UNIVERSITÀ DI PARMA

UNIVERSITÁ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN MATEMATICA

CICLO XXXVI

Learning Variational Models via Bilevel Optimization and Unfolded Algorithms

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

Relatore:
Chiar.ma Prof.ssa **Silvia Bonettini**

Correlatori:
Chiar.mo Prof. **Marco Prato**
Dott.ssa **Giorgia Franchini**

Dottorando: **Danilo Pezzi**

Anni Accademici 2020/2021 - 2022/2023

Abstract

Variational models are a classical tool to solve inverse problems in a multitude of contexts. In contrast to other modern methodologies, they lay their foundations on an established theoretical background. The advantage offered by their interpretability, combined with a less intense requirement in terms of computational resources makes the study and use of them still relevant to this day. To improve the performance of these models, we address one of their main weaknesses: parameters selection. These models are defined through an energy function which itself is characterized by one or more variables. The absence of broad rules to set these variables often leads to a tedious, and perhaps time consuming, empirical search for a good configuration. Borrowing ideas from the machine learning realm to automate this process, we effectively train the energy functional with the goal of finding a more than satisfying parameter setup. On one hand, we study a general formulation of a bilevel optimization problem to carry out this task in a variety of imaging applications. In detail, we develop an iterative general purpose inexact forward-backward algorithm which is able to converge to a stationary point of the bilevel problem. The experiments show that the algorithm is able to find more than valid solutions in a small amount of time. Although these results are promising, there are still limitations to the approach. To circumvent some of these, we also relax the lower level minimization problem by replacing it with the unfolding of an iterative optimization algorithm. The number of iterations that the chosen scheme performs is fixed a priori, so that the whole lower level resembles a very basic neural network. The training of this algorithm unfolding then aims at finding the best parameters (including some of the optimization algorithm itself!) to make the most out of the given iterations. We tested these ideas on different imaging problems, with surprising results, especially when compared to deep learning methods.

I modelli variazionali rappresentano uno degli strumenti classici per risolvere problemi inversi lineari in vari contesti. In contrasto con altri metodologie più moderne, essi poggiano basi teoriche che sono state studiate per decenni. La maggior interpretabili che offrono, combinata con una minor richiesta in termini di risorse computazionali fa si che lo studi di questi metodi sia ancora rilevante al giorno d'oggi. In questo lavoro, per migliorare la performance di questi modelli, ci soffermiamo sui metodi di

selezione dei loro parametri. È ben noto che i funzionali che costituiscono i modelli variazionali dipendono da uno o più parametri di cui però non si hanno regole precise su come sceglierli in modo efficace. Prendendo spunto da tecniche nell'ambito dell'apprendimento automatico, andiamo ad imparare i modelli variazioni con lo scopo di trovare in modo automatico una configurazione soddisfacente di parametri. In questo lavoro proponiamo un algoritmo di ottimizzazione di tipo forward-backward, di cui dimostriamo la convergenza ad un punto stazionario di un problema di ottimizzazione bilivello per l'apprendimento di un modello variazionale. Gli esperimenti numerici mostrano che l'algoritmo riesce a raggiungere una buona soluzione in poco tempo. Per aggirare alcuni limiti di questo approccio, modifichiamo il problema inferiore inserendo, invece che una minimizzazione completa, l'unfolding di un metodo di ottimizzazione iterativo. Il numero di iterazioni che compie tale metodo è fissato a priori. L'apprendimento di questo algoritmo "srotolato" mira a trovare la migliore configurazione di parametri (anche quelli dell'algoritmo stesso!) per sfruttare al meglio le iterazioni concesse. Abbiamo testato queste idee su diversi problemi di ricostruzione di immagini, con buoni risultati, soprattutto in confronto con altri metodi di tipo deep.

Keywords: Linear Inverse Problems, Variational Models, Bilevel Optimization, Algorithm Unfolding, GreenAI

Notation

- All vectors and matrices are denoted in **bold**. Matrices are upper case while vectors are in lower case (all exceptions are specified when necessary).
- If $\mathbf{A}$ is a matrix, then $\mathbf{A_i}$ is its i-th column and A_{ij} is the element in the i-th row and j-th column.
- If $\mathbf{a}$ is a vector, then a_i it its i-th component.
- Unless specified, we will always consider images in vector form, where the vector is formed by stacking the columns of the image.
- The matrix $\mathbf{A}_k$ is the k-th element of a sequence of matrices.
- $\mathbf{I_n}$ denotes the identity matrix of size $n \times n$.
- $\mathbf{1}_m$ and $\mathbf{0}_m$ denote the column vectors of length m made by all ones or zeros, respectively.
- $\mathcal{M}_\nu$ denotes the space of symmetric positive definite matrices with eigenvalues in $[\frac{1}{\nu}; \nu]$.
- $\|\mathbf{a}\|_\mathbf{A}$ is the norm induced by a symmetric and positive definite matrix $\mathbf{A}$, i.e., $\|\mathbf{a}\|_\mathbf{A}^2 = \mathbf{a}^T\mathbf{A}\mathbf{a}$.
- $\bar{\mathbb{R}}$ denotes the extended real line, i.e., $\bar{\mathbb{R}} = \mathbb{R} \cup \infty$
- $\mathbb{R}_+$ denotes the non-negative real numbers, while $\mathbb{R}_{++}$ denotes the positive real numbers.
- Given a function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$, f^* denotes its convex/Fenchel conjugate.
- Given a function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$, ∂f denotes the subdifferential operator associated to it.
- In the experiments, $\mathcal{D}$ denotes the dataset, where each sample always contains at least the corrupted image $\mathbf{y}_s$, whose corresponding ground truth to it, would be $\mathbf{x}_s$.

- The expression $\mathbf{a} * \mathbf{u}$ represents the convolution between the kernel $\mathbf{a}$ and the image $\mathbf{u}$. It will also be expressed as $\mathbf{Au}$ where $\mathbf{A}$ is the corresponding convolutional matrix.
- $\operatorname{dom} f$ denotes the *effective domain* of f, i.e., $\operatorname{dom} f := \{\mathbf{u} \colon f(\mathbf{u}) < +\infty\}$

Contents

Introduction

In many scientific disciplines, from archaeology to medicine, from physics to engineering, it is possible to encounter objects or phenomena that cannot be directly observable, but one is forced to study them only through the results that they produce in the environment. This methodology does, for instance, allow scientists to detect the presence of black holes in space. Indeed, black holes are not visible by conventional telescopes as they do not emit any light, but they heavily (pun intended) affect the matter surrounding them, and by studying these effects we can infer their position. Another space related example is the proof of heliocentrism by Galileo. The italian scientist, in the 17th century, observed that Venus exhibited different phases, like our Moon does, and those are explicable only by accepting that Venus is orbiting around the Sun and not the Earth. These problem of finding the cause by studying its effects is mathematically called an inverse problems. They are present also in everyday lives in many different shapes. In particular, one big branch is *Computational Imaging*, which develops techniques to form digital images through algorithms a significant computations. Computational imaging researcher have allowed us to be able to scan our bodies with an MRI, or to take sharp pictures with our phone cameras.

Although computational imaging is a field of research that has gathered its own attention only in the past three decades, one of its classical tools, the so called *Variational Models* have been around for quite some more time. As means to solve inverse problems, they are obtained through a Bayesian analysis of the measurement process. The winning traits of this procedure lie in the ability of incorporating into the model specific information of the object in question. This is done in two simultaneous ways. On one hand, the acquisition of the data is modelled according to the physical law of the measurement process. On the other, by properly defining the variational model, we have the ability to enforce properties that we know are true for the images of interest, that would otherwise be difficult to obtain.

Variational models have been around for so long, that of course other technologies have been developed, that yields the same, and in many cases better, results. We are talking about *Machine Learning* (ML) and *Deep Learning* (DL) methodologies, placed under the Artificial Intelligence label by the general public. Their fundamental strategy is to fit a model determined by thousands, if not millions, of parameters

using a pre-obtained dataset of samples. The results they bring are already outstanding, but they also have unavoidable flaws. This fixation on data also leads to a certain degree of overfitting, which means that there is always a non-zero chance that the ML/DL model produces, on unseen data, a prediction that is faulty, and possibly we may not even be able to detect these errors. This means that, no matter how much one tries to make the architecture as general as possible, they are permeated by instability, which cannot be predicted due to their black-box nature.
Variational models, with their solid and established theoretical background, may still be a better fit for some application. Whether it be because the human operator has better tools to understand what is incorrect in the image, or simply because they require a lot less computational resources, and we know how energy has been a quite central topic in the past decades, we believe that variational models are still worth studying.
One of the weaknesses of variational models is that they also depends on some hyperparameters, for which we do not really have broad and always valid strategies on how to select them. Hyperparameter selection represents also a big open problem in the ML and DL communities. This manuscript focuses on the recent trend of borrowing ideas from ML and DL and use them to improve the performance of the variational models. Since the number of their parameters is way lower than in ML or DL, the resulting training operation is far less expensive, but can still lead to a clear improvement over empirical techniques for hyperparameters selection.
The first chapter of this thesis is dedicated to describing the fundamental ideas behind variational models, including some examples of some of the most notable ones. Chapter 2 focuses on *Bilevel Optimization*. These kinds of problems were first introduced in 1952 by Stackelberg in its book "The Theory of the Market Economy" [89], and since then have come up in a variety of contexts. In our work, they are a natural tool to carry out the training of the variational model. Due to their complex and intricate structure, the mathematical analysis of bilevel optimization problems is seldom easy. In particular, we propose an algorithm that is able to solve a not insignificant class of bilevel optimization problems that arise in imaging applications. Chapter 3 studies the applications of algorithm unfolding as a mean to relax the issues of bilevel optimizations. In particular, we tackle two different inverse problems, a blind-deconvolution of text images and a super-resolution combined with deconvolution of microscopy images. Finally, the last chapter investigates the use of stochastic schemes to train variational models. These optimization algorithms are widely used in ML and DL, because it is simply impossible to process all the sheer amount of data that they involve. However, the lesser requirements of the proposed approaches in terms of number of data samples questions whether or not these schemes actually bring any benefit.

Overall this thesis presents different interesting contributions, both theoretical and practical, in the aforementioned topics.

- We propose a Forward-Backward algorithm to solve a bilevel optimization problem and prove its convergence to a stationary point.
- We develop a linesearch procedure to estimate the length of the step in a descent direction, but in the case where the objective function, its gradient and the proximal operator of its non-differentiable terms are not exact, but computable only up to a certain tolerance.
- We study an explainable framework for the learning of parameters in variational models and apply the relative techniques in two different scenarios: a blind deconvoltion challenge and super-resolution problems in the microscopy field.

Chapter 1

Variational Models: Motivations and Fundamentals

In this work, we always consider at the base a *linear inverse problem* arising from an imaging application. Let $\mathbf{y} \in \mathbb{R}^m$ be the raw captured image, i.e.,

$$\mathbf{y} = N(\mathbf{A}\mathbf{u}^{\text{true}} + \mathbf{bg}) \tag{1.1}$$

where the linear operator $\mathbf{A}\colon \mathbb{R}^n \to \mathbb{R}^m$ has the role of modeling both the observation process and the degradation that the acquisition instruments applies onto the image, $N(\cdot)$ represents the inevitable noise that occurs because of the imperfect measurements, and $\mathbf{bg}$ is the background distortion that can be seen in certain contexts. In a perfect world we would be able to recover the real image $\mathbf{u}^{\text{true}}$, but unfortunately here we have no hope of doing that: by nature, the noise is an aleatory phenomenon and there is no way of knowing exactly the realization, which means, in other words, that some information is lost.

Finding a high quality approximation is not as straightforward as it might seem. Direct inversion of the operator is a strategy almost never feasible. Consider the linear inverse problem $\mathbf{y} = \mathbf{A}\mathbf{u}^{\text{true}}$, where we assume to have exact data. The first issue would concern the well-posedness in the Hadamard sense [95]. It requires the existence and the uniqueness of the solution, as well as its continuity with respect to the data. Even in the case where these properties are satisfied, we have to remember that the data is not actually exact. A noisy image could be outside of the range of $\mathbf{A}$ or could still lead to a poor reconstruction due to a ill-conditioned operator. Let $\mathbf{v} = \mathbf{A}\mathbf{u}$ and $\mathbf{v}_\delta = \mathbf{A}\mathbf{u}_\delta$ be two vectors obtained by applying the operator $\mathbf{A}$ to vectors such that $\|\mathbf{u}_\delta - \mathbf{u}\|_2 = \delta$, then

$$\frac{\|\mathbf{v}_\delta\|_2}{\|\mathbf{v}\|_2} = \frac{\|\mathbf{A}\mathbf{u}_\delta\|_2}{\|\mathbf{A}\mathbf{u}\|_2} \leq \|\mathbf{A}^{-1}\|\|\mathbf{A}\|\frac{\|\mathbf{u}_\delta\|_2}{\|\mathbf{u}\|_2}.$$

If the condition number $\kappa(\mathbf{A}) = \|\mathbf{A}^{-1}\|\|\mathbf{A}\|$ is high, then for small changes δ in the input $\mathbf{u}$ one could possibly have striking differences in the outputs $\mathbf{v}_\delta$ and $\mathbf{v}$. All of this contributes to a non-negligible instability of the solution.
Regularization refers to a broad variety of procedures with the aim of resolving the issues of ill-posedness and ill-conditioning (as well as overfitting in machine learning contexts). The most natural way is to add a regularization term through a penalty or via constraints, making the solution unique, and, at the same time, enforcing desirable properties on the reconstruction. Unfortunately, as will become even more clear in the following sections, there is no general fit-all regularization procedure, and there is always the need to find the correct model for the situation of interest and the specific type of image.

1.1 Bayesian Formulation

In most images, pixels are not independent entities but often have some form of correlation with some, or all, of their neighbours. So, whether in the spatial domain or in some transformation space, we can deduce statistical properties on the images [27, 54, 64]. This justifies the use of a bayesian framework, where both the data $\mathbf{y}$ and the unknown ground truth $\mathbf{u}$ are considered as realizations of the random variables $\mathcal{Y}$ and $\mathcal{U}$ respectively, to obtain a possible reconstruction. Consider the probability that the original image is $\mathbf{u}$ given the observed sample $\mathbf{y}$, $\mathbb{P}(\mathcal{U} = \mathbf{u} \mid \mathcal{Y} = \mathbf{y})$. A *Maximum A Posteriori* (MAP) estimate is the solution $\mathbf{u}^*$ to the optimization problem

$$\bar{\mathbf{u}} = \underset{\mathbf{u}\in\mathbb{R}^n}{\operatorname{argmax}}\, \mathbb{P}(\mathcal{U} = \mathbf{u} \mid \mathcal{Y} = \mathbf{y}). \tag{1.2}$$

Unfortunately, basically nothing useful is known on this probability. The Bayes' theorem for the conditional probability can, however, shed some light on the situation:

$$\mathbb{P}(\mathcal{U} = \mathbf{u} \mid \mathcal{Y} = \mathbf{y}) = \frac{\mathbb{P}(\mathcal{U} = \mathbf{u} \wedge \mathcal{Y} = \mathbf{y})}{\mathbb{P}(\mathcal{Y} = \mathbf{y})} \cdot \frac{\mathbb{P}(\mathcal{U} = \mathbf{u})}{\mathbb{P}(\mathcal{U} = \mathbf{u})} = \frac{\mathbb{P}(\mathcal{Y} = \mathbf{y} \mid \mathcal{U} = \mathbf{u})\mathbb{P}(\mathcal{U} = \mathbf{u})}{C_{\mathbf{y}}}. \tag{1.3}$$

Here we have used the definition of conditional probability in the first equality and we note that the probability $\mathbb{P}(\mathcal{Y} = \mathbf{y})$ is in fact a constant for our problem. Overall, by taking the negative logarithm in this last equation, we can rewrite problem (1.2) as a composite minimisation problem:

$$\bar{\mathbf{u}} = \underset{\mathbf{u}\in\mathbb{R}^n}{\operatorname{argmin}}\; -\log(\mathbb{P}(\mathcal{Y} = \mathbf{y} \mid \mathcal{U} = \mathbf{u})) - \log(\mathbb{P}(\mathcal{U} = \mathbf{u})) \tag{1.4}$$

This reformulation is now something that can be dealt with.
As was said before, images are endowed with statistics. Thus, finding a probability $\mathbb{P}(\mathcal{U} = \mathbf{u})$, also called *prior distribution* (or simply prior), that can be a reasonable

model fit for the specific application one is interested in, is less complicated than it appears. The Gibbs distribution is the most widely employed prior:

$$\mathbb{P}(\mathcal{U} = \mathbf{u}) = \frac{1}{Z} e^{-\rho R(\mathbf{u})}, \tag{1.5}$$

where $Z = \int_{\mathbb{R}^n} e^{-\rho R(\mathbf{u})} \mathrm{d}\mathbf{u}$ is the normalization constant of the distribution, and $R \colon \mathbb{R}^n \to \mathbb{R}$ is, traditionally, a convex functional, even though in recent times this has bee shown to be more of a limitation than a perk and has been questioned quite a lot [27, 87].

Example 1.1. *Tikhonov.*
Tikhonov regularization has been around for quite some time now and has found use both in imaging and machine learning applications. It simply features a quadratic penalty:

$$R(\mathbf{u}) = \|\mathbf{W}\mathbf{u}\|_2^2 = \sum_{i=1}^{p} (Wu)_i^2,$$

Here, $\mathbf{W}$ is a matrix that maps the image into some appropriate domain, e.g., the Wavelet Transform [14]. This regularization term promotes smoothness either on the signal itself, when $\mathbf{W} = \mathbf{I}_n$, or on its representation through $\mathbf{W}$.
When combined in (1.4) with a least squares term, the resulting problem is also known as *Ridge Regression* [52].

Example 1.2. *ℓ_1-norm.*
ℓ_1-regularization, as the name suggests, employs the ℓ_1-norm:

$$R(\mathbf{u}) = \|\mathbf{W}\mathbf{u}\|_1 = \sum_{i=1}^{p} |(Wu)_i|,$$

It is well known that many signals, such as astronomic or text images, are sparse, or they have a sparse representation in some space with only a few non-zero pixels. Hence it is logical to promote this property through the regularizer. This is the main purpose of the ℓ_1 norm, and more in general of the ℓ_p-norms (or semi-norms) with $0 \leq p < 1$, thanks to the different topology they induce. However, while the ℓ_1-norm is a convex function, the ℓ_p-norm for $p < 1$ is not. In the extreme case of $p = 0$ we actually obtain the ℓ_0-norm:

$$R(\mathbf{u}) = \|\mathbf{W}\mathbf{u}\|_0 = |\{(Wu)_i \colon (Wu)_i \neq 0\}|,$$

which counts the number of non-zero components of the vector. However, we note that solving the minimization problem

$$\min_{\mathbf{u} \in \mathbb{R}^n} \|\mathbf{u}\|_0$$

is NP-hard.
As its counterpart, when the ℓ_1-norm is used together with a least squares term

in (1.4), the problem takes the name of *LASSO* (*Least Absolute Shrinkage and Selection Operator*) [94].

Example 1.3. *Total Variation.*
A notable special case for the previous example is the Total Variation regularizer [81], which imposes sparsity on the gradient of the image. For clarity, let us consider $\mathbf{u}$ in its matrix form of dimensions $n_1 \times n_2$, with $n_1 n_2 = n$. We can define the *finite differences* operator $\mathbf{D} : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n \times 2}$ as

$$\mathbf{Du} = \begin{pmatrix} u_{1,1} - u_{1,2} & u_{1,1} - u_{2,1} \\ u_{1,2} - u_{1,3} & u_{1,2} - u_{2,2} \\ \vdots & \vdots \\ u_{n_1,n_2} - u_{n_1,n_2+1} & u_{n_1,n_2} - u_{n_1+1,n_2} \end{pmatrix} \tag{1.6}$$

where we can use different boundary conditions whenever we have either $n_1 + 1$ or $n_2 + 1$ as the subscripts:

- *zero or Dirichlet*: simply assumes that the out-of bound pixel is zero.
- *periodic*: $u_{n_1+1,j} = u_{1,j}$ and $u_{i,n_2+1} = u_{i,1}$.
- *reflective*: $u_{n_1+1,j} = u_{n_1,j}$ and $u_{i,n_2+1} = u_{i,n_2}$.

It is worth noting that applying $\mathbf{D}$ to an image can be considered as applying two convolutional kernels to the image, one for the horizontal derivatives and one for the vertical ones. If we assume to penalize equally the horizontal and the vertical gradients, the regularizer has the form

$$TV(\mathbf{u}) := \|\mathbf{Du}\|_1 := \sum_{l=1}^{n} \|(\mathbf{Du})_l^T\|_2 = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \sqrt{(u_{i,j} - u_{i,j+1})^2 + (u_{i,j} - u_{i+1,j})^2}, \tag{1.7}$$

and is called the *isotropic* total variation. However, it is also possible to consider the *anisotropic* version

$$TV_a(\mathbf{u}) := \|(\mathbf{Du})_1\|_1 + \|(\mathbf{Du})_2\|_1 = \sum_{i=1}^{n_1} |(Du)_{i,1}| + \sum_{j=1}^{n_2} |(Du)_{j,2}|, \tag{1.8}$$

where $(\mathbf{Du})_c$ is the c-th column of $\mathbf{Du}$, for $c = 1, 2$. This second version allows for weighting one direction more than the other.
In Figure 1.1 the reader can visualize the effects of the finite differences operator in (1.6). Image (b) depicts the absolute value of the horizontal derivatives $\mathbf{Du}_1$, which in fact highlight elements such as the window frames, while image (c) contains the vertical derivatives that put more emphasis on the longitudinal details of the image. Image (d) shows the sum of (b) and (c), which is the sum of the absolute value of the two columns of $\mathbf{Du}$. Unfortunately, some elements are lost or barely picked up, such as the sky or the reflections in the water. Indeed, while this functional is

excellent at reconstructing edges, a known shortcoming is the lack of texture in the images as it tends to flatten image patches a bit too much.
Considering the already promising starting point offered by the TV, it is no wonder that many variants have been proposed, all keen at reducing or directly eliminating its shortcomings. One possible improvement is offered by the *Weighted Total Variation* (WTV) [25, 53].

$$WTV(\mathbf{u}) := \sum_{i=1}^{n} \rho_i \| \left((Du)_{i,1}, (Du)_{i,2}\right) \|_2,$$

were the amount of regularization is controlled pixel by pixel by the parameters ρ_i. Another strategy could be incorporating higher order derivatives, since they contain more information of the textures of the image, which is an idea pursued for instance by the *Total Generalized Variation* (TGV) in [23, 85].

The conditional probability $\mathbb{P}(\mathcal{Y} = \mathbf{y} \mid U = \mathbf{u})$ is usually chosen according to the noise $N(\cdot)$ that is present on the image, through a classical *maximum likelihood* approach. Here we present the two, arguably by far, most common types of noise.

Example 1.4. *Additive white Gaussian noise.*
The most common type of noise can be modelled in the following way:

$$\mathbf{y} = N(\mathbf{u}) = \mathbf{u} + \boldsymbol{\eta}$$

where each component of $\boldsymbol{\eta}$ is an i.i.d. random variable with distribution $\mathcal{N}(0, \sigma^2)$. Taking into account the acquisition process (1.1) we have that $\boldsymbol{\eta} = \mathbf{y} - \mathcal{N}(\mathbf{Au} + \mathbf{bg})$ and

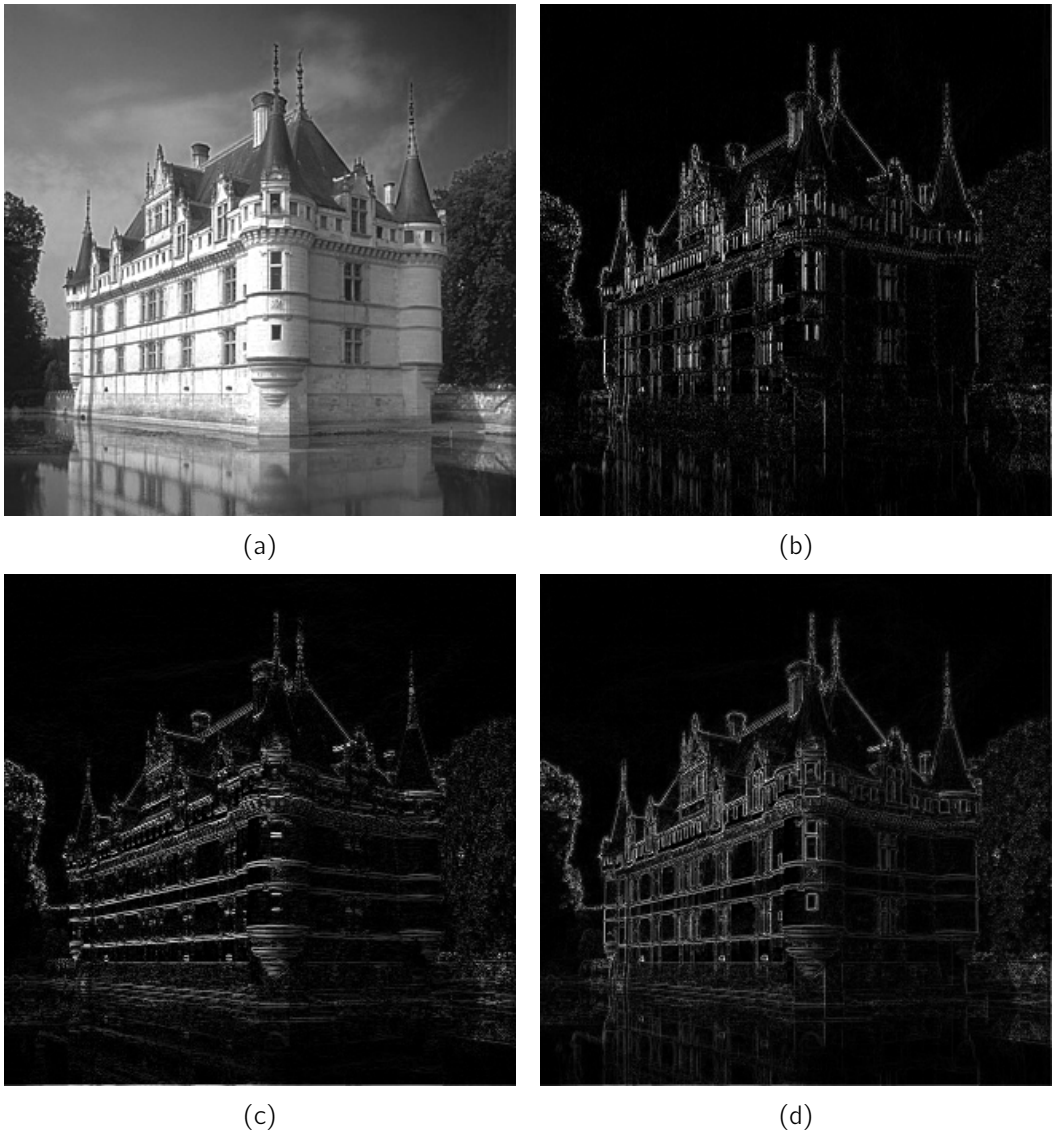
$$\mathbb{P}(\mathcal{Y} = \mathbf{y} \mid \mathcal{U} = \mathbf{u}) = \frac{1}{\sqrt{2\pi\sigma^2}} \prod_{i=1}^{m} e^{-\frac{(y_i - (Au+bg)_i)}{2\sigma^2}}$$
$$-\log(\mathbb{P}(\mathcal{Y} = \mathbf{y} \mid \mathcal{U} = \mathbf{u})) = \frac{C_{\boldsymbol{\sigma}}}{2} \|\mathbf{y} - (\mathbf{Au} + \mathbf{bg})\|_2^2.$$

The constant $C_{\boldsymbol{\sigma}}$ is always automatically incorporated in other parameters of the model and thus will be omitted from now on.

Example 1.5. *Poisson noise.*
Poisson noise is not additive, instead each pixel of the image becomes an independent random variable with Poisson distribution, governed by the true value of the pixel:

$$y_i \sim \mathcal{P}\Big((Au + bg)_i\Big),$$

i.e., the observed pixel y_i is sampled from the Poisson distribution with expected value $(Au + bg)_i$.



(a) (b)

(c) (d)

Figure 1.1: Illustration of the Finite Differences filter applied to (a). The images in (b) and (c) show, respectively, the horizontal and vertical gradients, while (d) contains the sum of image (b). and (c).

Similarly to the previous example

$$\begin{aligned}
\mathbb{P}(\mathcal{Y} = \mathbf{y} \mid \mathcal{U} = \mathbf{u}) &= \prod_{i=1}^{m} \frac{(Au + bg)_i^{y_i} e^{-(Au+bg)_i}}{y_i!} \\
-\log(\mathbb{P}(\mathcal{Y} = \mathbf{y} \mid \mathcal{U} = \mathbf{u})) &= \sum_{i=1}^{m} (Au + bg)_i + \log(y_i!) - y_i \log\Big((Au + bg)_i\Big) \\
&\simeq \sum_{i=1}^{m} (Au + bg)_i - y_i \log\left(\frac{(Au + bg)_i}{y_i}\right).
\end{aligned}$$

In the last line, the Stirling approximation formula yields $\log(y_i!) \simeq y_i \log(y_i) - y_i$.

Poisson noise is typically found in applications where the measurements to acquire the image are done via photon counting. Photons obey quantum mechanics laws, and thus, their behaviour is not deterministic but is more appropriately described by a probability distribution function. The primary cause of Poisson noise is not a miscalculation of the number of photons, but rather it is a good approximation of their distribution [9].

In light of this analysis, the formulation (1.4) is no other than a maximum likelihood problem but with a regularization term and can be cast as a *variational model*

$$\begin{aligned}\bar{\mathbf{u}} &= \operatorname*{argmin}_{\mathbf{u}\in\mathbb{R}^n} E_{\mathbf{y}}(\mathbf{u}) \equiv F_{\mathbf{y}}(\mathbf{u}) + \rho R(\mathbf{u}) \\ &= \operatorname*{argmin}_{\mathbf{u}\in\mathbb{R}^n} E_{\mathbf{y}}(\mathbf{u};\boldsymbol{\theta}) \equiv F_{\mathbf{y}}(\mathbf{u};\boldsymbol{\theta}) + R(\mathbf{u};\boldsymbol{\theta}) \qquad \text{(VM)}\end{aligned}$$

where the energy functional $E_{\mathbf{y}}$ is composed by a data fidelity $F_{\mathbf{y}}$, which measures the distance between $\mathbf{Au}$ and the data $\mathbf{y}$, and the regularizer R. The balance between the two terms is controlled by the so called regularization parameter ρ. In the second line we have made explicit the dependence of the functionals on a set of hyperparameters $\boldsymbol{\theta} \in \mathbb{R}^p$ (including ρ), which can be used to create more complex models.

1.2 Parameter Selection: an Overview

In the last few years, hyperparameter selection has been an important topic of research. The issue lies in the fact that there really are no guidelines on how to choose these quantities that work for every (or the vast majority of) task. This is also true in almost all applications of machine learning, deep learning and imaging. Most of the time one has to go on a case by case fashion and follow some empirical strategy. The ultimate scope of this work is to investigate strategies to make a sensible selection. Nonetheless, in the algorithms there will always be some variables that are left out and still need to be manually tuned, ideally those that have a lesser influence on the results. Thus, we are going to restrict the term hyperparameters only for these latter quantities, and simply call parameters all those that are automatically set by the methods proposed in this manuscript. In this section we are going to present some of the most known and general strategies for parameter selection, both in imaging and machine learning.

Grid Search. The first, most basic, and instinctive way of selecting the parameters is to discretize the domain in which they lie with an arbitrarily fine grid of equidistant points, and then try every possible value. While this is rather simple to implement, it suffers from the curse of dimensionality in multiple ways. Obviously, one would like to try as many parameters as possible by increasing the resolution of the grid, however,

this also potentially requires an extraordinary amount of time and/or computational effort. This inefficiency worsens exponentially when there is more than one parameter. Also, if one is dealing with an high dimensional problem or, in general, obtaining a possible solution is expensive, they still are forced to use a coarser grid.

Random Search. Another possibility is to draw randomly a set of parameters from a given distribution, usually the uniform one. The benefits of such choice have been empirically shown in [7], where for all experiments proposed the number of random trials needed to match or improve the performance of grid search is only a fraction of the total number of nodes in the grid. Intuitively, the advantage lies in the fact that this method better explores the space of more influential parameters while not wasting resources on the lesser ones. In other words, the cost of tuning parameters increases far less with their number, unlike grid search.

Bayesian Optimization. *Bayesian optimization*, also sometimes known as *Sequential Model-Based Optimization* (SMBO) is a more sophisticated approach and presents a notable improvement in the case where computing a solution or evaluating the loss function is rather costly [46]. It can be summarised in three steps:

1. Fit a surrogate function, that is supposed to approximate the loss, based on the points of the parameters space already established;
2. Use an acquisition function to find a promising new set of parameters;
3. Test the parameters and evaluate the loss function.

Unfortunately, at the end of the day SMBO is not as used as Random Search, despite being a more refined tool. When evaluating the loss function truly requires an unsustainable computational cost, then the method, with its philosophy of making the most out of each loss evaluation, can actually be beneficial. However, in most other cases, the efforts needed to put the approach in place are not worth the hassle. The surrogate and the acquisition functions do have an influence on the results, thus it is necessary to find a good fit and tune, which is what SMBO is supposed to do for the underlying model.

Discrepancy Principle. The *(Morozov) Discrepancy Principle* was established by V. A. Morozov in the past century, but still finds applications to this day [90, 91]. Its purpose is that of finding a meaningful regularization parameter ρ that balances the tradeoff between data fidelity and regularization in the energy $E(\mathbf{u};\boldsymbol{\theta}) = F(\mathbf{u};\boldsymbol{\theta}) + \rho R(\mathbf{u};\boldsymbol{\theta})$. Clearly, the minimum of E, i.e. $\bar{\mathbf{u}}^{\rho}$, depends of the value of ρ. Assuming that the noise level on the image is δ, i.e.

$$\|\mathbf{A}\mathbf{u}^{\text{true}} + \mathbf{b}\mathbf{g} - \mathbf{y}\|_2 \leq \delta,$$

the discrepancy principle revolves around finding a value ρ such that the restoration $\bar{\mathbf{u}}^\rho$ satisfies

$$c_1\delta \leq \|\mathbf{A}\bar{\mathbf{u}}^\rho + \mathbf{bg} - \mathbf{y}\|_2 \leq c_2\delta$$

or equivalently

$$\|\mathbf{A}\bar{\mathbf{u}}^\rho + \mathbf{bg} - \mathbf{y}\|_2 = C\delta,$$

for some constants $C > 0, 0 < c_1 \leq c_2$. Although the discrepancy principle requires, in theory, the level of the noise, there have been studies that show how it is possible to apply the principle a posteriori. In particular, in [8] the author show, for different standard regularization terms, how the method can be used in the presence of unknown Poisson noise.

Chapter 2

Bilevel Optimization

The advent of data driven methods, and their impressive results across many disciplines naturally leads to wonder whether there is a way to adapt some of their underlying ideas to improve variational models, but sacrificing as less as possible of their solid and established theoretical background. The most basic approach would involve using a loss function to measure the performance of the underlying variational model and penalize wrong sets of parameters. The resulting scheme is what is called a *bilevel optimization* problem. In this chapter, section 2.1 is dedicated to the general formulation of this kind of problems, along with a brief description of the principal strategies for their resolution in this form. Section 2.2 focuses on the specific bilevel problems that arise from imaging applications, especially when tied to variational models. Again, one of the most popular strategies is described. Section 2.3 is a natural follow up to the previous section, as it contains our main result in solving bilevel optimization problems for learning variational models.

2.1 General Formulation

Bilevel optimizatin problems, in the most general case, can be cast as

$$\begin{cases} \text{“ } \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \text{ ”} \; L(\mathbf{u}, \boldsymbol{\theta}) \\ \text{sub. to } G(\mathbf{u}, \boldsymbol{\theta}) \leq 0 \\ \qquad\qquad \mathbf{u} \in \underset{\mathbf{u}'}{\operatorname{argmin}} \{ E(\mathbf{u}'; \boldsymbol{\theta}) : g(\mathbf{u}', \boldsymbol{\theta}) \leq 0 \}, \end{cases} \tag{2.1}$$

where $G \colon \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^r$ and $g \colon \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^s$. The bilevel approach involves solving two optimization problems, where one appears as a constraint for the other. The main problem, i.e. the outer one, is also called the *upper level problem*, while the inner is called the *lower level problem*. The study and analysis of nested optimization problems is especially difficult due to the upper level being dependent on the outcome of the lower level, which usually does not have an explicit formula, i.e., the variable

$\mathbf{u}$ is implicitly defined. Unless the lower level objective function admits only one minimum, the problem as presented above is not well-defined, which is why the argmin operator is between quotes. Thus, it is important to specify the strategy that is used to obtain the lower minimum, as it heavily influences the general theoretical approach to study the problem. The total spectrum of such strategies can be placed within two extremum points. If we denote the *solution map* for the lower level problem with $S(\boldsymbol{\theta}) = \text{argmin}_{\mathbf{u}}\{E(\mathbf{u};\boldsymbol{\theta}) : g(\mathbf{u},\boldsymbol{\theta}) \leq 0\}$, then

- The *optimistic solution* $\mathbf{u}^+(\boldsymbol{\theta})$ is the one that produces the optimal value for the loss function, i.e., $\mathbf{u}^+(\boldsymbol{\theta}) = \text{argmin}_{\mathbf{u}}\{L(\mathbf{u},\boldsymbol{\theta}) \colon \mathbf{u} \in \mathcal{S}(\boldsymbol{\theta}), G(\mathbf{u},\boldsymbol{\theta}) \leq 0\}$. It is worth noting that, in this case, the bilevel problem can also be equivalently rewritten as

$$\begin{cases} \underset{\boldsymbol{\theta},\mathbf{u}}{\text{argmin}}\ L(\mathbf{u},\boldsymbol{\theta}) \\ \text{sub. to } G(\mathbf{u},\boldsymbol{\theta}) \leq 0 \\ \qquad\qquad \mathbf{u} \in \underset{\mathbf{u}'}{\text{argmin}}\{E(\mathbf{u}';\boldsymbol{\theta}) : g(\mathbf{u}',\boldsymbol{\theta}) \leq 0\}, \end{cases}$$

  where the upper level function is optimized also w.r.t. $\mathbf{u}$.

- The *pessimistic solution* $\mathbf{u}^-(\boldsymbol{\theta})$ is the one that produces the worst value for the loss function, i.e., $\mathbf{u}^-(\boldsymbol{\theta}) = \text{argmax}_{\mathbf{u}}\{L(\mathbf{u},\boldsymbol{\theta}) \colon \mathbf{u} \in \mathcal{S}(\boldsymbol{\theta}), G(\mathbf{u},\boldsymbol{\theta}) \leq 0\}$.

In applications, there are also cases where in practice the minimum of the lower level is not even available, as we will see. As natural as it sounds the optimistic solution is easier to deal with, and most of the time requires weaker assumptions to achieve convergence to the solution of the bilevel problem [86]. In the case of an optimistic approach, the two main strategies to solve bilevel optimization problems can be formulated as follows.

Karush-Kuhn-Tucker Conditions. In the case where the lower level objective function is convex and differentiable, one possibility lies in replacing the lower level constrained optimization problem with its *KKT optimality conditions*:

$$\nabla_{\mathbf{u}} E(\mathbf{u},\boldsymbol{\theta}) + \boldsymbol{\mu}^T \nabla_{\mathbf{u}} \mathbf{g}(\mathbf{u};\boldsymbol{\theta}) = 0 \tag{2.2}$$

$$\mu_i \geq 0 \qquad i = 1,\dots,s \tag{2.3}$$

$$g_i(\mathbf{u},\boldsymbol{\theta}) \leq 0 \qquad i = 1,\dots,s \tag{2.4}$$

$$\mu_i g_i(\mathbf{u},\boldsymbol{\theta}) = 0 \qquad i = 1,\dots,s. \tag{2.5}$$

Thus the bilevel problem is turned into a single-level *Mathematical Program with Complementary Constraints* (MPCC), that are given by equations (2.3), (2.4) and (2.5). Unfortunately, especially in imaging applications, the energy functional is often either non-smooth or non-convex, and sometimes both, which means that the algorithms used for solving MPCCs are seldom applicable as they are. Thus, researchers

of the field have mostly focused on determining the conditions for which the equivalence actually holds, and eventually on developing extensions and generalizations for the tools of MPCCs [31, 33].

Lower-Level Value Function. The *Lower-Level Value Function* (LLVF), also called Optimal Value Function, approach offers the possibility to obtain a single-level optimization problem. The LLVF is defined as

$$\begin{aligned} \varphi\colon \mathbb{R}^p &\longrightarrow \mathbb{R} \\ \boldsymbol{\theta} &\longmapsto \varphi(\boldsymbol{\theta}) := \min_{\mathbf{u}}\{E(\mathbf{u};\boldsymbol{\theta})\colon g(\mathbf{u},\boldsymbol{\theta}) \leq 0\}, \end{aligned} \tag{2.6}$$

so that the bilevel problem can be rewritten as

$$\begin{cases} \operatorname{argmin}_{\boldsymbol{\theta}} L(\mathbf{u},\boldsymbol{\theta}) \\ \text{sub. to } G(\mathbf{u},\boldsymbol{\theta}) \leq 0 \\ \qquad\quad g(\mathbf{u},\boldsymbol{\theta}) \leq 0 \\ \qquad\quad E(\mathbf{u};\boldsymbol{\theta}) \leq \varphi(\boldsymbol{\theta}). \end{cases} \tag{2.7}$$

This formulation was originally introduced by Outrata in [71] and a set of optimality conditions was first developed by Ye and Zhu in [98]. The downsides of the approach are that the LLVF φ may not be easy to handle. For starters, it is not always known in practice, and besides, it is often non-differentiable. Furthermore, the resulting problem in often non-convex even in the case where the objectives and the constraints are convex. However, in recent years, thanks to the fact that it does not present derivatives of the lower level functions, it has allowed to develop second order schemes for its solution [38, 39].

The book [32] by Stephan Dempe presents in detail and with much more depth the ideas summed up above. In the rest of this thesis, we are going to focus on a specific kind of bilevel problem, whose naturally suggests different approaches to reach its solution.

2.2 Learning Variational Models Via Bilevel Optimization

As we have already introduced, bilevel optimization is a suitable strategy to automatically select the parameters $\boldsymbol{\theta}$ of variational models. However, formulation (2.1) is actually more general, and in our applications we are naturally led to a simpler problem. In order to formulate the bilevel optimization problem for imaging applications, we need;

- A merit function L that is able to evaluate the quality of the lower level solution, whether it be in a supervised or unsupervised fashion;
- A dataset $\mathcal{D}$, where each of the $|\mathcal{D}|$ examples contains the features that we need. Usually, it includes the features of corrupted samples, and if the merit function is supervised then the corresponding ground truths are also required to be in $\mathcal{D}$.

$$\begin{cases} \bar{\boldsymbol{\theta}} \in \underset{\boldsymbol{\theta}\in\Theta\subseteq\mathbb{R}^p}{\operatorname{argmin}} \sum_{s=1}^{|\mathcal{D}|} L(\bar{\mathbf{u}}_s(\boldsymbol{\theta}), \boldsymbol{\theta}) \\ \text{subject to } \bar{\mathbf{u}}_s \in \underset{\mathbf{u}\in U}{\operatorname{argmin}} E_{\mathbf{y}_s}(\mathbf{u}; \boldsymbol{\theta}), \quad s = 1, \dots, |\mathcal{D}|, \end{cases} \tag{2.8}$$

where $\Theta \subseteq \mathbb{R}^p$ and $U \subseteq \mathbb{R}^n$ are the spaces where the parameters $\boldsymbol{\theta}$ and the images $\mathbf{u}_s$ lie, respectively. Basically, once the underlying variational model is established and the energy function in the lower level is set, the solution of the bilevel problem is a point $\bar{\boldsymbol{\theta}}$ that achieves the best performance according to some criterion expressed by the loss (or penalty) function L in the upper level. For instance, the loss function can be chosen directly as the quality measure used to evaluate the model, or as something that is strictly related to it, like the ℓ_2 norm can be used when the performance is evaluated in terms of (P)SNR (Peak Signal-to-Noise Ratio). Also, natural images may require a different loss functions than sparse images, or images that are known to be sparse in some domain, in which case one would probably achieve better results by transforming the solutions of the lower before feeding them to the upper level. In this work, we are going to explore different choices and approaches both in the upper and lower levels, for a variety of imaging problems.

In this thesis, we are going minimize the loss function via a gradient-based method. Since computing the gradient of the loss function is linear in the number of samples, for the rest of this manuscript we are always going to assume that $|\mathcal{D}| = 1$ and omit the subscript s. We are also going to assume the ability to split the loss function into two/three terms:

$$\begin{aligned} L(\mathbf{u}(\boldsymbol{\theta}), \boldsymbol{\theta}) &= F_L(\mathbf{u}(\boldsymbol{\theta})) + R_L^0(\boldsymbol{\theta}) + R_L^1(\boldsymbol{\theta}) && (2.9) \\ &= G_L(\mathbf{u}(\boldsymbol{\theta}), \boldsymbol{\theta}) + R_L^1(\boldsymbol{\theta}), && (2.10) \end{aligned}$$

The form for the loss function is actually quite reasonable in practice. Basically, the F is a fidelity which penalizes reconstructions that are more distant from their ground truth, or more in general it can be a quality measure, e.g. when the ground truths are not available. On the other hand, R_L^0 and R_L^1 act as the regularization for the variable $\boldsymbol{\theta}$. In particular, R_L^0 is a differentiable function, while R_L^1 is convex but non-differentiable. The second equation is a more operative and compact version of the first one: under the function G_L we have the differentiable terms F_L and R_L^0. In the following, to ease the notation, we are going to drop the subscript L.

Considering the second split, Algorithm 2.1 describes the very general framework that we are going to follow and try improve on.

Algorithm 2.1: Bilevel Optimization: general proximal gradient based scheme

1 **Input:** starting point $\boldsymbol{\theta}^{(0)}$, $k = 0$
2 **while** *stopping condition is not met* **do**
3 Compute $\bar{\mathbf{u}}(\boldsymbol{\theta}^{(k)}) \in \underset{\mathbf{u}\in U}{\operatorname{argmin}}\, E_{\mathbf{y}}(\mathbf{u}; \boldsymbol{\theta}^{(k)})$.
4 Compute $\boldsymbol{\nabla}_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)})$.
5 Compute a suitable steplength α_k and a scaling matrix $\mathbf{M}_k$.
6 Compute the updated $\boldsymbol{\theta}^{(k+1)}$ by means of

$$\boldsymbol{\theta}^{(k+1)} = \operatorname{prox}_{\alpha R^1}^{\mathbf{M}_k}\left(\boldsymbol{\theta}^{(k)} - \alpha_k \mathbf{M}_k^{-1} \boldsymbol{\nabla}_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)})\right). \tag{2.11}$$

7 Set $k = k + 1$.
end
8 **Output:** learned parameters $\bar{\boldsymbol{\theta}}$

The most critical steps are lines 3 and 4 due to their relation. The loss function and its gradient are dependent on $\bar{\mathbf{u}}$ which is a minimum point for $E_{\mathbf{y}}$. Using the definition of the solution map $S(\boldsymbol{\theta}) \in \operatorname{argmin}_{\mathbf{u}}\{E_{\mathbf{y}}(\mathbf{u}; \boldsymbol{\theta}) : \mathbf{u} \in U\}$ we can elaborate a bit more on the gradient $\boldsymbol{\nabla}_{\boldsymbol{\theta}} G$:

$$\begin{aligned} G(\bar{\mathbf{u}}(\boldsymbol{\theta}), \boldsymbol{\theta}) &= G(S(\boldsymbol{\theta}), \boldsymbol{\theta}) \\ \boldsymbol{\nabla}_{\boldsymbol{\theta}} G(S(\boldsymbol{\theta}), \boldsymbol{\theta}) &= \mathbf{J}_S(\boldsymbol{\theta})^T \boldsymbol{\nabla}_{\mathbf{u}} F(S(\boldsymbol{\theta})) + \boldsymbol{\nabla}_{\boldsymbol{\theta}} R^0(\boldsymbol{\theta}). \end{aligned} \tag{2.12}$$

The clear troublemaker is the first term, that would be $\boldsymbol{\nabla}_{\boldsymbol{\theta}} F$, which features the gradient of F with respect to $\mathbf{u}$, pre-multiplied by the Jacobian matrix of the solution map, $\mathbf{J}_S \in \mathbb{R}^{n\times p}$. Computing this Jacobian is not simple. Differentiating through the argmin or argmax operators has gathered attention for quite some time now [6, 50, 63], as it is useful in many fields, from image reconstruction [27, 92] to the machine learning [13, 72] realms and beyond.
Implicit differentiation exploits simple calculus rules to formulate an explicit expression for the jacobian $\mathbf{J}_S$ under somewhat standard, but not weak or mild, assumptions.

Theorem 2.1

Assume that $\forall \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^p$ the function $E_{\mathbf{y}}(\cdot; \boldsymbol{\theta}) : U \to \mathbb{R}$ is twice continuously differentiable and is strongly convex. If the gradient $\boldsymbol{\nabla}_{\mathbf{u}} E_{\mathbf{y}}$ is continuously differentiable with respect to $\boldsymbol{\theta}$, then the solution map S is continuously differentiable with respect to $\boldsymbol{\theta}$ and its jacobian can be written as

$$\mathbf{J}_S = -\underbrace{\boldsymbol{\nabla}^2_{\mathbf{uu}} E_{\mathbf{y}}(S(\boldsymbol{\theta}); \boldsymbol{\theta})^{-1}}_{n\times n} \underbrace{\boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E_{\mathbf{y}}(S(\boldsymbol{\theta}); \boldsymbol{\theta})}_{n\times p}. \tag{2.13}$$

Proof : For any $\boldsymbol{\theta} \in \Theta$, $S(\boldsymbol{\theta})$ is the unique minimizer of $E_{\mathbf{y}}(\cdot;\boldsymbol{\theta})$, hence if satisfies the first order optimality conditions:

$$\boldsymbol{\nabla}_{\mathbf{u}} E_{\mathbf{y}}(S(\boldsymbol{\theta});\boldsymbol{\theta}) = 0. \tag{2.14}$$

The implicit function theorem gives us the continuous differentiability of S. Differentiating with respect to $\boldsymbol{\theta}$ through this last equation and applying the chain rule yields

$$\boldsymbol{\nabla}^2_{\mathbf{uu}} E_{\mathbf{y}}(S(\boldsymbol{\theta});\boldsymbol{\theta})\mathbf{J}_S + \boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E_{\mathbf{y}}(S(\boldsymbol{\theta});\boldsymbol{\theta}) = 0.$$

Since the hessian is strongly convex, it is invertible and we can then easily arrive at

$$\mathbf{J}_S = -\boldsymbol{\nabla}^2_{\mathbf{uu}} E_{\mathbf{y}}(S(\boldsymbol{\theta});\boldsymbol{\theta})^{-1}\boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E_{\mathbf{y}}(S(\boldsymbol{\theta});\boldsymbol{\theta}).$$

□

Although the previous theorem gives us a fundamental result, there are still a few aspects that need to be considered before making it viable in practice. Once this result is implemented in equation (2.12) we obtain

$$\begin{aligned}\boldsymbol{\nabla}_{\boldsymbol{\theta}} G(S(\boldsymbol{\theta}),\boldsymbol{\theta}) &= \boldsymbol{\nabla}_{\boldsymbol{\theta}} R^0(\boldsymbol{\theta}) - \left(\boldsymbol{\nabla}^2_{\mathbf{uu}} E_{\mathbf{y}}(S(\boldsymbol{\theta});\boldsymbol{\theta})^{-1}\boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E_{\mathbf{y}}(S(\boldsymbol{\theta});\boldsymbol{\theta})\right)^T \boldsymbol{\nabla}_{\mathbf{u}} F(S(\boldsymbol{\theta})) \\ &= \boldsymbol{\nabla}_{\boldsymbol{\theta}} R^0(\boldsymbol{\theta}) - \boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E_{\mathbf{y}}(S(\boldsymbol{\theta});\boldsymbol{\theta})^T \boldsymbol{\nabla}^2_{\mathbf{uu}} E_{\mathbf{y}}(S(\boldsymbol{\theta});\boldsymbol{\theta})^{-1}\boldsymbol{\nabla}_{\mathbf{u}} F(S(\boldsymbol{\theta})) \end{aligned} \tag{2.15}$$

The most obvious is the inversion of the hessian, which should be avoided in the vast majority of cases. Denoting with

$$\mathbf{q} = \boldsymbol{\nabla}^2_{\mathbf{uu}} E_{\mathbf{y}}(S(\boldsymbol{\theta});\boldsymbol{\theta})^{-1}\boldsymbol{\nabla}_{\mathbf{u}} F_L(S(\boldsymbol{\theta})) \in \mathbb{R}^n,$$

we rather have to solve the linear system

$$\boldsymbol{\nabla}^2_{\mathbf{uu}} E_{\mathbf{y}}(S(\boldsymbol{\theta});\boldsymbol{\theta})\mathbf{q} = \boldsymbol{\nabla}_{\mathbf{u}} F_L(S(\boldsymbol{\theta})). \tag{2.16}$$

This is a common approach that was followed, for instance, in [72] and [92]. Just like in these two works, we avoid directly solving the linear system but rather we only look for a sufficiently good approximation of its solution.
Unfortunately the linear system (2.16) can not be handled as it is, since, in applications, the minimum $\bar{\mathbf{u}}(\boldsymbol{\theta})$ usually is not available in an explicit form that can be computed exactly. Hence, in practice, we have to obtain approximation of $\bar{\mathbf{u}}(\boldsymbol{\theta})$, denoted with $\tilde{\mathbf{u}}(\boldsymbol{\theta})$, via an optimization algorithm applied to the minimization of the energy functional $E_{\mathbf{y}}(\cdot,\boldsymbol{\theta})$. This operation, no matter how accurate $\tilde{\mathbf{u}}(\boldsymbol{\theta})$ is, does always introduce some error in both the computation of the gradient of G_L and of the loss itself. Then, the inexact jacobian of the solution maps becomes

$$\tilde{\mathbf{J}}_S = -\boldsymbol{\nabla}^2_{\mathbf{uu}} E_{\mathbf{y}}(\tilde{\mathbf{u}}(\boldsymbol{\theta});\boldsymbol{\theta})^{-1}\boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E_{\mathbf{y}}(\tilde{\mathbf{u}}(\boldsymbol{\theta});\boldsymbol{\theta}). \tag{2.17}$$

There is research in the literature that shows, under some strong but not unusual or unseen assumptions, that the approximate jacobian in (2.17) does indeed converge

to the true jacobian as $\tilde{\mathbf{u}}(\boldsymbol{\theta})$ becomes a better approximation of $\bar{\mathbf{u}}(\boldsymbol{\theta})$, and the convergence can also be as fast as the convergence of $\tilde{\mathbf{u}}(\boldsymbol{\theta})$ to $\bar{\mathbf{u}}(\boldsymbol{\theta})$. These assumptions, in general, are consistent with those of theorem 2.1, as the energy still needs to be twice continuously differentiable and to have a unique minimum. In the next section, following this approach, we are going to prove a very similar result to those, for instance, in [50, 63].

2.3 Solving Bilevel Optimization Via an Inexact Forward-Backward Scheme

In this section we are going to present an algorithm that solves the following bilevel optimization problem:

$$\begin{cases} \bar{\boldsymbol{\theta}} \in \underset{\boldsymbol{\theta}\in\mathbb{R}^p}{\operatorname{argmin}}\ L(\bar{\mathbf{u}}(\boldsymbol{\theta}), \boldsymbol{\theta}) \equiv F(\bar{\mathbf{u}}(\boldsymbol{\theta})) + R^0(\boldsymbol{\theta}) + R^1(\boldsymbol{\theta}) \\ \text{where } \bar{\mathbf{u}}(\boldsymbol{\theta}) = \underset{\mathbf{u}\in\mathbb{R}^n}{\operatorname{argmin}}\ E(\mathbf{u}; \boldsymbol{\theta}). \end{cases} \tag{2.18}$$

Since the upper level problem requires the minimization to be carried out with respect to the variable $\boldsymbol{\theta}$, with a slight abuse of notation, we will write $L(\boldsymbol{\theta})$ and $F(\boldsymbol{\theta})$ when we do not care that the dependence from $\boldsymbol{\theta}$ of those functions is through $\mathbf{u}$.

Assumptions 2.2.
For our scheme, we require the following properties:

(i) The function $L\colon (\mathbf{u}, \boldsymbol{\theta}) \in \mathbb{R}^n \times \mathbb{R}^p \mapsto \mathbb{R}$ is bounded from below.

(ii) The function $R^0\colon \boldsymbol{\theta} \in \mathbb{R}^p \mapsto \mathbb{R}$ is continuously differentiable and its gradient is $\Lambda_{\nabla R^0}$-Lipschitz continuous.

(iii) The function $R^1\colon \boldsymbol{\theta} \in \mathbb{R}^p \mapsto \bar{\mathbb{R}}$ is proper, lower semicontinuous and convex.

(iv) The function $F\colon \mathbf{u} \in \mathbb{R}^n \mapsto \mathbb{R}$ is Λ_F-Lipschitz continuous, continuously differentiable and its gradient is $\Lambda_{\nabla F}$-Lipschitz continuous.

(v) The function $E\colon \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ is twice continuously differentiable and $\forall \boldsymbol{\theta} \in \operatorname{dom} R$ the function $E(\cdot; \boldsymbol{\theta})$ is strongly convex with modulus μ.

(vi) $\forall \boldsymbol{\theta} \in \operatorname{dom} R$, the function $\boldsymbol{\nabla}^2_{\mathbf{uu}} E(\cdot; \boldsymbol{\theta})\colon \mathbb{R}^n \mapsto \mathbb{R}^{n\times n}$ is $\Lambda_{\nabla^2_{uu}E}$-Lipschitz continuous.

(vii) $\forall \boldsymbol{\theta} \in \operatorname{dom} R$, the function $\boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(\cdot; \boldsymbol{\theta})\colon \mathbb{R}^n \mapsto \mathbb{R}^{n\times p}$ is $\Lambda_{\nabla_{\theta u}E}$-Lipschitz continuous.

(viii) The function

$$f : \mathbb{R}^p \times \mathbb{R} \longrightarrow \bar{\mathbb{R}} \tag{2.19}$$

$$(\boldsymbol{\theta}, p) \longmapsto f(x) = L(\boldsymbol{\theta}) + \frac{1}{2}p^2 \tag{2.20}$$

is a Kurdyka-Łojasiewicz (KL) function (see definition 2.10).

Under these assumptions on the structure of the problem, we are able to develop an iterative algorithm whose generated sequence converges to a stationary point of L. Note that R^1 is not necessarily differentiable. Also, Assumption (v) implies that the solution of the lower level problem is unique and that $\boldsymbol{\nabla}^2_{\mathbf{uu}} E(\mathbf{u}; \boldsymbol{\theta})$ is invertible. Finally, regarding Assumption (viii), it is shown in Remark 1 of [22] that, if $L(\boldsymbol{\theta})$ satisfies the KL property, then the function f as its defined above is still a KL function. However, we do not have a sufficient condition to verify in practice that the function L is a KL in the first place. In particular, we do not have any theoretical guarantee that the composition of F with the solution map S is a KL function, especially because we have little information on S.
As we did in the previous section, we are going to make the notation more compact by denoting with $G(\mathbf{u}(\boldsymbol{\theta}), \boldsymbol{\theta}) = F(\mathbf{u}(\boldsymbol{\theta})) + R^0(\boldsymbol{\theta})$, that it is still has a $\Lambda_{\nabla G}$-Lipschitz continuous gradient.

2.3.1 The Algorithm

The method that we propose for the update of the parameters $\boldsymbol{\theta}$ involves a forward gradient step, and a backward, proximal step on the non-differentiable function R. Essentially, our algorithm can be cast under the umbrella of *Forward-Backward* (FB) schemes, that are characterized by the iteration

$$\mathbf{z}^{(k)} = \operatorname{prox}^{\mathbf{M}_k}_{\alpha_k R^1}\left(\boldsymbol{\theta}^{(k)} - \alpha_k \mathbf{M}_k^{-1} \boldsymbol{\nabla}_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)})\right) \tag{2.21}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \lambda_k(\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}), \tag{2.22}$$

where α_k is the steplength along the arc, λ_k is the steplength along the descent direction $\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}$ and $\mathbf{M}_k \in \mathcal{M}_\nu$ is a scaling matrix. The idea behind these schemes is to exploit the gradient when its available, while turning to the subdifferential and the convexity of the non-differentiable terms. Behind FB methods there is an established literature, where a variety of cases has been handled [19, 75]. Since we are going to use some of the tools of this literature, we are going to recall some properties as we go on, but we also refer the reader to Appendix A for the basics of convex analysis. Here we recall the definition of the *proximal operator*:

$$\operatorname{prox}^{\mathbf{M}}_{\alpha R^1}(\boldsymbol{\varphi}) := \operatorname*{argmin}_{\boldsymbol{\theta} \in \mathbb{R}^p} \left[R^1(\boldsymbol{\theta}) + \frac{1}{2\alpha} \|\boldsymbol{\theta} - \boldsymbol{\varphi}\|^2_{\mathbf{M}} \right] \tag{2.23}$$

We can highlight two main obstacles in order to solve our problem with an FB scheme:

- The gradient $\nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta}^{(k)})$, and by extension the gradient $\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)})$, cannot be computed exactly, as we do not have the explicit minimum $\bar{\mathbf{u}}^{(k)}$ of $E(\cdot; \boldsymbol{\theta}^{(k)})$.
- The proximal operator $\text{prox}_{\alpha R^1}^{\mathbf{M}}(\cdot)$ may not have an explicit form. In fact, the proximal operator is another case of a function defined though the argmin operator.

Inexact Proximal Operator. In the general case, the proximal operator of R does not have a closed form solution, and needs to be computed inexactly. In this specific context, even if the proximal operator does have an explicit formulation for its minimum, the resulting point still has to be considered as inexact due to the approximation of the gradient for the differentiable part. Provided that the approximation that is computed satisfies a specific tolerance that we will discuss in a moment, we can achieve convergence through the KL framework. In this paragraph, we are going to present the basics of this operation, which can be found in full details in [20]. Let us start with the definition of descent directions for non-differentiable functions ([78])

Definition 2.3. If $L : \mathbb{R}^p \to \bar{\mathbb{R}}$, is a proper function, then a vector $\mathbf{d} \in \mathbb{R}^p$ is a descent direction for L at $\boldsymbol{\theta} \in \text{dom}\, L$ when it satisfies

$$\limsup_{\lambda \downarrow 0^+} \frac{L(\boldsymbol{\theta} + \lambda \mathbf{d}) - L(\boldsymbol{\theta})}{\lambda} < 0. \tag{2.24}$$

where, essentially, the limit is the one-sided directional derivative $L'(\mathbf{u}; \mathbf{d})$. Naturally, the previous definition implies that $\exists \bar{\lambda} > 0$ such that

$$L(\boldsymbol{\theta} + \lambda \mathbf{d}) < L(\boldsymbol{\theta}) \qquad \forall \lambda \leq \bar{\lambda}.$$

When $\mathbf{z}^{(k)}$ is the exact proximal operator value, then its properties make $\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}$ a descent direction. The approximation $\tilde{z}^{(k)} \approx \mathbf{z}^{(k)}$ needs to be chosen so that it maintains this quality. To do this, one can place the burden of identifying descent directions, at the k-th iterate of a FB scheme, on the strongly convex function

$$h^{(k)}(\mathbf{z}) := \nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)})^T (\mathbf{z} - \boldsymbol{\theta}^{(k)}) + \frac{1}{2\alpha_k} \|\mathbf{z} - \boldsymbol{\theta}^{(k)}\|^2_{\mathbf{M}_k} + R^1(\mathbf{z}) - R^1(\boldsymbol{\theta}^{(k)}). \tag{2.25}$$

Indeed, if $h^{(k)}(\mathbf{z}) < 0$ then

$$\begin{aligned} L'(\boldsymbol{\theta}^{(k)}; \mathbf{z} - \boldsymbol{\theta}^{(k)}) &= \nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)})^T (\mathbf{z} - \boldsymbol{\theta}^{(k)}) + (R^1)'(\boldsymbol{\theta}^{(k)}; \mathbf{z} - \boldsymbol{\theta}^{(k)}) \\ &\leq \nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)})^T (\mathbf{z} - \boldsymbol{\theta}^{(k)}) + R^1(\mathbf{z}) - R^1(\boldsymbol{\theta}^{(k)}) \\ &\leq h^{(k)}(\mathbf{z}) < 0, \end{aligned}$$

where the first inequality is obtained from Theorem (23.1) in [78]. Also, it has a clear relationship with the proximal operator. Indeed for the step (2.21), we have

$$\mathbf{z}^{(k)} = \underset{\mathbf{z}\in\mathbb{R}^p}{\operatorname{argmin}}\ h^{(k)}(\mathbf{z}). \tag{2.26}$$

Considering the definition of h, it immediately holds that $h^{(k)}(\boldsymbol{\theta}^{(k)}) = 0$ and also $h^{(k)}(\mathbf{z}^{(k)}) < 0$ since $\mathbf{z}^{(k)}$ is a minimum of $h^{(k)}$. Unfortunately, finding $\tilde{\mathbf{z}}^{(k)}$ so that $h^{(k)}(\tilde{\mathbf{z}}^{(k)}) < 0$ is still not enough. Following the criterion used in [19] and we need to compute it so that

$$h^{(k)}(\tilde{\mathbf{z}}^{(k)}) - h^{(k)}(\mathbf{z}^{(k)}) \leq -\frac{\tau}{2} h^{(k)}(\tilde{\mathbf{z}}^{(k)}), \tag{2.27}$$

for some $\tau > 0$. While the presence of the true $\mathbf{z}^{(k)}$ makes it look like there is no way to satisfy this condition, it is actually not the case. Indeed, since solving the optimization problem in (2.23) in $\boldsymbol{\theta}^{(k)} - \alpha_k \mathbf{M}_k^{-1} \nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)})$ is equivalent to searching for the minimum of $h^{(k)}$, then turning to its dual problem and exploiting the duality gap to define the stopping criterion with $-\frac{\tau}{2} h^{(k)}(\tilde{\mathbf{z}}^{(k)})$ as its tolerance (see Appendix A and [19] for more details).
We now give some inequalities on the relationship between $\tilde{\mathbf{z}}^{(k)}$, $\mathbf{z}^{(k)}$ and $\boldsymbol{\theta}^{(k)}$ that will be useful later on (Lemma 2 from [19]).

Lemma 2.4

Consider a point $\boldsymbol{\theta}^{(k)} \in \mathbb{R}^p$ and the vectors $\tilde{\mathbf{z}}^{(k)}, \mathbf{z}^{(k)} \in \mathbb{R}^p$ defined according to the equations (2.26) and (2.27). For some $\tau \geq 0$, assume that there exist some $0 \leq \alpha_{min} \leq \alpha_{max}$ such that $\alpha_k \in [\alpha_{min}, \alpha_{max}]$ and some $\nu > 0$ that satisfies $\mathbf{M}_k \in \mathcal{M}_\nu$ for every $k \in \mathbb{N}$. Then the following inequalities hold

$$\frac{1}{2\alpha_{max}\nu} \|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 \leq -\left(1 + \frac{\tau}{2}\right) h^{(k)}(\tilde{\mathbf{z}}^{(k)}) \tag{2.28}$$

$$\frac{1}{2\alpha_{max}\nu} \|\tilde{\mathbf{z}}^{(k)} - \mathbf{z}^{(k)}\|_2^2 \leq -\frac{\tau}{2} h^{(k)}(\tilde{\mathbf{z}}^{(k)}) \tag{2.29}$$

$$\frac{C_\tau}{2\alpha_{max}\nu} \|\tilde{\mathbf{z}}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 \leq -h^{(k)}(\tilde{\mathbf{z}}^{(k)}), \tag{2.30}$$

where $C_\tau = \frac{1}{2(\tau+1)} \leq 1$.

Proof: A simple property of strong convexity (See Proposition A.4 in the Appendix) gives

$$h^{(k)}(\mathbf{z}) \geq h^{(k)}(\mathbf{z}^{(k)}) + \left(\nabla h^{(k)}(\mathbf{z}^{(k)})\right)^T (\mathbf{z} - \mathbf{z}^{(k)}) + \frac{\kappa}{2} \|\mathbf{z} - \mathbf{z}^{(k)}\|_2^2,$$

where actually $\nabla h^{(k)}(\mathbf{z}^{(k)}) = 0$ since $\mathbf{z}^{(k)}$ is the minimum of $h^{(k)}$. The constant κ is the strong convexity modulus of $h^{(k)}$, that thanks to the assumptions satisfies

$$\kappa \geq \frac{1}{\alpha_{\max}\nu},$$

Algorithm 2.2: Computation of $\tilde{\nabla}_\theta G$ at $\boldsymbol{\theta}^{(k)}$

1 **Input:** $\eta > 0$, $\eta_{\min}$, $\boldsymbol{\theta}^{(k)}$, $\boldsymbol{\theta}^{(k-1)}$

2 Set

$$\eta_k = \eta \min\{\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2, \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2, \eta_{\min}\} \leq \eta \cdot \eta_{\min} \tag{2.32}$$

3 Compute $\tilde{\mathbf{u}}^{(k)}$ such that

$$\|\nabla_{\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\|_2^2 \leq \eta_k \tag{2.33}$$

4 Compute $\tilde{\mathbf{q}}^{(k)}$, approximate solution of the linear system

$$\nabla^2_{\mathbf{uu}} E_{\mathbf{y}}(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\mathbf{q} = \nabla_{\mathbf{u}} F(\tilde{\mathbf{u}}^{(k)}) \tag{2.34}$$

so that

$$\|\nabla^2_{\mathbf{uu}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\tilde{\mathbf{q}}^{(k)} - \nabla_{\mathbf{u}} F(\tilde{\mathbf{u}}^{(k)})\|_2 \leq \eta \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2 \tag{2.35}$$

5 **Output:** $\tilde{\nabla}_\theta G(\boldsymbol{\theta}^{(k)}) = \nabla_\theta R^0(\boldsymbol{\theta}^{(k)}) - \nabla^2_{\boldsymbol{\theta}\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})^T \tilde{\mathbf{q}}^{(k)}$

so that

$$h^{(k)}(\mathbf{z}) - h^{(k)}(\mathbf{z}^{(k)}) \geq \frac{\kappa}{2}\|\mathbf{z} - \mathbf{z}^{(k)}\|_2^2 \geq \frac{1}{2\alpha_{\max}\nu}\|\mathbf{z} - \mathbf{z}^{(k)}\|_2^2. \tag{2.31}$$

Now, $h^{(k)}(\boldsymbol{\theta}^{(k)}) = 0$ with the previous property implies that

$$-h^{(k)}(\mathbf{z}^{(k)}) > \frac{1}{2\alpha_{\max}\nu}\|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2.$$

Using the condition (2.27) in the previous inequality yields the first result. Similarly, choosing $\mathbf{z} = \tilde{z}^{(k)}$ in (2.31) and applying again condition (2.27) yields the second one.
Finally, the third inequality is a direct consequence of the first two, as its just an application of the triangular inequality. In fact,

$$\|\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 \leq \left(\|\tilde{z}^{(k)} - \mathbf{z}^{(k)}\|_2 + \|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2\right)^2 \leq 2\|\tilde{z}^{(k)} - \mathbf{z}^{(k)}\|_2^2 + 2\|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2,$$

so that

$$\frac{1}{4\alpha_{\max}\nu} \leq -(1+\tau)h^{(k)}(\tilde{z}^{(k)})$$

and

$$\underbrace{\frac{1}{2(1+\tau)}}_{C_\tau}\frac{1}{2\alpha_{\max}\nu}\|\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 \leq -h^{(k)}(\tilde{z}^{(k)})$$

□

Inexact Gradient of G. Algorithm 2.2 describes, at step k, the details on how we compute the approximate gradient $\tilde{\nabla}_{\theta} G$. The fundamental ideas are those introduced in the previous section and in [72]. For each outer iteration, we need to compute the minimum of the lower level with a stopping criterion that never involves the true minimum. In particular, we want the norm of the gradient of the energy to be smaller than the distance between the last two outer iterates. The linear system (2.34) also shares the same accuracy for its solution. Here we show that, under these conditions, we can also bound the error between the approximate gradient $\tilde{\nabla}_{\theta} G(\theta^{(k)})$ and the true one $\nabla_{\theta} G(\theta^{(k)})$, again with the same tolerance.
The inexactness of the gradient can be handled with the described procedure, but it also requires to slightly modify the FB-scheme, as we are performing the gradient step with $\tilde{\nabla}_{\theta} G$ instead of $\nabla_{\theta} G$. In particular, equation (2.21) has to take into account the error in the computation of the gradient of F:

$$\mathbf{z}^{(k)} = \operatorname{prox}_{\alpha_k R^1}^{\mathbf{M}_k} \left(\theta^{(k)} - \alpha_k \mathbf{M}_k^{-1} \left(\nabla_{\theta} G(\theta^{(k)}) + \mathbf{e}^{(k)} \right) \right), \tag{2.36}$$

where

$$\mathbf{e}^{(k)} = \tilde{\nabla}_{\theta} G(\theta^{(k)}) - \nabla_{\theta} G(\theta^{(k)}) = \tilde{\nabla}_{\theta} F(\theta^{(k)}) - \nabla_{\theta} F(\theta^{(k)}), \tag{2.37}$$

is the said error. Note that $\nabla R^0(\theta^{(k)})$ can be computed exactly. The definition of the function $h^{(k)}$ in (2.25) also needs to reflect this change. With a minor abuse of notation, the new function is still denoted with $h^{(k)}$ but is defined as

$$h^{(k)}(\mathbf{z}) = \left(\nabla_{\theta} G(\theta^{(k)}) + \mathbf{e}^{(k)} \right)^T (\mathbf{z} - \theta^{(k)}) + \frac{1}{2\alpha_k} \|\mathbf{z} - \theta^{(k)}\|_{\mathbf{M}_k}^2 + R^1(\mathbf{z}) - R^1(\theta^{(k)}). \tag{2.38}$$

In this new version $h^{(k)}$ still holds its most useful properties. In particular, it still satisfies $h^{(k)}(\mathbf{z}^{(k)}), h^{(k)}(\tilde{z}^{(k)}) \leq 0$, equation (2.26) and the inequalities in Lemma 2.4.
Lemma 2.6, which is essentially Theorem 1 from [72]. However, we show that under reasonable conditions on the $(\theta^{(k)})_{k \in \mathbb{N}}$, the error $\mathbf{e}^{(k)}$ is itself bounded adaptively by $\|\theta^{(k)} - \theta^{(k-1)}\|_2$, whereas in [72] the bound is determined by a summable sequence chosen in advance. We just meed an auxiliary lemma first.

Lemma 2.5

Assume that $\mathbf{q}, \tilde{\mathbf{q}} \in \mathbb{R}^p$ are the solutions to the systems

$$\mathbf{A}\mathbf{q} = \mathbf{b}$$
$$(\mathbf{A} + \Delta\mathbf{A})\tilde{\mathbf{q}} = \mathbf{b} + \Delta\mathbf{b},$$

where $\mathbf{A}$ is invertible. Then

$$\|\mathbf{q} - \tilde{\mathbf{q}}\|_2 \leq \|\mathbf{A}^{-1}\| \left(\|\Delta\mathbf{b}\|_2 + \|\Delta\mathbf{A}\| \cdot \|\tilde{\mathbf{q}}\|_2 \right)$$

Proof: The second system reads $\mathbf{A}\tilde{\mathbf{q}} = \mathbf{b} + \mathbf{\Delta b} - \mathbf{\Delta A}\tilde{\mathbf{q}}$. Subtracting that to the first system and using basic linear operators properties we obtain

$$\begin{aligned}\mathbf{A}(\mathbf{q} - \tilde{\mathbf{q}}) &= -\mathbf{\Delta b} + \mathbf{\Delta A}\tilde{\mathbf{q}}.\\ \mathbf{q} - \tilde{\mathbf{q}} &= \mathbf{A}^{-1}\left(-\mathbf{\Delta b} + \mathbf{\Delta A}\tilde{\mathbf{q}}\right)\\ \|\mathbf{q} - \tilde{\mathbf{q}}\|_2 &\leq \|\mathbf{A}^{-1}\|\left(\|\mathbf{\Delta b}\|_2 + \|\mathbf{\Delta A}\| \cdot \|\tilde{\mathbf{q}}\|_2\right).\end{aligned}$$

□

Lemma 2.6

Under Assumptions 2.2, suppose that $\{\boldsymbol{\theta}^{(k)}\}_{k\in\mathbb{N}}$ is a bounded sequence and let $\tilde{\mathbf{u}}^{(k)}$ be computed according to Algorithm 2.2, then:

$$\|S(\boldsymbol{\theta}^{(k)}) - \tilde{\mathbf{u}}^{(k)}\|_2 \leq \frac{\eta_k}{\mu} \tag{2.39}$$

$$\|\mathbf{e}^{(k)}\|_2 = \|\nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta}^{(k)}) - \tilde{\nabla}_{\boldsymbol{\theta}} F(\boldsymbol{\theta}^{(k)})\|_2 \leq C_{\nabla F}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2. \tag{2.40}$$

Proof: By Assumption 2.2 (v) and a standard property of strongly convex functions (See Proposition A.4 in the Appendix) tells us that

$$\mu\|S(\boldsymbol{\theta}^{(k)}) - \tilde{\mathbf{u}}^{(k)}\|_2^2 \leq \left(\nabla_{\mathbf{u}} E(S(\boldsymbol{\theta}^{(k)}); \boldsymbol{\theta}^{(k)}) - \nabla_{\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\right)^T (S(\boldsymbol{\theta}^{(k)}) - \tilde{\mathbf{u}}^{(k)}).$$

The Cauchy-Schwarz inequality, the fact that $\nabla_{\mathbf{u}} E(S(\boldsymbol{\theta}^{(k)}); \boldsymbol{\theta}^{(k)}) = \mathbf{0}$ and equation (2.33) give us the first claim:

$$\|(S(\boldsymbol{\theta}^{(k)}) - \tilde{\mathbf{u}}^{(k)}\|_2 \leq \frac{1}{\mu}\|\nabla_{\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\|_2 \leq \frac{\eta_k}{\mu}. \tag{2.41}$$

Also, denoting with $\bar{\eta} = \eta \cdot \eta_{\min}$ we obtain that $\bar{\eta} \geq \eta_k$, by (2.32), and

$$\|S(\boldsymbol{\theta}^{(k)}) - \tilde{\mathbf{u}}^{(k)}\|_2 \leq \frac{\bar{\eta}}{\mu} \qquad \forall k \in \mathbb{N},$$

Now, since $\{\boldsymbol{\theta}^{(k)}\}_{k\in\mathbb{N}}$ is a bounded sequence, then there exists a compact set B such that $\{\boldsymbol{\theta}^{(k)}\}_{k\in\mathbb{N}} \in B$. Also, the solution map is continuously differentiable, which means that it is locally Lipschitz continuous, and specifically is it Lipschitz continuous in B with constant Λ_{S_B}, hence

$$\|S(\boldsymbol{\theta}^{(k)}) - S(\boldsymbol{\theta}^{(0)})\|_2 \leq \Lambda_{S_B}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(0)}\|_2. \tag{2.42}$$

So that $(S(\boldsymbol{\theta}^{(k)}))_{k\in\mathbb{N}}$ and consequently $(\tilde{\mathbf{u}}^{(k)})_{k\in\mathbb{N}}$ are also bounded. By definition, we can find an upper bound $M_{\tilde{\mathbf{u}}} > 0$ so that $\|\tilde{\mathbf{u}}^{(k)}\| \leq M_{\tilde{\mathbf{u}}}$ for every $k \in \mathbb{N}$. In turn, thanks to the continuity of $\nabla^2_{\boldsymbol{\theta}\mathbf{u}} E(\cdot;\cdot)$, we have that $\|\nabla^2_{\boldsymbol{\theta}\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\|_2 \leq M_{\nabla^2_{\boldsymbol{\theta}\mathbf{u}} E}$ for some $M_{\nabla^2_{\boldsymbol{\theta}\mathbf{u}} E} > 0$. Now, let $\mathbf{q}^{(k)}$ be an exact solution of the linear system in (2.34), i.e.,

$$\nabla^2_{\mathbf{uu}} E(S(\boldsymbol{\theta}^{(k)}); \boldsymbol{\theta}^{(k)})\mathbf{q}^{(k)} = \nabla_{\mathbf{u}} F(S(\boldsymbol{\theta}^{(k)}))$$

Now, since $\nabla^2_{\mathbf{uu}} E(\cdot; \boldsymbol{\theta}^{(k)})$ is continuous by assumption and $\|S(\boldsymbol{\theta}^{(k)}) - S(\boldsymbol{\theta}^{(0)})\|$ is bounded, the same arguments can be repeated to show that $\mathbf{q}^{(k)}$ is bounded as well. Let $M_{\mathbf{q}} > 0$ be

a value such that $\|\mathbf{q}^{(k)}\| \leq M_{\mathbf{q}}$ for every $k \in \mathbb{N}$.
At the same time, $\tilde{\mathbf{q}}^{(k)}$, is an approximate solution of the system:

$$\boldsymbol{\nabla}^2_{\mathbf{uu}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\mathbf{q} = \boldsymbol{\nabla}_{\mathbf{u}} F(\tilde{\mathbf{u}}^{(k)}).$$

Hence, it can be seen as the exact solution to the perturbed system

$$\begin{aligned}\boldsymbol{\nabla}^2_{\mathbf{uu}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\mathbf{q} &= \boldsymbol{\nabla}_{\mathbf{u}} F(\tilde{\mathbf{u}}^{(k)}) + \mathbf{r}^{(k)} \\ \Rightarrow \tilde{\mathbf{q}}^{(k)} &= \boldsymbol{\nabla}^2_{\mathbf{uu}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})^{-1}(\boldsymbol{\nabla}_{\mathbf{u}} F(\tilde{\mathbf{u}}^{(k)}) + \mathbf{r}^{(k)}).\end{aligned}$$

In particular, the hessian $\boldsymbol{\nabla}^2_{\mathbf{uu}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})$ is invertible because the function $E(\cdot; \boldsymbol{\theta}^{(k)})$ is strongly convex. Thanks to equation (2.35), the perturbation/residual satisfies

$$\|\mathbf{r}^{(k)}\|_2 \leq \eta \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2.$$

Again, exploiting the continuity of the hessian and of the gradient of F we deduce that the sequence $\{\tilde{\mathbf{q}}^{(k)}\}_{k\in\mathbb{N}}$ is also bounded by some constant denoted with $M_{\tilde{\mathbf{q}}} > 0$.
We can exploit the previous Lemma by choosing

$$\begin{aligned}\mathbf{A} &= \boldsymbol{\nabla}^2_{\mathbf{uu}} E(S(\boldsymbol{\theta}^{(k)}); \boldsymbol{\theta}^{(k)}) \\ \boldsymbol{\Delta}\mathbf{A} &= \boldsymbol{\nabla}^2_{\mathbf{uu}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)}) - \boldsymbol{\nabla}^2_{\mathbf{uu}} E(S(\boldsymbol{\theta}^{(k)}); \boldsymbol{\theta}^{(k)}) \\ \mathbf{b} &= F(S(\boldsymbol{\theta}^{(k)})) \\ \boldsymbol{\Delta}\mathbf{b} &= F(\tilde{\mathbf{u}}^{(k)}) - F(S(\boldsymbol{\theta}^{(k)})) + \mathbf{e}^{(k)},\end{aligned}$$

to show that $\|\mathbf{q}^{(k)} - \tilde{\mathbf{q}}^{(k)}\|_2$ is also bounded by $\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2$. Indeed

$$\begin{aligned}\|\mathbf{q}^{(k)} - \tilde{\mathbf{q}}^{(k)}\|_2 &\leq \|\boldsymbol{\nabla}^2_{\mathbf{uu}} E(S(\boldsymbol{\theta}^{(k)}); \boldsymbol{\theta}^{(k)})^{-1}\| \cdot \Big(\|\boldsymbol{\nabla}_{\mathbf{u}} F(\tilde{\mathbf{u}}^{(k)}) - \boldsymbol{\nabla}_{\mathbf{u}} F(S(\boldsymbol{\theta}^{(k)}))\|_2 + \|\mathbf{e}^{(k)}\|_2 \\ &\quad + \|\boldsymbol{\nabla}^2_{\mathbf{uu}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)}) - \boldsymbol{\nabla}^2_{\mathbf{uu}} E(S(\boldsymbol{\theta}^{(k)}); \boldsymbol{\theta}^{(k)})\| \cdot \|\tilde{\mathbf{q}}^{(k)}\|_2\Big) \\ &\leq \frac{1}{\mu}\Big(\Lambda_{\nabla F}\|\tilde{\mathbf{u}}^{(k)} - \mathbf{u}^{(k)}\|_2 + \|\mathbf{e}^{(k)}\|_2 + M_{\tilde{\mathbf{q}}}\Lambda_{\nabla^2_{uu}E}\|\tilde{\mathbf{u}}^{(k)} - \tilde{\mathbf{u}}\|_2\Big) \\ &\leq \frac{1}{\mu}\Big(\|\mathbf{e}^{(k)}\|_2 + (\Lambda_{\nabla F} + M_{\tilde{\mathbf{q}}}\Lambda_{\nabla^2_{uu}E}))\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2\Big) \\ &\leq \frac{\eta}{\mu}\Big(1 + \Lambda_{\nabla F} + M_{\tilde{\mathbf{q}}}\Lambda_{\nabla^2_{uu}E}\Big)\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2\end{aligned}$$

where, again, in the second inequality we used the Lipschitz continuity of the functions involved, that holds by assumption.
Finally, we can conclude that

$$\begin{aligned}\|\boldsymbol{\nabla}_{\boldsymbol{\theta}} F(\boldsymbol{\theta}^{(k)}) - \tilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} F(\boldsymbol{\theta}^{(k)})\|_2 &= \|\boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(S(\boldsymbol{\theta}^{(k)}); \boldsymbol{\theta}^{(k)})^T\mathbf{q}^{(k)} - \boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})^T\tilde{\mathbf{q}}^{(k)} \\ &\quad + \boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})^T\mathbf{q}^{(k)} - \boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})^T\mathbf{q}^{(k)}\|_2 \\ &\leq \|\boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(S(\boldsymbol{\theta}^{(k)}); \boldsymbol{\theta}^{(k)})^T\mathbf{q}^{(k)} - \boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})^T\mathbf{q}^{(k)}\|_2 \\ &\quad + \|\boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})^T\mathbf{q}^{(k)} - \boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})^T\tilde{\mathbf{q}}^{(k)}\|_2 \\ &\leq \|\boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(S(\boldsymbol{\theta}^{(k)}); \boldsymbol{\theta}^{(k)}) - \boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\| \cdot \|\mathbf{q}^{(k)}\|_2 \\ &\quad + \|\boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\| \cdot \|\mathbf{q}^{(k)} - \tilde{\mathbf{q}}^{(k)}\|_2. \\ &\leq \|\boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(S(\boldsymbol{\theta}^{(k)}); \boldsymbol{\theta}^{(k)}) - \boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\| \cdot \|\mathbf{q}^{(k)}\|_2 \\ &\quad + \underbrace{M_{\boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}}E}\frac{\eta}{\mu}\Big(1 + \Lambda_{\nabla F} + M_{\tilde{\mathbf{q}}}\Lambda_{\nabla^2_{uu}E}\Big)}_{C_1}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2.\end{aligned}$$

Once again, leaning on the assumed Lipschitz continuity we have that

$$\|\nabla^2_{\theta\mathbf{u}} E(S(\boldsymbol{\theta}^{(k)}); \boldsymbol{\theta}^{(k)}) - \nabla^2_{\theta\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\| \cdot \|\mathbf{q}^{(k)}\|_2 \leq M_{\mathbf{q}} \|S(\boldsymbol{\theta}^{(k)} - \tilde{\mathbf{u}}^{(k)})\|_2 \leq M_{\mathbf{q}} \frac{\eta_k}{\mu}.$$

Thus, we can conclude the claim hold for $C_{\nabla F} = C_1 + M_{\mathbf{q}}\bar{\eta}$ by recalling the condition (2.32) and combining it with the last findings. □

With these premises, we can illustrate in Algorithm 2.3 the full iterative FB-like scheme that includes the update of $\boldsymbol{\theta}^{(k)}$. While the parameters α_k and $\mathbf{M}_k$ are left as a mostly free choice in the proposed method, all the heavy weight is placed on the steplength λ_k.

The Linesearch Procedure. The steplength λ_k in (2.22) can be used to achieve the sufficient decrease of the objective function in the KL framework. Our method employs a backtracking loop until a sufficient decrease is verified for a tailored merit function. In fact, $\tilde{\mathbf{u}}^{(k)}$ being an approximation of the true minimum means that the objective function is computed inexactly as well and subsequently that the sufficient decrease condition cannot feature the true loss value. However, we are able to circumvent this issue thanks to condition (2.33) as we are able to prove that, at each iteration, we can compute an approximated value

$$\tilde{L}^{(k)} = F(\tilde{\mathbf{u}}^{(k)}) + R^0(\boldsymbol{\theta}^{(k)}) + R^1(\boldsymbol{\theta}^{(k)}) = G(\tilde{\mathbf{u}}^{(k)}, \boldsymbol{\theta}^{(k)}) + R^1(\boldsymbol{\theta}^{(k)}), \tag{2.43}$$

where $\tilde{\mathbf{u}}^{(k)}$ satisfies

$$\|\nabla_{\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\|_2 \leq \eta_k, \tag{2.44}$$

with $\eta_k = \eta \cdot \min\{\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2, \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2, \eta_{\min}\}$, and such that $\tilde{L}^{(k)}$ is sufficiently close to the true value:

$$|\tilde{L}^{(k)} - L(\boldsymbol{\theta}^{(k)})| \leq \rho \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2, \tag{2.45}$$

where $\rho > 0$ is a fixed constant and not an hyperparameter as we will show it depends on η, the constant Λ_F and the strong-convexity modulus μ. Essentially, from equation (2.45) we have that

$$L(\boldsymbol{\theta}^{(k)}) \leq \tilde{L}^{(k)} + \rho \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 \tag{2.46}$$

$$\tilde{L}^{(k)} \leq L(\boldsymbol{\theta}^{(k)}) + \rho \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2. \tag{2.47}$$

The next Lemma proves that the algorithm does indeed satisfy this condition.

Lemma 2.7

Under Assumptions 2.2 (iv) and (v), if $\boldsymbol{\theta}^{(k)} \in \operatorname{dom} R^1$ and $\tilde{\mathbf{u}}^{(k)}$ satisfies

$$\|\nabla_{\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\| \leq \eta_k,$$

for some $\eta_k > 0$, then

$$|F(S(\boldsymbol{\theta}^{(k)})) - F(\tilde{\mathbf{u}}^{(k)})| \leq \Lambda_F \frac{\eta_k}{\mu}. \tag{2.48}$$

Proof: By definition of Lipschitz continuity we have

$$|F(S(\boldsymbol{\theta}^{(k)})) - F(\tilde{\mathbf{u}}^{(k)})| \leq \Lambda_F \|S(\boldsymbol{\theta}^{(k)}) - \tilde{\mathbf{u}}^{(k)}\|_2.$$

Combining this inequality with the one in (2.39) we directly obtain the claim. □

In (2.45) we have that $\tilde{L}^{(k)} = F(\tilde{\mathbf{u}}^{(k)}) + R^0(\boldsymbol{\theta}^{(k)}) + R^1(\boldsymbol{\theta}^{(k)})$, with the second term being exact, so that $\tilde{L}^{(k)} - L(\boldsymbol{\theta}^{(k)}) = F(\tilde{\mathbf{u}}^{(k)}) - F(\boldsymbol{\theta}^{(k)})$. Hence, in light of (2.48) we have that (2.45) indeed holds with

$$\rho = \eta \frac{\Lambda_F}{\mu}.$$

Under this condition, the merit function that we use is

$$\Phi^{(k)} = \tilde{L}^{(k)} + \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2. \tag{2.49}$$

which is decreasing thanks to the While loop condition and the update rules. In fact we have that $\Phi^{(k+1)} \leq \Phi^{(k)} + \lambda_k \mathbf{d}^{(k)} h^{(k)}(\tilde{z}^{(k)}) < \Phi^{(k)}$.

The following Lemma guarantees, under a suitable choice of the hyperparameters, that Algorithm 2.3 is well defined, i.e., the condition of the While loop is verified in a finite number of steps.

Lemma 2.8

Under Assumptions 2.2 (iii) and (iv), given the points $\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k-1)} \in \operatorname{dom} R^1$ and $\tilde{\boldsymbol{z}}^{(k)} \in \operatorname{dom} R$ such that

$$h^{(k)}(\tilde{\boldsymbol{z}}^{(k)}) - h^{(k)}(\mathbf{z}^{(k)}) \leq -\frac{\tau}{2} h^{(k)}(\tilde{\boldsymbol{z}}^{(k)}),$$

then, if $\gamma > 2\rho$, we have that the While *loop in Algorithm 2.3 ends in a finite numbers of steps. More precisely, at most*

$$m = \left\lceil \max\left\{0, \log_\delta\left(\frac{1-\sigma}{\xi}\right)\right\}\right\rceil \tag{2.50}$$

where

$$\xi = \frac{2\nu\alpha_{max}}{C_\tau}\left(\frac{\Lambda_{\nabla F}}{2} + \rho + \frac{\gamma}{2} + \frac{C_{\nabla F}}{2(\gamma - 2\rho)}\right),$$

so that $\lambda_k \geq \lambda_{min} = \delta^m$.

Proof: Pick $\lambda \in (0, 1]$ and denote with $\tilde{L}_\lambda$ any value such that

$$\tilde{L}_\lambda \leq L(\boldsymbol{\theta}^{(k)} + \lambda \mathbf{d}^{(k)}) + \rho\lambda^2 \|\mathbf{d}^{(k)}\|_2^2,$$

with $\mathbf{d}^{(k)} = \tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{\theta}^{(k)}$. In light of this we have that

$$\begin{aligned}\tilde{L}_\lambda + \frac{\gamma}{2}\lambda^2\|\mathbf{d}^{(k)}\|_2^2 &\leq L(\boldsymbol{\theta}^{(k)} + \lambda \mathbf{d}^{(k)}) + \left(\rho + \frac{\gamma}{2}\right)\lambda^2\|\mathbf{d}^{(k)}\|_2^2. \\ &= G(\boldsymbol{\theta}^{(k)} + \lambda \mathbf{d}^{(k)}) + R^1((1-\lambda)\boldsymbol{\theta}^{(k)} + \lambda\tilde{\boldsymbol{z}}^{(k)}) + \left(\rho + \frac{\gamma}{2}\right)\lambda\|\mathbf{d}^{(k)}\|_2^2.\end{aligned}$$

Since G has a Lipschitz continuous gradient we can use the Descent Lemma [11], and we can also use the convexity of R^1 to obtain

$$\begin{aligned}\tilde{L}_\lambda + \frac{\gamma}{2}\lambda^2\|\mathbf{d}^{(k)}\|_2^2 &\leq G(\boldsymbol{\theta}^{(k)}) + \lambda\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)})^T\mathbf{d}^{(k)} + \left(\frac{\Lambda_{\nabla G}}{2} + \rho + \frac{\gamma}{2}\right)\lambda^2\|\mathbf{d}^{(k)}\|_2^2 \\ &\quad + (1-\lambda)R^1(\boldsymbol{\theta}^{(k)}) + \lambda R^1(\tilde{\boldsymbol{z}}^{(k)}) + \lambda(\mathbf{e}^{(k)})^T\mathbf{d}^{(k)} - \lambda(\mathbf{e}^{(k)})^T\mathbf{d}^{(k)} \\ &= L(\boldsymbol{\theta}^{(k)}) + \lambda\underbrace{\left(R^1(\tilde{\boldsymbol{z}}^{(k)}) - R^1(\boldsymbol{\theta}^{(k)}) + (\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)}) + \mathbf{e}^{(k)})^T\mathbf{d}^{(k)}\right)}_{\leq \Delta_k} \\ &\quad + \left(\frac{\Lambda_{\nabla G}}{2} + \rho + \frac{\gamma}{2}\right)\lambda^2\|\mathbf{d}^{(k)}\|_2^2 - \lambda(\mathbf{e}^{(k)})^T\mathbf{d}^{(k)}\end{aligned}$$

Following up with equation (2.46) we arrive at

$$\begin{aligned}\tilde{L}_\lambda + \frac{\gamma}{2}\lambda^2\|\mathbf{d}^{(k)}\|_2^2 &\leq \tilde{L}^{(k)} + \rho\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 + \lambda\Delta_k + \left(\frac{\Lambda_{\nabla G}}{2} + \rho + \frac{\gamma}{2}\right)\lambda^2\|\mathbf{d}^{(k)}\|_2^2 \\ &\quad - \lambda(\mathbf{e}^{(k)})^T\mathbf{d}^{(k)},\end{aligned}$$

where we recall that $\Delta_k = h^{(k)}(\tilde{\boldsymbol{z}}^{(k)}) < 0$. Now, if $c_1, c_2 > 0$ and $\mathbf{a}, \mathbf{b} \in \mathbb{R}^p$, then we have the basi inequality

$$c_1 c_2 \mathbf{a}^T\mathbf{b} \leq \frac{c_1^2}{2}\|\mathbf{a}\|_2^2 + \frac{c_2^2}{2}\|\mathbf{b}\|_2^2.$$

Using the previous inequality with $c_1 = \frac{\sqrt{\gamma - 2\rho}}{C_{\nabla F}}$, $c_2 = \lambda c_1^{-1}$, $\mathbf{a} = \mathbf{e}^{(k)}$ and $\mathbf{b} = \mathbf{d}^{(k)}$, as well as inequality (2.40) we obtain

$$\begin{aligned}\tilde{L}_\lambda + \frac{\gamma}{2}\lambda^2\|\mathbf{d}^{(k)}\|_2^2 &\leq \tilde{L}^{(k)} + \rho\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 + \lambda\Delta_k \\ &\quad + \left(\frac{\Lambda_{\nabla F}}{2} + \rho + \frac{\gamma}{2} + \frac{C_{\nabla F}}{2(\gamma - 2\rho)}\right)\lambda^2\|\mathbf{d}^{(k)}\|_2^2 + \frac{\gamma - 2\rho}{2C_{\nabla F}^2}\|\mathbf{e}^{(k)}\|_2^2 \\ &\leq \tilde{L}^{(k)} + \lambda\Delta_k + \left(\frac{\Lambda_{\nabla F}}{2} + \rho + \frac{\gamma}{2} + \frac{C_{\nabla F}}{2(\gamma - 2\rho)}\right)\lambda^2\|\mathbf{d}^{(k)}\|_2^2 \\ &\quad + \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2\end{aligned}$$

Finally, the definition of $\mathbf{d}^{(k)}$ and inequality (2.30) yield:

$$\begin{aligned}\tilde{L}_\lambda + \frac{\gamma}{2}\lambda^2\|\mathbf{d}^{(k)}\|_2^2 \leq \tilde{L}^{(k)} + \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 + \lambda\Delta_k \\ \underbrace{- \frac{2\nu\alpha_{\max}}{C_\tau}\left(\frac{\Lambda_{\nabla F}}{2} + \rho + \frac{\gamma}{2} + \frac{C_{\nabla F}}{2(\gamma - 2\rho)}\right)}_{\xi}\lambda^2\Delta_k \\ = \tilde{L}^{(k)} + \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 + \lambda(1 - \lambda\xi)\Delta_k.\end{aligned}$$

The While loop terminates if

$$\begin{aligned}\tilde{L}^{(k)} + \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 + \sigma\lambda\Delta_k \geq \tilde{L}^{(k)} + \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 + \lambda(1 - \lambda\xi)\Delta_k \\ \geq \tilde{L}_\lambda + \frac{\gamma}{2}\lambda^2\|\mathbf{d}^{(k)}\|_2^2,\end{aligned}$$

In the algorithm we have that that $\lambda = \delta^m$, which means that the loop ends, at worst, as soon as m hits the smallest positive integer such that

$$\sigma\delta^m\Delta_k \geq \delta^m(1 - \delta^m\xi)\Delta_k \tag{2.51}$$

$$\Rightarrow 0 < \delta^m \leq \frac{1 - \sigma}{\xi}. \tag{2.52}$$

□

Corollary 2.9

Under the same assumptions as the previous Lemma, we also obtain the following inequalities:

$$\tilde{L}^{(k+1)} + \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\|_2^2 \leq \tilde{L}^{(k)} + \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 + \sigma\delta^m\Delta_k \tag{2.53}$$

$$\tilde{L}^{(k+1)} + \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\|_2^2 \leq \tilde{L}_{\mathbf{z}}^{(k+1)} + \frac{\gamma}{2}\|\tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 \tag{2.54}$$

$$\leq L(\tilde{\boldsymbol{z}}^{(k)}) + \frac{\gamma + 2\rho}{2}\|\tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2, \tag{2.55}$$

where m is the smallest integer such that (2.52) holds.

Proof: This corollary was indirectly proves in the previous Lemma as a consequence of the linesearch procedure and of the subsequent updating rule in Algorithm 2.3. □

Before diving into the convergence analysis, let us discuss the results shown here with other similar methods in literature.

Our assumptions on the functions that compose the bilevel problem are very similar to those in [92]. In particular, in [92] the authors find the approximation $\tilde{\mathbf{u}}^{(k)}$ by applying one step of gradient descent on $E(\cdot; \boldsymbol{\theta}^{(k)})$, and then compute the product $\tilde{\mathbf{q}}^{(k)}$ in an inexact relaxed way. Therefore, in contrast to our proposed method, they never fully solve the lower level problem. Moreover, the algorithms proposed in [92] require the knowledge of the Lipschitz constants involved. In fact convergence is

assured when all the stepsizes involved satisfy certain conditions. These bounds, however, are seldom verifiable in practice, i.e., the hyperparameters are chosen by trial and error, as the authors confirm.
The scheme proposed in [72] also shares a good amount of similarities, but also notable differences. In [72] the same stragegy to compute the inexact gradient of the loss function is used, but as we have discussed already, the error in this computation is controlled with a summable sequence, while in our approach we use an adaptive strategy. Furthermore, the stepsize in [72] is chosen in an adaptive strategy, that is shown to work empirically but no formal analysis of the algorithm with said technique is available in the work. Last, but not least, the algorithm proposed in [72] is not shown to converge, but rather that the limit points of the sequence of the iterates satisfy the stationary point condition. By contrast, we are going to show that our algorithm does indeed converge to a stationary point of the loss function. In particular, we lean on a linesearch procedure which we are able to carry out even though we can only have inexact values of the objective function and its gradient. Moreover, we also include a non-differentiable term in the upper level as well as an inexact computation of its proximal operator.

Algorithm 2.3: Bilevel Line-Search Algorithm

Input: initial iterates $\boldsymbol{\theta}^{(-1)}$ and $\boldsymbol{\theta}^{(0)}$, hyperparameters $\eta, \eta_{\min}, \gamma > 2\rho, \rho = \eta\Lambda_F/\mu, \tau > 0$, $\sigma, \delta \in (0,1)$, $0 < \alpha_{\min} \leq \alpha_{\max}$, $\mu \geq 1$, MaxIter ≥ 1.

Set $\eta_1 = \eta \min\{\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^{(-1)}\|_2^2, \|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^{(-1)}\|_2, \eta_{\min}\}$.

Compute $\tilde{\mathbf{u}}^{(0)}$ such that $\|\boldsymbol{\nabla}_{\mathbf{u}} E(\tilde{\mathbf{u}}^{(0)}; \boldsymbol{\theta}^{(0)})\| \leq \eta_0$.

Set $\tilde{L}^{(0)} = F(\tilde{\mathbf{u}}^{(0)}) + R^0(\boldsymbol{\theta}^{(0)}) + R^1(\boldsymbol{\theta}^{(0)})$.

for $k = 0, 1, \ldots,$ MaxIter **do**

  Compute $\tilde{\mathbf{q}}^{(k)}$ such that $\|\boldsymbol{\nabla}^2_{\mathbf{uu}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})\tilde{\mathbf{q}}^{(k)} - \boldsymbol{\nabla}_{\mathbf{u}} F(\tilde{\mathbf{u}}^{(k)})\|_2 \leq \eta \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2$.

  Compute $\tilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)}) = \boldsymbol{\nabla}_{\boldsymbol{\theta}} R^0(\boldsymbol{\theta}^{(k)}) - \boldsymbol{\nabla}^2_{\boldsymbol{\theta}\mathbf{u}} E(\tilde{\mathbf{u}}^{(k)}; \boldsymbol{\theta}^{(k)})^T \tilde{\mathbf{q}}^{(k)}$.

  Choose $\alpha_k \in [\alpha_{\min}, \alpha_{\max}]$ and $\mathbf{M}_k \in \mathcal{M}_\mu$.

  Compute $\tilde{z}^{(k)} \simeq \operatorname{prox}^{\mathbf{M}_k}_{\alpha_k R^1}\left(\boldsymbol{\theta}^{(k)} - \alpha_k \mathbf{M}_k^{-1} \tilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)})\right)$ so that

$$h^{(k)}(\tilde{z}^{(k)}) - h^{(k)}(\mathbf{z}^{(k)}) \leq -\frac{\tau}{2} h^{(k)}(\tilde{z}^{(k)}).$$

  Set $\Delta_k = h^{(k)}(\tilde{z}^{(k)})$, $\mathbf{d}^{(k)} = \tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}$ and

$$\eta^{\mathbf{z}}_{k+1} = \eta \min\{\|\mathbf{d}^{(k)}\|_2^2, \|\mathbf{d}^{(k)}\|_2, \eta_{\min}\}.$$

  Compute $\tilde{\mathbf{u}}^{(k+1)}_{\mathbf{z}}$ such that

$$\|\boldsymbol{\nabla}_{\mathbf{u}} E(\tilde{\mathbf{u}}^{(k+1)}_{\mathbf{z}}; \tilde{z}^{(k)})\| \leq \eta^{\mathbf{z}}_{k+1}.$$

  Set $\tilde{L}^{(k+1)}_{\mathbf{z}} = F(\tilde{\mathbf{u}}^{(k+1)}_{\mathbf{z}}) + R^0(\tilde{z}^{(k)}) + R^1(\tilde{z}^{(k)})$, $\tilde{L}^{(k+1)}_{\lambda} = \tilde{L}^{(k+1)}_{\mathbf{z}}$, $\lambda_k = 1$ and $m = 0$.

  while

$$\min\{\tilde{L}^{(k+1)}_{\lambda} + \frac{\gamma}{2}\lambda_k^2 \|\mathbf{d}^{(k)}\|_2^2,\ \tilde{L}^{(k+1)}_{\mathbf{z}} + \frac{\gamma}{2}\|\mathbf{d}^{(k)}\|_2^2\} > \tilde{L}^{(k)} + \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 + \sigma\lambda_k\Delta_k$$

  do

    Update $m = m + 1$ and $\lambda_k = \delta^m$.

    Set $\eta^{\lambda}_{k+1} = \eta \min\{\lambda_k^2 \|\mathbf{d}^{(k)}\|_2^2, \lambda_k \|\mathbf{d}^{(k)}\|_2, \eta_{\min}\}$.

    Compute $\tilde{\mathbf{u}}^{(k+1)}_{\lambda}$ such that

$$\|\boldsymbol{\nabla}_{\mathbf{u}} E(\tilde{\mathbf{u}}^{(k+1)}_{\lambda}; \boldsymbol{\theta}^{(k)} + \lambda_k \mathbf{d}^{(k)})\|_2 \leq \eta^{\lambda}_{k+1}.$$

    Update $\tilde{L}^{(k+1)}_{\lambda} = F(\tilde{\mathbf{u}}^{(k+1)}_{\lambda}) + R^0(\boldsymbol{\theta}^{(k)} + \lambda_k \mathbf{d}^{(k)}) + R^1(\boldsymbol{\theta}^{(k)} + \lambda_k \mathbf{d}^{(k)})$.

  end

  if $\tilde{L}^{(k+1)}_{\mathbf{z}} + \frac{\gamma}{2}\|\mathbf{d}^{(k)}\|_2^2 \leq \tilde{L}^{(k+1)}_{\lambda} + \frac{\gamma}{2}\lambda_k^2 \|\mathbf{d}^{(k)}\|_2^2$ **then**

    Update $\boldsymbol{\theta}^{(k+1)} = \tilde{z}^{(k)}$.

    Update $\tilde{L}^{(k+1)} = \tilde{L}^{(k+1)}_{\mathbf{z}}$, $\eta_{k+1} = \eta^{\mathbf{z}}_{k+1}$ and $\tilde{\mathbf{u}}^{(k+1)} = \tilde{\mathbf{u}}^{(k+1)}_{\mathbf{z}}$.

  else

    Update $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \lambda_k \mathbf{d}^{(k)}$.

    Update $\tilde{L}^{(k+1)} = \tilde{L}^{(k+1)}_{\lambda}$, $\eta_{k+1} = \eta^{\lambda}_{k+1}$ and $\tilde{\mathbf{u}}^{(k+1)} = \tilde{\mathbf{u}}^{(k+1)}_{\lambda}$.

  end

end

2.3.2 Convergence

As we have already discussed, Algorithm 2.3 follows the two main steps of a Forward-Backward scheme. Whenever the proximal operator has to be computed inexactly, the convergence analysis becomes rather complicated. One efficient strategy to obtain the desired results is working with the *Kurdyka-Łojasiewicz* (KL) Property [15]

Definition 2.10. A proper, lower-semicontinuous function $f : \mathbb{R}^p \to \bar{\mathbb{R}}$ satisfies the *Kurdyka-Łojasiewiecz property* at a point $\bar{\boldsymbol{\theta}} \in \operatorname{dom} \partial f$ if there exist

- a constant $c \in (0, +\infty]$
- a neighbourhood B of $\bar{\boldsymbol{\theta}}$
- a continuous concave function $\varphi\colon [0, c) \to [0, +\infty)$, such that $\varphi \in C^1(0, c)$, $\varphi(0) = 0$, $\varphi' > 0$ in $(0, c)$ and

$$\varphi'(f(\boldsymbol{\theta}) - f(\bar{\boldsymbol{\theta}})) \cdot \operatorname{dist}(0, \partial f(\boldsymbol{\theta})) \geq 1 \quad \forall \boldsymbol{\theta} \in B \cup [f(\bar{\boldsymbol{\theta}}) < f < f(\bar{\boldsymbol{\theta}}) + c].$$

In particular, if f satisfies the property for every point in $\operatorname{dom} \partial f$ it is called a *Kurdyka-Łojasiewicz function* .

While this condition, at first glance, does not provide meaningful insights, it is especially useful in the convergence analysis. In fact, it is satisfied by the vast majority of objective functions that come up in imaging applications, and, at the same time, it excludes some of the pathological cases, e.g. the C^∞ mexican hat. Furthermore, it also provides a specific path that can be followed to achieve convergence [2, 67].

Theorem 2.11

If f is a KL function and $\{\boldsymbol{\theta}^{(k)}\}_{k\in\mathbb{N}}$ is a sequence that satisfies:

- $\boldsymbol{\theta}^{(k)}{}_{k\in\mathbb{N}}$ *is bounded*
- *Sufficient Decrease: there exist $a > 0$ such that*

$$f(\boldsymbol{\theta}^{(k)}) \leq f(\boldsymbol{\theta}^{(k-1)}) - a\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 \quad \forall k \in \mathbb{N}$$

- *Relative Error: there exist $b > 0$ and $\boldsymbol{\kappa}^{(k)} \in \partial f(\boldsymbol{\theta}^{(k)})$ such that*

$$\|\boldsymbol{\kappa}^{(k)}\|_2 \leq b\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2 \quad \forall k \in \mathbb{N}$$

- *Continuity: there exist a subsequence $\{\boldsymbol{\theta}^{(k_j)}\}_{j\in\mathbb{N}}$ and a point $\bar{\boldsymbol{\theta}} \in \mathbb{R}^p$ such that*

$$\lim_{j\to+\infty} \boldsymbol{\theta}^{(k)} = \bar{\boldsymbol{\theta}}, \quad \lim_{j\to+\infty} f(\boldsymbol{\theta}^{(k)}) = f(\bar{\boldsymbol{\theta}})$$

then $\{\boldsymbol{\theta}^{(k)}\}_{k\in\mathbb{N}}$ *has finite length, that is*

$$\sum_{k=0}^{+\infty} \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\| < +\infty.$$

As a consequence, it converges to a stationary point $\bar{\boldsymbol{\theta}}$ *of* f*.*

This framework has been extended in [22], in order to deal with a number of variations of the FB scheme, for instance by including an inertial term or, as in our case, a possible error in the gradient of the differentiable part of the objective function. Here we are going to work with an adaptation of Theorem 5 of [22]:

Theorem 2.12

Assume that $f : \mathbb{R}^p \times \mathbb{R}^m \to \bar{\mathbb{R}}$ *is a proper, lower semicontinuous KL function. Given the sequences* $\{(\boldsymbol{\theta}^{(k)}, \mathbf{p}_k)\}_{k\in\mathbb{N}} \subset \mathbb{R}^p \times \mathbb{R}^m$*,* $\{\Phi_k\}_{k\in\mathbb{N}}, \{r_k\}_{k\in\mathbb{N}} \subset \mathbb{R}$*,* $\{\boldsymbol{\zeta}^{(k)}\}_{k\in\mathbb{N}} \subset \mathbb{R}^p$ *and* $\{t_k\}_{k\in\mathbb{N}} \subset \mathbb{R}_+$*, if they satisfy the conditions*

- *[H0]: The sequence* $\{(\boldsymbol{\theta}^{(k)}, \mathbf{p}_k)\}_{k\in\mathbb{N}}$ *is bounded and* $\{\mathbf{p}_k\}_{k\in\mathbb{N}}$ *converges.*
- *[H1]:* $\{\Phi_k\}_{k\in\mathbb{N}} \subset \mathbb{R}$ *is bounded from below and admits a* $a > 0$ *such that*

$$\Phi_{k+1} + at_k^2 \leq \Phi_k \quad \forall k \geq 0.$$

- *[H2]:* $\lim_{k\to+\infty} r_k = 0$ *and*

$$\Phi_{k+1} \leq f(\boldsymbol{\zeta}^{(k)}, \mathbf{p}_k) \leq \Phi_k + r_k \quad \forall k \geq 0.$$

- *[H3]: For every* $k \geq 0$*, there exists a subgradient* $\boldsymbol{\kappa}^{(k)} \in \partial f(\boldsymbol{\zeta}^{(k)}, \mathbf{p}_k)$ *such that*

$$\|\boldsymbol{\kappa}^{(k)}\|_2 \leq b \sum_{i\in\mathcal{I}} \xi_i t_{k+1-i} \quad \forall k \geq 0,$$

  where $b > 0$*,* $\mathcal{I} \subset \mathbb{Z}$ *is non-empty,* $\xi_i \geq 0$ *for every* $i \in \mathcal{I}$ *with* $\sum_{i\in\mathcal{I}} \xi_i = 1$ *and* $t_k = 0$ *for* $k \leq 0$*.*

- *[H4]:*

$$\{(\boldsymbol{\theta}^{(k_j)}, \mathbf{p}_{k_j})\}_{j\in\mathbb{N}} \xrightarrow{j\to+\infty} (\bar{\boldsymbol{\theta}}, \bar{\mathbf{p}}) \Longrightarrow \begin{cases} \lim_{j\to+\infty} \|\boldsymbol{\zeta}^{(k_j)} - \boldsymbol{\theta}^{(k_j)}\|_2 = 0 \text{ and} \\ \lim_{j\to+\infty} f(\boldsymbol{\zeta}^{(k_j)}, \mathbf{p}_{k_j}) = f(\bar{\boldsymbol{\theta}}, \bar{\mathbf{p}}). \end{cases}$$

- *[H5]: There exists a* $w > 0$ *such that*

$$\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\|_2 \leq wt_k \quad \forall k \geq 0.$$

Then the sequence $\{(\boldsymbol{\theta}^{(k)}, \mathbf{p}_k)\}_{k\in\mathbb{N}}$ *converges to a stationary point* $(\bar{\boldsymbol{\theta}}, \bar{\mathbf{p}})$ *of* f*.*

Whenever this theorem holds, we can easily obtain the convergence of $\{\boldsymbol{\theta}^{(k)}\}_{k\in\mathbb{N}}$ to the stationary point. Indeed, since $(\bar{\boldsymbol{\theta}}, \bar{\mathbf{p}})$ is stationary for f, then $(\mathbf{0}, 0) \in \partial f(\bar{\boldsymbol{\theta}}, \bar{\mathbf{p}}) = \partial L(\bar{\boldsymbol{\theta}}) \times \{\bar{\mathbf{p}}\}$, where we have used the separability of f to split the subdifferential. We can then conclude that $\bar{\boldsymbol{\theta}}$ is stationary for L by definition, i.e., $\mathbf{0} \in \partial L(\bar{\boldsymbol{\theta}})$.
In the remainder of this section we are going to prove that the hypotheses of this theorem hold in our setting. In particular, we recall that by Assumption 2.2 (viii)

$$f(\boldsymbol{\theta}, p) = L(\boldsymbol{\theta}) + \frac{1}{2}p^2 \qquad p \in \mathbb{R} \tag{2.56}$$

$$\Phi_k = \tilde{L}^{(k)} + \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2, \tag{2.57}$$

where $\tilde{L}^{(k)}$ is value of the loss function computed at the approximated minimum of $E(\cdot; \boldsymbol{\theta}^{(k)})$ in step k. We also impose the following

Assumptions 2.13.
The sequence $\{\boldsymbol{\theta}^{(k)}\}_{k\in\mathbb{N}}$ generated by Algorithm 2.3 is bounded.

Hypothesis 0. This will be proved as we proceed. In particular the last observation that concludes the proof is at the end of Hypothesis 3.

Hypothesis 1. From the end of the proof of Lemma 2.8 we conclude that, for m as in equation (2.50), the value δ^m is, by all means, a lower bound for λ_k. The same proof also directly shows that, with

$$a = \sigma\delta^m \tag{2.58}$$

$$t_k = \sqrt{-\Delta_k}, \tag{2.59}$$

we have

$$\Phi_{k+1} \leq \Phi_k - at_k^2.$$

Furthermore, we also have that

$$\sum_{k=0}^{+\infty} t_k^2 \leq \frac{1}{a}\sum_{k=0}^{+\infty} \Phi_k - \Phi_{k+1} = \frac{1}{a}\left(\Phi_0 - \inf_k \Phi_k\right) < +\infty,$$

where the right-hand series is a telescopic series whose sum is the limit of $\{\Phi_k\}_{k\in\mathbb{N}}$, which is lower bounded (from Assumption 2.2 (ii)) and hence convergent to its infimum. We can conclude that the left-hand series is convergent, Δ_k tends to zero and from (2.30) we know that

$$0 = \lim_{k\to+\infty} \|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\|_2^2 = \lim_{k\to+\infty} \Delta_k \underbrace{=}_{(2.30)} \lim_{k\to+\infty} \|\tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2, \tag{2.60}$$

in fact, since $\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)} = \lambda_k(\tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{\theta}^{(k)})$ for $\lambda_k \in (0, 1]$, we have $\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\|_2 \leq \|\tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2$ with the latter converging to zero.

Hypothesis 2. We are going to use Lemma 10 from [22], that we report here slightly changed to be more apt to our context, to prove that [H2] holds with

$$\boldsymbol{\zeta}^{(k)} = \mathbf{z}^{(k)} \tag{2.61}$$

Lemma 2.14

Under Assumptions 2.2 (iv), consider the sequence $\{\boldsymbol{\theta}^{(k)}\}_{k\in\mathbb{N}}$ and $\{\tilde{\mathbf{z}}^{(k)}\}_{k\in\mathbb{N}}$ generated by Algorithm 2.3. There exist the constants $c_1, c_2, c_3, c_4 \in \mathbb{R}$ that depend only from α_{min}, α_{max}, τ, ν, $\Lambda_{\nabla G}$ and $C_{\nabla F}$ such that

$$L(\mathbf{z}^{(k)}) \geq L(\tilde{\mathbf{z}}^{(k)}) + c_1\Delta_k - c_2\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 \tag{2.62}$$

$$L(\mathbf{z}^{(k)}) \leq L(\boldsymbol{\theta}^{(k)}) - c_3\Delta_k + c_4\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 \tag{2.63}$$

Proof: The Descent Lemma [11] tells us that

$$G(\mathbf{z}^{(k)}) \geq G(\tilde{\mathbf{z}}^{(k)}) - \nabla_{\boldsymbol{\theta}} G(\mathbf{z}^{(k)})^T(\tilde{\mathbf{z}}^{(k)} - \mathbf{z}^{(k)}) - \frac{\Lambda_{\nabla G}}{2}\|\tilde{\mathbf{z}}^{(k)} - \mathbf{z}^{(k)}\|_2^2. \tag{2.64}$$

Also, by definition of $h^{(k)}$ (2.38) and simple computations we have that

$$\begin{aligned} h^{(k)}(\tilde{\mathbf{z}}^{(k)}) - h^{(k)}(\mathbf{z}^{(k)}) = R^1(\tilde{\mathbf{z}}^{(k)}) - R^1(\mathbf{z}^{(k)}) + \left(\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)}) + \mathbf{e}^{(k)}\right)^T(\tilde{\mathbf{z}}^{(k)} - \mathbf{z}^{(k)}) \\ + \frac{1}{2\alpha_k}\|\tilde{\mathbf{z}}^{(k)} - \boldsymbol{\theta}^{(k)}\|_{\mathbf{M}_k}^2 - \frac{1}{2\alpha_k}\|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_{\mathbf{M}_k}^2 \end{aligned} \tag{2.65}$$

Rearranging the terms and using the condition (2.27) on $h^{(k)}(\tilde{\mathbf{z}}^{(k)})$ we obtain

$$\begin{aligned} R^1(\mathbf{z}^{(k)}) \geq R^1(\tilde{\mathbf{z}}^{(k)}) + \left(\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)}) + \mathbf{e}^{(k)}\right)^T(\tilde{\mathbf{z}}^{(k)} - \mathbf{z}^{(k)}) + \\ + \frac{1}{2\alpha_k}\|\tilde{\mathbf{z}}^{(k)} - \boldsymbol{\theta}^{(k)}\|_{\mathbf{M}_k}^2 - \frac{1}{2\alpha_k}\|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_{\mathbf{M}_k}^2 + \frac{\tau}{2}\underbrace{h^{(k)}(\tilde{\mathbf{z}}^{(k)})}_{\Delta_k} \end{aligned} \tag{2.66}$$

Adding together this last inequality with (2.64) yields

$$\begin{aligned} L(\mathbf{z}^{(k)}) \geq L(\tilde{\mathbf{z}}^{(k)}) - \left(\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)}) - \nabla_{\boldsymbol{\theta}} G(\mathbf{z}^{(k)})\right)^T(\mathbf{z}^{(k)} - \tilde{\mathbf{z}}^{(k)}) - (\mathbf{z}^{(k)} - \tilde{\mathbf{z}}^{(k)})^T\mathbf{e}^{(k)} \\ + \frac{1}{2\alpha_k}\|\tilde{\mathbf{z}}^{(k)} - \boldsymbol{\theta}^{(k)}\|_{\mathbf{M}_k}^2 - \frac{1}{2\alpha_k}\|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_{\mathbf{M}_k}^2 + \frac{\tau}{2}\Delta_k \end{aligned} \tag{2.67}$$

To obtain the first result, we use the Cauchy-Schwarz inequality, the Lipschitz continuity of $\nabla_{\boldsymbol{\theta}} G$ and the inequalities (2.28) and (2.29) so that

$$\begin{aligned} \left(\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)}) - \nabla_{\boldsymbol{\theta}} G(\mathbf{z}^{(k)})\right)^T(\mathbf{z}^{(k)} - \tilde{\mathbf{z}}^{(k)}) &\leq \|\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)}) - \nabla_{\boldsymbol{\theta}} G(\mathbf{z}^{(k)})\|_2\|\mathbf{z}^{(k)} - \tilde{\mathbf{z}}^{(k)}\|_2 \\ &\leq \Lambda_{\nabla G}\alpha_{\max}\nu\sqrt{2\tau\left(1 + \frac{\tau}{2}\right)}(-\Delta_k). \end{aligned}$$

Furthermore, using the well known inequality $\mathbf{a}^T\mathbf{b} \leq \frac{1}{2}\|\mathbf{a}\|_2 + \frac{1}{2}\|\mathbf{b}\|_2^2$, we have that

$$(\mathbf{z}^{(k)} - \tilde{\mathbf{z}}^{(k)})^T\mathbf{e}^{(k)} \leq \frac{1}{2}\|\mathbf{e}^{(k)}\|_2^2 + \frac{1}{2}\|\mathbf{z}^{(k)} - \tilde{\mathbf{z}}^{(k)}\|_2^2.$$

Combining these two bounds with (2.67) and the properties imposed on $\mathbf{M}_k$ gives us

$$\begin{aligned}
L(\mathbf{z}^{(k)}) &\geq L(\tilde{z}^{(k)}) + \Lambda_{\nabla G}\alpha_{\max}\nu\sqrt{2\tau\left(1+\frac{\tau}{2}\right)}\Delta_k - \frac{1}{2}\|\mathbf{e}^{(k)}\|_2^2 - \frac{1}{2}\|\mathbf{z}^{(k)} - \tilde{z}^{(k)}\|_2^2 \\
&\quad + \frac{1}{2\alpha_k}\|\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_{\mathbf{M}_k}^2 - \frac{1}{2\alpha_k}\|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_{\mathbf{M}_k}^2 + \frac{\tau}{2}\Delta_k \\
&\geq L(\tilde{z}^{(k)}) + \Lambda_{\nabla G}\alpha_{\max}\nu\sqrt{2\tau\left(1+\frac{\tau}{2}\right)}\Delta_k - \frac{1}{2}\|\mathbf{e}^{(k)}\|_2^2 - \frac{1}{2}\|\mathbf{z}^{(k)} - \tilde{z}^{(k)}\|_2^2 \\
&\quad + \frac{1}{2\nu\alpha_{\max}}\|\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 - \frac{\nu}{2\alpha_{\min}}\|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 + \frac{\tau}{2}\Delta_k
\end{aligned}$$

Now, using (2.28)-(2.29) and (2.40) we have the first result

$$\begin{aligned}
L(\mathbf{z}^{(k)}) &\geq L(\tilde{z}^{(k)}) + \Lambda_{\nabla G}\alpha_{\max}\nu\sqrt{2\tau\left(1+\frac{\tau}{2}\right)}\Delta_k - \frac{C_{\nabla F}^2}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 \\
&\quad + \frac{\tau}{2}\alpha_{\max}\nu\Delta_k + \left(1+\frac{\tau}{2}\right)\frac{\alpha_{\max}}{\alpha_{\min}}\nu^2\Delta_k \\
&= L(\tilde{z}^{(k)}) + \underbrace{\left(\Lambda_{\nabla G}\alpha_{\max}\nu\sqrt{2\tau\left(1+\frac{\tau}{2}\right)} + \frac{\tau}{2}\alpha_{\max}\nu + \left(1+\frac{\tau}{2}\right)\frac{\alpha_{\max}}{\alpha_{\min}}\nu^2\right)}_{c_1}\Delta_k \\
&\quad - \underbrace{\frac{C_{\nabla F}^2}{2}}_{c_2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2.
\end{aligned}$$

For the second part, we restart from the Descent Lemma

$$G(\tilde{z}^{(k)}) \leq G(\boldsymbol{\theta}^{(k)}) + \boldsymbol{\nabla}_{\boldsymbol{\theta}}G(\boldsymbol{\theta}^{(k)})^T(\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}) + \frac{\Lambda_{\nabla G}}{2}\|\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2.$$

Adding and subtracting $R^1(\tilde{z}^{(k)})$ and $R^1(\boldsymbol{\theta}^{(k)})$ gives us

$$\begin{aligned}
G(\tilde{z}^{(k)}) + R^1(\tilde{z}^{(k)}) &\leq G(\boldsymbol{\theta}^{(k)}) \pm R^1(\boldsymbol{\theta}^{(k)}) + R^1(\tilde{z}^{(k)}) + \boldsymbol{\nabla}_{\boldsymbol{\theta}}G(\boldsymbol{\theta}^{(k)})^T(\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}) \\
&\quad + \frac{\Lambda_{\nabla G}}{2}\|\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 \\
L(\tilde{z}^{(k)}) &\leq L(\boldsymbol{\theta}^{(k)}) + R^1(\tilde{z}^{(k)}) - R^1(\boldsymbol{\theta}^{(k)}) + \boldsymbol{\nabla}_{\boldsymbol{\theta}}G(\boldsymbol{\theta}^{(k)})^T(\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}) \\
&\quad + \frac{\Lambda_{\nabla G}}{2}\|\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2
\end{aligned}$$

So that, using similar arguments as before, as well as inequality (2.30), we obtain

$$\begin{aligned}
L(\tilde{z}^{(k)}) &\leq L(\boldsymbol{\theta}^{(k)}) + h^{(k)}(\tilde{z}^{(k)}) + \frac{\Lambda_{\nabla G}}{2}\|\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 + (\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)})^T\mathbf{e}^{(k)} \\
&\leq L(\boldsymbol{\theta}^{(k)}) + \Delta_k + \frac{\Lambda_{\nabla G}}{2}\|\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 + \frac{1}{2}\|\mathbf{e}^{(k)}\|_2^2 + \frac{1}{2}\|\tilde{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 \\
&\leq L(\boldsymbol{\theta}^{(k)}) + \Delta_k - \frac{2\Lambda_{\nabla G}\alpha_{\max}\nu}{C_\tau}\Delta_k + \frac{C_{\nabla F}^2}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 \\
&\leq L(\boldsymbol{\theta}^{(k)}) - \underbrace{\left(\frac{2\Lambda_{\nabla G}\alpha_{\max}\nu}{C_\tau} - 1\right)}_{c_3}\Delta_k + \underbrace{\frac{C_{\nabla F}^2}{2}}_{c_4}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2.
\end{aligned}$$

□

From equations (2.54) and (2.55) we have

$$\begin{aligned}\Phi_{k+1} &\leq \tilde{L}_{\mathbf{z}}^{(k)} + \frac{\gamma}{2}\|\tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 \\ &\leq L(\tilde{\boldsymbol{z}}^{(k)}) + \frac{\gamma + 2\rho}{2}\|\tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2.\end{aligned}$$

Now, using the result in equation (2.30) we obtain

$$\Phi_{k+1} \leq L(\tilde{\boldsymbol{z}}^{(k)}) - \frac{(\gamma + 2\rho)\nu\alpha_{\max}}{C_\tau}\Delta_k,$$

and from the previous Lemma we get the inequality.

$$\Phi_{k+1} \leq L(\mathbf{z}^{(k)}) - \left(c_1 + \frac{(\gamma + 2\rho)\nu\alpha_{\max}}{C_\tau}\right)\Delta_k + c_2\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2.$$

So, if in the definition of f in (2.56) we choose

$$p_k = \sqrt{2}\left(-\left(c_1 + \frac{(\gamma + 2\rho)\nu\alpha_{\max}}{C_\tau}\right)\Delta_k + c_2\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2\right)^{\frac{1}{2}},$$

we have the first inequality of the claim.
For the other inequality, we use first the equation (2.63) of the previous Lemma and then the bound on the approximate value of the loss as in (2.46):

$$\begin{aligned}L(\mathbf{z}^{(k)}) + \frac{1}{2}p_k^2 &\leq L(\boldsymbol{\theta}^{(k)}) - c_3\Delta_k + c_4\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 + \frac{1}{2}p_k^2 \\ &\leq \tilde{L}^{(k)} - c_3\Delta_k + (c_4 + \rho)\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 \\ &\quad \pm \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 + \frac{1}{2}p_k^2 \\ &= \underbrace{\tilde{L}^{(k)} + \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2}_{\Phi_k} + r_k,\end{aligned}$$

where we have that

$$r_k = \frac{1}{2}p_k^2 - c_3\Delta_k + \left(c_4 + \rho - \frac{\gamma}{2}\right)\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2.$$

Lastly, the convergence of r_k to zero is assured by equation (2.60).

Hypothesis 3. We are going to show that there is a subgradient $\boldsymbol{\kappa}^{(k)} \in \partial f(\mathbf{z}^{(k)}, p_k)$ such that

$$\|\boldsymbol{\kappa}^{(k)}\|_2 \leq \bar{c}\left(\sqrt{-\Delta_k} + \sqrt{-\Delta_{k-1}}\right),$$

for some $\bar{c} \geq 0$. This would mean that [H3] is verified for $\mathcal{I} = \{1, 2\}$, $b = 2\bar{c}$ and $\xi_1 = \xi_2 = \frac{1}{2}$, with t_k given by (2.59).
Similarly to the previous hypothesis, we base the proof on Lemma 3 from [19] that is recalled here adapted to our case.

Lemma 2.15

Under Assumptions 2.2, if $\{\boldsymbol{\theta}^{(k)}\}_{k\in\mathbb{N}}$ is a sequence generated by Algorithm 2.3, then there exists a subgradient $\boldsymbol{\omega}^{(k)} \in \partial L(\mathbf{z}^{(k)})$ such that

$$\|\boldsymbol{\omega}^{(k)}\|_2 \leq c_5 \left(\|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2 + \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2\right) \tag{2.68}$$

$$\leq c_6 \left(\sqrt{-\Delta_k} + \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2\right), \tag{2.69}$$

with $c_5, c_6 \in \mathbb{R}$ depends only on α_{min}, α_{max}, $C_{\nabla F}$ and $\Lambda_{\nabla G}$.

Proof: By definition, $\mathbf{z}^{(k)} = \operatorname{argmin}_{\mathbf{z}} h^{(k)}(\mathbf{z})$. The relative optimality condition reads

$$\mathbf{0} \in \partial h^{(k)}(\mathbf{z}^{(k)}) \Leftrightarrow -\frac{\mathbf{M}_k(\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)})}{\alpha_k} - \left(\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)}) + \mathbf{e}^{(k)}\right) \in \partial R^1(\mathbf{z}^{(k)}) \tag{2.70}$$

Which in turn, considering simple subdifferential calculus rules, means

$$\boldsymbol{\omega}^{(k)} = -\frac{\mathbf{M}_k(\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)})}{\alpha_k} - \left(\nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)}) + \mathbf{e}^{(k)}\right) + \nabla_{\boldsymbol{\theta}} G(\mathbf{z}^{(k)}) \in \partial L(\mathbf{z}^{(k)}). \tag{2.71}$$

Then, a direct application of the triangular inequality and (2.40) yields

$$\begin{aligned}
\|\boldsymbol{\omega}^{(k)}\|_2 &\leq \frac{1}{\alpha_k}\|\mathbf{M}_k\|\|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2 + \|\mathbf{e}^{(k)}\|_2 + \|\nabla_{\boldsymbol{\theta}} G(\mathbf{z}^{(k)}) - \nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta}^{(k)})\|_2 \\
&\leq \left(\frac{\nu}{\alpha_{\min}} + \Lambda_{\nabla G}\right)\|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2 + C_{\nabla F}\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\| \\
&\leq \underbrace{\max\left\{\frac{\nu}{\alpha_{\max}} + \Lambda_{\nabla G}, C_{\nabla F}\right\}}_{c_5}\left(\|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2 + \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2\right),
\end{aligned}$$

and the desired result is obtained via (2.28) (recall $\Delta_k = h^{(k)}(\tilde{z}^{(k)})$)

$$\begin{aligned}
\|\boldsymbol{\omega}^{(k)}\|_2 &\leq c_5\left(\sqrt{-\Delta_k\left(1 + \frac{\tau}{2}\right)2\alpha_{\max}\nu} + \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2\right) \\
&\leq \underbrace{c_5 \cdot \max\left\{1, \sqrt{\left(1 + \frac{\tau}{2}\right)2\alpha_{\max}\nu}\right\}}_{c_6}\left(\sqrt{-\Delta_k} + \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2\right)
\end{aligned}$$

□

Recalling that $\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)} = \lambda_{k-1}(\tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{\theta}^{(k-1)})$ for $\lambda_{k-1} \leq 1$, from this Lemma we have that, for the subgradient $\boldsymbol{\omega}^{(k)}$

$$\begin{aligned}
\|\boldsymbol{\omega}^{(k)}\|_2 &\leq c_6\left(\sqrt{-h^{(k)}(\tilde{\boldsymbol{z}}^{(k)})} + \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2\right) \\
&\leq c_6\left(\sqrt{-h^{(k)}(\tilde{\boldsymbol{z}}^{(k)})} + \|\tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2\right).
\end{aligned}$$

Also, inequality (2.30) gives us

$$\begin{aligned}\|\boldsymbol{\omega}^{(k)}\|_2 &\leq c_6\sqrt{-\Delta_k} + c_6\sqrt{\frac{2\nu\alpha_{\max}}{C_\tau}\left(-\Delta_{k-1}\right)} \\ &\leq c_{\boldsymbol{\omega}}\left(\sqrt{-\Delta_k} + \sqrt{-\Delta_{k-1}}\right),\end{aligned}$$

with $c_{\boldsymbol{\omega}} = c_6 \cdot \max\{1, \frac{2\nu\alpha_{\max}}{C_\tau}\}$.
Due to basic subdifferential calculus rules, we need to check if a similar inequality holds for $|p_k| = p_k$. The same results that we just used lead us also to have that

$$\begin{aligned}p_k &= \sqrt{2}\left(-\left(c_1 + \frac{(\gamma+2\rho)\nu\alpha_{\max}}{C_\tau}\right)h^{(k)}(\tilde{\boldsymbol{z}}^{(k)}) + c_2\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2\right)^{\frac{1}{2}} \\ &\leq \sqrt{2}\left(-\left(c_1 + \frac{(\gamma+2\rho)\nu\alpha_{\max}}{C_\tau}\right)h^{(k)}(\tilde{\boldsymbol{z}}^{(k)}) + c_2\|\tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{\theta}^{(k-1)}\|_2^2\right)^{\frac{1}{2}} \\ &\leq \sqrt{2}\left(-\left(c_1 + \frac{(\gamma+2\rho)\nu\alpha_{\max}}{C_\tau}\right)\Delta_k - c_2\frac{2\nu\alpha_{\max}}{C_\tau}\Delta_{k-1}\right)^{\frac{1}{2}} \\ &\leq c_p\sqrt{-\Delta_k - \Delta_{k-1}},\end{aligned}$$

where

$$c_p = \sqrt{2}\max\left\{c_1 + \frac{(\gamma+2\rho)\nu\alpha_{\max}}{C_\tau}, c_2\frac{2\nu\alpha_{\max}}{C_\tau}\right\}$$

Finally, with a basic inequality of the square root we get

$$p_k \leq c_p\left(\sqrt{-\Delta_k} + \sqrt{-\Delta_{k-1}}\right). \tag{2.72}$$

Putting everything together we obtain the required inequality, in fact:

$$\begin{aligned}\|\boldsymbol{\kappa}^{(k)}\|_2 &= \|(\boldsymbol{\omega}^{(k)}, p_k)\|_2 \\ &\leq \|\boldsymbol{\omega}^{(k)}\|_2 + p_k \\ &\leq (c_{\boldsymbol{\omega}} + c_p)\left(\sqrt{-\Delta_k} + \sqrt{-\Delta_{k-1}}\right).\end{aligned}$$

Note that, in view of (2.60), what we have just shown also proves that $p_k \to 0$, so that Hypothesis 0 also holds true.

Hypothesis 4. Consider a limit point $(\bar{\boldsymbol{\theta}}, \bar{p})$ for the sequence $\{(\boldsymbol{\theta}^{(k)}, p_k)\}_{k\in\mathbb{N}}$ and let $\{(\boldsymbol{\theta}^{(k_j)}, p_{k_j})\}_{j\in\mathbb{N}}$ be any of the subsequences that converge to it. From inequality (2.28) we have that

$$\lim_{k\to+\infty}\|\mathbf{z}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2 = \lim_{k\to+\infty}\Delta_k = 0,$$

because we have already shown in (2.60) that the second limit is indeed zero. This limit also tells us that $\mathbf{z}^{(k_j)}$ also converges to $\bar{\boldsymbol{\theta}}$. As a direct consequence of Assumption 2.2 (iii) and (iv), f is lower semicontinuous, so that, by definition:

$$\lim_{j\to+\infty} f(\mathbf{z}^{(k_j)}, p_{k_j}) \geq f(\bar{\boldsymbol{\theta}}, \bar{p}).$$

In particular, the limit exists and is equal to $\lim_{k\to+\infty} \Phi_k = \inf_k \Phi_k \in \mathbb{R}$. thanks to Hypothesis 1 and 2 which have already been proved. Now we only need the inequality in the opposite direction.
The same reasoning leads us to conclude that $\|\boldsymbol{\kappa}^{(k)}\|_2$ tends to zero, as well as $\|\boldsymbol{\omega}^{(k)}\|_2$, $\boldsymbol{\omega}^{(k)}$ and p_k, which are the components of $\boldsymbol{\kappa}^{(k)}$.
Basic subdifferential calculus rules say that

$$\boldsymbol{\omega}^{(k)} = \boldsymbol{\nabla}_{\boldsymbol{\theta}} G(\mathbf{z}^{(k)}) + \mathbf{s}^{(k)}, \qquad \mathbf{s}^{(k)} \in \partial R^1(\mathbf{z}^{(k)}).$$

The gradient of G is continuous by assumption and $\mathbf{z}^{(k)}$ goes to $\bar{\boldsymbol{\theta}}$ means that $\mathbf{s}^{(k_j)} \xrightarrow{j\to+\infty} -\boldsymbol{\nabla}_{\boldsymbol{\theta}} G(\bar{\boldsymbol{\theta}})$. Now we recall the basic subgradient inequality (A.1)

$$R^1(\bar{\boldsymbol{\theta}}) \geq R^1(\mathbf{z}^{(k_j)}) + \left(\mathbf{s}^{(k_j)}\right)^T (\bar{\boldsymbol{\theta}} - \mathbf{z}^{(k_j)}).$$

We add $\frac{p_{k_j}^2}{2}$ to both sides and $\pm G(\mathbf{z}^{(k_j)})$ to the right-hand side:

$$R^1(\bar{\boldsymbol{\theta}}) + \frac{1}{2}p_{k_j}^2 \geq f(\mathbf{z}^{(k_j)}, p_{k_j}) - G(\mathbf{z}^{(k_j)}) + \left(\mathbf{s}^{(k_j)}\right)^T (\bar{\boldsymbol{\theta}} - \mathbf{z}^{(k_j)}).$$

Taking the limit and reordering the terms does indeed yield

$$\lim_{j\to+\infty} f(\mathbf{z}^{(k_j)}, p_{k_j}) \leq f(\bar{\boldsymbol{\theta}}, \bar{p}).$$

Hypothesis 5. Recalling that $t_k = \sqrt{-h^{(k)}(\tilde{\boldsymbol{z}}^{(k)})}$, we have that, thanks to how the algorithm updates the variable $\boldsymbol{\theta}^{(k)}$:

$$\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k-1)}\|_2^2 = \lambda_k^2 \|\mathbf{d}^{(k)}\|_2^2 = \lambda_k^2 \|\tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{\theta}^{(k)}\|_2^2 \leq -\lambda_k^2 \frac{C_\tau}{2\nu\alpha_{\max}} h^{(k)}(\tilde{\boldsymbol{z}}^{(k)}),$$

where we have also used (2.30). Hence, [H5] holds with $w = \lambda_k \sqrt{\frac{C_\tau}{2\nu\alpha_{\max}}}$.

2.3.3 Numerical Experience

Problem Setting. The proposed Algorithm 2.3 has been tested in a problem of *deblurring*. In this scenario, the image formation model can be formulated as

$$\mathbf{y} = \mathbf{H}\mathbf{u}^{\mathbf{true}} + \mathcal{N}(0, v^2) \in \mathbb{R}^n,$$

where the forward operator $\mathbf{H} \in \mathbb{R}^{n\times n}$ is a convolution operator, i.e., a matrix built from a convolution kernel $\mathbf{h} \in \mathbb{R}^{m\times m}$, which is also known as the *Point Spread*

Function (PSF). The effect of the PSF is, as the name suggests, spreading the intensity of every pixels across its neighbours. It can be visualised by applying the kernel to an image with just one pixel activated. The PSF is, more often than not, a low–pass filter, which means that the matrix $\mathbf{H}$ is usually very ill conditioned. Directly solving the inverse problem, even when it is feasible, would then lead to unmeaningful solutions. If the underlying convolution kernel $\mathbf{h}$ is completely unknown and needs to be retrieved together with $\mathbf{u}^{(\text{true})} \in \mathbb{R}^n$, then we are facing a *blind deconvolution* problem. In this experiment we can say we are in the presence of a semi-blind deconvolution problem, since we assume to have partial knowledge on $\mathbf{h}$. When looking for reconstructions $\bar{\mathbf{u}}$ it is important to also find a good approximation of the kernel. Indeed, simply solving a variational model with a wrong PSF often leads to undesirable artifacts and ringing effects on the restorations. This problem is more of a semi-blind deconvolution, as we assume to know the kernel size $m = 5$. We also take for granted that the corner pixels of $\mathbf{h}$ are zero, the center pixel has unknown value θ_2, its four nearest neighbours have all equal intensity $\theta_3/4$, and, similarly, $\theta_4/16$ is the value of the remaining components of the kernel. The parameters θ_2, θ_3 and θ_4 are unknown and need to be estimated. In the experiments, the true kernel is characterized by $\theta_2 = 0.15, \theta_3 = 0.1$ and $\theta_4 = 0.75$, and the standard deviation of the Gaussian noise is $5 \cdot 10^{-3}$.
The energy functional used is

$$E_{\mathbf{y}}(\mathbf{u}; \boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{H}(\boldsymbol{\theta})\mathbf{u} - \mathbf{y}\|_2 + C_1\theta_1\|\mathbf{D}\mathbf{u}\|_{1,\varepsilon}. \tag{2.73}$$

The operator $\mathbf{D}$ is the finite difference operator as in (1.6), with zero boundary conditions. Moreover $\|\cdot\|_{1,\varepsilon}$ is a C^2-differentiable Huber-like approximation of the ℓ_1-norm, and is defined as

$$\|\mathbf{v}\|_{1,\varepsilon} = \sum_{i=1}^{n} \psi_\varepsilon(v_i), \tag{2.74}$$

where

$$\psi_\varepsilon(s) = \begin{cases} -\frac{|s|^3}{3\varepsilon^2} + \frac{|s|^2}{\varepsilon} & |s| \leq \varepsilon \\ |s| - \frac{\varepsilon}{3} & |s| > \varepsilon. \end{cases} \tag{2.75}$$

In the experiments ε is set at 10^{-2}. The constant C_1 is not learned, and serves the sole purpose of rescaling the first component of the loss gradient so that the update of $\boldsymbol{\theta}$ does not require different steplengths for the parameters. As for the loss function, we have that

$$L(\mathbf{u}, \boldsymbol{\theta}) = \frac{1}{2N_s}\sum_{s=1}^{N_s}\|\mathbf{u}_s - \mathbf{x}_s\|_2^2 + C_2(\sum_{i=2}^{4}\theta_i - 1)^2 + \iota_{\{\boldsymbol{\theta}\geq\mathbf{0}\}}(\boldsymbol{\theta}), \tag{2.76}$$

C_2 is an eventual hyperparameter that controls the amount of regularization of the parameters, in order to force the kernel pixels to sum up to one, a fundamental property of PSFs. Finally, $\iota_\Omega(\cdot)$ represents the indicator function of a set:

$$\iota_\Omega(s) = \begin{cases} 0 & s \in \Omega \\ +\infty & s \notin \Omega. \end{cases} \tag{2.77}$$

Training the Model. The training of this variational model is carried out with the use of $N_s = 20$ samples $(\mathbf{y}_s, \mathbf{x}_s)$, where $\mathbf{y}_s$ is the blurred image and $\mathbf{x}_s$ its clean ground truth. All the images are from the Berkley Segmentation Dataset [62] and are cropped to have size 128×128. From the same dataset, a separate test set of 10 images with the same dimensions is selected. The final performance is then measured on the restoration of the test images in terms of the *Peak Signal-to-Noise Ratio* (PSNR) and of the Structural SIMilarity index (SSIM) [100]. For the energy minimization, once the parameters are learned, we used the *Scaled Gradient Projection* method [17]. The SGP scheme was allowed to run for at most 5000 iterations and was stopped when the relative difference between two consecutive energy values, the relative distance of two $\boldsymbol{\theta}$ values or the norm of the approximate gradient loss were less than 10^{-7}.
This setting is an extension to that proposed in [92], which also gives us an algorithm to compare against: the *Forward-Inexact-Forward-Backward* method (FIFB). To implement the FIFB method, we heavily relied on the code provided[1], having the algorithms share as much code as possible to make the comparison more fair.All tests were conducted on a remote server equipped with 48Gb of RAM, 48 AMD EPYC 7552 CPUs and MATLAB version 2018a.

We now turn to the details of the proposed Bilevel Linesearch Algorithm 2.3. The approximate minima were computed with SGP until the required stopping criterion (2.33) was satisfied. For safety purposes, a maximum number of 25000 iterations was placed, but it was enforced very rarely or in extreme cases. Also, the linear system was computed with the *conjugate gradient* method until the prescribed tolerance (2.35) on the residual was met. The hyperparameters of the model were set according to Table 2.1. We note that the first regularization term in the loss function is differentiable, hence the proximal operator is limited to that of the indicator function, which is the standard Euclidean projection. Hence, there was no need to set τ. The proposed method was set to perform no more than 250 iterations, with similar stopping criteria to those described above. In particular, it was stopped when the relative difference between two consecutive approximate loss values or the norm of the loss gradient were less than 10^{-6} or when the relative difference between two consecutive iterates was less than 10^{-4}. This weaker tolerance on the iterates is motivated by the fact that it also directly influences the inner problem stopping criterion, where the norm squared between consecutive iterates appears. Setting a really strict threshold for this means that the inner solver may end up running for a tremendous amount of time, but at the same time the overall performance does not show sufficient improvements to justify the extra computational efforts.
As for the FIFB, we followed the choices made by the authors in [92] and in the relative code, and was let run for at most 10^5 iterations. The significantly higher number of iterations is expected because of the different nature of the algorithms.

[1] https://zenodo.org/records/7974062

Each iteration of the FIFB method performs only one gradient step on the images, which obviously leads to smaller gains in terms of loss function. We also placed the same stopping criteria, but again with a stricter threshold of 10^{-12}.

η	$\eta_{\min}$	γ	δ	σ	$\alpha_{\min}$	$\alpha_{\max}$	C_1	C_2
10	0.1	1	0.3	10^{-4}	10^{-10}	0.01	1	1

Table 2.1: Hyperparameters chosen for Algorithm 2.3 in the proposed semi-blind deconvolution problem (2.73)-(2.76).

Results. Figure 2.1 illustrates the comparisons between the two algorithms. For the FIFB scheme, the initial values for $\mathbf{u}^{(0)}$ are pre-computed with an optimization algorithm, and then used to obtain also the starting point for the adjoint variable. We used this $\mathbf{u}^{(0)}$ to computed the initial approximated value for the loss function, and consequently the time required for iteration zero includes the time used for this "pre-processing". On the other hand, in our proposed method the value of the approximate loss function is plotted at time zero because we are able to obtain it indirectly in the required computations for the first iterate.
The graphs shows how both methods needs some time to adjust before seeing a significant decrease in the approximate loss function. The difference between the two is that our proposed method is faster in reaching this break-point while the FIFB method is slower but a bit steadier, only showing signs of an acceleration towards the end of its runtime. The very strict threshold for the stopping of FIFB is motivated by the behaviour exhibited by the method: if the threshold is set less severe, these small steps accidentally leads to a premature stop of the algorithm, at least compared to the performance shown by our proposed approach.

Table 2.2 reports the numbers for the two methods in question. It is interesting to notice how the improvement between the blurred images and the final reconstructions is more evident in terms of the SSIM. This is not by chance. In fact, the PSNR is notoriously not a good metric to measure the distortion caused by blur, while the same cannot be said for the SSIM [64]. Overall, the performance of the two algorithms in terms of validation is essentially the same.

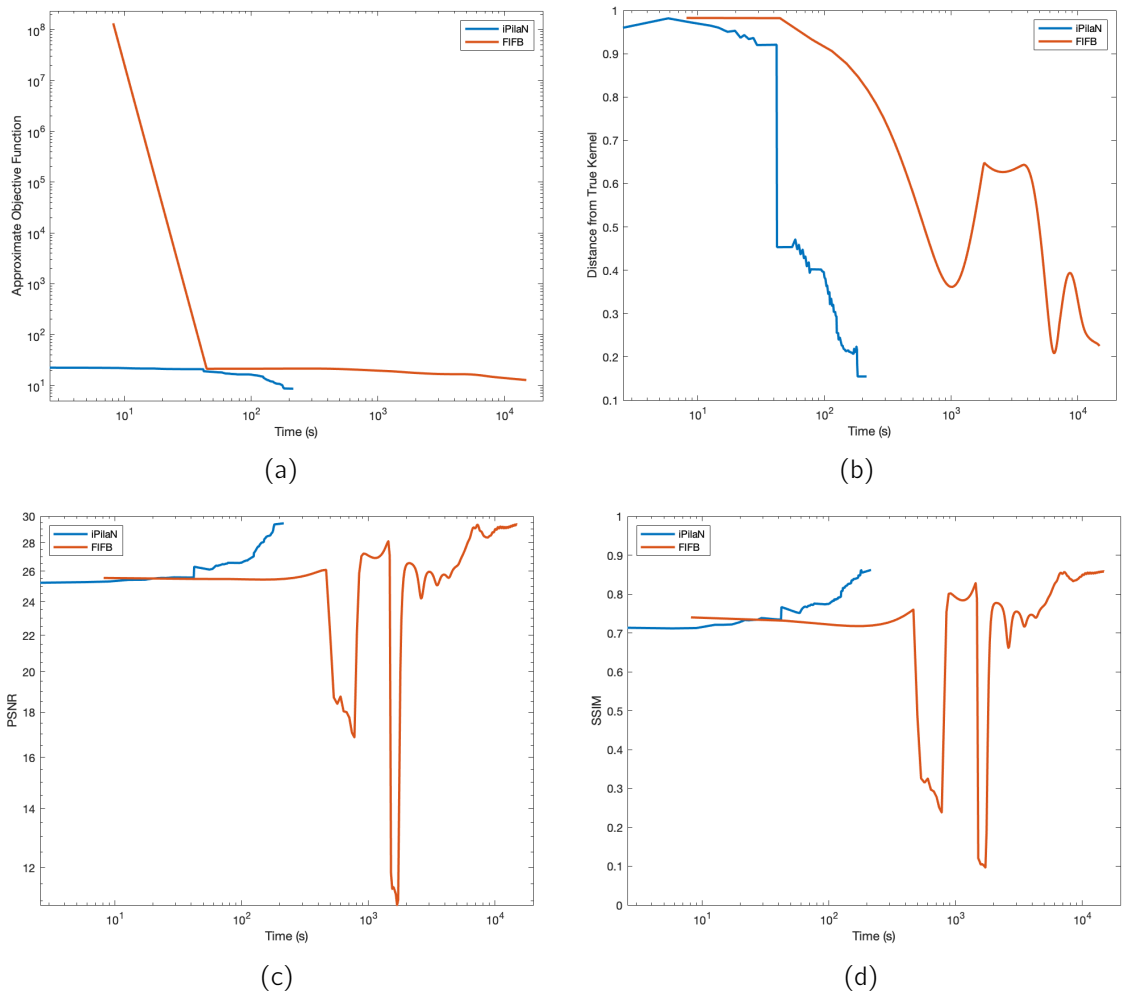


Figure 2.1: (a) Approximate value of the loss function through time; (b) Distance of the estimated kernel from the true one; (c) Evolution of the mean PSNR computed on the test images; (d) Evolution of the mean SSIM computed on the test images.

	Blurred	Ours	FIFB
Avg. PSNR	24.63	29.39	29.35
Avg. SSIM	0.701	0.860	0.858
Castle PSNR	18.75	24.70	20.54
Castle SSIM	0.592	0.871	0.723
Dog PSNR	25.98	30.13	28.32
Dog SSIM	0.769	0.870	0.823

Table 2.2: Numerical results in terms of PSNR and SSIM for the methods compared. The average PSNR and SSIM were computed for the 10 images of the test set, while "Castle" and "Dog" refer to the values for the images shown in Figures 2.2 and 2.3, respectively.

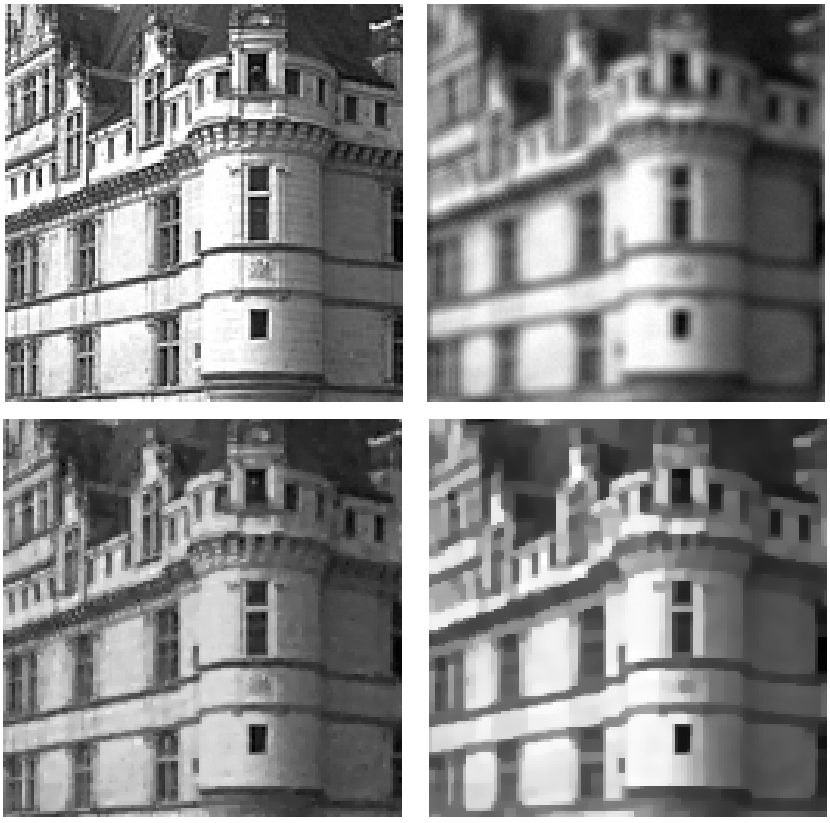


Figure 2.2: From the top left image, going clockwise: ground truth, blurred sample, reconstruction obtained with $\boldsymbol{\theta}$ learned by FIFB, reconstruction obtained with $\boldsymbol{\theta}$ learned by our proposed algorithm.

Summary and Possible Improvements.

In this chapter we have introduced bilevel optimization as a tool for the learning of parameters in variational models. In particular, we have described the implicit differentiation technique, one of the prevalent strategies in the literature to solve this kind of problems in imaging applications. Moreover, we have proposed a general optimization scheme to solve the bilevel problem. The algorithm is classified in the Forward-Backward framework, where the novelty lies in the fact that it can include some form of error in the computation of the gradient for the differentiable part. The algorithm was also compared to other schemes present in literature that solve the same problem under very similar, if not exactly the same, assumptions. The results of this comparisons are strongly promising. Our proposed bilevel linesearch algorithm exhibits a comparable performance to other bilevel optimization schemes, while doing that in a relatively short amount of time. This line of work has many natural continuations and directions of improvement, some of them clearly dictated by the strict assumptions placed on the objective functions. The obvious one is smoothness. The assumption that wants a twice continuously differentible energy

Figure 2.3: From the top left image, going clockwise: ground truth, blurred sample, reconstruction obtained with $\boldsymbol{\theta}$ learned by FIFB, reconstruction obtained with $\boldsymbol{\theta}$ learned by our proposed algorithm.

functional is fundamental and at the very base of the algorithm. However, the vast majority famous regularizers are not even differentiable, think of the Total Variation in Example 1.3 or the ℓ_1-norm in Example 1.2. Although it is possible to find many smooth and proficient approximations for many non-differentiable functions, it would be ideal to be able to deal with them without workarounds. Another crystal clear limitation is convexity, especially considering the recent advent of better performing and more suitable non-convex regularizers [27, 87]. Our scheme relies on the strong convexity of the lower energy to link the magnitude of the error in the approximate gradient to the difference between the last two iterates, which is a key passage to prove the convergence. However, this assumption also serves a second purpose. It forces the energy to have a unique minimizer. In the non-convex case there can be multiple non-global minima and the convergence analysis needs to take into account that the approximate solution obtained from the lower level may just be a local minimum.

Chapter 3

Algorithm Unrolling

The ideas presented in the previous chapter had the fundamental assumption that we are willing to carry out the energy minimization as far as necessary, in order to get an approximation of the minimum that satisfies certain convergence conditions. This had the obvious consequence of having to optimize the energy, sometimes even multiple times for each outer iteration, even though error still needs to be considered as the minimum is not exact. There is an alternative way of framing the problem of learning the parameters of the variational model, which is going to be the object of this chapter. We are talking about *algorithm unrolling/unfolding* which is going to be described in section 3.1 along with some other data driven methods that are natural comparisons to it. Sections 3.2 and 3.3 are dedicated to two different applications of this strategy.

3.1 Framework Definition

The problem of learning the unrolling of an algorithm can be formulated as

$$\begin{cases} \bar{\boldsymbol{\theta}} \in \underset{\boldsymbol{\theta}\in\Theta\subseteq\mathbb{R}^p}{\operatorname{argmin}} \sum_{s=1}^{|\mathcal{D}|} L(\bar{\mathbf{u}}_s(\boldsymbol{\theta}), \boldsymbol{\theta}) \\ \text{subject to } \bar{\mathbf{u}}_s = \mathcal{A}^{(K)}(\mathbf{u}_0; \mathbf{y}_s, \boldsymbol{\theta}). \end{cases} \tag{3.1}$$

Essentially, we give up the requirement that $\mathbf{u}$ is constrained to be within a certain distance from a minimum of the energy. Here, $\mathcal{A}$ represents any optimization algorithm, suitable for the energy functional E, that is applied K times starting from the initial point $\mathbf{u}_0$. Needless to say, the choice of $\mathcal{A}$ and K has to produce an image that is somewhat close to the true minimum, in order to feed the upper level with relevant information.
In practice, the composition of the K steps of $\mathcal{A}$ becomes the restoration algorithm, as once problem (3.1) is solved, we just need to apply the learned iterations to clean a new image $\mathbf{y}$. While the scheme is still tied to the energy E, it also has more room

for movement. For instance, one of the goals of problem (3.1) is still that of learning the parameters of E, but this time it can be extended so that $\boldsymbol{\theta}$ also includes (some of) those of the optimization algorithm itself.
This reformulation takes the name of *algorithm unrolling/unfolding*. As far as we are aware of, was originally proposed in [34], but as gained a lot of attention in both the imaging and machine learning communities [41, 47, 61, 69, 70, 74]. In particular, [34] shows in detail how to compute the loss gradient. In fact, as the lower level solution is now exact, the gradient of the loss function can now be obtained by repeated applications of the chain rule. For simplicity, we will consider the derivative w.r.t. only one parameter θ_j and for one sample. In this passage, we are going to make use of both the total derivative notation and the partial one to help make it more clear w.r.t. what we are differentiating. Then, we have that

$$\frac{\mathrm{d}L}{\mathrm{d}\theta_j}(\bar{\mathbf{u}}(\boldsymbol{\theta}),\boldsymbol{\theta}) = \boldsymbol{\nabla}_{\mathbf{u}} L(\bar{\mathbf{u}},\boldsymbol{\theta})^T \frac{\mathrm{d}\bar{\mathbf{u}}}{\mathrm{d}\theta_j}(\boldsymbol{\theta}) + \frac{\partial L}{\partial \theta_j}(\bar{\mathbf{u}},\boldsymbol{\theta}) \tag{3.2}$$

$$= \boldsymbol{\nabla}_{\mathbf{u}} L(\bar{\mathbf{u}},\boldsymbol{\theta})^T \frac{\mathrm{d}\mathcal{A}}{\mathrm{d}\theta_j}(\mathbf{u}^{(K-1)}(\boldsymbol{\theta});\boldsymbol{\theta}) + \frac{\partial L}{\partial \theta_j}(\bar{\mathbf{u}},\boldsymbol{\theta}). \tag{3.3}$$

If we return to the formulation (2.9) introduced in the previous chapter, then we can assume that $R_L^0(\boldsymbol{\theta})$ is differentiable and R_L^1 can be dealt with using an appropriate outer optimizer. Then, the main difficulty lies in obtaining the total derivative of $\mathcal{A}$ w.r.t. θ_j. The first thing we notice is that we actually do not need to form it, as we are interested in its product with a vector $\mathbf{v} \in \mathbb{R}^n$. To compute the scalar product

$$\mathbf{v}^T \frac{\mathrm{d}\mathcal{A}}{\mathrm{d}\theta_j}(\mathbf{u}^{(K-1)}(\boldsymbol{\theta});\boldsymbol{\theta}) \tag{3.4}$$

there are two main modalities.

- A *backward mode* or *reverse mode*, where, in a first phase, one needs to compute and store the sequence of iterates $(\mathbf{u}^{(k)})_{k=0,\dots,K}$, and then with a second loop the chain rule is applied starting from the last iterate. Observing that

$$\mathbf{v}^T \frac{\mathrm{d}\mathcal{A}}{\mathrm{d}\theta_j}(\mathbf{u}^{(K-1)}(\boldsymbol{\theta});\boldsymbol{\theta}) = \mathbf{v}^T \boldsymbol{\nabla}_{\mathbf{u}} \mathcal{A}(\mathbf{u}^{(K-1)};\boldsymbol{\theta}) \underbrace{\frac{\mathrm{d}\mathbf{u}^{(K-1)}}{\mathrm{d}\theta_j}}_{=\frac{\mathrm{d}\mathcal{A}}{\mathrm{d}\theta_j}(\mathbf{u}^{(K-2)}(\boldsymbol{\theta});\boldsymbol{\theta})} + \mathbf{v}^T \frac{\partial \mathcal{A}}{\partial \theta_j}(\mathbf{u}^{(K-1)}(\boldsymbol{\theta});\boldsymbol{\theta})$$

$$= \mathbf{v}^T \boldsymbol{\nabla}_{\mathbf{u}} \mathcal{A}(\mathbf{u}^{(K-1)};\boldsymbol{\theta}) \boldsymbol{\nabla}_{\mathbf{u}} \mathcal{A}(\mathbf{u}^{(K-2)}(\boldsymbol{\theta});\boldsymbol{\theta}) \frac{\mathrm{d}\mathbf{u}^{(K-2)}}{\mathrm{d}\theta_j}$$

$$+ \mathbf{v}^T \boldsymbol{\nabla}_{\mathbf{u}} \mathcal{A}(\mathbf{u}^{(K-1)};\boldsymbol{\theta}) \frac{\partial \mathcal{A}}{\partial \theta_j}(\mathbf{u}^{(K-2)};\boldsymbol{\theta}) + \mathbf{v}^T \frac{\partial \mathcal{A}}{\partial \theta_j}(\mathbf{u}^{(K-1)}(\boldsymbol{\theta});\boldsymbol{\theta}),$$

  it is possible to compute the product (3.4) following algorithm 3.1. This scheme has been used for quite some time in other fields as well. Namely it is known as backpropagation in deep learning contexts [59].

- A *forward* mode, which perhaps is a bit more straightforward as the derivatives are computed alongside the iterates, without the need for a second loop. Noting that the initial iterate $\mathbf{u}^{(0)}$ does not depend on $\boldsymbol{\theta}$, the product (3.4) can be obtained as the output of algorithm 3.2

Algorithm 3.1: Backward Mode

Input: $(\mathbf{u}^{(k)})_{k=0,\dots,K} \subset \mathbb{R}^n$

Set

$$\mathbf{z}^{(K)} = \mathbf{v}^T \in \mathbb{R}^{1\times n}$$
$$w = 0$$

for $k = K, K-1, \dots, 1$ **do**

Update

$$w_j = w_j + \mathbf{z}^{(k)} \frac{\partial \mathcal{A}}{\partial \theta_j}(\mathbf{u}^{(k-1)}; \boldsymbol{\theta})$$
$$\mathbf{z}^{(k-1)} = \mathbf{z}^{(k)} \nabla_{\mathbf{u}} \mathcal{A}(\mathbf{u}^{(k-1)}; \boldsymbol{\theta})$$

end

Output: $\mathbf{v}^T \frac{\mathrm{d}\mathcal{A}}{\mathrm{d}\theta_j}(\mathbf{u}^{(K-1)}(\boldsymbol{\theta}); \boldsymbol{\theta}) = w_j$.

Although the two modes yield the same final output, their implementations bear significant differences. From a theoretical point of view, the forward mode is, in general, preferable. The fact that the derivatives are compute online and also the inner iterates do not need to be stored makes the modality more efficient. However, in practice, this is often not the case. If the number of parameters p is quite large, in particular when $p >> n$, then the tables are turned. As a matter of fact, we have expressed two modalities only for one parameter, but obviously the computations are carried out simultaneously for all $\boldsymbol{\theta}_j, j = 1, \dots, p$. At each iteration of the backward mode needs to store the vector $\mathbf{w} \in \mathbb{R}^p$ and $\mathbf{z}^{(k)} \in \mathbb{R}^n$, which is the same for every parameter θ_j. On the other hand, the forward mode requires the update of the variables $\mathbf{r}^{(k)}, \mathbf{q}^{(k)} \in \mathbb{R}^{n\times p}$, where there is also a lot of redundant information stored multiple times.

Until this point we have not specified any kind of assumption on the functions in question. As we have said, the upper level loss function is assumed to be in the form (2.9), where the term $F_L(\mathbf{u})$ needs to be differentiable. Regarding $\mathcal{A}$ itself, in order to be able to apply either one of the algorithms here presented, we need that the general iteration of $\mathcal{A}$ is differentiable w.r.t. $\mathbf{u}$ and $\boldsymbol{\theta}$. This property is achieved, for instance, if $\mathcal{A}$ is a first order method, i.e., it only involves E or $\nabla_{\mathbf{u}} E$, and E is twice continuously differentiable. However, as we already discussed, this requirement is a bit strict and excludes a multitude of powerful functionals, unless some relaxation is

Algorithm 3.2: Forward Mode

Input: $\mathbf{u}^{(0)}$

Set

$$\mathbf{r}^{(0)} = \frac{\partial \mathcal{A}}{\partial \theta_j}(\mathbf{u}^{(0)}; \boldsymbol{\theta})$$
$$\mathbf{q}^{(0)} = \mathbf{r}^{(0)}$$

for $k = 1, 2, \ldots, K-1$ **do**

Compute

$$\mathbf{u}^{(k)} = \mathcal{A}(\mathbf{u}^{(k-1)}; \boldsymbol{\theta})$$
$$\mathbf{r}^{(k)} = \frac{\partial \mathcal{A}(\mathbf{u}^{(k)}; \boldsymbol{\theta})}{\partial \theta_j}$$
$$\mathbf{q}^{(k)} = \boldsymbol{\nabla}_{\mathbf{u}} \mathcal{A}(\mathbf{u}^{(k)}; \boldsymbol{\theta}) \mathbf{q}^{(k-1)} + \mathbf{r}^{(k)}$$

end

Compute $\mathbf{u}^{(K)} = \mathcal{A}(\mathbf{u}^{(K-1)}; \boldsymbol{\theta})$

Compute $\mathbf{v} = \boldsymbol{\nabla}_{\mathbf{u}} L(\mathbf{u}^{(K)}; \boldsymbol{\theta})$

Output: $\mathbf{v}^T \mathbf{q}^{(K-1)}$.

employed. There are, however, other ways to obtain the same regularity. If the energy can be written as $E(\mathbf{u}; \boldsymbol{\theta}) = f_0(\mathbf{u}; \boldsymbol{\theta}) + f_1(\mathbf{u}; \boldsymbol{\theta})$, where f_0 is a continuously differentiable function, and f_1 is proper, convex and lower semicontinuous, FB schemes as described in (2.21) and (2.22) represents a class of first order methods that can be rielaborated to yields the desired properties. In fact, the proximal operator as defined in (2.23) is based on the standard euclidean distance. Bregman distances can help to further generalize the proximal operator with the dual goal of obtaining an explicit closed formula for the minimization in the proximal operator [3, 4] and, at the same time, making the single iteration of $\mathcal{A}$ differentiable. Of course, this approach is not limited to FB schemes but can be applied to a variety of first order proximal algorithms [47, 70].

Algorithm unfolding has another, more subtle, improvement compared to the full bilevel optimization problem: the hessian of the energy functional, even when needed, is not inverted. Furthermore, in practice, the hessian does not even need to be completely formed, since in most cases one just needs its action against a vector.

There is also the obvious question is how to select the hyperparameter K. The most natural strategy is comparing problem (3.1), which is technically not a bilevel problem anymore, to its cousin (2.8). While it is possible, for various choices of $\mathcal{A}$, to prove the convergence of the "unrolled" loss derivative to the "full" one expressed in (2.12), as in [63] for instance, it is never going to be case as the number of iterations

K is fixed a priori, and ideally as low as possible. An interesting study was conducted in [84], where the authors considered the class of quadratic functions with $\mathcal{A}$ being a first order optimization scheme. They observed that the distance between the gradient computed according to algorithm 3.2 or 3.1 and the full one from (2.12) initially increases, and only after a sufficient number of inner iterations it becomes to decrease. These unwanted effects can be mitigated by reducing the inner steplength, which however then leads to remaining farther from the energy minimum and possibly an inferior quality in the final reconstruction.
There is, of course, also the strategy of somehow learning the lower steplengths as well. In [36], the authors observe of often the best image is not actually the minimum of the energy, but is obtained by early stopping. Following this argument, they develop a strategy to learn K from data by means of the optimal control theory. Overall, in most cases, K is left as an hyperparameter of the model. It is set according to the specific task in question as well as in relation to the steplengths in $\mathcal{A}$. Also, K cannot be chosen arbitrarily high, as after some point the procedure suffers of diminishing returns in terms of distance form the minimum of E, while the computational cost increases (especially when the backward mode is used to compute the loss gradient).

3.1.1 Extension to Deep Methodologies

The concept of unrolling an optimization algorithm leads to a final learned model that can be viewed as a very basic *Neural Network* (NN). A NN is essentially a composition of multiple functions $N_1, \dots, N_L$, where most depend on a set of parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_L)$

$$\mathcal{NN}(\mathbf{u}; \boldsymbol{\theta}) = N_L(N_{L-1}(\dots N_2(N_1(\mathbf{u}; \boldsymbol{\theta}_1); \boldsymbol{\theta}_2) \dots; \boldsymbol{\theta}_{L-1}); \boldsymbol{\theta}_L),$$

where L is the number of so called *layers*. Due to their versatility and capabilities of abstracting the images and the data they are presented with, NNs have attracted enormous attention for quite some time now. NNs, as many other *deep learning* methodologies, are able to set all of their parameters through a supervised or unsupervised learning procedure, and once these are learned, they can be used to directly process new data. The connection between NNs and algorithm unfolding is clearly drawn, but there are still differences as well as ways to fill the gaps in between.

Plug and Play Models. Let us consider a general variational model $E(\mathbf{u}) = F(\mathbf{u}) + R(\mathbf{u})$, where F is a differentiable term while R is a convex but non-differentiable function. In the optimization literature there exists many different schemes to tackle the corresponding minimization problem by performing a variable split [28]. One example is the family of FB schemes as in (2.21) and (2.22). Another popular strategy is the Half-Quadratic Splitting which rewrites the problem as

$$\min_{\mathbf{u},\mathbf{z}} F(\mathbf{u}) + R(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{z} - \mathbf{u}\|_2^2 \tag{3.5}$$

and then solves is with the alternating steps

$$\mathbf{u}^{(k+1)} = \text{prox}_{\frac{1}{\rho}F}(\mathbf{z}^{(k)}) \tag{3.6}$$

$$\mathbf{z}^{(k+1)} = \text{prox}_{\frac{1}{\rho}R}(\mathbf{u}^{(k+1)}). \tag{3.7}$$

At this point, the fundamental idea of *Plug and Play* (PnP) models is to replace one or more proximal operators with pre-learned denoisers, which act as the new prior for the images:

$$\mathbf{z}^{(k)} = \mathbf{u}^{(k)} - \alpha \nabla F(\mathbf{u}^{(k)}) \tag{3.8}$$

$$\mathbf{u}^{(k+1)} = \mathcal{NN}_R^{\boldsymbol{\theta}}(\mathbf{z}^{(k)}), \tag{3.9}$$

in the case of a FB scheme, or

$$\mathbf{u}^{(k+1)} = \mathcal{NN}_F^{\boldsymbol{\theta}}(\mathbf{z}^{(k)}) \tag{3.10}$$

$$\mathbf{z}^{(k+1)} = \mathcal{NN}_R^{\boldsymbol{\theta}}(\mathbf{u}^{(k+1)}), \tag{3.11}$$

for the half-quadratic split. This is motivated by the fact that known, explicitly formulated regularizers are not usually easy to handle, or have limitations in their performance. The first obstacle is finding and/or developing the right prior for the specific images in question. A second one is related to its properties. If the regularizer is non convex then the energy admits local minima, which may all lead to different levels of quality in the restoration. Another aspect is whether or not the proximal operator has a closed form solution to its own minimization problem. So, overall, a winning strategy might be that of the proximal operator with a NN, so that we could even draw from distribution that do not have explicit regularizer.
The question whether or not this NN, in the end, is truly a proximal maps always looms over this approach. However, there have been works where the denoiser is indeed shown to be a proximal map [55, 65, 88], in particular when the denoiser is nonexpansive. This, in turn, also ensures the convergence of the resulting iterative algorithm to a stationary point.

Deep Unfolding. *Deep unfolding/unrolling* is a natural variant of the PnP approach. In fact, the iterative schemes described in the previous paragraph can be truncated after a number of iterations, and then learned similarly to an normal algorithm unrolling. In general, the concept of deep unfolding refers to a broader multitude of strategies where the layers of one or more NNs, and the overall pipeline is guided by an optimization algorithm [10, 99], so that the uncertainty of the NNs is mitigated by the solid theoretical background of variational models.

Some considerations are in order. It is true that NNs have been, and still are, revolutionizing the literature thanks to their outstanding results across many disciplines. However, they still come with notable downsides.

What makes deep learning methods so powerful also constitutes their biggest limitation. The data driven nature that is at the core also enforces a non-negligible task dependence. While the same could also be said for variational models, and even more for learned variational models, deep architectures not only needs to be constructed based on the problem at hand, but are also heavily affected by the specific samples that are used in learning. The bias-variance tradeoff or, in broader terms, the generalization capabilities, are always relevant in the design and the training of a NN. Overfitting is a constant obstacle in many machine and deep learning applications, and although there are many tools to address and mitigate it, no clear ever-effective resolution exists. On the other hand, variational models, in virtue of being simpler entities, require less data and thus are a lot less at risk of adapting too much to what they see in training, i.e., there are less chances that a variational models performs way better on the training set than on the test set.
The data driven nature is also strictly connected to another weakness of NNs: they have no interpretability, or barely any. Hybrid schemes such as PnP and deep unfolding aim at giving sense to the structure and the millions of parameters of NNs, so that there is some hope in understanding their behaviour. In practical terms, this lack of understanding translates in an unpredictability of the results. There are countless examples where the NN returns an answer that is just plainly wrong [93]. When this happens because the network was fed with an image that was corrupted to levels beyond those that the network is able to handle, then it is still reasonable. However, when this happens on images that do not have anything wrong (at least to the human eye), then it is a not so insignificant problem. Studies on the stability of NNs have been conducted [37], but still, in the end, the black-box nature of NNs makes them never one hundred percent reliable and it does not allow us to safely predict when they are going to fail nor why. The well established theoretical grounds of variational models, both in terms of functional analysis and practical applications, allow us to have a more precise overall picture and better intuition regarding the outcome.
Finally, in order to have a decent level of stability as well as accuracy, a model usually undergoes multiple design changes and trainings. In light of the recent trend of a green ans sustainable artificial intelligence, the computational resources required for this, i.e., to develop a satisfying deep model should be taken into account. An entire branch of research, called *Neural Architercture Search* [45], is focused on finding an efficient way of designing NNs, exactly because otherwise it can truly be overwhelming.

3.2 The Helsinki Deblur Challenge

One suitable application in which we decided to learn the unrolling of an optimization algorithm was the *Helsinki Deblur Challenge* (HDC) of 2021[1]. This proved to be a

[1] https://www.fips.fi/HDC2021.php

solid benchmark for the proposed approach. This scenario, in fact, not only offers the possibility of learning the variational model, but it also adds a few interesting complications, namely, a blind deconvolution problem and the difficulty of working with real pictures.

3.2.1 Challenge Rules

The primary goal of the challenge was to deconvolve a collection of text images of size $n_1 \times n_2 = 1460 \times 2360$ pixels. These images portrayed black characters against a light background and were written in two distinct fonts: Verdana and Times New Roman. Each image contained a randomized text string comprising 10 characters distributed across three separate lines. These images were categorized into 20 levels, each corresponding to a different degree of blur. As one progressed to higher levels, the images exhibited progressively more degradation. Additionally, for each level, two subsets of 100 images were provided, one for each font. All these images were captured by two digital cameras, denoted as CAM1 and CAM2, simultaneously aimed at the same e-ink screen displaying the text string. CAM1 was in sharp focus and thus served as the benchmark, constituting the first subset of data. The second subset, on the other hand, contained images captured by CAM2, which experienced a misfocus, leading to images that were both blurry and noisy. Furthermore, these images suffered from various optical distortions. Effectively, the dataset consisted of two versions for each text string at every blur level: a high-quality image and a blurred, noisy, and distorted counterpart. Figure 3.1 depicts the setting in which the images were obtained.

Following the conclusion of the challenge, a separate test set was released, consisting of 40 images per level and constructed in a manner almost identical to the original dataset. The most significant difference was the addition of more possible characters, such as numbers and spaces. the quality of an image was assessed by subjecting it to Optical Character Recognition (OCR) software, which was included with the initial dataset. This assessment was based on the Levenshtein distance [60], which essentially returns the smallest possible number of substitutions, insertions or deletions needed to transform one string into another. It is also important to note that the OCR would always first try to identify the three lines, and if it recognized anything other than three, whethere it be more or less, it would instantly assign a score of 0.

Competing algorithms were evaluated against the test set, and achieving an average score of at least 70 out of 100 correctly recognized characters by the OCR software signified a successful pass at the respective level. Notably, the images captured by CAM2 in the first three levels managed to clear the OCR test as they were, without requiring any additional post-processing. However, in the later levels, the degree of blur reached an extreme level. It is important to highlight that the OCR score exhibited a higher sensitivity to blur as compared to noise or warp, making it a viable

metric for assessing image quality under these specific conditions.

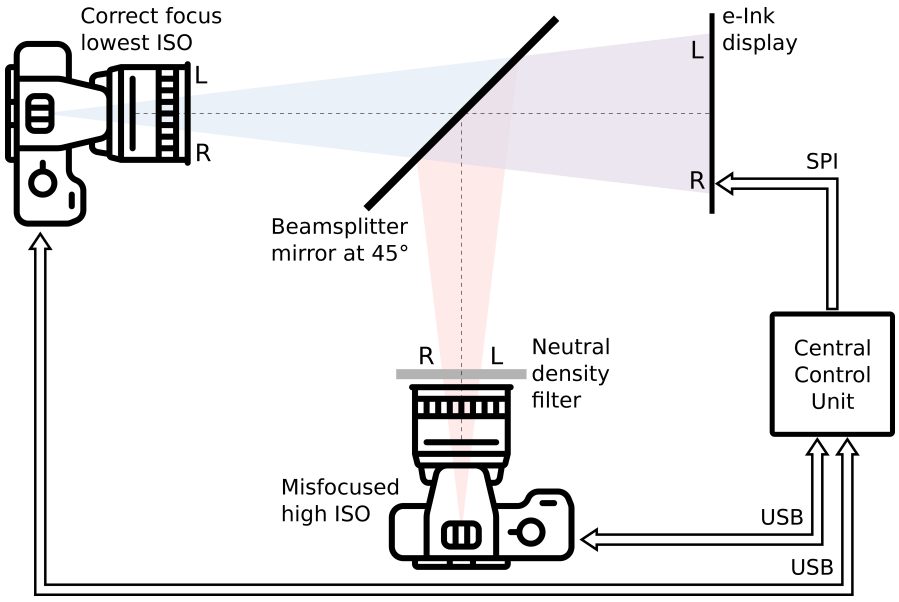


Figure 3.1: Experimental setup for the HDC.

3.2.2 The Variational Model

Let us first work on the underlying variational model, as it is at the center of our proposed framework. For this challenge, we assumed the following acquisition model for the images:

$$\mathbf{y} = \mathbf{H}(r)(\mathbf{u}^{(\text{true})} + \mathbf{bg}) + \boldsymbol{\sigma}, \qquad \mathbf{u}, \mathbf{y} \in \mathbb{R}^n, \tag{3.12}$$

with $n = n_1 \times n_2$. The noise $\boldsymbol{\sigma}$ is assumed to be a realization of a Gaussian distribution. Gaussian noise is the most common, and there are no other signs or indicators that the images are affected by other forms of noise. Both the ground truths and the noisy samples presented a background **bg** distortion, which if not addressed withing the energy functional would most probably cause issues. In particular, it would hinder the variational model in producing sharp, sparse and binary images, which are some of the main characteristics of text images. Last, but not least, the operator $H(r)$, that with an abuse of notation represents both the linear operator and the corresponding matrix, is the main culprit of the challenge and represents the convolution operator. The HDC is structured as a blind deconvolution problem, as the challenge itself did not provide any information regarding the PSF. However, a preliminary analysis showed us that the data was afflicted by an out-of-focus blur, or something very similar. This helped us giving a reasonable, although not perfect, structure to the kernel. An out-of-focus blurring kernel can be considered as a flat disc, with its radius r controlling the intensity of the degradation. We used the MATLAB built-in function `fspecial` to obtain, whenever needed, a disc kernel and its derivatives with

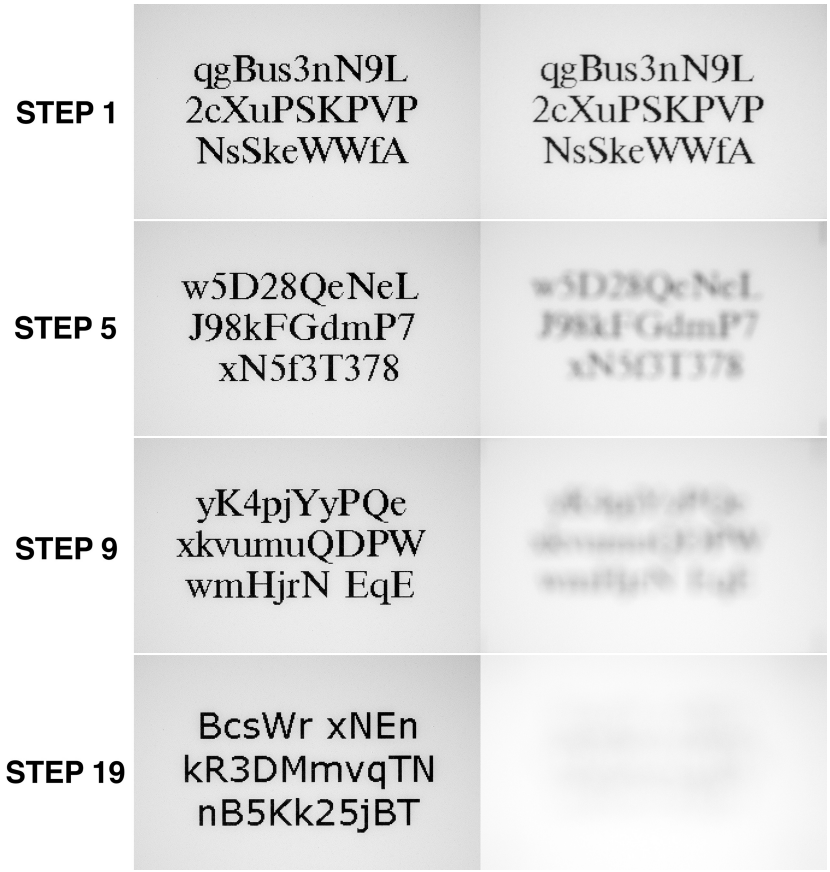


Figure 3.2: Sample images from the test set for selected levels. Level 19 is the last level, and in the blurred images (right column) even the human eye is incapable of discerining any character whatsoever.

respect to the parameter r.

The variational model featured the constraint $\mathbf{u} \in [0,1]^n$ in combination with two regularization terms:

- The indicator function
$$\iota_\Omega(\mathbf{u}) := \begin{cases} 0 & \mathbf{u} \in \Omega \\ +\infty & \mathbf{u} \notin \Omega, \end{cases} \tag{3.13}$$
when $\Omega = [0,1]^n$ is tasked with enforcing the boundaries for the pixels.
- The Total Variation (TV) is extremely well versed in this context. The main drawback of the TV is the unpleasing visual effect of having big image patches with constant pixels. Indeed, while it is proficient in reconstructing the edges of the images, sparsifying the gradient also produces the effect of having the region between the edges to be flat. This turns out to be a perk in recovering characters, as having a well defined edge is as important as having a solid black character

when feeding the image to the OCR. Formally, the TV term used was

$$TV_\delta(\mathbf{u}) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sqrt{(u_{i,j}-u_{i,j+1})^2+(u_{i,j}-u_{i+1,j})^2+\delta^2}. \tag{3.14}$$

Here, the small adjustment is the presence of δ to obtain a smooth functional.

- As a prior for binary images, we employed the following function

$$B(\mathbf{u}) = \frac{1}{2}\sum_{i=1}^{n} u_i(1-u_i). \tag{3.15}$$

This is a downward parabola which serves the purpose of forcing the pixels towards the boundary of the feasible region, i.e., to be either 0 or 1, in order to highlight and define as much as possible the characters.

To sum up, the energy functional involved was

$$E_{\mathbf{y}}(\mathbf{u};\boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{H}(r)\mathbf{u}+\mathbf{bg}-\mathbf{y}\|_2^2+\rho B(\mathbf{u})+\gamma TV(\mathbf{u};\delta), \tag{3.16}$$

where the parameters to learn are $\boldsymbol{\theta} = (r,\rho,\gamma,\delta)$.

3.2.3 Unrolling Technique

The *Fast Iterative Soft-Thresholding Algorithm* (FISTA) [5, 26] is an extremely renowned and effecient optimization methods which can be employed to solve an optimization problem of the form

$$\min_{\mathbf{u}\in\mathbb{R}^n} f_0(\mathbf{u}) + f_1(\mathbf{u}), \tag{3.17}$$

where $f_0\colon \mathbb{R}^n \to \mathbb{R}$ is a convex function that is also continuously differentiable on a closed convex set $\Omega \subset \mathbb{R}^n$, while $f_1\colon \mathbb{R}^n \to \bar{\mathbb{R}}$ is proper, lower semicontinuous and convex function with $\Omega \subset \operatorname{dom} f_1$. Since its first publication, it has attracted the attention of most of the optimization community, and several variants have been proposed. Here, we use the following FISTA iteration:

$$\begin{aligned}\mathbf{v}^{(k)} &= P_\Omega(\mathbf{u}^{(k)}+\beta(\mathbf{u}^{(k)}-\mathbf{u}^{(k-1)}))\\ \mathbf{u}^{(k+1)} &= \operatorname{prox}_{\alpha_k f_1}(\mathbf{v}^{(k)}-\alpha_k\nabla\mathbf{f_0}(\mathbf{v}^{(k)})).\end{aligned}$$

$P_\Omega(\cdot)$ represents the orthogonal projection onto the set Ω and the proximal operator is defined as in equation (2.23). Furthermore, α_k and β_k are the steplength and extrapolation parameters, respectively. The convergence properties of the above iteration have been established in [18], for a suitable set of $(\alpha_k)_{k\in\mathbb{N}}$ and $(\beta_k)_{k\in\mathbb{N}}$. Results in the non-convex set have also been established such as those in [68].
The optimization problem $\min_{\mathbf{u}} E_{\mathbf{y}}(\mathbf{u};\boldsymbol{\theta})$ can be dealt with the FISTA iteration described, by setting $\Omega = [0,1]^n$, $f_0 = E_{\mathbf{y}}$ and $f_1 = \iota_{[0,1]^n}(\mathbf{u})$. It is well known that

the proximal operator of the indicator function of a set reduces to the orthogonal projection onto said set. This proximal operator would not normally be problematic. However, computing the gradient loss is much easier to handle if the single algorithm iteration $\mathcal{A}$ is smooth, as we have desscribed in section 3.1. To achieve the desired differentiability, we developed the following component-wise projection function:

$$\Pi(s) = \begin{cases} \max\{\min\{s, 1\}, 0\} & \text{if } s \in [-\infty, 0] \cup [\epsilon, 1-\epsilon] \cup [1, +\infty] \\ \left(2 - \frac{s}{\epsilon}\right) \frac{s^2}{\epsilon} & \text{if } s \in [0, \epsilon] \\ 1 - \left(2 - \frac{1-s}{\epsilon}\right) \frac{(1-s)^2}{\epsilon} & \text{if } s \in [1-\epsilon, 1] \end{cases} \tag{3.18}$$

where ϵ is a positive parameter. In practice, the function coincides with the Euclidean projector everywhere outside of $[0, \epsilon]$, $[1-\epsilon, 1]$, a third degree polynomial interpolates through the points $(0, 0)$, (ϵ, ϵ) and, by symmetry, $(1-\epsilon, 1-\epsilon)$, $(1, 1)$. A behaviour that is as close as possible to the Euclidean projection is more synergistic with the concave function 3.15. Especially compared to other projection-like functions, like the interior barrier function from [10] or the sigmoid function, it maps the points outside of the box exactly to the closest bound of the box. In our experiments, we have found this to be a crucial aspect to recover good quality images. However, it also suffers from the unavoidable oscillations that appear when interpolating via polynomials. In Figure 3.3 we compare our proposed projection to the standard Euclidean one and that from [10] in the interval $[0, 0.1]$. In the experiments that will be discussed later, we always set $\epsilon = 10^{-4}$.

Algorithm 3.3 reports the general unrolling procedure based on the elements presented in this paragraph. Similarly to the convex case, we use the extrapolation step $\beta_k = \frac{k-1}{k+2}$ to accelerate the energy decrease through the iterations, as shown in [26, 66]. When dealing with a deblurring problem via variational models, it is important to get as close as possible to the minimum, in order to reduce the ringing effects in the reconstruction. This is especially true in a blind deconvolution problem where the PSF used in the model is not even the correct one, emphasizing the Gibbs phenomenon. In our energy the regularization terms, especially the concave function, help attenuating these artifacts, but it still proves necessary either apply the iteration $\mathcal{A}$ for a notable amount of times or to accelerate the convergence.
Regarding the steplength α_k we consider them as parameters to be learned along with the energy functional parameters $\boldsymbol{\theta}$. Then, denoting by $\boldsymbol{\alpha} \in \mathbb{R}^K$ the vector containing all the steplength parameters, we have the full picture with all the dependencies of the lower level solution:

$$\bar{\mathbf{u}} = \bar{\mathbf{u}}(\boldsymbol{\theta}, \boldsymbol{\alpha}).$$

The idea of including the FISTA iteration in a learning procedure has been already proposed in several contexts, often in combination with neural networks and deep learning techniques (see for example [1, 10, 51, 97] and references therein). As explained above, here we consider the FISTA unrolling in the framework of bilevel optimization, which is closer to the traditional variational approach.

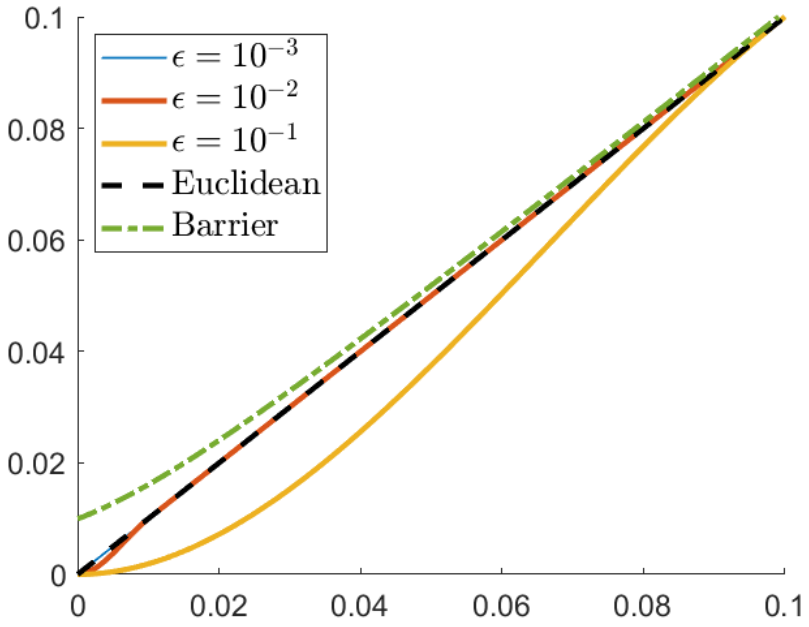


Figure 3.3: Plot of the projection-like function 3.18 in the interval $[0, 1]$, for different choices of the parameter ϵ. The dashed line shows the standard Euclidean projection, while the dash-dotted line represents the interior barrier projection proposed in [10] with parameter 10^{-4}.

Finally, we compute the gradient of the loss function with the backword mode described in Algorithm 3.1. Here we describe the details of said scheme in this specific instance, which can be summarised by following Algorithm 3.4. For the sake of ease of notation, we consider the partial derivative of the loss function L with respect to only one parameter θ_j. First, we can observe that

$$\begin{aligned}
\mathbf{v}^{(k)} &= \Pi((1+\beta_k)\mathbf{u}^{(k)} - \beta_k \mathbf{u}^{(k-1)}) \\
\frac{\partial \mathbf{v}^{(k)}}{\partial \theta_j} &= \boldsymbol{\Pi}'(\bar{\mathbf{v}}^{(k)}) \left((1+\beta_k)\frac{\partial \mathbf{u}^{(k)}}{\partial \theta_j} - \beta_k \frac{\partial \mathbf{u}^{(k-1)}}{\partial \theta_j} \right).
\end{aligned}$$

With a small abuse of notation, $\boldsymbol{\Pi}'$ denotes the jacobian matrix of the projection $\Pi(\cdot)$, which is diagonal since the projection acts component-wise. Exploiting the previous formula we get:

Algorithm 3.3: FISTA-like deconvolution procedure

Input: $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{u}^{(0)}, \mathbf{u}^{(-1)} = \mathbf{y}$
for $k = 0, \ldots, K$ **do**
  $\bar{\mathbf{v}}^{(k)} = \mathbf{u}^{(k)} + \beta_k(\mathbf{u}^{(k)} - \mathbf{u}^{(k-1)})$
  $\mathbf{v}^{(k)} = \Pi(\bar{\mathbf{v}}^{(k)})$
  $\mathbf{t}^{(k)} = \mathbf{v}^{(k)} - \alpha_k \nabla_{\mathbf{u}} E_{\mathbf{y}}(\mathbf{v}^{(k)}; \boldsymbol{\theta})$
  $\mathbf{u}^{(k+1)} = \Pi(\mathbf{t}^{(k)})$
end
Output: $\bar{\mathbf{u}} = \mathbf{u}^{(K+1)}$.

$$
\begin{aligned}
\frac{\partial \mathbf{u}^{(k+1)}}{\partial \theta_j} &= \boldsymbol{\Pi}'(\mathbf{t}^{(k)}) \left(\left(\mathbf{I_n} - \alpha_k \nabla^2_{\mathbf{uu}} E(\mathbf{v}^{(k)}; \boldsymbol{\theta}) \right) \frac{\partial \mathbf{v}^{(k)}}{\partial \theta_j} - \alpha_k \frac{\partial}{\partial \theta_j} \nabla_{\mathbf{u}} E(\mathbf{v}^{(k)}, \boldsymbol{\theta}) \right) \\
&= (1+\beta_k) \boldsymbol{\Pi}'(\mathbf{t}^{(k)}) \left(\mathbf{I_n} - \alpha_k \nabla^2_{\mathbf{uu}} E(\mathbf{v}^{(k)}; \boldsymbol{\theta}) \right) \boldsymbol{\Pi}'(\bar{\mathbf{v}}^{(k)}) \frac{\partial \mathbf{u}^{(k)}}{\partial \theta_j} + \\
&\quad - \beta_k \boldsymbol{\Pi}'(\mathbf{t}^{(k)}) \left(\mathbf{I_n} - \alpha_k \nabla^2_{\mathbf{uu}} E(\mathbf{v}^{(k)}; \boldsymbol{\theta}) \right) \boldsymbol{\Pi}'(\bar{\mathbf{v}}^{(k)}) \frac{\partial \mathbf{u}^{(k-1)}}{\partial \theta_j} + \\
&\quad - \alpha_k \boldsymbol{\Pi}'(\mathbf{t}^{(k)}) \frac{\partial}{\partial \theta_j} \nabla_{\mathbf{u}} E(\mathbf{v}^{(k)}; \boldsymbol{\theta}).
\end{aligned}
$$

This is the $j-th$ column of the Jacobian matrix for the function $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k+1)}(\boldsymbol{\theta})$. Using these findings and a smart variable notation, we can define a procedure to compute the gradient that avoids building explicitly the Jacobian of $\mathbf{u}^{(k+1)}$, and is also well versed for the practical experiments as all the computations can be done efficiently in the context of the HDC. The inizialization reads:

$$
\mathbf{z}^{(K+1)} = \nabla_{\mathbf{u}} L(\bar{\mathbf{u}}) \qquad \mathbf{r}^{(K+1)} = \mathbf{0} \qquad w_j^{(K+1)} = 0,
$$

Then

$$
\begin{aligned}
\frac{\partial L}{\partial \theta_j}(\bar{\mathbf{u}}) &= \left[\frac{\partial \bar{\mathbf{u}}}{\partial \theta_j} \right]^T \nabla_{\mathbf{u}} L(\bar{\mathbf{u}}) = \left[\frac{\partial \mathbf{u}^{(K+1)}}{\partial \theta_j} \right]^T \nabla_{\mathbf{u}} L(\bar{\mathbf{u}}) \\
&= \left[\frac{\partial \mathbf{u}^{(K+1)}}{\partial \theta_j} \right]^T \mathbf{z}^{(K+1)} + \left[\frac{\partial \mathbf{u}^{(K)}}{\partial \theta_j} \right]^T r^{(K+1)} + w_j^{(K+1)} \\
&= (1+\beta_K) \left[\frac{\partial \mathbf{u}^{(K)}}{\partial \theta_j} \right]^T \Pi'(\bar{\mathbf{v}}^{(K)}) \left(\mathbf{I_n} - \alpha_K \nabla^2_{\mathbf{uu}} E(\mathbf{v}^{(K)}, \boldsymbol{\theta}) \right) \boldsymbol{\Pi}'(\mathbf{t}^{(K)}) \mathbf{z}^{(K+1)} \\
&\quad - \beta_K \left[\frac{\partial \mathbf{u}^{(K-1)}}{\partial \theta_j} \right]^T \boldsymbol{\Pi}'(\bar{\mathbf{v}}^{(K)}) \left(\mathbf{I_n} - \alpha_K \nabla^2_{\mathbf{uu}} E(\mathbf{v}^{(K)}, \boldsymbol{\theta}) \right) \boldsymbol{\Pi}'(\mathbf{t}^{(K)}) \mathbf{z}^{(K+1)} \\
&\quad - \alpha_K \left[\frac{\partial}{\partial \theta_j} \nabla_{\mathbf{u}} E(\mathbf{v}^{(K)}, \boldsymbol{\theta}) \right]^T \boldsymbol{\Pi}'(\mathbf{t}^{(K)}) \mathbf{z}^{(K+1)} + \left[\frac{\partial \mathbf{u}^{(K)}}{\partial \theta_j} \right]^T \mathbf{r}^{(K+1)} + w_j^{(K+1)}.
\end{aligned}
$$

Algorithm 3.4: FISTA-like backpropagation procedure for a parameter θ_j

Input: $\bar{\mathbf{u}}$ from Algorithm 3.3
Set $\mathbf{z}^{(K+1)} = \boldsymbol{\nabla}_{\mathbf{u}} L(\bar{\mathbf{u}})$, $\mathbf{r}^{(K+1)} = \mathbf{0}$ and $w_j^{(K+1)} = 0$.
for $k = K, K-1 \ldots, 1$ **do**
  $\bar{\mathbf{z}}^{(k)} = \boldsymbol{\Pi}'(\mathbf{t}^{(k)})\mathbf{z}^{(k+1)}$
  $w_j^{(k)} = -\alpha_k \left[\frac{\partial}{\partial \theta_j} \boldsymbol{\nabla}_{\mathbf{u}} E(\mathbf{v}^{(k)}; \boldsymbol{\theta})\right]^T \bar{\mathbf{z}}^{(k)}$
  $\mathbf{q}^{(k)} = \boldsymbol{\Pi}'(\mathbf{v}^{(k)}) \left(\mathbf{I_n} - \alpha_k \boldsymbol{\nabla}^{\mathbf{2}}_{\mathbf{uu}} E_{\mathbf{y}}(\mathbf{v}^{(k)}; \boldsymbol{\theta})\right) \bar{\mathbf{z}}^{(k)}$
  $\mathbf{z}^{(k)} = (1 + \beta_k)\mathbf{q}^{(k)} + \mathbf{r}^{(k+1)}$
  $\mathbf{r}^{(k)} = -\beta_k \mathbf{q}^{(k)}$
end
Output: $\frac{\partial L}{\partial \theta_j}(\bar{\mathbf{u}}) = w_j^{(1)} - \alpha_0 \left[\frac{\partial}{\partial \theta_j} \boldsymbol{\nabla}_{\mathbf{u}} E(\mathbf{v}^{(0)}; \boldsymbol{\theta})\right]^T \boldsymbol{\Pi}'(\mathbf{t}^{(0)})\mathbf{z}^{(1)}$.

Now by defining

$$
\begin{aligned}
z^{(k)} &= (1 + \beta_k)\boldsymbol{\Pi}'(\bar{\mathbf{v}}^{(k)}) \left(\mathbf{I_n} - \alpha_k \boldsymbol{\nabla}^{\mathbf{2}}_{\mathbf{uu}} E(\mathbf{v}^{(k)}), \boldsymbol{\theta})\right) \boldsymbol{\Pi}'(\mathbf{t}^{(k)}))\mathbf{z}^{(k+1)} + \mathbf{r}^{(k+1)} \\
r^{(k)} &= -\beta_k \boldsymbol{\Pi}'(\bar{\mathbf{v}}^{(k)}) \left(\mathbf{I_n} - \alpha_k \boldsymbol{\nabla}^{\mathbf{2}}_{\mathbf{uu}} E(\mathbf{v}^{(k)}, \boldsymbol{\theta})\right) \boldsymbol{\Pi}'(\mathbf{t}^{(k)})\mathbf{z}^{(k+1)} \\
w_j^{(k)} &= -\alpha_k \left[\frac{\partial}{\partial \theta_j} \boldsymbol{\nabla}_{\mathbf{u}} E(\mathbf{v}^{(k)}, \boldsymbol{\theta})\right]^T \boldsymbol{\Pi}'(\mathbf{t}^{(k)})\mathbf{z}^{(k+1)} + w_j^{(k+1)},
\end{aligned}
$$

we get

$$
\frac{\partial L}{\partial \theta_j} = \left[\frac{\partial \mathbf{u}^{(K)}}{\partial \theta_j}\right]^T \mathbf{z}^{(K)} + \left[\frac{\partial \mathbf{u}^{(K-1)}}{\partial \theta_j}\right]^T \mathbf{r}^{(K)} + w_j^{(K)}.
$$

Proceeding with the recursion, and using the notation indicated, we get the desired derivative as written in Algorithm 3.4.

3.2.4 Learning the Model

We can now turn our attention to the learning problem, which is described only for one training sample. We need to solve

$$
\begin{aligned}
&\min_{\boldsymbol{\theta}, \boldsymbol{\alpha}} \; L(\bar{\mathbf{u}}(\boldsymbol{\theta}, \boldsymbol{\alpha})) \equiv L(\mathcal{A}^{(K)}(\mathbf{y}; \boldsymbol{\theta}, \boldsymbol{\alpha})) \\
&\text{s. to } \boldsymbol{\theta} \in \Theta \subset \mathbb{R}^4, \boldsymbol{\alpha} \in [\alpha_{\min}, \alpha_{\max}]^K.
\end{aligned} \tag{3.19}
$$

The parameters of the model are bounded to be in a reasonable space dictated by their role in the variational model.

A key aspect of the methodolgy that we used to solve the HDC is the choice of the merit function L. It is tasked with a number of objectives. The images obtained from $\mathcal{A}$ need to be sufficiently cleaned from the blur and noise in order to received the

highest score possible from the OCR. At the same time, it has to be differentiable and it would not hurt if it is also easy to handle with known optimization schemes. Retaining both there purposes, we devise two different strategies for this parameter learning phase: the first one is as standard as it can get, and revolves around comparing the reconstruction $\bar{\mathbf{u}}$ to the ground truth image $\mathbf{x} \in \mathbb{R}^n$, that is provided by the HDC; the second is focused on building a differentiable and "nice" enough OCR score predictor. The latter, in particular, works around a fundamental assumption that can be a weakness of the first method, i.e., the ground truth $\mathbf{x}$ needs available and suitable to use.

Supervised SSIM-Based Merit Function. Here we want to measure the distance of the reconstruction from its corresponding ground truth. An aspect that must not be neglected, is the choice of the metric function that penalizes errors. This can be influenced by many factors. In the HDC, due to the unknown distortion that afflicts each and every image, ground truths included, a loss function that compares the images pixel by pixel is not the best course of action. With these considerations we adopted the *Structural SIMilarity index* (SSIM) as it was presented in [100]:

$$SSIM(\mathbf{u}, \mathbf{x}) = \frac{1}{m} \sum_{i=1}^{m} S_i^{(1)}(\mathbf{u}, \mathbf{x}) \cdot S_i^{(2)}(\mathbf{u}, \mathbf{x}), \tag{3.20}$$

with

$$S_i^{(1)}(\mathbf{u}, \mathbf{x}) = \frac{2\mu_i^{\mathbf{u}}\mu_i^{\mathbf{x}} + C_1}{(\mu_i^{\mathbf{u}})^2 + (\mu_i^{\mathbf{x}})^2 + C_1}, \quad S_i^{(2)}(\mathbf{u}, \mathbf{x}) = \frac{2\sigma_i^{\mathbf{ux}} + C_2}{\sigma_i^{\mathbf{u}} + \sigma_i^{\mathbf{x}} + C_2}. \tag{3.21}$$

The quantities $\boldsymbol{\mu}^{\mathbf{u}}, \boldsymbol{\mu}^{\mathbf{x}}, \boldsymbol{\sigma}^{\mathbf{u}}, \boldsymbol{\sigma}^{\mathbf{x}}, \boldsymbol{\sigma}^{\mathbf{ux}}, \in \mathbb{R}^m$ are defined through a convolution matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ that acts as a mask by selecting the correct pixels:

$$\begin{aligned} \boldsymbol{\mu}_i^{\mathbf{u}} &= \sum_{j=1}^{n} W_{ij} u_j, & \mu_i^{\mathbf{x}} &= \sum_{j=1}^{n} W_{ij} x_j, \\ \sigma_i^{\mathbf{u}} &= \sum_{j=1}^{n} W_{ij} u_j^2 - (\mu_i^{\mathbf{u}})^2, & \sigma_i^{\mathbf{x}} &= \sum_{j=1}^{n} W_{ij} x_j^2 - (\mu_i^{\mathbf{x}})^2, \\ \sigma_i^{\mathbf{ux}} &= \sum_{j=1}^{n} W_{ij} \mu_j^{\mathbf{u}} \mu_j^{\mathbf{x}} - \mu_i^{\mathbf{u}} \mu_i^{\mathbf{x}}, & i &= 1, \dots, m. \end{aligned}$$

In the experiments $\mathbf{W}$ is a Gaussian filter with standard deviation 1.5 and size 11, $C_1 = 10^{-4}$ and $C_2 = 3 \cdot 10^{-4}$, which are standard choices for images with pixels in $[0, 1]$.

The gradient of the SSIM can be computed as follows:

$$\lambda_i = \frac{S_i^{(2)}(\mathbf{u},\mathbf{x})}{(\mu_i^{\mathbf{u}})^2 + (\mu_i^{\mathbf{x}})^2 + C_1}\left(\mu_i^{\mathbf{u}} - \mu_i^{\mathbf{x}}\frac{2\mu_i^{\mathbf{u}}\mu_i^{\mathbf{x}} + C_1}{(\mu_i^{\mathbf{u}})^2 + (\mu_i^{\mathbf{x}})^2 + C_1}\right)$$

$$\phi_i = \frac{S_i^{(1)}(\mathbf{u},\mathbf{x})}{\sigma_i^{\mathbf{u}} + \sigma_i^{\mathbf{x}} + C_2}\left(\mu_i^{\mathbf{x}} - \mu_i^{\mathbf{u}}\frac{2\sigma^{\mathbf{ux}} + C_2}{\sigma_i^{\mathbf{u}} + \sigma_i^{\mathbf{x}} + C_2}\right)$$

$$\nu_i = \frac{S_i^{(1)}(\mathbf{u},\mathbf{x})}{\sigma_i^{\mathbf{u}} + \sigma_i^{\mathbf{x}} + C_2}$$

$$\xi_i = S_i^{(2)}(\mathbf{u},\mathbf{x}) \cdot \nu_i,$$

for $i = 1, \ldots, m$, and then, for $j = 1, \ldots, n$,

$$\boldsymbol{\nabla}_{\mathbf{u}}\boldsymbol{SSIM}(\mathbf{u},\mathbf{x})_j = -\frac{2}{m}\sum_{i=1}^{m} W_{ij}(\lambda_i - \phi_i) + \frac{2}{m}x_j\sum_{i=1}^{m} W_{ij}\nu_i - \frac{2}{m}u_j\sum_{i=1}^{m} W_{ij}\xi_i.$$

The SSIM assigns to $\mathbf{u}$ a score of 1 when it is $\mathbf{x}$, and 0 when there is similarity whatsoever. Hence, a suitable loss function becomes:

$$L(\bar{\mathbf{u}}(\boldsymbol{\theta},\boldsymbol{\alpha})) = 1 - SSIM(\bar{\mathbf{u}}(\boldsymbol{\theta},\boldsymbol{\alpha}),\mathbf{x}). \tag{3.22}$$

Unsupervised OCR Score Predictor Based Merit Function. In real world application, having a dataset with clear ground truths and images with the same characteristics of those in object is definitely not always a given. In this scenario, an unsupervised training procedure becomes necessary. The ground truths of the HDC could not be considered the real clean images that we seek to obtain, as they were far from being clean. Indeed, as described in section 3.2.1, the so called "true images" from CAM1 are taken simultaneously as those from CAM2 and the true only difference in the acquisition procedure is the fact that CAM2 has misfocused lens. Hence, except for the blur, the "true images" were corrupted by noise and the inevitable camera distortion. This is enough to justify a more difficult to control unsupervised training model. The goal then becomes to produce an image that maximizes the OCR score. There is also a second reason that prompted us to follow this path. The judgment of the OCR is faulty. It is possible to have crystal clear images that still do not achieve a perfect score. Thus, predicting how good a reconstruction would score is a way to try to have the training adjusting to this uncontrollable flaw.
The OCR is given by the HDC, but as a black-box. Using directly that as merit function is not a road that can be pursued in our scheme. Instead, we trained a standard regression model by feeding it an image and the OCR score. The choice of the regressor is left to us. Differentiability still remains a requirement, thus all machine learning techniques based on binary decisions, like Random Forests, are automatically excluded. We also feel that deep learning frameworks, besides increasing

the computational cost for the backpropagation, would necessitate of an amount of data that is beyond the Green AI philosophy that is one of the underlying motivations of this work. A *Support Vector machine for Regression* (SVR) [29, 35] was then our function of choice. An SVR is a standard machine learning tool that has already been employed as a performance predictor, e.g. in [45], with similarly to how it is done here. To train the SVR, separate training dataset $\mathcal{D}_{\text{SVR}} = \{(\mathbf{y}_s, l_s)\}_{s=1}^{N_s}$, with $\mathbf{y}_s$ being an image and l_s its OCR score. Let us denote with l_{pred} the predicted score given to an image $\mathbf{u}$ according to

$$l_{\text{pred}} = P(\mathbf{u}) = \sum_{s=1}^{N_s} (\bar{\zeta}_s - \bar{\zeta}_s^*) K(\mathbf{u}^{(s)}, \mathbf{u}) + \bar{b}, \tag{3.23}$$

having $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ as a kernel function [83] and $\boldsymbol{\zeta}, \bar{\boldsymbol{\zeta}}^* \in \mathbb{R}^{N_s}$ as the solution to optimization problem

$$\begin{aligned} \min_{\boldsymbol{\zeta}, \boldsymbol{\zeta}^* \in \mathbb{R}^{N_s}} \quad & \frac{1}{2} \sum_{s,s'=1}^{N_s} (\zeta_s - \zeta_s^*)(\zeta_{s'} - \zeta_{s'}^*) K(\mathbf{y}_s, \mathbf{y}_{s'}) + \epsilon \sum_{s=1}^{N_s} (\zeta_s + \zeta_s^*) - \sum_{s=1}^{N_s} l_s (\zeta_s - \zeta_s^*) \\ \text{s. to} \quad & \sum_{s=1}^{N_s} \zeta_s - \zeta_z^* = 0 \\ & 0 \le \zeta_s \le C, \quad 0 \le \zeta_s^* \le C \quad s = 1, \dots, S. \end{aligned} \tag{3.24}$$

The bias $\bar{b}$ is determined by either one of the following relations:

$$\begin{aligned} \bar{b} &= l_{s'} - \sum_{s=1}^{N_s} (\bar{\zeta}_s - \bar{\zeta}_s^*) K(\mathbf{y}_s, \mathbf{y}_{s'}) - \epsilon \\ \bar{b} &= l_{s'} - \sum_{s=1}^{N_s} (\bar{\zeta}_s - \bar{\zeta}_s^*) K(\mathbf{y}_s, \mathbf{y}_{s'}) + \epsilon \end{aligned} \tag{3.25}$$

for a given training sample $(\mathbf{u}_s, l_s)$ for which we $0 < \zeta_{s'} < C$ or $0 < \zeta_{s'}^* < C$. The hyperparameters C and ϵ in (3.24)-(3.25) are tasked with balancing the tradeoff between accuracy and generalization. In particular C acts as an upper bound for the coefficients ζ_s, ζ_s^*, while ϵ determines the insensitive region for the SVR. In the practical case, we chose a Gaussian kernel function:

$$K(\mathbf{u}, \mathbf{u}') = e^{-\frac{\|\mathbf{u} - \mathbf{u}'\|^2}{2\sigma^2}}, \qquad \sigma > 0. \tag{3.26}$$

With these instruments, we can define the unsupervised merit function as:

$$L(\bar{\mathbf{u}}(\boldsymbol{\theta}, \boldsymbol{\alpha})) = e^{-\frac{P(\bar{\mathbf{u}}(\boldsymbol{\theta}, \boldsymbol{\alpha}))}{100}}, \tag{3.27}$$

where P is the function from equation (3.23). Since the OCR has a score between 0 and 100, we divided the predicted score by 100 as a normalization. The training of

the SVR was done in MATLAB. We followed its formulation to derive the formula for the gradient of the merit function:

$$\nabla_{\mathbf{u}} L(\bar{\mathbf{u}}(\boldsymbol{\theta},\boldsymbol{\alpha})) = \frac{1}{100} e^{-\frac{P(\bar{\mathbf{u}}(\boldsymbol{\theta},\boldsymbol{\alpha}))}{100}} \sum_{s=1}^{N_s} (\bar{\zeta}_s - \bar{\zeta}_s^*) e^{-\frac{\|\bar{\mathbf{u}}(\boldsymbol{\theta},\boldsymbol{\alpha}) - \mathbf{y}_s\|^2}{2\sigma^2}} \left(\frac{\bar{\mathbf{u}}(\boldsymbol{\theta},\boldsymbol{\alpha}) - \mathbf{y}_s}{\sigma^2} \right)$$

3.2.5 Numerical Experience

In this part we give the full details for the practical implementation of the tools described until now. Including how we built the dataset to train the SVR, a more precise definition of the distortion effect and how we dealt with it in the SSIM-based loss function and of course how our method compares to the results of the HDC when the winner was declared.

Dataset Definition for SSIM Optimization. For the reasons already introduced, the HDC dataset need to undergo a pre-processing procedure. The very first step was to invert the intensity of the pixels so that all the images featured white characters on a (ideally) black background and rescale the pixels values to be in $[0, 1]$. We also resized the images to an eighth of their original size. This operation had the dual goal of making handling the otherwise big images easier, and to help in reducing the blurring present. Indeed, considering what a PSF does, it is natural to deduce that any image resize algorithm is going to mitigate its effects.
The background was empirically estimated for each level of the challenge. This was done by extracting the frame around the three lines of text for all corrupted images and then averaging all these frames. As for the central part that was cut out, we filled it with by interpolation. This consituted the term **bg** in equation (3.16). The reader can visualize the estimated background for level 10 in Figure 3.4 (c) and (d).
The distortion effect present on the images is not a pressing issue when trying to obtain a high OCR score, but it certainly becomes one when trying to penalize wrong reconstructions through a supervised loss function. Figure 3.4 (e) contains the binarization of a reconstruction (highlighted in blue for visualization purposes), and in red the borders of the same characters of the ground truth. As the reader can see, there is not an exact match. This needs to be corrected. The type and degree of the distortion are properties inherent to the camera used. In fact, it is technically possible to estimate having the camera on hand, which unfortunately is not our case. We observed that a rather basic radial model determined by two hyperparameters, i.e., the center and radius of the distortion, is close enough to the true model. These hyperparameters were manually tuned for each blur level, all with the same strategy.
Let $\mathbf{x}_{\text{CAM1}}$ be the binarization of a clean image and let (p_i, q_j), with $i = 1, \dots, n_1$ and $j = 1, \dots, n_2$, be the spatial coordinates of the nodes of a grid corresponding the the image pixels. The we "correct" the distortion in $\mathbf{x}_{\text{CAM1}}$ with the function $x\colon \mathbb{R}^2 \to \mathbb{R}$:

$$x(\bar{p}_i, \bar{q}_j) = [x_{CAM1}]_{ij}, \quad \text{where} \begin{cases} \bar{p}_i &= c_p + R(p_i - c_p) \\ \bar{q}_j &= c_q + R(q_j - c_q) \end{cases},$$

where $R > 0$ and (c_p, c_q) are the radius and the center of the distortion. Abusing the notation a little, $\mathbf{x}$ also denotes the image obtained by sampling the interpolation function over the gridpoints.
The tuning of the hyperparameters R, c_x, c_y was carried out in the following way: a decent reconstruction from the corresponding image from CAM2 was obtained with a simple TV model, which takes into account also the estimated background; then we binarized said restoration, let us denote with $\hat{\mathbf{u}}$ said binarization; finally, the hyperparameters are moved so that the edges of the characters in $\hat{\mathbf{u}}$ match those in $\mathbf{x}$. See Figure 3.4 (f) for reference. Although this distorion model could have been included in the lower level, we opted to use it to preprocess the "ground truth" as to avoid further complications.

In summary, the preprocessing consisted in:

- Each blurred image $\mathbf{y}$ is flipped, resized and then the estimated background is subtracted from it.
- Each ground truth $\mathbf{x}$ is flipped, resized and warped with the radial distortion model.

OCR Predictor via Support-Vector machine for Regression. The SVR used as a performance predictor becomes a useful tool, that is however task dependent. In fact, it needs to be trained with a very significant dataset in order to be able to capture all the possible outcomes of the underlying model.
For the HDC we created a mixed dataset with images from the challenge as well as synthetic images with very similar properties. The synthetic images were included mainly to have a better representation in the dataset, as the images provided by the challenge, in most cases, achieved either a score above 60 or flat 0. This happened every time the OCR would not pick up exactly three lines, which was the norm from a certain level onward. As explained before, the SVR also wants to see possible outcomes from the unrolling, especially considering that an overestimated radius for the PSF leads to a ringing effect, which is detrimental to the OCR score. For this reason we also included reconstructions obtained with random set of parameters $(\boldsymbol{\theta}, \boldsymbol{\alpha})$ in the dataset $\mathcal{D}_{\text{SVR}}$. More specifically, it was composed as follows:

- 400 images obtained by CAM2, 50 for each level from 1 through 8.
- 1230 synthetic images. For each score interval $[0, 10],]10, 20], \ldots,]90, 100]$, 123 images were selected.
- 429 extra images. Most of them were obtained by applying the unrolling to the samples from CAM2 with random parameters. We also included some specific "structured" images, such as constant images, as we noticed the SVR would otherwise assign a high score to them.

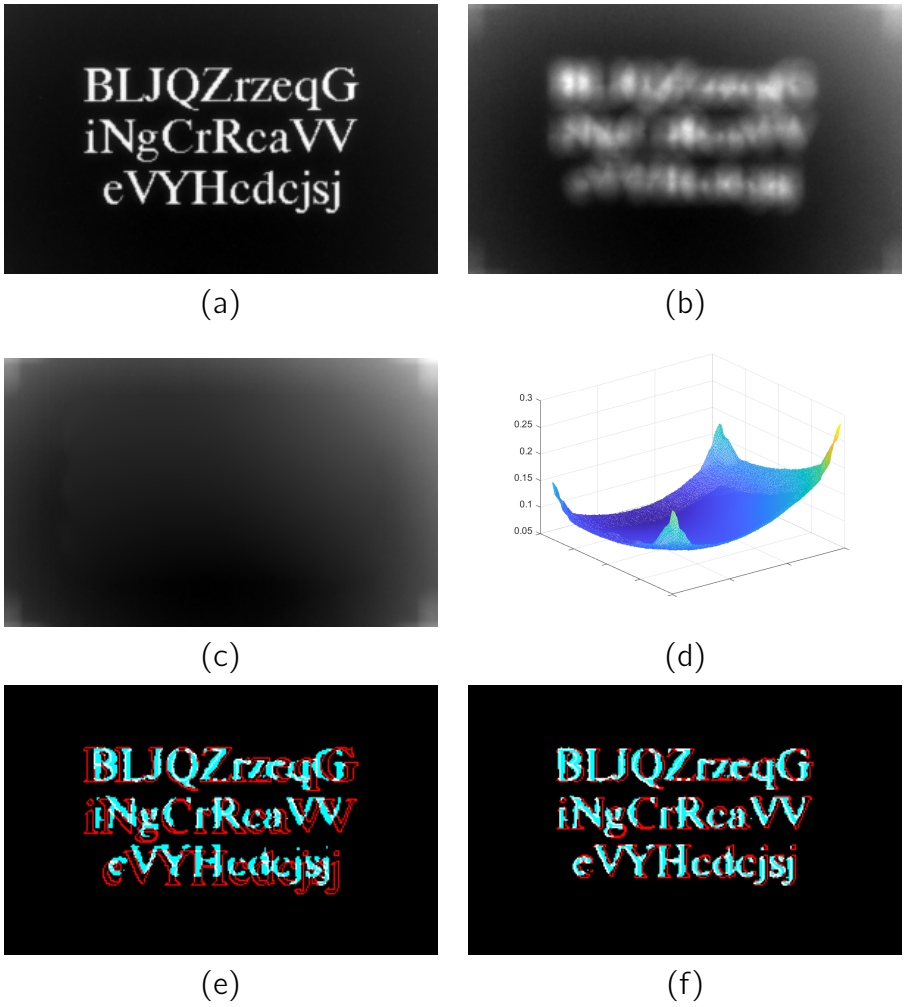


Figure 3.4: Dataset definition for the SSIM loss function. Upper row: original HDC data from step 10, Times New Roman, image n.51, CAM1 (a) and CAM2 (b). Middle row: estimated background, represented as an image (c) and as a surface (d). Bottom row: estimated distortion. Panel (e) shows the binarized reconstruction (light blue) of image (b) obtained with a variational method and the edges of the binarized version of (a) (red). The white lines correspond to the pixels where the two images are superimposed. In panel (f), the red edges are obtained after applying the estimated radial distortion to (a).

The images underwent a similar preprocessing procedure. In particular they were again flipped, resized to a quarter of their original size and rescaled to have pixels between 0 and 1. We also noticed that these operations, besides helping the variational model, also improved the performance of the SVR. However, before being given to the OCR, they were flipped again a upscaled to their original size. In Figure 3.5 we show one SVR sample per kind. The synthetic image does not have any distortion or noise, and unsurprisingly performs the best, while the last one clearly illustrates

how the ringing effect ruins the score.
The hyperparameters of the SVR were finetuned with crossvalidation strategies. The final values employed in the model were $\epsilon = 4.8$ and $C = 48$. The average error on an isolated test set was 16 OCR points.

(a) (b) (c)

Figure 3.5: Samples from the SVR training dataset. (A) Noisy image from the step 4 of the Helsinki dataset (OCR score = 52); (B) Synthetic image blurred with a disc PSF of radius 9 (OCR score = 84); (C) Reconstruction with wrong parameters for the variational model (OCR score = 11).

Setting of the Unrolling Framework. The unrolling scheme was tested for three different numbers of iterations, namely $K = 60, 70, 80$. As explained before, the more blur is present on the image, the more iterations we need to let $\mathcal{A}$ run for.
The parameters $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ were constrained to be in sensible ranges, which were determined easily in the process of choosing a decent starting point. This boundaries also serves as the only form of regularization, and limit the chances for the merit function of overfitting and introducing undesired artifacts on the restorations. The outer optimization problem was handled with the *Scaled Gradient Projection* method (SGP) [16, 17, 21]. The algorithm was stopped either when the distance of successive values of the merit function was less than 10^{-7} or after 50 iterations. Different steplengths were used, in particular for each element of $\boldsymbol{\theta}$, since the components of the loss gradient had different scaling levels.
The variational training phase only required 4 samples, indipendelty of the merit function used. This is a clear showing of the advantages that the framework and ideas presented in this elaborate possess, as other teams, including the eventual winners, had to obtain a not so insignificant amount of images exactly as those from the HDC. Using a limited number of training images also balanced the computational cost of the backpropagation. In particular, the optimization of the SSIM-based merit function (3.22) took about 15-30 minutes on a workstation equipped with a multicore CPU Intel(R) Core(TM) i7-6700 CPU @3.40GHz, on the other hand, for the SVR merit function (3.27) a few hours were required on the same architecture. All the routines have been implemented and run in Matlab R2021b.

Results. Although the variational model is just the one in (3.16), it was trained for levels 6, 8, 10, and 12, once for each level. This is not surprising, nor outside of the rules of the challenge. Afterall, the PSF does change through the levels, thus the algorithm needs to adapt to it. The testing phase, where we collected the reconstructions and their OCR score, was conducted on the test sets for the said levels. This phase was significantly less expensice, as most of the computational efforts were dedicated to the learning phase. Indeed, to infer a reconstruction one just needs to apply the learned iterations of $\mathcal{A}$ to the new corrupted image. It is worth pointing out that the background estimation was not repeated for the test images, but we kept the same one that was used for the training.
The time required to compute a new reconstruction by applying the iterations of $\mathcal{A}$ is clearly dependent on the number K of them. Another aspect to remember is that the SVR approach only resized each image dimension by a factor of $\frac{1}{4}$, while the SSIM loss function used a factor of $\frac{1}{8}$. This obviously affects the restoration time, even with the same number of iterations. Averaging the recorded times for 5 runs, the SVR-trained unrolling required 1.15, 1.29 and 1.57 seconds for $K = 60, 70$ and 80, respectively. On the other hand, the SSIM-trained unrolling took 0.37, 0.44 and 0.50 seconds.

Table 3.1 reports the final average true OCR scores for the images from the test sets. For each level, it was composed by a total of 40 images, 20 per font. By this time we have mentioned a few times how more iterations are, in theory, an advantage in terms of restoration quality when it comes to deconvolution. However, the numbers obtained for the different choices of K, a bit surprisingly, do not show any sign this. Also, there is not a clear choice to what the best number of K is across all levels. Perhaps this can be attributed in the learning of the steplength $\boldsymbol{\alpha}$ of $\mathcal{A}$, that can make up for a lesser number of iterations.

Table 3.2 offers a more in depth illustrations of the SVR as a performance predictor, while Figures 3.6 and 3.7 contain the images related to this table. The first column is the ground truth, the second is the blurred sample and the last one is the restoration obtained. Overall we observe a bit of conservatism in the scores assigned by the SVR, as they rarely get to either 100 or 0 (note that the SVR output does not have bounds, i.e., scores could have been much bigger or even in the negative numbers). The average error of 16 OCR points certainly seems significant, however, from the table we can appreciated how the SVR is still able to differentiate between very good images, decent reconstructions and the blurred samples. It is also worth noticing that the scores predicted by the SVR for the presented restorations are indeed close to the final performance of our algorithm.

Table 3.1: Average OCR scores obtained on the 40 test images of each step. Up until level 10 would be a pass, level 12 would have been failed.

	$K = 60$		$K = 70$		$K = 80$	
	SSIM	SVR	SSIM	SVR	SSIM	SVR
Step 6	85.20	85.60	85.60	82.45	85.08	83.28
Step 8	83.88	82.63	84.15	81.80	82.45	80.13
Step 10	70.88	73.90	71.35	76.30	72.72	73.23
Step 12	60.23	61.53	61.73	48.58	61.90	61.53

Table 3.2: Comparison between the SVR prediction and the true OCR score for the images of Figure 3.6 and 3.7.

	Ground truth		Noisy sample		Reconstruction	
	OCR	SVR	OCR	SVR	OCR	SVR
Step 6 Times	100	81.84	0	42.17	100	71.71
Step 6 Verdana	100	81.91	0	45.64	100	74.18
Step 8 Times	100	81.91	0	34.44	90	70.1
Step 8 Verdana	100	81.98	0	37.68	86	75.48
Step 10 Times	100	81.92	0	31.22	90	66.32
Step 10 Verdana	100	81.96	0	30.67	100	68.44
Step 12 Times	100	82.03	0	24.07	76	67.42
Step 12 Verdana	100	82.03	0	29.05	70	71.53

Regarding Table 3.3, without claiming to be fully exhaustive of extremely precise, we briefly discuss here the methodologies proposed by other teams:

- Team 15_A, from the Technische Universität Berlin, Institut für Mathematik in Berlin, Germany, proposed a end-to-end deblurring neural network, with a similar architecture to that of the standard U-Net [80], which involves around 2.6 million parameters. As one of the rules of the HDC was that the algorithms needed to be for deconvolution in general settings, and not just of text images, some expedients to avoid overfitting were incorporated.
- Team 12_B, from the Institution Department of Mathematics, National University of Singapore, proposed to use the a deep learning approach DeblurGanV2 [58] but without using the GAN training loss.
- Team 01, from Leiden University, Leiden, The Netherlands, used a Mixed-Scale Dense CNNs [73] to deblur text images.

Figure 3.6: From left to right: ground truth, noisy sample and reconstruction using the SVR merit function with $K = 70$ inner iterations. From top to bottom, two images, one per font, are selected for the steps 6 and 8. The corresponding scores are shown in Table 3.2.

- Team 11_C, from ZeTeM Uni Bremen Team, employed an inversion model, the StepNet, a fully-learned model that operates in a stratifies fashion. It consists in 20 sequential sub-networks. Each one of them is tasked with removing the blur from level i in order to produce an image with blur from level $i - 1$. All the sub-netoworks were small UNets[80].
- Team 06, from the University of Düsseldorf, Department of Computer Science, Germany, first augmented the data with the DIV2K dataset, then trained a neural network to deconvolve samples from both the DIV2K dataset and the HDC dataset itself. The neural network proposed is an adaptation from [40], with an architecture similar to a UNet (see [49]).
- Team 13, from the Federal University of ABC, Center for Engineering, Modeling and Applied Social Sciences, Brazil, based their approach on the Deep Image Prior (DIP), originally from [96], which is an unsupervised scheme that only requires the blurred image. However, Instead of using just the DIP, a secondary DNN with a

Figure 3.7: From left to right: ground truth, noisy sample and reconstruction using the SVR merit function with $K = 70$ inner iterations. From top to bottom, two images, one per font, are selected for the steps10 and 12. The corresponding scores are shown in Table 3.2.

bottleneck architecture (like an autoencoder) is proposed to facilitate the deblurring goal, by incorporating additional (prior) information from the clean images as well.

- Team 16_B, from the Technical University of Denmark, DTU Compute Denmark, implemented a deconvolution algorithm with an estimation procedure for the PSF radius [76].
- Team 09_B, from the University of Campinas (UNICAMP), School of Electrical and Computer Engineering, Brazil, in a first step, they obtained an estimation of the PSF from special dot images provided by the HDC, then used this information to deconvolve the image with the inverse-problems fixed-point scheme REgularization by Denoising (RED) [79].

If we compare them to those in Table 3.3, i.e., the final standings at the conclusion of the challenge in 2021, we can appreciated how our method places relatively well,

especially under the light shed by the analysis of the methods in Table 3.3. Our proposed method would have placed just above Team 11_C, as neither them nor us were able to go further than step 10. However, the striking difference is in the number of parameters that our unrolled algorithm uses. While $\mathcal{A}$ is dependent by at most 84 parameters, all the other deep approaches had no less than hundreds of thousands of parameters, averaging over one million. This of course can translates in a lot more computational resources used. Also, our algorithm still maintains all the established theoretical properties of variational models and SVRs, but still performs better than the purely classical ones from Teams 16_B and 09_B. This is a promising sign that this mixed strategy between machine learning and variational models can lead to satisfying results.

Table 3.3: Results of the HDC published in November 2021. The number of used parameters for the other teams is an estimation based on the model used or from the github codes in the HDC page.

Team	**Step 6**	**Step 8**	**Step 10**	**Step 12**	**#parameters**
15_A	94.03	93.12	93.75	91.42	2.6 millions
12_B	92.62	92.62	85.80	85.95	11 millions
01	91.75	91.65	88.67	87.12	0.187 millions
11_C	87.78	81.25	79.15	62.80	52 millions
06	94.33	85.92	70.17	0.00	2.6 millions
13	71.12	67.12	54.38	64.83	2.2 millions
16_B	76.45	68.35	4.03	7.42	3
09_B	6.33	2.27	2.62	4.03	4

Finally, we recall that the HDC, by virtue of being a general blind deconvolution challenge, wanted to promote algorithms that worked for as many image classes as possible, and not just text images. This was implemented in the rules of the challenge by means of a "sanity check", which consisted in applying the proposed methodologies to other types of images and controlling whether the restorations showed any signs that the algorithms were calibrated only for text images. The quality of the reconstruction did not influence if this sanity check was passed or not. Figure 3.8 includes some of the general images of this check and what happens when Algorithm 3.4 is used on them. In particular, the images shown are from step 6 of the challenge. We note that the training was done for text images, and the energy functional is tailored for that kind of samples. This is enough to explain why these reconstructions may not be satisfactory, like the deer picture. This is both a flaw and a perk of bilevel/unrolling approaches. While the energy needs to be carefully

chosen to cater toward the specific images in question, the philosophy and the ideas behind are transferable to a variety of applications and contexts.

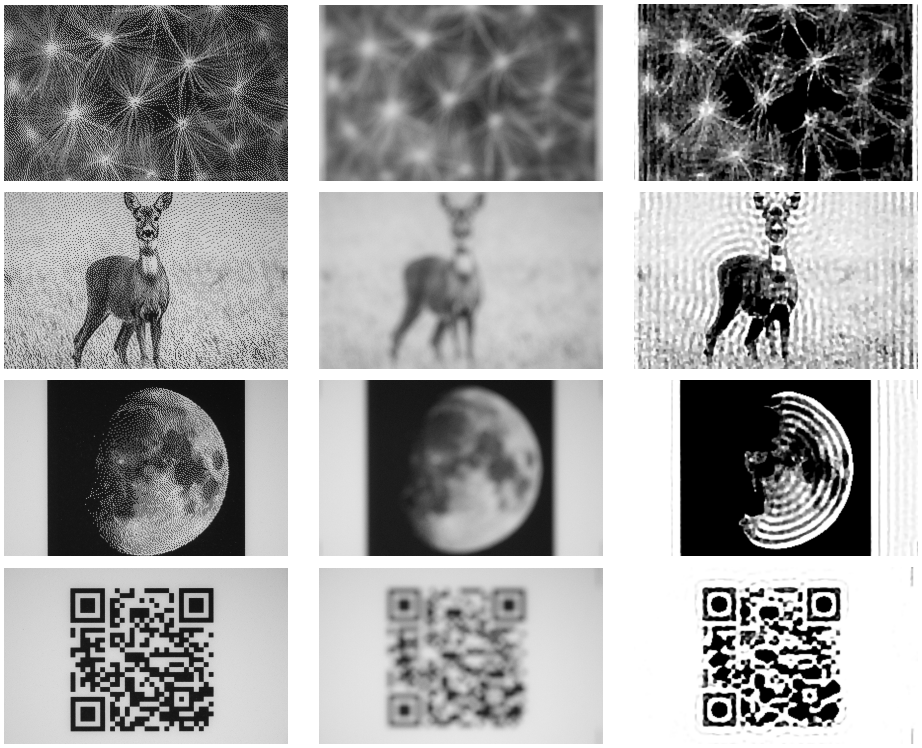


Figure 3.8: Results of Algorithm 3.3 with the SSIM loss function and 60 unrolled iterations applied to some of the test images employed for the sanity check (step 6).

Possible Improvements

The specific tools presented in this section can clear favor some modifications to obtain a better performance and/or more generalization.
For one, the modelling of the corruption presented are perhaps a little on the simpler side. A more complex model for the PSF, such as, for instance, an elliptic shape, would have probably led to better results and more levels being passed.
Another key aspect that is worth pursuing is the upper level unsupervised merit function. Indeed, being able to train the model without ever needing the exact ground truths of the images involved can definitely boost the versatility of the algorithm. In particular, what was done here had the help of having a clear reference in the OCR. A similar approach applied to natural images, say a PSNR-predictor, requires a lot more thought: withouth any form of feature extraction, natural images have way less in common among them than structured text images. Turning to No-Reference quality measures such as BRISQUE [64] may be an answer to this. Another advantage is that we could actually train the unrolling scheme $\mathcal{A}$ once for each corrupted sample,

in the same spirit as the DIP [96], something that does not make sense in a supervised setting. Finally, it may also free us from the constraint in the image dimensions: as it is, the SVR wants as input images always with $n = n_1 n_2$ pixels.

3.3 Single Molecule Localization Microscopy

In a variety of applications from the biomedical field to astronomy, the technical instruments are physically unable to produce images at an adequate resolution, due to light phenomena. This can cause the obvious issue where two objects become indistinguishable, such as two close stars in the distance. In the field of fluorescent microscopy, fluorophores are attached to the molecules of interest in order for them to be more easily observed. Since the information regarding the positions of said molecules is carried through light, it becomes necessary to develop techniques to recover the images at an higher resolution.
Single Molecule Localization Microscopy (SMLM) encompasses a large family of methods, such as PALM [12] or STORM [82], that aim at obtaining the desired resolution by capturing many under-sampled acquisitions of the object, but where only a handful of molecules are activated each time. Each image, is then passed through a *super-resolution* scheme, and at the same time is cleaned of the inevitable noise and blur present. Finally, all the frames can be summed together to obtain the full picture. Obviously, the success of this methodology is affected by the density of the molecules in each frame: more acquisitions can lead to better quality images, but can also damage the object that is to be observed; reducing the number of acquisitions, and thus increasing the density of the molecules, can again lead to having smaller features to be lost. One possible strategy to address the issue is to enforce sparsity when restoring each frame.
In this section we are going to explore the use of algorithm unrolling in the context of SMLM problems. In particular, the first three experiments in subsection 3.3.4 are all focused on testing different choices in super-resolution problems, while the last one offers a semi-blind deconvolution similar to that in 3.2, but in a different domain and, technically a different operator.

3.3.1 Problem Definition

Here we describe the general super-resolution problem which is going to be the object of the first three experiments (for the details of the last one we refer the reader directly to the dedicated paragraph in subsection 3.3.4). Super-resolution viewed under an inverse problem formulation means that, given an observation

$$\mathbf{y} = N(\mathbf{A}\mathbf{u}^{\text{true}} + \mathbf{bg}) \in \mathbb{R}^m = \mathbb{R}^{\sqrt{m}\cdot\sqrt{m}}, \tag{3.28}$$

we want to recover an image $\mathbf{u} \in \mathbb{R}^n = \mathbb{R}^{\sqrt{n}\cdot\sqrt{n}}$ with $n = m\lambda^2$, where λ is the desired super-resolution factor. We also assume that m and n are perfect squares

for simplicity in the description. This means that the operator $\mathbf{A}\colon \mathbb{R}^n \to \mathbb{R}^m$ is the composition of two operators:

- A convolution matrix $\mathbf{H}\colon \mathbb{R}^n \to \mathbb{R}^n$ which is caused by the lens of the microscope. In our experiments this is modeled as Gaussian kernel, as it is one of the more common types of blur in these microscopy applications.
- A super-resolution operator $\mathbf{S}\colon \mathbb{R}^n \to \mathbb{R}^m$, which is responsible for the down-sampling of the image. Obviously, the down-sampling can be carried out in different ways. In our experiments, we use the super-resolution operator as in [48], which is defined through

$$\mathbf{M}_\lambda = \begin{pmatrix} \mathbf{1}_\lambda^T & \mathbf{0}_\lambda^T & \cdots & \mathbf{0}_\lambda^T \\ \mathbf{0}_\lambda^T & \mathbf{1}_\lambda^T & \cdots & \mathbf{0}_\lambda^T \\ \vdots & & & \\ \mathbf{0}_\lambda^T & \mathbf{0}_\lambda^T & \cdots & \mathbf{1}_\lambda^T \end{pmatrix} \in \mathbb{R}^{\sqrt{m}\times\sqrt{n}}, \tag{3.29}$$

  so that, considering the images in their matrix form, we have

$$\mathbf{S}\mathbf{u} = \mathbf{M}_\lambda \mathbf{u} \mathbf{M}_\lambda^T. \tag{3.30}$$

  In practice, $\mathbf{S}$ computes the sum of non-overlapping patches of dimension $\lambda \times \lambda$ when applied to $\mathbf{u}$.

3.3.2 The Model

We now describe in detail the variational model that we employed to solve the SMLM problems. We also illustrate the training procedure, which algorithm $\mathcal{A}$ we decided to use.

3.3.3 The Proposed Model

The variational model at the center of this application is characterized by the following energy functional:

$$E_{\mathbf{y}}(\mathbf{u};\theta) = F(\mathbf{A}\mathbf{u};\mathbf{y},\mathbf{b}\mathbf{g}) + \rho\|\mathbf{u}\|_1 + \iota_{\mathbb{R}^n_+}(\mathbf{u}), \tag{3.31}$$

Where the indicator function $\iota_{\mathbb{R}^n_+}$ is defined as in (3.13). Note that the non-negativity constraint makes it so that the ℓ_1-norm is differentiable, as it becomes a simple sum of the pixels. One of the best possible regularizers that can be used in SMLM would be the ℓ_0-norm, which is technically not even a norm. Unfortunately, minimizing such functional is extremely hard, thus it is sensible to instead turn to a suitable approximation. One of the scopes of this work is to somehow test the limits ℓ_1-norm in super-resolution and deconvolution of sparse images in SMLM. Indeed, it is well

known that in this kind of problems the ℓ_1-norm can have issues in producing reconstructions that are sparse enough. Its core weakness, compared to other regularizers, is its convexity, which limits the ability of the ℓ_1-norm of approximating the ℓ_0-norm. By adopting a strategy that is supposed to find the best value for the regularization for ρ, we hope that the ℓ_1-norm can still produce decent results. Moreover, in order to further improve the quality of the reconstruction, we also developed other tools to help alleviate the flaws of the ℓ_1-norm.

Since we are going to work with images corrupted with either Gaussian or poisson noise, we tested two different data fidelity terms:

- $F(\mathbf{u};\mathbf{y}) = \frac{1}{2}\|\mathbf{Au}-\mathbf{y}\|_2^2$
- $F(\mathbf{u};\mathbf{y},bg) = KL(\mathbf{u};\mathbf{y},\mathbf{bg}) = \sum_{i=1}^{m}(Au)_i + bg - y_i - y_i \log\left(\frac{(Au)_i+bg}{y_i}\right)$,

where $bg \in \mathbb{R}$ represents a constant background when present, or a very small term to avoid numerical issues. The first fidelity is the quadratic penalty that is more suitable for Gaussian noise, while the second one is the *Kullback-Leibler* (KL) divergence that is obtained from the bayesian approach when poisson noise is present on the image as shown in example 1.5. In practice, it can happen that poisson noise is not as impacting as it should be, making the quadratic fidelity still a more preferable choice, due to it being more controllable. With a similar motivation, the training of the variational model may also be easier and provide better results. In the second and third experiments we placed part of our focus on checking when the KL divergence does truly show its benefits.

We followed the same choices for $\mathcal{A}$ as done in the previous application (Algorithm 3.3) as well as for its backpropagation (Algorithm 3.4), including the learning of the parameters $\boldsymbol{\alpha}$ for $\mathcal{A}$. The startinf point chosen was either the corrupt data $\mathbf{y}$, or $\mathbf{S}^T\mathbf{y}$ when downsampling is involved. The only difference is pertaining the differentiable projection. Considering that in SMLM we are primarily interested in finding the correct positions of the molecule, mostly disregarding the intensity of the light they omit, we only want to constraint the images into the non-negative orthant. This constraint also serves a secondary purpose: it makes the ℓ_1-norm regularizer differentiable.
We also point out that the $CEL0$ regularizer [87], which serves as a non-convex approximation of the ℓ_0-norm, and that is known to produce better results compared to the ℓ_1-norm, does not satisfy the necessary differentiability assumptions. Indeed, while the constraint allows for $CEL0$ to be differentiable, its gradient in non-differentiable with respect to the regularization parameter that governs it.

We turned to two approaches to penalize errors in the reconstruction: a common ℓ_2-norm and the ℓ_1-norm with a binarization of the image that results from the unrolling.

Given a dataset $\mathcal{D}\{(\mathbf{y}_t, \mathbf{x}_t)\}_{t=1}^{T}$, where $\mathbf{y}_t$ is the spoiled frame observed at time t and $\mathbf{x}_t$ is the underlying ground truth, we can express the two loss functions as:

$$L_2(\bar{\mathbf{u}}(\boldsymbol{\theta}); \mathbf{x}) = \frac{1}{2}\|\bar{\mathbf{u}}(\boldsymbol{\theta}) - \mathbf{x}_t\|_2^2, \tag{3.32}$$

$$L_1(\bar{\mathbf{u}}(\boldsymbol{\theta}); \mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} \psi_\gamma \left((B_{\delta,c,\epsilon}(\bar{\mathbf{u}}(\boldsymbol{\theta})) - \tilde{\mathbf{x}}_t)_i^2 \right). \tag{3.33}$$

Here, ψ_γ is the Huber function placed instead of the absolute value to achieve differentiability:

$$\psi_\gamma(s) := \begin{cases} \frac{s}{2\gamma} & \text{if } s \leq \gamma^2 \\ \frac{2\sqrt{s}-\gamma}{2} & \text{if } s > \gamma^2. \end{cases} \tag{3.34}$$

The motivation that prompted us towards trying the loss L_1 can again be found in the fact that we care more for the correct localization rather than having a good intensity estimation. This inconsistency between what the L_2 loss promotes and what we want is less emphasized in the L_1 loss. Intuitively, when the difference between a pixel in the restoration and in the ground truth is particularly high, then this error weighs way more heavily in L_2 than in L_1 that has means to ignore this potential gap. Considering also that we do not place any upper bound on the pixels value, it becomes clear how L_2 may end up prioritizing less the localization: a false positive caused by a spurious point very close to zero will have a minimal impact on L_2, while a true positive where however the pixel intensity is completely wrong will have clear effects on L_2, and we want the exact contrary to happen.
While the Huber-like function ψ is a step towards the desired direction, it is not enough. For this reason, we also developed a differentiable binarization $B_{\delta,c,\epsilon}$ that acts pixel-wise. The binarization is used as a post processing function, which sets the maximum pixel value at $\bar{p}$, while also trying to have as less pixels as possible in the interior points of $[0, \bar{p}]$. In this light, $\tilde{\mathbf{x}}$ is a binarized version of $\mathbf{x}$, which is however obtained by setting all non-zero entries to $\bar{p}$. The binarizaion is defined as

$$B_{\delta,c,\epsilon}(s) := \begin{cases} 0 & s \leq \delta \\ \left(2 - \frac{s-\delta}{\epsilon}\right) \frac{(s-\delta)^2}{\epsilon} \frac{\bar{p}}{2(c-\delta)} & \delta < s < \delta + \epsilon \\ \frac{\bar{p}}{2(c-\delta)}(s-\delta) & \delta + \epsilon \leq s \leq 2c - \delta - \epsilon \\ \bar{p} - \left(2 - \frac{2(c-\delta)-s+\delta}{\epsilon}\right) \frac{(2(c-\delta)-s+\delta)^2}{\epsilon} \frac{\bar{p}}{2(c-\delta)} & 2c - \delta - \epsilon < s < 2c - \delta \\ \bar{p} & s \geq 2c - \delta. \end{cases} \tag{3.35}$$

This binarization is dependent on three (hyper)parameters. The hyperparameter ϵ has the same philosophy and purpose as is then differentiable projection Π, c is the point that is going to be transformed into $\frac{\bar{p}}{2}$, and last, but definitely not least, δ sets the threshold under which everything is set to zero, and, by symmetry, it also controls those that will be set to $\bar{p}$. Figure 3.9 shows various shapes for this binarization for

three values of δ. Ideally, we want to operate in a situation where the slope is as steep as possible, provided we avoid any numerical issue, meaning that we want c to be really close to δ.
Effectively these are technically three more hyperparameters of the model. However, in practice, ϵ does not need any tuning as is should be set to fairly low value. Also, as we as just said, to obtain a steep slope we choose $c = \delta + c_0$, with $c_0 > \epsilon$, but still low, otherwise the definition of the binarization has no sense. With this choice, the binarization actually becomes differentiable with respect to δ, which means that we can (and we will) learn the parameter δ as well. For this specific parameter we have:

$$\frac{\partial L_1}{\partial \delta} = \sum_{i=1}^{n} \psi'_\gamma \left(((B_{\delta,c,\epsilon}(\bar{\mathbf{u}}(\boldsymbol{\theta})) - \tilde{\mathbf{x}}_t)_i^2) \right) \frac{\partial B_{\delta,c,\epsilon}(\bar{\mathbf{u}}(\boldsymbol{\theta}))_i}{\partial \delta} \tag{3.36}$$

where

$$\frac{\partial B_{\delta,c,\epsilon}(s)}{\partial \delta} = \begin{cases} 0 & \delta \geq s \\ \frac{\bar{p}}{2c_0} \left(3\frac{(s-\delta)^2}{\epsilon^2} - 4\frac{(s-\delta)}{\epsilon} \right) & s > \delta > s - \epsilon \\ -\frac{\bar{p}}{2c_0} & s - \epsilon \geq \delta \geq s + \epsilon - 2c_0 \\ -\frac{\bar{p}}{2c_0} \left(4\frac{2*c_0 - s + \delta}{\epsilon} - 3\frac{(2*c_0 - s + \delta)^2}{\epsilon^2} \right) & s + \epsilon - 2c_0 > \delta > s - c_0 \\ 0 & \delta \leq s - 2c_0 \end{cases} \tag{3.37}$$

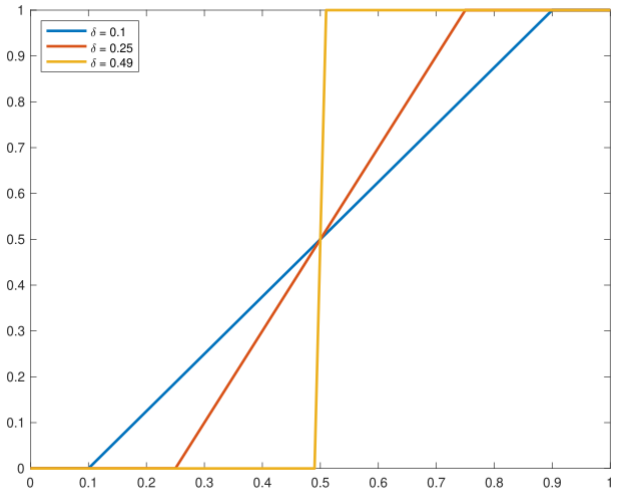


Figure 3.9: Binarization in [0,1] with $c = 0.5$, $\epsilon = 10^{-4}$ for three different values of δ: 0.1 (blue), 0.25 (red), 0.49 (yellow).

3.3.4 Numerical Experience

This section is dedicated to illustrate the results that we have obtained testing the tools that we have presented in the previous section. Since we are working in a

SMLM context, we are mostly interested in obtaining reconstructions that better localize the molecules, disregarding whether the intensity of the pixel matched that of the ground truth. For this reason, we evaluated the quality of the reconstructions with the Jaccard index, which is expressed as:

$$J_{\tilde{p}} = \frac{\#\text{ of correct detections}}{(\#\text{ of correct detections}) + (\#\text{ of false negatives}) + (\#\text{ of false positives})} \tag{3.38}$$

where $\tilde{p} \in \{0, 2, 4\}$ is the number of pixels of tolerance. A zero pixels tolerance means that a detection is counted as correct only if it is in the precise pixel, a tolerance of p means that a correct detection is valid even if the molecule is localized in a pixel that is within a radius of p pixels from the position of interest. In other practical terms, the Jaccard index is no other than the Intersection-over-Union measure. As the name suggests, it computes the ratio between the cardinality of the intersection of the set of detections in the reconstruction and the set of detections in the reference over the cardinality of the union of the same two sets.

In all experiments, the optimization of the upper level problem is carried out via the Scaled Gradient Projection method (SGP) [17, 21], so that we can impose reasonable bounds on the parameters. In particular, whenever possible we exploited proposition II.1 from [57] and chose the upper bound on ρ as

$$\rho_{\max} = \min_t \|\mathbf{A}^T \mathbf{y}_t\|_\infty. \tag{3.39}$$

Although it is technically possible to use the test set, as it is based on the noisy images, we computed this upper bound only using the images of the training set. This was implemented to prevent the variational model to admit only the trivial zero solution. At the same time, the upper bound on δ was set at half the maximum pixel value. Again, the goal was to avoid obtaining all null reconstructions. The lower bounds were all set at 10^{-10}. The training was stopped when the norm of the loss gradient was less than 10^{-5} or when the relative difference between two consecutive loss values was less than 10^{-7}. For safety purposes, a maximum number of 1000 iterations was also set.

Experiment 1: super-resolution and deconvolution of sparse images. The objective of this first toy experiment was twofold. On one hand we wanted to see that there is indeed hope for the ℓ_1-norm as the regularizer. On the other, we needed to see if the combination of the binarization with the smoothed ℓ_1-norm in the loss function does indeed a better job at promoting localization over intensity estimation. Each ground truth $\mathbf{x}_t$ contains between 75 and 150 activated pixels with integer values from 100 to 255. The downsampled images have size 64×64 and the super-resolution factor is $\lambda = 4$. We deal with the imaged in vector form. Let $\mathbf{A} = \mathbf{SH}$ be the acquisition operator. In particular, $\mathbf{H} \in \mathbb{R}^{n^2 \times n^2}$ is the convolution operator

with a Gaussian kernel of standard deviation 2.5, while $\mathbf{S}$ is the operator described in (3.29). The images were also afflicted by additive white Gaussian noise of standard deviation 0.15. The algorithm $\mathcal{A}$ was applied for a total of $k = 190$ iterations, while the parameter c was set as $\delta + 0.01$.

Although rather simple, this experiment shed some light onto our model. As we expected, the L_1 loss achieves significantly better results. Table 3.4 shows the results obtained on the 25 images of test set. What was not expected, was that in order to obtain meaningful results with the quadratic loss, we needed to train on a tailored dataset. Indeed, when using in training images with the same molecules density as those in the test set, the quadratic loss would struggle to find a big enough value for ρ, so that in the end the images would not be sufficiently sparse. However, it still deemed a better choice to keep ρ in a low range as the intensities estimation was good enough to have a small error when comparing the reconstruction to its ground truth. So we created a special dataset of very sparse images that contained exactly 4 activated pixels and used that to train the model with the L_2 loss. This had the wanted effect of improving its performance in terms of the Jaccard index.
Figure 3.10 contains a sample image, in all of its forms, from the test set.

Table 3.4: Results. R = Number of examples used in the training; RS = Number of special examples used in the training. The test set is made by 25 images.

Loss function	R / RS	J0	J2	J4	Avg. J	PSNR
L_2	0 / 8	0.3878	0.4179	0.4179	0.4079	35.76
L_1 (δ learned)	10 / 0	0.7452	0.8480	0.8583	0.8172	33.09

Experiment 2: super-resolution and deconvolution of images form the ISBI 2013 miscroscopy challenge. A more interesting problem for our model was the simulated dataset provided by the ISBI 2013 SMLM challenge. Unfortunately this dataset is not available anymore, but other similar datasets can be found on the challenge website[2]. The dataset featured 361 ground truth images. Each one had on average about 200 pixels activated, with values in $\{1, 2, 3\}$, which is the number of molecules activated in said pixel. For our experiments we rescaled them so that the maximum value was 255. The PSF was obtained using information provided by the challenge organizers. Specifically, in the simulation the camera had a resolution of $m \times m = 64 \times 64$ pixels of $100nm$ each. The PSF was a Gaussian convolution kernel with Full Width at Half Maximum (FWHM) of $258.21nm$. The resolution

[2] https://srm.epfl.ch/srm/index.php

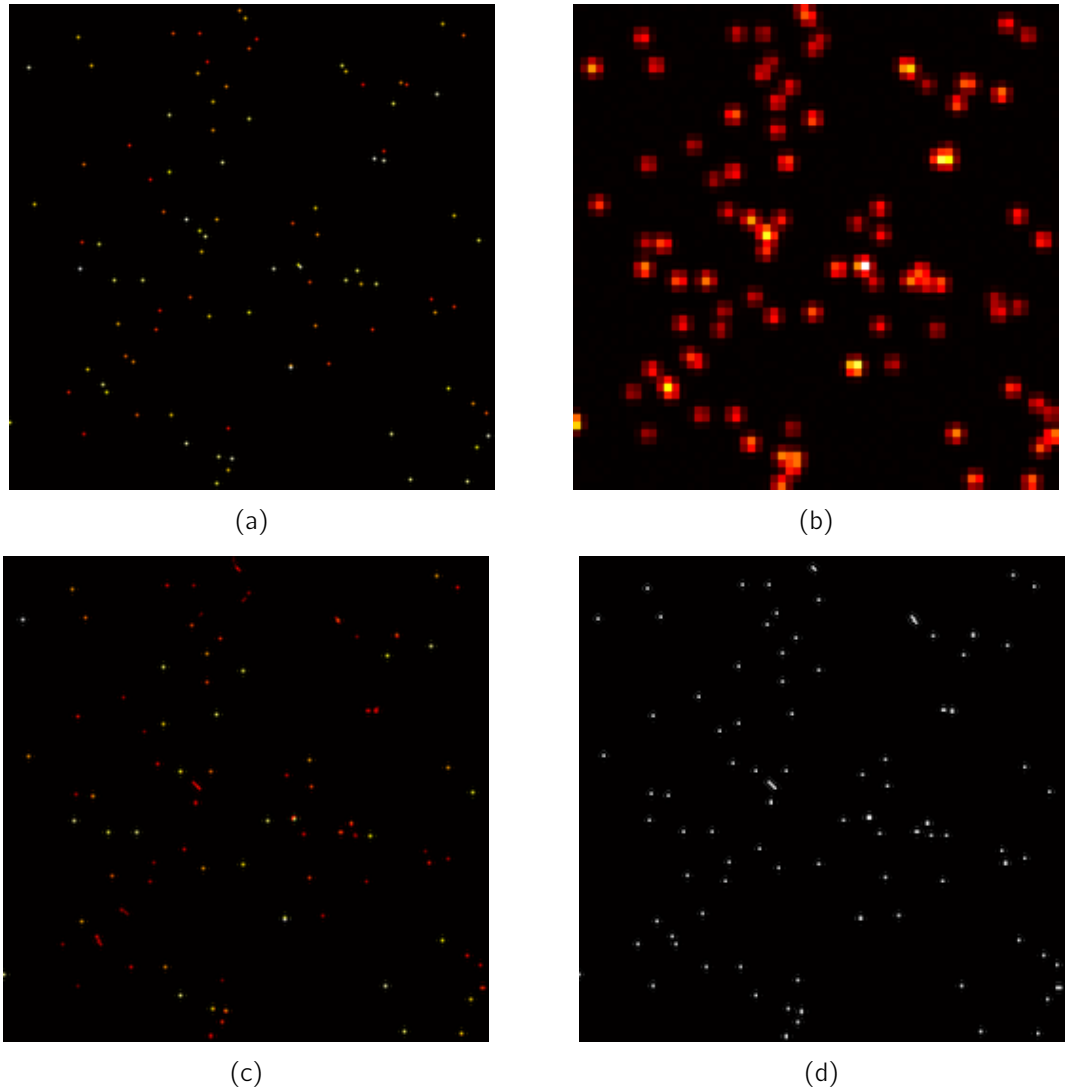


Figure 3.10: (a) Ground Truth; (b) Noisy; (c) Reconstruction from the L_2 loss training; (d) Reconstruction from the L_1 loss training.

factor considered was $\lambda = 4$.
This time, we also added different levels and types of noise:

- Gaussian noise with standard deviation 0.15.

- Poisson noise with background emission of intensity 0.1.

- Poisson noise with background emission of intensity 12.75.

By definition, Poisson noise does not truly allows the concept of "weak" and "strong" intensity, unlike Gaussian noise. However, the intensity of the background emission, when combined with this type of noise, can significantly impact the quality of the observed image. In this sense, when the background is higher, we talk about there being more or stronger Poisson noise. In this scenario, we wanted to see if the KL

divergence does provide the supposed benefits compared to a quadratic data fidelity. In this experiment we also trialled two extra empirical strategies to set the binarization threshold δ, besides the training approach. Both took advantage of the fact that, on average, there were 217 pixels per ground truth image, less than 0.05% of the total number of pixels in each image. One possibility is take the 99.5-th percentile of the pixels values of the observed data, possibly rounding the by defect to be more safe. The other is to take the same value, but from the reconstruction obtained from the starting point of the training when the L_2 loss is used. This makes sense since we are in a non-convex context, and we have to select a decent starting point as to not get stuck in a bad local minimum. For the parameter c, again we simply took $c = \delta + 0.01$ in all cases.
For the training dataset we used only 20 images of the stack. The FISTA-like algorithm was ran for 300 iterations, as this time the PSF was more aggressive.

All the results are reported in Tables 3.5, 3.6 and 3.7, while the images can be seen in figure 3.11 and 3.12. Regarding the choice whether to learn or not the value δ, we observe that, out of the two empirical rules, the one that selects δ from the reconstructions shows better results, often with a significant margin. However, including δ in the training procedure always nets the best results both in terms of Jaccard index and of PSNR. Although this is effectively another parameter to learn, the cost in the backpropagation is still limited as it only requires to compute one time the simple derivative of the binarization. In contrast, the empirical rule based on the reconstruction requires first a whole training phase and a validation phase yield the value of δ. In the tables the symbol † denotes a value of δ set according to the empirical strategy applied to the data, while $\star$ is for when it is obtained from the reconstructions.
In this experiment the KL divergence did not show any sign meaningful of improvement over a quadratic fidelity term, despite the theoretical background says otherwise. The worse performance can be attributed to the fact that we are in a non-convex optimization problem, and the ℓ_2-norm being easier to handle, may lead the training to a better local minimum. The only case where the KL divergence does better, is when it is paired with the L_2 loss, that however achieves only mediocre results, at best.

Experiment 3: super-resolution and deconvolution of images with random particle activation length. For this experiment we used the ground truth positions for the ISBI dataset `https://srm.epfl.ch/DatasetPage?name=MT1.N1.LD`. Specifically, the first operation involved discretizing the positions of the 8731 molecules acquired to obtain the full ground truth stack. The second focused on creating the singular ground truths images, where only a handful of molecules is activated. For this scope we selected, at each new synthetic frame, about 20 new random molecules

Table 3.5: Results in the case of additive white Gaussian noise. † and $\star$ mean that δ was read, respectively, from the noisy data and fro the L_2 reconstructions, according to the empirical rules.

Loss Fun.	δ	J0	J2	J4	Avg. J	PSNR
L_2	-	0.1147	0.2175	0.2194	0.1839	34.27
L_1	$9^\dagger$	0.1283	0.3764	0.4075	0.3041	20.57
L_1	$16.4796^\star$	0.1323	0.4743	0.5622	0.3896	21.96
L_1	learned	0.1309	0.4889	0.5921	0.4040	22.45

Table 3.6: Results in the case of Poisson noise with background 0.1. † and $\star$ mean that δ was read, respectively, from the noisy data and fro the L_2 reconstructions, according to the empirical rules.

Loss Fun.	Fidelity	δ	J0	J2	J4	Avg. J	PSNR
L_2	KL	-	0.0921	0.2633	0.3159	0.2238	33.94
L_2	$\|\cdot\|_2$	-	0.1057	0.1205	0.1211	0.1158	34.74
L_1	KL	$9^\dagger$	0.0895	0.1723	0.1882	0.1500	24.39
L_1	KL	$0.2592^\star$	0.0959	0.4000	0.5424	0.3460	21.89
L_1	KL	learned	0.0907	0.4250	0.5924	0.3694	22.66
L_1	$\|\cdot\|_2$	$9^\dagger$	0.1244	0.3688	0.4220	0.3051	20.81
L_1	$\|\cdot\|_2$	$15.970^\star$	0.1293	0.4407	0.5229	0.3643	21.93
L_1	$\|\cdot\|_2$	learned	0.1287	0.4494	0.5358	0.3713	22.52

Table 3.7: Results in the case of Poisson noise with background 12.75. † and $\star$ mean that δ was read, respectively, from the noisy data and fro the L_2 reconstructions, according to the empirical rules.

Loss Fun.	Fidelity	δ	J0	J2	J4	Avg. J	PSNR
L_2	KL	-	0.0589	0.0909	0.0987	0.0828	34.35
L_2	$\|\cdot\|_2$	-	0.0544	0.0778	0.0829	0.0717	34.36
L_1	KL	$3^\dagger$	0.0933	0.2896	0.3695	0.2508	20.84
L_1	KL	$3.8133^\star$	0.0940	0.3051	0.3905	0.2632	21.14
L_1	KL	learned	0.0982	0.3684	0.4771	0.3146	25.31
L_1	$\|\cdot\|_2$	$3^\dagger$	0.0620	0.2534	0.3463	0.2205	22.86
L_1	$\|\cdot\|_2$	$11.572^\star$	0.0557	0.2479	0.3397	0.2145	23.41
L_1	$\|\cdot\|_2$	learned	0.0950	0.3902	0.5012	0.3288	25.74

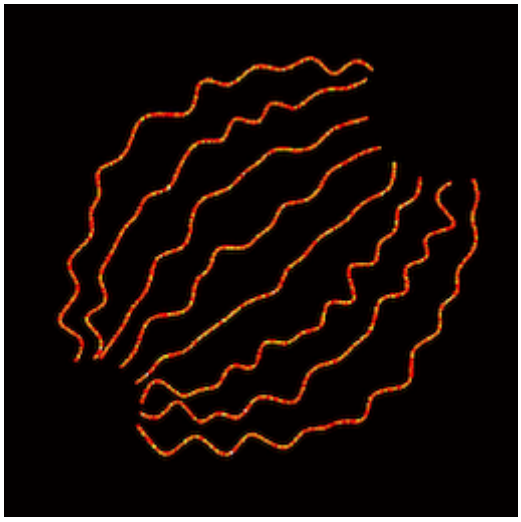
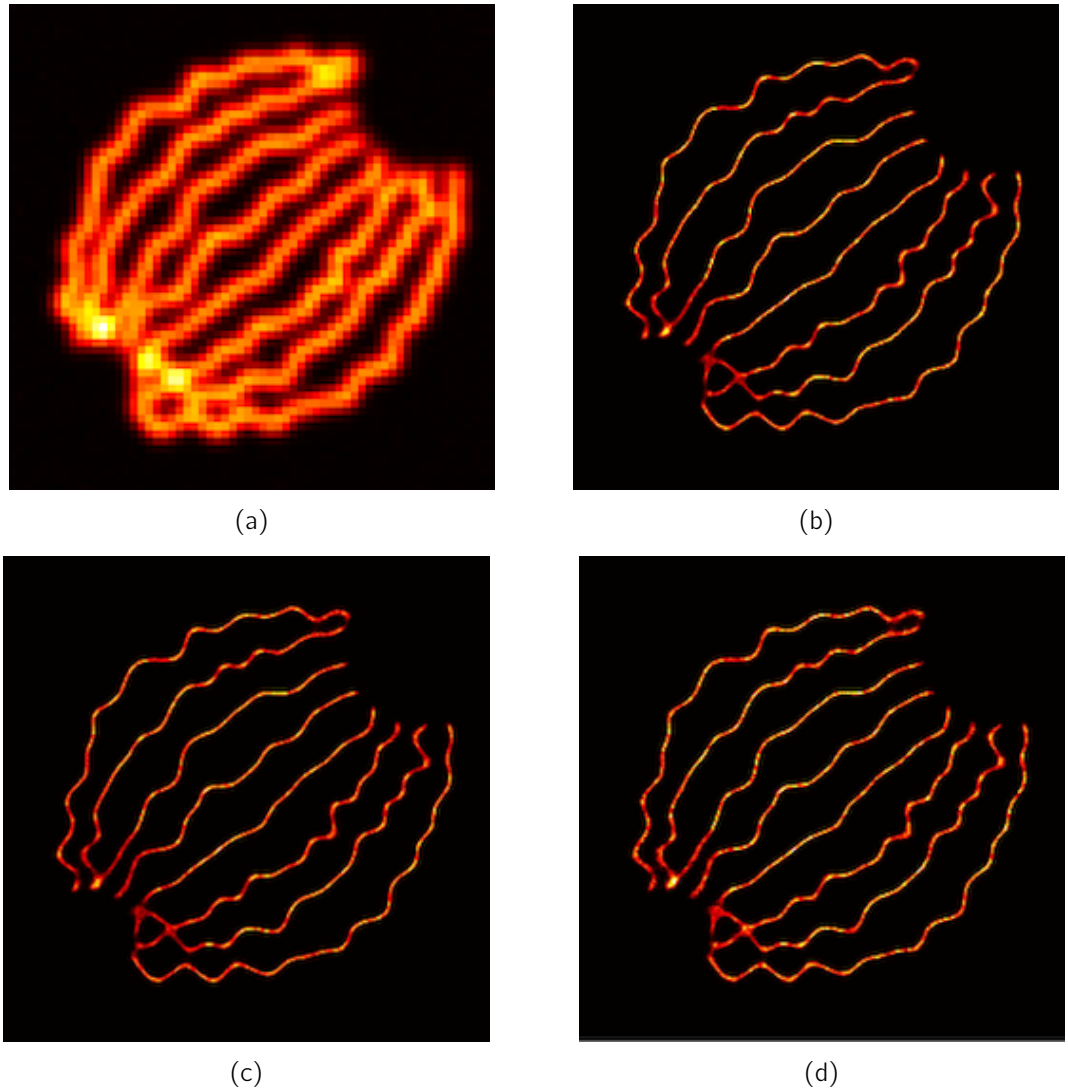


Figure 3.11: Stack of the ground truths images.

from the collection and simulated their activations. The particularity of this experiment is that we simulated a real phenomenon where the very same molecule, once activated, actually emits light for a few successive frames. This behaviour is rarely modelled when treating SMLM problem with variational models, thus we believe this experiment to be a meaningful step as this data is more realistic. In this optic, every time a molecule was activated for the first time it was assigned a counter in $\{0, 1, 2, 3, 4\}$ that signified how many more frames the molecule was going to be activated. The counter was drawn according to the probability density $[0.1, 0.25, 0.3, 0.25, 0.1]$ on said set of values. Regarding the forward operator, we used the same as in the previous experiment, while we only placed Poisson noise with background 10 on the images. For reference, the pixel values still lied in the interval $[0, 255]$. The binarization was fully learned again with $c = \delta + 0.01$, for the reasons explained in the previous paragraph. We tested the $\mathcal{L}_2$ loss function one more time as well as the two different fidelity terms. The unrolling of $\mathcal{A}$ still involved $K = 300$ iteration. Finally, the number of training samples used was, again, 20.

Table 3.8 contains the results, in terms of Jaccard index (3.38) and PSNR, while the visuals are contained in figures 3.13 and 3.14. Two things are worth noticing. The first is that the KL divergence outperforms the quadratic fidelity, by a margin. This may be attributed to the lower overall density of the molecules. Indeed the ISBI tubular structures contained almost ten times the number of molecules of this dataset. The other aspect that the results reveal, is the fact that almost, if not all, the errors of the L_1-trained model are false negatives, as can be deduced from Figure 3.14. This is not surprising as its the already discussed flaw of the ℓ_1-norm as the regularized for the energy.



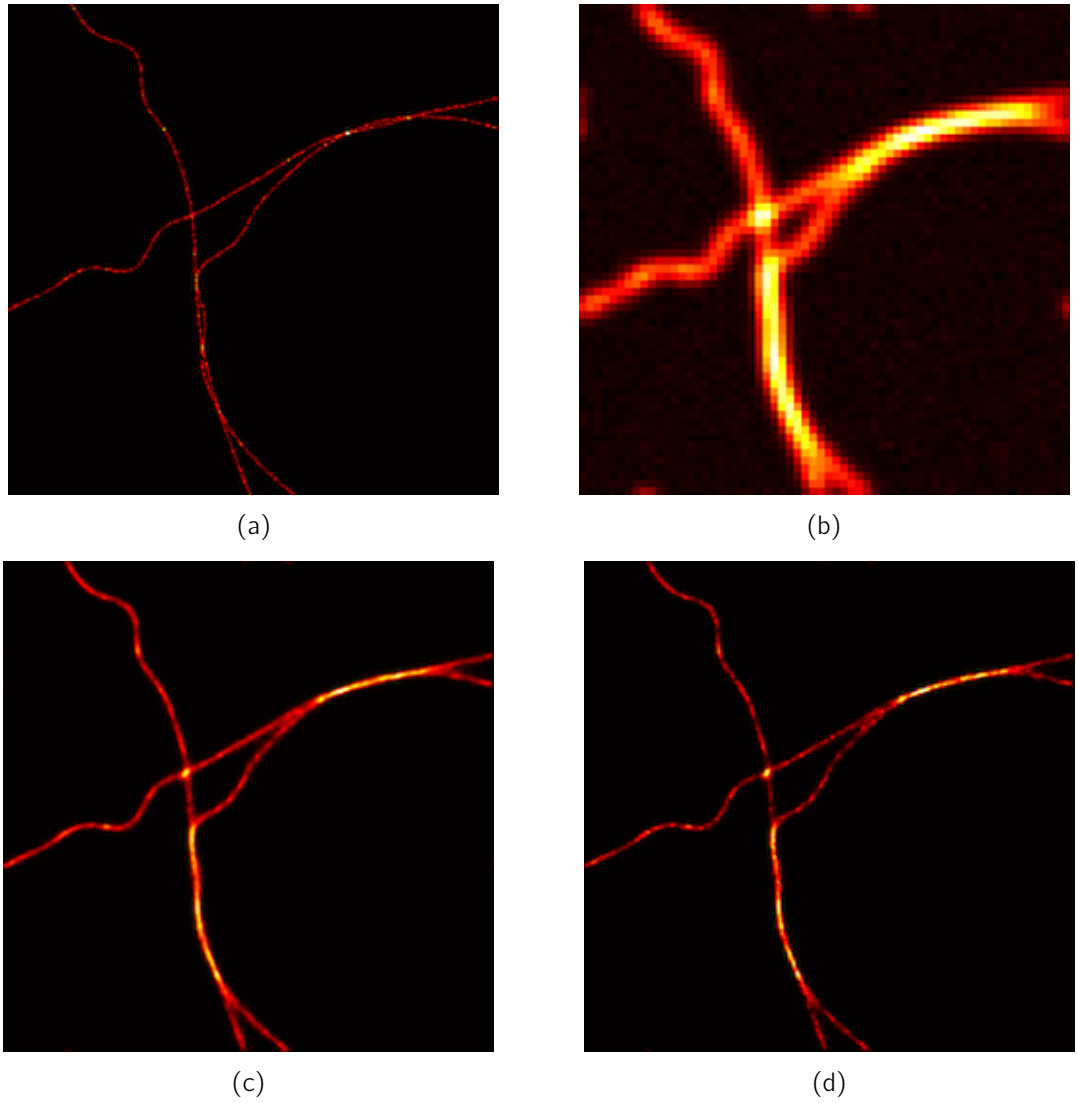
(a) (b)

(c) (d)

Figure 3.12: Image (a) contains the stack of the 361 noisy samples of the dataset in the case of Gaussian noise. Images (b), (c) and (d) shows the stacks of the reconstructions obtained (in order, Gaussian noise, Poisson noise with background 0.1 and Poisson noise with background 12.75).

Table 3.8: Results in the case of Poisson noise with background 10.

Loss function	Fidelity	δ	J0	J2	J4	Avg. J	PSNR
L_2	KL	-	0.0619	0.1285	0.1363	0.1089	39.88
L_2	$\|\cdot\|_2$	-	0.0545	0.0812	0.0837	0.0731	39.97
L_1	KL	learned	0.0776	0.3688	0.4966	0.3144	27.39
L_1	$\|\cdot\|_2$	learned	0.0661	0.2291	0.3259	0.2070	28.00



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

Figure 3.13: (a) Ground truth stack; (b) Noisy stack; (c) Reconstruction stack from the model trained with the $\mathcal{L}_2$ loss; (d) Reconstruction stack from the model trained with the L_1 loss

Experiment 4: Blind-deconvolution of images corrupted with a Gaussian PSF with unknown variance. In this application, we assume a slightly different image acquisition model that we now describe that is analogous to that in [90, 91]. In this case, the dataset $\mathcal{D} = \{(\mathbf{Y}_s; \mathbf{X}_s)\}_{s=1}^{S}$ is not made by single frames but by the variance of the stack. More in detail, each sample $(\mathbf{V}_{\mathbf{X}}^s, \mathbf{V}_{\mathbf{Y}}^s)$ is obtained with the following procedure:

- Step 1: a collection of $T = 1000$ true images is obtained for the molecular structure of interest, $\{\mathbf{x}_s^{(t)}\}_{t=1}^{T}$.
- Step 2: each frame is corrupted with a Gaussian PSF of standard deviation $\sigma = 3$ and additive white Gaussian noise with standard deviation $\eta = 3$. Effectively, this

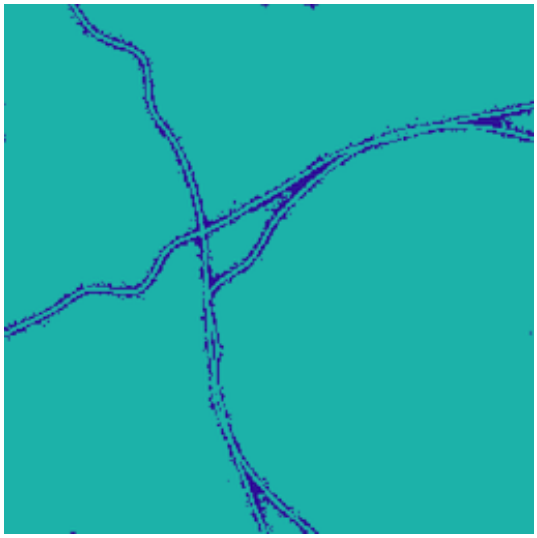


Figure 3.14: False negatives and false positives for the reconstruction obtained with the $\mathcal{L}_1$ loss. Yellow points denotes false negatives, while blue points are false positives.

lead to a second stack of the corrupted images:

$$\mathbf{y}_s^{(t)} = \mathbf{H}(\sigma)\mathbf{x}_s^{(t)} + \mathcal{N}(0, \eta^2).$$

- Step 3: the variance through the index t of the stack is computed for both the clean and noisy collections, giving us $\mathbf{V}_\mathbf{X}^s$ and $\mathbf{V}_\mathbf{Y}^s$ respectively:

$$\mathbf{M}_\mathbf{X}^s = \frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_s^{(t)} \qquad \mathbf{M}_\mathbf{Y}^s = \frac{1}{T}\sum_{t=1}^{T}\mathbf{y}_s^{(t)}$$

$$\mathbf{V}_\mathbf{X}^s = \frac{1}{T-1}\sum_{t=1}^{T}\left(\mathbf{x}_s^{(t)} - \mathbf{M}_\mathbf{X}^s\right)^2 \qquad \mathbf{V}_\mathbf{Y}^s = \frac{1}{T-1}\sum_{t=1}^{T}\left(\mathbf{y}_s^{(t)} - \mathbf{M}_\mathbf{Y}^s\right)^2$$

The variance image that is obtained is, in its appearance, not much different from what we would obtain by stacking directly the images instead of computing their variance through time. However, by working with the second order statistics of the image we are reducing the effects of the blur. In practice, if $\mathbf{h}(\sigma)$ is the kernel of the PSF, by turning to the variance domain the convolution kernel simply becomes $\mathbf{h}^2(\sigma)$, where the square operator acts pixel-wise. Let (i, j) denote the discrete coordinates of a pixel in the kernel matrix $\mathbf{h}$, then we have that

$$h(i,j) = \frac{1}{2\pi\sigma^2}e^{-\frac{\left(i-\frac{n+1}{2}\right)^2+\left(j-\frac{n+1}{2}\right)^2}{2\sigma^2}}$$

$$h(i,j)^2 = \frac{1}{4\pi^2\sigma^4}e^{-\frac{\left(i-\frac{n+1}{2}\right)^2+\left(j-\frac{n+1}{2}\right)^2}{\sigma^2}} \qquad i,j = 1,\ldots,n,$$

where we recall that a single frame has dimension $n \times n$. Unlike in the previous experiments, this time we are going to assume that the kernel is partially unknown, i.e., we take for granted a Gaussian shape, but we leave the parameter σ for the learning procedure.

This different acquisition process is also reflected in the energy functional. For a single stack, as in [90, 91], we adopt the following functional:

$$\underset{\mathbf{V_u}\in\mathbb{R}^{n^2}}{\operatorname{argmin}} \|\mathbf{H}^{(2)}(\sigma)\mathbf{V_u} + \eta^2\mathbf{1}_{n^2} - \mathbf{V_Y}\|_2^2 + \rho\|\mathbf{V_u}\|_1 + \iota_{\{\mathbf{V_u}\geq 0\}}, \tag{3.40}$$

where $\mathbf{H}^{(2)}$ is the convolution matrix corresponding to the kernel $\mathbf{h}^2$.

As we are going to consider σ a parameter of the model, in the backpropagation procedure we are going to need the derivative of the gradient of E w.r.t to σ:

$$\begin{aligned}\frac{\partial}{\partial\sigma}\nabla_{\mathbf{V_u}}E(\mathbf{V_u}) &= \frac{\partial}{\partial\sigma}\left(\left(\mathbf{H}^{(2)}\right)^T\left(\mathbf{H}^{(2)}\mathbf{V_u} + \eta^2\mathbf{1}_{n^2} - \mathbf{V_Y}\right)\right)\\ &= \frac{\partial}{\partial\sigma}\left(\mathbf{H}^{(2)}\right)^T\left(\mathbf{H}^{(2)}\mathbf{V_u} + \eta^2\mathbf{1}_{n^2} - \mathbf{V_Y}\right) + \left(\mathbf{H}^{(2)}\right)^T\left(\frac{\partial}{\partial\sigma}\mathbf{H}^{(2)}\mathbf{V_u}\right),\end{aligned} \tag{3.41}$$

where we have omitted the dependence on σ. Further dissecting the previous formula, we have that the derivative of the convolution w.r.t. σ is the convolution with the derivative of the kernel, i.e.,

$$\frac{\partial}{\partial\sigma}\mathbf{H}^{(2)}\mathbf{V_u} = \left(\frac{\partial}{\partial\sigma}\mathbf{H}^{(2)}\right)\mathbf{V_u} = \left(\frac{\partial}{\partial\sigma}\mathbf{h}^2\right), *\mathbf{V_u},$$

where $*$ denotes the convolution operation and

$$\frac{\partial}{\partial\sigma}h^2(i,j) = \frac{e^{-\frac{\left(i-\frac{n+1}{2}\right)^2+\left(j-\frac{n+1}{2}\right)^2}{\sigma^2}}}{2\pi^2\sigma^5}\left(\frac{1}{\sigma^2}\left(\left(i-\frac{n+1}{2}\right)^2+\left(j-\frac{n+1}{2}\right)^2\right)-2\right) \tag{3.42}$$

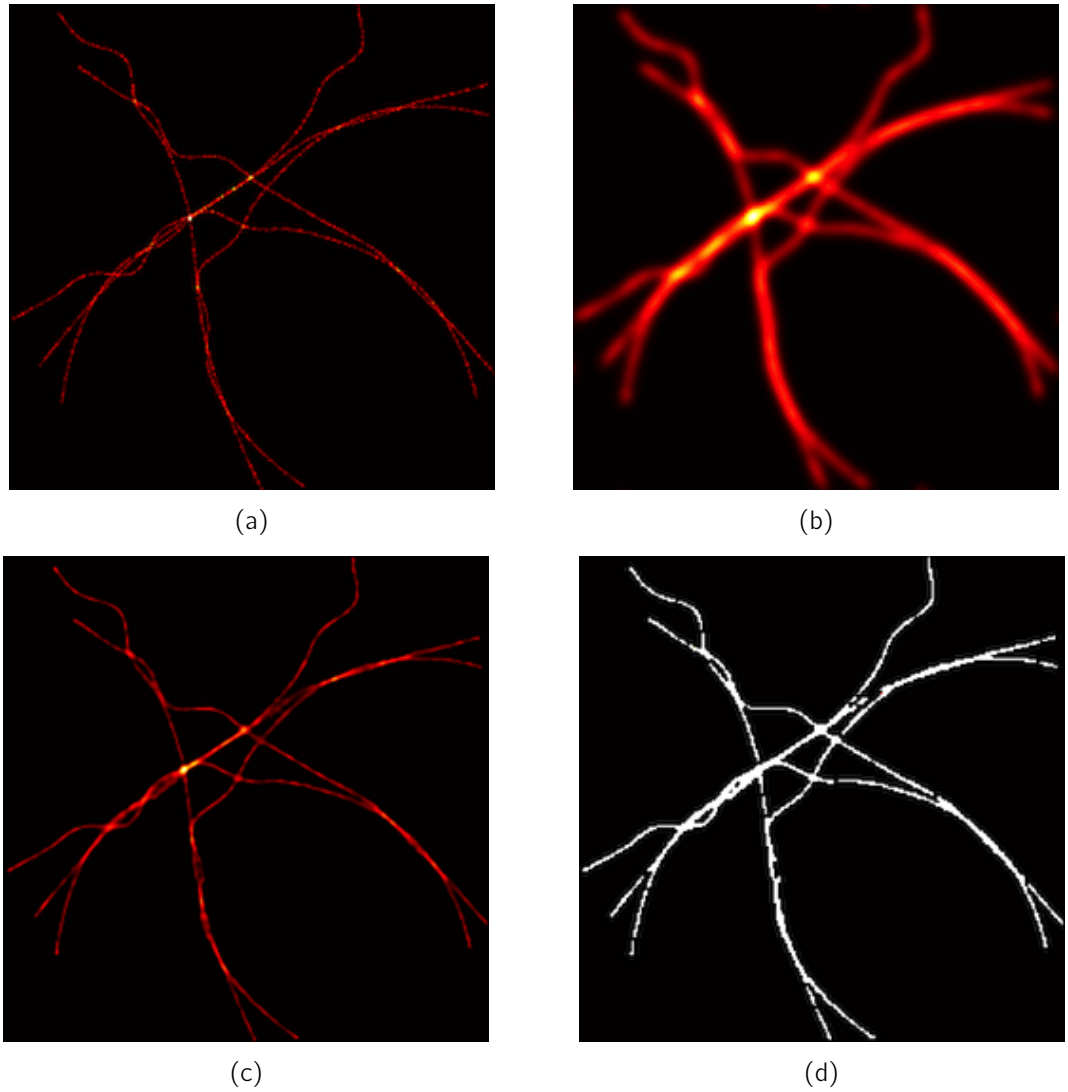
The dataset $\mathcal{D}$ is made of 30 different samples, and was split into a training set with 20 elements and a test set with the remaining 10 images. The specifics for the lower level model were kept as in the last two experiments, including the number of iterates for the unrolling. For the loss function we only used the L_1 penalty given, at this point, that it is clearly the better one between the two. The binarization was fully learned with $c = \delta + 0.01$.

Over the whole test set the average Jaccard index is reported in Table 3.9. It also contains the Jaccard index computed on the images before the binarization is applied,

so that its beneficial effect can be better appreciated.
The value learned for the parameter of the convolution kernel was $\sigma^* = 2.87$. Despite the fact that the parameter in slightly underestimated, the images do no show the typical artifacts of deconvolution with a wrong PSF, ringiing in particular. This is a positive effect of the regularization via ℓ_1-norm which helps in sharpening the edges and clear out these flaws. In Figure 3.15 we illustrate one reconstruction from the test dataset. As it can be seen again from the pictures, the areas where there is a higher concentration of molecules remain a little bit too much thick. This is probably caused by a mix of factors. The main contributor are the limits of the ℓ_1-norm as the regularizer. As previously discussed, it tends to not achieve the desired sparsity in areas where molecules are denser. This causes the filaments to appear as one when they are close, with the gaps between them not being recovered. However, it can also be a limitation of the whole model. In fact, this is still a non-convex optimization model, whose performance can vary as the local minimum that is found changes. Figure 3.16 specifically shows how the binarization is actually a bit too aggressive as it ends up deleting molecules that were not meant to be cancelled. It still serves its purpose of clearing all the spurious points. These pixels have values so low that are not actually visible in the pictures, but the count as any other false negative for the Jaccard index.

Table 3.9: Results over the whole dataset, including the numbers obtained from the reconstruction before the binarization is applied. PSNR is not reported as the model in (3.40) in only meant to find the localization of molecules completely disregarding the intensity.

Binarization	J0	J2	J4	Avg. J
No	0.7096	0.7164	0.7208	0.7156
Yes	0.7922	0.8362	0.8582	0.8289



(a) (b)

(c) (d)

Figure 3.15: (a) Ground truth variance; (b) Noisy variance; (c) Reconstruction pre-binarization; (d) Reconstruction post-binarization.

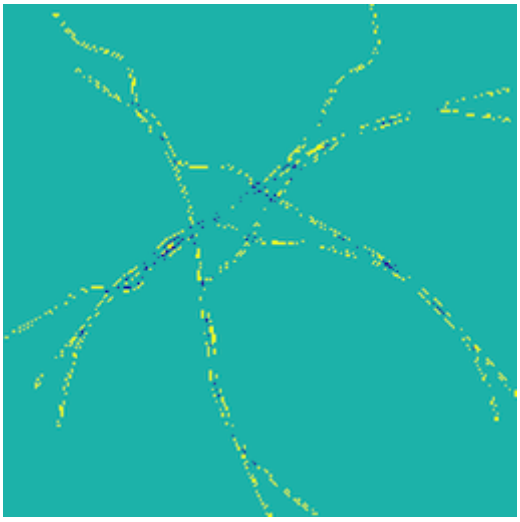


Figure 3.16: False negatives and false positives for the reconstruction obtained post-binarization. Yellow points denotes false negatives, while blue points are false positives.

Chapter 4

Stochastic Algorithms in Bilevel Optimization

In this chapter we are going to experiment on whether stochastic algorithms are a viable tool to optimize the upper level loss function in bilevel optimization 2.8 or for the learning of an unfolded algorithm 3.1. In the applications shown up until this point, the variational models depended only on a handful of parameters. Thus, a limited number of samples would suffice to train the model. If we are interested in using more sophisticated energies, with more parameters to be set, then more data is required. Even computing 200 approximate minimums and then solving the same amount of linear systems, in the spirit of the algorithm proposed in section 2.3, may prove to be computationally challenging. Similar arguments apply in the algorithm unrolling context. However, just because the problem presents itself in a way that is suitable to be addressed by stochastic optimization schemes, this does not automatically mean that they are going to provide meaningful results. The loss gradient obtained with the ideas presented in this thesis is noisy. Combining this flaw with the randomness of the stochastic gradient may prove to be destructive for the whole process. However, since we are going to work with a non-convex problem, the randomness introduced may also prove useful in escaping unwanted local minima.

4.1 Problem and Model Definition

Image denoising is the most basic type of linear inverse problem in imaging. This translates to having $\mathbf{A}$ as the identity operator in the formulation (1.1). Due to the simplicity of the problem, variational approaches are perhaps one of the best strategies to follow, as in this scenario they are able to produce great results with relatively low computational cost.

In this application we work with the MRF energy functional as proposed in [27], which is formulated as

$$E_{\mathbf{y}}(\mathbf{u};\boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{u}-\mathbf{y}\|_2^2 + \sum_{i=1}^{N_f} \alpha_i \phi(\mathbf{a}^i * \mathbf{u}). \tag{4.1}$$

The central piece of the functional is the regularizer. Essentially, it can be seen as a generalization of the TV functional. Instead of the finite differences operator, it generalizes it by featuring a weighted sum of N_f filters each encoded in the convolutional kernel $\mathbf{a}^i$. The convolution $\mathbf{a}^i * \mathbf{u}$ will also be written as the matrix-vector product $\mathbf{A}^i\mathbf{u}$. More specifically, each filter/matrix is not let completely free but is taken as the linear combination of the DCT basis. We work with filters of size 7×7, hence

$$\mathbf{A}^i = \beta_{ij}\mathbf{B}_j, \tag{4.2}$$

for $i = 1,\dots,N_f = 48$ and $j = 1,\dots,48$. Normally, a basis for 7×7 filters would include $7^2 = 49$ elements. However, for this application the first filter of the DCT basis is discarded as it is constant, and zero mean filters are shown to be more effective for natural images [54]. The representation of the image obtained through each filter is supposed to be sparse. Much like in the TV functional, after the filtering is done, sparsity is induce throug the function ϕ. While the ℓ_1-norm is still pretty good at inducing sparsity, its convexity is constantly hindering it. As shown in Figure 4.1 a non-convex penalty offers a far better approximation of the pixels distribution in the transformed domain. In particular, we worked with $\phi(t) = \log(1+t^2)$. This is a clear example of how, in general, convex regularizer are limited in their qualitative performance, as many distribution are, in fact, not convex at all.

Finally, given the dataset $\mathcal{D} = \{\mathbf{y}_s, \mathbf{x}_s\}_{s=1}^S$, the loss function used was

$$L(\bar{\mathbf{u}},\boldsymbol{\theta}) = \frac{1}{S}\sum_{s=1}^{S}\frac{1}{2}\|\bar{\mathbf{u}}_s(\boldsymbol{\theta}) - \mathbf{x}_s\|_2^2,$$

with $\bar{\mathbf{u}}_s$ being the solution of the lower level problem corresponding to the noisy sample $\mathbf{y}_s$ and $\mathbf{x}_s$ the relative ground truth.

4.2 Numerical Experience

Implementation Details. From the BSDS500 dataset [62], we took 200 image patches of size 64×64 that constituted the training set $\mathcal{D}$, and 68 different sample of size 481×321 to obtain a test set. The noisy samples were obtained by applying additive white Gaussian noise with a standard deviation that was 10% of the maximum pixel value.

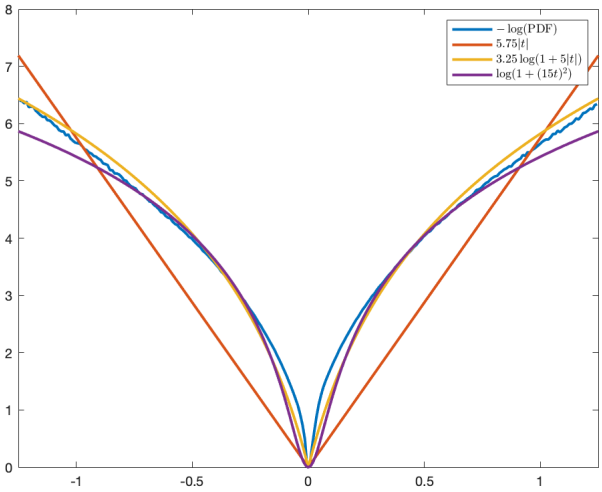


Figure 4.1: The blue line shows the empirically computed negative logarithm of the probability density function (PDF) for the pixels of the convolution between 100 natural images and one of the filters of the basis. The other lines represents the functions $c_1|t|$ (red), $c_2\log(1+c_3|t|)$ (yellow) and $c_4\log(1+(c_5t)^2)$ (purple). As it can be seen, for some fitted regularization constants $c_1,\dots,c_5$ it is possible to obtain a better approximation of the pursued prior distribution with non-convex functions rather than using the ℓ_1-norm.

For the energy presented in (4.1) and (4.2), the parameters that we want to learn are $\boldsymbol{\theta} = (\boldsymbol{\alpha},\boldsymbol{\beta})$, i.e., the weights $\{\alpha_i\}_{i=1}^{N_f}$ and the coefficients $\{\beta_{ij}\}_{i,j=1}^{48}$ of the linear combinations. In this application we followed both the approaches illustrated in Chapters 2 with equation (2.8) and 3 in the problem (3.1).

The energy functional in question is particularly complex, as it depends on $48^2+48 = 2352$ parameters. This mandates the use of a bigger training set. Both in the full bilevel optimization formulation and for algorithm unfolding, the computation of the loss function for hundreds of images becomes basically unfeasible, at least on an academic budget. In an optic of Green AI, we opt to use stochastic algorithms and process only a handful of samples for each outer iteration. We employed the *stochastic gradient* in its standard version (SGD) [77], the *stochastic gradient with momentum* (MOM) and AdaM [56]. It is well known that the performance of stochastic optimizers is stricly connected the their hyperparameters, in particular their *learning rate*. In the literature, various strategies to make this selection automatically have been proposed [42, 43], also including schemes [44] that aim at reducing the variance of the true gradient estimator. All these methods are particularly important in contexts such as deep learning, but in comparison our model in still on the simpler side, thus we deemed reasonable to only opt to process 20 images at a time for a constant minibatch size. Also, the learning rates were manually tuned.
In order to have a reference for our tests, we also used a deterministic optimizer

which processed the whole dataset at each outer iteration. The method used was a variation of the *Scaled Gradient Projection* method (SGP) [16, 17] which featured a BFGS scaling matrix in the spirit of [24].
The training phase was halted when the relative change in the loss function value between two subsequent iterations was less than 10^{-5}. In order to be sure to get close to a stationary point, the SGP method had a maximum number of 500 iterations. On the other hand, the stochastic schemes featured 100 epochs, a value that was determined empirically based on when we observe the typical stalling behaviour of the methods in question.

Regarding the lower level problem, in the bilevel optimization case we used the quasi-Newton *L-BFGS* algorithm [24][1], which was set to run for a maximum number of 500 iterations (rarely reached) or until $\|\nabla_{\mathbf{u}} E(\mathbf{u};\boldsymbol{\theta})\|_2 \leq 10^{-3}$. The gradient of the loss function was then computed following equations (2.15) and (2.16).
In the unrolling case we chose as $\mathcal{A}$ a simple gradient descent step

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} - \lambda_k \nabla_{\mathbf{u}} E_{\mathbf{y}}(\mathbf{u}^{(k)};\boldsymbol{\theta}),$$

where we also had the possibility of learning the steplengths λ_k. The algorithm $\mathcal{A}$ was unfolded for $K = 5, 10, 15, 20$ iterations and its gradient was learned according to the backward mode described in 3.1.

Numerical Results. Figure 4.2 reports the loss decrease for the bilevel optimization problem. The experiments confirmed what is expected of the stochastic methods from the literature: if the learning rate is too high, the loss has an unbearably oscillating behaviour, if it is too low, then the convergence is slow. Also, the inertial scheme show the desired acceleration in the first few epochs, and then reach the stalling point rather quickly. Overall, we found that all three stochastic schemes performed better than the quasi-Newton deterministic algorithm in the beginning, which was able to find lower loss values only after more than 50 epochs, and even then, after 100 epochs the gap was not too wide.
In table 4.1 we have summed up the final values, with the average PSNR computed on the 68 images of the test set for these experiments.

[1] We used the coded that is provided at `https://github.com/stephenbeckr/L-BFGS-B-C`

Table 4.1: Summary of the training results for the bilevel optimization problem. The final loss value and the relative average PSNR are computed after 100 epochs for the stochastic schemes. The values for the L-BFGS algorithm are taken from the original article [27]

Optimizer	**Final Loss Value**	**Average PSNR**
L-BFGS	388053	28.66
SGD 5e−8	392187	28.57
MOM 5e−8	400388	28.51
AdaM 1e−3	402266	28.50

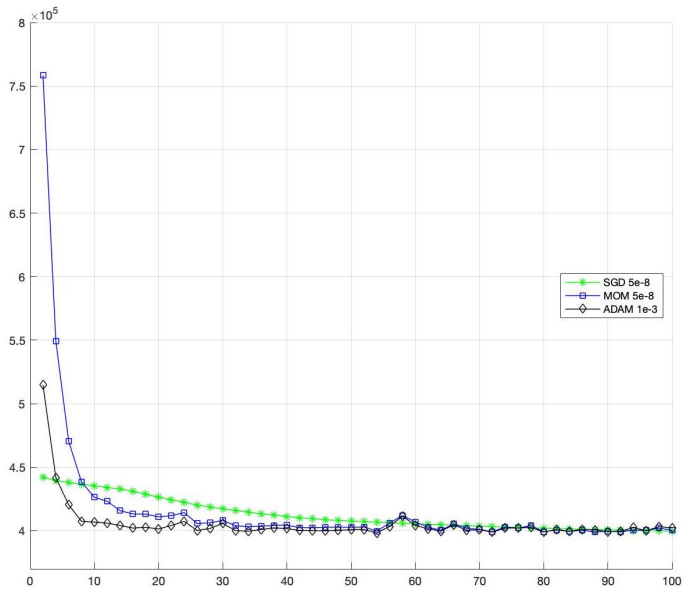


Figure 4.2: Plot of the loss decrease with the best fixed learning rates for each stochastic scheme tested.

Turning to the algorithm unrolling problem we confirmed that, as one would suppose, the more inner iterations the better. There is a significant increase in the performance when using 5 inner iterates or 10, as well as going from 10 to 15 (see figure 4.3). However, the same does not apply with more than 15 iterates, i.e., the increase in the quality of the reconstruction probably does not justify the extra computational effort.

In table 4.2 we have reported the final numbers for the unrolling experiments. The full-gradient loss optimizer is for reference, and again it does just slightly better, but only after quite some time and effort.

Table 4.2: Summary of the training results for the algorithm unfolding model.

Optimizer	**Final Loss Value**	**Average PSNR**
	$K = 10$	
Quasi-Newton SGP	387441	28.57
SGD 1e−7	396262	28.52
MOM 1e−8	400209	28.46
AdaM 1e−4	401220	28.49
	$K = 15$	
Quasi-Newton SGP	386698	28.6
SGD 1e−7	393670	28.55
MOM 1e−8	395823	28.52
AdaM 1e−4	398806	28.51

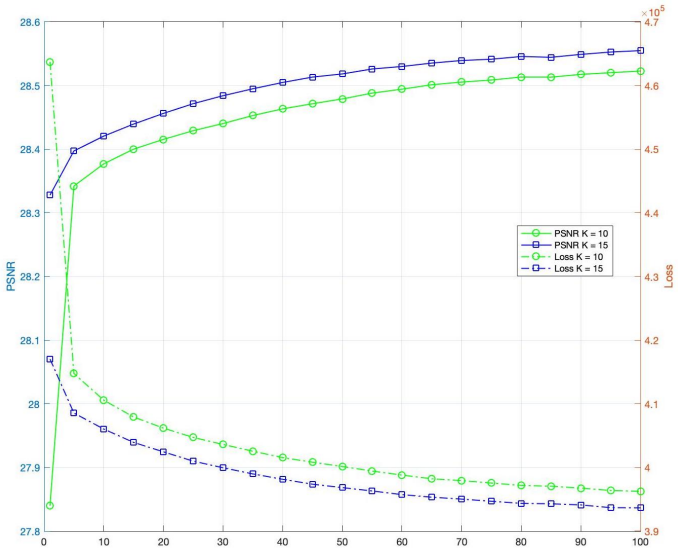


Figure 4.3: Loss and average PSNR behaviour for the cases $K = 10$ and $K = 15$. Loss optimizer: SGD with $\lambda = 1e{-}7$

In conclusion, our experiments showed comparable results to those originally shown in [27], which in tunr means that the method is comparable to notable algorithms such as BM3D [30] and GMMs [102]. In particular, the numerical experiments show that the results of a combination of the unrolling approach with the stochastic optimization for the loss minimization are comparable with those of the original bilevel approach in terms of accuracy. On the other side, the unrolling approach leads to a

computable, explicit form of the gradient of the loss function and also to a reduction of the complexity of the overall learning algorithm, since the lower level only requires a few iterations of $\mathcal{A}$. Technically the same can also be said for the processing of new, unseen images once the model is learned.

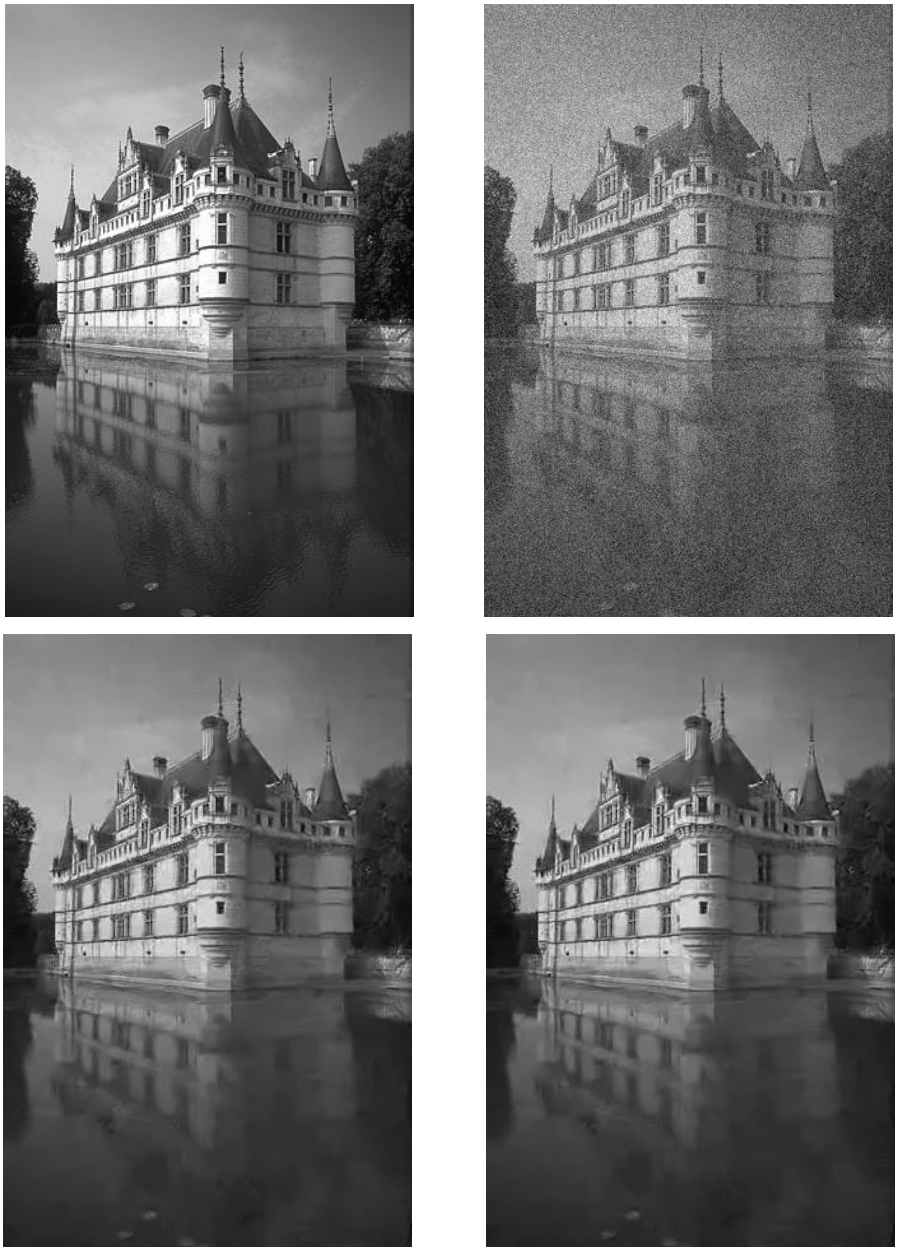


Figure 4.4: Example of a reconstruction: (a) ground truth; (b) noisy image; (c) and (d) reconstructions obtained with the unrolling of respectively 10 and 15 steps of gradient descent, where the parameters of the model were learned with SGD.

Appendix A

Convex Analysis.

In this brief appendix we are going to recall some basic notions that were used in Chapters 2 and 3 regarding convex optimization. For a more detailed illustration, including the proofs, we refer the reader to [78] or [101].
We start by recalling that, given a function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$, then its *effective domain* is the set $\operatorname{dom} f := \{\mathbf{u} \in \mathbb{R}^n : f(\mathbf{u}) < +\infty\}$ and f is called *proper* if $\operatorname{dom} f$ is non-empty.

Fundamentals of Convex Functions

Definition A.1. A set $\Omega \subset \mathbb{R}^n$ is said to be *convex* if

$$\lambda\mathbf{u} + (1-\lambda)\mathbf{v} \in \Omega \qquad \forall \mathbf{u}, \mathbf{v} \in \Omega \; \forall \lambda \in [0,1]$$

Essentially, given any two elements in Ω, then the segment that joins them lies completely in Ω. In this section $\Omega \subset \mathbb{R}^n$ will denote a convex set, as that is the base do define convex functions.

Definition A.2. A function $f : \Omega \subseteq \mathbb{R}^n \to \bar{\mathbb{R}}$ is *convex* if

$$f(\lambda\mathbf{u} + (1-\lambda)\mathbf{v}) \leq \lambda f(\mathbf{u}) + (1-\lambda) f(\mathbf{v}) \qquad \forall \mathbf{u}, \mathbf{v} \in \Omega \; \forall \lambda \in [0,1],$$

and is *stricly convex* if the previous inequality is strict. Furthermore, f is *strongly convex with modulus* $\nu > 0$ if

$$f(\mathbf{u}) - \frac{\nu}{2}\|\mathbf{u}\|_2^2$$

is convex.

As intuitively as it sounds, strong convexity $\Rightarrow$ strict convexity but the reverse is not true in general.
A notable property of convex functions is that every minimum is a global minimum. If the function is strictly convex, then the minimum is unique (provided it exists). Also, the following properties hold whenever the function is also differentiable.

Proposition A.3

If $f : \Omega \subseteq \mathbb{R}^n \to \bar{\mathbb{R}}$ is a differentiable convex function, then for every $\mathbf{u}, \mathbf{v} \in \Omega$

$$f(\mathbf{v}) \geq f(\mathbf{u}) + \boldsymbol{\nabla}_{\boldsymbol{u}} \boldsymbol{f}(\mathbf{u})^T (\mathbf{v} - \mathbf{u})$$

Proposition A.4

If $f : \Omega \subseteq \mathbb{R}^n \to \bar{\mathbb{R}}$ is a differentiable function, then the following conditions are equivalent

- *f is strongly convex with modulus ν*
- *For every $\mathbf{u}, \mathbf{v} \in \Omega$*

$$f(\mathbf{v}) \geq f(\mathbf{u}) + \boldsymbol{\nabla}_{\boldsymbol{u}} \boldsymbol{f}(\mathbf{u})^T (\mathbf{u} - \mathbf{v}) + \frac{\nu}{2} \|\mathbf{v} - \mathbf{u}\|_2^2$$

- *For every $\mathbf{u}, \mathbf{v} \in \Omega$*

$$\left(\boldsymbol{\nabla}_{\boldsymbol{u}} \boldsymbol{f}(\mathbf{v}) - \boldsymbol{\nabla}_{\boldsymbol{u}} \boldsymbol{f}(\mathbf{u})\right)^T (\mathbf{v} - \mathbf{u}) \geq \nu \|\mathbf{v} - \mathbf{u}\|_2^2$$

We also note that, from the definition of strongly convex functions, if f is twice continuously differentiable then $\nu \mathbf{I_n} \preceq \boldsymbol{\nabla}^{\mathbf{2}}_{\boldsymbol{uu}} \boldsymbol{f}$.
Now, these are all useful properties that can actually be generalized and obtained whenever f is convex but does not have a gradient. The key is replacing the gradient with the subgradient. A subgradient is based on extending the property in Proposition A.3 to non-differentiable functions.

Definition A.5. Given a proper function $f : X \to \bar{\mathbb{R}}$, a vector $\mathbf{w} \in \mathbb{R}^n$ is a *subgradient* for f at $\mathbf{u} \in \operatorname{dom} f$ if, for every $\mathbf{v} \in \Omega$, it satisfies the *subgradient inequality*:

$$f(\mathbf{v}) \geq f(\mathbf{u}) + \mathbf{w}^T (\mathbf{v} - \mathbf{u}). \tag{A.1}$$

The set of all subgradients for f at $\mathbf{u}$ is denoted with $\partial f(\mathbf{u})$. The multivalued operator

$$\begin{aligned} \partial f : \Omega &\longrightarrow \mathbb{R}^n \\ \mathbf{u} &\longmapsto \partial f(\mathbf{u}), \end{aligned}$$

is called the *subdifferential* of f.

In practice, any subgradient $\mathbf{w} \in \partial f(\mathbf{u})$ identifies a line tangent to the graph of f in $\mathbf{u}$, that is also never above the graph. Although any function f can have subgradients, convexity allows for an easier and systematic use of the subdifferential thanks to the following properties.

Proposition A.6

Given a proper convex function $f : \Omega \to \bar{\mathbb{R}}$, then the following properties hold
- *If $\mathbf{u}$ is a point in the interior of $\operatorname{dom} f$, then $\partial f(\mathbf{u})$ is non-empty.*
- *If f is continuous at $\mathbf{u}$ then $\partial f(\mathbf{u})$ is non-empty.*
- *If f is continuously differentiable at $\mathbf{u}$, then $\partial f(\mathbf{u}) = \{\nabla \boldsymbol{f}(\mathbf{u})\}$.*

The last property in particular, even though it seems rather natural, does not hold for any differentiable function. In fact, it may happen that $\partial f(\mathbf{u})$ is empty even if f is differentiable at $\mathbf{u}$, as the subgradient inequality for differentiable functions is always true only for convex function (Proposition A.3).
Last, but not least, we have a reformulation of the first order optimality condition in the case of non-differentiable functions. As in the case of convex functions, this is a direct consequence of the subgradient inequality (A.1).

Proposition A.7

Given a proper function $f : X \to \bar{\mathbb{R}}$, then $\mathbf{u} \in \operatorname{argmin} f$ if and only if $\mathbf{0} \in \partial f(\mathbf{u})$.

Proximal Operator

Definition A.8. The *proximal operator* of a function $f : X \to \bar{\mathbb{R}}$ is defined as

$$\operatorname{prox}_f(\mathbf{v}) = \operatorname*{argmin}_{\mathbf{u}} f(\mathbf{u}) + \frac{1}{2}\|\mathbf{u} - \mathbf{v}\|_2^2 \tag{P}$$

It is easy to show that the proximal operator of f is no other than what is called the *resolvent operator* of ∂f, i.e., $(I + \partial f)^{-1}$, where I is the identity operator.
Again, in the case of convex function, the proximal operator gains important properties:

Proposition A.9

If $f : \Omega \to \bar{\mathbb{R}}$ is proper, convex and lower semi-continuous, then the function

$$P_{\mathbf{v}}(\mathbf{u}) := f(\mathbf{u}) + \frac{1}{2}\|\mathbf{u} - \mathbf{v}\|_2^2$$

is strictly convex and admits a unique minimum. In other terms, the proximal operator is well-defined and is a single-valued operator.

The proximal operator as is defined, may not have a closed form solution and needs to be computed inexactly. In order to do so efficiently, we can turn to its *dual problem*.

Definition A.10. Given a function $f : X \to \bar{\mathbb{R}}$, its *convex conjugate* or *Fenchel conjugate* is the function

$$f^*(\mathbf{w}) = \sup_{\mathbf{u} \in X} \mathbf{w}^T\mathbf{u} - f(\mathbf{u}),$$

The convex conjugate is always convex (even if f itself is not) and is proper if f is proper, convex and lower semi-continuous. Also, we have the following notable results:

Theorem A.11 (Biconjugation Theorem)

A proper, convex and lower semi-continuous function $f : \Omega \to \bar{\mathbb{R}}$ coincides with is biconjugate, i.e.

$$f(\mathbf{u}) = f^{**}(\mathbf{u}) = \sup_{\mathbf{w} \in \operatorname{dom} f^*} \mathbf{u}^T\mathbf{w} - f^*(\mathbf{w}).$$

Proposition A.12

Let $f : \Omega \to \bar{\mathbb{R}}$ be a proper, convex and lower semicontinuous function, then the following are equivalent

- $f(\mathbf{u}) + f^*(\mathbf{w}) = \mathbf{w}^T\mathbf{u}$
- $\mathbf{w} \in \partial f(\mathbf{u})$
- $\mathbf{u} \in \partial f^*\mathbf{w}$

Now, the optimization problem in (P) can be referred to as the *primal problem*, whose is counterpart is the *dual problem*

$$\operatorname*{argmax}_{\mathbf{w}} \frac{1}{2}\|\mathbf{v}\|_2^2 - \frac{1}{2}\|\mathbf{v} - \mathbf{w}\|_2^2 - f^*(\mathbf{w}). \tag{D}$$

Let us denote with

$$Q_{\mathbf{v}}(\mathbf{w}) := \frac{1}{2}\|\mathbf{v}\|_2^2 - \frac{1}{2}\|\mathbf{v} - \mathbf{w}\|_2^2 - f^*(\mathbf{w}),$$

the objective function of the dual problem. Then, under reasonable hypothesis, a simple application of the results just recalled shows that $P_{\mathbf{v}}(\mathbf{u}) \geq Q_{\mathbf{v}}(\mathbf{w})$ for all $\mathbf{u}, \mathbf{w}$ and

$$\mathbf{u}^* = \operatorname*{argmin}_{\mathbf{u}} P_{\mathbf{v}}(\mathbf{u}) = \operatorname*{argmin}_{\mathbf{w}} Q_{\mathbf{v}}(\mathbf{w}) = \mathbf{w}^*$$

with $\mathbf{u}^* = \mathbf{v} - \mathbf{w}^*$. Furthermore, in the case where $f(\mathbf{u}) = g(\mathbf{A}\mathbf{u})$, then simple subdifferential and linear algebra calculus give us $\mathbf{u}^* = \mathbf{v} - \mathbf{A}^T\mathbf{w}^*$.
The advantage of turning to dual problems is that, in the majority of cases, they are simpler to solve. If $(\mathbf{w}^{(j)})_{j\in\mathbb{N}}$ is the sequence that converges to the solution of (D), then usually one would stop whenever

$$P_{\mathbf{v}}(\mathbf{v} - \mathbf{w}^{(j)}) - Q_{\mathbf{v}}(\mathbf{w}^{(j)}) \leq \epsilon,$$

for a given tolerance ϵ.

In Chapter 2 the tolerance in the inexact computation of the proximal operator actually depends on $h^{(k)}(\mathbf{z}^{(k)})$, where $\mathbf{z}^{(k)}$ is the exact solution. It is actually possible to circumvent this issue. We recall that

$$h^{(k)}(\mathbf{z}) = \nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta}^{(k)})^T(\mathbf{z} - \boldsymbol{\theta}^{(k)}) + \frac{1}{2\alpha_k}\|\mathbf{z} - \boldsymbol{\theta}^{(k)}\|^2_{\mathbf{M}_k} + R(\mathbf{z}) - R(\boldsymbol{\theta}^{(k)})$$

so that the dual function associated with the primal problem $\operatorname{argmin}_{\mathbf{z}} h^{(k)}(\mathbf{z})$ is

$$\begin{aligned} Q_{h^{(k)}}(\mathbf{w}) = &-\frac{1}{2\alpha_k}\|\alpha_k\mathbf{M}_k^{-1}\mathbf{w} - \boldsymbol{\theta}^{(k)} + \alpha_k\mathbf{M}_k^{-1}\nabla_{\boldsymbol{\theta}}F(\boldsymbol{\theta}^{(k)})\|^2_{\mathbf{M}_k} - R^*(\mathbf{w}) \\ &- R(\boldsymbol{\theta}^{(k)}) - \frac{1}{2\alpha_k}\|\alpha_k\nabla_{\boldsymbol{\theta}}F(\boldsymbol{\theta}^{(k)})\|^2_{\mathbf{M}_k} + \frac{1}{2\alpha_k}\|\boldsymbol{\theta}^{(k)} - \alpha_k\mathbf{M}_k^{-1}\nabla_{\boldsymbol{\theta}}F(\boldsymbol{\theta}^{(k)})\|^2_{\mathbf{M}_k} \end{aligned}$$

and with the same arguments as before, it can be shown that $Q_{h^{(k)}}(\mathbf{w}) \leq h^{(k)}(\mathbf{z})$ for all $\mathbf{w}, \mathbf{z}$, and that $\mathbf{z}^{(k)} = \boldsymbol{\theta}^{(k)} - \alpha_k\nabla_{\boldsymbol{\theta}}F(\boldsymbol{\theta}^{(k)}) - \alpha_k\mathbf{M}_k^{-1}\mathbf{w}^*$, where $\mathbf{w}^*$ is the solution of the dual problem. If $(\mathbf{w}^{(j)})_{j\in\mathbb{N}}$ is a sequence that converges to the dual maximum $\mathbf{w}^*$, then $\tilde{z}^{(k)} = \boldsymbol{\theta}^{(k)} - \alpha_k\nabla_{\boldsymbol{\theta}}F(\boldsymbol{\theta}^{(k)}) - \alpha_k\mathbf{M}_k^{-1}\mathbf{w}^{(j)}$ is the approximate solution of the primal problem that needs to satisfy

$$h^{(k)}(\tilde{z}^{(k)}) - h^{(k)}(\mathbf{z}^{(k)}) \leq h^{(k)}(\tilde{z}^{(k)}) - Q_{h^{(k)}}(\mathbf{w}^{(j)}).$$

Hence, by asking that $h^{(k)}(\tilde{z}^{(k)}) - Q_{h^{(k)}}(\mathbf{w}^{(j)}) \leq -\frac{\tau}{2}h^{(k)}(\tilde{z}^{(k)})$, we obtain the stopping criterion

$$h^{(k)}(\tilde{z}^{(k)}) - Q_{h^{(k)}}(\mathbf{w}^{(j)}) \leq \frac{2}{2+\tau}.$$

Bibliography

[1] S. Arridge et al. "Solving inverse problems using data-driven models". In: *Acta Numerica* 28 (2019), pp. 1–174. doi: 10.1017/S0962492919000059.

[2] H. Attouch, J. Bolte, and B. F. Svaiter. "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods". In: *Mathematical Programming* 137.1-2 (2013), pp. 91–129.

[3] A. Auslender and M. Teboulle. "Interior projection-like methods for monotone variational inequalities". In: *Mathematical Programming* 104.1 (2005), pp. 39–68. doi: 10.1007/s10107-004-0568-x.

[4] A. Auslender and M. Teboulle. "Projected subgradient methods with non-Euclidean distances for non-differentiable convex minimization and variational inequalities". In: *Mathematical Programming* 120.1 (2009), pp. 27–48. doi: 10.1007/s10107-007-0147-z.

[5] A. Beck and M. Teboulle. "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems". In: *SIAM Journal on Imaging Sciences* 2.1 (2009), pp. 183–202. doi: 10.1137/080716542.

[6] Y. Bengio. "Gradient-Based Optimization of Hyperparameters". In: *Neural computation* 12.8 (2000), pp. 1889–1900.

[7] J. Bergstra and Y. Bengio. "Random Search for Hyper-Parameter Optimization." In: *Journal of Machine Learning Research* 13.2 (2021).

[8] M. Bertero et al. "A discrepancy principle for Poisson data". In: *Inverse Problems* 26.10 (2010), p. 105004. doi: 10.1088/0266-5611/26/10/105004.

[9] M. Bertero, P. Boccacci, and V. Ruggiero. *Inverse Imaging with Poisson Data: From Cells to Galaxies*. Institute of Physics Publishing, 2018. isbn: 9780750314381.

[10] C. Bertocchi et al. "Deep unfolding of a proximal interior point method for image restoration". In: *Inverse Problems* 36.3 (2020), p. 034005. doi: 10.1088/1361-6420/ab460a.

[11] D. Bertsekas. *Nonlinear Programming*. Athena scientific optimization and computation series. Athena Scientific, 2016. isbn: 9781886529052.

[12] E. Betzig et al. "Imaging Intracellular Fluorescent Proteins at Nanometer Resolution". In: *Science* 313.5793 (2006), pp. 1642–1645. doi: 10.1126/science.1127344.

[13] M. Blondel et al. "Efficient and Modular Implicit Differentiation". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 5230–5242.

[14] A. Boggess and F. Narcowich. *A First Course in Wavelets with Fourier Analysis*. Wiley, 2011. isbn: 9781118211151.

[15] J. Bolte et al. "Clarke Subgradients of Stratifiable Functions". In: *SIAM Journal on Optimization* 18.2 (2007), pp. 556–572.

[16] S. Bonettini and M. Prato. "New Convergence Results for the Scaled Gradient Projection Method". In: *Inverse Problems* 31.9 (2015), p. 095008. doi: 10.1088/0266-5611/31/9/095008.

[17] S. Bonettini, R. Zanella, and L. Zanni. "A scaled gradient projection method for constrained image deblurring". In: *Inverse Problems* 25.1 (2008), p. 015002. doi: 10.1088/0266-5611/25/1/015002.

[18] S. Bonettini, S. Rebegoldi, and V. Ruggiero. "Inertial Variable Metric Techniques for the Inexact Forward–Backward Algorithm". In: *SIAM Journal on Scientific Computing* 40.5 (2018), A3180–A3210. doi: 10.1137/17M116001X.

[19] S. Bonettini, M. Prato, and S. Rebegoldi. "New Convergence Results for the Inexact Variable Metric Forward–Backward Method". In: *Applied Mathematics and Computation* 392 (2021).

[20] S. Bonettini et al. "Variable metric inexact line-search-based methods for nonsmooth optimization". In: *SIAM journal on Optimization* 26.2 (2016), pp. 891–921.

[21] S. Bonettini et al. "Recent Advances in Variable Metric First-Order Methods". In: *Computational Methods for Inverse Problems in Imaging*. Springer International Publishing, 2019, pp. 1–31. doi: 10.1007/978-3-030-32882-5_1.

[22] S. Bonettini et al. "An abstract convergence framework with application to inertial inexact forward–backward methods". In: *Computational Optimization and Applications* 84.2 (2023), pp. 319–362.

[23] K. Bredies, K. Kunisch, and T. Pock. "Total Generalized Variation". In: *SIAM Journal on Imaging Sciences* 3.3 (2010), pp. 492–526. doi: 10.1137/090769521.

[24] R. H. Byrd et al. "A Limited Memory Algorithm for Bound Constrained Optimization". In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208. doi: 10.1137/0916069.

[25] L. Calatroni et al. "Adaptive Parameter Selection for Weighted-TV Image Reconstruction Problems". In: *Journal of Physics: Conference Series* 1476.1 (2020), p. 012003. doi: 10.1088/1742-6596/1476/1/012003.

[26] A. Chambolle and C. Dossal. "On the Convergence of the Iterates of the "Fast Iterative Shrinkage/Thresholding Algorithm"". In: *Journal of Optimization Theory and Applications* 166.3 (2015), pp. 968–982. doi: 10.1007/s10957-015-0746-4.

[27] Y. Chen, R. Ranftl, and T. Pock. "Insights Into Analysis Operator Learning: From Patch-Based Sparse Models to Higher Order MRFs". In: *IEEE Transactions on Image Processing* 23.3 (2014), pp. 1060–1072.

[28] P. L. Combettes and J.-C. Pesquet. "Proximal Splitting Methods in Signal Processing". In: *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. Ed. by H. H. Bauschke et al. Springer New York, 2011, pp. 185–212. isbn: 978-1-4419-9569-8. doi: 10.1007/978-1-4419-9569-8_10.

[29] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[30] K. Dabov et al. "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering". In: *IEEE Transactions on Image Processing* 16.8 (2007), pp. 2080–2095. doi: 10.1109/TIP.2007.901238.

[31] J. C. De los Reyes. "Bilevel Imaging Learning Problems as Mathematical Programs with Complementarity Constraints: Reformulation and Theory". In: *SIAM Journal on Imaging Sciences* 16.3 (2023), pp. 1655–1686.

[32] S. Dempe. *Foundations of Bilevel Programming*. Springer New York, NY, 2002.

[33] S. Dempe and A. B. Zemkoho. "KKT Reformulation and Necessary Conditions for Optimality in Nonsmooth Bilevel Optimization". In: *SIAM Journal on Optimization* 24.4 (2014), pp. 1639–1669.

[34] J. Domke. "Generic Methods for Optimization-Based Modeling". In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*. Ed. by N. D. Lawrence and M. Girolami. Vol. 22. Proceedings of Machine Learning Research. PMLR, 2012, pp. 318–326.

[35] H. Drucker et al. "Support Vector Regression Machines". In: *Advances in Neural Information Processing Systems*. Ed. by M. Mozer, M. Jordan, and T. Petsche. Vol. 9. MIT Press, 1996.

[36] A. Effland et al. "Variational Networks: An Optimal Control Approach to Early Stopping Variational Methods for Image Restoration". In: *Journal of Mathematical Imaging and Vision* 62.3 (2020), pp. 396–416. doi: 10.1007/s10851-019-00926-8.

[37] F.-L. Fan et al. "On Interpretability of Artificial Neural Networks: A Survey". In: *IEEE Transactions on Radiation and Plasma Medical Sciences* 5.6 (2021), pp. 741–760. doi: 10.1109/TRPMS.2021.3066428.

[38] A. Fischer, A. B. Zemkoho, and S. Zhou. "Semismooth Newton-type method for bilevel optimization: global convergence and extensive numerical experiments". In: *Optimization Methods and Software* 37.5 (2022), pp. 1770–1804.

[39] J. Fliege, A. Tin, and A. B. Zemkoho. "Gauss–Newton-Type Methods for Bilevel Optimization". In: *Computational Optimization and Applications* 78 (2021), pp. 793–824.

[40] M. Forte and F. Pitié. "F, B, Alpha Matting". In: *CoRR* (2020). arXiv: 2003.07711.

[41] L. Franceschi et al. "Bilevel Programming for Hyperparameter Optimization and Meta-Learning". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1568–1577.

[42] G. Franchini, V. Ruggiero, and L. Zanni. "On the Steplength Selection in Stochastic Gradient Methods". In: *Numerical Computations: Theory and Algorithms*. Ed. by Y. D. Sergeyev and D. E. Kvasov. Cham: Springer International Publishing, 2020, pp. 186–197. isbn: 978-3-030-39081-5.

[43] G. Franchini, V. Ruggiero, and L. Zanni. "Ritz-like values in steplength selections for stochastic gradient methods". In: *Soft Computing* 24.23 (2020), pp. 17573–17588. doi: 10.1007/s00500-020-05219-6.

[44] G. Franchini, V. Ruggiero, and L. Zanni. "Steplength and Mini-batch Size Selection in Stochastic Gradient Methods". In: *Machine Learning, Optimization, and Data Science*. Ed. by G. Nicosia et al. Cham: Springer International Publishing, 2020, pp. 259–263. isbn: 978-3-030-64580-9.

[45] G. Franchini et al. "Neural architecture search via standard machine learning methodologies". In: *Mathematics in Engineering* 5.1 (2023), pp. 1–21.

[46] P. I. Frazier. *A Tutorial on Bayesian Optimization*. 2018. arXiv: 1807.02811 [stat.ML].

[47] J. Frecon, S. Salzo, and M. Pontil. "Bilevel learning of the Group Lasso structure". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[48] S. Gazagnes, E. Soubies, and L. Blanc-Féraud. "High density molecule localization for super-resolution microscopy using CEL0 based sparse approximation". In: *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*. 2017, pp. 28–31. doi: 10.1109/ISBI.2017.7950460.

[49] T. Germer, T. Uelwer, and S. Harmeling. "Deblurring photographs of characters using deep neural networks". In: *Inverse Problems and Imaging* 17.5 (2023), pp. 993–1007.

[50] S. Ghadimi and M. Wang. *Approximation Methods for Bilevel Programming*. 2018. arXiv: 1802.02246 [math.OC].

[51] K. Gregor and Y. LeCun. "Learning Fast Approximations of Sparse Coding". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 399–406. isbn: 9781605589077.

[52] M. Gruber. *Improving Efficiency by Shrinkage: The James–Stein and Ridge Regression Estimators*. Statistics: A Series of Textbooks and Monographs. Taylor & Francis, 1998. isbn: 9780824701567.

[53] M. Hintermüller, K. Papafitsoros, and C. N. Rautenberg. "Analytical aspects of spatially adapted total variation regularisation". In: *Journal of Mathematical Analysis and Applications* 454.2 (2017), pp. 891–935. doi: https://doi.org/10.1016/j.jmaa.2017.05.025.

[54] J. Huang and D. Mumford. "Statistics of natural images and models". In: *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. Vol. 1. 1999, 541–547 Vol. 1. doi: 10.1109/CVPR.1999.786990.

[55] S. Hurault, A. Leclaire, and N. Papadakis. "Proximal Denoiser for Convergent Plug-and-Play Optimization with Nonconvex Regularization". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 9483–9505.

[56] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[57] A. Koulouri, P. Heins, and M. Burger. "Adaptive Superresolution in Deconvolution of Sparse Peaks". In: *IEEE Transactions on Signal Processing* 69 (2021), pp. 165–178. doi: 10.1109/TSP.2020.3037373.

[58] O. Kupyn et al. "DeblurGAN-v2: Deblurring (Orders-of-Magnitude) Faster and Better". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 8877–8886. doi: 10.1109/ICCV.2019.00897.

[59] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444. doi: 10.1038/nature14539.

[60] V. I. Levenshtein et al. "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals". In: *Soviet physics doklady*. Vol. 10. 8. 1966, pp. 707–710.

[61] D. Maclaurin, D. Duvenaud, and R. Adams. "Gradient-based Hyperparameter Optimization through Reversible Learning". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. PMLR, 2015, pp. 2113–2122.

[62] D. Martin et al. "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics". In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 2. 2001, pp. 416–423. doi: 10.1109/ICCV.2001.937655.

[63] S. Mehmood and P. Ochs. "Automatic Differentiation of Some First-Order Methods in Parametric Optimization". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 1584–1594.

[64] A. Mittal, A. K. Moorthy, and A. C. Bovik. "No-Reference Image Quality Assessment in the Spatial Domain". In: *IEEE Transactions on Image Processing* 21.12 (2012), pp. 4695–4708. doi: 10.1109/TIP.2012.2214050.

[65] P. Nair, R. G. Gavaskar, and K. N. Chaudhury. "Fixed-Point and Objective Convergence of Plug-and-Play Algorithms". In: *IEEE Transactions on Computational Imaging* 7 (2021), pp. 337–348.

[66] Y. Nesterov. "Smooth minimization of non-smooth functions". In: *Mathematical Programming* 103.1 (2005), pp. 127–152. doi: 10.1007/s10107-004-0552-5.

[67] P. Ochs. "Unifying abstract inexact convergence theorems and block coordinate variable metric iPiano". In: *SIAM Journal on Optimization* 29.1 (2019), pp. 541–570.

[68] P. Ochs and T. Pock. "Adaptive FISTA for Nonconvex Optimization". In: *SIAM Journal on Optimization* 29.4 (2019), pp. 2482–2503. doi: 10.1137/17M1156678.

[69] P. Ochs et al. "Bilevel Optimization with Nonsmooth Lower Level Problems". In: *Scale Space and Variational Methods in Computer Vision*. Cham: Springer International Publishing, 2015, pp. 654–665.

[70] P. Ochs et al. "Techniques for Gradient-Based Bilevel Optimization with Non-smooth Lower Level Problems". In: *Journal of Mathematical Imaging and Vision* 56.2 (2016), pp. 175–194. doi: 10.1007/s10851-016-0663-7.

[71] J. V. Outrata. "On the Numerical Solution of a Class of Stackelberg Problems". In: *Zeitschrift für Operations Research* 34 (1990), pp. 255–277.

[72] F. Pedregosa. "Hyperparameter Optimization with Approximate Gradient". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 737–746.

[73] D. M. Pelt and J. A. Sethian. "A Mixed-Scale Dense Convolutional Neural Network for Image Analysis". In: *Proceedings of the National Academy of Sciences* 115.2 (2018), pp. 254–259. doi: 10.1073/pnas.1715832114.

[74] D. Ren et al. "Simultaneous Fidelity and Regularization Learning for Image Restoration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.1 (2021), pp. 284–299. doi: 10.1109/TPAMI.2019.2926357.

[75] A. Repetti and Y. Wiaux. "Variable Metric Forward-Backward Algorithm for Composite Minimization Problems". In: *SIAM Journal on Optimization* 31.2 (2021), pp. 1215–1241. doi: 10.1137/19M1277552.

[76] N. A. B. Riis, Y. Dong, and P. C. Hansen. "Computed tomography with view angle estimation using uncertainty quantification". In: *Inverse Problems* 37.6 (2021), p. 065007.

[77] H. Robbins and S. Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407.

[78] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[79] Y. Romano, M. Elad, and P. Milanfar. "The Little Engine That Could: Regularization by Denoising (RED)". In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1804–1844. doi: 10.1137/16M1102884.

[80] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by N. Navab et al. Cham: Springer International Publishing, 2015, pp. 234–241. isbn: 978-3-319-24574-4.

[81] L. I. Rudin, S. Osher, and E. Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: Nonlinear Phenomena* 60.1 (1992), pp. 259–268.

[82] M. J. Rust, M. Bates, and X. Zhuang. "Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM)". In: *Nature Methods* 3.10 (2006), pp. 793–796. doi: 10.1038/nmeth929.

[83] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[84] D. Scieur et al. "The Curse of Unrolling: Rate of Differentiating Through Optimization". In: *Neural Information Processing Systems*. 2022. url: +.

[85] D. di Serafino and M. Pragliola. *Automatic parameter selection for the TGV regularizer in image restoration under Poisson noise*. 2022. arXiv: 2205.13439 [math.NA].

[86] A. Sinha, P. Malo, and K. Deb. "A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications". In: *IEEE Transactions on Evolutionary Computation* 22.2 (2018), pp. 276–295.

[87] E. Soubies, L. Blanc-Féraud, and G. Aubert. "A Continuous Exact ℓ_0 Penalty (CEL0) for Least Squares Regularized Problem". In: *SIAM Journal on Imaging Sciences* 8.3 (2015), pp. 1607–1639. doi: 10.1137/151003714.

[88] S. Sreehari et al. "Plug-and-Play Priors for Bright Field Electron Tomography and Sparse Interpolation". In: *IEEE Transactions on Computational Imaging* 2.4 (2016), pp. 408–423. doi: 10.1109/TCI.2016.2599778.

[89] H. von Stackelberg. *The Theory of the Market Economy*. William Hodge, 1952.

[90] V. Stergiopoulou et al. "COL0RME: Super-resolution microscopy based on sparse blinking/fluctuating fluorophore localization and intensity estimation". In: *Biological Imaging* 2 (2022), e1.

[91] V. Stergiopoulou et al. "Fluctuation-Based Deconvolution in Fluorescence Microscopy Using Plug-and-Play Denoisers". In: *Scale Space and Variational Methods in Computer Vision*. Ed. by L. Calatroni et al. Springer International Publishing, 2023, pp. 498–510.

[92] E. Suonperä and T. Valkonen. "Linearly convergent bilevel optimization with single-step inner methods". In: *Computational Optimization and Applications* (2023). doi: 10.1007/s10589-023-00527-7.

[93] C. Szegedy et al. *Intriguing Properties of Neural Networks*. 2014. arXiv: 1312.6199 [cs.CV].

[94] R. Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. issn: 00359246.

[95] A. Tikhonov et al. *Numerical Methods for the Solution of Ill-Posed Problems*. Mathematics and Its Applications. Springer Netherlands, 2013. isbn: 9789401584807.

[96] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Deep Image Prior". In: *International Journal of Computer Vision* 128.7 (2020), pp. 1867–1888. doi: 10.1007/s11263-020-01303-4.

[97] J. Xiang, Y. Dong, and Y. Yang. "FISTA-Net: Learning a Fast Iterative Shrinkage Thresholding Network for Inverse Problems in Imaging." eng. In: *IEEE Trans Med Imaging* 40.5 (2021), pp. 1329–1339. doi: 10.1109/TMI.2021.3054167.

[98] J. Ye and D. L. Zhu. "Optimality Conditions for Bilevel Programming Problems". In: *Optimization* 33 (1995), pp. 9–27.

[99] K. Zhang, L. V. Gool, and R. Timofte. "Deep Unfolding Network for Image Super-Resolution". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[100] W. Zhou et al. "Image Quality Assessment: from Error Visibility to Structural Similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. doi: 10.1109/TIP.2003.819861.

[101] X. Zhou. *On the Fenchel Duality between Strong Convexity and Lipschitz Continuous Gradient*. 2018. arXiv: 1803.06573 [math.OC].

[102] D. Zoran and Y. Weiss. "From learning models of natural image patches to whole image restoration". In: *2011 International Conference on Computer Vision*. 2011, pp. 479–486. doi: 10.1109/ICCV.2011.6126278.