Master Math en Action

Internship report

Academic year 2022 - 2023

Studying circumstellar environments with deep learning for high-contrast imagery reconstruction.

Edouard Chappon

Under the supervision of Nelly Pustelnik and Julian Tachella

Oral presentation: September 2023

Preface and acknowledgements

Je tiens à remercier sincèrement mes tuteurs, Nelly et Julian, pour leur réactivité, leur aide lors de mes nombreux problèmes avec mes modèles, leurs conseils pour l'oral de l'école doctorale, leurs relectures de ce rapport et plus généralement pour ces 6 mois de stage au laboratoire de physique.

Merci à Clément Marteau pour sa bienveillance et ses conseils tout au cours de l'année ainsi que pour les lettres de recommandation.

Un immense merci à mes collègues de bureau et camarades de master pour les croissants, le soutien, et les parties d'échecs, et particulièrement à Afsin qui m'a fait progresser énormément en anglais, Jeremy qui a résolu de nombreux problèmes de code et de connexion au CBP, et Victor pour les soirées passées à discuter.

Merci également à Charlotte pour toutes ces pauses et les soirées. Merci à Juliana pour son oreille attentive et son aide, notamment lorsque j'avais certains doutes. Merci à Emmanuel qui a été patient et bienveillant à mon égard. Merci enfin à tous ceux avec qui j'ai partagé les repas, les cafés et les after-work.

Merci à mes amis, pour les soirées sur les quais, les concerts et le soutien essentiel qu'ils ont représenté.

Et bien sûr, je témoigne,
toute ma reconnaissance,
envers ma famille,
qui m'a toujours accompagné
et soutenu,
et qui continue,
de m'épauler,
même d'une autre ville,
cela n'a pas d'incidence,
sur à quel point ils me soignent.

Contents

I. Motivation

The study and direct observation of circumstellar environments are crucial for improving our understanding of exoplanets and unraveling the intricate processes underlying the formation of stellar systems. Despite remarkable strides in instrumental capabilities and signal processing, enabling higher resolution of these environments, the challenge of direct observation persists due to the very high contrast between these environments and their host stars. Indeed, the radiance of these stars can be a staggering 1,000 to 10,000 times, or even up to 10 million times, more intense than that of their environment.

Fortunately, some polarimetric properties of the light emitted by the star and reflected on its environment, make it theoretically possible to use polarimetric imaging to disentangle the intermingled light of the two sources with the double difference [19] or double ratio methods [1].
To enhance the process of capturing light and utilizing polarimetric data, the Spectro-Polarimetric High-contrast Exoplanet REsearch (SPHERE) and its instrument InfraRed Dual Imaging and Spectrograph (IRDIS) installed on the Very Large Telescope (VLT), offer the capability to acquire polarimetric images at high contrast and high resolution. While acquiring the data, various factors such as instrumental blur, noise, and other effects can lead to their corruption. The double difference and double ratio are not robust to those instrumental effects.

Thus, the objective of this internship is to reconstruct the Stokes parameters from the degraded image by using the inverse problem formalism. This type of problem is a well-studied problem, often addressed through the application of iterative methods to solve variational problems.
These variational methods have already been explored during Laurence Denneulin's Ph.D. (2017-2020) [6]. A model named RHAPSODIE is described and available in [7].

The directions we explored in this internship are based on the fact that variational methods depend on the choice of a prior which captures the set of plausible signals. The quality of the reconstruction will highly depend on this choice. Given the success of deep learning, research in signal processing has developed a wide range of methods aiming to learn certain aspects of the variational problem. This internship aimed to enhance the results from RHAPSODIE by leveraging the physics of iterative methods and the recent improvements in deep learning to learn the prior. Thus, we studied a range of models bridging these two worlds. Some models with many parameters resemble the black-box nature of deep learning, while others with few parameters are more akin to variational methods.

This report is organized as follows:

In the second part, I will explain circumstellar environments, how they are formed, how data we aim to process are collected, and why we need to treat them.

Then I discuss on the third part about various reconstruction methods. I give heuristics, algorithms, and some theory about methods, including: *i*) Variational methods, *ii*) Unfolded variational methods, *iii*) Plug and Play. Moving on to the fourth part, numerical experiments are conducted using two datasets. Explanations of datasets and comparisons are done with: *i*) Synthetic data *ii*) Pre-reconstructed data. Finaly the fifth part, is devoted to conclusions with a discussion on future perspectives.

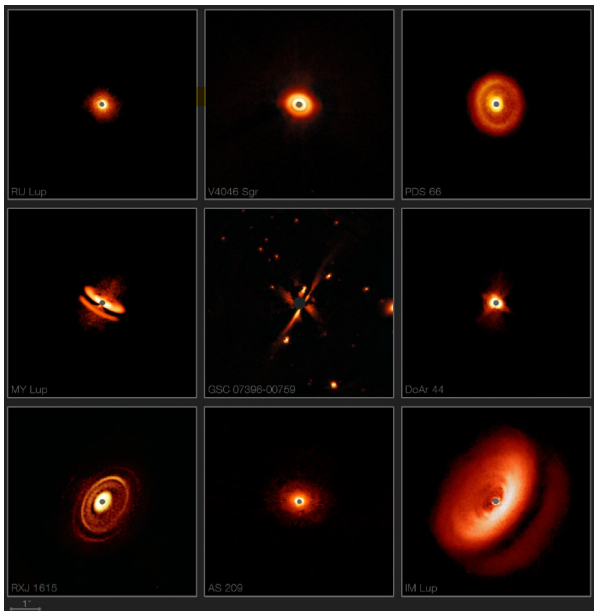


Figure 1: Circumstellar disks observed with SPHERE. We can observe the diversity of shapes and orientations of the disks.

II. Circumstellar environments

Formation:

Circumstellar environments encompass the regions surrounding stars, comprising residual gas, dust, and other materials originating from star formation. These environments exhibit remarkable diversity, as depicted in Figure 1, and serve as pivotal sites for the genesis of exoplanets, asteroids, and comets. Exploring these environments enables researchers to validate formation hypotheses for these disks against existing models. Additionally, it facilitates the pursuit of exoplanet research and characterization, which stands as a paramount domain within modern astronomy.

The variety of those structures is explained by the life cycle of a circumstellar environment:
- The birth of a star begins with the collapse of a molecular cloud. A molecular cloud is a large cloud of gas and dust that contains millions of times the mass of our sun. The shape of this cloud is a disk and it is called **protoplaneraty disk**.
- The protoplanetary disk's dust particles gradually come together to create rocky structures on the order of kilometers in size. The structure containing the star and the aggregated dust is called **transition disk**.
- Biggest asteroïds clean their orbit and they grow and attract more matter. At this stage, the dust has given way to planets. This is called the **debris disk** [1].

Objective:

Direct observation of the circumstellar environment is challenging because the light from the star that we observe is a mixture of light from the star itself and light that has been scattered on the star's disk. This scattered light is polarized, unlike the light coming directly from the star. This is a phenomenon where light waves align in a certain way[2]. This polarization enables a polarimetric imaging analysis using the Spectro-Polarimeter High-Contrast Exoplanet REsearch (SPHERE), located on the third VLT[3] and illustrated in Figure 2.a. The objective of polarimetric imaging is to obtain the Jones

[1] In our solar system, such a disk would correspond to the Kuiper Belt.
[2] In general, polarized light results from the scattering of unpolarized light by certain materials.
[3] The Very Large Telescope (VLT) operated by the European Southern Observatory (ESO), located on Cerro Paranal in the Atacama Desert of northern Chile consists of four individual telescopes

parameters (I_u, I_p, θ), defined as:

- I_u: Intensity of the unpolarised light, which is the light that directly arrives to us from the star.
- I_p: intensity of the polarized light, which is the light from the star scattered on the disk of dust. This is a useful parameter for studying the formation processes of a stellar environment and searching for exoplanets.
- θ: The polarization is caused by the reflection onto the circumstellar dust.

Thus with the parameters I_p and θ, we can observe circumstellar environments at Near InfraRed (NIR) wavelength.

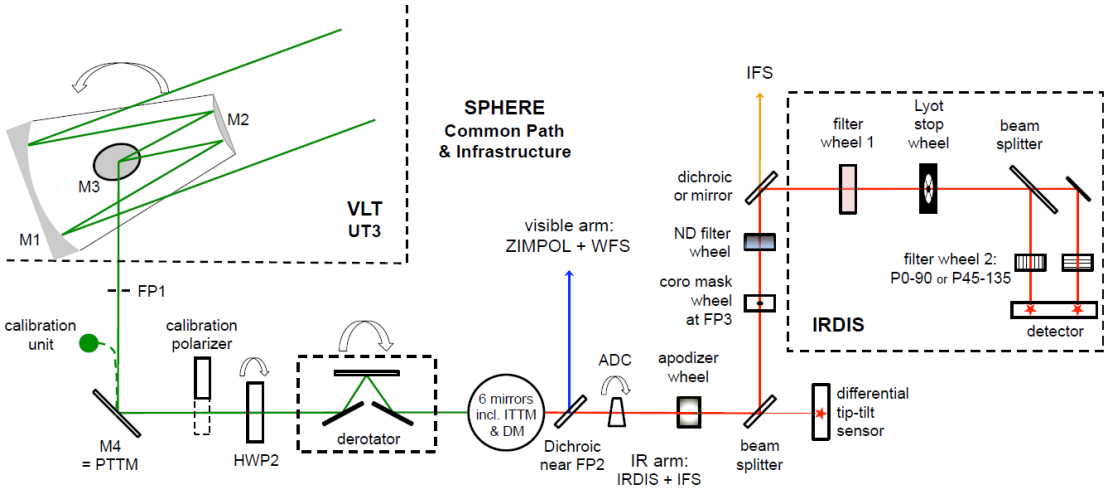


Figure 2: Pathway of light in ESO/SPHERE. [5]

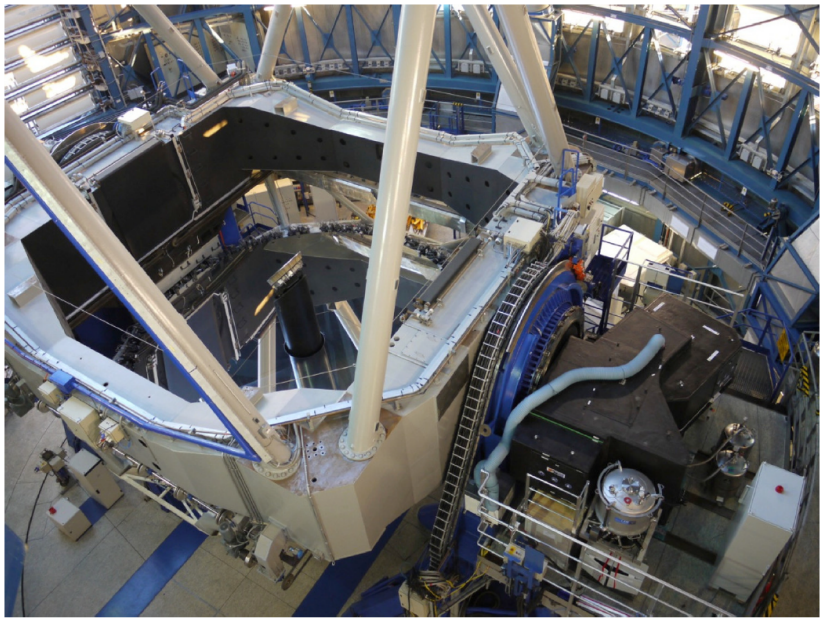
Difficulties during data acquisition:

During the acquisition of data by the instrument SPHERE, a lot of difficulties are typically encountered. Let's detail a few of them.

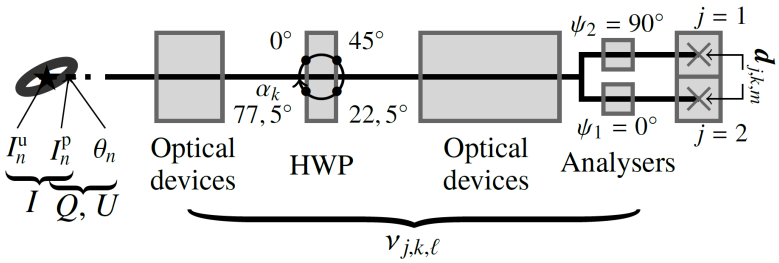
From our perspective, the star's disk can be observed from multiple perspectives, including face-on and edge-on orientations, capturing various stages of its transformation from a protoplanetary disk to a debris disk. This results in a wide variety of data, thus rich in information.

However, the angular size of those objects ranges between 0.01 and 2 arc seconds. Our eyes can perceive objects larger than 1 arc minute in size. To observe these environments, we use data from the VLT. The wavelength that can be directly observed with a telescope depends on its diameter. The VLT can achieve good resolution for light in the near-infrared ($1 - 30\ \mu m$) for observations at the desired angular size.

A second challenge arises from the fact that the light from these environments is thousands of times fainter compared to the light emitted by the host star. This significant difference in brightness poses observational difficulties. To address this, a mask called coronograph is employed, as shown in Figure 1. Nevertheless, although the mask diminishes starlight, the data still retains its high contrast, and it is polluted by residual light from the star.



(a)



(b)

Figure 3: (a) Picture of the ESO/VLT-SPHERE instrument on the Nasmyth platform of the VLT-UT3.(b) Pathway of light in IRDIS. [7]

Another challenge encountered by ground-based telescopes, like the VLT, is caused by atmospheric aberrations that lead to a degradation of resolution during observations. To counteract this, adaptive optics is implemented through the use of the SPHERE instrument. Their purpose is to mitigate atmospheric effects and residual light, thereby significantly augmenting the potential to capture high-resolution images of circumstellar environments.

This whole process requires a large collaboration of thousands of people throughout the world.

IRDIS data and model:

Notations:

- N number of pixels of one restored parameter.
- F is the number of pixels on one image taken by one analyzer.
- K number of acquisitions.

The data gathered by the VLT are processed through IRDIS, a sub-instrument of SPHERE, and undergo postprocessing to render it usable.

The electrical scalar field from a circumstellar environment recorded by the VLT is collected in several acquisitions. One acquisition is depicted in figure 3.b. We can see that at the k^{th} acquisition, the light is split and enters the two polarimetric analyzers, resulting in an image composed of two sub-images with different characteristics according to the settings of the analyzer that processed it. Each sub-image is composed of F pixels. Then the polarimetric data is denoted $(d_{j,k,f})_{j\in\{1,2\},k\in[\![1,K]\!],f\in[\![1,F]\!]}$. To summarize, K represents the number of acquisitions performed by the IRDIS instrument, $j = 1$ and $j = 2$ are the two analyzers and m is the number of pixels within a single acquisition of a sub-image. We refer to d for the whole observation, we refer to d_k for one acquisition composed of two sub-images, and $d_{j,k}$ for one acquired image by an analyzer. At each acquisition, the half-wave plate's (HWP) angle varies in $\{0°; 22.5°; 45°; 77.5°\}$. The number of acquisitions for one observation can vary between 32 and 512, and certain pre-processing is applied to rectify dead pixels and eliminate acquisitions that have failed (due to atmospheric issues, for example).

Each element of figure 3.b serves as a source of degradation of the electrical scalar field e_n recorded by the VLT. We denote M_1 and M_2 as the degradations produced by the optical devices, and J_k as the one introduced by the HWP. The effects of the analyzers can be likened to a projection onto the polarization basis $a_j \in \mathbb{R}^2$. When the field is recorded by sensors, the number of photons on each pixel is saved, which can be assimilated to a Gaussian random variable. Finally, the data is modeled as follows:

$$d_{j,k,n} = \mathbb{E}_t[|| < a_j, M_2 J_k M_1 e_n >_{\mathbb{C}^2} ||^2] + \epsilon_{j,k,n}; \quad \text{with } \epsilon \text{ a Gaussian noise.}$$

In [6], it is demonstrated that this model is not linear with respect to the Jones parameter θ, which complicates the consideration of dead pixels and allows to work only with pre-treated data and not on calibrated data, which is better to obtain quality results. To overcome this difficulty and achieve a better model fitting within the inverse problem framework, the common approach is to use Stokes parameters (I, Q, U), which can be expressed in terms of Jones parameters and vice versa. For a pixel $n \in \{1, ..., N\}$,

$$\begin{cases} I_n = I_n^u + I_n^p, \text{ the total intensity.} \\ Q_n = I_n^p \cos 2\theta_n, \text{ the horizontal polarized intensity.} \\ U_n = I_n^p \sin 2\theta_n, \text{ the vertical polarized intensity.} \end{cases}$$

If we denote $x_n = (I_n, Q_n, U_n)^T$, [6] proved that if we take in account instrumental Point Spread Function (PSF), the model can be expressed linearly with respect to the Stokes parameters as follows:

$$d_{j,k,n} = \sum_{z=1}^{3} \begin{bmatrix} T_{1,k} v_{1,k,z} \\ T_{2,k} v_{2,k,z} \end{bmatrix} (P x_z) + \epsilon_{j,k,n} \tag{1}$$

$$= T\,(Px) + \epsilon_{j,k,n} \tag{2}$$

where $(v_{j,k,z})_{j\in\{1,2\},k\in\{1,\dots k\},z\in\{1,\dots 3\}} \in \mathbb{R}$ is combinations of $(M_2 J_k M_1)^* a_j$, $P : \mathbb{R}^N \to \mathbb{R}^N$ is an operator for convolution by the instrumental PSF, and $(T_{j,k})_{j\in\{1,2\},k\in\{1,\dots K\}} : \mathbb{R}^N \to \mathbb{R}^F$ are linear transformation operators which map the space of parameters to the space of data. Thus $T : (\mathbb{R}^N)^3 \to (\mathbb{R}^{2F})^K$. We retrieve the classical formulation of the inverse problem:

$$d = Ax + \epsilon \tag{3}$$

with $A \in \mathbb{R}^{2FK\times 3N}$ and ϵ a Gaussian random variable. The overall objective is to estimate Stokes parameters $x = (I, Q, U)$ from $d = (d_{j,k,f})_{j\in\{1,2\},k\in[\![1,K]\!],f\in[\![1,F]\!]}$ to recover Jones parameters (I_u, I_p, θ).

This internship focuses on developing reconstruction methods for this problem, but to apply methods and test them more simply, we will use pre-reconstructed data $y = T^\dagger d$. Pre-reconstructed data are derived from observations d with a pipeline based on double difference [20]. With this method, we obtain a blurred and noised estimation of the signal we aim to recover. Reconstructing the parameter of interest from that pre-reconstructed data still fits with the very general formulation (3). In this case, A is a blur operator that can be represented through a convolution filter. This report focuses on reconstruction with supervised learning. The problem with pre-reconstructed data is that we do not have access to the ground truth. To overcome this problem we use a generated dataset based on Debris DIsks Tools (DDIT)[4][14]. It produces synthetic images of $(I_u^{disk}, I_p, \theta)$[5] of debris disks[6]. We used two datasets in the experimental part. The objective is to recover (I_p, Q, U) from their degraded version:

- Pre-reconstructed dataset is generated based on DDIT data with additional parameter I_u^{star} to obtain Jones parameters (I_u, I_p, θ). Parameters (I_p, Q, U) are derived from Jones parameters and are blurred with a PSF P[7]. The model developed in [6] produced observations d from blurred

[4]DDIT's code is available on GitHub: https://github.com/joolof/DDiT
[5]The unpolarized intensity I_u is the sum of unpolarized intensity coming from the star I_u^{star} and the one coming from the disk I_u^{disk}.
[6]which is the last stage of circumstellar environments formation
[7]I_u^{star} and P has been provided by Maud Langlois, a researcher at CNRS, CRAL, Université Lyon 1, ENS Lyon, Observatoire de Lyon.

parameters (I_p, Q, U). The double difference method produced $y = T^\dagger d$, the pre-reconstruction of (I_p, Q, U) parameters. Figure 4 shows a sample from this dataset.

- Synthetic dataset is generated from the blurred (I_p, Q, U) obtained in the pre-reconstructed dataset, to which I added Gaussian noise with a standard deviation of 0.1. Then $y = P(I_p, Q, U) + \epsilon$ Figure 5 shows a sample from this dataset.

In the next part, we introduce methods to solve such a problem.

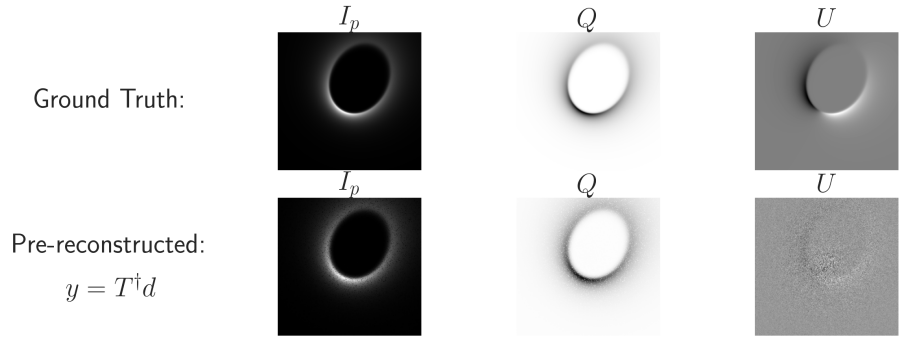


Figure 4: Sample from pre-reconstructed dataset.

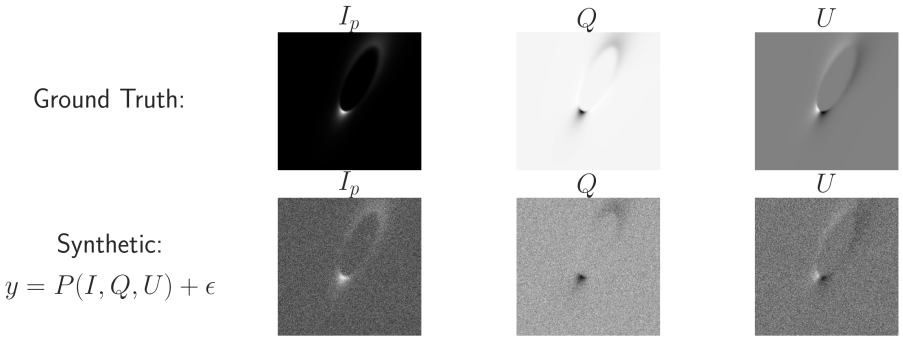


Figure 5: Sample from synthetic dataset.

III. Reconstruction methods

The aim of *inverse problem* solving is, from a certain set of observed measurements $\boldsymbol{y} \in \mathbb{R}^K$, to build an estimator $\widehat{\boldsymbol{x}} \in \mathbb{R}^L$ close from the ground truth data $\bar{\boldsymbol{x}}$.
In our case, we consider the model (1) that can be formulated as $\boldsymbol{y} = A\bar{\boldsymbol{x}} + \epsilon,$ with $A \in \mathbb{R}^{M \times L}$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$.

III.1 Variational methods

Notation:

L is the dimension of the space of the parameters of interest. $L = 3N$ in our case.

N is the number of pixels of one variable, then $\frac{L}{N}$ is the number of variable.

G is the dimension of the feature space.

M is the dimension of the degraded data.

Objective of the methods:

The estimator $\widehat{\boldsymbol{x}}$ of $\bar{\boldsymbol{x}}$ is classically obtained by solving a minimization problem, called a variational problem, of the form:

$$\widehat{\boldsymbol{x}} \in \underset{\boldsymbol{x} \in \mathbb{R}^L}{\operatorname{Arg\,min}} \left[\frac{1}{2} ||\boldsymbol{y} - A\boldsymbol{x}||_2^2 + \mathcal{R}(\boldsymbol{x}) \right] \tag{4}$$

Here $||\boldsymbol{y} - A\boldsymbol{x}||_2^2$ is the data fidelity term, which ensures that the degraded solution of the problem, $A\widehat{\boldsymbol{x}}$, is nearby the degraded signal $\boldsymbol{y}$. $\mathcal{R} : \mathbb{R}^L \to \mathbb{R}$ is a regularisation term, which is often decomposed as $\mathcal{R}(\boldsymbol{x}) = \lambda g(D\boldsymbol{x})$, with $D : \mathbb{R}^L \to \mathbb{R}^G$ a linear operator and $g : \mathbb{R}^G \to]-\infty, +\infty]$ a penalty function whose contribution must be adjusted by the parameter $\lambda \geq 0$.

The common interpretation of this problem is to assume that x is a realization of a random vector X following a certain probability distribution $p(x)$. A popular strategy for estimating $\bar{x}$ is to solve the Maximum A Posteriori (MAP) problem:

$$\hat{x} \in \underset{x \in \mathbb{R}^L}{\operatorname{Arg\,min}} \left(\underbrace{-\log p_{y|X=x}(y)}_{\text{likelihood}} \underbrace{-\log(p_X(x))}_{\text{prior}} \right) \tag{5}$$

Data fidelity term:

When ϵ is assumed to be a Gaussian noise, the likelihood is:

$$\forall \boldsymbol{x} \in \mathbb{R}^L, \quad p_{Y=y|X}(x) = \frac{\mathrm{e}^{-\frac{1}{2\sigma^2}||\boldsymbol{y}-A\boldsymbol{x}||_2^2}}{\sqrt{(2\pi)^K}\sigma}$$

or equivalently, minimize the negative log-likelihood, that is:

$$-\log p_{Y=y|X}(x) = \frac{1}{2\sigma^2}||\boldsymbol{y} - A\boldsymbol{x}||_2^2 + \frac{K\log(2\pi)}{2} + \log(\sigma) \tag{6}$$

When involved in (5) it turns to take the form of the data fidelity term in (4). If $\lambda = 0$ in (4), A^*A, the Gram matrix is invertible[8], the minimum of the data fidelity is expressed with the pseudo-inverse $A^\dagger$ which can blow up the noise if A is ill-conditioned[9]:

$$\begin{aligned}
\nabla_x \left(\frac{1}{2}\|\mathbf{y} - A\hat{\mathbf{x}}\|_2^2\right) = A^*(\mathbf{y} - A\hat{\mathbf{x}}) = \mathbf{0} \\
&\Leftrightarrow A^*A(\bar{\boldsymbol{x}} - \widehat{\boldsymbol{x}}) + A^*\epsilon = \mathbf{0} \\
&\Leftrightarrow \hat{\boldsymbol{x}} = \underbrace{\left((A^*A)^{-1}A^*\right)}_{A^\dagger}\epsilon + \bar{x} \\
&\Rightarrow ||\bar{\boldsymbol{x}} - \hat{\boldsymbol{x}}||_2 = ||A^\dagger\epsilon||_2
\end{aligned}$$

This issue can be addressed by introducing a regularization term to the data fidelity term, which improves conditioning and thereby reduces the variance of $\widehat{\boldsymbol{x}}$. This term incorporates a prior assumption about the probability distribution of $\bar{\boldsymbol{x}}$ and can enhance conditioning, ensuring robustness to numerical errors and ensuring the existence of a solution.

Regularisation term:

The choice of the type of regularization used to solve the inverse problem will depend on the expected structures in the images of the circumstellar environments. Most real images are piece-wise constant, it is the case of circumstellar environments that have black backgrounds. This assumption also permits to denoise and obtain a sharp edge and correct blur. This is well known that in the case of searching for a sparse parameter x, it simply involves using the ℓ_1 norm on the values of x. For a regularization encouraging piecewise constant behaviour, a popular approach is to impose sparsity on the horizontal and vertical finite differences of the images. For an image x, we denote the finite difference as $Dx \in \mathbb{R}^{2L}$ which is in the features space of dimension $G = 2L$. It consists of horizontal and vertical pixel differences illustrated on Figure 6 which are the results of convolution of the image with two filters f_h and f_v. For (l_1, l_2) a pixel of a parameter of interest x_z:

[8] If we consider pre-treated data, A is invertible, but with the global problem it isn't.
[9] i.e. A has little eigenvalues.

$$(D\boldsymbol{x_z})_{l_1,l_2} = \begin{pmatrix} \left(D^{\mathrm{h}}\boldsymbol{x_z}\right)_{l_1,l_2} \\ \left(D^{\mathrm{v}}\boldsymbol{x_z}\right)_{l_1,l_2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_{z,l_1+1,l_2} - \boldsymbol{x}_{z,l_1,l_2} \\ \boldsymbol{x}_{z,l_1,l_2+1} - \boldsymbol{x}_{z,l_1,l_2} \end{pmatrix}$$
$$= \begin{pmatrix} (f_h * \boldsymbol{x_z})_{l_1,l_2} \\ (f_v * \boldsymbol{x_z})_{l_1,l_2} \end{pmatrix} \quad \text{with } f_h = \begin{bmatrix} -1 & 1 \end{bmatrix}; f_v = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

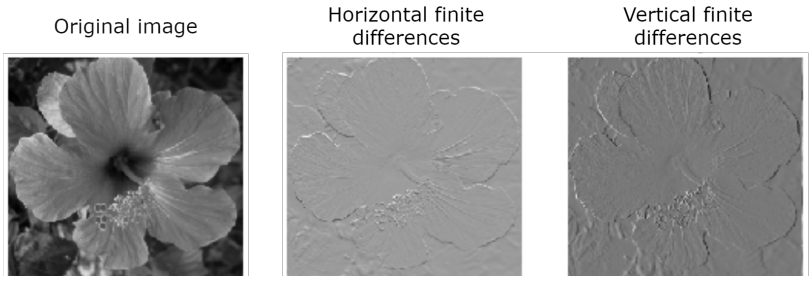


Figure 6: An image and the two sub-images constituting its horizontal and vertical finite difference.

Then in the experiments presented in Section 4, we use the anisotropic Total Variation (TV) regularization formulated as $\mathcal{R}(\boldsymbol{x}) = \lambda\|D\boldsymbol{x}\|_1$. The minimization problem becomes:

$$\widehat{\boldsymbol{x}} \in \underset{\boldsymbol{x}\in\mathbb{R}^L}{\operatorname{Arg\,min}} \left[\frac{1}{2}||\boldsymbol{y} - A\boldsymbol{x}||_2^2 + \lambda\|D\boldsymbol{x}\|_1\right] \tag{7}$$

where λ is the hyper-parameter which will controls the amount of regularization. This regularization method effectively promotes sparsity in the gradient of the reconstructed image, encouraging piecewise constant solutions.

The choice of regularisation parameters can be difficult, specifically when there are several of them like in RHAPSODIE. In this work, we will explore methods based on learning that solve this problem.

Proximal gradient descent algorithm:

In this section, we will introduce a simple algorithm to solve the general minimization problem (4) refered as Proximal Gradient Descent (PGD) algorithm also called Forward-Backward (FB). In Appendix VI.2, we introduce an important tool for non-differentiable optimization: the subdifferential ∂ and the proximity operator of a proper, lower semi-continues and convex function h that we denote by prox_h.

The PGD method is derived through a correspondence between fixed points of a specific operator and

the solution of problem (3) as follows:

$$\begin{aligned}
\widehat{\boldsymbol{x}} \in \underset{\boldsymbol{x}\in\mathbb{R}^L}{\operatorname{argmin}} \frac{1}{2}\|A\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \mathcal{R}(\boldsymbol{x}) &\Leftrightarrow 0 \in \tau\partial\left(\frac{1}{2}\|A\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \mathcal{R}\right)(\widehat{\boldsymbol{x}}) \\
&\Leftrightarrow 0 \in \{\tau A^*(A\widehat{\boldsymbol{x}}-\boldsymbol{y}) + \partial\tau\mathcal{R}(\widehat{\boldsymbol{x}})\} \\
&\Leftrightarrow (\widehat{\boldsymbol{x}}-\widehat{\boldsymbol{x}}) \in \{\tau A^*(A\widehat{\boldsymbol{x}}-\boldsymbol{y}) + \partial\tau\mathcal{R}(\widehat{\boldsymbol{x}})\} \\
&\Leftrightarrow (\widehat{\boldsymbol{x}}-\tau A^*(A\widehat{\boldsymbol{x}}-\boldsymbol{y})-\widehat{\boldsymbol{x}}) \in \partial\tau\mathcal{R}(\widehat{\boldsymbol{x}}) \\
&\Leftrightarrow \widehat{\boldsymbol{x}} = \operatorname{prox}_{\tau\mathcal{R}}(\widehat{\boldsymbol{x}}-\tau A^*(A\widehat{\boldsymbol{x}}-\boldsymbol{y})) \quad \text{according to property (VI.2.1)} \\
&\Leftrightarrow \widehat{\boldsymbol{x}} \in \operatorname{Fix}_\Phi \quad \text{with } \Phi = \operatorname{prox}_{\tau\mathcal{R}}(\operatorname{Id} - \tau A^*(A\cdot - \boldsymbol{y}))
\end{aligned}$$

This operator possesses certain properties that are sufficient to ensure the convergence of the Proximal Gradient Descent algorithm:

Algorithm 1: Proximal Gradient Descent algorithm (PGD)

Assumption: $\mathcal{R} \in \Gamma_0(\mathbb{R}^L)$
Input : $\boldsymbol{y} \in \mathbb{R}^M$, step size $\tau \in]0, 2\zeta^{-1}[$
Output : $\lim_{n\to\infty} x^n \in \underset{x\in\mathbb{R}^L}{\operatorname{Arg\,min}} \left[\frac{1}{2}||\boldsymbol{y}-A\boldsymbol{x}||_2^2 + \mathcal{R}(\boldsymbol{x})\right]$
$x^0 = A^*y$
for $n = 0, 1, 2, \ldots$ **do**
  $\tilde{x}^n = x^n - \tau A^*(Ax^n - \boldsymbol{y})$;
  $x^{n+1} = \operatorname{prox}_{\tau\mathcal{R}}(\tilde{x}^n)$;

In our optimization problem (7), $\mathcal{R} : x \to \lambda||\nabla x||_1$ has no closed form for its proximal operator. Thus we cannot use PGD. The next section introduces a new algorithm which overcomes this problem.

Condat-Vũ algorithm:

In this section, the goal is to address the Total Variation problem (Equation 3). In this context, we define $\mathcal{R}(x) = \lambda g(Dx)$. The challenge we encountered with PGD was computing the proximity operator of $g \circ D$.

Such a problem can then be solved using the primal-dual algorithm called Condat-Vũ [4], with the following iterations[10]:

Algorithm 2: Condat-Vũ algorithm (CV)

Assumption: $g \in \Gamma_0(\mathbb{R}^L)$
Input : $y \in \mathbb{R}^M, \lambda > 0, \tau > 0, \sigma > 0$, such that $\frac{1}{\tau} - \sigma||D||^2 > \frac{||A||^2}{2}$
Output : $\lim_{n\to\infty}(x^n) \in \underset{\boldsymbol{x}\in\mathbb{R}^L}{\operatorname{Arg\,min}} \left[\frac{1}{2}||y-Ax||_2^2 + \lambda g(Dx)\right]$
$x^0 = A^*y,\ u^0 = Dx^0$;
for $n = 0, 1, 2, \ldots$ **do**
  $x^{n+1} = x^n - \tau A^*(Ax^n - y) - \tau D^* u^n$;
  $u^{n+1} = \operatorname{prox}_{\sigma g^*}(u^n + \sigma D(2x^{n+1} - x^n))$;

Remark: τ and σ are the primal and dual stepsize.

[10] See Appendix VI.3 for more details.

This primal-dual method aims to reduce the complexity of computing the proximal operator associated with the composition of a non-smooth function with a linear operator by separating their contribution.

III.2 Unfolded variational methods

This section focuses on proposed informed neural networks, where the architecture is inspired by the minimization problem presented before. We will see that such networks can achieve better performances than variational methods because they are able to learn the prior for the specific dataset we use to train it. They are more interpretable than classical deep neural networks because they are developed via modeling the physical processes underlying the problem and capturing prior domain knowledge.

A neural network d_θ with K layers and parametrized by θ is defined as:

$$d_\theta(u) = \eta^{[K]}(W^{[K]} \dots \eta^{[1]}(W^{[1]}u + b^{[1]}) \cdots + b^{[K]}) \tag{8}$$

More details about neural networks are introduced in Appendix VI.5

The idea behind Unfolded networks [9] is to recognize the architecture of a classical layer of a neural network as one iteration of a variational method. Under this paradigm, neural network can be set as one iteration of an iterative algorithm, where some components have learnable parameters, which may include the step size of the algorithm and/or the dictionary of filters, or even the entire regularization term, which is replaced by a deep neural network acting as the proximal operator. Concatenating a finite number of these layers forms a deep neural network that is physics-driven by variational methods which is easy to evaluate and is free of choice of an arbitrary λ hyperparameter. The counterpart is that it requires a dataset $\{\bar{x}_i, y_i\}_{i\in\mathbb{I}}$. In this section, I will introduce several algorithms which can be divided into two distinct classes. The firsts are based on Condat-Vũ and the seconds are based on also called Forward Backward (FB). In the following, we suppose the number of iterations/layers equal to $K \in \mathbb{N}^*$.

Unfolded Condat-Vũ:

The problem (7) we aimed to solve with variational methods has a prior on the sparsity of the finite differences operator D. This prior is natural in image processing, but it has some undesirable properties which are explored further. Indeed, there is no trivial choice for the dictionary of filters we could use with our datasets. This is the reason why we proposed to learn it with Unfolded Condat-Vũ (UnfoldedCV). It is possible to interpret one iteration of the Condat-Vũ algorithm as a succession of affine transformations of the primal-dual initialized input $(x^0, u^0) \in \mathbb{R}^{M+P}$ followed by a non-linear activation [10]:

Proposition III.2.1:

The Condat-Vũ algorithm fits the network (8) when considering, for every $k \in \{1, \dots, K\}, W^{[k]} \in \mathbb{R}^{(L+G)\times(L+G)}, b^{[k]} \in \mathbb{R}^{L+G}$ and $\eta^{[k]} : \mathbb{R}^{L+G} \to \mathbb{R}^{L+G}$ such that:

$$\begin{cases} W^{[k]} = \begin{pmatrix} \mathrm{Id} - \tau A^* A & -\tau D^* \\ \sigma D\left(\mathrm{Id} - 2\tau A^* A\right) & \mathrm{Id} - 2\tau\sigma D D^* \end{pmatrix} \\ b^{[k]} = \begin{pmatrix} \tau A^* \mathrm{y} \\ 2\tau\sigma D A^* \mathrm{y} \end{pmatrix} \\ \eta^{[k]} = \begin{pmatrix} \mathrm{Id} \\ \mathrm{prox}_{\sigma g^*} \end{pmatrix} \end{cases}$$

The strategy we choose to parametrize this neural network is to define the D operator as the action by convolution of a set of $3 * 3$ spatial filters that mix the information of the 3 parameters of interest. We denote by P the number of filters. Figure (7) shows how D is acting on the signal set with $P = 2$.

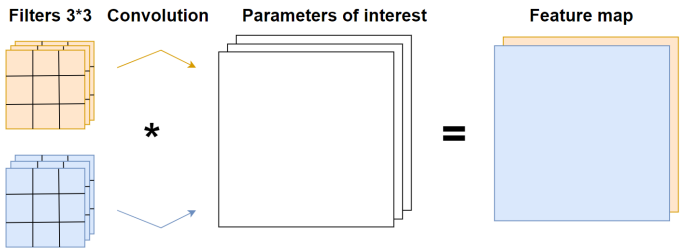


Figure 7: 2 filters 3×3 acting on an image, resulting in a mapping in $\mathbb{R}^{2N}$

We introduce several UnfoldedCV algorithms, we applying the network defined in proposition III.2.1 with the initialization $x^0 = A^* y$ and $u^0 = D_\theta x^0$ for fixed models and $u^0 = D_\theta^0 x^0$ for others. As we don't know the final value of D_θ before training, u^0 is initialized by a convolution between a random set of filters and u^0, which is not ideal.

- **unfoldedCV fix:** We apply the network with the same set D_θ^k of P filters. The number of parameters of such a network is about several hundreds of parameters.
- **unfoldedCV:** We apply the network with a different dictionary D_θ^k which contains P filters. The number of parameters of such a network is about several thousands of parameters.
- **unfoldedCV deep:** We apply the network with two sets of filters. We replace D with $D_\theta^{1,k} D_\theta^{2,k}$. $D_\theta^{1,k}$ consists of a set of P 3×3 filters applied to x^k, mapping the signal space $\mathbb{R}^L$ to the feature map $\mathbb{R}^G$, and $D_\theta^{2,k}$ is another set of P filters, also of size 3×3, applied to $D_\theta^{1,k} x^k$ to increase the number of parameters. This operation significantly increases the number of parameters to a million.
- **unfoldedCV deeprelu:** We apply the network with the non-linear function ReLU[11] function

[11] ReLU stands for Rectified Linear Unit, and it is a mathematical function applied pointwise, capturing only the positive part of the input data.

between the two convolutions of unfoldedCV_deep. We replace D by $D_\theta^{1,k} \circ \text{ReLU} \circ D_\theta^{2,k}$ to introduce non-linearity. This modification doesn't alter the number of parameters, which remains approximately one million.

Those networks have been trained using the Adam algorithm on several datasets. The parameter to optimize is $\Theta = \{\tau, \sigma, \theta\}$ for unfoldedCV fix, $\Theta = \{\tau^k, \sigma^k, \theta^k, k \in [\![1; K]\!]\}$ for unfoldedCV, $\Theta = \{\tau^k, \sigma^k, \theta^{i,k}, i \in \{1, 2\}, k \in [\![1; K]\!]\}$ for unfoldedCV deep and unfoldedCV deeprelu. Where τ and θ represent the primal and dual stepsize which are initialized to be equal to:
$\tau = 0.01$ and $\sigma = \dfrac{(\frac{1}{\tau} - \frac{1}{2})}{2||D_\theta||}$ for fixed models, $\sigma^k = \dfrac{(\frac{1}{\tau^k} - \frac{1}{2})}{2||D_\theta^k||}$ for not fixed models, $\sigma^k = \dfrac{(\frac{1}{\tau^k} - \frac{1}{2})}{2||D_\theta^{1,k} D_\theta^{2,k}||}$
for deep models or $\sigma^k = \dfrac{(\frac{1}{\tau^k} - \frac{1}{2})}{2||D_\theta^{1,k} \circ \text{ReLU} \circ D_\theta^{2,k}||}$ for deep ReLU models.

Unfolded Forward-Backward:

The principle of unfolded Forward-Backward (UnfoldedFB) is slightly different. We also can see one iteration of the PGD algorithm as one layer of a neural network like it is shown in the following Proposition:

Proposition III.2.2:

The PGD algorithm fits the network (8) when considering, for every $k \in \{1, \dots, K\}, W^{[k]} \in \mathbb{R}^{L \times L}, b^{[k]} \in \mathbb{R}^L$ and $\eta^{[k]} : \mathbb{R}^L \to \mathbb{R}^L$ such that:

$$\begin{cases} W^{[k]} = \text{Id} - \tau A^* A \quad + \tau D^* \\ b^{[k]} = -\tau A^* \text{y} \\ \eta^{[k]} = \text{prox}_{\tau \mathcal{R}} \end{cases}$$

but in this case, we will replace the proximal activation with an UNet which is a deep neural network architecture[12] which depends on a hyperparameter called scales. The higher the number scale is, the deeper the UNet is. Moreover, the prior is implicit through the UNet network. Once the proximal operation is replaced with an UNet, there is no longer any corresponding function to minimize. We are dealing with a kind of black box. We implemented two different versions of the forward pass: one fixed and one not fixed.

- **unfoldedFB_fix:** We apply at each iteration the same UNet network U_θ.

[12] see more in the appendix about UNet architecture

Algorithm 3: Unfolded Forward-Backward fixed (unfoldedFBfix)

Assumption: $\zeta := |||A^*A|||_2 < 2$, U_θ a Unet

Input : $y \in \mathbb{R}^M$, $\tau \in]0, 2\zeta^{-1}[$

Output : x^K

$x^0 = A^*y$

for $n = 0, 1, 2, \dots K-1$ **do**

$\tilde{x}^n = x^n - \tau A^*(y - Ax^n)$;

$x^{n+1} = U_\theta(\tilde{x}^n)$;

- **unfoldedFB:** We apply at each iteration a different UNet network $\{U_{\theta^k}\}_{k\in[\![1;K]\!]}$.

Algorithm 4: Unfolded Forward-Backward fixed (unfoldedFB)

Assumption: $\zeta := |||A^*A|||_2 < 2$, $\{U_{\theta^k}\}_{k\in[\![1;K]\!]}$, K Unet

Input : $y \in \mathbb{R}^M$, $\tau \in]0, 2\zeta^{-1}[$

Output : x^K

$x^0 = A^*y$

for $n = 0, 1, 2, \dots K-1$ **do**

$\tilde{x}^n = x^n - \tau A^*(y - Ax^n)$;

$x^{n+1} = U_{\theta^{n+1}}(\tilde{x}^n)$;

The parameter to optimize is $\Theta = \theta$ for unfoldedFBfix and $\{\Theta = \theta^k{}_{k\in[\![1;K]\!]}\}$ for unfoldedFB.

III.3 Plug and Play

The Plug-and-Play (PnP) approach [21] is a deep learning method that proposes the substitution of the proximal activation of the prior in any iterative method with a pre-trained denoiser. The proximal operator of a proper, convex, semi-continuous function behaves like a denoiser.

There exist several ways to implement PnP. We can use any pre-trained denoiser or train one tailored to our data to model the signal set. In our experiments, we chose to use the PGD. Once we have a denoiser D_θ, the PnP_FB algorithm is:

Algorithm 5: Plug and Play Forward Backward algorithm (PnP_FB)

Input : $y \in \mathbb{R}^M$, step size $\tau > 0$

Output: $\lim_{n\to\infty} x^n$

$x^0 = A^*y$

for $n = 0, 1, 2, \dots$ **do**

$\tilde{x}^n = x^n - \tau A^*(\boldsymbol{y} - Ax^n)$;

$x^{n+1} = D_\theta(\tilde{x}^n)$;

One of the methods to prove convergence in this algorithm is to assert that the denoiser D_θ is firmly nonexpansive[13], then the iteration converges sublinearly to its fixed point, then the iterations converge linearly to its unique fixed point[17].

[13] i.e. α-Lipschitz with $0 \leq \alpha < 1$

III.4 Pros and cons of each method:

Variational methods are simple, requiring no training or dataset. They are highly interpretable but may not be finely tuned for the specific task they aim to solve. Additionally, the reconstruction cost is high.

Unfolded methods are physically driven by Variational methods and leverage from neural network architectures that encountered success in recent years, like convolutional and Unet which are very flexible and adaptative to fit the task aimed to solve. They also offer cost-effective evaluation but necessitate training and a dataset. However, addressing new tasks with different degradations or datasets requires new training, and they may be challenging to interpret.

Plug and Play methods are also based on Variational methods and leverage top-performing neural, network architectures. However, they don't explicitly consider the signal's degradation. This means that no retraining is necessary if the dataset remains the same but with a different degradation. It also means that those methods are not finely tailored for the degradation of the signal. Moreover, the evaluation cost is high if we iterate methods until convergence.

IV. Numerical experiments

In this chapter, we present the results obtained using the models introduced in the section III, and we compare their performance on two datasets. In each part of this chapter, we explain the dataset, provide results for all model types, and compare the best performing models.

For Condat-Vũ models we manually tuned the hyper-parameter λ. For other models the same settings have been employed for each dataset:

- Unfolded Condat-Vũ models, including fix, not fixed, not fixed deep, and not fixed deep ReLU D, have been implemented with 4 different settings, where $\mathfrak{P}$ represents the number of dictionaries and $\mathfrak{L}$ the number of layers. We also provide the number of parameters of each of these configurations in the following table:

$\mathfrak{P}$	$\mathfrak{L}$	unfoldedCV_fix	unfoldedCV	unfoldedCV_deep	unfoldedCV_deeprelu
16	80	434	34 720	219 040	219 040
32	40	866	34 640	403 280	409 280
64	20	1730	34 600	771 880	771 880
128	10	3458	34 580	1 509 140	1 509 140

- Unfolded Forward-Backward models including fixed and not fixed U_θ, have been implemented with 6 different settings, where $\mathfrak{S}$ represents the number of scales and $\mathfrak{L}$ the number of layers:

$\mathfrak{S}$	2		3		4	
$\mathfrak{L}$	5	10	5	10	5	10
nb parameters fixed	446019		2070723		8564163	
nb parameters not fixed	2 230 095	4 460 190	10 353 615	20 707 230	42 820 815	85 641 630

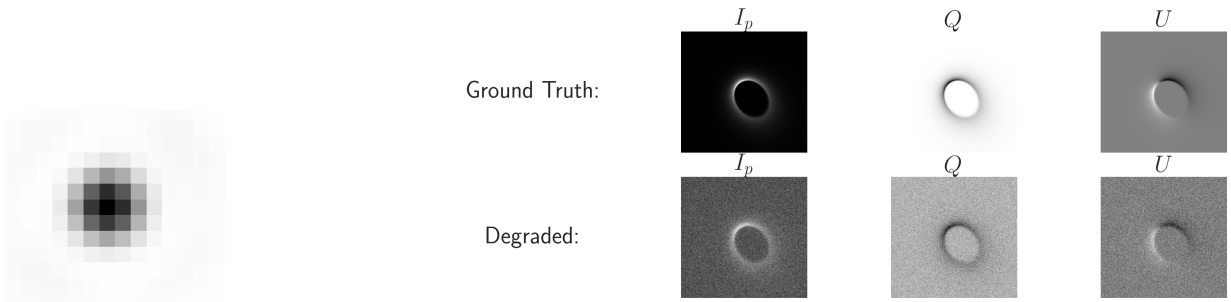
- Plug and Play Proximal Gradient Descent models have been implemented with pre-trained denoiser DeNoising Convolutional Neural Networks (DNCNN) [23] and Dilated-Residual U-Net (DRUNet) [8] which are available in the Python library Deep Inverse [18]. We used 2 different versions of DNCNN. One is Lipschitz to take advantage of the convergence results established in [17], the other has no such constraint. DNCNN has 668 227 parameters and DRUNet has 32 640 960.

IV.1 Synthetic data

Synthetic dataset is generated based on DDIT library [14]. It produces synthetic images of $(I_u^{disk}, I_p, \theta)$[14] and with additional parameter I_u^{star} we obtain Jones parameters (I_u, I_p, θ). Parameters (I_p, Q, U) are

[14]The unpolarized intensity I_u is the sum of unpolarized intensity coming from the star I_u^{star} and the one coming from the disk I_u^{disk}.

derived from Jones parameters, and after normalization, they are blurred with a PSF A[15], which is a Gaussian blur filter of size 17×17 as illustrated in Figure 8 (a). I added Gaussian noise with a standard deviation of 0.1. Then $y = A(I_p, Q, U) + \epsilon$. Thus each ground truth image consists of three channels (I_p, Q, U), each with dimensions $(244, 254)$. The mean Peak Signal-to-Noise Ratio (PSNR) of this dataset is 19.95. An example is presented in Figure 8.(b). Subsequently, the dataset has been divided into a training set, comprising 85% of the data (573 images), and a test set, consisting of the remaining 102 images.



(a) **(b)**

Figure 8: (a). Gaussian blur filter. (b). Sample from synthetic dataset

Variational methods

The Condat-Vũ algorithm have been employed to address the variational problem (7). The experiments encompass four distinct values of the parameter λ. For each experimental setup, a table similar to the one shown in Figure 9.(a) is presented. Each row in the table corresponds to a specific configuration, which is detailed in the first column, while the second column displays the PSNR (Peak Signal-to-Noise Ratio) achieved on the test set.

It's important to note that Condat-Vũ and PnP are deterministic algorithms, ensuring the reproducibility of results across experiments. However, this reproducibility does not extend to deep-learning methods. To assess the stability of these models, we train and evaluate them with three different initializations, leading to the presence of three columns (0, 1, 2) for deep-learning models and 2 others for the mean and standard deviation, respectively, across all initializations of the model. While Condat-Vũ and PnP, have only one column. Consequently, for Condat-Vũ and PnP, the mean is the PSNR over the test set and standard deviation is 0.

In the final column, we report the training time for deep-learning models and the time required to compute the iterations of the Condat-Vũ and PnP algorithm over the entire test set. This last column may not always be consistent due to the use of different Graphics Processing Units (GPUs), which can lead to variations in computational speed.

The red curves in our figures represent the PSNR values across iterations for variational methods

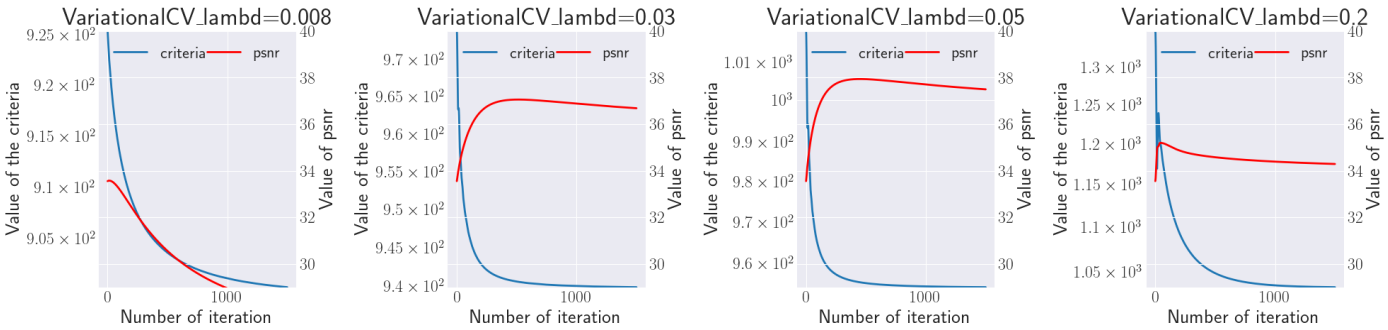
[15] I_u^{star} and A have been provided by Maud Langlois, an astrophysicist from CRAL.

or PnP and PSNR across training epochs for deep-learning methods. The blue curves represent the value of the objective function minimized by the Condat-Vũ method, the loss function for deep-learning methods, and the norm of residuals for PnP. Additionally, we present one or two reconstructed samples.

Upon examining the curves, it is clear that Condat-Vũ converges towards the minimum of the variational problem. It's worth noting that the PSNR is displayed on a logarithmic scale, which results in a longer stabilization period. The best performance is achieved with a regularization parameter of $\lambda = 0.05$. In Figure 10, the impact of the regularization parameter is observable. A lower λ value leaves the image undenoised, while a larger λ value introduces artifacts due to patch effects[16]. For $\lambda = 0.2$, the patch effect dominates, resulting in a too thick ring for channels I_p and U.

	mean	n_params	training_time (h)
lambd=0.008	28.15	0	0.06
lambd=0.03	36.69	0	0.06
lambd=0.05	37.5	0	0.06
lambd=0.2	34.29	0	0.06

(a)



(b)

Figure 9: (a) .Summary of the quantitative results obtained with the variational approach based on TV penalization and considering CV algorithm. (b). Evolution of PSNR and value of the function to minimize through iterations.

[16]The patch effect is characterized by two phenomena: *i*) structures with horizontal and vertical edges, which naturally emerge because diagonals are more penalized by the ℓ_1-norm than horizontal and vertical ones, *ii*) large areas with consistent colors, as the penalty primarily targets color variation.

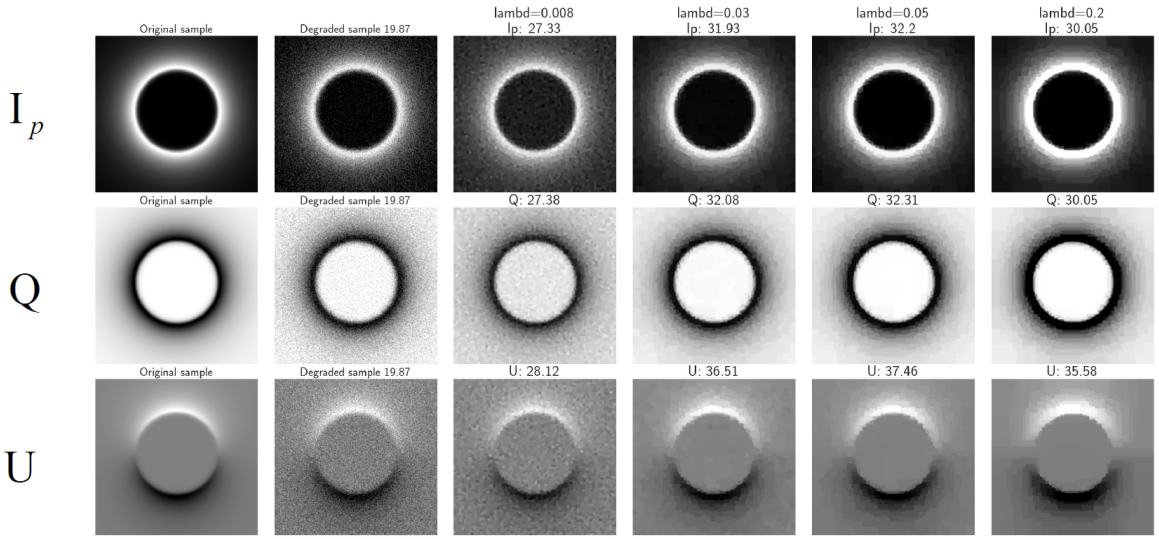


Figure 10: Reconstructed data with several regularisation parameters.

Unfolded Condat-Vũ:

The Unfolded Condat-Vũ models, including fix, not fix, and deep, were trained for 700 epochs, while the Unfolded Condat-Vũ deep-ReLU models were trained for 200 epochs. Therefore, we should exercise caution when interpreting training times. Nonetheless, it's evident that training is longer for deeper networks inner a model type, i.e., those with a depth of 80.

The table presented in Figure 11.(a) and Figure 12.(a) contains a total of 48 models. However, we only display the curves for the best-performing models across the four model types because the shape of the curves within each type doesn't vary significantly. Notably, models with a greater number of parameters tend to produce better results.

Moreover, the choice of depth and the number of dictionaries was made to obtain a number of parameters constant with non-fixed models. Then it appears that when the number of parameters is limited (i.e., not deep), models with higher depth achieve better results, even when the number of parameters is nearly 10 times lower with fixed models. With P=128 and depth=10, which amounts to 3458 parameters, the mean PSNR is 33.6, whereas with P=16 and depth=80, which comprises 434 parameters, the PSNR reaches 33.73.

The fixed models exhibit the highest stability. This can be attributed to the shape of the PSNR curves for fixed models in Figure 11.(b), where the PSNR increases over a few epochs until it reaches the minimum of the loss function and then remains stable until the end. This suggests that fixed models may lack complexity. While deep networks generally achieve slightly better performance than not fixed models, they come with significantly more parameters, ranging from 10 to 50 times more. It's worth noting that, based on the shape of the PSNR curve, deep-ReLU models could potentially outperform deep models if they were trained for the same number of epochs. Deep-ReLU models stand out as the best-performing model type from both a training time perspective and the results achieved, with PSNR values exceeding 45. Although these neural networks are challenging to interpret despite

their physically driven structure. With the exception of the fixed network, Figure 13 shows that all performances are close to perfection, with only some texture effects slightly visible on Q and U images.

	0	1	2	mean	std	n_params	training_time (h)
fixL_P=128_depth=10	33.62	33.58	33.61	33.6	0.02	3458	5.6
fixL_P=16_depth=80	33.79	33.81	33.6	33.73	0.09	434	6.13
fixL_P=32_depth=40	33.72	33.58	33.62	33.64	0.06	866	7.45
fixL_P=64_depth=20	33.65	33.58	33.65	33.63	0.03	1730	3.68
notfixL_P=128_depth=10	39.89	39.18	40.78	39.95	0.65	34580	2.97
notfixL_P=16_depth=80	45.19	44.42	44.65	44.75	0.32	34720	19.8
notfixL_P=32_depth=40	43.69	43.73	42.64	43.35	0.5	34640	10.2
notfixL_P=64_depth=20	41.84	42.7	42.07	42.2	0.36	34600	2.63

	0	1	2	mean	std	n_params	training_time (h)
deep_P=128_depth=10	41.8	43.28	43.28	42.79	0.7	1509140	10.23
deep_P=16_depth=80	43.75	43.85	43.59	43.73	0.11	219040	15.45
deep_P=32_depth=40	43.95	43.93	44.05	43.98	0.05	403280	10.9
deep_P=64_depth=20	43.46	43.86	43.47	43.6	0.19	771880	10.62
deeprelu_P=128_depth=10	44.51	45.44	44.41	44.79	0.46	1509140	2.21
deeprelu_P=16_depth=80	43.54	43.52	43.26	43.44	0.13	219040	4.8
deeprelu_P=32_depth=40	43.92	43.46	43.97	43.78	0.23	403280	2.53
deeprelu_P=64_depth=20	44.11	44.59	44.57	44.42	0.22	771880	1.87

Figure 11: Summary of the quantitative results obtained with UnfoldedCV models.

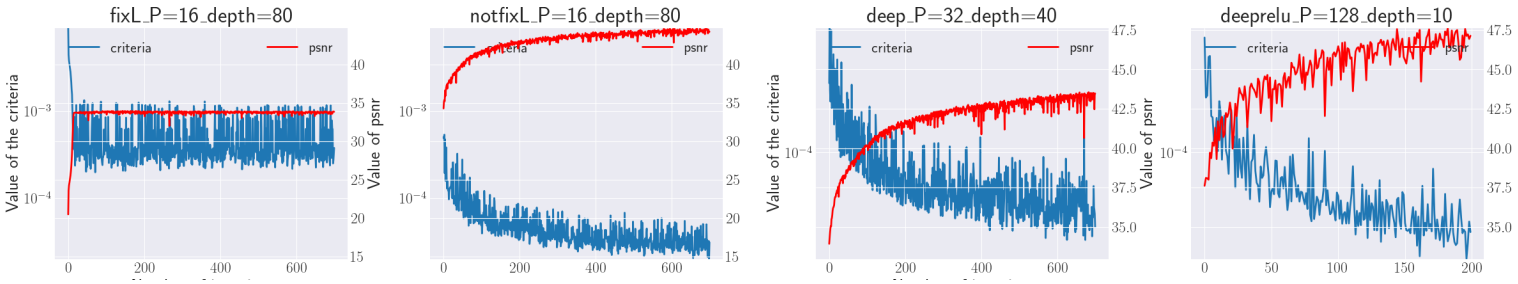


Figure 12: Evolution of PSNR and the loss function through epochs.

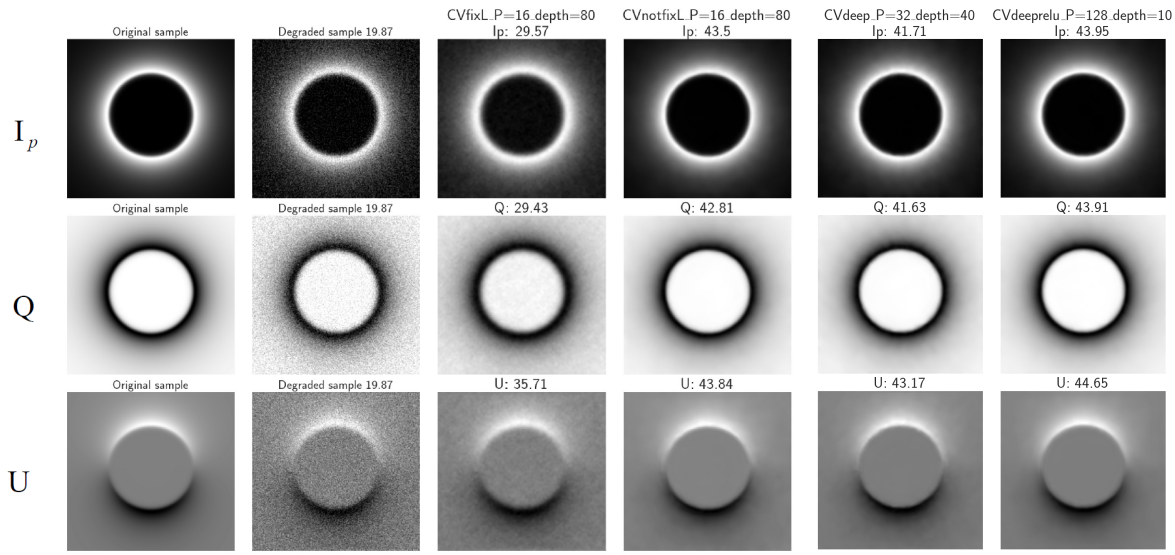


Figure 13: Reconstructed sample with unfolded CV deep.

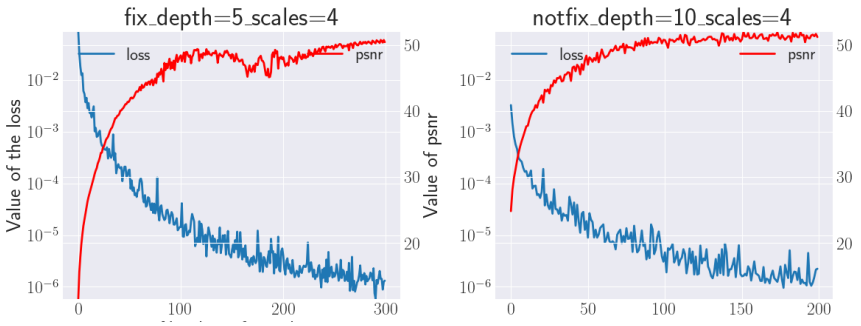
Unfolded Forward-Backward:

Unfolded FB are models with the best results but also with the higher number of parameters. The bigger one has 85 million parameters. They have been trained for 300 epochs for fixed networks and

200 for not-fixed models. for non-fixed ones and 300 for fixed ones. We hadn't time to run the 3 initializations on all settings for fixed models. The results between the two type of models are very similar, but non-fixed is cheaper to learn. from an eye point of view, even the most difficult structures. fixed networks seem to be not very stable except for 3 of them. Those models are expensive to train and interpretation is even harder than with previous networks.

	0	1	2	mean	std	n_params	training_time (h)
FBnotfix_depth=10_scales=2	48.74	48.58	48.94	48.75	0.15	4460190	5.93
FBnotfix_depth=10_scales=3	51.16	51.14	51.42	51.24	0.13	20707230	7.87
FBnotfix_depth=10_scales=4	51.95	52.64	52.06	52.22	0.3	85641630	6.88
FBnotfix_depth=5_scales=2	48.51	48.65	48.45	48.54	0.08	2230095	4.87
FBnotfix_depth=5_scales=3	50.71	50.93	51.15	50.93	0.18	10353615	6.21
FBnotfix_depth=5_scales=4	51.79	52.6	52.32	52.24	0.34	42820815	7.7
FBfix_depth=10_scales=2	48.08	47.95	48.28	48.1	0.14	446019	21.98
FBfix_depth=10_scales=3	50.75	50.67	50.94	50.79	0.11	2070723	31.66
FBfix_depth=10_scales=4	51.39	51.29	51.2	51.29	0.08	8564163	34.55
FBfix_depth=5_scales=2	49.03	49.05	48.86	48.98	0.09	446019	24.3
FBfix_depth=5_scales=3	51.62	51.52	51.75	51.63	0.09	2070723	30.49
FBfix_depth=5_scales=4	51.59	51.54	52.04	51.72	0.22	8564163	32.26

(a)



(b)

Figure 14: (a). Summary of the quantitative results obtained with UnfoldedFB models. (b). Evolution of PSNR and the loss function through epochs.

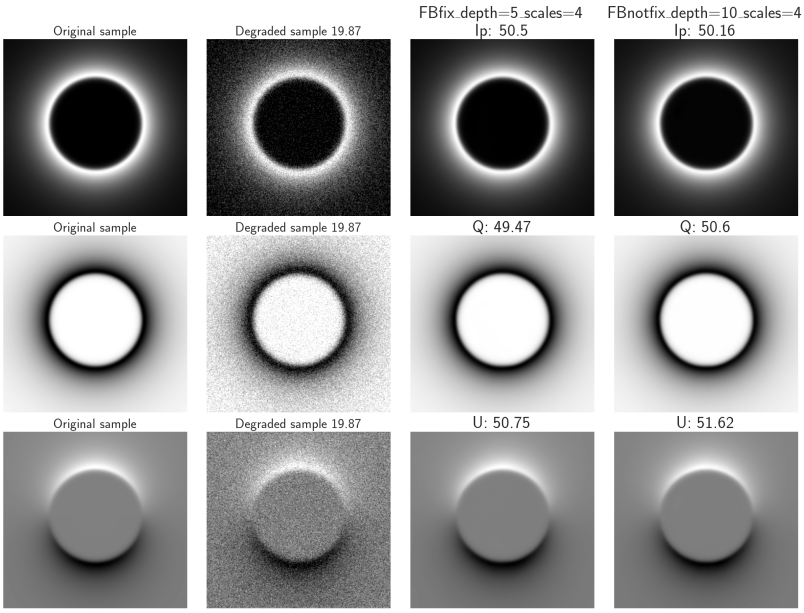


Figure 15: Reconstructed sample with unfolded FB

PnP

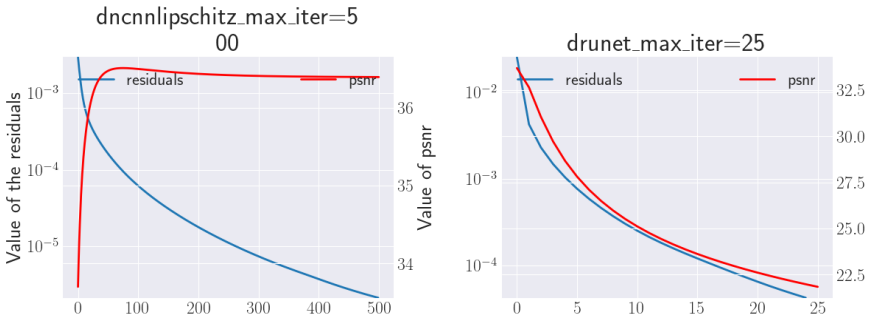
We employed three types of pre-trained denoising neural networks that were not specifically designed for circumstellar environments. Each configuration was tested for either 25 or 500 iterations using Proximal

Gradient Descent (PGD). While the heuristic of Plug-and-Play (PnP) implies running iterations until convergence, we opted for an intermediate test, as suggested in [22].

The most promising results were achieved with the DNCNN model, particularly when it was enforced to be firmly non expensive. However, other networks exhibited divergence, underlining the significance of firmly contracting denoiser, to obtain convergence guarantees, as established in [16]. This lack of convergence can be observed in the reconstructed data of the DRUnet model with 500 iterations, as illustrated in Figure 17.(b). Moreover, even the reconstructed sample of DNCNN Lipchitz is not totally satisfying. Furthermore, the reconstructed sample of DNCNN with Lipschitz constraints doesn't perform better than variational methods in terms of PSNR. However, we can notice that PnP does not suffer from the patch effect observed in Condat-Vũ. Instead, PnP exhibits some unwanted textures in its reconstructions. This could be attributed to the fact that the denoiser used in PnP was not specifically trained on the type of data we are evaluating. Additionally, DNCNN, has not as much parameters as DRUnet, if the Lipschitzianity were enforced we would expect that DRUnet would perform better than DNCNN.

	mean	std	n_params	training_time (h)
dncnn_max_iter=25	34.32	0.0	668227	0.01
dncnn_max_iter=500	23.97	0.0	668227	0.19
dncnnlipschitz_max_iter=25	36.04	0.0	668227	0.0
dncnnlipschitz_max_iter=500	36.33	0.0	668227	0.13
drunet_max_iter=25	21.08	0.0	32640960	0.04
drunet_max_iter=500	-107.65	0.0	32640960	0.85

(a)



(b)

Figure 16: (a). .Summary of the quantitative results obtained with PnP. (b). Evolution of PSNR and the residuals through epochs.

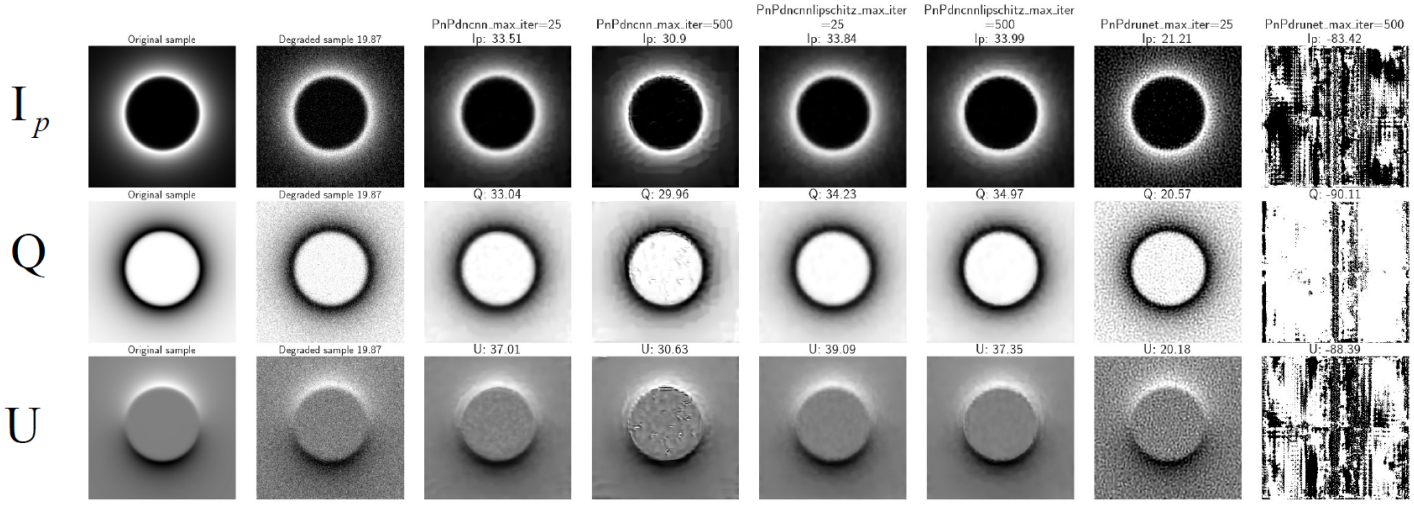


Figure 17: Reconstructed sample with PnP.

Comparisons

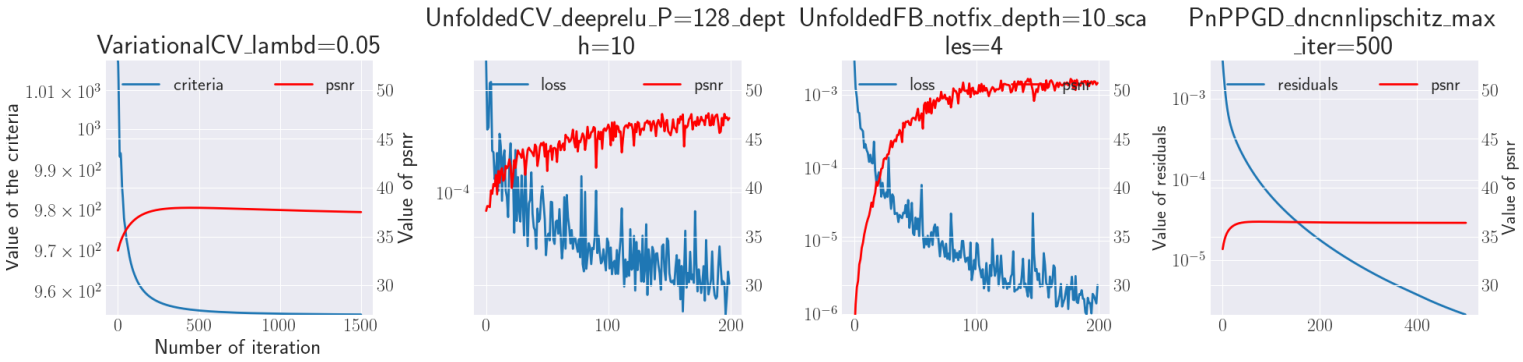
In this last section of experiments on the synthetic dataset, we compare the best model of each type of model.

PnP performs poorly compared to other models. On the other hand, the Unfolded Condat-Vũ surpasses Condat-Vũ. This is obvious on the Figure 19.(b) which is difficult to reconstruct, due to little structures. Although Unfolded Condat-Vũ exhibits some imperfections in the reconstruction of this complex image, it achieves remarkable results, approaching a level of quality that can be considered close to perfection on simpler images like those in Figure 19.(a). However, Unfolded Forward Backward surpasses all other models and excels to reconstruct even the most challenging images.

The correlation between model performance and the number of parameters is strong, as the Figure 20suggests. Thus increasing model complexity generally leads to better results. However, when the number of parameters grows, interpretability diminishes.

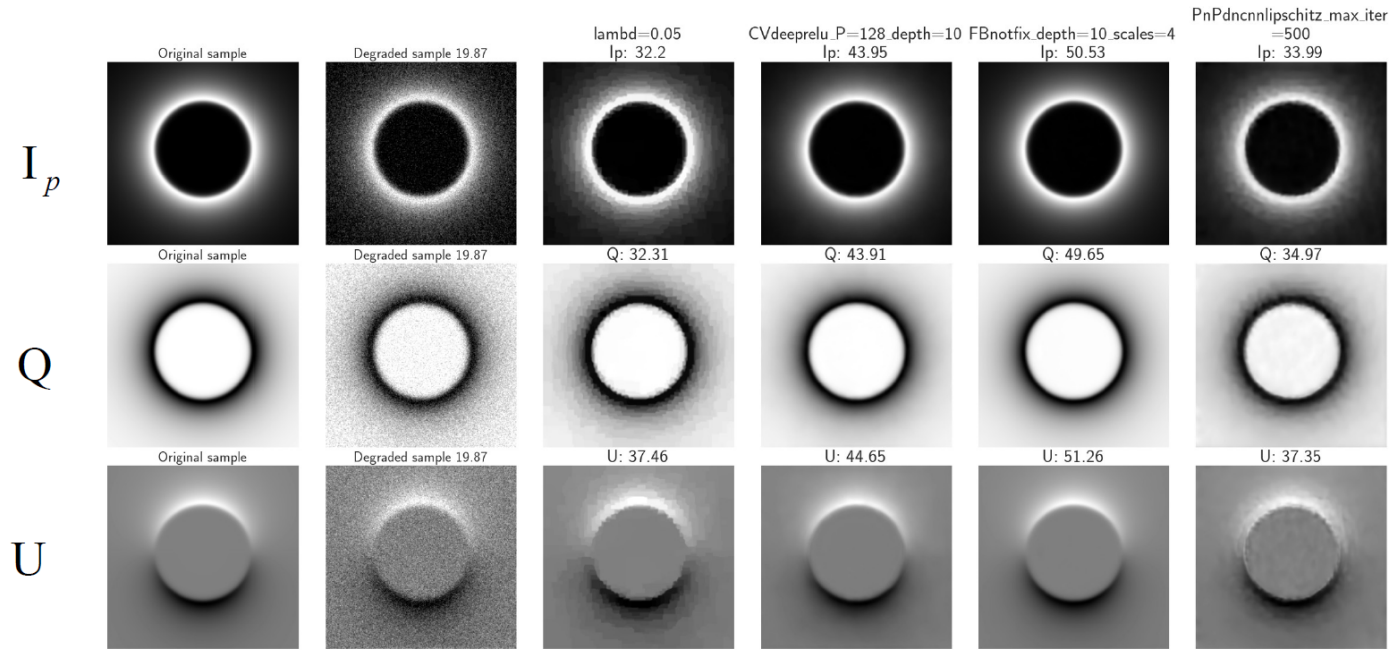
	0	mean	std	n_params	training_time (h)
lambd=0.05	37.5	37.5	0.0	0	0.06
FBnotfix_depth=10_scales=4	52.64	52.64	0.0	85641630	6.82
CVdeeprelu_P=128_depth=10	45.44	45.44	0.0	1509140	2.21
PnPdncnnlipschitz_max_iter=500	36.33	36.33	0.0	668227	0.13

(a)

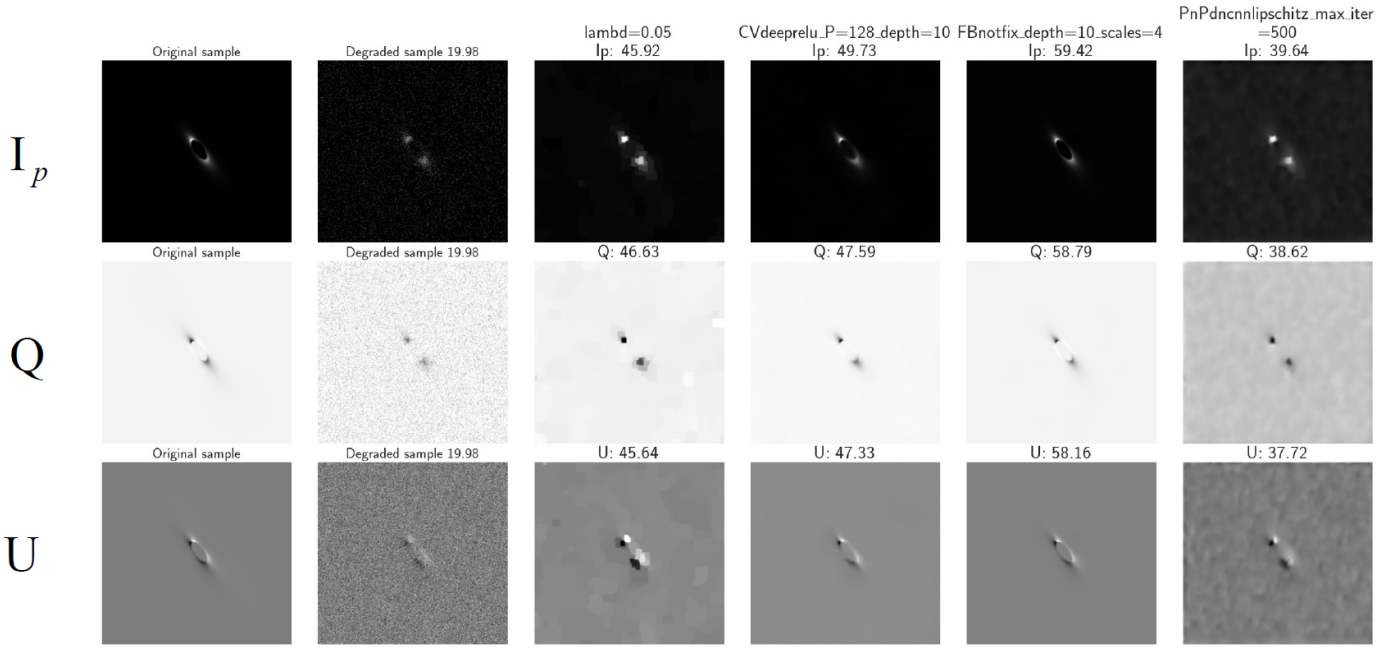


(b)

Figure 18: (a). Summary of the quantitative results obtained with best models. (b). Evolution of PSNR.



(a)



(b)

Figure 19: Reconstructed samples with best models.

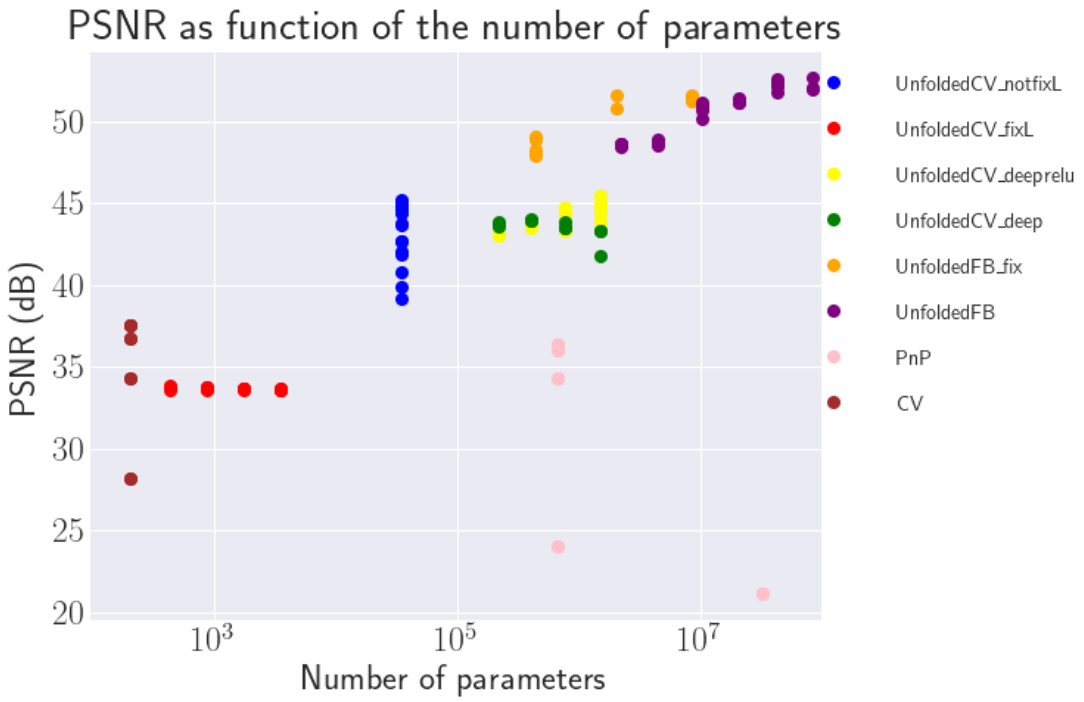


Figure 20: PSNR in function of the number of parameters of the models. CV as 0 parameters. The hyperparameters of UnfoldedCV have been selected to maintain a constant number of parameters while investigating the impact of high numbers of layers or a large number of filters on performance.

32

IV.2 Pre-reconstructed data

Pre-reconstructed dataset is generated based on DDIT data with additional parameter I_u^{star} to obtain Jones parameters (I_u, I_p, θ), like with synthetic dataset. Parameters (I_p, Q, U) are derived from Jones parameters and are blurred with the same PSF A as before. The model developed in [6] produced observations d from blurred parameters (I_p, Q, U). The double difference method produced $y = T^{\dagger}d$, the pre-reconstruction of observation d.

This dataset has been normalized in a different way than the previous one because our previous normalization has been done based on the biggest and lowest value of each data, which is not a good practice to be able to handle new data for which we don't know the ground truth. That means, we did unrealistic experiments, but it was a good toy model to test our architectures. The pre-reconstruted data have been normalized channel wise with the highest and lowest values over the whole training dataset. Training dataset contains 573 images and the test dataset 102 images. With an average PSNR of 35.8.

This dataset can be more difficult to deal with because there are big differences between values of certain degraded images and a disparity of amplitude in the value of the pixels as indicated by peaks and stairs value level of channel U in the Figure 21.(a).

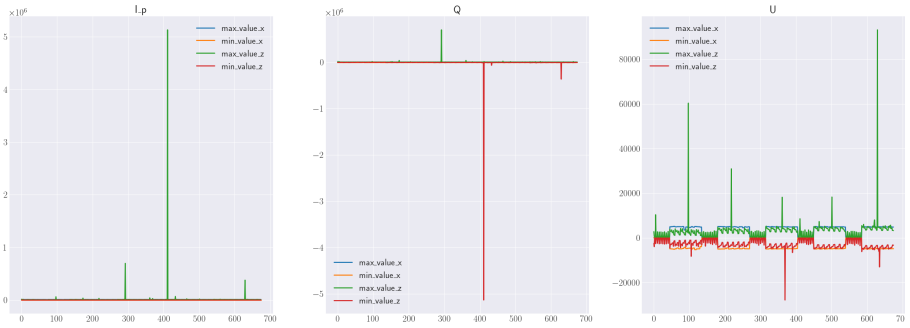


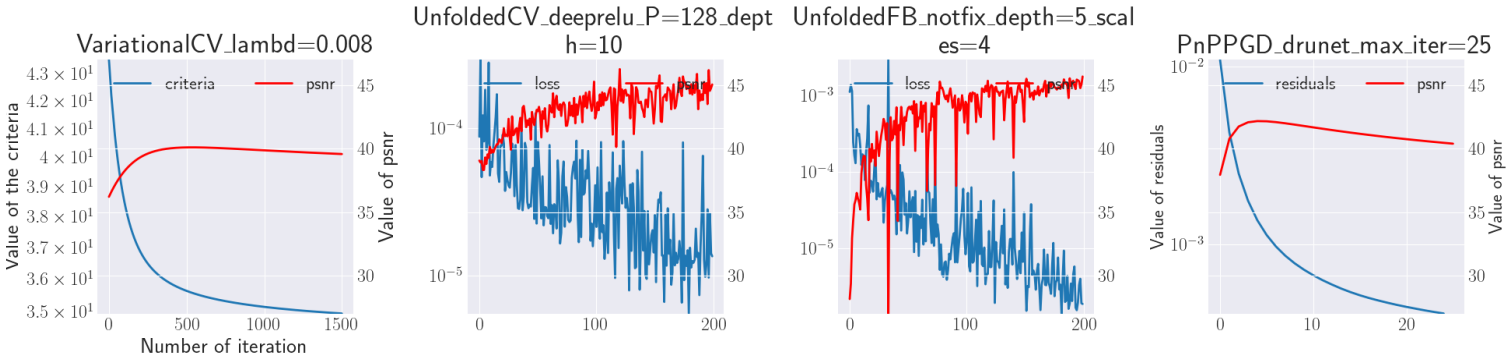
Figure 21: Minimum and maximum pixel values of the 675 original images (x) and degraded (z) on the three channels.

Comparisons:

We compare the best model of each type. More detailed results are displayed in Appendix VI.6. PnP with DRUNet achieves similar results to the best Condat-Vũ with cheaper evaluation. We can see on the PnP with DRUNet curve that it could be way better if we stopped iterations to 7 or 8 instead of 25. Condat-Vũ outperforms all other models on channel U of the image of the Figure 23.(b). However, those performances on the whole dataset are poor compared to performances achieved by deep networks. The Figure 24 highlights the correlation between the number of parameters and the PSNR. It is interesting to notice that this correlation doesn't occur with unfolded CV deep, but when a non-linearity between the two convolutions is added, which corresponds to deep-relu model, it produces a clean improvement according to the number of parameters.

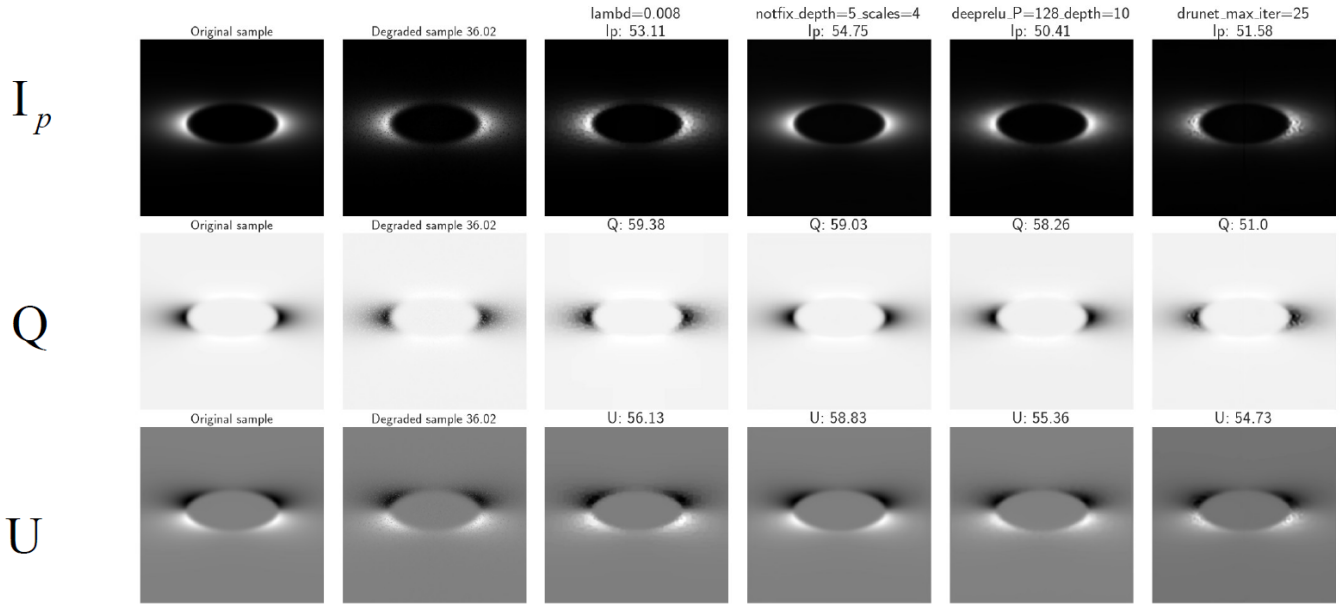
	0	mean	std	n_params	training_time (h)
lambd=0.008	39.57	39.57	0.0	0	0.06
FBnotfix_depth=5_scales=4	52.59	52.59	0.0	42820815	6.86
CVdeeprelu_P=128_depth=10	48.62	48.62	0.0	1509140	5.29
PnPdrunet_max_iter=25	39.4	39.4	0.0	32640960	0.04

(a)

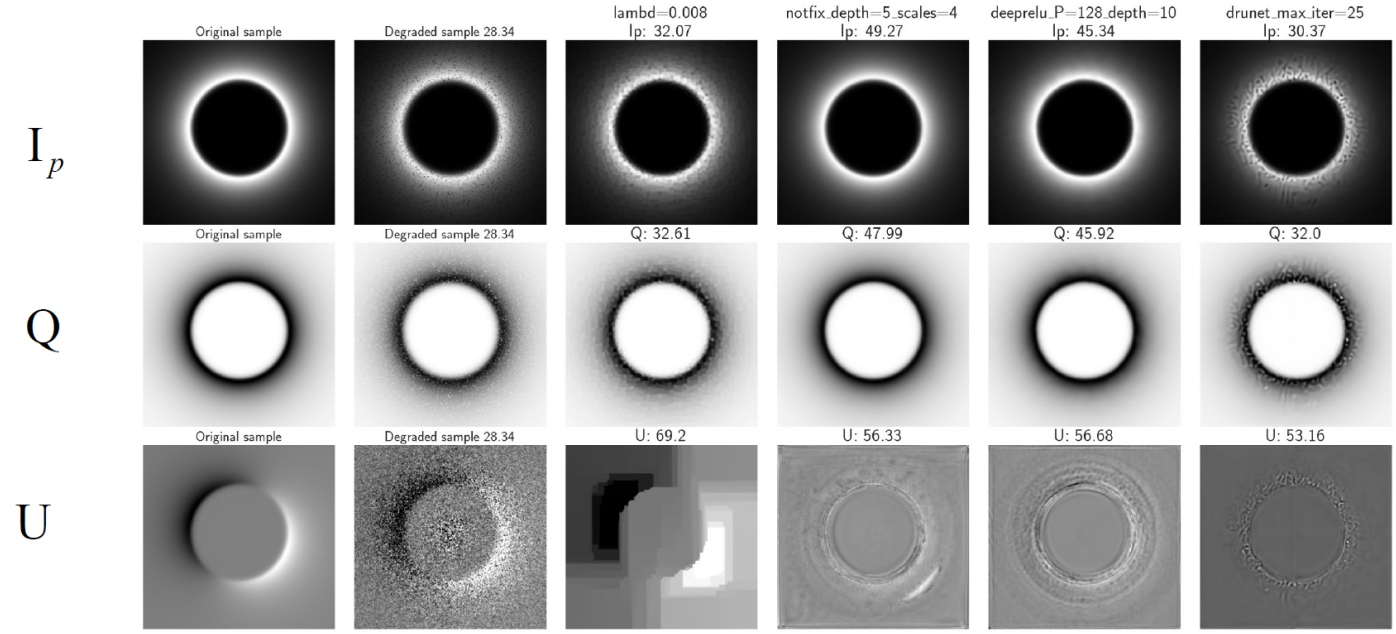


(b)

Figure 22: (a). Summary of the quantitative results obtained with the bests models. (b). Evolution of PSNR.



(a)



(b)

Figure 23: Reconstructed samples with best models.

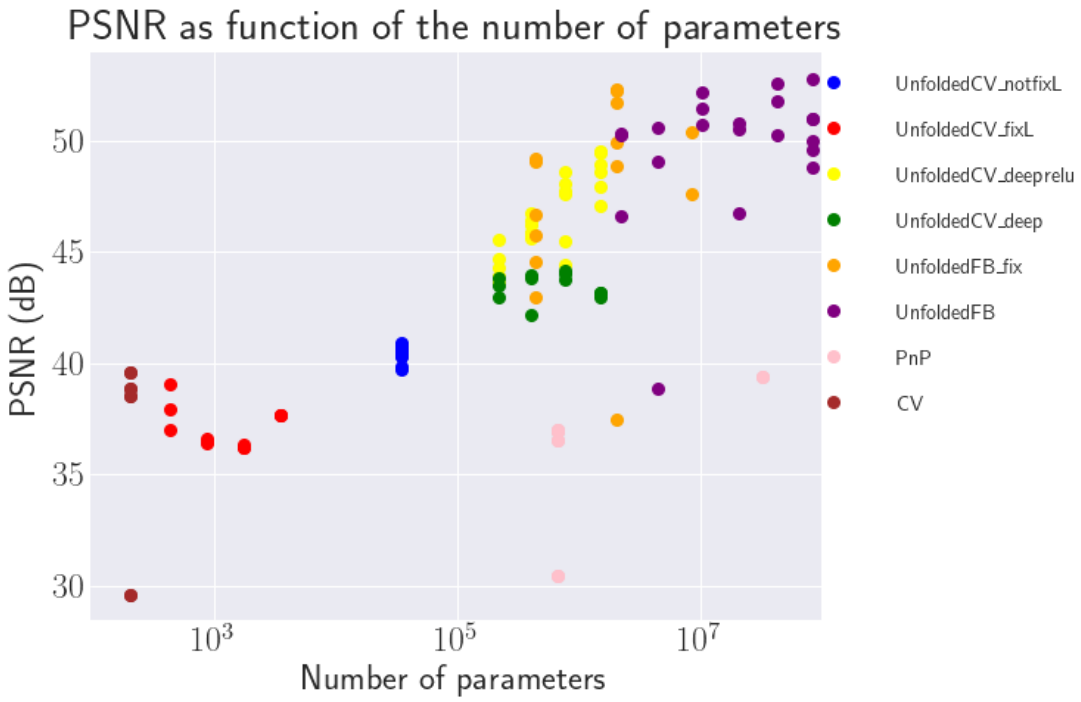


Figure 24: PSNR in function of the number of parameters of the models. CV has 0 parameters. The hyperparameters of UnfoldedCV have been selected to maintain a constant number of parameters while investigating the impact of high numbers of layers or a large number of filters on performance.

V. Conclusion and perspectives

In this report, we propose a study of a wide range of image reconstruction methods by combining the variational methods with deep learning. This approach leverages the strenghts of both fields, the physics of the variational methods, and the recent advancement in deep learning. The objective has been to enhance the quality of synthetic images depicting circumstellar environments by addressing issues related to blurring and noise. These investigations were carried out using two distinct datasets, and we compared the outcomes with those obtained from a baseline Total Variation minimization model, inspired by an existing model known as RHAPSODIE.
The initial approach we pursued is called Unfolded approach which consists of using iterative schemes to solve variational problems with a low number of iterations and parametrize parts to be learnable. The second approach comes from an interpretation of the proximal operator used in iterative methods, which can also be interpreted as a denoising function. In this approach, we pre-trained a denoiser and replaced the proximal operator with it. However, it's worth noting that this particular category of methods did not yield satisfactory results. To enhance these models, potential strategies include training denoisers on the train dataset we aim to reconstruct, enforcing Lipschitz continuity, or terminating iterations when the reconstruction reaches an optimal state could be explored.

On the other side, the unfolded models exhibited remarkable performances compared to traditional variational methods. We observed a strong correlation between the number of model parameters and their performance. The most successful models diverged significantly from the initial objective, which was to minimize a specific objective function encompassing data fidelity and a prior. This situation highlights a trade-off between interpretability and performance.

Moving forward, the next phase involves applying the most effective architectures to real data, which present greater complexity due to their structural characteristics, as described in the Section II. We necessitate an unsupervised learning approach with those data for which we lack ground truth. To address this challenge, it is considered to use a loss function such as Stein's Unbiased Risk Estimate, coupled with a prior to capture the symmetries in circumstellar environment images. A recent approach known as equivariant imaging could be used.

VI. Appendix

VI.1 Definitions:

Definition VI.1.1: (Non-expansive Operator)

ϕ is said to be non-expansive if it is $1-$Lipschitz continuous.

Definition VI.1.2: ($C_{\zeta}^{1,1}(\mathcal{H})$)

Let $\mathcal{H}$ be a Hilbert space. Let $\zeta \geq 0$. This denotes the class of functions $f : \mathcal{H} \to \mathbb{R}$ such that f is Gâteaux-differentiable in $\mathcal{H}$, and $\nabla f : \mathcal{H} \to \mathcal{H}$ is ζ-Lipschitz.

Definition VI.1.3: (Cocoercive Operator)

Let $\mathcal{H}$ be a Hilbert space. $M : \mathcal{H} \to \mathcal{H}$ is called cocoercive if there exists $\mu' > 0$ such that $\langle T(x) - T(y) \mid x - y \rangle \geq \mu' ||T(x) - T(y)||^2$ for all $x, y \in \mathcal{H}$.

Definition VI.1.4: (μ-Averaging Operator)

Let $\mathcal{H}$ be a Hilbert space. Let $\mu \in]0, 1]$. $\phi : H \to \mathcal{H}$ is called μ-averaging if for all $x \in \mathcal{H}$ and all $y \in \mathcal{H}$, $||\phi x - \phi y||^2 \leq ||x - y||^2 - (\frac{1-\mu}{\mu})||(Id - \phi)x - (Id - \phi)y||^2$.

We will say that ϕ is firmly non-expansive if Φ is $\frac{1}{2}$-averaged, and ϕ is non-expansive if Φ is 1-averaged.

Definition VI.1.5: ($\Gamma_0(\mathcal{H})$)

This refers to the class of functions $f : \mathcal{H} \to]-\infty; +\infty]$ such that f is proper, convex, and lower semi-continuous.

This class allows us to work with both non-smooth and smooth sets and functions.

VI.2 Minimizing a function:

Let $\mathcal{H}$ be a Hilbert space and $h : \mathcal{H} \longrightarrow \mathbb{R}$ a proper function. When h is not differentiable, finding for a minimizer $\widehat{x}$ cannot be done with Fermat's rule which states that $\nabla h(\widehat{\boldsymbol{x}}) = 0$. Thus we need to define tools for searching for minima in the non-differentiable case: the concepts of subdifferential and proximal operator, which extend the notion of gradient and gradient descent to the general case.

Definition VI.2.1:

Let $h : \mathcal{H} \to \mathbb{R}$ be a proper function, meaning that its domain is not empty. Then its subdifferential $\partial h : \mathcal{H} \to 2^{\mathcal{H}}$ is given by:

$$\forall x \in \mathcal{H} \quad \partial h(\boldsymbol{x}) = \{\boldsymbol{u} \in \mathcal{H} \quad \text{such that} \quad \forall y \in \mathcal{H}, \langle y - x, u \rangle + h(\boldsymbol{x}) \leq h(y)\}$$

With this definition of subdifferential, Fermat's rule can be written :

$$\widehat{\boldsymbol{x}} \in \underset{\boldsymbol{x}\in\mathcal{H}}{\operatorname{Arg\,min}}\, h(\boldsymbol{x}) \Leftrightarrow 0 \in \partial h(\widehat{\boldsymbol{x}}).$$

Whether in the differentiable or non-differentiable case, the problem of finding 0 of the gradient or subdifferential of h can be seen as searching for a fixed point of a certain function. Indeed, let $\tau \geq \mathbf{0}$:

$$\begin{aligned}\mathbf{0} \in \partial_h(\widehat{\boldsymbol{x}}) &\Leftrightarrow \widehat{\boldsymbol{x}} - \widehat{\boldsymbol{x}} \in \tau\partial h(\widehat{\boldsymbol{x}}) \\ &\Leftrightarrow \widehat{\boldsymbol{x}} \in (\mathrm{Id} - \tau\partial h)(\widehat{\boldsymbol{x}})\end{aligned}$$

This relation gives incentives to use sub-gradient descent algorithm, but for a non-differentiable function, this one requires a decreasing step size and is not stable enough [15]. For that reason, an implicit version of sub-gradient descent has been proposed:

$$x^{[t]} \in (\mathbf{Id} - \tau\partial h)\left(x^{[t+1]}\right)$$

Such a scheme corresponds to finding the fixed points of the proximal operator of the function h, whose definition is given by:

Definition VI.2.2:

Let $h : \mathcal{H} \to \mathbb{R}$ be a proper, convex, and lower semi-continuous function, and let $\tau \geq 0$. Then, its proximal operator is given by:

$$\forall x \in \mathcal{H} \quad \operatorname{prox}_{\tau h}(x) = \underset{y\in\mathcal{H}}{\operatorname{argmin}} \frac{1}{2\tau}\|x - y\|^2 + h(y).$$

We then have the relation:

Proposition VI.2.1:

With the same assumptions as in last definition

$$\forall x \in \mathcal{H} \quad \widehat{y} = \operatorname{prox}_{\tau\mathcal{G}}(x) \Leftrightarrow x - \widehat{y} \in \tau\partial_{\mathcal{G}}(\widehat{y}).$$

Proof.

$$\begin{aligned}\widehat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{Arg\,min}} \left\{\tau h(\mathbf{y}) + \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|^2\right\} &\Leftrightarrow \{\mathbf{0}\} = \partial\left(\tau h(\widehat{\mathbf{y}}) + \frac{1}{2}\|\widehat{\mathbf{y}} - \mathbf{x}\|^2\right) \\ &\Leftrightarrow \{\mathbf{0}\} = \partial(\tau h(\widehat{\mathbf{y}})) + (\widehat{\mathbf{y}} - \mathbf{x}) \\ &\Leftrightarrow \{\mathbf{x}\} = (I + \tau\partial h)\widehat{\mathbf{y}} = \widehat{\mathbf{y}} + \tau\partial h(\widehat{\mathbf{y}})\end{aligned}$$

□

Therefore, if $\widehat{\boldsymbol{x}}$ is a fixed point of $\text{prox}_{\tau h}$, denoted as Fix, we have:

$$\begin{aligned}\widehat{\boldsymbol{x}} \in \text{Fix}_{\text{prox}_h} &\Leftrightarrow \widehat{\boldsymbol{x}} = \text{prox}_h(\widehat{\boldsymbol{x}}) \\ &\Leftrightarrow \widehat{\boldsymbol{x}} - \widehat{\boldsymbol{x}} \in \partial h(\widehat{\boldsymbol{x}}) \\ &\Leftrightarrow 0 \in \partial h(\widehat{\boldsymbol{x}}) \\ &\Leftrightarrow \widehat{\boldsymbol{x}} \in \underset{\boldsymbol{x}\in\mathcal{H}}{\text{Arg min}}\, h(\boldsymbol{x}).\end{aligned}$$

Example VI.2.1: (ℓ_1 case)

The subdifferential of the ℓ_1 norm multiplied by a factor $\lambda \geq 0$ is given by:

$$\partial\lambda\|x\|_{\ell_1} = \bigotimes_{n\in\{1,\dots N\}} \begin{cases} -\lambda \text{ if } x_n < 0, \\ [-\lambda, \lambda] \quad \text{if } x_n = 0, \\ \lambda \quad \text{if } x_n > 0, \end{cases}$$

and its proximal operator is given by the soft-thresholding function:

$$\forall x \in \mathbb{R}^N, \quad \text{prox}_{\|\cdot\|_{\ell_1}}(x) = \left(\max\left\{0, 1 - \frac{\lambda}{|x_n|}\right\} x_n\right)_{n\in\{1,\dots N\}}$$

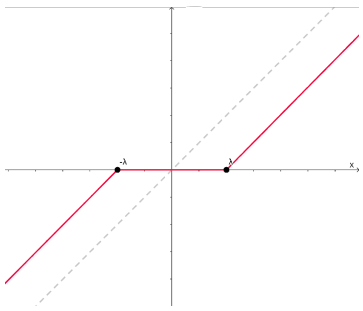


Figure 25: Soft-thresholding function

Our minimization problem becomes a fixed-point problem. Then we can use the fixed point theorems to solve them.

Theorem VI.2.1: (Banach-Picard)

Let $\mathcal{H}$ be a Hilbert space. Let $\omega \in [0, 1[$ and $\phi : \mathcal{H} \to \mathcal{H}$ be an ω-Lipschitz map. Let $x_0 \in \mathcal{H}$. For all $k \in \mathbb{N}$, we have $x_{k+1} = \phi(x_k)$.

Then, $Fix(\phi) = \hat{x}$ for $\hat{x} \in \mathcal{H}$ and for all $k \in \mathbb{N}$, $||x_k - \hat{x}|| \leq \omega^k ||x_0 - \hat{x}||$.

Moreover, the sequence of elements $(x_k)_{k\in\mathbb{N}}$ converges **strongly** to $\hat{x}$ with a linear convergence rate of ω.

Theorem VI.2.2:

Let $\mathcal{H}$ be a Hilbert space. Let $\mu \in]0, 1[$ and $\phi : \mathcal{H} \to \mathcal{H}$ be a μ-averaging operator such that $Fix(\phi) \neq \emptyset$.

Let $x_0 \in \mathcal{H}$. For all $k \in \mathbb{N}$, $x_{k+1} = \phi(x_k)$.

Then, the sequence of elements $(x_k)_{k\in\mathbb{N}}$ converges **weakly** to a fixed point of ϕ.

Remark: In finite dimension **Weak** and **strong** convergence are equivalent.

Thanks to those properties we can derive minimization algorithms:

Proposition VI.2.2:

Let $\mathcal{H}$ be a Hilbert space, $h \in \Gamma_0(\mathcal{H})$, and h is ζ-smooth with $\zeta \in]0, 2[$, then the operator $\Phi = \mathrm{Id} - \tau\nabla h$ is $\frac{\zeta}{2}$-averaged.

Then we derive the gradient descent algorithm:

Algorithm 6: (Gradient Descent Algorithm)

Assumption: $h \in \Gamma_0(\mathcal{H})$ and ζ-smooth with $\zeta < 2$
Input : Initial guess $x^0 \in \mathcal{H}$, step size $\tau \in]0, 2\zeta^{-1}[$
Output : Approximation of $\operatorname{Arg\,min}_{x\in\mathcal{H}} h$

for $n = 0, 1, 2, \ldots$ **do**
  $x^{n+1} = x^n - \tau\nabla h(x^n)n$;

Proposition VI.2.3:

Let $\mathcal{H}$ be a Hilbert space, $h \in \Gamma_0(\mathcal{H})$, then $\mathrm{prox}_{\tau h}$ is $\frac{1}{2}$-averaged

Then we derive the proximal point algorithm:

Algorithm 7: (Proximal Point Algorithm)

Assumption: $h \in \Gamma_0(\mathcal{H})$
Input : Initial guess x^0, $\tau > 0$
Output : approximation of $\arg\min_{x\in\mathcal{H}} h$

for $n = 0, 1, 2, \ldots$ **do**
  $x^{n+1} = \mathrm{prox}_{\tau h}(x^n)$;

These algorithms have limitations; they exhibit a low convergence rate. Additionally, for the Gradient Descent algorithm to be applicable, h must be differentiable, and it must implement a prox_h operation at each iteration. The latter involves solving a sub-optimization problem, which is a requirement for the Proximal Point algorithm. This next property will be used to introduce the Proximal Gradient Descent algorithm (see Section III.1), which involves decomposing h into two distinct parts:

Proposition VI.2.4:

Let D be a nonempty subset of $\mathcal{H}$. Let $\alpha_1 \in]0, 1[$ and $\alpha_2 \in]0, 1[$.

Let $\Phi_1 : D \to D$ be α_1-averaged and $\Phi_2 : D \to D$ be α_2-averaged.

Then $\Phi = \Phi_1\Phi_2$ is α-averaged with

$$\alpha = \frac{\alpha_1 + \alpha_2 - 2\alpha_1\alpha_2}{1 - \alpha_1\alpha_2} \in]0, 1[.$$

VI.3 Duality

Definition VI.3.1: (conjugate function)

Let $\mathcal{H}$ be a Hilbert space. Let $f : \mathcal{H} \to]-\infty; +\infty]$.

The function $f^* : \mathcal{H} \to]-\infty; +\infty]$ is called the *Fenchel conjugate* of f, defined for all $u \in \mathcal{H}$ as:

$$f^*(u) = \sup_{x \in \mathcal{H}} \{\langle x \mid u \rangle - f(x)\}.$$

Proposition VI.3.1:

With the same notations as before:

- If $h(x) = f(x - y)$ then $h^*(x) = f^*(x) + y^*x$.
- If $h(x) = f(Ax)$ with A linear and invertible, then $h^*(x) = f^*(A^{-1^*}x)$.
- If $h(x) = f(\lambda x)$ with $\lambda \in \mathbb{R}$, then $h^*(x) = \lambda f(\frac{y}{\lambda})$.

With that definition and properties, we are able to retrieve the equation 9:

- Our **primal problem** is $\underset{x \in \mathcal{H}}{\text{Arg min}}(f(x) + g(Dx))$
- Our **dual problem** is $\underset{u \in G}{\text{Arg min}}(f^*(-D^*u) + g^*(u))$

The advantage of dual algorithms is that the contribution of D is not in g anymore. Indeed, when $g(x) = \lambda||x||_1$ we have:

$$\begin{aligned} g^*(u) &= \iota_{\mathcal{B}_{\infty,\lambda}}(u) \\ &= \begin{cases} 0, & \text{if } \|u\|_\infty \leq \lambda, \\ +\infty, & \text{otherwise,} \end{cases} \end{aligned}$$

and most of the time $f(x) = \frac{1}{2}||y - Ax||_2^2$ thus:

$$\begin{aligned} f^*(x) &= \frac{1}{2}||A^{-1^*}x||_2^2 + y^*A^{-1^*}x \\ \Rightarrow f^*(-D^*u) &= \frac{1}{2}||-A^{-1^*}D^*u||_2^2 + y^*A^{-1^*}D^*u \end{aligned}$$

Finally, we obtain that the dual problem can be expressed as:

$$\underset{u \in G}{\text{Arg min}}(\frac{1}{2}||-A^{-1^*}D^*u||_2^2 + y^*A^{-1^*}D^*u + \iota_{\mathcal{B}_{\infty,\lambda}}(u))$$

Theoreme VI.3.1: (duality theorem)

There exists a primal solution if and only if there exists a dual solution.

In this case, $(\hat{x}, \hat{u})$ are the dual and primal solutions if and only if $-D^*\hat{u} \in \partial f(\hat{x})$ and $D\hat{x} \in \partial g^*(\hat{v})$

If $f(x) = \frac{1}{2}||y - Ax||_2^2$, we can express primal solution with the dual one:

$$-D^*\hat{u} = A^*(y - A\hat{x}) \Rightarrow \hat{x} = (A^*A)^{-1}(A^*y + D^*\hat{u})$$

Goal: Address the Total Variation problem (Equation 3). In this context, we define $\mathcal{R}(x) = \lambda g(Dx)$. The challenge we encountered with PGD was computing the proximity operator of $g \circ D$.

The conventional approach to overcoming such an issue is to solve the dual problem and retrieve the primal solution with the relation $\hat{\boldsymbol{x}} = (A^*A)^{-1}(A^*y + D^*\hat{\boldsymbol{u}})$.

$$\widehat{\boldsymbol{u}} \in \underset{\boldsymbol{u}\in\mathbb{R}^P}{\operatorname{Arg\,min}} \left[\frac{1}{2}||-A^{-1^*}D^*u||_2^2 + y^*A^{-1^*}D^*u + \iota_{\mathcal{B}_{\infty,\lambda}}(u)\right] \tag{9}$$

Where $P = 2L$, A^* represents the adjoint of A and $\iota_{\mathcal{B}_{\infty,\lambda}}$ the indicator function of the unity ball of radius λ. equation (9) is proven and we developed on duality in Appendix VI.3. However, this formulation can be problematic when the linear operator A is not invertible.
In this case, we generally solve the saddle point problem, which gives the primal-dual solution :

$$\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{u}} \in \underset{\boldsymbol{x}\in\mathbb{R}^L}{\operatorname{Arg\,min}} \underset{\boldsymbol{u}\in\mathbb{R}^P}{\operatorname{Arg\,max}} \left[\frac{1}{2}||y - Ax||_2^2 + \lambda g^*(\frac{\boldsymbol{u}}{\lambda}) + < D\boldsymbol{x}|\boldsymbol{y} >\right] \tag{10}$$

Duality allows us to reformulate a problem into another one, which is often easier to handle when the function we want to minimize is of the form $h(x) = f(x) + g(Dx)$, with $\mathcal{H}, \mathcal{G}$ are Hilbert spaces and $L \in \mathfrak{B}(\mathcal{H}, \mathcal{G}), f \in \Gamma_0(\mathcal{H}), g \in \Gamma_0(\mathcal{G}), \lambda \in \mathbb{R}$.

VI.4 RHAPSODIE:

In the Section III.1, we initially assumed that the noise ϵ follows an isotropic distribution. However, this assumption is restrictive. In practice, the calculations for the data fidelity term can be adapted to a more general case. This broader framework is adopted by the RHAPSODIE model, which was developed in [6] and [7].
It involves a weighted ℓ_2-norm in the data fidelity term. RHAPSODIE employs a variable metric Forward-Backward scheme [2] to minimize the sum of three functions:

- The weighted data fidelity term.
- Regularization, which is the combination of two hyperbolic edge-preserving regularizations, one for the image I and the other for the pair (U, Q). Each regularization term has two hyperparameters and can be seen as analogous to an $\ell_{1,2}$-norm.
- A third function, acting as a constraint with the epigraphic condition $I \geq \|(Q, U)\|_2^2$.

The sum of the data fidelity and regularization terms is differentiable and is updated using gradient-based methods. On the other hand, the constraint is enforced through an indicator function which is activated with its proximal operator. This overall optimization process results in a standard projected gradient descent [3].

VI.5 Neural Networks

Generality of Neural Networks:

A neural network is a predictive function $d_\theta : \mathcal{H} \to \mathcal{G}$, such that:

$$d_\theta(u) = \eta^{[K]}(W^{[K]} \dots \eta^{[1]}(W^{[1]}u + b^{[1]}) \cdots + b^{[K]}) \tag{11}$$

Where:

- Linear operators representing weights are $W^{[1]}, \dots, W^{[K]}$.
- Activation functions are $\eta^{[1]}, \dots, \eta^{[K]}$.
- Bias vectors are $b^{[1]}, \dots, b^{[K]}$.

The parameters of the neural network are $\theta = \{W^{[1]}, \dots, W^{[K]}, b^{[1]}, \dots, b^{[K]}\}$. Commonly, a neural network is a succession of layers $T_k = \eta_k(W_k \cdot + b_k)$. Then we have $d_\theta = T_K \circ \cdots \circ T_1$.

From a dataset $S = \{(x_i, y_i) \in \mathcal{H} \times G \mid i \in \mathcal{I}\}$, the neural network is tasked with predicting x_i based on y_i. Our goal is to have $d_\theta(y_i) \approx x_i$. In the context of this internship, the variable x_i represents the data of interest, while y_i represents data collected and degraded by the Very Large Telescope (VLT).

To achieve this, we define a loss function $\mathcal{L}(\theta)$, which serves as a measure of how well the neural network approximates our data. The objective is to minimize this function. This is the training part. In practice, we use a refinement of the gradient descent algorithm called Adam [11]. The main concept to be able to use gradient descent on such a function is the back-propagation method [12] which permits updating all layers from the last to the first one.

$$\hat{\theta} \in \operatorname*{Arg\,min}_\theta \{\mathcal{L}(\theta)\}$$

There are a lot of data collected nowadays, but we don't always have access to pairs (x_i, y_i). Then we can divide neural networks into two categories:

- We call a network **supervised** when the targets x_i of our dataset are available. This happens when we have some pairs (x_i, y_i). For example, in the case of image recognition, we can label by hand a set of images or if there are existing large datasets like meteorological data. In this case, the classical loss function used is the Mean Squared Error (MSE):

$$\begin{aligned} \mathcal{L}_{MSE}(\theta) &= \mathbb{E}_y[||x - d_\theta(y)||_2^2] \\ &= \frac{1}{\#\mathcal{I}} \sum_{i \in \mathcal{I}} ||x_i - d_\theta(y_i)||^2 \end{aligned}$$

- We call a network **unsupervised** when the targets are not available. Most data collected by humans fall in this case. For example, the data collected by VLT are degraded by instruments and random noise, in this case, we aim to reconstruct images of circumstellar environments without the original one. In this case, we cannot use x_i's to define our loss function. Instead, if we suppose $y \sim \mathcal{N}(x, \sigma^2 \boldsymbol{I})$ we can use Stein's Unbiased Risk Estimate (SURE) which verifies $\mathbb{E}_y[\mathcal{L}_{SURE}(\theta)] = \mathcal{L}_{MSE}(\theta)$:

$$\mathcal{L}_{SURE}(\theta) = \sum_{i \in \mathcal{I}} \|y_i - \Phi f_\theta(y_i)\|_2^2 - \sigma^2 m + 2\sigma^2 div(\Phi \circ f_\theta)(y_i)$$

During this internship, I worked with generated data representing circumstellar environments, giving us access to the ground truth x_i. In future work, our method will be applied to VLT data, as in [7], and the loss function will transition to SURE or a refined version thereof.

UNet architecture:

UNet is a convolutional neural network that was developed for image segmentation [13]. The network consists of a contracting path (encoder) and an expansive path (decoder). During the encoding part, the spatial information is reduced while feature information is increased. The decoder combines the feature and spatial information through a sequence of up-convolutions and concatenations with high-resolution features from the contracting path. The number of of contracting/expensive steps is the hyperparameter called scale. The specificity of UNet is the Skip connections between the encoder and decoder which permits no loss of details of spatial information.

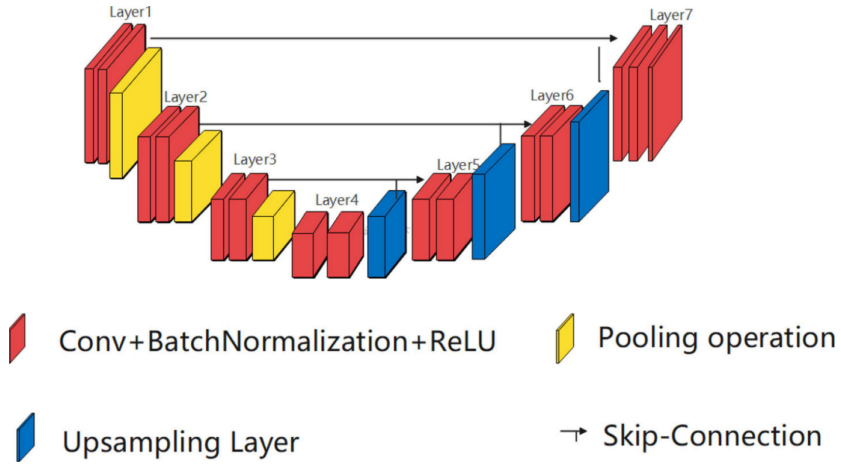


Figure 26: UNet network with scale equal to 3

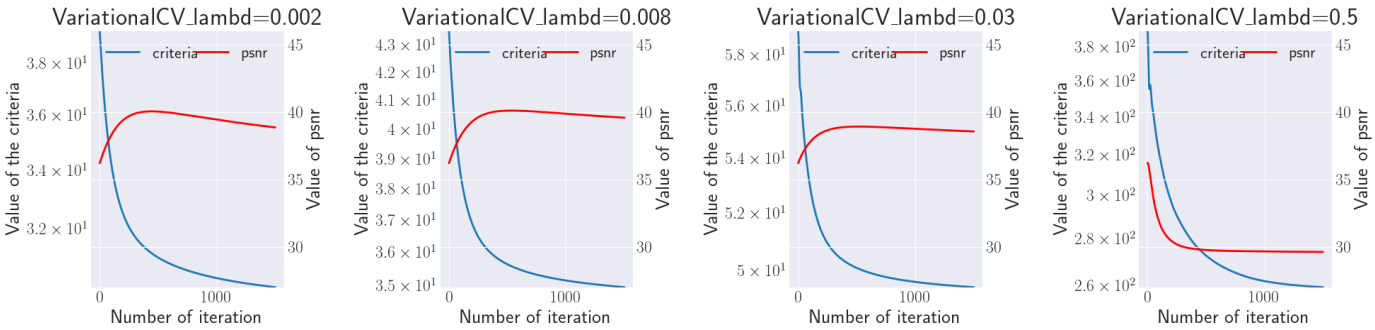
VI.6 Complementary experiment's results on pre-recontructed data:

Variational methods:

The best Condat-Vu has a PSNR of 35.8 dB while the reconstructed has 39.6 dB. With the previous dataset, the delta of PSNR was greater because the data were more homogeneous. We can observe in figure 28 that some images are very noisy and others not. It means that the optimal hyperparameter varies between images. This is why this dataset is more difficult to handle. The second reason to have a few improvements is that the PSNR is in a logarithmic scale, which means that it is way more complicated to gain the same amount of PSNR when it is equal to 20 than when it is equal to 35.
We retrieve the patch effects in the sample displayed in figure 28. The channel U of the right image is difficult to display because the values are in a tin range and if we choose to set the same colors for the same values as the original one, most of the time we obtain a uniform image. Then we choose to let the maximum and minimum values determine the scale of colors which can create aberations in the display even if the PSNR is high.

	0	mean	std	n_params	training_time (h)
lambd=0.002	38.85	38.85	0.0	0	0.06
lambd=0.008	39.57	39.57	0.0	0	0.06
lambd=0.03	38.55	38.55	0.0	0	0.2
lambd=0.5	29.61	29.61	0.0	0	0.06

(a)



(b)

Figure 27: (a). Resume table of CV models. (b). Evolution of PSNR and value of the function to minimize through iterations.

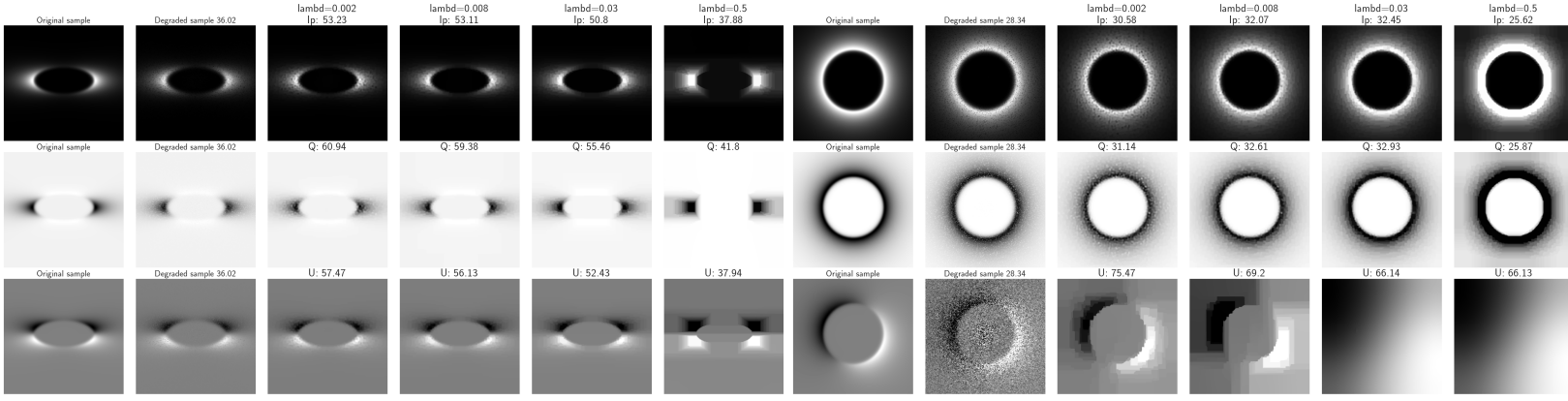


Figure 28: Reconstructed sample with CV.

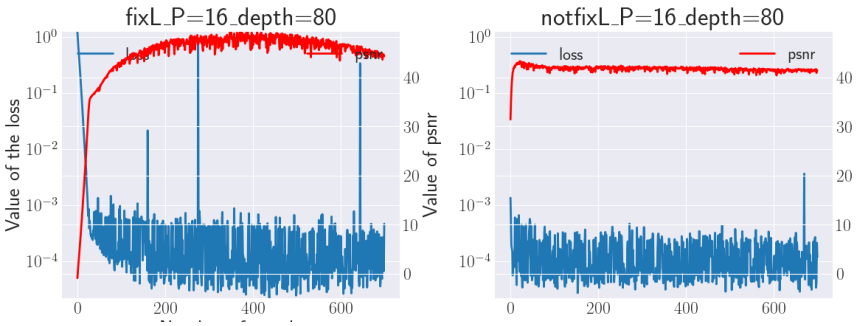
Unfolded Condat-Vu:

Unfolded CV has been trained for 700 epochs and Unfolded CV deep for 200. Even if the Unfolded CV models are very stable, the more the network is deep and has parameters, the more it is unstable and the more PSNR is high. Not fixed models increase on the few first epochs and stay constant for the rest of the train. With the synthetic data, it was fixed models which had this behavior. We also see that the higher the number of parameters is, the higher the PSNR is after the first epoch.

we can see in the ground truth of the I_p channel on figure 32 that it misses a circle inside the ellipse which is present on the degraded image. I don't know why it misses in ground truth, but it is interesting to remark that the best models can forget about it to reach better results where a network with few parameters preserves this part of the image which is supposed to belong to the image. Big neural networks can hallucinate.

	0	1	2	mean	std	n_params	training_time (h)
fixL_P=128_depth=10	37.69	37.63	37.64	37.65	0.03	3458	11.44
fixL_P=16_depth=80	36.99	37.93	39.06	37.99	0.85	434	18.82
fixL_P=32_depth=40	36.5	36.4	36.59	36.5	0.08	866	9.44
fixL_P=64_depth=20	36.21	36.31	36.19	36.24	0.05	1730	13.31
notfixL_P=128_depth=10	39.74	40.32	39.87	39.98	0.25	34580	12.28
notfixL_P=16_depth=80	40.3	40.93	40.62	40.62	0.26	34720	23.78
notfixL_P=32_depth=40	40.85	40.62	40.55	40.67	0.13	34640	6.85
notfixL_P=64_depth=20	40.71	40.7	40.5	40.64	0.1	34600	14.73

(a)

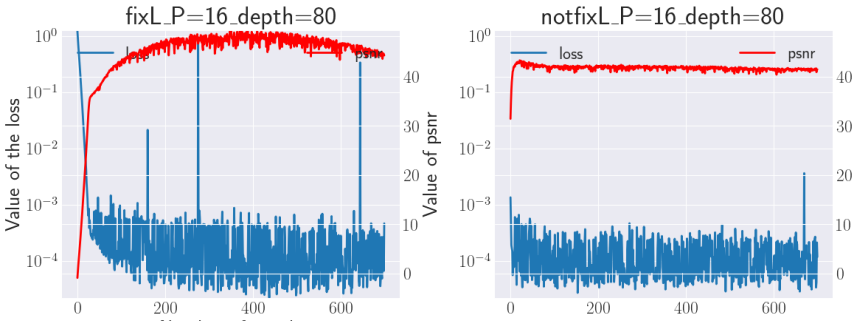


(b)

Figure 29: (a). Resume table of UnfoldedCV models. (b). Evolution of PSNR and the loss function through epochs.

	0	1	2	mean	std	n_params	training_time (h)
deep_P=128_depth=10	43.15	42.94	43.17	43.09	0.1	1509140	2.39
deep_P=16_depth=80	42.99	43.48	43.8	43.42	0.33	219040	4.41
deep_P=32_depth=40	42.15	43.98	43.8	43.31	0.82	403280	2.38
deep_P=64_depth=20	44.0	43.73	44.13	43.95	0.17	771880	2.07
deeprelu_P=128_depth=10	47.94	48.62	47.09	47.88	0.63	1509140	5.29
deeprelu_P=16_depth=80	43.91	44.28	43.64	43.94	0.26	219040	2.46
deeprelu_P=32_depth=40	46.19	45.81	45.63	45.88	0.23	403280	1.48
deeprelu_P=64_depth=20	44.45	47.59	48.04	46.69	1.6	771880	5.42

(a)



(b)

Figure 30: (a). Resume table of UnfoldedCV deep models. (b). Evolution of PSNR and the loss function through epochs.

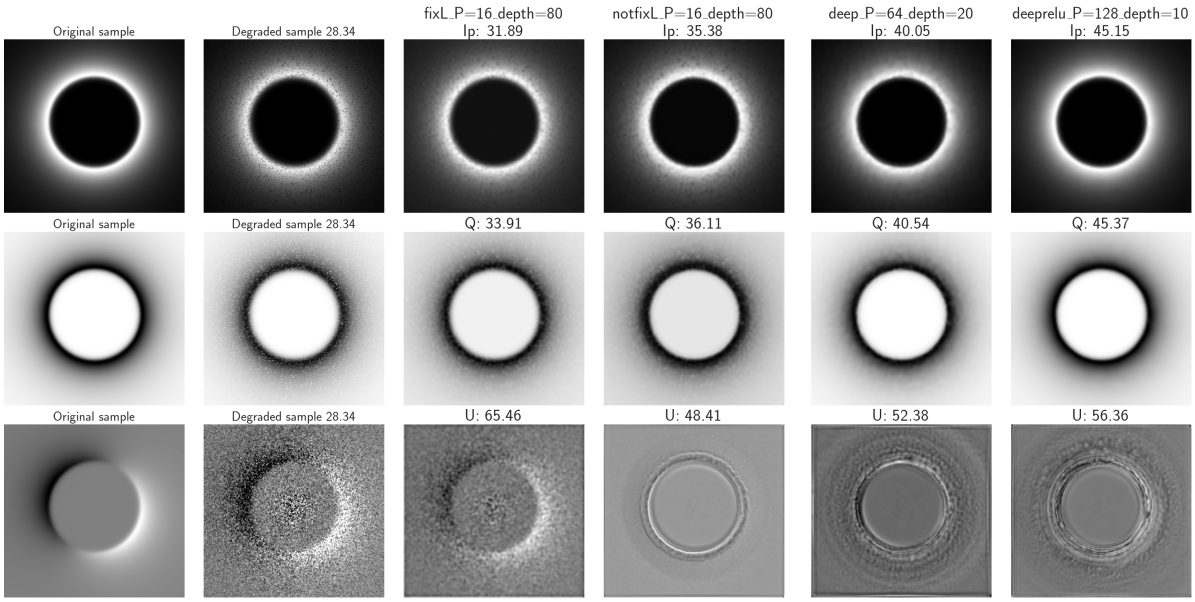


Figure 31: Reconstructed sample with unfolded CV deep.

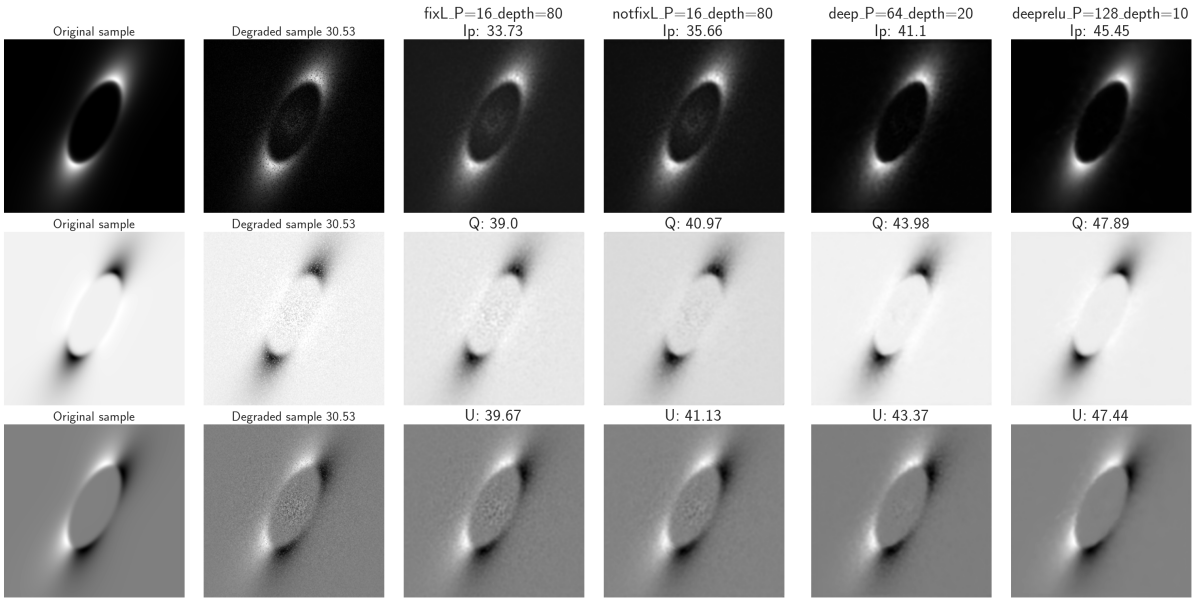


Figure 32: Reconstructed sample with unfolded CV deep.

Unfolded Forward-Backward:

Fixed models have been trained for 300 epochs and not fixed ones for 200. The last 4 models missing haven't been trained before the end of the writing of the report. The high training time for fixed models is due to a higher number of epochs and maybe some instability with GPU or some optimal coding. Nevertheless, those models reach good performances, even if some train obtain a low PSNR, like the first initialization of notfix depth=10 and scales=2 and the last one of fix depth=10 and scale=3. Surprisingly, either for fixed and not fixed models, for any scale value, and in contrast

with the synthetic dataset, the version with depth=5 outperforms the version with depth equal to 10. The reconstruction, for best models of both type, is greater than 50 of PSNR, which is close to undistinguishable from the ground truth from an eyes point of view.

Waiting for train

	0	1	2	mean	std	n_params	training_time (h)
notfix_depth=10_scales=2	38.88	50.55	49.07	46.17	5.19	4460190	3.6
notfix_depth=10_scales=3	50.79	50.53	46.72	49.35	1.86	20707230	5.31
notfix_depth=10_scales=4	48.8	49.96	51.01	49.92	0.9	85641630	6.78
notfix_depth=5_scales=2	50.29	46.62	50.26	49.06	1.72	2230095	10.19
notfix_depth=5_scales=3	52.14	51.41	50.69	51.41	0.59	10353615	6.87
notfix_depth=5_scales=4	51.79	52.59	50.26	51.55	0.97	42820815	6.64
fix_depth=10_scales=2	44.55	46.68	42.97	44.73	1.52	446019	20.09
fix_depth=10_scales=3	49.95	48.84	37.45	45.41	5.65	2070723	25.25
fix_depth=10_scales=4	47.58			47.58	0.0	8564163	36.39
fix_depth=5_scales=2	49.21	49.06	45.76	48.01	1.59	446019	23.98
fix_depth=5_scales=3	51.7	52.32	52.24	52.09	0.28	2070723	19.92
fix_depth=5_scales=4	50.36			50.36	0.0	8564163	41.33

Figure 33: Resume table of UnfoldedFB models

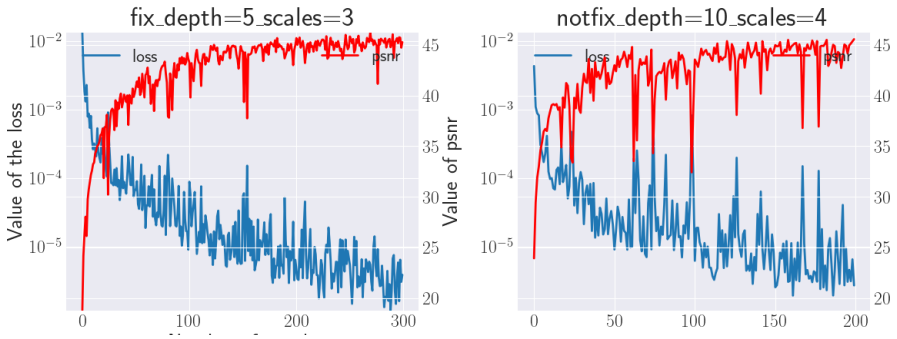


Figure 34: Evolution of PSNR and the loss function through epochs

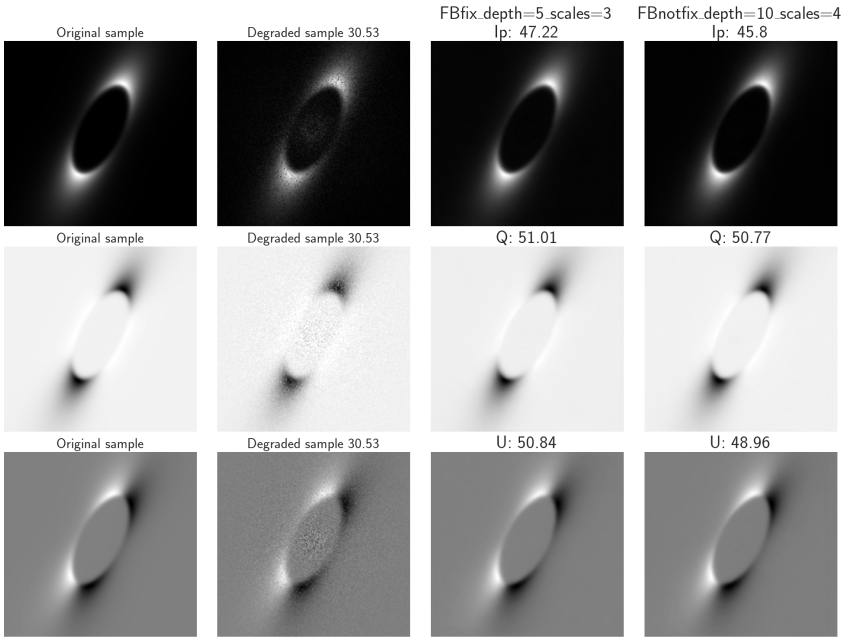


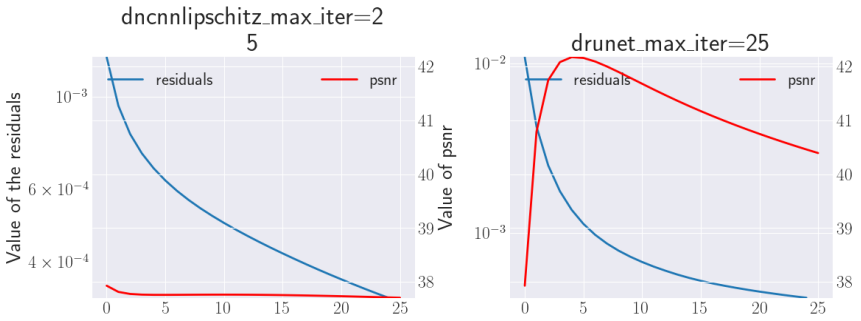
Figure 35: Reconstructed sample with unfolded FB

PnP:

The only model that converges is DNCNN Lipschitz as with the synthetic data. The performances of this convergent PnP is worse than the ones obtained with only 25 iterations with DRUNet, but if we continue the iterations, DRUNet diverges. PnP seems slightly worse than Condat-Vu in terms of PSNR, but, reconstructed images don't suffer from the patch effect and it is cheaper to evaluate.

	0	mean	std	n_params	training_time (h)
dncnn_max_iter=25	36.51	36.51	0.0	668227	0.01
dncnn_max_iter=500	30.41	30.41	0.0	668227	0.18
dncnnlipschitz_max_iter=25	37.01	37.01	0.0	668227	0.01
dncnnlipschitz_max_iter=500	36.91	36.91	0.0	668227	0.19
drunet_max_iter=25	39.4	39.4	0.0	32640960	0.04
drunet_max_iter=500	-42.08	-42.08	0.0	32640960	0.79

(a)



(b)

Figure 36: (a). Resume table of PnP. (b). Evolution of PSNR and the residuals through epochs.

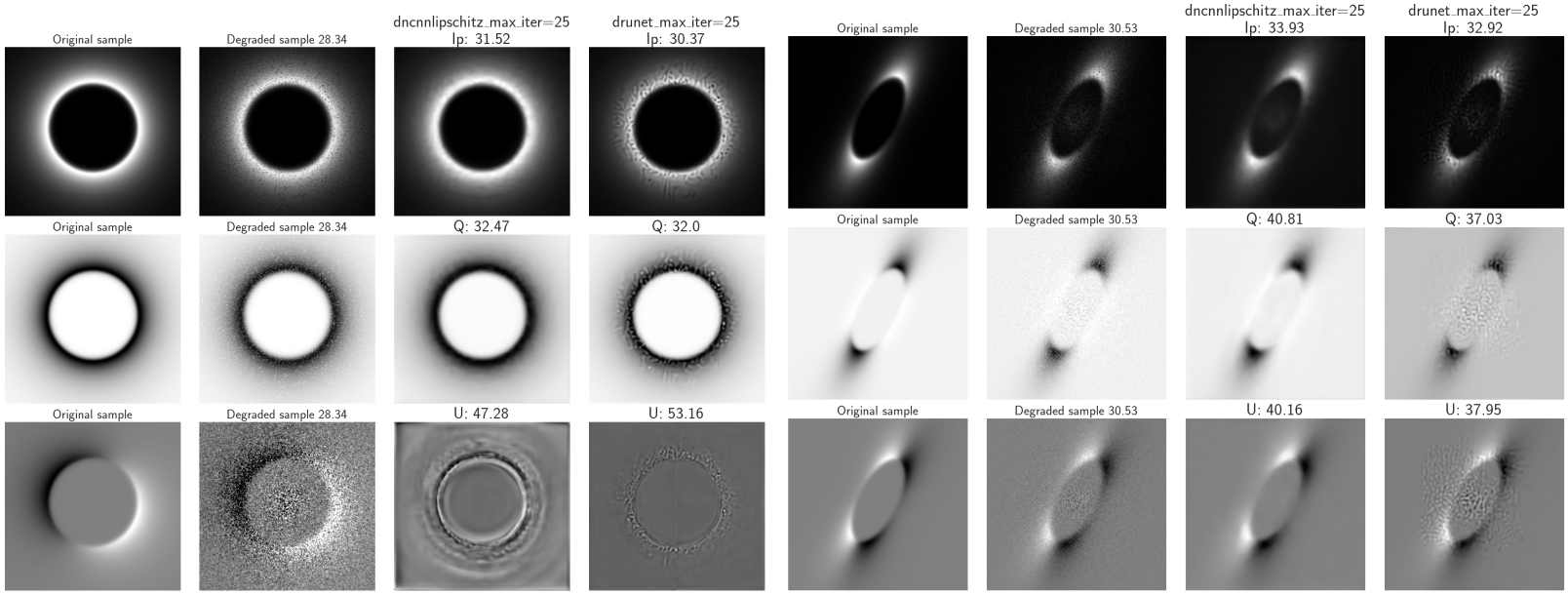


Figure 37: Reconstructed samples with PnP.

Bibliography

[1] H. Avenhaus, S. P. Quanz, H. M. Schmid, M. R. Meyer, A. Garu, S. Wolf, and C. Dominik. Structures in the protoplanetary disk of hd142527 seen in polarized scattered light. *The Astrophysical Journal*, 781(2):87–91, 2014.

[2] Emilie Chouzenoux, Jean-Christophe Pesquet, and Audrey Repetti. Variable metric forward–backward algorithm for minimizing the sum of a differentiable function and a convex function. *Journal of Optimization Theory and Applications*, 162(1):107–132, 2014.

[3] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale modeling & simulation*, 4(4):1168–1200, 2005.

[4] Laurent Condat. A primal-dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms. *Journal of optimization theory and applications*, 158(2):460–479, 2013.

[5] J. de Boer, M. Langlois, R. G. van Holstein, J. H. Girard, D. Mouillet, A. Vigan, K. Dohlen, F. Snik, C. U. Keller, C. Ginski, D. M. Stam, J. Milli, Z. Wahhaj, M. Kasper, H. M. Schmid, P. Rabou, L. Gluck, E. Hugot, D. Perret, P. Martinez, L. Weber, J. Pragt, J.-F. Sauvage, A. Boccaletti, H. Le Coroller, C. Dominik, T. Henning, E. Lagadec, F. Ménard, M. Turatto, S. Udry, G. Chauvin, M. Feldt, and J.-L. Beuzit. Polarimetric imaging mode of vlt/sphere/irdis: I. description, data reduction, and observing strategy. *Astronomy & Astrophysics*, 633:A63, 2020.

[6] L. Denneulin. Approche inverse pour la reconstruction des environnements circumstellaires en polarimétrie avec l'instrument d'imagerie directe eso/vlt sphere irdis. *Université Claude Bernard Lyon*, 2020.

[7] L. Denneulin, M. Langlois, E. Thiébaut, and N. Pustelnik. Rhapsodie: Reconstruction of high-contrast polarized sources and deconvolution for circumstellar environments. *Astronomy and Astrophysics*, 653:138–150, 2021.

[8] Sripad Krishna Devalla, Prajwal K. Renukanand, Bharathwaj K. Sreedhar, Shamira Perera, Jean-Martial Mari, Khai Sing Chin, Tin A. Tun, Nicholas G. Strouthidis, Tin Aung, Alexandre H. Thiery, and Michael J. A. Girard. Drunet: A dilated-residual u-net deep learning network to digitally stain optic nerve head tissues in optical coherence tomography images, 2018.

[9] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pages 399–406, 2010.

[10] Mingyuan Jiu and Nelly Pustelnik. A deep primal-dual proximal network for image restoration. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):190–203, 2021.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[12] Seppo Linnainmaa. *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. PhD thesis, Master's Thesis (in Finnish), Univ. Helsinki, 1970.

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[14] Johan Olofsson, Julien Milli, Amelia Bayo, Th Henning, and Natalia Engler. The challenge of measuring the phase function of debris discs-application to hr 4796 a. *Astronomy & Astrophysics*, 640:A12, 2020.

[15] B. T. Polyak. *Introduction to Optimization*. Optimization Software, Inc., Publications Division, New York, 1987.

[16] Ernest Ryu, Jialin Liu, Sicheng Wang, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Plug-and-play methods provably converge with properly trained denoisers. In *International Conference on Machine Learning*, pages 5546–5557. PMLR, 2019.

[17] Yu Sun, Brendt Wohlberg, and Ulugbek S Kamilov. An online plug-and-play algorithm for regularized image reconstruction. *IEEE Transactions on Computational Imaging*, 5(3):395–408, 2019.

[18] Julian Tachella, Dongdong Chen, Samuel Hurault, and Matthieu Terris. Deep inverse: A machine learning library for inverse problems. `https://github.com/deepinv/deepinv`, 2023.

[19] J. Tinbergen. Astronomical polarimetry. *Cambridge University Press*, 2005.

[20] Rob G van Holstein, Julien H Girard, Jozua De Boer, Frans Snik, Julien Milli, DM Stam, C Ginski, David Mouillet, Zahed Wahhaj, Hans M Schmid, et al. Polarimetric imaging mode of vlt/sphere/irdis-ii. characterization and correction of instrumental polarization effects. *Astronomy & Astrophysics*, 633:A64, 2020.

[21] Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. Plug-and-play priors for model based reconstruction. In *2013 IEEE global conference on signal and information processing*, pages 945–948. IEEE, 2013.

[22] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. Plug-and-play image restoration with deep denoiser prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6360–6376, 2021.

[23] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, jul 2017.