Neural Optical Beam Propagation

Chengkun Li

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Master of Science
of
University College London.

Department of Computer Science
University College London

September 12, 2021

I, Chengkun Li, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

The calculation of the beam propagation is highly demanded in computer-generated holography (CGH) and lensless imaging problems. However, some unexpected factors which are hard to control like the tilted angle between the source plane and observation plane may cause problems for the optical system. Thus, we propose a coarse-to-fine architecture of deep neural network (DNN) to simulate the beam propagation. Unlike the traditional mathematic method, the parameters of beam propagation should be pre-defined. The deep learning (DL) method could overcome this problem and directly learn all of these parameters, even some unexpected factors which we ignored before from the dataset collected. And our experiments show the great performance of our method.

Impact Statement

The final goal of our project is to simulate the beam propagation on the tilted planes by a DL method. The great ability of deep learning allows us to even fit this so much complex physical process, the beam propagation. In CGH and lensless researches, there is a wide requirement for the calculation of beam propagation. And always, it is very hard to consider all of the parameters of the mathematic function which may cause a mismatch between a real optical system and the mathematic model. However, our DL method will help overcome this problem and directly lean all of the factors from the dataset collected on the certain system. Thus, our method could be used in any place where the calculation of beam propagation is demanded in theory. Inside academia, it may improve the researches of VR and Holographic technology. And outside academia, our method may be used in the lensless camera field in the future. Besides, we wish to generate several papers related to this project and seek cooperation in academia and industry based on this technology.

Acknowledgements

During the whole project, Dr. Kaan Aksit gave me a lot of help and took great patience to answer my questions and guide the progress of the project. I would also like to thank the computer department of University College London for providing hardware devices for my research. Without their assistance, there is no chance for finishing this dissertation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the past few years, with the development of Virtual Reality (VR) and Augment Reality (AR), holography is also beginning to attract researchers' attention again. As a 3D stereoscopic display technology, holography can provide the most realistic three-dimensional visual effects for the human eye, accurate depth information and other information of a 3D object. Besides, comparing with other 3D stereoscopic display technology, it does not require wearing special viewing equipment which will cause visual fatigue. Thanks to all of these advantages, it has been the most important technology in the 3D displaying field [5].

Conventional optical holography uses the principle of interference of light waves to record the object light information by introducing a reference light wave that is coherent with the object light wave. The amplitude and phase of the object wave are recorded on a medium in the form of an interferometric fringe pattern. Then, the hologram (interferogram) is irradiated with reference light or its conjugate light to reproduce the original object wave which includes all of the information of the 3D object, by using the diffraction principle of the light wave. However, it means we need a very complex and robust optical system and complicated chemical treatment to deal with the recording medium. With the advent of the digital computer and spatial light modulators (SLMs), computer-generated holography is proposed to address these problems [6]. It uses a computer to perform numerical calculations to encode and generate holograms. It avoids the complex optical system and reduces the requirement for the accuracy of optical equipment. Also, it can produce various

virtual holograms of objects which even do not exist in the real world. Besides, low noise and high repeatability make it more flexible and easy, compared to the traditional optical holography.

In the past few years, lots of work focuses on phase retrieval problems. For example, Gerchberg-Saxton (GS) algorithm uses the iterative Fourier transformation method to get the hologram from the target image [7]. The phase image can also be obtained by the transport of intensity equation (TIE) [8]. Recently, the deep learning method has shown great capacity in many fields, like computer vision, robotics and so on. People can address a very hard question directly by using an end-to-end network easily. It makes the model learn the complex features and discover the potential relationships of the data in the latent space automatically. Thus, this method is also beginning to be adopted in holography and many works have been done [9, 10]. All of these works are been done base on the wave propagation function, such as angular spectrum function, the Rayleigh-Somrnerfeld formulation and the Kirchhoff formulation of diffraction [11]. However, the computational complexity and runtime are huge. The beam propagation is complex. Although there are already some good approximation functions to describe the optical propagation phenomena, the model mismatch between the real physical display system and the ideal optical propagation function will also cause problems like noise artifacts in the final result [12]. Considering the great power of deep learning. We thought it is necessary to do some researches on simulating the beam propagation by this method. In this way, not only we can use GPU to accelerate the computation of this optical process, but also can avoid mismatch problems like the tilted angle of the diffraction plane or non-linear parameters of SLMs.

In CGH, the conventional method is called the point-based method, which considers the 3D object as consisting of many point sources and the hologram is the end result of the sum of these point's optical fields [13]. However, the final display quality depends on the number of point sources, and this approach is too inefficient and computationally intensive for complex three-dimensional objects. Then, the polygon-based method is proposed to speed up the calculation, by making the 3D

object as many polygonal faces rather than the points [14]. These faces are not parallel with the hologram plane. In addition to the phase retrieval question for the 3D object, the hologram of 2D planar objects is also an important research area. And, it is also common that the hologram plane is tilted with an angle in practice. So, we argue that simulating the beam diffraction from a plane to a tiled hologram plane will be more general for the majority of cases.

In this paper, we do some researches to calculate the optical diffraction on the tilted planes directly by the deep learning method, and develop a net architecture for this problem. Unlike other research hot points in CGH, they pay more attention to the phase-retrieval questions which are actually doing the inverse beam propagation job based on the basic physical-digital wave propagation model. Here, we do the forward things and provide a new idea for doing this area. And as we know, we are the first to do this job with the deep learning approach. And the experiment results show us the powerful ability to simulate wave propagation in this way.

In summary, our contributions are shown below:

- By several experiments, we firstly demonstrate that the deep learning method has a great capacity to simulate the beam propagation.
- We propose a coarse-to-fine frame to calculate the beam diffraction pattern on the tilted plane. And in our experiments, our network shows a great performance for simulating beam propagation .
- We do some explorations to extend our network, and make it suitable for an exterior input, the tilted angle between the hologram plane and the source plane.

Chapter 2

Background

In this chapter, we provide an overview of some fundamental math theories and related technologies which would be used in our research for better comprehension. Firstly, we do the introduction of scalar diffraction theory in Section 2.1, and make a comparison between three basic wave propagation functions. Then, the outline of deep learning is provided in Section 2.2.

2.1 Scalar Diffraction Theory

One core part of CGH is doing the simulation of wave propagation in computers. The fundamental basis of these CGH algorithms is the Rayleigh-Somaffield diffraction theory and the angular spectrum (AS) theory, which respectively describes the diffraction process in the time and frequency domain [11]. Basing on this basic theory and implementation methods, diffraction algorithms can be divided into three types: the Fresnel approximation, convolution method and angular spectrum methods. All of them are called the basic wave diffraction algorithm. In the following, we first introduce the basic theory of numerical calculation. Then the three algorithms are described in detail separately. Besides, we introduce the discretization method for implementation in practice. Finally, we make a summary and comparison between these basic mathematical algorithms for calculating the beam propagation.

2.1.1 The Rayleigh-Somrnerfeld Diffraction Theory

In the scope of scalar diffraction, the Rayleigh-Somaffield diffraction formula is considered as the exact solution of the diffraction problem. Fig.2.1 shows the

diffraction process. We assume that the light wave propagates from the source plane (the $(x_s, y_s, 0)$ plane) to the observation plane (the (x_o, y_o, d) plane). Here, we set the distance between the source plane and observation plane as d. Then the Rayleigh-Somrnerfeld diffraction theory can be described as following mathematical function:

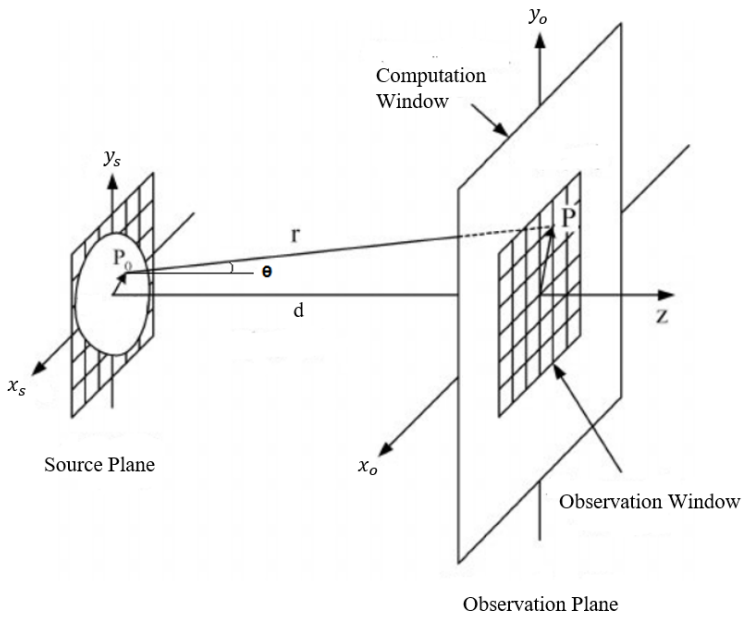


Figure 2.1: Scheme for the Rayleigh-Somrnerfeld diffraction theory.

$$U(x_o, y_o; d) = \frac{1}{j\lambda} \iint_{\Sigma} U(x_s, y_s; 0) \frac{\exp(jkr)}{r} \cos\theta dx_s dy_s \tag{2.1}$$

Where r is the Cartesian distance between point $(x_o, y_o; d)$ and $(x_s, y_s; 0)$. λ is the wave length of the light field and $k = \frac{2\pi}{\lambda}$ is wavenumber. Σ is the recording region of source plane. θ is the angle which has been shown in Fig.2.1. $U(x_s, y_s; 0)$ is the emergent complex optical field of the source plane, and $U(x_o, y_o; d)$ is the complex optical field which we can obtain on the observation plane. Both of them can be represent by its amplitude (A) and phase (ϕ) filed, for example:

$$U(x, y; 0) = A(x, y; 0) exp[j\phi(x, y; 0)] \tag{2.2}$$

2.1.2 The Fresnel Approximation

When the propagation distance d is far greater that the size of observation window. And the Fresnel approximation condition is satisfied, as shown below.

$$d^3 \gg \frac{\pi}{4\lambda}\left[(x_o - x_s)^2 + (y_o - y_s)^2\right]^2_{\max} \tag{2.3}$$

Then, r which is in the denominator part of the Eq 2.1 can be replace by distance d. However, mall changes in the molecule r will produce large phase errors and therefore cannot directly replace with d. With the help of binomial expansion of the square root, we can rewrite this parameter r as below:

$$r = \sqrt{d^2 + (x_o - x_s)^2 + (y_o - y_s)^2} \approx d + \frac{(x_o - x_s)^2}{2d} + \frac{(y_o - y_s)^2}{2d} \tag{2.4}$$

At the same time, the paraxial approximation is also satisfied, i.e. $\cos\theta \approx 1$. Thus, Eq 2.1 can be rewrite as,

$$\begin{aligned} U(x_o, y_o; d) &= \frac{1}{j\lambda d}\exp(ikd)\exp\left(i\frac{k}{2d}\left(x_o^2 + y_o^2\right)\right) \\ &\times \iint_\Sigma U(x_s, y_s; 0)\exp\left(j\frac{k}{2d}\left(x_s^2 + y_s^2\right)\right)\exp\left(-j\frac{k}{d}\left(x_s x_o + y_s y_o\right)\right) dx_s dy_s \end{aligned} \tag{2.5}$$

From the above analysis, the Fresnel diffraction approximation is actually an expansion of the Rayleigh-Sommerfield diffraction which is the paraxial approximation with substituting parameter r by its secondary expansion approximation. Neglecting the irrelevant constant phase factor outside the integral, the diffraction field can be simply expressed as a two-dimensional Fourier transform multiplied by a quadratic phase factor.

$$U(x_o, y_o; d) = \exp\left[j\frac{k}{2d}\left(x_o^2 + y_o^2\right)\right]\mathcal{F}\left\{U(x_s, y_s; 0) \times \exp\left[\frac{jk}{2d}\left(x_s^2 + y_s^2\right)\right]\right\} \tag{2.6}$$

Where $\mathcal{F}\{\cdot\}$ denotes the Fourier transformation operator.

2.1.3 The Convolution Method

The propagation of light in free space can be regarded as a linear invariant system. So, the diffraction formulation Eq 2.1 can also be written as the convolution form. By $\cos\theta = \frac{d}{r}$, Eq 2.1 will be transfer to:

$$U(x_o, y_o; d) = \frac{1}{j\lambda} \iint_{\Sigma} U(x_s, y_s; 0) \frac{\exp(jkr)}{r} \frac{d}{r} dx_s dy_s \tag{2.7}$$

And, we set a new function and simplify with Fresnel approximation:

$$\begin{aligned} h(x,y) &= \frac{d}{j\lambda} \frac{\exp\left[ik\sqrt{d^2+x^2+y^2}\right]}{d^2+x^2+y^2} \\ &\approx \frac{\exp(jkd)}{j\lambda d} \exp\left[\frac{jk}{2d}\left(x^2+y^2\right)\right] \end{aligned} \tag{2.8}$$

Then, we will obtain the convolution form of the Rayleigh-Sommerfield diffraction function.

$$\begin{aligned} U(x_o, y_o; d) &= \iint_{\Sigma} U(x_s, y_s; 0)\, h(x_o - x_s, y_o - y_s)\, dx_0 dy_0 \\ &= U(x_s, y_s, 0; d) \otimes h(x_s, y_s) \end{aligned} \tag{2.9}$$

Where $\otimes$ means convolution operation. According to the Fourier transform property of the convolution. Also, we can write it as following:

$$U(x_o, y_o; d) = \mathcal{F}^{-1}\left\{\mathcal{F}\left[U(x_s, y_s; 0)\right] \cdot \mathcal{F}\left[h(x_s, y_s)\right]\right\} \tag{2.10}$$

$\mathcal{F}^{-1}\{\cdot\}$ expresses the inverse Fourier transformation operation.

2.1.4 The Angular Spectrum Algorithm

The angular spectrum of a wave filed is given By its two-dimensional Fourier Transformation, Let $G(f_x, f_y; z)$ be the angular spectrum of the disturbance $U(x, y, z)$.

$$G(f_X, f_Y; z) = \iint_{-\infty}^{\infty} U(x, y, z) \exp\left[-j2\pi\left(f_X x + f_Y y\right)\right] dxdy \tag{2.11}$$

And its inverse transformation is given by:

$$U(x,y,z) = \iint_{-\infty}^{\infty} G(f_x, f_y; z) \exp[j2\pi(f_x x + f_y y)] df_x df_y \tag{2.12}$$

The propagation of a monochromatic light must satisfy the Helmholtz equation:

$$\nabla^2 U + k^2 U = 0 \quad \text{with} \quad k = \frac{2\pi}{\lambda} \tag{2.13}$$

And then, plug Eq 2.12 into the Helmholtz equation. The following equation will be obtained.

$$\frac{d^2}{dz^2} G + (\frac{2\pi}{\lambda})^2 [1 - (\lambda f_x)^2 - (\lambda f_y)^2] G = 0 \tag{2.14}$$

Which has the solution:

$$G(f_x, f_y; z) = G(f_x, f_y; 0) \exp\left[j \frac{2\pi z}{\lambda} \sqrt{1 - (\lambda f_x)^2 - (\lambda f_y)^2} \right] \tag{2.15}$$

Directly apply Eq 2.11 and Eq 2.12 to the equation above with the Fourier transformation property of convolution. The final optical filed on the observation plane can be obtained by following function.

$$U(x_o, y_o; d) = \mathcal{F}^{-1}\{\mathcal{F}[U(x_s, y_y; 0)]\, \mathcal{H}(f_x, f_y)\} \tag{2.16}$$

Where,

$$\mathcal{H}(f_x, f_y) = \begin{cases} \exp j\frac{2\pi d}{\lambda} \sqrt{1 - (\lambda f_x)^2 - (\lambda f_y)^2}, & \text{if } \sqrt{f_x^2 + f_y^2} < \frac{1}{\lambda} \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

2.1.5 Discretization

All of these three algorithms for calculating the beam propagation are related to the Fourier transform. Actually, we use the fast Fourier transform (FFT) to implement them in practice. And in reality, people use charge-coupled devices (CCDs) to take

photos, which could sense the light and converting images into digital signals. Thus, we should do discretization for integral and Fourier transformation operation.

Basing on the CCD parameters, the source optical filed will be sampled by a $M \times N$ equidistance grid points in $L_x \times L_y$ spacial region which actually denotes resolution of the image and effective optical sensitization area respectively. We set the sampling interval as Δx and Δy along x-axis and y-axis, which could also be recognised as the pixel size of CCDs. Thus, the consecutive wave field $U(x,y;z)$ can be written as $U(m\Delta x, n\Delta y; z)$. And m,n are the ordinal numbers of sampling points, with:

$$-\frac{M}{2} \leq m \leq \frac{M}{2} - 1, \quad -\frac{N}{2} \leq n \leq \frac{N}{2} - 1 \tag{2.18}$$

With the definition above, we use discrete Fourier transform (DFT) replacing the Fourier transform, which is:

$$F_{mn} = \sum_{s=-\frac{M}{2}}^{\frac{M}{2}-1} \sum_{t=-\frac{N}{2}}^{\frac{N}{2}-1} f_{st} \exp\left[-j2\pi\left(\frac{ms}{M} + \frac{nt}{N}\right)\right] \tag{2.19}$$

We suppose $\Delta x_s, \Delta y_s$ and $\Delta x_o, \Delta y_o$ are the sampling interval of the source plane and observation plane respectively. Then based on the sampling theorem, they will satisfy the following relationship for the Fresnel approximation method.

$$\Delta x_o = \frac{\lambda d}{M \Delta x_s}, \quad \Delta y_o = \frac{\lambda d}{N \Delta y_s} \tag{2.20}$$

Although, only one Fourier transform operation is required for the Fresnel approximation algorithm, the pixel size of the final observation wave field is related to the propagation distance. The larger the propagation distance, the larger the pixel size of the final optical field and the lower the resolution.

Since the convolution algorithm for beam propagation requires two Fourier transformations and one inverse Fourier transform, this makes the source wave plane and the observation plane have the same image element size, which means:

$$\Delta x_o = \Delta x_s, \quad \Delta y_o = \Delta y_s \tag{2.21}$$

It means diffraction pattern size is independent of propagation distance. Actually, since the sampling interval of the diffraction pattern is the same as the CCD image pixel size, it is also unable to perform a fine structure smaller than the image pixel size, which finally limits the resolution and restricts the use of convolution algorithm.

As for the angular spectrum function, its case is similar to the convolution method.

2.1.6 Summary of The Diffraction Theory

The Fresnel diffraction integral algorithm, the convolution method and angular spectrum reproduction algorithms are implemented with one, three and two Fourier transforms, respectively. So in terms of runtime speed, the Fresnel diffraction integral algorithm is the fastest and the convolution algorithm is the slowest. With regard to the applying condition, the Fresnel integral and convolution methods do the Fresnel approximation, so both of them should be used under this approximation condition. As for the angular spectrum method, it strictly follows the Helmholtz equation of scalar diffraction and has no limited conditions. Besides, the diffraction resolution is related to the CCD's pixel size for the Fresnel integral. For the other two methods, it keeps the same which make them more simple.

Thus, in our experiments, we use the angular spectrum algorithm as our baseline.

2.2 Deep Learning

Recently, with the growth of computer computing and algorithms like GPU and Backpropagation (BP) algorithm, the use of deep learning has been possible. And it has shown unprecedented abilities to deal with various problems, especially in computer vision area. In general, deep learning is also a kind of end-to-end learning. In the learning process, there is no sub-module or phased training, and then it directly optimizes the overall goal of the task. The training data is generally in the form of "input-output" pairs, without requiring any other additional information. Because it does not need to explicitly give the functions of the different modules or stages and no human intervention is demanded in the intermediate process, system error will be

avoided and more simple to build.

In this part, we will make a short introduction to artificial neural networks and convolutional neural networks (CNNs).

2.2.1 Artificial Neural Networks

The artificial neural network is a computational model designed to simulate the neural network of the human brain, which simulates the neural network of human brain in terms of structure, implementation mechanism and function. The basic neuron is known as McCullochPitts (MCP) neuron [15]. MCP takes a set of input $\{x_1, x_2, x_3, ..., x_n\}$ and combines these inputs with a set of weights $\{w_1, w_2, w_3, ..., w_n\}$ by a sum function, then a activation function is introduced to improve the non-linear ability, shown as Fig.2.2. Initially, a step function is used as the activation function, but a continuously derivable function is required by the modern neural networks. From the system point of view, artificial Neural networks are adaptive nonlinear dynamic systems composed of a large number of neurons by extremely rich and perfect connections.

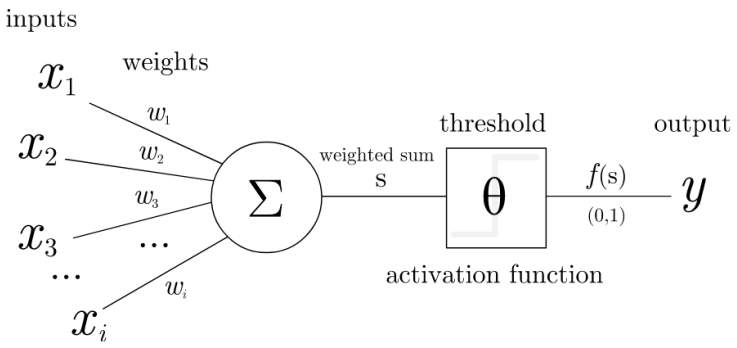
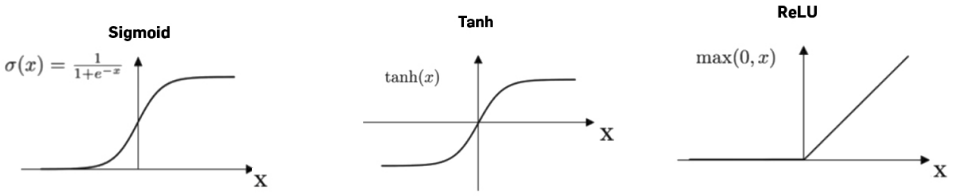


Figure 2.2: The structure of the MCP neuron

Although we can construct an artificial neural network relatively easily, it is not an easy task to train. Early neural network models do not have the ability to learn. Until 1986, Hinton proposed the backpropagation method to solve the learning problem for Multilayer Perceptron (MLP) and firstly used sigmoid function as activation function which shows powerful capacity in machine learning [16].

2.2.1.1 Activation Function

Generally, the computation of the network is a linear operation, and the emergence of the activation function provides the neural network with non-linear capability, which makes the learning capability of the network enhanced.



(a) (b) (c)

Figure 2.3: Activation functions.(a) Sigmoid; (b) Tanh; (c) Relu.

(1) Sigmoid function

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{2.22}$$

The Sigmoid function maps all inputs to the value between 0 and 1. This function is smooth and easy to do derivation. However, it is possible that the gradient will disappear when the number of the network layer is big. Since the output of the Sigmoid function is not zero-mean, the input of the next layer is also the non-zero mean signal. It means that the original distribution of the data will also be changed with the number of network layers increasing.

(2) Tanh function

$$\text{Tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.23}$$

The Tanh function is also called hyperbolic tangent function, which maps the input between -1 and 1, solving the problem that the output of the Sigmoid function is not zero-mean, but the gradient disappearance problem still exists as the layers of the neural network increasing.

(3) Relu function

$$\text{Relu}(x) = max(0, x) \tag{2.24}$$

The ReLU function is also known as a modified linear unit. As a kind of piecewise function, the gradient of the positive part of this function keeps one. It partly solves the problem of gradient disappearance of Sigmoid and Tanh functions, and the computation speed is much faster compared to the previous two non-linear activation functions. So, the Relu function is widely used in the design of modern artificial neural networks.

2.2.1.2 Loss Function

In machine learning, supervised learning keeps the output of a model close to the desired outcome. The loss function, an important component of supervised learning, gives the model an optimizing direction in the form of loss cost. The larger loss, the poorer fit of the model is, while the smaller cost, the better prediction of the model is. There are several common loss functions shown below.

(1) Mean Squared Loss

Mean Squared Loss (MSE) is also called l_2 loss which is commonly used in machine learning, deep learning.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.25}$$

(2) Mean Absolute Error Loss

Mean Absolute Error Loss (MAE) is also known as l_1 loss. The loss cost is linear with the absolute error between the true value and the predicted value. Compared to MSE, its convergence is slower. However, it is more robust to a few samples that deviate from the normal range.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{2.26}$$

(3) Cross Entropy Loss

This loss function is also called the softmax function. It will transfer the output vector as the probability distribution expression. The purpose of this function is to estimate the probability distribution of each sample that belongs to the true label

of each category. The smaller the value of the function, the higher the accuracy of the classification, and vice versa. The cross-entropy loss function is mainly used to solve binary classification problems or multi-classification problems.

$$CE = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{K-1} y_{i,k}\log p_{i,k} \tag{2.27}$$

$p_{i,k}$ represents the predicted probability of sample i belongs to the class k. $y_{i,k}$ is a flag which will only be 0 or 1. For example, if $y_{i,k} = 1$, it means the category of sample i is class k.

MAE and MSE is mainly used in the regression question, but for Cross Entropy Los, it is usually for classification problem.

2.2.1.3 Multilayer Rerceptron and Backpropagation

Multilayer Perceptron (MLP) generally consists of three-layer architecture, including Input Layer, Hidden Layer, and Output Layer. Fig.2.4 shows an example of MLP, and all neurons between layers in the network are connected. Therefore, it is also called a fully connection neural network.

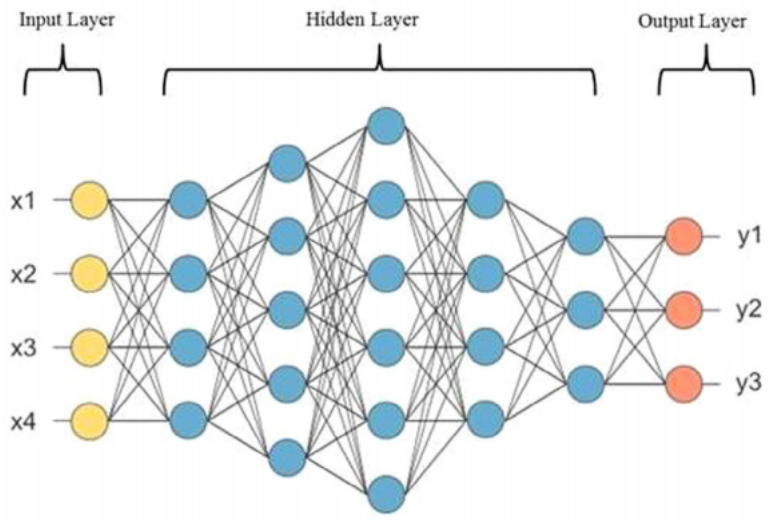


Figure 2.4: Multilayer Perceptron [1]

The goal of the model is to learn the mapping relationship from **x** to **y**, where **x** is the input to the network and **y** is the desired result. This process is called forward propagation. In order to make the predicted results and the target results are the

same, we should use a chain rule to update the weights **w** and bias **b**, which is called backpropagation. The most commonly used algorithm in deep learning training is the gradient descent algorithm, which firstly initializes the parameters of the model, and then descends in the direction of the gradient that minimizes the loss, during which the parameters are continuously updated by Eq 2.28.

$$\mathbf{w}' = \mathbf{w} - \alpha \frac{\partial l}{\partial \mathbf{w}}, \quad \mathbf{b}' = \mathbf{b} - \alpha \frac{\partial l}{\partial \mathbf{b}} \tag{2.28}$$

Where $\mathbf{w}'$ and $\mathbf{b}'$ are the updated parameters, **w** and **b** represent the parameters before updating, α denotes learning rate, $\frac{\partial l}{\partial \mathbf{w}}$ and $\frac{\partial l}{\partial \mathbf{b}}$ mean the direction of the gradient.

As the number of training layers increasing, it may cause the gradient disappearing or the gradient exploding, so it is necessary to design a reasonable network structure and choose a suitable loss function and the activation function to avoid the failure of the training and converging to the optimal solution.

2.2.2 Convolutional Neural Networks

Although the multi-layer perceptron network has shown great power, there are still some disadvantages hard to ignore.

(1) Too many parameters: In a fully connected feed-forward network, every neuron pairs between two adjacent layers have independent connections. Each connection corresponds to a weight parameter. As the number of hidden layer neurons growing, the size of the parameters increases dramatically. This leads to a very low training efficiency for the whole neural network, and it is easy to cause overfit problems.

(2) Local invariant characteristics: images of the real world have local invariant characteristics. The operations such as scale, translation and rotation do not affect their semantic information. But, the fully connected feed-forward network is difficult to extract these local invariant features, and data enhancement is generally needed to improve the performance.

In 1962, Hubei et al. conducted a study of cat visual cortex cells, in which

they showed objects of various shapes and brightness in front of the cat's eyes and changed the angle and position of the objects, then observed the visual cortex cells [17]. The results of these experiments revealed an "orientation-selective cell". People gradually began to understand how the visual nervous system works, from primitive signals to low-level abstraction, then to high-level abstraction. The concept of the perceptual field was first introduced. Photoreceptors in the retina are excited by stimulation and send neural impulse signals to the visual cortex. However, not all neurons in the visual cortex receive these signals. The receptive field of a neuron is a specific area of the retina where only stimulation in this area can activate the neuron.

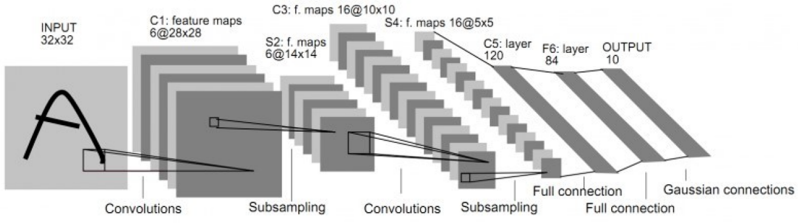


Figure 2.5: Architecture of LeNet-5, a convolutional neural network [2].

Then, this concept was adapted to the machine learning area, like Lenet-5 which was developed by Lecuu. By this method, the high-level features, like lines, edges and brightness, can be recognised in the higher layer neurons. Besides, it also makes networks insensitive whit translation, scaling and rotation. And because it shares the same weights in the same convolutional layer, the parameters are greatly reduced.

These excellent characters make it widely used in deep learning, especially computer vision. The current convolutional neural network is generally a feed-forward architecture cross-stacked by convolutional, pooling and fully connected layers. Next, we will make some introduction for some basic blocks that will be used in our network.

2.2.2.1 Convolutional Layer

In a convolutional neural network, the convolutional layer is the most basic operation that is used to extract features of the input image. And the output obtained after the convolution operation is the feature map. One convolutional layer is composed of multiple convolutional kernels, which are also called filters. Actually, the convolu-

tional operation is a kind of weighted summation operation of the pixels for every image patch. The size of the convolution kernel is also known as the perceptual field which has been introduced above.

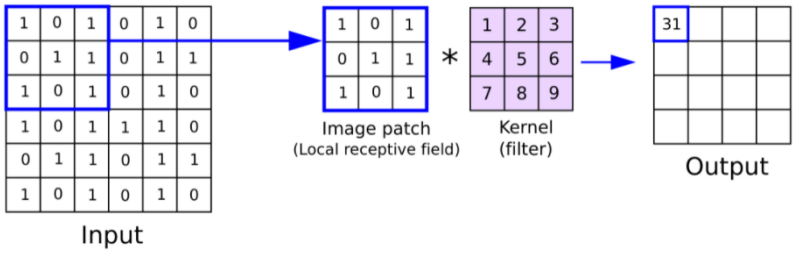


Figure 2.6: Convolution operation.

An example is shown in Fig.2.6. There is a 6×6 input matrix and a 3×3 convolution kernel used to do the convolution operation. It begins from the top-left of the input, and the figures in the image patch are multiplied with the numbers in the corresponding positions of the convolution kernel and then sum up to complete a convolution. After, traverse the image patch of the input to get the final output.

2.2.2.2 Pooling Layer

After the convolution operation, the feature map is usually big, thus a pooling layer is introduced to decrease the number of dimensions of the feature map. It can also help to keep to maintain feature invariance, like translation, rotation and scale. The pooling operation is a little similar to the scaling of an image. For example, people can still recognize it as a dog by doubly scaling down a dog picture. It means that the most important features of the dog have remained in the image, and people can still identify what it is. Apart from this, it can reduce the number of parameters and simplify the calculation, while keeping the main features to prevent overfitting and improve the model generalization ability.

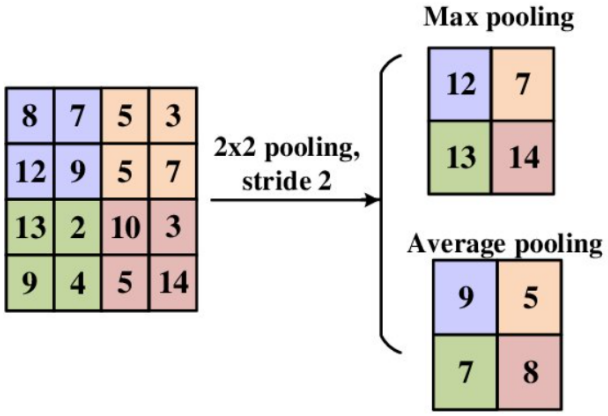


Figure 2.7: Max pooling and average pooling.

There are two main pooling methods including max and average pooling. Fig.2.7 shows us an example to do pooling operation for a 4×4 input and 2×2 kernel.

2.2.2.3 Fully Connected Layer

The convolutional layer and pooling layer are mainly applied to extract the feature map, and the fully connected layer is usually used to convert the feature map to a feature vector. Unlike the convolutional layer with local connections, within the fully connected layer, all neurons are connected with all of its adjacent upper and lower layers. Actually, the fully connected layer can also be regarded as a kind of multi-layer perceptron architecture. This makes it more efficient for processing nonlinear features, but accordingly, the number of parameters is also increasing.

2.2.2.4 Dilated Convolution

In CNN, the use of standard convolution has some problems in some cases. For example, in terms of the semantic segmentation task, the downsampling operation is often applied to reduce the network parameters and expand the network perceptual field. However, downsampling would lead to image resolution degradation and cause the loss of high-level features. For dealing with these questions, a novel convolution, dilated convolution is proposed [18]. It has some benefits below.

(1) Dilated convolution can expand the perceptual field of the network model at an exponential rate without increasing the network parameters and model complexity.

(2) The expanded convolution can better deal with feature mappings, and make

full use of the multi-scale contextual information.

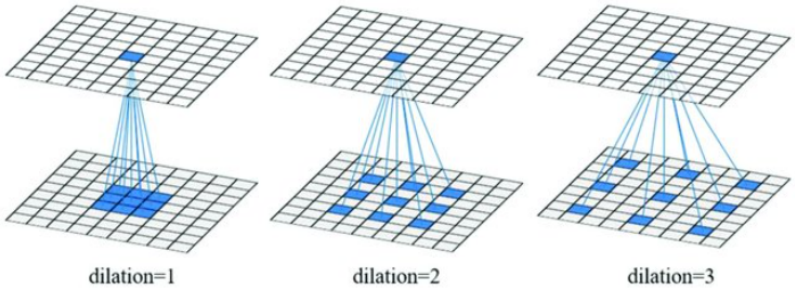


Figure 2.8: Dilated convolution.

Dilated convolution is easily obtained by just adding voids between the convolution kernels of a standard convolution. And there is an additional parameter for the dilated convolution, dilation rate, and the bigger it is, the bigger size perceptual field has. Just like Fig.2.8 showing, the number of parameters of the kernel will not change, however, the perceptual region augments.

2.2.2.5 Residual Model

It is generally believed that a trained deep neural network is capable of abstracting features layer by layer, and finally using simple classifiers or other learners to complete the final task. That is why deep learning is also called feature learning. With intuition, the natural idea is to train a very deep feed-forward neural network to accomplish the task. Intuitively, the deeper neural network's power is augmented by the nonlinear activation functions, which is more like to have an optimal solution. However, after deepening the network to a certain level in practice, it is not possible to continue adding layers. Because at this time, as the network continues augmenting, the results obtained will not only be optimized, but also may even become worse. One possible reason is that the deeper networks lead to overfitting phenomena. Beyond, neural networks are also more prone to cause some problems such as gradient disappearance and network degradation.

To be able to solve the above-mentioned problems researchers proposed the residual network (ResNet) [3]. The main idea of ResNet is to let redundant network layers complete constant mapping, i.e., without iteratively updating the network

weights, ResNet can also make the network deepen to a higher level. And it is possible to obtain more information, learn richer features while avoiding the gradient disappearance.

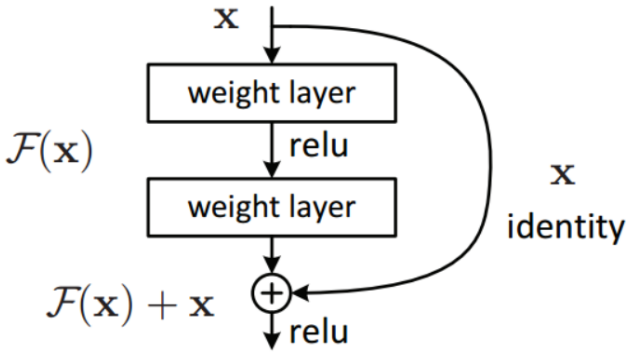


Figure 2.9: Single residual block [3]

The residual network adds a branch to the main forward path in the basic block, as Fig.2.9. This process can be expressed as a mathematical function: $\mathcal{H}(x) = \mathcal{F}(x) + x$. When $\mathcal{F}(x) = 0$, it means $\mathcal{H}(x) = x$. In other words, the output is the same as input in this calculation, and we say that it is a constant mapping. ResNet learns the residual loss between $\mathcal{H}(x)$ and x to update the parameters of $\mathcal{F}$. Besides, the cross-layer connection of the residual block promises the accuracy of the network, and does not decrease as the layer's number of the network increasing and remains the network at an optimal state, avoiding problems such as gradient vanishing.

2.2.3 Summary of deep learning

Due to the non-linearity and complexity of neural networks, it has a stronger expressive ability, i.e., it is possible to find the neural network that fits better to the specific data set and find the complex features which are hard for a person to notice. It is believed that this is an important reason why deep learning methods can yield a good set of results. We want to use this technology to deal with the beam propagation problem.

In theory, a MLP network can fit any function. However, it might meet some problems like vanishing gradient and too many parameters. And considering CNN architecture shows great performance in handling computational imaging (CI) prob-

lems. Here, we will try to use the CNN method to simulate a very complicate physical case, the beam propagation. We wish it can accelerate the calculation compared to traditional mathematical methods, and in this way, we can also avoid the mismatch problem in CGH problems.

Chapter 3

Methodology

Deep learning has been widely used in various aspects of human life, like face recognition, medical image analysis, image restoration and so on. Deep learning can deal well with any regression and classification problem. So, it encourages us to think about whether we can use this technology to simulate a very complicated law of physics, such as beam propagation. In this way, We can better allocate and utilize computing resources of computers, especially the usage of GPU. In this chapter, we will introduce how to generate the datasets and propose a basic architecture to do this job.

Worth to mention, there are many words to express beam propagation, like the wave or the optical propagation and the beam diffraction. Also, because that light has the property of wave-particle duality, beam can be also called wave, disturbance in many other papers. All in all, just need to know that they are referring to the same thing in this paper.

3.1 Datasets

Although, there are mathematical functions to express the beam propagation process, like angular spectrum function [11]. However, there are too many differences between theory and practice. Even trying to accurately set various parameters, there are still many factors we can not control in a real optical system. For example, the non-linear relationship between the voltage and phase delaying of the SLM device in a CGH system and the tilted angle between the hologram plane and source plane, all

of these unpredictable factors and mismatch may cause problems. Some researches [12] in CGH and lensless imaging area has shown this. These factors are even more complicated and hard added to the beam diffraction calculation. Actually, these parameters can be considered directly by a network, by just training it in the dataset generated by the certain system.

In this project, we just do a preliminary exploration for simulating the beam propagation by the deep learning method. Therefore, there is no need to make a real optical system to collect the dataset. And we can just use the fundamental mathematic functions to build the dataset, and verifying the powerful ability of our model to regress a complicated function even like these optical formulas.

In the CGH and lensless imaging area, computing the hologram of a two-dimensional image is highly demanded, especially in the phase retrieval questions, and it is hard to keep a strict parallel between the source plane and observation plane. Even for the calculation of a 3D object's hologram, the most useful method is the polygon-based algorithm, which makes the surface of the object as many polygonal faces and then calculates the diffraction pattern of these faces in the hologram plane [14]. And these faces are always not parallel with the hologram plane. So, dealing with the beam propagation problem between the tilted planes will be more useful and general. Above all, we pay more attention to the optical diffraction calculation in the tilted planes.

3.1.1 Physical Model of the beam Propagation

The geometric expression of the beam propagation in the tilted planes can be indicated by Fig.3.1. And there are several coordinate systems that need to be introduced firstly. We assume that the beam is defined in the $(x, y, 0)$ source coordinate system. And observe it diffraction pattern (or we say hologram) in the $(\hat{x}, \hat{y}, \hat{z})$ plane at a distance d between the source plane, we say this plane is the reference plane, while this plane is also be called observation plane or the hologram plane in some articles.

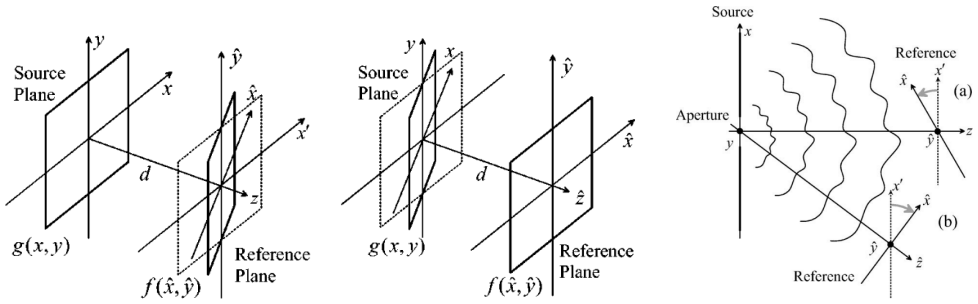


Figure 3.1: Scheme of the beam propagation in the tilted planes [4].

Fig.3.1 show two basic cases. One is the reference plane is tilted with an angle which is shown in Fig.3.1(a), and its inverse situation is shown in Fig.3.1(b). Actually, these two cases are the same and could be expressed as the same mathematical process which we will discuss in the following text. And in Fig.3.1(c), it shows these two situations from the top view in one image, and it helps simplify the wave propagation by placing an aperture in the source plane and assumes the beam exits from this aperture.

In both cases, the calculation of beam propagation is made up of two steps. The first step, compute the diffraction pattern in the intermediate plane $(x', \hat{y}, \hat{z})$ which is parallel with the source plane $(x, y, 0)$ by angular spectrum function which has been discussed in the introduction chapter. And then deduct the hologram in the reference plane $(\hat{x}, \hat{y}, \hat{z})$. In both scenes, the most essential steps are the same. So, we say there is no difference between these two cases in substance. Thus, we will focus on the case of Fig.3.1(a). And all of the following discussion is related to this scene.

3.1.2 Calculation of the beam propagation

For the calculation of optical diffraction in the tilted plane, there are several ways to do it. The traditional method is cutting the tilted reference plane into many parallel planes and convert this question into calculating the wave diffraction in the parallel planes. For speeding up this process, a fast calculation method is proposed which does a coordinate rotation in the Fourier domain [4, 19]. In this method, we need to do an interpolation operation for doing the rotation in the Fourier domain, and

Chenliang et al. substitutes the nonuniform fast Fourier transformation (NUFFT) for this interpolation operation and the results show well [20]. Generally, a dataset often includes tens of thousands of images, and considering the runtime, we use the fast calculation method by paper [4] to generate our dataset here.

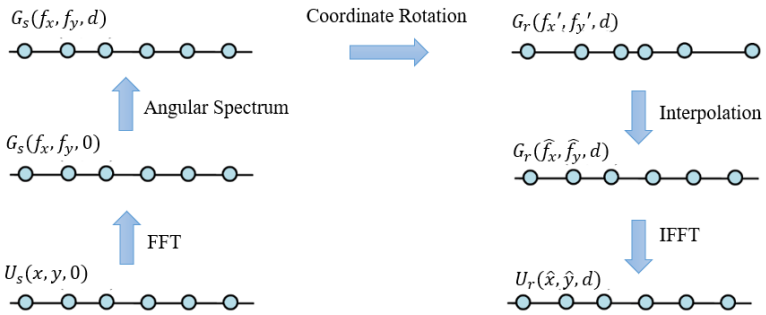


Figure 3.2: Schematic diagram of diffraction calculation of tilted plane.

Fig.3.2 shows us the calculation process of the fast rotate angular spectrum (RAS) algorithm. $U_s(x,y,0)$ and $U_r(\hat{x},\hat{y},d)$ mean the signals in the time domain of the source and reference planes respectively, and $G_s(f_x,f_y,0)$ and $G_r(\hat{f}_x,\hat{f}_y,d)$ are their the spectrum expressions. In this picture, the left shows the normal angular spectrum calculation, and the right indicates the rest operations that we should do to execute the rotate angular spectrum. Thus, the main steps we should notice is the following: (1) angular spectrum algorithm; (2) coordinate rotation; (3) interpolation; (4) inverse Fourier transformation. As for the angular spectrum algorithm, it has been introduced in detail in Section 2.1.4. The rest operations can be seen following.

3.1.2.1 Coordinate Rotation

For a complex plane wave $U(x,y,z=0;\mathbf{k})$, it can be expressed by the amplitude A and a wave vector $\mathbf{k}$. And its form can be defined in general as:

$$u(x,y,z=0;\mathbf{k}) = A\exp\left[i\frac{2\pi}{\lambda}(k_x x + k_y y)\right] \tag{3.1}$$

with

$$\mathbf{k} = \frac{2\pi}{\lambda}\begin{bmatrix} k_x & k_y & k_z \end{bmatrix} \tag{3.2}$$

And $\mathbf{k}$ should satisfy $|\mathbf{k}| = \frac{2\pi}{\lambda}$ and $k_x^2 + k_v^2 + k_z^2 = 1$. Now, we return back to the inverse operation of the AS algorithm (Eq.2.12), which transfers the optical wave's spectrum to the field of time domain. From a different perspective, this function can be interpreted as a superimposition of plane waves:

$$u(x,y,z;f_x,f_y) = G(f_x,f_v,z)\exp[i2\pi(f_x x + f_y y)] \tag{3.3}$$

Comparing with Eq.3.1, we can associate the wave vector $\mathbf{k}$ with Fourier frequencies by:

$$\mathbf{k} = 2\pi \begin{bmatrix} f_x & f_y & f_z \end{bmatrix} \tag{3.4}$$

Where

$$f_z = \sqrt{\lambda^{-2} - f_x^2 - f_y^2} \tag{3.5}$$

Notice that the definition of the Fourier frequencies is determined by the coordinate system used. This makes it possible use a simple transformation matrix $\mathbf{T}$ to transfer from one coordinate system to another. Here, we define $\hat{\mathbf{k}}$ is the wave vector in the reference plane and $\mathbf{k}$ is in the source plane.

$$\hat{\mathbf{k}} = \mathbf{T}\mathbf{k}, \quad \mathbf{k} = \mathbf{T}^{-1}\hat{\mathbf{k}} \tag{3.6}$$

Assuming the inverse transformation matrix is:

$$\mathbf{T}^{-1} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \tag{3.7}$$

We can obtain the spectrum field in the reference plane by that in the source plane.

$$\begin{aligned} f_x &= \alpha(\hat{f}_x,\hat{f}_y) = a_1\hat{f}_x + a_2\hat{f}_y + a_3\hat{f}_z(\hat{f}_x,\hat{f}_y) \\ f_y &= \beta(\hat{f}_x,\hat{f}_y) = a_4\hat{f}_x + a_5\hat{f}_y + a_6\hat{f}_z(\hat{f}_x,\hat{f}_y) \end{aligned} \tag{3.8}$$

Then the spectrum in the reference will be gained easily by that in the interme-

diate plane, which could be directly gotten by AS algorithm:

$$G_r(\hat{f}_x, \hat{f}_y, d) = G_s(\alpha(\hat{f}_x, \hat{f}_y), \beta(\hat{f}_x, \hat{f}_y), d) \tag{3.9}$$

3.1.2.2 Interpolation

The Fourier transformation is widely demanded in the fast RAS algorithm that we mention here. Actually, in practice, the FFT algorithm is commonly used for accelerating the calculation. However, there are several essential conditions for using FFT and one is an equidistance sampling grid. Obviously, it is also necessary for IFFT and non-uniform sampling will cause problems. As mentioned in Section 2.1.5, due to the sampling characteristics of CCDs, the optical field is represented by the image's pixels which is equal-sampled. Therefore, we can directly get its spectrum $G_s(f_x, f_y, 0)$ from $U_s(x, y, 0)$ of the source plane, and the spectrum field $G_s(f_x, f_y, d)$ of the intermediate plane will keep as a uniform sampling grid form. However, because of the tilted angle between the intermediate plane and the reference plane, we have to multiply with a transformation matrix $\mathbf{T}$ to get $G_r(f_x, f_y, d)$. And this operation will cause the non-uniform sampling grid.

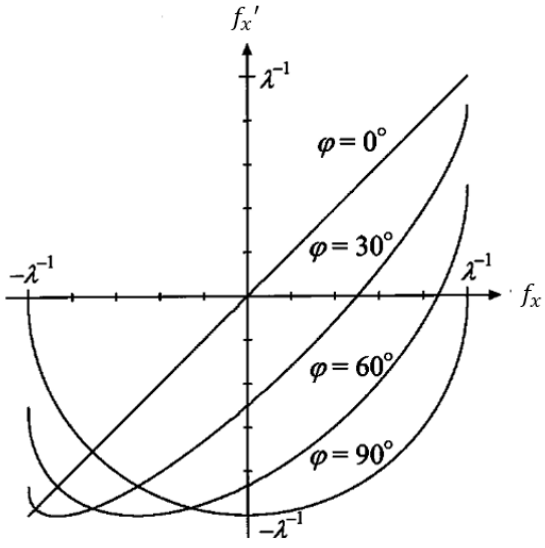


Figure 3.3: Curves for transforming Fourier frequency f_x in the intermediate plane into $\hat{f}_x$ in the reference plane; $f_y = 0$ [4].

Fig.3.3 shows the non-linear relationship between $\hat{f}_x$ and f_x when $f_y = 0$ in the

situation where we only rotate the y-axis of the intermediate plane with ϕ angle to get the reference plane. Because f_x is uniformly distributed, this non-linear relationship causes $\hat{f}_x$ non-uniform. Therefore, we will get a spectrum field on a non-uniform sample grid after coordination rotation. And our aim is to deduct a uniform spectrum field in the tilted plane by the interpolation method. So the question converts into two steps. Firstly, what this uniform grid looks like? Secondly, which interpolation should we use?

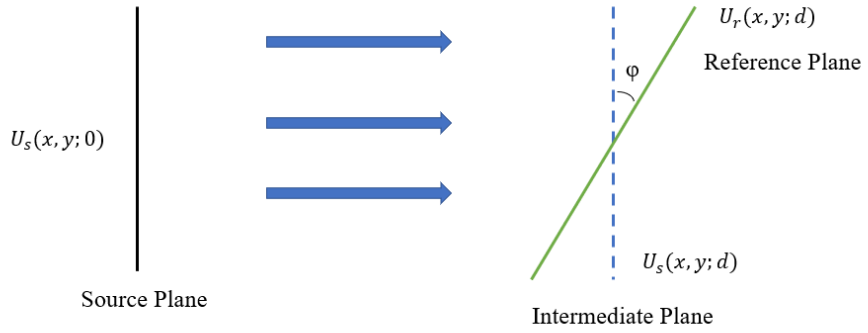


Figure 3.4: Schematic diagram of beam diffraction in the tilted plane.

Consider the situation as Fig.3.4, where the reference plane is inclined ϕ angle along the y-axis (for easily analysis, we just take into consideration that there is only one axis rotated). It is easily to get the image shape of the final result. We suppose that the size of source optical field is $L_x \times L_y$ with the resolution $M \times N$. It is obvious that the diffraction pattern size in the tilted plane will be $L'_x \times L_y$, where $L'_x = \frac{L_x}{\cos\phi}$. And the resolution will keep the same. Thus for the Fourier frequency sequence (f_x, f_y) , it will be:

$$\begin{aligned} f_x &= \left\{ \frac{1}{L_x}[-\frac{\lfloor N \rfloor}{2} + s(\frac{\lceil N \rceil}{2} - 1)] : 0 \le s \le N-1 \right\} \\ f_y &= \left\{ \frac{1}{L_y}[-\frac{\lfloor M \rfloor}{2} + s(\frac{\lceil M \rceil}{2} - 1)] : 0 \le s \le M-1 \right\} \end{aligned} \tag{3.10}$$

Where function $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the upward and downward rounding functions respectively, for dealing with the different situations that the resolution of image is

even or odd. Similarly, the Fourier frequency sequence in the inclined plane is:

$$\begin{aligned}\hat{f}_x &= \left\{ \frac{1}{L'_x}[-\frac{\lfloor N \rfloor}{2} + s(\frac{\lceil N \rceil}{2} - 1)] : 0 \le s \le N-1 \right\} \\ \hat{f}_y &= \left\{ \frac{1}{L_y}[-\frac{\lfloor M \rfloor}{2} + s(\frac{\lceil M \rceil}{2} - 1)] : 0 \le s \le M-1 \right\}\end{aligned} \tag{3.11}$$

And as for the Fourier frequencies in the intermediate plane, they will be:

$$\begin{bmatrix} f'_x \\ f'_y \\ f'_z\left(f'_x, f'_y\right) \end{bmatrix} = T \cdot \begin{bmatrix} f_x \\ f_y \\ f_z(f_x, f_y) \end{bmatrix} \tag{3.12}$$

Notice that (f'_x, f'_y) will compose a non-uniform grid. Apart from this, the inclined angle will also cause a shift in the center frequency in the spectrum $Gs(f_x, f_y, d)$. The corresponding Fourier $f'_x \neq 0$, when $f_x = 0$ and $f_y = 0$. We also call this shift as carrier frequency. Due to that FFT and IFFT can not work for the spectrum far from zero frequency, we should eliminate this carrier frequency firstly and compensate for this shift in the end.

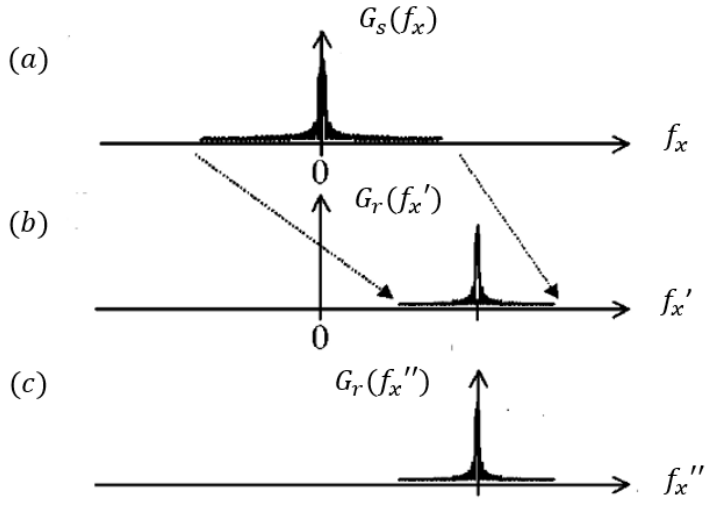


Figure 3.5: Schematic diagram of spectrum shift. (a) spectrum on in the intermediate plane, (b) reference plane, and (c) shift of spectral origin. [4].

From Fig.3.5, you can see that the transformation matrix will cause a shift,

which means that the centroid of f'_x is not zero (shown in (b)). So, we try to eliminate this carrier frequency and shift it back. And we compensate this shift in the next part. So we get a new non-uniform (f''_x, f''_y) grid:

$$\begin{bmatrix} f''_x \\ f''_y \\ f''_z\left(f''_x, f''_y\right) \end{bmatrix} = \begin{bmatrix} f'_x \\ f'_y \\ f'_z\left(f'_x, f'_y\right) \end{bmatrix} - \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.13}$$

Where the carrier frequency $(u, v, w)^T$ is

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = T \cdot \begin{bmatrix} f_x = 0 \\ f_y = 0 \\ f_z(0,0) \end{bmatrix} \tag{3.14}$$

Remember, we just shift the frequencies and do not change the spectrum value. Now, we have gotten a uniform gird $(\hat{f}_x, \hat{f}_y)$ and a non-uniform spectrum $G_r(f''_x, f''_y, d)$. Next, we do interpolation to obtain the uniform spectrum $G_r(\hat{f}_x, \hat{f}_y)$.

There are several interpolation algorithms we can use, like sinc interpolation, cubic interpolation and nth-order iterated linear interpolation [21]. According to the sampling theorem, the sinc interpolation could reach the best accuracy but waste calculation time. Thus, the author of paper [4] propose the ”cubic8” interpolation method and it can also reach a sufficient accuracy. However, we found that the cubic spline interpolation also perform great. And because it is well encapsulated by some libraries like SciPy [22], this interpolation method will run faster compared to the cubic8 method in practice.

3.1.2.3 Inverse Fourier Transformation

A simple inverse Fourier transformation could not reach our expected result. Because the total energy of the optical field will always decrease after the coordinate rotation operation. Therefore, it is necessary to add an additional item to fix the final result. Then, a Jacobian item $J(\hat{f}_x, \hat{f}_y)$ is introduced and convert the inverse Fourier

transformation of the angular spectrum function to:

$$U_r(\hat{x},\hat{y},z) = \iint_{-\infty}^{\infty} G_r\left(\hat{f}_x,\hat{f}_y;z\right)\exp\left[j2\pi\left(\hat{f}_x\hat{x}+\hat{f}_y\hat{y}\right)\right]\times|J(\hat{f}_x,\hat{f}_y)|df_xdf_y \quad (3.15)$$

with,

$$\begin{aligned} J(\hat{f}_x,\hat{f}_y) =& \frac{\partial\alpha}{\partial\hat{f}_x}\frac{\partial\beta}{\partial\hat{f}_y} - \frac{\sigma\alpha}{\partial\hat{f}_y}\frac{\partial\beta}{\partial\hat{f}_x} \\ =& (a_2a_6 - a_3a_5)\frac{\hat{f}_x}{\hat{f}_z(\hat{f}_x,\hat{f}_y)} \\ &+ (a_3a_4 - a_1a_6)\frac{\hat{f}_y}{\hat{f}_z(\hat{f}_x,\hat{f}_y)} + (a_1a_5 - a_2a_4) \end{aligned} \quad (3.16)$$

And for easily, we can rewrite as

$$U_r(\hat{x},\hat{y}) = \mathcal{F}^{-1}\{G_r(\hat{f}_x,\hat{f}_y)|J(\hat{f}_x,\hat{f}_y)|\} \quad (3.17)$$

In the prior step, we eliminate the carrier frequency for execute FFT. Here, we will consider this factor. And finally, the inverse Fourier transformation becomes

$$U_r(\hat{x},\hat{y}) = \mathcal{F}^{-1}\{G_r(\hat{f}_x,\hat{f}_y)|J(\hat{f}_x,\hat{f}_y)|\}\exp\left[j2\pi(u\hat{x}+v\hat{y})\right] \quad (3.18)$$

Obviously, the carrier frequency only exists in the phase factor. Even without this item, the intensity result $|U_r(\hat{x},\hat{y})|$ is affected. Moreover, we can only think about this compensatory item when it is needed by multiplying it after IFFT operation.

3.1.2.4 Validation

In the previous part, we have made a detailed introduction for calculation process of the RAS algorithm. In this part, we will do validation to verify the correction of our implementation.

There is a numerical simulation method to calculate the diffraction pattern in the tilted reference plane, shown in Fig.3.6. In this method, we treat the tilted plane as a series parallel planes. Then we compute the optical field in these planes, and finally combine all the points which are the intersection between the parallel plane

and the reference plane. Thus, we can easily obtain the optical field in these slides by directly using AS algorithm.

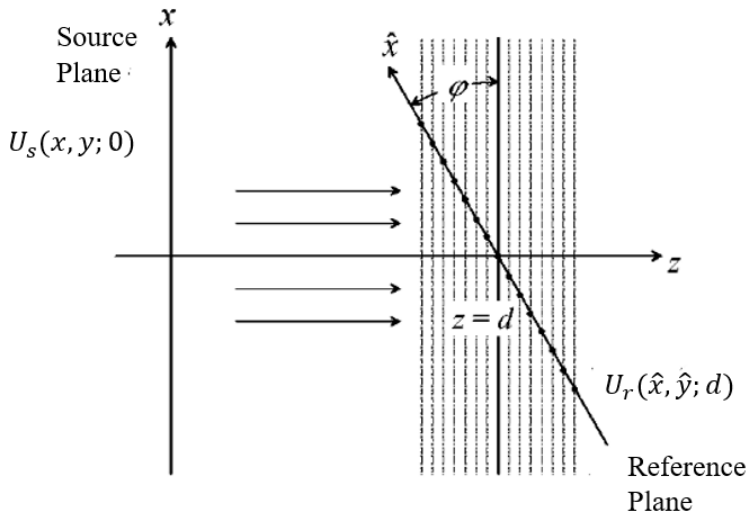


Figure 3.6: Schematic diagram for numerical simulation method of single-axis rotation [4]

For the validation, we assume there is only y-axis rotation and the tilted angle is $\phi = 30°$. The amplitude field of $U_s(x,y;0)$ is a randomly selected human face picture, but do normalize to transfer the range $[0,255]$ to range $[0,1]$. Similarly, we also randomly select a human face image as the phase field of $U_s(x,y;0)$ and limited values in range $[-\pi,\pi]$. As for other parameters, we set diffraction distance $d = 10mm$, beam wavelength as $632.8nm$, the pixel size of image as $5\mu m \times 5\mu m$ and the shape of image as 256×256.

Fig.3.7 and 3.8 show our experiments. We noticed that the size of optical field in the tilted plane has changed which satisfied our common sense. And there is almost no difference between the fast RAS method and the numerical simulation method. All of the results show the correction of our implementation.

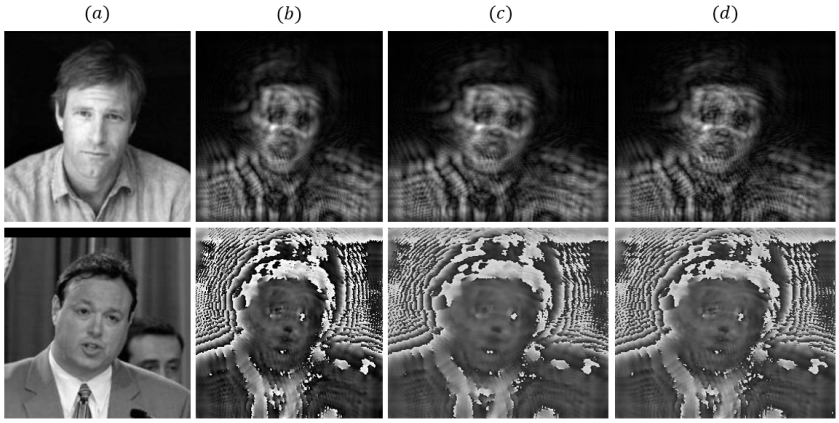


Figure 3.7: Validation of the beam propagation on tilted planes. The first row indicates the amplitude field, and the second express the phase field. (a) input images; (b) diffraction on the intermediate plane; (c) diffraction on the reference plane by the fast RAS method; (d) diffraction on the reference plane by the numerical simulation. Besides, in the phase image of (c) and (d), the phase factor caused by the carrier frequency has been eliminated.

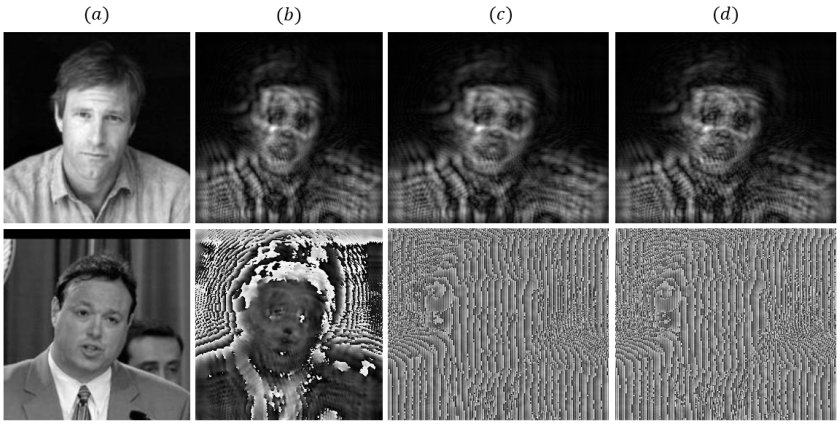


Figure 3.8: Validation of the beam propagation on tilted planes. The first row indicates the amplitude field, and the second express the phase field. (a) input images; (b) diffraction on the intermediate plane; (c) diffraction on the reference plane by the fast RAS method; (d) diffraction on the reference plane by the numerical simulation. Besides, in the phase image of (c) and (d), the carrier frequency has been compensated.

3.1.3 Generate Dataset

The ideal training dataset for dealing with our specific problem should contain all of classes of optical images and it is better to collect data on a certain professional optical system which is pre-defined by our problem. However, it is unrealistic and would waste too much time and money. So we propose to use synthesized data as

train set. Basing on the implementation of the fast RAS algorithm, we can generate datasets suitably. For verifying the great power of deep learning to simulate the optical diffraction on the tilted plane. We fix parameters of the beam propagation (wavelength, pixel size, tiled angle and propagation distance) to generate one dataset by the fast RAS algorithm. In this dataset, we use amplitude-phase image pair as input, and corresponding amplitude-phase pair as the target.

3.1.3.1 Basic Database

There are many datasets we can use, like ImageNet [23] dataset, Faces-LFW [24] dataset and Mnist [25] dataset. ImageNet is one of the most common databases, and it has been widely used in various machine learning problems especially in machine vision. However, it is too big which contains over fourteen thousand images. Considering our experiment condition, it is hard for training. In terms of Mnist database, it is too mall for this question. So, we select Faces-LFW here. Although, the data is limited in a small narrow class range (human face), it is still enough to demonstrate the ability of neural networks for simulating beam propagation.

Faces-LFW: This dataset is compiled by the Computer Vision Laboratory at the University of Massachusetts. It is mainly used to study the face recognition problem in unrestricted situations and mainly collected from the internet. There are 13233 images which belong to 5749 people, and each image's size is 250×250. Here, these images will be used to generate our dataset.

3.1.3.2 Synthesized Dataset

Essentially, our aim is to test the ability of neural networks to fit so complex functions like the beam propagation on the tilted plane. So for dealing with this problem, the dataset generated should contain amplitude-phase input pair and corresponding amplitude-phase target pair. And there are several parameters that should be pre-defined. In our experiments, the beam wavelength is $632.8nm$, the number of pixels is 256×256 and its size is $5\mu m \times 5\mu m$. Apart from this, we set the beam propagation distance as $10mm$ and the tilted angle between the reference plane and the source plane is 10° and rotate along the y-axis. And for the input, we randomly selected a human face image as amplitude input and do the same to get phase input, and we

resize the image size from 250×250 to 256×256. Normalization is necessarily needed before executing the fast RAS function. And the value range of amplitude input is limited in range $[0,1]$. As for the phase input, it is range $[-\pi,\pi]$.

Besides, for improving the calculation accuracy, it is better to add zero-padding for the input data before calculation. As shown in Fig.2.1, the computation window is twice the size of observation window because of the Nyquist sampling theorem.

Basing on the previous discussion, the image resolution will keep the same but the pixel size would change compared with the input data in the source plane. And the final complex image size is easily gotten by the geometry. Thus, we can keep the input and output image the same shape and resize the final result in the end. Besides, we can observe that non carrier frequency phase factor looks more meaningful for the phase output. And also, this factor could be easily added in the end by Eq.3.18. Therefore, for easily training, we use the non phase factor version as our database. People can add this item in the end. And in practice, we found that the network performs better if we map the phase input and output from range $[-\pi,\pi]$ to range $[0,1]$. Also, there is no need to worry about this change, because we can map it back in the end. Finally, Fig.3.9 shows one example of the synthesized dataset.

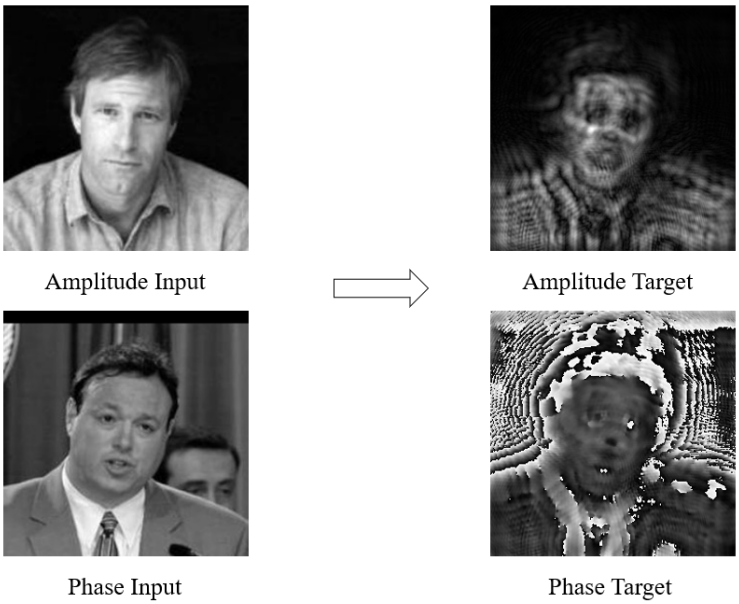


Figure 3.9: Synthesized training set. Left: amplitude-phase paire input; Right: amplitude-phase pair target.

3.1.4 Our Model

This section outlines the model we build for this specific problem and do some analysis. Firstly, we do some introduction for some related works. Then we propose a model pipeline for handling this problem, and more details are also discussed following, including training detail. And all of this model is implemented by Pytorch [26].

3.1.4.1 Related Works

Actually, there is not so much similar work. As we know, we are the first who try to simulate the beam propagation by the deep learning method. But, researchers have done many works related to phase retrieval by using the deep learning method [9, 10], which actually is the inverse problem of the beam propagation. In phase retrieval problems, there are diffraction intensity images, and we need to deduct the pattern of original phase objects. In fact, our ideas are closely related to these studies, and we try to do the forward process of the beam propagation and they do the inverse.

We assume the beam propagation function is $H(\cdot)$. Then for the phase retrieval problems, the question is to find a suitable expression for $H^{-1}(\cdot)$, so that they could find the phase object $\phi(x,y;0)$ from the intensity diffraction image $I(x,y;d)$, where d is the propagation distance.

$$\phi(x,y;0) = H^{-1}(I(x,y;d)) \tag{3.19}$$

There are several ways to solve this question, the classical method is to solve question

$$\tilde{\phi}(x,y) = \arg\min_{\phi} \|H(\phi) - I\|^2 + \rho(\phi) \tag{3.20}$$

Where $\rho(\phi)$ is the regularization item. As for the deep learning method, it attempts to learn a mapping function R which could get the predicted phase image from the input. But it is necessary to build a dataset $S_T = \{(\phi_k, I_k); k = 1,2,\ldots,K\}$

firstly. The mapping function's parameter θ could be obtained by solving:

$$R_{\theta^*} = \arg\min_{\theta\in\Theta} \|R_\theta(I_k) - \phi_k\|^2, \quad \forall(\phi_k, I_k) \in S_T \tag{3.21}$$

In terms of our problem, we directly learn the beam propagation function H by DL method from a labelled dataset $S = \{(\phi_s, A_s)_k, (\phi_o, A_o)_k; k = 1, 2, \ldots, K\}$, where A means amplitude image, foot marker s indicates the source and o expresses the observed images.

Thus, the solutions for the phase retrieval problem still have implications for our research. Sinha et al. [27] and Horisaki et al. [28] propose use U-Net [29] architecture neural network to solve end-to-end inverse beam propagation problems, and they have tested their networks on certain lensless image systems. Wang et al. [30] designs a network and combine it with the physical optical function, so that there is no need to pre-train the net and reduce the high demand for the dataset. Wu et al. [31] combines the DL method with the hardware device and make it possible that using a lensless camera to directly capture a real world photo. And he also proposes using DBPN [32] net to refine the image. Peng et al. [12] developed an iteration method to refine parameters of the model.

3.1.4.2 The Pipeline of Our Model

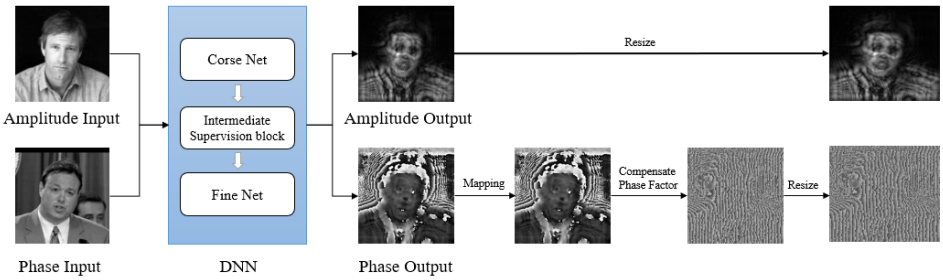


Figure 3.10: Schematic illustration of the pipeline of our model.

As shown in Fig.3.10. The deep neural network takes amplitude and phase images as input data, and will also predict the amplitude and phase field in the reference plane. However, the direct predicted results are not the final output. For easily training, we

map the value of phase field from $[-\pi, \pi]$ to range $[0, 1]$. Thus for getting the correct result, we have to map back for the phase image and limit the range from $= \pi$ to π. Besides, as discussion in Section.3.1.3.2, we ignore the carrier frequency here. So, we have to compensate for this phase factor. This step is easy, and we can directly time a phase factor to do this as Eq.3.18. Finally, we should notice that the pixel size in the inclined plane has changed compared to that in the source plane. Therefore, a resizing operation is essential for dealing with this question. From the geometric relationship, the size of the optical field in the reference plane changes as $\frac{L_x}{\cos\phi} \times L_y$.

The processing for the DNN output has been clearly stated in the previous chapter. Thus, we would not waste time to declare these processes and only focus on the DNN. And here, we suppose a coarse-to-fine architecture. There is no need to keep the same with us for each part of DNN, and we just propose an example next. More details will state in following sections.

3.1.4.3 The Coarse Net

Recently, U-Net [29] architecture is widely used in many fields of computer vision, like image segmentation and medical image processing. Essentially, U-Net is an encoder-decoder network with skip connection architecture. Downsample can increase the robustness to some small perturbations of the input image, such as image translation, rotation, etc.. Besides, it can also reduce the risk of overfitting and the number of operations, and increase the size of the perceptual field. The most important function of upsampling is to revert the abstract features to the original image size. Skip connection ensure the low-level features could also be used and provide more information for the upsample, so that increasing the learning accuracy.

Because of the huge advantages of U-Net, we use its architecture as our coarse net. In our net, we do four times downsampling and four times upsampling operations. Also, we do some small changes for this net. Consider that the cross-layer connection of the residual block could promise the accuracy of the network to remain the network at an optimal state, avoiding problems such as gradient vanishing. We use residual downsample and residual upsample block for doing encoder and decoder processes. Apart from this, the convolution layer also is replaced by the residual convolution

block. Moreover, we know that the optical field of one point on the reference plane is associated with all of the range of the spectrum. Thus, an additional dilated convolution block is used for increasing the receptive field. More details about these basic blocks could be found in Appendix Fig.A.1. And Fig.3.11 shows the architecture of our coarse net.

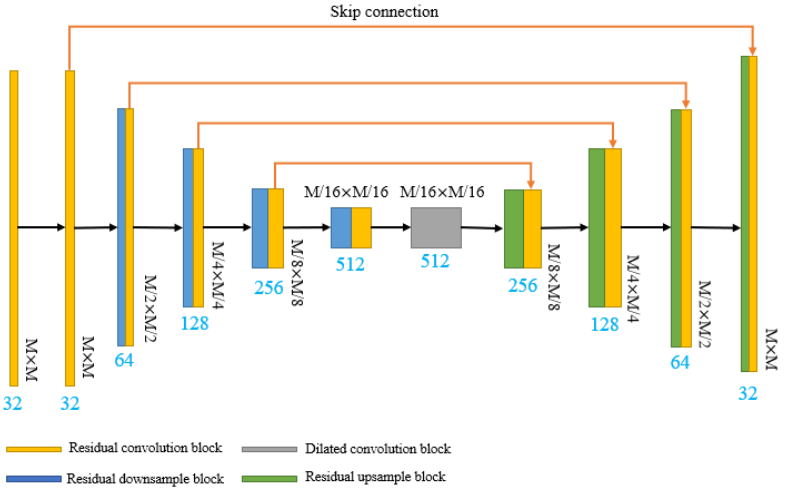


Figure 3.11: Architecture of the coarse net.

3.1.4.4 The Intermediate Supervision Block

In order to make the special net achievement stage purpose. We design an intermediate supervision block, so that we can have early supervision for the network. Therefore, after the coarse net, we could obtain coarse target images and refine these images in the next step. Inspired by [33], we design our intermediate supervision block as the following architecture.

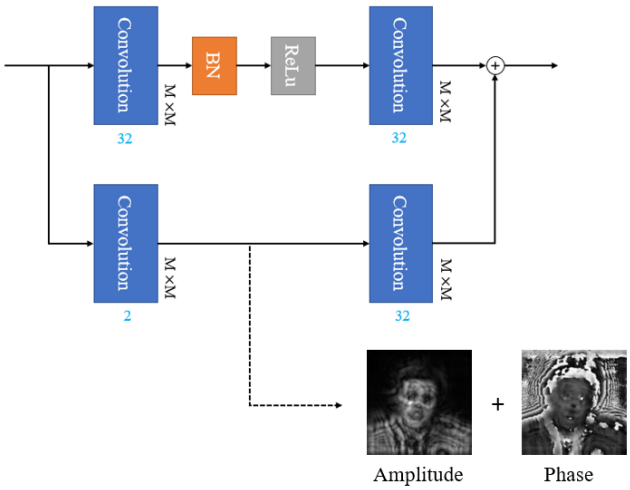


Figure 3.12: Architecture of the intermediate supervision block.

3.1.4.5 The Fine Net

The function of the fine Net is to refine the result of the coarse net. Naturally, we thought of using a super reconstruction (SR) neural network to do this work. Given a low-resolution image with coarse details, SR algorithm can convert it to the corresponding high-resolution image with better visual quality and fine details.

Here, we use DBPN [32] as our fine net to make the final prediction more clear and accurate. In DBPN, it uses mutually connected up- and down-sampling feed-forward and deep concatenation architectures. It not only uses the upsampling layer to generate diverse high resolution (HR) features but also uses the downsampling layer to map them to low revolution (LR) space. And deep concatenation architecture makes it possible to use different types HR and LR space features. For using this architecture correctly, we have to do a downsample operation to reduce the revolution of the coarse result and then follow a DBPN to do the super resolution. And the architecture of the fine net is shown in Fig.3.13

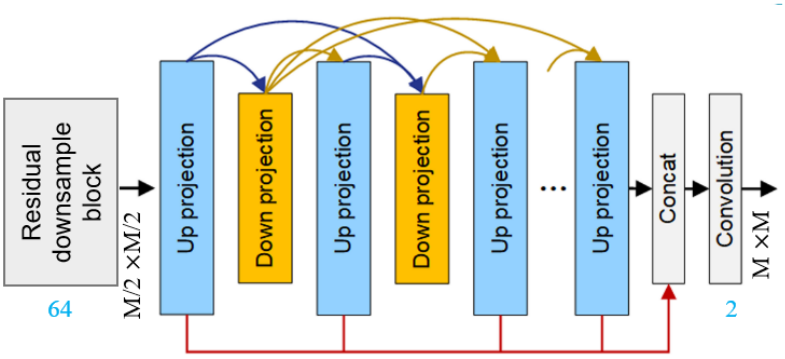


Figure 3.13: Architecture of the fine net basing on DBPN.

3.1.4.6 Loss Function

In this part, we define the loss function used for training. Actually, this problem is a regression question. Thus, we can simply use mean square error (MSE) loss. Assuming the intermediate supervision outputs are $A_{coarse}, \Phi_{coarse}$ respectively, where A represents the amplitude image and Φ indicates the phase image. And the final prediction results are expressed as A_{fine}, Φ_{fine}. Besides, A_{fine} and Φ_{fine} are the target images. Therefore, we define the loss function as,

$$\begin{aligned} Loss =& \lambda_1(\alpha\|A_{coarse} - A_{target}\|_2 + \beta\|\Phi_{coarse} - \Phi_{target}\|_2) \\ &+ \lambda_2(\alpha\|A_{fine} - A_{target}\|_2 + \beta\|\Phi_{fine} - \Phi_{target}\|_2) \end{aligned} \tag{3.22}$$

With,

$$\lambda_1 + \lambda_2 = 1, \quad \alpha + \beta = 1 \tag{3.23}$$

3.1.5 Chapter Summary

In this chapter, for our specific problem which is simulating the beam propagation on the tilted planes. We have generated a dataset $S = \{(\Phi_{input}, A_{input})_k, (\Phi_{target}, A_{target})_k; k = 1, 2, \ldots, K\}$, where every labeled data is a amplitude-phase pair. For dealing with this problem, we propose a pipeline and also suggest a neural network which takes an amplitude-phase image pair as input and outputs their diffraction pattern. Remember, the output of the DNN is not the final result, we still need some additional steps to

finish it shown in our pipeline.

Chapter 4

Experiments And Evaluation

In the previous chapter, we have introduced the dataset generation and the model's architecture. Next, we would do some experiments to test and evaluate the performance of our model.

4.0.1 Implementation Details

All of the code is implementated in Pycharm by Pytorch [26]. For training our model, we use one Nvidia GTX Titan X card on a linux machine, while the linux version of the system is CentOS 7.9. In practice, we train our model for 10 epochs with ADAM optimiser. Limited by the GPU RAM, we set the batch size as 4. Besides, we use the dynamic learning rate (LR) reducing strategy here, which ignores the first 2 epochs with no improvement for the learning rate and only decrease the LR by 10 times after the 3rd epoch if the loss still has not improved then. And we set the initial learning rate as 0.001.

Table 4.1: The hyper-parameters for training.

	epochs	batch size	learning rate	optimiser
Our Model	10	4	0.001	ADAM

As for the loss function Fig.3.22, we set $\alpha = \beta = 0.5$ and $\lambda_1 = 0.1, \lambda_2 = 0.9$. Besides, we separate our synthesized database as three different datasets, 70% train set, 20% validation set and 10% test set.

4.0.2 Qualitative Results

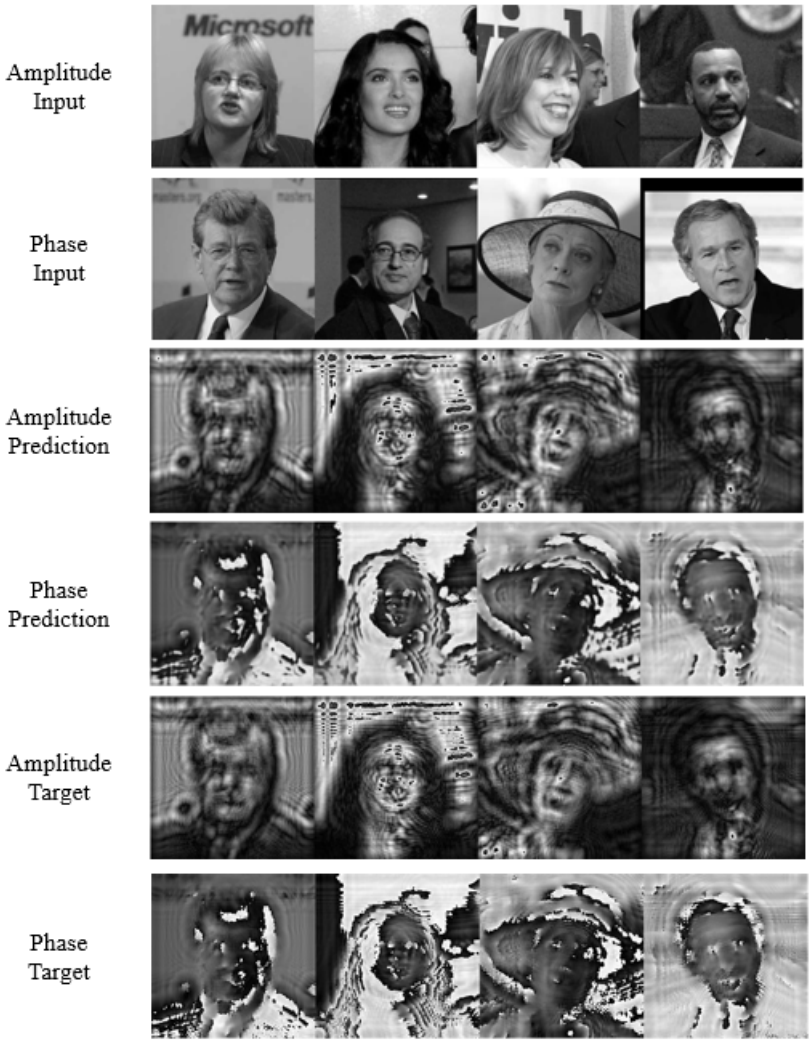


Figure 4.1: The experiment's results for the 10° tilted plane and 10*mm* propagation distance.

For this part, we test our approach on our generated dataset. And in this case, the tilted angle of the reference plane is 10° and the beam propagation distance is 10*mm*. Here, we test our trained net on the test set and Fig.4.1 shows the results. The qualities of our predictions are very close with the ground truth, which indicates the great ability of our model and fine performance for this regression problem.

4.0.3 Quantitative Evaluation

For testing our model's generalization, we generate several new datasets to discuss the influence of tilted angle and propagation distance for our DNN model. And we

use MSE loss to estimate the performance. In this section, we consider two types situations.

Table 4.2: Validation of our model with the different propagation distance on the 10° tilted reference plane.

Propagation Distance	10mm	60mm	120mm	180mm
MSE loss	0.00896	0.0129	0.0116	0.0118

Table 4.3: Validation of our model on the different tilted angle planes with 10*mm* propagation distance.

Tilted Angle	5°	10°	20°	30°
MSE loss	0.0128	0.00896	0.0106	0.0116

1. Test the performance of our model with different propagation distances. For doing We train the network on several different datasets. Keep the tilted angle as 10° and do not change other parameters, and we only substitute the propagation distance for different values. Table 4.2 shows the experiment's result.

2. Study the impact of tilted angle for our model. Here, we only change the tilted angle to generate datasets and other parameters keep the same as Chapter 3. The results of our experiments are shown in Table 4.3.

In the experiments above, all of them perform great and the MSE loss is close with each other, which means that our model is not only suitable for a certain setting. It also shows us the potential capacity to consider the unexpected factors we do not know or notice in a certain optical system.

4.0.4 Ablation Studies

For our specific problem, we propose to use a coarse-to-fine frame. And taking our neural network as an example, we do an ablation study in this section to discuss the necessity of this frame.

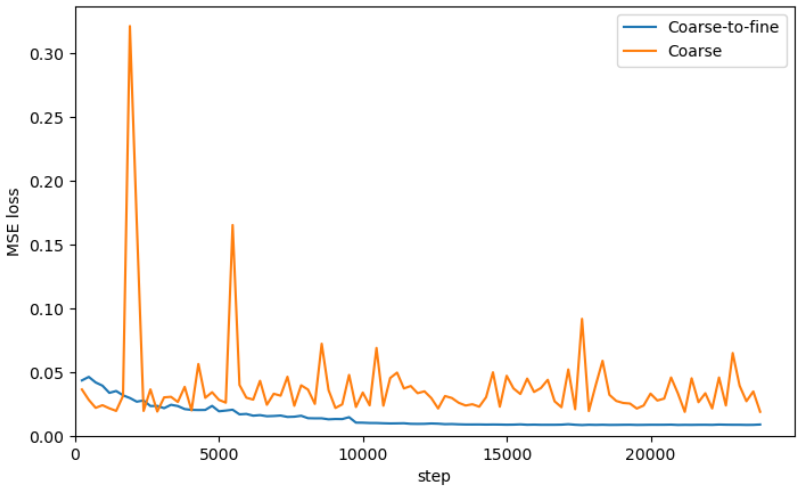


Figure 4.2: The evolution of the MSE with an increasing number of epochs.

In Fig.4.2, the x-axis means the optimizing step of our model, and the y-axis indicates the performance of the net by MSE loss. We notice that fluctuations are particularly huge in the convergence process for the only-coarse net. But for the integrated coarse-to-fine network, the performance is obviously better.

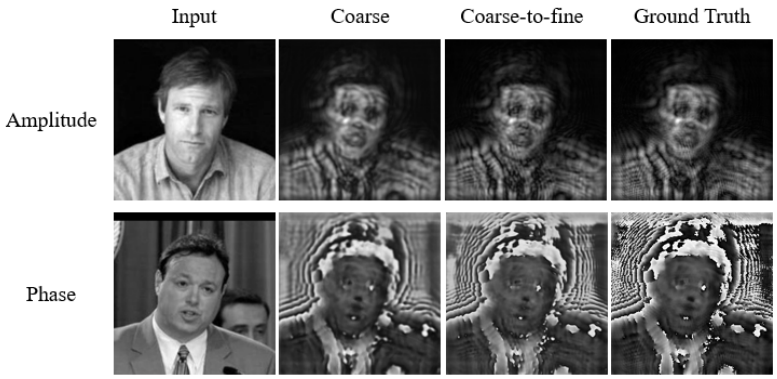


Figure 4.3: Qualitative results of the only-coarse net and coarse-to-fine net.

Fig.4.3 shows us the predictions of the coarse net and coarse-to-fine net. Both of them are trained by the same learning rate (0.001) and epochs (10). It is obvious that the result of coarse-to-fine net is better. And for the prediction of the coarse net, the image looks like being blurred and more coarse. Therefore, it shows that the following DBPN block of our model could indeed add more details for the output and refine the coarse result.

4.0.5 Running Speed

Because there is one-time FFT and one-time IFFT operation with an interpolation step, it is inescapable to waste calculation time for the traditional fast RAS method to compute the beam propagation on the tilted planes. And always, the DNN method could make good use of GPU resources which enable the parallel calculation. Fig.4.4 shows this running speed difference. And we notice that the running time increases rapidly for the RAS algorithm with the resolution (it could also represent as sampling number) of input data increases. However, it almost keeps the same which is less than 70ms for our DL method. All of these show us invincible advantages in running speed.

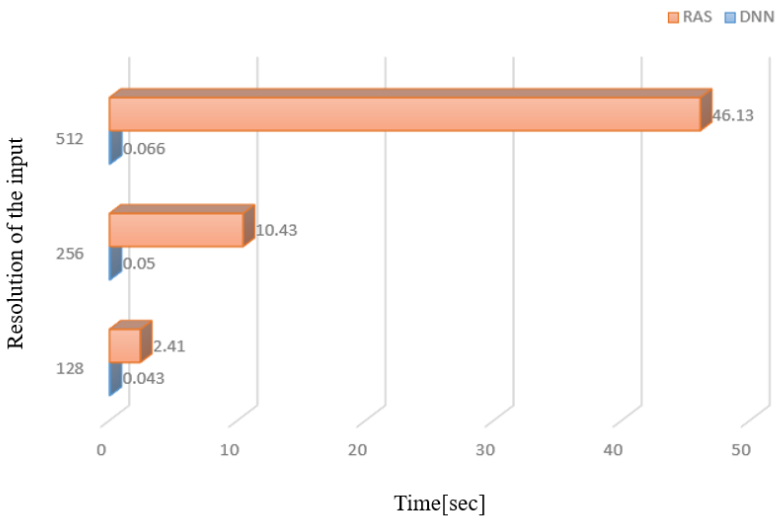


Figure 4.4: Running time of the fast RAS method and our DNN model with different resolutions of the amplitude-phase input.

4.0.6 Chapter Summary

In this chapter, we have done a comprehensive analysis for our model. we do both quantitative and qualitative evaluations. And the results show the great performance and generalization of our DNN model. Then, we also have done an ablation study to discuss the necessity of an additional fine net. Finally, the calculation time shows the invincible advantages of the DL method compared to the traditional RAS algorithm.

Chapter 5

Extension

In this chapter, we do an extension for the neural optical beam propagation. In Chapter 3, we simulate the beam propagation on the reference plane with a certain tilted angle. For this part, we try to introduce another input, the tilted angle, for the network architecture so that we can predict the optical diffraction in an arbitrary inclined plane.

5.0.1 Generate Dataset

Due to an additional input information is imported, we have to generate a completely new dataset for dealing with this specific problem. Like Chapter 3, the synthesized dataset should also contain an amplitude-phase image pair as input and the corresponding labelled amplitude-phase target pair. However, the tilted angle α should also be introduced into the input tuple. We represent the database as $S = \{(\alpha_{input}, \Phi_{input}, A_{input})_k, (\Phi_{target}, A_{target})_k; k = 1, 2, \ldots, K\}$, where Φ is the phase image and A is the amplitude image.

Also, many propagation parameters should be pre-defined. And most of them, we use the same here. In this case, the beam wavelength is $632.8nm$, the number of pixels is 256×256 and its size is $5\mu m \times 5\mu m$. And we still set the propagation distance as $10mm$. For the amplitude-phase input, we still randomly select from the Faces-LFW database, and select another human face photo as the phase input at the same time. And like Chapter 3, we resize the image shape from 250×250 to 256×256 here. However, we generate the titled angle along the y-axis from range $(0°, 45°)$. And then, the target amplitude-phase output is computed by the fast

RAS algorithm which is introduced in Section 3.1.2. Here, we still normalize the amplitude image's range as $(0, 1)$ and phase image's range also as $[0, 1]$.

5.0.2 The Model

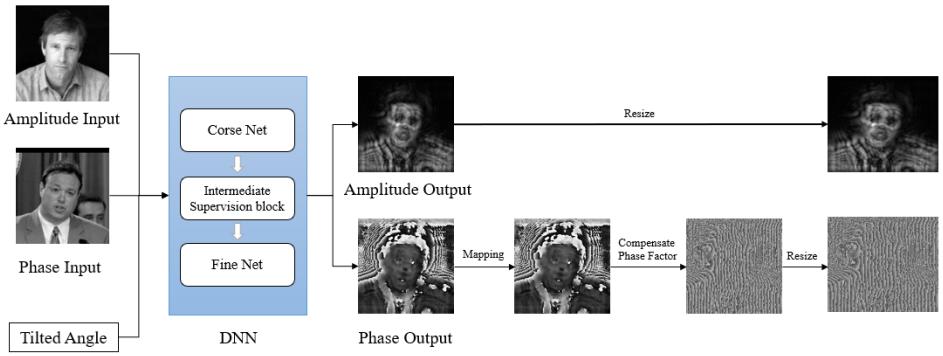


Figure 5.1: Schematic illustration of the pipeline of the model for the extension content.

There is non too much change for the pipeline. The only change is for the input of DNN, there is an additional tilted angle input. The subsequent operation after the DNN is the same as Chapter 3. So we do no further elaboration for this part, and only pay attention to the DNN.

This angle information is only added into the coarse net, and the following blocks (the intermediate supervision block and the fine net) keep the same. And the architecture of the coarse net is changed as following:

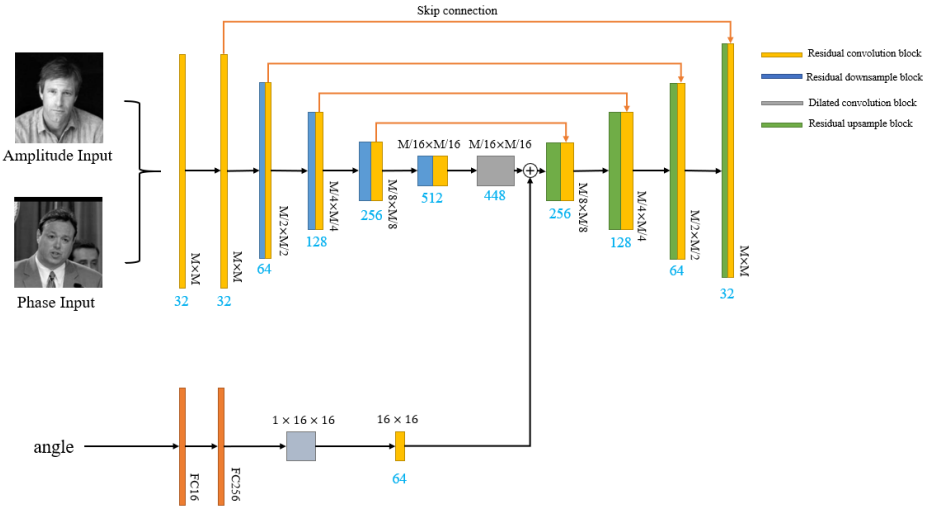


Figure 5.2: Architecture of the coarse net for an additional angle input.

We treat this tilted angle as a high-level feature, and try to embed it into the original coarse net. Thus, we firstly encode this angle information input a multi-dimensional vector. Here, we use two fully connection (FC) layers to transfer the single number input as a 256 length vector. The precise architecture of this step is: $FC(16) \rightarrow BN \rightarrow ReLU \rightarrow FC(256) \rightarrow BN \rightarrow ReLU$. And then, we resize this vector as a $1 \times 16 \times 16$ image, where 1 means the channel number. Considering the angle info is a high level feature, naturally, we should also embed it into the higher-level layer of the coarse net. And we put it into the bottleneck part of this net. Before doing this, a residual convolution block is used to increase the channel to 64. Notice that the selection of the FC layer should be flexible. In our experiments, the input data size is 256×256 which means $M = 256$. Therefore, the high and width of the torch of bottleneck part will also be 16×16. Thus we can concatenate them together.

5.0.3 Experiment Results

The majority of training details of this neural network is the same as Chapter 4. But we only train 5 epochs in this part. In our experiment, the training loss curve is still decreasing, thus it is completely fine to run more epochs.

Fig.5.3 shows us some experiment results. In this picture, every column shows one example on the reference plane with a certain titled angle. And the titled angle with the amplitude and phase input is this DNN's input data. Comparing the prediction results and the target images (the ground truth), the simulation performance has reached a great level in perceptual quality. And, use MSE loss to measure the prediction performance on the test set. The MSE value between the prediction amplitude-phase pair and the ground truth is 0.01274. In this situation, we could say that our model successfully simulate the beam propagation on the tilted plane.

Figure 5.3: Some experiment results for this extension content.

5.0.4 Chapter Summary

In this chapter, we introduce an additional input, the tilted angle, to test the ability of DL method to simulate the beam propagation. For dealing with this problem, we synthesized a new database $S = \{(\alpha_{input}, \Phi_{input}, A_{input})_k, (\Phi_{target}, A_{target})_k; k = 1, 2, \ldots, K\}$ where α is the titled angle. And we also do an extension for our model. Finally, this model performs great in our experiment.

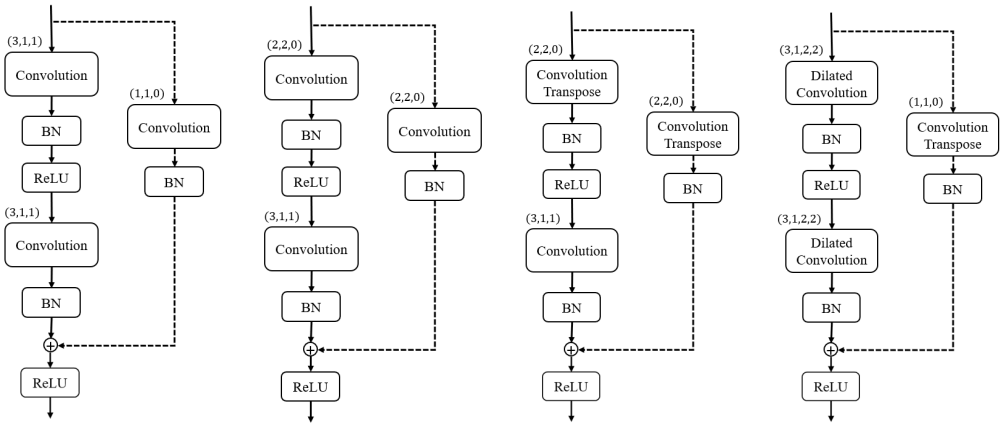
Chapter 6

Conclusions

In this thesis, we try to deal with a very new problem of how to simulate the beam propagation on the tilted planes by the DL method. For dealing with this problem, we propose a coarse-to-fine frame and designed a neural network basing on this frame. And then, we do a series of experiments including quantitative and qualitative evaluations, ablation studies and test running-time. All of these experiments demonstrate the great power of the deep learning method for this specific problem. Besides, we also do an extension which makes the network suitable for an additional input, the tiled angle, and the experiment results show great performance. We believe we provide for researchers a new perspective for solving related areas, like phase-retrieval or CGH problems.

Appendix A

The Basic Model Blocks



(a) (b) (c) (d)

Figure A.1: The basic residual blocks used in our DNN model. The number tuples which locate on the left-right corner of the convolution layers indicate (kernel, stride, padding) and (kernel, stride, padding, dilation) respectively. (a) residual convolution block; (b) residual downsample block; (c) residual upsample block; (d) dilated convolution block.

Appendix B

Code Listing

The code for this project is available in our Github:

https://github.com/mremilien/NeuralOpticalBeamPropagation

Appendix C

Colophon

This document was set in the Times Roman typeface using LaTeX and BibTeX, composed with a text editor.

Bibliography

[1] Vishwesh A Vyawahare, Gilberto Espinosa-Paredes, Gaurav Datkhile, and Pratik Kadam. Artificial neural network approximations of linear fractional neutron models. *Annals of Nuclear Energy*, 113:75–88, 2018.

[2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[4] Kyoji Matsushima, Hagen Schimmel, and Frank Wyrowski. Fast calculation method for optical diffraction on tilted planes by use of the angular spectrum of plane waves. *JOSA A*, 20(9):1755–1762, 2003.

[5] Chris Slinger, Colin Cameron, and Maurice Stanley. Computer-generated holography as a generic display technology. *Computer*, 38(8):46–53, August 2005.

[6] Stephen A Benton and V Michael Bove Jr. *Holographic imaging*. John Wiley & Sons, 2008.

[7] Ralph W Gerchberg. A practical algorithm for the determination of phase from image and diffraction plane pictures. *Optik*, 35:237–246, 1972.

[8] Michael Reed Teague. Deterministic phase retrieval: a green's function solution. *JOSA*, 73(11):1434–1441, 1983.

[9] George Barbastathis, Aydogan Ozcan, and Guohai Situ. On the use of deep learning for computational imaging. *Optica*, 6(8):921–943, 2019.

[10] Yair Rivenson, Yichen Wu, and Aydogan Ozcan. Deep learning in holography and coherent imaging. *Light: Science & Applications*, 8(1):1–8, 2019.

[11] Joseph W Goodman. Introduction to fourier optics. *Introduction to Fourier optics, 3rd ed., by JW Goodman. Englewood, CO: Roberts & Co. Publishers, 2005*, 1, 2005.

[12] Yifan Peng, Suyeon Choi, Nitish Padmanaban, and Gordon Wetzstein. Neural holography with camera-in-the-loop training. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020.

[13] Mark E Lucente. Interactive computation of holograms using a look-up table. *Journal of Electronic Imaging*, 2(1):28–34, 1993.

[14] Kyoji Matsushima and Sumio Nakahara. Extremely high-definition full-parallax computer-generated hologram created by the polygon-based method. *Applied optics*, 48(34):H54–H63, 2009.

[15] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[16] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[17] DH Hubei and TN Wiesel. Receptive fields, binocular interaction and functional arch i tec cur. *the cat's visual cortex. Journal of Physiology*, 1962.

[18] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[19] Sergio De Nicola, A Finizio, G Pierattini, P Ferraro, and D Alfieri. Angular spectrum method with correction of anamorphism for numerical reconstruction of digital holograms on tilted planes. *Optics express*, 13(24):9935–9940, 2005.

[20] Chenliang Chang, Jun Xia, Jun Wu, Wei Lei, Yi Xie, Mingwu Kang, and Qiuzhi Zhang. Scaled diffraction calculation between tilted planes using nonuniform fast fourier transform. *Optics express*, 22(14):17331–17340, 2014.

[21] Thomas Martin Lehmann, Claudia Gonner, and Klaus Spitzer. Survey: Interpolation methods in medical image processing. *IEEE transactions on medical imaging*, 18(11):1049–1075, 1999.

[22] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

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

[24] Gary B. Huang, Marwan Mattar, Honglak Lee, and Erik Learned-Miller. Learning to align from scratch. In *NIPS*, 2012.

[25] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc,

E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[27] Ayan Sinha, Justin Lee, Shuai Li, and George Barbastathis. Lensless computational imaging through deep learning. *Optica*, 4(9):1117–1125, 2017.

[28] Ryoichi Horisaki, Ryosuke Takagi, and Jun Tanida. Deep-learning-generated holography. *Applied optics*, 57(14):3859–3863, 2018.

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[30] Fei Wang, Yaoming Bian, Haichao Wang, Meng Lyu, Giancarlo Pedrini, Wolfgang Osten, George Barbastathis, and Guohai Situ. Phase imaging with an untrained neural network. *Light: Science & Applications*, 9(1):1–7, 2020.

[31] Jiachen Wu, Liangcai Cao, and George Barbastathis. Dnn-fza camera: a deep learning approach toward broadband fza lensless imaging. *Optics Letters*, 46(1):130–133, 2021.

[32] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1664–1673, 2018.

[33] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.