四川大学

本科毕业论文（设计）

论文题目 基于深度学习的光学散斑图像重建

学　　院 材料科学与工程学院

专　　业 生物医学工程

学生姓名 程圣福

学　　号 2015141221007

指导教师 Puxiang Lai（香港理工大学），陈科

教务处制表

二０一九 年 五月 五日

基于深度学习的光学散斑图像重建

专业：生物医学工程

学生：程圣福　　指导教师：Puxiang Lai，陈科

[摘要]：透过复杂的散射介质成像是一个很多场景下都能遇到的普遍问题，例如深层组织成像。光散射扰乱了在介质中传播的波，结果没有目标的图像产生，只得到看起来随机的图案称为散斑。人们已经付出了巨大的实验上的努力去探索透过散射介质成像，例如数字全息术、波前整形技术以及记忆效应，但这些都要求复杂的光学设置。 而就计算成像而言，目前的进展主要依赖传输矩阵（TM）的方法来对穿过散射介质后扰乱的图像进行解扰。 最近几年，深度学习技术已被越来越多地用于透过散射介质的计算成像。通过数据驱动的方式，深度学习能构建一种计算框架作为光在散射介质中传输过程的函数拟合。这为通过学习匹配透过散射介质的输入/输出波前，进而从探测的散斑图案中重建目标图像提供了可能。该毕业论文的研究重点是从测得的散斑强度图像中重建出目标图像。第一步就是获取包含目标图像和对应的散斑图像的数据集。开展了光学实验，实验中 MNIST 手写数字图像展示在空间光调制器（SLM）上用于对即将入射到扩散器的激光进行相位调制，并用相机采集到相应的散斑图案。除了 MNIST 数据集之外，还下载了一个公开的实验数据集。在尝试用深度学习做图像重建前，研究了透过散射介质成像情景下的相位复原问题。基于下载的数据集,利用数据集中的实验测量传输矩阵作为投影算子的GS迭代算法能够成功地复原输入图像。最后，使用改进了的 U-Net 模型，基于测得的散斑强度图像能有效地重建出空间光调制器上的 MNIST 数字图像，以及使用了一个残差网络 ResNet18 对散斑图像做识别。这证明看似随机的散斑图案包含输入数字体的信息，使得有效的重建与进一步识别成为可能。

[关键词]：深度学习；散斑；相位复原；图像重建

Deep Learning based Image Reconstruction from Optical Speckle Pattern

Department: Biomedical Engineering

Student：Shengfu Cheng Supervisor：Puxiang Lai, Chen Ke

[Abstract]: Imaging through scattering complex media is a pervasive problem encountered in many cases, such as deep tissue imaging. Light scattering scrambles the waves propagating inside the media, with the result of no image of the target being produced, but only seemingly random pattern known as speckle. There have been tremendous experimental efforts by exploiting digital holography, wavefront shaping techniques or memory effect to explore imaging through scattering media, while complex optical setups are usually required. In the computational imaging scenario, major progresses have been made by using transmission matrix (TM) method to descramble the transmitted images across scattering media. In the recent years, Deep Learning techniques have been increasingly employed for computational imaging through scattering media. Through the data-driven learning approach, deep learning can build a computational architecture as a generic function approximation of light propagation process across complex media. This opens up the avenue to reconstruct the target objects from the recorded speckle patterns via learning to map the input/output wavefronts through scattering media. In this thesis, the research focus is to reconstruct target images from intensity-only measurement of speckle patterns. The first step is obtaining image dataset that contains target images and corresponding speckle patterns. Optical experiment was conducted where the MNIST handwritten digit images were displayed on a phase-only Spatial Light Modulator (SLM) to manipulate the light incident onto a diffuser, with the resulting speckle patterns being measured by a camera. Apart from MNIST dataset, another public empirical dataset was also downloaded. Before the attempt to use deep learning for image reconstruction, the research explored phase retrieval in the scenario of imaging through scattering media. Based on the public dataset, iterative GS algorithm employing empirical TM measurement operator was able to retrieve SLM patterns successfully. Last but not least, a modified U-net model has been utilized to efficiently reconstruct the MNIST images displayed on SLM from the recorded intensity of speckle patterns, and further recognition for the speckle patterns via a ResNet18 model. This demonstrate that seeming random speckle patterns contain information about the input digit images, rendering it possible for efficient reconstruction and recognition.

[Key words]: deep learning; speckle; phase retrieval; image reconstruction

CONTENTS

CHAPTER 1 Introduction

1.1 Computational imaging through scattering complex media

Considering that coherent light passing through scattering media, the wavefronts are distorted by multiple light scattering inside inhomogeneous refractive-index profile, with the result of no image of the target being produced but only seeming random patterns known as speckles out of complex media. Imaging through scattering complex media is a pervasive problem encountered in many cases, especially for deep tissue optical imaging[1]. This problem is of particular in the field of biomedicine, as is our long-desired dream to see deep and seeing clearly into biological tissues with visible light. There have been tremendous experimental efforts by exploiting digital holography[2, 3], wavefront shaping techniques[4-6] or memory effect[7, 8] to explore imaging through scattering media, while complex optical setups are usually required.

In the computational imaging scenario, major progress has been made by using transmission matrix (TM) method[5, 9] to descramble the transmitted images through scattering media. A TM generally has vast number of elements to characterize the relationship between the input-output optical field of a scattering medium. Suppose the scattering process is linear, we can mathematically describe it as $E_m^{out} = \sum_n T_{mn} E_n^{in}$, where $E_n^{in} \in \mathbb{C}^N$ is the input and $E_m^{out} \in \mathbb{C}^M$ the output. It usually requires large-scale interferometric measurement of all input and output light modes to measure a TM of a scattering medium. Once the TM of the scattering medium is measured in advance, it can be inverted to descramble any transmitted image. Many studies have described TM methods for imaging or image transmission through complex media like diffusers[5, 10] and multimode optical fibers[11, 12]. However, such TM approach is usually of experimental and calculational complexity, along with high susceptibility to the perturbation of the imaging system itself. Besides, the TM method needs to set the assumption that the optical processes are linear and a single TM could be able to model them. However, this couldn't be met when the experimental environments are noisy and inadequate.

In addition, there are also phase retrieval approaches in the context of optical imaging. Phase retrieval aims to reconstruct signals from the intensity measurements of linear transformation of the signals[13], typically the Gerchberg-Saxton-Fienup-type iterative algorithm[14, 15]. The general phase retrieval problem can be formulated as solving the following optimization problem: when given the magnitude measurement $\mathbf{b} = |\mathbf{A}\mathbf{x}|$, and $\mathbf{A}$ represents the known measurement operator, to determine $\mathbf{x}$. Gerchberg and Saxton were the pioneers who proposed a practical solution which relies on

alternating projection, and now such algorithm is named after them. What's more, Fienup has made the major improvements to it. Most phase retrieval problems aim to reconstruct a signal from its Fourier magnitude, since Fourier transform is the measurement operator in many imaging modalities like X-ray crystallography, optical diffraction imaging and astronomical imaging. However, such Fourier transform-based measurement operator could not be applicable to our scenario of imaging through scattering medium. In other words, Fourier measurement in imaging modalities that rely upon Fraunhofer diffraction can be extend to TM measurement in imaging through scattering media[16]. It has been shown that iterative phase retrieval algorithm with TM as the measurement operator can be efficient to reconstruct images from speckle patterns produced due to multiply light scattering.

In the recent years, deep learning techniques have been increasingly employed for computational imaging through scattering media. Through the data-driven learning approach, deep learning can build a computational architecture as a generic function approximation of light propagation process across complex media. This opens up the avenue to reconstruct the target objects from the recorded speckle patterns via learning to map the input/output wavefronts through scattering media. Here, we'd like to report recent developments in learning-based computational imaging through scattering media, which proves to be simpler and more robust compared to the aforementioned TM method. The key of such inverse scattering process is to recover the target objects from the recorded speckles via "learning" to approximate solutions to inverse problems in computational imaging.

For the problem of computational imaging through scattering media, suppose f is the unknown target object, I is the recorded speckle pattern, and H is the forward imaging operator, with an additional regularization item $\Phi(\cdot)$, we can model it by the Tikhonov-Wiener optimization functional for solving the inverse problem: $\hat{f} = \arg\min_f \|Hf - I\|^2 + \alpha\Phi(f)$, where α serves as the regularization parameter, and $\hat{f}$ is the estimate of the target. To work out this ill-posed reconstruction functional, H and Φ must be known explicitly or parametrically. Although there are several approaches to determine these prior representations, the process is often complicated and prone to errors. Recent years have seen AI techniques in solving this inverse problem by learning these operators implicitly through the training of examples of targets imaged through scattering. In 2016, Horisaki *et al.* first demonstrated such idea by proposing a SVR architecture to learn the scatterer and the prior of faces being imaged through random medium, with the result of correctly reconstruct face objects. The authors also made it clear that the SVRs with shallow fully connected two-lay architectures suffers a generalization limitation. Such ML based approach in the context of computational imaging through scattering media has had a positive impact in boosting other related

works, especially for the successful introduction of DL techniques. A DNN can be trained to learn H, Φ and α implicitly, and to act as the inverse operator H^{inv} to recover the target object from the recorded speckle, with the expression $\widehat{f} = H^{inv} I$ mathematically. This learning approach requires no prior knowledge and is expected to be a more robust approximator to nonlinear operator, but a large training dataset of known image pairs of target objects and their corresponding speckle patterns must be available.

Meng Lyu *et al*.[17] used a DNN model with two reshaping layers, four hidden layers and one output layer to retrieve the MNIST handwritten digit images behind a 3mm thick white polystyrene slab. The method relies on the way of "end-to-end" mapping, that is, the network maps the speckle patterns observed behind the slab to the corresponding phase SLM objects before the slab. It is simple yet has opened up avenues to explore optical imaging through more general complex systems. Syan Sinha *et al*.[18] in 2017, for the first time, experimentally built and tested a lensless imaging system. In the computation part of the system, a DNN model was trained to recover phase objects displayed on SLM given the recorded intensity diffraction patterns. What's noticeable is that their network has actually learned a model of the underlying physical characteristics of the imaging system. This can be demonstrated by the fact that the network could accurately reconstruct images of other classes from Faces-LFW database while it was trained exclusively on images from the ImageNet database. The authors in 2018, further proposed a densely connected convolutional neural network architecture named "IDiffNet" to solve the problem of imaging through diffuse media.[19] They also introduced the negative Pearson correlation coefficient (NPCC) as the loss function which proves to be efficient for sparse image reconstruction. The "IDiffNets" trained on different databases all seems to learn automatically the physical characteristics of the scattering media, such as the degree of shift invariance and the priors restricting the objects. This convolutional architecture was shown to achieve higher space–bandwidth product of reconstructed images than the previously reported, and to exhibit a robustness toward the choice of the priors. In order to address the susceptibility to speckle decorrelation faced by using the deterministic TM method of "one-to-one" mapping, a "one-to-all" DL technique was proposed for imaging through scattering media in 2018 by Yunzhe Li *et al*.[20] The authors built a CNN model to learn the statistical information about the speckle intensity patterns measured on a series of diffusers with different microstructures yet the same macroscopic parameters. The trained CNN could invert speckle patterns recorded from different diffusers to the corresponding target object accurately. And this may be of great significance to scalable imaging through scattering media.

1.2 Research contents and arrangements of chapters

In this thesis, the research focus is to reconstruct target images from intensity-only measurement of speckle patterns.

Chapter 1 is an introduction about computational imaging through scattering media, with an emphasis on deep learning-based approaches. Besides, it makes arrangements for the subsequent chapter contents.

Chapter 2 is a detailed description about the optical experiment for speckle patterns collection using MNIST database. The MNIST handwritten digit images were displayed on a phase-only SLM to manipulate the light incident onto a diffuser, with the resulting speckle patterns being measured by a CMOS camera. Apart from MNIST dataset, another public empirical dataset was also downloaded for iterative phase retrieval algorithm.

Chapter 3 is about the research on phase retrieval for image reconstruction. We first review the generalized iterative phase retrieval algorithms that rely on alternating projection between real domain and Fourier domain. Then we give a MATLAB demo to show phase retrieval from optical diffraction pattern. Next, we further explore phase retrieval in the scenario of imaging through scattering media. As the downloaded public dataset comprises the transmission-matric that relays the input to the output across scattering medium, iterative phase retrieval algorithm employing the empirical TM measurement operator on this dataset was shown to retrieve SLM patterns successfully.

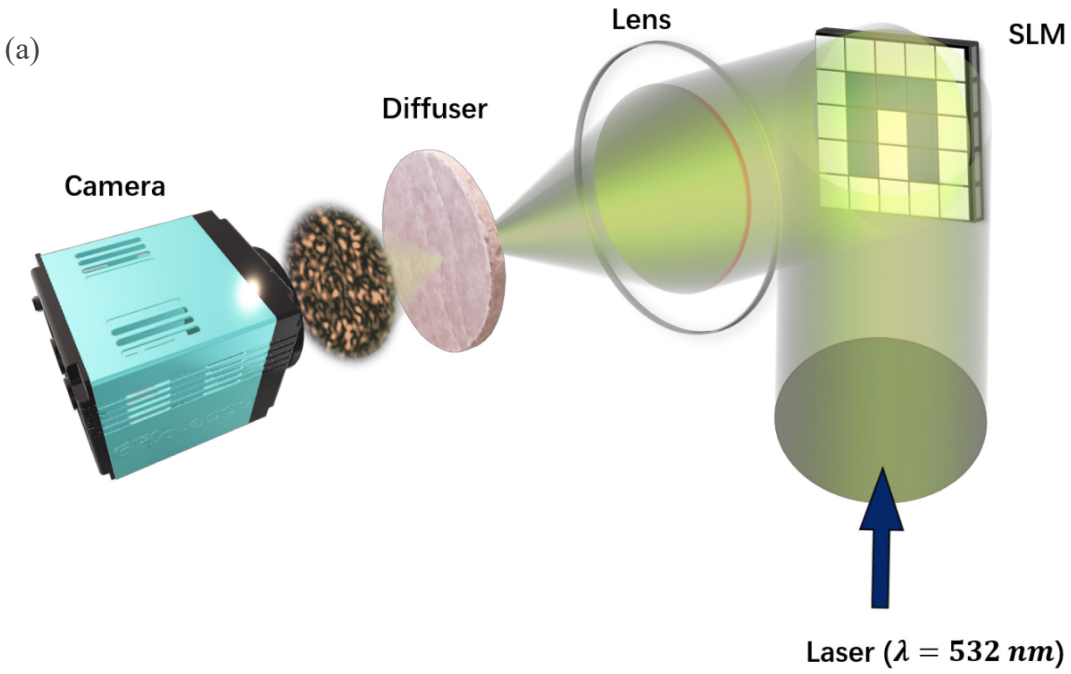
Chapter 4 gives the methods and results of the implementation of deep neural network (DNN) models in the reconstruction of MNIST images displayed on SLM from the speckle patterns, and further recognition for speckle patterns. In detail, a modified U-net model that employs dense blocks has been utilized to efficiently reconstruct the SLM input images from the intensity measurement of speckle patterns. In addition, with the reconstruction task done, a pretrained ResNet18 model will do transfer learning for image classification of the speckle patterns. All the deep learning programming was under the Pytorch framework.

Chapter 5 is the discussion and conclusion on this research, with further envision about future advances in learning-based image recovery and reconstruction in the scenario of computational imaging through scattering media.

CHAPTER 2 “Object-Speckle” Image Dataset Acquisition

2.1 Optical experiment for MNIST database’ speckle patterns collection

Our optical experimental apparatus is configured as depicted in Fig 1, with both simplified illustration and the real picture. Images extracted from MNIST database are used as a phase objects and are displayed on a phase-only modulation spatial light modulator (SLM) (Pluto Holoeye 1080p, German). A collimated continuous-wave coherent laser beam functioning at $\lambda = 532$ nm (Spectra Physics) is expanded so that the screen of SLM can be fully illuminated. The magnified laser beam is then modulated by the SLM uploading phase object. Since the handwritten digits in MNIST have the dimension of 28-by-28, these images are up-sampled by enlarging to the dimension of 1064-by-1064 and displayed on the SLM sequentially. The phase-only SLM converts the 8-bit grayscale (0-255) to phase delay (0-2π in radian). As the images in MNIST are also quantified by 8-bit grayscale, the images will be rescaled from 0-255 to 0-127 and then uploaded to the SLM, for strengthening the modulation efficiency. Modulated by the image on SLM, the laser beam is incident onto a diffuser (220-grid Thorlabs, US) by a lens. Light scattering-induced speckle patterns are captured by a CMOS camera (PointGrey Canada). The camera and SLM are synchronized by a MATLAB program during data acquisition.



(b)

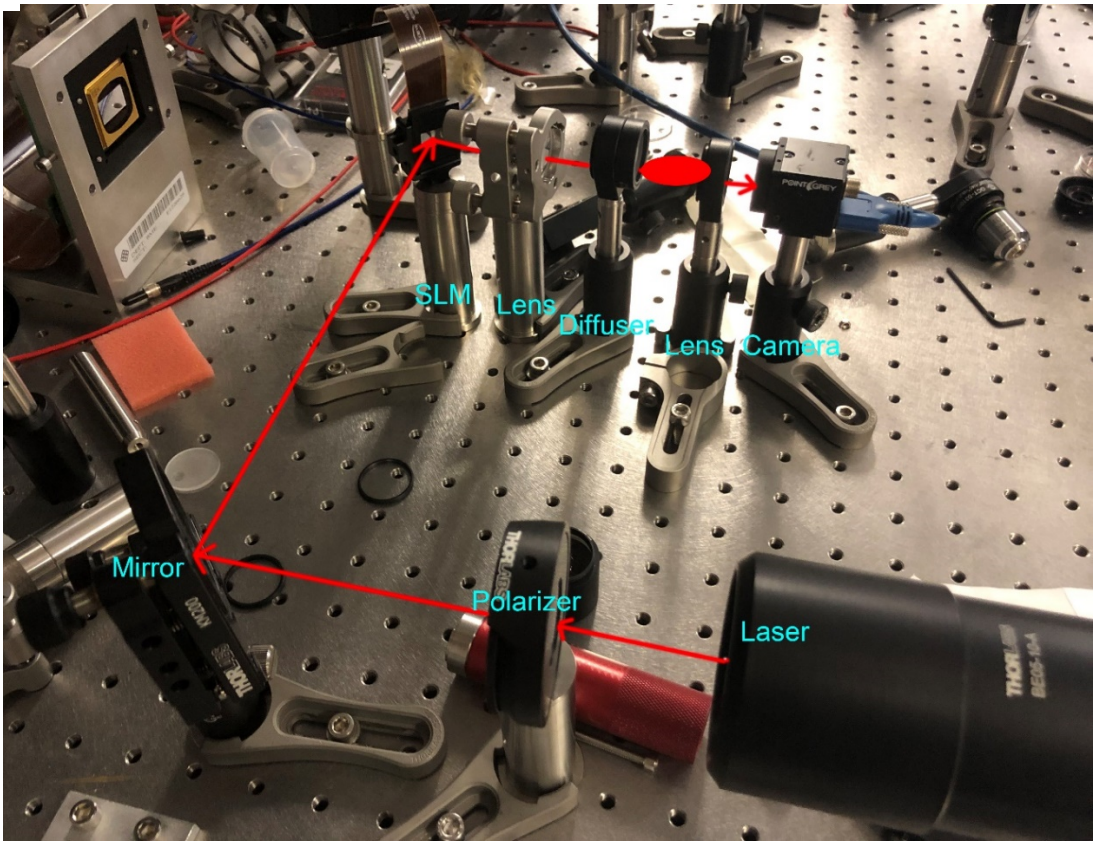


Fig 1. Experimental setup for collecting speckle patterns. Phase objects are displayed on the SLM and illuminated by an expanded laser beam (λ=532 nm).

We have collected 20,000 "object-speckle" image pairs altogether and they are saved in .mat format processed by MATLAB. In detail, our MNIST dataset consists of the training set and the test set. In the training set, there are 1,9800 image pairs containing MNIST images from 1st to 9800th and their corresponding speckle patterns. As for the test set, MNIST images from 9801st to 1,000th and their corresponding speckle patterns.

2.2 A publicly available dataset containing transmission matrix

According to the Ref[16], the authors have publicized a dataset that contains the input SLM patterns and the corresponding speckle patterns, and especially the transmission-matrix corresponding to the scattering medium from which the speckle patterns were produced. The dataset can be downloaded from the website: http://dsp.rice.edu/research/transmissionmatrices/. Since the dataset includes empirical transmission matrixes, we can employ it for phase retrieval in the scenario of imaging through scattering media, see details in Chapter 3.

CHAPTER 3 Phase Retrieval for image reconstruction

In this chapter, we introduce iterative phase retrieval method using transmission matrix as the measurement operator, with the purpose of reconstructing image from speckle patterns produced from scattering media. We firstly give a brief review of iterative phase retrieval algorithms, based on which is a demo of phase retrieval from optical diffraction pattern. In the scenario of imaging through scattering media, thanks to the dataset that contains transmission matrix, we want to use the GS algorithm using TM as the measurement operator to retrieve the phase displayed on SLM incident on scattering medium. With TM functions as FT and its pseudoinverse functions as IFT in alternating project method, we can successfully reconstruct the SLM phase patterns.

3.1 Iterative algorithms for phase retrieval

3.1.1 Introduction

As known, the phase is usually more important compared to the amplitude as the former contains the contour information of an object. However, in many optical settings, typical optical detection devices such as CCD and CMOS camera could only measure the intensity, instead of the phase, of the light wave. This is because what the devices measure actually is the photon flux and it is proportional to the magnitude square of the optical field, and electronic device that convert photons to electrons could not capture the phase of electromagnetic field ($\sim 10^{15}$ Hz).[21] Consequently, it is necessary to develop computational algorithms to recover the phase of object given the measured intensity. Phase retrieval aims to reconstruct a complex-valued signal from the magnitude of its linear measurement.[13] The general phase retrieval problem can be modeled as: When given the magnitude measurement $b = |\mathbf{Ax}|$, where $\mathbf{A}$ is the known measurement operator, to determine $\mathbf{x}$.[22]

For many imaging modalities that rely upon Fraunhofer diffraction such as X-ray crystallography, optical diffraction imaging and astronomical imaging, the far filed of electromagnetic fields correspond to the Fourier transform of their near field.[21] Thus, optical measurement devices essentially measure the Fourier magnitude in the far field. Most phase retrieval problems reconstruct a signal from its Fourier magnitude, which means Fourier transform is the measurement operator. Basically, there are three categories of phase retrieval methods: digital holography (DH), the transport of intensity equation (TIE) and iterative algorithms.[23] Here, we focus on the iterative algorithms for phase retrieval.

Let's see the mathematical model of phase retrieval. Suppose that $|f(\mathbf{r})|$ is the magnitude of

object in the real plane and $|F(\mathbf{p})|$ the Fourier magnitude in measurement plane. Phase retrieval problem can be formulated as solving the following optimization function:

$$\min_{\phi(\mathbf{r})} \left\| |F(\mathbf{p})| - |\mathcal{F}\{|g(\mathbf{r})| \exp[j\phi(\mathbf{r})]\}| \right\|$$

for the missing phase $\phi(\mathbf{r})$, and $\|\cdot\|$ is the norm.[23] Early approaches to phase retrieval were based on alternating projections, pioneered by Gerchberg and Saxton who first provided a practicable solution to this problem in 1972.[13]

The Gerchberg-Saxton (GS) algorithm attempts to recover a complex object from the intensity measurement at the real (object) plane and the Fourier (image) plane. There are iteratively forward and inverse Fourier transforms between the two planes, in which the real domain $(|g|)$ and Fourier amplitude constraints $(|\mathcal{F}|)$ are respectively imposed.[23] The algorithm minimizes the aforementioned objective by starting with a random initialization and iteratively imposing the two constraints using projections.[13] The detailed steps are illustrated in the following Fig.

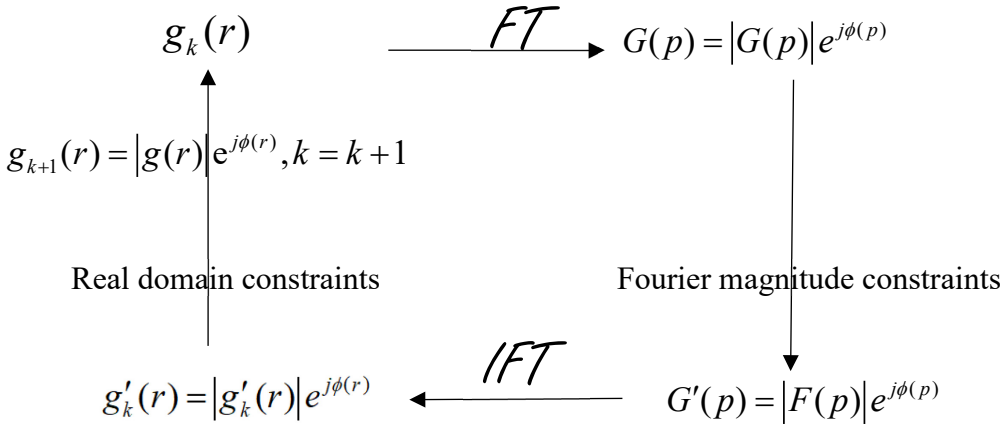


Fig 2. GS algorithm based on iterative Fourier transform

The GS algorithm can be viewed alternatively as the projection onto constraint sets in a Hilbert space[23]:

$$g_{k+1}(r) = P_s P_m g_k(r)$$

Here, $g_k(r)$ is the object retrieved in k^{th} iteration. P_m is the projector onto amplitude constraint sets in the Fourier domain, which preserves the phase of the Fourier transform and combine it with *a prior* amplitude, based on which new transform is conducted iteratively. It is given by:

$$P_m g_k(r) \leftrightarrow F(p)e^{j\phi(p)}$$

and P_s is the projector onto support constraint sets in the real domain. As the projections are between a convex set (for the real domain constraints) and a non-convex set (for the Fourier magnitude constraints), the convergence is often to a local minimum.[13]

Fienup[24] has extended this alternating projection method in 1980s. In order to solve convergence stagnation or trapping, he added different real domain constraints and introduced correction steps to the real domain projection, known as the Error Reduction (ER) and Hybrid Input-Output (HIO) algorithm. We can describe the two improved algorithms as follows:

ER $$g_{k+1}(r) = P_s g'_k(r) = \begin{cases} g_k(r), |r| \le a \\ 0, |r| > a \end{cases}$$ (support constraint)

HIO $$g_{k+1}(r) = P_s g'_k(r) = \begin{cases} g'_k(r), r \in S \\ g_k(r) - \beta g'_k(r), r \notin S \end{cases} \quad S = \{r \mid g'_k(r) > 0\}$$ (positivity constraint)

Furthermore, more projection algorithms with various support domain and error constraints have been proposed, summarized in Ref[25], we refer to it as the following table.

011301-5 Phase retrieval: Projection algorithms

TABLE I. Summary of various algorithms.

Algorithm	Iteration $\rho^{(n+1)}=$
ER	$P_s P_m \rho^{(n)}$
SF	$R_s P_m \rho^{(n)}$
HIO	$\begin{cases} P_m \rho^{(n)}(r) & r \in S \\ (I - \beta P_m)\rho^{(n)}(r) & r \notin S \end{cases}$
DM	$\{I + \beta P_s[(1+\gamma_s)P_m - \gamma_s I] - \beta P_m[(1+\gamma_m)P_s - \gamma_m I]\}\rho^{(n)}$
ASR	$\frac{1}{2}[R_s R_m + I]\rho^{(n)}$
HPR	$\frac{1}{2}[R_s(R_m + (\beta-1)P_m) + I + (1-\beta)P_m]\rho^{(n)}$
RAAR	$\left[\frac{1}{2}\beta(R_s R_m + I) + (1-\beta)P_m\right]\rho^{(n)}$

3.1.2 MATLAB demo for iterative Phase Retrieval

Here we give a toy demo of using different alternating projection algorithms to explore phase retrieval problem. Our scenario is a MNIST image of digit “5” is displayed on SLM as the phase object to modulate an incident optical field which propagates in FREE SPACE to produce a diffraction pattern. To demonstrate the retrieval of the phase of object, assuming the amplitude of object is 1, we use a randomly initialized 2D array in the interval (0,1) to represent the phase of object, and the 2D Fourier transform of the object to represent the diffraction pattern. Now given the amplitude of the diffraction pattern of object and the randomly initialized object phase, we want to retrieve the phase of object using various iterative Fourier transform algorithms. Fig. shows our phase retrieval results using Difference Map algorithm, and the code of our MATLAB demo is in the Appendix 1.

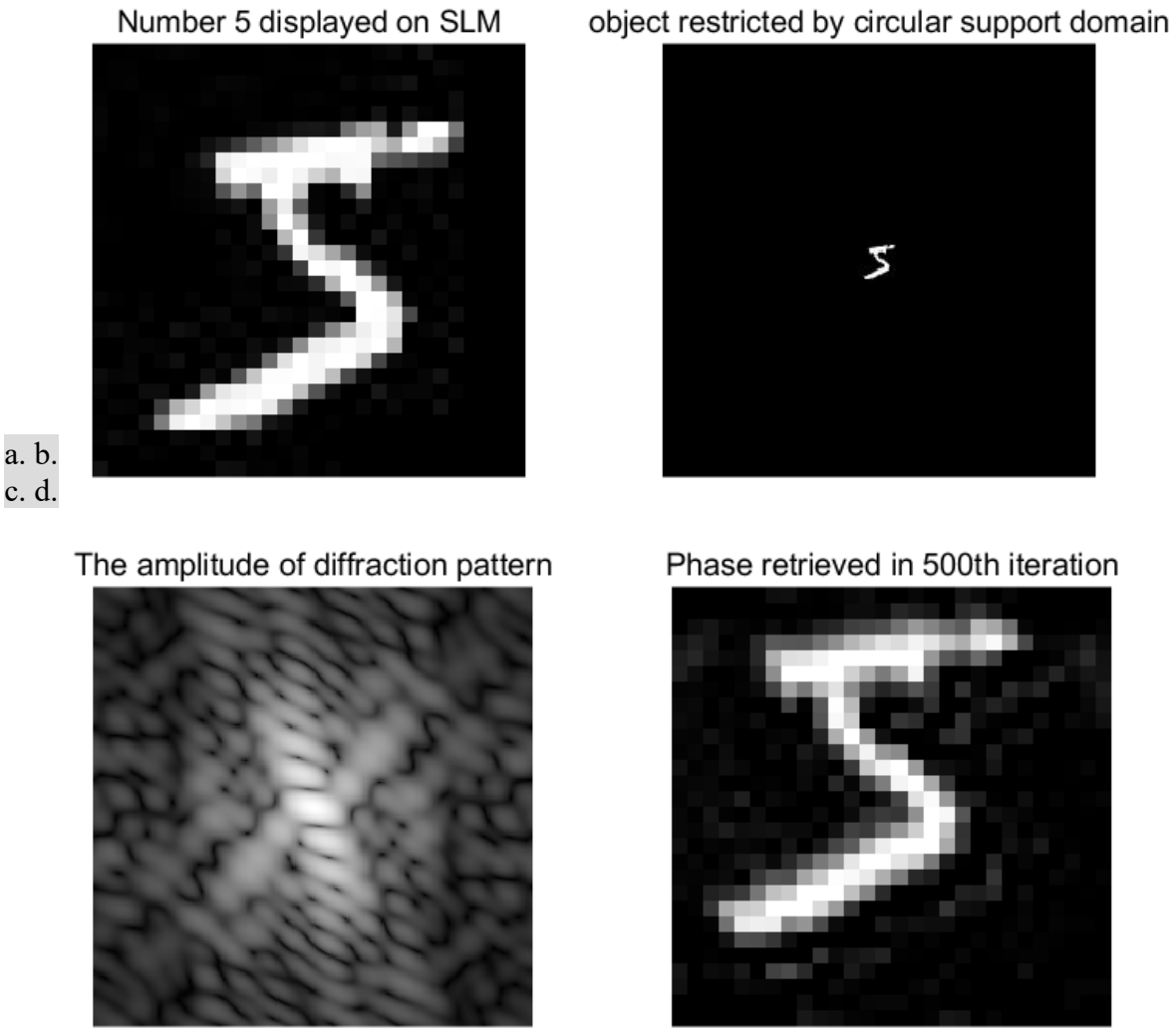


Fig 3. Results of phase retrieval using DM algorithm

3.2 Iterative phase retrieval using TM measurement operator

Now we come back to the problem of imaging through scattering media. In this scenario, it is the transmission matrix (TM) that describes the input-output relationship of a complex optical wavefront as it propagates through a scattering medium. And we want to reconstruct the input wavefront, that is, the phase of object used to modulate the incident light, from the intensity measurement of speckle pattern. Therefore, Fourier measurement in imaging modalities that rely upon Fraunhofer diffraction can be extend to TM measurement in imaging through scattering media. For our problem, suppose the scattering process is linear, the input wavefront $\mathbf{x} \in \mathbb{C}^N$, now we're given the intensity measurement $\mathbf{b} = |\mathbf{A}\mathbf{x} + \epsilon|$, where $\mathbf{A} \in \mathbb{C}^{M \times N}$ represents the TM measurement operator and ϵ is the noise, to determine $\mathbf{x}$.

Once we know the TM, we can use the aforementioned iterative phase retrieval algorithms to reconstruct the phase of object from the magnitude measurement, thus imaging through scattering media could be achieved. Thanks to the public dataset containing the TM, which has been mentioned at the end of Chapter 2, we can conduct iterative phase retrieval using TM measurement operator. The steps could be illustrated as Fig.

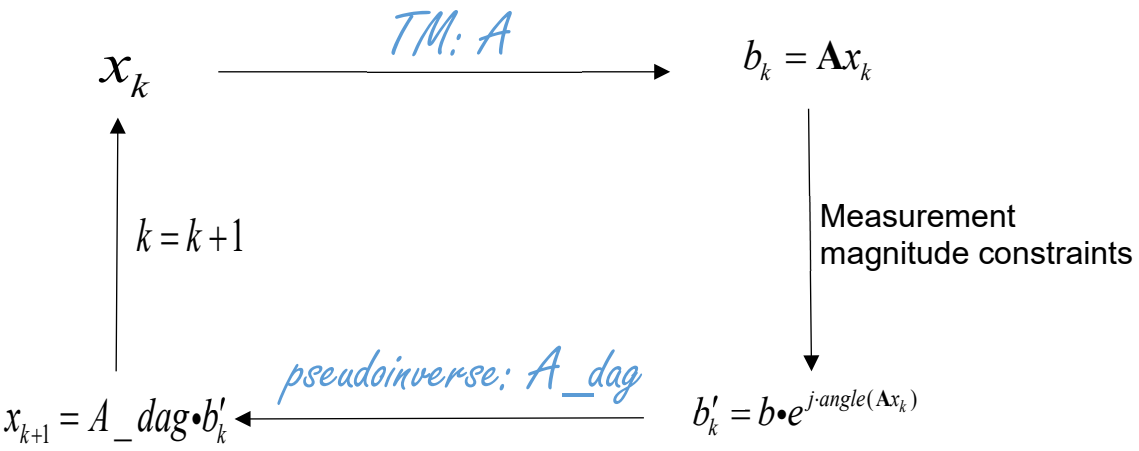


Fig 4. GS algorithm using TM measurement operator

When given the TM of the scattering medium and the intensity measurement of speckle patterns, we can reconstruct the phase SLM patterns used to modulate the incident light. The phase retrieval results using GS algorithm and TM after 100 iterations is shown in Fig 5, and the code is available in Appendix 2.

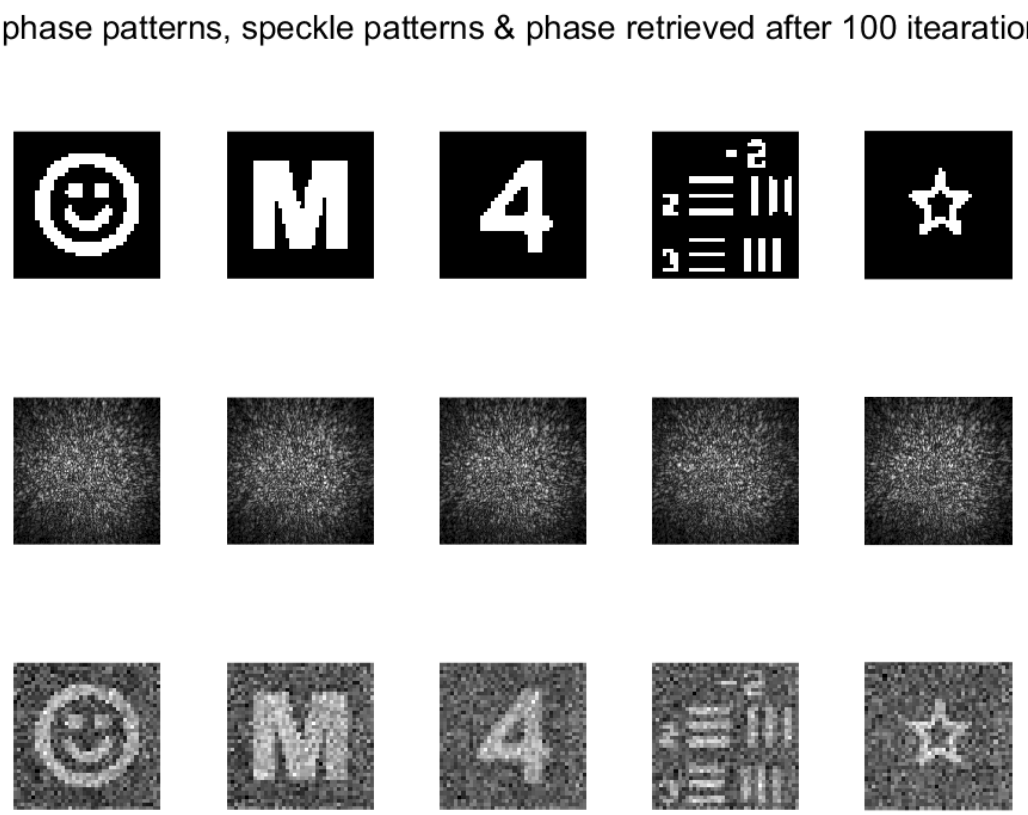


Fig 5. The first row consists of 5 40 x 40 phase SLM patterns incident on scattering medium. The second row is composed of the corresponding 256 x 256 speckle patterns captured by camera. The last row corresponds to the reconstructions after 100 iterations by GS algorithm with TM measurement operator employed.

Compared to imaging or image transmission through scattering media based on TM method described in Ref[9], our iterative phase retrieval using GS algorithm and TM has several advantages. The first is that we only rely on the magnitude information, instead of the complex-valued speckle pattern in the case of ordinary TM method. As the measurement is usually noisy, the use of inverse transmission matrix is very unstable to descramble the transmitted image. Hence, there are several reconstruction operators proposed to tackle the measurement with noise. By contrast, our iterative algorithm is simpler and more robust to reconstruct the SLM patterns in a noisy measurement environment.

CHAPTER 4 DNN Models for Reconstruction and Recognition for Speckle Patterns

This chapter details the methods and results of the implementation of our DNN models to reconstruct the MNIST digit images from the recorded intensity of the speckle patterns, and further recognition for speckle patterns. In a word, we use two types of networks for the reconstruction and recognition tasks, respectively. Firstly, we built a modified "U-Net"[26] type DNN that employs dense blocks to make predictions of MNIST digits from their speckle patterns. With the reconstruction task done, we then propose to do further recognition of speckle patterns, and there two approaches for the recognition task. One is the classification of the reconstructed MNIST images, the other takes speckle patterns directly for image classification.

We want to use an 18-layer DNN based on residual networks[27] (ResNet18) to train on our preprocessed MNIST dataset first. With a pretrained weighting parameters that have fitted the MNIST image classification, the ResNet18 then performs classification task on our reconstruction results, which can demonstrate the reconstruction validity. In addition, the pretrained ResNet18 is capable to do transfer learning to recognize the speckle patterns directly. Our proposal of reconstruction and recognition for speckle patterns is illustrated in Fig 6.

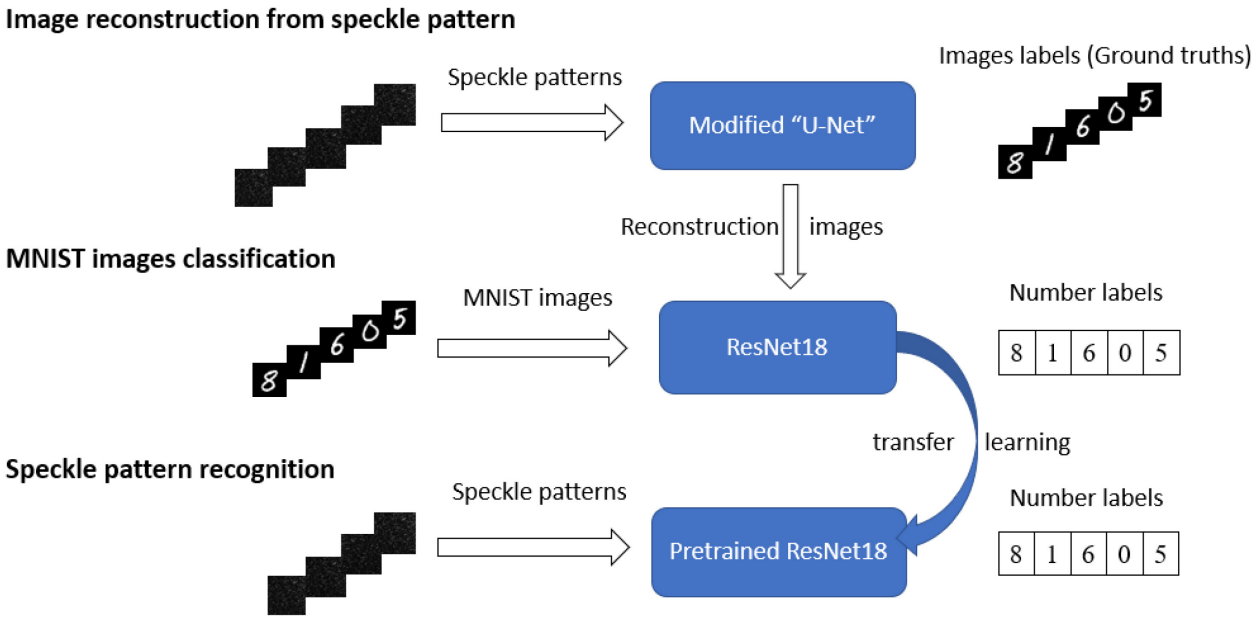


Fig 6. Proposal of reconstruction and recognition for speckle patterns with two types of networks employed.

4.1 Introduction of several network architectures

4.1.1 Encoder–decoder “U-Net” architecture

The U-Net[26] architecture is built upon the Fully Connected Network (FCN) and modified in order to yield more precise biomedical image segmentation. It was originally created by Olaf Ronneberger *et.al* in 2015 for the ISBI challenge, in which only few training images are given for segmentation of neuronal structures in electron microscopic stacks. The elegant yet powerful encoder–decoder “U” type network architecture and the training strategy relying on data augmentation have won the ISBI challenge for the authors in that year. U-Net has then gained much popularity in biomedical image segmentation, with many variants and applications in recent years.

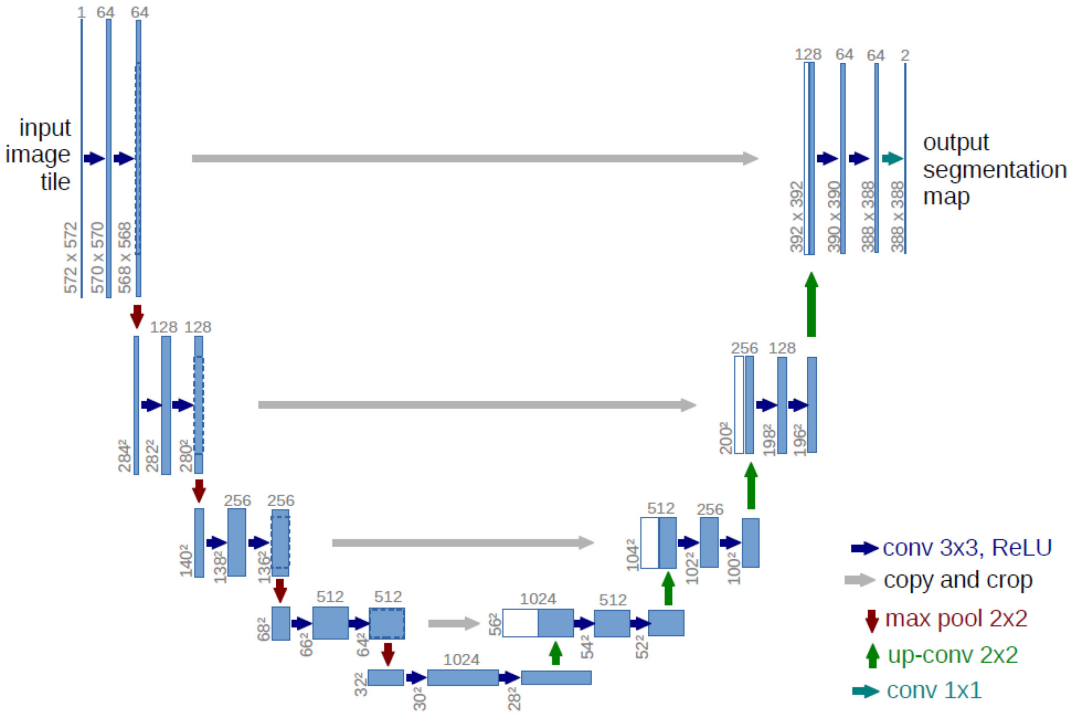


Fig 7. The diagram of U-Net architecture. Reproduced from Ref[26].

The network is composed of 3 parts: a contracting/down-sampling path, a bottleneck, and an expansive/up-sampling path, which makes it a U-shaped architecture (see Fig 7). Each layer of the contracting path is actually a convolutional network that comprises repeated applications of 3x3 convolution layer, followed by a rectified linear unit (ReLU) function and a 2x2 max pooling operation. The spatial information will be reduced while feature information will be enhanced during

the contraction. There are successive layers of up-sampling operators to the contracting network part. The expansive pathway is composed of repeated applications of 2x2 up-convolution and 3x3 convolution Layers (with ReLU). This up-sampling path combines both the feature and spatial information through a strand of up-convolutions and concatenations with high-resolution feature maps from the contracting path. Also, large amounts of feature channels are in the up-sampling part to propagate the context information to the follow-up layers. Finally, a successive convolution layer can then assemble a high-resolution output on the basis of the context information.

The two main differences when we compare U-Net to FCN-8 upon which the U-Net is built are: (1) the U-Net is symmetric; (2) the skip connections between the down-sampling path and the up-sampling path apply a concatenation operator between the channel instead of a sum along each feature channel.[28] These skip connections are intended to provide feature location information to the up-sampling path. The network enjoys large amounts of feature maps in the up-sampling path because of its symmetry, and this allows for more efficient information propagation. By contrast, the basic FCN architecture has only a quantity of classes of feature maps in the up-sampling path.

There are several dominant advantages of U-Net and its training strategy. First, the location information from the down-sampling path is combined with the context information in the up-sampling path to finally help obtain a general information for the prediction of a good segmentation map. In addition, there is no dense layer in U-Net, which makes it possible for the network input with images of different sizes. Last but not the least, the train strategy utilizing massive data augmentation undoubtedly facilitates the training, and this is of great significance in biomedical cases where the number of available annotated samples is usually limited.

4.1.2 Deep residual networks for image recognition

Deep residual network (ResNet)[27] has proven to be one of the most pioneering work in the communities of computer vision and deep learning. Since it became very successful for image classification in ImageNet competition of 2015, there have also been many different variants and interpretations of the residual network architecture in recent years. Apart from classification task, many other computer vision applications have also been boosted, like object detection, image segmentation and face recognition, due to the powerful representational ability of ResNets.

As there is a common trend to deepen our network architecture, simply stacking more layers together usually makes it hard to train on account of the tricky vanishing gradient problem. This

problem can be tackled by using a residual learning architecture that allows for extremely deep networks along with considerable accuracy. The key idea of ResNets is the introduction of "shortcut connection" that skips some layers to directly map the identity to the output, while these layers fit for residual mapping. This can be shown as the following building block (see Fig 8). It appears to be easier to optimize the residual mapping than to directly optimize the original underlaying mapping, as there are obviously less parameters to train. Then we can stack such identity mappings upon the current network. The resulting architecture proves not to produce a higher training error than its shallower counterparts, as the inserted layers don't do anything and directly maps the gradient to earlier layers during its backpropagation.

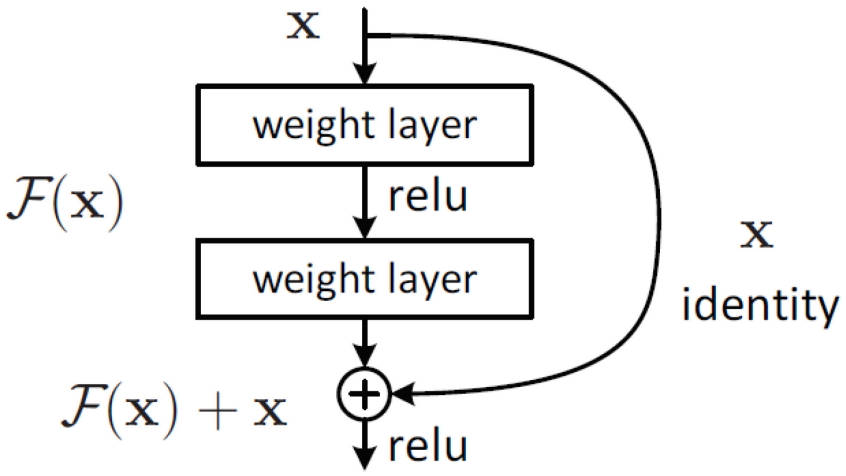


Fig 8. Residual learning unit: a building block, where x is the identity shortcuts and $\mathcal{F}(x)$ is the residual function. Reproduced from Ref[27].

What's more, there are two kinds of residual connections. On the case when the input x and the residual function output $\mathcal{F}(x)$ are of the same dimensions, the identity shortcuts x can be directly used. This is shown in building block, defined as:

$$y = \mathcal{F}\left(x, \{W_i\}\right) + x$$

Here, W_i is the layer weight parameter and function $\mathcal{F}\left(x, \{W_i\}\right)$ represents the residual mapping. The other case is when the input/output channels change, and there are two options: ①The shortcut still performs identity mapping, with extra zero-padding for the increased dimension. ②A projection shortcut W_s which is essentially done by 1*1 conv is used to match the dimension using the following formula:

$$y = \mathcal{F}\left(x, \{W_i\}\right) + W_s x$$

Now we can see the network architecture. There are two types of ResNet blocks: building block

for 2-layer deep which is used in small networks like ResNet 18, 34; and bottleneck for 3-layer deep (ResNet 50, 101, 152).

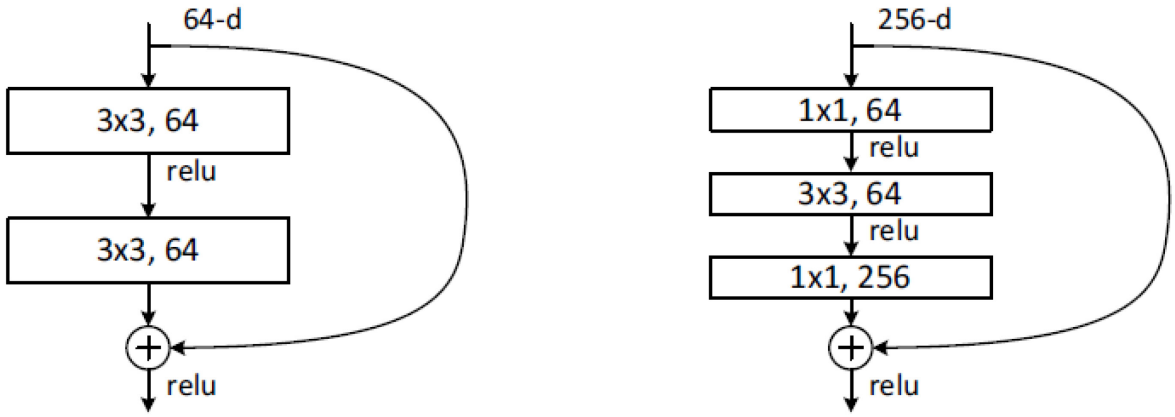


Fig 9. Two types of residual functions. Left: a building block for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152. Reproduced from Ref[27].

The ResNets design is based on VGG19 with the insertions of residual shortcut connections. There are three features about our residual networks. First, the 3*3 filters are used mostly. Next, the down-sampling is done using a convolutional layer with the stride of 2, during which the amounts of feature maps double while their size reduce half. Lastly, there is a successive global average pooling layer, followed by a 1000-class fully connected layer and softmax function in the end.

4.1.3 Densely connected convolutional networks (DenseNets)

As known, ResNets insert "residual shortcut connections" between layers to avoid vanishing gradient problem in the error back propagation, which makes it possible to train deeper CNNs for higher accuracy in generic function approximation. The ResNets connectivity is an element-wise addition between inputs and residuals for identity mapping. DenseNets[29], like ResNets, also create short paths from early layers to later layers, but in a dense connection way. In detail, each layer takes the feature maps of all preceding layers for channel-wise concatenation as the inputs, while its own feature maps are also the inputs to all subsequent layers. Such difference in shortcut connection between ResNets and DenseNets is illustrated in Fig. A compelling feature about DenseNets is that they encourage features reuse because of channel-wise feature concatenation. What's more, the number of parameters has been substantially reduced, while high performance to strengthen feature

propagation and training efficiency are achieved.

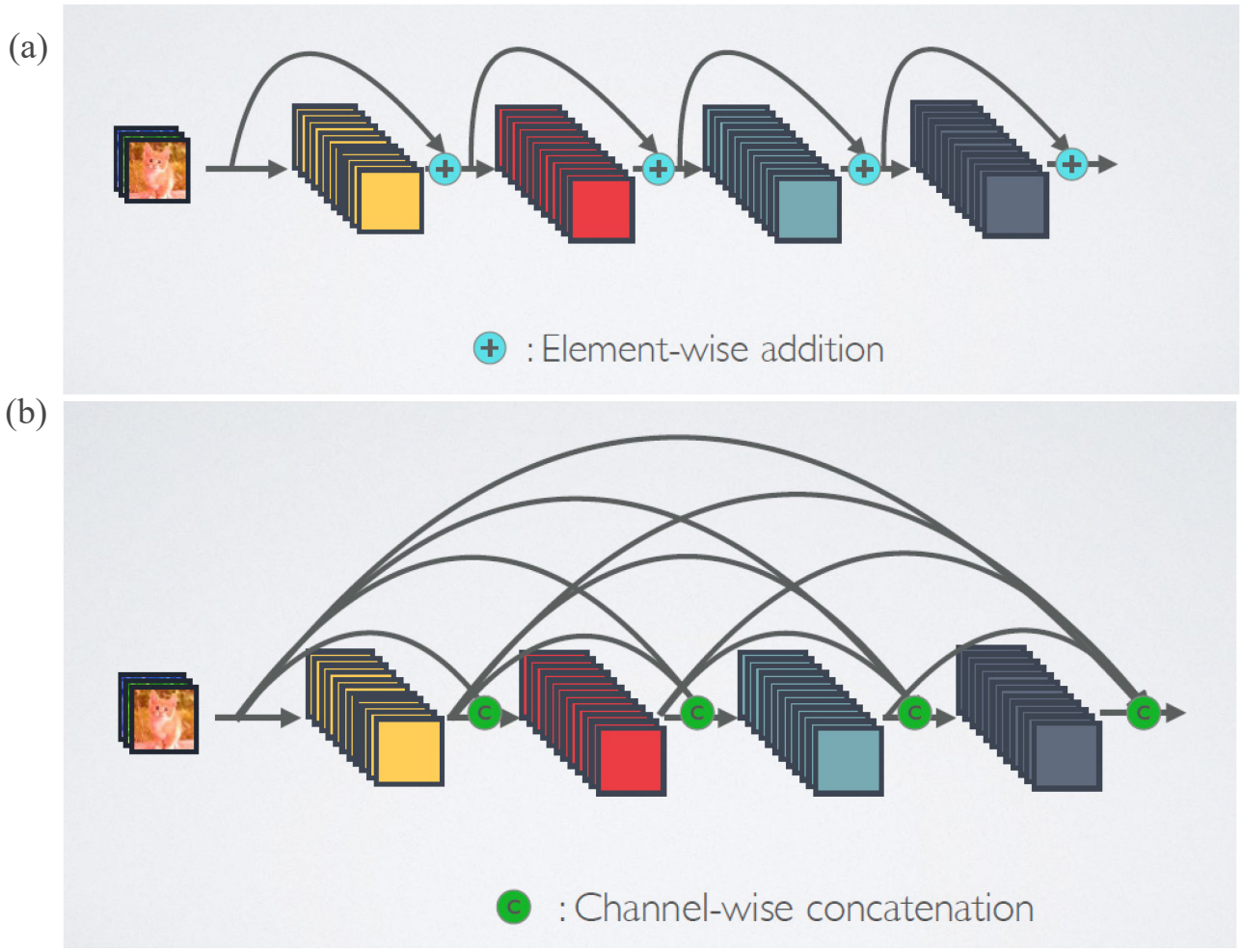


Fig 10. (a) ResNets connectivity. (b) DenseNets connectivity, also represents a dense block in DenseNets. Reproduced from Ref[29].

Now let's come to the DenseNet architecture, which is mainly composed of dense blocks and transition layers. Fig 11 shows a deep DenseNet with three dense blocks. A dense block is a module consisting of many layers whose feature maps are of the same size and the layers are connected in a dense way. For each dense block, the non-linear transformation function H_ℓ used to transform $(\ell-1)^{th}$ layer to ℓ^{th} layer within it is a composition of Batch Normalization (BN), ReLU and 3 x 3 Conv. Hence, the ℓ^{th} layer takes the concatenation of the feature-maps of all preceding layers as inputs:

$$x_\ell = H_\ell\left(\left[x_0, x_1, \ldots, x_{\ell-1}\right]\right)$$

There will be k feature maps produced by all function H_ℓ. Consequently, suppose the feature maps

of input layer is k_0, the input feature maps of ℓ^{th} layer will be $k_0 + k \times (\ell - 1)$. k here is referred as the growth rate of the network, and it can be relatively small so that Densets can have narrow layers while obtain high performance. To reduce the feature-map number of layers and model complexity in dense block, there will be bottleneck, that is, BN-ReLU-1×1 Conv is done before BN-ReLU-3×3 Conv.

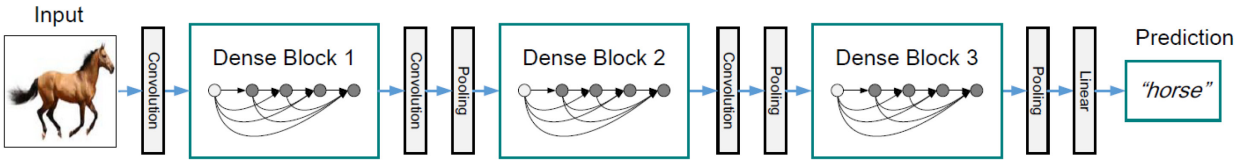


Fig 11. A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling. Reproduced from Ref[29].

With respect to transition layers, they are referred to those between dense blocks that do convolution and pooling. Transition layers are indispensable to facilitate down-sampling in DenseNets where the size of feature-maps may change among different dense blocks. A batch normalization and a 1×1 Conv followed by a 2×2 average pooling constitute the transition layer. In the end, a global average pooling layer and a softmax classifier are attached after the last dense block.

There are several advantages of DenseNtets. The first is its strong gradient flow. As the author pointed out, "Each layer has direct access to the gradients from the loss function and the original input signal, leading to an implicit deep supervision."[29] Hence, both the input signal and the error signal can be easily and more directly propagated. Next is its parameter and computational efficiency as a result of feature reuse and tricks like bottleneck, transition layers and small growth rate. Lastly, there are more diversified features in DesNets. As each layer in dense block receives feature-maps of all preceding layers as the inputs, such diversified features may result in richer output patterns.

4.2 Data Processing

4.2.1 Padding, up-sampling and image normalization

Just as illustrated in Chapter 2, we choose the database of MNIST handwritten digits based on which corresponding speckle patterns were collected. The resolution of original MNIST digit images are 28-by-28, different from the speckle patterns of size 256 x 256 pixels. To facilitate the applicability of different DNN models, we want to make the MNIST images and speckle patterns the same size. We first pad the edges of MNIST digit images with 0 values to the size of 32-by-32 and then up-sample the images to the size of 256 x256 by a factor of 8. In addition, image normalization is also indispensable for data running through network with many operations, which helps the network to converge (find the optimum) a lot faster. In our cases, we use nearest neighbor algorithm for the up-sampling of NNIST images. Both MNIST images and speckle patterns are normalized to (0,1).

There are two obvious benefits for the MNIST images and speckle patterns to have the same 256 x 256 pixels. For image reconstruction task, the modified U-Net is trained with pairs of speckle patterns and MNIST images. With a bigger output size like 256 x 256 for network training, the pixels of an input speckle pattern are processed with more kernel operations to make it quicker for approximation convergence. And for further recognition of speckle pattern, we first input MNIST images to the ResNet18 and then use the pretrained weighting parameters to initialize the DNN model that takes speckle patterns as inputs for further recognition. Thus the same size between MNIST images and speckle patterns makes it possible for transfer learning. Code for our MNIST dataset processing is provided in Appendix 3.

4.2.2 Loading dataset with Pytorch DataLoader

Pytorch framework has provided three classes in torch.utils.data.Dataset module (i.e. Dataset, DataLoader and DataLoaderlter), which are really hopeful to data loading and preprocessing. torch. utils.data.Dataset is an abstract class representing a dataset. We can inherit Dataset and override the following methods to obtain our custom dataset:

1) __len__ so that len(dataset) returns the size of the dataset.
2) __getitem__ to support the indexing such that dataset[i] can be used to get ith sample

While Dataset class represents the custom dataset and return s sample by __getitem__ function, torch. utils.data.DataLoader is an iterator which wraps Dataset class and provides the features like: batching the data, shuffling the data, and loading the data in parallel way. And for torch. utils.data. dataloader. DataLoaderIter, it is the framework that wraps Dataloader class and plays roles in the

iteration on Dataloader, as dataloader's __iter__() method returns a DataLoaderIter. We get the batch data by calling the __next__() method of DataLoaderIter. The code for loading our MNIST dataset for image reconstruction is provided in Appendix 4.

The data processing comprising preprocessing and loading data is illustrated as Fig 12.

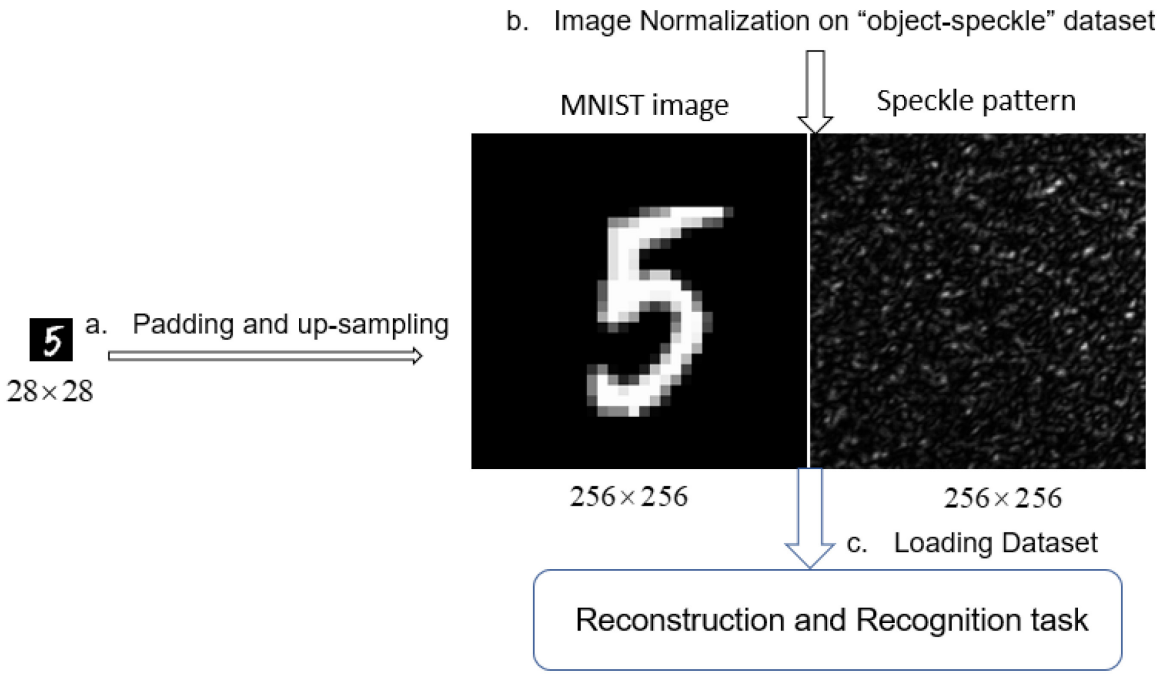


Fig 12. The diagram of data processing

4.3 Modified "U-Net" for image reconstruction from speckle pattern

4.3.1 Introduction

Here we're dealing with optical phase retrieval in the scenario of computational imaging through scattering media, and we can describe it mathematically[18]. Suppose $\psi(x,y;z=0)=\exp[i\phi(x,y)]$ represent the optical field that's phase-only modulated by an unknown target object $\phi(x,y)$ at the origin $z=0$ of the optical axis. The measurement is an intensity image $I(x,y)=\left|\Psi(x,y;z)\right|^2$ at the distance z known as speckle pattern. We use the forward operator H to relate the phase object at the origin to the speckle pattern $I(x,y)$ at the distance z as $I(x,y)=H\phi(x,y)$.[18] Phase retrieval problem is to formulate an inverse transform H^{inv} to obtain an acceptable estimate of the phase object $\hat{\phi}(x,y)$ like:

$$\hat{\phi}(x,y) = H^{inv} I(x,y)$$

We can solve the following inverse problem known as Tikhonov–Wiener optimization functional to obtain H^{inv}:

$$\hat{\phi}(x,y) = \arg\min_{\phi} \left\| H\phi - I \right\|^2 + \alpha\Phi(\phi)$$

where Φ is the regularizer that expresses *a prior* information about the object, and α is the regularization parameter.

To work out this ill-posed reconstruction functional, H and Φ must be known explicitly or parametrically. Although there are several approaches to determine these prior representations, the process is often complicated and prone to errors. Recent years have seen deep learning techniques in solving this inverse problem by learning these operators implicitly through the training of examples of targets imaged through scattering media. A DNN can be trained to learn H and Φ implicitly, and to act as the inverse operator H^{inv} to recover the target object from the recorded speckle. This learning approach requires no prior knowledge and is expected to be a more robust approximator to nonlinear operator. But a large training dataset of known image pairs of target objects and their corresponding speckle patterns must be available.

It has been shown by many researches[18-20] that a DNN model is able to discover the complex relationship between the inputs and outputs, through learning to map the input/output wavefronts of the scattering medium. In this part, we train a modified "U-Net" model to reconstruct the MNIST digit images containing phase information from the intensity measurements of the speckle patterns.

4.3.2 Implementation of modified "U-Net"

We have built a DNN model that follows the encoder-decoder "U-Net"[26] architecture with modification of replacing every two linked convolutional layers with a dense block[29] in which each layer is connected to the others within the same block in a feed-forward way. As there are more direct connections between the layers in DenseNets, the feature propagation is strengthened, also the feature reuse is encouraged. Hence, our modified U-net can have better generalization capability for image reconstruction, and its architecture is shown as Fig 13.

The input to the DNN is the normalized speckle pattern in greyscale and of the size 256 x 256. It first passes through the "encoder" path that is composed of five dense blocks which're connected by 2 x 2 max pooling layers for down-sampling. Every dense block is a module consisting of 4 composite layers (i.e. Conv2d-BN-ReLU-Dropout) with dense connections to each other. After the encoder, an

intermediate output gets small dimension (i.e. 16 x 16) while encoding rich information along its feature channel (i.e. 1024 channels). Then the low-resolution feature map passes through the "decoder" path that comprises another four dense blocks between which there are transpose convolutions for up-sampling. What's more, there are skip connections through the encoder-decoder paths to pass the high-frequency information learned in the initial layers toward the output reconstruction. The transpose convolution results and the skip connection results are concatenated along the channel for the inputs of corresponding dense blocks. Finally, after the decoder path, two successive convolutional layers follows the last dense block and produces the network output that is of the same size and channel as the input.

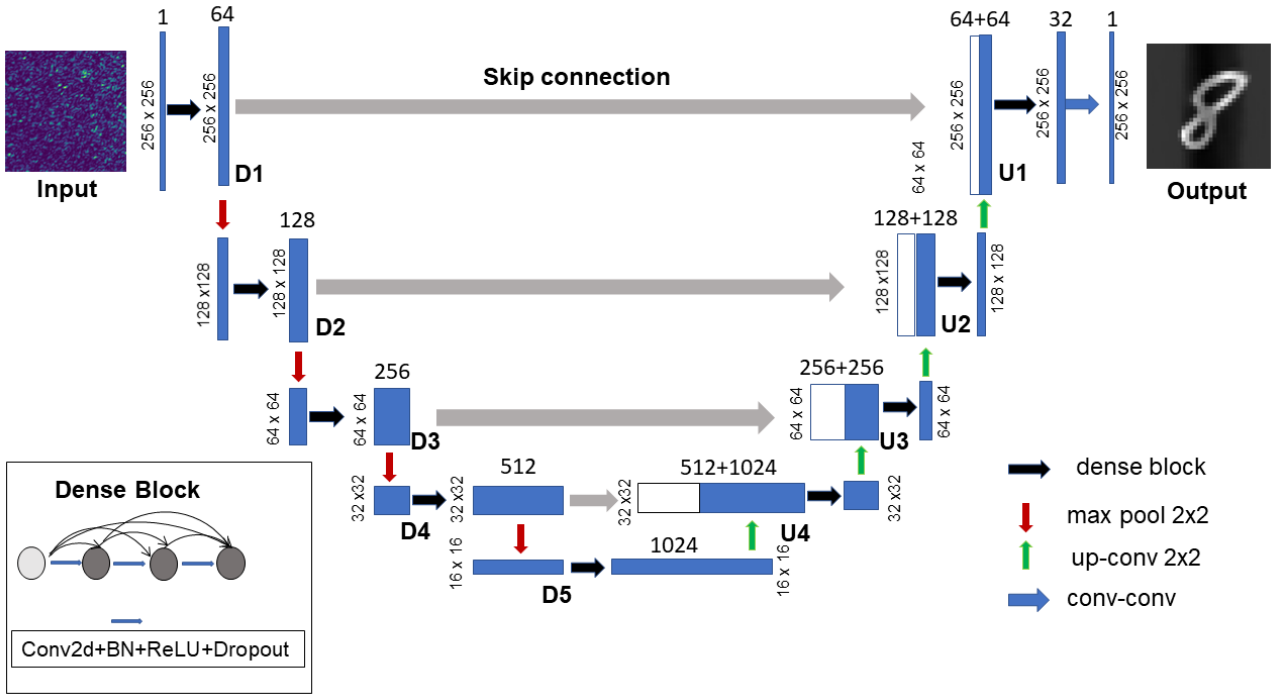


Fig 13. The architecture of our modified U-Net for MNIST image reconstruction

The modified U-Net was established using Pytorch framework. The python code to construct our DNN model, including the dense block and two type of layers (i.e. one consisting of dense block and down-sampling, the other one comprising transpose convolution and dense block) is provided in my Github.

4.3.3 Training and generalization test for reconstruction

Our DNN model training was performed on a lab PC with an Intel Xeon® CPU of 3.50GHz,

RAM 56GB and a1070Ti GPU, using Pytorch framework. 12000 pairs of preprocessed "object-speckle" data were randomly split to 10000 training set and 200 validation set. There is a validation every epoch to test the model on the untrained dataset and calculate the reconstruction loss. Once the validation loss is less than before, the model weight parameters will be saved. Our U-net was trained with 20 epochs using SGD optimizer with the initial learning rate set as 0.01 and decaying by 0.1 every 10 epochs. It generally takes 11hours for the training. After the training was done, 200 speckle patterns were then tested for reconstruction using the best model parameters saved during the training epochs. Although the network training is time consuming, the generalization test for a trained network can be very fast. The complete python code of the modified U-Net training and testing, as well as the well-pretrained weight parameters can be accessed in my Github, see https://github.com/Ford666.

4.3.4 Reconstruction results using different loss functions

Here, we have tried two different loss function to calculate the loss between the reconstruction image and the corresponding ground truth. The first is the widely used mean square error (MSE), defined as:

$$MSE=\frac{1}{M\cdot N}\sum_{i=1}^{M}\sum_{j=1}^{N}(R(i,j)-G(i,j))^2$$

Where $R(i,j)$ represents the value of (i,j) pixel in the reconstruction image and $G(i,j)$ the corresponding value in the ground truth. Recently a research[19] have demonstrated that the negative Pearson correlation coefficient (NPCC) is better for image reconstruction, especially in the scenario of imaging sparse objects. The NPCC is defined as:

$$NPCC=-\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(R(i,j)-\bar{R})(G(i,j)-\bar{G})}{\sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}(R(i,j)-\bar{R})^2}\sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}(G(i,j)-\bar{G})^2}}$$

Here, $\bar{R}$ and $\bar{G}$ are the mean value of reconstruction image and ground truth, respectively. There is nn.MSELoss() module while no ready-made module for calculating NPCC between input and target tensors and doing derivation to network weight parameters automatically in Pytorch. So we provide a NPCCloss module in Appendix 5.

Note that our DNN model is used to reconstruct MNIST images with many 0 values for black background, thus NPCC loss function may be promising to force the network output to approximate the ground truth as possible as it can. Here, from different point of view, we compare qualitatively

and quantitatively the reconstruction performance on test set when the network is trained using MSE and NPCC as the loss function, respectively.

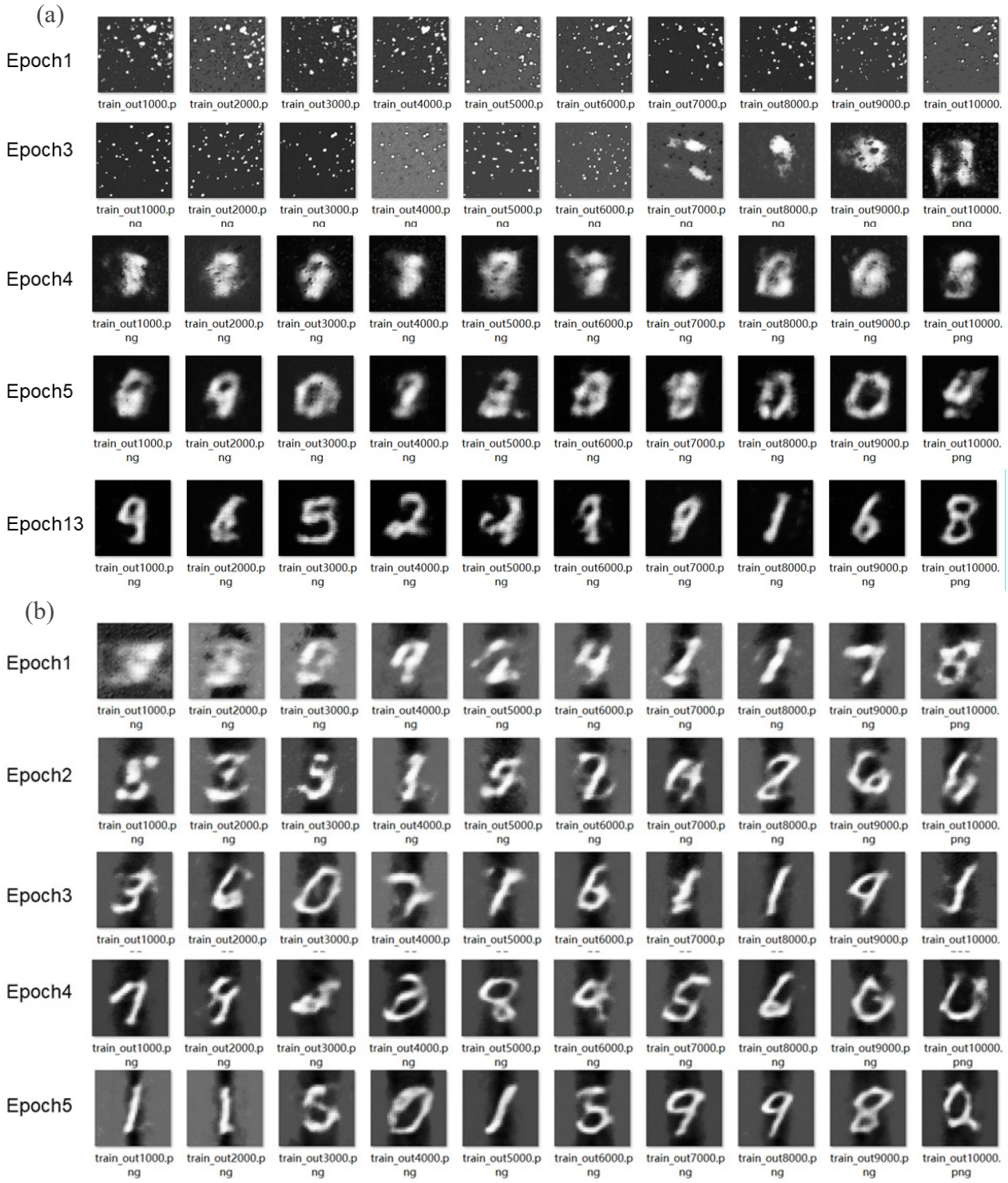


Fig 14. Visualization of network training process for MNIST image reconstruction from speckle pattern. Using (a) MSE and (b) NPCC as the loss function respectively, the network outputs every 1000 iterations during mini-batch training in different epochs.

What really make sense for the reconstruction performance of a trained network is to test it on an untrained dataset and to see whether it has obtained the generalization capability. Here we compare the reconstruction on test set using model weight parameters saved during different training epochs, when the network is trained using MSE and NPCC as the loss function respectively. The following Fig 15. shows two examples of reconstruction on MNIST digit “6” and “8” of test set. The results have reflected roughly common tendency of reconstruction performance in terms of different loss function.

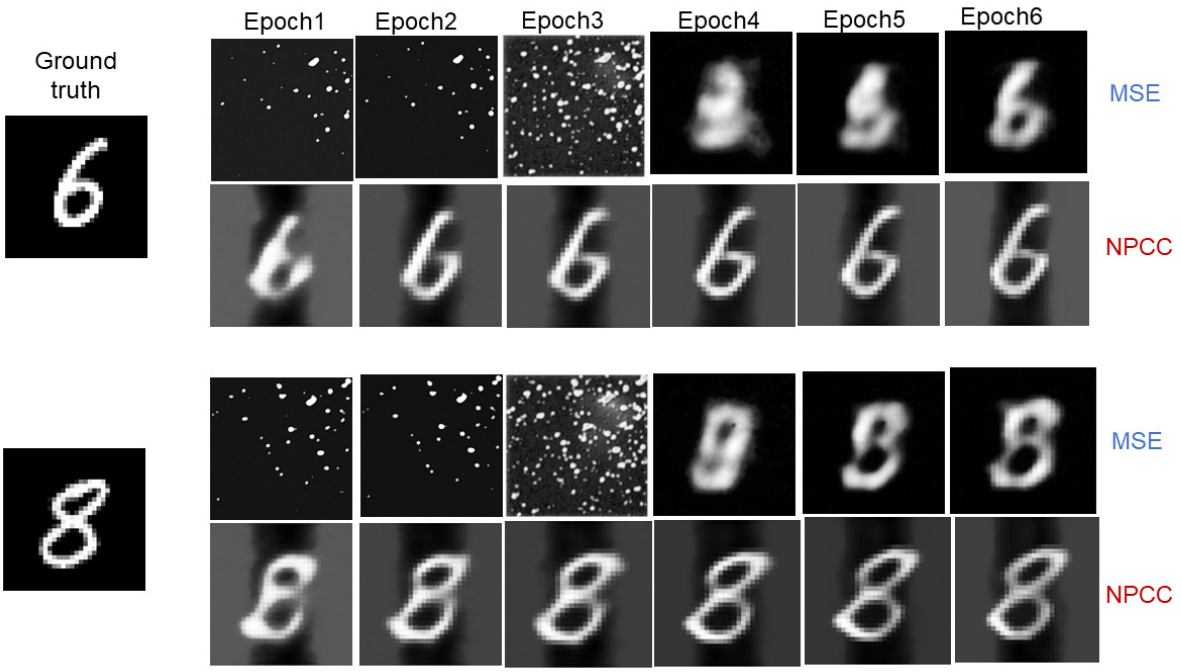


Fig 15. Reconstruction on MNIST digit “6” and “8” of test set using model weight parameters saved during different epochs when network trained with MSE and NPCC loss function respectively.

Using the weight parameter saved in 1st epoch of network trained with NPCC loss, we can visually recognize the reconstruction images. And for that of MSE, the least epoch for recognition is 5th. Also, for the former, the changing of reconstruction results is progressive in which a speckle pattern is transformed to its target object step by step. But for the latter one, the image quality improvement is not so obvious when inputted to a network initialized with weight parameters saved during different training epochs. The comparison of reconstruction results on test set show that NPCC-trained network is better than MSE-trained one in terms of subjective visual effect.

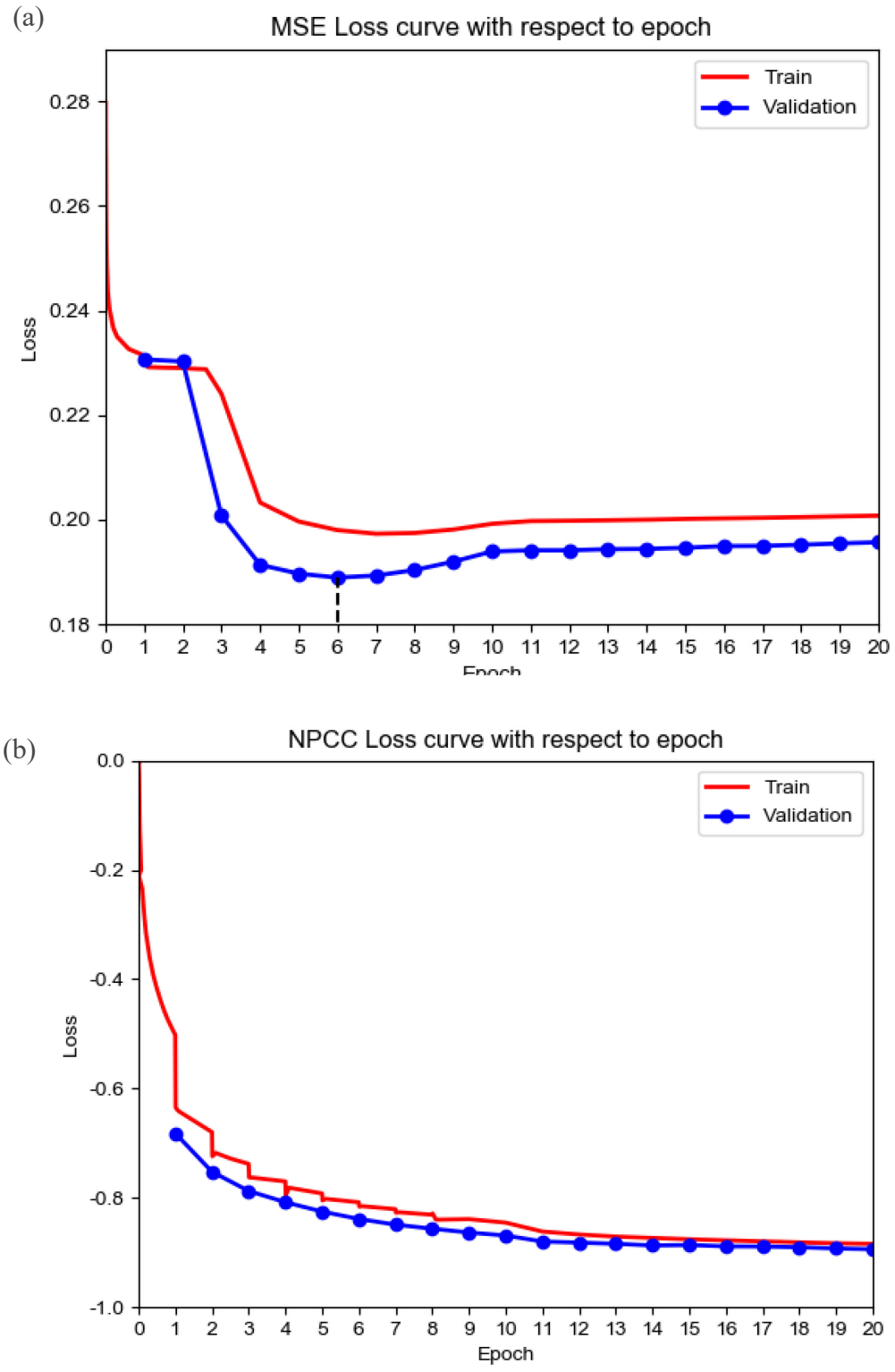


Fig 16. Loss curve with respect to epoch during training and validation, with network trained using (a) MSE and (b) NPCC as the loss function, respectively.

Let's have a look at the two loss curves during training and validation epochs. When MSE is used as the loss function, the loss reduces smoothly first and then roses slightly after falling to its minimum. By and large, the loss curve is not sharp, there is a short period from 1st to 3rd epochs when the loss keeps stable, and the loss reduction is not obvious. It is in 6th epoch that the MSE loss descend to the bottom and we use the weight parameters saved this time to initialize our U-net for reconstruction on test set. However, when it comes to the case when NPCC is the loss function, the loss drops down dramatically in the first several epochs and keeps descending in the subsequent epochs in a smooth way. What's especially interesting is that there is a sharp drop at the first few epoch nodes in which the loss curve appears nearly vertical. This is because the weight parameters fully update for all the iterations in last epoch, and the network with updated parameters performs better for its function approximation. We also load the weight parameters saved in 20th epoch to initialize our U-net for reconstruction on the test set. We make a quantitative comparison between the reconstruction results on test set using MSE-trained and NPCC-trained network, summarized in table 1.

Table 2. Comparison of reconstruction image quality of the test set when using network trained with MSE and NPCC loss function, respectively.

Loss function	visual recognition rate	MSE of test set	PCC of test set	Least epoch for recognition
MSE	122/200 (61%)	0.1971776	0.7035744	5
NPCC	200/200 (100%)	0.6607372	0.905956	1

Both subjective visual effect and objective standard evaluation were employed for the assessment of reconstruction results. The PCC between two positively linearly dependent variables is 1, and the MSE between two variables of identical values is 0. The quantitative analyses show that when tested with MSE-trained network, the reconstruction images got less mean square error and less Pearson correlation coefficient (PCC) compared with the ground truths. And to the case of using NPCC-trained network, it turns out just the opposite. The results have revealed that MSE is not so related to our subjective visual effect as NPCC does.

4.4 ResNet18 for MNIST classification and recognition of speckle pattern

4.4.1 Inheriting and adjusting ResNet class

Deep residual networks (ResNet) were originally designed for ImageNet competition, which was a color (3-channel) image classification task with 1000 classes. The MNIST images and the speckle patterns however are in the grayscale (1-channel) and only contain 10 classes. Here we take ResNet18 from torchvision library, to implement MNIST images classification first. The standard input image size for this network is 224 x 224 pixel, while our image size is 256 x 256 pixels. So there are a few adjustments to the existing network to fit our needs. The MyResNet class inheriting ResNet from torchvision library with adjustments is provided in Appendix 6.

In fact, not only have we made aforementioned adjustments while inheriting ResNet class from torchvision laibary, we also add some modules to the ResNet class, such as nn.Dropout2d layers and nn.LayerNorm layers, to avoid network overfitting and to quicken up the convergence.

4.4.2 ResNet18 training and evaluation for the MNIST database

The ResNet18 training loop with evaluation is similar to the case of image reconstruction which has been detailed in last section, while there are still some main differences.

1) For image classification task, apart from error between network prediction and image label, the accuracy of prediction is also widely used for classification assessment.
2) The loss function we use for image classification here is nn.Crossentropy() which proves to be fit for multi-classification task.
3) The setting of optimizer and the learning rate varies from cases to case. I've tried several optimizers, like Adadelta, Adam and SGD in MNIST image classification.

Here, I feel compelled to point out that it requires expertise and extensive trial and error to set the hyper parameters well. As a deep learning model is usually full of hyper-parameters, finding the best configuration for things like optimizer, learning rate and weight decay can never be trivial in such a high dimensional space. The result of our MNIST dataset classification task is failing: the training accuracy still remained a low level (around 0.10) after many iterations, let alone the generalization test! Although we've spent lots of time in trials, it still didn't work. Hence, we cannot obtain a pretrained ResNet18 fitting our MNIST image classification, further proposal for the recognition of speckle pattern has to be aborted.

Image classification, actually, is a far more difficult task compared to image reconstruction. As the network eventually outputs only one class label (i.e. the index of biggest value of the output probability distribution on all classes). Generally speaking, for MNIST classification, it usually takes tens of thousands of images and hundreds of epochs to train a network, with countless trial and error to set the hyper-parameters. In our cases, the original MNIST dataset has been preprocessed, including padding with 0 values and up-sampling using nearest neighbor algorithm, the training set and the training epochs are not large enough, the time left for recognition task is limited…. All of them may account for the failure of our MNIST dataset classification task.

4.4.3 Transfer learning for recognition of speckle pattern

Since we cannot obtain a pretrained ResNet18 fit for our MNIST dataset classification by ourself, further proposal to test on our reconstructed images and to do transfer learning for direct speckle pattern recognition becomes infeasible. However, Pytorch has provided pretrained ResNet model that's trained on ImgaeNet dataset, and this may be hopeful for our speckle pattern recognition.

For our speckle pattern dataset, it is of grayscale with one channel, the pattern size is 256 x256 pixels, and the expected classes are 10. However, for ResNet, the expected inputs are 3-channelk RGB images with the size of 224 as default. The final classes are 1000 for ImageNet DTASET. Hence, we have to make several changes to the architecture of the pretrained ResNet18.

1) The first layer of ResNet is a convolutional layer that takes the input of 3-channel, we propose to add a new layer that takes 1-chaenel input and outputs 3-channel image before the first convolutional layer.

2) Replace the last two layers. The layer second to last is average pool with the kernel size set as 7, and we propose to replace it with a new average pool layer with size 8 for kernel, since the input to this layer is of size 8 x 8 instead of 7 x 7. In addition, we replace the last fully-connected layer with a new one that predicts 10 classes.

Transfer learning uses the trained weight parameters to initialize an untrained model. The saved weight parameters are used to initialize only the layers in our own model same as those in the original model. And these initialized layers have to set fixed without updates during training. The other layers in our own model made by ourselves to fit our needs will not be initialized and to update parameters during training, this is what finetuning means. Code of transfer learning of pretrained ResNet18 from torchvision library is provided in Appendix 7.

Sad thing happened again. It turns out that simply transfer learning from the ResNet18 fit for ImageNet dataset classification cannot be trivially finetuned for our speckle pattern recognition. The final test accuracy after 20-epoch training is only around 0.14. Much efforts are still required for further network finetuning.

CHAPTER 5 Discussion and Conclusion

In this thesis, the research scenario is when given the intensity-only measurements of speckle patterns, how can we reconstruct the SLM target images incident on the scattering medium. We've tried two methods, that is, iterative phase retrieval in the context of imaging through scattering media and deep learning to map the SLM patterns to the corresponding speckle patterns. For the former, thanks to a publicly available dataset that contains empirical transmission matrix, GS algorithm with TM measurement operator is able to reconstruct SLM patterns with good results. Although iterative phase retrieval with TM employed has shown some advantages toward ordinary TM method to descramble speckle pattern, *a prior* TM of the scattering medium is still required. In contrast, deep learning requires no prior knowledge and is expected to be a more robust approximator to the inverse scattering operator, while a large training dataset must be available. We have used MSE and NPCC loss functions to train our modified U-Net, respectively, and the reconstruction results on test set are qualitatively and quantitatively analyzed and compared. With image reconstruction task done well, we were supposed to do further recognition for the speckle patterns with two technical routes proposed, while both didn't work within a short time. Trial and error are still wanted in future.

From my point of view, the method for deep learning to tackle imaging through scattering media is very simple, nothing more than "end-to-end" mapping through data-driven way, yet it is effective and even state-of-art. The training of network, however, is never trivial. Many things, including the setting of hyper-parameters, the choice of loss function, and data processing/augmentation, will make a difference to the result of generalization test.

For further developments in learning-based methods for computational imaging through scattering media, there are two guidelines here. The first is the network architecture we employ needs to model and to fit the physical scattering process in order to generate the desired target images after training. Also, learning from the field of Computer Vision to train and exploit image data for image enhancement, restoration and reconstruction, is very of great significance to this area.

REFERENCES

1. Ntziachristos, V., "Going deeper than microscopy: the optical imaging frontier in biology," Nature Methods. 7(8), 2010: 603-614.
2. Vinu, R.V., K. Kim, A.S. Somkuwar, Y. ParkR.K. Singh, "Imaging through scattering media using digital holography," Optics Communications. 439 2019: 218-223.
3. Zhang, Y.Z., G.H. Situ, G. Pedrini, D.Y. Wang, B. Javidi, and W. Osten, "Application of short-coherence lensless Fourier-transform digital holography in imaging through diffusive medium," Optics Communications. 286 2013: 56-59.
4. Vellekoop, I.M.A.P. Mosk, "Focusing coherent light through opaque strongly scattering media," Optics Letters. 32(16), 2007: 2309-2311.
5. Popoff, S.M., G. Lerosey, R. Carminati, M. Fink, A.C. Boccara, and S. Gigan, "Measuring the Transmission Matrix in Optics: An Approach to the Study and Control of Light Propagation in Disordered Media," Physical Review Letters. 104(10), 2010.
6. Yaqoob, Z., D. Psaltis, M.S. FeldC.H. Yang, "Optical phase conjugation for turbidity suppression in biological samples," Nature Photonics. 2(2), 2008: 110-115.
7. Bertolotti, J., E.G. van Putten, C. Blum, A. Lagendijk, W.L. Vos, and A.P. Mosk, "Non-invasive imaging through opaque scattering layers," Nature. 491(7423), 2012: 232-234.
8. Katz, O., P. Heidmann, M. FinkS. Gigan, "Non-invasive single-shot imaging through scattering layers and around corners via speckle correlations," Nature Photonics. 8(10), 2014: 784-790.
9. Popoff, S., G. Lerosey, M. Fink, A.C. BoccaraS. Gigan, "Image transmission through an opaque material," Nature Communications. 1 2010.
10. Conkey, D.B., A.M. Caravaca-AguirreR. Piestun, "High-speed scattering medium characterization with application to focusing light through turbid media," Optics Express. 20(2), 2012: 1733-1740.
11. Caravaca-Aguirre, A.M., E. Niv, D.B. ConkeyR. Piestun, "Real-time resilient focusing through a bending multimode fiber," Optics Express. 21(10), 2013: 12881-12887.
12. Choi, Y., C. Yoon, M. Kim, T.D. Yang, C. Fang-Yen, R.R. Dasari, K.J. Lee, and W. Choi, "Scanner-Free and Wide-Field Endoscopic Imaging by Using a Single Multimode Optical Fiber," Physical Review Letters. 109(20), 2012.
13. Jaganathan, K., Y.C. EldarB.H.J. Mathematics, "Phase Retrieval: An Overview of Recent Developments," 2015.
14. Bauschke, H.H., P.L. CombettesD.R. Luke, "Phase retrieval, error reduction algorithm, and Fienup variants: a view from convex optimization," Journal of the Optical Society of America a-Optics Image Science and Vision. 19(7), 2002: 1334-1345.
15. Marchesini, S., "Invited article: A unified evaluation of iterative projection algorithms for phase retrieval (vol 78, art no 011301, 2007)," Review of Scientific Instruments. 78(4), 2007.
16. Metzler, C.A., M.K. Sharma, S. Nagesh, R.G. Baraniuk, O. Cossairt, and A. Veeraraghavan, "Coherent Inverse Scattering via Transmission Matrices: Efficient Phase Retrieval Algorithms and a Public Dataset," 2017 Ieee International Conference on Computational Photography (Iccp 2017), 2017: 51-66.
17. Lyu. Meng, W.H., Li. Guowei, Situ. Guoha, "Exploit imaging through opaque wall via deep learning," arXiv:1708.07881, 2017.
18. Sinha, A., J. Lee, S. LiG. Barbastathis, "Lensless computational imaging through deep learning," Optica. 4(9), 2017: 1117-1125.

19. Li, S., M. Deng, J. Lee, A. SinhaG. Barbastathis, "Imaging through glass diffusers using densely connected convolutional networks," Optica. 5(7), 2018: 803-813.
20. Li, Y.Z., Y.J. XueL. Tian, "Deep speckle correlation: a deep learning approach toward scalable imaging through scattering media," Optica. 5(10), 2018: 1181-1190.
21. Shechtman, Y., Y.C. Eldar, O. Cohen, H.N. Chapman, J.W. Miao, and M. Segev, "Phase Retrieval with Application to Optical Imaging," Ieee Signal Processing Magazine. 32(3), 2015: 87-109.
22. Chandra, R., Z.Y. Zhong, J. Hontz, V. McCulloch, C. Studer, and T. Goldstein, "PhasePack: A Phase Retrieval Library," 2017 Fifty-First Asilomar Conference on Signals, Systems, and Computers, 2017: 1617-1621.
23. Situ, G.H., J.L. SuoQ.H. Dai, "Generalized Iterative Phase Retrieval Algorithms and Their Applications," Proceedings 2015 Ieee International Conference on Industrial Informatics (Indin), 2015: 713-720.
24. Fienup, J.R., "Phase Retrieval Algorithms - a Comparison," Applied Optics. 21(15), 1982: 2758-2769.
25. Marchesini, S., "A unified evaluation of iterative projection algorithms for phase retrieval," Review of Scientific Instruments. 78(1), 2007.
26. Ronneberger, O., P. FischerT. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," Medical Image Computing and Computer-Assisted Intervention, Pt Iii. 9351 2015: 234-241.
27. He, K.M., X.Y. Zhang, S.Q. RenJ. Sun, "Deep Residual Learning for Image Recognition," 2016 Ieee Conference on Computer Vision and Pattern Recognition (Cvpr), 2016: 770-778.
28. "DeepLearning 0.1 documentation." [Online]. Available: http://deeplearning.net/tutorial/unet.html.
29. Huang, G., Z. Liu, L. van der MaatenK.Q. Weinberger, "Densely Connected Convolutional Networks," 30th Ieee Conference on Computer Vision and Pattern Recognition (Cvpr 2017), 2017: 2261-2269.

致谢

川大四年的本科时光匆匆而过，一路磕磕碰碰地走来，一不留神就到了终点站。回首过往，虽然有很多遗憾不甘，更多的还是成长的收获与对师长同窗的感恩与留恋。

首先要感谢我们生物医学工程专业的几位老师。犹记得自己大二开始刚从物理系转来时，C++编程的基础掌握不多，课上课下便经常提问，邹远文老师给予我的耐心解答和亲切鼓励；犹记得自己曾翘课 Windows 程序设计甚至在数据结构的期末考卷上留言责怪老师出卷子不行，陈科老师对我的谅解与包容；犹记得医学图像课一次次编程大作业让我们心焦力瘁，不断看书编程调试后站在讲台上侃侃而谈时林江莉老师对我的赞许与期望。这些专业老师是我求学与人生路上的领路人，你们对我的严格培养和悉心教诲令人难忘。毕设期间，林老师和陈老师在各个环节都给了我们图像组的同学许多指导和帮助。陈老师更是我的校内导师，积极支持我去香港做毕设，并在线上为我答疑，在此向你们表示衷心的谢意！

然后我要感谢香港理工大学的赖溥祥老师，谢谢你愿意接收我做他的博士研究生并邀请我去生物医学光子学实验室完成毕业设计。在港理工一个多月的时间，每周一的组会督促我不敢懈怠，师兄们带我熟悉新的生活与科研环境。在此尤其要感谢李焕浩师兄，由于缺乏 GPU，你把自己的工作站腾出来给我跑深度学习，在网络搭建和训练过程中更是给了大量的反馈和指导，使得我能够顺利完成散斑图像的重建。还要感谢候旋迪和余志鹏师兄，初到港理工不大适应，谢谢你们一起开导我，还带我出去玩。做毕设的那段时光，紧张焦虑之余，也有轻松和喜悦，实在充实难忘！

其次感谢物理系和生医系的同学好友们，那些年我们一起在川大度过的青葱岁月，都将镌刻入我的美好青春记忆里。最后感谢我的父母对我的养育之恩，是你们一直对我的无私关爱和永远支持，我才能一步步地成长成熟；同时感谢母校川大对我的四年培养，让我有机会站在更高的平台上去开创精彩的人生旅程！

2018.05.18 于川大工学馆

APPENDICES

Appendix 1 MATLAB demo for iterative Phase retrieval

```
clear; close all; clc;

%To upload an object MNIST image on SLM
N=256; m=28;
object = imread('object.jpg');object = MatMap(object,0,1);
figure
subplot(121)
imshow(object,[]);title('Number 5 displayed on SLM');
% support domain constraint
sup = circle_mask(N,m,N/2,N/2); % generate a circle support domain
S = zeros(N,N);
S(N/2-m/2+1:N/2+m/2,N/2-m/2+1:N/2+m/2) = object;
S1 = S.*sup;
subplot(122)
imshow(S1,[]);title('object restricted by circular support domain');

% Use the 2D-FT of the object as the diffraction pattern
S2 = abs(fftshift(fft2(S1)));
figure
subplot(121)
imshow(log(1+S2),[]);title('The amplitude of diffraction pattern');
% Gerchberg-Saxton-Fienup-type algorithm
iter_num = 500;
g = rand(N,N); % Phase-only modulated optical field

for i = 1:iter_num
    %================Original GS==================
%       g = projectM(g,S2);
    %================Error Reduction==================
```

```
%     g = projectM(g.*sup,S2);  %support domain constraint

    %===============Hybrid Input-Output=============
%     g1 = projectM(g,S2);
%     g1 = g1.*sup; %support domain constraint
%     g = (g1>=0).*g1 + (g1<0).*(g-0.7.*g1); %positivity constraint

    %================Difference Map=================
    g = g + projectM(2*g.*sup-g,S2) - g.*sup;

    %======display the retrieved object=======
    % the central part of a square
    subplot(122)
    imshow(rot90(g(N/2-m/2+1:N/2+m/2,N/2-m/2+1:N/2+m/2),2),'InitialMagnification',1000);
%     imshow(abs(g),'InitialMagnification',200);
    title(sprintf('Phase retrieved in %dth iteration',i))
    pause(0.001);
end
```

Appendix 2 Iterative phase retrieval using TM measurement operator

```
clear
close all
clc

%% load an object-speckle image dataset and a empirical measurement matrix
load 'YH_squared_test.mat'
load 'XH_test.mat'
load 'A_GS.mat'
m = 256^2;
n = 40^2;
```

```
X_object = double(XH_test)'/255;
Y_TrueSpeckle = double(YH_squared_test)'; %speckle intensity pattern
Y_TrueSpeckle = Y_TrueSpeckle.^.5;

figure
for i=1:size(X_object,2)
    subplot(3,5,i);
    imshow(reshape(X_object(:,i),sqrt(n),sqrt(n)),[]);
end
for i=1:size(Y_TrueSpeckle,2)
    subplot(3,5,5+i);
    imshow(reshape(Y_TrueSpeckle(:,i),sqrt(m),sqrt(m)),[]);
end
%% reconstruct SLM patterns from synthetic speckle patterns using GS algorithm.

%Throw out the ill-conditioned lines of TM
good_inds = find(residual_vector<.4);
A = A(good_inds,:);
Y_TrueSpeckle = Y_TrueSpeckle(good_inds,:);

A_dag = pinv(A); %The Moore-Penrose pseudoinverse

GS_iters = 100;
disp('started computations')
t0=tic;
X_recon = A_dag*(Y_TrueSpeckle.*exp(2*pi*rand(size(Y_TrueSpeckle))));
for i = 1:GS_iters
    z=Y_TrueSpeckle.*exp(1i*angle(A*X_recon));
    X_recon=A_dag*z;
    % display reconstruction results
    for j=1:size(X_recon,2)
        subplot(3,5,10+j);
```

```
        imshow(reshape(real(X_recon(:,j)),sqrt(n),sqrt(n)),[]);
        pause(0.001);
    end
end
suptitle('SLM phase patterns, speckle patterns & phase retrieved after 100 iterations')
t1=toc(t0);
fprintf('It takes %4.2f seconds\n',t1);
```

Appendix 3 Data preprocessing

```
#MNIST image(19800,28,28) => Padding to (19800,32,32)
train_image = np.pad(train_image,((0,0),(2,2),(2,2)),'constant',constant_values = (0,0))

#Normalize the image/speckle values into interval[0,1]
image_max = (np.amax(np.amax(train_image,axis=1),axis=1)).reshape(19800,1,1)
image_min= (np.amin(np.amin(train_image,axis=1),axis=1)).reshape(19800,1,1)
train_image= (train_image-image_min)/(image_max-image_min)
speckle_max = (np.amax(np.amax(train_speckle,axis=1),axis=1)).reshape(19800,1,1)
speckle_min= (np.amin(np.amin(train_speckle,axis=1),axis=1)).reshape(19800,1,1)
train_speckle= (train_speckle-speckle_min)/(speckle_max-speckle_min)
#MNIST image upsampling during mini-batch training
for i, (inputs, labels) in enumerate(train_loader):
    optimizer.zero_grad()
    inputs, labels = inputs.to(device), labels.to(device)
    outputs = model(inputs) #(batch_size,1,256,256)
    labels = labels.unsqueeze(1) #(batch_size,32,32) => (batch_size,1,32,32)
    labels=nn.UpsamplingNearest2d(scale_factor=8)(labels) #(batch_size,1,256,256)
    //....
```

Appendix 4 Loading the data for MNIST image reconstruction

```
from torch.utils.data import Dataset, DataLoader
class MNIST_data(Dataset):
    # Dataset wrapping X_data and y_data(tensors)
    def __init__(self, X_data, y_data):    #initialization
        self.X_data = X_data
        self.y_data = y_data
        self.len = X_data.shape[0]
    def __getitem__(self, index): #support the indexing
        return self.X_data[index], self.y_data[index]
    def __len__(self): #return the data length
        return self.len


  # instantiate MNIST_data class
    MNIST_train = MNIST_data(X_train, y_train)
    MNIST_val = MNIST_data(X_val, y_val)
    MNIST_test = MNIST_data(X_test, y_test)
    # Dataloader begins
    batch_size = 1
    train_loader = DataLoader(dataset=MNIST_train, batch_size=batch_size,
                                        shuffle=True, num_workers=2)
    val_loader = DataLoader(dataset=MNIST_val, batch_size=batch_size,
                                        shuffle=True, num_workers=2)
    test_batch_size = 1
    test_loader = DataLoader(dataset=MNIST_test, batch_size=test_batch_size,
                                        shuffle=True, num_workers=2)
   #Get the batch data by iterations on DataLoader
    for i, (inputs,labels) in enumerate(train_loader):
        inputs,labels = Variable(inputs), Variable(labels)
       //….
```

Appendix 5 NPCCloss module

```
class NPCCloss(nn.Module):
    def __init__(self):
        super(NPCCloss, self).__init__()
    def forward(self, x, y):
        x = x.view(256**2,-1)
        y = y.view( 256**2,-1)
        x_mean = torch.mean(x, dim=0, keepdim=True)
        y_mean = torch.mean(y, dim=0, keepdim=True)
        vx = (x-x_mean)
        vy = (y-y_mean)
        c = torch.mean(vx*vy, dim=0)/(torch.sqrt(torch.mean(vx**2, dim=0)+1e-08) * torch.sqrt(torch.mean(vy
** 2,dim=0)+1e-08))
        output = torch.mean(-c)
        return output
```

Appendix 6 Our ResNet class to inherit ResNet from torchvision library

```
from torchvision.models.resnet import ResNet
class MyResNet(ResNet): #ResNet18
    def __init__(self):
        super(MyResNet, self).__init__(BasicBlock, [2, 2, 2, 2], in_channels=1, num_classes=10)
        self.avgpool = nn.AvgPool2d(8, stride=1) #fit the input size(N,1,256,256)

    def forward(self, x):
        return super(MyResNet, self).forward(x)
```

Appendix 7 Transfer learning of pretrained ResNet18 from torchvision library.

```
    from torchvision import models

    # Pretrained ResNet18 for transfer learning
```

```
class Net(nn.Module):
    def __init__(self, model):
        super (Net,self).__init__()
        self.up_channel = nn.Conv2d(1, 3, kernel_size=3, padding=1)
        self.resnet_layer = nn.Sequential(*list(model.children())[:-2])
        self.pool_layer = nn.AvgPool2d(kernel_size=8, stride=1) #fit the input size(N,1,256,256)
        self.fc = nn.Linear(in_features=512, out_features=10)
    def forward(self,x):
        x = self.pool_layer(self.resnet_layer(self.up_channel(x)))
        x = x.view(x.size(0),-1)
        x = self.fc(x)
        return x
resnet = models.resnet18(pretrained=True)
model = Net(resnet)
model = model.float().cuda()

for para in list(model.parameters())[:-1]:
    para.requires_grad=False
for param in model.up_channel.parameters():
    param.requires_grad = True
for param in model.fc.parameters():
        param.requires_grad = True
optimizer = torch.optim.SGD(params=[model.up_channel.weight, model.up_channel.bias,
                                    model.fc.weight,model.fc.bias],lr=1e-3, momentum=0.9)
//…
```