# Machine learning for image-based wavefront sensing

**Auteur :** Vanberg, Pierre-Olivier
**Promoteur(s) :** Absil, Olivier; Louppe, Gilles
**Faculté :** Faculté des Sciences appliquées
**Diplôme :** Master en ingénieur civil physicien, à finalité approfondie
**Année académique :** 2018-2019
**URI/URL :** http://hdl.handle.net/2268.2/6800

# Machine learning for image-based wavefront sensing

Master thesis conducted by

**Pierre-Olivier Vanberg**

with the aim of obtaining the degree of Master in Physical Engineering

University of Liège
Faculty of Applied Sciences

Academic year 2018-2019

# Machine learning for image-based wavefront sensing

University of Liège - Faculty of Applied Sciences

**Pierre-olivier Vanberg**

**Abstract:** Astronomical images are often degraded by the disturbance of the Earth's atmosphere. Indeed, light propagating to ground-based telescopes is distorted by the random motions of the turbulent atmosphere. Adaptive Optics (AO) partly solves this problem by flattening in real time the wavefront. The role of AO is first to acquire information about the distortions and then to adjust a deformable mirror to compensate these optical aberrations. Notwithstanding the efficiency of AO systems, some intrinsic limitations remain. In particular, the presence of unseen aberrations called Non-Common Path Aberrations (NCPA) still represent a real challenge for extreme AO observation tasks such as exoplanet direct imaging.

Image-based wavefront sensing methods are specifically designed algorithms dedicated to estimate the wavefront from science camera intensity measurements. This approach is particularly desirable as it requires no additional components and solely depends on the observed images. This thesis thus proposes to improve image-based wavefront sensing techniques using machine learning algorithms. Deep convolutional neural networks (CNN) have been trained to estimate the wavefront using one or multiple intensity measurements. The efficiency of state-of-the art architectures is reviewed and compared on two different data sets: 20 Zernike modes and 100 Zernike modes. In both configuration, the direct pixel-wise estimation of the phase map has been shown to outperform the modal phase reconstruction.

# Acknowledgements

First, I would like to thank Olivier Absil and Gilles Orban De Xivry for the time dedicated to this master thesis and the continuous feedback provided. I would also like to thank them for the provision of a perfectly maintained GPU server (Fenrir).

Thereafter, I would like to express my thanks to Gilles Louppe whose deep learning expertise has been particularly useful for the accomplishment of this work.

Finally, I would like to thank Marc Van Droogenbroeck for agreeing to read the manuscript and to participate in the defense of this thesis.

# Contents

# Abbreviations

| | |
|---|---|
| **AO** | Adaptive optics |
| **WFS** | Wavefront sensor |
| **PSF** | Point spread function |
| **OTF** | Optical transfer function |
| **NCPA** | Non-common path aberrations |
| **DFT** | Discrete Fourier transform |
| **FFT** | Fast Fourier transform |
| **IFFT** | Inverse fast Fourier transform |
| **DNN** | Deep neural network |
| **CNN** | Convolutional neural network |
| **MLP** | Multilayer perceptron |
| **RMSE** | Root mean squared error |

# Chapter 1

# Introduction

Astronomical images are often degraded by the disturbance of the Earth's atmosphere. Since the invention of the first telescopes in the 1600s, it has been known that light propagating to ground-based telescopes is affected by the random motions of the atmospheric turbulence. These disturbances lead to small changes in the refractive index of the air and alter optical waves as they pass through the atmosphere. In 1730, Isaac Newton wrote:

> "If the Theory of making Telescopes could at length be fully brought into Practice, yet there would be certain Bounds beyond which Telescopes could not perform. For the Air through which we look upon the Stars, is in perpetual Tremor; as may be seen by the tremulous Motions of Shadows cast from high Towers, and by the twinkling of the fix'd Stars."
>
> Isaac Newton, *Opticks*, 1730.

By these few lines, Newton predicted that a turbulent atmosphere would ultimately limits the resolution and the contrast of optical systems. The resolution of ground-based telescope is primarily determined by its optical components ($R \sim \lambda/D$): the observing wavelength and the aperture diameter. Increasing the diameter of optical elements thus improve the achievable resolution up to a given threshold whereby the image quality starts to depend on the atmospheric condition. Since the 1950s, such limitations have been reached and the astronomers have investigated two main solutions: avoid the atmosphere by going into space or directly compensate the atmospheric disturbances on the ground. Among the first category, one of the best-known example is the Hubble Space Telescope which provides extremely high-resolution images of unprecedented quality.

The second category focuses on the correction of the distorted wavefront thanks to a deformable mirror. This technique is known as Adaptive Optics (AO) and intends to flatten in real time the wavefront. The role of AO is first to acquire information about the distortions and then adjust a deformable mirror to compensate these optical aberrations. The phase profile of the wave is typically estimated with a wavefront sensor (WFS) based on light intensity measurements. The retrieved information is then passed to a control system which maintains the mirror shape as close as possible to the optimal configuration. Over the years, AO has reinvented ground-based astronomy and improved the imaging quality up to the diffraction limit.

Notwithstanding the efficiency of AO systems, some intrinsic limitations remain. In particular, the presence of unseen aberrations called Non-Common Path Aberrations (NCPA) still represent a real challenge for extreme AO observation tasks such as exoplanet direct imaging. Indeed to achieve the best level of correction, the wavefront sensor must be positioned as close as possible to the science camera in order to avoid optical path differences. Moreover as those aberrations are introduced after the beam splitting between the WFS and the imaging plane, they are not directly sensed and need a proper correction. Among the most efficient tools, image-based wavefront sensing has proven to be one of the most efficient and reliable technique. This method measures the wavefront error using a parameterized physical model and science camera intensity measurements. This approach is particularly desirable as it requires no additional components and solely depends on the observed images.

This thesis thus proposes to improve image-based wavefront sensing techniques using machine learning algorithms. Since 2000s, deep learning has indeed been proven to generalize well on numerous classification and regression tasks. In this perspective, deep convolutional neural networks (CNN) have been trained to estimate the wavefront error using one or multiple intensity measurements.

# Chapter 2

# Background

In this first chapter, the ground-based telescope imaging theory is briefly reviewed. The main results are presented and limiting factors are discussed. A special attention is given to differentiate external phenomena (atmospheric turbulence, NCPA) and fundamental limits imposed by physics (diffraction limit). In the second part of this chapter, deep learning optimization theory and convolution neural networks will be succinctly introduced.

## 2.1 Imaging theory

In the following section, light will be described as an electromagnetic wave governed by the Maxwell equations. The quantum properties of light will be discarded until the noise section. Starting then from the electromagnetism description, light is expressed as a complex wave.

$$U(\boldsymbol{r}, t) = U_0(\boldsymbol{r}) \exp\left(i(\boldsymbol{k} \cdot \boldsymbol{x} - \omega t)\right) \tag{2.1}$$

where $\boldsymbol{r}$ indicates the spatial coordinate, $\boldsymbol{k}$ the wave vector and $\omega$ the frequency.

### 2.1.1 Fresnel-Kirchhoff diffraction

In a wide range of configurations, the propagation of light can be described using the Fresnel-Kirchhoff diffraction theory. The equation follows from the Maxwell equations and the Kirchhoff integral theorem for monochromatic waves.

The Fresnel diffraction integral is expressed as

$$U(x,y) = \int\int A(x',y')U(x',y') \exp\left[ik\left(\frac{x'^2 - 2x'x}{2z} + \frac{y'^2 - 2y'y}{2z}\right)\right]dx'dy' \tag{2.2}$$

$x', y'$ indicate the pupil plane coordinates, $x, y$ the image plane coordinates and $z$ the distance between the two planes. $A(x', y')$ is a one-zero function describing the pupil aperture.

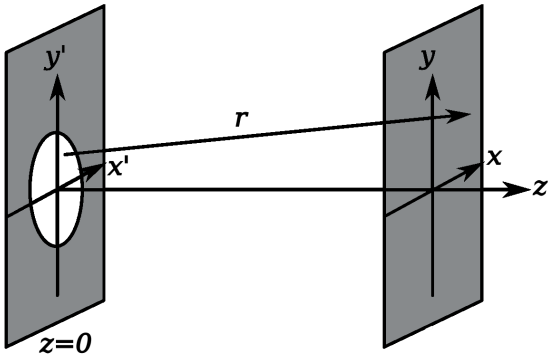



FIGURE 2.1: Geometry of imaging, pupil and image planes.

This expression can then be simplified using the Fraunhofer approximation:

$$z \gg \frac{k(x'^2 + y'^2)_{max}}{2} \qquad \forall x', y' \in \text{entrance aperture} \tag{2.3}$$

and by applying a first order approximation ignoring the quadratic terms.

$$U(x,y) = \int\int A(x',y')U(x',y') \exp\left[-\frac{ik}{z}\left(x'x + y'y\right)\right]dx'dy' \tag{2.4}$$

$$= \mathcal{F}\Big[A(x',y')U(x',y')\Big] \tag{2.5}$$

It is thus recognized that the Fresnel-Kirchhoff diffraction theory results in direct a 2D Fourier transform. The field in the image plane is Fourier transform of the field in the aperture with the corresponding spatial frequencies: $f_x = \frac{kx}{z}$ and $f_y = \frac{ky}{z}$.

### 2.1.2 Point spread function

The point spread function (PSF) is the 2D irradiance profile in the image plane of an astronomical point source. It can be viewed as the impulse response of an optical system

with circular aperture.

$$\mathrm{PSF}(x, y) = |h(x, y)|^2 \tag{2.6}$$

with $h(x, y)$ the Fourier transform of the complex electromagnetic field,

$$h(x, y) = \mathcal{F}\Big[A(x', y') \exp\big(i\theta(x', y')\big)\Big] \tag{2.7}$$

$A(x', y')$ is a zero-one function corresponding to the pupil function while $\theta(x', y')$ represent the wavefront phase over the entrance pupil.

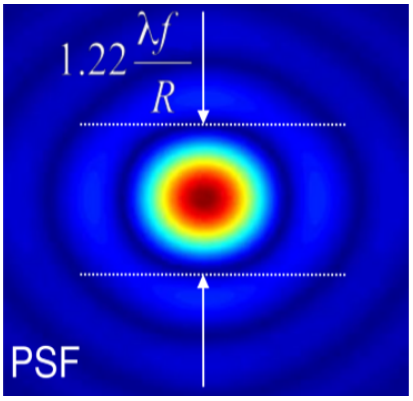



FIGURE 2.2: Airy disk. [1]

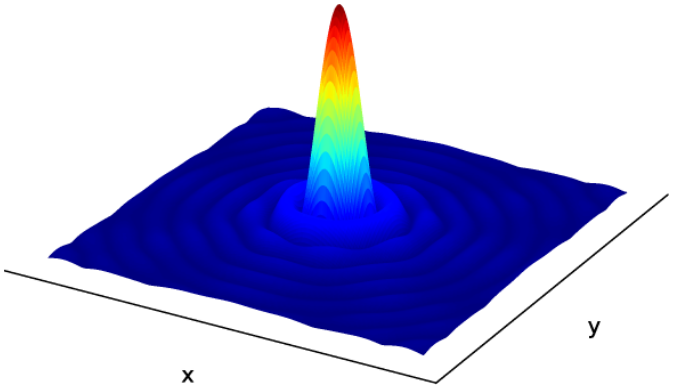



FIGURE 2.3: PSF$(x, y)$, point spread function.

The PSF for a perfect and unaberrated optical system with circular aperture is known as the «Airy pattern» and follow:

$$PSF(u) = \frac{1}{(1-\varepsilon^2)^2}\left[\frac{2J_1(u)}{u} - \varepsilon^2\frac{2J_1(\varepsilon u)}{\varepsilon u}\right] \tag{2.8}$$

where $u = \frac{\pi}{\lambda}D\theta$ is the dimensionless distance from the optical axis, $D$ is the primary aperture diameter, $\theta$ is the angular radius, $J_1$ is the first order Bessel function of the first kind and $\varepsilon$ is the fractional obscured radius (0 in the previous figures). The first minimum of the Airy pattern is commonly referred as the airy disk and contains about 86% of the total light. Its diameter is given by:

$$\text{Airy diameter} = 1.22\frac{\lambda z}{D} \tag{2.9}$$

It establishes a physical limit to the resolution power of optical instrument. The Rayleigh criterion states that two point source objects must be separated by at least half the diameter of the airy disk to be fully distinguishable.

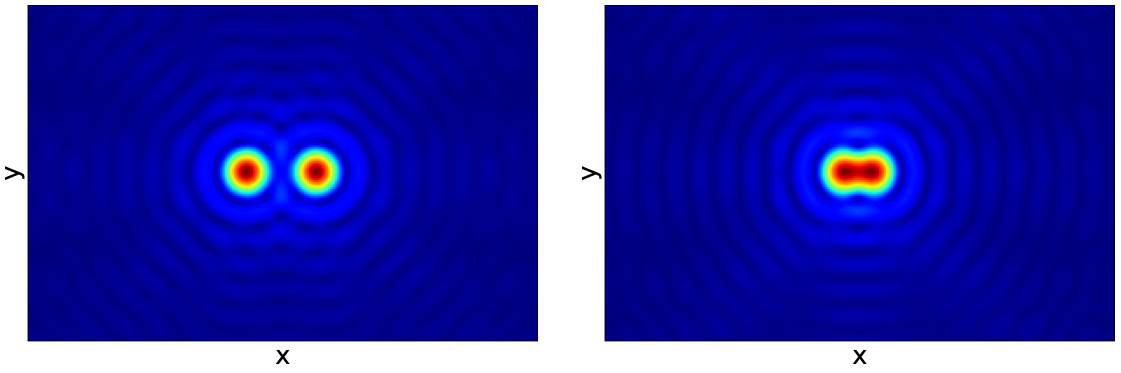



FIGURE 2.4: Illustration of the Rayleigh criterion. **Left:** Fully distinguishable objects **Right:** Objects separated by half their airy diameter. (resolution limit)

Up to now only perfect PSF has been considered. Nevertheless, a phase aberration $\theta(x', y') \neq 0$ tends to flatten the PSF peak and thus reduces even more the resolution power.

### 2.1.3 Zernike polynomials

The phase aberration $\theta(x', y')$ can be described in several manners. One of the most commonly used methods consists in decomposing the wavefront phase onto an orthogonal basis called the Zernike basis.

$$\theta(x', y') = \sum_{i=0}^{\infty} c_i Z_i(x', y') \tag{2.10}$$

where $c_i$ are the Zernike expansion coefficients. The Zernike polynomials $Z_i$ are defined on the unit circle and are particularly well suited to characterize optical aberrations: tip, tilt, coma... Many normalizations and labeling currently exist for defining the Zernike polynomials, along this thesis the Noll convention has been chosen. [2] For convenience, they are expressed in polar coordinates as the product of angular functions and radial polynomials.

$$\begin{aligned} Z_i(r, \varphi) &= \sqrt{n+1} R_0^n(r) && m = 0 \\ Z_i(r, \varphi) &= \sqrt{n+1} R_m^n(r) \sqrt{2} \cos(m\varphi) && m \neq 0, i \text{ even} \\ Z_i(r, \varphi) &= \sqrt{n+1} R_m^n(r) \sqrt{2} \sin(m\varphi) && m \neq 0, i \text{ odd} \end{aligned} \tag{2.11}$$

where

$$R_n^m(r) = \sum_{k=0}^{\frac{n-m}{2}} \frac{(-1)^k\,(n-k)!}{k!\left(\frac{n+m}{2}-k\right)!\left(\frac{n-m}{2}-k\right)!}\, r^{n-2\,k} \tag{2.12}$$

$n$ corresponds to the radial order and $m$ to the azimuthal order. $m, n$ are positive integer satisfying $m \leq n$ and $n - |m| =$ even. The index $i$ is a mode ordering number depending on $m$ and $n$. One of the main characteristic of the Zernike polynomials is their orthogonality property. It allows to express any continuous function by a unique set of coefficients as in (3.11).

$$\frac{1}{\pi}\int_0^1 \int_0^{2\pi} Z_i(r,\varphi) Z_j(r,\varphi) dr d\varphi = \delta_{i,j} \tag{2.13}$$

### 2.1.4 Strehl ratio

The Strehl ratio is a measure of the performances of an optical system. The Strehl ratio ($S$) is defined as the ratio of the peak intensity of a measured PSF to the peak intensity of a perfect, diffraction-limited PSF. [3]

$$S = \frac{I(x,y)}{P(x_0,y_0)} \tag{2.14}$$

where $x_0, y_0$ is the position of the maximum of the diffraction limited PSF and $x, y$ is the position of the maximum of the measured PSF. The Strehl ratio quantifies the amount of light contained within the Airy disk as a ratio of the theoretical maximum that would have been contained in a diffraction limited Airy disk. The main advantage of the Strehl ratio is the fact that it asses the optical quality in terms of the theoretical performances and within a single number. Furthermore, the Strehl ratio can be related to the wavefront errors via the Maréchal approximation. [4]

$$S \approx |< \exp\left[i(\theta - < \theta >)\right] >|^2 \tag{2.15}$$

where $\theta$ is the wavefront phase over the pupil aperture.

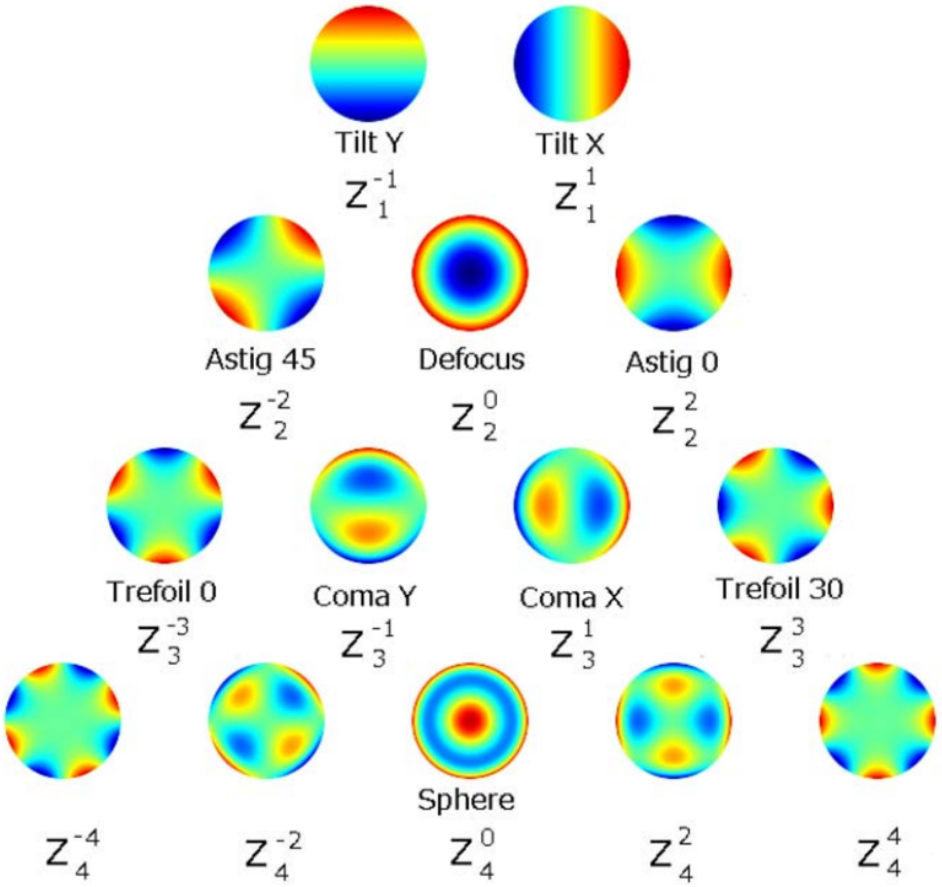



FIGURE 2.5: Color map diagram of the Zernike polynomials up to the 4th order. [5]

| j | n | m | $Z_j$ | Name |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | Piston |
| 2 | 1 | -1 | $2r\cos(\varphi)$ | Tilt |
| 3 | 1 | 1 | $2r\sin(\varphi)$ | Tilt |
| 4 | 2 | 0 | $\sqrt{3}(2r^2-1)$ | Defocus |
| 5 | 2 | 2 | $\sqrt{6}r^2\sin(2\varphi)$ | Astigmatism |
| 6 | 2 | -2 | $\sqrt{6}r^2\cos(2\varphi)$ | Astigmatism |
| 7 | 3 | -1 | $\sqrt{8}(3r^3-2r)\sin(\varphi)$ | Coma |
| 8 | 3 | 1 | $\sqrt{8}(3r^3-2r)\cos(\varphi)$ | Coma |
| 9 | 3 | 3 | $\sqrt{8}r^3\sin(3\varphi)$Ttrefoil | |
| 10 | 3 | -3 | $\sqrt{8}r^3\cos(3\varphi)$ | Trefoil |

TABLE 2.1: Zernike polynomials up to third radial order, Noll (1976).

## 2.2 Phase retrieval

Nowadays, most of the light detectors consist of charge-coupled devices (CCD) or closely related equipment. Such devices convert the light intensity into electrical charges which can then be easily manipulated. Nonetheless, light has not only an amplitude but also phase, see equation (2.1). All the information about the phase is therefore systematically lost in the measurement. The way in which light interacts with matter therefore introduces a fundamental limit.

### 2.2.1 Mathematical description

Wavefront sensing by phase retrieval is the process of algorithmically retrieving the wavefront, i.e. the phase $\theta(x', y')$ from one or multiple intensity measurements. For instance along this thesis, phase retrieval consists in extracting the entrance pupil phase distortion $\theta(x', y')$ from $PSF(x, y)$ images.

$$\mathrm{PSF}(x, y) = |h(x, y)|^2 \tag{2.16}$$

The presence of the modulus indicates that all phase information related to $h(x, y)$ have been lost. The inverse Fourier transform is thus not sufficient to uniquely reconstruct the wavefront.

$$H(x', y') = \mathcal{F}^{-1}\Big[h(x, y)\Big] \tag{2.17}$$

$$= A(x', y') \exp\big(i\theta(x', y')\big) \tag{2.18}$$

$H(x', y')$ represents here the coherent transfer function of the system and must not be confused with the optical transfer function (OTF) which is the direct Fourier transform of the point spread function. As previously mentioned, $A(x', y')$ is often known (e.g. entrance aperture), the problem therefore becomes equivalent to finding $\theta(x', y')$ or equivalently $H(x', y')$ from

$$|h(x, y)| = \sqrt{\mathrm{PSF}(x, y)} \tag{2.19}$$

and

$$|H(x', y')| = A(x', y') \tag{2.20}$$

Overall, the phase retrieval problem consists in recovering the phase of a complex function from its modulus in two Fourier conjugated planes.

Several extensions may also be considered. For instance the problem can easily be prolonged to noisy measurement $n(x, y)$ or extended sources $o(x, y)$.

$$\mathrm{p}(x, y) = (PSF * o)(x, y) + n(x, y) \tag{2.21}$$

where $*$ denotes the standard convolution product.

### 2.2.2 Algorithms

In 1971, R. W. Gerchberg and W. O. Saxton [6] have proposed the first algorithm for solving the phase problem using a space-frequency approach. Subsequently, many improvements were gradually proposed: Gonsalves [7], Devaney [8], Fienup [9]. In this section, the main algorithms are reviewed and their limitations are discussed.

First it is important to mention that in practice, one deals with discrete data. Typically, the point spread functions are sampled using a regular grid, $x_i = 0, 1, ..., N-1$ and $y_j = 0, 1, ..., N-1$. The sampling may arise from the light capturing devices or simply for numerical considerations. Naturally, one also uses the discrete Fourier transform (DFT).

$$H(x'_i, y'_j) = \mathcal{F}\Big[h(x_n, y_m)\Big] \tag{2.22}$$

$$= \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} h(x_n, y_m) \exp\left[ -i\frac{2\pi}{N}\left(x_n x'_i + y_m y'_j\right)\right] \tag{2.23}$$

and its inverse

$$h(x_i, y_j) = \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} H(x'_n, y'_m) \exp\left[ -i\frac{2\pi}{N}\left(x'_n x_i + y'_m y_j\right)\right] \tag{2.24}$$

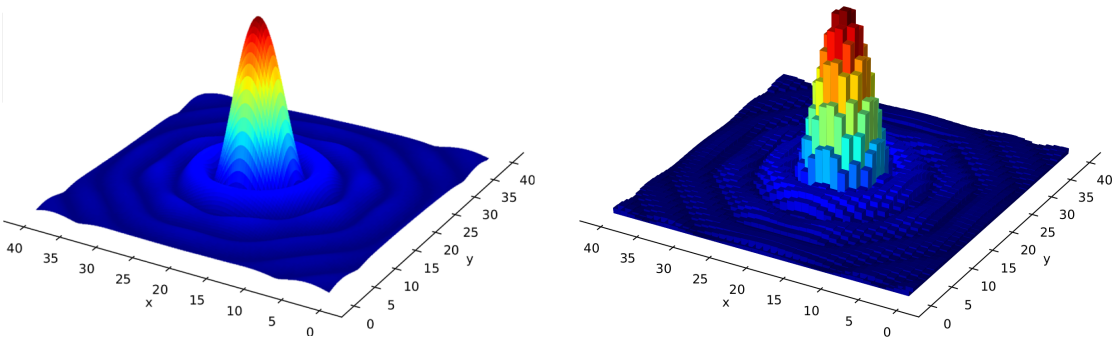



FIGURE 2.6: **Left:** Initial point spread function. **Right:** Sampled point spread function over $40 \times 40$ pixels.

#### 2.2.2.1 Gerchberg-Saxton

The Gerchberg-Saxton algorithm [6] [9] is an iterative algorithm designed to reconstruct the phase $\theta(\boldsymbol{u})$ from conjugated intensity measurements: $|h(\boldsymbol{x})|$ and $|H(\boldsymbol{x})|$. After the $k-th$ iteration, $h_k(\boldsymbol{x})$, $H_k(\boldsymbol{u})$ and $\theta_k(\boldsymbol{u})$ are respectively the estimate of $h(\boldsymbol{x})$, $H(\boldsymbol{u})$ and $\theta(\boldsymbol{u})$. The GS algorithm consists of 4 main steps:

1. Fourier transform the object estimation

$$H_k(\boldsymbol{u}) = |G_k(\boldsymbol{u})|e^{i\theta_k(\boldsymbol{u})} = \mathcal{F}[h_k(\boldsymbol{x})] \tag{2.25}$$

2. Replace the modulus of the estimation by the measured one

$$H'_k(\boldsymbol{u}) = |H(\boldsymbol{u})|e^{i\theta_k(\boldsymbol{u})} \tag{2.26}$$

   The prime notation indicates a temporary variable.

3. Inverse Fourier transform of the constrained estimation

$$h'_k(\boldsymbol{x}) = |h'_k(\boldsymbol{x})|e^{i\phi_k(\boldsymbol{x})} = \mathcal{F}^{-1}[H'_k(\boldsymbol{u})] \tag{2.27}$$

4. Replace the modulus of the estimation by the measured one

$$h_{k+1}(\boldsymbol{x}) = |h(\boldsymbol{x})|e^{i\phi_k(\boldsymbol{x})} \tag{2.28}$$

The initial choise of $\theta_k(\boldsymbol{u})$ is problem dependant. A good practice consists in starting from randomly distributed values. By iterating back and forth between the Fourier spaces, on can then demonstrate that the estimated object will in fine satisfy the object domain constraint. [9]

$$E_k^2 = \frac{1}{N^2}\sum_{\boldsymbol{u}} |h_k(\boldsymbol{u}) - h'_k(\boldsymbol{u})|^2 \tag{2.29}$$

or equivalently the Fourier domain constraint

$$E_k^2 = \frac{1}{N^2}\sum_{\boldsymbol{u}} |H_k(\boldsymbol{u}) - H'_k(\boldsymbol{u})|^2 \tag{2.30}$$

The algorithm convergence may also be demonstrated

$$E_{k+1}^2 \leq E_k^2 \tag{2.31}$$

#### 2.2.2.2 Error-reduction

The error-reduction algorithm proposed by J. R. Fienup in 1982 [9] is the direct generalization of the Gerchberg-Saxton. The three first steps are identical while the fourth step becomes:

$$h_{k+1}(\boldsymbol{x}) = \begin{cases} h'_k(\boldsymbol{x}) & \boldsymbol{x} \notin \Omega \\ 0 & \boldsymbol{x} \in \Omega \end{cases} \tag{2.32}$$

where $\Omega$ is the space region in which $h'_k(\boldsymbol{x})$ violates the object domain constraint. For example, one can enforce a compact support or a non-negativity requirement. It is the direct generalization of the GS algorithm to single intensity measurement.

#### 2.2.2.3 Hybrid Input-Output

The error-reduction algorithm however exhibits slow convergence in multiple configuration. [9][10] One improvement therefore consists in introducing feedback information from the previous iteration.

$$h_{k+1}(\boldsymbol{x}) = \begin{cases} h'_k(\boldsymbol{x}) & \boldsymbol{x} \notin \Omega \\ h_k(\boldsymbol{x}) - \beta h'_k(\boldsymbol{x}) & \boldsymbol{x} \in \Omega \end{cases} \tag{2.33}$$

$\beta$ is the feedback factor $\in [0, 1]$. This formulation also reduces the probability of stagnation.

#### 2.2.2.4 Limitations

The previously presented algorithms have been demonstrated to be globally efficient methods for the phase reconstruction. Nevertheless, these implementations may stagnate under some circumstances.

- First, the algorithm may stagnates due to the twin-image problem. It results from the fact that an object $h(\boldsymbol{x})$ and its complex conjugated counterpart $h(-\boldsymbol{x})$ have the same Fourier modulus. Starting from random initial guess, the iterative algorithm has equal probability of reconstructing one of these two objects. However, in some cases, the features of both images may become equally important and the algorithm may stagnate trying to reconstruct simultaneously both images. Experimentally, this mode of stagnation has been significantly encountered when the object support is centrosymmetric.

- The second possible mode of stagnation is characterized by the appearance of a superimposed pattern of stripes. This pattern follows a sinusoidal perturbation in a given direction while remaining constant in the orthogonal direction. It can be interpreted as being stuck in a strong local minimum of the error metric.
- A third possible stagnation problem may arises from inconsistent support constraint. In particular, if the reconstructed object is translated, the algorithm will partially cut the image. Subsequently, the algorithm has a tendency to stagnate.
- Finally, the algorithm may stagnates in local minimum due to inadequate initialization. This problem is know as the «capture range problem». If the starting guess is too far apart from the true solution, the likelihood of stagnation rapidly becomes significant. One can demonstrate that: «the probability of stagnation is related to the ratio of the surface area of the hypersphere cap for which all points are within the capture range to the total surface area of the hypersphere. This probability decreases as approximately $1/R^{n+1}$ , where n is the dimensionality of our hypersphere and $R$ is its radius». [11] The radius $R$ is proportional to the RMS wavefront error while $n$ can be related to the number of Zernike modes. When dealing with a large number of Zernike polynomials (for example 100), the probability of finding a random guess in the capture range is thus particularly low. The same problem occurs with high wavefront aberrations (RMS WFE $\approx$ 1 rad or greater) leading to large value of $R$ and low converging probability.

### 2.2.3 Phase diversity

As previously mentioned, the wavefront reconstruction from a single focused image does not ensure the uniqueness of the solution. Phase diversity [7][12] thus intends to remove this indetermination by adding additional information. The idea is to collect two or more images perturbed by an additional known phase. For instance, a simple translation of the detector along the optical axis can be used to introduce a known amount of defocus $\varphi(x', y')$. In this case, it is assumed that nor the aberrations nor the object evolve during the translation. For this reason, phase diversity is often used to characterize static aberrations such as NCPA.

$$h_1(x, y) = \mathcal{F}^{-1}\Big[A(x', y') \exp\big(i\theta(x', y')\big)\Big] \tag{2.34}$$

$$h_2(x, y) = \mathcal{F}^{-1}\Big[A(x', y') \exp\big(i\theta(x', y') + i\varphi(x', y')\big)\Big] \tag{2.35}$$

Alternatively phase diversity can be introduced using a beam splitter and two detectors.

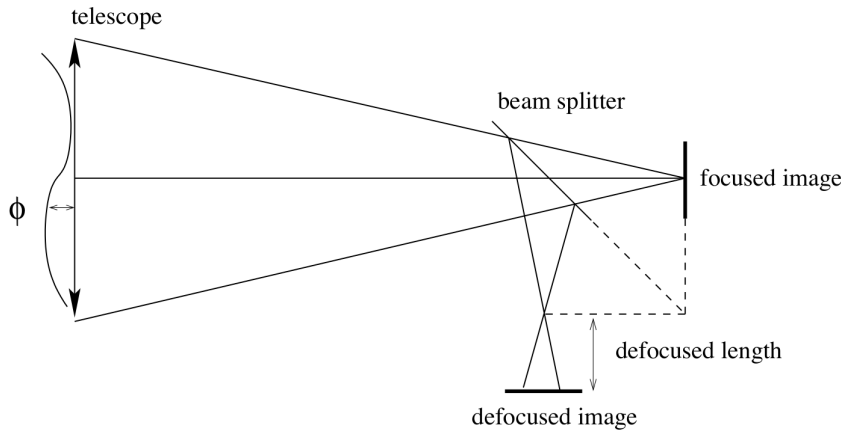



FIGURE 2.7: Phase diversity principle. [13]

The goal of phase diversity algorithms is then to identify a phase aberration combination $\theta(x', y')$ that is consistent with all the intensity measurement $|h_1(x, y)|$ and $|h_2(x, y)|$. In particular, for point source objects.

$$PSF_1(x, y) = |h_1(x, y)|^2 \tag{2.36}$$

$$PSF_2(x, y) = |h_2(x, y)|^2 \tag{2.37}$$

One of the classical optimization strategy consists in using a Joint Maximum A Posteriori (JMAP) approach. [13]

$$\hat{\theta}_{\text{JMAP}} = \arg\max_{\theta} \quad p\big(|h_1|, |h_2|, \theta\big) \tag{2.38}$$

where $p\big(|h_1|, |h_2|, \theta\big)$ is the joint probability density function. Moreover, to reduce the dimensionality of optimization space, the phase function is expanded onto the $k$ first Zernike polynomials.

$$\theta(x', y') = \sum_{i=0}^{k} c_i Z_i(x', y') \tag{2.39}$$

The JMAP can then be expressed as

$$\hat{\boldsymbol{c}}_{\text{JMAP}} = \arg\max_{\boldsymbol{c}} \quad p\big(|h_1|, |h_2|, \boldsymbol{c}\big) \tag{2.40}$$

One can finally demonstrate that the maximization a posteriori is equivalent to the minimization of the negative log-likelihood. [12] Under its minimization form, the problem can then be iteratively solved using classical gradient descent algorithms. (Conjugate Gradient, Fast Gradient Method, ...)

Over the years, phase diversity has been proven to be efficient and reliable. It requires none or minimal extra optical hardware making it easily implementable. The aberration parameters have been successfully estimated in a wide range of configurations. [7][12] For example, it has been used to calibrate the NAOS-CONICA instruments of the very large telescope. [13]

## 2.3 Machine learning

Machine learning is a subset of artificial intelligence focused on the supervised or unsupervised learning of algorithms. On the contrary to classical algorithms, the training task relies on pattern and features identification without explicit instructions. The algorithms are iteratively trained based on sample data and their internal parameters are progressively tuned to achieve high level of performance. In this section, the theoretical background related to one class of machine learning algorithms, called artificial neural networks are introduced and the current state-of-the-art applications to wavefront sensing are reviewed.

### 2.3.1 Neural networks

#### 2.3.1.1 Description

Neural networks or artificial neural networks were initially proposed to mimic the neural function of the human brain. In 1943, Warren McCulloch and Walter Pitts introduces one of the first modelling nowadays famously known as the threshold logic unit. [14] More than 10 years later, in 1957, Frank Rosenblatt proposes in turn a second pioneering architecture called the perceptron. [15]

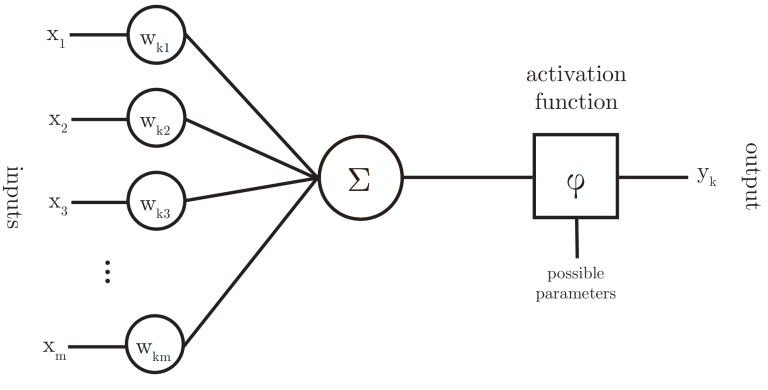



FIGURE 2.8: Computational graph of an artificial neuron. [14]

An artificial neuron $k$ receives $m$ inputs variables denoted $x_i \quad \forall x = 1, ..., m$. These inputs variables are modulated using internal parameters $w_{k,i}$ called weights. The inputs and weights are then summed and fed to an activation function $\varphi$.

$$y_k = \varphi\left(\sum_{i=0}^{m} w_{k,i} x_i\right) \tag{2.41}$$

The weights are trainable parameters which are carefully tuned to achieve the desired output. These single units can then be arranged in parallel to form layers and the layers can themselves be arranged in a series to create a fully-connected multilayer neural network, also called multilayer perceptron (MLP). It typically includes an input layer, one or multiple hidden layers and an output layer. Such neural networks are called fully-connected due to the fact that each neuron is connected to every units in the previous and the nex layers, see figure 2.9. It enables a great neuronal connectivity but can easily becomes very computational expensive.

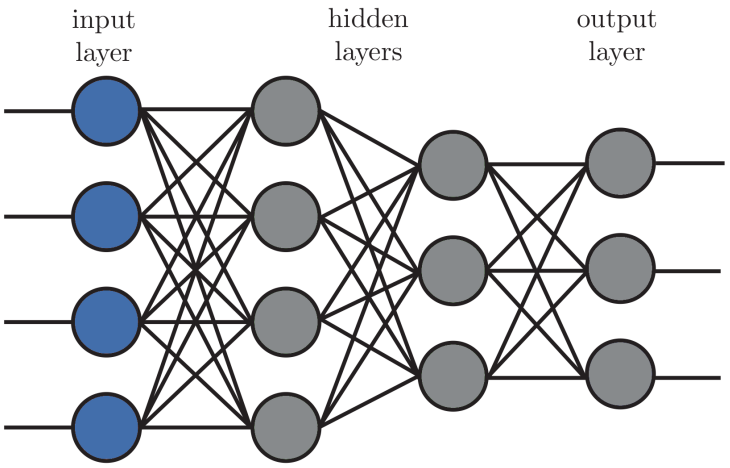
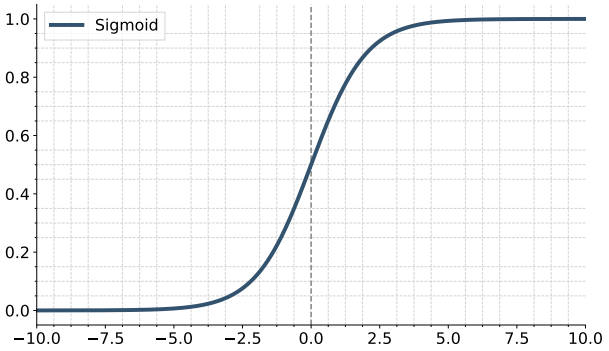



FIGURE 2.9: Fully-connected multilayer neural network.

One can also demonstrates that a feedforward network with single layer is a universal function approximator (assuming that the layer is large enough). [16] This property is partially due to the non-linearities introduced by the activation functions. Among the most popular ones:

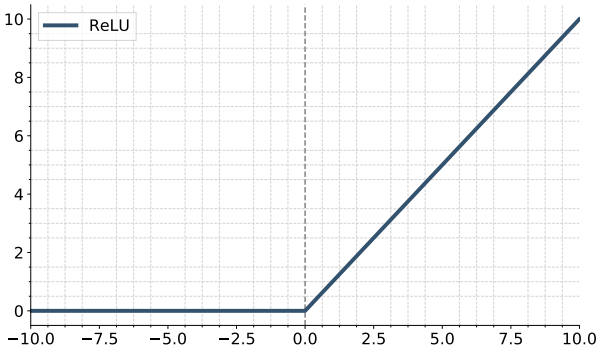
Sigmoid activation function:

$$\varphi(x) = \frac{1}{1 + \exp(-x)} \qquad (2.42)$$




Rectified linear unit (ReLU):

$$\varphi(x) = \max(0, x) \qquad (2.43)$$




#### 2.3.1.2 Application to wavefront sensing

Since 1990s, neural networks have been used to estimate wavefront aberrations. Angel et al. [17] first trained a fully connected neural network to perform multiple telescope array correction. They showed that using a pair of in-focus and out-of-focus images, a neural network could infer the wavelength path variations and the wavefront tilt between the array elements. The neural network architecture was fairly simple with a single layer of hidden nodes and sigmoid activation functions. Nevertheless, this first step showed that the neural-net approach could be used for atmospheric compensation or alignment error sensing.

Shortly after this first successful use of neural network, multiple researches were initiated. Among the most notable, Sandler et al. [18] reiterates the experiment based on a real star. The network was this time train to estimate the first 10 Zernike modes of the wavefront. The objective function was the squared error between the real and estimated Zernike coefficients. A illustration of the neural network architecture is given in the figure 2.11. The neural network performance was also compared to conventional Hartmann sensor.

Finally, artificial neural networks have also been used to estimate the optical aberration of space telescopes. In 1993, Barret et al. [19] used a similar network architecture to recover

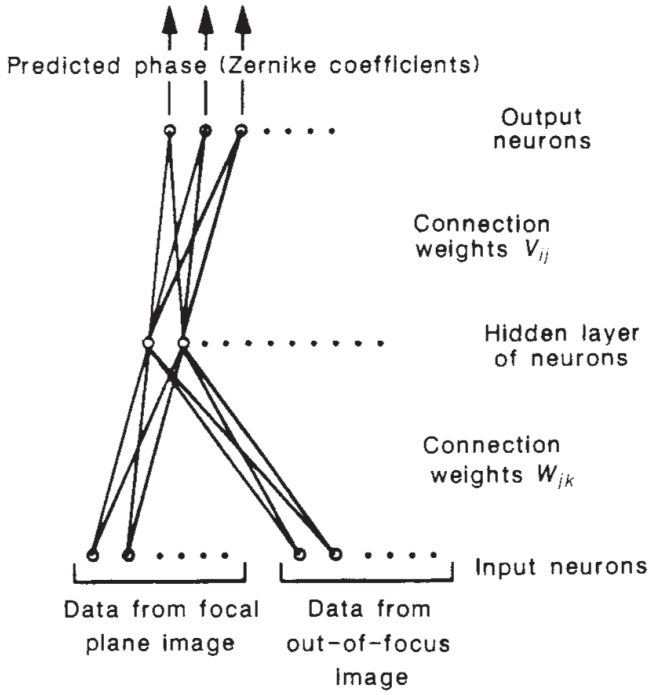



FIGURE 2.10: Schematic representation of the operation of the artificial neural network used to recover the phase. The first layer of neurons takes the intensity (camera data) from the two focal planes. [18]

the static aberrations in the Hubble Space Telescope (HST) primary mirror. Identically to Sandler et al. the network was trained to estimate the first Zernike coefficients and optimized with respect to the mean squared error. They thus successfully recovered phase distortion within only a small residual error.

### 2.3.2 Convolutional neural networks

#### 2.3.2.1 Description

Regular fully-connected neural networks do not scale well to high dimension images. For example, let us consider a fully-connecteded network with a single layer composed of $k$ nodes. For an input image of $128 \times 128 \times 2$, the number of interconnections (and weights) is given by $k \times 128 \times 128 \times 2$. If $k$ is small it may not be a problem but it highly limits the representation capacity of the network, therefore large values of $k$ are often needed and cause fully-connected networks to be particularly inefficient to process images. Furthermore, fully-connected networks do not preserve the spatial organization of the images through the network. They are therefore difficult to deal with and to interpret.

Convolutional neural network (CNN or ConvNet) considers another approach, they take advantage of the hierarchical pattern in the images. The idea behind CNN is inspired by a biological concept: the receptive field. The receptive field a portion of sensory space which can trigger the activation of neuronal cell. It basically acts as a detector sensitive to a particular type of stimuli. For instance, an edge or a color. Convolutional neural networks approximates this biological function using the convolution operation.

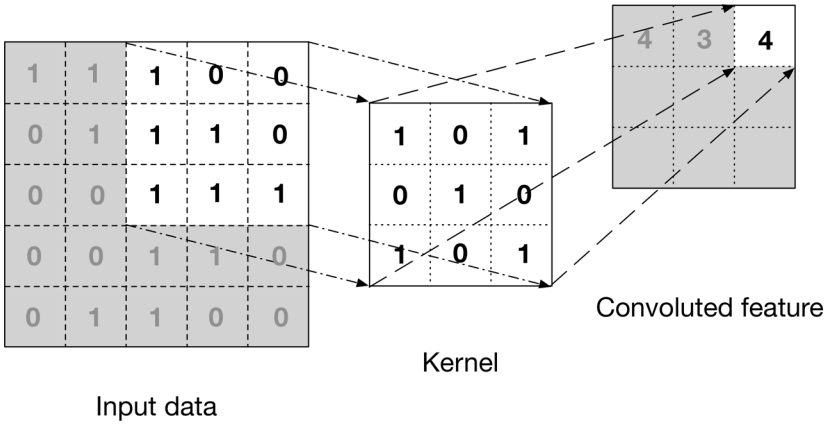



FIGURE 2.11: Example of convolution operation for input data $\boldsymbol{x} \in \mathbb{R}^{1\times5\times5}$ and a convolution kernel $\boldsymbol{u} \in \mathbb{R}^{1\times3\times3}$. [Li Yin, Computer vision blog]

Let us consider an input $\boldsymbol{x}$ of size $\mathbb{R}^{C\times H\times W}$ (e.g. $2 \times 128 \times 128$) and a convolution kernel $\boldsymbol{u}$ of receptive field $\mathbb{R}^{C\times h\times w}$ (e.g. $2 \times 3 \times 3$). The convolution operation consists in sliding the kernel across the input image and sum the element-wise product between the overlapping input elements and the kernel weights ($\boldsymbol{u}$, $\boldsymbol{b}$).

$$\boldsymbol{o}_{i,j} = \boldsymbol{b}_{i,j} + \sum_{c=0}^{C-1} (\boldsymbol{u_c} * \boldsymbol{x_c})[i,j] = \boldsymbol{b}_{i,j} + \sum_{c=0}^{C-1}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} \boldsymbol{u}_{c,n,m}\boldsymbol{x}_{c,n+i,m+j} \tag{2.44}$$

The output $\boldsymbol{o}$ has the dimension $\mathbb{R}^{C\times(H-h+1)\times(W-\omega+1)}$. A multilayer convolutional neural network can then be built by repeating the convolution operation on the output.

### Pooling layers

Along with the standard convolution layers, pooling layers are often used to down-sample the feature map. It is mainly used to reduce the input dimension while preserving the spatial organization. Considering an input tensor $\boldsymbol{x} \in \mathbb{R}^{C\times(rh)\times(sw)}$ and pooling filter of

size $h \times w$, the max pooling operation is defined as

$$\boldsymbol{o}_{i,j} = \max_{n<h,m<w} \boldsymbol{x}_{c,ri+n,sj+m} \tag{2.45}$$

The output $\boldsymbol{o}$ has the dimension $\mathbb{R}^{C \times r \times s}$. Pooling layers also tends to decrease potential over-fitting effects.

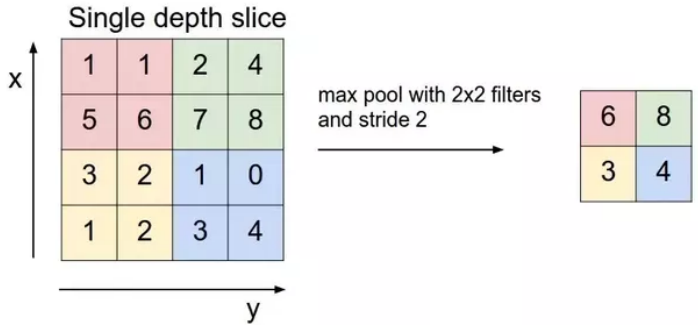



FIGURE 2.12: Max-pooling operation (kernel 2x2, stride 2x2) [20]

### Batch normalization layers

Batch normalization layers [21] are also often used to control the output of the convolution layers. During the training, the distribution of the activations is constantly changing and tends to slow down the training as the layers must adapt themselves to changing distributions. Batch normalization layers solve this issue by normalizing the input of each layer. First the mean and the variance of the layers are computed over a batch

$$\mu_B = \frac{1}{B} \sum_{b=1}^{B} \boldsymbol{x}_b \tag{2.46}$$

$$\sigma_B^2 = \frac{1}{B} \sum_{i=b}^{B} (\boldsymbol{x}_b - \mu_B)^2 \tag{2.47}$$

Then, the input $x$ is normalized, scaled and shifted

$$\boldsymbol{o}_b = \gamma \frac{\boldsymbol{x}_b - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} + \beta \tag{2.48}$$

$\gamma$ and $\beta$ are parameters optimized during the learning.

#### Optimization

Neural networks performances are assessed with respect to a given metric $L(\theta)$, for example the root mean squared error (RMSE). This metric depends on the network parameters $\theta$ and is typically computed over a small subset of the data called a batch,

$$L(\theta) = \frac{1}{B}\sum_{b=1}^{N} \ell\big(\boldsymbol{y}_b, f(\boldsymbol{x}_b;\theta)\big) \tag{2.49}$$

where $\boldsymbol{x}_b$ represents the network input, $\boldsymbol{y}_b$ the label and $f$ the network function. The goal of the training is to minimize the error between $\boldsymbol{y}_b$ and the network output value $f(\boldsymbol{x}_b;\theta)$. The network parameters are thus carefully tuned by batch gradient descent

$$\nabla L(\theta) = \frac{1}{B}\sum_{b=1}^{N} \nabla_\theta \ell\big(\boldsymbol{y}_b, f(\boldsymbol{x}_b;\theta)\big) \tag{2.50}$$

$$\theta_{t+1} = \theta_t - \gamma \nabla L(\theta) \tag{2.51}$$

where $\gamma$ is called the learning rate. The learning rate defines the step length taken in the inverse direction the gradient. A high learning rate enables to train the model faster but increase the probability of missing some optimum. At opposite, a small learning rate has higher probability of converging to an adequate minimum but results in slower training. Furthermore, if the slope of the optimization landscape varies too much along different directions, a constant learning rate is often inappropriate and yield poor training performances. To counteract this problem, different flavours of the stochastic gradient descent algorithms have been proposed: SGD with momentum, Adaptive Moment Estimation (Adam), RMSProp...

#### 2.3.2.2 State-fo-the-art architecture

#### VGG-16

VGG is a very deep convolutional network appeared in 2014 after reaching the second place of the ILSVRC 2015 challenge. It achieved 92.7% top-5 test accuracy in ImageNet, a data set containing more than 14 millions of images belonging to 1000 different classes. It was proposed by Simonyan and Zisserman from the university of Oxford. [22] It consists of successive $3 \times 3$ convolutional layers followed by max pooling layers and regularly repeated. In this thesis, the same architecture is used, only the last fully connected layers are removed and replaced to perform regression instead of classification. The

training of the network will also take advantages of the pretrained weights of the model on Imagenet to speed-up the training.

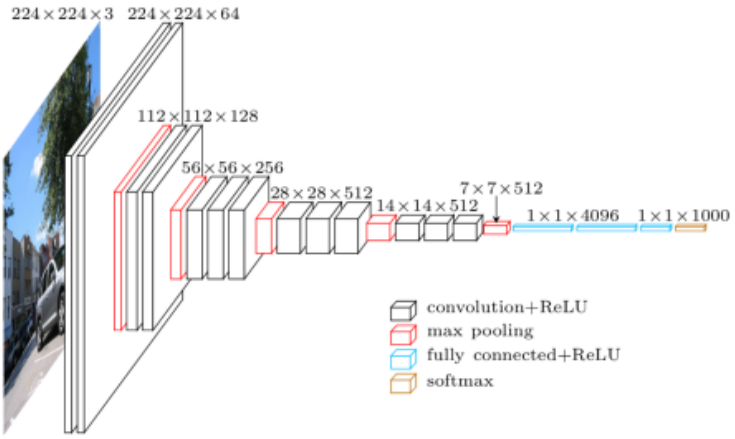



FIGURE 2.13: VGG-16 architecture. [22]

### Resnet

About a year later in 2015, Residual Neural Network (ResNet) [23] appeared and introduced an innovative architecture based on skip connections and systematic batch normalization. One of the motivations of this architecture is the vanishing gradient problem. In very deep neural networks, the intensity of gradients may decrease from layers to layers during the backpropagation steps. To overcome this issue, the introduction of skip connections ensure the smooth and easy flow of the gradients. This architecture enables to create very deep networks with up to 152 successive layers.

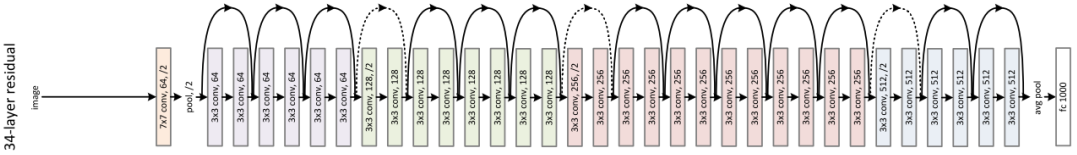



FIGURE 2.14: Resnet-34 architecture. [23]

### Inception v3

The inception v3 architecture [24] proposes an alternative to the assembly conventional layers: instead of creating very deep networks which are particularly difficult to optimize, the idea is to make the network wider instead of deeper. The solution is to use at the same level multiple filters of different sizes and shapes. For example, $1\times1$, $3\times3$ and $1\times7$ convolution kernels are applied to the same feature map and each one detects specific patterns depending on its receptive field properties. Secondly, it also uses an auxiliary

classifier as regularizer. The architecture is shown in the Figure 2.15. For this project, a input convolutional layer and a pooling layer have been added at the start of the network. It ensure that the input features map contains 3 channels (as in standard RGB images). The last fully connected layers have also been modified to perform regression.

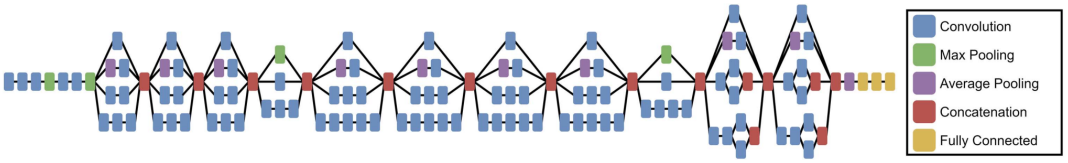



FIGURE 2.15: Adapted Inception v3 architecture used to predict Zernike coefficients. The layers flow from left to right, where the input is fed to the furthest left convolutional block, and the output comes from the furthest right fully connected block. [11]

### Unet

The U-Net is a convolutional neural network initially proposed to perform biomedical image segmentation. Its name come from the U-shaped geometry of its architecture. It can be divided in three parts: the contraction, the bottleneck and the expansion. First, the contraction part is made of successive $3 \times 3$ convolution layers followed $2 \times 2$ max pooling layers. The operation is repeated multiple times to progressively downsample the feature map. Secondly from the extracted features contained in the bottleneck, the expansion part progressively upsamples the data using unpooling layers and skip connections. At the end, a image of the same size than the input image is recovered. In this project, to perform regression instead of segmentation, the last Softmax layer has been removed.

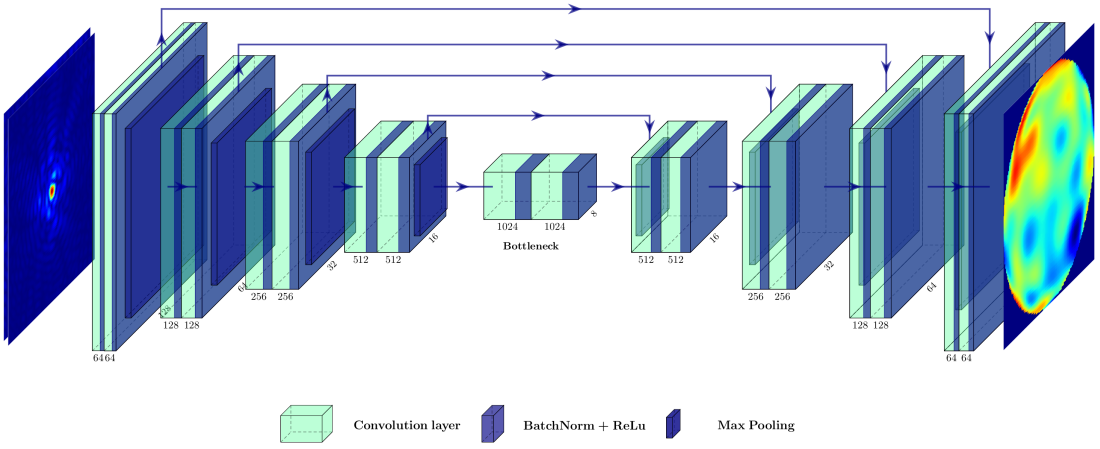



FIGURE 2.16: Unet architecture.

#### 2.3.2.3 Application to wavefront sensing

One of the first use of deep neural networks to the phase problem was performed in 2017 by A. Sinha et al. [25] They demonstrated for the first time in computational imaging that a deep neural network (DNN) can be trained to solve inverse problem. Using raw intensity images, they showed that their network architectures was able to recover the phase object. Even thought this research is not directly focused on astronomical images, it suggests new opportunities for image-based wavefront sensing.

Following this idea in march 2018, S. Paine and J. R. Fienup applied deep conventional neural networks to estimate the Zernike coefficients from aberrated PSFs. [11] Using $256 \times 256$ px input images, they successfully trained a state-of-the-art CNN architecture called Inception V3. They showed that convolutional neural networks could be used in collaboration with iterative algorithms to solve the «capture range problem» and produce initial guess close enough to the exact solution.

Finally in January 2019, Y. Nishizaki et al. presented a new class of wavefront sensors based on conventional neural networks. [26] Using different preconditioners: overexposure, defocus or scattering, they showed that the estimation of the Zernike coefficients from point spread function could be improved. Furthermore, they expanded the method to extended sources and demonstrated that despite a small accuracy drop, the Zernike coefficients were still well estimated.

# Chapter 3

# Simulations

In this chapter, we will start by introducing the data set, the data simulation methods and the data preprocessing. Then, the performance of several deep convolution neural network architectures (CNN) on this specific data set are reviewed. Two configurations are investigated. First, the estimation of the Zernike coefficients from distorted point spread functions (Resnet, Inception, ...) and secondly the direct estimation of the corresponding phase without decomposition on a modal basis (Unet). Finally, the overall accuracy and robustness of CNNs are investigated and their performance is compared with conventional iterative algorithms.

## 3.1 Data description

The data consists of a set of numerically simulated and aberrated PSF pairs: in-focus and out-of-focus. The two images are then concatenated along the Z-axis and form an input feature map of size $128 \times 128 \times 2$. Each pair is labelled by the corresponding phase map, i.e. the wavefront aberrations. Two distinct data set of $100,000$ images are constituted. The first one encompasses only the first 20 Zernike modes while the second has been extended up to 100 Zernike modes. In both situations, the piston mode is discarded.

Concerning the physical parameters, the observation wavelength $\lambda$ has been chosen in the near-infrared at $2.2\mu m$ and the telescope diameter has been set to 10m. This is for example representative of existing instruments such as NIRC-2 at Keck Observatory which operates in-between 1 and $5\mu m$ with a primary mirror of $10m$. The pixel scale of the science camera is fixed to 0.01 arcseconds per pixel. It ensures that the sampling rate is slightly greater than the Nyquist sampling, i.e. several pixels cover the main PSF peak. The number of phase points across the pupil is also set to 128

### 3.1.1 Simulation process

The numerical generation of the data set is performed in Python (v3.7). To begin, the first $n$ (20 or 100) Zernike polynomials are computed using the Aotools[1] library. The pupil mask is also sampled over $128 \times 128$ pixels. The choice of the pupil size results from the trade-off between a good pupil modelling (Figure 3.1) and a reasonable computation time.

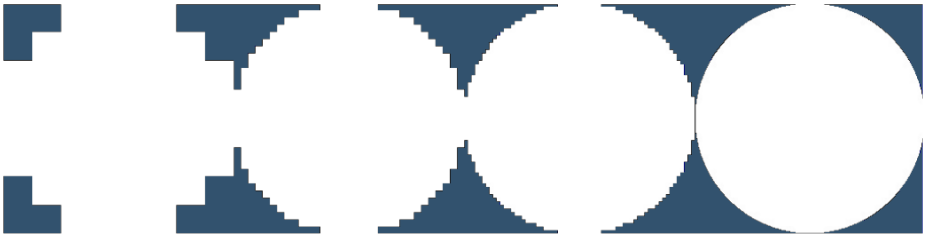


FIGURE 3.1: Pupil shape discretized over respectively 8, 32, 128 and 256 pixels. [13]

Secondly, to numerically reproduce non-common path aberrations, the phase map must reflect typical errors of conventional optics. This is achieved by randomly generating the Zernike coefficients and dividing each coefficient by its radial order (Figure 3.2). This procedure approximates a power spectral density profile, $S \approx 1/f^2$ (where $f$ is the spatial frequency). Such profile is for instance frequently produced by polishing errors. [27] The phase map $\theta(x', y')$ can then be constructed as a linear combination of the Zernike polynomials.

$$\theta(x', y') = \sum_{i=0}^{\infty} c_i Z_i(x', y') \tag{3.1}$$

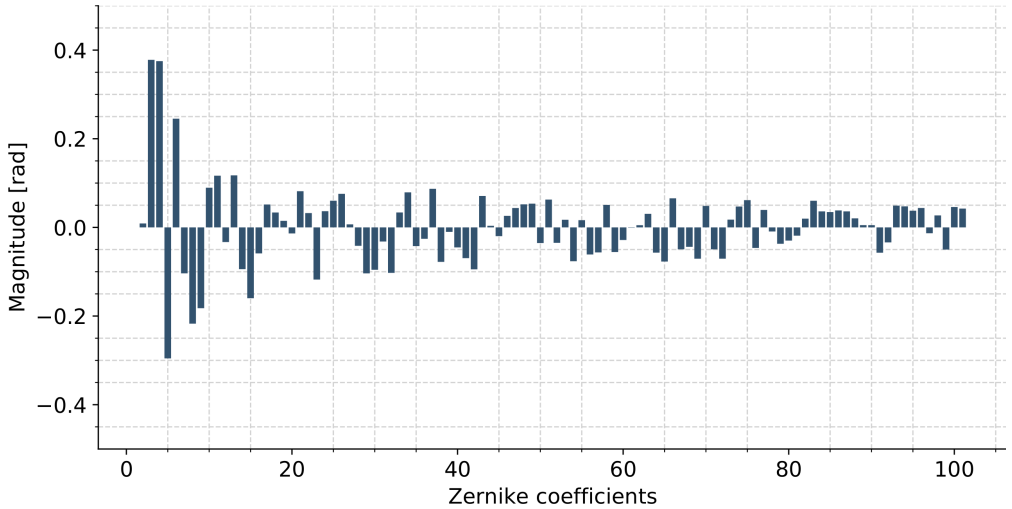



FIGURE 3.2: Example of randomly generated Zernike coefficients following a $1/f^2$ PSD.

[1] `https://github.com/AOtools/aotools`

And the corresponding power spectral density profile is illustrated in the Figure 3.3.

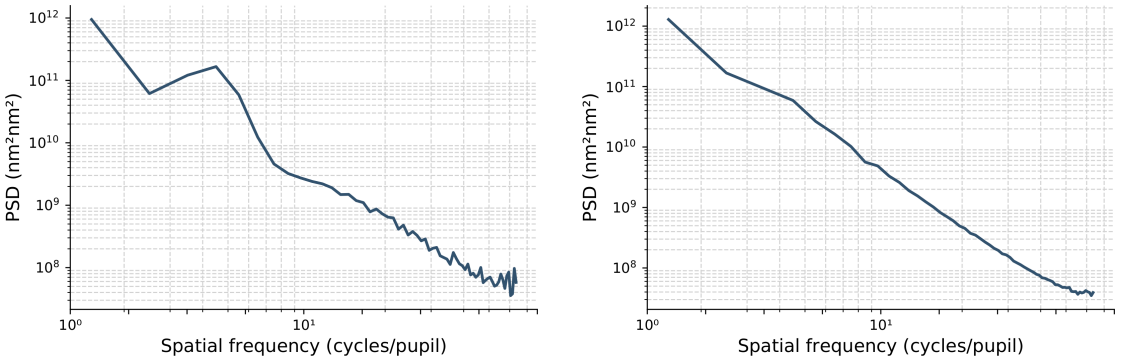



FIGURE 3.3: Power Spectral Density profile, **Left:** slice **Right:** Azimutal averaging

Once the phase map obtained, the optical aberrations are numerically propagated through the optical system. The point spread functions were thus obtained using a Monte-Carlo adaptive optics simulation toolkit, called Soapy[2]. Illustrations of the generated point spread functions and the corresponding phase maps are available in the Table 3.1 (20 Zernike modes) and in the Table 3.2 (100 Zernike modes). The extension to 100 Zernike polynomials introduces higher order modes which creates more rapidly varying phase maps.

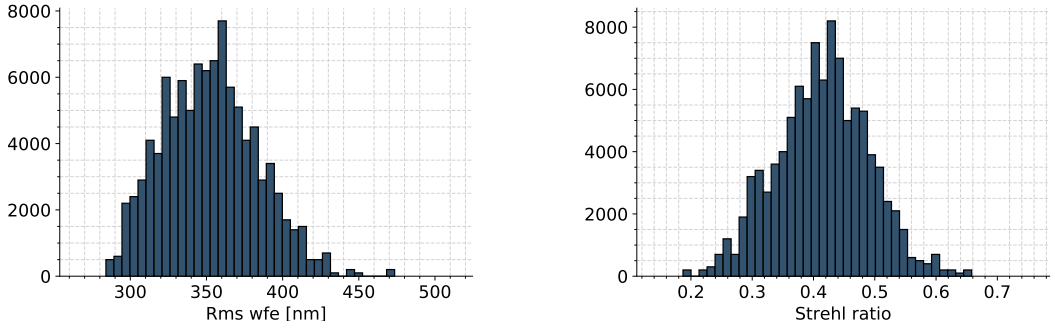



FIGURE 3.4: First data set, 20 Zernike modes. **Left:** Root mean squared wavefront error (RMS WFE) of the phase map **Right:** Strehl ratio of the in-focus PSF.

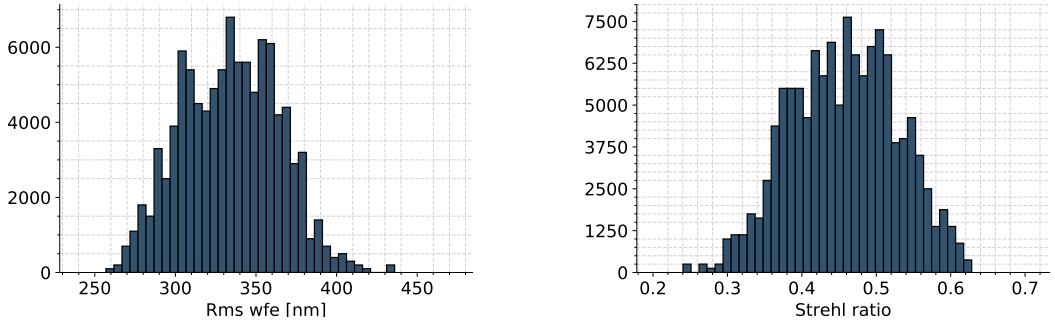



FIGURE 3.5: Second data set, 100 Zernike modes. **Left:** Root mean squared wavefront error (RMS WFE) of the phase map **Right:** Strehl ratio of the in-focus PSF.

[2] `https://github.com/AOtools/soapy`

To characterize the generated data set, the root mean squared wavefront error (RMS WFE) distribution of the phase map and the distribution of the strehl ratio (in-focus PSF) are computed. It indicates us that the optical aberrations in both data set extend from 290 to 400 $nm$ RMS and roughly follow a Gaussian distribution. Also note that both distributions can be related using the Maréchal approximation (Eq 2.15).

### 3.1.2 Noise

Finally, noise is added to the point spread functions. The noise encountered in astronomical images may be split into various contributions. First assuming that a only fraction of the incoming photons is converted into electrons the total number of electrons $N_{tot}$ collected at a given site (i.e. pixels) can be expressed as [28]

$$N_{tot} = N + N_S + N_{DC} + N_R \tag{3.2}$$

- $N$, corresponds to the number of electrons generated by light at a given site. It corresponds to a fraction $\eta$ of the number of photons. The ratio electron/photon $\eta$ is called the quantum efficiency.

- $N_S$, the shot noise results from the quantum properties of light and express an uncertainty in the number of electrons stored at a collection site. It is a fundamental limit and cannot be removed. In practice, the number of electrons follows a zero-mean Poisson distribution $P$ (counting process). In particular, its variance equals its mean $N$.

  $$P(k) = \frac{N^k e^{-k}}{N!} \tag{3.3}$$

  Assuming then $N >> 1$, the Poisson distribution tends towards a Gaussian distribution with a mean of $N$, and a standard deviation of $\sigma = \sqrt{N}$,

  $$P(k) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(k-N)^2/2\sigma^2} \tag{3.4}$$

  Therefore if we detect $N$ photons per second from a star, it will ultimately remains a fundamental uncertainty of $\sqrt{N}$. As this uncertainty scale as the square root of the number of photons, it will mainly affect low intensity measurements.

- $N_{DC}$, is the dark-current noise. It originates from the thermal ionization of the detector itself.

- $N_R$, corresponds to the read noise. It is associated to the electronics of the detectors. More precisely, the output amplifier transforms the charges collected into a

measurable voltage. During this process, a Gaussian zero mean read noise may be generated. Moreover, this noise is independent of the number of collected electrons.

In this project, we consider the cases of bright astronomical objects such as stars. This particular regime is dominated by the shot noise while the other noise sources remain negligible.

$$N_{tot} \approx N + N_S + 0 + 0 \tag{3.5}$$

Furthermore, assuming long enough exposure time, the signal-to-noise ratio (SNR) has been adjusted to match an average value of 100 over the main PSF peak.

$$SNR_{\text{peak}} = \frac{\text{Signal}}{\text{Noise}} \approx 100 \tag{3.6}$$

As illustrated in Figure 3.6, the noise effects are much more noticeable in peripheral and low intensity regions. The point spread functions, the phase map and the corresponding Zernike coefficients are then stored under the fits format (Flexible Image Transport System[3]) with a float 32 precision.

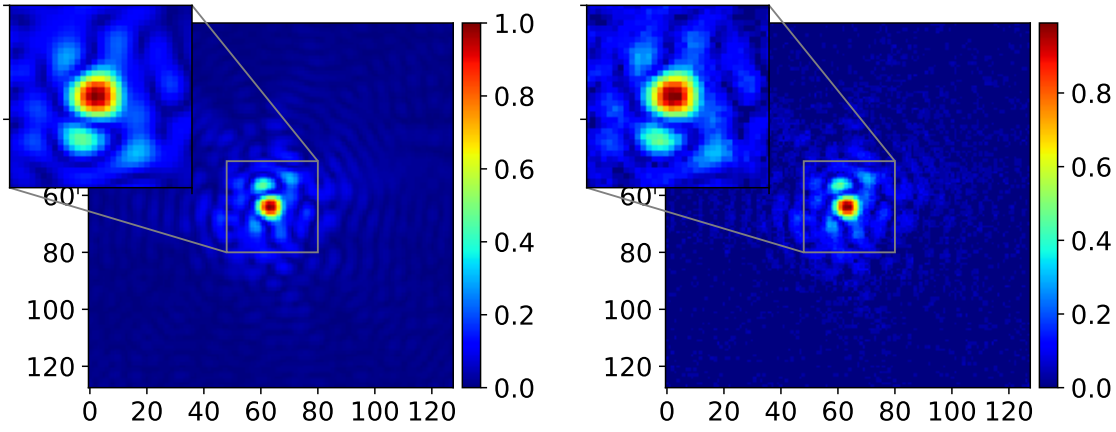



FIGURE 3.6: **Left:** Noise free aberrated PSF **Right:** Noisy aberrated PSF (shot noise)

### 3.1.3 Preprocessing

Finally, before being fed to the convolutional neural networks, the point spread functions will be preprocessed. The first step consists in normalizing the in-focus and out-of-focus PSFs independently. The simulated PSFs may indeed records varying intensity, the

[3] `https://fits.gsfc.nasa.gov/fits_documentation.html`

normalization therefore ensure the uniformity of the input data.

$$PSF'_i = \frac{PSF_i - \min(PSF)}{\max(PSF) - \min(PSF)} \tag{3.7}$$

Alternatively the standardization of the inputs may also be consider, in the particular case of this data set, both methods exhibited similar performances. Then, a stretching operation is also applied to the PSFs. Its main goal is to improve the number of informative pixels and by consequences improve the ability of the CNN to identify important features.

Stretching functions:

$$PSF'_i = PSF_i \tag{3.8}$$
$$PSF'_i = \sqrt{PSF_i} \tag{3.9}$$
$$PSF'_i = \log\left(PSF_i\right) \tag{3.10}$$

Note that the square root preserves the norm $[0, 1] \to [0, 1]$.

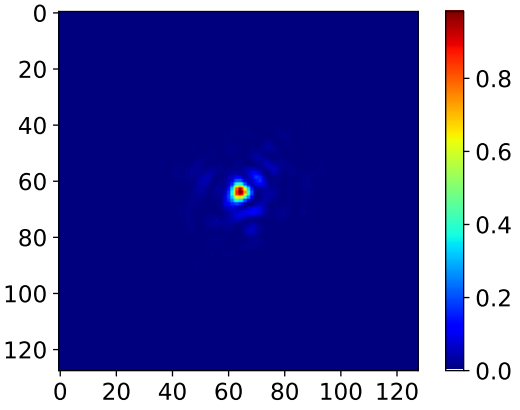



FIGURE 3.7: No stretching

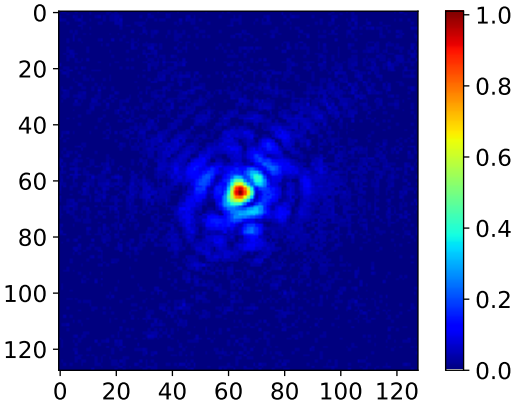



FIGURE 3.8: Square root stretching

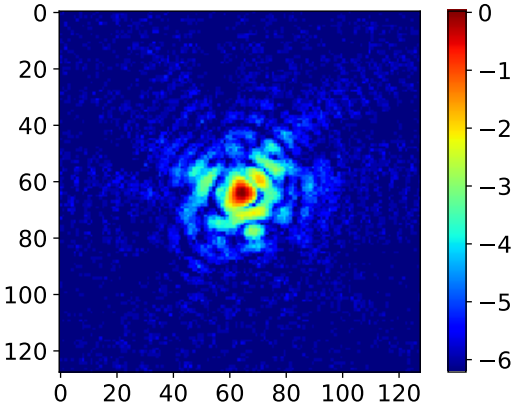



FIGURE 3.9: Logarithmic stretching

To preserve a good signal-to-noise ratio while improving the number of informative pixels, the square root stretching was chosen. The phase map and the Zernike coefficients have also been expressed in radians $[-\lambda, \lambda] \to [-\pi, \pi]$. The standard error functions such as the mean square error are scale sensitive, the range of each target pixels can thus affect the neural network learning. By imposing a physically meaningful scaling, the relative importance of each pixel is therefore constrained.

| Input feature map | | |
|---|---|---|
| In-focus PSF | Out-of-focus PSF | Phase map |

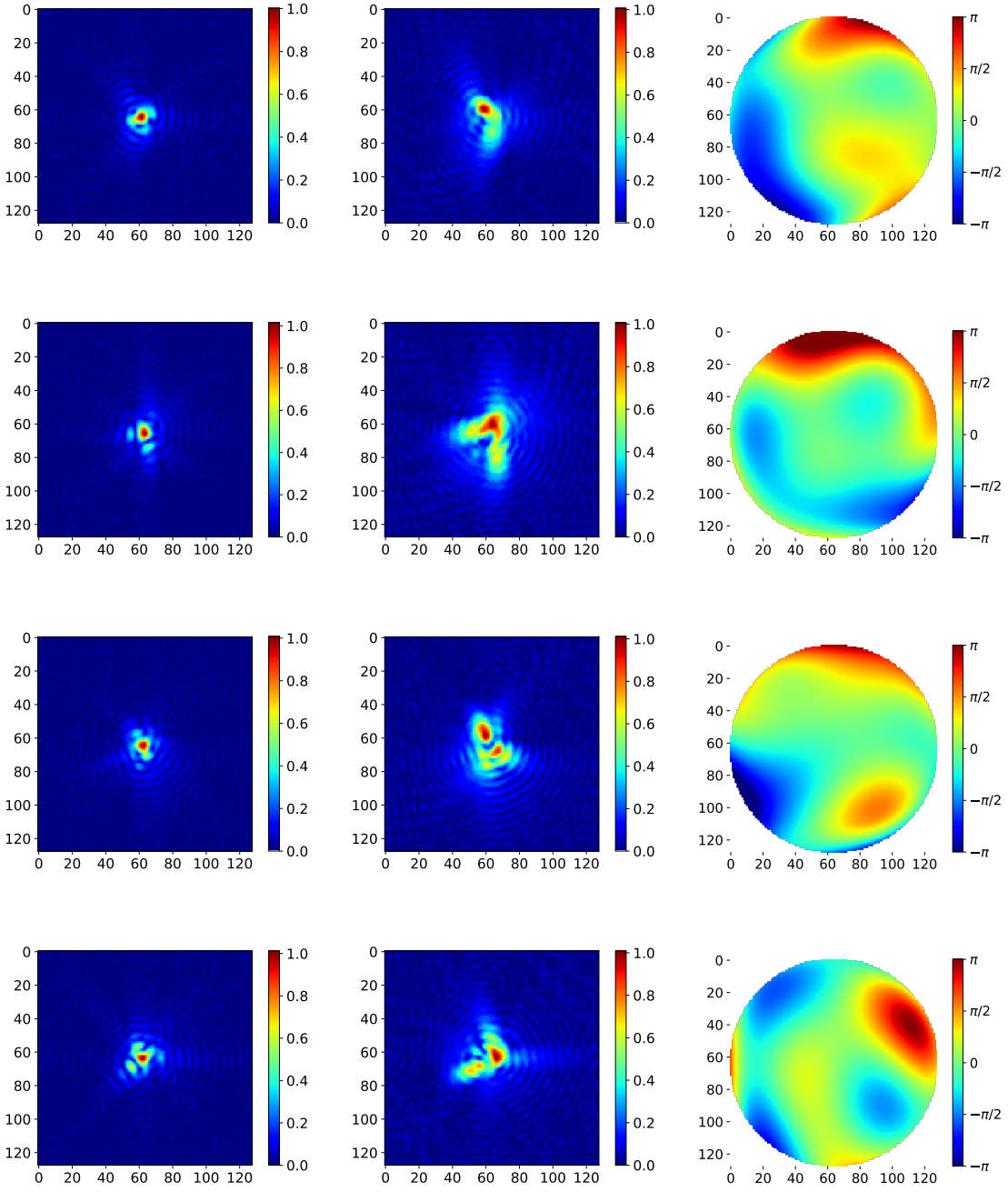


TABLE 3.1: Data set samples randomly drawn from the 100,000 available PSF pairs. The phase map is generated from the 20 first Zernike polynomials (piston excluded).

| | | |
|---|---|---|
| **Average Strehl ratio:** | in-focus 0.4679 | out-of-focus 0.0908 |
| **Average rms wfe:** | in-focus 349.18 $nm$ | out-of-focus 638.98 $nm$ ($\lambda/4$ defocus) |

The pairs of point spread functions constitute the input of the CNN. The phase map is the estimation objective, the networks are trained to minimize the error between the exact and the estimated phase map.

| Input feature map | | |
|---|---|---|
| In-focus PSF | Out-of-focus PSF | Phase map |

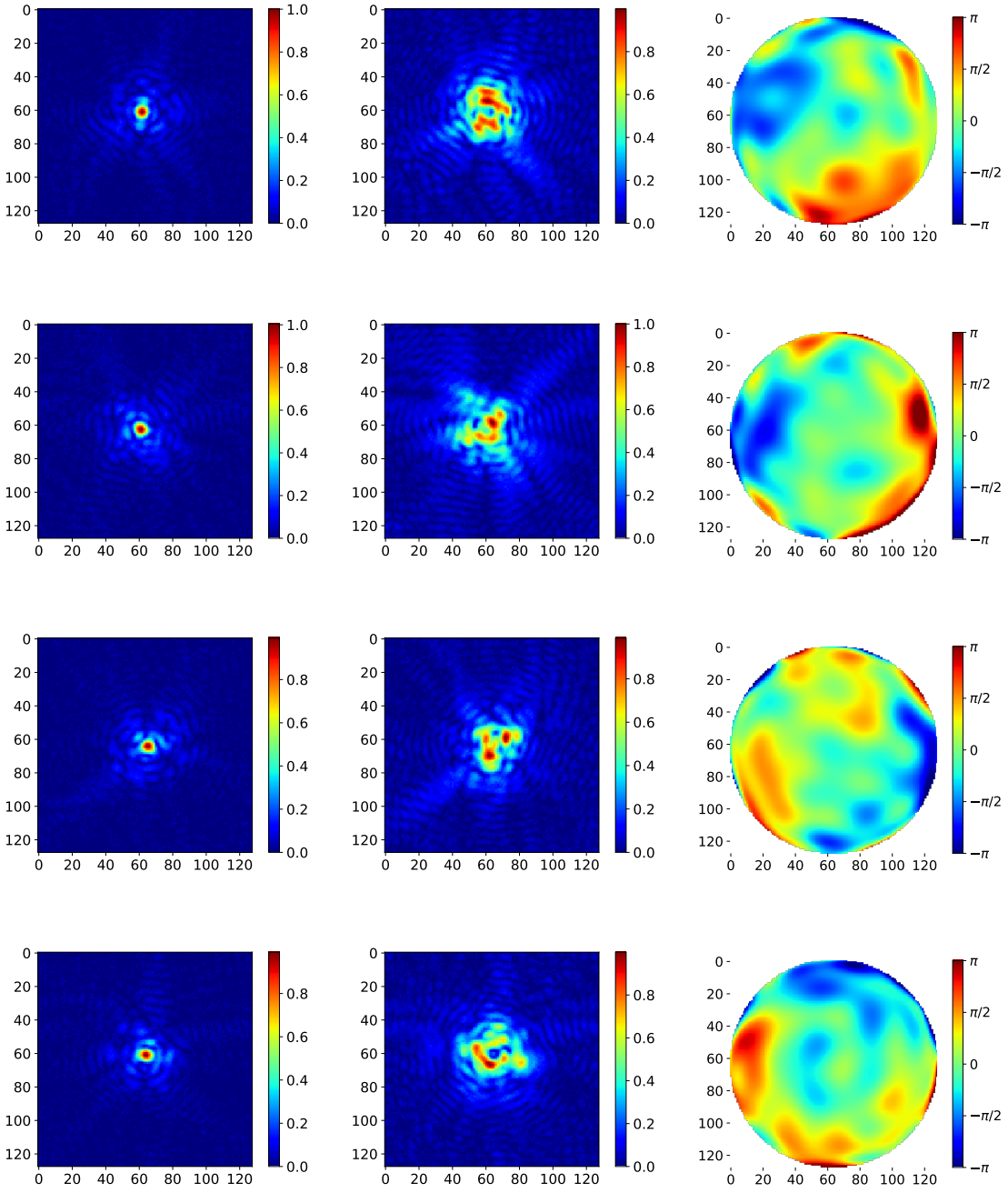



TABLE 3.2: Data set samples randomly drawn from the 100,000 available PSF pairs. The phase map is generated from the 20 first Zernike polynomials (piston excluded).

| | | |
|---|---|---|
| **Average Strehl ratio:** | in-focus 0.5019 | out-of-focus 0.1870 |
| **Average rms wfe:** | in-focus 308.77 $nm$ | out-of-focus 517.14 $nm$ ($\lambda/4$ defocus) |

The pairs of point spread functions constitute the input of the CNN. The phase map is the estimation objective, the networks are trained to minimize the error between the exact and the estimated phase map.

## 3.2 Results 20-Zernike

### 3.2.1 Introduction

In this section, the different networks architectures are compared on the first data set. As reminder, the first data set is generated from 20 Zernike coefficients and an average rms wavefront error of 349.18 $nm$. The data set is respectively split as $90,000$ training images, $5,000$ test images and $5,000$ validation images. The training is performed using the Pytorch library (v1.1) and the networks are trained using 2 GTX 1080. Two approaches are considered. First, convolutional neural networks are used to estimated the Zernike coefficients and the phase map is reconstructed as linear combinations of the estimated coefficients.

$$\theta(x,y) = \sum_{i=0}^{\infty} c_i Z_i(x,y) \tag{3.11}$$

The networks are trained with respect to root mean squared error (RMSE) between the exact ($c_i$) and the estimated ($c_i'$) coefficients.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{20}(c_i - c_i')^2}{20}} \tag{3.12}$$

The state-of-the-art architectures considered are: VGG-16, Resnet-50 and Inception-V3. Secondly, a particular CNN architecture, Unet is trained to directly estimate the phase map. Instead of estimating the Zernike coefficients and then reconstructing the phase, this network will directly provide a pixel-wise estimation of the phase. The Zernike coefficients can then be recovered by projecting the phase onto the Zernike orthogonal basis.

$$c_i = \frac{< \theta(x,y), Z_i(x,y) >}{< Z_i(x,y), Z_i(x,y) >} \tag{3.13}$$

This network is trained with respect to pixel-wise root mean squared error (RMSE) between the exact ($\theta$) and the estimated ($\theta'$) phase map.

$$\text{RMSE} = \sqrt{\frac{1}{N(\Omega)} \sum_{i,j \in \Omega} (\theta(x_i, y_i) - \theta'(x_j, y_j))^2} \tag{3.14}$$

where $\Omega$ is the sampled pupil aperture and $N(\Omega)$ is the total number of pixels within the pupil. For each architecture, the training curve, the error on the estimated Zernike coefficients and rms wavefront error is presented.

### 3.2.2 Models

#### 3.2.2.1 VGG-16

The VGG architecture has been trained over 300 epochs using the Adam optimizer. The batch size has been fixed to 128 and the learning rate has been set to $10^{-5}$. The learning rate is kept constant, decreasing it or increasing the number of epochs do not improved the results. Also, note that the training do not show characteristics of overfitting even though no regularization was applied. The network is initialized using Imagenet [29] pretrained weights.

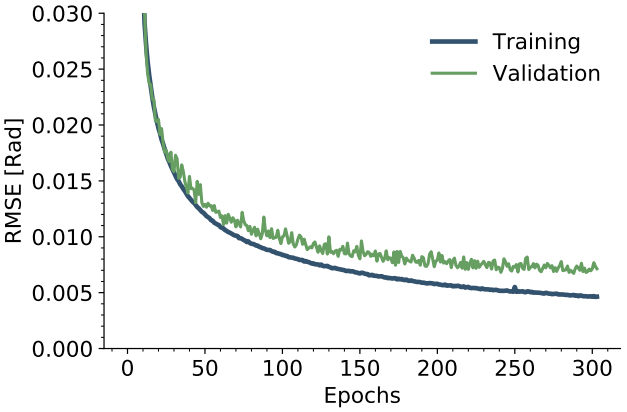



FIGURE 3.10: VGG-16 validation and training learning curve of the RMSE over the Zernike coefficients. Optimizer=Adam, Learning rate $= 10^{-5}$ and batch size=128

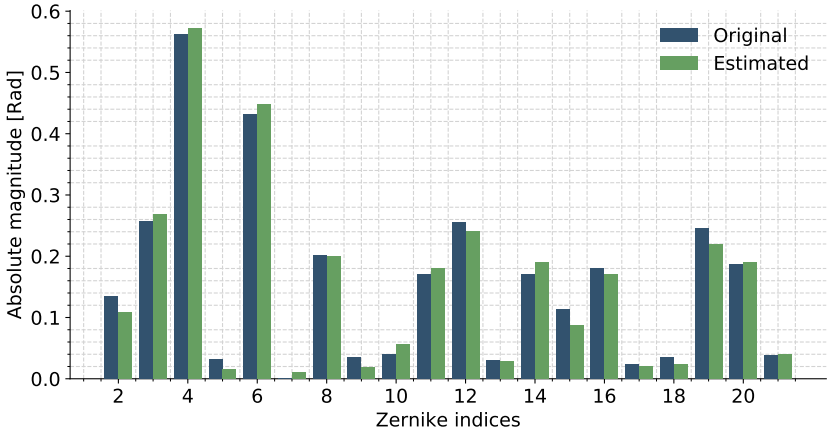



FIGURE 3.11: Comparison between the exact and the estimated Zernike coefficients from in-focus and out-of-focus PSF using VGG-16.

The final rms error over the Zernike coefficients and the phase map are computed using Monte Carlo simulations over the test set.

| | RMSE | Unit | RMSE | Unit |
|---|---|---|---|---|
| Coefficients | $0.0107 \pm 0.0018$ | $[Rad]$ | $3.746 \pm 0.630$ | $[nm]$ |
| Phase | $0.0387 \pm 0.0059$ | $[Rad]$ | $13.505 \pm 1.815$ | $[nm]$ |

TABLE 3.3: Root mean squared error over the Zernike coefficients and the phase map for the VGG-16 architecture.

#### 3.2.2.2 Inception v3

The Inception architecture has been trained over 350 epochs using the SGD optimizer with momentum=0.9. The training of this network is a bit more tricky, the training is first performed from scratch. The batch size has been fixed to 64 and the learning rate has been set to $10^{-4}$ for the convolution layers and $10^{-3}$ for the fully connected layers. The learning rate is kept constant along the epochs. The most noticeable difference with the VGG is the higher variability of the validation error.

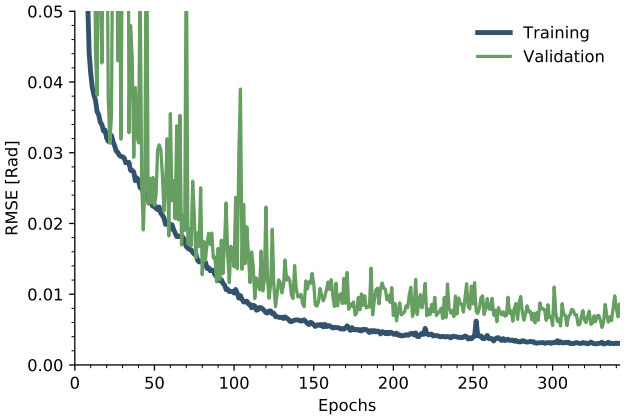



FIGURE 3.12: Inception v3 validation and training learning curve of the RMSE over the Zernike coefficients. Optimizer=SGD, momentum = 0.9, Learning rate = $10^{-4}$ (Conv) and $10^{-3}$ (FC), batch size=128

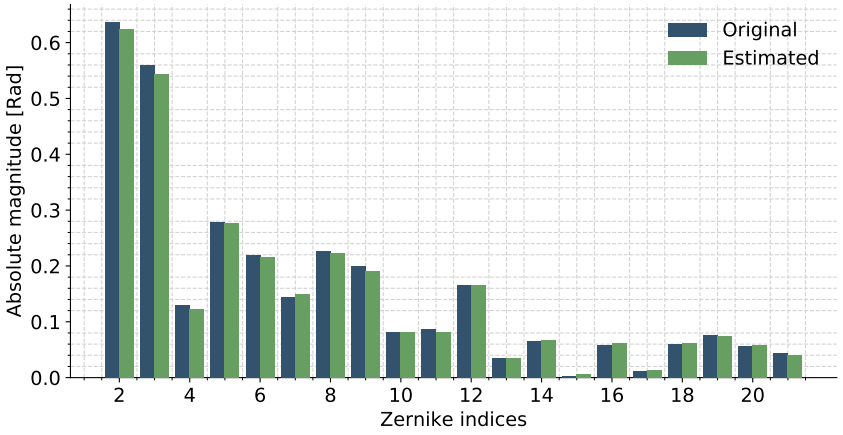



FIGURE 3.13: Comparison between the exact and the estimated Zernike coefficients from in-focus and out-of-focus PSF using Inception v3.

The final rms error over the Zernike coefficients and the phase map are computed using Monte Carlo simulations over the test set.

| | RMSE | Unit | RMSE | Unit |
|---|---|---|---|---|
| Coefficients | $0.0094 \pm 0.0030$ | $[Rad]$ | $3.293 \pm 1.054$ | $[nm]$ |
| Phase | $0.0240 \pm 0.0051$ | $[Rad]$ | $8.432 \pm 1.816$ | $[nm]$ |

TABLE 3.4: Root mean squared error over the Zernike coefficients and the phase map for the Inception architecture.

#### 3.2.2.3 Resnet-50

The Resnet-50 architecture was trained over 400 epochs using the SGD optimizer with momentum=0.9. The weights have been initialized from pretrained values on Imagenet. Nevertheless, no difference in term of the convergence speed nor in term of final accuracy were noticed when training from scratch. The batch size has been fixed to 64 and the learning rate has been initially set to $10^{-4}$. After the 300 first iterations the learning rate is lowered to $10^{-5}$, it results in a direct accuracy gain.

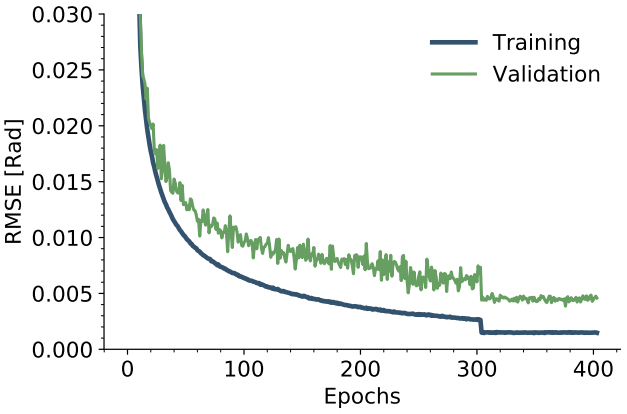



FIGURE 3.14: Resnet-50 validation and training learning curve of the RMSE over the Zernike coefficients. Optimizer=SGD, momentum = 0.9, Learning rate = $10^{-4}$ (then $10^{-5}$), batch size=128

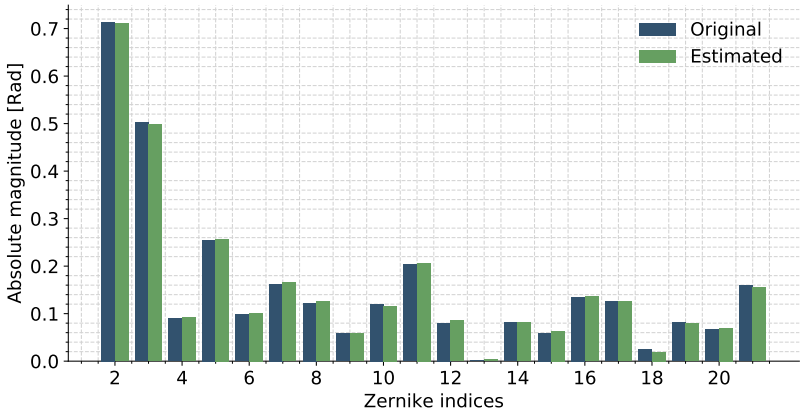



FIGURE 3.15: Comparison between the exact and the estimated Zernike coefficients from in-focus and out-of-focus PSF using Resnet-50.

The final rms error over the Zernike coefficients and the phase map are computed using Monte Carlo simulations over the test set.

| | RMSE | Unit | RMSE | Unit |
|---|---|---|---|---|
| Coefficients | $0.0058 \pm 0.0021$ | $[Rad]$ | $2.030 \pm 0.735$ | $[nm]$ |
| Phase | $0.0187 \pm 0.0039$ | $[Rad]$ | $6.543 \pm 1.365$ | $[nm]$ |

TABLE 3.5: Root mean squared error over the Zernike coefficients and the phase map for the Resnet-50 architecture.

#### 3.2.2.4 Unet

The Unet network was trained over 300 epochs using the SGD optimizer with momentum=0.9. The best results were achieved using Xavier initialization. [30] The batch size has been fixed to 64 and the learning rate has been set to $10^{-3}$. As previously mentioned, the network is directly trained with respect to the pixel-wise RMSE between the exact and the estimated phase.

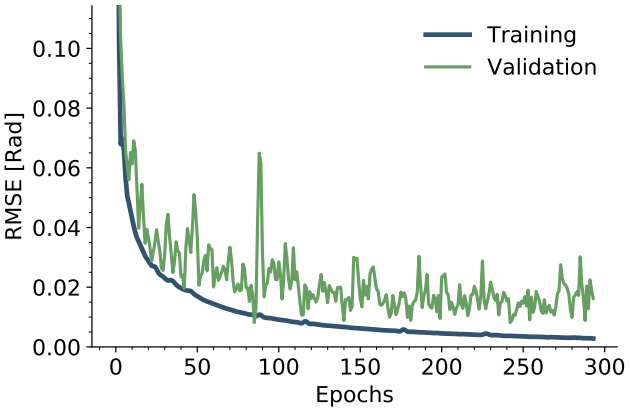



FIGURE 3.16: Unet validation and training learning curve of the RMSE over the phase map. Optimizer=SGD, momentum = 0.9, Learning rate = $10^{-3}$ , batch size=128

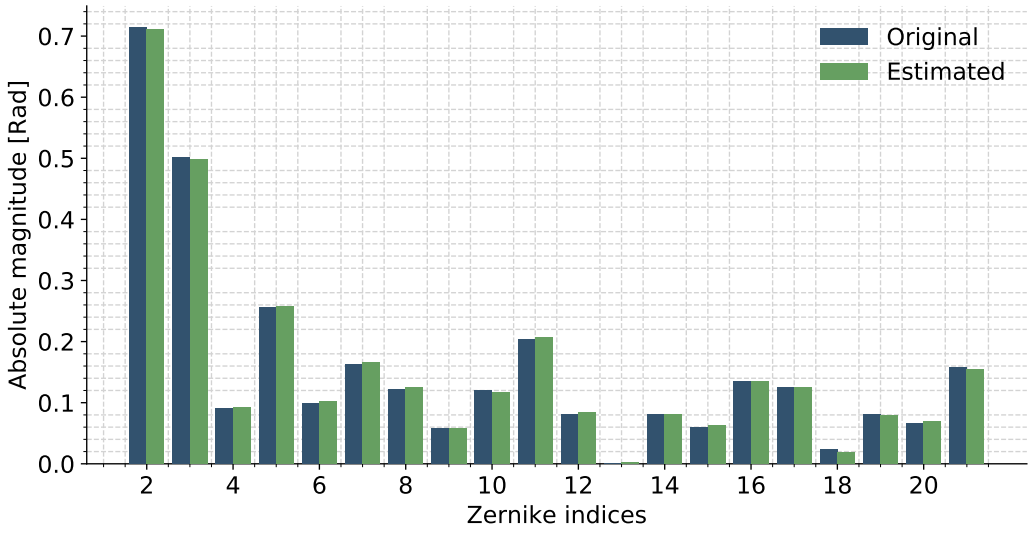



FIGURE 3.17: Comparison between the exact and the estimated Zernike coefficients from in-focus and out-of-focus PSF using Unet.

The final rms error over the Zernike coefficients and the phase map are computed using Monte Carlo simulations over the test set.

| | RMSE | Unit | RMSE | Unit |
|---|---|---|---|---|
| Coefficients | $0.0040 \pm 0.0018$ | $[Rad]$ | $1.400 \pm 0.630$ | $[nm]$ |
| Phase | $0.0132 \pm 0.0019$ | $[Rad]$ | $4.621 \pm 0.665$ | $[nm]$ |

TABLE 3.6: Root mean squared error over the Zernike coefficients and the phase map for the Unet architecture.

### 3.2.3 Comparison

The previous results first confirmed that convolutional neural networks may indeed be used to perform wavefront sensing. In each cases, the wavefront has been successfully estimated within 10 nm RMS error. A performance gap is however noticeable between the investigated architectures. For instance, the Resnet-50 almost reduce by a factor 2 the phase error with respect to the VGG-16 architecture. This is probably due to the inherent simplicity of the VGG architecture while the residual network is built upon a more complex pattern and skip connections. In summary, the final performance of each CNN is shown in the following figure.

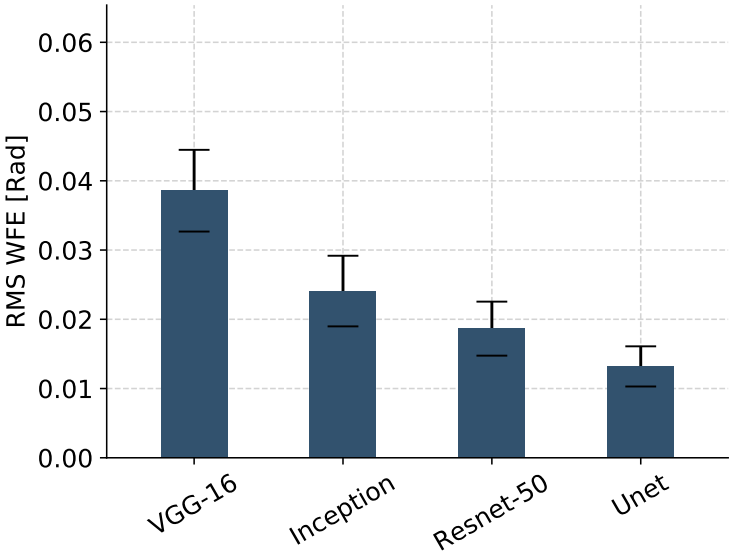



FIGURE 3.18: RMS wavefront error between the exact and the estimated phase map for the different architectures explored.

The network performances may also be compared in term of memory consumption, inference time and in term of the total number of internal parameters.

| Architecture | Nbr of parameters | Size (MB) | Inference time (s) |
|---|---|---|---|
| VGG-16 | $51,713,757$ | 886.86 | $0.3469s$ |
| Inception V3 | $25,153,253$ | 322.69 | $0.1954s$ |
| Resnet-50 | $23,712,941$ | 378.78 | $0.2093s$ |
| Unet | $27,395,265$ | 484.22 | $0.2793s$ |

TABLE 3.7: Average values of the network characteristics for a single input (batch size = 1). The inference time (forward pass) is calculated on CPU, Intel Xeon e3-1230v5.

It is shown that reconstructing directly the phase with a neural net (Unet) outperforms reconstructed phase from estimated Zernike coefficients. One may however have feared that the projection of pixel-wise approximation introduces high order frequency variations. Experimentally, the phase map reconstructed using Unet have shown great smoothness and small pixel to pixel variation. To illustrate these remarks, an example is provided. Let us consider the initially aberrated PSF and the corresponding wavefront.

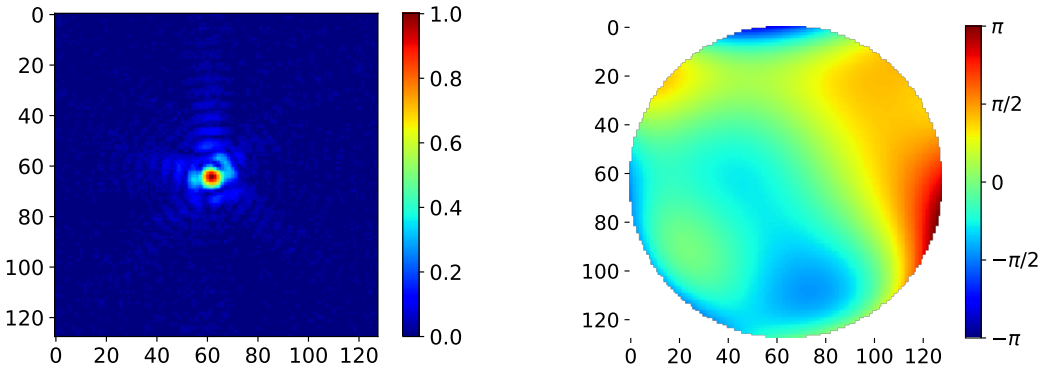



FIGURE 3.19: **Left:** Aberrated in-focus PSF **Right:** Corresponding phase map

Then, using our two best approaches (Unet and Resnet), the previous PSFs (in- and out-of-focus) are fed to the networks and the phase map is estimated. The residual phase map $\theta - \theta'$ is finally compared using the two approaches.

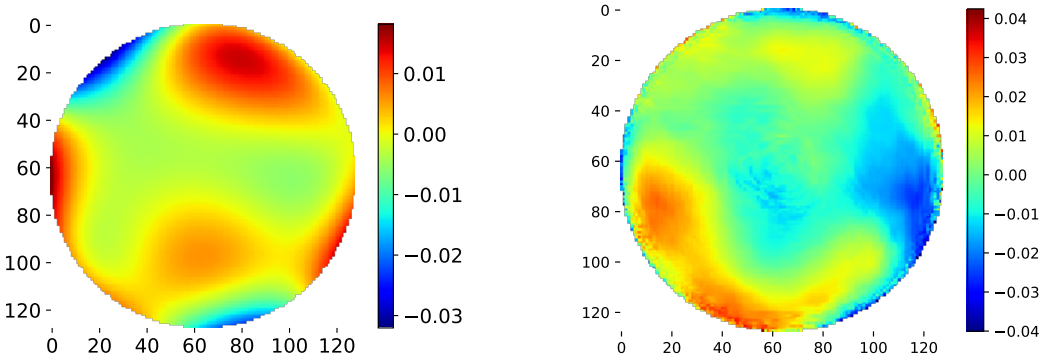



FIGURE 3.20: Residual phase map **Left:** Resnet-50 **Right:** Unet. The colorbar is expressed in [Rad].

First, we notice that both networks provide completely different residual errors. Even though, the networks have been trained using the exact same data, the learning process is closely related to the architecture. Similar results were observed with the remaining architectures (VGG, Inception). Secondly, we also notice that the residual phase map obtained with Unet suffers from pixels-to-pixels discontinuities and a lack of smoothness. Nevertheless due to the small range of the discontinuities in comparison of the initial wavefront, the discontinuities may be considered negligible and are hardly noticeable.

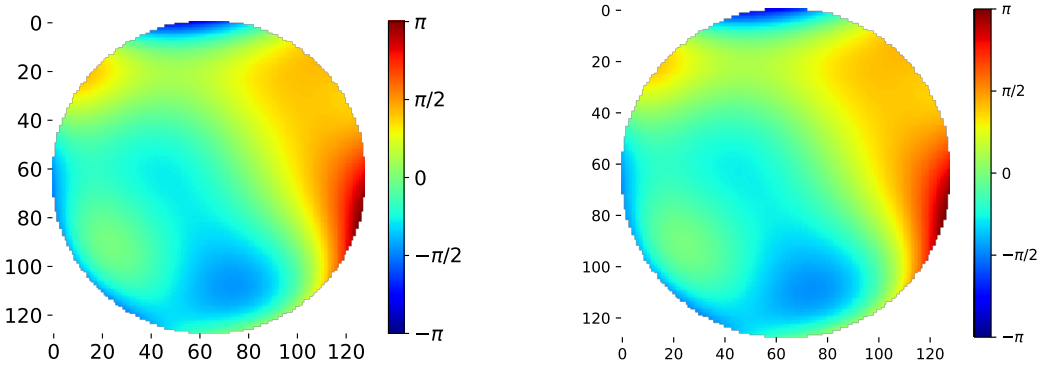



FIGURE 3.21: Estimated phase map **Left:** Resnet-50 **Right:** Unet.

Finally, assuming that we could perfectly apply the estimated phase to correct the wavefront, the point spread functions converge towards the diffraction limits.

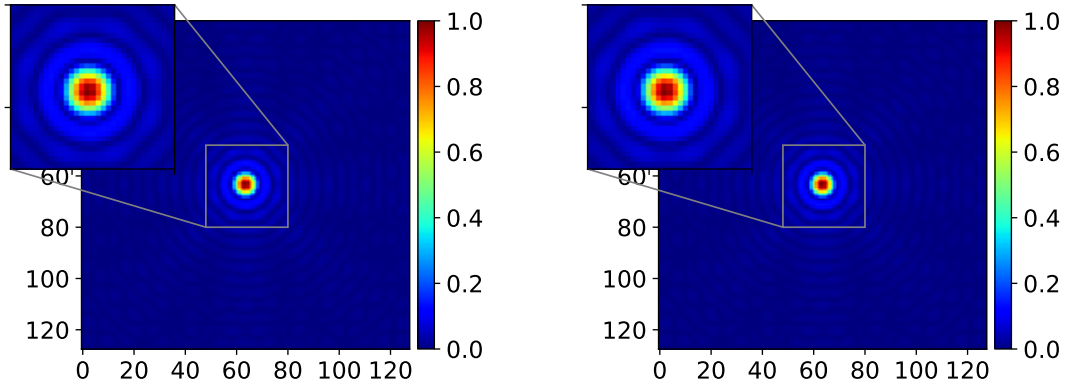



FIGURE 3.22: Corrected PSF **Left:** Resnet-50, strehl ratio=0.9907 **Right:** Unet, strehl ratio=0.9934.

## 3.3 Results 100-Zernike

### 3.3.1 Introduction

In this section, the network robustness is evaluated on a more complex data set. The number of Zernike polynomials has been extended up to 100. The piston mode is still not considered. The average RMS wavefront error is kept similar with the previous data set at an average value of 308.77 $nm$. The data set is respectively split as $90,000$ training images, $5,000$ test images and $5,000$ validation images. As demonstrated, in the first part, the VGG-16 model is slow and comparatively inefficient, we therefore removed it from the list of investigated architectures. Then, to explore even more the capability of deep neural network to directly infer the wavefront map. A slightly modified and improved version of Unet naturally called Unet++ is also considered. [31]

### 3.3.2 Models

#### 3.3.2.1 Resnet-50

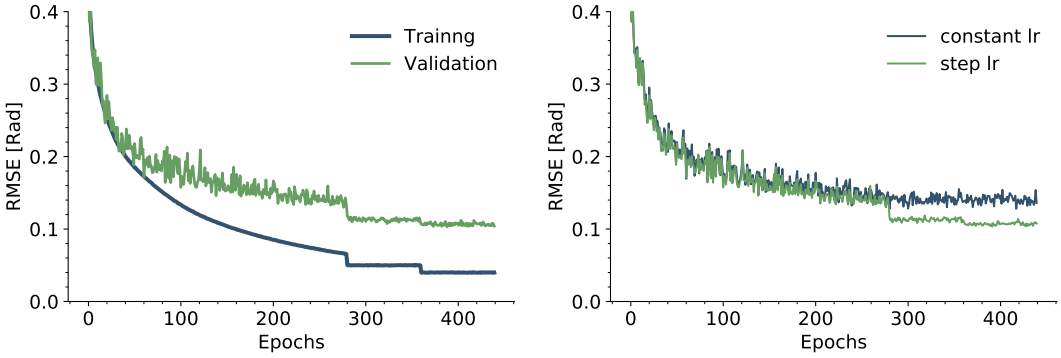



FIGURE 3.23: Resnet-50 validation and training learning curve of the RMSE over the phase map. Optimizer=SGD, momentum = 0.9, batch size=128

The Resnet-50 architecture was trained over 450 epochs using the SGD optimizer with momentum=0.9 (Nesterov). The weights have been initialized from pretrained values on Imagenet. The batch size has been fixed to 64 and the learning rate has be initially set to $10^{-3}$. After the 280 first iterations, the learning rate is lowered to $10^{-4}$ and 80 iterations later again divided by 10. This update technique is known as step learning rate. [23] At start the learning rate is high allowing a fast training of the network. Then the network reaches a point where the optimization landscape requires smaller steps to converge. By reducing progressively the learning rate, we therefore improve the final performance and ensure the convergence to, at least, a local optimum.

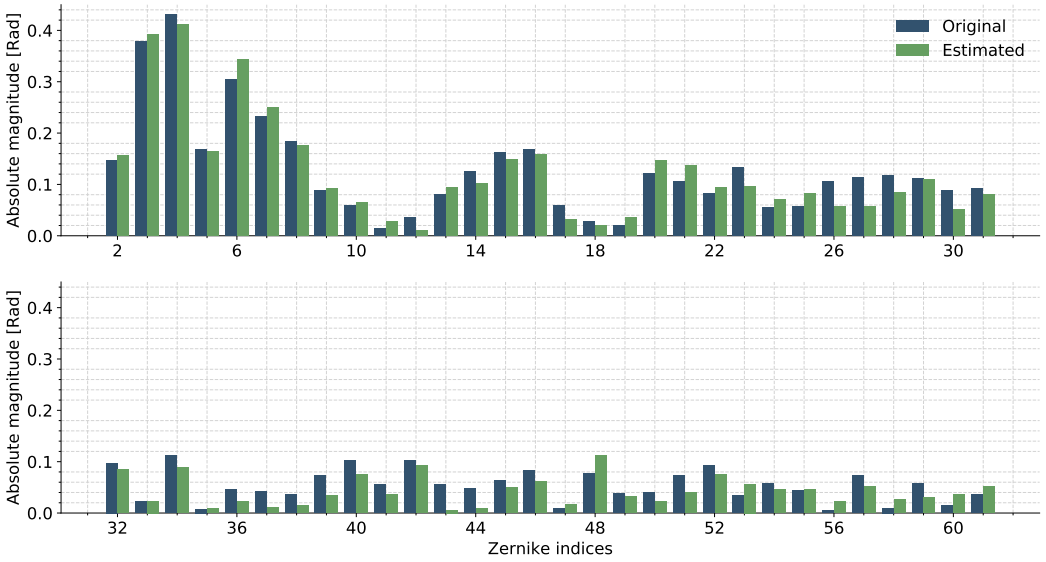



FIGURE 3.24: Comparison between the exact and the estimated Zernike coefficients from in-focus and out-of-focus PSF using Resnet-50.

The estimation of the Zernike coefficients is much more difficult with this second data set. Nonetheless, the order of magnitude of the original ones is well recovered. We also notice that the first coefficients are better estimated than the latest. The larger magnitude of the first coefficients ($1/f^2$ law) increases their visible effects on the PSFs. By consequences, the network recognizes more easily the low order modes.

| | RMSE | Unit | RMSE | Unit |
|---|---|---|---|---|
| Coefficients | $0.1978 \pm 0.0387$ | $[10^{-3}Rad]$ | $0.0665 \pm 0.0105$ | $[nm]$ |
| Phase | $0.114 \pm 0.0133$ | $[Rad]$ | $39.916 \pm 4.689$ | $[nm]$ |

TABLE 3.8: Root mean squared error over the Zernike coefficients and the phase map for the Resnet architecture.

Note that as the number of Zernike coefficients has been increased from 20 to 100, the RMSE metric over the Zernike coefficients is not comparable to the previous data set. The higher order modes have smaller norm, they thus lower the error value while the coefficients are on average less accurately estimated than in the first data set. The relevant comparison metric is the RMSE over the phase map.

#### 3.3.2.2 Unet

The Unet network was trained over 300 epochs using the SGD optimizer with momentum=0.9. Other optimizers such as Adam have also been tested and yield similar results. The network weights were initialized using Xavier initialization. The batch size has been fixed to 64 and the learning rate has be set to $10^{-3}$. As previously mentioned, the network is directly trained with respect to the pixel-wise RMSE between the exact and the estimated phase.

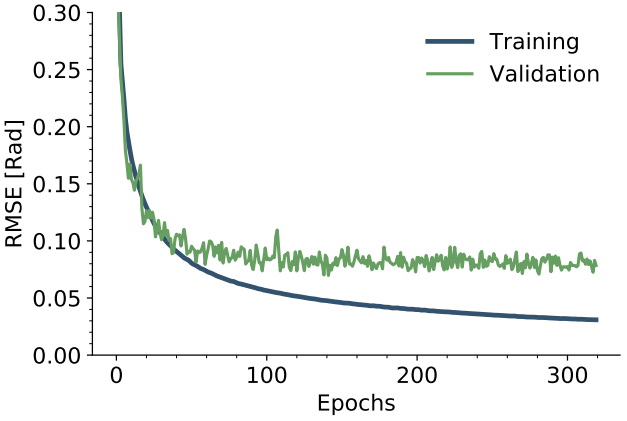



FIGURE 3.25: Unet validation and training learning curve of the RMSE over the phase map. Optimizer=SGD, momentum = 0.9, Learning rate = $10^{-3}$ , batch size=64

Once again, the direct estimation of the phase map (Unet) outperformed the modal basis reconstruction (Resnet). The projection of the phase map on the Zernike basis allows the estimation of the coefficients without issues.

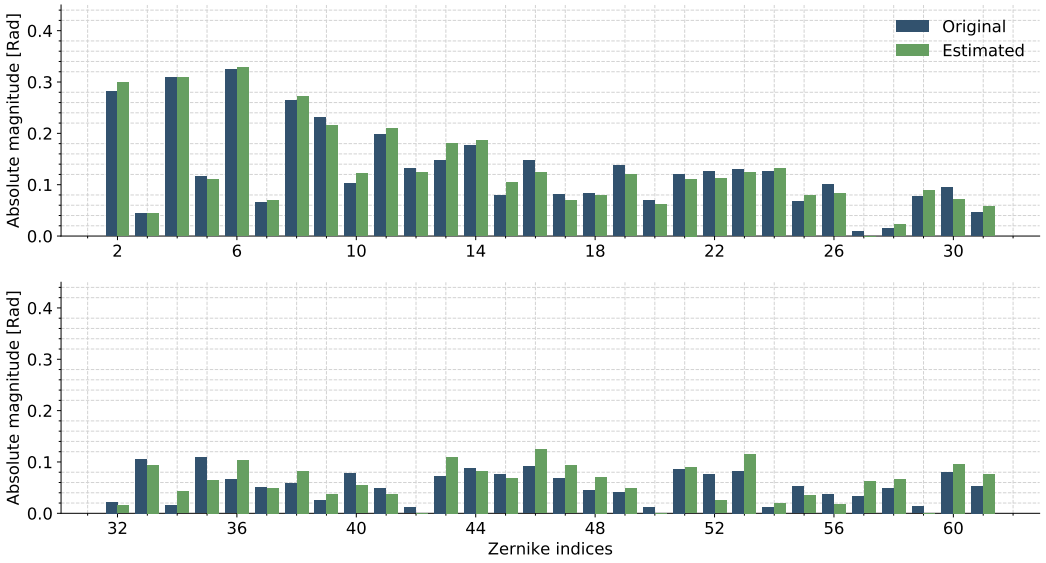



FIGURE 3.26: Comparison between the exact and the estimated Zernike coefficients from in-focus and out-of-focus PSF using Unet.

As the results between Unet and Unet++ are very close and the training method is similar, the final accuracy is expressed together.

| Unet | RMSE | Unit | RMSE | Unit |
|---|---|---|---|---|
| Coefficients | $0.1123 \pm 0.0210$ | $[10^{-3}Rad]$ | $0.0350 \pm 0.0077$ | $[nm]$ |
| Phase | $0.097 \pm 0.0109$ | $[Rad]$ | $33.963 \pm 3.816$ | $[nm]$ |
| Unet++ | RMSE | Unit | RMSE | Unit |
| Coefficients | $0.1091 \pm 0.0223$ | $[10^{-3}Rad]$ | $0.0382 \pm 0.0078$ | $[nm]$ |
| Phase | $0.094 \pm 0.0133$ | $[Rad]$ | $32.910 \pm 4.551$ | $[nm]$ |

TABLE 3.9: Root mean squared error over the Zernike coefficients and the phase map for the Unet and the Unet++ architecture.

Both architectures are essentially deep encoder-decoder networks. Unet++ simply differs by the introduction of a series of nested, dense skip pathways between the encoder and the decoder. The number of filters is also identical in both networks ($[64, 128, 256, 512]$). This similarity explain the close results. The slightly improvement of Unet++ is mainly due to two factors: 1) the skip connections which improves the gradient flow [31] 2) the higher number of trainable parameters, see Table 3.10. The

number of parameters and especially the memory consumption of the nested architecture (Unet++) is much more important. There is thus a trade-off between the accuracy gain and the training time/model size.

| Architecture | Nbr of parameters | Size (MB) | Inference time (s) |
|---|---|---|---|
| Unet | $27,395,265$ | 484.22 | $0.2793s$ |
| Unet++ | $34,829,505$ | 2444.22 | $0.413s$ |

TABLE 3.10: Average values of the network characteristics for a single input (batch size = 1). The inference time (forward pass) is calculated on CPU, Intel Xeon e3-1230v5.

### 3.3.3 Comparison

This extended set of data first demonstrated that convolutional neural networks can be trained to estimate a large number of Zernike coefficients. This extension has however a non negligible cost, it reduces significantly the network performances. In both data set, the direct pixel-wise wavefront estimation (Unet) exhibited increased performances. Finally, it can be noticed that the Inception architecture performs slightly better than the Resnet architecture on the second data set (100 Zernike) while the opposite behaviour is observed on the first data set. (20 Zernike) The exact reason is still unclear and further investigations may be conducted.

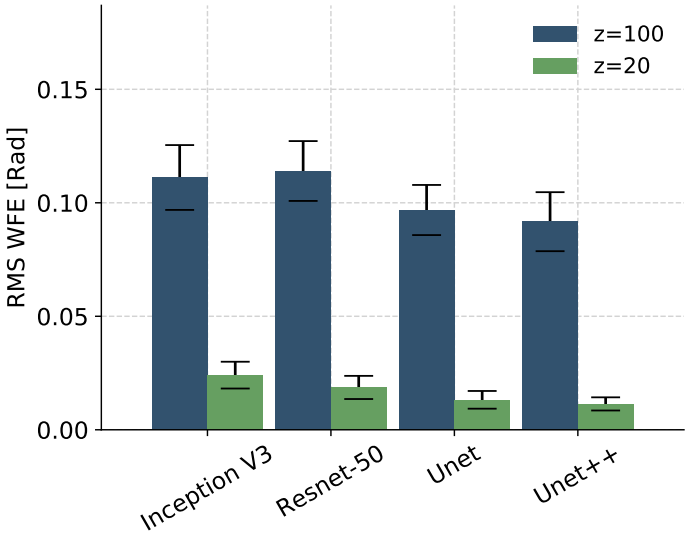



FIGURE 3.27: RMS wavefront error between the exact and the estimated phase map for the different architectures explored.

### Example

To illustrate the network performances, let us consider a randomly sampled in-focus and out-of-focus PSFs. It constitutes the input of our neural network.

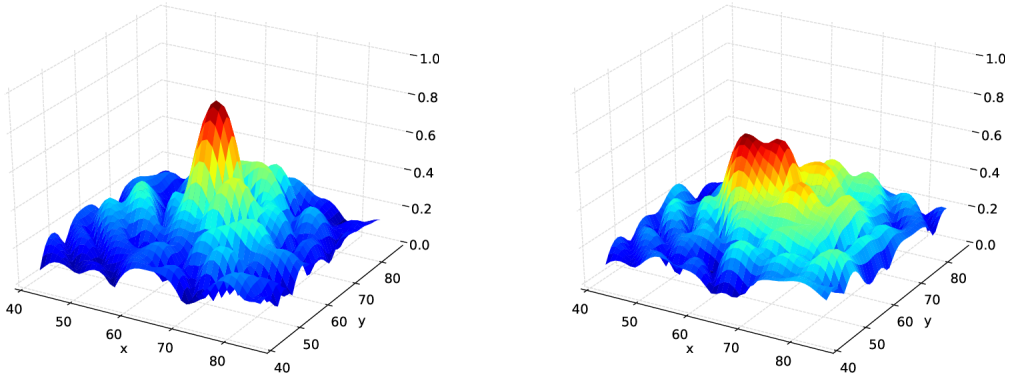



FIGURE 3.28: **Left:** In-focus PSF (Strehl=0.63) **Right:** Out-of-focus PSF (Strehl=0.44) randomly drawn from the validation set with 100 Zernike modes.

From these inputs, we compare the estimated phase map $\theta'$ recovered by Unet with the exact phase map $\theta$.

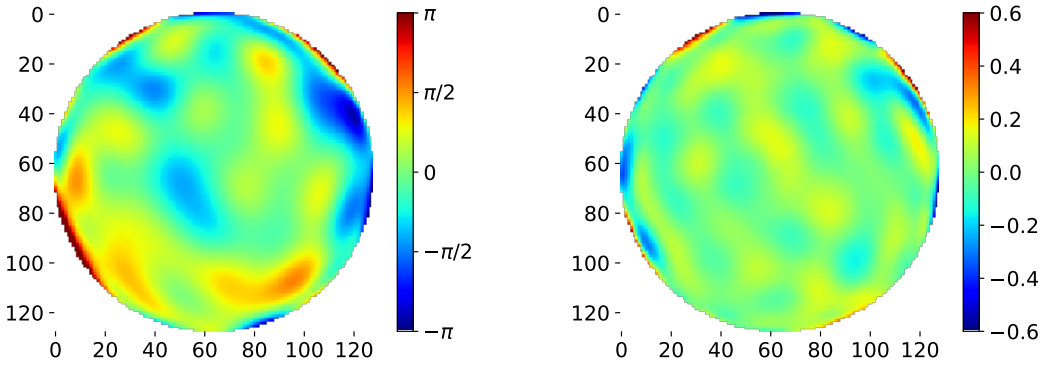



FIGURE 3.29: **Left:** Estimated phase map $\theta'$ **Right:** Residual phase map: $(\theta - \theta')$ obtained with Unet.

From the residual phase map (Figure 3.36), it is noticed that the regions at the edges of the map constitute the main source of error. It indeed corresponds to the higher order modes which are more likely to be mises-timated. Overall, as shown in the Figure 3.30, the point spread function after correction is close to the diffraction limit. (i.e. generated from the residual phase)

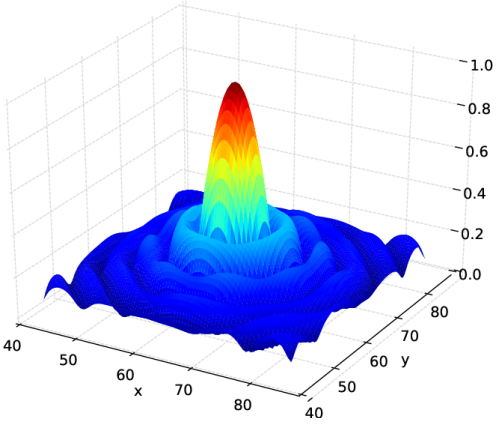



FIGURE 3.30: Recovered PSF (Strehl 0.963) after correction with a perfect AO system.

### 3.3.4 Influence of the training size

The number of training samples also affects the final accuracy. Figure 3.31 clearly shows that a larger training set improves the final network performances. Nevertheless, after $100,000$ examples, increasing furthermore the training size results in minimal accuracy gain while requiring significantly longer training time. There is thus a trade-off between the final performance and the training time. Finally, note that the network inference time (i.e. time required to perform a prediction) is independent of the training size and always requires the same time. It indeed solely relies on the number of parameters and the network architecture.

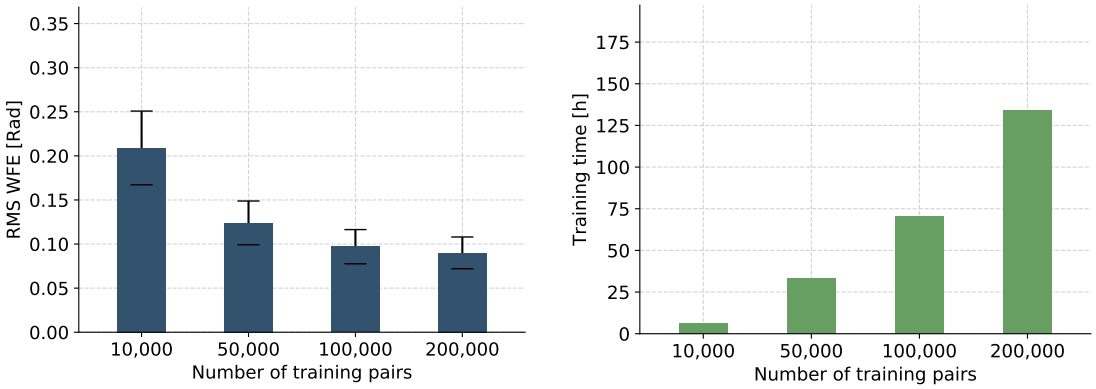



FIGURE 3.31: Unet metrics as function of the training data set size. **Left:** RMS wavefront error between the exact and the estimated phase map (100 Zernike modes) **Right:** Training time for 400 epochs in hours.

### 3.3.5 Influence of the phase diversity

Phase diversity has been shown to be particularly useful for conventional iterative algorithms but what about neural networks? To asses the impact of the phase diversity we have trained Unet with only in-focus PSFs then with only out-of-focus PSFs and finally with both as in the previous configurations.

First, Figure 3.32 clearly shows that the simultaneous use of in-focus and out-of-focus images improves significantly the final accuracy. We also notice that when working with a single intensity measurement, it is preferable to use out-of-focus images rather than in-focus images. This joins the observations of Y. Nishizaki et al. [26] which showed that the use of a defocus preconditioner may be exploited to improve the final accuracy of convolutional networks. It indeed increases the number of informative pixels and highlights the PSF distortions. Finally, unlike iterative algorithms (GS, HIO, ...) neural networks do not seem to suffer from the twin-images problem even from single intensity measurements.

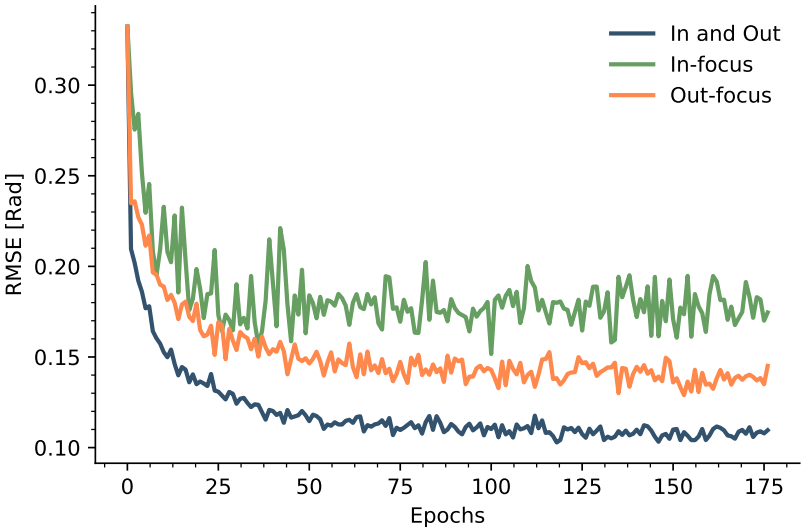



FIGURE 3.32: Training curve of Unet with 100,000 in-focus PSFs, out-of-focus PSFs and both simultaneously. Optimizer=SGD, momentum = 0.9, Learning rate = $10^{-3}$, batch size=64.

### 3.3.6 Robustness

The network performances may also be evaluated for different ranges of simulated wavefront. First, let us compare a network trained with RMS wavefront errors extending from 0.71 to 1.28 [Rad]. This is exactly the results of the previous sections, the network is simply evaluated this time with aberrations of lower and higher magnitudes than encountered in the training set. Then, the same network (Unet) is trained with PSFs generated from phase maps extending from 0.23 to 1.95 [Rad]. The data set has exactly the same size in both configurations: 100,000 in-focus and out-of-focus PSF pairs.

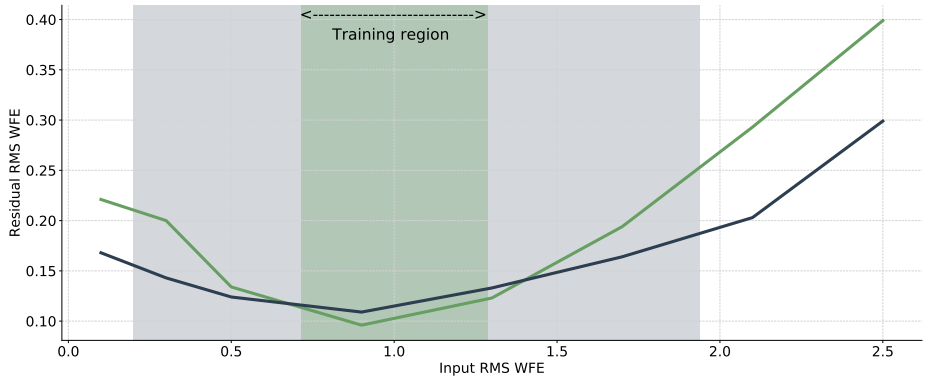



FIGURE 3.33: Residual RMS wavefront error as function of the input RMS wavefront error. The green curve and green area correspond to the first training region (0.71 to 1.28 Rad) while the blue curve and the blue area correspond to the extended training region (0.23 to 1.95 Rad). The network architecture is Unet and the training data set contains 100,000 images for each region.

From Figure 3.33, we first notice that the error rapidly increases when exiting the training area. The figure also shows that extending the training area improves the network accuracy on a wider RMS range while preserving a relatively close accuracy.

### 3.3.7 Improvements

As previously demonstrated, training the same network on a wider training region does not improve the final accuracy of the network. Moreover, it does not enable the same network to correct simultaneously high and small wavefront errors. In this section, we will therefore train a second network to correct the predictions of the first network and achieve in fine better accuracy. The second network is trained from the 100, 000 corrected in-focus and out-of-focus point spread functions of the first network (100 Zernike modes).

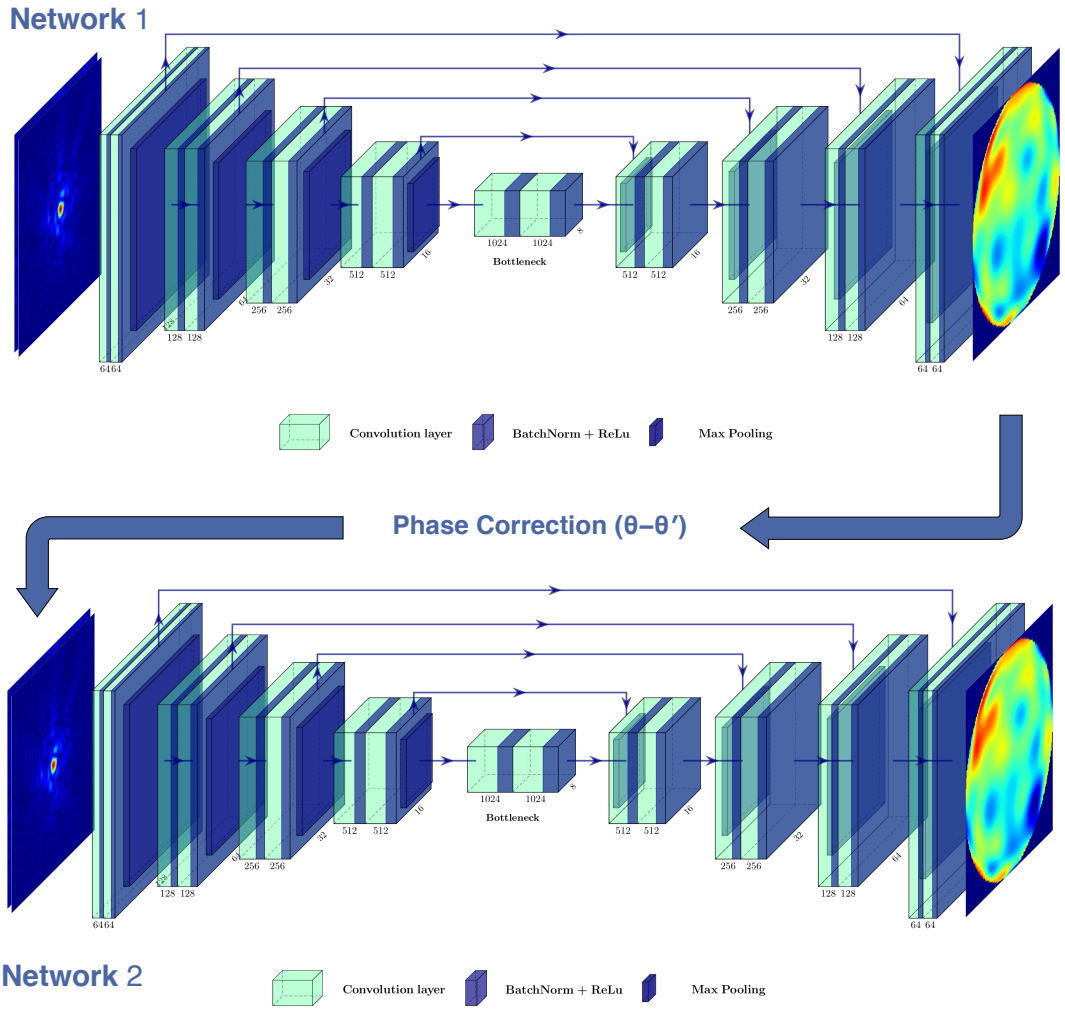



FIGURE 3.34: Illustration of the iterative application of the two neural networks. The input of the second neural network are the corrected in-focus and out-of-focus PSFs by the first network.

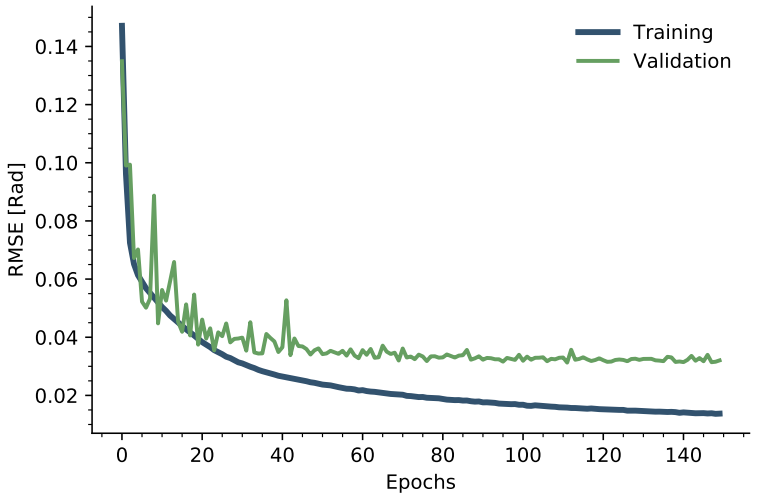



FIGURE 3.35: Second Unet network validation and training learning curve of the RMSE over the phase map. Optimizer=SGD, momentum = 0.9, Learning rate = $10^{-5}$, batch size=128

Both network architectures are identical (Unet) and are trained using similar procedure. The only difference is the preprocessing of the second network input PSFs. As the aberrations have already been partially corrected, the distortions are barely visible, a logarithmic stretching is therefore preferred to the square root stretching. The Figure 3.35 illustrates that the training of a second network may be used to improve the prediction. Both networks correct very different range of wavefront errors: $\approx 1 Rad$ for the first network and $0.1 Rad$ for the second network. By combining both networks, a final accuracy of $0.0334 \pm 0.00451$ Rad RMSE is reached. This is close to the correction level achieved with a single network on 20 Zernike coefficients.

As example, let us consider a randomly selected phase map and let us evaluate the residual phase map after the first and the second network corrections.

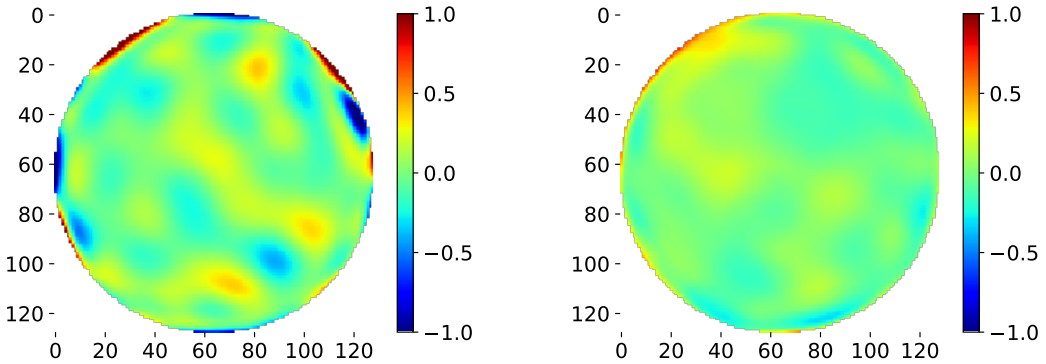



FIGURE 3.36: Residual phase map: $(\theta - \theta')$ **Left:** First network **Right:** First and second network.

# Chapter 4

# Conclusion

In this thesis, a new approach to wavefront sensing based on deep convolutional neural networks have been proposed. Two procedures have been considered. First, the phase map has been projected onto an orthogonal basis, the Zernike basis and deep neural networks have been trained to estimate the coefficients of the phase expansion. (VGG, Inception v3, Resnet) Secondly, instead of estimating the Zernike coefficients and then reconstructing the phase map, neural networks (Unet, Unet++) have been trained to directly provide a pixel-wise estimation of the phase map.

Two distinct data sets have been considered. The first one encompasses the first 20 Zernike modes while the second has been extended up to 100 Zernike modes. In both configurations, the direct phase map estimation has been shown to outperform the modal phase reconstruction. Secondly, it is highlighted that the extension to a large number of Zernike modes reduces significantly the network performances. Nonetheless, it is shown that a second neural network can be trained to correct the predictions and to reach the same level of performances than obtained with 20 Zernike modes. Finally, phase diversity has been shown to improve the CNN final accuracy without being a necessary requirement.

Finally, future works may consider different levels of optical aberrations, different noise realizations or even extended sources. Neural networks may also be trained using experimental data rather than simulated data.

# Appendices

# Appendix A

# VGG-16 architecture

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv2d-1 | [-1, 3, 128, 128] | 9 |
| AdaptiveMaxPool2d-2 | [-1, 3, 256, 256] | 0 |
| Conv2d-3 | [-1, 64, 256, 256] | 1,792 |
| BatchNorm2d-4 | [-1, 64, 256, 256] | 128 |
| ReLU-5 | [-1, 64, 256, 256] | 0 |
| Conv2d-6 | [-1, 64, 256, 256] | 36,928 |
| BatchNorm2d-7 | [-1, 64, 256, 256] | 128 |
| ReLU-8 | [-1, 64, 256, 256] | 0 |
| MaxPool2d-9 | [-1, 64, 128, 128] | 0 |
| Conv2d-10 | [-1, 128, 128, 128] | 73,856 |
| BatchNorm2d-11 | [-1, 128, 128, 128] | 256 |
| ReLU-12 | [-1, 128, 128, 128] | 0 |
| Conv2d-13 | [-1, 128, 128, 128] | 147,584 |
| BatchNorm2d-14 | [-1, 128, 128, 128] | 256 |
| ReLU-15 | [-1, 128, 128, 128] | 0 |
| MaxPool2d-16 | [-1, 128, 64, 64] | 0 |
| Conv2d-17 | [-1, 256, 64, 64] | 295,168 |
| BatchNorm2d-18 | [-1, 256, 64, 64] | 512 |
| ReLU-19 | [-1, 256, 64, 64] | 0 |
| Conv2d-20 | [-1, 256, 64, 64] | 590,080 |
| BatchNorm2d-21 | [-1, 256, 64, 64] | 512 |
| ReLU-22 | [-1, 256, 64, 64] | 0 |
| Conv2d-23 | [-1, 256, 64, 64] | 590,080 |
| BatchNorm2d-24 | [-1, 256, 64, 64] | 512 |
| ReLU-25 | [-1, 256, 64, 64] | 0 |
| MaxPool2d-26 | [-1, 256, 32, 32] | 0 |
| Conv2d-27 | [-1, 512, 32, 32] | 1,180,160 |
| BatchNorm2d-28 | [-1, 512, 32, 32] | 1,024 |
| ReLU-29 | [-1, 512, 32, 32] | 0 |
| Conv2d-30 | [-1, 512, 32, 32] | 2,359,808 |
| BatchNorm2d-31 | [-1, 512, 32, 32] | 1,024 |
| ReLU-32 | [-1, 512, 32, 32] | 0 |
| Conv2d-33 | [-1, 512, 32, 32] | 2,359,808 |
| BatchNorm2d-34 | [-1, 512, 32, 32] | 1,024 |
| ReLU-35 | [-1, 512, 32, 32] | 0 |
| MaxPool2d-36 | [-1, 512, 16, 16] | 0 |

```
           Conv2d-37          [-1, 512, 16, 16]       2,359,808
      BatchNorm2d-38          [-1, 512, 16, 16]           1,024
             ReLU-39          [-1, 512, 16, 16]               0
           Conv2d-40          [-1, 512, 16, 16]       2,359,808
      BatchNorm2d-41          [-1, 512, 16, 16]           1,024
             ReLU-42          [-1, 512, 16, 16]               0
           Conv2d-43          [-1, 512, 16, 16]       2,359,808
      BatchNorm2d-44          [-1, 512, 16, 16]           1,024
             ReLU-45          [-1, 512, 16, 16]               0
        MaxPool2d-46            [-1, 512, 8, 8]               0
AdaptiveAvgPool2d-47            [-1, 512, 7, 7]               0
           Linear-48                 [-1, 4096]      34,764,544
             ReLU-49                 [-1, 4096]               0
      BatchNorm1d-50                 [-1, 4096]           8,192
           Linear-51                 [-1, 1024]       4,195,328
             ReLU-52                 [-1, 1024]               0
      BatchNorm1d-53                 [-1, 1024]           2,048
           Linear-54                   [-1, 20]          20,500
================================================================
Total params: 121,713,757
Trainable params: 51,713,757
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.12
Forward/backward pass size (MB): 422.43
Params size (MB): 464.30
Estimated Total Size (MB): 886.86
----------------------------------------------------------------
```

# Appendix B

# Inception-v3 architecture

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv2d-1 | [-1, 3, 128, 128] | 9 |
| AdaptiveMaxPool2d-2 | [-1, 3, 299, 299] | 0 |
| Conv2d-3 | [-1, 32, 149, 149] | 864 |
| BatchNorm2d-4 | [-1, 32, 149, 149] | 64 |
| BasicConv2d-5 | [-1, 32, 149, 149] | 0 |
| Conv2d-6 | [-1, 32, 147, 147] | 9,216 |
| BatchNorm2d-7 | [-1, 32, 147, 147] | 64 |
| BasicConv2d-8 | [-1, 32, 147, 147] | 0 |
| Conv2d-9 | [-1, 64, 147, 147] | 18,432 |
| BatchNorm2d-10 | [-1, 64, 147, 147] | 128 |
| BasicConv2d-11 | [-1, 64, 147, 147] | 0 |
| Conv2d-12 | [-1, 80, 73, 73] | 5,120 |
| BatchNorm2d-13 | [-1, 80, 73, 73] | 160 |
| BasicConv2d-14 | [-1, 80, 73, 73] | 0 |
| Conv2d-15 | [-1, 192, 71, 71] | 138,240 |
| BatchNorm2d-16 | [-1, 192, 71, 71] | 384 |
| BasicConv2d-17 | [-1, 192, 71, 71] | 0 |
| Conv2d-18 | [-1, 64, 35, 35] | 12,288 |
| BatchNorm2d-19 | [-1, 64, 35, 35] | 128 |
| BasicConv2d-20 | [-1, 64, 35, 35] | 0 |
| Conv2d-21 | [-1, 48, 35, 35] | 9,216 |
| BatchNorm2d-22 | [-1, 48, 35, 35] | 96 |
| BasicConv2d-23 | [-1, 48, 35, 35] | 0 |
| Conv2d-24 | [-1, 64, 35, 35] | 76,800 |
| BatchNorm2d-25 | [-1, 64, 35, 35] | 128 |
| BasicConv2d-26 | [-1, 64, 35, 35] | 0 |
| Conv2d-27 | [-1, 64, 35, 35] | 12,288 |
| BatchNorm2d-28 | [-1, 64, 35, 35] | 128 |
| BasicConv2d-29 | [-1, 64, 35, 35] | 0 |
| Conv2d-30 | [-1, 96, 35, 35] | 55,296 |
| BatchNorm2d-31 | [-1, 96, 35, 35] | 192 |
| BasicConv2d-32 | [-1, 96, 35, 35] | 0 |
| Conv2d-33 | [-1, 96, 35, 35] | 82,944 |
| BatchNorm2d-34 | [-1, 96, 35, 35] | 192 |
| BasicConv2d-35 | [-1, 96, 35, 35] | 0 |
| Conv2d-36 | [-1, 32, 35, 35] | 6,144 |

```
      BatchNorm2d-37           [-1, 32, 35, 35]              64
      BasicConv2d-38           [-1, 32, 35, 35]               0
       InceptionA-39          [-1, 256, 35, 35]               0
           Conv2d-40           [-1, 64, 35, 35]          16,384
      BatchNorm2d-41           [-1, 64, 35, 35]             128
      BasicConv2d-42           [-1, 64, 35, 35]               0
             .                        .
             .                        .
             .                        .
           Conv2d-279            [-1, 384, 8, 8]         786,432
      BatchNorm2d-280            [-1, 384, 8, 8]             768
      BasicConv2d-281            [-1, 384, 8, 8]               0
           Conv2d-282            [-1, 384, 8, 8]         442,368
      BatchNorm2d-283            [-1, 384, 8, 8]             768
      BasicConv2d-284            [-1, 384, 8, 8]               0
           Conv2d-285            [-1, 384, 8, 8]         442,368
      BatchNorm2d-286            [-1, 384, 8, 8]             768
      BasicConv2d-287            [-1, 384, 8, 8]               0
           Conv2d-288            [-1, 448, 8, 8]         917,504
      BatchNorm2d-289            [-1, 448, 8, 8]             896
      BasicConv2d-290            [-1, 448, 8, 8]               0
           Conv2d-291            [-1, 384, 8, 8]       1,548,288
      BatchNorm2d-292            [-1, 384, 8, 8]             768
      BasicConv2d-293            [-1, 384, 8, 8]               0
           Conv2d-294            [-1, 384, 8, 8]         442,368
      BatchNorm2d-295            [-1, 384, 8, 8]             768
      BasicConv2d-296            [-1, 384, 8, 8]               0
           Conv2d-297            [-1, 384, 8, 8]         442,368
      BatchNorm2d-298            [-1, 384, 8, 8]             768
      BasicConv2d-299            [-1, 384, 8, 8]               0
           Conv2d-300            [-1, 192, 8, 8]         393,216
      BatchNorm2d-301            [-1, 192, 8, 8]             384
      BasicConv2d-302            [-1, 192, 8, 8]               0
       InceptionE-303           [-1, 2048, 8, 8]               0
           Linear-304                  [-1, 20]          40,980
       Inception3-305     [[-1, 20], [-1, 1000]]               0
     Phase2DLayer-306            [-1, 128, 128]               0
================================================================
Total params: 25,153,253
Trainable params: 25,153,253
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.12
Forward/backward pass size (MB): 226.62
Params size (MB): 95.95
Estimated Total Size (MB): 322.69
----------------------------------------------------------------
```

# Appendix C

# Resnet-50 architecture

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv2d-1 | [-1, 3, 128, 128] | 9 |
| AdaptiveMaxPool2d-2 | [-1, 3, 224, 224] | 0 |
| Conv2d-3 | [-1, 64, 112, 112] | 9,408 |
| BatchNorm2d-4 | [-1, 64, 112, 112] | 128 |
| ReLU-5 | [-1, 64, 112, 112] | 0 |
| MaxPool2d-6 | [-1, 64, 56, 56] | 0 |
| Conv2d-7 | [-1, 64, 56, 56] | 4,096 |
| BatchNorm2d-8 | [-1, 64, 56, 56] | 128 |
| ReLU-9 | [-1, 64, 56, 56] | 0 |
| Conv2d-10 | [-1, 64, 56, 56] | 36,864 |
| BatchNorm2d-11 | [-1, 64, 56, 56] | 128 |
| ReLU-12 | [-1, 64, 56, 56] | 0 |
| Conv2d-13 | [-1, 256, 56, 56] | 16,384 |
| BatchNorm2d-14 | [-1, 256, 56, 56] | 512 |
| Conv2d-15 | [-1, 256, 56, 56] | 16,384 |
| BatchNorm2d-16 | [-1, 256, 56, 56] | 512 |
| ReLU-17 | [-1, 256, 56, 56] | 0 |
| Bottleneck-18 | [-1, 256, 56, 56] | 0 |
| Conv2d-19 | [-1, 64, 56, 56] | 16,384 |
| BatchNorm2d-20 | [-1, 64, 56, 56] | 128 |
| ReLU-21 | [-1, 64, 56, 56] | 0 |
| Conv2d-22 | [-1, 64, 56, 56] | 36,864 |
| BatchNorm2d-23 | [-1, 64, 56, 56] | 128 |
| ReLU-24 | [-1, 64, 56, 56] | 0 |
| Conv2d-25 | [-1, 256, 56, 56] | 16,384 |
| BatchNorm2d-26 | [-1, 256, 56, 56] | 512 |
| ReLU-27 | [-1, 256, 56, 56] | 0 |
| Bottleneck-28 | [-1, 256, 56, 56] | 0 |
| Conv2d-29 | [-1, 64, 56, 56] | 16,384 |
| BatchNorm2d-30 | [-1, 64, 56, 56] | 128 |
| ReLU-31 | [-1, 64, 56, 56] | 0 |
| Conv2d-32 | [-1, 64, 56, 56] | 36,864 |
| BatchNorm2d-33 | [-1, 64, 56, 56] | 128 |
| ReLU-34 | [-1, 64, 56, 56] | 0 |
| Conv2d-35 | [-1, 256, 56, 56] | 16,384 |
| BatchNorm2d-36 | [-1, 256, 56, 56] | 512 |

```
              ReLU-37          [-1, 256, 56, 56]               0
        Bottleneck-38          [-1, 256, 56, 56]               0
            Conv2d-39          [-1, 128, 56, 56]          32,768
       BatchNorm2d-40          [-1, 128, 56, 56]             256
              ReLU-41          [-1, 128, 56, 56]               0
            Conv2d-42          [-1, 128, 28, 28]         147,456
       BatchNorm2d-43          [-1, 128, 28, 28]             256
              ReLU-44          [-1, 128, 28, 28]               0
            Conv2d-45          [-1, 512, 28, 28]          65,536
       BatchNorm2d-46          [-1, 512, 28, 28]           1,024
            Conv2d-47          [-1, 512, 28, 28]         131,072
       BatchNorm2d-48          [-1, 512, 28, 28]           1,024
              ReLU-49          [-1, 512, 28, 28]               0
        Bottleneck-50          [-1, 512, 28, 28]               0
             .                         .
             .                         .
             .                         .
              ReLU-160           [-1, 512, 7, 7]               0
            Conv2d-161          [-1, 2048, 7, 7]       1,048,576
       BatchNorm2d-162          [-1, 2048, 7, 7]           4,096
              ReLU-163          [-1, 2048, 7, 7]               0
        Bottleneck-164          [-1, 2048, 7, 7]               0
            Conv2d-165           [-1, 512, 7, 7]       1,048,576
       BatchNorm2d-166           [-1, 512, 7, 7]           1,024
              ReLU-167           [-1, 512, 7, 7]               0
            Conv2d-168           [-1, 512, 7, 7]       2,359,296
       BatchNorm2d-169           [-1, 512, 7, 7]           1,024
              ReLU-170           [-1, 512, 7, 7]               0
            Conv2d-171          [-1, 2048, 7, 7]       1,048,576
       BatchNorm2d-172          [-1, 2048, 7, 7]           4,096
              ReLU-173          [-1, 2048, 7, 7]               0
        Bottleneck-174          [-1, 2048, 7, 7]               0
AdaptiveAvgPool2d-175          [-1, 2048, 1, 1]               0
            Linear-176                [-1, 100]         204,900
            ResNet-177                [-1, 100]               0
      Phase2DLayer-178           [-1, 128, 128]               0
================================================================
Total params: 23,712,941
Trainable params: 23,712,941
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.12
Forward/backward pass size (MB): 288.20
Params size (MB): 90.46
Estimated Total Size (MB): 378.78
----------------------------------------------------------------
```

# Appendix D

# Unet architecture

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv2d-1 | [-1, 64, 128, 128] | 1,216 |
| BatchNorm2d-2 | [-1, 64, 128, 128] | 128 |
| ReLU-3 | [-1, 64, 128, 128] | 0 |
| Conv2d-4 | [-1, 64, 128, 128] | 76,928 |
| BatchNorm2d-5 | [-1, 64, 128, 128] | 128 |
| ReLU-6 | [-1, 64, 128, 128] | 0 |
| double_conv-7 | [-1, 64, 128, 128] | 0 |
| inconv-8 | [-1, 64, 128, 128] | 0 |
| MaxPool2d-9 | [-1, 64, 64, 64] | 0 |
| Conv2d-10 | [-1, 128, 64, 64] | 143,856 |
| BatchNorm2d-11 | [-1, 128, 64, 64] | 256 |
| ReLU-12 | [-1, 128, 64, 64] | 0 |
| Conv2d-13 | [-1, 128, 64, 64] | 347,584 |
| BatchNorm2d-14 | [-1, 128, 64, 64] | 256 |
| ReLU-15 | [-1, 128, 64, 64] | 0 |
| double_conv-16 | [-1, 128, 64, 64] | 0 |
| down-17 | [-1, 128, 64, 64] | 0 |
| MaxPool2d-18 | [-1, 128, 32, 32] | 0 |
| Conv2d-19 | [-1, 256, 32, 32] | 595,168 |
| BatchNorm2d-20 | [-1, 256, 32, 32] | 512 |
| ReLU-21 | [-1, 256, 32, 32] | 0 |
| Conv2d-22 | [-1, 256, 32, 32] | 1,090,080 |
| BatchNorm2d-23 | [-1, 256, 32, 32] | 512 |
| ReLU-24 | [-1, 256, 32, 32] | 0 |
| double_conv-25 | [-1, 256, 32, 32] | 0 |
| down-26 | [-1, 256, 32, 32] | 0 |
| MaxPool2d-27 | [-1, 256, 16, 16] | 0 |
| Conv2d-28 | [-1, 512, 16, 16] | 2,180,160 |
| BatchNorm2d-29 | [-1, 512, 16, 16] | 1,024 |
| ReLU-30 | [-1, 512, 16, 16] | 0 |
| Conv2d-31 | [-1, 512, 16, 16] | 4,359,808 |
| BatchNorm2d-32 | [-1, 512, 16, 16] | 1,024 |
| ReLU-33 | [-1, 512, 16, 16] | 0 |
| double_conv-34 | [-1, 512, 16, 16] | 0 |
| down-35 | [-1, 512, 16, 16] | 0 |
| MaxPool2d-36 | [-1, 512, 8, 8] | 0 |
| Conv2d-37 | [-1, 512, 8, 8] | 4,359,808 |
| BatchNorm2d-38 | [-1, 512, 8, 8] | 1,024 |

```
              ReLU-39            [-1, 512, 8, 8]               0
            Conv2d-40            [-1, 512, 8, 8]       4,359,808
       BatchNorm2d-41            [-1, 512, 8, 8]           1,024
              ReLU-42            [-1, 512, 8, 8]               0
       double_conv-43            [-1, 512, 8, 8]               0
              down-44            [-1, 512, 8, 8]               0
          Upsample-45          [-1, 512, 16, 16]               0
            Conv2d-46          [-1, 256, 16, 16]       4,359,552
       BatchNorm2d-47          [-1, 256, 16, 16]             512
              ReLU-48          [-1, 256, 16, 16]               0
            Conv2d-49          [-1, 256, 16, 16]       1,090,080
       BatchNorm2d-50          [-1, 256, 16, 16]             512
              ReLU-51          [-1, 256, 16, 16]               0
       double_conv-52          [-1, 256, 16, 16]               0
                 up-53          [-1, 256, 16, 16]               0
          Upsample-54          [-1, 256, 32, 32]               0
            Conv2d-55          [-1, 128, 32, 32]         589,952
       BatchNorm2d-56          [-1, 128, 32, 32]             256
              ReLU-57          [-1, 128, 32, 32]               0
            Conv2d-58          [-1, 128, 32, 32]         347,584
       BatchNorm2d-59          [-1, 128, 32, 32]             256
              ReLU-60          [-1, 128, 32, 32]               0
       double_conv-61          [-1, 128, 32, 32]               0
                 up-62          [-1, 128, 32, 32]               0
          Upsample-63          [-1, 128, 64, 64]               0
            Conv2d-64           [-1, 64, 64, 64]         347,520
       BatchNorm2d-65           [-1, 64, 64, 64]             128
              ReLU-66           [-1, 64, 64, 64]               0
            Conv2d-67           [-1, 64, 64, 64]          76,928
       BatchNorm2d-68           [-1, 64, 64, 64]             128
              ReLU-69           [-1, 64, 64, 64]               0
       double_conv-70           [-1, 64, 64, 64]               0
                 up-71           [-1, 64, 64, 64]               0
          Upsample-72         [-1, 64, 128, 128]               0
            Conv2d-73         [-1, 64, 128, 128]          73,792
       BatchNorm2d-74         [-1, 64, 128, 128]             128
              ReLU-75         [-1, 64, 128, 128]               0
            Conv2d-76         [-1, 64, 128, 128]          36,928
       BatchNorm2d-77         [-1, 64, 128, 128]             128
              ReLU-78         [-1, 64, 128, 128]               0
       double_conv-79         [-1, 64, 128, 128]               0
                 up-80         [-1, 64, 128, 128]               0
            Conv2d-81          [-1, 1, 128, 128]             577
           outconv-82          [-1, 1, 128, 128]               0
================================================================
Total params: 27,395,265
Trainable params: 27,395,265
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.12
Forward/backward pass size (MB): 433.00
Params size (MB): 51.10
Estimated Total Size (MB): 484.22
----------------------------------------------------------------
```

# Bibliography


[1] Nick Fang. Lecture notes on wave optics, June 2014.

[2] Robert J. Noll. Zernike polynomials and atmospheric turbulence. *J. Opt. Soc. Am.*, 66(3):207–211, Mar 1976.

[3] Lewis C. et al. Is that really your strehl ratio? *Proceedings: Advancements in Adaptive Optics*, Vol 5490, 2004.

[4] J W Hardy. *Adaptive Optics for Astronomical Telescope*. 01 1998.

[5] P. Hickson. Fundamentals of Atmospheric and Adaptive Optics. *The University of British Columbia*, 2008.

[6] R. W. Gerchberg and W. O. Saxton. A practical algorithm for the determination of phase from imageand diffraction plane pictures. *Optik*, Vol. 35, No.2, 1972.

[7] Robert A. Gonsalves. Phase retrieval and diversity in adaptive optics. *Optical Engineering*, 21(5):829 – 832 – 4, 1982.

[8] M. H. Maleki and A. J. Devaney. Phase-retrieval and intensity-only reconstruction algorithms for optical diffraction tomography. *J. Opt. Soc. Am. A*, 10(5):1086–1092, May 1993.

[9] J. R. Fienup. Phase retrieval algorithms: a comparison. *Appl. Opt.*, 21(15):2758–2769, Aug 1982.

[10] J. R. Fienup and C. C. Wackerman. Phase-retrieval stagnation problems and solutions. *J. Opt. Soc. Am. A*, 3(11):1897–1907, Nov 1986.

[11] Scott W. Paine and James R. Fienup. Machine learning for improved image-based wavefront sensing. *Opt. Lett.*, 43(6):1235–1238, Mar 2018.

[12] John H. Seldin and Richard Paxman. Joint estimation of amplitude and phase from phase-diversity data. 06 2005.

[13] M Hartung, A Blanc, Thierry Fusco, Francois Lacombe, Laurent Mugnier, G Rousset, and Rainer Lenzen. Calibration of naos and conica static aberrations. *A&A*, 399(1), 02 2003.

[14] W. S. McCulloch and W. H. Pitts. A logical calculus of the ideas immanent in nervous activity. 5:115–133, 1943.

[15] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[16] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989. ISSN 0893-6080.

[17] M. Lloyd-Hart J. R. P. Angel, P. Wizinowich and D. Sandler. Adaptive optics for array telescopes using neural-network techniques. *Nature*, 348:221–224, 1990.

[18] D. G. Sandler et al. Use of a neural network to control an adaptive optics system for an astronomical telescope. *Nature*, 351:300–302, 1991.

[19] Todd K. Barrett and David G. Sandler. Artificial neural network for the determination of hubble space telescope aberration from stellar images. *Appl. Opt.*, 32(10): 1720–1727, Apr 1993.

[20] Fei-Fei Li et al. Cs231n: Convolutional neural networks for visual recognition, June Spring 2019.

[21] Christian Szegedy Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[24] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

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

[26] Ryoichi Horisaki Katsuhisa Kitaguchi-Mamoru Saito Jun Tanida Yohei Nishizaki, Matias Valdivia and Esteban Vera. Deep learning wavefront sensing. *Opt. Express*, 27(1):240–251, Jan 2019.

[27] Masen Lamb, David R. Andersen, Jean-Pierre Véran, Carlos Correia, Glen Herriot, Matthias Rosensteiner, and Jason Fiege. Non-common path aberration corrections for current and future ao systems. volume 9148, page 914857, 07 2014.

[28] G. E. Healey and R. Kondepudy. Radiometric ccd camera calibration and noise estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16 (3):267–276, March 1994.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[30] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[31] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. *CoRR*, abs/1807.10165, 2018.