UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
"MATEMATICA"

CICLO XXXVII

Novelties in the Deep Image Prior framework for image restoration and segmentation

Coordinatore:
Chiar.mo Prof. **Leonardo Biliotti**

Relatrice:
Chiar.ma Dott.ssa **Federica Porta**

Dottoranda: **Ambra Catozzi**

Anni Accademici 2021/2022 - 2023/2024

To Daniel

Abstract

The so-called Deep Image Prior (DIP) approach is an unsupervised deep learning methodology which has gained great interest in recent years due to its effectiveness in tackling imaging problems, such as denoising, inpainting, and super-resolution, without the need for extensive pre-training. A key limitation of DIP is its tendency to overfit noise if optimization runs too long, leading to the so called semiconvergence effect, where the model begins to capture noise rather than improve image quality. Although DIP has proven effective in tasks like denoising and super-resolution, its potential in areas like blind deconvolution and segmentation remains under-researched.

The aim of this thesis is to contribute to the DIP framework by exploring new possibilities for applying this approach to blind deconvolution and segmentation tasks, as well as developing efficient early-stopping techniques that automatically halt network optimization once an optimal reconstruction is achieved.

For addressing challenges in blind deconvolution and segmentation, the original DIP framework has been extended to the variational models specifically tailored to capture the unique structural and statistical characteristics of these imaging applications, enhancing DIP's adaptability to complex image restoration tasks.

On the other hand, two early stopping strategies have been developed, each based on a distinct approach. The first is based on a Neural Architecture Search to generate optimal hyperparameter configurations for the neural network used in Deep Image Prior, helping to prevent the semiconvergence effect. The second strategy relies on a modified version of the BRISQUE metric, a no-reference image quality measure, to track the behavior of the PSNR curve produced by Deep Image Prior, without requiring the ground truth image.

Acknowledgments

I would like to thank my thesis supervisor, Dr. Federica Porta, and my tutor Prof. Marco Prato for having followed my entire PhD career along these three years and for being always proactive and available.

I also thank Prof. Valeria Ruggiero, Dr. Giorgia Franchini, and Dr. Alessandro Benfenati for their collaboration and for sharing their knowledge during the research activities.

Furthermore, I extend my appreciation to all the professors, colleagues, friends, and family members who have supported me in various ways throughout this academic pursuit.

Lastly, I would like to dedicate this thesis to the person who is always by my side, loving me and encouraging me to be better, Daniel Selvatici.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence.

ANN Artificial Neural Network.

AT Ambrosio-Tortorelli.

BN Batch Normalization.

CNN Convolutional Neural Network.

DIP Deep Image Prior.

DL Deep Learning.

ES Early Stopping.

MAP Maximum A Posteriori.

ML Machine Learning.

MS Mumford-Shah.

MS-SSIM Multiscale Structural Similarity Index Measure.

NAS Neural Architecture Search.

PGDA Proximal Gradient Descent Ascent.

PSNR Peak Signal-to-Noise Ratio.

SSIM Structural Similarity Index Measure.

CHAPTER 1

Introduction

The field of image reconstruction is crucial across a wide range of scientific domains. The ongoing need for improved techniques to address imaging inverse problems has led to significant advancements in computational methods, particularly in the realm of Deep Learning.

Deep Image Prior is a recent approach that leverages a specific type of artificial neural network as a prior for image restoration tasks, including denoising, inpainting, and super-resolution. Unlike traditional deep learning techniques, which require large training datasets, Deep Image Prior works without any pre-training, relying on the structure of the network itself to encode prior information about data, like statistical properties. Indeed, in Deep Image Prior, the model does not generalize from a prior dataset but optimizes itself on the single image data during runtime to reconstruct a clean version of that image. This approach is effective because it leverages the network's tendency to reconstruct meaningful structures (like edges, textures, and object continuity) before noise, which typically appears as random patterns. This is a significant advantage in real case scenarios where large labeled datasets are not available. Other advantages include its flexibility to adapt to different loss functions based on the task and its reduced need for intensive hyperparameter tuning, since the process is mainly based on the natural behaviour of network optimization during inference.

One major drawback of Deep Image Prior is its tendency to overfit the noise if optimization continues for too long, leading to the so called "semiconvergence" effect. Since DIP operates directly on the noisy or corrupted image, after a certain number of iterations, the network starts fitting the noise itself, which degrades the quality of restoration. This makes it crucial to apply early stopping techniques to halt optimization at the right point. Another limitation is that DIP only uses the corrupted image and lacks external data to guide the learning process; thus, it may struggle with highly noisy or degraded images, especially when degradation is significant, as the model might fail to capture the underlying structure. Additionally, the choice of network architecture is important, as a suboptimal architecture can reduce performance or add complexity to practical application. Finally, although Deep Image Prior has recently emerged as a landmark approach for addressing various imaging challenges, such as denoising, JPEG artifact removal, inpainting, and super-resolution, its application to other areas like blind deconvolution and segmentation remains relatively unexplored.

The aim of this thesis is to contribute to the Deep Image Prior framework by exploring new possibilities for applying this approach to blind deconvolution and segmentation tasks, as well as developing efficient early-stopping techniques that automatically halt network optimization once an optimal reconstruction is achieved.

To provide a comprehensive foundation for the thesis, Chapter 2 reviews the background of Artificial Neural Networks, summarizing key concepts behind Deep Learning strategies for solving minimization problems.

Chapter 3 provides a comprehensive description of the Deep Image Prior approach, detailing how regularization can be implicitly achieved by the structure of a neural network. It also covers the optimization model used to solve imaging tasks via Deep Image Prior, and presents numerical results showing the advantages and limitations of this approach.

Chapter 4 is devoted to analyzing how Deep Image Prior can be exploited to address a blind deconvolution problem where the data are corrupted by Poisson noise. This results in a min-max problem depending on the weigths of two different neural networks. Proper optimization methods are discussed to solve this problem. The effectiveness of the proposed approach is evaluated on blind deconvolution problems arising from confocal microscopy.

The aim of Chapter 5 is to extend the Deep Image Prior idea to the segmentation of noisy images in order to benefit of both traditional variational models and new deep learning techniques. Indeed the resulting method consists of an unsupervised deep learning approach based on the minimization of very well known variational energies (such as the Mumford-Shah functional and its approximation proposed by Ambrosio and Tortorelli) properly parametrized in terms of the weights of convolutional neural networks. The implicit regularization provided by the network allows to make the traditional variational models more robust with respect to both the noise corrupting the data and the selection of the parameters which balance the role of the regularization terms. Several numerical experiments on noisy segmentation problems show promising results of the suggested approach.

In Chapter 6, two early stopping procedures for Deep Image Prior are illustrated. The first technique relies on the Neural Architecture Search strategy by generating hyperparameters configurations for the neural network employed in Deep Image Prior, configurations that shall be able to provide clean images comparable to those obtained by the standard configuration, optimally stopped, but with significantly fewer iterations. The second proposed early stopping strategy is based on a modified version of the so-called BRISQUE metric, a no-reference image quality measure, and it aims to track the behaviour of the PSNR curve, obtained by applying Deep Image Prior, without knowing the ground truth image. While the NAS-based early stopping technique is particularly suited in those situations where the computational time is limited, this latter one is also relevant when a larger number of iterations is allowed. Several numerical experiments on different denoising applications show a promising performance of Deep Image Prior combined with the suggested early stopping procedures.

The last chapter sums up the achievements and the conclusions of this thesis.

Related publications

This thesis presents a selection of articles and results that have been published or are under review in journals and conference proceedings. It represents the main part of the research conducted during the doctoral program.

- Benfenati A., Catozzi A. and Ruggiero V., ”Neural blind deconvolution with Poisson data”, Inverse Problems 39.5, 2023.
- Benfenati A., Catozzi A., Franchini G., Porta F., ”Early stopping strategies in Deep Image Prior”, submitted, 2024.
- Benfenati A., Catozzi A., Franchini G., Porta F., ”Piece-wise Constant Image Segmentation with a Deep Image Prior Approach”, International Conference on Scale Space and Variational Methods in Computer Vision. Cham: Springer International Publishing, 2024.
- Benfenati A., Catozzi A., Franchini G., Porta F., ”Unsupervised noisy image segmentation using Deep Image Prior”, submitted, 2024.

Other results obtained during the PhD have led to the following publications:

- Bubba T.A., Calatroni L., Catozzi A., Crisci S., Pock T., Pragliola M., Rautio S., Riccio D., Sebastiani A., ”Bilevel Learning of Regularization Models and Their Discretization for Image Deblurring and Super-Resolution”, Advanced Techniques in Optimization for Machine Learning and Imaging, 2024;
- Franchini G., Verucchi M., Catozzi A., Porta F., Prato M., ”Biomedical Image Classification via Dynamically Early Stopped Artificial Neural Network”, Algorithms 15(10):386, 2022.

CHAPTER 2

Background on Artificial Neural Networks

Artificial Intelligence (AI) describes a system's ability to exhibit human-like capabilities such as reasoning, learning, planning, and creativity. Today, AI is pervasive across various fields, including Medicine, Philosophy, Information Technology, and the Arts. A specific branch of AI focuses on investigating and implementing algorithms necessary for modeling observed phenomena, processing large amounts of data, or predicting events. This field is known as *Machine Learning* (ML), which aims to leverage statistical tools and concepts to analyze, evaluate, and optimize mathematical functions used for making decisions or predictions based on data.

Specifically, the aim of ML is to efficiently train models by learning from available data. In the literature, *learning* is described as a process in which performance on a specific task improves with experience in relation to a set of tasks and a performance measure. The *performance* is usually measured by a metric, which depends on the type of problem, indicating whether the method produces good results. Common tasks in ML include classification, regression, image restoration, and segmentation.

To facilitate learning, ML models rely on a *training set*, which is a collection of data examples provided to the algorithm to learn patterns or relationships. The quality, size, and diversity of the training set play a crucial role in the model's ability to generalize to unseen data.

ML algorithms can be classified in two categories: *supervised* and *unsupervised* methods. In the supervised approach, the input training data are mapped to corresponding outputs, called *labels*, allowing the model to learn from input-output pairs. In contrast, in the unsupervised approach, the model identifies inherent structures or relationships within the data without reference to any output or label.

Deep Learning (DL), a subset of ML, is designed to handle large datasets and solve tasks with specific structures by leveraging different levels of abstraction. DL can address problems such as image classification, segmentation, restoration, regression, and natural language processing, all of which involve vast amounts of data, requiring a model capable of feature extraction ([1]).

2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are DL techniques that allow modeling phenomena and predicting events, finding application in numerous fields ([2]). Depending on the type of problem and the nature of the data to be examined, like time series, images or categorical data, just to name a few, specific network architectures can be constructed.

The structure of various artificial neural networks will be described below, starting with the basic unit, the *artificial neuron*. Each neuron processes a weighted sum of its inputs, which represents the information being processed, and this signal is passed through an *activation function* to produce an output. There is a clear analogy between the chemical transmission of information in a human neuron, represented in Figure 2.1, and the logical operations in an artificial neuron, which was first formalized in 1943 (see [3] and Figure 2.2 for a visual inspection). In details, a single unit is constituted by:

- an input $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$,
- some weights $\theta = (\theta_1, \theta_2, \dots, \theta_n) \in \mathbb{R}^n$,
- an activation function $f_a \colon \mathbb{R}^n \to \mathbb{R}^m$,

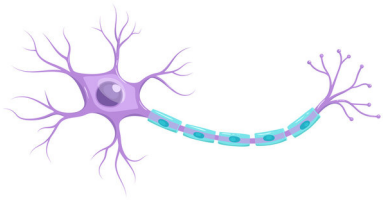


Figure 2.1: Representation of a neuron.

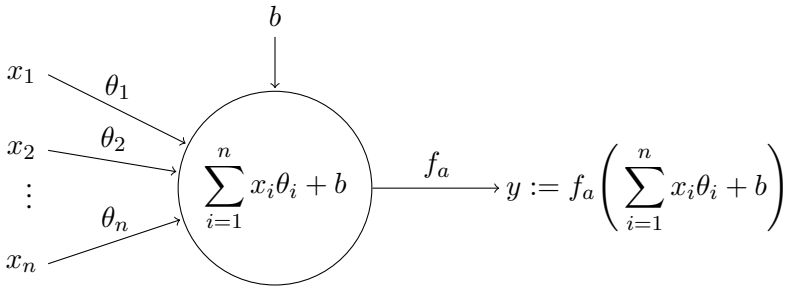


Figure 2.2: Mathematical representation of an artificial neuron.

- a bias $b \in \mathbb{R}$,
- an output $y \in \mathbb{R}^m$.

In Table 2.1 are listed the most popular examples of activation functions ([4, 5]).

Inputs can be processed in parallel and distributed across neurons at the same level, forming a structure known as a layer. Typically, an artificial neural network consists of an *input layer*, an *output layer*, and a certain number of *hidden layers* in between. The input layer receives and processes the data, the output layer generates the final result—such as a prediction, image reconstruction, or response—and the hidden layers transform the input data and perform computations. Additionally, these layers extract features by learning hidden patterns from the inputs. In particular, a network can be seen as the composition of many functions, so much so that it can be described as a directed acyclic graph. For example, in Figure 2.3 there are three hidden layers, that are the functions h_1, h_2 and h_3 linked in chain: $f(x) = h_3(h_2(h_1(x)))$, whose length is called *depth*, while the dimensionality of the hidden layers is the *width*.
To summarize, a deep neural network is structured as follows.

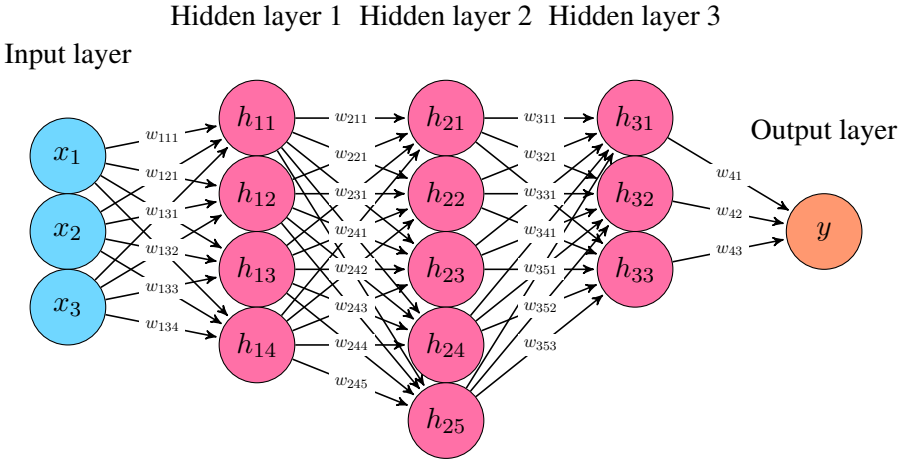


Figure 2.3: Example of feedforward neural network.

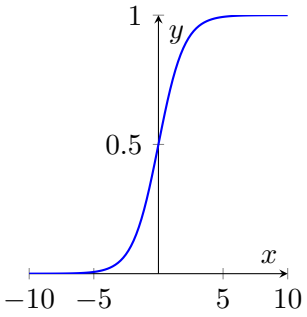
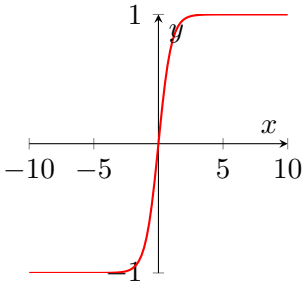
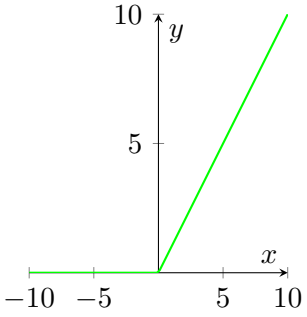
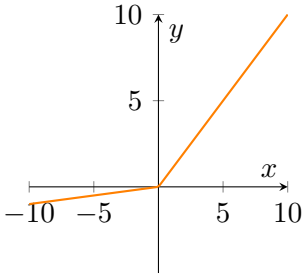
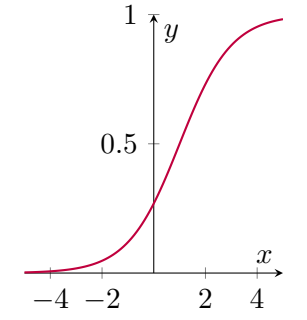
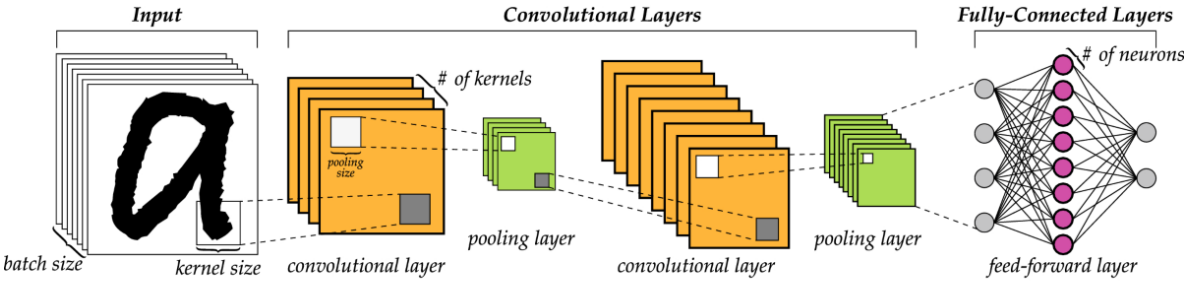
Name	Domain and codomain	Definition	Graph
Sigmoid	$\mathbb{R} \to (0, 1)$	$f_a(x) = \frac{1}{1+e^{-x}}$	
Hyperbolic tangent	$\mathbb{R} \to (-1, 1)$	$f_a(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	
Rectified Linear Unit (ReLU)	$\mathbb{R} \to [0, \infty)$	$f_a(x) = \max(0, x)$	
Leaky ReLU	$\mathbb{R} \to \mathbb{R}$	$f_a(x) = \begin{cases} x & \text{if } x \geq 0 \\ ax & \text{if } x < 0, a \in \mathbb{R} \end{cases}$	
Softmax	$\mathbb{R} \to (0, 1)$	$f_a(x) = \frac{e^x}{e^x + e^{x'}}$	

Table 2.1: Examples of activation functions in one dimensional case, $x \in \mathbb{R}$.



(c) Convolutional.

Figure 2.4: Example of convolution in CNNs from [7].

- **Input layer**. This layer consists of N units. These input units do not perform any computations; instead, they simply transmit information to the subsequent layers.
- **Hidden layers**. The network contains a series of L hidden layers, where $L \geq 2$.
- **Output layer**. The final layer is the output layer, which is made up of K neurons, where $K \geq 1$.
- **Synaptic weights**. There are synaptic weights that denote the strength of the connections between the neurons in adjacent layers.

By denoting with N_i the size of each layer, the synaptic weights are organized in weight matrices $\boldsymbol{W}^{(i)} \in \mathbb{R}^{N_i \times N_{i-1}}$ where an element $\boldsymbol{W}^{(i)}_{sr}$ encodes the weight between the r-th neuron in the $(i-1)$-th layer and the s-th neuron in the i-th layer. Hence each matrix $\boldsymbol{W}^{(i)}$ for $i = 1, \dots, L$ contains the weights of the edges which connect the $(i-1)$-th layer to the i-th. Moreover we denote the bias vectors as $\boldsymbol{b}^{(i)} \in \mathbb{R}^{N_i}$ for $i = 1, \dots, L$. Note that activation functions are specific to each neuron, and they can vary between layers to selectively transmit processed information to the next neuron.
The weight matrices and the bias vectors defining a neural network are learned during the training process, as discussed in Section 2.3. On the other hand, the architecture of a neural network - including the size of each layer, the activation function for each neuron, and the network's depth - is predetermined based on heuristic strategies and the specific application.

Since the arguments investigated in this thesis concern inverse problems in image processing, the focus will now shift to Convolutional and Encoder-Decoder Neural Networks, which are the most suitable for image analysis.

2.1.1 Convolutional Neural Networks

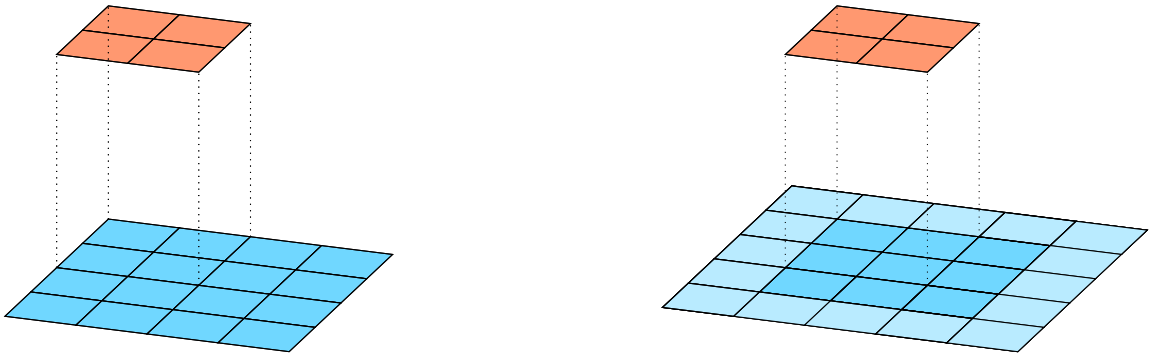
For solving Computer Vision tasks, Convolutional Neural Networks (CNNs, [6]) present the best architecture, because they resemble the biological behaviour of the human visual cortex. Such structures are capable of successfully capturing spatial dependencies in an image through the application of specific filters that reduce its complexity. A visual example of CNN is reported in Figure 2.4. This kind of nets are characterized by a specific mathematical operation called *convolution* which is suitable for analyzing spatial relationships in data, differing from the classical product between matrices.

Definition 2.1. *The convolution between two well defined functions* $i\colon \mathbb{R}^2 \to \mathbb{R}$ *and* $k\colon \mathbb{R}^2 \to \mathbb{R}$ *is given by*

$$(i * k)(x, y) = \int_{\mathbb{R}^2} i(s, r)k(x - s, y - r)dsdr \tag{2.1}$$

where the convolution is indicated by $*$.

It is possible to express (2.1) in a discrete domain, essential for practical implementation such as in image processing, where the domain is represented by a grid of pixels. In real-world applications,



(a) Valid padding

(b) Average padding

Figure 2.5: Examples of valid and same padding.

signals like images must be represented digitally, employing a finite process for storing discrete sets. Moreover (2.1) involves an integral over an infinite range, while 2.2 simplifies the computation to a finite sum, making it manageable for digital processing.

Definition 2.2. *The discrete convolution between two functions* $I\colon \mathbb{Z}^2 \to \mathbb{R}$ *and* $K\colon \mathbb{Z}^2 \to \mathbb{R}$ *is given by*

$$(I * K)(x, y) = \sum_s \sum_r I(s, r)K(x - s, y - r) \tag{2.2}$$

where the discrete convolution is indicated by $*$ *and the input data is* I *and the so-called kernel or filter* K *is a feature extraction matrix with smaller dimensions.*

The output of the operation described in (2.2) is called *feature map*, because it is able to enhance and extract the characteristics of the input. For example, in image processing the feature map brings out patterns, pixel intensities or boundaries.
Since also in the discrete context the operations are performed near to the boundary pixels, it is mandatory adding new pixels framing the input image before the convolution. This technique is called *padding* and it allows to maintain the spatial dimensions of the image after the convolution and there exist two different methods. The first is the *valid padding* where the output dimensions decrease, since the kernel K is applied only at pixels that allow it to be contained entirely in the input data I (see Figure 2.5(a)). The second way is the *same padding*, where the output admits the same size of the input image; the number of pixels for framing the input is computed using the dimensions of the kernel and the required feature map dimensions (see Figure 2.5(b)). An example is the *zero-padding*, where zero pixels are added to the border of the input.

The dimensions of the feature map can be reduce through the *pooling layers*, cutting off the number of net parameters, thus the computational cost, and it can be seen as a downsampling procedure. The most common pooling layers acting on the feature map patches are:

- *max pooling*, which saves the maximum value in each patch, that is the pooling window (see Figure 2.6(a)), and it aids in making feature detection in input data robust to changes in scale and orientation;
- *average pooling*, which saves the average value in each patch (see Figure 2.6(b)).

2.1.2 Encoder-decoder

An *encoder-decoder* network [8] is a type of neural network architecture that is widely used for tasks where the input data needs to be transformed into some intermediate representation (encoded), and then the output data is generated from that representation (decoded). This architecture is particularly useful

(a) Max pooling

(b) Average pooling

Figure 2.6: Examples of max and average pooling: the input matrix is 4×4 with integer values in $[1, 16]$, the pooling windows are 2×2 and the output matrix is the orange 2×2 matrix, containing the maximum or the average values from each pooling window.

in applications where the input and output are of different formats or sizes, such as in image processing, NLP, and signal processing. The first part of the architecture is called *encoder* and it compresses the input data into a lower-dimensional representation or *latent space*, thanks to the presence of many layers, then its output is a compact and abstract representation of the input that captures the most significant information. The second part, namely the *decoder*, reconstructs the data from the latent space.

There exists a subset of encoder-decoder architectures that are trained through unsupervised learning to reconstruct their input data; they are called *autoencoders* and usually are applied in denoising problems. Furthermore, *variational autoencoders* (VAEs) are encoder-decoder networks that are able to generate new data samples by learning a probabilistic representation of the input data.

An example of encoder-decoder is the U-Net architecture (introduced in [9] and designed for biomedical images). It is a CNN highly effective in image segmentation tasks, where the goal is to assign a class label to each pixel in an image, because it combines high-resolution spatial information from the encoder with contextual information from the decoder. Its ability to provide precise, pixel-level segmentation has made it a popular choice for many Computer Vision problems. Its name comes from the shape of the architecture since it is composed by two symmetric parts:

- the encoder, responsible for capturing the image features by downsampling (reducing the spatial dimensions) while increasing the number of feature channels, *i.e.* the output of a convolutional filter applied to an input image or feature map, thanks to convolutional layers followed by max-pooling operations to reduce the size of the feature maps. It uses repeated applications of two 3×3 convolutional layers, followed by a ReLU activation and a 2×2 max-pooling layer to downsample the image. Each step doubles the number of feature channels.

- the decoder, responsible for upsampling the image back to the original size while using the features extracted by the encoder, exploting the so-called *skip connections* between the encoder and the decoder parts. It upscales the feature maps using transposed convolutions (or up-convolutions) and halves the number of channels. At each step, the corresponding feature map from the contracting path is concatenated with the upsampled map via skip connections.

The skip connections allow the network to transfer high-resolution features from the contracting path directly to the corresponding upsampling layers in the expanding path. This helps the network recover fine details lost during the downsampling process.

2.2 Batch Normalization

A technique for improving the velocity and the accuracy of ANNs in general is the *Batch Normalization* (BN, [10]), which normalizes the layers' inputs through scaling and centering procedures such that they have zero mean and variance one. In the BN step, the mean and the variance are calculated over a *mini-batch* $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, i.e. a subset of samples from the training set:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i \quad \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \mu_B). \tag{2.3}$$

Thus the input normalization is given by

$$\hat{\mathbf{x}}_i = \frac{\mathbf{x}_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{2.4}$$

where ε is a small constant for avoiding the denominator annihilation. This practice is useful for increasing the stability, allowing higher learning rates, and the problems related to the exploding or vanishing gradient are reduced, making more robust the models [1].

2.3 Training deep neural networks

A deep neural network can be viewed as a function that relies on a set of weights and biases, represented as $\mathcal{Z} = \{(\boldsymbol{W}^{(i)}, \boldsymbol{b}^{(i)}) \text{ for } i = 1, \ldots, L\}$. Therefore, a deep neural network is formally defined as a function $\mathcal{N}(\theta, x)$ parameterized by $\theta \in \mathbb{R}^d$, exploiting the column vectorization [1].

In supervised learning, the aim is to adjust the parameters θ in order that $\mathcal{N}(\theta, \cdot)$ represent a prediction function from an input space $\mathbb{R}^{d_x}$ to an output space $\mathbb{R}^{d_y}$ such that, given $x \in \mathbb{R}^{d_x}$, the value $\mathcal{N}(\theta, x)$ offers an accurate prediction about the true output y. To achieve this, a so called loss function $\ell : \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \to \mathbb{R}$ is introduced; it is such that, for a given input-output pair (x, y), calculates the loss $\ell(\mathcal{N}(\theta, x), y)$, where $\mathcal{N}(\theta, x)$ is the predicted output and y is the true output. By assuming that the input-output space $\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ is endowed with a probability distribution $P : \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \to [0, 1]$, representing the true relationship between inputs and outputs, the parameters θ should minimize the following objective function

$$R(\theta) = \int_{\mathbb{R}^{d_x}\times\mathbb{R}^{d_y}} \ell(\mathcal{N}(\theta, x), y)\, dP(x, y) = \mathbb{E}[\ell(\mathcal{N}(\theta, x), y)]. \tag{2.5}$$

The function $R : \mathbb{R}^d \to \mathbb{R}$ is called the expected risk, given a parameter vector θ, with respect to the probability distribution P.
Although it would be ideal to minimize (2.5) directly, this is often not feasible due to incomplete information about the distribution P. Therefore, in practice, one aims to solve a problem based on an estimate of the expected risk R. In supervised learning, a set of $n \in \mathbb{N}$ independently drawn input-output samples $\{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ is available (either all at once or incrementally). Using this dataset, one can define the empirical risk function $R_n : \mathbb{R}^d \to \mathbb{R}$ as

$$R_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}\ell(\mathcal{N}(\theta, x_i), y_i), \tag{2.6}$$

where $\ell(\mathcal{N}(\theta, x_i), y_i)$ denotes the loss for the predicted output $\mathcal{N}(\theta, x_i)$ relative to the true output y_i. The set of input-output samples $\{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ is referred to as the *training set* and the process of finding the optimal parameter θ is called *training the network*.
Thus, the two fundamental components of learning in a network are its architecture - essentially, the composite function it represents - and the optimization algorithm used for training.

A widely recognized and effective algorithm is gradient descent, which optimizes a function by moving in the direction opposite to its gradient. This approach is commonly referred to as steepest descent, and its general iteration for minimizing (2.6) can be expressed as

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \nabla R_n(\theta^{(k)}) = \theta^{(k)} - \frac{\alpha}{n} \sum_{i=1}^{n} \nabla_\theta \ell(\mathcal{N}(\theta, x_i), y_i), \text{ given } \theta^{(0)}, \text{ for all } k \geq 0$$

where α is positive learning rate. For a complete overview on numerical optimization methods refer to [11]. It is worth noting that the analytical expression for the gradient can be derived using the chain rule. However, evaluating such expressions can often be computationally intensive. The *backpropagation* algorithm enables the computation of partial derivatives in a time that scales linearly with the depth of the computational graph[1] associated with the neural network. A detailed explanation of the backpropagation algorithm can be found in [12].

Given that n in (2.6) can be extremely large, calculating all the terms of the objective function $R_n(\theta)$ or its gradient can be prohibitively costly. Additionally, the entire dataset may be too large to fit into memory. In the context of online learning, where the dataset is not fully available from the very beginning and is instead gathered throughout the learning process, it becomes impractical to work directly with $R_n(\theta)$. In these situations, the minimization of $R_n(\theta)$ is addressed using stochastic approximations of the gradient, leading to the class of stochastic gradient descent (SGD) methods [13]. At each iteration k, a sample $\mathcal{S}_k$ of size $n_k \ll n$ is randomly and uniformly selected from the set $\{1, \dots, n\}$. The SGD algorithm for minimizing (2.6) can be expressed as:

$$\theta^{(k+1)} = \theta^{(k)} - \alpha g(\theta^{(k)}) \text{ given } \theta^{(0)}, \text{ for all } k \geq 0$$

where the stochastic direction $g(\theta^{(k)})$ is computed as

$$g(\theta^{(k)}) := \frac{1}{n_k} \sum_{i \in \mathcal{S}_k} \nabla_\theta \ell(\mathcal{N}(\theta, x_i), y_i).$$

The sample $\mathcal{S}_k$ represents the *mini-batch* at the k-th iteration, and its size n_k is referred to as the mini-batch size.

Selecting an appropriate learning rate for the SGD algorithm is quite challenging. Indeed, a learning rate that is too large can prevent convergence, while one that is too small can result in a very slow training process. A very popular alternative to SGD is represented by the Adam method [14], which falls under the class of adaptive stochastic gradient schemes. Given $\alpha > 0$, $\hat{\epsilon}, \beta_1, \beta_2 \in (0, 1]$, $\theta^{(0)}$ and setting $m_{-1} = 0$, $v_{-1} = 0$, the Adam k-th iteration is provided by the following recurrence formulas for $k \geq 0$:

$$\begin{aligned} m_k &= \beta_1 m_{k-1} + (1 - \beta_1) g(\theta^{(k)}) \\ v_k &= \beta_2 v_{k-1} + (1 - \beta_2) g^2(\theta^{(k)}) \\ \alpha_k &= \alpha \frac{\sqrt{1 - \beta_2^k}}{1 - \beta_1^k} \\ \theta^{(k+1)} &= \theta^{(k)} - \alpha_k \frac{m_k}{\sqrt{v_k} + \hat{\epsilon}} \end{aligned}$$

where the gradient squaring is to be intended element-wise. In [14], the authors proposed to set $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Unsupervised training of a neural network involves using unlabeled input data to discover inherent patterns and structures within the dataset. During this process, the network learns to represent the data

[1] A *computational graph* of an ANN is a structured representation of the sequence of operations and computations performed by the network during both the forward pass and in the backpropagation pass.

by identifying similarities and differences among the input features without any explicit guidance from labeled outputs. Chapter 3 is devoted to the analysis of a very recently proposed unsupervised approach especially tailored for image restoration problems.

CHAPTER 3

Deep Image Prior

Over the past decade, supervised deep learning methods have achieved state-of-the-art performance in addressing imaging inverse problems. This success can be attributed to their ability to learn the relationships between degraded images and their corresponding cleaned versions by utilizing highly representative models, such as deep neural network architectures, along with an external training set of paired degraded and clean examples. However, these approaches often face challenges, including limited generalization when trained on insufficient data. Additionally, in many practical scenarios, such as medical imaging, it can be nearly impossible to create a labeled dataset containing both ground truth and degraded data. These challenges have prompted researchers to explore unsupervised deep learning techniques that eliminate the need for training sets. Among these, Deep Image Prior [15, 16] stands out as one of the most promising methods in this category.

The Deep Image Prior (DIP) is an unsupervised approach tailored for solving a large class of imaging inverse problems. Such approach employs artificial neural networks, particularly U-Nets equipped with skipped connections [9]: the claim of the original work states that the *inner structure* of this type of networks is capable to capture a great amount of image statistics, without a training on large datasets. In [15, 16] the authors fit an untrained network to a single degraded image, addressing several problems such as image denoising, image restoration, inpainting and super-resolution. This chapter presents the DIP framework, particularly detailing the architecture of the neural network used in the DIP approach and the optimization problem that must be solved to address an imaging task.

3.1 The DIP neural network

The Deep Image Prior approach introduced in [15, 16] shows that generative neural architectures are capable to capture image statistics even in absence of a training phase and without using large datasets. The original work employed U-Net structures for tackling several imaging problems: indeed these architectures have gained interest since they showed to have remarkable performances in several tasks, such as image segmentation [17, 18] in medical imaging, in image generation [19, 20], precipitation nowcasting [21]. Before detailing the unsupervised DIP approach, the neural network employed on which DIP is based is described (see Figure 3.1). A U-Net can be interpreted as an autoencoder: the first part encodes the information by applying several operations (convolutions, batch normalization, leaky ReLU) and each stage halves the spatial dimensions of the image. The second part, formally the decoder, doubles the spatial dimensions to retrieve the original image size and reconstruct the information. Moreover, each stage of the encoder is linked to the relative stage of the decoder, acting on the same spatial dimensions, via skip connections: on one hand, these blocks allow to transmit information among the different levels of encoding-decoding, and moreover they seem to improve the learning phase, boosting the search for the minimum [22, 23]. Figure 3.1 depicts the architecture used in [15, 16], where both the encoder and the decoder have 5 stages. The first layer of an encoder stage is a convolution operation that halves the spatial dimensions, followed by a batch normalization layer and by a leaky ReLU layer with parameter -0.6. This triplet is repeated, with the sole difference that the second convolution layer does not halve the spatial dimensions. The last stage of the encoder (the lighter bottom block in Figure 3.1) ends with an upsampling layer, for having the correct spatial dimensions for using the skip connections. Each decoder stage starts with a depth concatenation layer, in order to combine their input with the output of the skip connections, then 2 triplets of convolution,

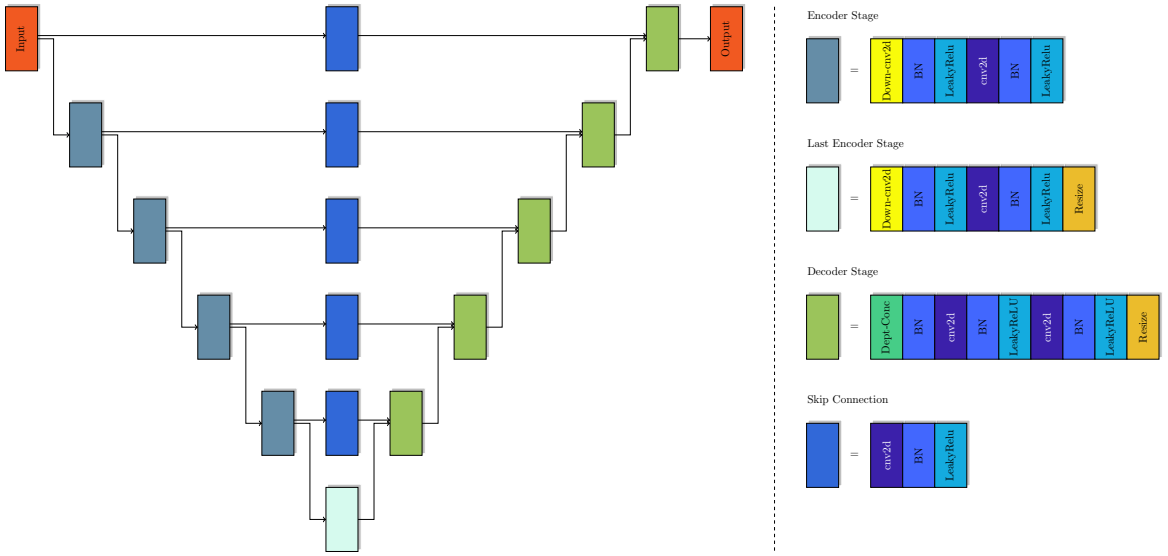


Figure 3.1: The Deep Image Prior is characterized by a U-Net architecture. The picture shows the original architecture of [16], with 5 stages of downsampling and relative upsampling, together with the skip connections. The last stage of the encoder part, which corresponds to the latent feature, is depicted in light blue and differs from the other encoder stages by the presence of a final upsampling layer.

batch normalization and leaky ReLU operations are combined. The last operation is an upsampling operation for doubling the dimensions, since one wants to recover the original image size. The structure of the skip connections is rather simple: they combine convolution, batch normalization and leaky ReLU layers.

3.2 The DIP optimization model

In DIP framework such U-Net is used as a generator: it is fed with a multichannel random Gaussian noise, which is used to recover the image of interest by learning the weights of the net under the minimization of a fit-to-data functional. Hence, the DIP approach is based on a new concept towards the regularization of inverse problems, achieved by forcing the recovered image to be a generated image from a learned network. As a consequence, the classical variational model which formalizes an imaging problem is rewritten in terms of the weights of a neural network and the structure of the network itself is implicitly used to obtain a regularization effect on the solution.
The classical variational approach for imaging problems consists in searching for the solution of

$$\underset{u}{\operatorname{argmin}}\ \mathcal{D}(u, g)$$

where $\mathcal{D}$ is the data fidelity and g is the recorded image. In particular, the recorded data is given by the following model

$$g = \mathcal{P}\left(H * u^{\star} + b\right) \tag{3.1}$$

where $u^{\star}$ is the ground truth, H is a linear operator describing the blur effects-such as out-of-focus, motion blur-of the acquisition system, b is a constant background term, $*$ denotes the convolution and $\mathcal{P}$ models the statistical noise affecting the data, which can be signal-dependent or signal-independent.

The recorded data can be a grayscale, binary or an RGB image, hence $g \in \mathbb{R}^{p\times q\times d}$, with $p \times q$ pixels and $d \in \{1, 3\}$ channels. DIP framework suggests to reparameterize u with a generative network parameterized by θ. In particular, let $z \sim \mathcal{N}(0, \sigma^2)$, $z \in \mathbb{R}^{p\times q\times h}$ a random Gaussian input with same spatial dimensions and h channels. The U-Net described before can be seen as a function

$f : \Theta \times \mathbb{R}^{p\times q\times h} \to \mathbb{R}^{p\times q\times d}$, where Θ is the space of the net weights θ. The DIP approach is based on numerically solving an optimization problem, searching for the optimal weights $\theta^\star$:

$$\theta^\star \in \underset{\theta\in\Theta}{\operatorname{argmin}}\ \mathcal{D}\left(f(\theta, z), g\right) \tag{3.2}$$

where $\mathcal{D}$ is the discrepancy functional chosen in dependence of the noise affecting the image. Once the optimal weights $\theta^\star$ are found by applying an iterative algorithm to (3.2), properly stopped before the degraded image starts to be over fitted, the resulting approximation of $u^\star$ can be computed as $f(\theta^\star, z)$. The complete scheme of the DIP approach is reported in the algorithm 1.

Algorithm 1: Deep Image Prior

Select the network f, initialise the network's weights $\theta^{(0)}$, select the discrepancy function $\mathcal{D}$. Set σ^2 for the input's variance. Choose the learning algorithm $Algo$, its hyperparameters and the loss function ℓ. Select the maximum number of iterations N.;

for $k = 0, 1, \ldots, N-1$ **do**

$z \sim \mathcal{N}(0, \sigma^2)$, a realization from a Gaussian probability distribution;

$\ell(\theta^{(k)}) \leftarrow \mathcal{D}\left(f(\theta^{(k)}, z), g\right)$;

$\theta^{(k+1)} \leftarrow Algo(\theta^{(k)}, \nabla\ell(\theta^{(k)}))$;

end

Recover the approximation of u^* as $f(\theta^N, z)$.

Notice that it is possible to add a regularization term to Problem (3.2), like in [24, 25, 26], for example the anisotropic Total Variation function:

$$TV(u) = \sum_i \|A_i\, u\|, \tag{3.3}$$

where $n = pq$, $A_i \in \mathbb{R}^{2\times n}$ is the two dimensional discrete first order difference operator at the pixel i of the vector-reshaped image $u \in \mathbb{R}^n$, and the matrices A_i, $i = 1, \ldots, n$ are submatrices of the complete difference matrix $A \in \mathbb{R}^{2n\times n}$, $A = \left(A_1^\top, \ldots, A_N^\top\right)^\top$. This functional is used for preserving characteristics on the reconstructed image, such as sharp edges.

The resulting DIP model is built upon the solution of the following optimization problem:

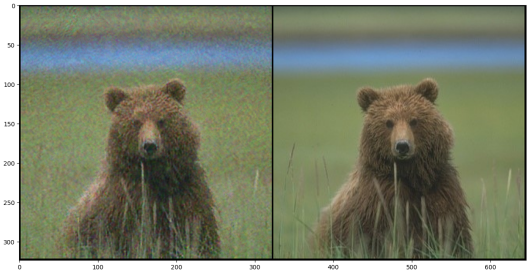
$$\underset{\theta\in\Theta}{\operatorname{argmin}}\ \mathcal{D}\left(f(\theta, z), g\right) + \lambda TV(f(\theta, z)), \tag{3.4}$$

where λ is the regularization term that balances the trade off between the discrepancy function and the Total Variation.

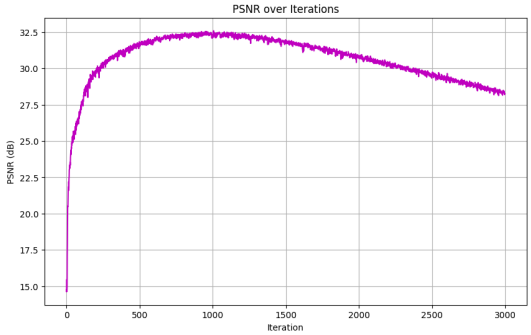
3.3 Semi-convergence behaviour of vanilla DIP

The aim of this section is to illustrate the so called semi-convergence behaviour typical of DIP. Indeed if optimization continues for too long DIP tends to overfit the noise. The numerical experiments reported in this section are carried out by running the standard DIP[1] on denoising problems. The maximum number of iterations is set to 3000, the optimization algorithm is Adam with learning rate 10^{-2} and the discrepancy functional $\mathcal{D}$ in (3.2) coincides with the Mean Squared Error. The input z at each iteration is corrupted by an additive normal noise with zero mean and standard deviation $\frac{1}{30}$. The considered test images, reported in Figure 3.2, are picked from Berkeley Segmentation Dataset and Benchmark (BSDS500, [27]) and the input noisy data g are corrupted by Gaussian noise with standard deviation $\sigma = \frac{25}{255}$. In Figure 3.2 the results of the denoising process achieved by DIP are reported with the

[1]The code is available at `https://github.com/DmitryUlyanov/deep-image-prior` ([15]).

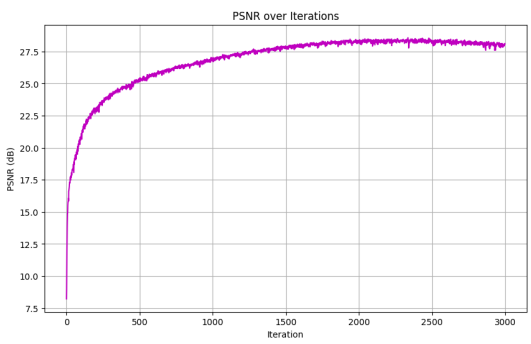


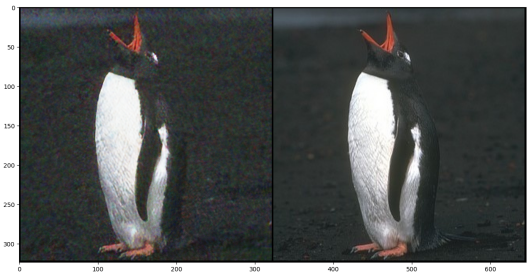
(a)



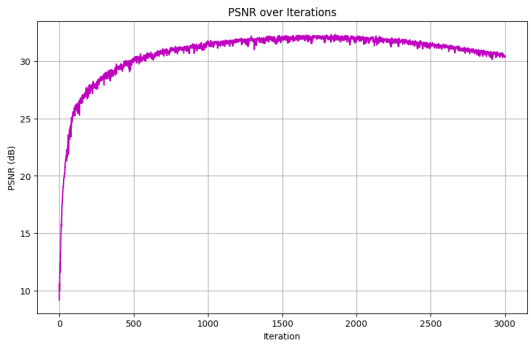
(b)

(c)





(d)



(e)

Figure 3.2: Examples of Deep Image Prior for denoising task. The images are corrupted by Gaussian noise with 0 mean and $\sigma = \frac{25}{255}$ is the standard deviation. The number of iterations is set at 3000. The first column reports on the left the recovered images and on the right the original images from the dataset, while the second column illustrates the variations of the PSNR metrics.

corresponding behaviour of the PSNR among the iterations. It is clear that all the reconstructions on the left are very distant from the respective clean data on the right. The recovered image is the output at 3000 iterations and it has a poor quality, but it doesn't correspond to the best metric measure. The PSNR plot clearly shows that after a certain number of iterations, the PSNR begins to decrease. In addition, the optimal number of iterations is not the same for all the images but depends on the image.

Moreover if an higher standard deviation value is chosen for the corruption of the input images, like $\sigma = \frac{50}{255}$ in the example in Figure 3.3, the effect of the semiconvergence on the PSNR is more evident, thus the recovered images have again a very low quality.

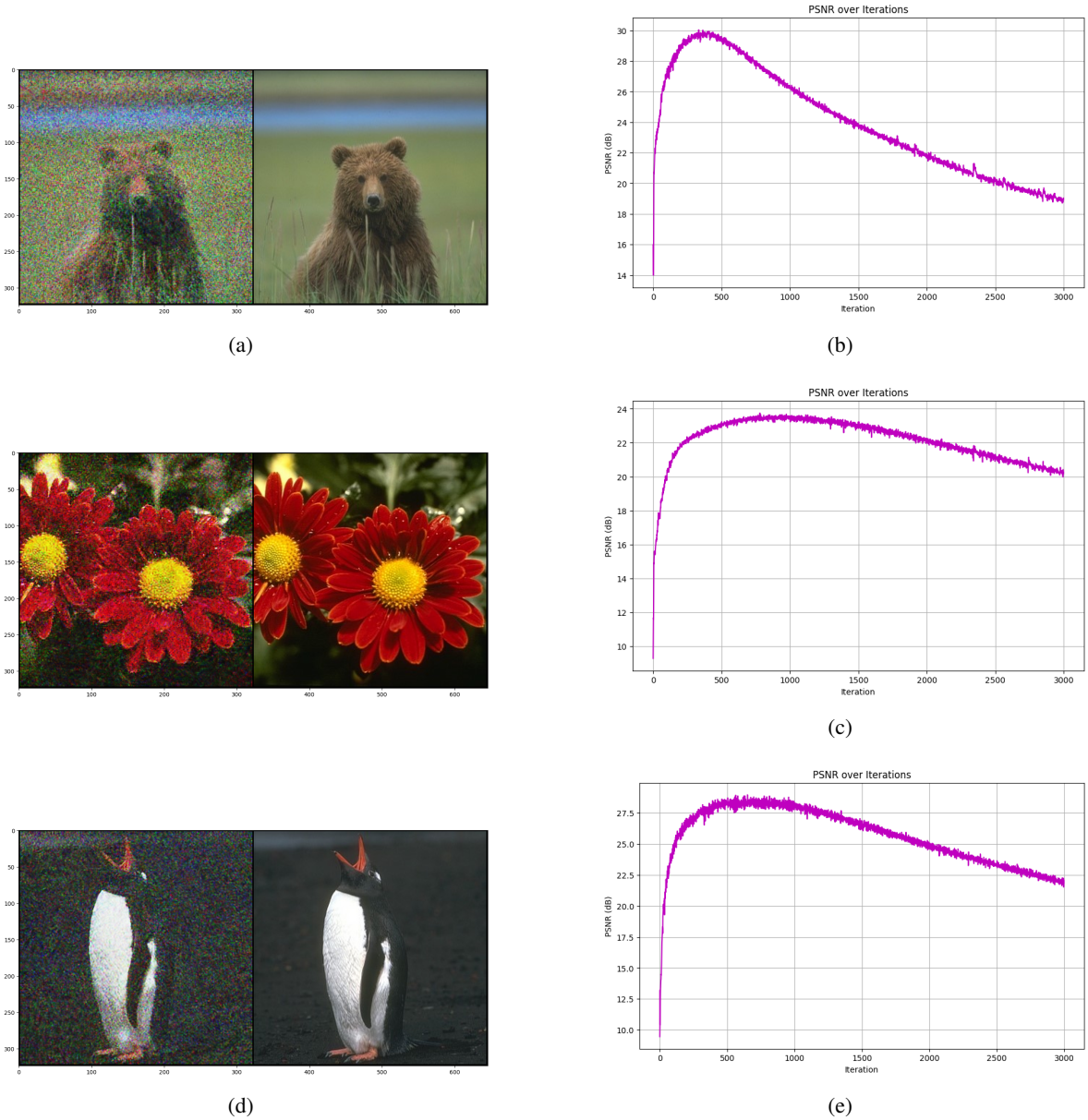


Figure 3.3: Examples of Deep Image Prior for denoising task. The images are corrupted by Gaussian noise with 0 mean and $\sigma = \frac{50}{255}$ is the standard deviation. The number of iterations is set at 3000. The first column reports on the left the recovered images and on the right the original images from the dataset, while the second column illustrates the variations of the PSNR metrics.

CHAPTER 4

Neural blind deconvolution with Poisson data

The aim of this chapter is to exploit the Deep Image Prior approach to address a Blind Deconvolution problem where the data are corrupted by Poisson noise.

4.1 Preliminary notions

Blind Deconvolution (BD) consists in a image deblurring problem when the *Point Spread Function* (PSF) is not known or only approximately known, *e.g.* only a rough approximation from observation or a mathematical model, containing a limited number of unknown parameters, is known. For a linear model of the acquisition process, the näive formulation of the BD is to solve the equation $\mathbf{g} = \mathbf{h} * \mathbf{u}$, where $*$ denotes the convolution product and both the PSF $\mathbf{h}$ and the object $\mathbf{u}$ are unknown and must be estimated from the recorded image $\mathbf{g}$.

Even when the PSF is accurately reported or known, BD is advantageous because the PSF provided by instrumentation may not fully account for real-world conditions, such as vibrations or optical imperfections, which can cause deviations from the theoretical PSF. Moreover, even with expensive equipment, the measured PSF might only be an approximation, subject to calibration errors, noise, or resolution limits. Other factors that can influence data acquisition include inherent optical aberrations, lens imperfections, or diffraction limits. This method also enhances the output images by accounting for the possibility of biological specimen movement during Microscopy analysis.

A convenient and realistic assumption is to consider a space-invariant model for the PSF. Nevertheless, the problem is extremely under-determined and there exists an infinite set of solutions. One of them is the trivial one, i.e., $\mathbf{u} = \mathbf{g}$ and $\mathbf{h} = \boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is the Dirac's delta, which is 1 at the center and zero elsewhere. Moreover, if the pair $(\tilde{\mathbf{h}}, \tilde{\mathbf{u}})$ is a solution and R is an injective linear operation commuting with the cyclic convolution, then the pair $(R\tilde{\mathbf{h}}, R^{-1}\tilde{\mathbf{u}})$ is also a solution [28].

BD is the subject of a wide literature and the different approaches concern specific classes of images and PSFs. For instance, approaches applicable to natural images may not be suitable in Microscopy or Astronomy; approaches developed for motion blur are not applicable to other classes of blur, and so on [29]. The obvious reason is that, since the problem is extremely ill-posed, specific kinds of prior knowledge must be introduced, for both the object and the PSF, in order to reduce the class of possible solutions [30]; as concerns natural images, the survey paper [31] contains a critical analysis of proposed methods as well as several relevant references. While methods that rely on a pre-calibrated PSF or supervised approaches can offer faster and potentially more deterministic results, they are often limited by the fidelity of the assumed PSF or the generalizability of the training data. Blind deconvolution trades off computational complexity and the potential for slower convergence in exchange for: greater robustness to PSF inaccuracies and the ability to leverage implicit priors (e.g., sparsity, structural consistency[1]) that are not easily encoded in pre-trained models or calibrated PSFs (see [32, 33, 34]).

In this thesis, the discussion is restricted to images acquired by a confocal microscopy, studying the methods of BD proposed in the framework of Poisson data. Poisson noise is often generated in low-light conditions and when an image is captured by digital devices. It consists in a multi-valued Poisson process: the outcome is influenced by the random incidence of photons on the sensor of the detection instrument (photon counting), with each pixel's value being independent of the others. A

[1] Structural consistency refers to the preservation of the key spatial, geometric, and relational features of an image or scene during a transformation, optimization, or reconstruction process.

further error source, that can be viewed as another Poisson process, is the photon-electron counting, which transforms the light information into the digital one. The intensity of the noise thus depends on the number of photons: as the number of photons increases, the level of noise affecting the image decreases ([35, 36]). The discrepancy functional used for restoring images perturbed by this kind of noise is the generalized Kullback-Leibler function, note that the Poisson noise is signal-dependent.

The model of the acquisition process then becomes

$$\mathbf{g} = P\left(\mathbf{h} * \mathbf{u} + b\right) \tag{4.1}$$

where the acquired image $\mathbf{g}$ is a realization of a Poisson multivalued random variable whose average is $\mathbf{h} * \mathbf{u} + b$; here b is a background function, usually a given constant function[2]. For a description of a confocal microscopy system and the related mathematical modelling of the imaging process, see [29]. In the Bayesian approach, the PSF $\mathbf{h}^*$ and the object $\mathbf{u}^*$ are considered realizations of multivalued random variables, respectively $\mathcal{H}$ and $\mathcal{X}$, so that the conditional probability of $\mathbf{h}, \mathbf{u}$ for a given value $\mathbf{g}$ of the random variable $\mathcal{G}$ of the image domain is given by

$$\mathbb{P}_{\mathcal{H},\mathcal{X}}(\mathbf{h}, \mathbf{u}|\mathbf{g}) = \frac{\mathbb{P}_{\mathcal{G}}(\mathbf{g}|\mathbf{h}, \mathbf{u})\mathbb{P}_{\mathcal{H}}(\mathbf{h})\mathbb{P}_{\mathcal{X}}(\mathbf{u})}{\mathbb{P}_{\mathcal{G}}(\mathbf{g})}, \tag{4.2}$$

where $\mathbb{P}_{\mathcal{H}}(\mathbf{h})$ and $\mathbb{P}_{\mathcal{X}}(\mathbf{u})$ are the priors of the PSF and the object, respectively. By assuming Gibb's priors and replacing $\mathbb{P}_{\mathcal{G}}(\mathbf{g}|\mathbf{h}, \mathbf{u})$ with the likelihood function, i.e., the Poisson probability distribution of $\mathcal{G}$ for the realization $\mathbf{g}$, the *Maximum A Posteriori* (MAP) estimates $\bar{\mathbf{h}}$, $\bar{\mathbf{u}}$ of $\mathbf{h}^*$ and $\mathbf{u}^*$, respectively, can be obtained by maximizing (4.2) with respect to $\mathbf{h}$ and $\mathbf{u}$, or, equivalently, by minimizing the negative logarithm of the conditional probability $\mathbb{P}_{\mathcal{H},\mathcal{X}}(\mathbf{h}, \mathbf{u}|\mathbf{g})$:

$$\min_{\mathbf{h}\in C_h,\, \mathbf{u}\in C_u} f^B(\mathbf{h}, \mathbf{u}; \mathbf{g}) \equiv f_0^B(\mathbf{h}, \mathbf{u}; \mathbf{g}) + f_1(\mathbf{h}) + f_2(\mathbf{u}). \tag{4.3}$$

Under the Poisson noise framework, the data-fidelity function $f_0^B(\mathbf{h}, \mathbf{u}; \mathbf{g})$ is the generalized Kullback-Leibler (KL) divergence, expressed as

$$f_0^B(\mathbf{h}, \mathbf{u}; \mathbf{g}) = KL(\mathbf{g}; \mathbf{h} * \mathbf{u} + b) \tag{4.4}$$

$$= \sum_i g_i \ln\left(\frac{g_i}{(\mathbf{h} * \mathbf{u} + b)_i}\right) + (\mathbf{h} * \mathbf{u} + b)_i - g_i, \tag{4.5}$$

where $g_i \ln g_i = 0$ if $g_i = 0$. Furthermore, f_1 and f_2 are the regularization functions of the PSF and object, respectively. In order to devise a meaningful solution between spurious local minima, simple physical constraints C_h and C_u for the two blocks of variables can be specified, as, for example, $C_h = \left\{\mathbf{h} \in \mathbb{R}^{d_h} : 0 \le h_i, \sum_{i=1}^{d_h} h_i = 1\right\}$ and $C_u = \{\mathbf{u} \in \mathbb{R}^{d_u} : 0 \le x_i, i = 1, ..., d_u\}$, with d_h and d_u related to the sizes of the convolution kernel and the images, respectively.

In all cases, the aim is to solve a constrained and non-convex minimization problem. Assuming that f_1 and f_2 are convex functions, since the objective function is convex with respect to each blocks of variables, keeping the other fixed, a standard approach to the solution of (4.3) is the so-called *alternating optimization method*, also known as *non-linear block Gauss-Seidel* or *block coordinate descent method* ([39, Chapter 2]); it consists in solving problem (4.3) by successively minimizing the objective function with respect to each block of variables, over the corresponding constraint set, by keeping the other fixed. Remarkable convergence results are given in [40] under the assumption that the exact solutions of the two minimization subproblems can be obtained at any iteration and, above all, in [41] where the stationarity of limit points of the sequence of inexact solutions is stated. This *inexact* block coordinate descent approach, where at each iteration a gradient projection step based on

[2]In Microscopy, a background emission can arise from auto-fluorescence, inadequate removal of fluorescence staining material, offset levels of the detector gain or other electronic sources [37, 38].

an Armijo line-search along the feasible direction with variable step-size is performed for any block, is very useful in the practical applications (see for example [42]). In the deterministic framework, many advances have been made to deal with BD problems in the case of natural images degraded by additive noise, by introducing novel regularization terms or suitable strategies to improve the alternating schemes (see, for example [43, 44, 45, 46] and the references therein).

Recently, deep Convolutional Neural Networks trained on large datasets are used to predict either the blur or the sharp image directly from the blurred examples (see [47, 48, 49, 50] and the references therein, and the surveys [51, 52]). In [53], the BD problem is addressed by training two separate generative networks, the first to produce sharp images and the second to generate blur kernels. Then, the BD problem is formulated as a minimization problem with respect to the inputs of the two pre-trained networks which provide as output the recovered image and the blur kernel; employing suitable regularization terms, this problem is solved by an alternating gradient descent scheme. A further proposal is to train only the network related to the blur kernel, using an unsupervised technique for training the network for the image. This technique is very similar to the one proposed in [54] and described in the upcoming sections. These data-driven methods are limited by the capacity of the training datasets and then they exhibit a lack of generalization when not trained with enough data. Moreover, in many real applications, it is practically impossible to build a sufficiently large dataset with both ground truth and degraded data.

Borrowing the DIP approach, it is natural to parameterized both the estimated image and the blur separately by generative networks, following the *Double-DIP* approach in [55]. These nets are not pre-trained on any datasets; they are adopted merely to capture the priors of either the image and the blur for the optimization process, performing a so-called Neural Blind Deconvolution. Indeed, this strategy is adopted by [54, 56]; the two approaches differ for the structure of the two nets. In particular in [56] the two nets are convolutional encoder-decoder networks, while in [54] only the image is the output of a convolutional U-Net and the blur kernel is generated by a more simple 2-layers fully connected net. This last method is called *SelfDeblur* and it is able to produce high-quality results for blurred images, but multiple runs can produce outputs with a wide range in quality. This is due to the random initialization of the neural nets, giving rise to a non-deterministic method. In [57] the authors improve the method in terms of consistency[3], by introducing a suitable initialization of the nets, multiscale processing and regularization. In [58], others techniques are discussed to address blurred and noisy images, also in presence of kernel with unknown size.

The contribution of this chapter is to tailor the ideas developed in [54, 57, 58] for NBD in confocal microscopy. In particular, the PSF $\mathbf{h}$ and the image $\mathbf{u}$ are derived from the parametric outputs of two generative networks $\mathcal{N}_h(\boldsymbol{\theta}_h)$ and $\mathcal{N}_u(\boldsymbol{\theta}_u)$; the problem (4.3) is reformulated so that the minimization is performed with respect to the weights $\boldsymbol{\theta}_h, \boldsymbol{\theta}_u$ of the two nets. For modeling $\mathbf{u}$, an autoencoder with skip connections $\mathcal{N}_u$ can be selected, so that $\mathbf{u} = \mathcal{N}_u(\boldsymbol{\theta}_u)$ while, for the more simple blur kernel $\mathbf{h}$, a fully connected network can be adopted, although the activation functions are sin functions and not standard ones. The input data of the two nets are random tensors. Following the suggestion in [42] (see also [29, Section 8.3]), in order to reduce the set of possible solutions of the NBD problem, a further constraint provided by the knowledge of the Strehl ratio[4] of the optical instrument is introduced. This information implies an upper bound on the PSF, in addition to the lower bound of non-negativity and to the normalization; therefore, with such a constraint, the trivial solution provided by the $\boldsymbol{\delta}$-array as PSF is not allowed. Suitable regularizations are introduced in the reformulated objective function. In view of a sharp image restoration, an edge-preserving non-smooth term is used, such as a Total Variation-like function. Furthermore, the minimization problem can be restate in the form of a nonconvex-concave min-max problem, similarly to what is described in [59]. For its numerical solution, a tailored version of the alternating *Proximal Gradient Descent-Ascent* (PGDA) method [60, 61] is developed. In order to drive the method to a meaningful solution, an initialization technique is needed to determine the weights

[3] In the context of DIP, the consistency ensures that the essential patterns, shapes, edges, and textures of an image are retained or reconstructed accurately while eliminating unwanted distortions or artifacts.

[4] The Strehl ratio is the ratio between the maximum value of an aberrated PSF versus that of a perfect PSF.

related to suitable starting values for $\mathbf{u}$ and $\mathbf{h}$. Lastly, the effectiveness of the proposed approach is evaluated by describing the numerical results of the NBD of synthetic images and of real images, showing the effect of the additional upper bound introduced on the blur kernel.

In details, the contributions in this chapter are organized as follows: Section 4.2 is devoted to present the problem, the network's structure and the choice for the regularization functionals; Section 4.3 presents the theoretical framework of the PGDA method employed for the minimization of (4.3) along with its implementation details in Algorithm 2; the numerical tests are presented in Section 4.4.

Notation The symbol $\|\cdot\|$ denotes the standard Euclidean norm, $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. Bold letters refer to vectors or tensors (it will be made clear by the context), whilst Greek and Latin letters refer to scalar. The Euclidean scalar product can be denoted by $\langle\cdot,\cdot\rangle$. The proximal operator of a convex function $F : \mathbb{R}^d \to \mathbb{R}$ at a vector $\mathbf{u} \in \mathbb{R}^d$ is defined as

$$\mathrm{prox}_F(\mathbf{u}) = \underset{\mathbf{w}\in\mathbb{R}^d}{\mathrm{argmin}} \left\{ F(\mathbf{w}) + \frac{1}{2}\|\mathbf{w}-\mathbf{u}\|^2 \right\}. \tag{4.6}$$

The proximal operator is well-defined for any convex function [62]. The proximal operator of the indicator function ι_C of a closed and convex set C at a vector $\mathbf{u}$ is the standard projection of $\mathbf{u}$ on C, i.e., $\mathrm{prox}_{\iota_C}(\mathbf{u}) = \Pi_C(\mathbf{u})$. Furthermore, for any vector $\mathbf{w} \in \mathbb{R}^2$, the shrinkage operator of $\mathbf{w}$ with parameter $\beta > 0$ is $\mathrm{shrink}_\beta(\mathbf{w}) = \underset{\mathbf{u}\in\mathbb{R}^2}{\mathrm{argmin}} \{\beta\|\mathbf{u}\| + \frac{1}{2}\|\mathbf{u}-\mathbf{w}\|^2\} = \frac{\mathbf{w}}{\|\mathbf{w}\|}\max(0, \|\mathbf{w}\| - \beta)$. The symbol $\mathbf{1}$ denotes an array with all entries equal to 1.

4.2 The Neural Blind Deconvolution problem

In a Bayesian approach, the BD problem can be formulated as the optimization problem (4.3), where, in presence of Poisson data, the data-fidelity function is expressed as (4.5). Motivated by the DIP idea [15, 16], following [54], in the BD problem the unknown blur kernel $\mathbf{h}$ and the image $\mathbf{u}$ can be replaced by the outputs of two generative networks $\mathcal{N}_h(\boldsymbol{\theta}_h)$ and $\mathcal{N}_u(\boldsymbol{\theta}_u)$, whose input data $\mathbf{z}_h$ and $\mathbf{z}_u$ are random samples. Combining this approach with the MAP minimization (4.3), the NBD problem can be formulated as

$$\min_{\boldsymbol{\theta}_h,\boldsymbol{\theta}_u} f_0^B(\mathcal{N}_h(\boldsymbol{\theta}_h), \mathcal{N}_u(\boldsymbol{\theta}_u); \mathbf{g}) + f_1\left(\mathcal{N}_h(\boldsymbol{\theta}_h)\right) + f_2\left(\mathcal{N}_u(\boldsymbol{\theta}_u)\right), \tag{4.7}$$

$$\text{s. t.} \quad 0 \le \mathcal{N}_u(\boldsymbol{\theta}_u) \le 1, \quad \mathcal{N}_h(\boldsymbol{\theta}_h) \ge 0, \quad \sum_i (\mathcal{N}_h(\boldsymbol{\theta}_h))_i = 1 \tag{4.8}$$

where f_1 and f_2 are regularization terms for the outputs of the two nets; the constraints in (4.8) can be automatically meet, by setting as output layers of the two nets the sigmoid nonlinearity for $\mathcal{N}_u$ and the SoftMax nonlinearity for $\mathcal{N}_h$ respectively. By denoting the solution of the minimization problem as $(\boldsymbol{\theta}_h^*, \boldsymbol{\theta}_u^*)$, the restored image $\mathbf{u}^*$ can be viewed as the output of the generative network $\mathcal{N}_u$ where the network weights $\boldsymbol{\theta}_u^*$ are a parametrization of $\mathbf{u}^*$, i.e., $\mathbf{u}^* = \mathcal{N}_u(\boldsymbol{\theta}_u^*)$. Similarly, the blur kernel $\mathbf{h}^* = \mathcal{N}_h(\boldsymbol{\theta}_h^*)$ and the network weights $\boldsymbol{\theta}_h^*$ are a parametrization of $\mathbf{h}^*$. The formulation (4.8) can include the trivial solution provided by the $\boldsymbol{\delta}$-array as PSF. Indeed, when an iterative method is used to find an reliable approximate solution, numerical experiments show how crucial it is to determine at which iteration one should stop, given that, in general, the iterative procedure can lead to the trivial solution. Hence, the numerical experience highlights that the sequence of iterates $\mathcal{N}_h(\theta_h^{(k)})$ tends to collapse to a single point with value 1 and, consequently, the quality of the recovered images deteriorates. This fact can be observed in the numerical experiments of Section 4.4, in particular in figures 4.3, 4.4 and 4.5. Here, for any test problem of the considered dataset, the peak values of the PSF sequence exceed the maximum value of $\mathbf{h}^*$, continuing to increase and collapsing the iterates into one point (see the behaviour of the PSF peak value with respect the iterations in figures 4.3(a), 4.4(a) and 4.5(a)). To avoid this drawback, following the suggestion in [42], the outputs of the network $\mathcal{N}_h$

are bounded from above by a prefixed peak value as well as being constrained by the non-negativity and normalization to 1. Thus, the problem (4.8) can be restated as follows:

$$\min_{\boldsymbol{\theta}_h,\boldsymbol{\theta}_u} f_0^B(\Pi_{C_{\mathbf{V}}}(\mathcal{N}_h(\boldsymbol{\theta}_h)),\mathcal{N}_u(\boldsymbol{\theta}_u);\mathbf{g}) + f_1\left(\Pi_{C_{\mathbf{V}}}(\mathcal{N}_h(\boldsymbol{\theta}_h))\right) + f_2\left(\mathcal{N}_u(\boldsymbol{\theta}_u)\right), \tag{4.9}$$

$$\text{s. t.} \quad 0 \leq \mathcal{N}_u(\boldsymbol{\theta}_u) \leq 1, \tag{4.10}$$

with $C_{\mathbf{V}} = \{0 \leq \mathbf{V} \leq H\mathbf{1}, \quad \sum_i(\mathbf{V})_i = 1\}$, $0 < H < 1$. While the output of the net $\mathcal{N}_u$ automatically satisfies the box constraints, the output of $\mathcal{N}_h$ has to be projected in the convex set, $C_{\mathbf{V}} = \{0 \leq \mathbf{V} \leq H\mathbf{1}, \quad \sum_i(\mathbf{V})_i = 1\}$; this condition requires to be able to compute the projection operator on the set $C_{\mathbf{V}}$ and this can be implemented by exploiting efficient algorithms, such as [63, 64, 65]. Unlike what holds for (4.8), in this case $\mathbf{h}^* = \Pi_{C_{\mathbf{V}}}(\mathcal{N}_h(\boldsymbol{\theta}_h^*))$. The peak value H is strictly dependent on the optical instrument; its physical features enable to define an ideal PSF. The ideal PSF is different from the real PSF and it is possible to estimate the Strehl ratio, finding a reliable value for H (see for example [66, 67]).

4.2.1 The two generative networks

The networks employed in the experimentation are inspired by the architectures used in [54, 57]. The network $\mathcal{N}_u$ is the U-Net described in Chapter 3 and illustrated by Figure 3.1. The seminal paper on DIP [16] points out that the choice of the upsampling method (bilinear, nearest neighbour or transposed convolution) does not impact on the final performance, but in this experimentation the transposed convolution works slightly better. The transposed convolution is then followed by a Batch Normalization and by a LeakyReLU. The input of this network is a random 3D tensor of dimension $n \times n \times 32$ (see Section 4.4 for the details of the probability distribution of such tensor). All the convolutional layers use convolutional filters of dimension 3×3.

In the original papers [54, 57] the network for recovering the PSF is a shallow, fully connected (FC) one: the motivation is based on the fact that the PSF does not have particular structures that require convolutional filters to be captured; hence a simple FC network is sufficient. The very recent work [58] implements the different architecture SIREN [68], which uses the sin function as neuronal activation. Consequently in the numerical experiments such strategy has been followed, since it has been observed in the numerical experiments and in real world scenarios that SIREN networks can learn high-frequency components in a better way than DIP [68]. The structure of the SIREN network used in this work is depicted in figure 4.1: an input layer of 200 random components is followed by four instances of a fully connected (FC) layer followed by a sin activation layer. The last two layers are a FC one (in order to retrieve the desired dimension for the PSF) and a SoftMax layer. The initialization of such network is done following the suggestion depicted in [68].

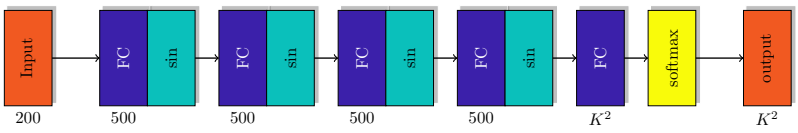


Figure 4.1: SIREN network for the reconstruction of the PSF. K^2 is the number of elements of the PSF.

4.2.2 Regularization terms

Although the generative nets have high impedance to image noise [69], numerical experience shows that multiple runs on the same input for the minimization of the objective data-fidelity function without regularization terms produce very different results [57]. This can be motivated by the random initialization of the nets, but also by the non-convexity of both the original problem and the reformulated one, that can admit many solutions. In order to address these drawbacks and make the approach more

consistent, in (4.3) and, consequently in (4.8) and (4.10), two regularization terms are introduced, as in [57, 58]. For both the terms, a pixel-wise weighted regularization is adopted (see [70, 71]).

It is assumed for simplicity to deal with a 2D $m \times m$ blur kernel, reshaped as vector of $\mathbb{R}^M$, $M = m \cdot m$, and $n \times n$ image, reshaped as vector of $\mathbb{R}^N$, $N = n \cdot n$. For the blur kernel an adaptive Tikhonov-like regularization of order 0 is considered, which is defined as the sum of a vector $\overline{\boldsymbol{\lambda}}$ of positive parameters, locally adapted to the kernel pattern, combined with the terms of the standard regularization, that is

$$f_1(\mathbf{h}) = \frac{1}{2} \sum_{i=1}^{M} \overline{\lambda}_i \mathbf{h}_i^2. \tag{4.11}$$

Similarly, to devise a restored sharp image, a pixel-wise weighted regularization term is used, locally adapted to the image pattern, combined with the terms of a standard regularization function, well-suited to microscopy images, such as the ℓ_1 function [72] or the Total Variation (TV) [59]. Specifically f_2 can be written as

$$f_2(\mathbf{u}) = \sum_{i=1}^{N} \tilde{\lambda}_i R_i(\mathbf{u}), \tag{4.12}$$

where $R_i(\mathbf{u})$ is defined as follows

$$R_i(\mathbf{u}) = |\mathbf{u}_i| \qquad \text{for } \ell_1 \text{ regularization,} \tag{4.13}$$

$$R_i(\mathbf{u}) = \|\mathbf{A}_i \mathbf{u}\| \qquad \text{for TV regularization.} \tag{4.14}$$

Here $\mathbf{A}_i$ is the 2D discrete first order difference operator at the pixel i of the vector-reshaped image $\mathbf{u} \in \mathbb{R}^N$ and the matrices $\mathbf{A}_i \in \mathbb{R}^{2\times N}$, $i = 1, ..., N$ are sub-matrices of a matrix $\mathbf{A} \in \mathbb{R}^{2N\times N}$, $\mathbf{A} = \left(\mathbf{A}_1^\top, \ldots, \mathbf{A}_N^\top\right)^\top$.

Both the selected regularization terms are convex, but the f_2 term is non-smooth. Thus, it should be inappropriate to use a gradient-like method, to address the problems (4.8) and (4.10), due to the non-existence of the f_2 gradient in neighborhoods with constant values of the image iterate. Nevertheless, the proximal operator of a suitable reformulation of f_2 can be computed by a closed formula. Consequently, the problem (4.10) is addressed by using a suitable scheme in the framework of alternating PGDA methods.

4.3 PGDA-like approach

Both the problems (4.8) and (4.10) can be restated as min-max problems, by introducing auxiliary variables. In particular, the case of problem (4.10) is discussed, pointing out that for (4.8) the reformulation is based on the introduction of only one auxiliary variable, the one which deals with the regularization term f_2. In the case of (4.10), two auxiliary variables must be considered, the first to realize the final projection of the output of $\mathcal{N}_h$ on $C_{\mathbf{V}}$ and the second to deal with the regularization term f_2. By considering as f_2 the case of the TV variant (4.12) and (4.14), the auxiliary variables are defined as follows:

$$\mathbf{V} = \mathcal{N}_h(\boldsymbol{\theta}_h) \in \mathbb{R}^M, \qquad \mathbf{Y} = \mathbf{A}\,\mathcal{N}_u(\boldsymbol{\theta}_u) \in \mathbb{R}^{2N}. \tag{4.15}$$

Taking into account that the constraints on $\mathcal{N}_u(\theta_u)$ in (4.10) are automatically meet by the output of the U-Net, the problem can be restated in the following form:

$$\min_{\boldsymbol{\theta}_h, \mathbf{V}, \boldsymbol{\theta}_u, \mathbf{Y}} \quad f_0^B(\mathbf{V}, \mathcal{N}_u(\boldsymbol{\theta}_u); \mathbf{g}) + \frac{1}{2} \sum_{i=1}^{M} \overline{\lambda}_i \mathbf{V}_i^2 + \iota_{C_{\mathbf{V}}}(\mathbf{V}) + \sum_{i=1}^{N} \tilde{\lambda}_i \|\mathbf{Y}_i\| \tag{4.16}$$

$$\text{s. t.} \quad \mathbf{V} = \mathcal{N}_h(\boldsymbol{\theta}_h), \quad \mathbf{Y} = \mathbf{A}\,\mathcal{N}_u(\boldsymbol{\theta}_u). \tag{4.17}$$

where $\iota_{C_{\mathbf{V}}}(\mathbf{V})$ is the indicator function of the convex set $C_{\mathbf{V}} = \{0 \leq \mathbf{V} \leq H\mathbf{1}, \quad \sum_i (\mathbf{V})_i = 1\}$. By introducing penalty terms for the equality constraints, the corresponding augmented Lagrangian

function can be written as

$$\mathcal{L}_{\gamma_{\mathbf{V}},\gamma_{\mathbf{Y}}}(\boldsymbol{\theta}_h,\mathbf{V},\boldsymbol{\theta}_u,\mathbf{Y},\boldsymbol{\mu}_{\mathbf{V}},\boldsymbol{\mu}_{\mathbf{Y}}) = f_0^B(\mathbf{V},\mathcal{N}_u(\boldsymbol{\theta}_u);\mathbf{g}) + \frac{1}{2}\sum_{i=0}^{M}\overline{\lambda}_i\mathbf{V}_i^2 + \iota_{C_{\mathbf{V}}}(\mathbf{V}) + \sum_{i=1}^{N}\tilde{\lambda}_i\|\mathbf{Y}_i\| + \tag{4.18}$$

$$+\frac{\gamma_{\mathbf{V}}}{2}\|\mathbf{V}-\mathcal{N}_h(\boldsymbol{\theta}_h)\|^2 + \langle\boldsymbol{\mu}_{\mathbf{V}},\mathbf{V}-\mathcal{N}_h(\boldsymbol{\theta}_h)\rangle + \tag{4.19}$$

$$+\frac{\gamma_{\mathbf{Y}}}{2}\|\mathbf{Y}-\mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u)\|^2 + \langle\boldsymbol{\mu}_{\mathbf{Y}},\mathbf{Y}-\mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u)\rangle\,, \tag{4.20}$$

where $\gamma_{\mathbf{V}}$ and $\gamma_{\mathbf{Y}}$ are positive penalty parameters and $\boldsymbol{\mu}_{\mathbf{V}}$, $\boldsymbol{\mu}_{\mathbf{Y}}$ are the multipliers of the equality constraints. By setting $\mathbf{P} = (\boldsymbol{\theta}_h,\mathbf{V},\boldsymbol{\theta}_u,\mathbf{Y})$ and $\boldsymbol{\mu} = (\boldsymbol{\mu}_{\mathbf{V}},\boldsymbol{\mu}_{\mathbf{Y}})$, the numerical solution of problem (4.10) is substituted by that of the following saddle point problem

$$\min_{\mathbf{P}}\max_{\boldsymbol{\mu}}\ \mathcal{L}_{\gamma_{\mathbf{V}},\gamma_{\mathbf{Y}}}(\mathbf{P},\boldsymbol{\mu}). \tag{4.21}$$

Furthermore the augmented Lagrangian function can be subdivided into two terms:

$$\mathcal{L}_{\gamma_{\mathbf{V}},\gamma_{\mathbf{Y}}}(\mathbf{P},\boldsymbol{\mu}) = L(\mathbf{P},\boldsymbol{\mu}) + R(\mathbf{P}), \tag{4.22}$$

where

$$L(\mathbf{P},\boldsymbol{\mu}) = f_0^B(\mathbf{V},\mathcal{N}_u(\boldsymbol{\theta}_u);\mathbf{g}) + \frac{1}{2}\sum_{i=0}^{M}\overline{\lambda}_i\mathbf{V}_i^2 + \tag{4.23}$$

$$+\frac{\gamma_{\mathbf{V}}}{2}\|\mathbf{V}-\mathcal{N}_h(\boldsymbol{\theta}_h)\|^2 + \langle\boldsymbol{\mu}_{\mathbf{V}},\mathbf{V}-\mathcal{N}_h(\boldsymbol{\theta}_h)\rangle + \tag{4.24}$$

$$+\frac{\gamma_{\mathbf{Y}}}{2}\|\mathbf{Y}-\mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u)\|^2 + \langle\boldsymbol{\mu}_{\mathbf{Y}},\mathbf{Y}-\mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u)\rangle \tag{4.25}$$

$$R(\mathbf{P}) = \sum_{i=1}^{N}\tilde{\lambda}_i\|\mathbf{Y}_i\| + \iota_{C_{\mathbf{V}}}(\mathbf{V}). \tag{4.26}$$

The function $L(\mathbf{P},\boldsymbol{\mu}_{\mathbf{Y}})$ is smooth; the non-smooth $R(\mathbf{P})$ term is actually a convex separable function of $\mathbf{V}$ and $\mathbf{Y}$; there exists a closed formula to compute the proximal operator of each term $\|\mathbf{Y}_i\|$, $i = 1,...,N$, whereas the proximity operator of $\iota_{C_{\mathbf{V}}}(\mathbf{V})$ is the projection on the convex set $C_{\mathbf{V}}$, numerically obtainable by means of efficient algorithms, as for example the ones in [63, 64, 65]. Thus the saddle point problem (4.21) can be formulated as

$$\min_{\mathbf{P}}\max_{\boldsymbol{\mu}}\ L(\mathbf{P},\boldsymbol{\mu}) + R(\mathbf{P}), \tag{4.27}$$

which is the typical form of the problems addressed by alternating PGDA methods, recently developed in [60, 61] and already adopted for DIP models in [59]. Upon suitable initialization of the variables, the k-th iteration of PGDA iterative algorithm reads as

$$\mathbf{P}^{(k+1)} = \mathrm{prox}_{\alpha_{\mathbf{P}}R}(\mathbf{P}^{(k)} - \alpha_{\mathbf{P}}\nabla_{\mathbf{P}}K(\mathbf{P}^{(k)},\boldsymbol{\mu}^{(k)})) \tag{4.28}$$

$$\boldsymbol{\mu}^{(k+1)} = \boldsymbol{\mu}^{(k)} + \alpha_{\boldsymbol{\mu}}\nabla_{\boldsymbol{\mu}}K(\mathbf{P}^{(k+1)},\boldsymbol{\mu}^{(k+1)}), \tag{4.29}$$

where $\alpha_{\mathbf{P}}$ and $\alpha_{\boldsymbol{\mu}}$ are positive step lengths (learning rates). Due to the separability of the functions $L(\mathbf{P}, \boldsymbol{\mu})$ and $R(\mathbf{P})$, the general scheme (4.29) can be detailed as follows:

$$\boldsymbol{\theta}_h^{(k+1)} = \boldsymbol{\theta}_h^{(k)} - \alpha_{\mathbf{P}} \nabla_{\boldsymbol{\theta}_h} \left[\frac{\gamma_{\mathbf{V}}}{2} \|\mathbf{V}^{(k)} - \mathcal{N}_h(\boldsymbol{\theta}_h^{(k)})\|^2 + \left\langle \boldsymbol{\mu}_{\mathbf{V}}^{(k)}, \mathbf{V}^{(k)} - \mathcal{N}_h(\boldsymbol{\theta}_h^{(k)}) \right\rangle \right]$$

$$\begin{aligned}\mathbf{V}^{(k+1)} = \operatorname{prox}_{\alpha_{\mathbf{P}} \iota_{C_{\mathbf{V}}}} \Big(\mathbf{V}^{(k)} - \alpha_{\mathbf{P}} \nabla_{\mathbf{V}} \Big[f_0^B(\mathbf{V}^{(k)}, \mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}); \mathbf{g}) + \frac{1}{2} \sum_{i=0}^{M} \overline{\lambda}_i \mathbf{V}_i^2 + \\ \frac{\gamma_{\mathbf{V}}}{2} \|\mathbf{V}^{(k)} - \mathcal{N}_h(\boldsymbol{\theta}_h^{(k)})\|^2 + \left\langle \boldsymbol{\mu}_{\mathbf{V}}^{(k)}, \mathbf{V}^{(k)} - \mathcal{N}_h(\boldsymbol{\theta}_h^{(k)}) \right\rangle \Big] \Big)\end{aligned}$$

$$\begin{aligned}\boldsymbol{\theta}_u^{(k+1)} = \boldsymbol{\theta}_u^{(k)} - \alpha_{\mathbf{P}} \nabla_{\boldsymbol{\theta}_u} \Big[f_0^B(\mathbf{V}^{(k)}, \mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}); \mathbf{g}) + \\ + \frac{\gamma_{\mathbf{Y}}}{2} \|\mathbf{Y}^{(k)} - \mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u^{(k)})\|^2 + \left\langle \boldsymbol{\mu}_{\mathbf{Y}}^{(k)}, \mathbf{Y}^{(k)} - \mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}) \right\rangle \Big]\end{aligned}$$

$$\begin{aligned}\mathbf{Y}^{(k+1)} = \operatorname{prox}_{\alpha_{\mathbf{P}} f_2} \Big(\mathbf{Y}^{(k)} - \alpha_{\mathbf{P}} \nabla_{\mathbf{Y}} \Big[\frac{\gamma_{\mathbf{Y}}}{2} \|\mathbf{Y}^{(k)} - \mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u^{(k)})\|^2 + \\ + \left\langle \boldsymbol{\mu}_{\mathbf{Y}}^{(k)}, \mathbf{Y}^{(k)} - \mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}) \right\rangle \Big] \Big)\end{aligned}$$

$$\boldsymbol{\mu}_{\mathbf{V}}^{(k+1)} = \boldsymbol{\mu}_{\mathbf{V}}^{(k)} + \alpha_{\boldsymbol{\mu}} \left(\mathbf{V}^{(k+1)} - \mathcal{N}_h(\boldsymbol{\theta}_h^{(k+1)}) \right)$$

$$\boldsymbol{\mu}_{\mathbf{Y}}^{(k+1)} = \boldsymbol{\mu}_{\mathbf{Y}}^{(k)} + \alpha_{\boldsymbol{\mu}} \left(\mathbf{Y}^{(k+1)} - \mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u^{(k+1)}) \right).$$

Then, the explicit formulation of the above iteration is depicted as follows:

$$\boldsymbol{\theta}_h^{(k+1)} = \boldsymbol{\theta}_h^{(k)} - \alpha_{\mathbf{P}} \nabla_{\boldsymbol{\theta}_h} \mathcal{N}_h(\boldsymbol{\theta}_h^{(k)}) \left(\gamma_{\mathbf{V}} (\mathcal{N}_h(\boldsymbol{\theta}_h^{(k)}) - \mathbf{V}^{(k)}) - \boldsymbol{\mu}_{\mathbf{V}}^{(k)} \right) \tag{4.30}$$

$$\begin{aligned}\mathbf{V}^{(k+1)} = \Pi_{C_{\mathbf{V}}} \Big(\mathbf{V}^{(k)} - \alpha_{\mathbf{P}} (\nabla_{\mathbf{V}} [f_0^B(\mathbf{V}^{(k)}, \mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}); \mathbf{g})] + \\ + \sum_{i=0}^{M} \overline{\lambda}_i \mathbf{V}_i + \gamma_{\mathbf{V}} (\mathbf{V}^{(k)} - \mathcal{N}_h(\boldsymbol{\theta}_h^{(k)})) + \boldsymbol{\mu}_{\mathbf{V}}^{(k)}) \Big)\end{aligned} \tag{4.31}$$

$$\begin{aligned}\boldsymbol{\theta}_u^{(k+1)} = \boldsymbol{\theta}_u^{(k)} - \alpha_{\mathbf{P}} \nabla_{\boldsymbol{\theta}_u} \mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}) \left(\nabla_{\mathcal{N}_u(\boldsymbol{\theta}_u)} \left[f_0^B(\mathbf{V}^{(k)}, \mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}); \mathbf{g}) \right] + \right. \\ \left. - \gamma_{\mathbf{Y}} \mathbf{A}^\top \left(\mathbf{Y}^{(k)} - \mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}) + \frac{\boldsymbol{\mu}_{\mathbf{Y}}}{\gamma_{\mathbf{Y}}} \right) \right)\end{aligned} \tag{4.32}$$

$$\mathbf{Y}_i^{(k+1)} = \underset{\alpha_{\mathbf{P}} \tilde{\lambda}_i}{\operatorname{shrink}} \left(\mathbf{Y}_i^{(k)} - \alpha_{\mathbf{P}} \gamma_{\mathbf{Y}} \left(\mathbf{Y}_i^{(k)} - \mathbf{A}_i \mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}) \right) - \alpha_{\mathbf{P}} (\boldsymbol{\mu}_{\mathbf{Y}}^{(k)})_i \right) \tag{4.33}$$

$$\boldsymbol{\mu}_{\mathbf{V}}^{(k+1)} = \boldsymbol{\mu}_{\mathbf{V}}^{(k)} + \alpha_{\boldsymbol{\mu}} \left(\mathbf{V}^{(k+1)} - \mathcal{N}_h(\boldsymbol{\theta}_h^{(k+1)}) \right) \tag{4.34}$$

$$\boldsymbol{\mu}_{\mathbf{Y}}^{(k+1)} = \boldsymbol{\mu}_{\mathbf{Y}}^{(k)} + \alpha_{\boldsymbol{\mu}} \left(\mathbf{Y}^{(k+1)} - \mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u^{(k+1)}) \right) \tag{4.35}$$

where the shrinkage operator is applied to any sub-vector $\mathbf{Y}_i \in \mathbb{R}^2$ corresponding to $\mathbf{A}_i\mathcal{N}_u(\boldsymbol{\theta}_u)$, $i = 1, ..., N$. The steps (4.30) and (4.32) can be easily approximated by one step of a stochastic gradient method applied to the functions $\frac{\gamma_{\mathbf{V}}}{2}\|\mathbf{V}^{(k)} - \mathcal{N}_h(\boldsymbol{\theta}_h)\|^2 + \left\langle \boldsymbol{\mu}_{\mathbf{V}}^{(k)}, \mathbf{V}^{(k)} - \mathcal{N}_h(\boldsymbol{\theta}_h) \right\rangle$ and $f_0^B(\mathbf{V}^{(k)}, \mathcal{N}_u(\boldsymbol{\theta}_u); \mathbf{g}) + \frac{\gamma_{\mathbf{Y}}}{2}\|\mathbf{Y}^{(k)} - \mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u)\|^2 + \left\langle \boldsymbol{\mu}_{\mathbf{Y}}^{(k)}, \mathbf{Y}^{(k)} - \mathbf{A}\mathcal{N}_u(\boldsymbol{\theta}_u) \right\rangle$ with learning rate $\alpha_{\mathbf{P}}$, respectively; the updating rule (4.31) for $\mathbf{V}$ can be obtained by applying the algorithm proposed in [64]; in view of the physical assumption $MH > 1$, the projection sub-problem is well-defined; the computation of the proximal operator related to the regularization term (4.12) (with parameter $\alpha_{\mathbf{P}}$) can be easily obtained in a closed form; the updating rule is well-defined, since f_2 is a convex function. Finally, the last updating rules for the multiplier vectors are ascent steps with learning rate $\alpha_{\boldsymbol{\mu}}$.

Remark 1. *The selection of parameters $\overline{\lambda}_i$ and $\tilde{\lambda}_i$ is described in [59, 70] and they vary along the iterations, according to the Uniform PENalty principle [72], in particular*

$$\overline{\lambda}_i^{(k)} = \frac{1}{M}\frac{f_0^B(\mathbf{V}^{(k)}, \mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}); \mathbf{g})}{(\mathbf{V}_i^{(k)})^2}, \quad \tilde{\lambda}_i^{(k)} = \frac{1}{2N}\frac{f_0^B(\mathbf{V}^{(k)}, \mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}); \mathbf{g})}{\|\mathbf{A}_i\mathcal{N}_u(\boldsymbol{\theta}_u^{(k)})\|}. \tag{4.36}$$

As a consequence, the pixels where the value of the local components $(\mathbf{V}_i^{(k)})^2$ and $\|\mathbf{A}_i\mathcal{N}_u(\boldsymbol{\theta}_u^{(k)})\|$ are smaller, have a greater regularization. In the experimentation, the denominator is modified as $\frac{1}{2}(\mathbf{V}_i^{(k)})^2 + \rho^2$ and $\|\mathbf{A}_i\mathcal{N}_u(\boldsymbol{\theta}_u^{(k+1)}) + \rho^2\|$ respectively, with $\rho \ll 1$, in order to avoid the annihilation of the denominators.

Remark 2. *When the regularization term* (4.12)-(4.13) *is adopted for the image, the auxiliary variable $\mathbf{Y}$ is defined as the N dimensional vector $\mathbf{Y} = \mathcal{N}_u(\boldsymbol{\theta}_u)$. Consequently, the formulation of the problem* (4.17) *and the definition of augmented Lagrangian function* (4.20) *are accordingly simplified. The discussion on the implementation of PGDA remains unchanged, by adopting small modifications in* (4.32)*,* (4.33) *and* (4.35) *(replacement of $\mathbf{A}$ with the identity of order N and consistent computation of the proximal operator). Furthermore, the definition of the regularization parameters is*

$$\tilde{\lambda}_i^{(k)} = \frac{1}{2N}\frac{f_0^B(\mathcal{N}_h(\boldsymbol{\theta}_h^{(k)}), \mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}); \mathbf{g})}{|(\mathcal{N}_u(\boldsymbol{\theta}_u^{(k)}))_i| + \rho^2}, \tag{4.37}$$

with $\rho \ll 1$, to prevent the annihilation of the denominator.

Remark 3. *The theoretical properties of the alternating PGDA method are discussed in [60]. With reference to* (4.22)*, under the assumptions that $L(\mathbf{P}, \boldsymbol{\mu})$ is ρ-weakly convex ($\rho > 0$) and Lipschitz continuous in $\mathbf{P}$ uniformly in the second component, concave with Lipschitz continuous gradient in $\boldsymbol{\mu}_{\mathbf{Y}}$ uniformly in the first component, and $R(\mathbf{P})$ is proper, convex, lower semicontinous and Lipschitz continuous on its domain, it is stated that an ε-stationary point [60] can be visited in a finite number of iterations depending on ε. The learning rates $\alpha_{\mathbf{P}}$ and $\alpha_{\boldsymbol{\mu}}$ have to satisfy upper bounds involving the unknown Lipschitz parameters. In practice, sufficiently small values of the two learning rates are fixed a priori.*

4.3.1 Starting vectors

In order to obtain meaningful results by the iterative scheme (4.30)–(4.35), a crucial point is to initialize the weights of the nets $\mathcal{N}_h$ and $\mathcal{N}_u$ so that the deblurring loop has a suitable starting point (see [57] for more insights and deeper explanation). Thus, $\mathbf{u}^{(0)} = \mathbf{g}$, whereas $\mathbf{h}^{(0)}$ is chosen accordingly to the application at hand, for example a Gaussian kernel or via a preset model (see Section 4.4 for more details). To initialize the parameters of the two nets, the following mean square error problems must be solved

$$\min_{\boldsymbol{\theta}_h} \|\mathcal{N}_h(\boldsymbol{\theta}_h) - \mathbf{h}^{(0)}\|^2, \tag{4.38}$$

$$\min_{\boldsymbol{\theta}_u} \|\mathcal{N}_u(\boldsymbol{\theta}_u) - \mathbf{u}^{(0)}\|^2. \tag{4.39}$$

Problems (4.38),(4.39) are quickly solved via the Adam method: the procedure is stopped upon reaching the maximum number of iteration T_h, T_u for (4.38) and (4.39), respectively, or when the loss function reaches a value at most equal to a percentage of its initial values (in the experiments, 0.1%).

4.3.2 Algorithm details

The whole procedure of the Double Deep Image Prior for Poisson data (DDIPP) is depicted in Algorithm 2.

Instead of running the algorithm until it reaches the maximum number of iteration T, there exists an implementation of the windowed-moving-variance strategy of [58, 73]. Such strategy consists in storing, at iteration k, the last $W \in \mathbb{N}$ recovered images in a queue $\mathcal{Q}$: $\{\mathcal{N}_u(\boldsymbol{\theta}_u^{(k-i)})\}_{i=0,\dots,W-1}$ and compute their variance. The last $p \in \mathbb{N}$ minimum variances are memorized: if these p variances stagnate, then the procedure is stopped and the output consists in the last achieved recovered image and its related PSF. The parameter p takes the name of *patience*. The variance at iteration k has the following expression

$$\mathrm{WMV}(k) = \frac{1}{W}\sum_{i=0}^{W-1}\left\|\mathbf{u}^{(k-i)} - \frac{1}{W}\sum_{j=0}^{W-1}\mathbf{u}^{(k-j)}\right\|_{\mathrm{F}}^2, \tag{4.40}$$

where $\mathbf{u}^{(k)} = \mathcal{N}_u(\boldsymbol{\theta}_u^{(k)})$. The implementation of this procedure consists in checking that the last p minimum variances have an absolute difference less than a given tolerance ε_p.
A further suggestion about the stopping of the DDIPP can be provided from [29, Lemma 4.1] (see also [74, 75, 76]), where it is stated that, since $\mathbf{g}_i$ can be viewed as a realization of a Poisson random variable with expected value $(\mathbf{h} * \mathbf{u} + b)_i$, if $\mathbf{g}_i$ is sufficiently large, the expected value of $2\left(\mathbf{g}_i \log\left(\frac{\mathbf{g}_i}{(\mathbf{h}*\mathbf{u}+b)_i}\right) + (\mathbf{h} * \mathbf{u} + b)_i - \mathbf{g}_i\right)$ is approximately 1. When the current values of $\mathbf{V}^{(k)}$ and $\mathbf{u}^{(k)}$ are near to the solutions, one can expect that the value of normalized discrepancy, i.e.,

$$\mathcal{D}(k) = \frac{2}{N} f_0^B(\mathbf{V}^{(k)}, \mathbf{u}^{(k)}; \mathbf{g}), \tag{4.41}$$

decreases and tends to fluctuate around 1. As remarked in [29], the use of the normalized discrepancy is strictly related to the assumption that the data satisfy Poisson statistics and that the blur kernel provides is a good approximation of the blur process. However, it seems reasonable to assume that the region in which an approximate solution is found is the one for which $\mathcal{D}(k) < S$, with S of the order of the unity, for example $S = 2$. The stopping test based on the windowed-moving-variance strategy can be performed only when the normalized discrepancy is less than S. As shown in the following numerical experiments, this strategy enables us to avoid to prematurely stop the method: indeed, it may happen that in the early iterations the recovered images are actually not completely reliable but at the same time their differences are minimal. This leads the criterion (4.40) to be satisfied, but the reconstruction is still far away from a reliable solution.

Remark 4. *Two technical observations about the actual implementation of the proposed procedure are reported below.*

1. *Algorithm 2 requires a gradient step for the update of the networks' parameters. The numerical experience showed that such simple step is not sufficient to achieve reliable results in a reasonable amount of time, hence Adam algorithm is suitable for updating the nets' parameters in Algorithm 2.*

2. *The numerical experience showed that updating $\mathbf{Y}^{(k+1)}$ using $\mathcal{N}_u(\boldsymbol{\theta}_u^{(k+1)})$ instead of $\mathcal{N}_u(\boldsymbol{\theta}_u^{(k)})$ induces an appreciable speed-up.*

Algorithm 2: DDIPP: Double DIP for Poisson noise

Set the parameters $\alpha_{\mathbf{P}}, \alpha_{\boldsymbol{\mu}}, \gamma_{\mathbf{V}}, \gamma_{\mathbf{Y}}$;
Set $r_{\min} = +\infty$ and choose ε_p and S;
Choose the length W of the queue $\mathcal{Q}$ and the patience p; choose T_h, T_u, T;
Initialise $\mathcal{N}_u$ solving (4.39) with a max number of iterations equal to T_u;
Initialise $\mathcal{N}_h$ solving (4.38) with a max number of iterations equal to T_h;
for $k = 1, 2, \ldots, T$ **do**
  Update $\boldsymbol{\theta}_h^{(k+1)}$ as in (4.30);
  Update $\mathbf{V}^{(k+1)}$ as in (4.31);
  Update $\boldsymbol{\theta}_u^{(k+1)}$ as in (4.32);
  Store $\mathcal{N}_u(\boldsymbol{\theta}_u^{(k+1)})$ in the queue $\mathcal{Q}$;
  if $|\mathcal{Q}| \geq W$ **then**
    Compute the WMV of the last W elements in $\mathcal{Q}$;
    $\hat{\mathbf{u}} = \mathcal{N}_u(\boldsymbol{\theta}_u^{(k+1)})$;
    $\hat{\mathbf{h}} = \mathbf{V}^{(k+1)}$;
    if *WMV*$< \min\{r_{\min}\}$ **then**
      $r_{\min} \hookleftarrow$WMV;
    end
    if $\sum$ *diff*$(r_{\min}) < \varepsilon_p(p-1)$ *and* $\mathcal{D}(k) < S$ **then**
      Output $\hat{\mathbf{u}}, \hat{\mathbf{h}}$;
    end
  end
  Set $\bar{\lambda}_i$ as in (4.36);
  Update $\mathbf{Y}^{(k+1)}$ as in (4.33), by using $\boldsymbol{\theta}_u^{(k+1)}$;
  Update $\boldsymbol{\mu}_{\mathbf{V}}^{(k+1)}$ as in (4.34);
  Update $\boldsymbol{\mu}_{\mathbf{Y}}^{(k+1)}$ as in (4.35);
end

4.4 Numerical experiments

This section is devoted to assess the performance of the proposed procedure on synthetic images and on real microscopy images. The code has been implemented in MATLAB® R2022b, using the Deep Learning and Image Processing toolboxes. The code is available online at `https://github.com/AleBenfe/DDIPP`. Regarding the computational times of Algorithm 2, it should be noted that the code has been implemented in MATLAB, a language that is not ideally suited for coding methods involving artificial neural networks due to its limited support for advanced machine learning frameworks. Consequently, the implementation can be translated into Python, which is more appropriate for such tasks, as it offers a wide range of libraries optimized for artificial intelligence and neural network development, such as TensorFlow, PyTorch, and Keras. Additionally, Python's compatibility with GPU acceleration and parallel processing can significantly reduce computation times, further improving the efficiency and scalability of the algorithm. The settings for Algorithm 2 and its simplified version for model (4.8) are detailed in the following paragraphs; the steplengths $\alpha_{\mathbf{P}}$ and $\alpha_{\boldsymbol{\mu}}$ are set to small values in order to ensure the convergence of PGDA, without a special tuning to decrease the number of iterations; furthermore, in all cases, $T_h = T_u = 200$ and the maximum number T of iteration is 1000. The random inputs of the networks, namely $\mathbf{z}_u$ for $\mathcal{N}_u$ and $\mathbf{z}_h$ for $\mathcal{N}_h$, are initially drawn from a uniform distribution in $[0, 1]$ and in $[-1, 1]$ respectively; at each iteration $\mathbf{z}_u$ is perturbed with Gaussian noise with zero mean and variance equal to 0.001 whilst $\mathbf{z}_h$ is left unperturbed, according to the strategy depicted in [54, 57]. The patience p is set to 20 and the queue length W is 10, $\varepsilon_p = 10^{-3}$

and $S = 2$. The regularization parameters for the blur kernel and the image are adaptively updated as in (4.36)-(4.37). Following the strategy depicted in [58], the size of the recovered PSF is overestimated, depending on the image size, even if, in case of synthetic data, the true PSF has been generated with a lower number of pixels. For synthetic data, the Peak Signal-to-Noise Ratio (PSNR) between the recovered data and the ground truths $\mathbf{u}^*, \mathbf{h}^*$ is employed as a figure of merit, as well as the Structural Similarity Index Measure (SSIM) of the reconstructed image with respect to $\mathbf{u}^*$.
In the following, all the images are displayed in the range $[0, 1]$, whereas the renderings of the blur kernel show the scale on the right.

4.4.1 Synthetic dataset

The first experimentation is carried on the following dataset, where each blurred image is perturbed with Poisson noise via the MATLAB® function `imnoise`:

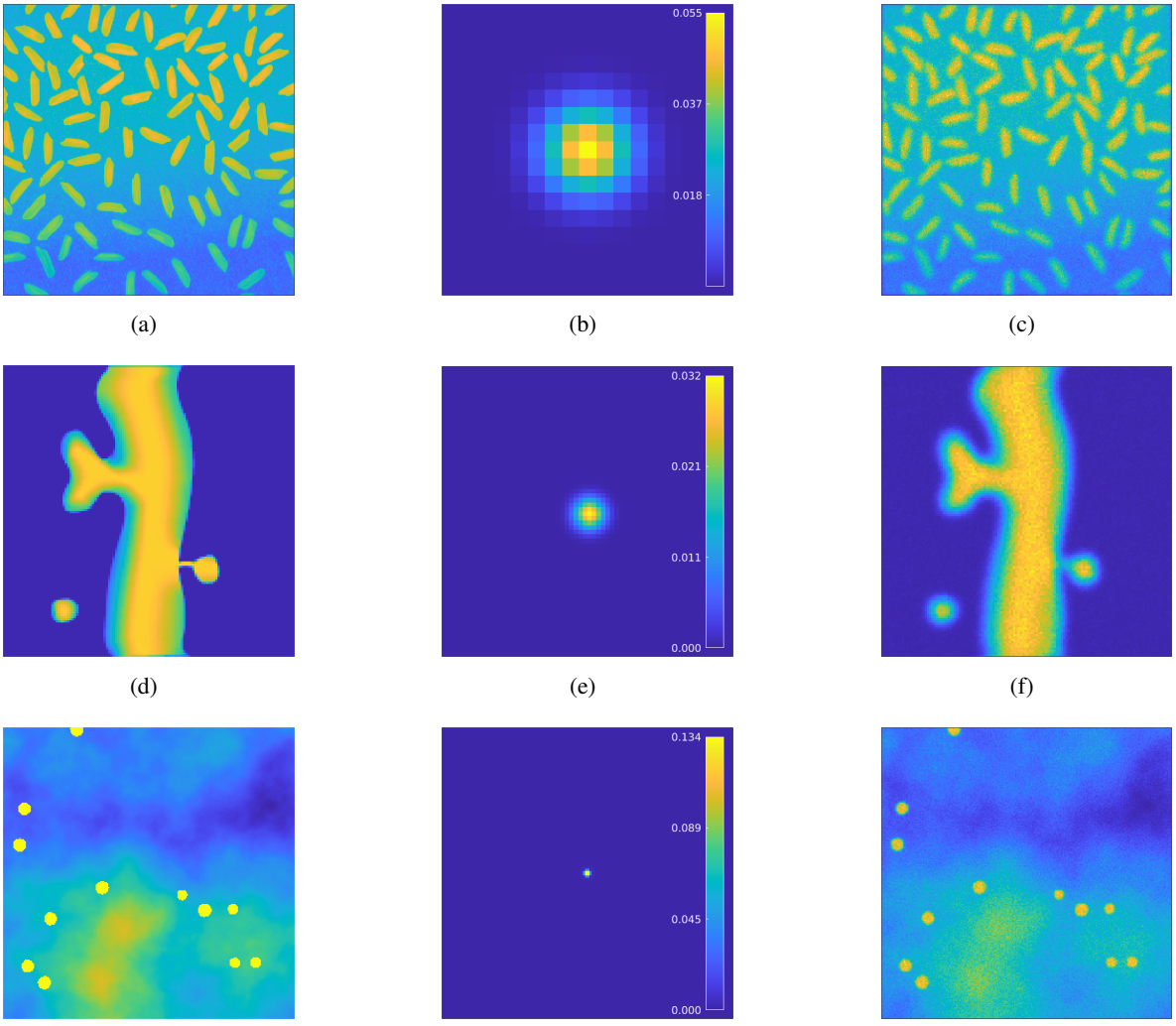
- `rice`: this image belongs to the MATLAB® Image Processing Toolbox. It is a 256×256 image with pixel values in $[0, 255]$; the blur operator consists in a Gaussian kernel with size 17×17 and standard deviation equal to 1.7; the peak value of the PSF is 0.055; furthermore, $PSNR(\mathbf{g}) = 22.67$, $SSIM(\mathbf{g}) = 0.51$.
- `micro`: the original image is a phantom of size 128×128 described in [77]; the PSF consists in a Gaussian blur with standard deviation equal to $\sqrt{5}$ and peak value 0.032; furthermore, $PSNR(\mathbf{g}) = 24.51$, $SSIM(\mathbf{g}) = 0.84$.
- `synth001`: this is a synthetic simulation of real-world microscopy images; the procedure employed for the generation of such image is explained in [78] and the code is available at `https://github.com/AleBenfe/upU-net_Perlin`. The PSF used for blurring these images is obtained via the software available at `http://bigwww.epfl.ch/algorithms/psfgenerator/` (see [79, 80, 81] for more technical details). A 3D PSF (with size $64 \times 64 \times 11$) is generated using the Born and Wolf 3D model: then the central frame is normalized to 1 and used as blur operator; the peak value of PSF is 0.134; furthermore, $PSNR(\mathbf{g}) = 21.96$, $SSIM(\mathbf{g}) = 0.52$.

Figure 4.2 provides a visual inspection of the dataset.

Comparison between the models (4.8) **and** (4.10)**.** The first experiment shows the comparison between the model (4.8) and the novel reformulation in (4.10), in order to check the effectiveness of the additional constraint on the output of $\mathcal{N}_h$. For both the models, the regularization terms (4.11) and (4.12),(4.14) are used for the PSF and the image respectively. In the following, this choice is denoted with the suffix $\ell_2 - TV$.
In both cases, the numerical solution is addressed with the alternating PGDA method: in the latter the implementation of Algorithm (2) is used, while in the former a tailored version of this algorithm for the model (4.8) is applied. In Figures 4.3, 4.4 and 4.5 there are for both the models the behaviour of the PSNR of the PSF and of the image, i.e., $PSNR(\mathbf{h}^{(k)})$ and $PSNR(\mathbf{u}^{(k)})$, the images and the blur kernels recovered at the iteration K where the stopping criterion is satisfied (or at an intermediate iteration if it is not satisfied), and at the last iteration T. Table 4.1 shows some figures of merit of the recovered images and PSFs the three test problems for both models (4.8) and (4.10) $\ell_2 - TV$. For test problems `rice` and `synth001`, the stopping criterion of PGDA is not satisfied in the case of model (4.8). Consequently, for `rice` and `synth001` the results related to the iteration 200 and 240, respectively, are reported.

From the Table 4.1 and the previous figures, it clearly appears that:

- the plots of the behaviour of PSNRs show the convergent trend of the method PGDA, although this convergence is very slow due to the small value of the steplengths; for the model (4.10) the differences between the recovered images at the iteration K and at the iteration T are visually unappreciable, showing that the stopping criterion seems effective;



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 4.2: (a): ground truth image of `rice` test problem (256×256). (b) PSF of `rice` test problem (peak value 0.055). (c) noisy blurred image $\mathbf{g}$ for `rice` test problem. (d): ground truth image of `micro` test problem (128×128). (e) PSF of `micro` test problem (peak value 0.032). (f) noisy blurred image $\mathbf{g}$ for `micro` test problem. (g): ground truth image of `synth001` test problem (512×512). (h) PSF of `synth001` test problem (peak value 0.134). (i) noisy blurred image $\mathbf{g}$ for `synth001` test problem.

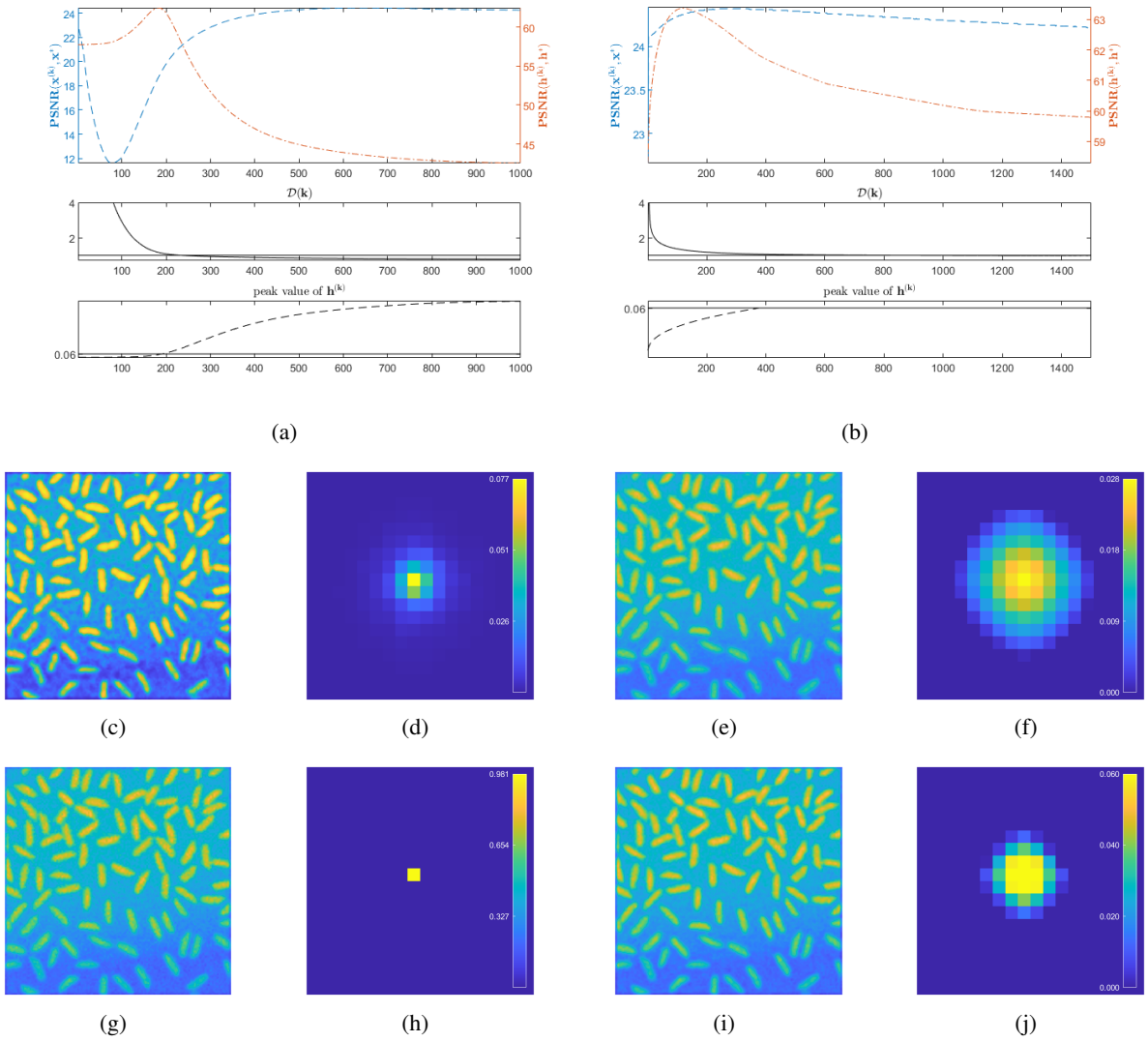


Figure 4.3: `rice` test problem - The panels on the left show the behaviour of PGDA method for the model (4.8) $\ell_2 - TV$ ($\alpha_{\mathbf{P}} = 5 \cdot 10^{-5}, \alpha_{\boldsymbol{\mu}} = 5 \cdot 10^{-4}, \gamma_{\mathbf{Y}} = 1$): (a) PSNR of $\mathbf{u}^{(k)}$ and $\mathbf{h}^{(k)}$, normalized discrepancy $\mathcal{D}(k)$ and peak value of $\mathbf{h}^{(k)}$, (c)-(d) reconstructions of the image and the PSF at the iteration 200, (g)-(h) reconstructions of the image and the PSF at the iteration $T = 1000$. The panels on the right show the behaviour of PGDA method for the model (4.10) $\ell_2 - TV$: ($\alpha_{\mathbf{P}} = 10^{-6}, \alpha_{\boldsymbol{\mu}} = 10^{-5}, \gamma_{\mathbf{Y}} = 1, \gamma_{\mathbf{V}} = 10^{-3}$): (b) PSNR of $\mathbf{u}^{(k)}$ and $\mathbf{h}^{(k)}$, normalized discrepancy $\mathcal{D}(k)$ and peak value of $\mathbf{h}^{(k)}$, (e)-(f) reconstructions of the image and the PSF at the iteration $K = 60$, (i)-(j) reconstructions of the image and the PSF at the iteration $T = 1000$.

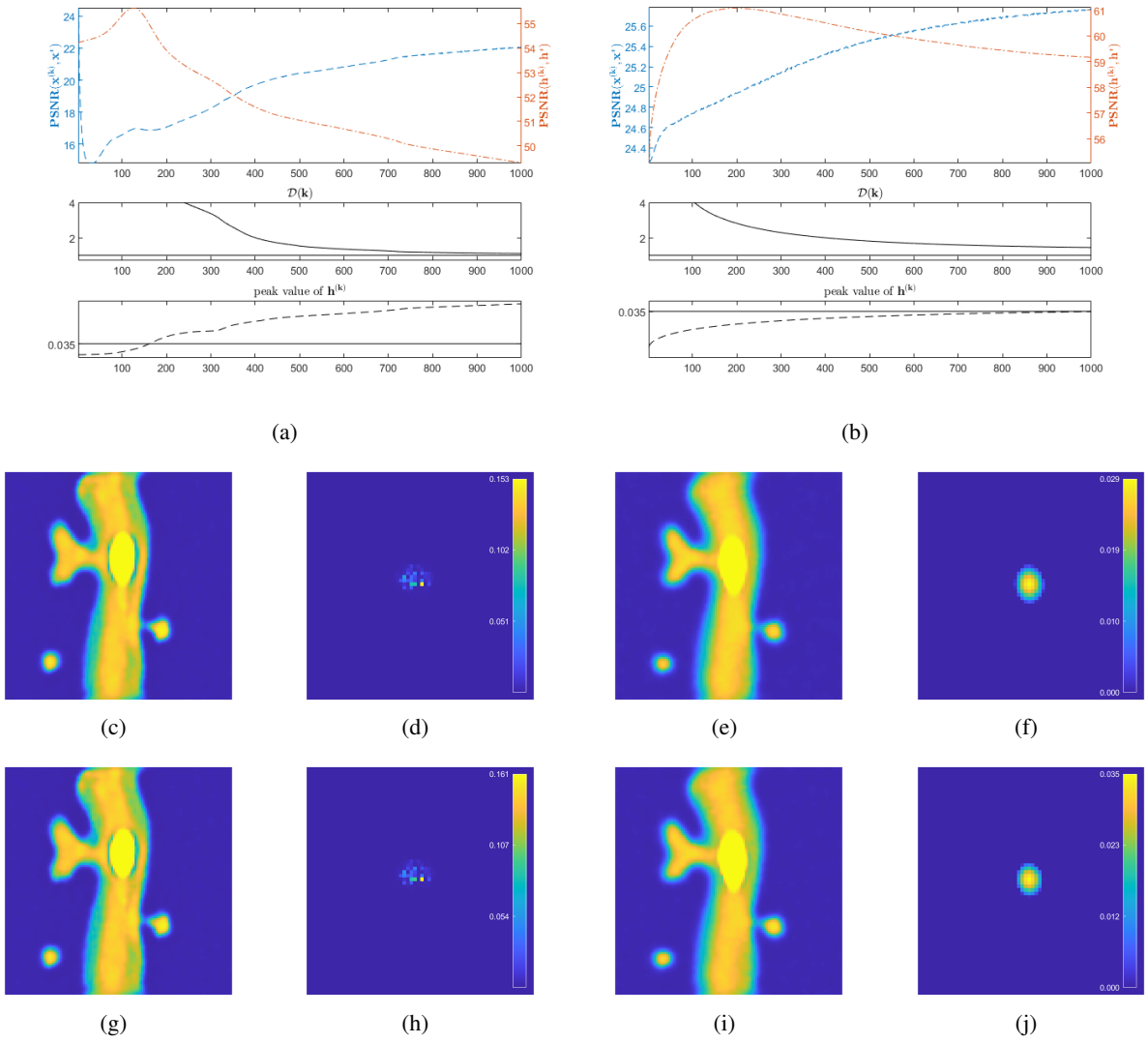


Figure 4.4: `micro` test problem - The panels on the left show the behaviour of PGDA method for the model (4.8) $\ell_2 - TV$ ($\alpha_{\mathbf{P}} = 5 \cdot 10^{-5}, \alpha_{\boldsymbol{\mu}} = 5 \cdot 10^{-4}, \gamma_{\mathbf{Y}} = 1$): (a) PSNR of $\mathbf{u}^{(k)}$ and $\mathbf{h}^{(k)}$, normalized discrepancy $\mathcal{D}(k)$ and peak value of $\mathbf{h}^{(k)}$, (c)-(d) reconstructions of the image and the PSF at the iteration $K = 860$, (g)-(h) reconstructions of the image and the PSF at the iteration $T = 1000$. The panels on the right show the behaviour of PGDA method for the model (4.10) $\ell_2 - TV$: ($\alpha_{\mathbf{P}} = 10^{-6}, \alpha_{\boldsymbol{\mu}} = 10^{-5}, \gamma_{\mathbf{Y}} = 1, \gamma_{\mathbf{V}} = 10^{-3}$): (b) PSNR of $\mathbf{u}^{(k)}$ and $\mathbf{h}^{(k)}$, normalized discrepancy $\mathcal{D}(k)$ and peak value of $\mathbf{h}^{(k)}$, (e)-(f) reconstructions of the image and the PSF at the iteration $K = 400$, (i)-(j) reconstructions of the image and the PSF at the iteration $T = 1000$.

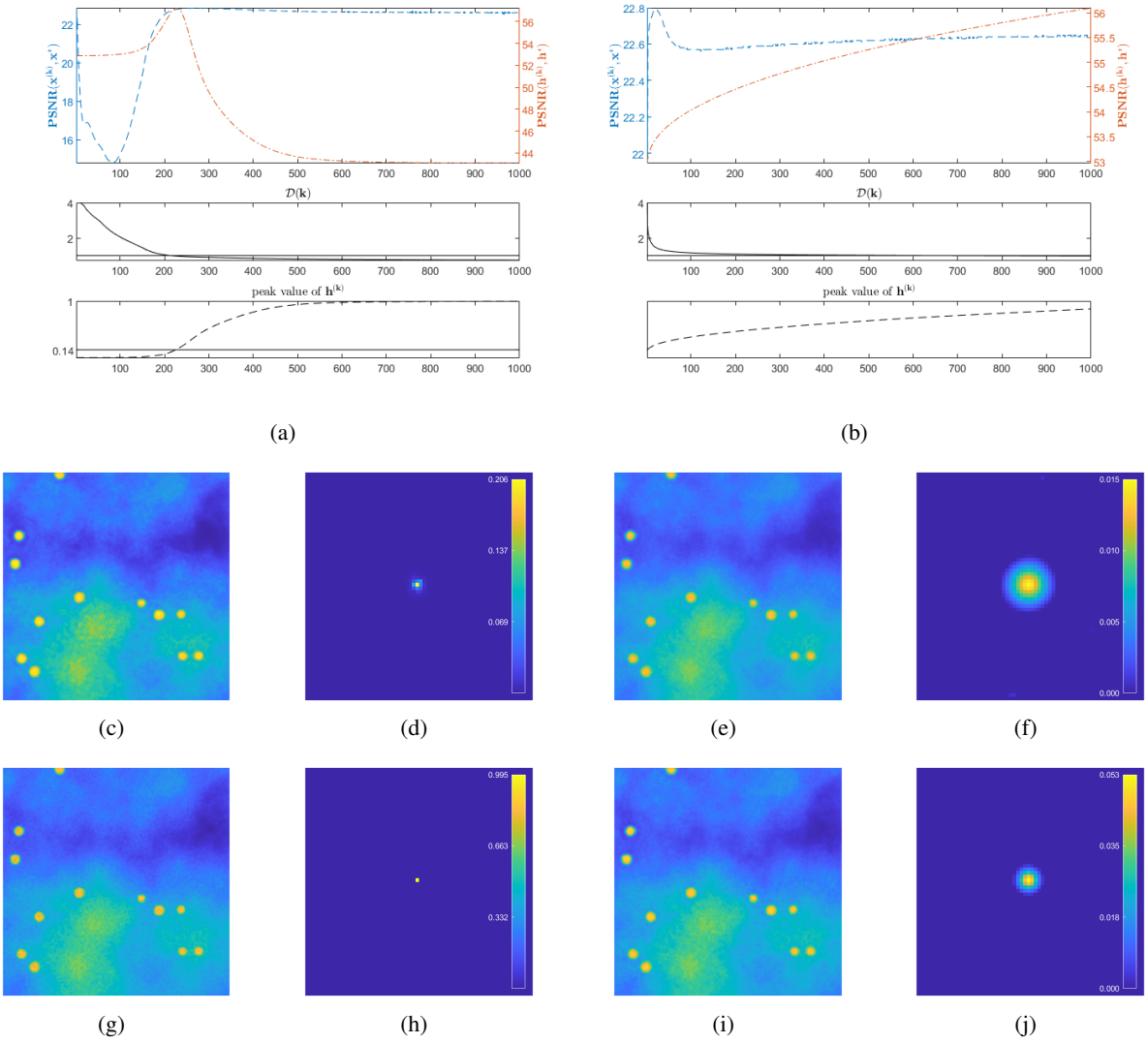


Figure 4.5: `synth001` test problem - The panels on the left show the behaviour of PGDA method for the model (4.8) $\ell_2 - TV$ ($\alpha_{\mathbf{P}} = 5 \cdot 10^{-5}, \alpha_{\boldsymbol{\mu}} = 5 \cdot 10^{-4}, \gamma_{\mathbf{Y}} = 1$): (a) PSNR of $\mathbf{u}^{(k)}$ and $\mathbf{h}^{(k)}$, normalized discrepancy $\mathcal{D}(k)$ and peak value of $\mathbf{h}^{(k)}$, (c)-(d) reconstructions of the image and the PSF at the iteration 240, (g)-(h) reconstructions of the image and the PSF at the iteration $T = 1000$. The panels on the right show the behaviour of PGDA method for the model (4.10) $\ell_2 - TV$: ($\alpha_{\mathbf{P}} = 10^{-6}, \alpha_{\boldsymbol{\mu}} = 10^{-5}, \gamma_{\mathbf{Y}} = 1, \gamma_{\mathbf{V}} = 10^{-3}$): (b) PSNR of $\mathbf{u}^{(k)}$ and $\mathbf{h}^{(k)}$, normalized discrepancy $\mathcal{D}(k)$ and peak value of $\mathbf{h}^{(k)}$, (e)-(f) reconstructions of the image and the PSF at the iteration $K = 70$, (i)-(l) reconstructions of the image and the PSF at the iteration $T = 1000$.

Model	Iters.	Time [s]	$PSNR(\mathbf{u}^{(k)})$	$PSNR(\mathbf{h}^{(k)})$	$SSIM(\mathbf{u}^{(k)})$
			`rice`		
(4.8)	$k = 200$		19.81	61.81	0.59
	$T = 1000$		24.26	42.54	0.64
(4.10)	$K = 60$	307	24.26	62.85	0.65
	$T = 1000$	5125	24.30	60.18	0.66
			`micro`		
(4.8)	$K = 860$		21.77	49.70	0.84
	$T = 1000$		22.09	49.30	0.84
(4.10)	$K = 400$	206	25.32	60.49	0.90
	$T = 1000$	516	25.76	59.16	0.91
			`synth001`		
(4.8)	$k = 240$		22.82	56.77	0.83
	$T = 1000$		22.61	43.07	0.82
(4.10)	$K = 70$	670	22.60	53.86	0.87
	$T = 1000$	9577	22.65	56.09	0.86

Table 4.1: Figures of merit for the results obtained for the models (4.8) and (4.10) $\ell_2 - TV$. K denotes the iteration for which the stopping criterion is satisfied, k is an iteration intermediate which exhibits suitable results, although the stopping criterion is not satisfied, T is the iteration corresponding to the maximum number of steps.

- regarding model (4.8), the PSNR of $\mathbf{h}^{(k)}$ reaches an high peak and then decreases in a very fast manner; in general, the peak does not correspond to a value of the recovered image satisfying the stopping criterion, although at this iteration the normalized discrepancy is close to 1 and the PSF peak value is near to the true upper bound of $\mathbf{h}^*$; the additional constraint of the PSF imposed in (4.10) seems to avoid this drawback; indeed, the stopping criterion provides more reliable results, above all on the reconstruction of the PSF. Furthermore, the reconstructions of the images obtained by solving (4.8) present several artifacts, for example in `rice` an undesired high contrast and in both `micro` and `synth001` some halos are present.

DDIPP with different regularization terms on the image. Figure 4.6 shows the results obtained by DDIPP equipped with the regularization (4.11) for the blur and the ℓ_1-like for the image (equations (4.12),(4.13)), i.e., $\ell_2 - \ell_1$. These results are obtained with the same settings used for the model (4.10) $\ell_2 - TV$. For completeness, in Table 4.2, are illustrated the merit figures obtained for the three test problems. From the comparison of Figures 4.3, 4.4 and 4.5 with Figure 4.6, it is clear that model (4.10) provides very similar results when it is combined with the TV-like or the ℓ_1-like regularization terms for the image. This is confirmed by the comparison of merit figures in Tables 4.1 and 4.2.

Starting vectors comparison Section 4.3.1 describes the necessity of the initialization of the net parameters, choosing suitable starting vectors. If one skip this phase, the weights of the networks can't address the problem, since the network can't reach the solution, *i.e.* the minimum of the model. Furthermore the stopping criteria is not satisfied, so the maximum number of iterations $T = 1000$ is always reached.

Behaviour for different noise levels. For data corrupted by Poisson noise, the noise level depends on the values of any pixel of the object [29, Section 3.4]. For example, for the `rice` object, with pixel

Iters	$PSNR(\mathbf{u}^{(k)})$	$PSNR(\mathbf{h}^{(k)})$	$SSIM(\mathbf{u}^{(k)})$
	`rice`		
$K = 120$	24.39	63.37	0.66
$T = 1000$	24.43	59.76	0.66
	`micro`		
$K = 600$	25.17	59.12	0.90
$T = 1000$	25.23	58.21	0.90
	`synth001`		
$K = 60$	22.62	53.82	0.87
$T = 1000$	22.65	56.40	0.86

Table 4.2: Figures of merit for the results obtained for the model (4.10) $\ell_2 - \ell_1$.

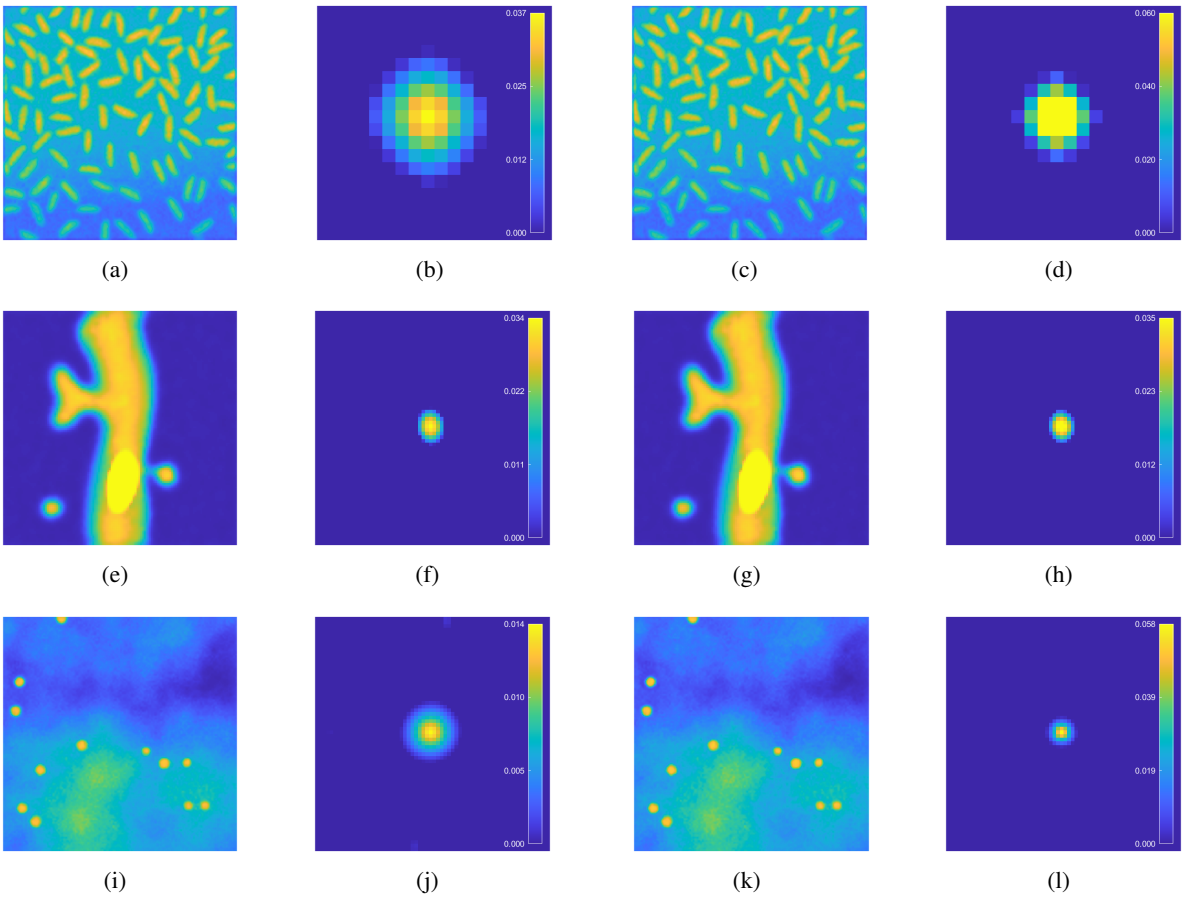


Figure 4.6: Results obtained for the model (4.10) $\ell_2 - \ell_1$. The first row shows the results for the problem `rice`: (a), (b) image and PSF at the iteration $K = 120$; (c), (d) image and PSF at the iteration $T = 1000$. The second row shows the results for the problem `micro`: (e), (f) image and PSF at the iteration $K = 600$; (g), (h) image and PSF at the iteration $T = 1000$. The third row shows the results for the problem `synth001`: (i), (l) image and PSF at the iteration $K = 60$; (m), (n) image and PSF at the iteration $T = 1000$.

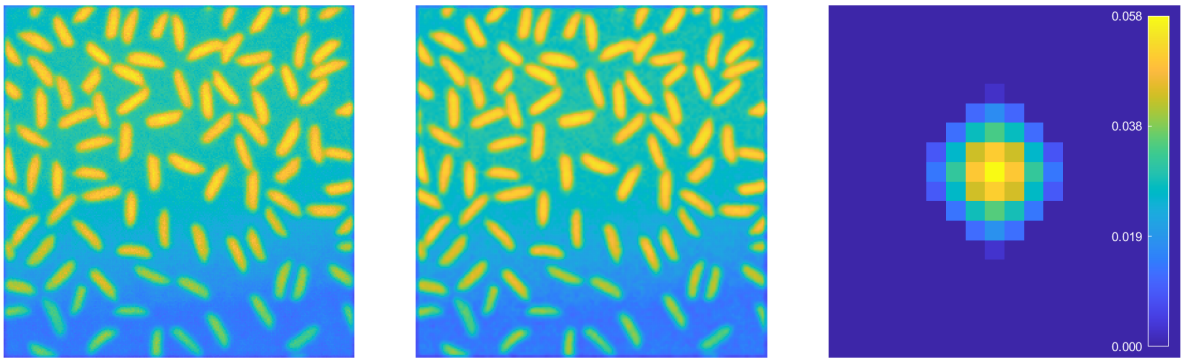
(a) (b) (c)

Figure 4.7: Results without the parameter initialization described in subsection 4.3.1 for the synthetic database using DDIPP $\ell_2 - TV$.

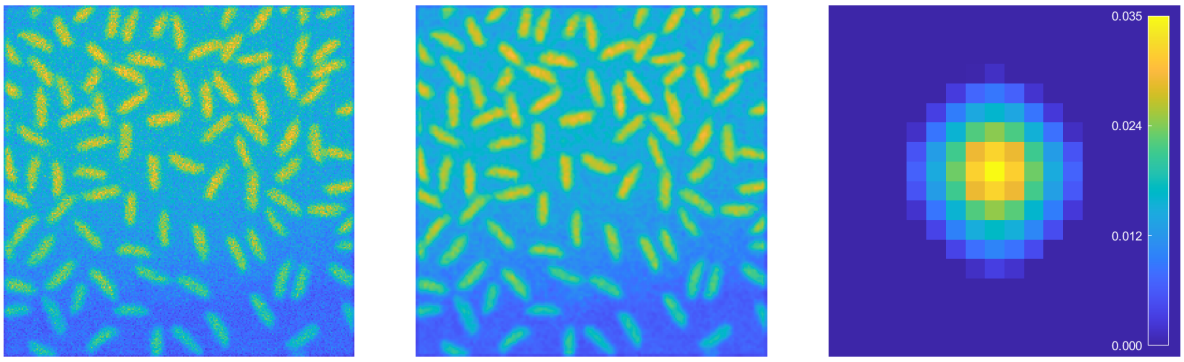
values in the interval $[0, 255]$, a synthetic detected image is the one in Figure 4.2(c); if the object is rescaled by a factor 10 or 0.5, the noise free image has the same morphology but the noisy blurred images are very different, as it can seen by comparing Figure 4.2(c) with Figures 4.8(a) and 4.8(d) corresponding to the object rescaled by 10 and 0.5 respectively. The merit figures for the test problem rescaled by 10 are $PSNR(\mathbf{g}) = 23.08$ and $SSIM(\mathbf{g}) = 0.63$, whereas they are $PSNR(\mathbf{g}) = 21.73$ and $SSIM(\mathbf{g}) = 0.44$ when the object is rescaled by 0.5. The numerical results obtained for the model (4.10) $\ell_2 - TV$ are reported in Figures 4.8(b), 4.8(c) for the object rescaled by 10 and in Figures 4.8(e) and 4.8(f) for the one rescaled by 0.5. For the former case, the stopping criterion is satisfied at the iteration $K = 400$ and the merit figures are $PSNR(\mathbf{u}^{(K)}) = 23.19$, $PSNR(\mathbf{h}^{(K)}) = 61.76$ and $SSIM(\mathbf{u}^{(k)}) = 0.65$; for the latter case, the stopping criterion is satisfied at the iteration $K = 90$ and the merit figures at this iteration are $PSNR(\mathbf{u}^{(K)}) = 24.16$, $PSNR(\mathbf{h}^{(K)}) = 63.39$ and $SSIM(\mathbf{u}^{(K)}) = 0.64$. The numerical results allow to observe that the model (4.10) appears robust with respect to the different noise levels. In particular, the reconstruction of the PSF appears to be sufficiently accurate in all the numerical tests. A similar behaviour has been observed in the other test problems.

Comparison between DDIPP, SelfDeblur, improved SelfDeblur [57] and Algorithm 3 in [53]. A further experiment is aimed to compare the results obtained by DDIPP with the ones achieved via other approaches. From what is known in literature, no method for NBD bases its model on the Kullback-Leibler generalized divergence. Furthermore, most of the approaches use pre-trained networks. Therefore it seems difficult to identify methods with which to carry out a comparison. Based on these reasons, two methods are considered which inspired DDIPP, namely SelfDeblur [54] and its improved version proposed in [57], referred to in this work as ImpSD. As regards SelfDeblur, the code for the experiments is available at `https://github.com/csdwren/SelfDeblur` and it can be run on Google Colab. On the other hand, since the code for ImpSD is not available online, the method has been implemented in MATLAB®, by replicating the networks of SelfDeblur. The procedure in [57], while preserving the basic approach of SelfDeblur, adds regularization terms and suggests the initialization of the nets, as depicted in 4.3.1. In the code, the MATLAB® implementation of the functionals employed in the formulation (Least square, SSIM, TV) are used. In particular, the objective functional is modified among the iterations: following [57], the first 2000 iterations employ the Least Square loss together with the ℓ_2 regularization on the PSF, then it switches to the SSIM functional coupled with the TV on the image. The regularization parameters are chosen as in [57]: 0.01 and 0.1 for TV and ℓ_2, respectively.

Another recent proposal is the third algorithm in [53], denoted in the following as ASA3. Indeed, the authors in [53] primarily use a different approach: the two networks $\mathcal{N}_h$ and $\mathcal{N}_u$ are *pre-trained* generative networks, and the procedure updates the *inputs* $\mathbf{z}_h$ and $\mathbf{z}_u$ of the networks and not their



(a) (b) (c)



(d) (e) (f)

Figure 4.8: Results for different noise levels - In the upper panel: (a) $\mathbf{g}$ for the test problem `rice` when the object is rescaled by a factor 10; (b), (c) recovered image and PSF for the model (4.10) $\ell_2 - TV$ at the iteration $K = 400$. In the lower panel: (d) $\mathbf{g}$ for the test problem `rice` when the object is rescaled by a factor 0.5; (e), (f) recovered image and PSF for the model (4.10) $\ell_2 - TV$ at the iteration $K = 90$.

weights. One must have *a-priori* information on the type of images and on the blur type, together with large relative datasets, in order to have properly trained networks. Then, to overcome the issue of having a large image dataset for the training of $\mathcal{N}_u$, the authors propose to update also the weights of this network, together with its input and with $\mathbf{z}_h$. The functional employed in this setting is

$$\mathcal{F}(\mathbf{z}_h, \mathbf{z}_u, \boldsymbol{\theta}_u) = \|\mathbf{g} - \mathcal{N}_h(\mathbf{z}_h; \boldsymbol{\theta}_h) * \mathcal{N}_u(\mathbf{z}_u; \boldsymbol{\theta}_u)\|^2 + \beta_h \|\mathbf{z}_h\|^2 + \beta_u \, TV(\mathcal{N}_u(\mathbf{z}_u; \boldsymbol{\theta}_u)). \tag{4.42}$$

With an abuse of notation, in (4.42) the dependence of the nets on the random inputs $\mathbf{z}_h$ and $\mathbf{z}_u$ is explicit. In the following experimentation, in order to implement the ASA3 method, the network $\mathcal{N}_h$ has been trained for generate Gaussian PSFs; this network presents a different architecture with respect to the one used in [53] since the structure of the PSF is less articulate than the motion blurs used in [53]. Since the code for ASA3 in [53] is not available online, it has been re-coded entirely in MATLAB®. For clearness, the details of ASA3 are reported in Algorithm 3. In the numerical experiment, as in [53], different steplengths are set for the descent step: namely $\eta = 10^{-3}$ for updating the nets' inputs and $\eta = 10^{-4}$ for updating the weights. Moreover, $\beta_h = 10^{-2}$ and $\beta_u = 10^{-3}$. The number of iterations is set to $T = 5000$ and $T_u = 500$.

Algorithm 3: ASA3 [53]

Set η, draw $\mathbf{z}_h \sim \mathcal{N}(0,1), \mathbf{z}_u \sim \mathcal{N}(0,1)$;
Choose T_u, T;
Initialise $\mathcal{N}_u$ by solving (4.39) with a max number of iterations equal to T_u;
for $t = 1, 2, \ldots, T$ **do**
  $\mathbf{z}_h^{(k+1)} \leftarrow \mathbf{z}_h^{(k)} - \eta \nabla_{\mathbf{z}_h} \mathcal{F}(\mathbf{z}_h^{(k)}, \mathbf{z}_u^{(k)}, \boldsymbol{\theta}_u^{(k)})$;
  $\mathbf{z}_u^{(k+1)} \leftarrow \mathbf{z}_u^{(k)} - \eta \nabla_{\mathbf{z}_u} \mathcal{F}(\mathbf{z}_h^{(k)}, \mathbf{z}_u^{(k)}, \boldsymbol{\theta}_u^{(k)})$;
  $\boldsymbol{\theta}_u^{(k+1)} \leftarrow \boldsymbol{\theta}_u^{(k)} - \eta \nabla_{\boldsymbol{\theta}_u} \mathcal{F}(\mathbf{z}_h^{(k)}, \mathbf{z}_u^{(k)}, \boldsymbol{\theta}_u^{(k)})$;
end

Figure 4.9 and Table 4.3 present the results of SelfDeblur, ImpSD and ASA3 carried on the syntetic database; bearing in mind Figures 4.3, 4.4, 4.5 and Table 4.1 (see also Figure 4.6 and Table 4.2), it is possible to make a comparison between the results of DDIPP and the ones obtained by the aforementioned methods. As noted in [58, Section 3.1.2], the cropping procedure in SelfDeblur produces artifacts and misplacements in the final reconstruction. Indeed, unlike SelfDeblur, the proposed DDIPP strategy considers a larger size of just the PSF and not of the image, since the boundary conditions are encompassed in the convolutional filters. ImpSD instead provides slightly better results, in terms of placement, but several details are missed; in particular, the image `synth001` suffers from several artifacts. Finally, the algorithm ASA3 seems to recover reliable image and PSF at an intermediate iteration. Indeed, Figure 4.9 shows the results of ASA3 at the iteration K where the highest PSNR value of the image $\mathbf{u}^{(K)}$ over 5000 iterations is observed, but for real applications, where the ground truth is not known, this is not generally possible. In some cases, the recovered image has a high PSNR, despite having artifacts (see `micro`), in other cases it remains noisy (see `rice`, `synth001`). At the iteration K, the recovered blur kernels also appear reliable, although slightly asymmetrical. However, after the iteration K, a subsequent degradation to the point form is observed. Finally, DDIPP in general provides a better PSF reconstruction.

4.4.2 Real dataset

This section is devoted to apply the proposed DDIPP to real-world Microscopy images. Such images are the scanning of a 3D volume, with dimension $64 \times 64 \times 4.1$ μm: this volume is then recorded in an array with dimension $512 \times 512 \times 10$ voxels. The radius of the spherical particles is 1.5 μm and they are suspended in a $\sim 70\% - 30\%$ glycerol/water mixture (viscosity of approximately 0.017 Pa s). The microscope employed for acquiring this dataset is a Zeiss LSM 700 with a 100×NA 1.4 oil

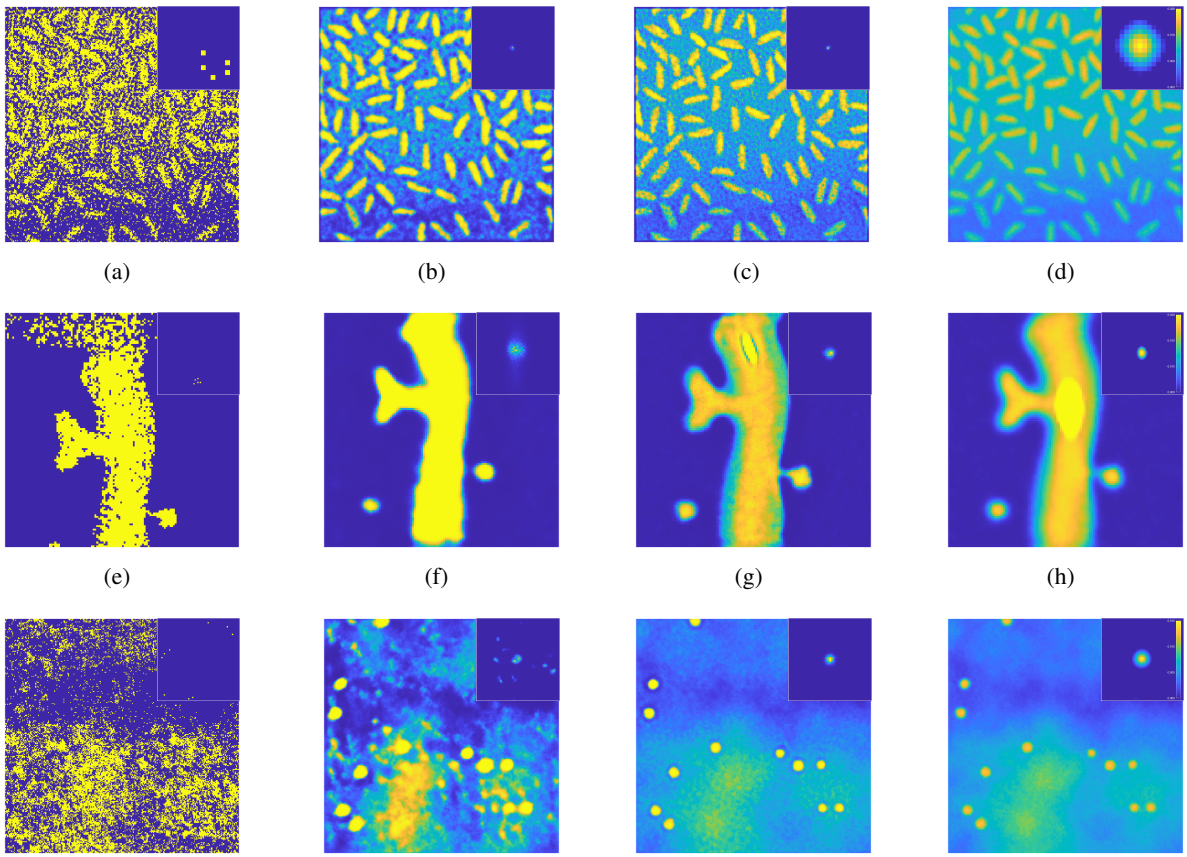
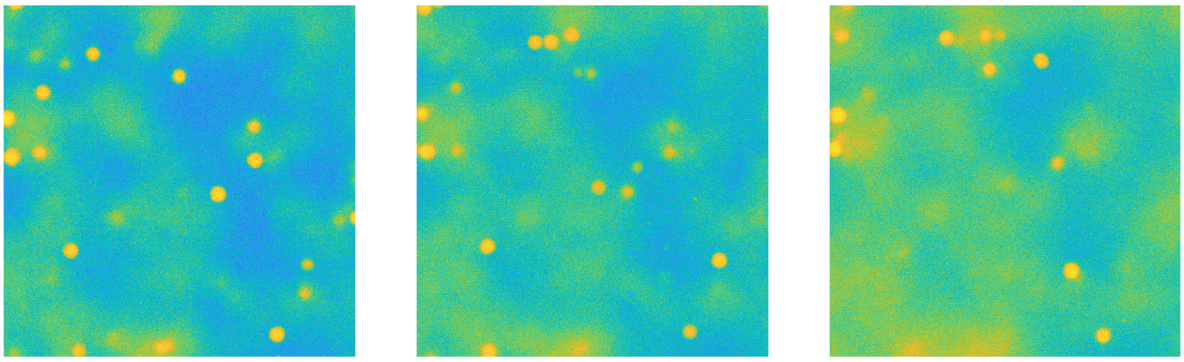


Figure 4.9: Results of the comparison for the synthetic database - In the first row `rice`: (a) recovered image via SelfDeblur, (b) recovered image via ImpSD; (c) recovered image via ASA3; (d) recovered image via DDIPP $\ell_2 - TV$. In the second row `micro`: (e) recovered image via SelfDeblur, (f) recovered image via ImpSD; (g) recovered image via ASA3; (h) recovered image via DDIPP $\ell_2 - TV$. In the third row `synth001`: (i) recovered image via SelfDeblur, (j) recovered images via ImpSD; (k) recovered image via ASA3; (l) recovered image via DDIPP $\ell_2 - TV$.

Algorithm	Iters	$PSNR(\mathbf{u}^{(k)})$	$PSNR(\mathbf{h}^{(k)})$	$SSIM(\mathbf{u}^{(k)})$
		`rice`		
SelfDeblur	$T = 5000$	6.45	17.67	0.01
ImpSD	$T = 5000$	14.54	60.70	0.41
ASA3	$K = 400$	17.30	64.98	0.42
		`micro`		
SelfDeblur	$T = 5000$	9.34	29.12	0.20
ImpSD	$T = 5000$	16.24	55.53	.71
ASA3	$K = 300$	26.48	65.03	.88
		`synth001`		
SelfDeblur	$T = 5000$	5.41	26.54	0.08
ImpSD	$T = 5000$	15.15	53.12	0.30
ASA3	$K = 150$	22.74	55.45	0.66

Table 4.3: Figures of merit for the results obtained with SelfDeblur (5000 iterations), ImpSD (5000 iterations) and ASA3. For the ASA3 method the results obtained are reported at the iteration K which provides the maximum PSNR of the image within 5000 iterations.

immersion objective (Zeiss Plan–APOCHROMAT). The architectures for $\mathcal{N}_u$ and $\mathcal{N}_h$ take as input just 2D images, not 3D volumes: hence one frame of the 3D volume is considered at time. This dataset contains several hundreds of images, but due to hardware limitation only 3 of them are considered and shown in Figure 4.10. On the base of the technical documentation of the instrument, the parameter H that imposes the upper bound on the peak of the PSF is approximately estimated as 0.2. The three



(a) (b) (c)

Figure 4.10: Detected Microscopy images; their size is 512×512.

problems employed DDIPP for the model (4.10) $\ell_2 - TV$ and $\ell_2 - \ell_1$. The results are carried out with $\alpha_{\mathbf{P}} = 10^{-6}, \alpha_{\boldsymbol{\mu}} = 10^{-5}, \gamma_{\mathbf{Y}} = 1, \gamma_{\mathbf{V}} = 10^{-3}$; the other parameters are as specified at the beginning of Section 4.4. The results are depicted in Figure 4.11 and Figure 4.12 in the case of $\ell_2 - TV$ and $\ell_2 - \ell_1$, respectively. In both the figures, the recovered images and related PSFs are reported. The image and PSF referring to the first test problem (first column) are obtained at the maximum number $T = 1000$ of iterations for both cases; the results related to the second and third test problems (second and third columns respectively) are obtained at the iterations satisfying the stopping criterion, i.e., $K = 570$ and $K = 680$, respectively, for the model (4.10) $\ell_2 - TV$ and at the maximum number $T = 1000$ of iterations for the model (4.10) $\ell_2 - \ell_1$. Although the value of the maximum number of iterations has been reached, the stopping criterion based on the condition (4.40) has been satisfied in all cases but the

value of the normalized discrepancy (4.41) remained greater than 2 (but less than 3).

It is clear that, as anticipated from the experiments on synthetic data, the contrast is improved, the particles are more enhanced with respect to the background, both in the more diffuse regions and in the darker ones. Furthermore, as expected, the recovered PSF is very similar in all cases, with a peak value between 0.12 e 0.15. No substantial differences are noticed between the use of $\ell_2 - TV$ and $\ell_2 - \ell_1$; in the case of some balls not very distinct from the background in the detected image, the reconstruction with ℓ_1 seems to recover slightly better their position (see Figures 4.11(c) and 4.12(c)).

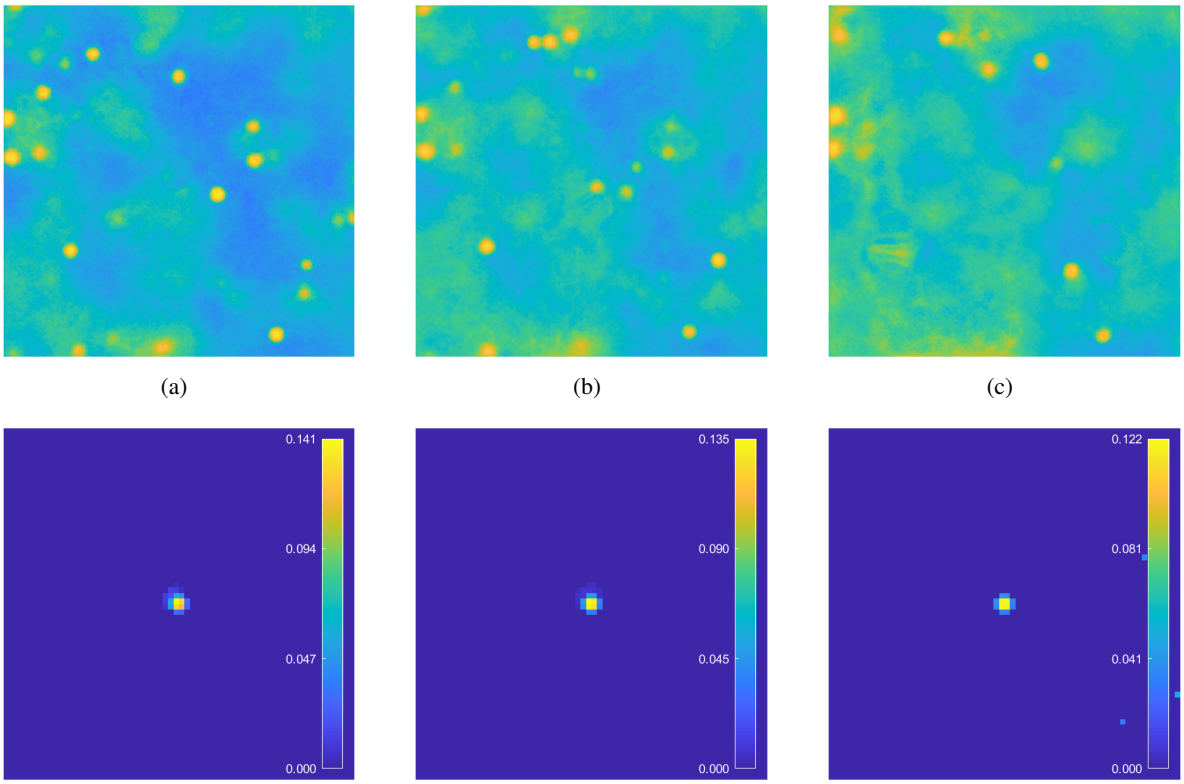
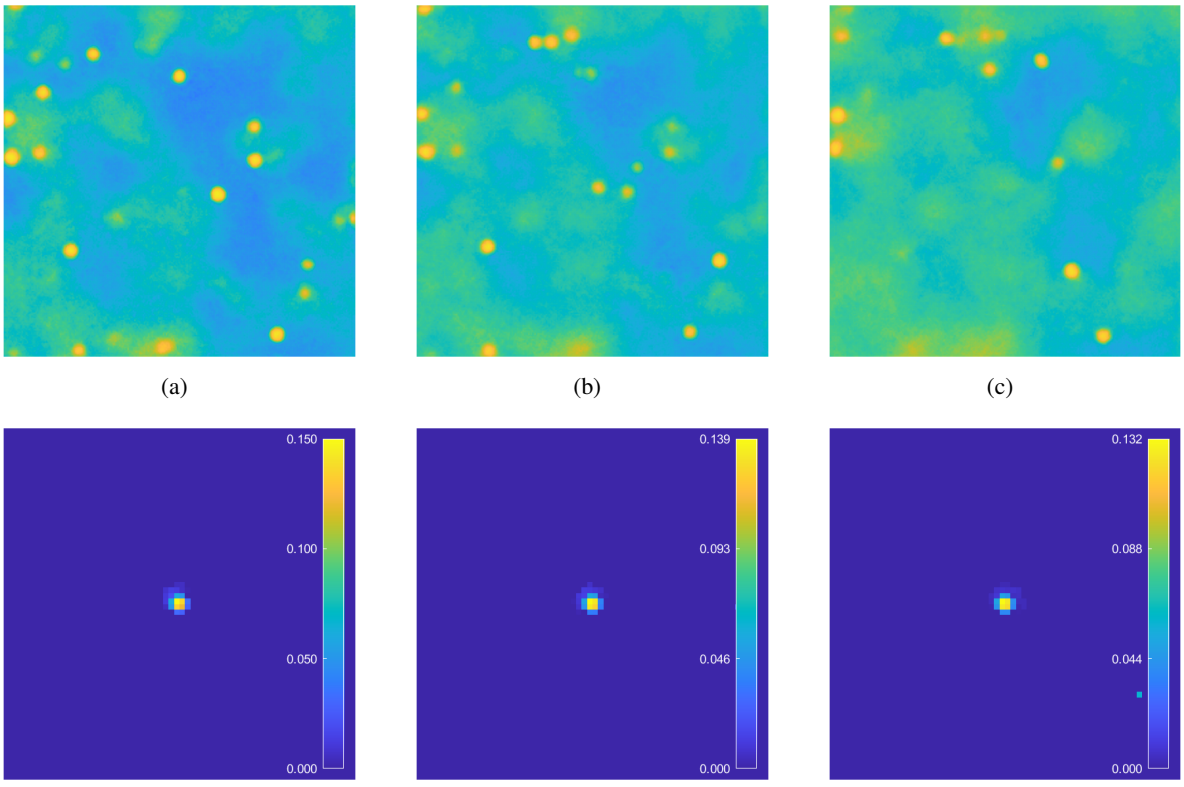


Figure 4.11: Real data: results for the model (4.10) $\ell_2 - TV$; in particular (a),(d): recovered image and PSF for the detected image in Figure 4.10(a) at the iteration 1000; (b),(e): recovered image and PSF for the detected image in Figure 4.10(b) at the iteration $K = 570$; (c), (f): and PSF for the detected image in Figure 4.10(c) at the iteration $K = 680$.

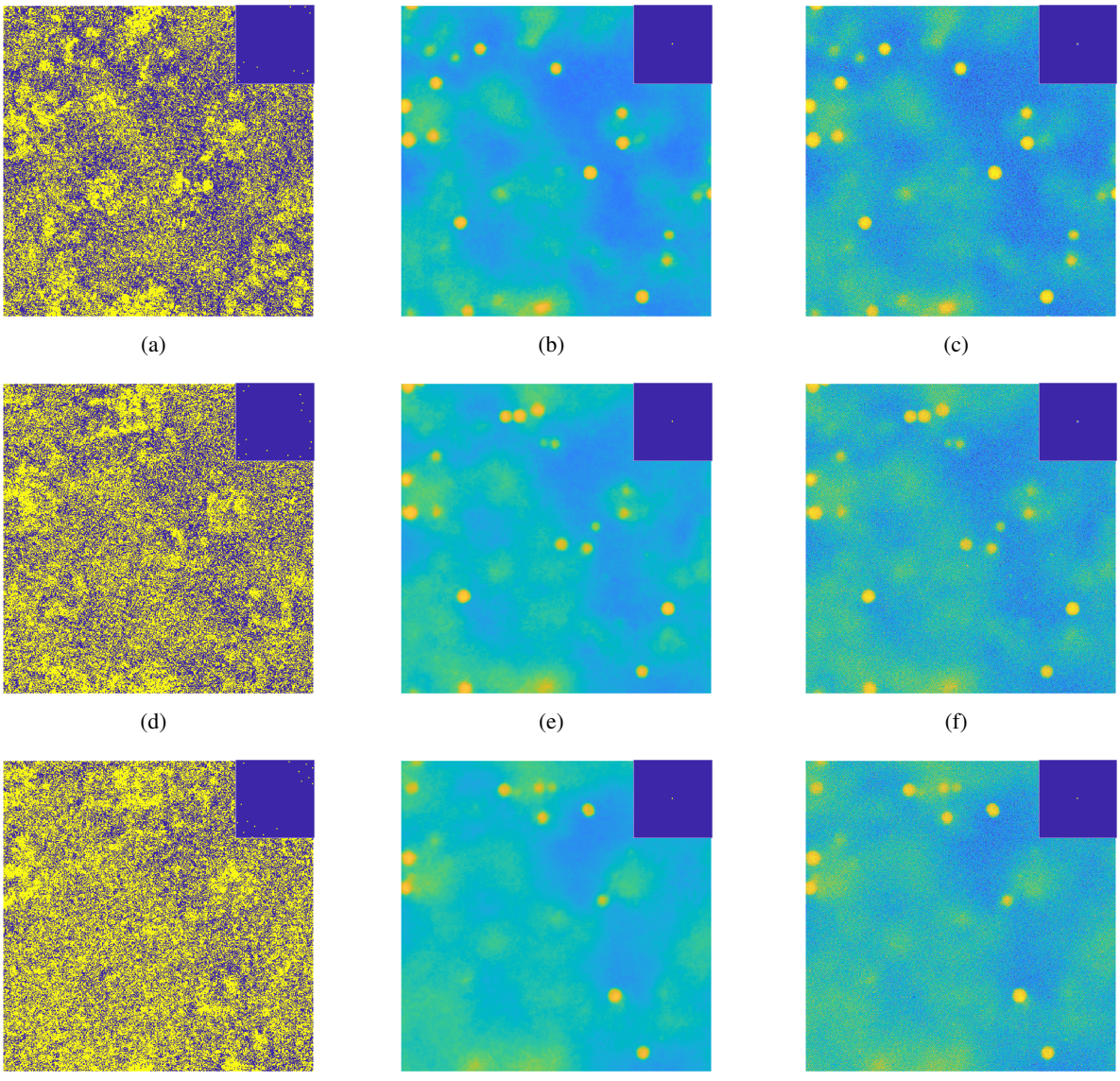
The importance of the right choice for the functional to minimize and the role of the regularizers are again evident from the results achieved by SelfDeblur, ImpSD and ASA3 after 5000 iterations, depicted in Figure 4.13. In general, the behaviour of ImpSD seems to be just denoising and not deblurring, since the recover of the PSF is the Dirac's delta and the only perceivable effect is the one by the TV.



(a) (b) (c)

(d) (e) (f)

Figure 4.12: Real data: results for the model (4.10) $\ell_2 - \ell_1$; in particular (a),(d): recovered image and PSF for the detected image in Figure 4.10(a) at the iteration 1000; (b),(e): recovered image and PSF for the detected image in Figure 4.10(b) at the iteration 1000; (c), (f): recovered image and PSF for the detected image in Figure 4.10(c) at the iteration 1000.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 4.13: Recovered images for the detected data in Figures 4.10(a), 4.10(b) and 4.10(c) achieved after 5000 iterations via SelfDeblur (first column), via ImpSD (second column) and via ASA3 (third column).

CHAPTER 5

Unsupervised noisy image segmentation via Deep Image Prior

Image segmentation is one of the most important and ubiquitous tasks in image analysis and its applications span across various domains. In medical imaging it is extensively used to isolate anatomical structures and regions of interest [82]. For autonomous driving, it plays a crucial role in assisting vehicles to navigate their surroundings effectively [83]. Additionally, segmentation also enables face recognition [84], satellite analysis and video surveillance [85] to count a few.

Deep learning methods for image segmentation (see for example [86, 87, 88, 89, 90]) have become very popular over the last few years mainly due to rapid technology improvements, (faster GPUs) and to the availability of larger and larger datasets. The majority of the segmentation approaches are based on Convolutional Neural Networks, which are usually trained in a supervised manner. This usually requires a complete dataset of appropriate dimension with the related ground truth labels, which typically can be rather expensive and/or time consuming to acquire. An alternative approach consists in considering unsupervised deep learning, where the networks are not trained on pre-labelled datasets. In this framework, several techniques have been proposed in recent years. A popular approach is to use generative models, such as Variational Autoencoders (VAEs) [91] or Generative Adversarial Networks (GANs) [92]. A further recent approach has been presented in [93], where the authors introduced a Mumford-Shah functional tailored for unsupervised image segmentation using neural networks. It is worth to remark that the approach in [93] still requires a training phase and a large dataset: indeed the training is pursued by minimizing the Mumford-Shah function (tailored for the neural networks) with respect to the outputs of the network and over mini-batches of the input images, however without using any ground truth label. Therefore, the technique suggested in [93] needs a training which is carried out in an unsupervised fashion.

Applying Deep Image Prior to image segmentation implies to combine the regularizing effects of neural networks with traditional variational models. The variational models classically employed to address image segmentation imply the minimization of tailored energy functionals. One of the most well-known energy function is the so-called Mumford-Shah (MS) functional [94], whose minimization determines an approximation of the image by means of a piecewise smooth function. The MS model represents the most general way to formulate a segmentation problem and the theory developed over the past two decades on this model encourages its application to provide robust and effective software for large-scale analysis. However, the original MS formulation of the segmentation problem is difficult to be treated from the numerical point of view due to the non-convexity and the non-smoothness nature of the model. For this reason, large efforts have been devoted to propose MS approximations prone to be numerically implemented. Ambrosio and Tortorelli (AT) proposed one of the most general approximations of the MS model [95, 96]. The AT strategy is based on a sequence of simpler elliptic functionals which Γ-converges (in the continuous setting) to the MS functional. Since in their model Ambrosio and Tortorelli replaced the unknown discontinuity set by an auxiliary function which smoothly approximates its indicator function, gradient-based methods have in general satisfying performance for its minimization. Nevertheless, it is well known that good results are strongly dependent on suitable choices for the values of the regularization parameters involved in the definition of the AT functional.

Several numerical experiments on image segmentation problems have been performed for both biomedical and natural images datasets. Furthermore the Ambrosio-Tortorelli formulation for the Mumford-Shah functional was considered, embedding it in the Deep Image Prior framework, reparametriz-

ing its two variables, namely the restored image and the auxiliary function, with two different neural networks. Despite the computational burden, *i.e.* optimizing the weights of two different nets, thanks to the inherent regularization behaviour of the DIP framework this approach is robust with respect to the regularization parameter: indeed, the same setting provides reliable result on different noise level and type, and moreover on images with different characteristics.

5.1 Classical variational models for image segmentation

Before detailing the unsupervised method for image segmentation, it is necessary to recall the main features of the variational models. Let Ω be an open and bounded domain in $\mathbb{R}^2$. Let $u_0 : \Omega \to \mathbb{R}^m$ with $m \geq 1$, $u_0 \in L^\infty(\Omega)$, be a given bounded input image, possibly noisy. In the variational approach to the segmentation problem proposed by Mumford and Shah [94], the aim is to find a pair (u, D) where $D \subset \bar{\Omega}$ is a closed set representing the contours reconstructed from the discontinuities of u_0 and u is a smooth representation of u_0 outside D. Such a pair can be found, among all possible pairs (u, D) with $D \subset \bar{\Omega}$ closed and $u \in C^1(\Omega \setminus D)$, as an optimal solution to the following minimization problem

$$\min_{u,D} \mathcal{E}(u, D) \equiv \int_\Omega (u(x) - u_0(x))^2 \, dx + \lambda \int_{\Omega \setminus D} |\nabla u(x)|^2 \, dx + \mu \mathcal{H}^1(D) \tag{5.1}$$

where $\lambda, \mu > 0$ are fixed parameters, weighting the different terms in the energy functional, and $\mathcal{H}^1(D)$ denotes the 1-dimensional Hausdorff measure of the set D. In (5.1) the first term acts as a data fidelity term and forces the approximation u to be close to the input image u_0, the second term forces u to be smooth except at the locations K of the edges and the last term penalizes the length of the set of discontinuities K. In order to show that the MS functional admits minimizers, De Giorgi et al. considered a weak formulation depending only on one variable [97]. Indeed, they worked with a relaxed version of the MS functional defined for a function $u \in SBV(\Omega)$, the space of special functions of bounded variation on Ω, by

$$\mathcal{E}_w(u) = \int_\Omega (u(x) - u_0(x))^2 \, dx + \lambda \int_\Omega |\nabla u(x)|^2 \, dx + \mu \mathcal{H}^1(J_u), \tag{5.2}$$

where J_u denotes the set of jumps of u, i.e. the set of discontinuities of u. The minimization of the functional in (5.2) is still delicate due to the computation of the 1-dimensional Hausdorff measure of J_u. By exploiting the notion of Γ-convergence, Ambrosio and Tortorelli [95] proposed a variational model in which they approximate the Hausdorff measure of the set J_u by a continuous function v with values close to 0 near J_u, and values close to 1 away from J_u. In more detail, they introduced the following sequence of approximating energies depending on the parameter $\varepsilon > 0$

$$\begin{aligned} \mathcal{E}_\varepsilon(u, v) = &\int_\Omega (u(x) - u_0(x))^2 \, dx + \lambda \int_\Omega v(x)^2 |\nabla u(x)|^2 dx + \\ &+ \mu \int_\Omega \Big(\varepsilon |\nabla v(x)|^2 + \frac{(v(x) - 1)^2}{4\varepsilon} \Big) \, dx, \end{aligned} \tag{5.3}$$

defined on functions $u, v \in H^1(\Omega)$ such that $0 \leq v \leq 1$. The functionals $\mathcal{E}_\varepsilon(u, v)$ in (5.3) Γ-converge to the MS functional (5.1) as ε tends to 0.

5.2 Unsupervised segmentation via Deep Image Prior

The DIP paradigm in the image segmentation framework can be exploited, like in the previous scenarios, for both the simpler case of piecewise constant image segmentation and the more general case of piecewise smooth segmentation.

The network architecture used in numerical experiments is described in Chapter 3 and it is depicted in Figure 3.1.

5.2.1 Piecewise constant segmentation

This section is devoted to presenting a notable instance of the model (5.2), when the segmented image u is piecewise constant.

Denote by Ω_i, $i = 1, \dots, N$, the connected components of $\Omega \setminus D$, u takes the values c_i within Ω_i, leading to $\nabla u(x) = 0$, $\forall x \in \Omega \setminus D$. Hereafter c represents the vector $c = (c_1, ..., c_N)^T$. In this case, by assuming N piecewise constant regions in the input image, the objective functional in (5.1) reduces to

$$\mathcal{E}(c, D) = \sum_{i=1}^{N} \int_{\Omega_i} (c_i - u_0(x))^2 \, dx + \mu \mathcal{H}^1(D) \tag{5.4}$$

with $D = \cup_{i=1}^{N} \partial\Omega_i$, where $\partial\Omega_i$ is the boundary of the i-th region. For a given D, it is straightforward to show that the functional in (5.4) is minimized with respect to the variables c_i by setting each c_i to the mean intensity of the input image u_0 within the region Ω_i. Introducing the characteristic function of the i-th region, denoted by $\chi_i(x)$, this allows to rewrite (5.4) as follows:

$$\mathcal{E}_{LS}(\chi) = \sum_{i=1}^{N} \int_{\Omega} (c_i - u_0(x))^2 \chi_i(x) \, dx + \mu \sum_{i=1}^{N} \|\chi_i(x)\|_{\mathrm{TV}} \, dx, \tag{5.5}$$

where $\| \cdot \|_{\mathrm{TV}}$ denotes the Total Variation functional [98]. The average pixel value c_i is given by

$$c_i(\chi) = \frac{\int_{\Omega} u_0(x) \chi_i(x) \, dx}{\int_{\Omega} \chi_i(x) \, dx}. \tag{5.6}$$

In particular, it is possible to reformulate (5.5) to the presence of Poisson Noise, which typically arises in low-light conditions and during image capture by digital devices. This noise originates from a Poisson process, where the variation is due to the random arrival of photons on the sensor of the detection device (photon counting). The noise level is inversely related to the number of photons: as the photon count increases, the noise affecting the image diminishes ([35, 36]). For image restoration in the presence of this noise, the generalized Kullback-Leibler function is used as the discrepancy measure, which for denoising problems is formulated as

$$KL(u, u_0) = \int_{\Omega} u_0(x) \log\left(\frac{u_0(x)}{u(x)}\right) + u(x) - u_0(x) \, dx. \tag{5.7}$$

As a consequence, supposing that the corrupted image is affected by Poisson noise and u is piecewise constant, the MS functional can be rewritten as

$$\begin{aligned} \mathcal{E}_{KL}(\chi) = \sum_{i=1}^{N} \int_{\Omega} \left[u_0(x) \left(\log(u_0(x)) - \log(c_i)\right) + c_i - u_0(x)\right] \chi_i(x) \, dx + \\ + \mu \sum_{i=1}^{N} \|\chi_i(x)\|_{\mathrm{TV}}. \end{aligned} \tag{5.8}$$

As for (5.5), once the functions χ_i, $i = 1, \dots, N$ have been found by minimizing (5.8), then it is simple to prove that the values c_i, $i = 1, \dots, N$ can be found as in (5.6). Indeed, once χ_i, $i = 1, \dots, N$ are fixed, c_i, $i = 1, \dots, N$ are found by simply solving

$$-\frac{1}{c_i} \int_{\Omega} u_0(x) \chi_i(x) dx + \int_{\Omega} \chi_i(x) dx = 0$$

which leads to the same result as in (5.6).

To apply the proposed method based on the minimization of either (5.5) or (5.8), the discrete framework must be introduced. Given $O \subset \mathbb{R}^d$ with $d = d_1 \cdot d_2$ a discretized version of the continuous domain Ω, let $u_0 : O \to \mathbb{R}$ represents the vectorized input image. Let be O partitioned into the

components O_i, such that $O = \cup_{i=1}^{N} O_i$. Here, O_i represents both Ω_i and its boundary $\partial\Omega_i$. Denoting by $X_i : O \to \{0, 1\}$ the discrete counterpart of the characteristic function of the i-th component O_i, i.e.:

$$X_i(k) = \begin{cases} 1 & \text{if } k \in O_i \\ 0 & \text{otherwise} \end{cases}, \tag{5.9}$$

the discrete version of (5.5) and (5.8) reads respectively as

$$E_{LS}(X) = \sum_{i=1}^{N} \left(\| (u_0 - c_i) \odot X_i \|_2^2 + \mu \sum_{k=1}^{d} \|\nabla_k X_i\|_2 \right) \tag{5.10}$$

and

$$E_{KL}(X) = \sum_{i=1}^{N} \left(\sum_{k=1}^{d} \left(\left[u_0 \log\left(\frac{u_0}{c_i}\right) + c_i - u_0 \right] \odot X_i \right)_k + \mu \sum_{k=1}^{d} \|\nabla_k X_i\|_2 \right) \tag{5.11}$$

where

$X \in \mathbb{R}^{d \times N}$ is the matrix whose i-th column is the vector-reshaped characteristic function X_i, here X_i is a vector containing binary values corresponding to O;

$\odot$ denotes the component-wise product of two vectors;

$\nabla_k \in \mathbb{R}^{2 \times d}$ represents the discrete-gradient operator at the element k defined through the standard finite difference with periodic boundary conditions.

The values c_i, $i = 1, \ldots, N$ can be retrieved using the discrete counterpart of (5.6):

$$c_i = \frac{\langle u_0, X_i \rangle}{\|X_i\|_0}. \tag{5.12}$$

Note that the functionals to be minimize in both cases of Gaussian and Poisson noise reads in a similar manner. Indeed, with some abuse of notation, both (5.10) and (5.11) can be rewritten as

$$\tilde{\mathcal{E}}(X) + \mu \mathcal{R}(X) \tag{5.13}$$

where $\tilde{\mathcal{E}}$ is the data fidelity depending on the noise: for Poisson noise one has

$$\tilde{\mathcal{E}}(X) = \sum_{i=1}^{N} \left(\sum_{k=1}^{d} \left(\left[u_0 \log\left(\frac{u_0}{c_i}\right) + c_i - u_0 \right] \odot X_i \right)_k \right) \tag{5.14}$$

while for Gaussian noise

$$\tilde{\mathcal{E}}(X) = \sum_{i=1}^{N} \| (u_0 - c_i) \odot X_i \|_2^2. \tag{5.15}$$

The functional $\mathcal{R}$ acts as the regularisation functional, and in this particular case

$$\mathcal{R}(X) = \sum_{i=1}^{N} \sum_{k=1}^{d} \|\nabla_k X_i\|_2.$$

The next step consists in moving all the above formulation under the DIP framework, considering a CNN generator:

$$\begin{aligned} f_X \quad : \quad & \mathbb{R}^s \times \mathbb{R}^t \to \mathbb{R}^{dN} \\ & (\theta_X, z_X) \mapsto f_X(\theta_X; z_X) \end{aligned}$$

which depends on the weights θ_X and it is fed by the random input z_X: its output must represent a labeling procedure based on the number N of classes in the image. Therefore, the activation function of the output layer of the vanilla DIP was replaced with a softmax, because it mimics the role of the characteristic function, allowing the network to provide the probability of each pixel to belong to one of the N classes. The problem to be solved becomes hence

$$\underset{\theta_X}{\operatorname{argmin}}\ \tilde{\mathcal{E}}(f_X(\theta_X; z_X)) + \mu\mathcal{R}(f_X(\theta_X; z_X)) \equiv \Phi(\theta_X) \tag{5.16}$$

and the values $\{c_i\}_{i=1,\dots,N}$ are computed by using (5.12).

The general procedure for the piecewise constant segmentation is summarized in Algorithm 4.

Algorithm 4: DIP-based Piecewise Segmentation

Data: Select the network f_X, initialise the weights $\theta_X^{(0)}$. Select the objective function Φ depending on the noise; set the initial values for $\{c_i\}_{i=1,\dots,N}$ via (5.12) using the output of $f_X(\theta_X^{(0)}, z_X^{(0)})$. Choose the learning algorithm $Algo$ and its hyperparameters. Fix σ^2, and the maximum number of iterations T.

Result: $X^\star$: one–hot enconding of the input image u_0.

Network initialization for f_X, initial values for $\{c_i\}_{i=1,\dots,N}$;
for $k = 1 : T$ **do**
  $z_X^{(k)} \sim \mathcal{N}(0, \sigma^2)$;
  $\theta_X^{(k+1)} \leftarrow Algo(\theta_X^{(k)}, \nabla\Phi(\theta_X^{(k)}))$;
  Update $\{c_i\}_{i=1,\dots,N}$ via (5.12);
end
$X^\star \leftarrow f_X(\theta_X^{(k+1)}, z_X^{(k)})$;

5.2.2 Piecewise smooth segmentation

In this section is instead considered the more general setting, where the image to be segmented is not piecewise constant. In more detail the aim is the minimization of the AT functional defined in (5.3) in order to recover both the denoised image and its contours. In the discrete framework, the rectangular domain $\Omega \subset \mathbb{R}^2$ is discretized by a lattice of $d_1 \times d_2$ points and the values of u_0, u and v are defined only on the grid points. Moreover, given $d = d_1 \cdot d_2$, the images u_0 and u are considered as vectors in $\mathbb{R}^{dm}$ where m represents the number of image channels. The variable $v \in \mathbb{R}^d$ denotes the edges between nodes and whose value is 0 when a contour change is detected and 1 otherwise. In this setting, a discrete counterpart of (5.3) can be written as

$$\underset{u\in\mathbb{R}^{dm},\, v\in\mathbb{R}^d}{\operatorname{argmin}}\ E(u,v) \equiv \underset{u\in\mathbb{R}^{dm},\, v\in\mathbb{R}^N}{\operatorname{argmin}}\ \|u-u_0\|^2 + \lambda\|v\odot Du\|^2 + \mu\mathcal{R}(v), \tag{5.17}$$

where $D \in \mathbb{R}^{d\times dm}$ models a finite difference operator and $\mathcal{R}$ denotes a regularization term which penalizes the length of the contours. Particularly,

$$\mathcal{R}(v) = \varepsilon\|Dv\|^2 + \frac{1}{4\varepsilon}\|1-v\|^2. \tag{5.18}$$

Following a similar approach to that described in [99], a generalized version of (5.17) can be considered. This especially allows to take into account different fidelity terms depending on the noise affecting the data.

Definition 5.1. *Let $\mathcal{L} : \mathbb{R}^{dm} \to (-\infty, +\infty]$ be a fidelity term to the data $u_0 \in \mathbb{R}^{dm}$ and $\mathcal{R} : \mathbb{R}^d \to (-\infty, +\infty]$ be a regularization term enforcing sparsity and acting as a length term. Let $\mathcal{S} : \mathbb{R}^d \times \mathbb{R}^{dm}$*

be a coupling term which penalizes strong variations except at the edges. The general discrete AT-like problem can be written as

$$\min_{u\in\mathbb{R}^{dm},\ v\in\mathbb{R}^{d}} \mathcal{L}(u,u_0)+\lambda\mathcal{S}(v,u)+\mu\mathcal{R}(v). \tag{5.19}$$

According to Definition 5.1, problem (5.17) is simply recovered by setting $\mathcal{L}(u,u_0)=\|u-u_0\|^2$, $\mathcal{S}(v,u)=\|v\odot Du\|^2$.

Starting from (5.19), the DIP approach can be applied to find both u and v. By generalizing the original DIP idea, the unknowns, u and v, are the outputs of two different generative deep networks. In particular, there exist two CNN generators

$$\begin{aligned} f_u \quad &: \quad \mathbb{R}^s\times\mathbb{R}^t\to\mathbb{R}^{dm} \\ &\quad\ (\theta_u,z_u)\mapsto f_u(\theta_u;z_u) \end{aligned}$$

and

$$\begin{aligned} f_v \quad &: \quad \mathbb{R}^s\times\mathbb{R}^t\to\mathbb{R}^{dm} \\ &\quad\ (\theta_v,z_v)\mapsto f_v(\theta_v;z_v) \end{aligned}$$

where θ_u and θ_v stand for the networks parameters to be learned while z_u and z_v are random input vectors sampled from a uniform distribution. By following the DIP paradigm, u and v are replaced by $f(\theta_u;z_u)$ and $f(\theta_v;z_v)$ and the AT-like problem (5.19) can be reformulated as

$$\begin{aligned} \min_{\theta_u\in\mathbb{R}^s,\theta_v\in\mathbb{R}^s} &\mathcal{L}(f_u(\theta_u;z_u),u_0)+\lambda\mathcal{S}(f_v(\theta_v;z_v),f_u(\theta_u;z_u))+\mu\mathcal{R}(f_v(\theta_v;z_v)) \\ &\equiv \Psi(\theta_u,\theta_v). \end{aligned} \tag{5.20}$$

Let the solution of the previous minimization problem be (θ_u^*,θ_v^*). The restored image u^* can be interpreted as the output of the generative network f_u, with the network weights θ_u^* which serve as a parametrization for u^*, i.e. $u^*=f_u(\theta_u^*;z_u)$. Similarly, the contours $v^*=f_v(\theta_v^*;z_v)$ can be represented as $v^*=f_v(\theta_v^*;z_v)$, where the network weights θ_v^* parametrize v^*. This procedure is unsupervised because it operates without an ideal label for the learning process. Instead, training is directed by striving to align the networks output as closely as possible with the observed image.

A standard approach to solve problem (5.20) is represented by the so-called *Gauss-Seidel iteration scheme* [100, 101, 102, 103], which is also well-known in the literature as *alternating minimization*. That is, starting from some given initial point $(\theta_u^{(0)},\theta_u^{(0)})$, the algorithm generates a sequence $\{(\theta_u^{(k)},\theta_u^{(k)})\}_{k\in\mathbb{N}}$ via the scheme

$$\begin{aligned} \theta_u^{(k+1)} &= \operatorname*{argmin}_{\theta_u}\ \Psi(\theta_u,\theta_v^{(k)}) \\ \theta_v^{(k+1)} &= \operatorname*{argmin}_{\theta_v}\ \Psi(\theta_u^{(k+1)},\theta_v) \end{aligned}.$$

Each step of the previous alternating scheme has been solved through a prefixed number of iterations of a gradient based optimization algorithm. The overall method which combines the AT model with the DIP paradigm and described in this section is summarized in Algorithm 5.

5.3 Numerical experiments

In this section the results obtained are reported, illustrating the behaviour of the method with respect to different noisy image segmentation problems through the DIP-based approaches described in Section 5.2. Both piecewise constant and piecewise smooth segmentation are considered.
All the experiments have been carried out on a Windows 11 machine, equipped with 13th Gen Intel Core i5-13500, 32 GB RAM and GeForce RTX 4070 VENTUS 2X 12G OC. MATLAB 2023b and its toolboxes were used for the coding.

Algorithm 5: DIP-based Ambrosio Tortorelli segmentation

Data: Select the networks f_u and f_v, initialise the weights $\theta_u^{(0)}$ and $\theta_v^{(0)}$. Select the objective function Ψ. Choose the learning algorithm $Algo$ and its hyperparameters. Fix σ^2, m, T.
Result: Approximated image $u^\star$ and segmented image $v^\star$

Networks initialization for f_u and f_v;
for $k = 1 : T$ **do**
  for $j = 1 : m$ **do**
    $z_u \sim \mathcal{N}(0, \sigma^2)$;
    $\ell(\theta_u^{(k)}) \leftarrow \Psi(\theta_u^{(k)}, \theta_v^{(k)})$;
    $\theta_u^{(k+1)} \leftarrow Algo(\theta_u^{(k)}, \nabla\ell(\theta_u^{(k)}))$;
  end
  for $j = 1 : m$ **do**
    $z_v \sim \mathcal{N}(0, \sigma^2)$;
    $\ell(\theta_v^{(k)}) \leftarrow \Psi(\theta_u^{(k+1)}, \theta_v^{(k)})$;
    $\theta_v^{(k+1)} \leftarrow Algo(\theta_v^{(k)}, \nabla\ell(\theta_v^{(k)}))$;
  end
end
$u^\star \leftarrow f_u(\theta_u^{(k+1)}, z_u)$;
$v^\star \leftarrow f_v(\theta_v^{(k+1)}, z_v)$;

5.3.1 Piecewise constant segmentation

The first numerical experiments concern the case where the segmented image u is piecewise constant. In the following experiments, different type of noise are considered. The images corrupted by Gaussian noise are taken from two datasets:

- the White Blood Cell (WBC) database [104], which consists in 50 leukocytes images and the labelled ground truths (for performance measurements): such ground truths specify the nucleus, the cytoplasm and the background;

- the GrabCut dataset [105], which includes 50 natural images (animals, people, objects), each of different dimensions, together with the ground truths.

The experiments with *salt and pepper noise* and Poisson noise are tested on images from WBC and MATLAB builtin demo images and on generated *ad hoc* geometric images. The first type of noise is also known as *impulse noise* and it presents itself as sparsely occurring white and black pixels, so it can be modeled as an impulse function that randomly replaces some pixel values in the image with either the maximum or minimum pixel value. The noisy input images for this experiment are obtained via the application of the MATLAB function `imnoise` with the specific setting for the salt and pepper noise. Its density $\delta \in \mathbb{R}^+$, that is the level of noise, was chosen $\delta \in \{0.05, 0.005\}$, hereafter called *high* and *low* noise, respectively. The image corruption thus is applied to approximately $\delta \cdot dm$ pixels, where d and m are the dimensions of the image. In Figures 5.4 and 5.5 the initial data and the respective outcomes are reported: it is evident that independently from the level of noise, a reliable segmentation is achieved. For numerically evaluating the segmentation the Rand index RI is the metric for the reference [106], which addresses the reliability of the classification.

In details, $RI \in [0, 1]$, and the closer RI is to 1, the more reliable is the result. For this experiment, the regularization parameter is $\mu = 10^{-3}$, while the chosen optimizer is Adam with learning rate $\eta = 10^{-3}$. The maximum number of iterations is set to $T = 500$.

Architecture of the network. The convolutional network architecture is inspired by the one in the seminal work [15], keeping the same original settings (see Figure 3.1). The sole difference with the network of [15] consists in having replaced the final layer, originally a Sigmoid layer, with Softmax activation function.

The input z is a 3D tensor having the same spatial dimension of the input image and with 3 channels. In the experiments, the input z is perturbed at each iteration by a component sampled from a Gaussian distribution with zero mean and standard deviation equal to $\frac{1}{100}$, like in the standard DIP framework.

Training and hyperparameters setting. All the results are obtained using Adam for the backpropagation phase with learning rate $\eta = 10^{-3}$. The regularization parameters λ and ν in (5.10) have been set equal to 1 and 10^{-2}, respectively. Moreover the smoothed version of the second term in (5.10) is considered by fixing $\delta = 10^{-4}$. The total number of iterations is equal to 500.

Gaussian noise

Biomedical images are often corrupted by noise due to the acquisition process through the imaging system. In order to evaluate the robustness of the suggested approach with respect to the presence of noise, both clean and noisy input images were considered for the WBC dataset. In particular, the starting noisy input images are created by considering additive white Gaussian noise of zero mean and standard deviation σ equals to $\frac{25}{255}$ and $\frac{50}{255}$. The presence of noise in the input images allows to

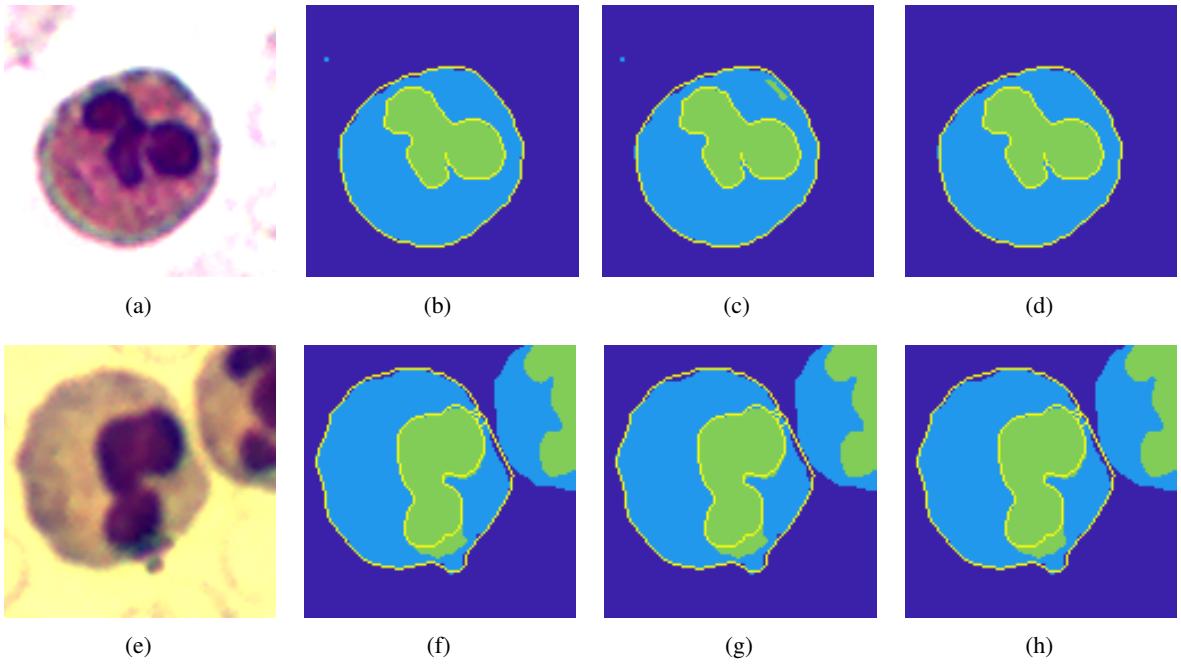


Figure 5.1: Visual inspection of unsupervised segmentation on samples from WBC dataset. (a), (e): true RGB image. (b), (f) Segmentation result with $\sigma = 0$. (c) and (g) Segmentation result with $\sigma = 25/255$. (d), (h) Segmentation result with $\sigma = 50/255$. The light lines depict the boundary of ground truth regions. The results in the second row show how the proposed method is able to segment regions of interest outside the given ground truth.

evaluate if the implicit regularization provided by DIP in the restoration tasks can be also observed in this setting.

Among the measures for establishing the performance of a segmentation process (i.e. Jaccard index, precision, recall), the choice fell on the *Rand index* which measures the similarity between two segmentations of the same image [106]. The Rand index ranges from 0 to 1, with 1 indicating

that the two clusterings are identical and 0 indicating that they are completely different. Table 5.1 reports the mean and the standard deviation of the Rand indices obtained on 50 elements of the WBC dataset for different values of σ. It is possible to conclude that the averaged Rand index is high in all the three cases and the results are robust with respect to the noise. The well known regularization behaviour of DIP can be actually appreciated. Figure 5.1 shows two examples of images taken from the

σ	**Mean**	**St. Deviation**
0	0.9369	0.0673
25/255	0.9333	0.0676
50/255	0.9403	0.0631

Table 5.1: Statistical results of Rand index based on 50 images from WBC dataset.

WBC dataset, together with the results obtained by the segmentation process with different level of noise. Particularly, in 5.1(a) the original RGB image is reported, while in 5.1(b), 5.1(c) and 5.1(d) the segmentation obtained with this method ($\sigma = 0, 25/255, 50/255$) can be appreciated compared to the ground truth. The corresponding Rand indexes are equal to 0.9878, 0.9867 and 0.9880, respectively. The suggested method shows to be very stable with respect to noise.

The results achieved on the cell image of Figure 5.1(e) shed some insights on the numerical performances showed in Table 5.1. Indeed, in this case the image 5.1(f) related to the clean case ($\sigma = 0$) achieves a Rand index of 0.7813, while the version 5.1(g) with corresponding to $\sigma = 25$ has a Rand index of 0.7808 and finally the version without the highest noise level ($\sigma = 50$) 5.1(h) has a Rand index of 0.7807. This relatively poor result is due to the fact that the ground truth considers only the cell in the center of the image, while the proposed procedure is able to segment even the regions of interest which have not been considered when the true labels where created.

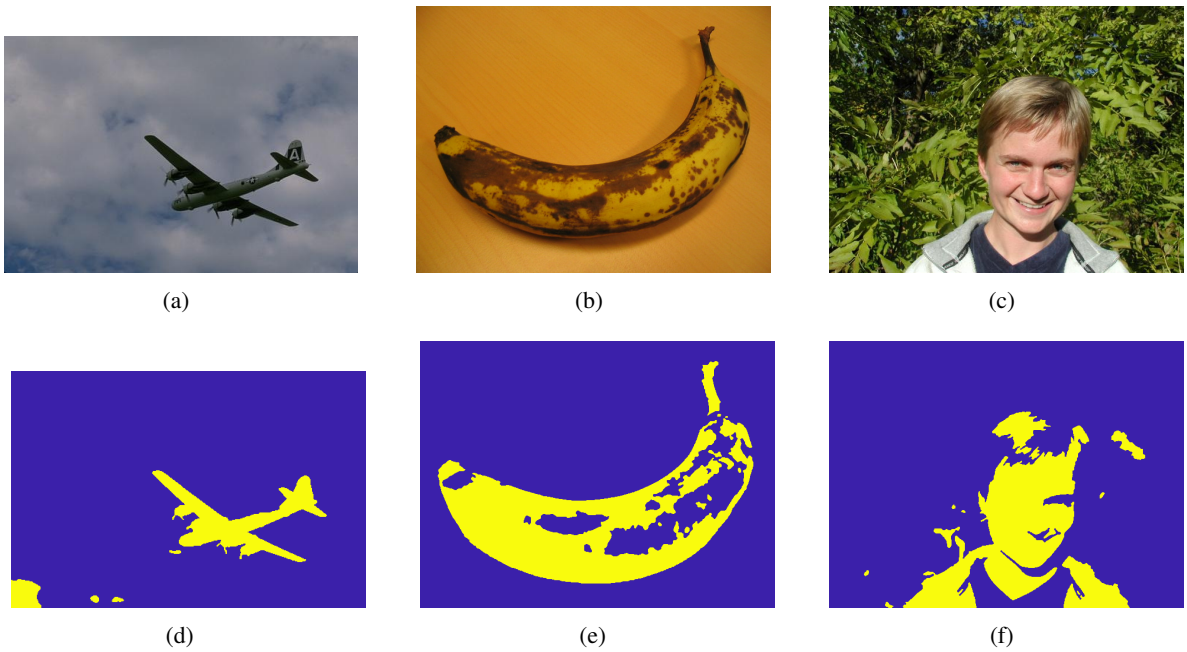
(a) (b)

(c) (d)

Figure 5.2: 3-class unsupervised segmentation with input data corrupted by Gaussian noise. (a) RGB image; (b) segmentation result (classes sky, animals, and landscape).

This subsection is devoted to discuss the qualitative results of the described approach on five images of the Grab-Cut dataset. Figure 5.2 presents the result for the classification where the number of classes is set to 3. Figure 5.3 refers to the background/foreground segmentation task. The segmentation output

for the plane is almost perfect, only a small portion of dark sky is recognized as belonging to the same class of the plane. The banana image presents some spots inside the fruit that are recognized as belonging to the background, due to the very similar color. The case of the portrait of the person in figure 5.3(c) presents some artifacts: indeed, some parts of the person (the hair and the t-shirt) have some common color components with the foliage in the background, hence they are classified as belonging to the same class. Nonetheless, the profile of the person is quite well recognized.



(a) (b) (c)

(d) (e) (f)

Figure 5.3: Foreground-Background extraction with input data corrupted by Gaussian noise. (a)-(b)-(c): Input RGB images; (d)-(e)-(f) relative unsupervised binary segmentation.

Salt and pepper and Poisson noise

Exploring several different values for μ leads to achieve similar results, as depicted in Figure 5.6, so the proposed method is robust with respect to the magnitude of the parameter μ.

Moreover, the same behaviour is observed on data corrupted by Poisson noise, illustrated in Figure 5.7, which refers to the solution to problems as in (5.11). It is evident from Figure 5.8 that the parameter μ does not play a huge role in achieving reliable results: the proposed method is robust with respect to the choice of the regularization parameter regardless of the noise type.

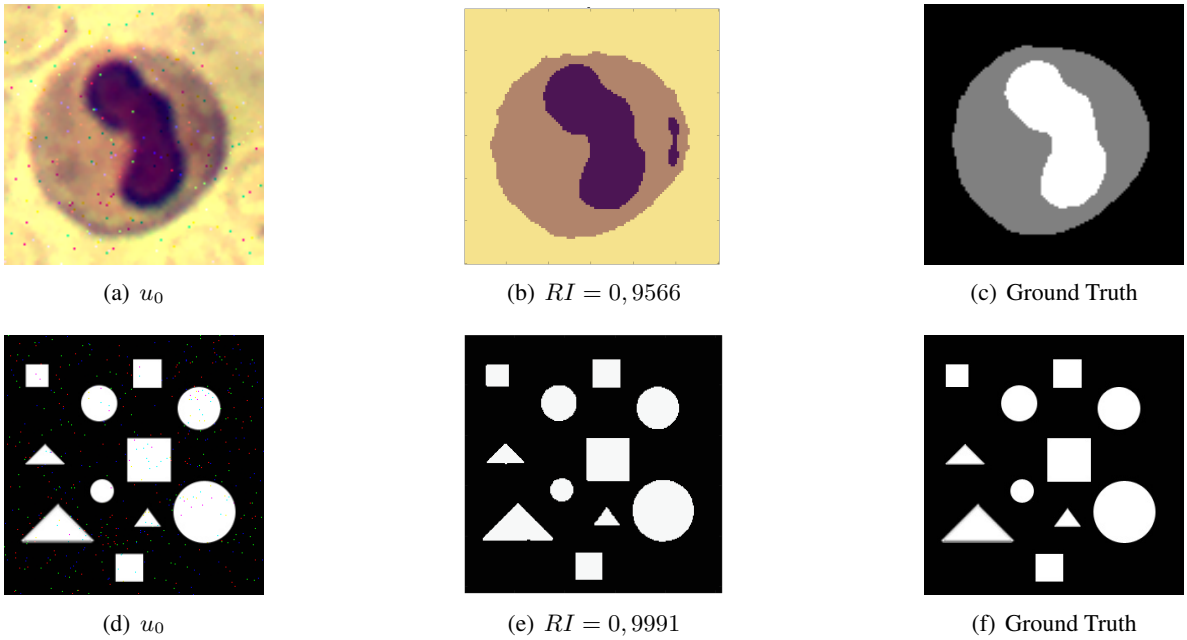
5.3.2 Piecewise smooth segmentation

This subsection presents the numerical results carried out by using Algorithm 5 for piecewise smooth segmentation problems with noisy data, referring to the minimization of (5.17), by changing only the type of noise on the input image.

This second group of experiments is performed over some images from the following datasets:

- Berkeley Segmentation Dataset and Benchmark (BSDS500, [27]);
- MATLAB builtin demo images;
- generated *ad hoc* geometric images.

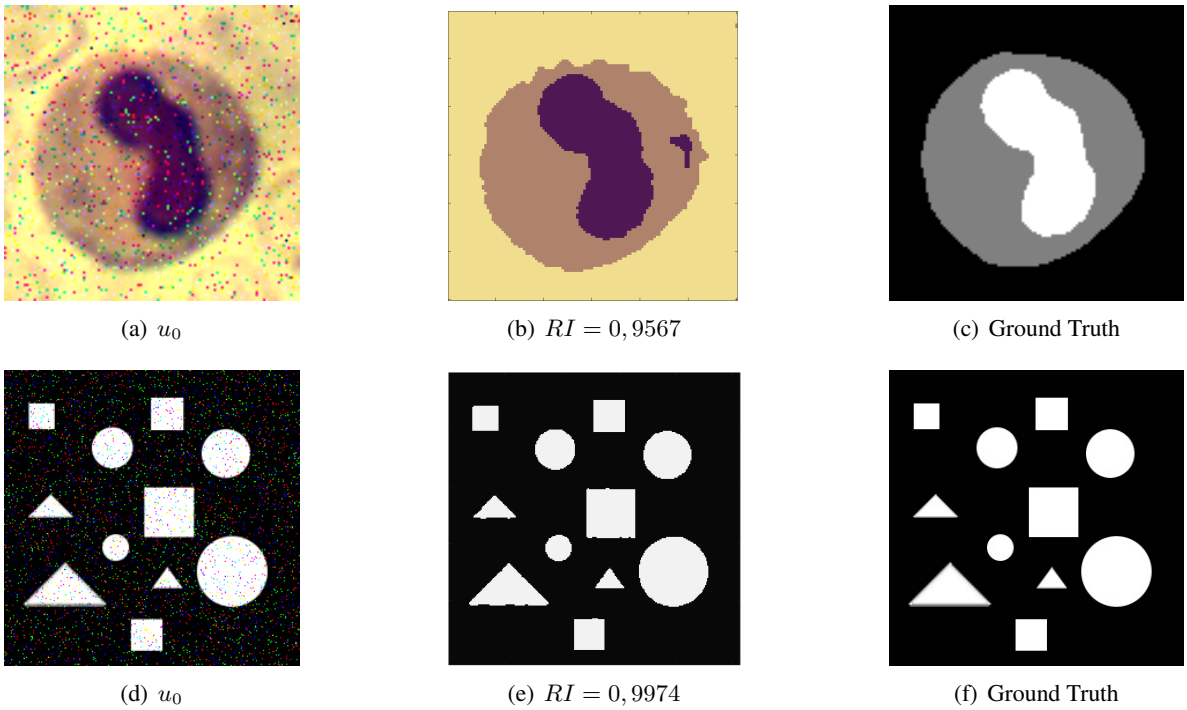
Two different levels of noise, namely *low* and *high*, are proposed. In order to generate u_0, the clean images from either The Berkeley Segmentation Dataset and Benchmark (BSDS500) [27] or



(a) u_0 (b) $RI = 0,9566$ (c) Ground Truth

(d) u_0 (e) $RI = 0,9991$ (f) Ground Truth

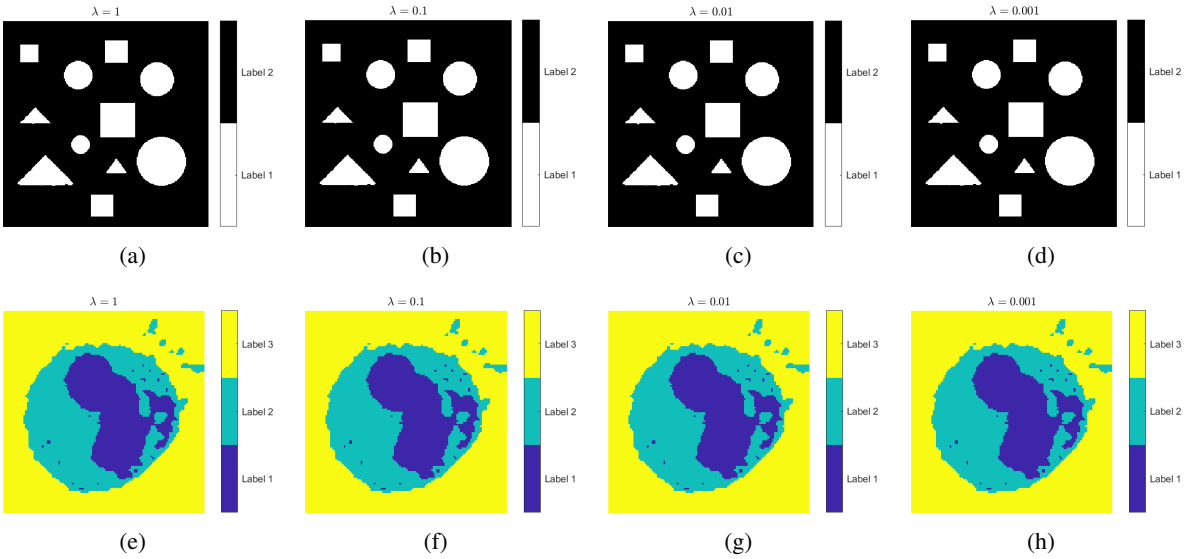
Figure 5.4: Segmentation results with low salt and pepper noise $\delta = 0.005$. The first column shows the noisy input images, the second one the resulting segmented images and the last one the ground truths for the segmentation. The first row illustrates a segmentation with three classes, while the second one a binary segmentation, i.e. there are only two classes. RI is the Rand Index measure.



(a) u_0 (b) $RI = 0,9567$ (c) Ground Truth

(d) u_0 (e) $RI = 0,9974$ (f) Ground Truth

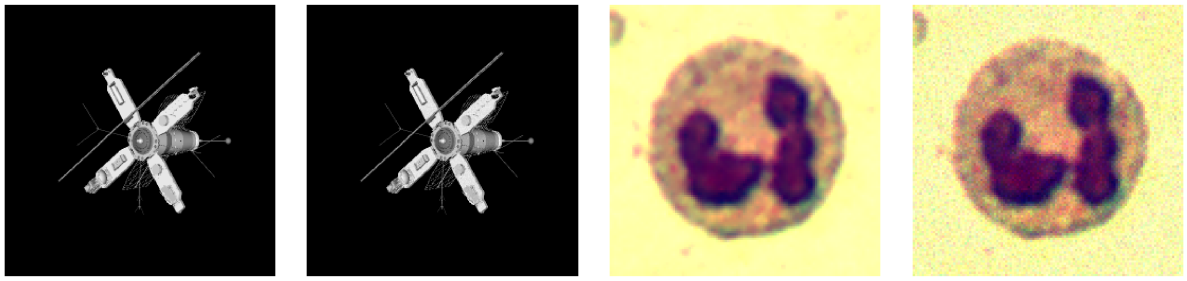
Figure 5.5: Segmentation results with high salt and pepper noise $\delta = 0.05$. The first column shows the noisy input images, the second one the resulting segmented images and the last one the ground truths for the segmentation. The first row illustrates a segmentation with three classes, while the second one a binary segmentation, i.e. there are only two classes. RI is the Rand Index measure.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5.6: Segmentation results with salt and pepper noise through Mumford-Shah minimization. The columns show the segmented images by varying the parameter $\mu \in \{1, 10^{-1}, 10^{-2}, 10^{-3}\}$ and the level of noise is $\delta = 0.05$ (high).



(a) (b) (c) (d)

Figure 5.7: Test images for the experiments with Poisson noise through Mumford-Shah minimization. From left to right: satellite ground truth image; satellite image corrupted by Poisson noise; white blood cell ground truth image; white blood cell image corrupted by Poisson noise.

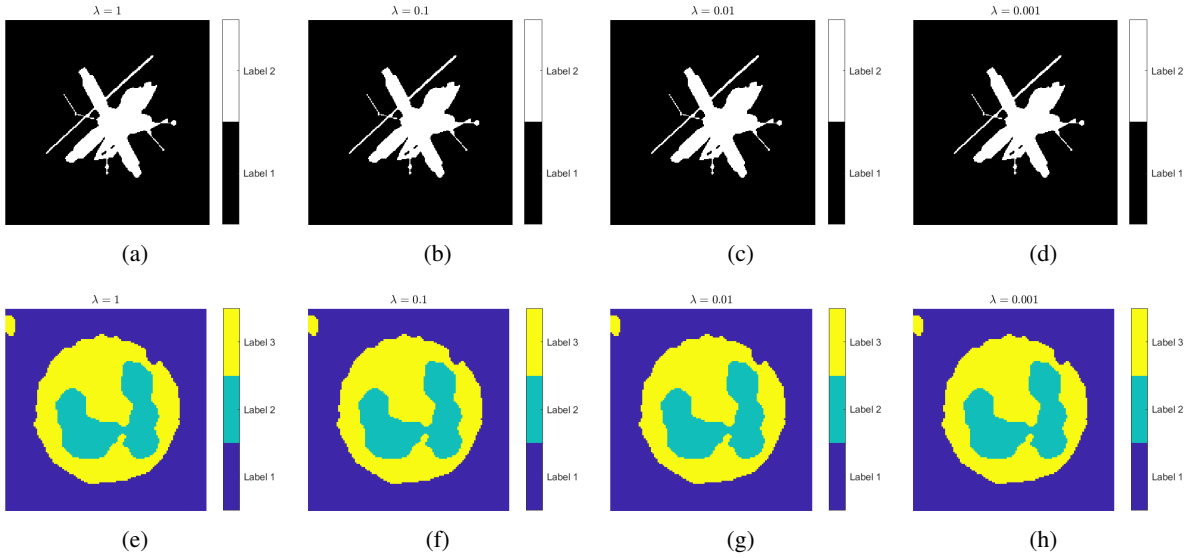


Figure 5.8: Segmentation results with Poisson noise through Mumford-Shah minimization. The columns show the segmented images by varying $\mu \in \{1, 10^{-1}, 10^{-2}, 10^{-3}\}$.

the MATLAB built-in demo were corrupted by adding either Gaussian or Cauchy noise with proper MATLAB functions. Cauchy noise mainly appears in remote sensing, radar, sonar applications and biomedical image acquisition ([107, 108]). The Cauchy noise is characterized by its level γ: the higher the value, the higher the level of perturbation. Moreover, such noise type is signal independent. The discrepancy function employed for denoising images corrupted by Cauchy noise reads as:

$$C(x, g) = \sum_i \log\left(\gamma^2 + (g - x)_i^2\right). \tag{5.21}$$

The considered ground truth images and their corresponding distorted data are reported in the first and the second columns of Figure 5.9, respectively.

For any given u_0, the restored image u and its contours v are obtained by running Algorithm 5 with the following settings: the learning algorithm $Algo$ has been chosen as the Adam optimizer with learning rate equal to 10^{-2}; the total number T of iterations and the number m of inner iterations have been fixed to 100 and 5 respectively. Additionally, following the common practice in the DIP framework, at each iteration the input z is perturbed by a component sampled from a Gaussian distribution with zero mean and standard deviation equal to $\sigma = \frac{1}{100}$. Finally, for the sake of completeness, the expression of the objective function $\Psi(\theta_u, \theta_v)$, discussed in Section 5.2, becomes for both the Gaussian and the Cauchy noise:

$$\begin{aligned}\Psi^{Gauss}(\theta_u, \theta_v) = \|f_u(\theta_u, z_u) - u_0\|^2 + \lambda\|f_v(\theta_v, z_v) \odot Df_u(\theta_u, z_u)\|^2 + \\ + \mu\left(\varepsilon\|Df_v(\theta_v, z_v)\|^2 + \frac{1}{4\varepsilon}\|1 - f_v(\theta_v, z_v)\|^2\right)\end{aligned}$$

$$\begin{aligned}\Psi^{Cauchy}(\theta_u, \theta_v) = \frac{1}{2}\sum_{i=1}^{dm}\log(\gamma^2 + (f_u(\theta_u, z_u) - u_0)^2) + \lambda\|f_v(\theta_v, z_v) \odot Df_u(\theta_u, z_u)\|^2 + \\ + \mu\left(\varepsilon\|Df_v(\theta_v, z_v)\|^2 + \frac{1}{4\varepsilon}\|1 - f_v(\theta_v, z_v)\|^2\right)\end{aligned}$$

To select suitable values for the parameters λ, μ and ε in the standard formulation of the AT functional is typically a challenging problem since numerical evidence shows that they strongly depend on the

image to be segmented. As an example, subsection 5.3.2 reports the results obtained by means of an alternating minimization scheme directly applied to (5.17). On the other hand, all the results obtained through Algorithm 5 and shown hereafter are obtained by always considering $\lambda = 1$, $\mu = 10^{-4}$ and $\varepsilon = 10^{-3}$. The same configuration of these parameters yields good results for images with different features and types of corrupting noise. The inclusion of the DIP approach improves the robustness of AT segmentation with respect to parameter selection due to the intrinsic regularization properties of deep neural networks.

Figure 5.9 shows the results achieved by Algorithm 5 for noisy data u_0 affected by Gaussian noise with zero mean and standard deviation $\sigma = 0.1$. The restored images u and their contours v can be appreciated in the third and the fourth columns respectively. From these results, the proposed approach is able to effectively remove the noise from the input data and to correctly find the contours. Similar considerations can be deduced from the results presented in Figure 5.11 obtained by applying Algorithm 5 to images corrupted by Cauchy noise with parameter $\gamma = 5$. For this second type of noise the maximum number of iteration is set to 200, even if pretty good results are reached with only 100 iterations. The computational times of the results in Figure 5.11 are listed in Table 5.2 and they are referred to 200 iterations.

#Row in Figure 5.11	Time [s]
1	2474
2	758
3	2310
4	2817
5	2311

Table 5.2: Computational times of the results in Figure 5.11.

Classical AT segmentation

This subsection describes the numerical behaviour of an alternating minimization algorithm applied to a classical AT-like optimization problem as in Definition 5.1, focusing on the robustness with respect to the values of the parameters λ, μ an ε. Particularly, the Semi-Linearized Proximal Alternating Minimization (SL-PAM) method [99] has been considered whose MATLAB implementation is available online[1]. In [99], the objective function to be minimized by SL-PAM considers $\mathcal{L}(u, u_0)$ and $\mathcal{S}(v, u)$ as defined in (5.17), while the regularization term $\mathcal{R}$ is fixed as an approximation of the one defined in (5.18), allowing for the computation of its proximal operator using a closed-form solution.

The input data u_0 were generated as previously described, starting from clean images of the BSDS500 dataset, which were then corrupted by Gaussian noise. Figure 5.12 shows the segmentation results obtained by the SL-PAM algorithm in 10000 iterations on input data corrupted with Gaussian noise, where the standard deviation σ is 0.05. The parameters defining the objective function, which depends on the input image, are specified in the figure captions and they are those suggested in the SL-PAM MATLAB code.

The results in Figure 5.12 allow to claim that the SL-PAM method is effective in restoring the ground truth and providing its contours, provided that the parameters λ, μ and ε are properly selected. Indeed, if the standard deviation of the Gaussian noise is slightly modified, the same setting of parameters does not provide satisfactory results, as shown in Figure 5.13. On the other hand, the results depicted in Figure 5.10 show that employing the DIP framework with regularization parameter set on a different noise level allows to obtain reliable results.

Algorithm 5 and SL-PAM are both based on an alternating minimization scheme applied to an AT-like objective function. However, the DIP paradigm applied in devising Algorithm 5 reduces the method's performance dependence on the regularization parameter settings.

[1] https://github.com/mfoare/discrete-mumford-shah

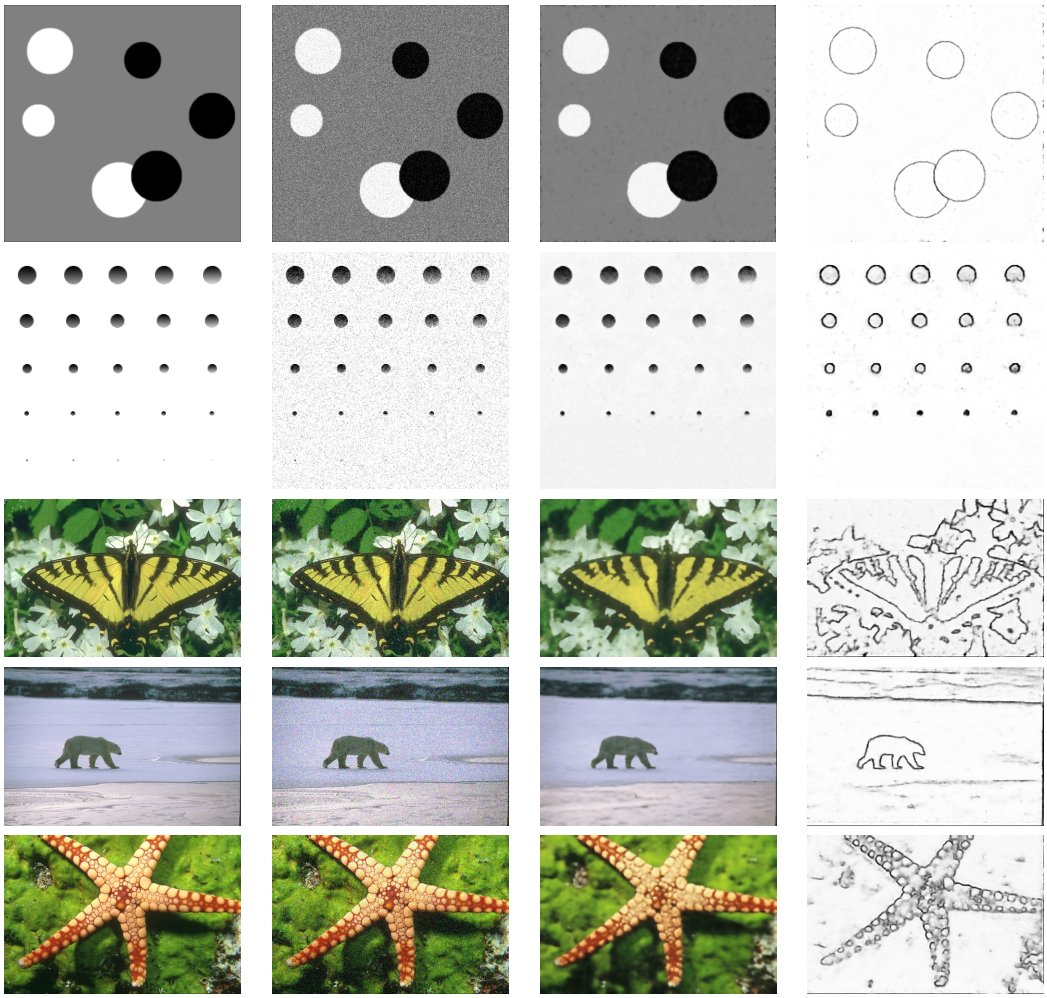


Figure 5.9: Segmentation results obtained by combining the AT functional and the DIP approach through Algorithm 5. The first and the second columns report, respectively, the original images and the corresponding corrupted ones by Gaussian noise with zero mean and standard deviation $\sigma = 0.1$. The third column shows the results of the image approximation u and the fourth column illustrates the contours v.

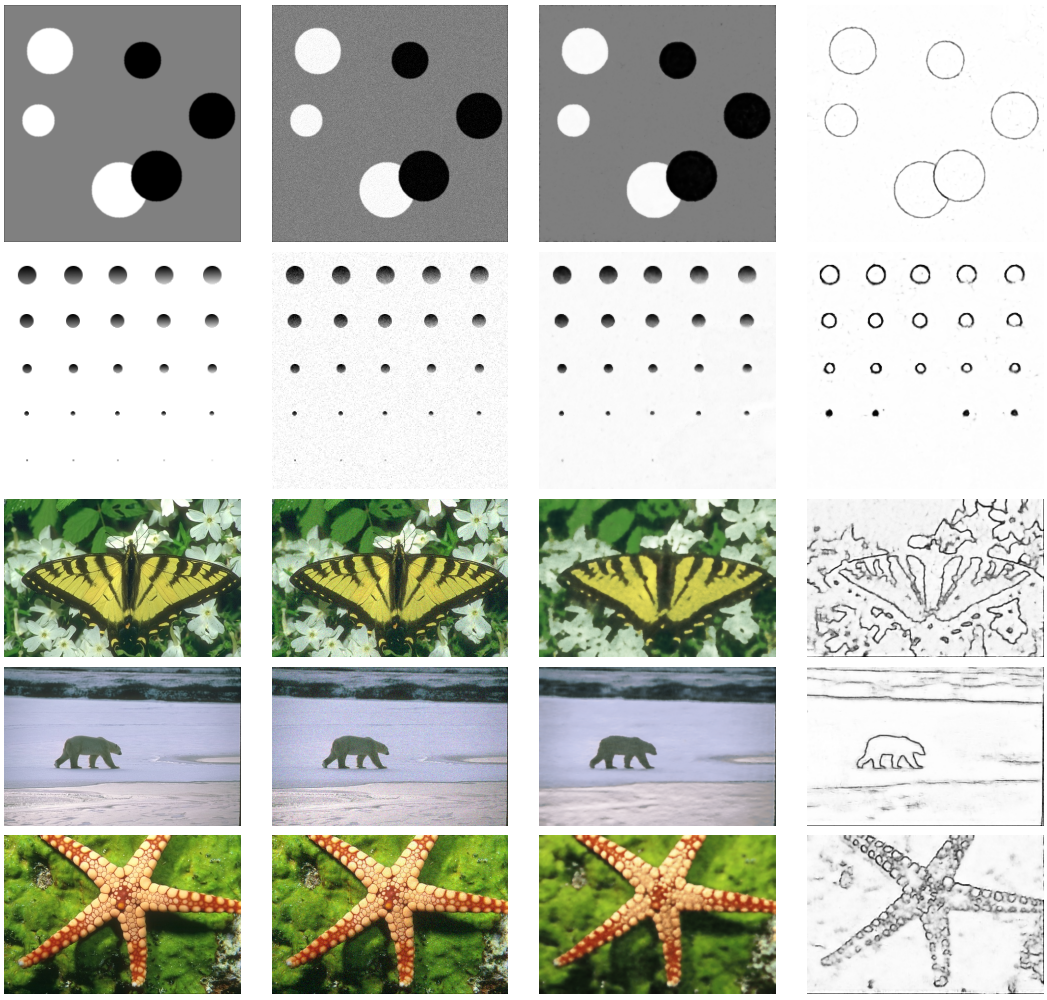


Figure 5.10: Segmentation results obtained by combining the AT functional and the DIP approach through Algorithm 5. The first and the second columns report, respectively, the original images and the corresponding corrupted ones by Gaussian noise with zero mean and standard deviation $\sigma = 0.05$. The third column shows the results of the image approximation u and the fourth column illustrates the contours v.

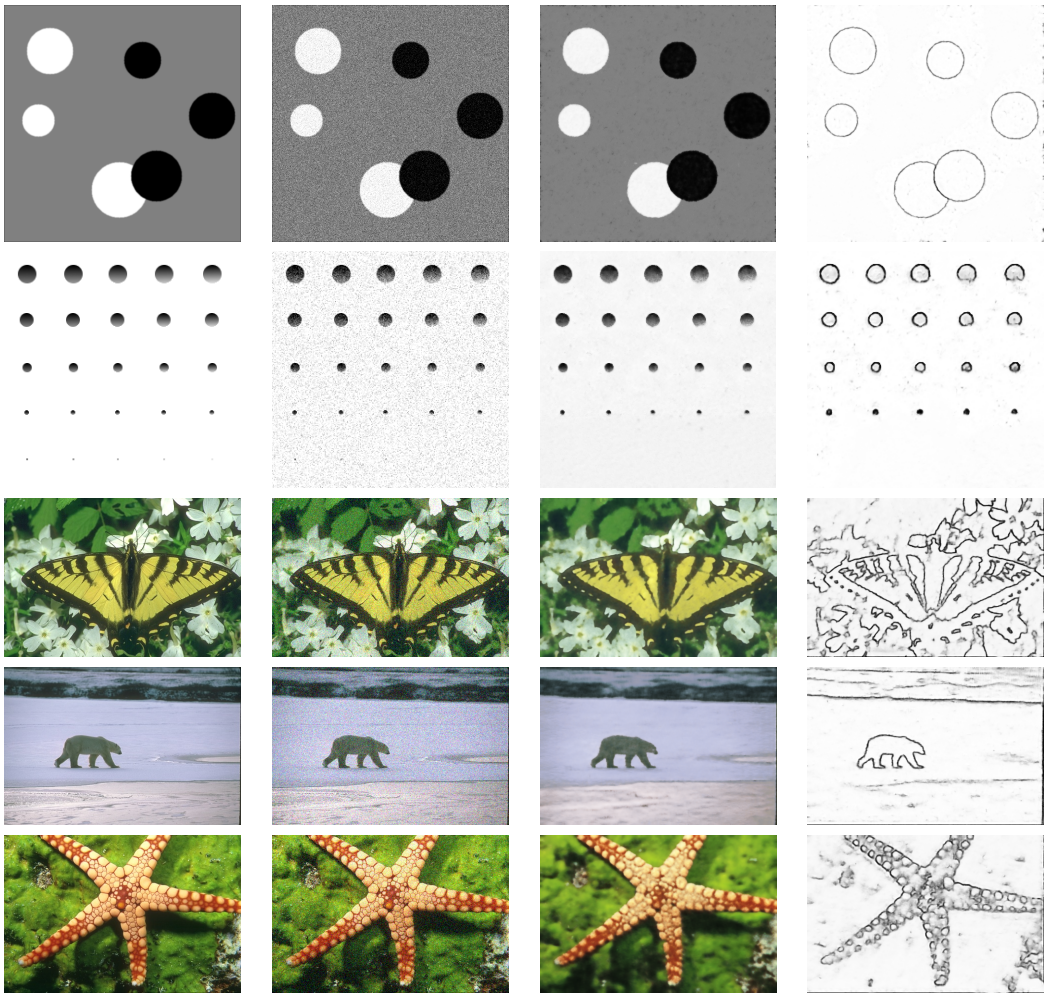
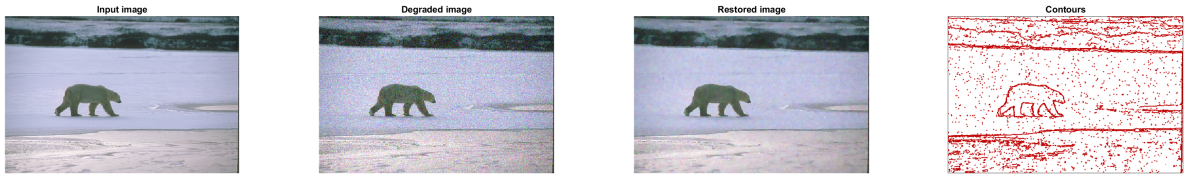


Figure 5.11: Segmentation results obtained by combining the AT functional and the DIP approach through Algorithm 5. The first and the second columns report, respectively, the original images and the corresponding corrupted ones by Cauchy noise with $\gamma = 5$. The third column shows the results of the image approximation u and the fourth column illustrates the contours v.

(a) Clean image (b) Noisy image (c) Approximation u (d) Contours v

(e) Clean image (f) Noisy image (g) Approximation u (h) Contours v

Figure 5.12: Results obtained with SL-PAM for Gaussian noise with standard deviation $\sigma = 0.05$. The parameters involved in the definition of the objective function are different: for the bear image they are fixed as $\lambda = 6$ and $\mu = 0.01$, $\varepsilon = 0.5$; for the starfish image they are set as $\lambda = 5$ and $\mu = 0.005$, $\varepsilon = 0.5$.

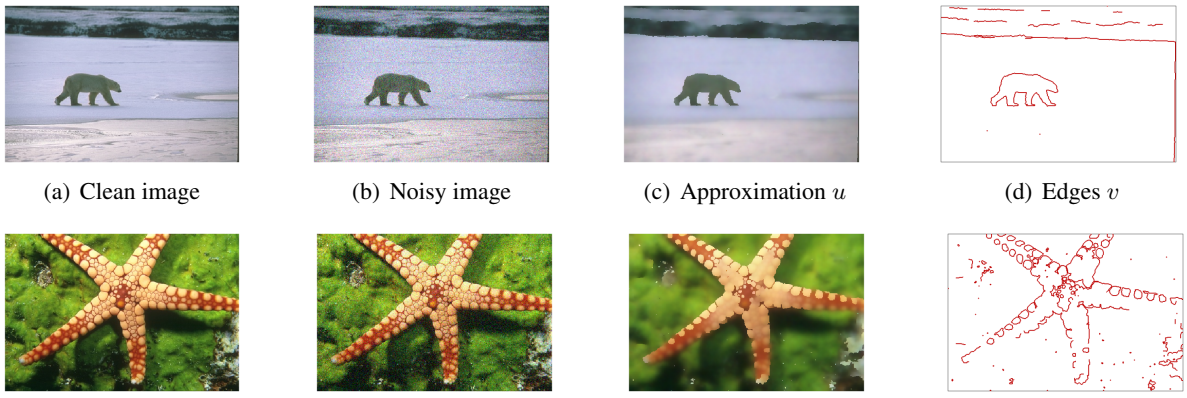


(a) Clean image (b) Noisy image (c) Approximation u (d) Edges v

(e) Clean image (f) Noisy image (g) Approximation u (h) Edges v

Figure 5.13: Results obtained with SL-PAM for Gaussian noise with standard deviation $\sigma = 0.1$. The parameters involved in the definition of the objective function are different: for the bear image they are fixed as $\lambda = 6$ and $\mu = 0.01$, $\varepsilon = 0.5$. For the starfish image are fixed as $\lambda = 5$ and $\mu = 0.005$, $\varepsilon = 0.5$.

If the Gaussian noise level is incremented, by using standard deviation $\sigma = 0.1$, it appears that the original parameter setting is not suitable, like in Figure 5.13, thus finding the best functional parameter configuration becomes a time consuming fine tuning procedure. Trying different parameter configurations with this level of noise, the best is characterized by $\lambda = 50$ and $\mu = 0.1$, $\varepsilon = 0.5$, even if the approximation of the image is very smooth and the edges are missing of some details, like reported in Figures 5.14(c) and 5.14(d), respectively. If the same parameters are used for another natural image, it is evident that the image approximation in Figure 5.14(g) is far from the original one and a good edge reconstruction can't be achieved, see Figure 5.14(h).



(a) Clean image (b) Noisy image (c) Approximation u (d) Edges v

(e) Clean image (f) Noisy image (g) Approximation u (h) Edges v

Figure 5.14: Test using the same optimal parameters for SL-PAM for Gaussian noise with standard deviation $\sigma = 0.1$. The parameters involved in the definition of the objective function have been fixed as $\lambda = 50$ and $\mu = 0.1$, $\varepsilon = 0.5$.

CHAPTER 6

Early stopping strategies in Deep Image Prior

The remarkable performances of Deep Image Prior may be hindered by the so called semi-convergence behaviour. Indeed DIP initially provides good reconstructed images, but after some iterations, depending on the amount of noise in the data, the iterative process starts to overfit the degraded data. Hence, there is the need of an automatic early stopping criterion to select the iteration number at which a reliable result can be obtained. Indeed the original paper provided, among the other hyperparameters, the optimal number of iterations for each task.

In this chapter two different early stopping strategies are described under the DIP framework. The first one is based on a *Neural Architecture Search* (NAS) approach. NAS is a technique within the field of machine and deep learning that automates the design of neural network architectures [109]. NAS methods typically involve exploring the so called *search space* of possible architectures, which can include variations in the number of layers, types of layers, connections between layers, and other architectural hyperparameters. The goal of NAS is to find neural network architectures that achieve high performance on a given task while minimizing the need for human intervention in the design process. The main idea consists of exploiting a NAS strategy to predict an hyperparameters configuration for the DIP net able to halt the learning process before the semi-convergence appears and in a prefixed (typically limited) number N of iterations.

In more detail, once N is fixed, the search space of the original DIP U-Net can be explored for generating the most performing net within N iterations. This NAS-based early stopping strategy not only avoids the semi-convergence phenomenon and the resulting tuning of the optimal number of iterations, but also enables the generation of network configurations that yield favorable outcomes within potentially fewer iterations than those required by the original DIP approach, optimally stopped. Finally in this specific case, the search space, besides the architectural ones, also include the hyperparameters related to both the optimization process and the regularizing term in the loss function. As for the *search strategy*, a random search walk was exploited for exploring the search space.

Specifically a proper performance predictor was trained that, given a test image, is able to provide the best hyperparameter configurations among a pool of different options by observing only the behaviour of the corresponding networks in the very early stage of the training. Moreover, the NAS-based early stopping procedure has been implemented by exploring the search space with two approaches: either keeping fixed the image on which the performance predictor is trained, or by varying it. In the first case it is possible to evaluate the ability of the performance predictor to generalize to very different test images. With the second approach the performance predictor learns to provide image-dependent hyperparameters configurations. Several NAS approaches have been applied to the DIP framework, but they differ from the proposed one with respect to both the steps for realizing the search and the goal they aim to achieve. In [110] the author propose a well-designed binary search space and use as a search strategy a genetic algorithm. They prove that the structure of the network is content-style dependent. The paper [111] restricts the search space to the network components related to the up-sampling part and modifies the original architecture by considering cross-scale residual connections, *i.e.*, the skip connections link blocks of the encoder part with blocks of the decoder part at different spatial dimensions. In [112] the focus is shifted on the metrics used to evaluate the performance: the authors develop several novel metrics that allow to reduce the costs of NAS procedures. It is worth to notice that, differently from the approach proposed in this paper, in none of the previously cited works on NAS for DIP, a performance predictor has been used to improve the DIP behaviour.

The second early stopping strategy relies on an idea inspired by the Blind/Referenceless Image

Spatial Quality Evaluator (BRISQUE) [113] metric, a no-reference image quality measure. In this case the aim is to track the behaviour of the Peak Signal-to-Noise Ratio (PSNR) curve, obtained by applying the standard DIP, without knowing the ground truth image. In particular, using the hyperparameters settings suggested in the original papers [15, 16], a customized version of the BRISQUE metric can be applied to estimate the iteration at which the most reliable result can be obtained under the PSNR measure.

Other early stopping procedures for DIP can be found in the literature and are briefly recalled here below. The variance of the DIP iterates is used for developing an early stopping method in [114], which has been successfully applied in DDIPP approach [115], where blind deconvolution problems for Poisson data have been addressed under the DIP framework. The paper [116] adopts a Bayesian framework and shows that DIP is equivalent to a Gaussian stationary process as soon as the number of input channels goes to infinity: this leads to adopt a Stochastic Gradient Langevin Dynamic algorithm, that avoids the early stopping, but at the cost of a large number of iterations-between 27000 and 37000-which does not meet Green AI principles [117]. In [118] the authors propose an early stopping technique based on the loss function values. However, the semi-convergence behaviour typical of DIP is also reflected in the values of the objective function, unless a proper regularizer is considered. Selecting the regularization term and its corresponding parameter, as well as determining the appropriate number of iterations to stop the DIP algorithm, poses a challenging task.

This chapter is organized as follows: Section 6.1 presents the proposed early stopping strategies, while Section 6.2 is devoted to the numerical validation of the developed techniques.

6.1 Proposed early stopping procedures

This section details two distinct early stopping strategies to be combined with the DIP framework. The perspectives of the two proposed procedures are different. The first one aims not only to prevent the semi-convergence behaviour of DIP, but also to get a neural network configuration able to provide effective results in a small number of iterations, accordingly to the Green AI principles. This aspect is very relevant in all those applications where the computational time is limited or a real-time solution is needed, just consider the biomedical or the autonomous driving fields to name a few. On the other hand, the second strategy does not modify the structure of the neural network on which the original DIP is based, but allows to monitor the generated PSNR behaviour without knowing the ground truth image. The optimization problem described in this section is the one defined in (3.4).

6.1.1 NAS-based early stopping

The first approach is an early stopping strategy *de facto*. Given the particular imaging application and, hence, the specific definition of the discrepancy functional in (3.4), a number of iterations is fixed and NAS techniques search the architecture that can provide the most reliable results in terms of several performance measures, namely the Peak Signal-to-noise Ratio (PSNR) [119], the Structure Similarity Index Measure (SSIM), its multiscale version (MS-SSIM) [119] and the reconstruction relative error (RRE). The search space for the U-Net network of Figure 3.1 encompasses different features:

1. the variance of the input distribution,
2. the depth of the network-the number of stages for the encoder and decoder parts,
3. the number of filters for the convolutional layers,
4. the number of channels for the random input,
5. the number of filters for the skip connections,
6. the value for the regularization parameter,

7. the type of learning algorithm,

8. the learning rate for the learning algorithm.

Since the actual search space is potentially infinite, the focus is on one of its discrete subsets, considering discrete intervals for each of the above options. This search subspace with $\Omega = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_8$, where $\mathcal{A}_j$ is the interval containing the possible values for the j-th item in the above list. Ω can be considered as a subset of $\mathbb{R}^8$. Section 6.2 (Tables 6.1 and 6.3) reports the complete list considered in the numerical experiments of this part. It is quite clear that the possible number of configurations to be taken into account is of course too large to be tested in a brute force manner. Testing them all, to find the most effective one for a given number of iterations, is time consuming and goes against the principles of Green AI [117]. For these reasons, by following a strategy similar to the one suggested in [120], a performance predictor able to provide effective hyperparameters configurations is trained by knowing only the performance of the networks corresponding to a very small subset of all the possible hyperparameters configurations. Below it is detailed how to build the performance predictor and to use it for an early stopping procedure. The main steps of this procedure are summarized in Algorithm 6, each step is detailed below.

Algorithm 6: NAS-based early stopping, single image

Set up.
Set $N, M, B \in \mathbb{N}$. Set Ω, $s \in (0,1)$ and select $Q = \lfloor s\,|\Omega| \rfloor$ hyperparameter configurations: $\{h_q\}_{q=1,\dots,Q}$. Select the c checkpoints $t_1, \dots, t_c$. Select a representative image I of the set $\mathcal{I}$. Choose the performance predictor ρ;

Dataset Creation.
for $q = 1, \dots, Q$ **do**
  Given I, run Algorithm 1 under the h_q configuration, store $P^{(q)}$ and $p_N^{(q)}$;
end

Training.
Train ρ on the dataset $\{(h_q, P^{(q)}), p_N^{(q)}\}_{q=1,\dots,Q}$ such that $\rho(h_q, P^{(q)}) \sim p_N^{(q)}$;

Qualification Phase.
Select M configurations $\tilde{h}_m, m = 1, \dots, M$.;
for $m = 1, \dots, M$ **do**
  Given I, run Algorithm 1 for t_c iterations and store $\tilde{P}^{(m)}$;
  Predict the performance under $\tilde{h}_m$ configuration: $\tilde{p}_N^{(m)} = \rho(\tilde{h}_m, \tilde{P}^{(m)})$;
end

Best Configuration Selection.
Choose the best B configurations $\{\hat{h}_b\}_{b=1,\dots,B}$ according to the predictions $\tilde{p}_N^{(m)}$;
for $b = 1, \dots, B$ **do**
  Given I, run Algorithm 1 under $\hat{h}_b$ configuration; store $\hat{p}_N^{(b)} \equiv p_N(\hat{h}_b)$;
end
Select $h^\star$ such that $h^\star = \underset{b=1,\dots,B}{\operatorname{argmax}}\, \hat{p}_N(\hat{h}_b)$;

Reconstruction of the images from the dataset $\mathcal{I}$.
Apply Algorithm 1 on each image of the dataset $\mathcal{I}$ using the configuration $h^\star$;

Set up. First of all the maximum number N of allowed iterations for Algorithm 1 and the set Ω are fixed. As previously noted, the number of possible configurations could be too large, hence a small percentage of them is randomly selected, namely Q, where $Q = \lfloor s\,|\Omega| \rfloor$, with $s \in (0,1)$, $|\Omega|$ being the cardinality of Ω and $\lfloor\cdot\rfloor$ is the floor function. O is the subset of Ω containing the Q selected configurations. Moreover, c is the selected number of checkpoints for storing the 4 performance metrics (PSNR, SSIM, MS-SSIM, RRE) at iterations $t_1, t_2, \dots, t_c$, which shall be used for training a prediction model ρ. In the experiments a performance predictor based on both a Random Forest (RF) algorithm and a Support Vector Machine for Regression (SVR) algorithm is used. Particularly, the prediction is the averaged values provided by RF and SVR. $M \in \mathbb{N}$ denotes the number of configurations $\{\tilde{h}_m\}_{m=1,\dots,M} \in \Omega \setminus O$ on which the predictor will be tested. $B \in \mathbb{N}, B < M$ denotes the best configurations, called $\{\hat{h}_b\}_{b=1,\dots,B} \subset \{\tilde{h}_m\}_{m=1,\dots,M}$ as predicted by ρ, that will be run for N iterations.

The aim is to search for the best configuration for recovering images of a set $\mathcal{I}$ sharing common characteristics, such as astronomical images corrupted by Poisson noise (see Section 6.2): hence a representative image I is selected, containing the most prominent features of this set.

Dataset creation. Algorithm 1 runs Q times on the image I, under the q-th hyperparameter configuration $h_q \in O$, $q = 1, \dots, Q$. Moreover, the metrics at each checkpoint $t_1, \dots, t_c$ are stored in the columns of $P^{(q)} \in \mathbb{R}^{4\times c}$. The vector $p_N^{(q)} \in \mathbb{R}^4$ stores the metrics at the end of the run, that is at the N-th iteration.

Training. The dataset $\{(h_q, P^{(q)}), p_N^{(q)}\}_{q=1,\dots,Q}$ is used for the training of the performance predictor such that

$$\rho(h_q, P^{(q)}) \sim p_N^{(q)}.$$

Qualification Phase. After these initial phases, a suitable number M of hyperparameters configurations $\{\tilde{h}_m\}_{m=1,\dots,M}$ (different from the configurations considered to train the predictors) are randomly selected. The corresponding M DIP procedures on the image I are invoked to get a good approximation of the example image and the related performance metrics $\tilde{P}^{(m)} \in \mathbb{R}^{4\times c}$ achieved at checkpoints $t_1, \dots, t_c$ are stored. Note that Algorithm 1 runs only for t_c iterations, then for each hyperparameters configuration $\tilde{h}_m$ the pair $(\tilde{h}_m, \tilde{P}^{(m)})$ is used for predicting the metrics $\tilde{p}_N^{(m)} = \rho(\tilde{h}_m, \tilde{P}^{(m)})$ at the N-th iteration.

Best Configuration Selection. Once all these steps are performed, the best $B < M$ configurations $\{\hat{h}_b\}_{b=1,\dots,B}$ are selected among the ones used in the Qualification Phase. Algorithm 1 is run for N iterations for each $\hat{h}_b$: then there are B final performances $\{\hat{p}_N^{(b)} \equiv \hat{p}_N(\hat{h}_b)\}_{b=1,\dots,B}$, where it is made explicit the dependence of the final metrics on the hyperparameters configuration. The best configuration $h^\star$ is taken as the one that corresponds to the highest performance, namely

$$h^\star = \underset{b=1,\dots,B}{\operatorname{argmax}}\, \hat{p}_N(\hat{h}_b).$$

The most effective hyperparameters configuration is then exploited to recover a good reconstruction for all the other images of the dataset in N iterations. As demonstrated in the numerical experiments of the next section, the benefit of this approach consists in finding an effective hyperparameters configuration for the DIP neural network such that, given any image of the dataset, Algorithm 1 is able to provide a good reconstruction in a relative small number of iterations. The predicted best configuration allows to avoid semi-convergence effect for Algorithm 1 and the consequent need for tuning the optimal number of iterations.

Reconstruction of the images from the dataset $\mathcal{I}$. Now Algorithm 1 is applied to all the images of $\mathcal{I}$ using the best configuration $h^\star$: the latter is the setting for DIP that allows us to obtain the best performance in N iterations. When N is small, *e.g.*, $N = 1000$, this amounts to an early stopping strategy, since reliable results are achieved in a low number of iterations.

One can consider a different version of the previously discussed NAS-based early stopping technique where the performance predictor is trained by varying the example image. The aim is to obtain an optimal configuration of hyperparameters that depends on the image itself. In order to reach this goal, the dataset needs to be adjusted on which the predictors are trained. More in detail, besides the values of the hyperparameters and the performance metrics related to the c checkpoints, each sample in the dataset must take into account some peculiar features of the image to be recovered. In order to obtain this image information, the same 36 features are exploited on which the no-reference image quality measure called Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) is based [113].

Notice that the BRISQUE metric is not used at this level, but only the image's features extraction model required to get the BRISQUE values. More details on BRISQUE can be found in Section 6.1.2.

Algorithm 7 reports the scheme of the image dependent NAS-based early stopping procedure. In the following, the main differences between Algorithms 6 and 7 are reported.

Set up. The dataset $\mathcal{I}$ of corrupted images is divided into two disjoint subsets: a training set $\mathcal{T}_r$ and a test set $\mathcal{T}_s$. For each image I_f in $\mathcal{T}_r$, with $f = 1, \dots, |\mathcal{T}_r|$, $F^{(f)} \in \mathbb{R}^{36}$ is the vector storing its 36 significant features which have been extracted by following the same strategy used to train BRISQUE. Moreover, for each image I_f in $\mathcal{T}_r$ Q different hyperparameters configurations $\{h_q^{(f)}\}_{q=1,\dots,Q}$ are considered. The value of Q is fixed as in Algorithm 6. In this approach O is the subset of Ω containing the $Q\,|\mathcal{T}_r|$ selected configurations. $N \in \mathbb{N}$ and $M \in \mathbb{N}$ represent again the maximum number of allowed iterations and the number of configurations needed for the **Qualification phase**, respectively. $B \in \mathbb{N}$ denotes instead the number of training images to be identified as the B most similar to a given test image. This parameter is needed for the reconstruction of the corrupted images.

Dataset creation. Algorithm 1 runs Q times for each of the images taken from the training set $\mathcal{T}_r$. In the columns of $P^{(f,q)} \in \mathbb{R}^{4\times c}$ the metrics at each checkpoint $t_1, \dots, t_c$ obtained by considering the configuration $h_q^{(f)}$ are stored. The vector $p_N^{(f,q)} \in \mathbb{R}^4$ stores the metrics at the end of the run, that is at the N-th iteration.

Training. Differently from Algorithm 6, the predictor ρ is trained on samples that take into account not only the hyperparameters configuration and the metrics in the first checkpoints, but also the features of the image used to get $p_N^{(f,q)}$, $f = 1, \dots, |\mathcal{T}_r|$, $q = 1, \dots, Q$. Particularly, it must hold that

$$\rho(F^{(f)}, h_q, P^{(f,q)}) \sim p_N^{(f,q)},\ f = 1, \dots, |\mathcal{T}_r|, q = 1, \dots, Q.$$

Qualification Phase. The qualification phase of Algorithm 7 differs from the one of Algorithm 6 only from the fact that it is repeated on M different configurations $\{\tilde{h}_m^{(f)}\}_{m=1,\dots,M}$ for every image belonging to the training set. Recall that these M configurations are different from the ones used during the Training.

Reconstruction of the images from the dataset $\mathcal{I}$. This phase is the most distinctive step of Algorithm 7 compared to Algorithm 6. Given a new image I of the test set $\mathcal{T}_s$, the most similar images belonging to the training set $\mathcal{T}_r$ must be identified. To reach this goal, the features of the test image I (stored in a vector called $F \in \mathbb{R}^{36}$) are compared with those of the training images by computing the Mean Squared Error (MSE) between F and $F^{(f)}$, for all $f = 1, \dots, |\mathcal{T}_r|$, namely

$$MSE(f) = \frac{1}{\#(F)}\|F - F^{(f)}\|^2,\ b = 1, \dots, |\mathcal{T}_r|.$$

Algorithm 7: NAS-based early stopping, multiple images

Set up.
Set $N, M, B \in \mathbb{N}$. Divide the dataset $\mathcal{I}$ into $\mathcal{T}_r$ and $\mathcal{T}_s$. Set $\Omega, s \in (0,1)$ and select $Q = \lfloor s\,|\Omega| \rfloor$ hyperparameter configurations for each image in $\mathcal{T}_r$: $\{h_q^{(f)}\}_{q=1,\dots,Q}$. Select the c checkpoints $t_1, \dots, t_c$. Choose the performance predictor ρ;

Dataset Creation.
for $f = 1, \dots, |\mathcal{T}_r|$ **do**
  Select $I_f \in \mathcal{T}_r$;
  for $q = 1, \dots, Q$ **do**
    Given I_f, run Algorithm 1 under the $h_q^{(f)}$ configuration, store $P^{(f,q)}$ and $p_N^{(f,q)}$;
  end
end

Training.
Given the features $\{F^{(f)}\}_{f=1,\dots,|\mathcal{T}_r|}$ of the images considered in the previous step, train ρ on the dataset $\{(F^{(f)}, h_q^{(f)}, P^{(f,q)}), p_N^{(f,q)}\}_{f=1,\dots,|\mathcal{T}_r|;q=1,\dots,Q}$ such that $\rho(F^{(f)}, h_q, P^{(f,q)}) \sim p_N^{(f,q)}$;

Qualification Phase.
for $f = 1, \dots, |\mathcal{T}_r|$ **do**
  Select $I_f \in \mathcal{T}_r$;
  Select M configurations $\{\tilde{h}_m^{(f)}\}_{m=1,\dots,M}$;
  for $m = 1, \dots, M$ **do**
    Given I_f, run Algorithm 1 under $\tilde{h}_m^{(f)}$ for t_c iterations and store $\tilde{P}^{(f,m)}$;
    Predict the performance under the $\tilde{h}_m^{(f)}$ configuration: $\tilde{p}_N^{(f,m)} = \rho(F^{(f)}, \tilde{h}_m^{(f)}, \tilde{P}^{(f,m)})$;
  end
end

Reconstruction of the images from the dataset $\mathcal{I}$.
Given a new image $I \in \mathcal{I}$, compute its features F;
for $s = 1, \dots, |\mathcal{T}_s|$ **do**
  Extract the features $F^{(s)}$ of $I_s \in \mathcal{T}_s$.;
  Compute $MSE(s) = \frac{1}{\#(F)}\|F - F^{(s)}\|^2$;
end
Select those images $\{\hat{I}_b\}_{b=1,\dots,B} \subset \mathcal{T}_s$ corresponding to the B lowest $\{MSE(s)\}_{s=1,\dots,|\mathcal{T}_s|}$;
for $b = 1, \dots, B$ **do**
  Given $\hat{I}_b$, select the best configuration $\hat{h}^{(b)}$ according to the predictions $\{\tilde{p}_N^{(b,m)}\}_{m=1,\dots,M}$;
  Given I, run Algorithm 1 under the configuration $\hat{h}^{(b)}$ for t_c iterations and store $\hat{P}^{(b)}$;
  Predict the performance: $\hat{p}_N^{(b)} = \hat{p}_N^{(b)}(\hat{h}^{(b)}) = \rho(F, \hat{h}^{(b)}, \hat{P}^{(b)})$;
end
Select $h^\star$ such that $h^\star = \underset{b=1,\dots,B}{\arg\max}\, \hat{p}_N^{(b)}(\hat{h}^{(b)})$.;
Given I, run Algorithm 1 under $h^\star$;

After the computation of $MSE(f)$, $f = 1, \dots, |\mathcal{T}_r|$, the B images $\{\hat{I}_b\}_{b=1,\dots,B}$ of the training set corresponding to the B lowest values of $MSE(f)$ are selected. For each $\hat{I}_b$, $b = 1, \dots, B$, the corresponding best configuration $\hat{h}^{(b)}$ predicted in the **Qualification phase** is exploited to run Algorithm 1 for t_c iterations. Based on the metrics at the c checkpoints stored in $\hat{P}^{(b)} \in \mathbb{R}^{4\times c}$, the final performance $\hat{p}_N^{(b)} \equiv \hat{p}_N^{(b)}(\hat{h}^{(b)})$ is predicted as $\rho(F, \hat{h}^{(b)}, \hat{P}^{(b)})$. Finally, the best configuration $h^\star$ for $I \in \mathcal{T}_s$ is given by the one that maximizes $\hat{p}_N^{(b)}$, $f = 1, \dots, B$, namely

$$h^\star = \underset{b=1,\dots,B}{\arg\max}\, \hat{p}_N^{(b)}(\hat{h}^{(b)}).$$

Such configuration $h^\star$ possibly varies with the images in the test set.

6.1.2 BRISQUE-based early stopping

The second early stopping strategy does not aim to modify the structure of the original U-Net architecture considered in the DIP approach and described in Chapter 3. The goal is to monitor the semiconvergence behaviour of DIP in order to stop the algorithm when the most reliable result can be obtained in terms of PSNR. Of course this goal must be reached without exploiting the ground truth image. For this reason, the idea is to exploit a no-reference image quality measure such as the already mentioned BRISQUE metric [113].

BRISQUE is a metric that measures image quality without any reference or ground truth image. The BRISQUE score lies in $[0, 100] \subset \mathbb{R}$: small values correspond to an high perceptual quality, as the one corresponding to clean images, instead large values indicate a worse perception. Real scenarios really benefit from the BRISQUE score, since usually the reference image is not available. The BRISQUE metric is based on a SVR model, typically trained on a dataset of images rated for quality by a human jury, to predict the perceived quality of the image. Indeed, to build the BRISQUE prediction, various statistical properties of an image are firstly extracted, including measures related to natural scene statistics, such as blur, contrast, and noise. Then the model is trained to correlate these extracted features with human subjective quality scores, allowing it to make quality predictions for new images without needing a reference image for comparison.

A naive approach could use BRISQUE as it is defined in the original paper [113]. In more detail, at each iteration k of Algorithm 1, the BRISQUE value related to $f(\theta^{(k)}, z)$ can be computed and once this value starts to increase, Algorithm 1 is stopped. However, as highlighted in [114] and confirmed in the numerical experiments of the next section, this simple approach is not able to improve standard DIP. For this reason, a customized version of the BRISQUE metric is considered by differently training the prediction algorithm on which BRISQUE itself is built. The details of this approach and its combination with DIP is proposed in Algorithm 8.

The main steps are described below.

Set up A collection $\mathcal{C}$ of corrupted images $\{I_f\}_{f=1,\dots,|\mathcal{C}|}$ with known ground truth $\{G_f\}_{f=1,\dots,|\mathcal{C}|}$ is selected, needed to train the customized version of BRISQUE. Moreover all the parameters related to DIP must be selected, particularly the structure of the network f, the initial weights $\theta^{(0)}$, the discrepancy function $\mathcal{D}$, the input variance σ^2, the learning algorithm $Algo$ and its hyperparameters and the maximum number N of iterations. For the numerical experiments, the hyperparameters related to the structure of the network and the optimization algorithm are kept as in the original paper [15]. Finally, in order to early stop DIP, the so-called patience parameter, denoted by τ, must be introduced.

Training of a customized BRISQUE The prediction algorithm to train the customized version of BRISQUE is not simply based on a SVR model as in [113]. Indeed a PSNR prediction which exploits both SVR and RF methods is considered, as explained hereafter. Both a SVR and a RF procedures are trained based on a dataset whose samples are the 36 features of the images $\{I_f\}_{f=1,\dots,|\mathcal{C}|}$ and the labels are the related PSNR values. The features of $\{I_f\}_{f=1,\dots,|\mathcal{C}|}$ are stored in the vectors $\{F^{(f)}\}_{f=1,\dots,|\mathcal{C}|}$. In more detail, given the dataset $\{F^{(f)}, \text{PSNR}(I_f)\}_{f=1,\dots,|\mathcal{C}|}$, the SVR and the RF algorithms are trained such that

$$\text{SVR}(F^{(f)}) \sim \text{PSNR}(I_f),\ f = 1, \dots, |\mathcal{C}|$$

and

$$\text{RF}(F^{(f)}) \sim \text{PSNR}(I_f),\ f = 1, \dots, |\mathcal{C}|.$$

Algorithm 8: BRISQUE-based early stopping

Set up

Fix a collection $\mathcal{C}$ of corrupted images with known ground truth. Select the network f, initialise the network's weights $\theta^{(0)}$, select the discrepancy function $\mathcal{D}$. Set σ^2 for the input's variance. Choose the learning algorithm $Algo$ and its hyperparameters. Select the maximum number of iteration N. Select the patience τ ;

Training of a customized BRISQUE

Given the pairs $\{(I_f, G_f)\}_{f=1,\dots,|\mathcal{C}|}$ of the collection $|\mathcal{C}|$ and the features $\{F^{(f)}\}_{f=1,\dots,\mathcal{C}}$ of $\{I_f\}_{f=1,\dots,|\mathcal{C}|}$, train SVR and RF algorithms on the dataset $\{F^{(f)}, \text{PSNR}(I_f)\}_{f=1,\dots,|\mathcal{C}|}$ such that $\text{cBRISQUE}_1(I_f) \equiv \text{SVR}(F^{(f)}) \sim \text{PSNR}(I_f)$ and $\text{cBRISQUE}_2(I_f) \equiv \text{RF}(F^{(f)}) \sim \text{PSNR}(I_f)$. Moreover $\text{cBRISQUE}_3(I_f) \equiv (\text{SVR}(F^{(f)}) + \text{RF}(F^{(f)}))/2$;

Reconstruction of the images from the dataset $\mathcal{I}$

Set $\tau_1 = \tau_2 = \tau_3 = 0$. Set $\beta_1^\star(\theta^{(0)}) = \beta_2^\star(\theta^{(0)}) = \beta_3^\star(\theta^{(0)}) = 0$;

for $k = 0, 1, \dots, N-1$ **do**

- $z \sim \mathcal{N}(0, \sigma^2)$, a realization from a Gaussian probability distribution.;
- $\ell(\theta^{(k)}) \leftarrow \mathcal{D}\left(f(\theta^{(k)}, z), g\right)$;
- $\theta^{(k+1)} \leftarrow Algo(\theta^{(k)}, \nabla\ell(\theta^{(k)}))$ $\beta_1(\theta^{(k+1)}) = \text{cBRISQUE}_1(f(\theta^{(k+1)}, z))$;
- $\beta_2(\theta^{(k+1)}) = \text{cBRISQUE}_2(f(\theta^{(k+1)}, z))$;
- $\beta_3(\theta^{(k+1)}) = \text{cBRISQUE}_3(f(\theta^{(k+1)}, z))$;
- **if** $\beta_1(\theta^{(k+1)}) > \beta_1^\star(\theta^{(k)})$ & $\tau_1 < \tau$ **then**
  - $\beta_1^\star(\theta^{(k+1)}) = \beta_1(\theta^{(k+1)}), \quad \tau_1 = 0;$
- **else**
  - $\beta_1^\star(\theta^{(k+1)}) = \beta_1(\theta^{(k)}), \quad \tau_1 = \max\{\tau_1 + 1, \tau\};$
- **end**
- **if** $\beta_2(\theta^{(k+1)}) > \beta_2^\star(\theta^{(k)})$ & $\tau_2 < \tau$ **then**
  - $\beta_2^\star(\theta^{(k+1)}) = \beta_2(\theta^{(k+1)}), \quad \tau_2 = 0$
- **else**
  - $\beta_2^\star(\theta^{(k+1)}) = \beta_2^\star(\theta^{(k)}), \quad \tau_2 = \max\{\tau_2 + 1, \tau\}$
- **end**
- **if** $\beta_3(\theta^{(k+1)}) > \beta_3^\star(\theta^{(k)})$ & $\tau_3 < \tau$ **then**
  - $\beta_3^\star(\theta^{(k+1)}) = \beta_3(\theta^{(k+1)}), \quad \tau_3 = 0$
- **else**
  - $\beta_3^\star(\theta^{(k+1)}) = \beta_3(\theta^{(k)}), \quad \tau_3 = \max\{\tau_3 + 1, \tau\}$
- **end**
- **if** $\tau_1 = \tau$ & $\tau_2 = \tau$ & $\tau_3 = \tau$ **then**
  - $\theta^\star = \text{argmax}_{\theta^{(k+1)}}\{\beta_1^*(\theta^{(k+1)}), \beta_2^*(\theta^{(k+1)}), \beta_3^*(\theta^{(k+1)})\}$;
  - break
- **else**
  - $\theta^\star = \theta^{(k+1)}$
- **end**

end

Recover the approximation of $u^\star$ as $f(\theta^\star, z)$;

Given these two trained models and a new image I with features vector F, it is possible to consider different customized versions of the BRISQUE metric defined as

$$\begin{aligned}\text{cBRISQUE}_1(I) &= \text{SVR}(F),\\ \text{cBRISQUE}_2(I) &= \text{RF}(F),\\ \text{cBRISQUE}_3(I) &= \frac{\text{SVR}(F)+\text{RF}(F)}{2}.\end{aligned}$$

As a consequence, cBRISQUE_i, $i = 1, 2, 3$, are able to estimate the PSNR of a new image without knowing the corresponding ground truth and, hence, they can be used to properly stop DIP when new datasets of corrupted images are considered.

Reconstruction of the images from the dataset $\mathcal{I}$ Given a dataset $\mathcal{I}$ of corrupted images, possibly with unknown ground truth, the goal is to apply the DIP algorithm to recover proper reconstructions by avoiding semi-convergence effect. For this reason it occurs a modified version of Algorithm 1 where the cBRISQUE_i, $i = 1, 2, 3$, metrics are employed. Particularly, once the new iterate $\theta^{(k+1)}$ is updated, the cBRISQUE_i values β_i^{k+1}, $i = 1, 2, 3$, of the corresponding image $f(\theta^{(k+1)}, z)$ are computed. When all the cBRISQUE_i metrics are not reduced after a number of successive iterations equal to the value of the patience τ, the DIP procedure is stopped even if the maximum number of iterations is not reached. In this case, the approximation $u^\star$ is computed by means of that vector of weights corresponding to the largest PSNR value predicted by cBRISQUE_1, cBRISQUE_2 and cBRISQUE_3.

6.2 Numerical experiments

This section is devoted to show the numerical experiments validating the proposed approaches on denoising problems. In particular two types of noise, namely Poisson and Cauchy noise, are considered. Hereafter, the vectorized version of a 2D image is defined, namely $x \in \mathbb{R}^n, n = pq$.

As introduced in Section 4.1, the discrepancy functional used for restoring images perturbed by this kind of noise is the generalized Kullback-Leibler function, that for denoising problems reads as:

$$KL(x, g) = \sum_i g_i \ln\left(\frac{g_i}{(x+b)_i}\right) + \sum_i (x + b - g)_i\,, \tag{6.1}$$

where $g_i \ln g_i = 0$ if $g_i = 0$.

In the experiments with Cauchy noise, as stated in subsection 5.3.2, the discrepancy function employed for denoising images reads as:

$$C(x, g) = \sum_i \log\left(\gamma^2 + (g - x)_i^2\right). \tag{6.2}$$

The datasets tested in the experiments are the following:

- Top 100 Images of the Hubble telescope [121, 122], hereafter called NASA dataset. The images are processed, shrinking them into $N \times M \times 3$ and then perturbed with Poisson noise by means of a custom MATLAB function;
- Berkeley Segmentation Dataset and Benchmark (BSDS500, [27]). 49 images are selected and corrupted via Cauchy noise with parameter $\gamma = 1, 5, 10$ via a custom MATLAB function.

All the experiments have been carried out on a Windows 11 machine, equipped with 13th Gen Intel Core i5-13500, 32 GB RAM and GeForce RTX 4070 VENTUS 2X 12G OC. Python 3.10 and Torch libraries were employed for the deep learning procedures.

6.2.1 NAS-based early stopping

Two distinct numerical experiments were performed to validate Algorithm 6 and Algorithm 7. In testing Algorithm 6 and Algorithm 7 different noise affecting the images and different datasets of images were considered.

The first set of experiments exploits the NASA dataset. Hereafter the search space underlying Algorithm 6 is detailed. The hyperparameters space described in Section 6.1 is discretized as reported in Table 6.1. In details, the set Ω consists of the hyperparameters configurations composed by: the standard deviation σ for the input Gaussian noise and its number of channels C; the number of upsampling and downsampling stages of the network, for the encoder and the decoder, respectively; the number of filters for the skip connections and for the convolutional layer; the regularization parameter λ; the choice of the optimization algorithm for minimizing the loss, such as Stochastic Gradient Descent (SGD), SGD with momentum with $\beta = 0.9$ (SGDM) or Adam; the learning rate lr. This discretized search subspace leads us to engage 145152 samples, see Table 6.1 for details on the values for each hyperparameter.

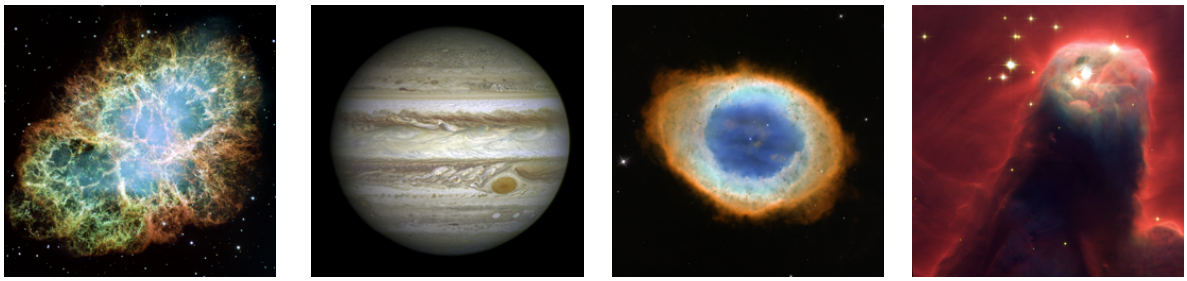
Network Architecture						Optimization Algorithm	
σ	C	# Stages	#filters (skip)	#filters (conv)	λ	optimizer	lr
$1/50$	8	3	2	8	**0**	SGD	$\mathbf{10^{-3}}$
$1/30$	16	4	4	16	$3.16 \cdot 10^{-5}$	SGDM	10^{-4}
$\mathbf{1/20}$	**32**	**5**	**8**	**32**	10^{-3}	**Adam**	10^{-2}
$1/10$	64	6		64	10^{-4}		10^{-5}
				128	$3.16 \cdot 10^{-4}$		$5 \cdot 10^{-4}$
				256	10^{-5}		$5 \cdot 10^{-6}$
							$5 \cdot 10^{-3}$

Table 6.1: Space of the hyperparameters for Algorithm 6. The values in bold are the default ones for the vanilla DIP.

For the **Set up** part of Algorithm 6 from Ω is sampled the $0,04\%$ ($s = 0.0004$) of all possible configurations, namely $Q = 58$, mimicking the approach in [120]. Let $M = 200$ and $B = 20$; moreover, $c = 3$ checkpoints are chosen, namely the PSNR, the SSIM, the MS-SSIM and the RRE are stored at iterations $t_1 = 50, t_2 = 100$ and $t_3 = 150$. The total number of iteration is $N = 1000$. The used label for the performance predictor training is the PSNR at the N-th iteration. The performance predictor is based on both the SVR and the Random Forest algorithms. In particular, the averaged values provided by RF and SVR is considered as a prediction. The representative image is the one depicted in Figure 6.1(a): this image was chosen because it presents both the characteristics of images with planets (centered object with dark background) and those with nebulae (stars in the background). Some examples are reported in Figures 6.1(b)-6.1(d).

In the **Training** phase of Algorithm 6 Algorithm 1 is applied to the image 6.1(a) using the configurations $\{h_q\}_{q=1,\dots,58}$, then the **Qualification Phase** is performed on other $M = 200$ configurations $\{\tilde{h}_m\}_{m=1,\dots,200}$. The best $B = 20$ configurations are chosen and, for each of them, Algorithm 1 is run for $N = 1000$ iterations. We emphasize that 1000 iterations represent a relatively low number in this context. Finally, in the **Best Configuration Selection** phase the best configuration $h^\star$ (Table 6.2) is picked and the last step (**Reconstruction of the images from the dataset** $\mathcal{I}$) of Algorithm 6 is run: Algorithm 1 is applied to each image of the NASA dataset under the configuration $h^\star$.

Figure 6.2 illustrates the results obtained in 1000 iterations by applying Algorithm 1 combined with both the standard DIP hyperparameters configuration and the $h^\star$ one (found by Algorithm 6) on the images 6.1(b), 6.1(c), 6.1(d). Also the corresponding PSNR values are reported. Similar results have been obtained for the other images of the NASA dataset. It is evident that within an economic perspective on the number of iterations, that is within 1000 iterations, a larger PSNR value is achieved, indicating a better quality in the recovered image. The results propose sharper images, without loosing



(a) Crab Nebula (b) Jupiter (c) Ring Nebula (d) Cone Nebula

Figure 6.1: Images from the NASA dataset. The image in the first panel has been used in the Dataset Creation phase of Algorithm 6.

details. Notice that the possibility of a semi-convergence behaviour is a priori excluded, as Algorithm 6 systematically explores effective configurations within the given N iterations. Finally it is worth noting that one might think that simply increasing the learning rate for the optimization algorithm would suffice to outperform standard DIP. However, this thesis is contradicted by Table 6.2, where it is clear that the learning rate for $h^\star$ is even smaller than that used in vanilla DIP.

σ	C	# Stages	#filters (skip)	#filters (conv)	λ	optimizer	lr
0.02	16	3	2	128	0	Adam	0.0005

Table 6.2: Best configuration $h^\star$ obtained by Algorithm 6 on the NASA dataset.

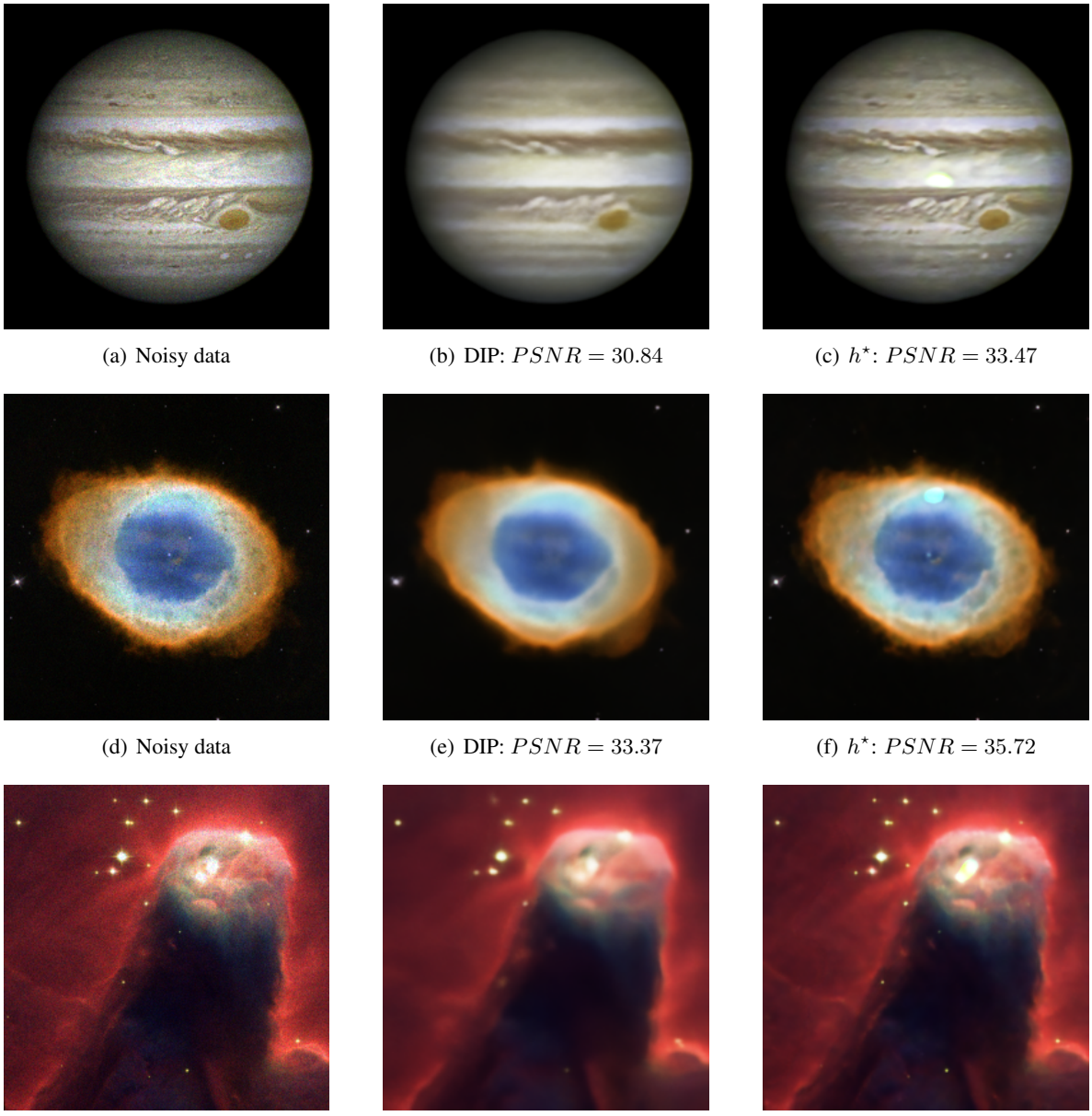
The second numerical experiment regards Algorithm 7 and it is carried out on the BSDS500 dataset, whose images have been corrupted by Cauchy noise. Since natural images, such as the ones belonging to this dataset, do not present common features as in the previous astronomical case, Algorithm 7 is applied, which includes in the search subspace several images from the dataset. This allows to provide optimal configurations depending on the image to be treated.

For the **Set up** phase of Algorithm 7 the standard deviation for Gaussian input noise is $\sigma = 1/20$ and the number of channels of the input is $C = 32$ as in vanilla DIP approach, since the BRISQUE features increase the dimension of the search space Ω. The space of the hyperparameters is reported in Table 6.3. Then $N = 1000$, $M = 100$ and $B = 3$, while the training set and the test set are such that $|\mathcal{T}_r| = 120 = 40 \times 3$ and $|\mathcal{T}_s| = 9$. Particularly, the training set is formed by taking 40 images from the BSDS500 dataset; each one is then corrupted by Cauchy noise at levels $\gamma = 1, 5, 10$, yielding then a grand total of 120 images. As for Algorithm 6, $c = 3$ and $t_1 = 50, t_2 = 100, t_3 = 150$; on the other hand, s is selected such as $Q = 360$.

Network Architecture				Optimization Algorithm	
# Stages	#filters (skip)	#filters (conv)	λ	optimizer	lr
3	2	8	**0**	SGD	$\mathbf{10^{-3}}$
4	4	16	10^{-2}	SGDM	10^{-4}
5	**8**	**32**	10^{-3}	**Adam**	$5 \cdot 10^{-5}$
		128			10^{-5}
		256			

Table 6.3: Space of the hyperparameters for Algorithm 7. The values in bold are the default ones for the vanilla DIP.

The **Dataset Creation** phase encompasses the generation of the dataset needed to train the performance predictor. Each row of the dataset refers to a particular image in the training set and a



(a) Noisy data (b) DIP: $PSNR = 30.84$ (c) $h^\star$: $PSNR = 33.47$

(d) Noisy data (e) DIP: $PSNR = 33.37$ (f) $h^\star$: $PSNR = 35.72$

(g) Noisy data (h) DIP: $PSNR = 34.33$ (i) $h^\star$: $PSNR = 36.80$

Figure 6.2: Results on images from Figure 6.1 after 1000 iterations. Left panel: vanilla DIP results. Middle panel: noisy image. Right panel: image obtained via the best configuration $h^\star$ for the hyperparameters reported in Table 6.2.

(a) (b)

Figure 6.3: Test images from the BSDS500 dataset.

particular hyperparameters configuration. Indeed it consists of the 36 features $F^{(f)}$ of the considered image belonging to $\mathcal{T}_r$, the configuration $h_q^{(f)}$ of 6 hyperparameters from Table 6.3, the 4 metrics at $c = 3$ checkpoints stored in $P^{(f,q)}$ and the label $p_N^{(f,q)}$, namely the final performance corresponding to $(F^{(f)}, h_q^{(f)}, P^{(f,q)})$.

For the **Training** phase the predictor ρ is considered as in the previous numerical experiment, that is, the prediction is the averaged values provided by a RF and a SVR algorithms trained on the dataset $\{(F^{(f)}, h_q^{(f)}, P^{(f,q)}), p_N^{(f,q)}\}_{f=1,\dots,|\mathcal{T}_r|,q=1,\dots,Q}$. The **Qualification Phase** requires, for each image in the test set $\mathcal{T}_s$, to run Algorithm 1 under $M = 100$ different configurations for 150 epochs[1]: thus the prediction is $\{\tilde{p}_N^{(f,m))}\}_{f=1,\dots,|\mathcal{T}_r|,m=1,\dots,M}$ values, namely the final performance of the 12000 configurations.

For the last phase, the **Reconstruction of the images from the dataset** $\mathcal{I}$, by using the ℓ^2 norm, the distances are measured between the BRISQUE statistics of each image in the test set $\mathcal{T}_s$ and those in the training set $\mathcal{T}_r$. In this way, the *nearest* three images belonging to $\mathcal{T}_r$ contribute to find $B = 3$ possible optimal configurations among the $\tilde{h}_m$ ones. Algorithm 1 combined with these B configurations is run on the considered test image until 150 iterations. Then, based on the predictions provided by ρ, Algorithm 1 is run for 1000 iterations with the configuration predicted as the best among the three.

Figure 6.3 shows two of the nine test images of $\mathcal{T}_s$ used to evaluate the performance of Algorithm 7. Figures 6.4 and 6.5 depict the achieved results in 1000 iterations with the vanilla DIP and the different best configurations. In detail, the first column reports the noisy images with different level of noise, the second column shows the recovered image via the vanilla DIP and the last one the result gained by DIP equipped with the optimal configuration provided by the performance predictor. The best predicted configurations for the images shown in Figures 6.4 and 6.5 and corrupted by different level of Cauchy noise are listed in Table 6.4. It is evident that the classical DIP has a large smoothing behaviour, while an optimal architecture is able to restore more sharp details and visual feature of the image. In those results, the optimal configuration is similar to the vanilla DIP, but the number of stages is lower with a greater number of filters for the convolutional layers.

Finally, Figure 6.6 summarizes the results obtained for all the test images considered in this second experiment. In particular, a comparison among the PSNR values obtained by DIP combined either with the original hyperparameters configuration or the best one provided by Algorithm 7 is depicted. The index identifying the test image is reported on the x-axis, the dots correspond to the PSNR values and the lines represent the averaged PSNR value computed on the nine test images. From Figures 6.6(a) and 6.6(b) it is evident that DIP equipped by the the configuration $h^\star$ outperforms the standard DIP in terms of PSNR for all the test images corrupted by Cauchy noise with both $\gamma = 1$ and $\gamma = 5$. For $\gamma = 10$, the results of the two approaches are comparable (see Figure 6.6(c)).

[1] An epoch refers to one complete pass through the entire training dataset during the training of a model.

test image	lr	optimizer	#filters (conv)	#filters (skip)	#stages	λ
Figure 6.3(a), $\gamma = 1$	$0,001$	Adam	256	8	3	0
Figure 6.3(a), $\gamma = 5$	$0,001$	Adam	128	4	3	0
Figure 6.3(a), $\gamma = 10$	$0,001$	Adam	128	4	3	0
Figure 6.3(b), $\gamma = 1; 5$	$0,001$	Adam	256	2	3	0
Figure 6.3(b), $\gamma = 10$	$0,001$	Adam	256	8	3	0

Table 6.4: Best configurations provided by Algorithm 7 with $N = 1000$ iterations for the test images in Figure 6.3 corrupted by Cauchy noise of several levels.

(a) Noisy data, $\gamma = 1$ (b) DIP $PSNR = 27.27$ (c) $h^\star$ $PSNR = 29.27$

(d) Noisy data, $\gamma = 5$ (e) DIP $PSNR = 25.36$ (f) $h^\star$ $PSNR = 25.77$

(g) Noisy data, $\gamma = 10$ (h) DIP $PSNR = 23.64$ (i) $h^\star$ $PSNR = 23.77$

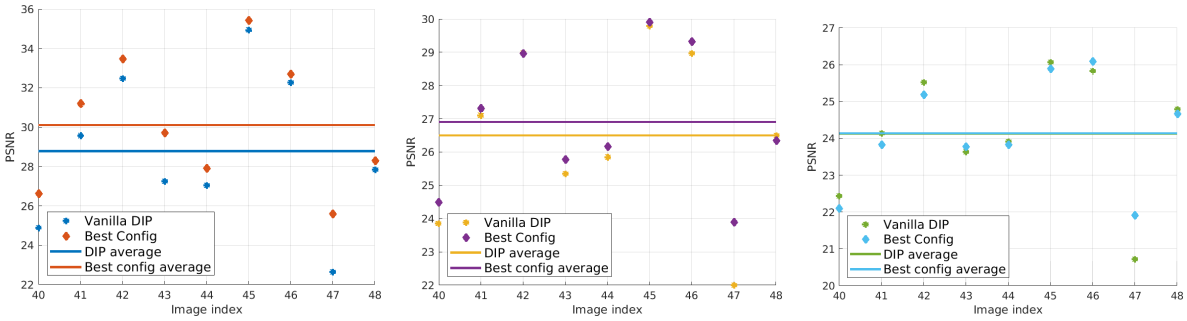
Figure 6.4: Results achieved on Figure 6.3(a) corrupted by Cauchy noise for $\gamma = 1$ (top row), $\gamma = 5$ (middle row) and $\gamma = 10$ (bottom row) via standard DIP (second column) and DIP with the best configuration reported in Table 6.4 (third column).

(a) Noisy data, $\gamma = 1$ (b) DIP $PSNR = 22.66$ (c) $h^\star$ $PSNR = 25.58$

(d) Noisy data, $\gamma = 5$ (e) DIP $PSNR = 22.01$ (f) $h^\star$ $PSNR = 23.89$

(g) Noisy data, $\gamma = 10$ (h) DIP $PSNR = 20.72$ (i) $h^\star$ $PSNR = 21.91$

Figure 6.5: Results achieved on Figure 6.3(b) corrupted by Cauchy noise for $\gamma = 1$ (top row), $\gamma = 5$ (middle row) and $\gamma = 10$ (bottom row) via standard DIP (second column) and DIP with the best configuration reported in Table 6.4 (third column).



(a) $\gamma = 1$ (b) $\gamma = 5$ (c) $\gamma = 10$

Figure 6.6: Results on test images from BSDS500 dataset with Cauchy noise.

6.2.2 BRISQUE-based early stopping

This section is devoted to evaluate the effectiveness of the BRISQUE-based early stopping technique, described in Algorithm 8, in preventing the semiconvergence behaviour typical of the original DIP approach. The collection $\mathcal{C}$ needed in Algorithm 8 has been built starting from both the NASA and the BSDS500 datasets considered in the previous numerical experiments. In particular 8 images from the NASA dataset are taken into account, corrupted by Poisson noise, and 40 images from the BSDS500 dataset, each one perturbed by three different noise levels, i.e. $\gamma = 1, 5, 10$. For each of these images, Algorithm 1 has been applied with $N = 10000$ and, every 50 iterations, the PSNR value of the current reconstruction and the corresponding features vector $F^{(f)}$ have been stored. By means of this phase the dataset $\{F^{(f)}, \text{PSNR}(I_f)\}_{f=1,\dots,|\mathcal{C}|}$ required to train cBRISQUE has been created. The reason why the just described $\mathcal{C}$ collection is considered is that there are already available checkpoints of the images, obtained from the previous numerical experiments by running Algorithm 1 on the 8 NASA images and the 120 BSDS500 images. However, the collection $\mathcal{C}$ can be created on other available pairs of corrupted images and related ground truths. The trained customized BRISQUE metrics can now be exploited to properly stop Algorithm 1 when used to reconstruct any other image. In particular, the trained customized BRISQUE metrics (cBRISQUE$_i$, $i = 1, 2, 3$) will be made available upon suitable request to the authors.

Number of iterations	Average PSNR	PSNR standard deviation
1000	27.04	3.78
1800	28.18	4.27
3000	28.28	5.25
5000	26.92	6.28
7000	25.50	5.74
10000	24.82	5.21

Table 6.5: Average PSNR and standard deviation achieved by Algorithm 1 on the test images employed to evaluate Algorithm 8.

For the last phase of Algorithm 8, the dataset $\mathcal{I}$ consits of the images of the NASA and BSDS500 datasets excluded from the creation of $\mathcal{C}$ and corrupted by Poisson noise, and Cauchy noise with $\gamma = 1, 5, 10$, respectively. Before presenting the results obtained with Algorithm 8, the results achieved using the original DIP scheme (Algorithm 1) are shown, both without early stopping and with a naive early stopping based on the standard BRISQUE metric.

Table 6.5 reports the averaged value of the PSNR obtained by Algorithm 1 on the test dataset $\mathcal{I}$ at different number of iterations. The semiconvergence effect of DIP is clearly highlighted by the results shown in Table 6.5. A good performance of Algorithm 1 can be achieved between 1800 and 3000 iterations. Of course this evaluation can be only deduced a posteriori, once the PSNR has been computed with respect to the ground truths.

Table 6.6 reports the results obtained by Algorithm 1 stopped by means of a criterion based on the original BRISQUE. More in detail, Algorithm 1 has been stopped when the BRISQUE value of the approximated images does not improve for τ successive iterations.

Based on the results presented in Table 6.6, it is possible to conclude that an early stopping procedure based on the original BRISQUE metric is not effective. Indeed, regardless the value of the patience, Algorithm 1 is stopped too early, without approaching the optimal PSNR window.

Table 6.7 shows the results obtained by applying Algorithm 8 with different values of the patience τ. The average PSNR values does not change significantly with τ, by showing a very robust behaviour of Algorithm 8 with respect to the patience. Moreover, it is possible to appreciate that Algorithm 8 allows to stop the DIP procedure in a number of iterations very close to the optimal one suggested by the results of Table 6.5. Indeed, the average PSNR values provided by Algorithm 8 are in line with the optimal ones reported in Table 6.5.

τ	$\bar{N}$	Average PSNR	PSNR standard deviation
5	850	24.91	3.83
10	1650	25.88	3.45
15	2550	26.24	3.76
20	2900	25.34	6.00

Table 6.6: Results achieved by Algorithm 1 stopped by means of a criterion based on the original BRISQUE approach on the test set employed to evaluate Algorithm 8. τ is the value of the patience and $\bar{N}$ is the average number of iterations.

τ	$\bar{N}$	Average PSNR	PSNR standard deviation
5	1600	27.51	4.15
10	3150	27.52	4.85
15	3350	27.29	4.75
20	3500	27.62	4.79

Table 6.7: Results achieved by Algorithm 8 on the test set. τ is the value of the patience and $\bar{N}$ is the average number of iterations.

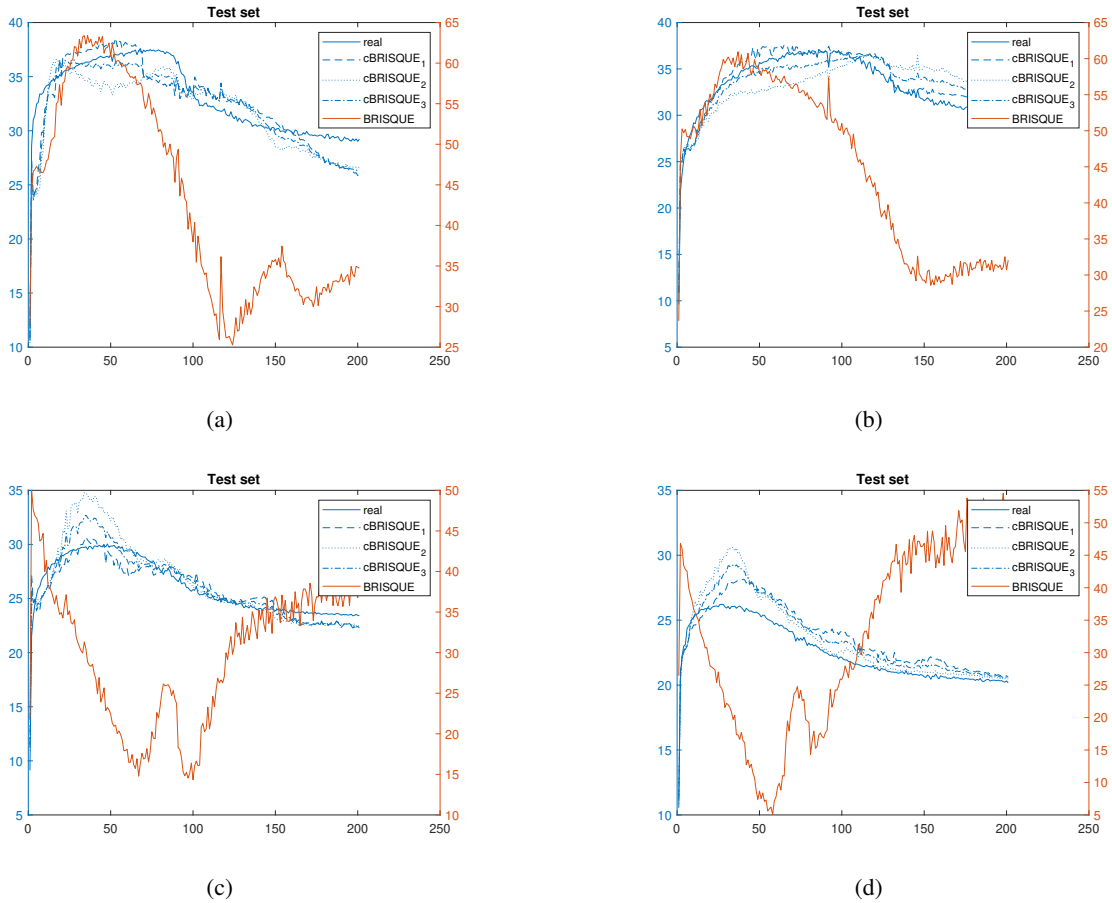
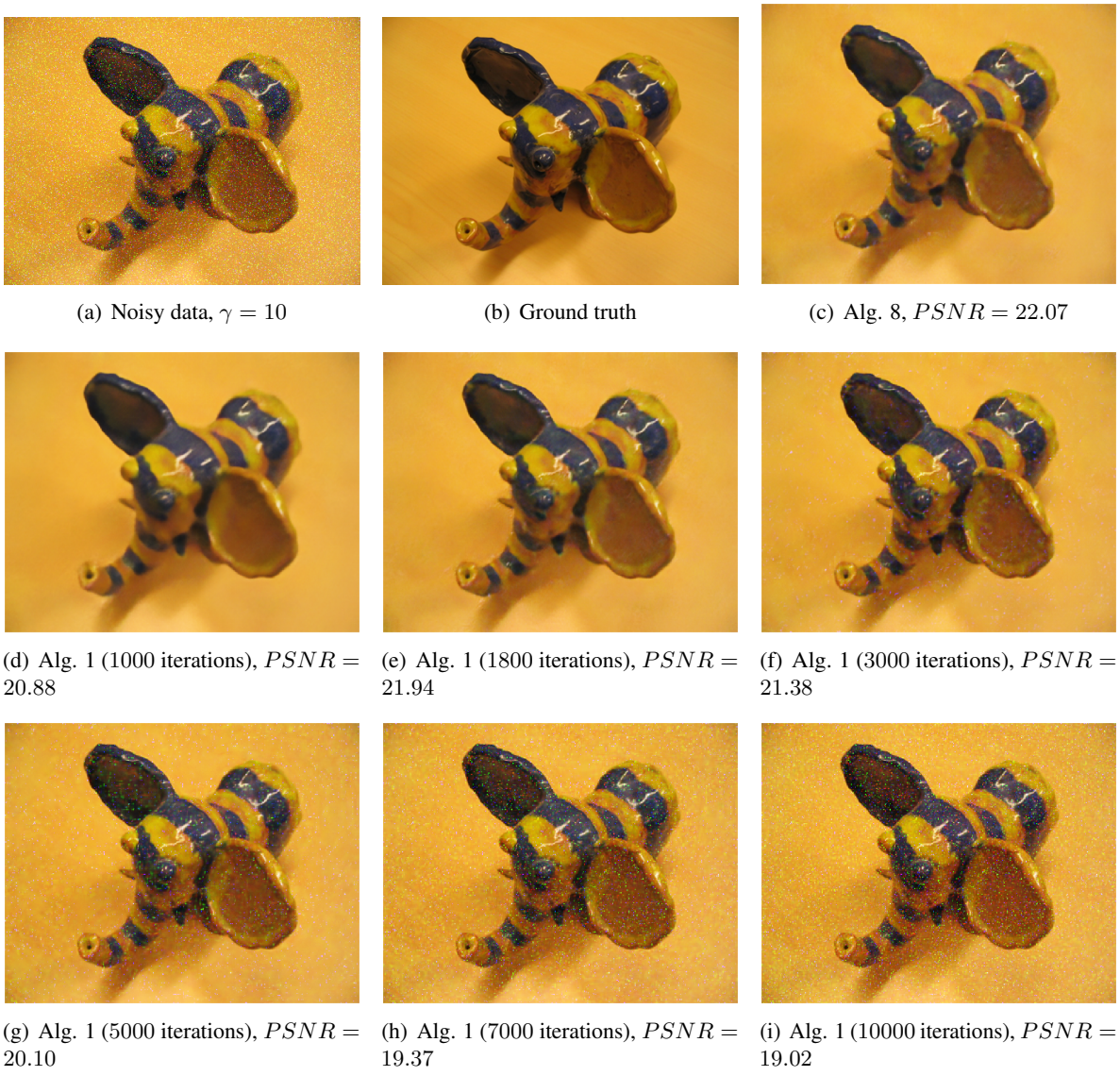


Figure 6.7: PSNR curves predicted by cBRISQUE_1, cBRISQUE_2 and cBRISQUE_3 compared to the real PSNR and the BRISQUE values on four different images of the test dataset.

Figure 6.7 reports the PSNR curves predicted by cBRISQUE_1, cBRISQUE_2 and cBRISQUE_3 when running the DIP procedure on four different images of the test dataset $\mathcal{I}$ corrupted by different type and level of noise. Both real PSNR and BRISQUE values corresponding to the different iterates have been also shown. All the three customized PSNR predictors follow a trend very similar to that of the original PSNR, differently from the BRISQUE metric.

Finally Figure 6.8 presents the reconstruction of one of the corrupted images of the dataset $\mathcal{I}$ obtained by applying both Algorithm 1 with varying number of iterations and Algorithm 8. In terms of PSNR, the best approximation has been provided by Algorithm 8 which performed 1900 iterations. A very similar image has been provided by Algorithm 1 in 1800 iterations. This example shows a very promising behaviour of Algorithm 8: indeed the DIP approach has been optimally stopped without knowing the ground truth.



(a) Noisy data, $\gamma = 10$ (b) Ground truth (c) Alg. 8, $PSNR = 22.07$

(d) Alg. 1 (1000 iterations), $PSNR = 20.88$ (e) Alg. 1 (1800 iterations), $PSNR = 21.94$ (f) Alg. 1 (3000 iterations), $PSNR = 21.38$

(g) Alg. 1 (5000 iterations), $PSNR = 20.10$ (h) Alg. 1 (7000 iterations), $PSNR = 19.37$ (i) Alg. 1 (10000 iterations), $PSNR = 19.02$

Figure 6.8: Results achieved by Algorithm 1 (at different iterations) and Algorithm 8 on a test image corrupted by Cauchy noise with $\gamma = 10$. The first row illustrates the input noisy data, the ground truth and the reconstruction reached by Algorithm 8 (1900 iterations). Images from 6.8(d) to 6.8(i) report the DIP reconstructions. The corresponding PSNR values are also reported.

CHAPTER 7

Conclusions

The results underscore the versatility of the Deep Image Prior (DIP) approach, presenting promising solutions for practical applications across various scientific disciplines and a wide range of problem types. Notably, this technique does not require pre-trained models or extensive training on large datasets, making it a highly adaptable and efficient method.

A novel procedure, DDIPP, has been introduced as a Blind Deconvolution method for recovering microscopy images affected by Poisson noise. This approach leverages a PGDA-inspired algorithm and employs a dual Deep Image Prior strategy, simultaneously applied to the image and the Point Spread Function (PSF). The loss function incorporates the Kullback-Leibler divergence to address Poisson noise, augmented by two adaptive regularization terms: a squared ℓ_2 norm for the PSF and either a Total Variation or an ℓ_1 norm for the image. Furthermore, technical details about the estimated peak of the blur kernel are incorporated to prevent trivial or degenerate solutions.

Numerical experiments conducted on both synthetic and real images have demonstrated encouraging outcomes. The results showed significant improvements in contrast and successful recovery of objects of interest, such as spherical beads in the tested cases. Future research directions include enhancing the model by integrating additional features of the image formation system, such as flux preservation. Another avenue for development involves investigating alternative, more efficient network architectures and identifying improved early stopping criteria. A crucial objective is to optimize the implementation of the code to enable the processing of 3D images, facilitating the estimation of realistic PSFs in three dimensions.

The challenging task of image segmentation has been effectively addressed by coupling the DIP approach with appropriate variational functionals, such as the Mumford-Shah and Ambrosio-Tortorelli formulations. In this framework, the input image is modeled as a piecewise constant or piecewise smooth function, enabling the detection of edges corresponding to gradient discontinuities. This thesis demonstrates empirically that the proposed strategy exhibits robust performance across various noise types. The intrinsic regularization behavior of DIP allows for consistent parameter settings regardless of the noise type, noise level, or the specific image under consideration.

The method has also been extended to the Ambrosio-Tortorelli formulation, employing two separate networks: one for the recovered image and another for the auxiliary variable. As with the piecewise constant case, empirical results confirm the robustness of this approach across different noise levels and image types. Comparisons with state-of-the-art methods further validate these findings. Future work aims to explore the theoretical foundations of this approach. While the numerical results are promising and the outcomes remarkable, deeper insights into the theoretical underpinnings could provide guidance on optimizing network architectures and functional parameters.

Despite the effectiveness of DIP in image reconstruction, the phenomenon of semi-convergence often arises. This occurs when the reconstruction begins to incorporate noise, reducing the Peak Signal-to-Noise Ratio (PSNR) and overall image quality. To address this, the use of early stopping techniques is essential, yielding improved outcomes. The final chapter highlights the importance of selecting optimal hyperparameter values, including the number of iterations, to train neural networks efficiently. Although this usually requires computationally expensive numerical exploration, two proposed early stopping methods for the DIP framework in image denoising tasks have proven effective.

The first method, based on a Neural Architecture Search (NAS) strategy with performance prediction, demonstrates how to fine-tune the network architecture (e.g., number of layers, filters), the training algorithm, and the loss function's regularization parameter under an early stopping framework.

The second approach, employing a customized BRISQUE metric, shows how to halt training of the vanilla DIP network (i.e., the original U-Net architecture proposed in the seminal work) at an optimal point. Notably, this method is adaptable to other network architectures beyond U-Net.

Future research could explore the application of the DIP framework to alternative strategies, such as positional encoding inspired by [123], to facilitate more comprehensive comparisons. Since the DIP framework exhibits inherent regularization properties, future studies might investigate the effects of including or excluding various terms in the regularization component of the functional. Additionally, segmentation methods could be coupled with appropriate early stopping strategies to enhance performance. Finally, the application of this approach to other noise types could be studied to extend its utility in recovering images degraded by different noise profiles.

Bibliography

[1] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Cambridge, MA, USA, 2017.

[2] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[3] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[4] Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural Networks*, 138:14–32, 2021.

[5] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503:92–108, 2022.

[6] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

[7] Alejandro Baldominos, Yago Saez, and Pedro Isasi. On the automated, evolutionary design of neural networks: past, present, and future. *Neural computing and applications*, 32:519–545, 2020.

[8] Yuzhu Ji, Haijun Zhang, Zhao Zhang, and Ming Liu. Cnn-based encoder-decoder networks for salient object detection: A comprehensive review and recent advances. *Information Sciences*, 546:835–857, 2021.

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[10] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. *Advances in neural information processing systems*, 31, 2018.

[11] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[12] Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[13] L. Bottou, F.E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Rev.*, 60(2):223–311, 2018.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[16] Ulyanov Dmitry, Andrea Vedaldi, and Lempitsky Victor. Deep image prior. *International Journal of Computer Vision*, 128(7):1867–1888, 2020.

[17] N Renuka. Semantic segmentation-based skin cancer detection. *Soft Computing*, 27(16):11895–11903, 2023.

[18] Said Charfi, Mohamed El Ansari, Lahcen Koutti, Ayoub Ellahyani, and Ilyas Eljaafari. Modified residual attention network for abnormalities segmentation and detection in WCE images. *Soft Computing*, pages 1–14, 2024.

[19] Dmitrii Torbunov, Yi Huang, Haiwang Yu, Jin Huang, Shinjae Yoo, Meifeng Lin, Brett Viren, and Yihui Ren. Uvcgan: Unet vision transformer cycle-consistent gan for unpaired image-to-image translation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 702–712, January 2023.

[20] Alessandro Benfenati, Davide Bolzi, Paola Causin, and Roberto Oberti. A deep learning generative model approach for image synthesis of plant leaves. *Plos one*, 17(11):e0276972, 2022.

[21] Kevin Trebing, Tomasz Stańczyk, and Siamak Mehrkanoon. SmaAt-UNet: Precipitation nowcasting using a small attention-unet architecture. *Pattern Recognition Letters*, 145:178–186, 2021.

[22] Ruoyu Sun, Dawei Li, Shiyu Liang, Tian Ding, and Rayadurgam Srikant. The global landscape of neural networks: An overview. *IEEE Signal Processing Magazine*, 37(5):95–108, 2020.

[23] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[24] Daniel Otero Baguer, Johannes Leuschner, and Maximilian Schmidt. Computed tomography reconstruction using deep image prior and learned reconstruction methods. *Inverse Problems*, 36(9):094004, 2020.

[25] Jiaming Liu, Yu Sun, Xiaojian Xu, and Ulugbek S. Kamilov. Image restoration using total variation regularized deep image prior. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7715–7719, 2019.

[26] Dave Van Veen, Ajil Jalal, Mahdi Soltanolkotabi, Eric Price, Sriram Vishwanath, and Alexandros G. Dimakis. Compressed sensing with deep image prior and learned regularization, 2020.

[27] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[28] Marlio Paredes, Jorge Villamizar Morales, Lola Xiomara Bautista Rozo, and Domingo Rodriguez. The structure of the computational signal algebra and its application in digital image processing. *Dyna*, 78(166):118–132, 2011.

[29] M. Bertero, P. Boccacci, and V. Ruggiero. *Inverse Imaging with Poisson Data*. IOP Publish., 2018.

[30] P. Campisi and K. Egiazarian, editors. *Blind Image Deconvolution: Theory and Applications*. CRC Press, 2016.

[31] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithm. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 1964–1971, 2009.

[32] S Derin Babacan, Rafael Molina, Minh N Do, and Aggelos K Katsaggelos. Bayesian blind deconvolution with general sparse image priors. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*, pages 341–355. Springer, 2012.

[33] R JL Fétick, LM Mugnier, Thierry Fusco, and Benoit Neichel. Blind deconvolution in astronomy with adaptive optics: the parametric marginal approach. *Monthly Notices of the Royal Astronomical Society*, 496(4):4209–4220, 2020.

[34] Jan Kotera, Václav Šmídl, and Filip Šroubek. Blind deconvolution with model discrepancies. *IEEE transactions on image processing*, 26(5):2533–2544, 2017.

[35] Mario Bertero, Patrizia Boccacci, and Valeria Ruggiero. *Inverse imaging with Poisson data: from cells to galaxies*. IOP Publishing, 2018.

[36] Riccardo Zanella, Patrizia Boccacci, Luca Zanni, and Mario Bertero. Efficient gradient projection methods for edge-preserving removal of Poisson noise. *Inverse Problems*, 25(4):045010, 2009.

[37] G. M. P. van Kempen, L. J. van Vliet, P. J. Verveer, and H. T. M. van der Voort. A quantitative comparison of image restoration methods for confocal microscopy. *J. Microsc.*, 185:354–365, 1997.

[38] G. M. P. van Kempen and van Vliet. Background estimation in nonlinear image restoration. *J. Opt. Soc. Am. A*, 17:425–433, 2000.

[39] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

[40] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, 26:127–136, 2000.

[41] S. Bonettini. Inexact block coordinate descent methods with application to the nonnegative matrix factorization. *IMA J. of Num. Analysis*, 31:1431–1452, 2011.

[42] M. Prato, A. La Camera, S. Bonettini, and M. Bertero. A convergent blind deconvolution method for post-adaptive-optics astronomical imaging. *Inverse Problems*, 29:065017 (22pp), 2013.

[43] W Zuo, D. Ren, D. Zhang, S. Gu, and L. Zhang. Learning iteration-wise generalized shrinkage–thresholding operators for blind deconvolution. *IEEE Transactions on Image Processing*, 25(4):1751–1764, 2016.

[44] J. Pan, D. Sun, H. Pfister, and M. H. Yang. Deblurring images via dark channel prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(10):2315–2328, 2018.

[45] M. Zhang, Y. Fang, G. Ni, and T. Zeng. Pixel screening based intermediate correction for blind deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5892–5900, June 2022.

[46] J. Liu, M. Yan, and T. Zeng. Surface-aware blind image deblurring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(3):1041–1055, 2021.

[47] S. Nah, T. Hyun Kim, and K. Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 257–265, 2017.

[48] X. Tao, H. Gao, X. Shen, J. Wang, and J. Jia. Scale-recurrent network for deep image deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 8174–8182, 2018.

[49] J. Zhang, J. Pan, J. Ren, Y. Song, L. Bao, R. W. H. Lau, and M. Yang. Dynamic scene deblurring using spatially variant recurrent neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 2521–2529, 2018.

[50] H. Gao, X. Tao, X. Shen, and J. Jia. Dynamic scene deblurring with parameter selective sharing and nested skip connections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, page 3843–3851, 2019.

[51] J. Koh, J. Lee, and S. Yoon. Single-image deblurring with neural networks: A comparative survey. *Computer Vision and Image Understanding*, 203:103134, 2021.

[52] P. Tran, A. Tran, Q. Phung, and M. Hoai. Explore image deblurring via encoded blur kernel space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[53] M. Asim, F. Shamshad, and A. Ahmed. Blind image deconvolution using deep generative priors. *IEEE Transactions on Computational Imaging*, 6:1493–1506, 2020.

[54] D. Ren, K. Zhang, Q. Wang, Q. Hu, and W. Zuo. Neural blind deconvolution using deep priors. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 3338–3347, 2020.

[55] Y. Gandelsman, A. Shocher, and M. Irani. "double-DIP": Unsupervised image decomposition via coupled deepimage- priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[56] Z. Wang, Z. Wang, Q. Li, and H Bilen. Image deconvolution with deep image and kernel priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[57] J. Kotera, F. Šroubek, and W. Šmídl. Improving neural blind deconvolution. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1954–1958, 2021.

[58] Z. Zhuang, T. Li, H. Wang, and J. Sun. Blind image deblurring with unknown kernel size and substantial noise, 2022.

[59] P. Cascarano, G. Franchini, E. Kobler, F. Porta, and A. Sebastiani. Constrained and unconstrained deep image prior optimization models with automatic regularization. *Computational Optimization and Applications*, pages 1–25, 2022.

[60] Radu Ioan Boţ and Axel Böhm. Alternating proximal-gradient steps for (stochastic) nonconvex-concave minimax problems. *SIAM Journal on Optimization*, 33(3):1884–1913, 2023.

[61] Z. Chen, Y. Zhou, T. Xu, and Y. Liang. Proximal gradient descent-ascent: Variable convergence under kl geometry. *arXiv preprint arXiv:2102.04653*, 2021.

[62] H. H. Bauschke and P. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer Science & Business Media, 2011.

[63] P.M. Pardalos and N. Kovoor. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, 46:321–328, 1990.

[64] Y. H. Dai and R. Fletcher. New Algorithms for Singly Linearly Constrained Quadratic Programming Problems Subject to Lower and Upper Bounds. *Mathematical Programming*, 106(3):403–421, 2006.

[65] Krzysztof C Kiwiel. Breakpoint searching algorithms for the continuous quadratic knapsack problem. *Mathematical Programming*, 112:473–491, 2008.

[66] W.T. Chen, A. Y. Zhu, M. Khorasaninejad, Z. Shi, V. Sanjeev, and Capasso F. Immersion Meta-Lenses at Visible Wavelengths for Nanoscale Imaging. *Nano Lett.*, 17(5):3188–3194, 2017.

[67] Y. Ashida, Y. Honma, N. Miura, T. Shibuya, H. Kikuchi, Y. Tamada, Y. Kamei, A. Matsuda, and M. Hattori. Imaging performance of microscopy adaptive-optics system using scene-based wavefront sensing. *J Biomed Opt.*, 25(12):123707, 2020.

[68] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *CoRR*, abs/2006.09661, 2020.

[69] Pasquale Cascarano, Giorgia Franchini, Federica Porta, and Andrea Sebastiani. On the First-Order Optimization Methods in Deep Image Prior. *Journal of Verification, Validation and Uncertainty Quantification*, 7(4), 01 2023. 041002.

[70] G. C Borgia, R.J.S. Brown, and P Fantazzini. Uniform-penalty inversion of multiexponential decay data. *Journal of Magnetic Resonance*, 132(1):65–77, 1998.

[71] M. Grasmair. Locally adaptive total variation regularization. In *Scale Space and Variational Methods in Computer Vision*, number 5567 in Lecture Notes in Computer Science, pages 331–342. Berlin/Heidelberg, Springer, 2009.

[72] V. Bortolotti, R. Brown, P. Fantazzini, G. Landi, and F. Zama. Uniform Penalty inversion of two-dimensional NMR relaxation data. *Inverse Problems*, 33(1):103134, 2016.

[73] H. Wang, T. Li, Z. Zhuang, T. Chen, H. Liang, and J. Sun. Early stopping for deep image prior. *CoRR*, abs/2112.06074, 2021.

[74] R Zanella, P Boccacci, L Zanni, and M Bertero. Efficient gradient projection methods for edge-preserving removal of Poisson noise. *Inverse Problems*, 25(4):045010, 2009.

[75] M Bertero, P Boccacci, G Talenti, R Zanella, and L Zanni. A discrepancy principle for Poisson data. *Inverse Problems*, 26:105004, 2010.

[76] R Zanella, P Boccacci, L Zanni, and M Bertero. Corrigendum: Efficient gradient projection methods for edge-preserving removal of Poisson noise. *Inverse Problems*, 29:119501, 2013.

[77] R. M. Willett and R. D. Nowak. Platelets: A multiscale approach for recovering edges and surfaces in photon limited medical imaging. *IEEE Transactions on Medical Imaging*, 22:332–350, 2003.

[78] A. Benfenati. upU-Net approaches for background emission removal in fluorescence microscopy. *Journal of Imaging*, 8(5), 2022.

[79] H. Kirshner, F. Aguet, D. Sage, and M. Unser. 3-D PSF fitting for fluorescence microscopy: Implementation and localization application. *Journal of Microscopy*, 249(1):13–25, January 2013.

[80] D. Sage, L. Donati, F. Soulez, D. Fortun, G. Schmit, A. Seitz, R. Guiet, C. Vonesch, and M. Unser. DeconvolutionLab2: An open-source software for deconvolution microscopy. *Methods—Image Processing for Biologists*, 115:28–41, February 15, 2017.

[81] D. Sage, T.-A. Pham, H. Babcock, T. Lukes, T. Pengo, J. Chao, R. Velmurugan, A. Herbert, A. Agrawal, S. Colabrese, A. Wheeler, A. Archetti, B. Rieger, R. Ober, G.M. Hagen, J.-B. Sibarita, J. Ries, R. Henriques, M. Unser, and S. Holden. Super-resolution fight club: Assessment of 2D and 3D single-molecule localization microscopy software. *Nature Methods—Techniques for Life Scientists and Chemists*, 16(5):387–395, May 2019.

[82] Jun Ma, Yuting He, Feifei Li, Lin Han, Chenyu You, and Bo Wang. Segment anything in medical images. *Nature Communications*, 15(1):654, 2024.

[83] Di Feng, Christian Haase-Schütz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaeser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360, 2020.

[84] Yinglong Li. Research and application of deep learning in image recognition. In *2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA)*, pages 994–999, 2022.

[85] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018.

[86] S. Afshari, A. BenTaieb, Z. Mirikharaji, and G. Hamarneh. Weakly supervised fully convolutional network for pet lesion segmentation. In *Med. Imag. 2019: Imag. Process.*, page 109491K, 2019.

[87] Chen L.-C. Collins M. Zhu Y. Papandreou G. Zoph B. Schroff F. Adam H. Shlens J. Searching for efficient multi-scale architectures for dense image prediction. *Advances in Neural Information Processing Systems*, pages 8713–8724, 2018.

[88] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3340, 2015.

[89] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(07):3523–3542, 2022.

[90] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 1520–1528, 2015.

[91] Zongyao Li, Ren Togo, Takahiro Ogawa, and Miki Haseyama. Variational autoencoder based unsupervised domain adaptation for semantic segmentation. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2426–2430, 2020.

[92] Siyi Xun, Dengwang Li, Hui Zhu, Min Chen, Jianbo Wang, Jie Li, Meirong Chen, Bing Wu, Hua Zhang, Xiangfei Chai, Zekun Jiang, Yan Zhang, and Pu Huang. Generative adversarial networks in medical image segmentation: A review. *Computers in Biology and Medicine*, 140:105063, 2022.

[93] Boah Kim and Jong Chul Ye. Mumford–Shah loss functional for image segmentation with deep learning. *IEEE Transactions on Image Processing*, 29:1856–1866, 2019.

[94] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. Pure Appl. Math.*, 42:577–685, 1989.

[95] L. Ambrosio and V. M. Tortorelli. Approximation of functional depending on jumps by elliptic functional via t-convergence. *Comm. Pure Appl. Math.*, 43(8):999–1036, 1990.

[96] L. Ambrosio and V. M. Tortorelli. On the approximation of free discontinuity problem by elliptic functionals. *Boll Un Mat Ital B*, 6:105–123, 1992.

[97] E. De Giorgi, M. Carriero, and A. Leaci. Existence theorem for a minimum problem with free discontinuity set. *Arch. Ration. Mech. Anal.*, 108:195–218, 1989.

[98] Antonin Chambolle, Vicent Caselles, Daniel Cremers, Matteo Novaga, and Thomas Pock. An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227, 2010.

[99] M. Foare, N. Pustelnik, and L. Condat. Semi-linearized proximal alternating minimization for a discrete Mumford–Shah model. *IEEE Transactions on Image Processing*, 29:2176–2189, 2019.

[100] A. Auslender. Méthodes numériques pour la décomposition et la minimisation de fonctions non différentiables. *Numerische Mathematik*, 18:213–223, 1971.

[101] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. PrenticeHall, New Jersey, 1989.

[102] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New-York, 1970.

[103] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.*, 109:475–494, 2001.

[104] Xin Zheng, Yong Wang, Guoyou Wang, and Jianguo Liu. Fast and robust segmentation of white blood cell images by self-supervised learning. *Micron*, 107:55–71, 2018.

[105] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabCut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.

[106] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[107] Federica Sciacchitano, Yiqiu Dong, and Tieyong Zeng. Variational approach for restoring blurred images with cauchy noise. *SIAM Journal on Imaging Sciences*, 8(3):1894–1922, 2015.

[108] Jin-Jin Mei, Yiqiu Dong, Ting-Zhu Huang, and Wotao Yin. Cauchy noise removal by nonconvex ADMM with convergence guarantees. *Journal of Scientific Computing*, 74:743–766, 2018.

[109] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[110] Kary Ho, Andrew Gilbert, Hailin Jin, and John Collomosse. Neural architecture search for deep image prior. *Computers & graphics*, 98:188–196, 2021.

[111] Yun-Chun Chen, Chen Gao, Esther Robb, and Jia-Bin Huang. NAS-DIP: Learning deep image prior with neural architecture search. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 442–459. Springer, 2020.

[112] Metin Ersin Arican, Ozgur Kara, Gustav Bredell, and Ender Konukoglu. ISNAN-DIP: Image-specific neural architecture search for deep image prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1960–1968, 2022.

[113] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on image processing*, 21(12):4695–4708, 2012.

[114] Hengkang Wang, Taihui Li, Zhong Zhuang, Tiancong Chen, Hengyue Liang, and Ju Sun. Early stopping for deep image prior. *Transactions on Machine Learning Research*, 2023.

[115] Alessandro Benfenati, Ambra Catozzi, and Valeria Ruggiero. Neural blind deconvolution with Poisson data. *Inverse Problems*, 39, 2023.

[116] Zezhou Cheng, Matheus Gadelha, Subhransu Maji, and Daniel Sheldon. A bayesian perspective on the deep image prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[117] Ibrahim Alshubaily. Efficient neural architecture search with performance prediction, 2021.

[118] Taihui Li, Zhong Zhuang, Hengyue Liang, Le Peng, Hengkang Wang, and Ju Sun. Self-validation: Early stopping for single-instance deep generative priors. In *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021*, page 108. BMVA Press, 2021.

[119] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

[120] Giorgia Franchini, Valeria Ruggiero, Federica Porta, and Luca Zanni. Neural architecture search via standard machine learning methodologies. *Mathematics in Engineering*, 5(1)):1–21, 2023.

[121] Top 100 images esa-hubble. `https://esahubble.org/images/archive/top100/`.

[122] Top 100 hubble telescope images. `https://www.kaggle.com/datasets/redwankarimsony/top-100-hubble-telescope-images`.

[123] Nimrod Shabtay, Eli Schwartz, and Raja Giryes. Pip: Positional-encoding image prior. *arXiv preprint arXiv:2211.14298*, 2022.