Focal plane phase retrieval using deep convolutional neural networks.

A study on the feasibility of phase retrieval in free space optical communications from a single out of focus intensity measurement using a deep convolutional neural network.

M. Noppen

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Focal plane phase retrieval using deep convolutional neural networks.

A study on the feasibility of phase retrieval in free space optical communications from a single out of focus intensity measurement using a deep convolutional neural network.

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

M. Noppen

August 9, 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by TNO.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Focal plane phase retrieval using deep convolutional neural networks.

by

M. Noppen

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: August 9, 2019

Supervisor(s):

prof.dr.ir. M.Verhaegen

Dr.ir. R. Saathof

Reader(s):

dr.ir. C. Smith

dr.ing. J. Kober

Abstract

Adaptive optics are widely used to correct the wavefront distortion imposed by atmospheric turbulence. Focal plane phase retrieval from intensity measurements has advantages due to the ease of implementation, potential broader application, less computations, low cost, high system bandwidth, simplified hardware and less calibration. To cope with the non-linear relation between focal plane intensity and wavefront phase the use of Machine Learning is investigated. A supervised learning deep Convolutional Neural Network is used to assess the feasibility for deriving a direct mapping between a single out of focus CCD intensity measurement and the Zernike modes belonging to it. A model of a typical free space optical communication system is used to asses 13 different CNN architectures. The first 35 Zernike modes(disregarding tip/tilt) were retrieved from Kolmogorov based CCD intensity measurements of size 70x70 with constant amplitude, turbulence strength of $1 \leq D/r_0 \leq 6$, $7 \leq D/r_0 \leq 12$ and $13 \leq D/r_0 \leq 18$ and a SNR of 22dB. The convergence of a 128 channel six layer CNN with kernal size 3 using stride resulted in a mapping providing a residual wavefront error of $5nm$, $21nm$ and $74nm$ after reconstruction of the wavefront. The results prove that a CNN can be used to map out of focus intensity data directly onto the Zernike coefficients of the wavefront. The CNN is validated by an experimental setup which was used to generate real input and output data. With a turbulence strength of $5 \leq D/r_0 \leq 15$ the mean squared phase error was found to be $72nm$. The use of a deep CNN for phase retrieval implementation in free space optical communications is promising and can provide fast and accurate phase retrieval with relatively simple hardware and faster computations.

Table of Contents

List of Figures

List of Tables

Preface

This thesis report is the result of a year part-time work at TNO Stieltjesweg and meant to conclude my masters program Mechanical Engineering at TU Delft where I followed the track Biomechanical Design(BMD). Before January 2018 I would not have though to graduate at the Optomechatronics department of TNO and at a different TU department then I study. But as I got in touch with the people at TNO via my current employer to discuss the implementation of laser communications in space an idea for a thesis topic grew. Since I followed some topics on optimization and machine learning the idea to perform wavefront corrections by the implementation of machine learning came up.

As I had a constraint on having to be of added value to TNO, TU Delft, RNLAF and not to forget myself I was very glad to be able to exploit this idea into a thesis. Although I must say that I almost regretted getting this started because I had zero experience in the field of optics which unpacked in a lot of time consuming 'side-snacks' down the road. But now that it's done I am satisfied with the result.

To cope with the snacks I would like to thank the people at TNO for helping me out where needed and I would like to apologize for not getting to know you better as my presence was very limited. Though I still have experienced the high level of expertise available at the department which I could make use of from time to time. As I hate asking for help I spend a lot of time cracking my brain on some topics but when needed I could rely on a concise answer of Rudolf so a special thanks to you for that. From the Airforce I want to thank Bernard for trusting me to combine both work and study when I applied at the department and for being flexible when I had 'stuff' to do at the university. Also I want to thank Bas for stepping in early to unload me from my tasks as my end at the office was coming near.

Overall I can conclude to myself that the past few years were very busy and consumed a lot of free time but when I look at the skills I mastered it was definitely worth it. In the next challenge that is already ahead of me this will help me have a smooth and great time. Though the next time I will remember myself, and everybody who is thinking of combining full-time study with full-time work, don't do it(is what I say at this moment).

"♫♫ Lekker belangrijk ♫♫ "

— *Edwin Evers*

Chapter 1

Introduction

1-1 Background

Already in 1608 the first telescope was invented in the Netherlands by Hans Lippershey after which new developments in the field of optics took a flight. Over the years the quality and aperture of telescopes kept increasing, partially due to the curious minds of astronomers. After many advances Huygens found in the mid 50's that starlight is mostly coherent until it reaches the earths atmosphere causing seeing conditions to degrade significantly. After a period of research to this it is now known that due to the chemical composition and state of the atmosphere electromagnetic waves undergo dispersion, distortion, absorption, reddering and refraction when propagating through this medium[2], where the contribution of each individual phenomena is also largely dependent on the wavelength of the subjected electromagnetic wave.

In the contemporary society we live in we highly rely and depend on a backbone of terrestrial and free space means of communication[3] using these electromagnetic waves or more specifically Radio Frequency (RF) waves. High bandwidth RF data links make sure that global communication and data exchange is of low latency and high throughput[3]. Various global players such as Intelsat, Eutelsat and Echostar make use of large, complex and costly systems to guarantee our dependency on communications. Though with the ever growing demand on availability and throughput the systems providing these services are growing bigger, more complex and more expensive[4]. Apart from that, free space RF spectrum has become congested making it difficult for the International Telecommunications Union (ITU) to allocate frequency bands to users without constraints on usage due to the risk of interference.

Due to these indicated issues high throughput, low interference systems in the visible light spectrum and abutting frequencies are finding their way to the stage. These optical

systems, using a focused beam penetrating through free space, enabled the development of communication systems transceiving at orders of magnitude higher data-rates. This without having to de-conflict different users due to the low divergence of the emitted beam and with that the need to file for a license at the ITU. Additionally, due to the low divergence of the beam less power is required which opens up opportunities in space based implementations where the available power budgets are often constraining link throughput.

Unfortunately the wavelength used for such systems brings new challenges to overcome when transceiving through the atmosphere. Atmospheric conditions are constantly changing and with that many variables influence the performance of a Free Space Optical Communications (FSOC) system. Clouds, fog, snow and rain absorb and scatter the laser beam causing attenuation of the transmitted beam and with that increasing bit error rate[2] or in the case of cloud deck making communications impossible. Atmospheric turbulence is a form of scattering causing various phenomena that make the optical wavefront deform[5]. This turbulence can cause beam wander or spreading and scintillation which further degrades the link. Also other influences such as beam divergence, pointing loss, system noise, vibrations cause the optical link to degrade further. Though apart from clouds, for which no apparent cloud penetrating solution is available, atmospheric turbulence is often the most severe[2].

This turbulence causes the light to travel different path lengths resulting in a phase shift in the emitted bundle. In the 1950's the concept of Adaptive Optics (AO) was already found to be suited for the correction of this phase aberration in the distorted wavefront[6]. Though it took another 20 years before the first techniques were implemented which were continuously improved from of that moment.

1-2 The principle of adaptive optics

Figure 1-1 shows two implementations of an AO system. On the left the Wavefront Sensor (WFS) is placed after the Deformable Mirror (DM) so that a closed loop system can be realized since a residual error can be measured. On the right the open loop control implementation is shown, this has as advantage that the control system does not have to account for the corrections applied by the DM. A downside is that in order to implement open loop control the DM has to be modeled very accurately to prevent DM states that not fully counter the wavefront due to a lack of feedback.

AO provide a means to correct a distorted wavefront so that the incoming electromagnetic waves can be effectively propagated to a focal plane where often a science instrument or detector is placed. By implementing a DM in the AO system the path length of the Electro Magnetic (EM) waves can be altered by deflecting the mirror surface. This way a phase shift in the wavefront can be countered as much as possible. As the wavelength of visible light and abutting frequencies are in the range of Nanometers the DM needs to be able to realize deflections on this small scale too.

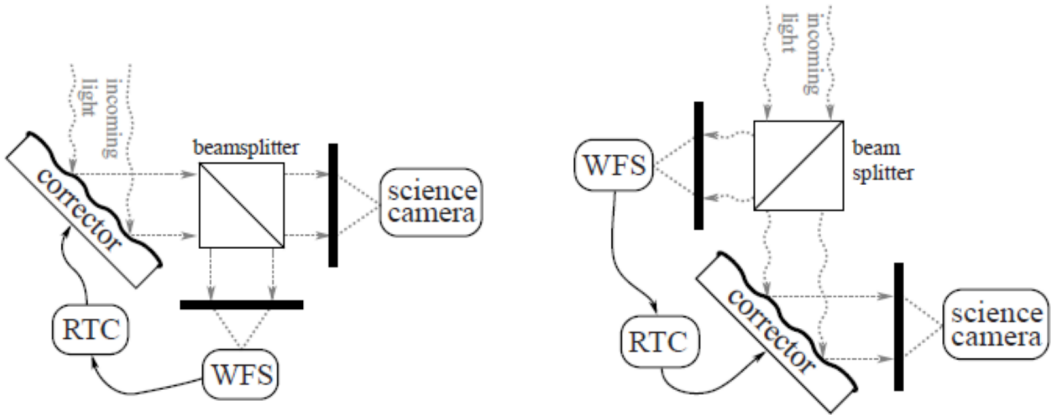

Figure 1-1: Two general implementations of an AO system. On the left a closed loop implementation where the WFS is placed after the DM. On the right an implementation in open loop with the WFS placed before the DM.

1-3 Goal of this research

1-3-1 Problem

Various wavefront sensing techniques exist such as pyramid sensors[7] and interferometers[8] but FSOC AO systems most often rely on a Shack-Harmann (SH) wavefront sensor[5] to retrieve the phase. Although good performance is achieved with this method it has downsides too. Due the lenslet array the resolution is limited and since the dynamic range and sensitivity are inherently coupled as (1-1) and (1-2) present. Here θ_{min} is the sensitivity and θ_{max} the dynamic range. Increasing the focal length f results in a higher sensitivity but also a lower dynamic range. Further is a SH wavefront sensor often more suited for setups of high optical power which is often not available on FSOC systems. Also in challenging environments(e.g. low elevation angle, high turbulence) a limited count of contributing lenslets can cause difficulty in retrieving the phase. Further significant computational power must be made available for the phase retrieval algorithms, careful calibration is required and hardware is costly. For these reasons Focal Plane (FP) WFS can be a viable alternative when comparable performance to a SH WFS can be reached.

$$\theta_{min} = \frac{\delta y_{min}}{f} \tag{1-1}$$

$$\theta_{max} = \frac{d/2}{f} \tag{1-2}$$

A CCD detector in the FP can only measure intensity, making it impossible to directly retrieve the wavefront phase. For this reason iterative algorithms such as the Gerchberg-Saxton (GS) calculate the wavefront based on a cycle of Fourier transforms and its reverse

to iteratively determine the wavefront phase based on a measured intensity. Other more advanced algorithms exist but the iterative nature limits the computation speed[9][10]. Implementing such algorithms on state of the art systems as under development at De Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek (TNO) is challenging as requirements demand an operating frequency of 5000Hz for high turbulent environments.

As the transformation from FP intensity data to wavefront phase is non-linear the use of Machine Learning (ML) can be interesting as it might can provide a means to find a direct mapping between intensity measurement and phase so that iterative algorithms become superfluous, a possible broader range of turbulent states can be handled, hardware is simplified and calibration is of less essence.

1-3-2 Goal

Neural networks have already been used often to find (non-linear) mappings between sensory input and system output. Especially in the last decade ML established itself in a very broad field of applications, such as health care, manufacturing, education, financial modeling, policing, and marketing. This is partially due to software packages that bring these techniques to the doorstep of a non-computer scientist[11][12]. To investigate whether ML, or in this case more specifically a Neural Network (NN) can provide a solution to the presented non-linear focal plane phase retrieval problem the goal of this research is to;

Investigate if a Feed Forward Neural Network (FFNN) can be used to find a mapping between FP intensity images and the wavefront phase for an open loop adaptive optics setup meant for free space optical communications.

1-3-3 Report structure

Figure 1-2 presents the processing sequence from incoming electromagnetic field E to the commanding of the DM. Here X represents a 2D array holding pixel values of the out of focus plane, $\hat{y}$ is a vector holding the wavefront mode coefficients retrieved by the NN and $\hat{y}_{DM}$ a vector holding the command voltages for the DM actuators. The figure also shows the DM control as a step but as the DM commanding is straight forward and not defined as a research goal, the training of a NN for this task is superfluous and out of scope.

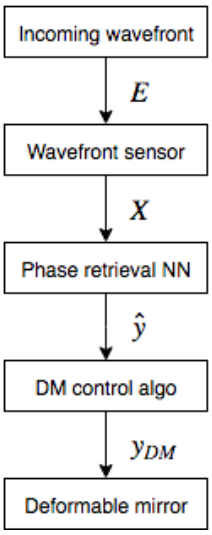

Figure 1-2: The steps from incoming wavefront to DM actuation by the use of NN's

To enable the training of a NN chapter 2 of this report focuses on the generation of E, X and reference NN output y in the shape of the Zernike modes present in the wavefront. By the use of this model data can be generated with which an investigation to the phase retrieval from the focal plane data can be started. Chapter 3 focuses on the generation of a set of NN architectures by training a basic architecture to distill which hyperparameters are important. Thereafter the NN architectures are evaluated on input data for different turbulence parameters in chapter 4. To validate the results of chapter 4 an experimental setup is build to prove the working principle of the found results as will be elaborated on in chapter 5. In chapter 6 the found results are discussed and conclusion are drawn.

Chapter 2

Input- output data model

Before a Neural Network (NN) for phase retrieval or prediction can be trained, input and output data has to be available for training. Therefore a model is created that provides input data X and reference output y, this chapter describes the developed model for this purpose. The NN to be designed for phase retrieval would then have as goal to find a mapping of X onto y.

2-1 Model layout

The model will be build as a sequence of operations. First phase screens representing the turbulence have to be created as source of the distortion. Secondly this turbulence has to be propagated through the optical system towards the Wavefront Sensor (WFS) which is a defocused Charge Coupled Device (CCD). Thirdly the complex electromagnetic field has to be translated to a 2D pixel array by modeling the CCD.

Figure 2-1 shows a general Adaptive Optics (AO) setup of which the red part has to be modeled to acquire input and output data. The camera at the end of the chain can be either a single mode fiber or a CCD for pointing, acquisition and tracking camera but is not relevant for the data model. For the purpose of generating X and y the AO setup from source to WFS has to modeled as will be elaborated on in the following subsections.

In order to work with representative dimensions and quantities the De Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek (TNO) Ofelia AO breadboard [13] is used as reference. Ofelia stands for Optical Feeder Link Adaptive optics and is the most recent Free Space Optical Communications (FSOC) ground station breadboard at TNO and is meant to evolve into a full scale FSOC ground station in the near future. In the basis the system is build as full duplex closed loop system incorporating a

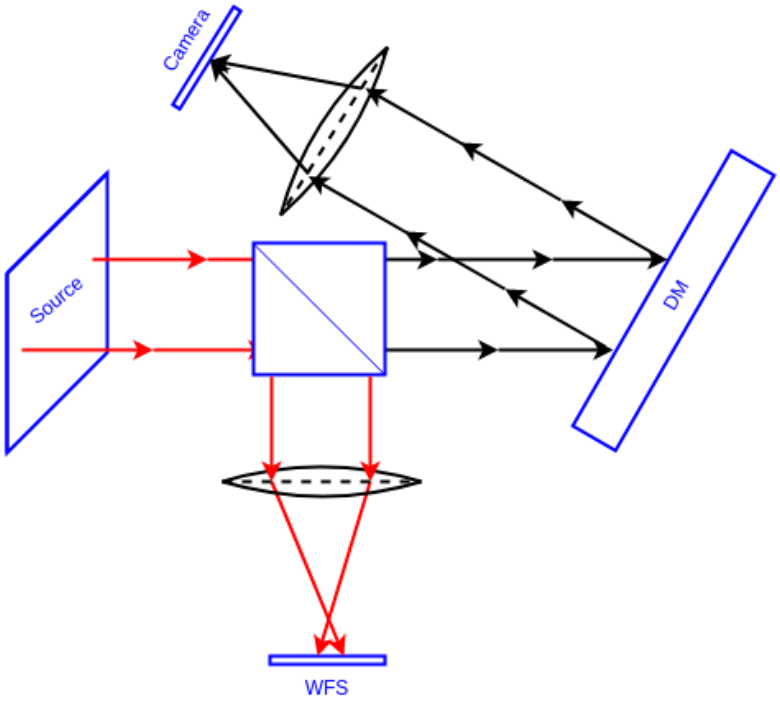

Figure 2-1: The schematic layout of the optical system that is partially modeled for simulations.

$0.1m$ Aperture, a Shack-Harmann (SH) WFS, tip-tilt sensor, Focus Camera (FC), Point Ahead Mirror (PAM), fast steering mirror and Deformable Mirror (DM).

2-2 Phase screens

The first step is to provide the source disturbance to the model. At TNO a computational tool called Helsinki is available to generate realistic turbulence data based on a Kolmogorov[14] spectrum and for which various turbulent layers can be stacked based on a C_n^2 profile. The output of the tool is both a sequence of 2D phase and amplitude frames. For the purpose in this research a disadvantage of Helsinki is that, depending on the parameters used, it takes a long time to generate the frames even on a cluster of PC's(500 frames in 3 days on TNO cluster). Training a NN without the risk of overfitting would require a multiple in the range of 10000 to 100000 when referring similar problems with 2D imagery input[15]. For this reason it was chosen to not use this high fidelity model but to generate a custom input.

Since the main question to answer is whether the phase can be retrieved from the Focal Plane (FP) in the shape of Zernike coefficients it is decided to create a model specifically for this research purpose. This has as advantage that full control of the input is

guaranteed and the relation between the Zernike modes to retrieve and the presented phase disturbance is pure and without noise(apart from CCD detector). To prove the concept it is also chosen to fix the amplitude of the turbulence to one. This again is to isolate the problem to only phase retrieval.

The following subsections describe the functions that have been used to generate the phase screens based on the stated here above.

2-2-1 Zernike modes

The incoming aberrated wavefront is often expressed by the use of Zernike coefficients[16] developed by the Dutch physicist Frits Zernike. These coefficients are used extensively in various fields to describe the wavefront. Also for phase retrieval the coefficients are used to express the wavefront phase and often form an intermediate step before translating these to DM commands. There are positive and negative Zernike polynomials, the positive Zernikes are defined by 2-1 and the negative by 2-2.

$$Z_n^m(\rho,\phi) = R_n^m(\rho)cos(m\phi) \tag{2-1}$$

$$Z_n^{-m}(\rho,\phi) = R_n^m(\rho)sin(m\phi) \tag{2-2}$$

Here the radial polynomial R is provided by;

$$R_n^m(\rho) = \sum_{k=0}^{(n-m)/2} \frac{(-1)^k(n-k)!}{k!((n+m)/2-k)!((n-m)/2-k)!}\rho^{n-2k} \tag{2-3}$$

Here ρ represents the radius and ϕ azimuthal angle. Further $n \geq m$. for values where $m \subset 0$ the polynomial is named negative.

A specific Zernike is often referred to by the J number which represents the polynomial ordering number. This number is provided by Eq. (2-4). Though often the polynomial ordering number is simply called a mode, as will also be the case in this report.

$$J = \frac{n(n+2)m}{2} \tag{2-4}$$

Zernikes are orthogonal on the unit disk and often the low order modes are depicted by a specified name. The first five modes are piston(Z_0^0), tip(Z_1^{-1}), tilt(Z_1^1), defocus(Z_2^0), and astigmatism(Z_2^{-2} and Z_2^2).

The Ofelia DM with 57 actuators is a very capable device though it is chosen to design the model for the retrieval of the first 35 Zernike modes as these already represent 98.8% of the total phase error and is enough to prove the concept. For clarity the modes to retrieve are shown in Figure 2-2.

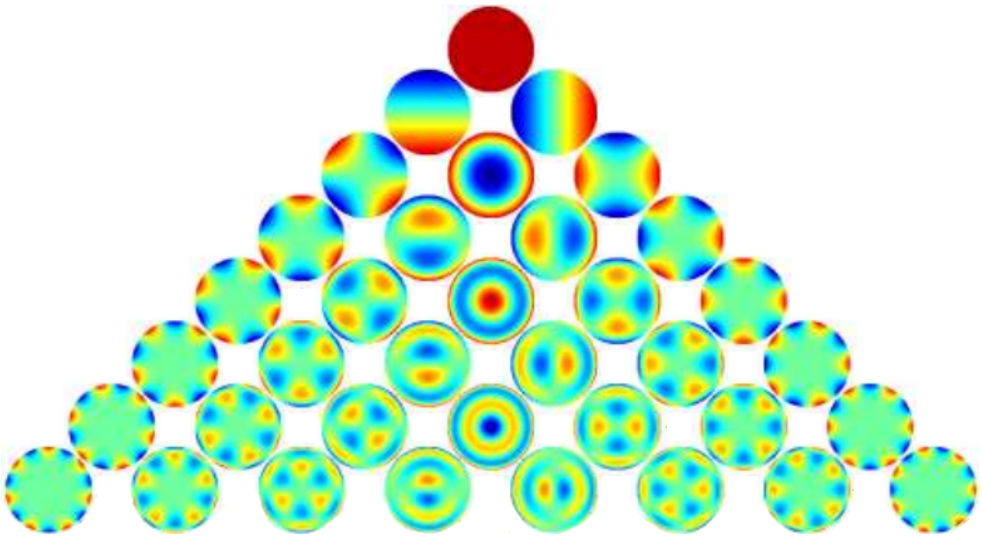

Figure 2-2: The first 35 modes that the NN will be tough to retrieve.

2-2-2 Randomizing phase screens

With increasing mode number less contribution to the overall phase ϕ is delivered. In order to generate a realistic phase pattern this has to be incorporated in the model. To do so an analysis of a Kolmogorov[14] turbulence by Noll[1] provides us with equations for the mean square residual phase error as are provided here in Table 2-1.

Table 2-1: Zernike-Kolmogorov mean square residual phase errors(Δ_J) as by Noll[1]

$\Delta_1 = 1.0299(D/r_0)^{5/3}$	$\Delta_{12} = 0.0352(D/r_0)^{5/3}$
$\Delta_2 = 0.582(D/r_0)^{5/3}$	$\Delta_{13} = 0.0328(D/r_0)^{5/3}$
$\Delta_3 = 0.134(D/r_0)^{5/3}$	$\Delta_{14} = 0.0304(D/r_0)^{5/3}$
$\Delta_4 = 0.111(D/r_0)^{5/3}$	$\Delta_{15} = 0.0279(D/r_0)^{5/3}$
$\Delta_5 = 0.088(D/r_0)^{5/3}$	$\Delta_{16} = 0.0267(D/r_0)^{5/3}$
$\Delta_6 = 0.0648(D/r_0)^{5/3}$	$\Delta_{17} = 0.0255(D/r_0)^{5/3}$
$\Delta_7 = 0.0587(D/r_0)^{5/3}$	$\Delta_{18} = 0.0243(D/r_0)^{5/3}$
$\Delta_8 = 0.0525(D/r_0)^{5/3}$	$\Delta_{19} = 0.0232(D/r_0)^{5/3}$
$\Delta_9 = 0.0463(D/r_0)^{5/3}$	$\Delta_{20} = 0.0220(D/r_0)^{5/3}$
$\Delta_{10} = 0.0401(D/r_0)^{5/3}$	$\Delta_{21} = 0.0208(D/r_0)^{5/3}$
$\Delta_{11} = 0.0377(D/r_0)^{5/3}$	$\Delta_J \sim 0.2944J^{-\sqrt{3}/2}$

Based on these residual errors a randomization method for the generation of phase screens can be derived with Frieds parameter r_0 and aperture D as input. By using this method unlimited test frames can be generated which is convenient for training NN's.

The Kolmogorov-Zernike residual error from Table 2-1 can be written as;

$$\Delta_J = c_J(D/r_0)^{5/3} \tag{2-5}$$

Here c_J represents the coefficients used for each J mode. Δ_J is the summed Kolmogorov-Zernike residual phase error of the modes higher then J. When a mode is picked and all higher modes are subtracted the phase contribution of that specific mode is found as is defined in (2-6).

$$\phi_J = c_J(D/r_0)^{5/3} - c_{J+1}(D/r_0)^{5/3} = (c_J - c_{J+1})(D/r_0)^{5/3} \tag{2-6}$$

With that random phase screens can be generated based on a random uniform distribution for each J-mode. Since ϕ_J is a variance the square root has to be taken so that sigma becomes: $\sigma_J = \sqrt{\phi_J}$ and zero mean μ.

$$y_J \sim \mathcal{N}(\mu, \sigma_J) \tag{2-7}$$

Here y are the random Gaussian J-mode coefficients that can be multiplied by the corresponding Zernike mode profile which is scaled to a value so that the standard deviation equals one. With that an overall input phase screen ϕ_{in} is defined by:

$$\phi_{in} = \sum_{J=1}^{J_{max}} y_J \frac{Z_J}{\sqrt{\frac{\sum_{i=1}^{n} |Z_{J,i} - \bar{Z}_J|^2}{n}}} \tag{2-8}$$

Here Z_J is a 2D array holding the corresponding Zernike of mode J, n is the number of elements in Z_J, $\bar{Z}_J$ the average and $\boldsymbol{\phi_{in}}$ the resulting phase screen that can be used to propagate to the focal plane as described in the next section.

The reference output for the NN is defined by a vector y holding a specified range of Zernike coefficients as defined in Eq. (2-9) where $y_{J_{min}}$ is the lowest mode to retrieve and $y_{J_{max}}$ the maximum mode.

$$y = [y_{J_{min}}, y_{J_{min}+1}, y_{J_{min}+2}, ..., y_{J_{max}}]^T \tag{2-9}$$

2-3 Propagation

With ϕ_{in} and A_{in} being respectively the input phase and amplitude of the incoming wavefront, the propagation towards the WFS detector can be described using Fourier theory as provided by Goodman[17]. First the complex electromagnetic field E_{in} is determined by;

$$E_{in} = A_{in}e^{i\phi_{in}} \tag{2-10}$$

Next a simplified beam splitter is modeled for which the phase shift and the absorption by the reflective layer are neglected.

$$E_{prop} = E_{in} r \tag{2-11}$$

Here r is the reflectance of the beam splitter. Next a convex lens converges the light to the out of focus WFS. Assuming a perfect lens the following equations are added.

$$\phi_d = \frac{R_d - \sqrt{x^2 + y^2 + R_d^2}}{2\pi\lambda} \tag{2-12}$$

Here ϕ_d is the phase shift due to the local difference in path length created by the lens. R_d is the divergence radius of the lens, x and y are the local coordinates in the focal plane and λ is the wavelength. Next the Fourier electric field in the focal plane is provided by the discrete 2D Fourier transform.

$$E_{FPf} = \mathcal{F}(E_{prop} + e^{i\phi_d})\mathcal{F}(E_l) \tag{2-13}$$

Eq. (2-13) provides a unique solution as the Fourier Transform is performed on a 2D array[18]. Further E_l is an extra term accounting for the out of focus setup as is provided by Eq. (2-14).

$$E_l = \frac{l e^{\frac{2i\pi}{\lambda} R_{FP}}}{R} \tag{2-14}$$

Here $R = x^2 + y^2 + l^2$, $R_{FP} = \sqrt{R}$ and l is the distance between the lens and the camera sensor.

The final EM field at the focal plane then becomes;

$$E_{FP} = \mathcal{F}(E_{FPf})^{-1} \tag{2-15}$$

2-4 CCD

A CCD is modeled to find the digital intensity pattern, for this an existing TNO Matlab script "Add_noise.m" by R. Saathof is adopted and slightly altered.

Starting with the presented electric field E_{FP} at the CCD only the intensity I_d can be measured in W/m^2 so that;

$$I_d = \mid E_{FP} \mid^2 \tag{2-16}$$

This intensity has to be transformed to a digital signal in the format of pixel values in an image. To do so the detector is modeled by the following equations.

First the number of received photons n on the detector surface is calculated based on the received power P_i and the sampling rate f_s,

$$n_r = \frac{P_i}{E_p f_s} \tag{2-17}$$

Here $E_p = \frac{ch}{\lambda}$, with c being the speed of light and h Planck's constant. From the received photons only a part is detected due to the detector properties;

$$n = n_r Q_e r_a f_f t \tag{2-18}$$

Here n_r is the read out noise in electrons, Q_e the quantum efficiency in the range 0 to 1, r_a the duty cycle, f_f the fill factor and t the transmission factor.

Knowing the total number of photons collected and the intensity distribution over the number of photons per pixel is calculated by;

$$I_{pixel} = \frac{I_d}{\sum I_d} n \tag{2-19}$$

Due to dark current d_c noise is added to the above array.

$$I = I_{pixel} + d_c f_s \tag{2-20}$$

Now that the number of photons per pixel including dark current is know Analog to Digital (AD) conversion is done for $0 < I < f_w$, where f_w is the maximum number of photons one pixel on the detector can hold.

$$D_{dig} = (2^b - 1)\frac{I}{f_w} + 1 \tag{2-21}$$

In Ofelia the Xenix Cheetah-640-CL[19] detector is used as WFS with properties as in Table 2-2. Using these parameters combined with a received power $P_{in} = 1 * 10^{-6}$, out of focus of 20% with respect to the focal length of the lens and various turbulence strengths the input frames X become as depicted in Figure 2-3

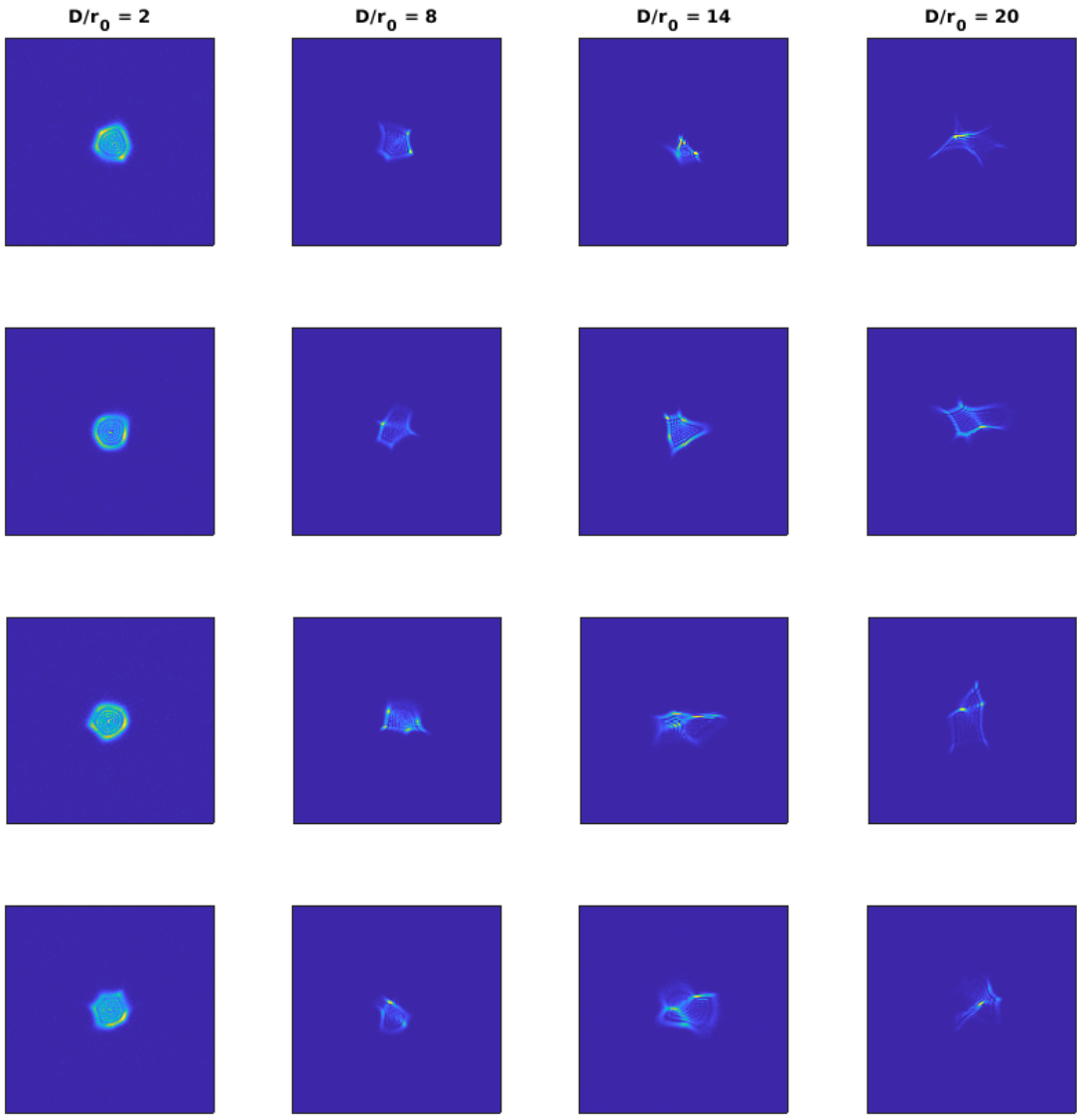

Figure 2-3: Example detector images for four different turbulence strengths ranging from very low ($D/R_0 = 2$) to very($D/r_0 = 20$) turbulence based on the first 35 modes and a 20% defocus.

Table 2-2: Xenixs Cheetah-640-CL properties

Property	Value
Quantum efficiency	0.7
Dark current	$1.9e5e^{-}/s/pix$
Framerate(@full res.)	800
Max resolution	640 x 512
Pixel size	$20\mu m$
Readout noise	$180e^{-}$
Duty cycle	0.1
ADC resolution	14
Full well	$80000photons$
Sample rate	$2000Hz$
Fill factor	0.9

Chapter 3

CNN designs and hyperparameter selection

The 2D pixel intensity pattern X, as captured by a Charge Coupled Device (CCD), is an image holding features on the wavefront phase ϕ which is often retrieved by iterative algorithms such as the Gerchberg-Saxton algorithm[20] or more advanced variants[9] and alternatives[21]. This chapter will focus on retrieving ϕ from X by the use of a Convolutional Neural Network (CNN).

With out of focus input image $X \in \mathbb{R}^2$ and corresponding wavefront phase ϕ expressed in Zernike coefficients by $y \in \mathbb{R}$ as defined in Chapter 2, the goal is to find a mapping of X onto y. This mapping is to be found by the Neural Network $\mathcal{P}$ as depicted in Eq. (3-1)

$$\mathcal{P} : X \rightarrow y \tag{3-1}$$

Depending on the chosen architecture, $\mathcal{P}$ holds multiple coefficient matrices that are to be found by a learning process based on gradient descent algorithms typically named backpropagation in Machine Learning (ML). Various so called optimizers can be used and will be tested in this chapter, though this report shall only focus on the forward propagation through the NN as this defines the architecture of the NN. A basic description of the optimization algorithms used for backpropagation are added as reference[22][23].

In the field of ML the best way to extract spatial information in so called features from images is by using a CNN[24]. Many success stories are available partly due to the availability of image datasets such as MNIST[25], SVHN[26], CIFAR-10/100[27] and Imagenet[28] on which many Neural Network (NN)'s are benchmarked and competitions are held. Since matching a CNN architecture to a specific problem is not an exact science, lessons learned and practitioners rule of thumb have to be used in order to

reduce the number of tuneable hyper-parameters to confine the search space for an optimal architecture.

Figure 3-1 reveals the task of the CNN, here the first 35 Zernike modes(piston removed) propagated from pupil to the out of focus plane are shown. Any random wavefront phase can be represented increasingly accurate by combining increasingly higher order Zernikes[1]. For this thesis the first 35 modes are selected due to the hardware available and the fact that these modes already represent 98.8% of the presented wavefront(see chapter 2) and is enough to prove the concept as a more accurate estimation of lower modes would be more beneficial then retrieving even higher order modes. The task of the CNN is to find features in intensity images that provide a measure for the contribution of each Zernike mode in the wavefront phase, the mode coefficient. The higher the radial mode of the Zernike the more difficult it will be to extract the coefficient since the incorporated power of higher modes decreases fast.

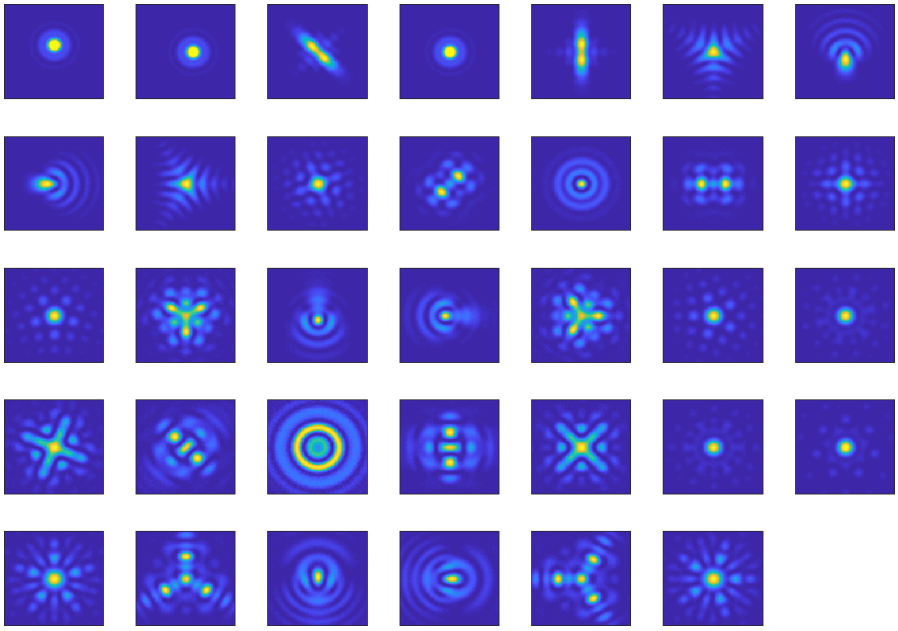

Figure 3-1: Propagation from pupil to out of focus plane for the first 35 Zernike modes(piston discarded)

The out of focus plane is used because it provides a better spread of the information. In an ideal in focus situation all the information concentrates in just a few pixels, which makes it very unlikely that features of the wavefront can be extracted or distinguished in enough detail. Figure 3-2 shows this principle, here the right image shows the out of focus intensity measurement of a wavefront which is likely to provide a better information source then the in focus(near perfect PSF) image on the left. In chapter 4 this hypothesis has been evaluated.

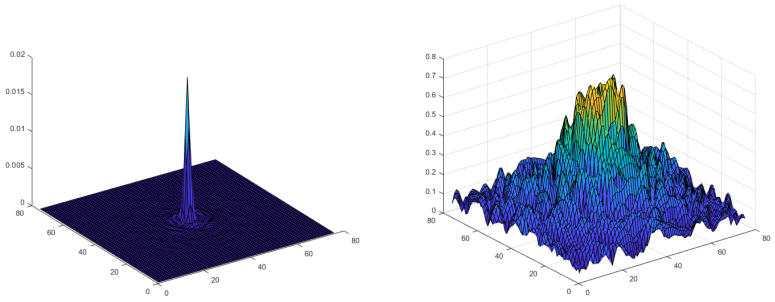

Figure 3-2: Example image of a wavefront propagated to the in focus plane(left) and out of focus plane(right). The right image shows better spatial separation of the information as it is spread more over the detector.

3-1 CNN building blocks

The following subsections provide a general introduction to the elements the CNN consists of and elaborates on choices for the CNN architecture.

3-1-1 Convolutional layers

A standard, commonly used convolutional layer consists of three subsequent parts. First there is the convolution, secondly the neuron activation and in order to reduce the array size pooling layers can be added. Since the input size of the CNN is a square array of large size(100x100 up to 200x200) it is evident that a deep CNN will have to be used when referring to achievements on the Imagenet dataset[28]. This dataset consists of 256x256 images on which successful CNN implementation had at least four subsequent convolutions and often more[29][30].

Convolutional layers are computationally expensive due to the many matrix multiplications involved. For a square input array with leg size M, kernel size K and number of channels C the required number of element wise matrix multiplications Q are;

$$Q = C(M - K + 1)^2 \tag{3-2}$$

and the needed Floating Point Operations (FLOPS) per element wise matrix multiplication of the filter and partial image, summation and averaging are:

$$FLOPS = 2K^2 + 1 \tag{3-3}$$

With that the total amount of FLOPS for one full convolution becomes:

$$FLOPS = 2CK^2(M - K + 1)^2 + 1 \tag{3-4}$$

After at least one convolution one can speak of a feature map which is a mapping of the input data onto the feature space holding accentuated or diminished characteristics of the

input. To the featuremap a bias is added to provide flexibility to the activation functions as it allows a lateral shift in the activation function ad described in 3-2-1 . Each element in the feature map is pulled through an activation function after each convolution to provide non-linearity to the otherwise linear transformation of the convolutions. Out of a broad range of activation functions the Rectified Linear Unit (ReLu) is baselined for the design and is provided in (3-5)

$$f(x) = \begin{cases} 0 & for \quad x < 0 \\ x & for \quad x \geq 0 \end{cases} \tag{3-5}$$

The reason to choose the ReLu above sigmoids, tanh or others is because ReLu's function well in deep networks due to the capacity to handle the phenomena of fading gradients when back propagating through the network for a weight update. Sigmoids as in (3-6) or tanh as in (3-7) have less suited derivatives since the gradients weaken exponentially when back-propagating through multiple activation layers while the derivative of a ReLu is a constant. Due to the expected deep architecture this is a critical argument to choose ReLu's for adding non-linearity.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3-6}$$

$$\sigma(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3-7}$$

Another advantage of the ReLu is that it is computationally efficient, it comprises one comparison and one multiplication making a 2 FLOPS total. Tanh and sigmoids require more sophisticated methods for approximating the outcome since the Central Processing Unit (CPU) needs to approximate these by a combination of simpler functions. Depending on the input value a tanh can easily consume more then 10 FLOPS[31] making them computationally expensive.

With this the computational cost of adding bias and the execution of the ReLu becomes:

$$FLOPS = 2C(M - K + 1)^2 \tag{3-8}$$

The third component of the standard convolution layer is the pooling layer. This operation reduces the size of the feature maps by averaging the sum of multiple neighboring matrix entries and replacing these with one average value. This way a 2D array can be resized from 200x200 to 100x100 by using a pooling size $P = 2$ (meaning a 2x2 pooling). The operation for either average pooling or maximum pooling comprises the following amount of FLOPS:

$$FLOPS = P^2(M/P)^2 \tag{3-9}$$

When summing and rearranging (3-4), (3-8) and (3-9) the computational cost of one full convolutional layer becomes:

$$FLOPS_{conv} = (2CK^2 + 2C)(M - K + 1)^2 + P^2(M/P)^2 + 1 \tag{3-10}$$

As indication for one convolution, activation and pooling with an input image of size 200x200, 32 channels and filtersize 3x3 this results in $24877761 FLOPS \approx 0.025 GFLOPS$. Knowing that mid-end contemporary GPU's can do up to $4 TFLOPS$(NVIDEA GTX1060), 160000 of these convolution can theoretically be performed per second.

3-1-2 Kernal considerations

More recent implementations of CNN's disposed the pooling layers and uses a stride higher then one for the 2D convolution in order to reduce size. Stride represents the stepping size of the kernal over the input image or featuremap as is shown in Figure 3-3.

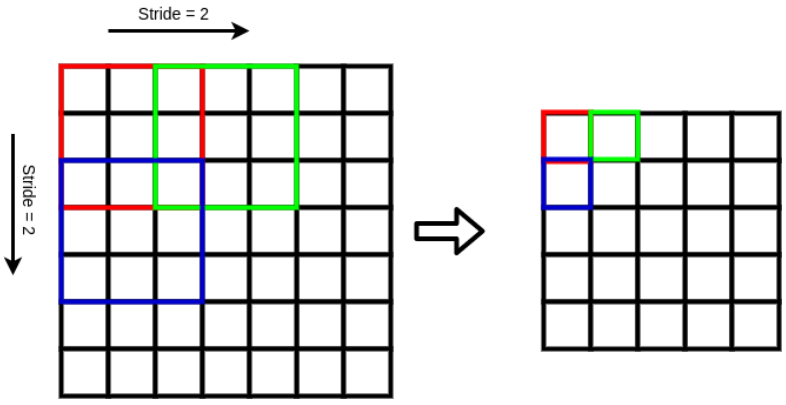

Figure 3-3: A filter moving over an image with a stride of two with the resulting featuremap on the right.

Since in theory a higher stride causes a loss of information Springenberg et al[32] investigated how this compares to using pooling for reducing the resolution of the features in the feature maps. Springenberg stated 'In particular, as opposed to previous observations, including explicit (max-)pooling operations in a network does not always improve performance of CNNs. This seems to be especially the case if the network is large enough for the dataset it is being trained on and can learn all necessary invariances just with convolutional layers.' With this in mind some of the to be tested CNN architectures will be designed on this principle.

In addition to the above Simonyan[33] experimented with the implementation of multiple subsequent small 3x3 filters in between of pooling layers instead of using one larger 7x7 filter and having a pooling layer after each convolution. The reason to do this was to reduce the computational cost of the convolutions since a 7x7 filter requires $2 * 7^2 + 1 = 99 FLOPS$ according to (3-3) while 3 subsequent 3x3 convolutions require $3 * (2 * 3^2 + 1) = 57 FLOPS$. The architectures are trained on a classification task for 224x224 Red Green Blue (RGB) images. It was found that the datasets generalizes well over a variety of different datasets and have matching or outperforming results compared to more complex but less deep CNN's. This shows the importance of having enough dept in the NN which will be accounted for in the test architectures which follow in 3-2.

Another recent method presented by Rastegari et al[34] demonstrated the use of binary kernals, meaning that the kernal weights could only be one or zero. This dramatically decreases the computational load and showed to perform well on the ImageNet dataset but again information loss is evident due to which this method will not be regarded as the primary goal is to first be able to retrieve the phase from intensity measurements.

3-1-3 Fully connected layers

Convolutions are needed to extract features from the input pixels and output a representation of the data with reduced dimensions. This representation has to be followed up by dense layers to provide the abiltiy to translate the representation of the data to Zernike modes. Fully connected or often called dense layers can provide this ability. In Figure 3-4 three subsequent layers of neurons are shown. The output of each neuron is fed to all neurons in the subsequent layer but not before it is multiplied by a weight. When zooming in on one neuron as in Figure 3-5 the internal workings are shown. In

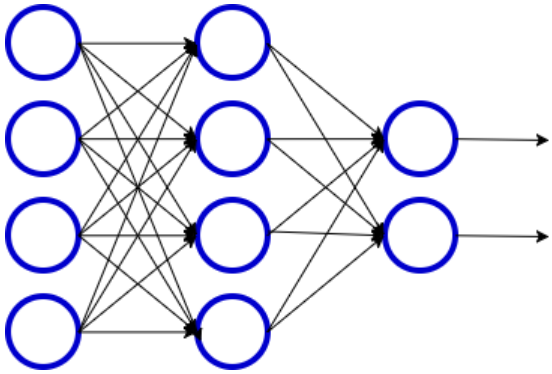

Figure 3-4: Three subsequent fully connected layers

the neuron the inputs are multiplied with the weights, are summed, bias is added and the activation function g provides the output of the neuron. With that the activation value a of neuron i becomes:

$$a_i = g(\sum_{1}^{i} x_i w_i + b) \tag{3-11}$$

Here x_i is a vector with the incoming values from the foregoing layer and w_i the weight vector for neuron i. One can decide to not connect all neurons to each other to create subregions. Though when looking at the objective and Figure 3-1 the input is a modal image on which each pixelvalue is a combination of all modes due to which no regions for specific modes exist. Therefore fully connected layers are a must to be able to differentiate between modes.

Generally the first dense layer after the CNN has the same size as the CNN output after which the size is quickly reduced in three to four layers to the wanted output size which

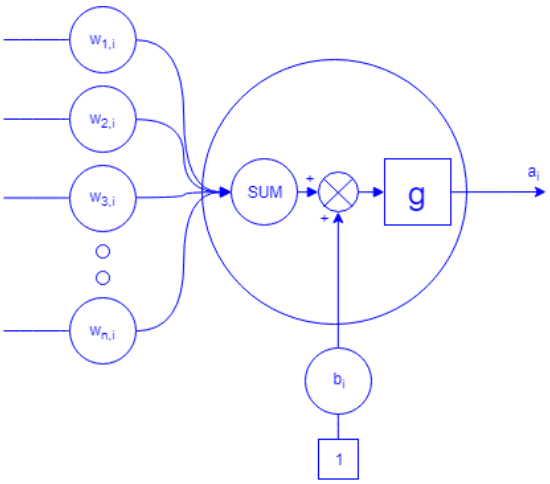

Figure 3-5: The components of a neuron in a dense layer

is the number of Zernike modes to retrieve. As this is common and since there is no reason to do otherwise this implementation is chosen.

3-2 Mathematical representation of the starting architecture

To gain insight in the set of architectures to further analyze a more general CNN is defined first to eliminate hyper parameters that have little to no influence on the networks convergence. For an input X with size 150x150 often three to five subsequent standard convolutional layers are used with max-pooling layers in between followed by a few dense layers. Therefore a basic configuration based on three convolutions is chosen to start with as schematically depicted in Figure 3-6.

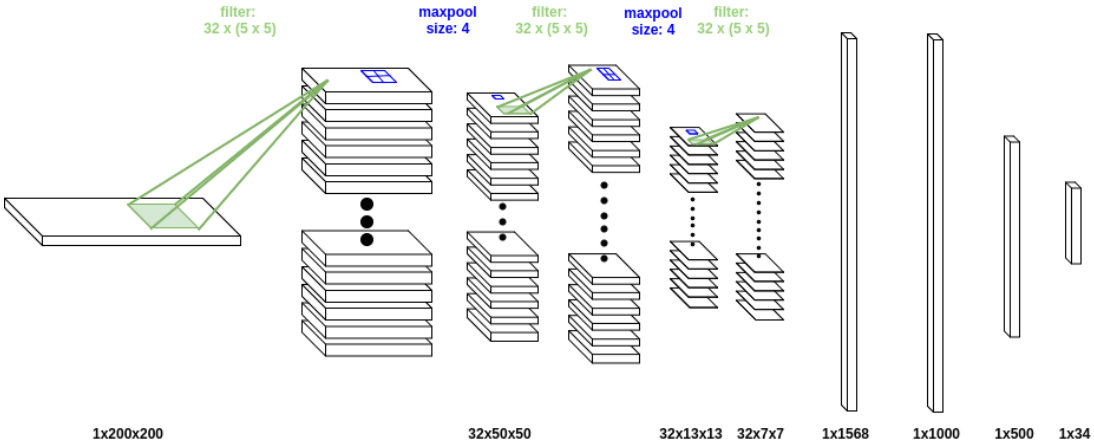

Figure 3-6: Starting architecture for instantaneous mode retrieval from intensity measurements. On the left the input image is shown which is processed by three convolutions to size 32x7x7, flattened to a 1D vector of size 1568 and fed through three dense layers.

To provide detailed insight to the computations the CNN performs the mathematical operations of one discrete forward pass executed for the mapping $\mathcal{P}$ as defined in 3-1 are described below.

First the 2D pixel data X of size 150x150 is normalized to prevent spurious weight updates during training as provided in (3-12).

$$X_{norm} = \frac{X}{2^{nbits}} \tag{3-12}$$

Here $nbits$ is the resolution of the detector in bits and X_{norm} is the normalized input to the first 2D discrete convolution as provided in (3-13).

$$Y_1(i,j,c) = \sum_{c=1}^{C} \sum_{m=0}^{M} \sum_{n=0}^{N} K_1(m,n,c) X_{norm}(i-m, j-n) \tag{3-13}$$

Where; $0 \leq i < M+N-1$ and $0 \leq j < M+N-1$

Y_1 is a 3D array holding C featuremaps of size ij, K_1 is a 3D array holding C 2D kernals. M and N are respectively the legsize of the square input array and kernal.

After the convolution the bias term $B_1 \in \mathbb{R}^3$ is added to the output of the convolutions to provide flexibility for the subsequent ReLu activation function;

$$Y_2 = \begin{cases} 0 \quad for \quad Y_1 + B_1 < 0 \\ Y_1 + B_1 \quad for \quad Y_1 + B_1 \geq 0 \end{cases} \tag{3-14}$$

Thirdly the 3D array holding the activation output Y_3 is downsized by an average pooling function described by;

$$Y_3(i,j,c) = \sum_{c=1}^{C} \frac{\sum_{m=0}^{P} \sum_{n=0}^{P} Y_3(i-m, j-n, c)}{P^2} \tag{3-15}$$

Since the architecture consists of three convolutional layers, Eq. (3-13), (3-14) and (3-15) are repeated another two times in order to arrive at Y_9;

$$Y_4(i,j,c) = \sum_{c=1}^{C} \sum_{m=0}^{M} \sum_{n=0}^{N} K_2(m,n,c) Y_3(i-m, j-n, c) \tag{3-16}$$

$$Y_5 = \begin{cases} 0 \quad for \quad Y_4 + B_2 < 0 \\ Y_4 + B_2 \quad for \quad Y_4 + B_2 \geq 0 \end{cases} \tag{3-17}$$

$$Y_6(i,j,c) = \sum_{c=1}^{C} \frac{\sum_{m=0}^{P} \sum_{n=0}^{P} Y_5(i-m, j-n, c)}{P^2} \tag{3-18}$$

$$Y_7(i,j,c) = \sum_{c=1}^{C} \sum_{m=0}^{M} \sum_{n=0}^{N} K_3(m,n,c) Y_6(i-m, j-n, c) \tag{3-19}$$

$$Y_8 = \begin{cases} 0 \quad for \quad Y_7 + B_3 < 0 \\ Y_7 + B_3 \quad for \quad Y_7 + B_3 \geq 0 \end{cases} \tag{3-20}$$

$$Y_9(i,j,c) = \sum_{c=1}^{C} \frac{\sum_{m=0}^{P} \sum_{n=0}^{P} Y_8(i-m, j-n, c)}{P^2} \tag{3-21}$$

Due to the matrix size reducing pooling layers of Eq. (3-15), (3-18) and (3-21) the resulting matrix Y_9 is of size 7x7x32 and has to be vectorized to the column vector a_0 of size 1568 before feeding to the dense layers.

$$a_0 = vec(Y_9)^T \tag{3-22}$$

a_0 is fed to the first dense layers with a tangens hyperbolicus activation so that the CNN output $\hat{y}$ becomes;

$$a_1 = \tanh(a_0 W_0 + b_0) \tag{3-23}$$

$$a_2 = \tanh(a_1 W_1 + b_1) \tag{3-24}$$

$$\hat{y} = \tanh(a_2 W_2 + b_2) \tag{3-25}$$

Here a_1 and a_2 are the actuation values of dense layer one and two, $W_0 \in \mathbb{R}^2$ is of size 1568x1000, $W_1 \in \mathbb{R}^2$ of size 1000x500 and $W_2 \in \mathbb{R}^2$ of size 500x34. b_0, b_1 and b_2 are vectors respectively of size 1568, 1000 and 500.

During the CNN training the Zernike vector output $\hat{y}$ is used to calculate the error or loss with the reference output y as defined in Eq. (2-9) by;

$$loss = \sum(\hat{y} - y)^2 \tag{3-26}$$

With input X and y being an example pair (X, y) of a larger dataset $X \in \mathbf{X}$ and $y \in \mathbf{y}$ with V entries, the CNN mapping $\mathcal{P}$ as defined in Eq. (3-1) for this CNN architecture becomes $\mathcal{P}(X_i) = \hat{y}_i$ for a single input pair . Combined with the error function Eq. (3-26) the value U to minimize by the backpropagation process is defined by;

$$U = \frac{1}{V} \sum_{i=1}^{V} (\sum(\mathcal{P}(X_i) - y_i)^2) \tag{3-27}$$

So that the minimization problem is defined by;

$$\min_{X \in \mathbf{X}} U \tag{3-28}$$

U is minimized using the process of backpropagation to update the weight matrices W_2, W_1, W_0 and biases b_2, b_1 and b_0 in the dense layers, followed by the kernals K_3, K_2, K_1 and biases B_3, B_2 and B_1 of the convolutions. This update process is based on gradient descent principles which are out of scope of this thesis though various often used inbuilt algorithms of Tensorflow are tested in the next section.

3-2-1 Findings for the starting architecture

For the evaluation of the starting architecture the in- and output model defined in chapter 2 is used to generate data for the first 35 Zernike modes with a turbulence strength of $D/r_0 = 10$. The starting architecture was evaluated for the following combinations of hyper-parameters:

- Batchsizes of 2, 4, 8, 16 and 32.
- Learningrate varying from 10^{-3}, 10^{-4}, 10^{-5} and 10^{-6}

- Optimizers: A method for stochastic optimization (ADAM)[35], Gradient Descent (GD), Stochastic Gradient Descent (SGD)

After training the CNN with the varying parameters it is found that the batch-size has little to no influence on the training process. Further it is found that the Learning Rate (LR) greatly influences the convergence when chosen too big. Just as with common gradient optimizers a too large stepsize can cause the algorithm to step over a (local) minimum or even diverge. For the CNN it is found that for a LR of 10^{-4} or smaller no noticeable decrease in the loss function is noticed. To remain on the safe side a LR of 10^{-5} is chosen for the to be tested CNN's.

ADAM is an advanced optimizer that computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. It gained a lot of popularity since 2014 and is suited for a broad range of NN's. For the starting architecture it provides convergence to a loss 30% lower then the other optimizers. Therefore it is the baselined optimizer for the architectures where the learning rate $\alpha = 10^{-5}$, combined with a first moment exponential decay rate $\beta_1 = 0.9$ and second moment exponential decay rate $\beta_2 = 0.999$ are chosen as these are common parameters to use.

Increasing the amount of dense layers resulted in no significant change to the loss, therefor the amount of dense layers is fixed to three with a tangens hyperbolycus as activation function and the layer size is to decrease gradually. Since the activation a_l of a tangens hyperbolicus is always $-1 \leq a_l \leq 1$ a final layer is added in which only a linear scaling is performed by a weight vector.

3-3 Architectures

With the learned in the above sections a subset of architectures is defined for which the most important hyperparameters are set. This set of architectures is non-exhaustive and merely a subset of all possible definitions based on gathered information in this chapter. The architectures are provided a numbered name and are briefly described in the list below. Table 3-3 provides the detailed architectures and the layers they consist of.

- M001: With a kernal size of three, stride as implemented by Springenberg[32] is used to downsize the input by using six sequential convolutions.
- M002: The same architecture as M001 but with double the channels.
- M003: With a stride of three as implemented by Springenberg[32] the input is flattened to a 1024 size vector after 4 convolutions. To support the stride of 3 a kernal of size 5 is used.
- M004: The same architecture as M003 but with double the channels.

- M005: With a stride of three as implemented by Springenberg[32] the input is flattened to a 512 size vector after 3 convolutions. To support the stride of 5 a kernal of size 7 is used.
- M006: The same architecture as M005 but with four times the number of channels.
- M007: Based on Simonyan[33] multiple small kernals instead of one bigger one is used in between of pooling layers to reduce size. 2 pooling layers of size 4 are needed to downsize the input within a reasonable amount of layers.
- M008: The same architecture as M007 but with double the channels.
- M009: Conventional CNN setup with a kernal of size 3 followed by a pooling layer of size 4. Four layers are implemented and the last pooling layer is of size 3 so that a vector of size 512 is created.
- M010: Conventional CNN setup with a kernal of size 3 followed by a pooling layer of size 3. 5 layers are needed to downsize to a vector of size 512.
- M011: The same architecture as M010 but with double the channels.
- M012: Conventional three layer CNN setup with kernal size 5 and pooling layers of size 4. The last pooling layer is of size 5 so that a vector of size 1024 can be created.
- M013: The same architecture as M012 but with double the channels.

Table 3-1: Specification of CNN architectures to be evaluated(part 1). Abbreviations are; k = kernal size, s = stride or size for pooling layers, c = number of channels, p = padding, o = layer output size.

Model Name	FLOPS	Layers(activation function skipped for estatics)											
M001	36.6 MFLOPS	k = 3*3 s = 2 c = 32 p = 2 o=75*75*32	k = 3*3 s = 2 c = 64 p = 2 o=38*38*64	k = 3*3 s = 2 c = 128 p = 2 o=19*19*128	k = 3*3 s = 2 c = 256 p = 2 o=9*9*256	k = 3*3 s = 2 c = 512 p = 2 o=5*5*512	k = 3*3 s = 2 c = 512 p = 2 o=3*3*512	flatten 1*4608	dense 1*1000	dense 1*500	dense 1*y	scalar 1*y		
M002	72.2 MFLOPS	k = 3*3 s = 2 c = 64 p = 2 o=75*75*64	k = 3*3 s = 2 c = 128 p = 2 o=38*38*128	k = 3x3 s = 2 c = 256 p = 2 o=19*19*256	k = 3*3 s = 2 c = 512 p = 2 o=9*9*512	k = 3*3 s = 2 c = 1024 p = 2 o=5*5*1024	k = 3*3 s = 2 c = 1024 p = 2 o=3*3*1024	flatten 1*9216	dense 1*1000	dense 1*500	dense 1*y	scalar 1*y		
M003	46.8 MFLOPS	k = 5*5 s = 3 c= 32 p = 3 o=50*50*32	k = 5*5 s = 3 c= 64 p = 3 o=17*17*64	k = 5x5 s = 3 c= 128 p = 3 o=6*6*128	k = 5x5 s = 3 c= 256 p = 3 o=2x2x256	flatten s=2 1x1024	dense 1*1000	dense 1*500	dense 1*y	scalar 1*y				
M004	92.5 MFLOPS	k = 5*5 s = 3 c= 64 p = 3 o=50*50*64	k = 5*5 s = 3 c= 128 p = 3 o=17*17*128	k = 5x5 s = 3 c= 256 p = 3 o=6*6*256	k = 5x5 s = 3 c= 512 p = 3 o=2x2x512	flatten s=2 1x2048	dense 1*1000	dense 1*500	dense 1*y	scalar 1*y				
M005	70.9 MFLOPS	k = 7*7 s = 5 c = 32 p = 2 o=30*30*32	k = 7*7 s = 5 c = 64 p = 2 o=6*6*64	k = 7*7 s = 3 c = 128 p = 2 o=2*2*128	flatten 1x512	dense 1x256	dense 1*128	dense 1*y	scalar 1*y					
M006	293.7 MFLOPS	k = 7*7 s = 5 c = 128 p = 2 o=30*30*128	k = 7*7 s = 5 c = 256 p = 2 o=6*6*256	k = 7*7 s = 3 c = 512 p = 2 o=2*2*512	flatten 1x2048	dense 1x1000	dense 1*500	dense 1*y	scalar 1*y					
M007	36.7 MFLOPS	k = 3*3 s = 1 c = 32 p = 2 o=150*150*32	k = 3*3 s = 1 c = 32 p = 2 o=150*150*32	pool s = 4 o=38*38*32	k = 3*3 s = 1 c = 64 p = 2 o=38*38*64	k = 3*3 s = 1 c = 64 p = 2 o=38*38*64	pool s = 4 10*10*64	k = 3*3 s = 1 c = 128 p = 2 o=10*10*128	pool s = 4 o=3*3*128	flatten 1*1152	dense 1*576	dense 1*288	dense 1*y	scalar 1*y
M008	82.1 MFLOPS	k = 3*3 s = 1 c = 64 p = 2 o=150*150*64	k = 3*3 s = 1 c = 64 p = 2 o=150*150*64	pool s = 4 o=38*38*64	k = 3*3 s = 1 c = 128 p = 2 o=38*38*128	k = 3*3 s = 1 c = 128 p = 2 o=38*38*128	pool s = 4 o=10*10*128	k = 3*3 s = 1 c = 256 p = 2 o=10*10*256	pool s = 4 o=3*3*256	flatten 1*2304	dense 1*1152	dense 1*576	dense 1*y	scalar 1*y

Table 3-2: Specification of CNN architectures to be evaluated(part 2). Abbreviations are; k = kernal size, s = stride or size for pooling layers, c = number of channels, p = padding, o = layer output size.

Model Name	FLOPS	Layers(activation function skipped for estetics)													
M009	38.5 MFLOPS	k = 3*3 s = 1 c = 64 p = 2 o=150*150*64	pool s = 4 o=38*38*64	k = 3*3 s = 1 c = 128 p = 2 o=38*38*128	pool s = 4 o=10*10*64	k = 3*3 s = 1 c = 256 p = 2 o=10*10*256	pool s = 4 o=3*3*256	k = 3*3 s = 1 c = 512 p = 2 o=3*3*512	pool s = 3 o=1*1*512	dense 1*256	dense 1*128	dense 1*y	scalar 1*y		
M010	28.2 MFLOPS	k = 3*3 s = 1 c = 32 p = 2 o=150*150*32	pool s = 3 o=50*50*32	k = 3*3 s = 1 c = 64 p = 2 o=50*50*64	pool s = 3 o=17*17*64	k = 3*3 s = 1 c = 128 p = 2 o=17*17*128	pool s = 3 o=6*6*128	k = 3*3 s = 1 c = 256 p = 2 o=6*6*256	pool s = 3 o=2*2*256	k = 3*3 s = 1 c = 512 p = 2 o=2*2*512	pool s = 2 o=1*1*512	dense 1*256	dense 1*128	dense 1*y	scalar 1*y
M011	75.5 MFLOPS	k = 3*3 s = 1 c = 64 p = 2 o=150*150*64	pool s = 3 o=50*50*64	k = 3*3 s = 1 c = 128 p = 2 o=50*50*128	pool s = 3 o=17*17*128	k = 3*3 s = 1 c = 256 p = 2 o=17*17*256	pool s = 3 o=6*6*256	k = 3*3 s = 1 c = 512 p = 2 o=6*6*512	pool s = 3 o=2*2*512	k = 3*3 s = 1 c = 1024 p = 2 o=2*2*1024	pool s = 2 o=1*1*1024	dense 1*512	dense 1*256	dense 1*y	scalar 1*y
M012	41.2 MFLOPS	k = 5*5 s = 1 c = 32 p = 2 o=150*150*32	pool s = 4 o=38*38*32	k = 5*5 s = 1 c = 64 p = 2 o=38*38*64	pool s = 4 o=10*10*64	k = 5*5 s = 1 c = 128 p = 2 o=10*10*128	pool s = 5 o=2*2*128	flatten 1*512	dense 1*512	dense 1*256	dense 1*y	scalar 1*y			
M013	84.0 MFLOPS	k = 5*5 s = 1 c = 64 p = 2 o=150*150*64	pool s = 4 o=38*38*32	k = 5*5 s = 1 c = 128 p = 2 o=38*38*128	pool s = 4 o=10*10*128	k = 5*5 s = 1 c = 256 p = 2 o=10*10*256	pool s = 5 o=2*2*256	flatten 1*1024	dense 1*1024	dense 1*512	dense 1*y	scalar 1*y			

Chapter 4

Evaluation of CNN architectures

4-1 Evaluation parameters

For the evaluation of the Convolutional Neural Network (CNN)'s defined in section 3-3, input X and reference output y are generated with the model as defined in chapter 2. Pixel frames of size $150x150$ are created as cutout from a larger pixel array where padding was added to prevent border effects by the Fourier Transform. The first 35 modes are incorporated for $D/r_0 = 2, 8, 14, 20$ representing a very low to very high turbulent state. The batch size is fixed to 8 and the Learning Rate (LR) to 1^{-5} as found in chapter 3-2-1. Further an out of focus of $50mm$ on a focal length of $300mm$ is used. During training tip and tilt are dismissed by not incorporating these modes in the reference output vector so that $y = [y_{J_{min}}, y_{J_{min}+1}, y_{J_{min}+2}, ..., y_{J_{max}}]^T$ with $y_{J_{min}} = 4$ and $y_{J_{max}} = 35$ at the output to prevent these significant modes from dominating the learning process, also the extraction of higher modes is also more interesting from of a research perspective.

The only noise in the modeled system is introduced by the detector of which the Signal to Noise Ratio (SNR) is defined by Eq. (4-1) and is $22dB$ for a representative input power for Free Space Optical Communications (FSOC) of $P_i = 1mW$ and the Xenix cheetah CCD as with specifications as provided in Table 2-2.

$$SNR = \frac{Q_e(P_i/E_p/f_s)}{\sqrt{Q_e(P_i/E_p/f_s) + N_{dark} + \delta^2_{readout}}} \tag{4-1}$$

The CNN is coded in Python and uses the Tensorflow[36] package in a dedicated Conda environment with Cuda drivers installed as in appendix A. With Python as main execution environment the Matlab engine runs in the background to generate input data on the fly. The computation graphs for the architectures are defined in a dedicated python

file and all hyper parameters can be controlled from of a Microsoft Excel worksheet. The training process can be followed by a very simple Graphical User Interface (GUI). Screenshots of the GUI and worksheet can be found in appendix B. The full code is delivered on USB with this thesis.

With an unlimited source of input data due to the created model new epoch data frames and reference outputs are generated freshly during training. Due to this there is no need to separate an existing dataset in a train, test and evaluation part which will reduce the risk of over fitting significantly.

4-2 Evaluation with modeled input and output

13 models have been evaluated, each for 4 different turbulence strengths. On average the training took 14 hours per model. Models 1 to 7 were run for 600 Epochs with an epoch size of 1000 images and models 8 to 13 were run 400 epochs as these CNN's already converged by then. The loss is defined by Eq. (4-2), where y is the reference output, $\hat{y}$ the Neural Network (NN) output and J the Zernike mode number.

$$loss = \sum_{J=4}^{35} (y_J - \hat{y_J})^2 \tag{4-2}$$

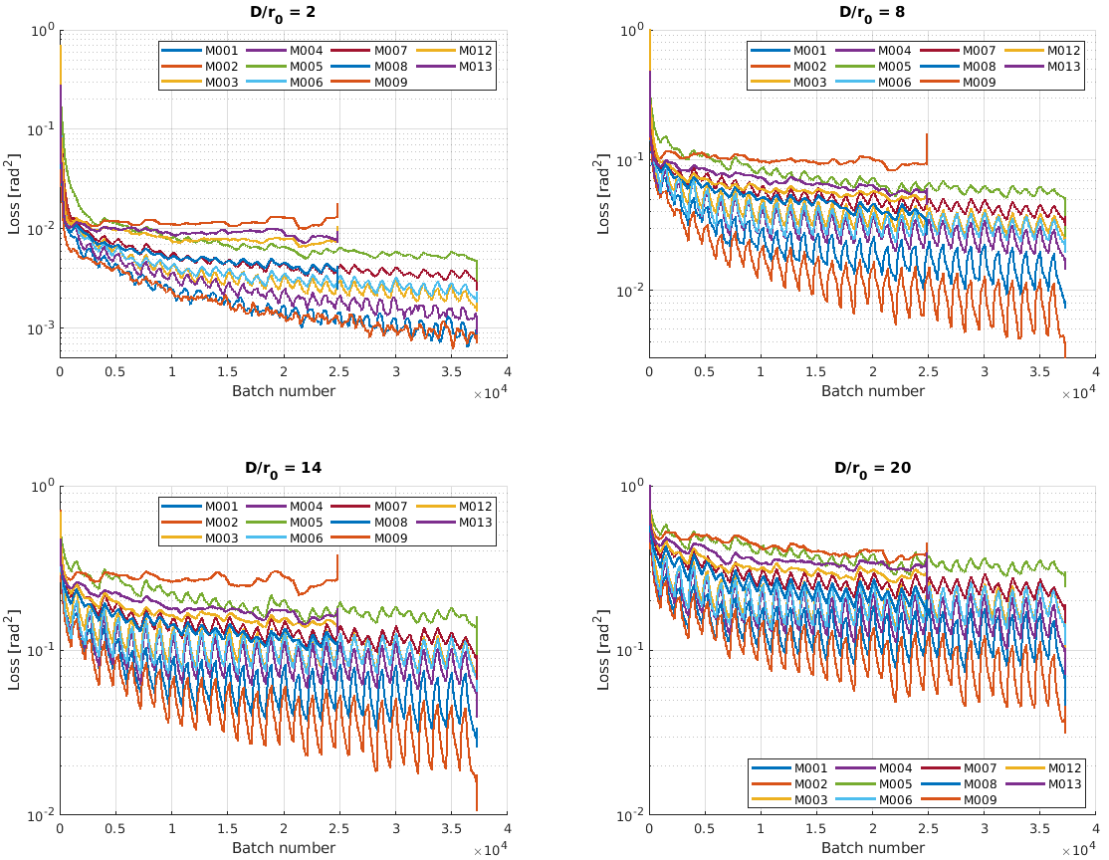

Figure 4-1: The training loss for all models and the various turbulence strenghts filtered by a moving average filter of size 500.

Figure 4-1 presents the training loss for each turbulence strength in a separate subplot and is filtered by a moving average filter of framesize 500. Model M010 and M011 are dismissed as these networks where unable to converge. Loading epoch train data dynamically during training appears advantageous as it denies the model to overfit to the input data as can be seen by the intermediate spikes in the plot, which indicate the loading of a fresh epoch.

It is observed that the performance degrades with increasing turbulence strength and that M001 and M002 outperform the other models. M002 has the same architecture as M001 but with double the channels. Therefore this indicates that more channels(feature maps) benefit the model outcome, this is also supported by the other architectures where the same phenomena is observed. More channels provide the ability to extract more features which seems logical as the input data has a more or less arbitrary shape instead of defined features as is often the case for a classification task where for instance the lines of a car form a feature. For this problem it seems that the featuremaps function as a bin system and with that a higher amount of bins results in a more accurate/less

generic placement in a bin.

Moreover, using stride outperforms pooling layers for reducing the dimensions of the input. Increasing the kernel size has a negative effect on the performance due to which we can conclude that using kernel size 3x3 for feature extraction performs best as choosing even smaller kernels eliminates the working principle of the convolution.

As M002 proved the best performance, Figure 4-2 depicts the loss of M002 in more detail. The loss for respectively $D/r_0 = 2$, $D/r_0 = 8$, $D/r_0 = 14$ and $D/r_0 = 20$ are 0.0008, 0.007, 0.02 and $0.07 rad^2$. When running the model on a fresh data set comprising 40000 new input images the loss is higher but with a same ratio between the turbulence strengths. The test losses are provided in Table 4-1.

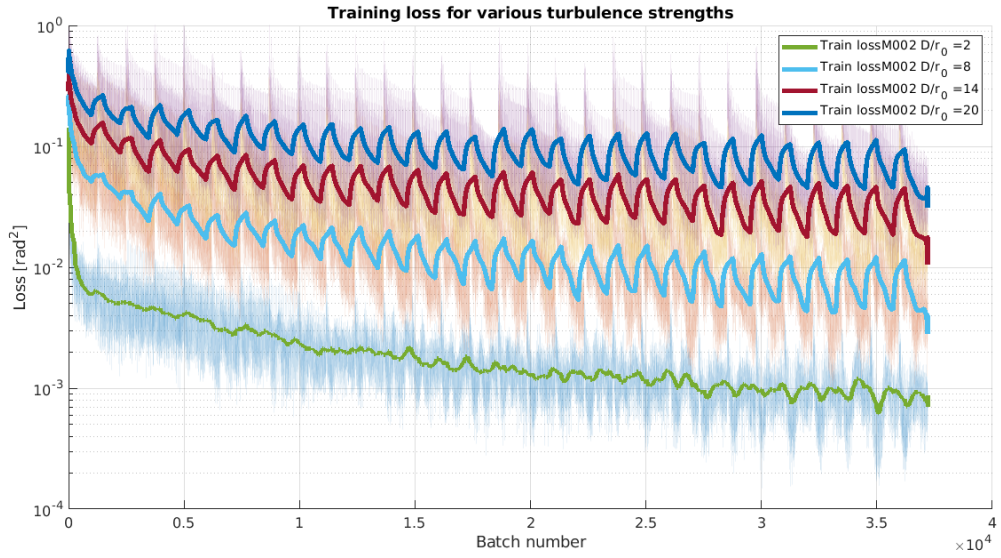

Figure 4-2: Filtered train loss for model M002 on four different turbulence strengths

Figure 4-3 shows the Mean Squared Error (MSE) per mode from of the fourth mode(no piston, tip and tilt). Overall it is noted that the error variance is of the same order of magnitude as the error itself which results in uncertainty in the retrieved phase modes.

When transforming $\hat{y}$ to a phase image in $\mathbb{R}^2$ and taking the MSE with the reference phase y one arrives at ϕ_{MSE} as presented in Eq. (4-3) with ϕ_{ref} and $\hat{\phi}$ representing respectively the reference phase and CNN output phase in radians and in $\mathbb{R}^2$ as provided by Eq. (4-4) and Eq. (4-5). For $D/r_0 = 2$ the resulting phase error is 15nm which is equal to a $\phi_{MSE}/\lambda = 1\%$ of the wavelength making it feasible to implement the CNN in a real setup. For $D/r_0 = 8$ the error is 15% of the wavelength which is to large for only a measurement error especially when other error sources are still to be included.

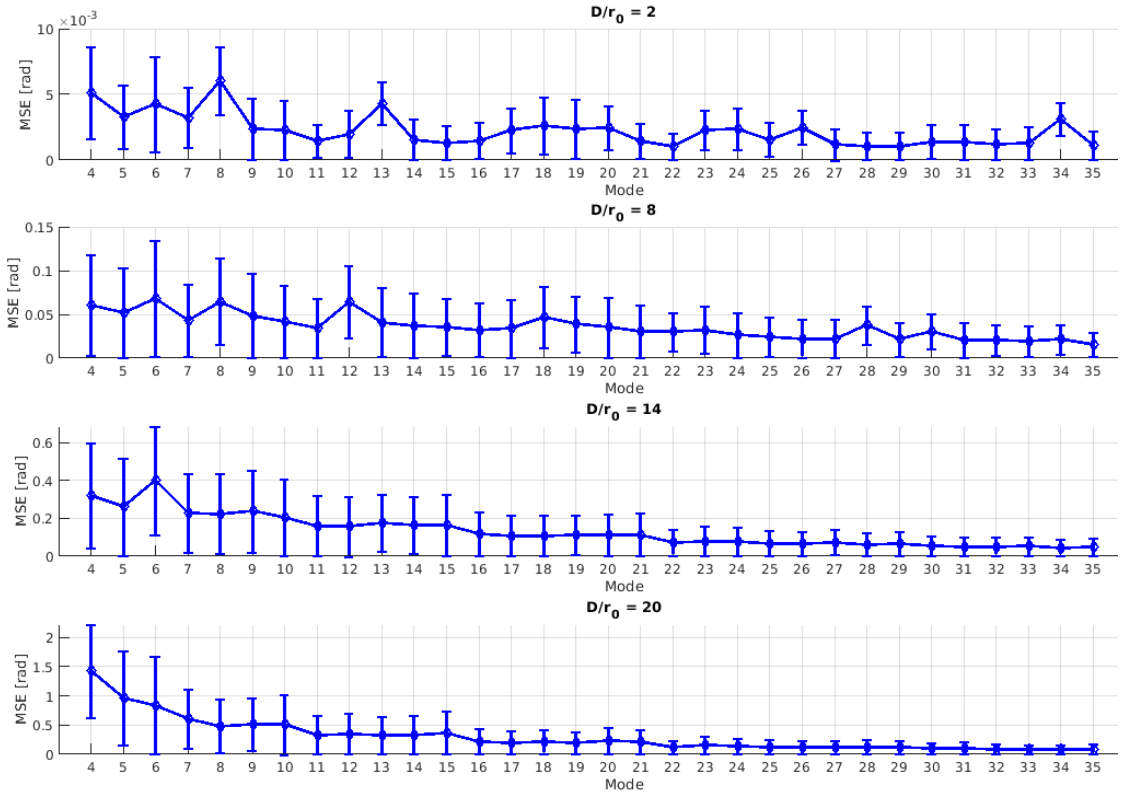

Figure 4-3: Mean squared error and variance per mode and turbulence strength.

$$\phi_{MSE} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(\phi_{ref}(i,j) - \hat{\phi}(i,j))^2}{n} \tag{4-3}$$

$$\phi_{ref} = \sum_{J=1}^{J_{max}} y_J \frac{Z_J}{\sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|Z_J(i,j)-\bar{Z}_J|^2}{n}}} \tag{4-4}$$

$$\hat{\phi} = \sum_{J=1}^{J_{max}} \hat{y_J} \frac{Z_J}{\sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|Z_J(i,j)-\bar{Z}_J|^2}{n}}} \tag{4-5}$$

.

Turbulence strength	Test loss	Mean squared phase error
$D/r_0 = 2$	$0.002rad^2$	$15nm$
$D/r_0 = 8$	$0.036rad^2$	$226nm$
$D/r_0 = 14$	$0.133rad^2$	$896nm$
$D/r_0 = 20$	$0.307rad^2$	$2238nm$

Table 4-1: Test loss and mean squared phase error for model M002 and various turbulence strengths.

A detailed analysis of the model convergence revealed that the biases of the kernals have an even spread in negative and positive values which indicates that sufficient flexibility in the model was available during training. Also Figure 4-4 shows the trained weights of the fifth convolution. From this seemingly random figure we can conclude that all weights in the kernals have converged and that no dead areas(all zeros or ones) exist in the CNN.

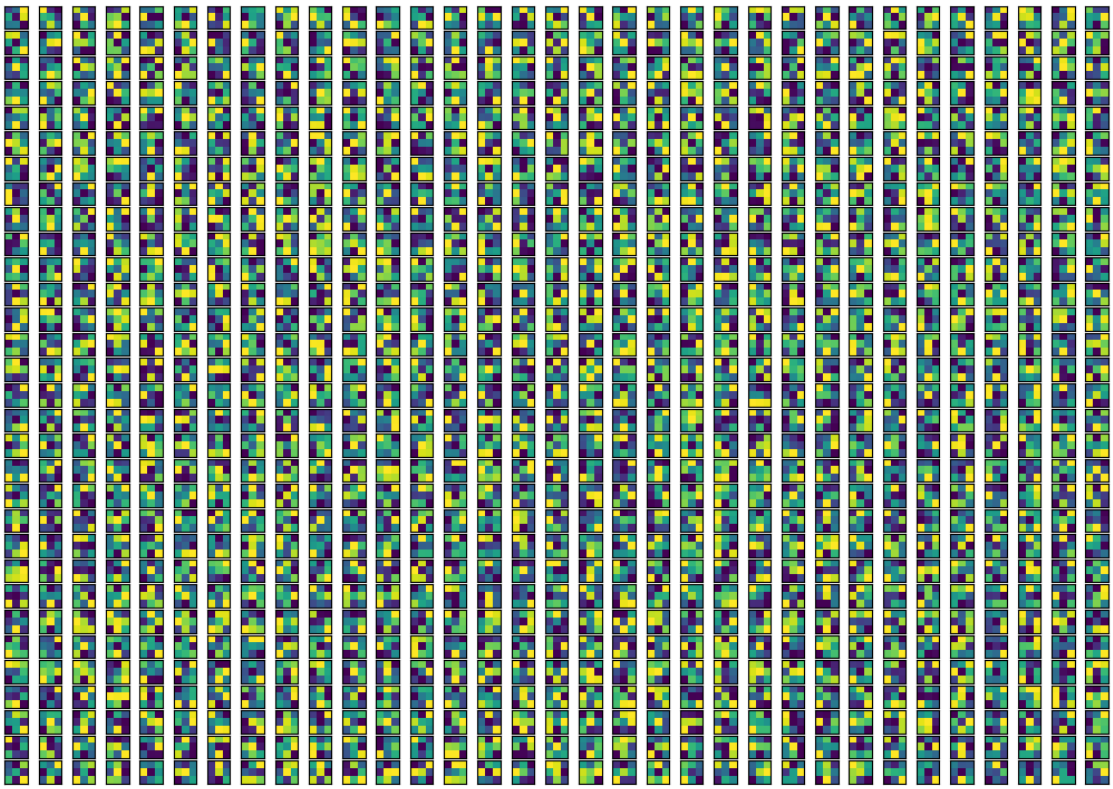

Figure 4-4: The trained weights of the 3x3 kernals in the fifth convolution. Yellow represents a 1 and blue a 0.

4-3 Evaluation with a kolmogorov phase

Now that a model is selected on a near perfect propagation of Zernike phase screens it is also tested on a Kolmogorov[14] turbulence. Also the model is augmented with more channels in the first layers to provide more 'bins' for feature extraction. Of the first three convolutional layers the number of channels is doubled so that these layers have respectively 128, 256, 512 and 1024 channels as can be seen in Figure 4-5. Further the input image size is reduced to a cutout of 70x70 pixels as little to no information is present on the image edges and the model is named M102. With these alterations the computational load sums to a total of 40 MFLOPS for one forward pass. This theoretically means that a Geforce GTX1060 graphics card would be able to run it at approximately 10kHz. Although the computations on a GPU will not be performed this efficient as latencies and memory bandwidth can be very restricting the order of magnitude of the load is realistic nevertheless.

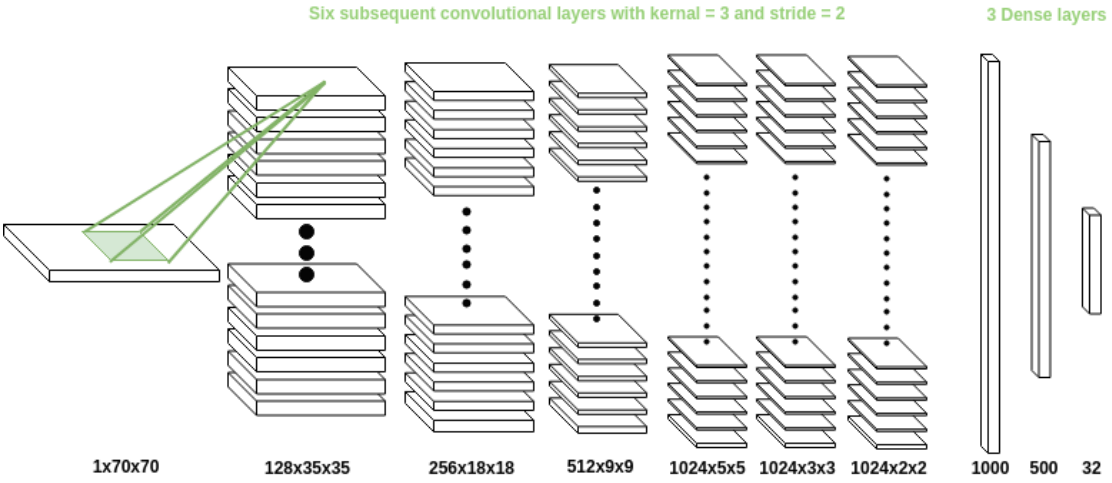

Figure 4-5: Model M102 consisting of six convolutional layers with kernalsize = 3x3, stride = 2 and ReLu activation.

Phase screens based on a Kolmogorov structure function are generated with an outer scale $L0 = 2m$ and an inner scale $l0 = 0.005m$. The aperture remains $D = 0.1m$ and r_0 is varied to generate turbulence strengths D/r_0 of 2, 4, 6 and 8. These lower turbulence strengths are chosen as a more severe turbulence was not feasible in section 4-2. The reference Zernike mode output y is created by a linear optimization algorithm that fits the Zernike profiles to the Kolmogorov phase.

Figure 4-6 presents the input data as it would be for the higher turbulence strenghts up to $D/r_0 = 20$ as used in 4. Figure 4-7 depicts the input to the CNN which is used for the evaluation of the Kolmogorov turbulence. It can be seen that the speckle pattern due to the turbulence is more realistic using a Kolmogorov spectrum as compared to using Zernikes.

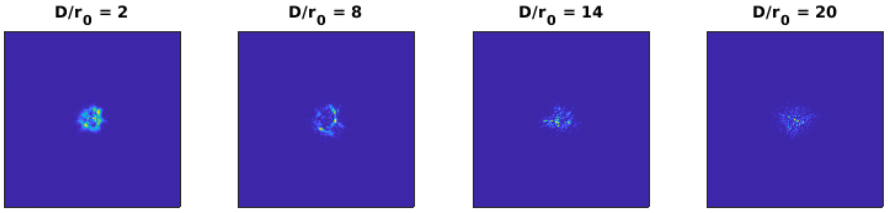

Figure 4-6: A Kolmogorov turbulence consisting of three phase layers of various strengths propagated to the 20% out of focus plane. Compared to pure Zernikes the speckle pattern is better visible.

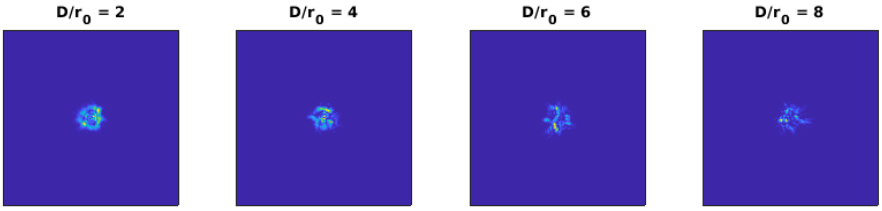

Figure 4-7: A Kolmogorov turbulence consisting of three phase layer for lower turbulence strengths propagated to the 20% out of focus plane. Compared to pure Zernikes the speckle pattern is better visible

4-3-1 In focus intensity measurements

Figure 4-8 shows the in focus intensity images of the same phase screens. Extracting the phase information from an in focus image has practical advantages because many Adaptive Optics (AO) systems already have a Focus Camera (FC) in their basic design and the input frame could be chosen much smaller which reduces the size of the CNN.

Intuitively one can see that it would be more difficult to extract information from the intensity pattern as the spatial spread of information is very limited. An evaluation of the M102 model(M002 with double the feature maps) showed that the CNN is unable to find a mapping as all weights in the model converge to zero. This indicates that using an in focus camera in combination with a CNN to retrieve the phase is not feasible, even for lower modes($3 \leq J \leq 14$). Training a CNN for tip and tilt retrieval from the in focus image might still be possible but is considered outside the scope of this report.

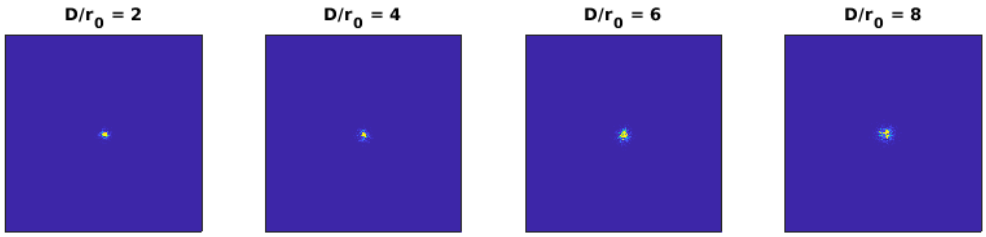

Figure 4-8: A Kolmogorov turbulence consisting of three phase layer of various strengths propagated to the in-focus plane. Compared to pure Zernikes the speckle pattern is better visible

4-3-2 Results

Training model M102 for 700 epochs with an epochsize of 1000 resulted in the training loss as depicted in Figure 4-8. Due to the extra feature maps the model was able to converge further, though the spikes in the unfiltered data in the background indicate the model experiences difficulties due to overfitting. This is also confirmed as the mean squared wavefront error on a 40000 frame test dataset is 94nm, 364nm, 718nm and 1263nm for respectively the lowest to highest turbulence strength.

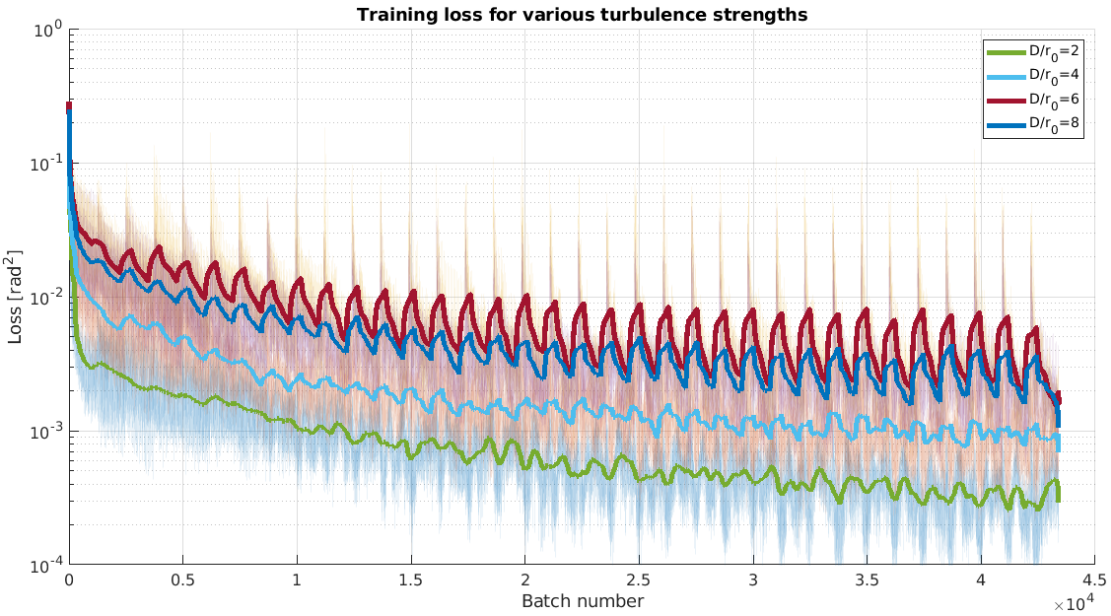

Figure 4-9: A Kolmogorov turbulence consisting of three phase layer of various strengths propagated to the in-focus plane. The large spikes in the background indicate overfitting to the epoch data.

A rerun on the exact same model but with a very small epoch size of 100 frames and refreshing this epoch every 10 epochs improves the training significantly as the test

results are comparative to the train results. Table 4-2 presents the outcome. It is noted that the MSE wavefront error is higher compared to the Zernike phase which can be explained as the Kolmogorov phase introduces a more sophisticated phase screen including high frequency components that are not captured in the first 35 modes due to which it can be regarded an additional noise signal to handle by the CNN.

Turbulence strength	Test loss	Mean squared phase error
$D/r_0 = 2$	$0.004 rad^2$	$37nm$
$D/r_0 = 4$	$0.019 rad^2$	$133nm$
$D/r_0 = 6$	$0.037 rad^2$	$287nm$
$D/r_0 = 8$	$0.069 rad^2$	$455nm$

Table 4-2: Test loss and mean squared phase error for model M102 and a Kolmogorov phase without overfitting.

The convergence and reduction of the loss fucntion during training of the CNN as shown in figure 4-9, but also as found in section 4-2, has proven that a mapping $\mathcal{P}$ can be found to directly relate the out of focus intensity image to the Zernike modes present in the wavefront.

4-4 Kolmogorov phase with varying turbulence strength

In a practical implementation multiple CNNs could be trained and switched among in real time when the strength of the turbulence changes. Therefore model M102 is also trained against a dataset with varying D/r_0. For training the same parameters as in 4-3 are used but for each phase frame D/r_0 is randomly generated on a uniform distribution within the range $1 \leq D/r_0 \leq 6$, $7 \leq D/r_0 \leq 12$ and $13 \leq D/r_0 \leq 18$. After training the mean squared residual error per mode was found and presented in Figure 4-10.

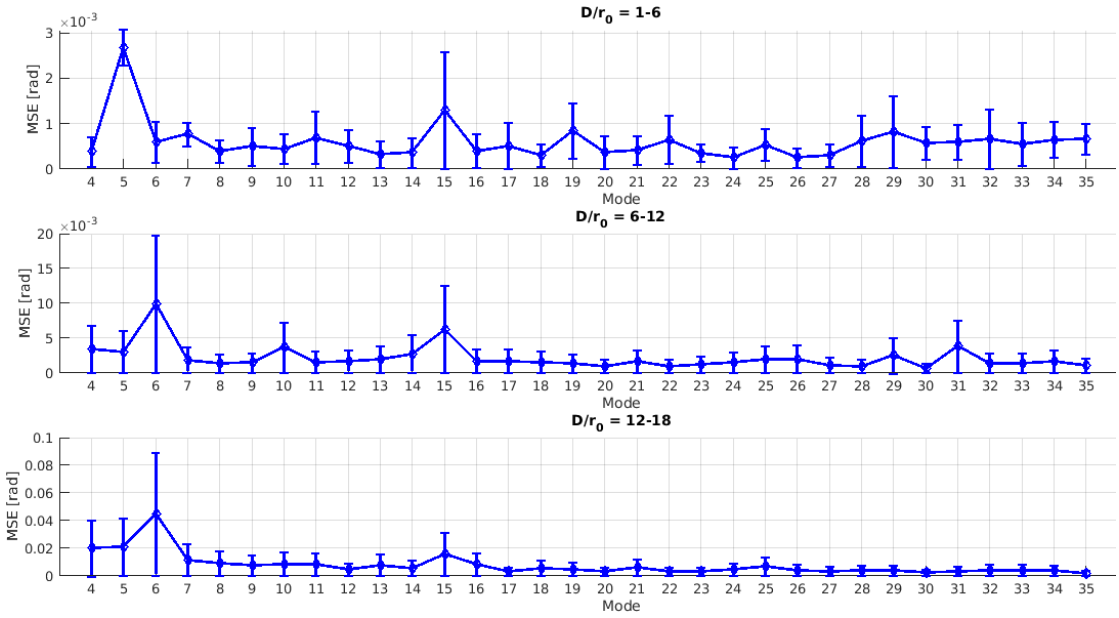

Figure 4-10: Mean squared residual phase error after training with bounded D/r_0

Frankly the mean squared residual phase error is drastically reduced compared to training on fixed turbulence strengths. Apparently training on a more diverse dataset ameliorates the convergence of the kernals. The overall performance indicators are shown in Table 4-3 where the mean squared phase error again is based on (4-3).

Turbulence strength	Test loss	Mean squared phase error
$D/r_0 = 1-6$	$0.00059 rad^2$	$5nm$
$D/r_0 = 6-12$	$0.0022 rad^2$	$21nm$
$D/r_0 = 12-18$	$0.0077 rad^2$	$74nm$

Table 4-3: Test loss and mean squared phase error for model M102 and a Kolmogorov phase without overfitting.

Figure 4-11 presents a selection of the featuremaps that are the result of the convolutions. As expected no clear features are shown as would be the case with for instance a car where often lines or edges are highlighted in the featuremaps. Although low order aberrations

can be distilled from Focal Plane (FP) images by the human eye no clear distinction of these modes is found in the featuremaps.

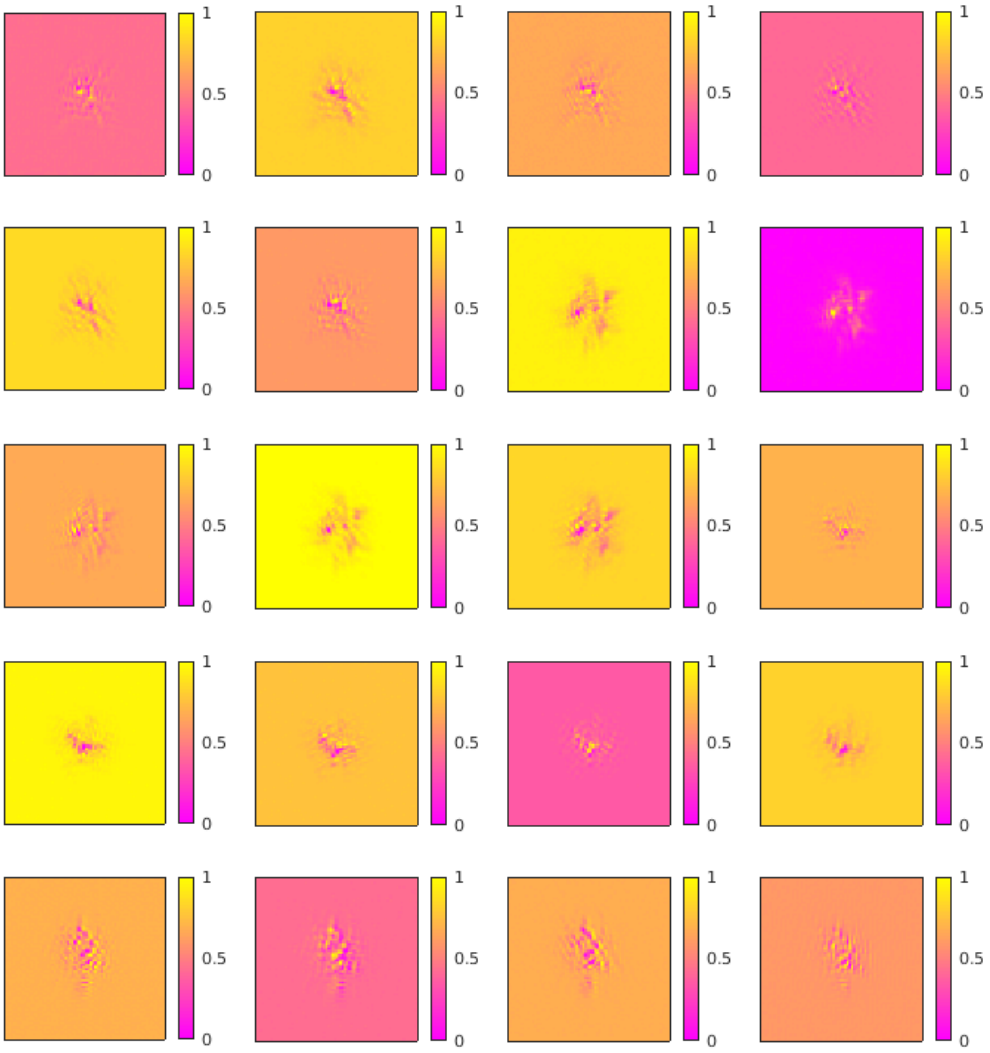

Figure 4-11: Four different featuremaps on the columns for five subsequent convolutional layers on the rows.

Chapter 5

Experiment

5-1 Goal of the experiment

With the found results based on modeled input and output data an experimental setup is build to generate real data on which the found Convolutional Neural Network (CNN) is trained and evaluated to demonstrate the working principle. This chapter describes the experimental setup used, the tests performed and presents the results.

5-2 Hardware setup

Figure 5-1 presents the schematic layout of the experimental setup. An existing TNO setup was used as basis and used a visible light spectrum laser which aided the calibration process. The hardware items in the setup are listed here below:

1. 0.02mW 635nm laser point source.
2. 50mm convex lens for collimating the beam(f=250)
3. Diaphragm resizing the beamwidth to 13mm.
4. Deformable mirror
5. Convex lens(f = 200mm)
6. Convex lens(f = 60mm) to resize the beam to 4mm.
7. Beamsplitter

8. Imperx 400 microlenslet Shack-Hartmann wavefront sensor as in Figure 5-2.

9. FO442SB IEEE 1394 monochrome CCD focus camera with a resolution of 1392x1024 pixels, framerate of 15fps and a pixelsize of 6.45μm. The camera is placed 20% out of focus on a focal length of f=150mm of the preceding lens.

The used Deformable Mirror (DM) is an OKO 37 channel mirror with the piezo electric actuators arranged in a hexagonal grid covering a circular surface of 15mm. The maximum deflection at the center of the mirror is 9μm.

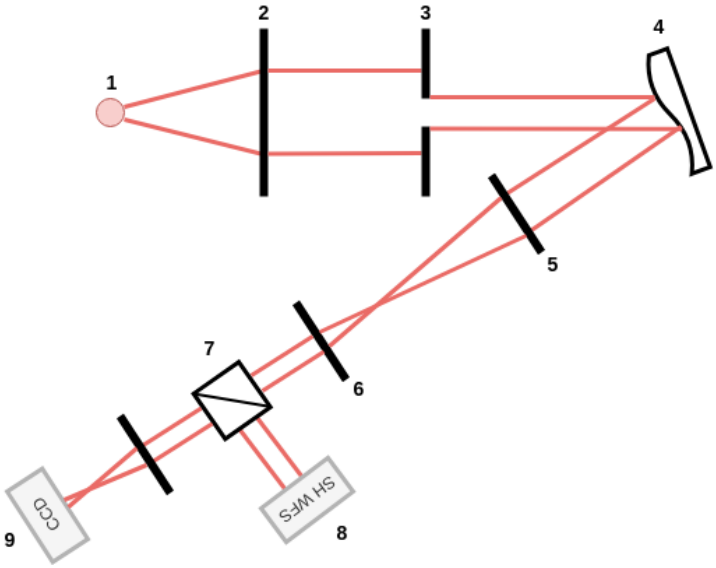

Figure 5-1: Hardware layout of experimental setup.

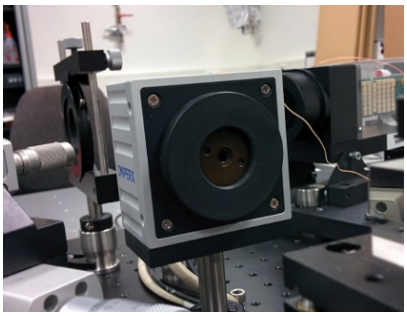

Figure 5-2: 400 microlenslet Schack-Hartmann wavefront sensor used in the experimental setup.

5-2-1 Calibration

Before the DM was placed the setup was calibrated using a static mirror. After the correct alignment procedures were followed to line up the optics the Shack-Harmann (SH) Wavefront Sensor (WFS) was calibrated with existing software. A new reference grid was generated and the power of the laser was adjusted for optimal performance. Figure 5-3 presents the CCD intensity pattern gathered by the WFS with an overlaying grid of 304 active lenslet spots used to reconstruct the wavefront from. Figure 5-4 shows a snapshot of the 20% out of focus Focus Camera (FC) with the static mirror installed.

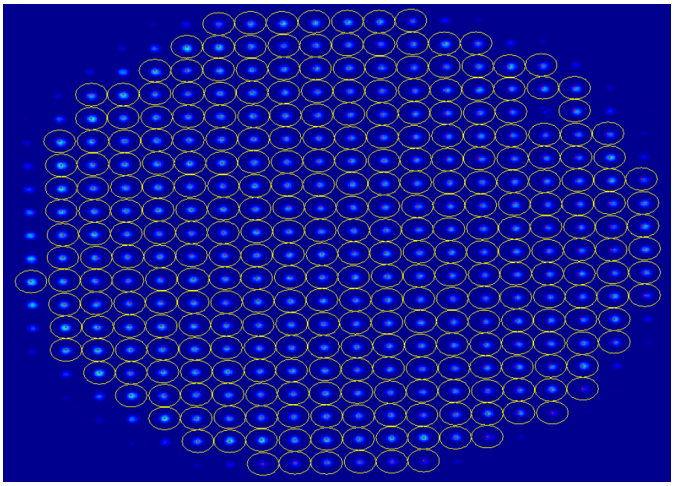

Figure 5-3: Calibrated SH WFS with the grid points defined for the static mirror setup.

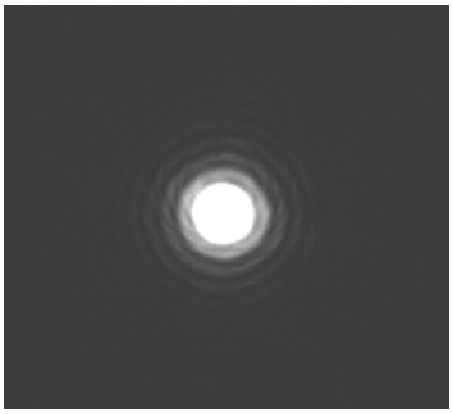

Figure 5-4: Out of focus snapshot of the focus camera using the static mirror(right)

5-2-2 Process steps

Due to time and resource restrictions the decision was made to separate the learning process from the data generation with the test setup. Therefore the test setup was only used to generate the in- and output data X and y which was then stored to the hard drive, after which the training of the CNN was performed separately. This speeded up the data generation and allowed for an intermittent learning process due to which risk on setup failure(long experiment) was mitigated.

Figure 5-5 provides the steps required to acquire the data. First a Zernike phase screen is generated based on the input model as defined in Chapter 2. Based on this phase screen a DM actuator command is generated and send via USB to a Digital Analog Converter (DAC), generating the input voltages for the DM. After confirmation that the DM is set a snapshot is taken with the out of focus camera which is then stored as an input image X.

Simultaneously with the taken snapshot the wavefront as generated by the DM is reconstructed by the SH WFS on a realtime computer(existing hardware and software[37]) and sent to the Control PC on which the main loop runs. From the reconstructed phase the Zernike coefficients are calculated and stored to disk as reference output y. This is needed because the generated Zernike phase screens with the model deviate from the actual wavefront set by the DM due to suboptimal DM software control.

Matlab is used to control the overall software loop and the generation of the phase screens. Python is used to connect the FC and the extraction of the subregion from a snapshot. The DM control is initiated by Matlab which calls a custom windows executable that takes a vector of size 37 representing the DM actuator setting for a specific wavefront.

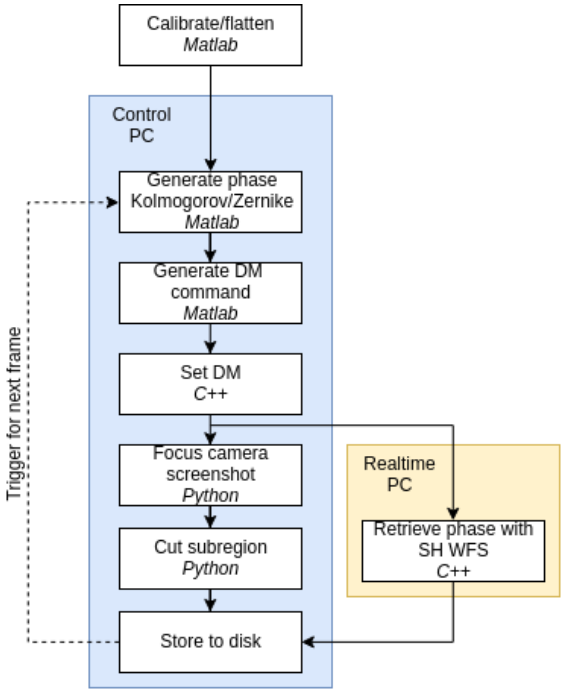

Figure 5-5: Process flow for the generation of experimental data.

5-3 Generated data

Due to latency in the system(DM controller), used hardware, computations, shutter and acquisition time it takes an average of 2.5 seconds to process a single frame. Therefore the amount of frames that could be generated was limited, though a representative dataset was generated consisting of 85000 frames. As no advanced drivers were available for the DM control a basic program was created that maintained the individual ratios of the Zernike modes in accordance with Noll but could not accurately set the DM with a specific turbulence strength. Therefore D/r_0 was calculated afterwards based on the retrieved phase of the SH WFS using Eq. (5-2) which is typically used for a Kolmogorov spectrum but will also provide a good estimate here as the individual Zernike mode contributions are proportional to the Noll residual error. Figure 5-6 presents a histogram of the turbulence strength distribution in the dataset. As seen in 4-4, training on a spread of turbulence strenghts improves the convergence of the CNN. 90% of the data is generated between $5 < D/r_0 < 15$ which supports this finding.

$$\phi_{var} = 1.03(D/r_0)^{5/3} \tag{5-1}$$

So that D/r_0 becomes;

$$D/r_0 = \frac{\phi_{var}^{3/5}}{1.03} \tag{5-2}$$

Figure 5-7 shows a few example plots of the generated data with on top the retrieved phase by the SH WFS and at the bottom the recorded FP image corresponding to this phase. To fully capture the intensity pattern the cutout from a full size snapshot was set to 170x170.

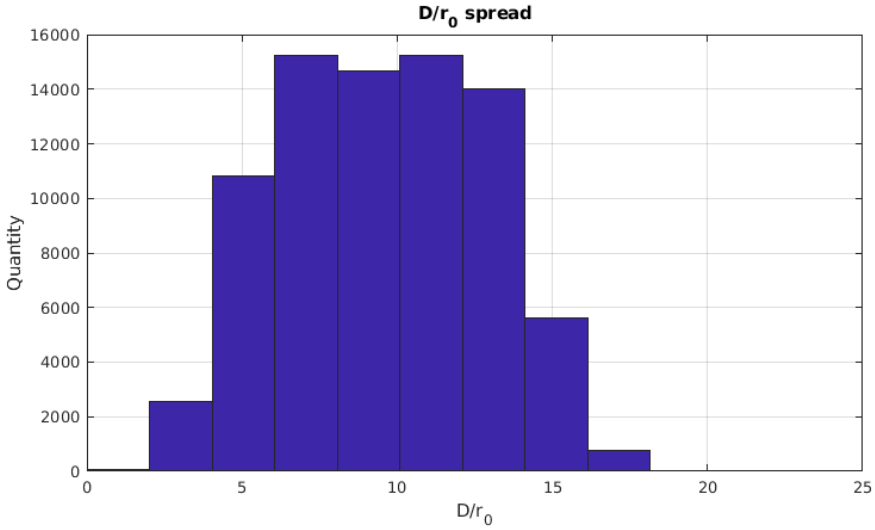

Figure 5-6: Histogram of the experimental dataset showing the distribution of the turbulence strength.

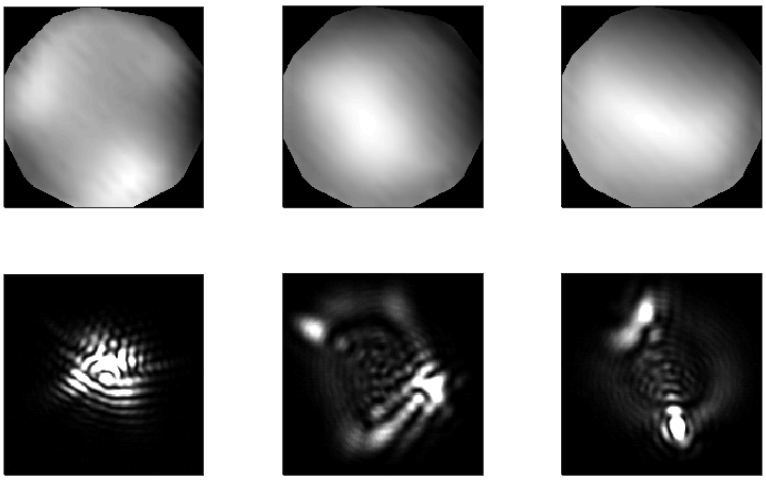

Figure 5-7: Example data generated by the experimental setup. On top the phase screen as retrieved by the SH WFS and at the bottom the recorded FP intensity pattern of size 170x170 pixels.

5-3-1 Eperimental results

Due to the enlarged input image size X the model is slightly altered to fit this input. With the convolutional layers as in M102 the resulting output size of the sixth convolution has become 3x3x1024, therefore the weight matrix to the first dense layer is altered to a size of 1000x9216 to provide a proper fit. Also the computational cost has increased to 154MFLOPS which is nearly a four time increase.

Figure 5-8 presents the convergence of to a loss of $0.61 rad^2$ after approximately 100000 batches.

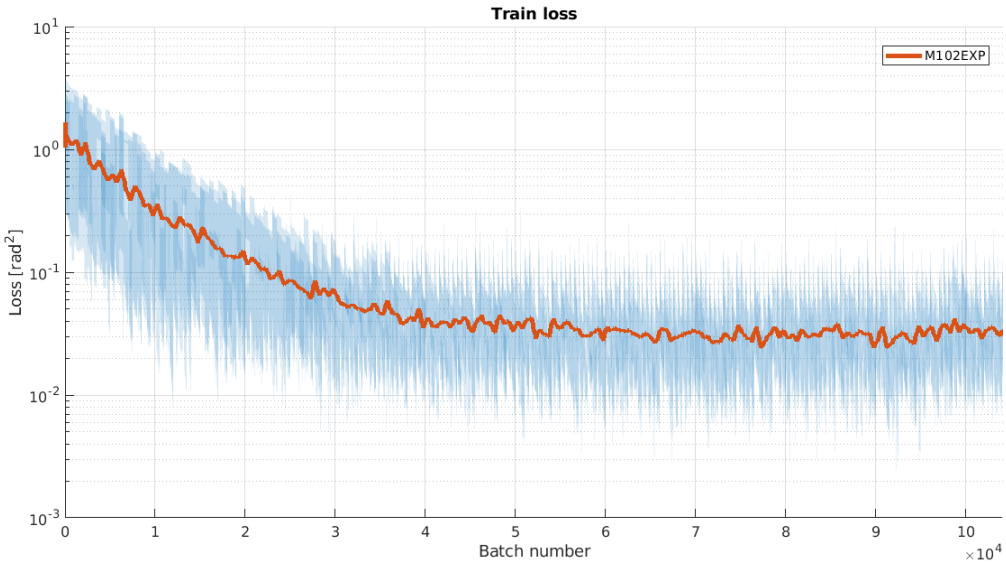

Figure 5-8: Train loss on the experimental data with convergence to a loss of $0.061 rad^2$.

The reference Zernike modes to which the CNN was trained are retrieved from the SH WFS phase screens by a gradient descent routine that minimizes the residual phase error to a 4 decimal accuracy on the Zernike mode coefficients.

The first 20 modes were retrieved on a test dataset(not seen by the CNN before) of 5000 frames with an accuracy as provided in Figure 5-9.

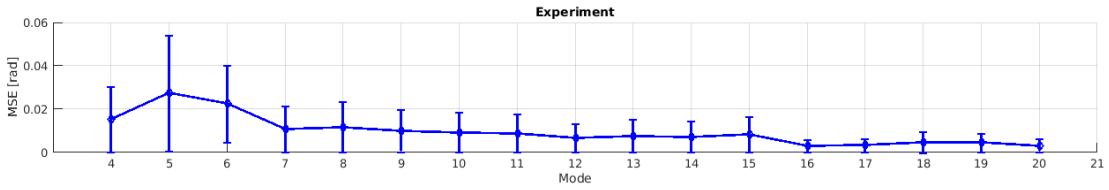

Figure 5-9: Test loss per mode on the experimental dataset

After reconstruction of the wavefront phase based on $\hat{y}$ and the use of Eq. (4-3) to (4-5) the mean squared phase error becomes $72nm$. The used SH WFS introduces an average wavefront error between $\lambda/50$ to $\lambda/25$[38], which for the laser used equals to $12.7nm$ to $25.4nm$. The results on the Kolmogorov phase for comparable turbulence strength resulted in a wavefront error of $21nm$ as shown in table 4-3. This error is purely due to the CNN mapping $\mathcal{P}$, therefor the total error introduced by the experimental setup is estimated to be $\sim 51nm$. Taking this into account it is shown that the CNN is able to derive a direct mapping on a real dataset, which answers the primary research question. Namely, that a CNN can be used for phase retrieval from a single out of focus focal plane image and provides competitive accuracy with respect to SH WFS.

Chapter 6

Discussion & Conclusions

As the relationship between focal plane intensity measurements and wavefront phase is non-linear but of value for usage in free space optical communications, this research was conducted to investigate whether a neural network is able to find a direct mapping from the one onto the other. More specifically, whether a deep convolutional neural network is able to find a fast and accurate mapping from out of focus intensity measurements onto the Zernike mode coefficients of the presented phase.

A model approach is used to generate perfect zero noise Zernike representations of the wavefront phase using the residual mean squared phase error per Zernike mode as provided by Noll[1] on which random Gaussian representations of the turbulence phase were generated(Eq. (2-5) to (2-8)). The phase screens are propagated with a constant amplitude to the out of focus plane where the electromagnetic field is transformed to an intensity measurement by a CCD camera. Representative system parameters were used for the optical model based on the TNO Ofelia breadboard in which for realism only detector noise was added with a SNR of $22dB$.

Using the model 13 CNN architectures were designed and evaluated. For turbulence strengths of $D/r_0 = 2, 8, 14$ and 20 the model named M001 proved best convergence to a mean squared phase error as defined in 4-3 of respectively $15nm, 226nm, 896nm$ and $2238nm$ on a wavelength of $\lambda = 1550nm$ and disregarding tip and tilt. The architecture consisted of six sequential convolutional layers using a stride of 2 to reduce the input dimensions, followed by 3 dense layers with an output layer of size 32 representing the Zernike modes 4 to 35. A Rectified Linear Unit (ReLu) activation function was used.

From the evaluated architectures it is found that using stride outperforms pooling layers, also increasing the amount of kernels benefits the accuracy. A learning rate of $lr < 0.0001$ for the ADAM optimizer proves to be sufficiently small and the batchsize has little influence on the learning process. A kernal size of $3x3$ performs best, especially when regarding the high computational load concerned with larger kernels.

Subsequently the model is evaluated for the phase retrieval of a Kolmogorov phase with a doubling of the kernels in the first three layers so that 128, 256, 512, 1024, 1024 and 1024 channels for respectively layer 1 to 6 are used. The model is renamed to M102 and the computational load is increased to $30.4MFLOPS$ for an input image size of $70x70$. Trained on a random uniformly distributed range of turbulence strengths between $1 \leq D/r_0 \leq 6$, $7 \leq D/r_0 \leq 12$ and $13 \leq D/r_0 \leq 18$ the model converged to a mean squared residual phase error of respectively $5nm, 21nm$ and $74nm$ for the retrieval of the first 35 Zernike modes when disregarding tip and tilt. Training on a range of turbulence strengths greatly improves the training process. Although difficult to prove this is likely due to the spread in the input data that causes the gradients that update the weight matrices to be more agile, and with that it becomes less likely for the weights to converge to local minima. Moreover, the fact that a Zernike mode coefficient representation of the phase is not necessarily unique to the wavefront could actually aid the training process as it functions as dropout. This might also support the fact that only one intensity measurement suffices for retrieving the phase.

To validate the modeled results an experimental setup was build to generate real data that served as in- and output to the CNN. A dataset of 80000 frames for training and 5000 frames for evaluation with a turbulence strength varying roughly between $5 \leq D/r_0 \leq 15$ was generated. With the SH WFS accuracy between $12.7nm - 25.4nm$ serving as reference error tolerance, the phase was retrieved for the first 20 Zernike modes with a mean squared residual phase error of $72nm$. The final error is larger then the modeled implementation as the PSF is larger and more noise is present. Since the CNN implementation, used hardware and hardware configuration have not yet been fully optimized, enough potential for further testing is available when regarding aforementioned benefits of the concept.

The CNN architecture as implemented on the experimental setup has a computational load of $154MFLOPS$ for one forward pass which is largely due to the big input image size of 170x170. On a contemporary mid-end GPU like the NVIDEA GTX1060(4.0TFLOPS on 32bit floating point numbers) this results in a theoretical operating frequency of 25.9kHz, which is a five-fold of the Ofelia breadboard requirement. Although this number is merely a rough estimate as factors such as memory bandwidth, memory latency and level of parallel processing will limit the performance, it is realistic to believe that a trained CNN for FP phase retrieval can be run sufficiently fast for demanding high turbulent environments.

Overall it can be concluded that focal plane phase retrieval by the use of a CNN is feasible for a real setup. The dimensions of the CNN make realtime execution possible, especially when regarding recent CNN implementations on FPGA's[39][40] and the suitability for parallel processing. Placing the CCD closer to the focal point can also cause a significant speed up as the the dimensions of the input image can then be reduced. How this translates to CNN performance should be investigated. This research can provide new ideas to wavefront retrieval in FSOC systems as it can simplify hardware requirements and induce cost reductions. There is no need for wavefront detection and the

corresponding reconstruction calculations that often need iterative optimization in the realtime system. Although the experiment did not reach competing performance to SH WFS and no fluctuation in received intensity was incorporated, the method shows good potential when further optimization of the CNN architecture and hardware configuration is conducted. Considering the CNN the most important conclusions are summed below.

6-1 Conclusions

- Deep Convolutional Neural Networks are able to retrieve the wavefront phase Zernike modes from a single out of focus intensity measurement and could be implemented in a real system.
- The convergence of the CNN and with that the residual phase error is improved when using stride instead of pooling layers for size reduction. Increasing the amount of kernals(and thus featuremaps) is beneficial for the convergence too.
- The used dataset for training has great influence on model convergence. For training and evaluation a diverse dataset of varying turbulence strengths should be used.
- The non-uniqueness of the Zernike mode coefficients with respect to the intensity image is beneficial for the training process as it tends to functions as drop out, providing agility to the weight matrices.

6-2 Recommendations

The performed research has been conducted using a fixed amplitude but this would not be the case in a real implementation. Therefore follow on research on how this impacts the performance of the CNN should be prioritized.

The CCD detector has been placed 20% out of focus to capture the spatial pattern though this position was only based on an educated estimatation. The ideal position in the focal plane proved not feasible but a near focal placement should be investigated.

High accuracy tip and tilt retrieval by a CNN in the focal plane has not been investigated but from the found results and practical perspective/benefit this should be investigated.

Appendix A

Python and driver setup

In order to make use of fast computation on a NVIDIA GPU the following steps need to be undertaken for creating a suited python environment.

First the right driver and back-end needs to be installed in order to run Tensorflow. Therefore do the following on a Linux machine:

- Open new terminal and install 'sudo apt install nvidia-cuda-toolkit'
- install nvidia driver: 'sudo apt install nvidia-384 nvidia-384-dev'
- install other import packages: 'sudo apt-get install g++ freeglut3-dev build-essential libx11-dev libxmu-dev libxi-dev libglu1-mesa libglu1-mesa-dev'
- CUDA 9 requires gcc 6: 'sudo apt install gcc-6' and 'sudo apt install g++-6'
- Download the Cuda toolkit version 9.0:
  'wget https://developer.nvidia.com/compute/cuda/9.0/Prod/local_installers/cuda_9.0.176_384.81_linux-run'
- 'chmod +x cuda_9.0.176_384.81_linux-run'
- 'sudo ./cuda_9.0.176_384.81_linux-run –override'

Next a Python environment needs to be created which allows the written code for this project to be run.

- Install anaconda from https://www.anaconda.com/download/linux and choose the newest version

- Restart machine
- Open terminal
- Create a new python 3.5 environment: 'conda create –name 'python35Tensorflow' python=3.5'
- Activate the environment: 'source activate python35Tensorflow'
- Install the packages below by:
- 'pip install tensorflow-gpu==1.8' OR 'pip install tensorflow' for cpu version
- 'pip install keras'
- 'pip install pandas'
- 'pip install matplotlib'
- 'pip install matlab'
- 'pip install tkinter'
- 'pip install xlrd'

The packages here above require other dependencies, Conda will intall these necessary packages automatically.

In order to make python communicate with Matlab and vice versa the python engine needs to be installed by Matlab. To do so undertake the following steps:

- Make sure you are using Matlab 2014b or higher to have support of the Matlab/Python API.
- Open Matlab and insert 'matlabroot' in the command window. Append this directory with 'extern/engines/python'
- In a terminal: cd "matlabroot/extern/engines/python"
- Install the engine: python setup.py install
- It can be that the engine is not installed for the right python environment. If so go to the site-packages folder of the default python installation and copy the folder 'matlab' to the location where the required python version is installed.

Appendix B

Training GUI and Parameter sheet

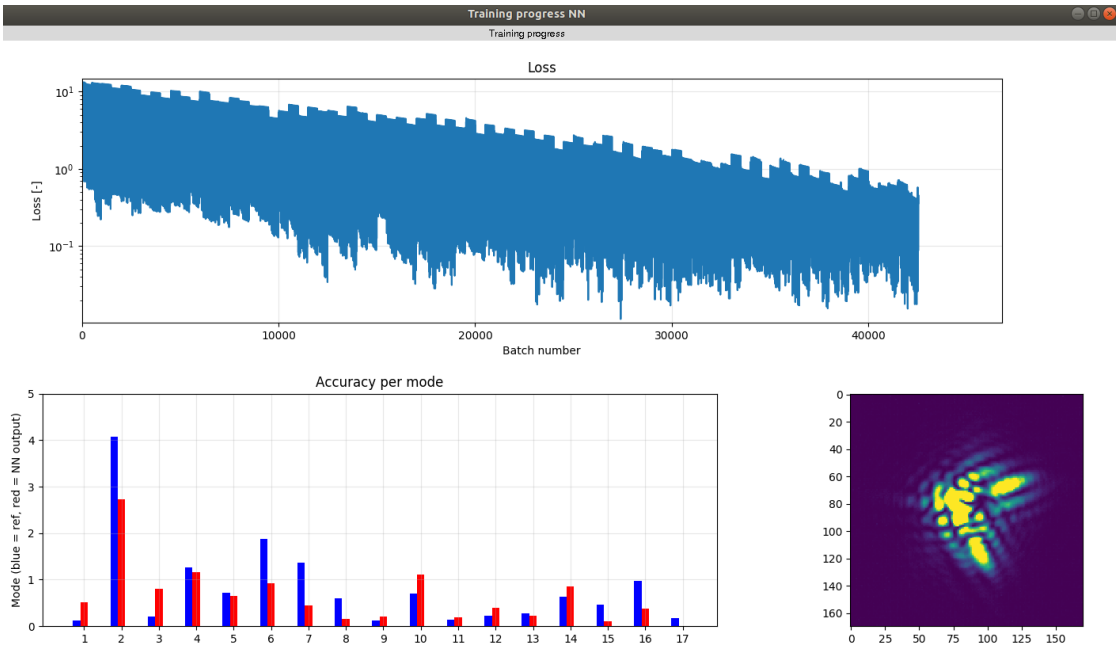

Figure B-1: GUI to follow the training progress. The blue bars indicate the reference phase and the red bars the CNN output.

ITEM	SELECTION	OPTIONS
Model	fromPython	fromPython, loadExisting
Mode	ExpData	TrainingWithData, Realtime, ReversTraining, NollFromMatlab, VonKarmenPhase, VonKarmenPhase_bounded, ExpData
ModelLocation		Provide path of where model is stored(use only when a previously stored model is used)
Modelname	M102EXP	Any model from the file 'NNmodels.py'(use only when a fresh model needs to be loaded)
Tensorboard	on	on or off for monitoring
Optimizer	ADAM	Choose the tensorflow optimizer to use
Monitor	on	store intermediate results for plotting later
InputX	/media/gebruiker/D/ExpData_all	folder location if applicable, no location is none
Outputy	/media/gebruiker/D/ExpData_all	folder location if applicable, no location is none
Ratio	0,8	ratio between train and test data
Plotting	on	plot intermediate results in figure window
PlottingFreq	10	update plot every n frames
InputImageSize	170	
Padding	1500	
Cutoff		
n_modes	1	
Tiptilt	20	
TurbBoundLow	0	0 = present in phase screens but not trained for
TurbBoundHigh	13	minimum D/r0 for VonKarmenPhase_bounded
	18	maximum D/r0 for VonKarmenPhase_bounded

SETUPS TO RUN(add columns for multiple runs)					
SCENARIOS	<- KEEP cell!				
LearningRate	2E-05	4E-05	2E-05	2E-05	
BatchSize	8	8	8	8	
Epochs	2500	3000	3000	80	
EpochSize	100	100	100	3000	
RefreshEpoch	10	10	10	10	
D	0,1	0,1	0,1	0,1	
r0	0,01677	0,01667	0,025	0,0125	

Figure B-2: Configuration Excel file that holds the settings and hyper parameters needed for training the CNN.

Bibliography

[1] R. J. Noll, "Zernike polynomials and atmospheric turbulence," *Journal of the Optical Society of America*, vol. 66, p. 207, mar 1976.

[2] H. Kaushal and G. Kaddoum, "Free space optical communication: Challenges and mitigation techniques,"

[3] J. H. Reed, J. T. Bernhard, and J. Park, "Spectrum access technologies: The past, the present, and the future," *Proceedings of the IEEE*, vol. 100, pp. 1676–1684, may 2012.

[4] D. Cabric, S. Mishra, and R. Brodersen, "Implementation issues in spectrum sensing for cognitive radios," in *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, IEEE.

[5] J. W. Hardy, *Adaptive Optics for Astronomical Telescopes*. OXFORD UNIV PR, 1998.

[6] H. W. Babcock, "The possibility of compensating astronomical seeing," *Publications of the Astronomical Society of the Pacific*, vol. 65, p. 229, oct 1953.

[7] S. R. Chamot, C. Dainty, and S. Esposito, "Adaptive optics for ophthalmic applications using a pyramid wavefront sensor," *Optics Express*, vol. 14, no. 2, p. 518, 2006.

[8] S. Velghe, J. Primot, N. Guérineau, M. Cohen, and B. Wattellier, "Wave-front reconstruction from multidirectional phase derivatives generated by multilateral shearing interferometers," *Optics Letters*, vol. 30, p. 245, feb 2005.

[9] V. Korkiakoski, C. U. Keller, N. Doelman, M. Kenworthy, G. Otten, and M. Verhaegen, "Fast & furious focal-plane wavefront sensing," *Applied Optics*, vol. 53, p. 4565, jul 2014.

[10] C. E. Carrizo, R. M. Calvo, and A. Belmonte, "Intensity-based adaptive optics with sequential optimization for laser communications," *Optics Express*, vol. 26, p. 16044, jun 2018.

[11] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, pp. 255–260, jul 2015.

[12] A. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, vol. 29, pp. 31–44, mar 1996.

[13] R. Saathof, R. den Breeje, W. Klop, S. Kuiper, N. Doelman, F. Pettazzi, A. Vosteen, N. Truyens, W. Crowcombe, J. Human, I. Ferrario, R. M. Calvo, J. Poliak, R. Barrios, D. Giggenbach, C. Fuchs, and S. Scalise, "Optical technologies for terabit/s-throughput feeder link," in *2017 IEEE International Conference on Space Optical Systems and Applications (ICSOS)*, IEEE, nov 2017.

[14] A. N. Kolmogorov, "A refinement of previous hypotheses concerning the local structure of turbulence in a viscous incompressible fluid at high reynolds number," *Journal of Fluid Mechanics*, vol. 13, p. 82, may 1962.

[15] J. P. A. Gibson, *Deep learning, a practitioners approach*. O'Reilly, 2017.

[16] F. Zernike, "The concept of degree of coherence and its application to optical problems," *Physica*, vol. 5, pp. 785–795, aug 1938.

[17] J. W. Goodman, *Introduction to Fourier Optics*. W.H.Freeman & Co Ltd, 2005.

[18] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev, "Phase retrieval with application to optical imaging: A contemporary overview," *IEEE Signal Processing Magazine*, vol. 32, pp. 87–109, may 2015.

[19] Xenics, "Xenics cheetah-640-cl data sheet, http://www.xenics.com/sites/default/files/documents/cheetah-640-cl.zip."

[20] R. S.-W. Gerchberg, "A practical algorithm for the determination of phase from image and diffraction plane pictures," *Optik 35*, 1972.

[21] T. R. Ellis and J. D. Schmidt, "Wavefront sensor performance in strong turbulence with an extended beacon," in *2010 IEEE Aerospace Conference*, IEEE, mar 2010.

[22] R. Rojas, *Neural networks: a systematic introduction*. Springer, 1996.

[23] R. HECHT-NIELSEN, "Theory of the backpropagation neural network based on "nonindent" by robert hecht-nielsen, which appeared in proceedings of the international joint conference on neural networks 1, 593–611, june 1989. © 1989 IEEE.," in *Neural Networks for Perception*, pp. 65–93, Elsevier, 1992.

[24] A. G. J. Patterson, *Deep learning, a practitioners's approach*. O'Reilly, 2017.

[25] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, pp. 141–142, nov 2012.

[26] N. Y. W. T. C. et al., "Reading digits in natural images with unsupervised feature learning," *In NIPS workshop on deep learning and unsupervised feature learning, pp. 4,*, 2011.

[27] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Master's thesis, Department of Computer Science, University of Toronto, 2009.

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2009.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, may 2017.

[30] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun, "Deep image: Scaling up image recognition,"

[31] N. H. F. Beebe, "Accurate hyperbolic tangent computation," tech. rep., University of Utah, Department of Mathematics, 1993.

[32] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net,"

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition,"

[34] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-net: ImageNet classification using binary convolutional neural networks," in *Computer Vision – ECCV 2016*, pp. 525–542, Springer International Publishing, 2016.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization,"

[36] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation*, (Savannah, GA), pp. 265–283, USENIX Association, 2016.

[37] J. Wildschut, "Characterization and control of a deformable mirror for adaptive optics," Master's thesis, Technische Universiteit Eindhoven, 2015.

[38] A. Chernyshov, U. Sterr, F. Riehle, J. Helmcke, and J. Pfund, "Calibration of a shack–hartmann sensor for absolute measurements of wavefronts," *Applied Optics*, vol. 44, p. 6419, oct 2005.

[39] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA - based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ACM Press, 2015.

[40] W. Zhao, H. Fu, W. Luk, T. Yu, S. Wang, B. Feng, Y. Ma, and G. Yang, "F-cnn: An fpga-based framework for training convolutional neural networks," in *2016 27th International Conference on Application-specific Systems Architectures and Processors*, IEEE, jul 2016.

Glossary

List of Acronyms

AD	Analog to Digital
AO	Adaptive Optics
ADAM	A method for stochastic optimization
CCD	Charge Coupled Device
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DM	Deformable Mirror
DAC	Digital Analog Converter
EM	Electro Magnetic
FC	Focus Camera
FFNN	Feed Forward Neural Network
FLOPS	Floating Point Operations
FP	Focal Plane
FSOC	Free Space Optical Communications
GUI	Graphical User Interface
GD	Gradient Descent
GS	Gerchberg-Saxton

ITU	International Telecommunications Union
LR	Learning Rate
ML	Machine Learning
MSE	Mean Squared Error
NN	Neural Network
PAM	Point Ahead Mirror
ReLu	Rectified Linear Unit
RGB	Red Green Blue
RF	Radio Frequency
SH	Shack-Harmann
SGD	Stochastic Gradient Descent
SNR	Signal to Noise Ratio
TNO	De Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek
WFS	Wavefront Sensor

Nomenclature

Table B-1: Nomenclature codes part 1

Symbol	Meaning
a_0	connecting vector from convolutional to dense layer
c_J	Noll residual error coefficient
d	Lens diameter
D	Aperture
D_{dig}	Digital 2D detector output
τ_0	Atmospheric coherence time
A	Amplitude of the electric field
f_s	Sample frequency
f	Focal length
f_w	Full well photon count
r_0	Fried's parameter
ϕ	Wavefront phase
ϕ_d	Wavefront phase after propagation
ϕ_{in}	Wavefront phase of input data
ϕ_{MSE}	Mean squared wavefront error
E	Electrical field
E_{in}	Electrical field of input data
E_{FP}	Electrical field in the focal plane
I_d	Intensity
P_i	Received power
$\mathcal{P}$	Neural Network mapping
n_r	Number of received photons
h	Planck's constant
Q_e	Quantum efficiency
f_f	Pixel fill factor

Table B-2: Nomenclature codes part 2

Symbol	Meaning
r_a	Duty cycle
I_{pixel}	Number of photons received on one detector pixel
f_w	Full well capacity of detector
d_c	Dark current on detector
α	Learning rate
β_1	First moment exponential decay rate
β_2	Second moment exponential decay rate
θ_{min}	Sensitivity
ρ	Radius
Δ_J	Mean squared resisual phase error
J	Polynomial ordering number(mode number)
F	2D mirror surface
S	MSE
λ	Wavelength
L_0	Outer scale of the Kolmogorov spectrum
l_0	Inner scale of the Kolmogorov spectrum
D	Aperture
W_i	Weight matrix of layer i
X	2D intensity pixel array
$\mathbf{X}$	Dataset holding multiple X
X_{norm}	Normalized 2D input array
Y_δ	Matrix output of CNN layer δ
M	Leg size of input array
K	Kernal size
K_δ	3D kernal for CNN layer δ
P	Pooling layer size
C	Number of channels in a NN
C_n^2	Atmospheric turbulence strength profile
Q	Number of matrix multiplications
$FLOPS$	Floating point operations
$FLOPS_{conv}$	Floating point operations for one convolutional layer
a_i	Activation of neuron i
x_i	Neuron input from neuron i in precessing layer
w_i	weight for input i
b_i	bias vector of layer i
B	bias matrix

Table B-3: Nomenclature codes part 3

Symbol	Meaning
y	Reference Zernike modes of the wavefront
$\hat{y}$	Neural Network output Zernike modes
y_J	Zernike mode coefficient J
$y_{J_{min}}$	Lowest Zernike mode coefficient
$y_{J_{max}}$	Highest Zernike mode coefficient
y_{DM}	DM command signal
N	Number of elements in array
R_n^m	Radial polynomial of order m and n
U	CNN minimization objective
V	Number of entries in dataset
Z_n^m	Zernike polynomial of order m and n