Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Magnus Hovde Eriksen

Fourier ptychographic reconstruction using neural networks trained on stock images

Master's thesis in MTFYMA
Supervisor: Dag Werner Breiby
June 2023

NTNU
Norwegian University of
Science and Technology

Magnus Hovde Eriksen

Fourier ptychographic reconstruction using neural networks trained on stock images

Master's thesis in MTFYMA
Supervisor: Dag Werner Breiby
June 2023

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

NTNU
Norwegian University of
Science and Technology

ABSTRACT

Fourier ptychography is a computational imaging technique for capturing images at high-resolution while maintaining a wide field of view. The technique consists of gathering information about the sample's Fourier spectrum by illuminating the sample from different incidence angles. Each incidence angle then corresponds to a low-resolution intensity image with information about a particular area in the sample's high-resolution spectrum. The set of captured low-resolution intensity images can then be used to synthesize the sample's high-resolution complex amplitude. Since the complex amplitude is recovered the phase image of the sample will also be recovered.

The reconstruction procedure is conventionally done with an iterative method, but recently alternative deep learning-based approaches with neural networks have been explored. This project explores training neural networks to do Fourier ptychographic reconstruction, but with the twist that all training data used to optimize the parameters of the neural network is created by simulation. The simulation procedure emulates Fourier ptychography imaging of a complex object where the values used for the amplitude and phase arrays are chosen to be grayscale images drawn from a dataset of stock images.
Two different neural network architectures were explored.

Reconstruction results on simulated data created from a validation dataset showed good results, recovering the high-resolution object with high accuracy. The performance of the neural networks was also tested on real data by reconstructing an FP image set of a bone and cartilage sample. The neural networks produced images with increased resolution, but not the resolution gain expected from theory. A comparison with the high-resolution image reconstructed by the conventional iterative method showed that the neural networks did not produce images of comparable quality, with the high-resolution images reconstructed by the iterative method showing a larger amount of detail which suggests a higher resolution than the reconstructed neural network solutions.

SAMMENDRAG

Fourier ptykografi er en beregningsbasert avbildningsteknikk for å oppnå høy oppløsning og samtidig et bredt synsfelt (FoV). Teknikken består av hente informasjon om prøvens Fourier spektrum ved å belyse prøven fra forskjellige innfallsvinkler. Hver innfallsvinkel produserer et lavoppløst intensitetsbilde som innneholder informasjon om et bestemt område i prøvens Fourier spektrum. Settet av lavoppløste intensitetsbilder kan deretter brukes til å syntetisere et høyoppløst bilde av prøvens komplekse amplitude ved bruk av en rekonstruksjonsmetode. Siden den komplekse amplituden rekonstrueres vil også prøvens fasebilde gjenfinnes.

Rekonstruksjonsmetodene som vanligvis benyttes er iterative metoder basert på overlappende informasjon i Fourier rommet, men alternative dyplærings baserte metoder med nevrale nettverk har også blitt utforsket. Dette prosjektet utforsker bruken av nevrale nettverk for fourier ptykografis rekonstruksjon, men med vrien at de nevrale nettverkene trenes utelukkende på et datasett konstruert med simulering. Det kunstige datasettet til bruk for trening av de nevrale nettverkene ble konstuert ved å simulere fourier ptykografisk avbildning av et komplekst objekt, der verdiene benyttet for amplitude og fase til det komplekse objektet var gråskalabilder hentet fra et utvalgt datasett.
To forskjellige nevrale nettverksarkitekturer ble utprøvd.

Rekonstruksjonsresultater på simulerte data, der det avbildede komplekse objektet var kontruert med data hentet fra et valideringsdatasett, viste gode resultater og lyktes i å rekonstruere det komplekse objektet med høy nøyaktighet. De nevrale nettverkene ble også testet på virkelig FP data ved å rekonstruere en FP avbildning av en ben- og bruskprøve. NN metoden genererte rekonstruerte bilder med økt oppløsning, men ikke den oppløsningsøkningen som er forventet av teorien. En sammenligning med høyoppløstbildet rekonstruert av den konvensjonelle iterative metoden viste at de nevrale nettverkene ikke genererte bilder av sammenlignbar kvalitet. Rekonstruert høyoppløstbilde med den iterative metoden viste høyere oppløsning enn høyoppløstbildet generert av den NN baserte metoden som indikerer at NN metoden ikke rekonstruerer Fourier spekteret til det høyoppløste objektet tilstrekkelig.

PREFACE

This thesis was written in fulfilment with the final part of the MTFYMA study programme.
I would like to thank Prof. Dag Werner Breiby for being my supervisor for this thesis project and the specialization project the preceding autumn semester. I appreciate it!

Magnus Hovde Eriksen
Trondheim, 8.June, 2023

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

List of all abbreviations in alphabetic order:

- **CNN** Convolutional Neural Network
- **FP** Fourier Ptychography
- **GPU** Graphics Processing Unit
- **HR** High Resolution
- **LR** Low Resolution
- **MAE** Mean Absolute Error
- **MSE** Mean Square Error
- **NA** Numerical Aperture
- **NN** Neural Network
- **OA** Optical Axis
- **PSNR** Peak Signal to Noise
- **SBP** Space-Bandwidth Product
- **SGD** Stochastic Gradient Descent
- **SSIM** Structural Similarity Index Measure

CHAPTER

ONE

INTRODUCTION

Fourier ptychography [ZHY13] is an computational imaging technique for achieving high-resolution while maintaining a large FoV, resulting in a larger SBP for the system than would be possible conventionally. It is simultaneously also a phase retrieval technique. The FP imaging procedure consist of illuminating the object from a set of different angles, resulting in a set of intensity images with each image corresponding to a certain illumination direction. Each illumination angle collects information about a certain area of the Fourier domain of the imaged object and is stored in the corresponding intensity image. The captured images containing information about the Fourier spectrum of the object are then used to reconstruct a high-resolution complex image of the object.
The reconstruction procedure takes in the captured images as input and outputs the reconstructed high-resolution complex object. The most common of these reconstruction procedures are iterative in nature can consists of iterative updates to the Fourier domain of the high-resolution object until it is consistent with information contained in the collected images.

Deep learning based methods have been explored as an alternative to the iterative methods for performing Fourier ptychographic reconstruction ([Ngu+18],[Jia+18] ,[Zha+19],[Lu+21]). Deep learning methods differs from the iterative methods in that they are optimised once prior use and then reconstructs the high-resolution object in a few forward passes. If one could train netural networks to produce reconstruction results of similar quality to the iterative methods then aided with modern hardware like GPUs one could hope to achieved faster reconstruction that are just as accurate as the iterative method.

1.1 Aim of work

This project aims to train neural networks for the purpose of Fourier ptychography reconstruction without the need for the gathering of training data beforehand. This is archived by simulating Fourier ptychographic imaging of the complex object $O(x, y) = A(x, y) \cdot \exp(i \cdot \phi(x, y))$ computationally. The arrays $A(x, y)$ and $\phi(x, y)$ are chosen to be stock images from selected datasets.
The hope with this approach is that the resulting optimized neural networks have

learned the reconstruction function well enough that they generalize to real FP image data captured in the lab. The advantage of this approach is that it eliminates the time consuming process of collecting training data for use in network optimisation.

CHAPTER

TWO

THEORY

2.1 Imaging theory

2.1.1 Linear systems and transfer functions

Linear systems is used to describe a variety of different physical systems, including optics.
Optical systems very commonly use linear system theory in describing and reasoning about the systems properties, which warrants a review of the key concepts. The following discussion follows the one given in [Goo96, pp. 19–22].

The condition for an operator $\mathcal{S}$ to be linear is

$$\mathcal{S}\left[(af(x,y)+bg(x,y)\right] = a\mathcal{S}\left[f(x,y)\right] + b\mathcal{S}\left[g(x,y)\right]. \tag{2.1}$$

Using delta-functions one can write a function $g(x,y)$ as

$$g(x,y) = \iint_{-\infty}^{+\infty} g(x',y')\delta(x-x',y-y')\,dx'dy'. \tag{2.2}$$

Using the δ-function trick from equation (2.2), the effect of the linear operator $\mathcal{S}$ on $g(x,y)$ can now be described by

$$\begin{aligned}
\mathcal{S}\{g(x,y)\} &= \mathcal{S}\left\{\iint_{-\infty}^{+\infty} g(x',y')\delta(x-x',y-y')\,dx'dy'\right\} \\
&= \iint_{-\infty}^{+\infty} \mathcal{S}\left\{g(x',y')\delta(x-x',y-y')\right\}\,dx'dy' \quad \text{(by linearity)} \\
&= \iint_{-\infty}^{+\infty} g(x',y')\mathcal{S}\left\{\delta(x-x',y-y')\right\}\,dx'dy' \quad \text{(since } \mathcal{S} \text{ operates on } x,y) \\
&= \iint_{-\infty}^{+\infty} g(x',y')h(x-x',y-y')\,dx'dy' = h(x,y) \circledast g(x,y).
\end{aligned} \tag{2.3}$$

where $h(x, y)$ is defined by $h(x, y) = \mathcal{S}\{\delta(x - x', y - y')\}$.
This means that the effect a linear operator $\mathcal{S}$ has on a function $g(x, y)$ can be described by a convolution with kernel h as defined in equation (2.3).
We further note that

$$\mathcal{F}\{g_2(x, y)\} = G_2(\nu_x, \nu_y) = \mathcal{F}\{h(x, y) \circledast g(x, y)\} = H(\nu_x, \nu_y)G_1(\nu_x, \nu_y). \quad (2.4)$$

We see that we can describe the effect of the operator $\mathcal{S}$ on the system by a multiplication by function $H(\nu_x, \nu_y)$ in the frequency-domain.
$H(\nu_x, \nu_y)$ is know as the *transfer function* of the system.

Figure 2.1.1: Transfer function as a black box.

2.1.2 Coherence

Coherence is the correlation between the phases of EM signals. Two signals that are coherent will have constant (or near constant) relative phase and thus will oscillate in union with each other.
Coherent leads to a number of phenomena, notably the production of fringe patterns resulting from interfering signals.

2.1.2.1 Temporal coherence

Temporal coherence is the measure of the phase correlation of a signal $U(t)$ with the signal at a later time $U(t+\tau)$. If we express the signal as $U(t) = U_0 e^{i\phi} e^{i\omega_0 t}$ one sees that for a monochrome signal, the phase difference between points $(t, t + \tau)$ on the sinusoidal signal in figure (2.1.2) is constant and equal to $\omega_0\tau$.
In practice there are no perfect monochrome signals, which means for large τ, the signal at points $(t, t + \tau)$ will not be coherent, i.e no constant phase difference.

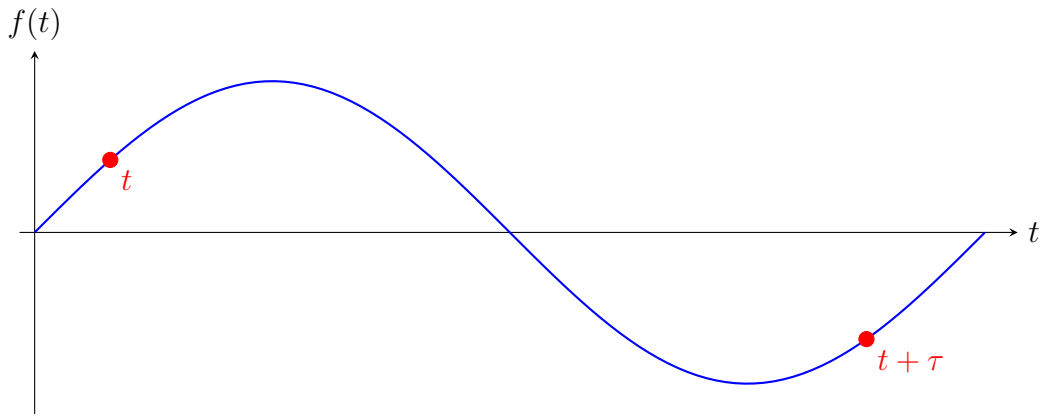


Figure 2.1.2: Two points along the signal

A fundamental reason for the fact that no monochrome signal exist is the signal emitted by a source have a finite duration, which inevitably makes a monochrome signal impossible.

Following the example in [PPP18, pp. 230–231], it is easy to show that for a finite wave-train of the type illustrated in figure (2.1.3)

$$f(t) = \begin{cases} e^{-\omega_0 t}, & -\frac{\tau_0}{2} < t < \frac{\tau_0}{2} \\ 0, & \text{elsewhere} \end{cases} \tag{2.5}$$

the fourier transform $F(\omega)$ of the is given by

$$F(\omega) = \frac{\tau_0}{2\pi} \left[\frac{\sin[(\tau_0/2)(\omega - \omega_0)]}{(\tau_0/2)(\omega - \omega_0)} \right] \tag{2.6}$$

with power spectrum as illustrated which can be seen to be concentrated at ω_0, but still containing a continuum of frequencies.

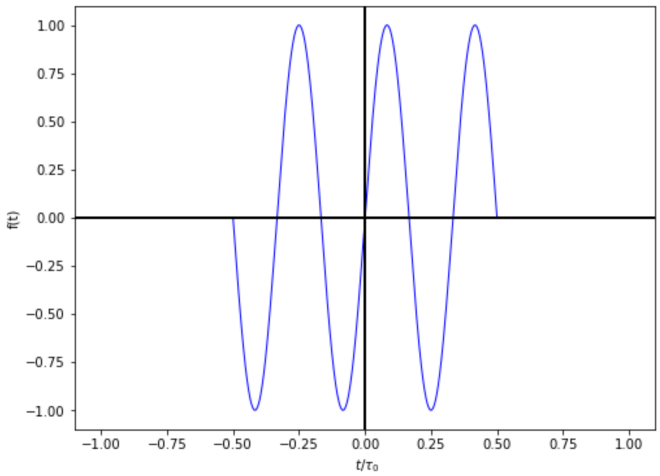


Figure 2.1.3: Finite wave train

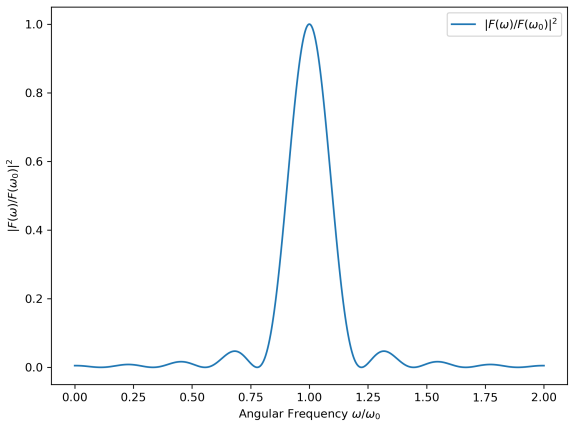


Figure 2.1.4: Power spectrum for the finite wave train.

The fact that no perfect monochrome signal exists due to the finiteness of the

wave-trains emitted by sources justifies the concept of a coherence time τ_0 for which points $(t, t+\tau)$ with $\tau \leq \tau_0$ the signal can be considered coherent. The corresponding coherence length l_t is defined $l_t = c\tau_0$.

2.1.2.2 Spatial coherence

Spatial coherence is the correlation in phase between laterally separated points on the wavefront. This is in contrast to temporal coherence, which measures the degree of correlation down the propagation path.
Figure (2.1.5) shows a double-slit setup. If the light reaching the slits are coherent the light emanating from the slits will produce interference fringes. It can be shown ([PPP18, pp. 237–239]) that an approximate condition for the separation between the slits l_s such that a interference pattern is produced on the screen due to an source of dimension s is given by

$$l_s < \frac{r\lambda}{s}. \tag{2.7}$$

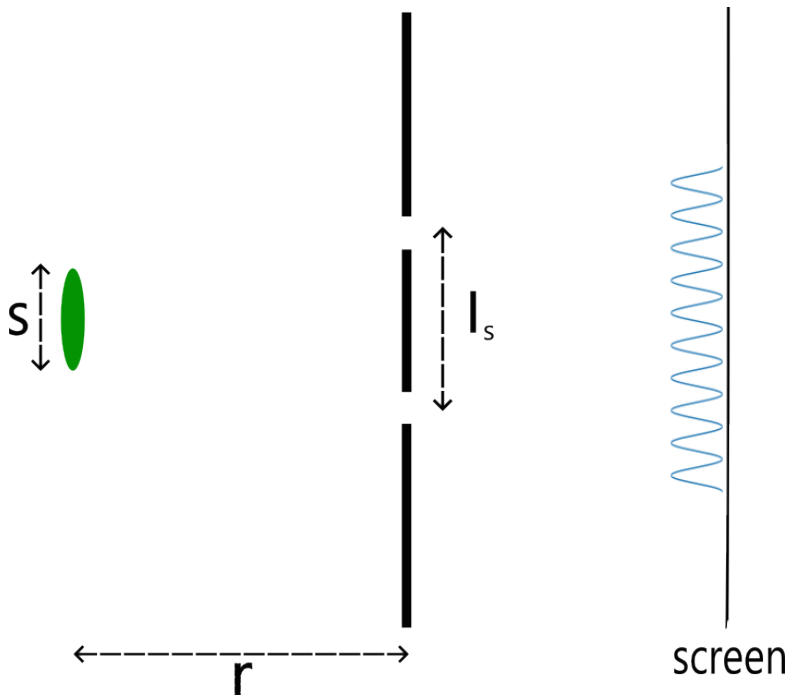


Figure 2.1.5: Double-slit setup illuminated by an extended source of dimension s.

Equation (2.7) can be considered a condition on the spatial separation l_s for two points on the wavefront to be coherent.

2.1.2.3 Partial coherence

In practice it is hard to achieve fully coherent radiation, usually the coherence condition is only approximately met and the signal is neither fully coherent nor

fully incoherent.
The degree of coherence between two electromagnetic fields $E_1(t), E_2(t)$ is measured by the mutual correlation function ([Hec17, p. 600])

$$\Gamma_{12}(t) \equiv \langle E_1(t) E_2^*(t+\tau) \rangle_T. \tag{2.8}$$

The normalized version of the mutual correlation function, $\gamma(\tau)$, is often convenient to use, and is given by

$$\gamma(\tau) \equiv \frac{\Gamma_{12}(t)}{\sqrt{[\Gamma_{11}(0)\Gamma_{22}(0)]}}. \tag{2.9}$$

with

$$\Gamma_{11}(\tau) = \langle E_1(t) E_1^*(t+\tau) \rangle_T$$

and

$$\Gamma_{22}(\tau) = \langle E_2(t) E_2^*(t+\tau) \rangle_T.$$

It can be shown that the relation between $\gamma(\tau)$ and the total intensity of the interfering fields is [Hec17, p. 601]

$$I = I_1 + I_2 + 2\sqrt{I_1 I_2}\,\mathrm{Re}\{\gamma(\tau)\}. \tag{2.10}$$

The degree of coherence is given by $|\gamma(\tau)|$ and defines the coherence regime:

$$\begin{aligned} &|\gamma(\tau)| = 1 \quad \text{fully coherent} \\ &0 < |\gamma(\tau)| < 1 \quad \text{partially coherent} \\ &|\gamma(\tau)| = 0 \quad \text{incoherent.} \end{aligned}$$

2.1.3 Coherent imaging

For coherent illumination there is a constant relative phase between each output signal $O(x, y)$ in the image-plane.
Due to the constant phase difference the output signal $O_{\text{out}}(x, y)$ vary in unison at all points (x, y) and can be expressed as a convolution between the input signal $O_{\text{in}}(x, y)$ and the PSF $h(x, y)$ of the system

$$O_{\text{out}}(x, y) = h(x, y) \circledast O_{\text{in}}(x, y). \tag{2.11}$$

In the frequency domain the relation becomes

$$G_{\text{out}}(\nu_x, \nu_y) = H(\nu_x, \nu_y) G_{\text{in}}(\nu_x, \nu_y). \tag{2.12}$$

2.1.4 Propagation in free space

2.1.4.1 Transfer function of free space

By definition $H(\nu_x, \nu_y)$ is the factor an spatial harmonic is multiplied with to produce the output spatial harmonic. Considering a spatial harmonic plane wave in the plane $z = z_1$ given by $U(x, y, z_1) = \exp[-i2\pi(k_x x + k_y y)]$ and in plane $z = z_2$ a distance $d = z_2 - z_1$ along the optical axis given by $U(x, y, z_1) = \exp[-i(k_x x + k_y y + k_z z_2)]$. $H(\nu_x, \nu_y)$ is the found to be

$$H(\nu_x, \nu_y) = \frac{U(x, y, z_2)}{U(x, y, z_1)} = \exp[-ik_z d] = \exp\left[-i2\pi d\sqrt{\lambda^{-2} - \nu_x^2 - \nu_y^2}\right] \tag{2.13}$$

where it has been used that $k_z = \sqrt{k^2 - k_x^2 - k_y^2} = 2\pi\sqrt{\lambda^{-2} - \nu_x^2 - \nu_y^2}$.

2.1.4.2 Impulse response function of free space under Fresnel conditions

In general the impulse response function (PSF) $h(x, y)$ is the inverse Fourier transform of $H(\nu_x, \nu_y)$ from eq(2.13). The expression of $H(\nu_x, \nu_y)$ can be simplified under certain conditions. Assuming that $k_z >> k_x^2 + k_y^2$ it follows that $\lambda^{-2} >> \nu_x^2 + \nu_y^2$ and the squre root can be expanded as $\sqrt{\lambda^{-2} - (\nu_x^2 + \nu_y^2)} = \frac{1}{\lambda}\sqrt{1 - \lambda^2(\nu_x^2 + \nu_y^2)} \approx \frac{1}{\lambda}\left(1 - \frac{\lambda^2(\nu_x^2+\nu_y^2)}{2} + \frac{\lambda^4(\nu_x^2+\nu_y^2)^2}{8} - \dots\right)$. In the Fresnel approximation all the terms in the sum following the second is neglected and one is left with

$$\begin{aligned} H(\nu_x, \nu_y) &\approx \exp\left[-i\frac{2\pi d}{\lambda}\right] \exp\left[i\pi d\lambda(\nu_x^2 + \nu_y^2)\right] \\ &= \exp\left[-ikd\right] \exp\left[i\pi d\lambda(\nu_x^2 + \nu_y^2)\right] \end{aligned} \tag{2.14}$$

$$h(x, y) = \mathcal{F}^{-1}\left\{H(\nu_x, \nu_y)\right\} \approx \frac{i}{\lambda d} \exp\left[-ikd\right] \exp\left[-ik\frac{x^2 + y^2}{2d}\right]. \tag{2.15}$$

2.1.5 Fraunhofer diffraction

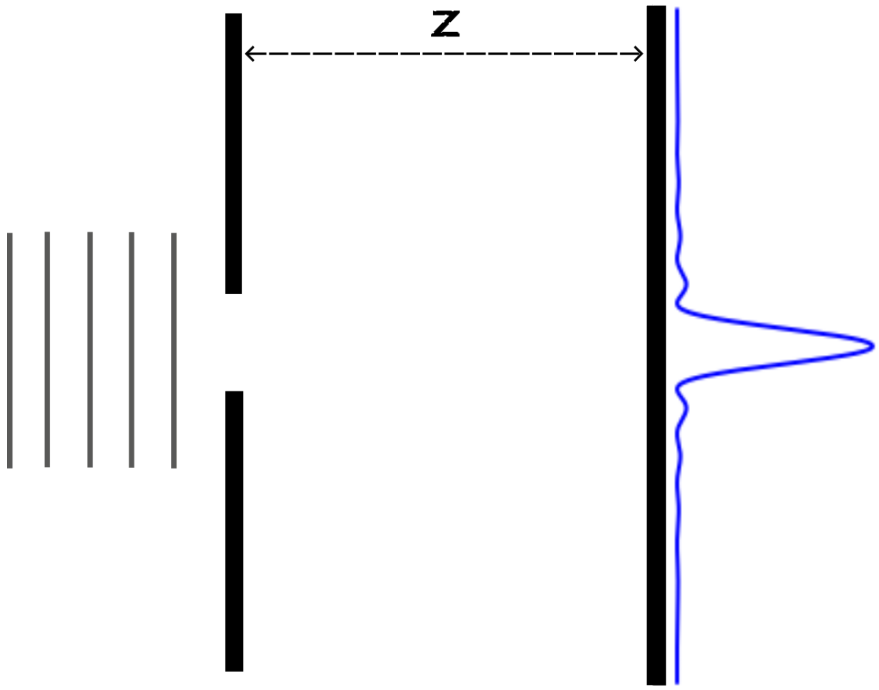


Figure 2.1.6: Plane wave passing through the horizontal aperture plane Σ and is collected on a screen a distance z away. The aperture-to-screen distance z is assumed large. Figure not to scale.

Fraunhofer diffraction is the regime describing the diffraction pattern of waves when collected at distances sufficiently far from the diffracting object, i.e in the far field.
Figure (2.1.6) illustrates the setup considered in this section where a plane wave is

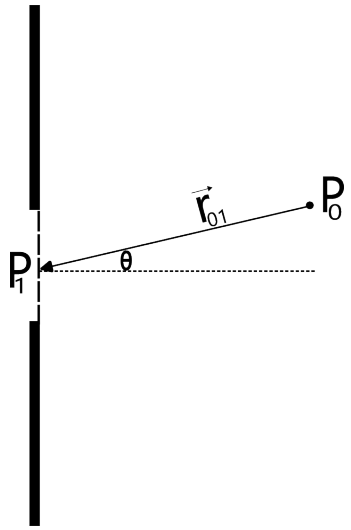


Figure 2.1.7: Diffraction by plane screen.

incident on an aperture and collected on a screen, where the separation z between the aperture plane and the screen is assumed to be large. Following the derivation in [Goo96, pp. 72–73] we start with the Huygens-Fresnel principle from scalar diffraction theory describing the amplitude $U(P1)$ at point P_1 when an incoming wave U is diffracted by an aperture Σ:

$$U(P_1) = \frac{1}{j\lambda} \iint\limits_{\Sigma} U(P_0) \frac{\exp(ikr_{01})}{r_{01}} \cos(\theta)\, ds. \tag{2.16}$$

With θ and r_{01} defined as indicated in figure (2.1.7).
Substituting z/r for $\cos(\theta)$ and series expanding the distance $r = \sqrt{z^2 + (x-\xi)^2 + (y-\eta)^2} = z\sqrt{1 + \frac{(x-\xi)^2}{z^2} + \frac{(y-\eta)^2}{z^2}}$ as $r \approx z\left[1 + \frac{1}{2}\left(\frac{x-\xi}{z}\right)^2 + \frac{1}{2}\left(\frac{y-\eta}{z}\right)^2\right]$ and neglecting other terms than z in the denominator one ends up with

$$U(P_1) = \frac{e^{ikz}}{i\lambda z} \iint\limits_{\xi,\eta} U(\xi,\eta) \exp\left[i\frac{k}{2z}\left[(x-\xi)^2 + (y-\eta)^2\right]\right] d\xi d\eta. \tag{2.17}$$

Considering the exponent inside the integral expression

$$i\frac{k}{2z}(x^2 + y^2 + \xi^2 + \eta^2 - 2x\xi - 2y\eta)$$

we note that if the far-field $z >> \frac{k(\xi^2+\eta^2)}{2}$ we can approximately disregard the ξ^2, η^2 terms and we are left with the fraunhofer diffraction equation:

$$U(x,y) = \frac{e^{ikz} e^{i\frac{k}{2z}(x^2+y^2)}}{i\lambda z} \iint\limits_{\xi,\eta} U(\xi,\eta) \exp\left[-i\frac{k}{z}(x\xi + y\eta)\right] d\xi d\eta. \tag{2.18}$$

We note that the integral in equation (2.18) is the Fourier transform of U with support only over the aperture Σ.

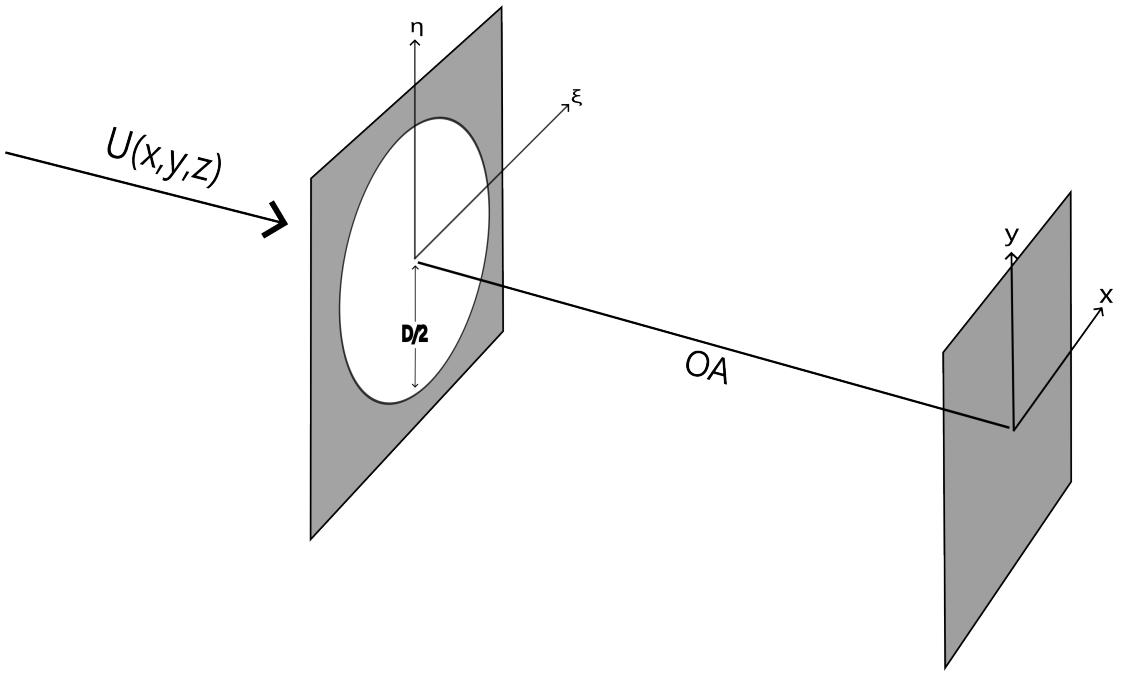


Figure 2.1.8: Plane wave passing through a plane with a circular aperture and collected on a screen a distance z away.

2.1.5.1 Circular aperature Fraunhofer diffraction pattern

Considering a case where a plane wave U_{in} is incident on a circular aperture as shown in Figure (2.1.8) the wave just to the right of the circular aperture can be described by the equation $U_{\text{in}}(x,y)p(x,y)$ where $p(x,y)$ is the circ function

$$p(x,y) = \begin{cases} 1 & \text{if} \quad x^2+y^2 < \left(\frac{D}{2}\right)^2 \\ 0 & \text{elsewhere} \end{cases} \tag{2.19}$$

Substituting the aperture distribution into equation (2.18) we get the fraunhofer equation for a circular aperture

$$U(x,y) = \frac{e^{ikz}e^{i\frac{k}{2z}(x^2+y^2)}}{i\lambda z} \iint\limits_{\xi,\eta} U_{\text{in}}(\xi,\eta)p(\xi,\eta)\exp\left[-i\frac{k}{z}(x\xi+y\eta)\right]\,d\xi d\eta. \tag{2.20}$$

If $U_{in}(x,y)$ is a plane wave travelling in the z-direction of unit amplitude, one has $U(x,y) = \sqrt{I_{\text{in}}} = 1$ and equation (2.20) becomes

$$U(x,y) = \frac{e^{ikz}e^{i\frac{k}{2z}(x^2+y^2)}}{j\lambda z} \iint\limits_{\xi,\eta} p(\xi,\eta)\exp\left[-i\frac{2\pi}{z\lambda}(x\xi+y\eta)\right]\,d\xi d\eta \tag{2.21}$$

where $\frac{2\pi}{\lambda}$ has been substituted for k.
It is worth noting in passing that we can identify the integral in equation (2.21) as the Fourier transform of the aperture distribution $p(x,y)$, and we can write

$$U(x,y) = \frac{e^{ikz}e^{j\frac{k}{2z}(x^2+y^2)}}{i\lambda z} P\left(\frac{x}{\lambda z}, \frac{y}{\lambda z}\right) \tag{2.22}$$

where $P = \mathcal{F}\{p(x, y)\}$.
The resulting intensity pattern observed on the screen can thus be given as

$$I = I_0 \left| P\left(\frac{x}{\lambda z}, \frac{y}{\lambda z}\right)\right|^2 \tag{2.23}$$

for a constant I_0. The expression given in equation (2.23) can be calculated to be ([Goo96, p. 77])

$$I = I_0 \left| \frac{2J_1(\pi D\rho/\lambda z)}{(\pi D\rho/\lambda z)}\right|^2 \tag{2.24}$$

where J_1 is the first-order bessel-function, D is the diameter of the circular aperture and ρ is the distance from the screen centre to the point (x, y).

The first zero of the bessel-function J_1 defines the radius of the bright central

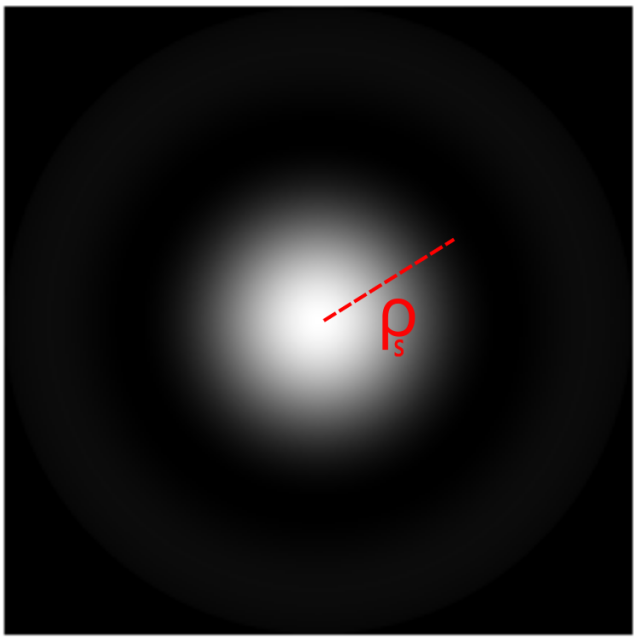


Figure 2.1.9: Airy disk with radius ρ_s.

lobe of the diffraction pattern. The zero is found at approximately

$$\rho_s = 1.22\frac{\lambda z}{D}. \tag{2.25}$$

2.1.6 The single lens imaging system as a low-pass filter

The main takeaway from this section will be that the a single lens imaging system acts as a low-pass filter. In terms of the transfer functions in the frequency domain, it will be shown that

$$\begin{aligned} G_{\text{out}}(\nu_x, \nu_y) &= H(\nu_x, \nu_y)G_{\text{in}}(\nu_x, \nu_y) \\ \text{with } H(\nu_x, \nu_y) &= p(\nu_x/w_{\text{cutoff}}, \nu_y/w_{\text{cutoff}}) \end{aligned} \tag{2.26}$$

where $p(x, y)$ is a scaled circ-function introduced in equation (2.19). Equation (2.26) shows that there is a sharp cutoff in the frequencies that the system can capture, and it follows that the part of the signal containing frequencies $\nu > \frac{D}{2} \cdot w_{\text{cutoff}}$

will be lost.

We follow the derivation given in [Bah19, pp. 141–144].

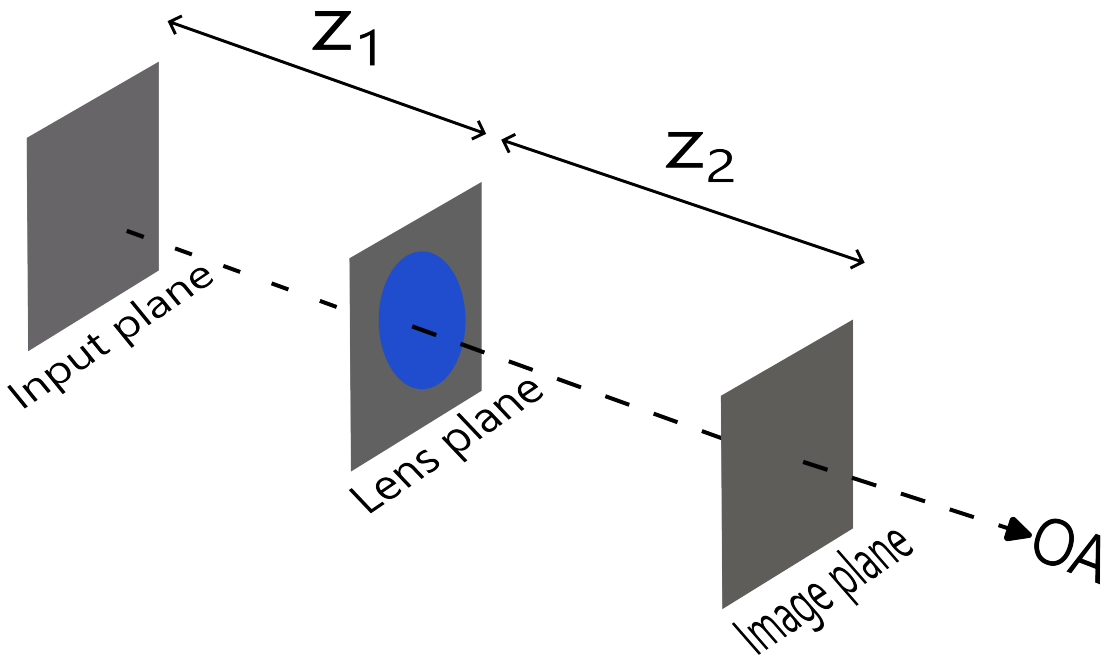


Figure 2.1.10: Single lens imaging system.

We recall that the point spread function $h(x, y)$ is the system response to a unit impulse given as $h(x, y) = \mathcal{S}\{\delta(x, y)\}$.
We find $h(x, y)$ by following the response from an point source placed at (0,0) on the optical axis in the input plane in Figure (2.1.10) and follow the signal to the output plane. The resulting output-signal will be the desired impulse response function $h(x, y)$.

Using equation (2.15) propagation into the lens-plane results in

$$U_{L^-}(x, y) = h_1 \exp\left[-ik\frac{x^2 + y^2}{2z_1}\right] \tag{2.27}$$

with $h_1 = (i/\lambda z_1) \exp(-ikz_1)$.

Right after crossing the lens aperture we have the output

$$U_{L^+} = U_{L^-}(x, y)p(x, y) \exp\left[ik\frac{x^2 + y^2}{2f}\right] \tag{2.28}$$

where $p(x, y)$ is the circ-function from equation (2.19) and $\exp\left[ik\frac{x^2+y^2}{2f}\right]$ is a quadratic phase-factor from the lens ([Goo96, p. 100]), with f being the focal length of the lens.

Using the Fraunhofer equation (2.18) to find the output signal in the image plane,

it leads to the impulse function

$$\begin{aligned} h(x,y) &= \frac{e^{ikz}}{i\lambda z} \iint\limits_{\xi,\eta} U_{L^+}(\xi,\eta) \exp\left[-i\pi \frac{(x-\xi)^2+(y-\eta)^2}{\lambda z_2}\right] d\xi d\eta \\ &= \frac{e^{ikz}}{i\lambda z} \iint\limits_{\xi,\eta} h_1 U_{L^-}(\xi,\eta) p(\xi,\eta) \exp\left[ik\frac{\xi^2+\eta^2}{2f}\right] \exp\left[-i\pi \frac{(x-\xi)^2+(y-\eta)^2}{\lambda z_2}\right] d\xi d\eta \\ &= h_1 h_2 \exp\left[i\frac{k}{2z_2}(x^2+y^2)\right] \iint\limits_{\xi,\eta} p(\xi,\eta) \exp\left[-i\epsilon\pi\frac{\xi^2+\eta^2}{\lambda}\right] \exp\left[-i\frac{2\pi}{z_2\lambda}(x\xi+y\eta)\right] d\xi d\eta \end{aligned} \tag{2.29}$$

with factors $h_1 = (i/\lambda z_1)\exp(ikz_1)$, $h_2 = (i/\lambda z_2)\exp(ikz_2)$ and ϵ given by

$$\epsilon = \frac{1}{z_1} + \frac{1}{z_2} - \frac{1}{f}. \tag{2.30}$$

Observing that the integral in equation (2.29) is the Fourier transform of $p(x,y)\exp\left[-i\epsilon\pi\frac{x^2+y^2}{\lambda}\right]$ we can write

$$h(x,y) = h_2 h_1 \exp\left[-i\pi\frac{x^2+y^2}{\lambda z_2}\right] P_1(x/\lambda z_2, y/\lambda z_2) \tag{2.31}$$

with $P_1 = \mathcal{F}\left\{p(x,y)\exp\left[-i\epsilon\pi\frac{x^2+y^2}{\lambda}\right]\right\}$.
Assuming $\frac{x^2+y^2}{\lambda z_2} << 1$ and a focused system $\epsilon = 0$, we get the approximate expression

$$h(x,y) \approx h_1 h_2 P(x/\lambda z_2, y/\lambda z_2) \tag{2.32}$$

where P is the Fourier transform $\mathcal{F}\{p(x,y)\}$ of the circ-function $p(x,y)$ from equation (2.19).

Now that we have obtained $h(x,y)$ we can find the frequency domain transfer function of the system

$$\begin{aligned} H(\nu_x,\nu_y) = \mathcal{F}\{h(x,y)\} &= \mathcal{F}\left\{\iint\limits_{-\infty}^{+\infty} p(\xi,\eta)\exp\left[-j\frac{2\pi}{\lambda z_2}(x\xi+y\eta)\right] d\xi d\eta\right\} \\ &= \text{const}\cdot\ p(-\lambda z_2\nu_x, -\lambda z_2\nu_y). \end{aligned} \tag{2.33}$$

We see that the transfer function $H(\nu_x,\nu_y) \sim p(\nu_x/w_{\text{cutoff}}, \nu_y/w_{\text{cutoff}})$, with $w_{\text{cutoff}} = \frac{1}{\lambda z_2}$, is a scaled circ-function as was to be shown.
This means that the cutoff frequency ν_{cutoff} becomes

$$\nu_{\text{cutoff}} = \frac{D}{2}\cdot w_{\text{cutoff}} = \frac{D}{2\lambda z_2}. \tag{2.34}$$

For coherent imaging the cutoff frequency can be given as NA/λ ([Zhe16, p. 1.1]), which is often convenient to phrase in terms of wave vector notation:

$$k_{\text{cutoff}} = k_0 \cdot \text{NA} \tag{2.35}$$

with $k_0 = 2\pi/\lambda$.

2.1.7 Space-bandwidth product (SBP)

If an input signal $i(x, y)$ is band-limited in the sense that the entire spectrum I is contained within the rectangle $-B_x \leq \nu_x \leq B_x, -B_y \leq \nu_y \leq B_y$, and $i(x, y)$ is only significant on a rectangular spatial region $-L_x \leq x \leq L_x, -L_y \leq y \leq L_y$, then $i(x, y)$ can be recovered with good accuracy [Goo96, pp. 26–27] by sampling at a total of

$$M = \frac{2L_x}{1/(2B_x)} \cdot \frac{2L_y}{1/(2B_y)}$$

points by sampling at the Nyquist frequencies $2B_x, 2B_y$.
This motivates the definition of the space-bandwith product (SBP) for an optical system:

$$\text{SBP} = 16L_xL_yB_xB_y. \tag{2.36}$$

It is clear that the SBP is a performance measure of the amount of information the optical system can capture.

2.2 Fourier ptychography imaging

Fourier ptychography (FP) is a computational imaging technique for resolution enhancement, phase retrieval and high SBP microscopy.
The SBP of the total imaging system in enhanced by gathering information in the Fourier-domain of the high-resolution complex object $O(x, y)$ by sequentially illuminating the object from several incident directions (k_x, k_y).

The angled illumination from direction (k_x, k_y) can be modelled as a coherent imaging process with the equation

$$O_{\text{out}}(x, y) = h(x, y) \circledast \left(O_{\text{in}}(x, y)e^{ik_xx+ik_yy}\right) \tag{2.37}$$

for the PSF of the system $h(x, y)$.
Transforming equation (2.37) into the Fourier domain we get

$$G_{out} = H(\nu_x, \nu_y)G_{in}(\nu_x - k_x, \nu_y - k_y) \tag{2.38}$$

by remembering the fact from fourier theory that $\mathcal{F}(f(x)e^{ikx}) = F(\nu - k)$.

Recalling the discussion from section (2.1.6), it was shown that the Fourier-domain transfer function $H(\nu_x, \nu_y)$ for a single-lens imaging system was effectively a low-pass filter described by $H(\nu_x, \nu_y) = p(\nu_x/w_{\text{cutoff}}, \nu_y/w_{\text{cutoff}})$.
Using the fact that the cutoff frequency for an objective lens with numerical aperture NA is $k_0 \cdot \text{NA}_{\text{obj}}$ as discussed in section (2.1.6), we can reasonably assume that the imaging equation equation (2.38) represents the extraction of a circular patch of radius $k_0 \cdot \text{NA}_{\text{obj}}$ centred at location (k_x, k_y) from the Fourier spectrum of the high-resolution object $O_{in}(x, y)$. Again we have defined $k_0 = 2\pi/\lambda$.
If the illumination source used is a LED-array then each image captured by a LED in the LED-array corresponds to a direction (k_x, k_y) that provide information about a region (a disk) in the Fourier spectrum of the object. Figure (2.2.1) illustrates the sampling of the Fourier spectrum of $O_{\text{in}}(x, y)$ where each illumination source captures a disk in the spectrum of $O_{\text{in}}(x, y)$. More precisely, each

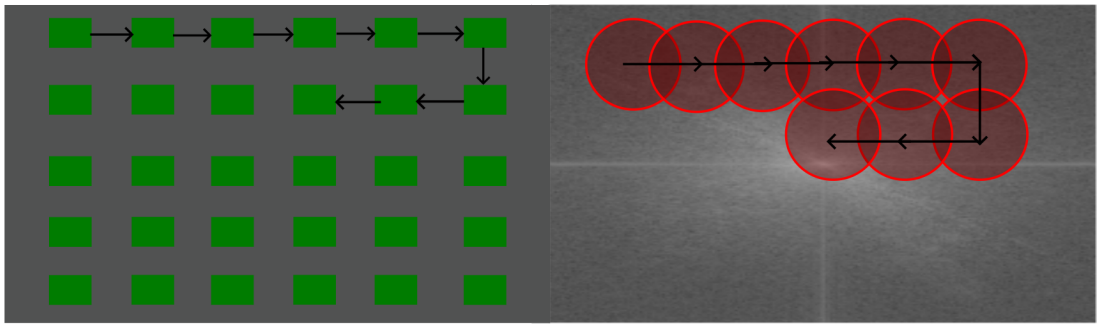


Figure 2.2.1: Conceptual Fourier ptychography illustration: Illuminating a high-resolution object by each LED to capture Fourier spectrum sequentially

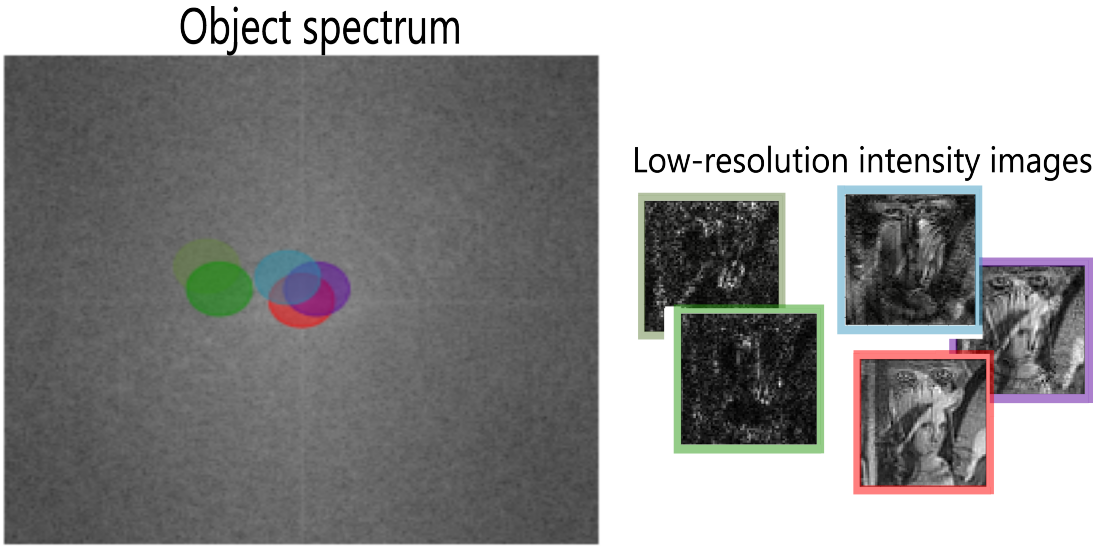


Figure 2.2.2: Correspondence between disks in the fourier spectrum of the high-resolution object $O(x, y)$ and the captured low-resolution images in fourier ptychography.

captured image, referred to in this project as as "low-resolution images", corresponds to a circular patch in the Fourier spectrum of the high-resolution object. This is formulated mathematically in equation (2.39).

$$|O_{\text{out}}(x, y)|^2 = |\mathcal{F}^{-1}\{H(\nu_x, \nu_y)\mathcal{F}\{O_{\text{in}}(x, y)\}\}|^2. \tag{2.39}$$

$H(\nu_x, \nu_y)$ is the transfer function for the imaging system.
This correspondence is illustrated in figure (2.2.2).

The angled imaging is typically implemented in practice by the use of a LED-array as shown in figure (2.2.3). The complex object $O_{in}(x, y)$ is illuminated by each LED successively, with each LED corresponding to a different direction with incident wave-vector components (k_x, k_y). The image corresponding to a specific LED is collected by the imaging system as a low-resolution intensity image as discussed previously.

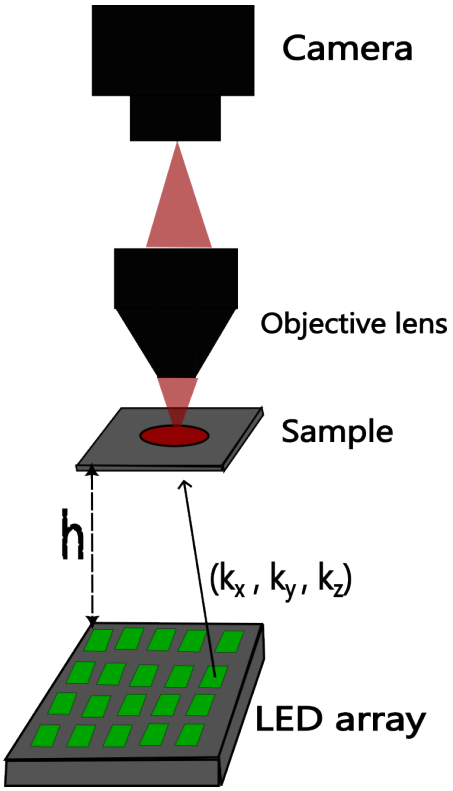


Figure 2.2.3: Typical setup for fourier ptychography imaging.

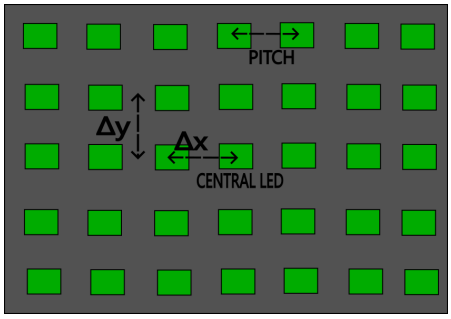


Figure 2.2.4: LED-matrix illustration.

The values for the wave-vector components (k_x, k_y) can be given as

$$\begin{aligned} k_{xn} &= \frac{2\pi}{\lambda} \frac{\Delta x_n}{(\Delta x_n)^2 + (\Delta y_n)^2 + h^2} \\ k_{yn} &= \frac{2\pi}{\lambda} \frac{\Delta y_n}{(\Delta x_n)^2 + (\Delta y_n)^2 + h^2} \end{aligned} \tag{2.40}$$

where $\Delta x_n, \Delta y_n$ is the offset from the central LED as shown in figure (2.2.4), h is the distance from the LED-matrix to sample as given in figure (2.2.3) and λ is the wavelength of the illuminating light.

The information captured about the Fourier spectrum of the object $O_{\text{in}}(x, y)$ in the form of a set of low-resolution images $\mathbf{I}_{\text{LR}}$ is used to by various reconstruction algorithms to recover $O_{\text{in}}(x, y)$.
As discussed earlier this section, the low-resolution intensity images captured by the camera have lower resolution since the transfer function of the imaging system essentially functions as a low-pass filter. This means that recovering $O_{\text{in}}(x, y)$ from $\mathbf{I}_{\text{LR}}$ can be seen as enhancing the resolution of the imaging system.

Another way to formulate the resolution enhancement by FP imaging is in terms of the numerical aperture. The NA of the system in increased by FP imaging from NA_{obj} of the objective lens to the synthesized numerical aperture NA_{sys}.
The synthesized aperture of the system is given by [Ou+15]

$$\text{NA}_{\text{sys}} = \text{NA}_{\text{obj}} + \text{NA}_{\text{illu}} \tag{2.41}$$

where $\mathrm{NA}_{\mathrm{illu}}$ is the increase due to the computational enhancement using the information gathered about the high-resolution object's Fourier spectrum.
The increase in numerical aperture $\mathrm{NA}_{\mathrm{illu}}$ is given by [Ou+15]

$$\mathrm{NA}_{\mathrm{illu}} = \max_n \frac{\sqrt{k_{xn}^2 + k_{yn}^2}}{k_0} = \sin\left(\max_n \theta_n\right) \tag{2.42}$$

where n iterates over all the low-resolution images and θ_n is the angle of illumination of low-resolution image n.
We see from equation (2.42) that the increase in NA of the system $\mathrm{NA}_{\mathrm{sys}}$ is independent of the $\mathrm{NA}_{\mathrm{obj}}$ of the employed objective lens.
This property can be utilised by selecting a low-NA objective lens with a large field-of-view (FoV) and rely on the $\mathrm{NA}_{\mathrm{illu}}$ term to increase the resolution sufficiently.
In this way it is possible achieve a large increase in the SBP of the imaging system.

2.2.1 Bright-field and dark-field FP images

The low-resolution images can be classified as either *bright-field* or *dark-field*. As discussed previously each low-resolution image corresponds to an incident direction that is characterised by its wave-vector components (k_{xn}, k_{yn}), which in turn defines an illumination angle θ_n

$$\sin(\theta_n) = \frac{\sqrt{k_{xn}^2 + k_{yn}^2}}{k_0}. \tag{2.43}$$

The criterion for an illumination angle to result in a bright-field image can be stated [Sun+18]

$$\sin(\theta_n) \leq \mathrm{NA}_{\mathrm{obj}} \tag{2.44}$$

after which it naturally follows that for a dark-field image

$$\sin(\theta_n) > \mathrm{NA}_{\mathrm{obj}}. \tag{2.45}$$

Its clear that when θ_n is large enough the imaging mode changes from bright-field to dark-field. The criterion for bright-field imaging can be restated in the Fourier domain as the condition that the zero-frequency is contained in the disk with radius $k_0 \cdot \mathrm{NA}_{\mathrm{obj}}$ (cutoff frequency) centred at (k_{xn}, k_{yn}):

$$(k_{xn} - 0)^2 + (k_{yn}^2 - 0) < k_0 \cdot \mathrm{NA}_{\mathrm{obj}}. \tag{2.46}$$

Figure (2.2.5) illustrates which shifts (k_{xn}, k_{yn}) are characterized as bright-field and dark-field for a uniform sampling of the Fourier spectrum.
It is clear from equation (2.46) that the bright-field images contain information about the low frequencies in the Fourier spectrum of the high-resolution object, while the dark-field images contain information about the higher frequencies and hence the finer details in the high-resolution object $O(x, y)$.

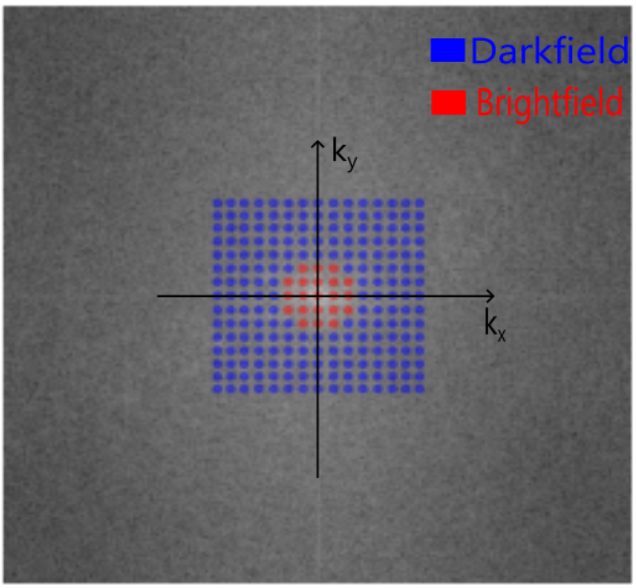


Figure 2.2.5: Bright-field shift centres (red) and dark-field shift centres (blue).

2.2.2 Maximal achievable FP resolution enhancement

The numerical aperture of the system increased to $\mathrm{NA}_{\mathrm{sys}}$ due to computational enhancement using the information provided about the fourier spectrum of the high-resolution object $O(x, y)$.
As observed in section (2.1.6) the resolution for coherent imaging system is given λ/NA, which implies that the final resolution for an FP imaging system is given by

$$\delta r = \frac{\lambda}{\mathrm{NA}_{sys}} = \frac{\lambda}{\mathrm{NA}_{\mathrm{obj}} + \mathrm{NA}_{\mathrm{illu}}}. \tag{2.47}$$

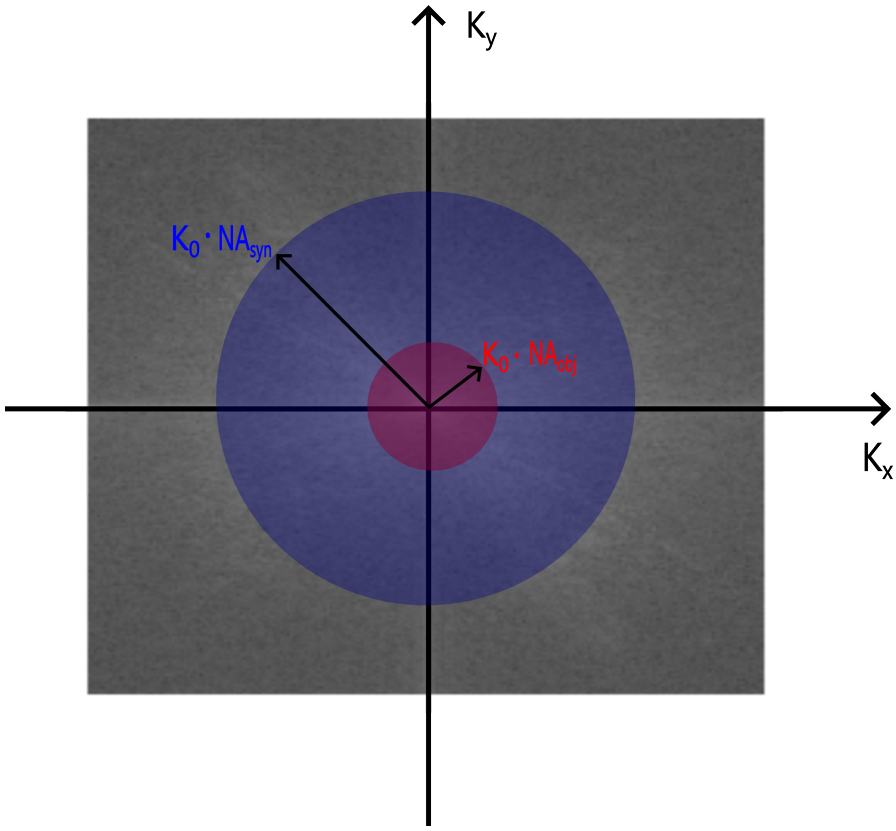


Figure 2.2.6: Illustration of resolution enhancement from $\mathrm{NA_{obj}}$ to $\mathrm{NA_{sys}}$ by FP imaging.

The largest pixel size the low-resolution intensity images can have and still fully capture the full signal bandwidth $\lambda/\mathrm{NA_{obj}}$ without anti-aliasing is given by the Nyquist sampling theorem as $\lambda/(2 \cdot \mathrm{NA_{obj}})$. Using FP imaging the synthetic numerical aperture of the high-resolution complex amplitude is given $\mathrm{NA_{sys}}$. The high-resolution intensity image $|O(x, y)|^2$ then corresponds to a convolution in the Fourier domain which doubles the pass-band to $2 \cdot k_0 \cdot NA_{sys}$. It follows then from the Nyquist sampling theorem that the largest pixel-size that samples the signal without anti-aliasing is given by $\lambda/(4 \cdot \mathrm{NA_{sys}})$.
This gives a total pixel-enhancement for FP imaging of

$$\text{pixel enhancement} = 2 \cdot \frac{\mathrm{NA_{sys}}}{\mathrm{NA_{obj}}}. \tag{2.48}$$

Related to the pixel-wise is the increase in information gathering capacity, measured by the space-bandwidth product (SBP). The SBP enhancement is given by [Sun+17]

$$\text{SBP enhancement} = \left(1 + \frac{\mathrm{NA_{illu}}}{\mathrm{NA_{obj}}}\right)^2 = \left(\frac{\mathrm{NA_{sys}}}{\mathrm{NA_{obj}}}\right)^2. \tag{2.49}$$

2.2.3 Iterative FP reconstruction method

The method of reconstruction in this section was developed by Zheng([Zhe16]).
It is an iterative algorithm that recovers the Fourier spectrum of the high-resolution

object, $\mathcal{F}(O(x,y))$. The method therefore also recovers $O(x,y)$ which can be computed from the recovered spectrum. Since the complex amplitude $O(x,y)$ is recovered the phase of the high-resolution object is also retrieved.

The fundamental idea of the iterative reconstruction method is to impose consistency between the information about the sampled patches of the Fourier spectrum of the high-resolution object provided by the low-resolution images gathered.
The fact that the method relies on enforcing consistency among the collected data requires there to be a certain amount of overlap between the sampled disks in the Fourier domain as illustrated in Figure (2.2.7).

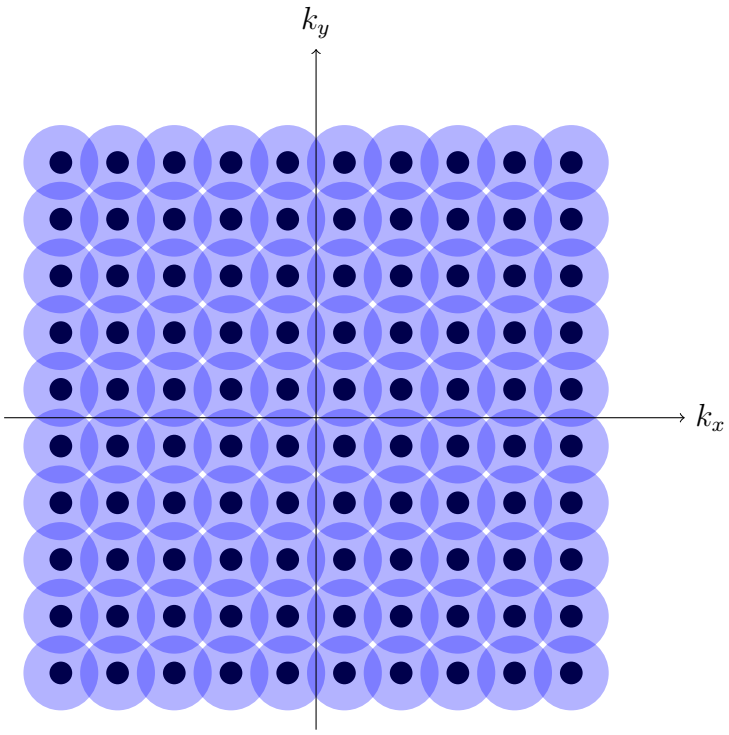


Figure 2.2.7: Fourier domain overlap between disks centred at (k_{xm}, k_{ym}).

The main steps of the algorithm can be given as follows:
1.Initialize the current estimated high-resolution image $\sqrt{I_{hr}}e^{i\phi_{hr}}$ with an initial guess.
2.For a given incident direction extract a circular patch $D_{k_{xn},k_{yn}}$ centred at (k_{xn}, k_{yn}).
3.Compute the low-resolution complex amplitude corresponding to the extracted patch $D_{k_{xn},k_{yn}}$: $\sqrt{I_{ln}}e^{i\phi_{ln}} = \mathcal{F}^{-1}(D_{k_{xn},k_{yn}})$.
4.Make the computed low-resolution image from the extracted patch consistent with the actual low-resolution amplitude $\sqrt{I_l}$ gathered during imaging by rescaling:

$$\sqrt{I_{ln}}e^{i\phi_{ln}} \rightarrow \frac{\sqrt{I_l}}{\sqrt{I_{ln}}} \cdot \sqrt{I_{ln}}e^{i\phi_{ln}} \tag{2.50}$$

5.Compute $\mathcal{F}\left\{\sqrt{I_{ln}}e^{i\phi_{ln}}\right\}$ of the intensity adjusted image $\sqrt{I_{ln}}e^{i\phi_{ln}}$, and update the corresponding circular area centred at (k_x, k_y) in the fourier spectrum of the current estimate of the high-resolution object $\sqrt{I_{hr}}e^{i\phi_{hr}}$.

6.Iterate through all incident directions (k_{xn}, k_{yn}) (indexed by variable n).
7. Repeat until convergence.
A more detailed version of the iterative reconstruction method is given in alg.(1).

Algorithm 1 Fourier Ptychography Reconstruction Algorithm

procedure RECONSTRUCT($I, N_{\text{LEDs}}, NA, NITERS$)
  $m \leftarrow$ number of low-resolution images $N_{\text{LEDs}} \times N_{\text{LEDs}}$ in I
  $Ks \leftarrow$ positions of LEDs in frequency domain
  $\sqrt{I_{HR}}e^{i\phi_{HR}} \leftarrow$ initialize array of size $M \times N$
  $F_{\text{HR}} \leftarrow \mathcal{F}\{\sqrt{I_{HR}}e^{i\phi_{HR}}\}$
  for $i \leftarrow 0$ to $NITERS$ **do**
    for j in RANGE(LENGTH(Ks)) **do**
      $[k_x, k_y] \leftarrow Ks[j]$
      disk $\leftarrow$ EXTRACT_CIRCULAR($[k_x, k_y], k_0 \cdot \text{NA}$)
      $\sqrt{I_{lj}}e^{i\phi_{lj}} \leftarrow \mathcal{F}^{-1}\{\textbf{disk}\}$
      $\sqrt{I_{lj}}e^{i\phi_{lj}} \leftarrow \frac{\sqrt{I_l}}{\sqrt{I_{ln}}} \cdot \sqrt{I_{ln}}e^{i\phi_{ln}}$
      $F_{HR}[kxl : kxh, kyl : kyh] \leftarrow \sqrt{I_{HR}}e^{i\phi_{HR}}$
    end for
  end for
  $\sqrt{I_{HR}}e^{i\phi_{HR}} = \mathcal{F}^{-1}\{F_{HR}\}$
  return $\sqrt{I_{HR}}e^{i\phi_{HR}}$
end procedure

2.3 Neural networks

2.3.1 Fundamentals

Machine learning models can usually be described as a parametrized function $f(x; \boldsymbol{\theta})$ where the parameters $\boldsymbol{\theta}$ can be optimized to minimze a given cost function $L(y, f(x; \boldsymbol{\theta}))$.
Neural networks are a specific type of machine learning models consisting of a connected graph of nodes. Each node in the graph represents a specific function and can thus be thought of as extracting a certain feature from the input data.

Neural networks are usually organized into layers as illustrated in figure (2.3.1). The network illustrated in figure (2.3.1) is such the the next layer is a function of only the previous layer. In such a feed-forward architecture the network can be considered a composition of functions

$$\mathbf{x} \xrightarrow{f_1} f_1(\mathbf{x}) \xrightarrow{f_2} f_2(f_1(\mathbf{x})) \xrightarrow{f_3} \cdots \xrightarrow{f_{n-1}} f_{n-1}(\cdots f_2(f_1(\mathbf{x}))) \xrightarrow{f_n} f_n(f_{n-1}(\cdots f_2(f_1(\mathbf{x})))$$

where f_n is the function that computes layer n from layer $n-1$.
In this perspective the layers of the network represents successive transformations of the data in a way which helps solve the end goal, which is to minimize the loss function L. Not all neural networks needs to be strictly feed-forward like the network in figure (2.3.1) but the intuition of layer-wise processing and successive feature extraction is something that applies to many types of neural networks.

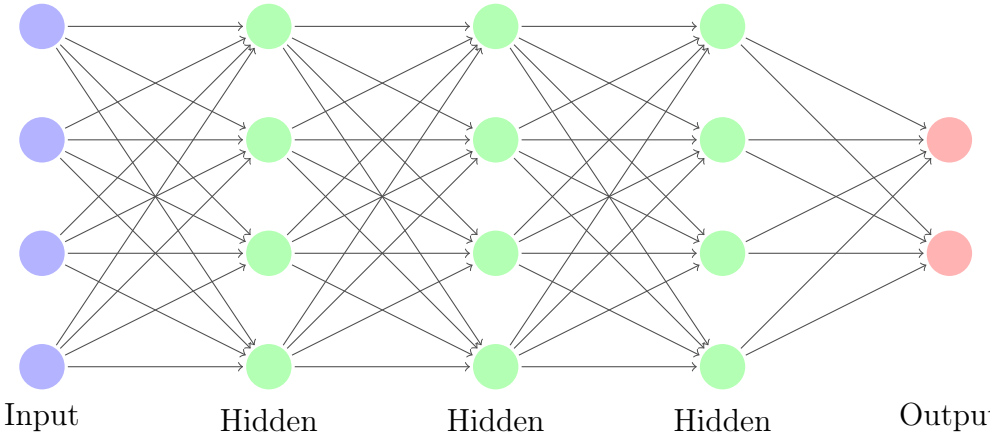


Figure 2.3.1: Fully connected feed-forward network with three hidden layers and two output nodes. Each arrow can be associated with a weight parameter $w_{ij}^{(n)}$ between the two nodes.

We denote a node number j in a hidden layer n by $h_j^{(n)}$.
For a feed-forward network the value of each node in the hidden layer is computed in a two-step computation:

$$\begin{aligned} h_j'^{(n)} &= \sum_{i=1}^{m} w_{ij}^{(n)} h_i^{(n-1)} + b_j^{(n)} \qquad (1) \\ h_j^{(n)} &= \rho(h_j'^{(n)}) \qquad (2) \end{aligned} \tag{2.51}$$

where (1) is a weighted sum of the node values in the previous layer $n-1$ plus a bias b_j, followed by (2) which is the application of a nonlinear function ρ.

We see that all the nodes $h_j^{(n)}$ in the next layer can be computed as a matrix multiplication where the weights w_{ij} is replaced with a weight matrix $\mathbf{W}$ and the biases b_j is replaced with a bias vector $\mathbf{b}$

$$\begin{aligned} \mathbf{h}'^{(n)} &= \mathbf{W}^{(n)\top} \mathbf{h}^{(n-1)} + \mathbf{b}^{(n)} \qquad (1) \\ \mathbf{h}^{(n)} &= \rho\left(\mathbf{h}'^{(n)}\right). \qquad (2) \end{aligned} \tag{2.52}$$

Where now all the nodes $h_j^{(n)}$ in hidden layer n is represented by the vector $\mathbf{h}^{(n)}$, and $\rho(\mathbf{h}^{(n)})$ denotes the element-wise application of the nonlinear function ρ.

The fact that matrix multiplication and vector addition can be used to do the main bulk of the computation in the network is one of the main reasons why neural network computations can be done so quickly on hardware like GPUs that are made for parallellizable computation such as matrix multiplications. For these reasons neural networks usually operates on vectorized data, and vectors, matrices and multidimensional arrays are the default language in neural network literature. The fast computations enabled by specialized hardware such as GPUs is a major reason why the very large neural networks that are common today have become feasible.

2.3.2 Optimization of neural networks

The parameters $\boldsymbol{\theta}$ in the neural network $f(\mathbf{x};\boldsymbol{\theta})$ will be optimized to minimize the loss function of the problem $L(y, f(\mathbf{x};\boldsymbol{\theta}))$ where y is the desired output for the input $\mathbf{x}$.
In many cases one has a dataset of datapoints $(\mathbf{x}_i, y_i)_{i=1}^N$ and hence it is desired to find the parameters $\boldsymbol{\theta}^*$ that minimizes the sum the loss function over the entire dataset

$$\boldsymbol{\theta}^* = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i;\boldsymbol{\theta})). \tag{2.53}$$

Equation (2.53) defines the optimal values for the parameters $\boldsymbol{\theta}$, and any optimization prodcedure for neural networks would hope to discover these values. Since optimizing neural networks is a difficult high-dimensional optimization problem, reaching the optimal solution is not always realistic. Luckily one does not need to find the optimal set of parameters $\boldsymbol{\theta}^*$ for the neural networks to be useful.

Most neural networks optimization methods only use the first order derivatives $\frac{\partial L}{\partial \theta_k}$ when computing updates to the parameters. The reason for this is that computing second order derivatives $\frac{\partial L}{\partial \theta_{k1} \partial \theta_{k2}}$ is very expensive when the number of parameters in the network is very large, which is often the case.
Thus it remains to find a way to compute the gradient $\frac{\partial L}{\partial \theta_k}$ of the loss with respect to each parameter $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_K)$ in the network.
If one uses all the datapoints when computing the gradients we are seeking a way to compute the derivative

$$\frac{\partial L_{\text{tot}}}{\partial \theta_k} = \frac{\partial \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i;\boldsymbol{\theta}))}{\partial \theta_k} \tag{2.54}$$

that can be used for computing updating the parameters in an optimization routine.

2.3.2.1 Backpropagation

The conventional way of computing these gradient is with the backpropagation algorithm.
As an illustration of the backpropagation algorithm we consider how the weights w_{ij} from the feed-forward described by equation (2.51) is done with backpropagation.
We are interested in computing $\frac{\partial L}{\partial w_{ij}}$ since this derivative can be used for updating the parameters w_{ij}, b_j using one of the first-order optimization methods that are common in neural network optimization.
Using the chain rule we can write

$$\begin{aligned} \frac{\partial L}{\partial w_{ij}} &= \frac{\partial L}{\partial h_j} \frac{\partial h_j}{\partial w_{ij}} \\ \frac{\partial L}{\partial b_j} &= \frac{\partial L}{\partial h_j} \frac{\partial h_j}{\partial b_j}. \end{aligned} \tag{2.55}$$

From equation (2.55) we observe that if we can compute $\frac{\partial L}{\partial h_j}$ we can find $\frac{\partial L}{\partial w_{ij}}$ and $\frac{\partial L}{\partial b_j}$ and hence compute the first-order derivatives for all parameters $\boldsymbol{\theta} =$

$\left(\mathbf{W}^{(1)} \cdots \mathbf{W}^{(K)}, \mathbf{b}^{(1)} \cdots \mathbf{b}^{(K)}\right)$.

$$\begin{aligned}
\frac{\partial L}{\partial h_j^{(n-1)}} &= \sum_{i=1}^{H_n} \frac{L}{\partial h_i^{(n)}} \frac{\partial h_i^{(n)}}{\partial h_j^{(n-1)}} \\
&= \sum_{i=1}^{H_n} \left[\frac{L}{\partial h_i^{(n)}} \rho' \left(\sum_{l=1}^{H_{n-1}} w_{lj}^{(n)} h_l^{(n-1)} \right) \frac{\partial}{\partial h_j^{(n-1)}} \left(\sum_{l=1}^{H_{n-1}} w_{lj}^{(n)} h_l^{(n-1)} \right) \right] \\
&= \sum_{i=1}^{H_n} \left[\frac{L}{\partial h_i^{(n)}} \rho' \left(\sum_{l=1}^{H_{n-1}} w_{lj}^{(n)} h_l^{(n-1)} \right) w_{ij}^{(n)} \right]
\end{aligned} \tag{2.56}$$

The weight parameters $w_{ij}^{(n)}$ denotes the weight parameters in the matrix $\mathbf{W}^{(n)}$ between hidden layers $(n-1), n$. The number of nodes in layer n is denoted H_n.

The last line in equation (2.56) shows that given all the $\frac{\partial L}{\partial \mathbf{h}^{(n)}}$ from layer n we can compute also compute $\frac{\partial L}{\partial \mathbf{h}^{(n-1)}}$ in the previous layer $n-1$.
Hence the derivatives $\frac{\partial L}{\partial w_{ij}}, \frac{\partial L}{\partial b_j}$ for all parameters can be computed by successively passing the $\frac{\partial L}{\partial \mathbf{h}^{(n)}}$ derivatives backward through the network, explaining the name backpropagation.
This derivation only showcased the case where both $\mathbf{h}^{(n)}$ and $\mathbf{h}^{(n-1)}$ were hidden layers in a feed-forward network but it serves to illustrate the idea behind the backpropagation method.
It is possible to derive a vectorized form of equation (2.56) which formulates the equation in terms of matrix multiplications so that backpropagation can be done efficiently by taking advantage of hardware like GPUs just as was the case for the forward pass.

Optimizing the network now consists of
(1) Forward pass given input data $\mathbf{x}_i$.
(2) Computing derivatives using backpropagation.
(3) Using the derivatives to compute the parameter update

2.3.2.2 Mini-batches

Many machine learning problems, such as classification problems, consists of minizing a loss function L over a dataset. In the case of a image classification problem each datapoint $(\mathbf{x}_i, y_i)$ would consist of input image $\mathbf{x}_i$ and the corresponding true label y_i.
Computing the true gradient given in equation (2.57) of such a problem

$$\nabla_{\boldsymbol{\theta}} \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i; \boldsymbol{\theta})) \tag{2.57}$$

would require iterating trough the entire dataset $(\mathbf{x}_i, y_i)_{i=1}^N$ which for large dataset with large N would be unfeasible due to time and memory constraints.
The solution often opted for instead is computing the gradient over a smaller subset of the dataset $(\mathbf{x}_i, y_i)_{i=1}^n$ of size n, where n is much smaller than the dataset

size such that $n << N$. The gradient that is being computed is

$$\nabla_{\boldsymbol{\theta}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i; \boldsymbol{\theta})). \tag{2.58}$$

Since not all datapoints are used, this is not the true gradient of the problem but instead a noisy estimate.
Such a selection for datapoints of size n is often referred to as a mini-batch, and n is referred to as the batch size.

2.3.2.3 Optimization routines

As mentioned most of the common neural network optimization methods are first-order and only requires the derivatives $\frac{\partial L}{\partial \theta_k}$ as input.

Stochastic gradient descent (SGD) is a rather simple method defined by

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \alpha \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i; \boldsymbol{\theta}^{(t)})) \tag{2.59}$$

where n is the mini-batch size. The name *stochastic* in SGD derives from he fact that the gradients computed using mini-batches is a noisy estimate of the true gradient.

More sophisticated routines than SGD have been developed. One notable and commonly used method is Adam [KB14].
The Adam optimizer adapts the learning rate individually for each parameter using information from running estimates of the gradients $\mathbf{m}_t$ and squared gradients $\mathbf{v}_t$
At each step t, the running averages are updated using the gradient from step t, $\mathbf{g}_t = \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i; \boldsymbol{\theta}^{(t)}))$:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\mathbf{g}_t \tag{2.60}$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)\mathbf{g}_t^2 \tag{2.61}$$

Here, β_1 and β_2 are hyperparameters that is input into the Adam method at the start. β_1, β_2 are the decay rates of the gradient average and squared gradient average respectively. Since the running gradients are zero initialized they are inherently biased ([KB14]) and must be bias-corrected. The bias-corrected gradients $\hat{\mathbf{m}}_t$ and gradients squared $\hat{\mathbf{v}}_t$ are computed as:

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t} \tag{2.62}$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t} \tag{2.63}$$

The parameters of the model is update in the following way, where α is the learning rate:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} \tag{2.64}$$

Where ϵ is a small constant added to the denominator to avoid division by zero.

2.3.3 Convolutional neural networks

In a fully connected feed-forward network is shown in figure (2.3.1) the next layer $\mathbf{h}^{(n)}$ is computed from the the previous layer using a matrix multiplication

$$\mathbf{h}^{(n)} = \mathbf{W}^{(n)\top}\mathbf{h}^{(n-1)} + \mathbf{b}^{(n)}. \tag{2.65}$$

If layer n has k nodes and layer $n-1$ has l nodes the number of learnable parameters in the weight matrix $\mathbf{W}^{(n)\top}$ will be $l \times k$. If the layers in the neural network are large (large l, k) each layer requires a large amount of parameters. If for instance the dimensions of the layers $\mathbf{h}^{(n)}, \mathbf{h}^{(n-1)}$ is in the thousands the parameters in the weight matrix $\mathbf{W}^{(n)}$ will be in the millions.
This is one of the reasons alternative networks to the fully-connected feed-forward network from figure (2.3.1) have been developed.

A very common type of network, especially for image data, that have been developed is the convolutional neural networks (CNN).
A convolution can be thought of as an filter of given spatial dimensions sliding over a feature map. Mathematically this can be described as convolution of the feature map I and a filter K resulting in an output S [GBC16]

$$S(i,j) = (I \circledast K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{2.66}$$

The convolution operation occurs frequently in image processing, and many well-known convolutional filters for various image processing tasks exists.
Examples of such filters are the edge-detection filter and the gaussian blur filter.

The convolution operation potentially changes the spatial dimensions of the input feature maps on account locations (i,j) in the input image I parts of the filter K would extend beyond the border of I. Such cases could be handled by padding the input image I before applying the convolution. Both padding the input image with zeros and replicating the edge-values are common choices. The spatial dimensions of the output given the chosen padding, stride and filter size can be found by [PyT21]

$$\begin{aligned} H_{\text{out}} &= \left\lfloor \frac{H_{\text{in}} + 2 \times \text{padding} - \text{filter\_size}}{\text{stride}} + 1 \right\rfloor \\ W_{\text{out}} &= \left\lfloor \frac{W_{\text{in}} + 2 \times \text{padding} - \text{filter\_size}}{\text{stride}} + 1 \right\rfloor. \end{aligned} \tag{2.67}$$

In a convolutional layer in a neural network the filter weights $K(i,j)$ are learnable parameters that are optimized during network training. In this way the network can learn its own filters to extract features useful to solve a given task.
A convolutional layer in a neural network consist of a given number of filters each one producing an output map S as given by equation (2.66).

Convolutional layers have several benefits. They cost considerably less parameters than fully-connected layers, which is due to the fact that the convolution

filters work locally on a very restricted number of values from the input feature map I, contrasted with fully-connected layers with connections between each input unit and each output unit.
CNNs utilize the fact that features in image data are often local. Detecting edges for instance usually only requires local data in the input image.
They also feature parameter reuse in the sense that a filter can extract the same features at different spatial locations in the input image I.
For instance if a filter have learned to extract a feature like edges it be used to detect edges anywhere on the input image.

2.3.4 Neural network architectures

2.3.4.1 ResNet

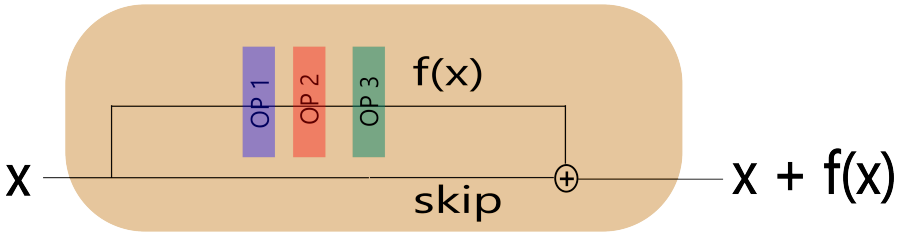


Figure 2.3.2: Resnet layer.

ResNets ([He+16]) were introduced to enable training of deep neural networks. One of the reasons why training deep neural networks can be challenging is that the learning signal has to be backpropagated through a lot of layers, where in each layer it would pick up multiplicative factors that could potentially be quite small. This could cause the learning signal to become very weak after passing backwards through many layers, resulting in the infamous vanishing gradient problem.
The main trick used in ResNets to mitigate this problem is to let there be a skip connection as illustrated in figure (2.3.2). The derivative of a layer, where the non-skip connection is represent by f is then given by

$$\frac{\partial(\text{layer})}{\partial x} = \frac{\partial}{\partial x}(x + f(x)) = 1 + f'(x). \tag{2.68}$$

As equation (2.68) shows, that even if $f'(x)$ turns out to be small the learning signal backwards will not be diminished.

2.3.4.2 DenseNet

DenseNets ([Hua+17]) is another approach to designing neural networks that does not suffer from the vanishing gradient problem.
DenseNets are organized into dense blocks. Each dense block contains a given number of convolutional layers. As shown in figure (2.3.3) dense nets concatenates channel-wise the output of each convolutional layer in the dense block to the current state. This leads to direct forward connections between every layer in the dense block. The direct connections help with gradient flow backwards during training and helps prevent vanishing gradients. DenseNet architecture encourages

feature reuse since it does not change the feature maps, but adds new feature maps to the current state.
The number of output channels from convolutional layer in a dense block that is concatenated to the state is referred to as the growth-rate k. If the input array has dimensions (C, W, H) and is passed through a dense-block of L layers with growth-rate k the output array will have dimensions $(C + L \cdot k, W, H)$.

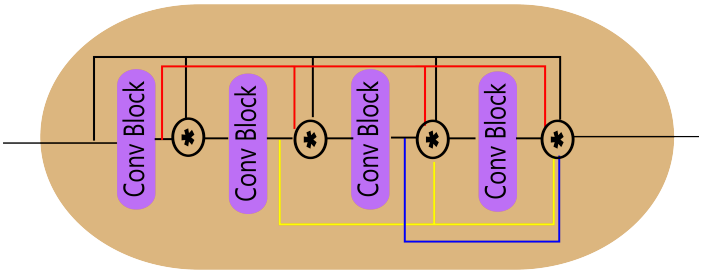


Figure 2.3.3: Dense block in a DenseNet architecture. Concatenation is denoted by $\circledast$.

2.3.5 Common layers in neural networks

2.3.5.1 Upsampling

Upsampling refers to increasing the spatial dimensions of the feature maps.
Upsampling can be implemented in neural networks in various ways such as transposed convolutions ([Eva16]) or pixelshuffle layers ([Shi+16]).
Another option is simply using nearest neighbour upsampling. The advantages of this approach is that no additional parameters is introduced, as opposed to tranposed convolution layers which contains learnable parameters.

2.3.5.2 Downsampling

Downsampling refers to decreasing the spatial dimensions of the feature maps.
One common way of doing this is by using a strided convolution. As shown in equation (2.67) we see that choosing stride $= 2$, padding $= 1$, filter size $= 3$ for the convolutional layer results in halving the spatial dimensions of the input array.
Other options include max-pooling and average pooling. Average pooling is a strided convolution where all the filter weights are given by $w_{ij} = 1/F^2$ for a filter of dimensions $F \times F$.

2.3.5.3 Instance normalization

Normalization layers are layers that normalize the input in various different ways. The goal of normalization layers is to ensure that the units have a similar distribution and value ranges.
Experience has shown that such normalization layers can make neural network training more stable and faster which is why the inclusion of various types of normalization layers has become common practice.

Instance normalization ([UVL16]) is a normalization layer that normalizes over the spatial dimensions of the input units.
Specifically if the input is of shape (C, W, H), where W, H represent the spatial dimensions instance norm computes normalization over each channel independently [UVL16]:

$$y_{c,i,j} = \frac{x_{c,i,j} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}},\ \mu_c = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} x_{c,i,j},\ \sigma_c^2 = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} (x_{c,i,j} - \mu_c)^2. \tag{2.69}$$

$y_{c,i,j}$ is the output of the instance normalization layer for input $x_{c,i,j}$. The index c runs over the channel-dimension and i, j indexes the spatial location in the feature maps. μ_c is the mean of the c-th channel and σ_c^2 is the variance of the c-th channel, and ϵ is a small constant added to avoid zero-division.

2.3.6 Generative adversarial networks (GANs)

Generative adversarial networks ([Goo+14]) are a class of generative models that consists of two neural networks, a generator G and a discriminator D.
The fundamental idea is to train the networks G, D in a adversarial way defined by the minimax equation [Goo+14]

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]. \tag{2.70}$$

z is randomly generated noise that is input into the generator with the interpretation that the generator is sampling from an latent distribution.

Put in words the generator network G in a GAN tries to generate output that will maximize the probability that the discriminator network D classifies the output as real. The discriminator network will output a probability that estimates whether an input is real or fake.
The hope is that this will give us a generator network G that can produce realistic data $\mathbf{x}$ indistinguishable from the real distribution p_{data}.

The loss functions for the generator network becomes

$$L_{\text{gen}} = \log(1 - D(G(z))) \tag{2.71}$$

and the loss function for the discriminator becomes

$$L_{\text{disc}} = -\log D(x) - \log(1 - D(G(z))). \tag{2.72}$$

This gives us the following gradients used for updating the generator and discriminator network:

$$\begin{aligned} &\nabla_{\theta_G} \frac{1}{n} \sum_{i=1}^{n} \log(1 - D(G(z))) \\ &\nabla_{\theta_D} \frac{-1}{n} \sum_{i=1}^{n} [\log D(x) + \log(1 - D(G(z)))] \end{aligned} \tag{2.73}$$

where n is the mini-batch size. The log-probability can be used instead of the probability since the parameters maximizing the log-probability is the same paramters that maximizes the probability.

2.3.6.1 LSGAN

Least squares generative adversarial networks (LSGAN) is a purposed ([Mao+17]) modification of regular GANs, where the loss function for the discriminator is a least squares loss function. This means that the output from the discriminator in an LSGAN is interpreted as a real number and not a probability.
The loss functions for the generator and discriminator can be chosen to be

$$L_{\text{gen}} = [D(G(z)) - 1]^2 \tag{2.74}$$

for the generator and

$$L_{\text{disc}} = \frac{1}{2}[D(x) - 1]^2 + \frac{1}{2}[D(G(z)) - 0]^2 \tag{2.75}$$

for the discriminator.
With this choice of loss functions the discriminator tries to optimize its parameters so that it outputs 1 for real data and 0 for fake data.
LSGANs was shown in [Mao+17] to lead to better and more stable learning than regular GAN for certain problems and represents an alternative method to regular GANs that is often easier to train.

CHAPTER

THREE

CREATING A FP IMAGING DATASET USING STOCK IMAGE DATA

Training neural networks often requires considerable amounts of training data. State of the art generative models are often trained on datasets containing millions of datapoints. Gathering real training data for fourier ptychography reconstruction would be time consuming endeavour since each datapoint in the dataset would involve imaging the object from many different incident directions, often numbering in the hundreds. This chapter presents a way of generating a dataset that can by used to train neural networks for fourier ptychography reconstruction. Using synthetic data constructed from simulation of the imaging process and stock-image data as input is an alternative way of creating training data removing the need to gather real training data. Datasets with stock image data are numerous and available since computer vision problems and tasks have always been at the forefront of tasks that one is interested in automating with machine learning models.

3.1 Generating a synthetic FP training set from stock images

For real FP imaging conducted in the lab the sample is illuminated sequentially from a set of incident directions $\{(k_{xn}, k_{yn})\}$. These incident directions is defined by the relative angle compared to the the on-axis illuminating source. As has been discussed in section (2.2) each of these different incident illumination directions (k_{xn}, k_{yn}) gives rise to shifts in the Fourier-domain such that

$$\mathcal{F}\{O_{\text{in}}\}(\nu_x, \nu_y) = H(\nu_x, \nu_y)\mathcal{F}\{O_{\text{out}}\}(\nu_x - k_{xn}, \nu_y - k_{yn}).$$

Combined with the fact that the objective lens of the imaging system acts as a low-pass filter, i.e that the transfer function of the optical system has the property $H(\nu_x, \nu_y) \sim p(\nu_x/w, \nu_y/w)$ for some cutoff frequency w, then each illumination direction corresponds to circular disk in the Fourier spectrum of $O_{\text{in}}(x, y)$. The cutoff frequency, i.e the radius of the extracted disks, is chosen to be $k_0 \cdot \text{NA}_{\text{obj}}$ as discussed in section (2.1.6).

Simulation of FP imaging can then be done by extracting these disks from a high-resolution object $O_{\text{in}}(\nu_x, \nu_y)$ and using the extracted disks to create the simulated low-resolution intensity images.

3.1.1 FP simulation algorithm

The method for simulating the forward imaging process is adapted from the one suggested in [Zhe16, pp. 2.3–2.6].
A complex object $O_{\text{in}}(x, y)$ is constructed from an amplitude $A(x, y)$ and a phase $\phi(x, y)$. The values for the amplitude $A(x, y)$ and the phase $\phi(x, y)$ arrays is chosen to be from a dataset of grayscale images. Explicitly if we want to construct a complex object of dimensions (m, n), then $A(x, y), \phi(x, y)$ will be real arrays of dimensions (m, n) with their values given by a grayscale images randomly selected from a dataset. The complex object is then formed as $O_{\text{in}}(x, y) = A(x, y) \exp[i \cdot \phi(x, y)]$.
With a high-resolution complex object constructed the simulation of the imaging is done by
1.Computing the fourier transform of $\mathcal{F}\{O_{\text{in}}(x, y)\}$
2.Extracting circular patches $\{D_n\}$ of radius $k_0 \cdot \text{NA}_{\text{obj}}$ from $\mathcal{F}\{O_{\text{in}}(x, y)\}$ for a given set of shifts $\{(k_{xn}, k_{yn})\}$.
3.Compute the low-resolution intensity images $|\mathcal{F}^{-1}\{D_n\}|^2$ from the extracted patches.
For each shift (k_{xn}, k_{yn}) there has now been created a corresponding low-resolution image. The set of created low-resolution images $\mathbf{I}_{\text{LR}}$, combined with the high-resolution complex object $A(x, y) \exp[i \cdot \phi(x, y)]$ now constitute one datapoint $(\mathbf{I}_{\text{LR}}, [A(x, y), \phi(x, y)])$ in the training dataset. The simulation procedure is visualized in figure (3.1.1).

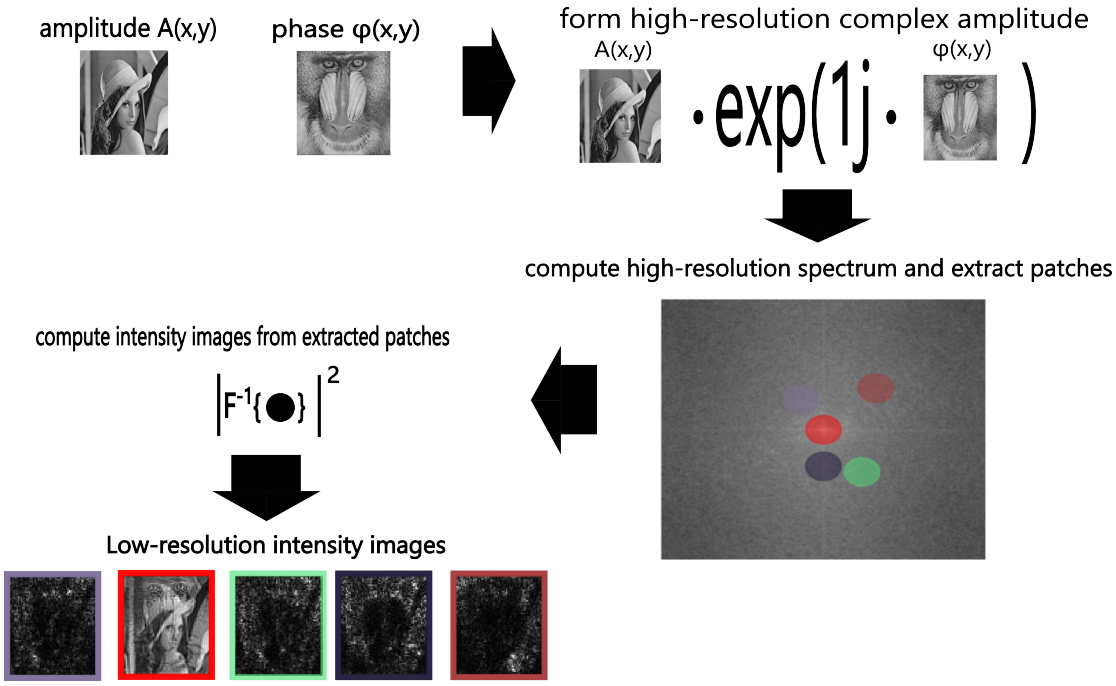


Figure 3.1.1: FP imaging simulation.

3.1.2 Dataset creation

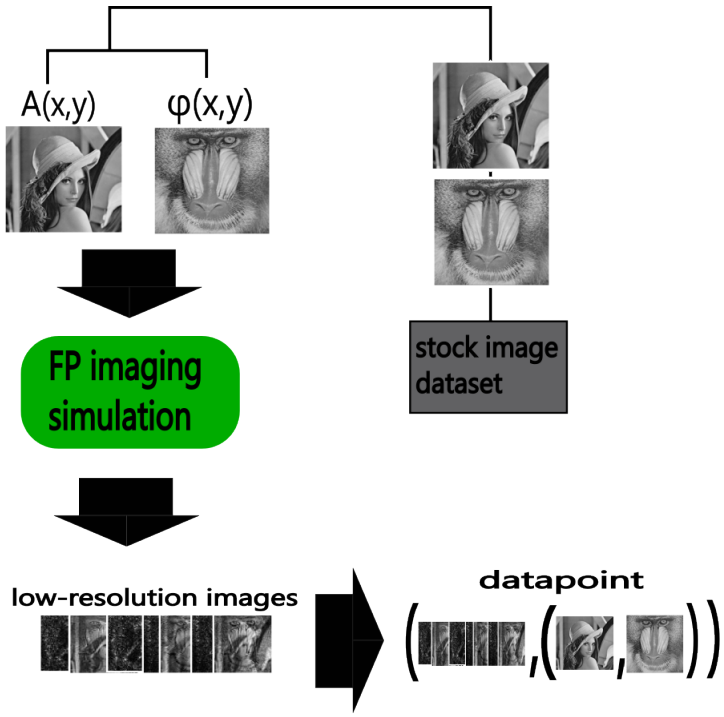


Figure 3.1.2: Synthetic FP dataset creation.

The forward FP simulation method described in the previous section showed how to create a datapoint given two image arrays that are identified as $A(x, y), \phi(x, y)$. In this way an entire dataset for FP reconstruction can be made from an set of stock images obtained from a dataset as illustrated in figure (3.1.2).
Each FP datapoint $(\mathbf{I}_{\mathrm{LR}}, [A(x, y), \phi(x, y)])$ contains a set of low-resolution images and two high-resolution images, making the training data memory intensive and requiring large storage capacity for the dataset sizes requires to train a neural network of the required capacity. This means that the dataset needs to be continuously generated during training time.

CHAPTER

FOUR

FOURIER PTYCHOGRAPHIC RECONSTRUCTION USING NEURAL NETWORKS

The goal of a FP reconstruction is using the information from the set of low-resolution intensity images $\mathbf{I}_{\text{LR}}$ to recover the high-resolution complex object $O(x, y) = A(x, y) \exp(i \cdot \phi(x, y))$, where $A(x, y)$ is the object amplitude $\phi(x, y)$ is the object phase.
The input to the neural network will therefore be the set of low-resolution FP images $\mathbf{I}_{\text{LR}}$.
Using the set of low-resolution images as input the task of the neural network will be to generate a reconstructed high-resolution object $\tilde{O}(x, y)$.
Many neural network designs for the FP reconstruction task choose to output two real arrays $\tilde{A}(x, y)$,$\tilde{\phi}(x, y)$, representing the reconstructed amplitude and the reconstructed phase of the high-resolution complex object, instead of outputting a complex-valued array $\tilde{O}(x, y)$. This is also the approach taken in this project.
As suggested in figure (4.0.1) a neural network for reconstructing the high resolution object from a number of C low-resolution images will thus be a function:

$$G_{\boldsymbol{\theta}} : \mathbb{R}^{C \times m \times n} \to \mathbb{R}^{2 \times M \times N} \tag{4.1}$$

where $\boldsymbol{\theta}$ represent the network parameters, (m, n) are the spatial dimensions of the low-resolution images, (M, N) are the spatial dimensions of the output high-resolution objects. The two in the output dimensions $(2, M, N)$ stems from the fact that the neural network outputs two reconstructed high-resolution arrays $\tilde{A}(x, y), \tilde{\phi}(x, y)$ of spatial dimensions (M, N). The reconstructed complex object is then formed as $\tilde{O}(x, y) = \tilde{A}(x, y) \exp\Big(i \cdot \tilde{\phi}(x, y)\Big)$.

4.1 Patchwise reconstruction of images

Large image sizes make neural network operations very memory expensive, especially for training but also for inference. It is not uncommon for fourier ptychography that the low-resolution intensity images have spatial dimensions in the thousands, for instance (1024×1024) or (2048×2048). This makes working on the full-size images unfeasible due to the large amount of GPU memory it would require. This is especially true during network training when intermediate results

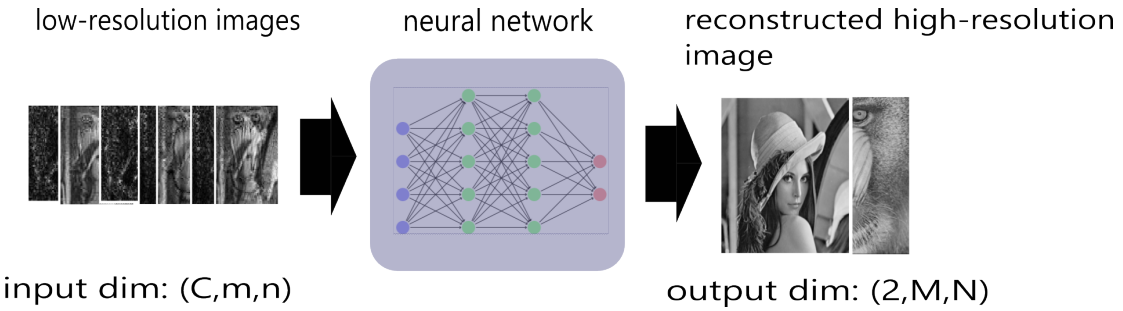


Figure 4.0.1: Reconstruction of low-resolution intensity images by neural network.

would have to be stored for gradient computation.
The solution to this problem will be to reconstruct the high-resolution image patch-wise. Specifically, given an upscale factor r, the neural network $G_{\boldsymbol{\theta}}$ would take in inputs of shape (C, m, n) and output reconstructed patches $(2, r \cdot m, r \cdot n)$.

As a concrete example, consider an imaging setup using a camera with a resolution of $(2048) \times (2048)$ and using a 15×15 LED array for illuminating the object, resulting in $15^2 = 225$ low-resolution images. Assume an upscale factor of $r = 4$. Reconstruction patch-wise could be done by choosing patches of spatial dimensions $(64, 64)$ for input into the neural network. Then the neural network would take as input patches of shape $(15^2, 64, 64)$ and output patches of shape $(2, 256, 256)$.
Reconstruction of the entire image would by done by a sliding window, where the window has spatial dimensions of 64×64.
Reconstructing the high-resolution object patch-by-patch in this particular example would require $(2048/64)^2 = 1024$ patches passed through the neural network. Its important to note that this would not require 1024 separate passes since neural network software and GPUs are able to do this in parallel.

4.2 Loss functions for FP reconstruction

The loss function defines the minimization problem the neural network parameters are optimised after. Explicitly we are seeking the network $G_{\boldsymbol{\theta}^*}$ with parameters $\boldsymbol{\theta}^*$ such

$$\boldsymbol{\theta}^* = \arg\min_{\theta} \sum L(G_\theta(\mathbf{I}_{\mathrm{LR}})) \tag{4.2}$$

where L is the chosen loss function to minimize and $\mathbf{I}_{\mathrm{LR}}$ is the input low-resolution intensity images.
The quality of the reconstruction results one obtains with a trained generator network $G_{\boldsymbol{\theta}}$ is highly dependent on choosing a good loss function since it defines the optimization criteria for the parameters $\boldsymbol{\theta}$.
The network is designed to output the reconstructed complex valued high-resolution object, with the high-resolution object being represented with separate channels for the amplitude and the phase, so $G(\mathbf{I}_{\mathrm{LR}}) = (\tilde{A}(x, y), \tilde{\phi}(x, y))$. The network output will thus be arrays having the shape $(2, M, N)$ where M, N is the spatial dimensions of the high-resolution complex image.

4.2.1 Distance loss

The most obvious choice of loss function is arguably using the distance between the reconstructed high-resolution image and the true high-resolution image as the loss.
Distance can be measured in different ways, but the most natural and commonly used would be the mean squared error (MSE) or the mean absolute error (MAE). We refer to these distance measures as the L_2 loss and L_1 loss respectively.
The L_2 loss (MSE) is defined as squared differences between the corresponding pixel values in the reconstructed and true images, divided by the total number of pixels:

$$L_2 = \frac{1}{MN} \sum_{x,y} [p(x,y) - \tilde{p}(x,y)]^2 = \|p(x,y) - \tilde{p}(x,y)\|_2 \tag{4.3}$$

Similarly, the L_1 loss (MAE) is defined :

$$L_1 = \frac{1}{MN} \sum_{x,y} |p(x,y) - \tilde{p}(x,y)| = \|p(x,y) - \tilde{p}(x,y)\|_1 \tag{4.4}$$

in both cases M, N is the spatial dimensions of the images.

4.2.2 Adversarial loss

In the framework of standard GANs one can define an adversarial loss for the generator G as

$$L_{\text{adv}} = \log [1 - D(G(\mathbf{I}_{\text{LR}}))]. \tag{4.5}$$

or in the case of LSGANs as discussed in (2.3.6.1):

$$L_{\text{adv}} = [1 - D(G(\mathbf{I}_{\text{LR}}))]^2. \tag{4.6}$$

In both cases the adversarial loss measures how good the generator network G is at fooling the discriminator network D.
Adversarial losses like defined in equations only encourages the generator network G to output images of high visual quality, but does not enforce the images to be solutions to the FP reconstruction problem and therefore needs to be used in combination with other loss function that enforce closeness to the solution of FP reconstruction problem.

4.2.3 Composite loss functions for FP reconstruction

Empirically it is often the case that using using only distance losses such as L_2 or L_1 losses for image reconstruction problems usually does not lead to high-quality results. It is frequently found in practice that the solutions obtained from only using L_2 or L_1 loss functions in the neural network training leads to blurry images, due to the tendency L_2 and L_1 losses has of encouraging averaged solutions. Reconstructed images appear blurry when the highest frequency components of the target image have not been recovered. Stated another way; when trained with

only L_2 or L_1 losses the neural network fails to reconstruct the highest frequencies in the image and lacks finer details. This is detrimental to the image quality and can render the reconstructed images unusable.
The aforementioned reasons motivates using a composite loss function incorporating both a distance loss, such as the L_1 or L_2 loss and an adversarial loss.
Such a loss function can be given as a weighted sum of the distance loss function and the adversarial loss function:

$$L = w_1 L_1 + w_2 L_{\text{adv}}. \tag{4.7}$$

The L_1 distance term ensures that the reconstructed image is close pixel-wise to the real image and the adversarial loss L_{adv} encourages reconstruction of the higher frequency part by also training the generator network G to produce reconstructed images that are hard to distinguish from the true value. Typically the L_1 term would be weighted higher than the L_{adv} since the reconstructed solutions staying close to the true values pixel-wise is the most important criteria for good solutions.

4.3 Network architectures

The network architecture is an important parameter for the performance of the network. This section presents the chosen architectures for both the generator and discriminator networks.

The input to the generator networks is the collected set of low-resolution intensity images $\mathbf{I}_{\text{LR}}$, which is a real array of dimension (C, m, n) where C is the number of low-resolution images and (m, n) is the spatial dimensions of low-resolution images.
To avoid confusion it is again mentioned that since the reconstruction is done patch-wise, so (m, n) will be the spatial dimensions of a chosen patch size. The output of the generator network will be high-resolution patches of dimension $(2, M, N)$ where $(M, N) = (r \cdot m, r \cdot n)$. Upper bounds for the upscale factor r is discussed in section (2.2.2).

The discriminator only evaluates the perceptual quality of the output images, not how well the output solves the FP reconstruction problem. The input to the discriminator network is therefore the reconstructed high-resolution image patches of dimension $(1, M, N)$.

4.3.1 Generator networks

Two architectures for the generator network is explored in this project.

The first is a rather deep ResNet network as illustrated in figure (4.3.1). The architecture of this network is inspired by the generator network architecture from the paper [BDS18]. The 'non-local' block in the network architecture is defined the same way as in [BDS18]. This network outputs only a single high-resolution image of shape $(1, M, N)$ so two networks are needed to reconstruct the high-resolution complex object $O(x, y)$, one for reconstructing the amplitude $A(x, y)$ and another

network for reconstructing the phase $\phi(x, y)$.
The ResNet network used with architecture as given in figure (4.3.1) had 41.08 M parameters.

The other generator architecture explored is a DenseNet ([Hua+17]) with a U-net architecture ([RFB15]) adapted from the paper [Lu+21]. This generator architecture reconstructs the amplitude $A(x, y)$ and the phase $\phi(x, y)$ simultaneously and hence outputs an array of dim $(2, M, N)$.
The DenseNet network used with architecture as given in figure (4.3.2) had 68.32 M parameters.

4.3.1.1 ResNet generator

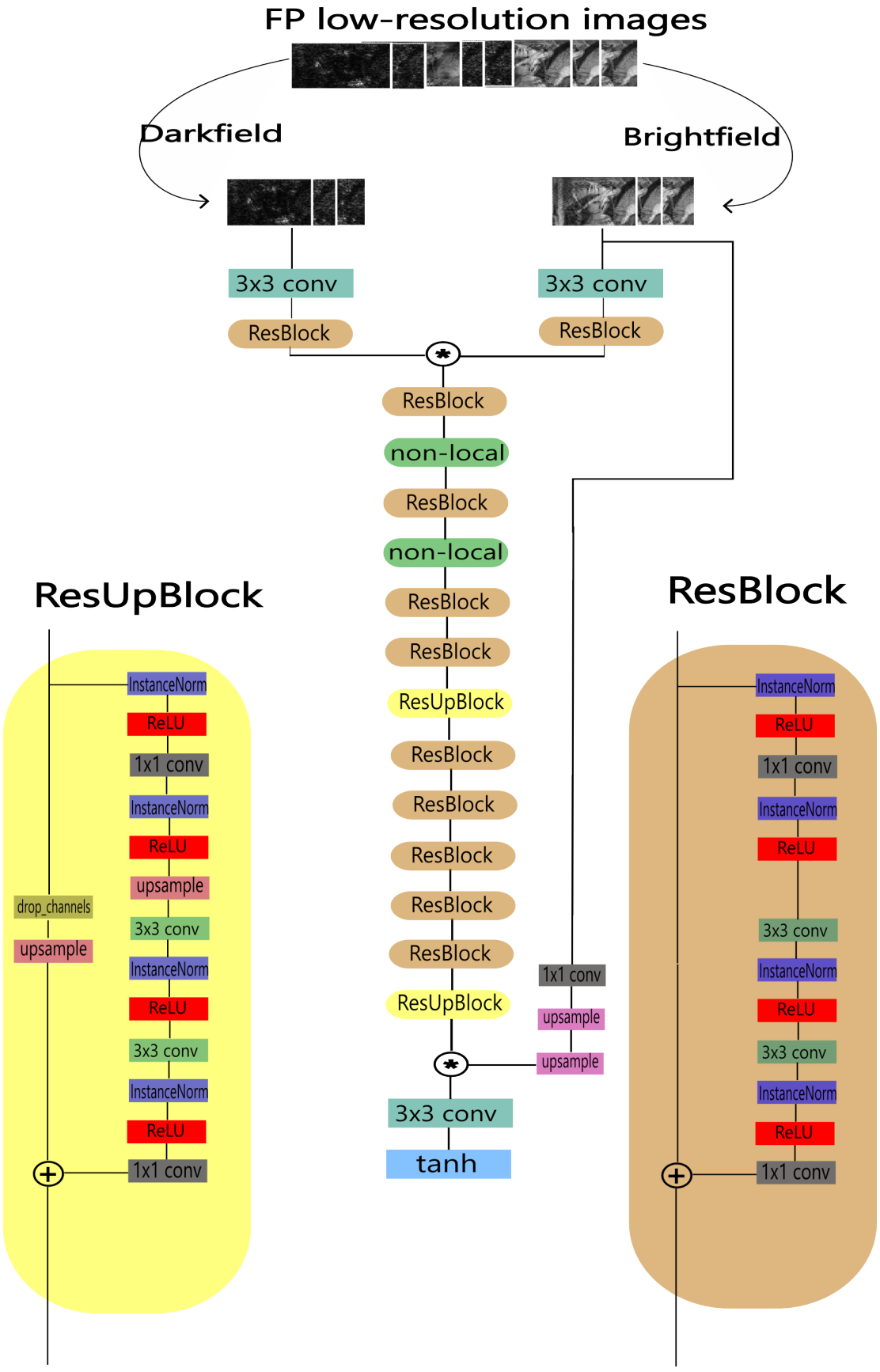


Figure 4.3.1: ResNet generator network architecture.

4.3.1.2 DenseNet generator

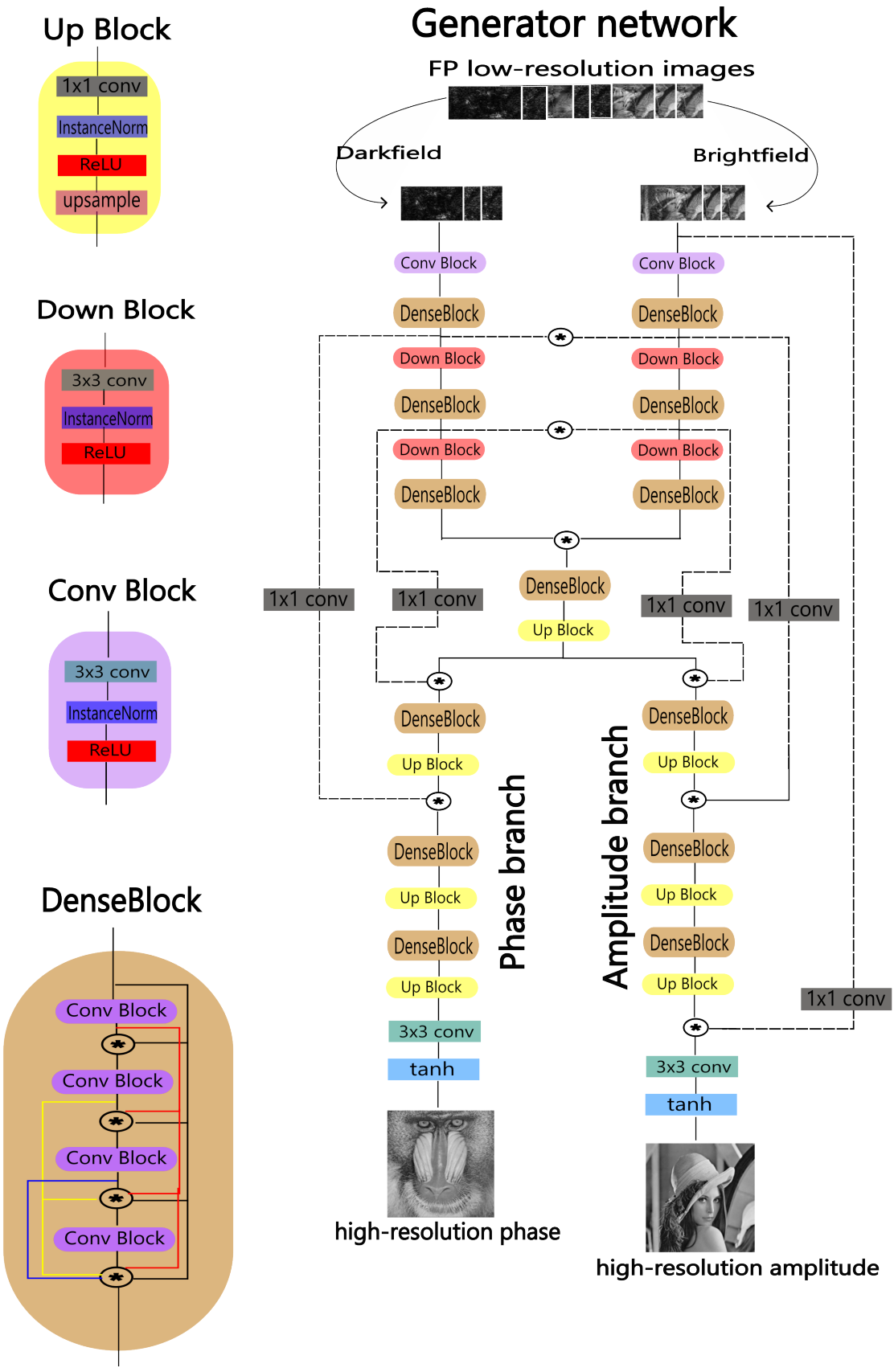


Figure 4.3.2: DenseNet generator network architecture.

4.3.2 Discriminator networks

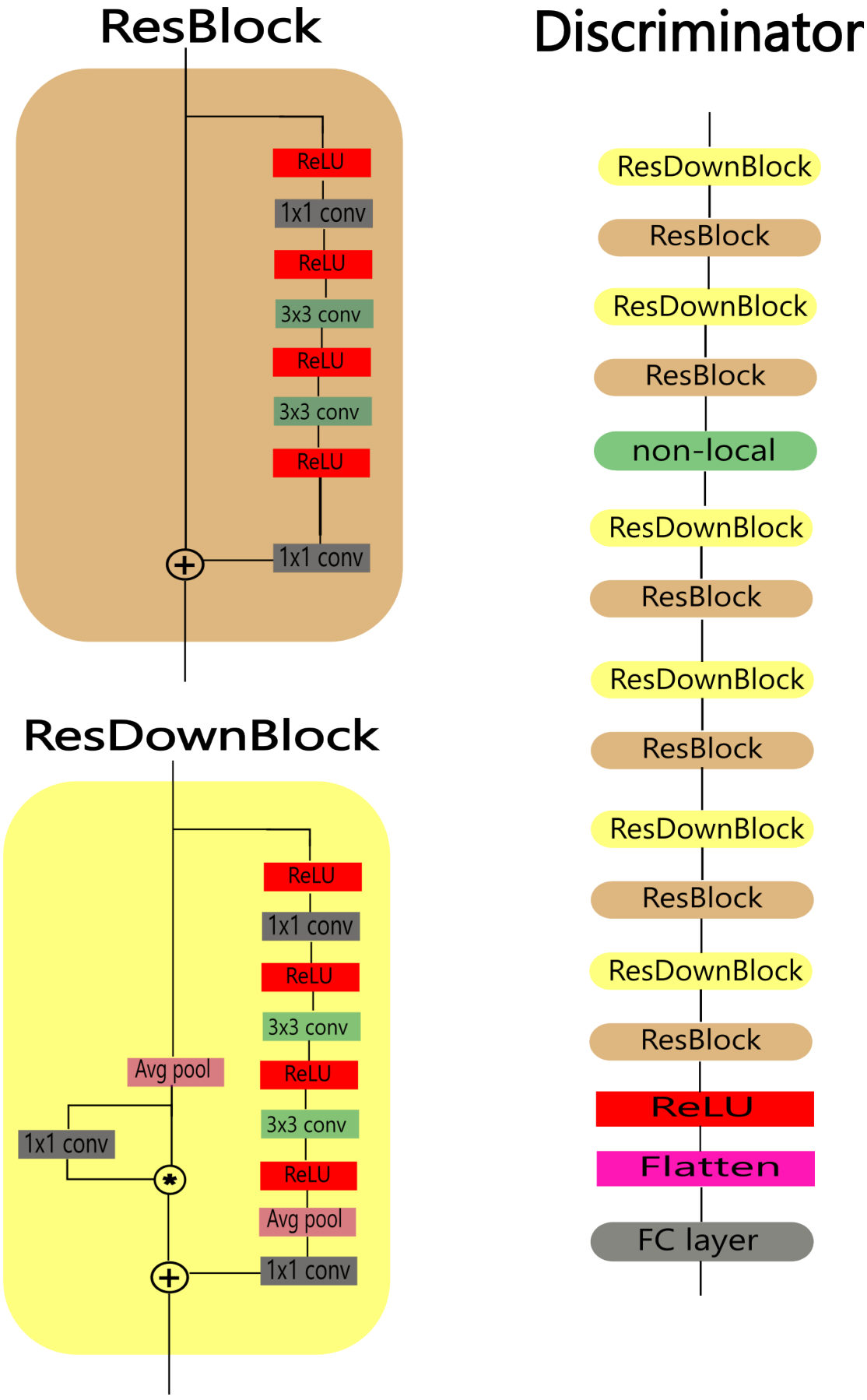


Figure 4.3.3: Discriminator network architecture. The network layer 'non-local' is defined the same way as in [BDS18].

4.4 Network training

As described in section (3.1.2) the training data is generated continuously as needed using the simulation method from section (3.1.1). As was also discussed, the arrays $A(x,y), \phi(x,y)$ which is used to create the stimulated FP imaging datapoints $(\mathbf{I}_{\text{LR}}^{(i)}, [A(x,y), \phi(x,y)]^{(i)})$ will be chosen to be stock images. The arrays $A(x,y)$ and $\phi(x,y)$ will be grayscale images read in from a chosen dataset. The FP datapoints $(\mathbf{I}_{\text{LR}}^{(i)}, [A(x,y), \phi(x,y)]^{(i)})$ are continuously generated as needed using stock images as values for $A(x,y), \phi(x,y)$. The grayscale images are scaled into proper ranges, $[0,1]$ for $A(x,y)$ and $[0,2\pi]$ for $\phi(x,y)$ before begin input into the FP simulation method.

It is likely that the datasets chosen will have a large impact on the performance of the final network. One might hypothesize that a more diverse training dataset leads to a network that is better at generalizing since it has learned to output a greater range of image classes, hence a rather large diverse dataset is chosen.
The dataset used is the ILSVRC2014 dataset from the 'ImageNet Large Scale Visual Recognition Challenge 2014' competition. The dataset contains 456567 RGB images of 200 different classes.
The images in the datasets is converted to grayscale, resized to spatial dimensions $(256, 256)$ and separated into a training set and a validation set with a 85%-to-15% split.
The FP image simulation method are set to simulate FP imaging with a 15x15 LED array, corresponding to $15^2 = 225$ different (k_{xn}, k_{yn}) shifts and 225 low-resolution intensity images $\mathbf{I}_{\text{LR}}$. An upscale factor of $r = 4$ is chosen, which gives means low-resolution images have spatial dimensions $(64, 64)$.
Summarising; the input to the neural network will be arrays $\mathbf{I}_{\text{LR}}$ of dimensions $(225, 64, 64)$, the neural network will be trained to reconstruct the high-resolution amplitude $A(x,y)$ and phase $\phi(x,y)$ of spatial dimensions $(256, 256)$ and hence the dimensions of the output will be $(2, 256, 256)$.

Some preprocessing of the datapoints $(\mathbf{I}_{\text{LR}}^{(i)}, [A(x,y), \phi(x,y)]^{(i)})$ is applied before training starts.
The low-resolution images $\mathbf{I}_{LR}$ are normalized before being input into the generator netowork G:

$$\mathbf{I}_{\text{LR}}^{\prime(i)} = \frac{\mathbf{I}_{\text{LR}}^{(i)} - \mu_{\mathbf{I}_{\text{LR}}}}{\sigma_{\mathbf{I}_{\text{LR}}}}. \tag{4.8}$$

All generator network ends in a tanh-function which has range $[-1, 1]$ as it is a convenient way to constrain the output. For this reason image arrays $A(x,y)$,$\phi(x,y)$ are mapped into $[-1, 1]$ before being input to the loss-function. This does not lose generality since the arrays can be re-scaled back to the proper ranges when needed.

The loss function used is a weighted sum of L_1 loss and an adversarial GAN-loss L_{adv}, $L = w_1 L_1 + w_2 L_{\text{adv}}$ as justified in section (4.2.3). The values used are $w_1 = 100, w_2 = 1$.
The adversarial loss arising from the GAN means that a discriminator network D will be trained alongside the generator network G.

The networks are trained using a batch size of 3. The training is divided into epochs where each epoch contains 300 parameters updates. The initial learning rate is set to 1.5×10^{-5} and after the first 150 epochs the learning rate is decayed by multiplying the learning rate by 0.9 every 10 epochs.
The optimization routine used is Adam ([KB14]) for both the generator and discriminator networks.
The learning of the discriminator is slowed down by multiplying the discriminator loss by 0.5, as suggested in [Iso+17].
This is to prevent the discriminator for overpowering the generator too much. We hypothesise that since the generator has to minimise the L_1 loss in addition to the GAN adversarial loss L_{adv} it makes the task of learning to fool the discriminator harder, and hence it might be beneficial to slow down the learning of the discriminator network.

4.5 Related work

The use of neural networks for fourier ptychographic reconstruction has already accumulated an extensive body of work. A variety of different methods have been explored, with many different choices when it comes to network architectures, training data and loss functions.

The approach taken in this project has been inspired by the previous work [Lu+21] and their work on the use of neural networks for FP reconstruction. Similarly to the work done in this thesis project they use simulation of the imaging process to generate the training data used for network optimization. Their purposed network is a U-net with dense connections similar the network shown in figure (4.3.2). One of the main features of their network architecture is the splitting up of the dark-field and bright-field images and processing these images independently for the first layers in the network.The intuition behind this is that the intensity values of the bright-field images are quite different form the dark-field images. This results in a quite large dynamic range for the input data if they are input into the neural network on equal basis as one array and could be detrimental to the learning process. The idea of separating the dark-field and bright-field images has also been applied to the network architectures used in this project.
Another similarity as mentioned is the use of simulation of the FP imaging process for constructing a training data set. In [Lu+21] random patches from a set of ground truth high-resolution images is extracted. The random patches is then used in a forward FP imaging simulation, obtaining 225 low-resolution FP images. The approach taken to create a dataset is thus similar to the one used in this project, with the difference being that instead of using ground-truth high-resolution objects obtained from experiment, stock images are put into the FP simulation algorithm. There is a key difference in that the training set constructed in this project is purely synthetic in contrast to the work by [Lu+21] in which the training data stems from real experimental FP images.

[Ngu+18] reconstructs FP microscopy images of live-cells with a deep learning approach. For training data they use a single FPM dataset (one set of low-resolution

images and corresponding high-resolution images) where the high-resolution ground truth images have been reconstructed with an iterative solution method. The dimensions of the high-resolution images are 12800×10800 with low-resolution images of dimensions 2560×2160, corresponding to an upscale factor of $r = 5$. The training dataset is created by selecting random crops from the ground-truth high-resolution phase image along with the corresponding patches from the low-resolution intensity images. Each cropped patch then gives a data-point (X, Y) for use network training
. The loss function described in their paper is quite extensive, being a weighted sum of several components other than the usual L1-loss function. Similarly to many other approaches to reconstructing FP images with NNs, they also choose to incorporate an adversarial component in the loss function to avoid blurry output images. Specifically they train a cGAN where the conditional discriminator computes $D(G(I)|I)$ where x is the input low-resolution FP images. In words the discriminator outputs a probability that the output of the generator $G(I)$ is real given the low-resolution images I. In addition to the adversarial loss form the discriminator they include an L1 loss in the fourier domain between reconstructed high-resolution images and the ground truth high-resolution images, to further improve the reconstruction of higher frequencies and avoid blurry output images. Concretely the loss function is given as a weighted sum of individual loss functions

$$l = \lambda_1(l_{\text{MAE}} + l_{\text{FMAE}}) + \lambda_2 l_G + \lambda_3 l_{\theta_G} \tag{4.9}$$

with the individual loss functions given by

$$\begin{aligned} l_{\text{MAE}} &= \frac{1}{r^2WH}\|\phi - G(I)\|_1 \\ l_{\text{FMAE}} &= \frac{1}{r^2WH}\|\mathcal{F}\{\phi\} - \mathcal{F}\{G(I)\}\|_1 \\ l_G &= -\log_{\theta_G}(D(G(I)|I)) \\ l_{\theta_G} &= -\|\theta_G\|_2 \end{aligned} \tag{4.10}$$

l_{MAE} is the l_1 pixel-wise loss, l_{FMAE} is the l_1 distance in the fourier domain between the reconstructed and ground truth, l_G is the adversarial loss and $l_{\theta_G} = -\|\theta_G\|_2$ is an l2-regularization term often used when training neural networks.

CHAPTER

FIVE

RESULTS AND DISCUSSION

5.1 Reconstruction of synthetic FP images

The two different networks was trained on a simulated dataset as described in section (3.1.2).
This section presents how well the neural networks are able to reconstruct unseen FP datapoints derived from the validation dataset, i.e datapoints which the neural network has not been trained on.
The FP datapoints are obtained by by the simulation procedure given in section (3.1.1), drawing the $A(x, y), \phi(x, y)$ input from the validation set.

The parameters for the forward FP simulation procedure used to create the network training data is selected in such a way that the k-shifts $\{(k_{nx}, k_{ny})\}_{i=1}^{225}$ are the same as the k-shifts produced by the experimental parameters given in table (5.2.1). This enables the network trained on the resulting simulated data dataset to also reconstruct real FP image data for the real dataset presented in section (5.2), and hence the same trained networks are used in both sections.

Table 5.1.1: Network Performance in terms of average L1 loss, average Peak Signal To Noise (PSNR) and average Structural Similarity Index Measure (SSIM).

Network	L1 Loss (avg)	PSNR (avg)	SSIM (avg)
ResNet phase	0.1588	28.73 dB	0.6222
ResNet amp	0.0710	31.43 dB	0.7466
DenseNet phase	0.1226	29.30 dB	0.6977
DenseNet amp	0.0722	31.36 dB	0.7441

The performance metrics in table (5.1.1) was calculated by creating 20,000 FP datapoints using stock images from the validation dataset, hence all datapoints were previously unseen by the generator networks.
As can be seen from table (5.1.1) the performance of both network architectures are very similar.The reconstructed phase images are of lower quality than the reconstructed amplitude. It is not unexpected that recovering the phase is a more

difficult problem than recovering the amplitude. Furthermore a lot of information about the high-resolution amplitude is already present in the on-axis low resolution intensity image since the high-resolution is usually similar to an upscaled version of the on-axis image.

The most important criteria for the reconstructed images is closeness to the true values in the L1 or L2 sense, i.e that the distance between the reconstructed image and true image is small. It is however also important that the quality of the reconstructed images are high in terms of noise-level, sharpness and visible details. Figure (5.1.1) shows reconstruction results for FP-datapoints simulated by the FP simulation method with input $A(x, y), \phi(x, y)$ randomly selected from the validation dataset.
The on-axis low resolution intensity images is up-scaled to the same dimensions 256×256 using nearest neighbourhood up-sampling and shown for context.
A qualitative inspection of the reconstructed images in figure (5.1.1) shows that the reconstructed images $\tilde{A}(x, y), \tilde{\phi}(x, y)$ have a strong resemblance with the input $A(x, y), \phi(x, y)$, and are hard to distinguish at the image scale used in figure (5.1.1). The low-level features in the images seems to be recovered very well.
The PSNR metric also indicates that the images have been reconstructed fairly accurately. The reconstructed amplitude images $A(x, y)$ have values roughly in the range $30\,\mathrm{dB} - 34\,\mathrm{dB}$ while the reconstructed phases $\phi(x, y)$ have lower PSNR scores in the range $28\,\mathrm{dB} - 30\,\mathrm{dB}$ which indicates lower image quality compared to the reconstructed amplitude. This suggest that the image quality of the reconstructed amplitude is higher than the image quality of the reconstructed phase, but this is not immediately apparent when viewing the images at the selected scale in figure (5.1.1).

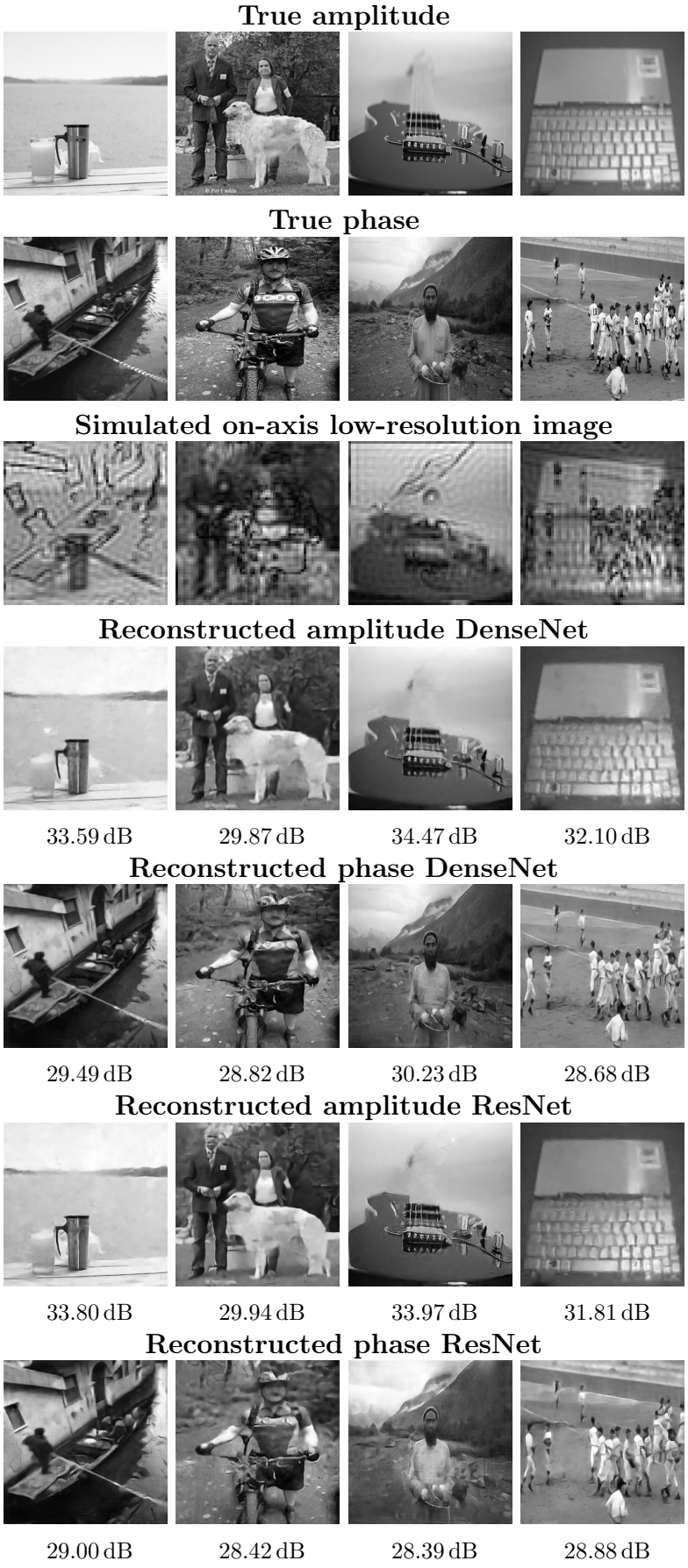


Figure 5.1.1: Reconstruction of simulated FP data randomly selected from the validation dataset. Reconstructed images are captioned with the Peak Signal to Noise (PSNR) score.

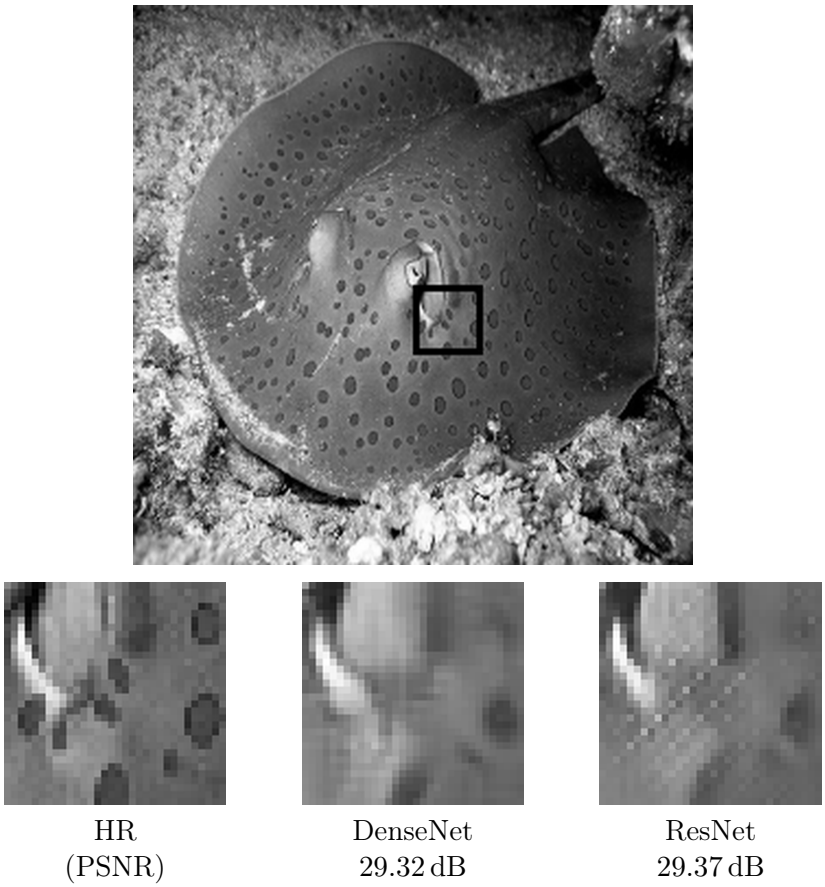
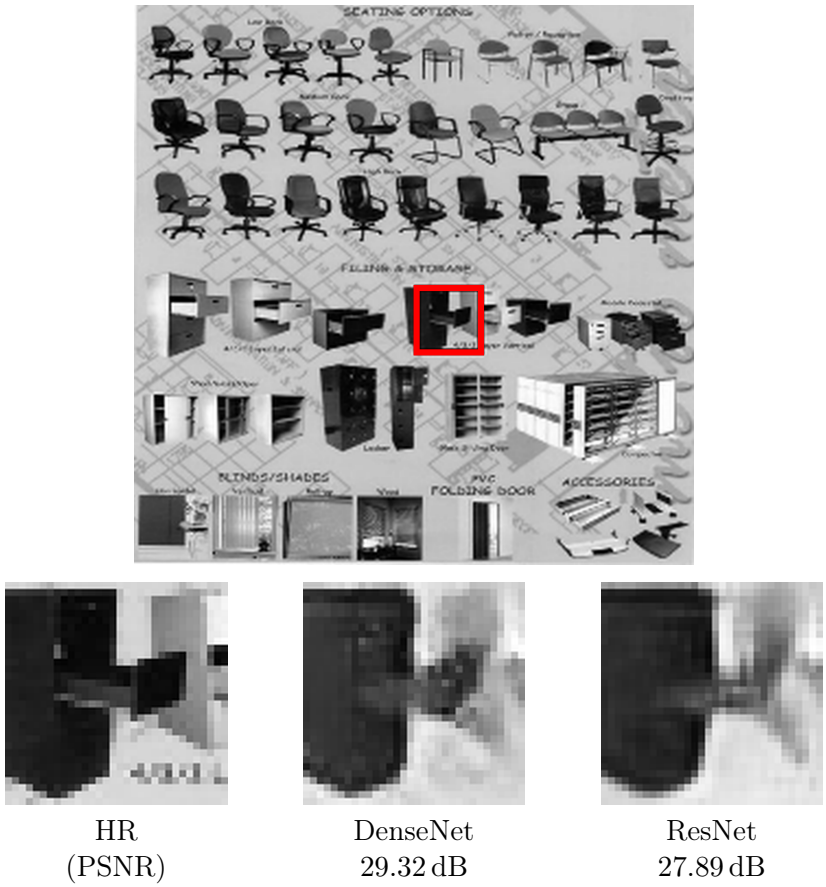


Figure 5.1.2: Zoomed in view of amplitude image reconstruction patches.

Figure (5.1.2) shown the reconstructed amplitude by the DenseNet and ResNet for a particular image part of the image as indicated by the square in the large high-resolution image.
Inspecting the upscaled patch of the reconstructed high-resolution amplitude image reveals that some of the finer details are lost in the amplitudes reconstructed by the neural networks. The amplitudes output by the neural networks furthermore looks more noisy and blurred. As discussed earlier the blurry images indicate that the higher-frequency components of the spectrum of the image $A(x, y)$ have not been fully recovered.
A similar comparison for a phase reconstruction is shown in figure (5.1.3). Figure (5.1.3) shows the reconstruction results for a FP-datapoint generated from randomly selected validation data. The observations made about the image equality of the reconstructed phase images mirror those made about the reconstructed amplitude image. The finer details in the image have not been recovered and this is reflected in the PSNR score of the reconstructed images.



HR (PSNR) | DenseNet 29.32 dB | ResNet 27.89 dB

Figure 5.1.3: Zoomed in view of amplitude image reconstruction patches.

5.2 Recontruction of real FP images

The neural networks trained on stock image data was tested on real FP imaging data and the results are presented in this section.
For clarity we repeat that the neural networks take the set of FP low-resolution images as input and output the reconstructed high-resolution amplitude $\tilde{A}(x, y)$ and high-resolution phase $\tilde{\phi}(x, y)$. Together the amplitude and the phase make up the reconstructed high-resolution object $\tilde{O}(x, y) = \tilde{A}(x, y) \cdot \exp(i \cdot \tilde{\phi}(x, y))$.
The neural networks was trained on channel-wise normalized input as described in section (4.4), and hence the FP low-resolution images was normalized before being fed to the neural networks.
Reconstruction is also performed with the iterative reconstruction method and the reconstructed high-resolution object is compared to the corresponding reconstructed high-resolution images as produced by the neural networks.

5.2.1 Bone and cartilage dataset

The FP imaging dataset used for the reconstruction test is an FP microscopy imaging of a bone and cartilage sample.
The size of the LED-array used was 15×15 resulting in 225 low-resolution intensity images. The pixel size of the camera sensor used was $6.5\mu m$ and the magnification factor $M = 2$ results in an effective pixel size of $3.25\mu m$.

Additional parameters for the experiment is provided in table (5.2.1).

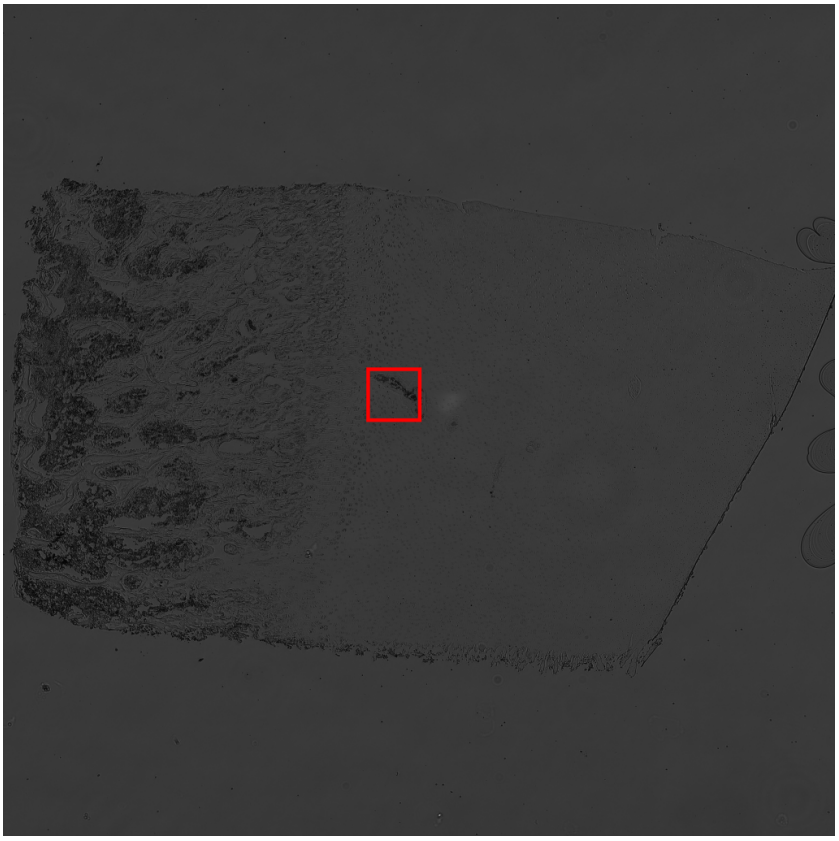


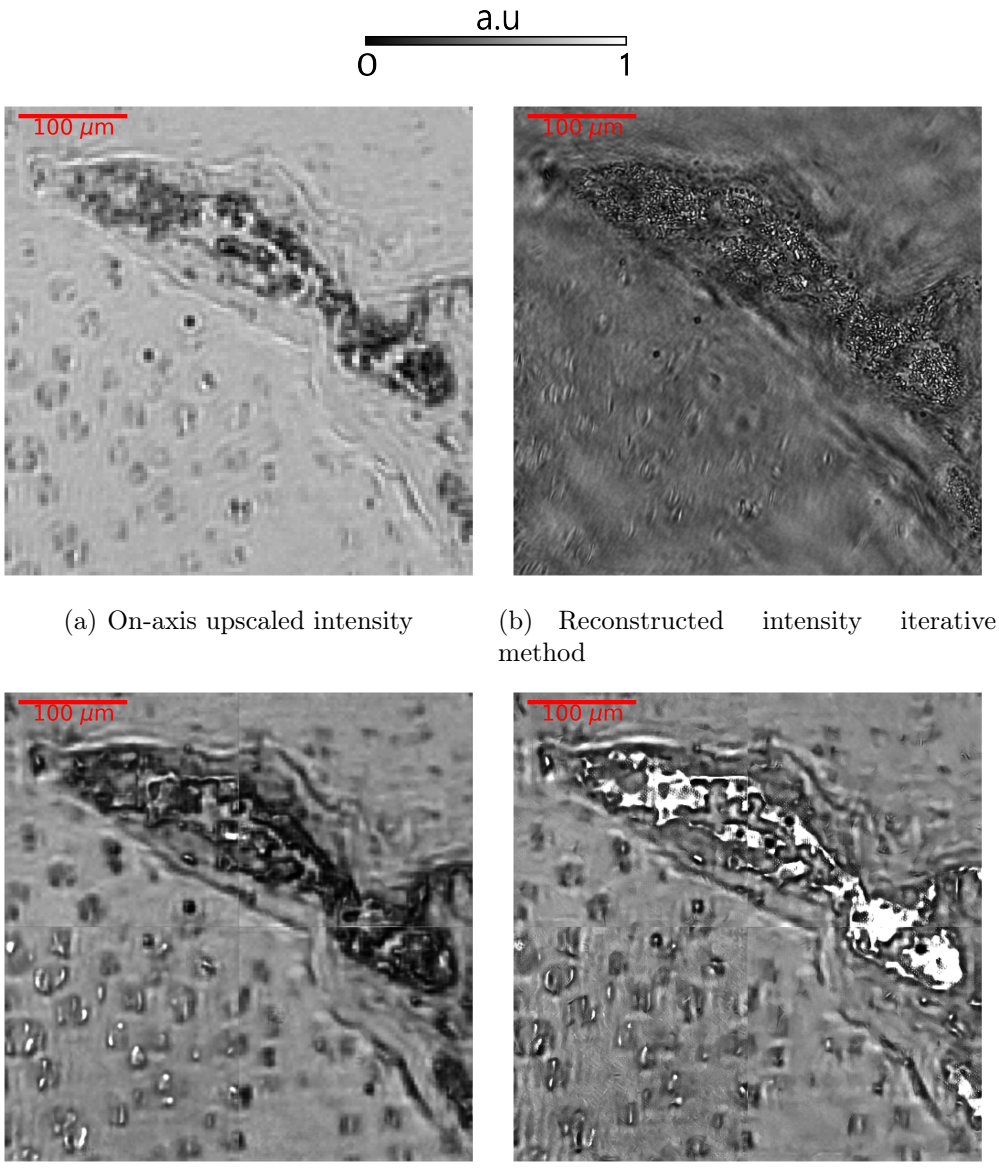
Figure 5.2.1: On-axis low-resolution intensity image. The region enclosed by the red square will be used for testing the reconstruction performance of the optimized neural networks.

Table 5.2.1: Experimental Parameters for FP imaging of the bone and cartilage sample.The setup is similar to the example given in figure (2.2.3).

NA	Magnification	Pixel size (μm)	Wavelength	Image Size	Distance to sample
0.055	2	6.5	530 nm	2048 × 2048	200 mm

5.2.2 Reconstruction results

The region enclosed by the red square in figure (5.2.1) have been reconstructed and the resulting high-resolution intensity images displayed in figure (5.2.2) together with contents of the red square region from the on-axis low-resolution intensity image shown in figure (5.2.1).
We reconstruct the high-resolution intensity with the iterative method (b) as well as with the two different optimized generator neural networks shown in (c) and (d).



(a) On-axis upscaled intensity

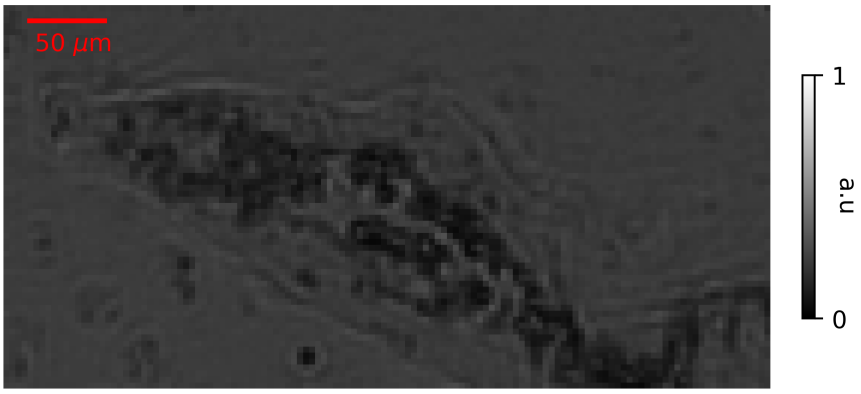
(b) Reconstructed intensity iterative method

(c) Reconstructed intensity DenseNet generator

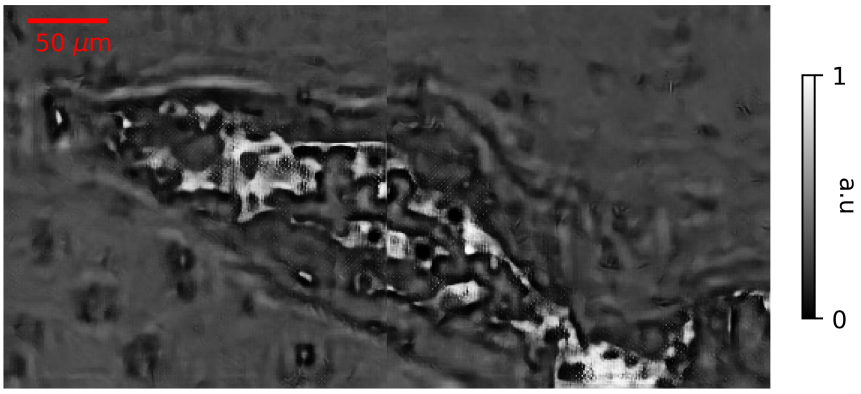
(d) Reconstructed intensity ResNet generator

Figure 5.2.2: (a) is the upscaled region from the on-axis low-resolution intensity image corresponding to the reconstructed high-resolution region shown in (b),(c) and (d). All images shown have dimensions 512×512. All images have been contrast adjusted and scaled to fit the entire 16-bit range to enhance image clarity.

From figure (5.2.2) we observe that both the reconstructed intensity images generated by the neural networks contain details not visible in the on-axis image (a) in figure (5.2.2). A general qualitative visual inspection also suggest the reconstructed intensity images appear to have higher resolution than the upscaled on-axis image.
The resolution enhancement becomes even clearer in figure (5.2.3), which provides an enlarged view of the top half of the region shown in figure (5.2.2). Consequently we conclude with relatively high confidence that the neural networks have succeeded in enhancing the original resolution. The intensity image reconstructed by the iterative method, shown in (b) figure (5.2.2), contain finer details not discernible in the intensity images reconstructed by the neural networks.
The finer patterns present in the intensity reconstructed by the iterative method seems 'fused' together in the intensities reconstructed by the neural networks. This leads us to conclude that the neural networks have failed to enhance the resolution sufficiently to reproduce these details. This suggests that the NNs have not learned to utilize all the available information in the input set of low-resolution FP images.

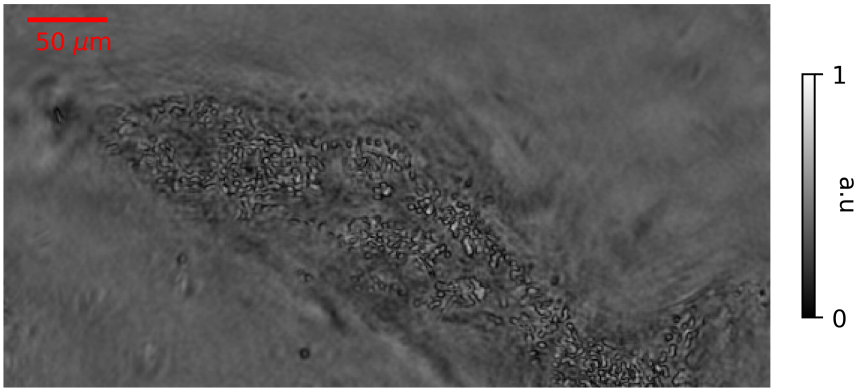


(a) Upscaled on axis intensity.

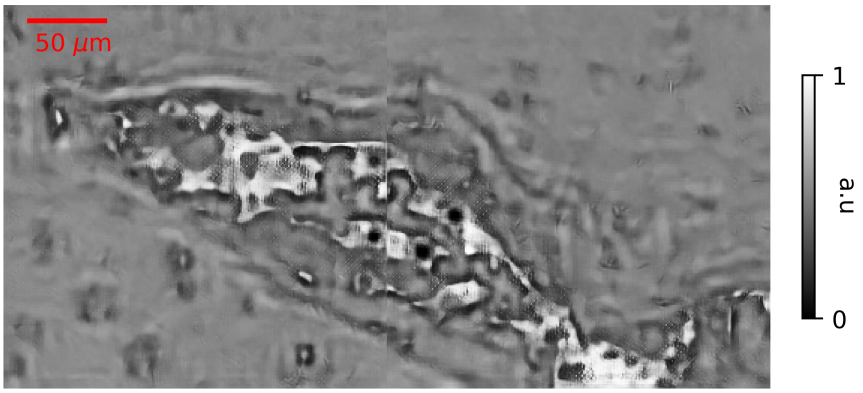


(b) Reconstructed intensity by ResNet generator.

Figure 5.2.3: Enlarged view of the 512×256 top half of the image patch shown in figure (5.2.2)



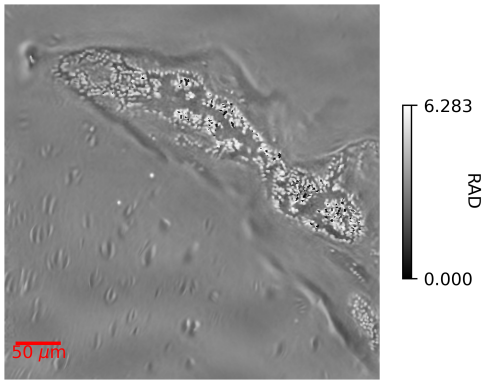
(a) Reconstructed amplitude by the iterative method.



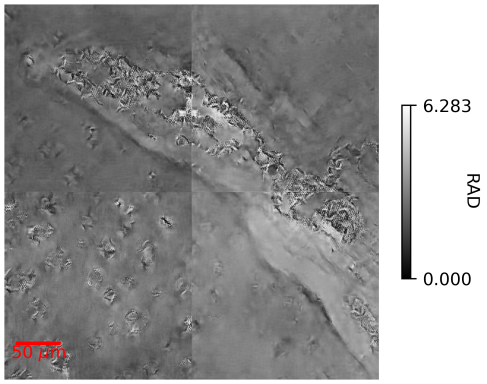
(b) Reconstructed amplitude by the ResNet generator.

Figure 5.2.4: Reconstructed amplitude patch of dimension 512×256 for the top half of the image patch enclosed by the red square in figure (5.2.1).

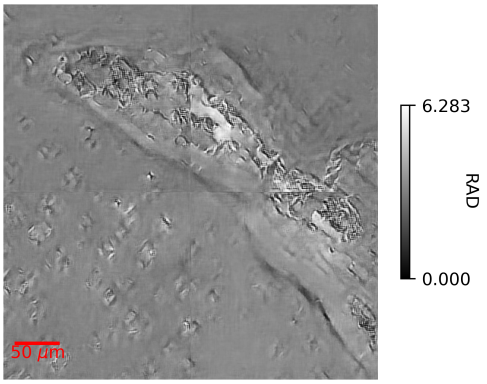
Comparing the amplitudes reconstructed by the iterative method and the ResNet neural network in figure (5.2.4) it shows more clearly that neural network has not managed to reconstruct the fine-grained details that can be observed in the amplitude reconstructed by the iterative method show in (a) figure (5.2.4).
However the amplitude reconstructed by the neural network are to be able to reproduces edges and the overall 'structure' of the image.



(a) Reconstructed phase image ϕ_{iter} by iterative method



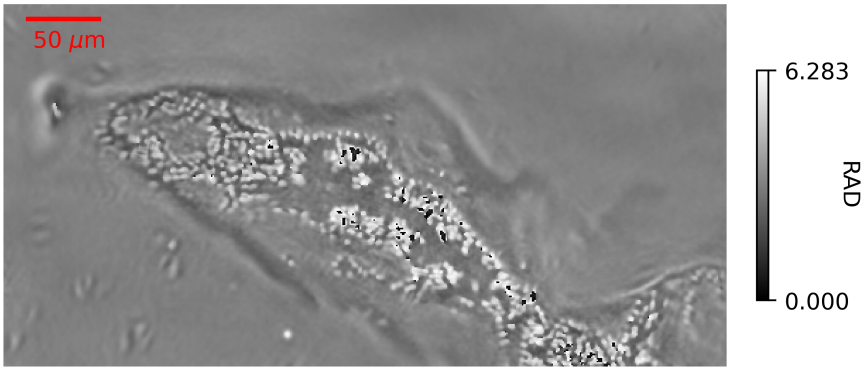
(b) Reconstructed phase image ϕ_{resnet} by ResNet generator.



(c) Reconstructed phase image ϕ_{dense} DenseNet generator.

Figure 5.2.5: Phase image reconstructions of dimensions 512×512 for the 128×128 patch enclosed by the red square in figure (5.2.1).

Reconstructed phase images are shown in figure (5.2.5). All reconstructed phases have been transformed into the range of $[0, 2\pi)$.
Qualitatively we see that the phase images reconstructed by the neural networks are able to detect edges and overall structure fairly well. But again the neural networks fail to reconstruct the finer details.
We also observe significant phase differences at various regions when comparing the phase image reconstructed by the neural networks to the phase reconstructed by the iterative method.
Edge effects between upscaled patches are also apparent in the phase images produced the the neural networks, these edge effects also degrades the image quality.



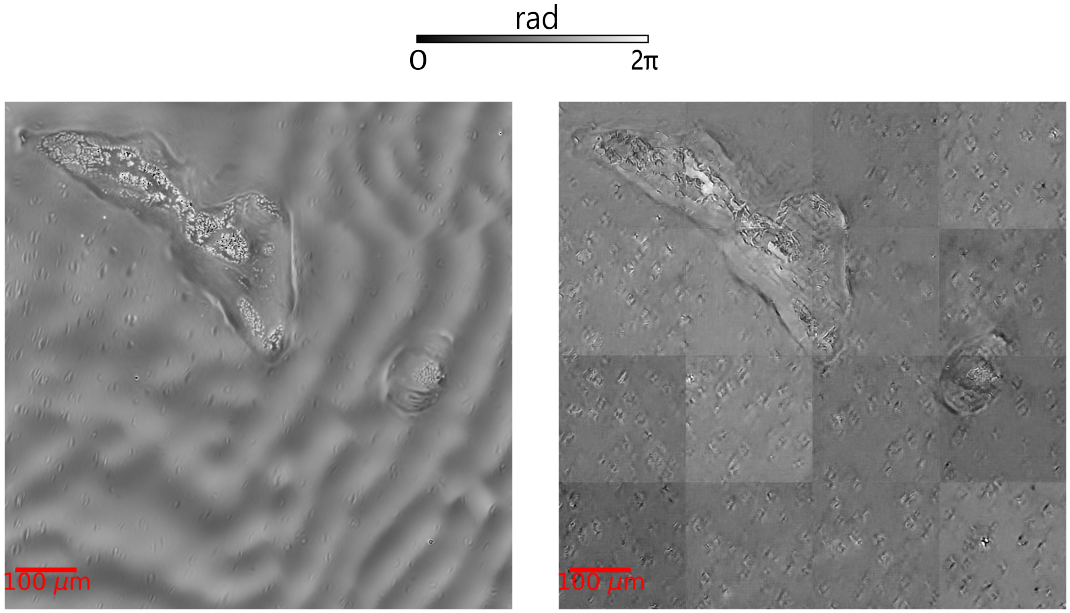
(a) Reconstructed phase image ϕ_{iter} iterative method.

(b) Reconstructed phase image ϕ_{resnet} ResNet generator.

(c) $|\phi_{\text{iter}} - \phi_{\text{resnet}}|$

Figure 5.2.6: Enlarged view of 256×512 patch from figure (5.2.5).

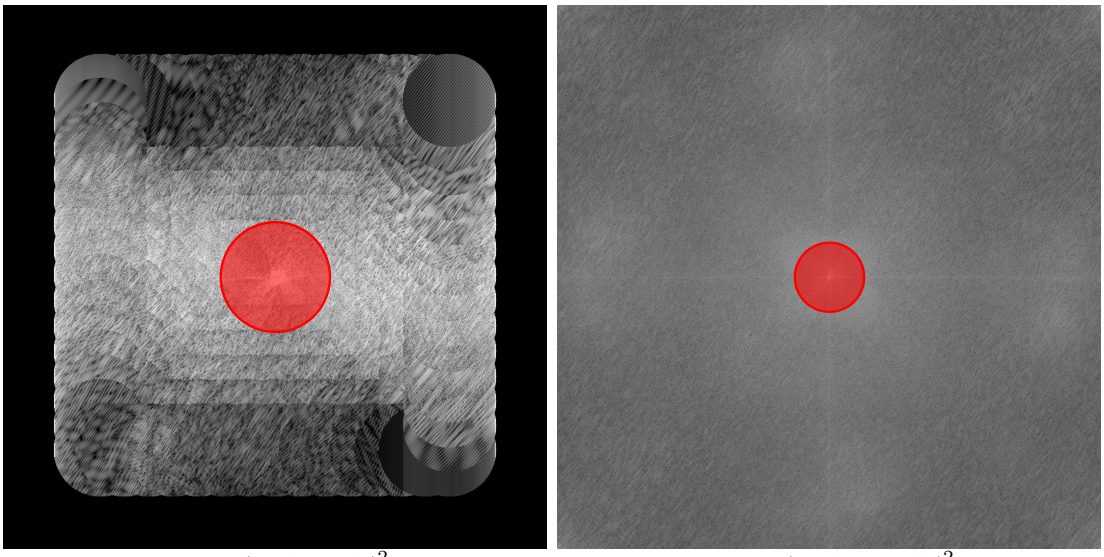
The enlarged view in figure (5.2.6) highlight the previous observations about discrepancies in phase value and details between the phase reconstructed by the iterative method and the phase reconstructed by the neural network.
Figure (5.2.6) (c) shows the absolute phase difference between the reconstructed phase images for the iterative method and the ResNet generator $|\phi_{\text{iter}} - \phi_{\text{resnet}}|$. The plot of $|\phi_{\text{iter}} - \phi_{\text{resnet}}|$ indicates that the neural networks outputs phase values close to ϕ_{iter} for the 'background' region, i.e regions with little variations in phase. In regions with high variation in phase ϕ_{resnet} varies markedly from ϕ_{iter}.



(a) Reconstructed phase iterative method. (b) Reconstructed phase ResNet generator.

Figure 5.2.7: Reconstructed 1024×1024 phase images by the (a) iterative method and (b) the ResNet neural network. The top-left quadrant corresponds to the region in the on-axis image enclosed by the red square in figure (5.2.1).

The phase image reconstructed by the neural network seems to have the roughly the same edges as the phase image produces by the iterative method. Areas with phase considerably different from the average ('background') phase value difference also appear in the same spatial locations in the two phase images. However there are again there are significant differences in the phase values between the phase recovered by the neural network and the phase recovered by the iterative method.
Figure (5.2.7) also showcases some interesting differences between the reconstructed phase image by the iterative method and the phase image reconstructed by the neural network.
The phase image recovered by the iterative method have some 'wave-like' artefacts which is not present in the phase reconstructed by the neural network, while the phase reconstructed by the neural network have jittery artefacts throughout the image.



(a) $\log\left|\mathcal{F}\left\{\tilde{O}_{\text{iter}}\right\}\right|^2$ (b) $\log\left|\mathcal{F}\left\{\tilde{O}_{\text{resnet}}\right\}\right|^2$

Figure 5.2.8: Log power spectrum for reconstructed complex amplitude $\tilde{O}(x, y)$ of the same image patch as figure (5.2.7).

The log power spectra of the recovered complex amplitude $\tilde{O}(x, y)$ corresponding to the same image patch as in figure (5.2.7) is shown in figure (5.2.8).
The encircled areas in figure (5.2.8) are defined such that the fraction of the spectral density outside of the area is less than 1×10^{-6}, in other words it defines a soft frequency cutoff radius. The soft cutoff for the spectrum reconstructed by the iterative method is $0.063k_0$ and for the spectrum reconstructed by the ResNet generator $0.040k_0$. This show that the spectrum reconstructed by the iterative method have a larger portion of the spectral density at higher frequencies than the spectrum reconstructed by the neural networks, and could explain why the recovered images by iterative method contain finer details.

5.3 Reconstruction time and performance

Reconstruction time can be an important factor in FP reconstruction, especially if one needs to reconstruct a long sequence of FP images.
Table (5.3.1) compares reconstruction times for the iterative method and for neural network based reconstruction. Most software for neural networks are made to process mini-batches of input data as discussed in section (2.3.2.2), which means it will upscale several patches in the same forward pass.
The neural network architectures used in this project will takes input batches of dimension $(\text{batch\_dim}, 225, 64, 64)$ and outputs high-resolution patches of dimensions $(\text{batch\_dim}, 256, 256)$. The number of forward passes required to reconstruct the entire high-resolution image is then $2048^2/(\text{batch\_dim} \cdot 64^2)$. Obviously the larger batch dimension the faster the reconstruction of the entire image will be. The VRAM size of the GPU used is the limiting factor of the batch dimension. Considering batch sizes of the form 2^n , it was found that the largest possible batch dimensions for the DenseNet was 64, while the largest batch size possible

for the ResNet architecture turned out to be 32 when tested on a NVIDIA RTX 3090 GPU with 24 GB VRAM
The comparison with the iterative method shows that reconstruction with neural networks is considerably faster than reconstruction with the iterative method. The speedups are quite large and its $7-12$ times faster to reconstruct an entire 8192×8192 complex high resolution image from a set of 2048×2048 low-resolution images using neural networks instead of 5 iterations of the iterative method.
If the image quality and accuracy of the neural network reconstructions could be improved to be of comparable quality as the output of the iterative method this represents a significant faster method, especially for long image sequences.
Drawbacks to the neural network method as presented in this project is the time it takes to train the network, often taking days of training to converge.
A situation where training a neural network beforehand is viable could be a case where a long sequence of FP image data is to collected over time, one such case could be monitoring the evolution of a dynamically changing sample.
Another drawback is the need to retrain the network anytime changes in the experimental parameters that determine the k-shifts $\{(k_{xn}, k_{yn})\}$ occurs, since this changes the definition of the input to the neural network. The iterative method on the other hand does not suffer from this problem.

Table 5.3.1: Comparison of reconstruction times for the iterative method and for the generator neural networks. Patch size refers to the spatial dimensions of the input patch extracted from the low-resolution images. Total reconstruction time is defined as the time requires to reconstruct the entire high-resolution object from low-resolution images of dimension 2048×2048.

Method	Batch size	Patch size (spatial)	Forward pass time	Total reconstruction time
Iterative_5loops	-	256	3.891s	249.0s
Iterative_5loops	-	512	17.53s	280.5s
Iterative_10loops	-	256	8.166s	522.6s
Iterative_10loops	-	512	35.470s	567.5s
DenseNet	64	64	1.303s	20.85s
DenseNet	32	64	0.655s	20.96s
ResNet	32	64	1.130s	36.16s
ResNet	16	64	0.571s	36.55s

CHAPTER

SIX

CONCLUSIONS

Neural networks have been implemented and trained for Fourier ptychographic reconstruction. The entire training dataset used for optimizing the network parameters is created by simulation. The simulation procedure used to create the dataset emulates fourier ptychographic imaging of a complex object $O(x, y) = A(x, y) \cdot \exp(i \cdot \phi(x, y))$ where the arrays $A(x, y), \phi(x, y)$ are grayscale stock images drawn from a dataset. In this way a synthetic dataset is created and removes the need for time consuming collection of training data by doing FP imaging in the lab.
Two neural network architectures were explored.
After network optimization the performance of the neural networks were tested on previously unseen FP datapoints created by selecting data from a validation dataset. The network performance was measured by averaging over a total 20,000 FP datapoints created with input data from a validation dataset. Both the neural network models showed overall good performance reconstructing previously unseen simulated FP data.
In terms of average L_1 distance on validation data between reconstructed images and ground truth images both networks had an average L_1 loss of around 0.07 for amplitude reconstructions, while the phase reconstructions had slightly higher L_1 loss, with about 0.12 average L_1 loss for the DenseNet generator and 0.16 average L_1 loss for the ResNet generator.
The image quality of the reconstructed high-resolution images was also found acceptable in terms of the image quality metrics PSNR and SSIM.

The neural networks that had been trained only on synthetic training data was tested by reconstructing a real FP image dataset. The real FP data was a FP imaging set of a bone and cartilage sample.
The high-resolution image reconstructed showed that the neural networks succeeded in enhancing the resolution, but not to the level theoretically possible, meaning the networks did not manage to fully recover the high-resolution object $O(x, y)$.
The quality of the reconstructed high-resolution neural networks was considerably worse than the corresponding high-resolution image reconstructed by the conventional iterative method. The images reconstructed by the iterative method

contained more details not visible in the reconstructed images produced by the neural networks.
The neural network based reconstruction method considered in this project was considerably faster in reconstructing the high-resolution images compared to the iterative method, but with the image quality of the reconstructions produced by the networks it cannot considered a viable option for Fourier ptychographic reconstruction.

No major performance difference was observed for the two neural network architectures tested, namely the ResNet-style network and the Dense-net style network. It was observed that the DenseNet had somewhat better phase recovery performance. Given that the DenseNet recovers the entire complex object in one forward pass and has a total of 68.32M parameters while the ResNet generator uses two separate networks with 41.08M paramters each. It was also observed that it was possible to reconstruct larger batches with the DenseNet than the ResNet, indicating that the ResNet uses more VRAM on the GPU during a forward pass than the DenseNet. Taking these facts into account the DenseNet is the preferred architecture of the two considered.

6.1 Future work

The architecture of the generator networks could probably be improved by further testing and experimenting. This could be a part of the solution to improving the reconstruction peformance.
Another possible improvement concerns the FP simulation method for creating a FP dataset from stock images. There is likely possible to improve the simulation method to produce FP datapoints with low-resolution intensity images that are more similar to real FP data.

REFERENCES

[Bah19] Malvin C. Teich Bahaa E. A. Saleh. *Fundamentals of photonics*. 3rd ed. Wiley, 2019.

[BDS18] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *arXiv preprint, arXiv:1809.11096* (2018). URL: https://doi.org/10.48550/arXiv.1809.11096.

[Eva16] Trevor Darrell Evan Shelhamer Jonathan Long. "Fully Convolutional Networks for Semantic Segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016). URL: https://doi.org/10.48550/arXiv.1605.06211.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 1st. MIT Press, 2016. ISBN: 978-0262035613.

[Goo+14] Ian Goodfellow et al. "Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014, pp. 2672–2680.

[Goo96] Joseph W. Goodman. *Introduction to fourier optics*. 2nd ed. McGraw-Hill, 1996.

[He+16] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016). URL: https://doi.org/10.48550/arXiv.1512.03385.

[Hec17] Eugene Hecht. *Optics*. 5th ed. Pearson, 2017.

[Hua+17] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708. URL: https://doi.org/10.1109/CVPR.2017.243.

[Iso+17] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5967–5976. URL: https://doi.org/10.1109/CVPR.2017.632.

[Jia+18] Shaowei Jiang et al. "Solving Fourier ptychographic imaging problems via neural network modeling and TensorFlow". In: *Biomed. Opt. Express* 9.7 (July 2018), pp. 3306–3319. DOI: 10.1364/BOE.9.003306. URL: https://opg.optica.org/boe/abstract.cfm?URI=boe-9-7-3306.

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014). URL: https://arxiv.org/abs/1412.6980.

[Lu+21] Xin Lu et al. "Deep learning for fast image reconstruction of Fourier ptychographic microscopy with expanded frequency spectrum". In: *Proceedings of SPIE*. 2021. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11781/117810M/Deep-learning-for-fast-image-reconstruction-of-Fourier-ptychographic-microscopy/10.1117/12.2591381.full?SSO=1.

[Mao+17] Xudong Mao et al. "Least squares generative adversarial networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. IEEE. 2017, pp. 2794–2802. DOI: 10.1109/ICCV.2017.304.

[Ngu+18] Thanh Nguyen et al. "Deep learning approach to Fourier ptychographic microscopy". In: *Optics express*. 2018.

[Ou+15] Xiaoze Ou et al. "High numerical aperture Fourier ptychography: principle, implementation and characterization". In: *Optics Express Vol. 23, Issue 3, pp. 3472-3491* (2015). URL: https://opg.optica.org/oe/viewmedia.cfm?uri=oe-23-3-3472&seq=0.

[PPP18] Frank L. Pedrotti, Leno M. Pedrotti, and Leno S. Pedrotti. *Introduction to Optics*. 3rd ed. Cambridge University press, 2018.

[PyT21] PyTorch. *Convolutional Layers*. https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html. Accessed: September 2021. Facebook, 2021.

[RFB15] 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*. Springer. 2015, pp. 234–241. URL: https://doi.org/10.48550/arXiv.1505.04597.

[Shi+16] Wenzhe Shi et al. "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016).

[Sun+17] Jiasong Sun et al. "Resolution-enhanced Fourier ptychographic microscopy based on high-numerical-aperture illuminations". In: *Sci Rep 7, 1187*. 2017. URL: https://doi.org/10.1038/s41598-017-01346-7.

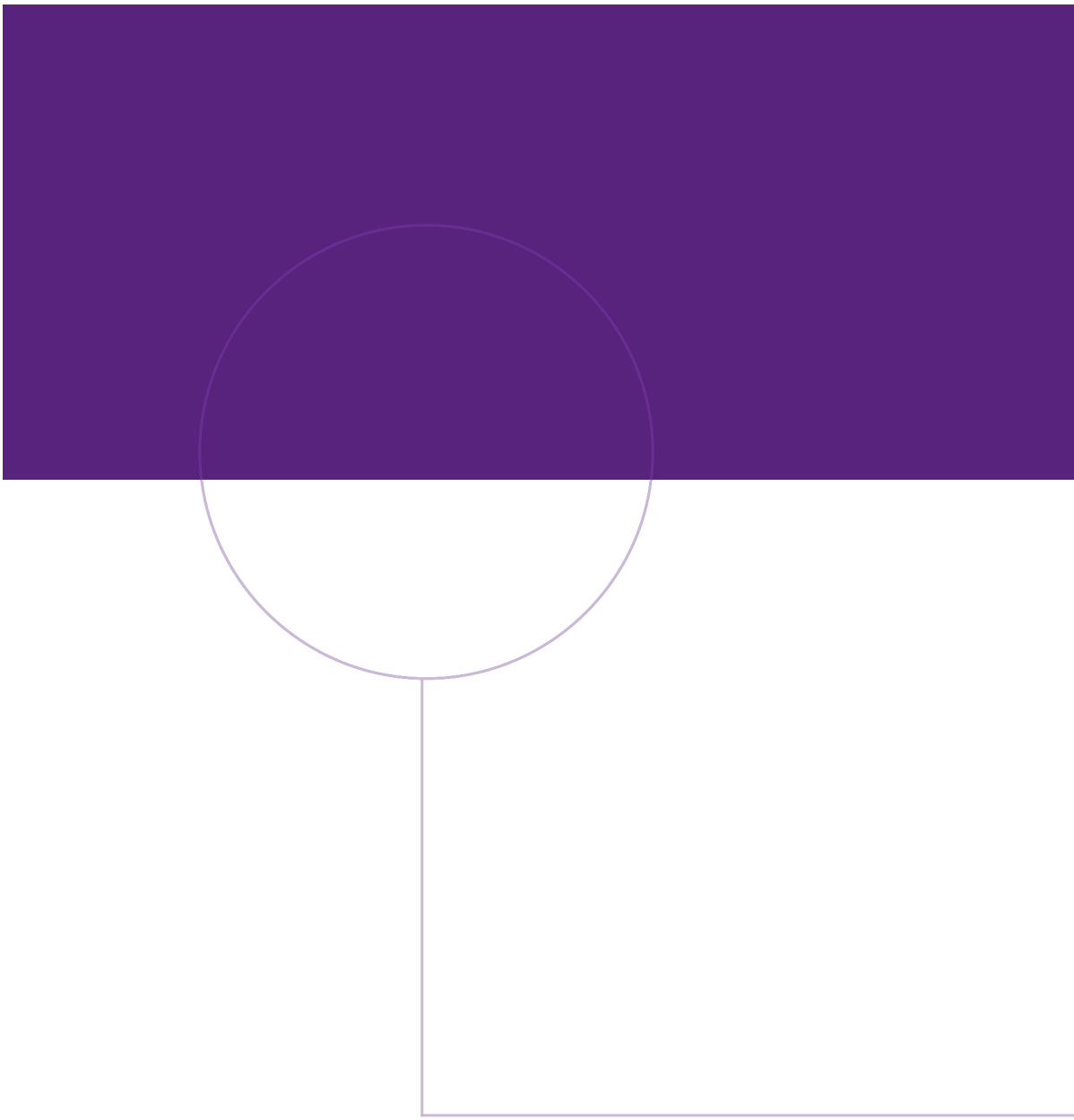
[Sun+18] Jiasong Sun et al. "High-speed Fourier ptychographic microscopy based on programmable annular illuminations". In: *Nature Sci Rep 8, 7669* (2018). DOI: https://doi.org/10.1038/s41598-018-25797-8.

[UVL16] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Instance normalization: The missing ingredient for fast stylization". In: *arXiv preprint arXiv:1607.08022* (2016). URL: https://arxiv.org/abs/1607.08022.

[Zha+19] Jizhou Zhang et al. "Fourier ptychographic microscopy reconstruction with multiscale deep residual network". In: *Opt. Express* 27.6 (Mar. 2019), pp. 8612–8625. DOI: 10.1364/OE.27.008612. URL: https://opg.optica.org/oe/abstract.cfm?URI=oe-27-6-8612.

[Zhe16] Guoan Zheng. *Fourier Ptychographic Imaging*. 1st ed. Morgan Claypool Publishers, 2016.

[ZHY13] Guoan Zheng, Roarke Horstmeyer, and Changhuei Yang. "Wide-field, high-resolution Fourier ptychographic microscopy". In: *Nature Photonics* 7.9 (2013), pp. 739–745.



NTNU

Norwegian University of
Science and Technology