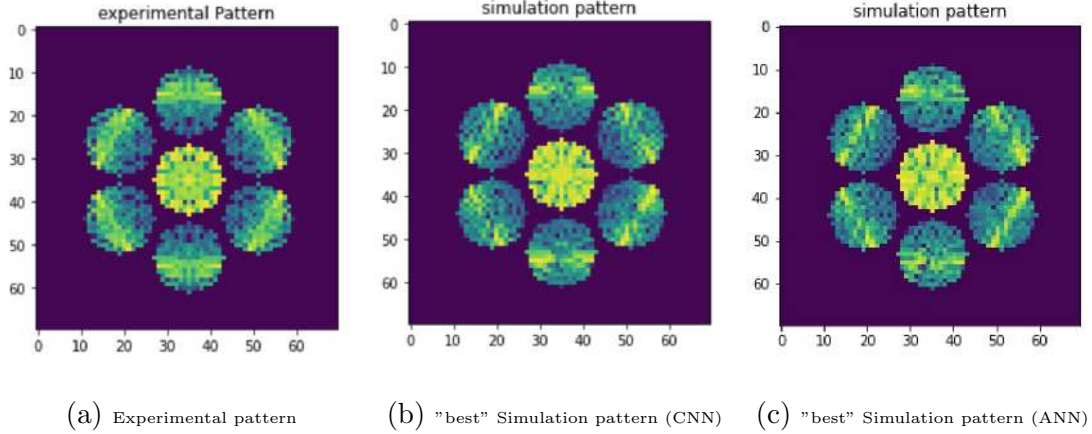


Figure 4.14: (a) Experimental pattern of  $ZnO-1$ ; (b) the "best" simulation pattern retrieved from the predicted structure factors of CNN, the R-factor calculated between this simulation pattern and the experimental pattern is 0.495; (c) the "best" simulation pattern retrieved from the predicted structure factors of ANN, having a R-factor of 0.531.

To further investigate the ability of our approach to determine the correct structure factors of different crystal categories, the best two simulation LARBED patterns ( $SrTiO_3-2$  and  $ZnO-1$ ) are presented with their experimental pattern in Figure 4.15 and Figure 4.16. The potential maps corresponding to the real part and the imaginary part of the structure factors are shown in (b),(c),(e) and (f) and the unit cells are outlined in red. The crystal structures of  $SrTiO_3$  and  $ZnO$  are presented on the bottom of the figures, where the unit cell are also outlined. The diffraction parameters are given in Table 4.10. It can be observed from Figure 4.15 that the reconstructed potential field of  $SrTiO_3$  does not agree reasonably with the experimental one. The atoms in the outlined unit cell are incorrectly positioned, from which we can conclude that the phases of the structure factors have been incorrectly reconstructed. Compared to the diffraction intensities of the experimental pattern, the intensities in the simulation pattern are much stronger. This is due to a larger amplitude of the predicted structure factors. The simulation pattern of  $ZnO$  agrees better with the experimental pattern than that of  $SrTiO_3$ . The atoms in the unit cell are in the correct positions, except that the atom in the upper right is split in two.

In terms of running time, the CNN spent 20-40 minutes for the entire iterative process, depending on the quantity of diffraction intensity data. The larger the data size, the longer the running time. The running time of the ANN is more sensitive to the data size. For a LARBED pattern with only 13 beams and 441 tilts (that is,  $13 * 441 = 5,733$  intensity data points), it only takes 15 minutes, while for a pattern composed of 69 beams and 441 tilts ( $69 * 441 = 30429$ ), it takes about 175 minutes,

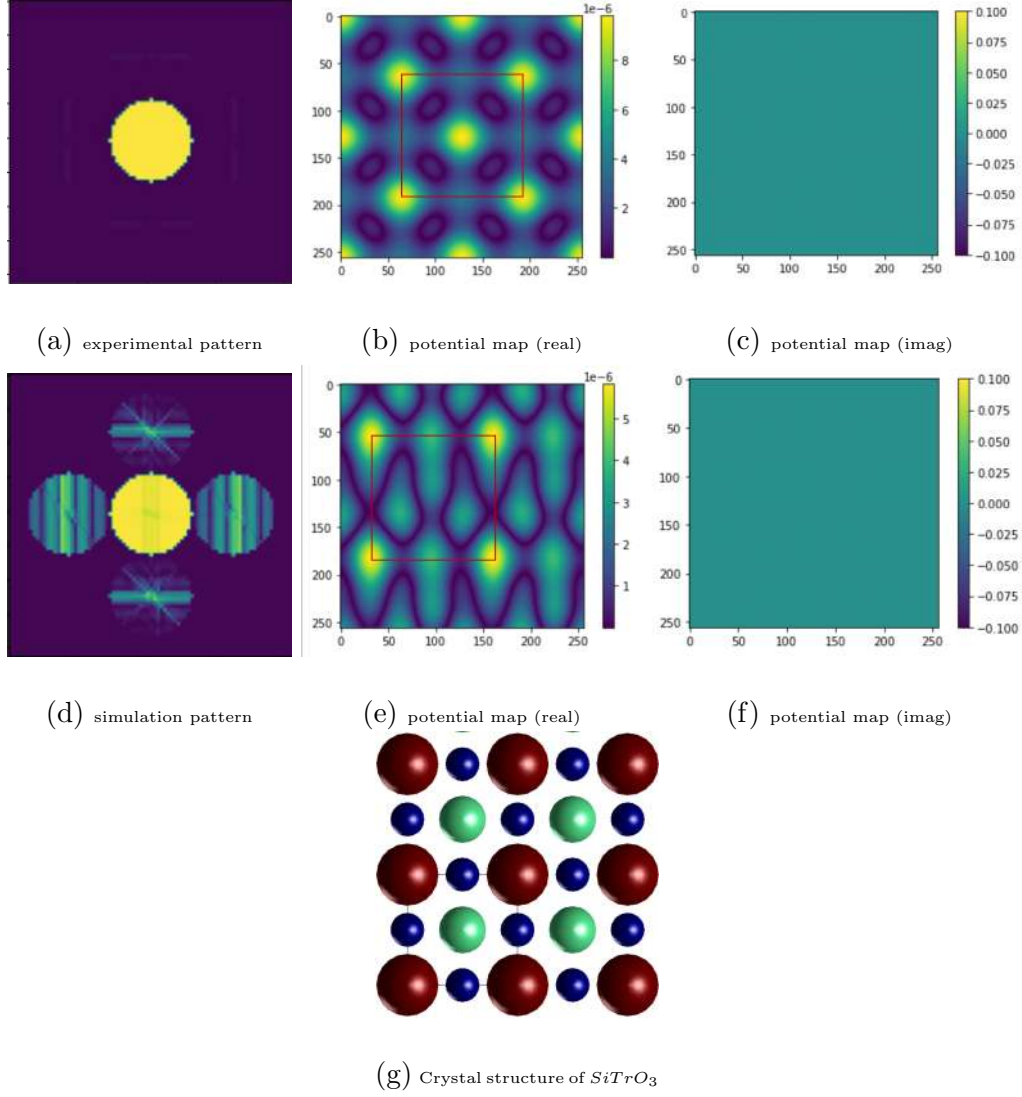


Figure 4.15: Potential maps of the corresponding LARBED pattern of  $SrTiO_3 - 2$ . (a) Experimental pattern of  $SrTiO_3 - 2$  (one unit cell is outlined in red); (b) real part potential of the experimental pattern; (c) imaginary part of the potential of the experimental pattern; (d) simulation pattern of  $SrTiO_3 - 2$  (in CNN); (e) real part potential of the simulation pattern (one unit cell is outlined in red); (f) imaginary part of the simulation pattern; (g) crystal structure of  $SrTiO_3$  (the unit cell is outlined).



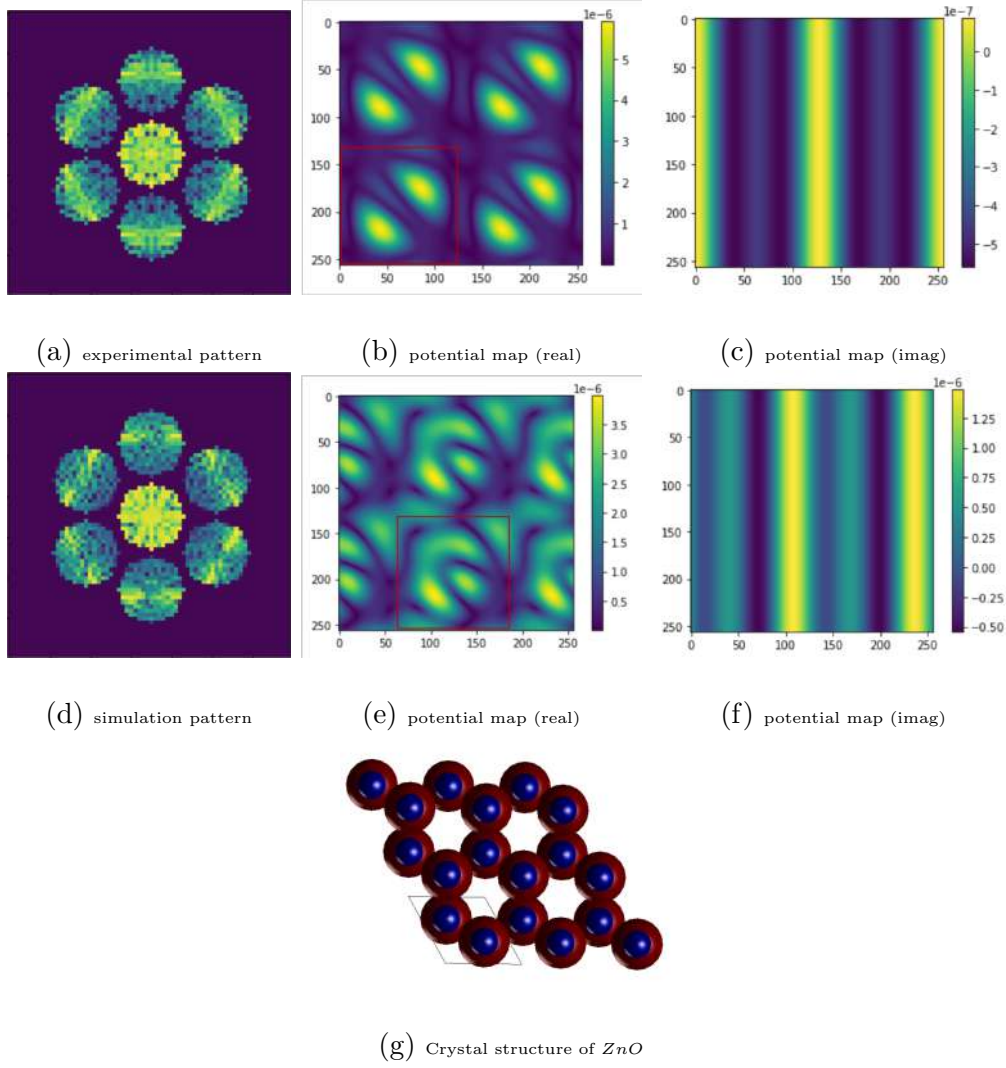


Figure 4.16: Potential maps of the corresponding LARBED pattern of  $ZnO - 1$ . (a) Experimental pattern of  $ZnO - 1$  (one unit cell is outlined in red); (b) real part potential of the experimental pattern; (c) imaginary part of the potential of the experimental pattern; (d) simulation pattern of  $ZnO - 1$  (in CNN); (e) real part potential of the simulation pattern (one unit cell is outlined in red); (f) imaginary part of the simulation pattern; ; (g) crystal structure of  $ZnO$  (the unit cell is outlined).

File Name	Number of Beams	Number of Tilts	Thickness ( <i>nm</i> )	Tilt Range ( <i>mrad</i> )	Voltage ( <i>KV</i> )
<i>SrTiO</i> <sub>3</sub> – 2	13	441	30	100	120
<i>ZnO</i> – 1	95	197	50	30	100

Table 4.10: diffraction parameters of file *SrTiO*<sub>3</sub> – 2 and *ZnO* – 1

which is nearly four times long as the time ran by CNN. Using the intensities in the LARBED pattern as the input of ANN will save a lot of time, but the prediction performance will not differ greatly.

In summary, the results of Task 2 show that the ability of the CNN and ANN models to obtain the structure factors from a LARBED pattern is dependent on the number of beams and the tilt range. Both networks have a good performance ( $R \leq 0.3$ ) in predicting structure factors of the LARBED patterns that are formed by less than 25 beams and a tilt range of 100 *mrad*. However, the thickness prediction is far from correct. In terms of the running time, CNN is more efficient when processing LARBED patterns with a larger amount of intensity data.

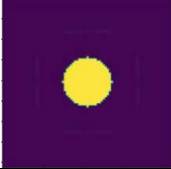
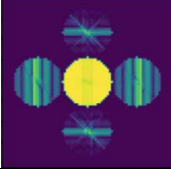
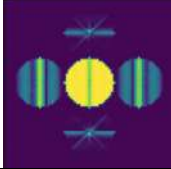

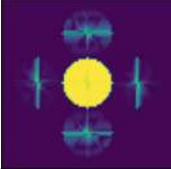
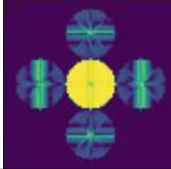

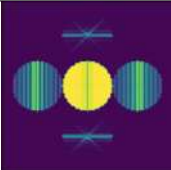
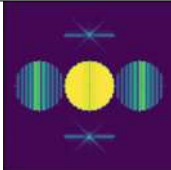

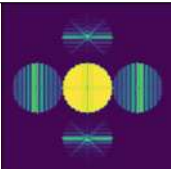
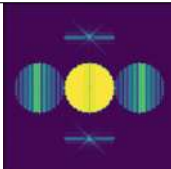
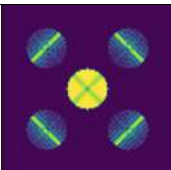
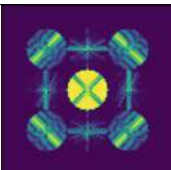
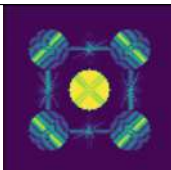
Parameter file	Experimental pattern	Simulation pattern (CNN)	Simulation pattern (ANN)
$SrTiO_3-2$			
R value	0	0.197	0.219
$SrTiO_3-3$			
R value	0	0.215	0.176
$SrTiO_3-6$			
R value	0	0.221	0.177
$SrTiO_3-7$			
R value	0	0.199	0.169
$SrTiO_3-9$			
R value	0	0.301	0.262

Table 4.11: The "best" simulation patterns of  $SrTiO_3$  that have an R value smaller than 0.3.

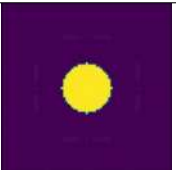
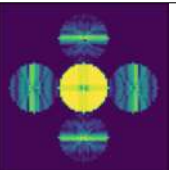
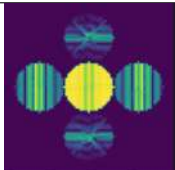

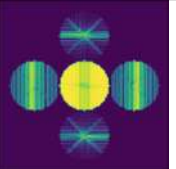
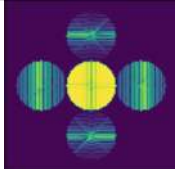

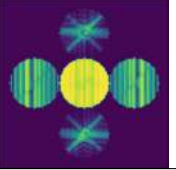
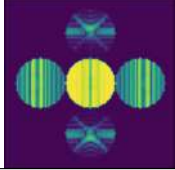
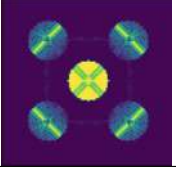
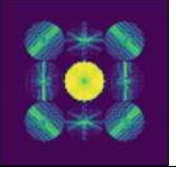
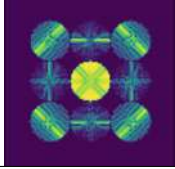
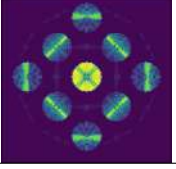
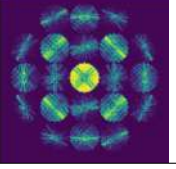
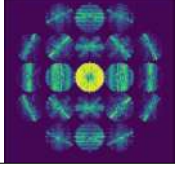
Parameter file	Experimental pattern	Simulation pattern (CNN)	Simulation pattern (ANN)
$SrTiO_3 - 1$			
R value	0	0.303	0.423
$SrTiO_3 - 4$			
R value	0	0.397	0.353
$SrTiO_3 - 5$			
R value	0	0.515	0.510
$SrTiO_3 - 8$			
R value	0	0.505	0.479
$SrTiO_3 - 10$			
R value	0	0.630	0.711

Table 4.12: The "best" simulation patterns of  $SrTiO_3$  that have an R value greater than 0.3.

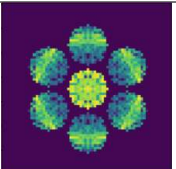
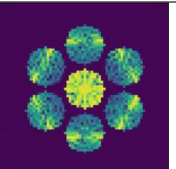
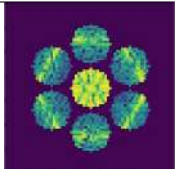
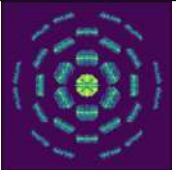
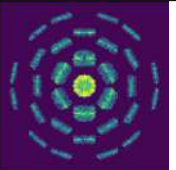
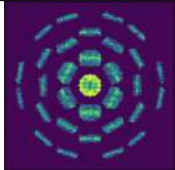
Parameter file	Experimental pattern	Simulation pattern (CNN)	Simulation pattern (ANN)
$ZnO - 1$			
R value	0	0.495	0.531
$ZnO - 2$			
R value	0	0.666	0.713

Table 4.13: The "best" simulation patterns of  $ZnO$ .

# Chapter 5

## Discussion

### 5.1 Discussion of the Training Performance

This section discusses the problems that might occur during the training, such as overfitting, local minimum etc.; in addition, the training performance of two network models are compared.

#### Overfitting

Overfitting is a central problem in supervised learning. It's defined as the phenomenon that the model learns the detail and noise in the training data rather than finding a general predictive rule, which indicated by a greater training loss with respect to the testing loss. It usually happens when the training data is insufficient or a neural network is too complex [34][35]. Thanks to the shortcut in the ResNet architecture, a CNN with 50-layers is successfully trained on 100 training data without overfitting. The ANN also avoids the overfitting by using the shallow models. Further more, an added random offset also prevents the training from overfitting. From the model loss graphs of the two tasks (Figure 4.1 and Figure 4.7), it can be observed that both the training and the testing loss converge to a same level. It means that both models have successfully learned the general rules of the training data set that are also applicable for the new data set.

#### Local Minimum & Saddle Point

However, although the overfitting problem is avoided, the initial training loss of Task 2 (taking  $SrTiO_3 - 1$  as example) can only be reduced to 0.007, which is much higher than  $e^{-6}$  (achieved in Task 1). On the one hand, completely random diffraction data are used in the initial training of Task 2, which makes the neural network hard to train. On the other hand, we need to consider whether the problem of falling into a saddle point or local minimum in the learning process occurs.

The gradient descent methods do not undergo further change when they reach a local minimum of a non-convex function because the first derivative of the cost function is zero at a local minimum point [36][37]. Although the MSE cost function used in this project is convex in terms of parameters, it is not convex anymore in a neural network due to the nonlinear activation function. The Adam algorithms

solved the problem of trapping in a local minimum during the training by applying the momentum method and controlling the distance of each learning step.

In addition to the local minimum problem, the network training process may also be trapped in a saddle point. The gradient value vanishes at the saddle point. Through shuffling the mini-batches of the training set, it might be solved by Adam. By calculating the gradient and momentum in different directions, the optimal route might be found to escape the saddle point.

By iteratively replacing the random diffraction data of the initial training set with the predicted ones, the training and testing loss of models are continuously reduced to  $e^{-4}$  and  $e^{-5}$ . The reason why the final training loss in Task 2 is still higher than the value in Task 1 is that the training data used in Task 1 is to some extent similar to the same experimental pattern. The new training data used to replace the initial training data in Task 2 is only similar to the previous prediction of the neural network, but not necessarily similar to the experimental data to the same extent. The network is easier to train with a set of data of a higher similarity.

## CNN v.s. ANN

According to the prediction results of both tasks, ANN model is comparable with the CNN model. For different LARBED data, CNN and ANN both give  $U_g$  predictions, whose simulation patterns carry a relatively consistent R value. The architecture of the ANN model is much simpler than that of CNN. However, there are some advantages of the CNN model over the ANN model:

1. Efficiency: Even though the CNN model is much deeper than the ANN model, it takes less time to analyze the LARBED data that contains more information, because CNN has a reduced number of parameters due to parameter sharing and sparse connections. The transfer learning method and the ResNet structure used in the CNN model also speed up the learning time in the deeper network layers.
2. Stability: Since the CNN model is trained based on a well trained initial weights through transfer learning, it is much more stable than the ANN that is trained from random initial weights. In addition, the CNN model is much deeper and avoids degradation problem through shortcuts. For a more complex input, CNN will give a more accurate prediction.

## 5.2 Discussion of the Prediction

In this section, factors influencing the network performance are discussed, such as the choice of the target function, the type of the crystal categories etc.. In addition the influence of diffraction parameters including the tilt range, the number of tilt, the number of beams and the sample thickness are discussed.

### 5.2.1 Factors influencing Prediction Performance

Testing our neural network models on several LARBED data, we find that only for patterns with a large tilt range and a small number of beams, the neural network is able to give an accurate prediction of structure factors, i.e., a final R value less than 0.3. The main reason for the underwhelming prediction performance may be the inconsistent choice of target function and the evaluation function.

#### Target Function & Evaluation Function

The neural network gives the optimal parameters through minimizing the MSE function, which calculates the averaged squared difference between the true structure factors and thickness vectors against the predicted ones:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \|U_{g-t_{pred}^{(i)}}(\theta) - U_{g-t^{(i)}}\|^2. \quad (5.1)$$

Where  $m$  is the size of a mini-batch (in our case  $m = 20$ ),  $\theta$  are the learnable parameters. However, the evaluation function for the prediction results is R-factor, which calculate the difference between experimental pattern and simulation pattern:

$$R = \frac{\sum |I^{exp} - I^{sim}|}{\sum I^{exp}}. \quad (5.2)$$

The model is dedicated to train the network to predict a set of structure factors closest to the true value. However, for a real experimental pattern, we can only evaluate the network performance by comparing the difference between the simulation pattern and the experimental pattern. Although a simulation pattern with a low R value always corresponds to an accurate structure factors-thickness vector, a structure factors-thickness vector with a smaller error does not necessarily correspond to a simulation pattern with a smaller R value. This can be proved by the R-iteration curve obtained in Task 2 (Figure 4.11). As the training error becomes smaller and smaller with increasing number of iterations, the R value does not gradually decrease.

#### Thickness Prediction

The thickness prediction of the neural network models is far from accurate. Regardless of the true sample thickness, the network models always give a value around 50 nm. The reason might be that, the sample thickness is not predicted

separately, but with the structure factors. For the  $SrTiO_3-1$  pattern constructed by 13 beams, the output size of the neural network, i.e., the size of the structure factors-thickness vector is 27. However, only one of the 27 neurons is used to calculate the thickness, this means that the prediction error of thickness contributes little during the back propagation. We know that the crystal thickness is crucial in determining the intensity of diffraction pattern [14], but the network doesn't treat it as such. The lack of accuracy in the weights of the sample thickness causes the network to awkwardly predict the thickness as the average value of the true label, i.e., the average thickness of the random patterns. For Task 1, the training data generated based on the target data has a thickness close to the true value, so the thickness prediction in Task 1 is always correct.

In multi-output regression, the outputs are typically dependent not only on the input but also each other [38] [39]. Hence an extremely inaccurate prediction of thickness certainly affects the prediction of the structure factors. A straightforward way to solve this problem is to use another neural network or an alternative method to predict the sample thickness separately. However, one drawback of this solution is that the structure factors and sample thickness cannot be learned at the same time, despite the fact that they are correlated.

### 5.2.2 Diffraction Parameters

The prediction results of Task 2 show that the neural network method is not sensitive to the sample thickness and the number of tilts, whereas the tilt range and the number of beams affect the prediction performance of the neural network substantially. We summarize the impact of all diffraction parameters in Task 2 below:

Maximum Tilt Range	the larger the better
Number of Beams	the smaller the better
Number of tilts	no influence
Thickness	no influence

Table 5.1: Parameters influencing Network Performance

(Remark: Although no influence was seen with the given parameters and network model in this work, the thickness and number of tilts will definitely have an impact.)

#### Maximum Tilt Range

The prediction results show that the prediction accuracy increases with an increasing tilt range. Among the tested LARBED patterns, only the simulation patterns with a largest tilt range of 100 *mrad* show a good agreement with the experimental patterns. This result is reasonable because in LARBED, the discs are produced by tilting the incident beam and recording the diffraction patterns. More diffraction information is gained by an increasing range of illumination tilt angles [5]. Richer information helps neural networks to learn better.



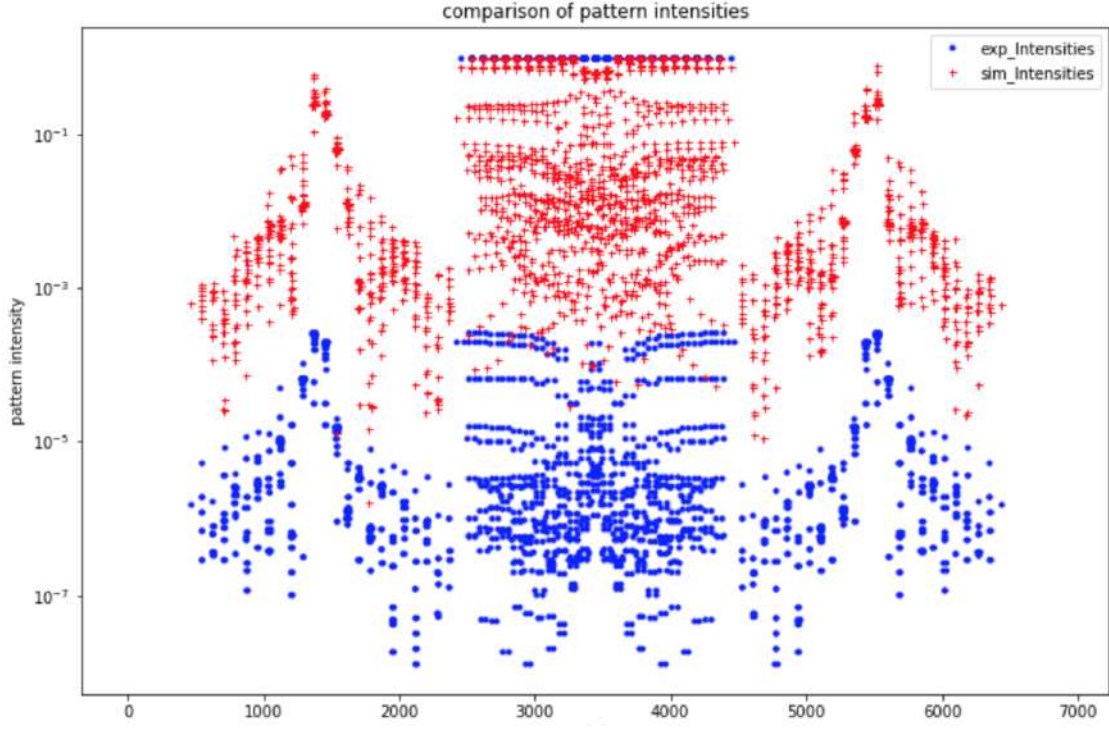
## Number of Beams

The prediction performance of the neural network gets worse with an increasing number of beams. Since the size of the network's output is the number of structure factors of the LARBED pattern plus one, more neurons are required in the output layer for a pattern formed by more beams. The training process will be more difficult, because the network not only needs to predict more targets, but also to consider more complex relationships between these targets. Moreover, the thickness prediction could get worse for the same reason.

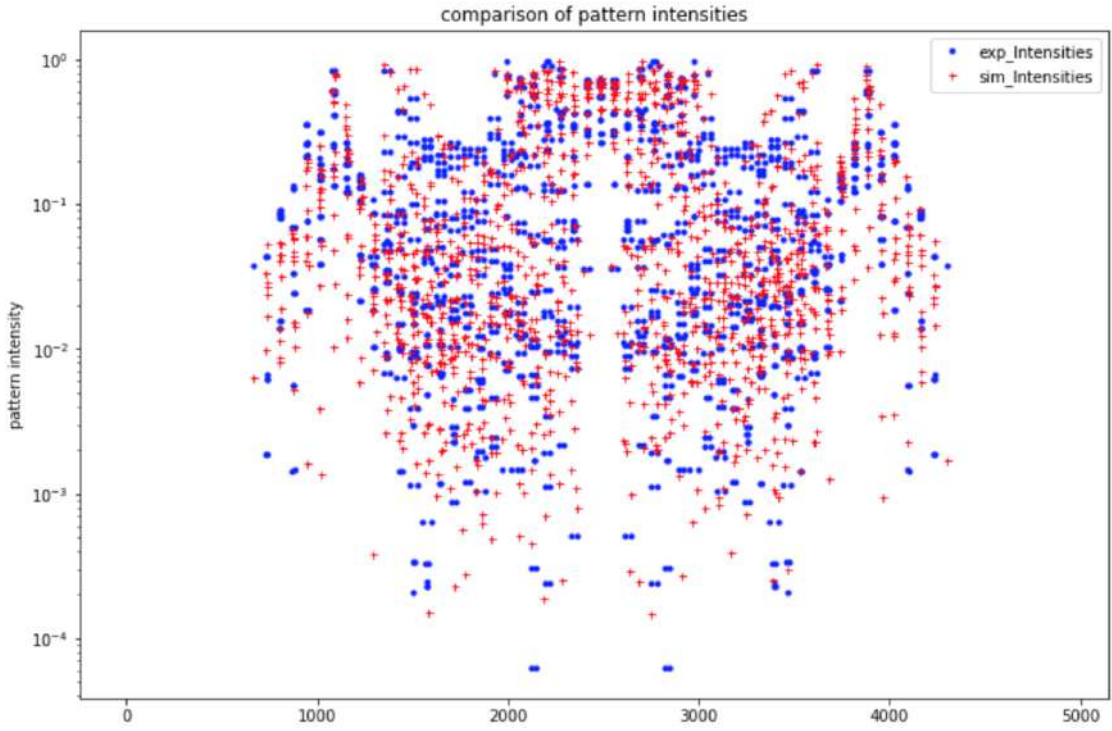
### 5.2.3 $ZnO$ v.s. $SrTiO_3$

By comparing the simulation patterns of  $ZnO$  and  $SrTiO_3$  in Task 2, we found that a simulation pattern of  $ZnO$  looks more similar to the experimental pattern than  $SrTiO_3$  for a comparable R value. That because  $SrTiO_3$  patterns have more vanishing intensities than  $ZnO$  patterns, and the neural network has a worse performance to analyze the weak intensity part of the pattern. Figure 5.1 shows a detailed comparison of the simulation pattern and experimental pattern for a  $SrTiO_3$  pattern and a  $ZnO$  pattern. The blue points and red points represent the value of integrated intensities of the experimental pattern and simulation pattern, respectively. Although the simulation pattern  $SrTiO_3 - 1$  (Figure 5.1 (a)) has a lower R value, the weak intensity points have a bad match with the experimental pattern. The simulation value for an intensity point of magnitude  $10^{-5}$  is up to  $10^{-1}$ . As contrast, a simulation pattern of  $ZnO$  carrying a R value of 0.495 matches the experimental pattern better overall. The major difference between the simulation pattern and the experimental pattern of  $SrTiO_3$  at weak intensities can also be observed in Table 4.11 and Table 4.12. All the weak intensities in the experimental pattern, corresponding to weak reflections with index odd sum ( $h + k + l$ ), appear strong in the simulation pattern. However, the R factor calculated as  $R = \frac{\sum |I^{exp} - I^{sim}|}{\sum I^{exp}}$  is dominated by the error in strong intensities. That's why a  $SrTiO_3$  pattern always has a lower R value than  $ZnO$ .

From the results of Task 1, we come to the conclusion that for all patterns of the same crystal category, the maximal acceptable deviation scale determined by the models are same. From this result, it can be inferred that the acceptable "randomness" of the training data for a correct prediction depends on the crystal category. For diffraction patterns that contain more intense reflections, the model has a higher tolerance for the training data, i.e., the training data can be more random.



(a)  $SrTiO_3 - 1$ ,  $R=0.303$



(b)  $ZnO - 1$ ,  $R=0.495$

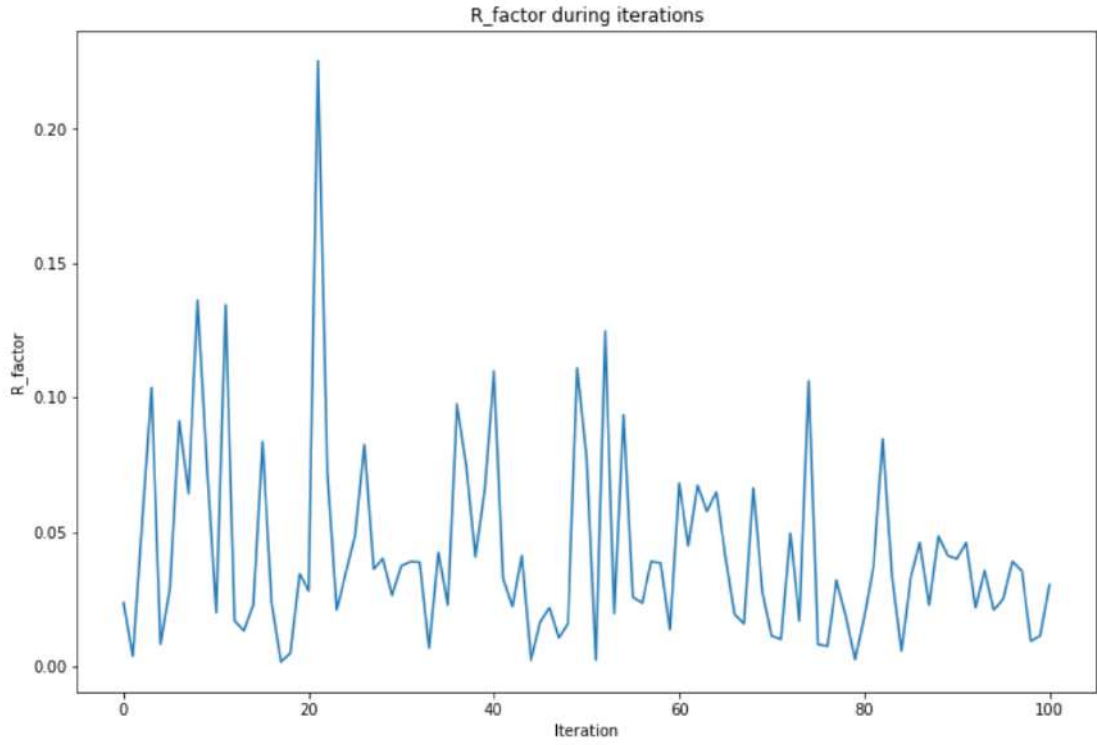
Figure 5.1: Details of the integrated intensities in simulation pattern and experimental pattern for (a)  $SrTiO_3 - 1$  pattern and (b)  $ZnO - 1$  pattern. The blue points and red points in both figures represent the value of intensities of the experimental pattern and simulation pattern, respectively. The simulation pattern of  $SrTiO_3 - 1$  constructed from 13 beams has a  $R$  value of 0.303, the simulation of  $ZnO - 1$  constructed from 95 beams has a  $R$  value of 0.495.

### 5.3 Discussion for the Fundamental Idea

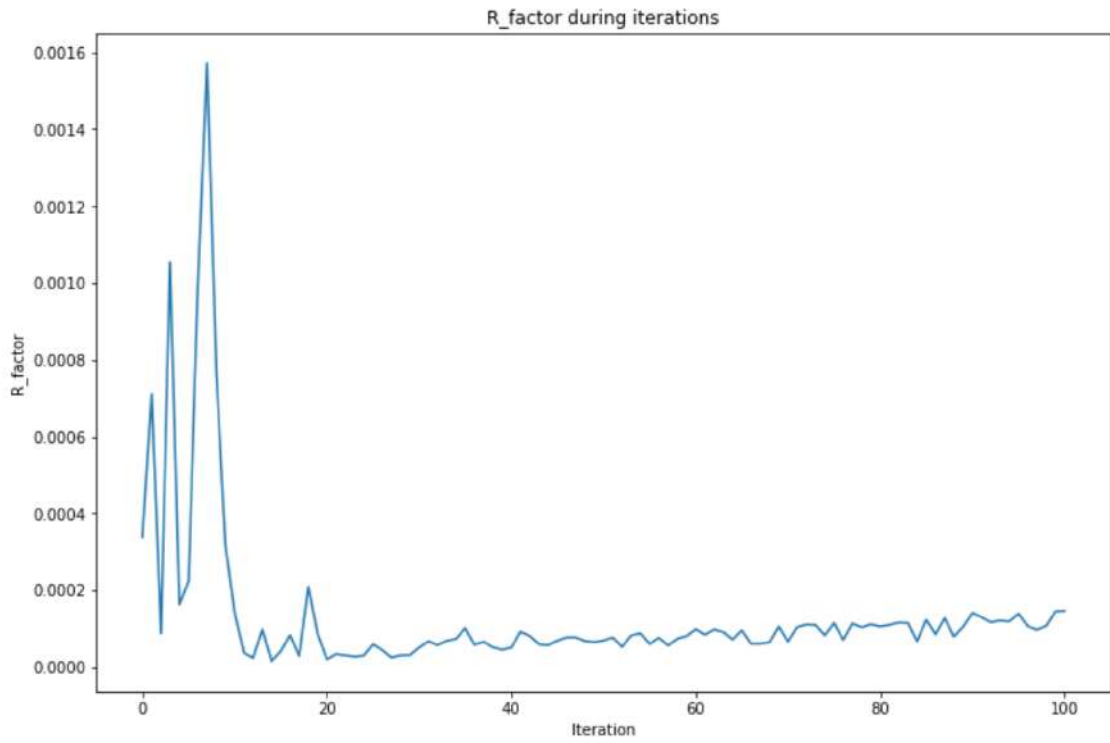
In Task 2, a loop is applied to replace the original random training data with new training data and retrain the neural network. However, the results of the task are not as good as we expected. In this section, we discuss the utility of this idea.

In the case of  $SrTiO_3 - 1$ , a training set deviated from the experimental data at a scale larger than 10.24 increases the R value of the simulation pattern (shown in Figure 4.6) rapidly. To investigate whether adding new training data can increase the acceptable deviation scale, we initialize the algorithm of Task 2 with the training set at the scale of 10.24 and 20.48 from Task 1 instead of the randomly generated training set. The R-iteration curves are shown in Figure 5.2 and 5.3. The R value becomes smaller as new training data is added, which is more evident in the case of ANN. The smallest R values are 0.0016 for CNN and 0.0001 for ANN at a deviation scale of 10.24, and 0.0008 for CNN and 0.0005 for ANN at a deviation scale of 20.48. The "best" simulation patterns with the smallest R values are shown in Figure 5.4 with the experimental pattern. Compared to the results in Task 1 with the same deviation scale of 10.24 (see Table 4.1), the reconstruction of the LARBED pattern is significantly improved by gradually replacing the initial training set.

The change of R value presented in Figure 4.6 are shown again in Figure 5.5 with the new data points. The new points are clearly labelled, showing that the R values for the large deviation scale of 10.24 and 20.48 are still small. We can conclude that the acceptable deviation scale from Task 1 can be larger with this procedure. This result also indicates that retraining the network with newly added pairs of  $\{I_{pred}^{exp}, U_g - t_{pred}^{exp}\}$  in Task 2 can, in principle, help the neural network to give a better prediction.

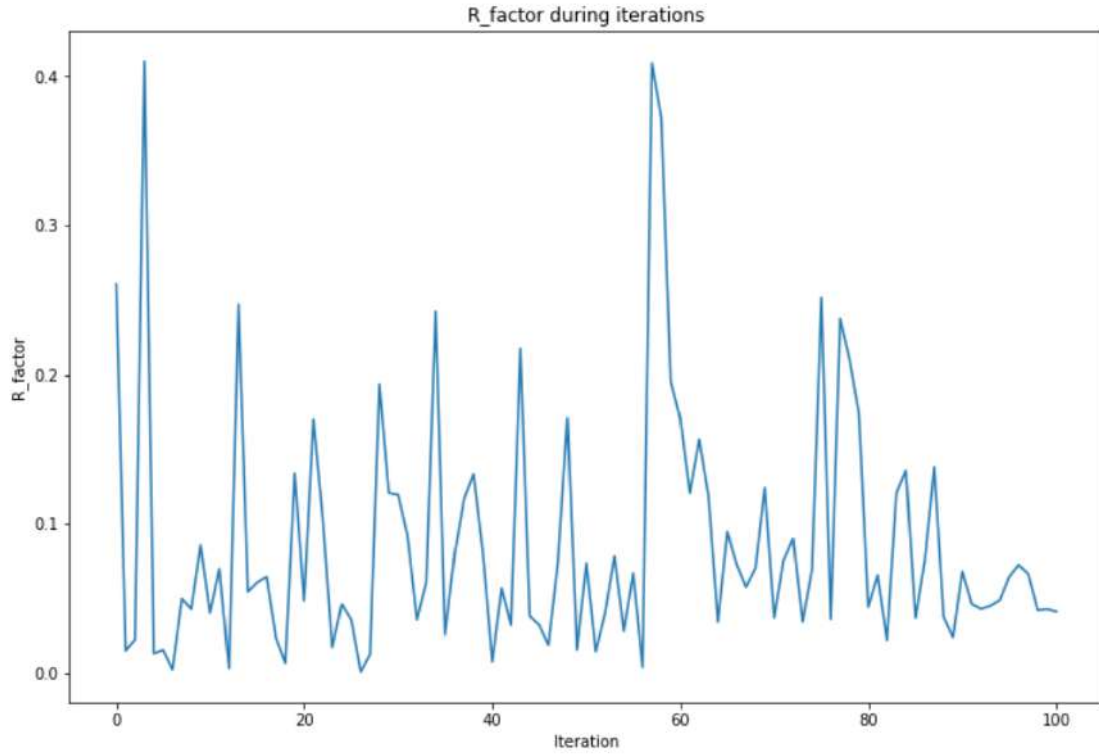


(a) CNN

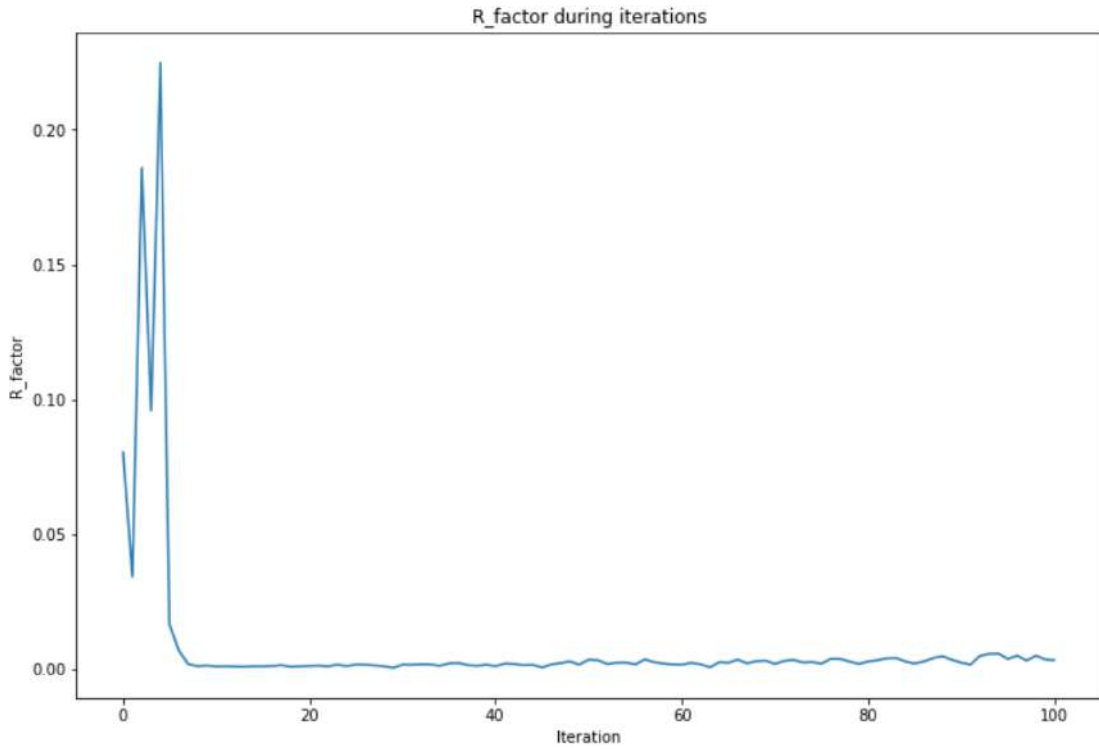


(b) ANN

Figure 5.2: R-iteration curve for (a) CNN and (b) ANN. The R factors are calculated between the simulation and experimental patterns of  $SrTiO_3 - 1$ . The simulation patterns are generated following the procedure of Task 2 with an initial training set at a deviation scale of 10.24 from Task 1.



(a) CNN



(b) ANN

Figure 5.3: R-iteration curve for (a) CNN and (b) ANN. The simulation patterns are generated following the procedure of Task 2 with an initial training set at a deviation scale of 20.48 from Task 1.

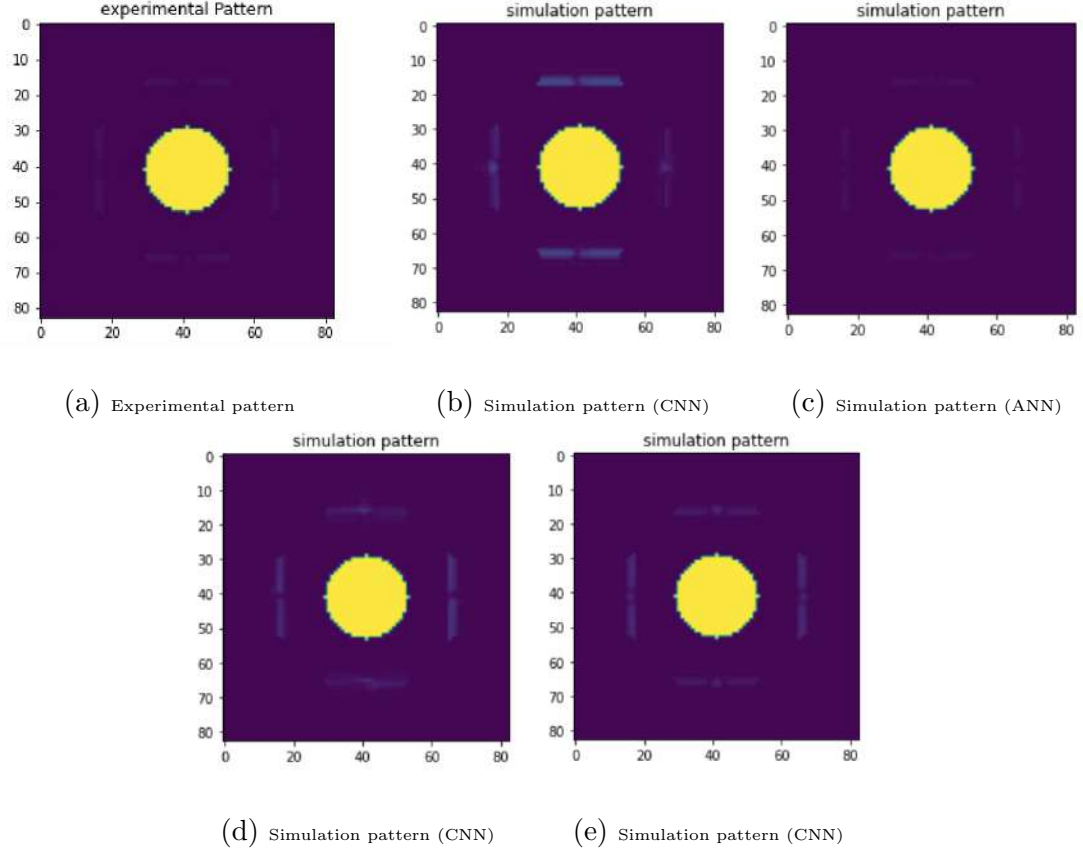
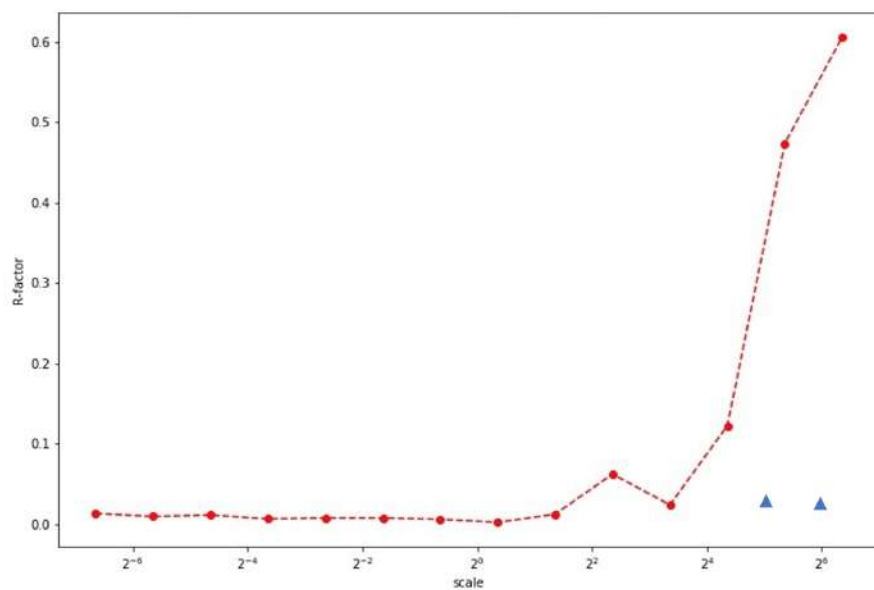
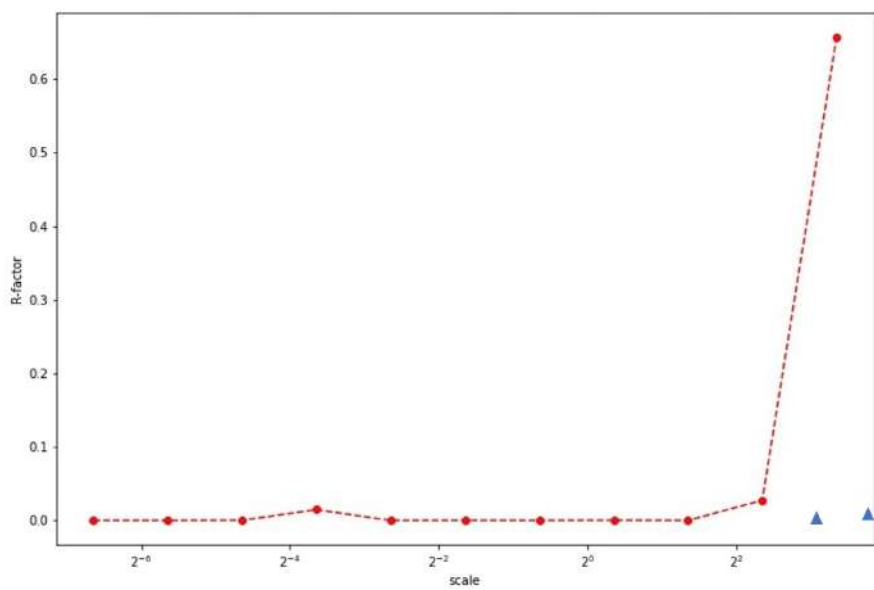


Figure 5.4: (a) The experimental pattern of  $SrTiO_3 - 1$ ; (b)(c) the best simulation patterns obtained by CNN and ANN at a deviation scale of 10.24, which have a R value of 0.0016 and 0.0001 respectively; (d)(e) the best simulation patterns obtained by CNN and ANN at a deviation scale of 20.48, which have a R value of 0.0008 and 0.0005 respectively.



(a) CNN



(b) ANN

Figure 5.5: Change of the R-factor value calculated from the simulation and the experimental pattern with increasing deviation scale. The new R values calculated in this Section are labelled in triangles.

# Chapter 6

## Conclusion and Outlook

In this project, two neural network models have been provided to determine the structure factors from LARBED patterns. The first model is a CNN with a ResNet-50 architecture, the second model is a simple ANN. Both networks have been well trained for the prediction in two tasks.

In Task 1, the capability of the neural network approach to determine correct structure factors has been explored. Both networks are able to obtain a correct solution of an experimental pattern if the training data has a deviation scale smaller than 5 from the target data. However, in the section 5.3, we proved that it is able to improve the scale of deviation by replacing the initial training set with the newly added pairs of  $\{I_{pred}^{exp}, U_g - t_{pred}^{exp}\}$ . In Task 2, the networks have been applied to determine structure factors and the sample thickness of several experimental LARBED patterns, which are constructed by different diffraction parameters. From the prediction results we demonstrated that the networks make a more accurate estimation of structure factors for a LARBED pattern formed by less beams with a larger tilt range. They are able to achieve a reasonable prediction accuracy ( $R \leq 0.3$ ) in analyzing LARBED patterns covering a maximum tilt range of 100 *mrad*. Both networks have equivalent prediction performances in both tasks.

A number of avenues for future work have been proposed to further improve the network performance. Since an incorrect prediction of sample thickness affects the determination of structure factors, a separate network or an alternative technique can be used to determine the sample thickness. In both tasks, this neural network approach shows a different prediction performance for different crystal categories. It would also be meaningful to apply this approach to other crystal categories. In addition, neural networks can be combined with other machine learning methods to get a better estimation of structure factors.



## Acknowledgements

First of all, I would like to express my appreciation to my supervisor Prof. Christoph T. Koch who provides me a position to complete my master thesis in his group. He gives me guidance and suggestions with patience all the time.

Then I would like to extend my sincere thanks to Prof. Dr.-Ing. Peter Eisert for taking over the second advisor for this work.

Besides, I would like to extend my sincere thanks to my colleague, Sam Fairman, who helps me throughout my thesis work. The completion of my master thesis would not have been possible without his support.

My sincere thanks also goes to Dr. Markus Kühbach and Marcel Schloz for helping me with the neural network architecture.

Finally, I would like to thank the whole "Structure Research and Electron Microscopy" group for their friendness and encouragement in the last one year.

## Statement of authorship

I declare that I completed this thesis on my own and that the information which has been directly or indirectly taken from other sources has been noted as such. Neither this nor a similar work has been presented to an examination committee.

Berlin, 07.01.2022

Jie Chen 陈洁 07.01.2022

Jie Chen

# Bibliography

- [1] Dajun Dua, Kang Li, and Minrui Fei. “Electron Microdiffraction”. In: *ScienceDirect* 26 (1978), pp. 1–53.
- [2] Renliang Yuan et al. “Training artificial neural networks for precision orientation and strain mapping using 4D electron diffraction datasets”. In: *Ultramicroscopy* 113256 (2021).
- [3] R. Beanland et al. “Digital electron diffraction - seeing the whole picture”. In: *Acta Cryst.* **A69** (2013), pp. 427–434.
- [4] Feng Wang, Robert S. Pennington, and Christoph T. Koch. “Inversion of Dynamical Scattering from Large-Angle Rocking-Beam Electron Diffraction Patterns”. In: *Phys. Rev. Lett.* **117**.015501 (2016).
- [5] Christoph T.Koch. “Aberration-compensated large-angle rocking-beam electron diffraction”. In: *Ultramicroscopy* **111**.2011 (1979), pp. 828–840.
- [6] Christoph T. Koch, V. Burak Özdöl, and Kazuo Ishizuka. “Quantitative Four-Dimensional Electron Diffraction in the TEM”. In: *Wiley Analytical Science* (2012).
- [7] Z.Ding, E.Pascal, and M.De Graef. “Indexing of electron back-scatter diffraction patterns using a convolutional neural network”. In: *Acta Materialia* 199 (2020), pp. 370–382.
- [8] Kevin Kaufmann et al. “Paradigm shift in electron-based crystallography via machine learning”. In: *Materials Science* 367 (2020), pp. 564–568.
- [9] Alex Foden, Alessandro Previero, and Thomas Benjamin Britton. “Advances in electron backscatter diffraction”. In: *Computational Physics* (2019).
- [10] Christoph T. Koch. “Determination of Core Structure Periodicity and Point Defect Density along Dislocations”. PhD thesis. Arizona State University, (2002).
- [11] B.K.Vainshtein. “Structure analysis by electron diffraction”. In: *Pergamon press* (1964).
- [12] Thomas E. Weirich, János L. Lábár, and Xiaodong Zou. “Electron Crystallography: Novel Approaches for Structure Determination of Nanosized Materials”. In: *NATO Science Series* (2004).
- [13] Christoph T. Koch and V. Burak Özdöl. “4D electron diffraction by large-angle rocking-beam electron diffraction”. In: (2012).
- [14] M. Vijayalakshmi, S. Saroja, and R. Mythili. “Convergent beam electron diffraction — A novel technique for materials characterisation at sub-microscopic levels”. In: *Sadhana* **28**.2011 (2003), pp. 763–782.

- [15] R.F. Egerton. “Physical Principles of Electron Microscopy”. In: *Springer* (2016).
- [16] David B. WilliamsC. and Barry Carter. *The Transmission Electron Microscope*. Springer, (2009).
- [17] C J Humphreys. “The scattering of fast electrons by crystals”. In: *Rep. Prog. Phys.* **42**.1825 (1979).
- [18] Jerome Karle. “Recovering phase information from intensity data”. In: *Nobel lecture* (1985).
- [19] John M.Cowley. “Diffraction Physics”. In: *NH PL* (1995).
- [20] Zhihua Zhou. *Machine Learning*. Qinghua university press, (2016).
- [21] Yann A. LeCun et al. *Efficient BackProp*. Springer, (1998).
- [22] Diederik P. Kingma and Jimmy Lei Ba. “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION”. In: *ICLR* (2015).
- [23] Kylee Santos and Shashank Ojha. “Parallelizing Gradient Descent”. In: (2018).
- [24] Leon Bottou. “Stochastic Gradient Learning in Neural Networks”. In: *Neuro-Nimes* (1991).
- [25] Alex Yu. “How To Teach A Computer To See With Convolutional Neural Networks”. In: (2018).
- [26] Jürgen Adamy. *Fuzzy Logik, Neuronale Netze und Evolutionäre Algorithmen*. Shaker, (2011).
- [27] Li Yin. “A Summary of Neural Network Layers”. In: (2018).
- [28] Afshine Amidi and Shervine Amidi. “Convolutional Neural Networks cheat-sheet”. In: (2018).
- [29] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CVPR* (2016), pp. 770–778.
- [30] Kaiming He et al. *Identity Mappings in Deep Residual Networks*. Springer, (2016).
- [31] Stéphane Lathuilière et al. “A Comprehensive Analysis of Deep Regression”. In: *IEEE* **42.9** (2019), pp. 2065–2081.
- [32] Emilio Soria Olivas et al. *Handbook Of Research On Machine Learning Applications and Trends*. Information Science Reference, (2009).
- [33] James Bergstra, Dan Yamins, and David D. Cox. “Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms”. In: *PYTHON IN SCIENCE CONF.* (2013), pp. 13–20.
- [34] Tom Dietterich. “Overfitting and Undercomputing in Machine Learning”. In: (1995).
- [35] JIN Long, KUANG Xueyuan, HUANG Haihong, et al. “Study on the Overfitting of the Artificial Neural Network Forecasting Model”. In: *ACTA METEOROLOGICA SINICA* **19** (2004).
- [36] Dokkyun Yi, Jaehyun Ahn, and Sangmin Ji. “An Effective Optimization Method for Machine Learning Based on ADAM”. In: *Applied Sciences* (2020).
- [37] Imran Khan Mohd Jais, Amelia Ritahani Ismail, and Syed Qamrun Nisa. “Adam Optimization Algorithm for Wide and Deep Neural Network”. In: *KEDS* **2.1** (2019).

- [38] Hanen Borchani et al. “A survey on multi-output regression”. In: *IREs Data Mining Knowl Discov* (2015).
- [39] Dajun Dua, Kang Li, and Minrui Fei. “A fast multi-output RBF neural network construction method”. In: *Neurocomputing* 73 (2010), pp. 2196–2202.