HAL open science

An Unrolled Half-Quadratic Approach for Sparse Signal Recovery in Spectroscopy

Mouna Gharbi, Emilie Chouzenoux, Jean-Christophe Pesquet

To cite this version:

Mouna Gharbi, Emilie Chouzenoux, Jean-Christophe Pesquet. An Unrolled Half-Quadratic Approach for Sparse Signal Recovery in Spectroscopy. Signal Processing, 2024, 218, 10.1016/j.sigpro.2023.109369 . hal-04229774v2

HAL Id: hal-04229774

https://inria.hal.science/hal-04229774v2

Submitted on 22 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

An Unrolled Half-Quadratic Approach for Sparse Signal Recovery in Spectroscopy

Mouna Gharbi[a], Emilie Chouzenoux[a], Jean-Christophe Pesquet[a]

[a] *CVN, CentraleSupélec, Inria Saclay, University Paris Saclay, Gif-sur-Yvette, France*

Abstract

This paper addresses the problem of sparse signal reconstruction frequently arising in analytical chemistry. Spectroscopic restoration is an important task that was tackled using two distinct approaches. First, model-based iterative methods were deployed. These methods rely on a strong theoretical background. However they face practical obstacles in terms of hyperparameter tuning. Conversely, approaches based on deep neural networks are fast and easy to implement. Nonetheless, they are often described as black-box tools, and their stability/robustness are ongoing research topics. In this paper, we propose a deep neural network based on unrolling a Majorization-Minimization (MM) Half-Quadratic (HQ) algorithm. This allows us to build interpretable layers mirroring iterations, making it possible to learn automatically data-driven hyperparameters such as regularization and stepsizes. Furthermore, we propose a dictionary of custom activation functions derived from potentials used in the original variational model. This interpretation of activations can be useful for analyzing the stability of neural networks. The efficiency of our method in comparison to iterative and learning-based methods is showcased through various experiments conducted on realistic mass spectrometry (MS) databases with various blur kernels and noise levels. All experiments show an improved reconstruction quality of chemical signals, while maintaining a low execution time at the test phase. All the results are reproducible, through the code shared on a repository available at `https://github.com/GHARBIMouna/Unrolled-Half-Quadratic`.

Keywords: Unrolled architecture, Signal reconstruction, Half-Quadratic, Spectroscopy.

1. Introduction

Spectroscopy gathers data acquisition/processing pipelines aiming to decipher the chemical properties of a material or a compound. Let us cite, for instance, mass spectrometry (MS) [1] and nuclear magnetic resonance (NMR) [2]. Despite constant improvements in physical acquisition devices, the acquisition process often remains noisy and prone to degradation. More formally, let $\overline{x} \in \mathbb{R}^n$ be an original chemical signal we aim to estimate, and let $y \in \mathbb{R}^m$ be an acquired measurement. In many spectroscopy applications (e.g., RMN DOSY [3], RMN relaxometry [4], MS [5]), both are linked by the following model

$$y = H\overline{x} + e, \tag{1}$$

where $H \in \mathbb{R}^{m \times n}$ represents a measurement degradation process modeling for example a blur kernel [6] or a discrete Laplace transform [7], and $e \in \mathbb{R}^m$ is an additive noise corrupting the data.

To draw relevant chemical interpretations, measurements y need to be processed to recover an estimate of the original data $\overline{x}$, which amounts to inverting Model (1).

In a real-life application context, this inversion process may encounter several difficulties. On the one hand, the ill-posedness of (1), linked to the poor conditioning of H, prevents the use of basic inversion or filtering approaches. On the other hand, the necessity to solve (1) multiple times, for each new acquisition, induces hard constraints on the processing time. To cope with the aforementioned challenges, two main strategies in literature have been considered. A first set of methods relies on optimization, where a criterion combining a data fidelity term and a penalization term is defined and minimized to estimate the signal of interest. This approach was employed in many spectroscopy restoration works such as [5, 8] for MS data processing, [4, 9, 10, 11, 12] in the context of NMR relaxometry, and in DOSY NMR [3].

Recently, a second set of approaches has emerged, which relies on supervised machine learning techniques applied on real or synthetic chemical signal databases to learn mappings (typically, deep neural networks) capable of producing an estimate of the sought signal from the degraded data. For example, dense convolutional neural networks (CNNs) were used to remove artifacts from high-dimensional NMR signals in [13] and, in [14], a residual convolutional neural network, was designed for compressive sensing spectroscopy. The work [15] reviews various deep networks such as LSTMs,

CNNs and DNNs, used in the context of analytical chemistry. Let us also mention the survey paper [16], presenting an overview of various deep learning methods for NMR spectra processing, including the recent works from [17, 18, 19]. Other spectroscopy modalities, such as Raman and FTIR, have also been considered, from the side of deep learning, for instance in [20] with residual Unets and [21] via autoencoders.

Supervised learning-based methods can lead to high-quality results, as soon as a sufficiently large labeled database is available. Furthermore, they have proven to be fast since they benefit from the advances of GPU-based deep learning programming frameworks. However, the interpretability and explainability of these methods remain open issues (see [22, 23, 24] for recent works on these aspects). They might suffer from instability issues [25, 26], and guaranteeing their robustness remains a challenge [27]. In contrast, optimization-based methods offer the required theoretical guarantees, but tuning their hyperparameters might be a tedious task, and their execution time for CPU based implementations may be too high in large scale applicative contexts. For instance, the Matlab implementation[1] of the MS signal restoration method from [5] requires more than 1 minute to restore an MS spectra on a grid of $n = 10^3$ mass values.

Lately, the two previous approaches have been combined, yielding an appealing class of techniques based on 'unrolling' (or 'unfolding'). The core idea is to re-interpret existing iterative optimization algorithms as neural network architectures [28], where layers mirror iterations. A recent survey of unrolling techniques can be found in [29]. This new paradigm presents several advantages. First, new architectures can be proposed, which enjoy an interpretable network structure. Second, prior and/or physical knowledge about the problem at hand can be naturally embedded into the architecture. Third, a reduced execution time is obtained during the test phase, thanks to deep learning GPU-based frameworks. Lastly, unrolled architectures are more flexible than optimization algorithms, and in particular, they offer the possibility to learn some hyperparameters and/or operators of the original algorithm. For instance, stepsizes and regularization weights are learnt in [30], operators and penalty function parameters in [31, 32], and shrinkage functions and biases are learnt in [33].

[1] `https://fr.mathworks.com/matlabcentral/fileexchange/88897-spoq-smooth-sparse-p-over-q-ratio-regularization-toolbox`

In various applications [29], unrolling techniques have been shown to outperform (sometimes, by far) existing methods, with a noticeable gain in explainability and robustness with respect to plain deep learning techniques, while leading to similar (fast) computing times [34]. This paper investigates unrolled algorithms dedicated to the resolution of spectroscopy inverse problems. We explore specifically the unrolling of a popular class of optimization algorithms called Majorization-Minimization (MM) approaches [35]. Namely, we focus on the MM half-quadratic (HQ) approach [36] which minimizes sequentially quadratic majorizing surrogates of the objective function.

HQ was initially developed for image restoration applications [37], then deployed for chemical signal recovery in [38, 39]. Processing large-scale data with MM HQ has been explored in [37] by relying on subspace acceleration strategy [40, 41], or parallel CPU-based implementations [42, 43, 44, 45].

Our contributions in this work are as follows:

- The proposition of an unrolled MM HQ algorithm, for which an efficient GPU-based implementation is available.
- The construction of new activation functions, with established connection with choices of penalty functions.
- The design of a supervised architecture allowing us to learn the involved regularization and stepsize parameters.
- The validation of the proposed approach in the restoration of signals arising from realistic databases of MS signals.

Link with existing works: First, as we mentioned, deep learning strategies or optimization-based ones, have already been considered for tackling spectroscopy inverse problems. Up to our knowledge, deep unrolling has been only scarcely explored in the field of spectroscopy, and its deployment for MS signal processing remains unprecedented. We can mention [46, 47, 48] proposing deep unfolded neural networks to solve inverse problems in γ-ray spectroscopy, photothermal radiometry, and non-uniformly sampled NMR, respectively. In the broader context of sparse 1D signal recovery, we can cite [49, 50, 51]. Although the deep unrolling paradigm in these papers is common to our study, the methods highly differ in the physical model they are considering, in the signal processing problem that is tackled, and in the iterative algorithm that is unrolled. Second, unrolling of MM methods is also not

much explored in the literature. We can only mention [52] which proposes to unroll an HQ method for an application to image deblurring. Third, our analysis connecting penalty functions and activation layers generalizes the one presented in [53].

The paper is organized as follows, Section 2 introduces the optimization problem, and the MM HQ algorithm to solve it. Section 3 details our contributions, mainly the unrolling of HQ algorithm, the proposition of new activation functions, and the design of architectures for hyperparameter learning. Section 4 presents our experimental results and discussions. Lastly, Section 5 concludes the work.

2. Background on MM-based restoration methods

2.1. Problem formulation

A classical strategy for solving inverse problem (1) and producing an estimate $\widehat{x} \in \mathbb{R}^n$ of $\overline{x}$ from observed data y, consists in solving a penalized least-squares minimization problem [54] of the form

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \Big(F(x) = \frac{1}{2} \|Hx - y\|^2 + \lambda_1 \Psi_1(x) + \lambda_2 \Psi_2(x) \Big). \tag{2}$$

Hereabove, the regularization functions $\Psi_i \colon \mathbb{R}^n \to \mathbb{R}$, $i \in \{1, 2\}$, aim at ensuring the stability of the solution $\widehat{x}$, by enforcing *a priori* assumptions related to sparsity. A generic expression for such sparsity-promoting regularization functions is given, for $i \in \{1, 2\}$, by

$$(\forall x \in \mathbb{R}^n) \quad \Psi_i(x) = \sum_{s=1}^{n} \psi_i([x]_s), \tag{3}$$

where $\psi_i : \mathbb{R} \to [0, +\infty)$ is some positive-valued potentials, and $[\cdot]_s$ with $s \in \{1, \ldots, n\}$ denotes the s-th component of a vector in $\mathbb{R}^n$. $\lambda_1 > 0$ and $\lambda_2 > 0$ are regularization hyperparameters balancing the weight between both regularization terms $(\Psi_i)_{i \in \{1,2\}}$, and the least-squares fidelity in the objective function F.

Let us provide the rationale for adopting such a hybrid two-part penalty term. Sparsity-based priors are widely used in the signal processing literature so as to promote few nonzero entries in the sought signal (or in its representation in a transformed domain [55]). Two main families of such priors can be distinguished, namely (i) convex ones, (ii) non-convex ones. Convex sparse

penalties include, as a notorious example, the ℓ_1 norm (related to the LASSO approach in machine learning [56]). An extension of this penalty is the elastic net regularizer [57]. Smooth approximations for it have also been very popular, such as the Huber penalty or the Fair potential [58]. Convex penalties are convenient since they lead to more tractable optimization problems associated with efficient solvers with sounded convergence guarantees. However, they may lead to a bias in the estimated non-zero coefficients, which might be detrimental to the final quality of the results or to their quantitative interpretation. There is a plethora of non-convex priors, starting with the popular ℓ_0 quasi-norm up to its various approximations through norm ratios [59], piecewise defined penalties [60], and smoothed ℓ_0 potentials [61]. Such choices are closer, in essence, to the true goal of sparse restoration, as they directly aim at minimizing the number of significant components in the sought signal, without altering (too much) their amplitudes. Thus, they can lead to much better restoration quality, especially for very sparse signals, and avoid the biasing effect of the convex penalties. However, non-convexity has a significant cost in terms of difficulty of the optimization procedure, making the algorithms sometimes unstable and prone to getting stuck in spurious local minima. In this work, we propose to get the best of both worlds by mixing the two strategies, setting ψ_1 as a convex sparsity-enhancing potential and ψ_2 as a non-convex one. Furthermore, we opt for smooth (i.e., differentiable) regularizers, so as to preserve an efficient optimization procedure. It can be understood that, in such a context, the choice of proper regularization parameters (λ_1, λ_2) will play a critical role. The unrolling approach will serve to decipher, in an automatic, supervised manner, the setting of them, as we will explain in Section 3.

2.2. Quadratic majorization-minimization algorithm

The estimate $\widehat{x}$ of $\overline{x}$ is obtained by minimizing the penalized least squares formulation (2) by assuming that this optimization problem admits a solution. This is, in particular, guaranteed if the cost function in (2) is coercive, which is satisfied, for standard choices of function Ψ_1, provided that $\lambda_1 > 0$. Given our choices for ψ_1 and ψ_2, a popular class of minimization schemes rely on the key concept of quadratic majorization-minimization we introduce hereafter.

Before presenting our algorithm, let us be more specific about the assumption on our potential functions ψ_1 and ψ_2.

Assumption 1. *For every $i \in \{1, 2\}$, $\psi_i : \mathbb{R} \to \mathbb{R}$ is*

(i) a differentiable, even function,

(ii) increasing on $[0, +\infty)$*,*

(iii) such that $\psi_i(\sqrt{\cdot})$ *is concave on* $[0, +\infty)$*.*

In addition, ψ_1 *is convex.*

Assumption 1 is satisfied by a wide class of convex, and non-convex potentials, listed in the second column of Table 1 (see also [62]).

In the context of Problem (2) involving penalties satisfying Assumption 1, an efficient strategy consists of adopting a half-quadratic (HQ) optimizer [63, 36]. HQ belongs to the class of Majorization-Minimization (MM) algorithms, which is a prominent family of methods in the field of optimization [64, 65], with successful applications in the context of signal/image processing, telecommunications, and machine learning [66, 67]. MM techniques substitute the often complex problem of minimizing F by a sequence of more tractable problems, based upon the minimization of sequences of *majorant functions* (also called surrogate functions or auxiliary functions), satisfying the following conditions: at every iteration $k \in \mathbb{N}$ of the algorithm,

$$(\forall x \in \mathbb{R}^n) \quad F(x) \leq Q(x, x_k), \quad F(x_k) = Q(x_k, x_k), \tag{4}$$

where $x_k \in \mathbb{R}^n$ is the current iterate. Under Assumption 1, a quadratic majorant function for F at $x_k \in \mathbb{R}^n$ is given by

$$(\forall x \in \mathbb{R}^n)\ Q(x, x_k) = F(x_k) + \nabla F(x_k)^\top (x - x_k) + \frac{1}{2}(x - x_k)^\top A(x_k)(x - x_k), \tag{5}$$

where $A(x_k)$ is symmetric definite positive matrix in $\mathbb{R}^{n \times n}$, defined as

$$(\forall x \in \mathbb{R}^n) \quad A(x) = H^\top H + \lambda_1 \operatorname{Diag}\{(\omega_1([x]_s))_{1 \leq s \leq n}\} + \lambda_2 \operatorname{Diag}\{(\omega_2([x]_s))_{1 \leq s \leq n}\},$$

where, for every $i \in \{1, 2\}$,

$$(\forall u \in \mathbb{R} \setminus \{0\}) \quad \omega_i(u) = \frac{\varrho_i(u)}{u}, \quad \omega_i(0) = \lim_{u \to 0} \frac{\varrho_i(u)}{u}, \tag{6}$$

with $\varrho_i \colon \mathbb{R} \to \mathbb{R}$ the derivative of the scalar potential function ψ_i. Note that the above limit is well-defined, thanks to Assumption 1 [63]. The expression of the derivatives for a bunch of potential functions satisfying Assumption

1 is provided in the third column of Table 1. Additional properties for the potential functions, also given in the table, will be discussed in Subsection 3.2.

Solving Problem (2) using the MM algorithm associated with quadratic majorant function (5) yields

$$(\forall k \in \mathbb{N}) \quad x_{k+1} = x_k - \gamma_k A(x_k)^{-1} \nabla F(x_k), \tag{7}$$

where $x_0 \in \mathbb{R}^n$ and, for every $k \in \mathbb{N}$, $A(x_k)^{-1}$ is the inverse of $A(x_k)$,

$$\nabla F(x_k) = H^\top (Hx_k - y) + \lambda_1 \Omega_{1,k} x_k + \lambda_2 \Omega_{2,k} x_k,$$

with, for $i \in \{1, 2\}$,

$$\Omega_{i,k} = \mathrm{Diag}\{(\omega_i([x_k]_s))_{1 \leq s \leq n}\}. \tag{8}$$

$\{\gamma_k\}_{k \in \mathbb{N}} \subset (0, 2)$ is a sequence of stepsizes. Algorithm (7) can be viewed as a preconditioned gradient method, or a quasi-Newton method, where the scaling matrix multiplying the gradient descent term is built following the MM principle, in such a way that $(F(x_k))_{k \in \mathbb{N}}$ decreases monotonically. It can also be interpreted as an alternating minimization method acting in an extended space, where the function to minimize is quadratic with respect to some auxiliary vector variable. This interpretation explains the name given to HQ algorithm, and is actually a key element for proving its convergence [68]. The performance of HQ method has been illustrated in the context of signal restoration [63].

The matrix inversion required in (7) can be detrimental to a stable and fast practical implementation of this method. Moreover, setting the stepsize sequence, and the penalization hyperparameters (λ_1, λ_2) in the hybrid regularizer might be tedious. To this end, we propose in the next section an unrolling procedure that helps reducing computational time by creating an architecture with a given number of layers, benefiting from the advantages of implementations on GPUs, and with automatized hyperparameter tuning.

3. Proposed unrolled network

3.1. Unrolled HQ architecture

The first important step for algorithm unrolling consists in translating the algorithm of interest into a multi-layer network architecture. Let us consider

a given number $K > 0$ of iterations for HQ algorithm. We will show in this section that each iteration $k \in \{0, \dots, K-1\}$ of it can be mirrored by a layer $\mathcal{L}_k$ of a feedforward network with a residual connection [69]. Let us rewrite the update rule of HQ algorithm (7) as

$$(\forall k \in \mathbb{N}) \quad x_{k+1} = x_k - W_k \nabla F(x_k), \tag{9}$$

where $x_0 \in \mathbb{R}^n$,

$$W_k = \gamma_k A(x_k)^{-1}, \tag{10}$$

and the gradient expression is given in (8). We now introduce linear and non linear operators that will play the role of linear layers and non-linear functions in the unrolled architecture. Let

$$V_0 = H^\top H, \quad V_1 = I_n, \quad V_2 = I_n, \quad V_3 = [I_n \quad \lambda_1 I_n \quad \lambda_2 I_n], \tag{11}$$

$$b_0 = -H^\top y, \quad b_1 = 0, \quad b_2 = 0, \quad b_3 = 0, \tag{12}$$

where I_n is the identity matrix of size $n \times n$. Moreover, let R_0 to R_3 be defined as, for every $x \in \mathbb{R}^n$,

$$R_0(x) = R_3(x) = x, \; R_1(x) = \big(\varrho_1([x]_s)\big)_{1 \le s \le n}, \; R_2(x) = \big(\varrho_2([x]_s)\big)_{1 \le s \le n}. \tag{13}$$

Note these activations operate componentwise. Activations R_0 and R_3, both reducing to identity operators, are introduced for the sake of completeness of the presentation, so they play no role in the final output expression. Given the expressions (8) and (9), and the above definitions, we can rewrite the K first iterations of the HQ algorithm (7) as

$$(\forall k \in \{0, \dots, K-1\}) \quad x_{k+1} = x_k - W_k \left(R_3 \left(V_3 \begin{bmatrix} R_0(V_0 x_k + b_0) \\ R_1(V_1 x_k + b_1) \\ R_2(V_2 x_k + b_2) \end{bmatrix} + b_3 \right) \right), \tag{14}$$

with $x_0 \in \mathbb{R}^n$. Equation (14) has a remarkable structure, where we can identify fundamental bricks of a standard deep neural network architecture. For every $k \in \{0, \dots, K-1\}$, the input x_k of layer $\mathcal{L}_k$ first goes simultaneously through the 3-channel linear layer with weights V_0, V_1, V_2, and biases b_0, b_1, b_2, then through the activation operators R_0, R_1, R_2. The three outputs of these activation blocks are concatenated and go through a linear layer with

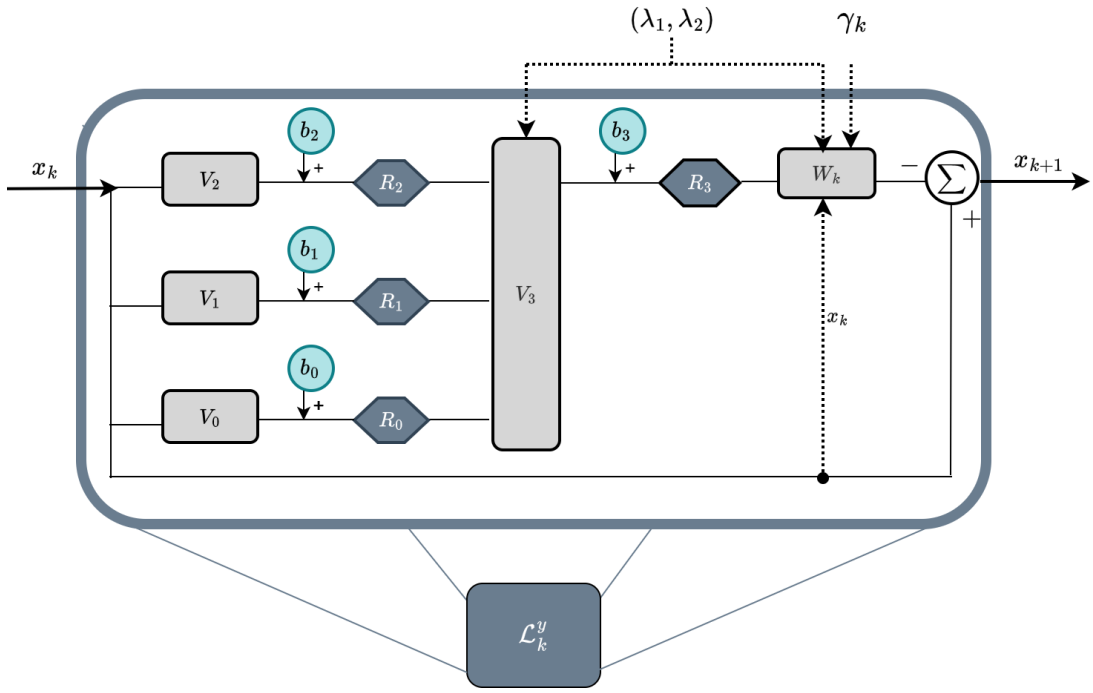

Figure 1: Unrolled HQ architecture: Layer $\mathcal{L}_k^y$ mimicks the k-th iteration of the HQ algorithm (7), for $k \in \{0, \ldots, K-1\}$, and $K > 0$, as described in (14). V_0, V_1, V_2 and V_3 and W_k are weight operators while b_0, b_1, b_2, b_3 are biases. R_0, R_1, R_2 and R_3 are activation functions. Black dotted arrows showcase the dependency of the pointed block to the mentioned value. For instance, block V_3 requires the knowledge of (λ_1, λ_2).

weight matrix V_3, aggregating them. This layer has bias b_3, and activation R_3 (here reducing to identity). The output x_{k+1} is finally obtained by computing the difference between the input x_k and the output of R_3 composed with the linear operator W_k. Such a last step is similar to a residual connection. Each layer of the network depends on the noisy observation. For every $k \in \{0, \ldots, K-1\}$ the k-th layer is denoted as $\mathcal{L}_k^y$, so that, for every $k \in \{0, \ldots, K-1\}$,

$$x_{k+1} = \mathcal{L}_k^y(x_k). \tag{15}$$

Figure 1 summarizes the structure of layer $\mathcal{L}_k^y$ for some $k \in \{0, \ldots, K-1\}$.

This equivalent formulation of the HQ algorithm presents several advantages. As we will demonstrate in Subsection 3.3, it is a first step towards the supervised learning of its hyperparameters such as the regularization pa-

rameters and the stepsizes. The optimization of these parameters is based on a dedicated loss function directly related to the application at hand (e.g., mean squared error on the restored signal). This deep unrolling strategy leads to the design of a deep architecture with interpretable structure, which contrasts with traditional deep networks. Namely, the involved weight and bias operators have specific forms derived from the original inverse problem formulation. The nonlinear activation functions are not chosen in an ad hoc manner, but they are fully interpretable in an optimization sense. This will be further discussed in Subsection 3.2. Another specificity of our architecture with respect to conventional DL ones is that the weight operator W_k depends on the k-th layer input. This weight operator thus exhibits some similarity to a multivariate attention mechanism. Moreover, relying on the unrolling paradigm offers certain mathematical guarantees of stability and robustness as it has been proven in [30, 70]. Finally, from a practical viewpoint, the neural network-based interpretation of the HQ method allows the deployment of GPU-friendly tools to implement the different bricks of the optimizer. This is beneficial in terms of computational efficiency, during the training process, when supervised unrolling is performed. This is also useful for computational time reduction at the inference stage, with or without trained parameters, as we will demonstrate in our numerical experiments.

3.2. Activation functions

The architecture displayed in Fig. 1 makes use of several activation functions whose properties are further analysed in this section. Since R_0 and R_3 reduce to identity functions, we focus our discussion on R_1 and R_2, which are directly related to our choice for the penalization functions Ψ_1 and Ψ_2 in (2). As can be seen in (13), R_1 and R_2 are applied componentwise. Their expressions involve the scalar potentials ϱ_1 and ϱ_2 that are the derivatives of ψ_1 and ψ_2, respectively. We list in the third column of Table 1 the derivative ϱ for typical choices of function ψ satisfying Assumption 1. Interestingly, one can recognize that some correspond to commonly used activation functions in neural networks, while others are less familiar. For example, the hyperbolic tangent is obtained for the Green potential, Eliott activation function corresponds to the Fair potential, and the Huber penalty leads to a shifted version of capped ReLU. As shown in [26], proximal properties of activation functions play an important role in the mathematical understanding of neural networks, in particular in their stability analysis. We analyse further the properties of ϱ in that respect. Let $\Gamma_0(\mathbb{R})$ denote the class of lower-

semicontinuous convex functions from $\mathbb{R}$ to $\mathbb{R} \cup \{+\infty\}$ which are proper (i.e., at least finite in one point). We recall that, if φ belongs to $\Gamma_0(\mathbb{R})$, its proximity operator $\operatorname{prox}_\varphi \colon \mathbb{R} \to \mathbb{R}$ is defined as

$$(\forall x \in \mathbb{R}) \quad \operatorname{prox}_\varphi(x) = \underset{t \in \mathbb{R}}{\operatorname{argmin}}\, \varphi(t) + \frac{1}{2}(t-x)^2. \tag{16}$$

We are now ready to state the following property.

Proposition 1. *Let $\psi \colon \mathbb{R} \to \mathbb{R}$ be a differentiable even function and let ϱ be its derivative. Assume that ϱ is 1-Lipschitz on $\mathbb{R}$, i.e.*

$$(\forall (x,t) \in \mathbb{R}^2) \quad |\varrho(x) - \varrho(t)| \leq |x-t|.$$

1. *There exists $\alpha \in [1/2, 1]$ and an even function $\varphi \in \Gamma_0(\mathbb{R})$ such that*

$$(\forall x \in \mathbb{R}) \quad \varrho(x) = x + 2\alpha(\operatorname{prox}_\varphi(x) - x). \tag{17}$$

2. *Let*

$$(\forall x \in \mathbb{R}) \quad \widetilde{\varphi}(x) = \varphi(x) + \frac{x^2}{2} \tag{18}$$

and let $\widetilde{\varphi}^* \in \Gamma_0(\mathbb{R})$ be the Fenchel-Young conjugate of $\widetilde{\varphi}$. [2] *Then*

$$(\forall x \in \mathbb{R}) \quad \psi(x) \overset{\mathrm{c}}{=} (1-2\alpha)\frac{x^2}{2} + 2\alpha\widetilde{\varphi}^*(x), \tag{19}$$

where $\overset{\mathrm{c}}{=}$ designates equality up to an additive constant.

3. *If ψ is convex, then $\alpha = 1/2$.*

Proof. See Section S1 of our supplementary material. □

The interplay between penalty function ψ, the activation ρ, and the underlying convex function φ is summarized in Fig. 2. The previous proposition ensures that, when ψ is a convex penalty function satisfying Assumption 1, there exists $\varphi \in \Gamma_0(\mathbb{R})$ such that

$$\varrho = \operatorname{prox}_\varphi. \tag{20}$$

[2] The conjugate of $\widetilde{\varphi}$ is defined as $(\forall u \in \mathbb{R})\ \widetilde{\varphi}^*(u) = \sup_{x \in \mathbb{R}} xu - \widetilde{\varphi}(x)$.

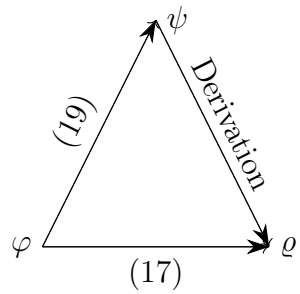

Figure 2: Interplay between penalty ψ, activation ϱ and convex function φ.

In this case, ϱ enjoys the property of being firmly nonexpansive, i.e. increasing and 1-Lipschitz [71, Proposition 4.4] . As emphasized in [26], it turns out that most of the existing activation functions used in neural networks are proximity operators of convex functions (e.g. ReLU activation function). On the other hand, if ψ is non convex, then (17) shows that ϱ is the overrelaxation of the proximity operator of a function in $\Gamma_0(\mathbb{R})$. According to (E1) in Section S1 of the supplementary material, ϱ is then α-averaged, which means that it satisfies the inequality [71, Proposition 4.35] [72],

$$(\forall (x,t) \in \mathbb{R})^2 \quad |\varrho(x) - \varrho(t)|^2 \leq |x-t|^2 - \frac{1-\alpha}{\alpha}|x - \varrho(x) - t + \varrho(t)|^2. \quad (21)$$

The smallest α, the tightest the upper bound above. In other words, a small value of α induces more stability. Equation (19) gives another interpretation of α. It shows that ψ is a weakly-convex function [73, Proposition 4.3] with modulus $2\alpha - 1$. Thus, α can be viewed as a measure of the degree of nonconvexity of potential ψ. To conclude, the activation functions in our unrolled architecture benefit from specific stability properties inherited from their optimization-based interpretation. This is in line with recent works [26, 53]. In [53], diffusion methods, wavelet approaches, and variational regularization are employed to perform a denoising task. The link with CNN architectures is made through residual neural blocks. Diffusitivies, shrinkage functions, and regularizers are reinterpreted as activation functions. The authors emphasize the prospects of nonmonotone functions, rarely used in the deep learning community. Through our method, we obtain a dictionary of novel activation functions, associated to usual convex and non-convex sparsity promoting penalizations. We provide, the values for α in the last column of Table 1. The convex function φ associated to a penalty ψ is implicitly defined by (18) and

(19). As shown in Section S3 of our supplementary material, it is possible for most of the convex penalties in Table 1 to derive a closed form expression for φ.

Penalty Name	Penalization $\psi(t)$, $t \in \mathbb{R}$	Activation $\varrho(t)$, $t \in \mathbb{R}$	α
Convex penalties			
Fair potential	$\delta(\lvert t\rvert - \delta \log(\frac{\lvert t\rvert}{\delta} + 1))$	$\frac{\delta t}{\lvert t\rvert + \delta}$	$\frac{1}{2}$
Huber	$\begin{cases} \frac{t^2}{2} & \text{if } \lvert t\rvert < \delta \\ \delta\lvert t\rvert - \delta^2/2 & \text{otherwise} \end{cases}$	$\begin{cases} t & \text{if } \lvert t\rvert < \delta \\ \delta \text{sign}(t) & \text{otherwise} \end{cases}$	$\frac{1}{2}$
Green	$\log(\cosh(t))$	$\tanh(t)$	$\frac{1}{2}$
Hyperbolic	$\frac{\delta^2}{\kappa}\left((1 + \frac{t^2}{\delta^2})^{\frac{\kappa}{2}} - 1\right)$	$t(1 + \frac{t^2}{\delta^2})^{\frac{\kappa}{2}-1}$	$\frac{1}{2}$
Nonconvex penalties			
Welsch	$\delta^2\left(1 - \exp(-\frac{t^2}{2\delta^2})\right)$	$t\exp(-\frac{t^2}{2\delta^2})$	$\exp(-\frac{3}{2}) + \frac{1}{2}$
Geman-McClure	$\frac{\delta^2 t^2}{2\delta^2 + t^2}$	$\frac{4\delta^4 t}{(2\delta^2 + t^2)^2}$	$\frac{5}{8}$
Tukey biweight	$\begin{cases} \delta^2(1 - (1 - \frac{t^2}{6\delta^2})^3) & \text{if } \lvert t\rvert \leq \sqrt{6}\delta \\ 1 & \text{otherwise} \end{cases}$	$\begin{cases} t(1 - \frac{t^2}{6\delta^2})^2 & \text{if } \lvert t\rvert \leq \sqrt{6}\delta \\ 0 & \text{otherwise} \end{cases}$	$\frac{9}{10}$
Hyperbolic tangent	$\delta^2 \tanh(\frac{t^2}{2\delta^2})$	$\frac{t}{(\cosh(\frac{t^2}{2\delta^2})^2}$	0.9581
Cauchy	$\frac{\delta^2}{2}\log(1 + \frac{t^2}{\delta^2})$	$\frac{\delta^2 t}{t^2 + \delta^2}$	$\frac{9}{16}$

Table 1: Examples of penalties ψ satisfying Assumption 1. In the third column, we provide also the expression of their derivatives (i.e., nonlinear activation in our architecture) ϱ and, in the last column, the averaging constants α (see Section 3.2). All expressions are valid for every $t \in \mathbb{R}$, $(\lambda, \delta) \in]0, +\infty[^2$ and $\kappa \in [1, 2]$. The calculations for the averaging constants in non-convex cases are provided in Section S2 of our supplementary material.

3.3. Learning strategy

As aforementioned, the rewriting of HQ algorithm as a multi-layer neural network structure (see Fig. 1) paves the way for supervised learning of its hyperparameters. In this subsection, we showcase our proposed strategy to tune automatically such parameters.

First, regularization parameters (λ_1, λ_2) are untied into two sequences $(\lambda_{i,k})_{0 \leq k \leq K-1}$, $i \in \{1, 2\}$, varying with each layer of the proposed architecture. This is in contrast with classical optimization-based restoration procedures, which rely on setting those hyperparameters over all iterations, relying on empirical or statistical tuning techniques often difficult to implement [74]. Here, we propose instead to learn automatically data-driven parameters. For a given number of layers $K > 0$, each layer $\mathcal{L}_k^y$ indexed by $k \in \{0, \ldots, K-1\}$

is attached to learnable units defined as

$$(\forall i \in \{1,2\}) \quad \lambda_{i,k} = \mathrm{ReLU}\left(\mathrm{FC}_{i,k}\left(([Hx_k - y]_\ell)^2)_{1\leq \ell \leq m}\right)\right). \tag{22}$$

Hereabove, ReLU is the rectified linear unit [75] applied componentwise, aiming at enforcing positivity on regularization parameters, and defined as follows:

$$(\forall z \in \mathbb{R}^m) \quad \mathrm{ReLU}(z) = (\max(0, [z]_\ell))_{1\leq \ell \leq m}. \tag{23}$$

Moreover, $\mathrm{FC}_{i,k}$, $i \in \{1,2\}$, $k \in \{0,\ldots,K-1\}$, are fully connected layers acting on component-wise squares of the data fit residual at the entry of each layer $\mathcal{L}_k^y$. In such way, the regularization parameters automatically adapt to the noise level at hand, without any required prior knowledge. A similar strategy was employed in [30] in the context of image restoration. The rationale behind this choice is derived from the Bayesian interpretation of the objective function, which leads to having regularization hyperparameters proportional to the Gaussian noise variance.

Second, we suggest to also learn, in a supervised fashion, the stepsize sequence $(\gamma_k)_{0\leq k\leq K-1}$ involved in the linear weight layer sequence $(W_k)_{0\leq k\leq K-1}$. The rationale is to tune optimally the contribution of each layer for delivering the best estimate $\widehat{x}$ at the output of the unrolled architecture. In order to reduce spurious local minima in the training phase, we adopt an overparametrization strategy [76], that consists in defining

$$(\forall k \in \{0,\ldots,K-1\}) \quad \gamma_k = \mathrm{ReLU}(c_{k,1} \times c_{k,2} \times \ldots \times c_{k,N}), \tag{24}$$

with $N \geq 1$ (typically, $N = 10$), and $(c_{k,j})_{0\leq k\leq K-1, 1\leq j\leq N}$ are scalars quantities that are learned. Vector $\theta = (\theta_k)_{0\leq k\leq K-1} \in \mathbb{R}^{K(2m+N)}$ gathers the parameters to be learnt, namely $2mK$ weights for the fully-connected layers in (22), and NK values to learn the stepsize quantities as in (24). Simple constraints, such as range or sum-to-one, on the estimates can be easily imposed by appending a dedicated activation function at the end of the unrolled architecture. Such strategy will be employed in our experimental section, using a ReLU as last layer, to enforce the positivity of the restored MS spectra.

The global multi-layer architecture, called **U-HQ** (**U**nrolled **H**alf-**Q**uadratic), is summarized in Fig. 3, with a detailed overview of a layer $\mathcal{L}_k^{\theta_k}$ structure displayed in Fig. 4, responsible for learning regularization and stepsize hyperparameters to be fed to the layer $\mathcal{L}_k^y$ simulating one iteration of the HQ scheme, as shown in Fig. 1. Given a training set $\{(\overline{x}^{(i)}, y^{(i)}),\ i \in \{1,\ldots,S\}\}$ comprised of S pairs of groundtruth and degraded data samples, the prediction

feedforward model for one instance $i \in \{1, \ldots, S\}$ is given by

$$f_\theta(x_0^{(i)}; y^{(i)}) = \mathcal{L}_{K-1}^{y^{(i)}} \circ \cdots \circ \mathcal{L}_k^{y^{(i)}} \circ \cdots \circ \mathcal{L}_0^{y^{(i)}}(x_0^{(i)}), \tag{25}$$

$x_0^{(i)}$ denoting the input for the first layer of the network for the i-th sample. Then, the learnt hyperparameter vector $\widehat{\theta}$ is defined as the solution to the optimization problem

$$\underset{\theta \in \mathbb{R}^{K(2m+1)}}{\text{minimize}} \quad \frac{1}{S} \sum_{i=1}^{S} \mathcal{L}(f_\theta(x_0^{(i)}; y^{(i)}), \overline{x}^{(i)}), \tag{26}$$

with $\mathcal{L}$ a predefined loss, for instance, the mean squared error. Problem (26) can be solved end-to-end through backpropagation with standard neural network training algorithms such as stochastic gradient or ADAM [77]. At the inference stage, for a given degraded signal y, and an initial prediction x_0, the inferred output is given by $\widehat{x} = f_{\widehat{\theta}}(x_0; y)$. The proposed approach benefits from automatic parameter tuning with layer-dependent and data driven structure, while having an interpretable structure.

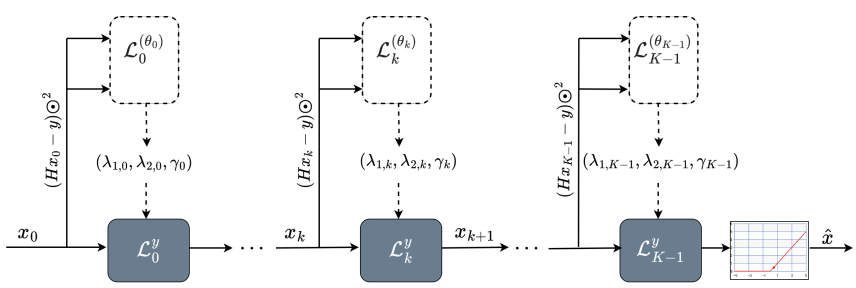

Figure 3: Supervised **U-HQ** network: For every $k \in \{0, \ldots, K-1\}$, the detailed view of layer $\mathcal{L}_k^y$ is given by Fig. 1. Details of the learning block $\mathcal{L}_k^{\theta_k}$ are given in (22) and (24).

4. Restoration of mass spectrometry signals

We now evaluate the performance of **U-HQ** on the restoration of MS sparse signals. We focus on the resolution of inverse problem (1) where $\overline{x} \in \mathbb{R}^n$ is a ground truth signal with length $n = 2000$, $H \in \mathbb{R}^{m \times n}$ is a Toeplitz matrix mimicking a circular padded convolution with a blur kernel, and $e \in \mathbb{R}^m$ corresponds to an additive i.i.d. Gaussian noise with zero-mean and standard deviation $\sigma > 0$.

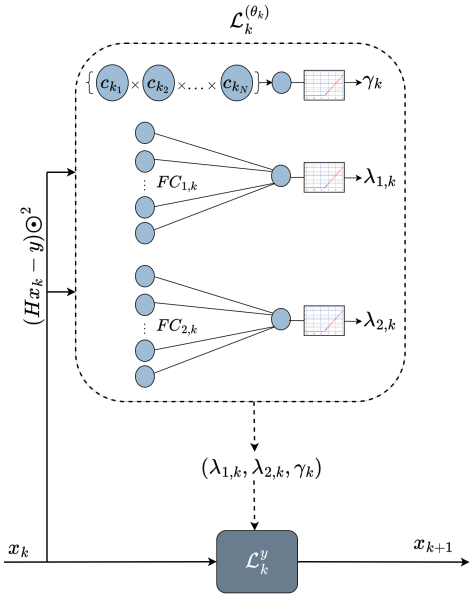

Figure 4: Overview of one layer of proposed architecture **U-HQ**.

4.1. Datasets

Let us first introduce the three datasets we employ to evaluate our method as well as its competitors. In all cases, the dataset is made of original signals $\overline{x}$ paired with degraded signals y obtained through Model (1). Moreover, the blur support size is 50, so that $m = 2049$.

Dataset 1. Signals $\overline{x}$ are real MS spectra extracted from the mass bank inventory of MS spectra [78], considering a monoisotopic mass range of [0,200] ppm and a charge state +1. The spectra intensities are rescaled to reach a maximum value of 100. The blur kernel varies for each signal. It has a "Ricker" shape with spreading factor chosen randomly uniformly in $(0.25, 1)$ [79]. The noise standard variation is also varying throughout the dataset, and is chosen randomly uniformly between 0 and 0.5. The blur kernel is known, while the noise level is assumed to be unknown. Training, validation, and test sets contain 900, 100 and 100 pairs $(\overline{x}, y)$, respectively.

Dataset 2. Signals $\overline{x}$ are also real MS spectra extracted from the same mass spectrometry bank inventory [78], we consider again the monoisotopic mass range [0,200] ppm and a charge state +1. Hereagain, the spectra intensities

are rescaled to reach a maximum value of 100. The blur kernel is chosen as for Dataset 1. The noise standard deviation now varies uniformly between 0.5 and 1. Training, validation, and test sets contain 900, 100 and 100 pairs $(\overline{x}, y)$, respectively.

Dataset 3. Groundtruth signals $\bar{x}$ have a monoisotopic mass between 0 and 200 ppm, similarly to Dataset 1 and Dataset 2. We now use blur kernel with shape "Fraser Suzuki" [80], whose spreading factor is chosen randomly uniformly between 0.25 and 1 and the asymmetry factor between 0.2 and 0.6. The noise standard deviation varies between 0 and 0.5. Training, validation, and test sets contain 900, 100 and, 100 pairs $(\overline{x}, y)$, respectively.

4.2. Experimental settings

Evaluation of the results. All compared supervised models are trained so as to solve (26), where $\mathcal{L}$ is the averaged mean squared error (MSE) between the estimated signal $\widehat{x}$ and the ground truth $\overline{x}$, $\mathrm{MSE}(\overline{x}, \widehat{x}) = (1/n)\|\overline{x} - \widehat{x}\|^2$. MSE metric is also used to finetune regularization hyperparameters, as explained in a forthcoming section. In our comparative tables, we provide the signal-to-noise ratio (SNR, in dB) defined as $\mathrm{SNR}(\overline{x}, \widehat{x}) = 20 \log_{10} \left(\|\overline{x}\| / \|\overline{x} - \widehat{x}\| \right)$, as well as the thresholded SNR metric (TSNR) in dB, defined as the SNR computed only on the support (i.e., indices of non-zero entries) of the sought signal $\overline{x}$. We display both the mean and standard deviation (STD) of these quantitative metrics, computed over the test sets.

Penalties. The regularization functions Ψ_1 and Ψ_2 in (2), are defined as in (3), with ψ_1 the (convex) Fair potential and ψ_2 the (non-convex) Cauchy penalty, whose mathematical expressions and properties are listed in Table 1. They depend on smoothing parameters δ_1 and δ_2, respectively, which are set through a procedure precised hereafter. Within an ablation study, we will also compare to cases when only one of these penalty functions is employed (i.e., when we remove the network branch related to either Ψ_1 or Ψ_2). Results when using a quadratic (i.e., Tikhonov) penalty for ψ_1 (while $\psi_2 = 0$) will also be presented. This experimental plan allows, in particular, to study the role and effect of the layer W_k, and the activations R_1 and R_2, in our **U-HQ** architecture.

Initialization. All the compared approaches are initialized by the null vector, in particular, all the compared neural networks take $x_0 = 0$ as an input.

Programming framework. The methods are implemented using Pytorch (version 1.8.0) under Python (version 3.8.5). The GPU-based code is run on an Nvidia DGX server using a V100 SXM2 GPU card (32 GB of RAM). The CPU is an Intel@ Xeon(R) W-2135 CPU @ 3.70 GHz with 12 active cores.

4.3. Comparison settings

The proposed **U-HQ** approach is compared to various methods, that can be classified into three categories, namely (i) optimization-based methods which rely on the HQ scheme to minimize (2), (ii) deep unrolling techniques relying on similar or distinct architecture than ours, with various training strategies, and (iii) deep neural networks inspired from state-of-the-art architectures deployed for the restoration of sparse and/or spectroscopic signals. Let us specify hereafter the tuning for all the competing methods, including ours.

Optimization-based methods. We implement the HQ algorithm (7) described in Section 2.2 to minimize (2). We run HQ until convergence, that is either until it reaches a maximal number of iterations (here, 100), or until it shows a stabilization of its iterates through $\|x_{k+1} - x_k\|^2/\|x_k\|^2 < 10^{-5}$. We denote **HQ-SC** this approach. We also consider a modified variant of HQ algorithm, where the number of iterations is fixed to a value K identical to the number of layers retained in the unrolled architecture. This method is inspired from early stopping (ES), therefore we name it **HQ-ES**.

For both methods, the stepsize is set to $\gamma_k \equiv 1$ for simplicity. Moreover, the hyperparameters (λ_1, λ_2) are finetuned manually. Specifically, a gridsearch is performed on 200 signal randomly extracted from the training set to minimize the averaged MSE. The resulting parameters are retained later to assess the performance of these methods on the test set. Specifically, we perform the gridsearch using the convex and non-convex potentials separately to finetune both the regularization and smoothness hyperparameters $(\lambda_1, \lambda_2, \delta_1, \delta_2)$. Then, we run HQ algorithm using the hybrid penalty, with (δ_1, δ_2) fixed according to the previous disjoint gridsearch and we finetune the regularization hyperparameters $(\lambda_1,\ \lambda_2)$. The grid used for λ_1 and λ_2 is $[0, 0.01, 0.1, 1, 2, 4, 6, 8, 10, 20]$, while for δ_1 and δ_2 it is $[0.01, 0.1, 0.5, 1, 2, 4, 6, 8]$.

Learning-based methods. In order to assess the benefits of the proposed architecture in **U-HQ**, we provide comparisons with simplified versions of it,

with an increasing number of free parameters, so as to perform an ablation study. We start with a low parametrized model inspired from deep equilibrium (DE) models [81]. Namely, the stepsize is set as $\gamma_k \equiv 1$, and the trained architecture shares the same two parameters (λ_1, λ_2) for all layers. So, for a fixed number of layers K, we have

$$\mathcal{L}_0^{(\theta_0)} = \mathcal{L}_1^{(\theta_1)} = \ldots = \mathcal{L}_{K-1}^{(\theta_{K-1})} = \mathcal{L}^{(\theta)}. \tag{27}$$

This yields the so-called **U-HQ-DE** method. We then implement two architectures entitled **U-HQ-FixS** (**U-HQ Fix S**tepsize) and **U-HQ-FixN** (**U-HQ Fix**ed **N**oise). In both cases, the noise estimation layer from Figure 4 is discarded, and the regularization hyperparameters $(\lambda_{i,k})_{0\leq k\leq K-1}$, are simply defined as outputs of ReLU unit with $2K$ learnable inputs $(a_{i,k})_{0\leq k\leq K-1}$, $i \in \{1, 2\}$. Moreover, in **U-HQ-FixS**, we furthermore impose $\gamma_k \equiv 1$, so that we learn only $(\lambda_{i,k})_{0\leq k\leq K-1}$, while **U-HQ-FixN** learns stepsize hyperparameters $(\gamma_k)_{0\leq k\leq K-1}$ with enforced positivity using ReLU. Finally, we implement **U-HQ-FixN-OverP**. This last architecture uses **U-HQ-FixN** as a baseline, and includes the overparametrization strategy similar as the one described in (24), for the learning of both the regularization and stepsize hyperparameters per layer.

The unrolled schemes **U-HQ-FixS**, **U-HQ-FixN**, **U-HQ-FixN-OverP**, and the proposed **U-HQ**, are experimented using either ψ_1 (convex) or ψ_2 (non-convex), or (ψ_1, ψ_2) (hybrid), in their activation layers. Note that using only one of these two penalizations practically results in training half the number of weights. Regarding **U-HQ-DE** we present the results only for the hybrid penalty pair (ψ_1, ψ_2), for the sake of conciseness. Furthermore, we make comparisons with two unrolled methods recently introduced for sparse signal recovery in [82], namely **U-ISTA** and **U-PD**, based on an unrolling of the ISTA algorithm [83] and the primal dual algorithm [84], respectively. Both methods rely on an ℓ_1 (thus convex) penalization on the signal. Regularization and stepsize parameters of both methods are learned, with enforced positivity using ReLU activations.

Deep learning methods. We compare our proposed approach **U-HQ** to three deep neural network architectures. First, we implement a fully connected network **FCNet**, based on the concatenation of 4 blocks of layers, each composed of a linear map and a ReLU activation layer. Second, we create an autoencoder structure, inspired from [21], named **AE**. Similarly to **FCNet**,

the **AE** composes linear maps and ReLU layers. The difference is that the linear maps are such that the encoder reduces, with a factor 2 per layer, the dimension of the input then the decoder maps, through up-samplings with factor 2, the low-dimensional signals to the output. We set 4 layers in both encoder and decoder parts. We finally deploy a handcrafted Residual UNet, named **ResUNet**, reminiscent from the one proposed in [20], in the context of Raman spectra recovery. The latter architecture relies on an autoencoder structure like UNet [85]. We adapt the code available in the github of [20], by reducing the number of parameters of the network, to limit over-fitting effects. We provide the details of the resulting **ResUNet** architecture in Section S6 of the supplementary material as well as the two comparative deep learning methods.

The acronyms of all the compared approaches are summarized in Table 2.

Method's name	Acronym definition
U-HQ	Unrolled Half Quadratic approach
HQ-SC	Half Quadratic algorithm with Stopping Criteria
HQ-ES	Half Quadratic algorithm with Early Stopping rule
U-HQ-DE	Unrolled Half Quadratic approach with Deep Equilibrium model.
U-HQ-FixS	Unrolled Half Quadratic approach with Fixed Stepsize
U-HQ-FixN	Unrolled Half Quadratic approach with Fixed Noise model
U-HQ-FixN-OverP	Unrolled Half Quadratic approach with Fixed Noise model using over-parametrization
U-PD	Unrolled Primal Dual approach
U-ISTA	Unrolled Iterative Soft Threshholding Algorithm
FCNet	Fully Connected Network
AE	AutoEncoder
ResUNet	Residual UNet

Table 2: Glossary of terms

Training settings. Let us now present the training specifications used for the learning-based methods described in the previous subsection. The number of layers is set, for each dataset and each method, through empirical search, to maintain a trade-off between a low validation loss, a comparable number of parameters between all tested methods, a reasonable computational training time and a stable learning. For architectures **U-HQ**, **U-HQ-FixS**, **U-HQ-FixN**, **U-HQ-FixN-OverP** and **U-HQ-DE**, we set $(\delta_1, \delta_2) = (0.1, 1)$ for Dataset 1 and Dataset 2 and $(\delta_1, \delta_2) = (0.01, 0.1)$ for Dataset 3 in the unrolled HQ methods (while these parameters were finetuned in optimization-based methods). We set initial values of learnable weights/parameters to 1, and the parameter N to 10 in all our overparametrized settings. Training is performed using Adam optimizer [77], with mini-batch size 5. Regarding the training strategy, we set a unique learning rate lr to update the weights in most architectures, at the exception of **U-HQ**. As this latter uses completely different architectures for the stepsize (i.e., simple ReLU layer) and the regularization weights (i.e., fully connected layers), we use two different learning rates, namely lr_γ and lr_λ, respectively. Regarding the deep learning methods, Adam optimizer is also used to minimize the MSE loss. Training parameters, including learning rates, batch sizes, and layer number, for all architectures, are summarized in Sections S4 (for unrolled models) and S6 (for DL models), in our supplementary material.

For reproducibility purposes, we share a repository available at `https://github.com/GHARBIMouna/Unrolled-Half-Quadratic` with our implementation in PyTorch of all the compared algorithms.

4.4. Results

Table 4 summarizes the results in terms of mean and standard deviation of the SNR and TSNR scores computed on test sets, for the three datasets. We recall that the higher the SNR (resp. the TSNR), the better the estimation. Best quantitative results are highlighted in bold style. At a first glance, this confirms the competitiveness of our proposed architecture **U-HQ** with respect to all other compared methods. Let us inspect and discuss more deeply the results.

U-HQ *vs optimization-based methods.* First, let us inspect the results of optimization-based methods. Using the proposed hybrid penalty term with iterative methods **HQ-SC** and **HQ-ES** reaches better metrics compared to using convex or non-convex potentials separately in most cases on all

datasets. This demonstrates the advantage of a hybrid penalty in this applicative problem, as it brings flexibility in the regularization model. However, it is at the price of a tedious grid search. Second, from a general glance, we can notice that optimization-based methods are outperformed by most learning-based ones. This is because learning-based competitors perform a supervised data-driven tuning of its hyperparameters to optimize the quality metric (here, MSE, which is directly related to SNR). In particular, our proposed approach **U-HQ** yields better restoration quality compared to the use of a standard iterative HQ algorithm **HQ-SC**, and to its early stopped variant **HQ-ES** (up to 3dB improvement in both SNR/TSNR).

*Computational complexity and stability of **U-HQ**.* The interpretation of HQ algorithm as layers of a feed-forward neural network has allowed us to implement it efficiently on the GPU-based Pytorch environment, which considerably reduces its execution time, when compared to a more standard CPU implementation. Since it requires, in average, a lower number of layers/iterations than its optimization-based counterpart, **U-HQ** benefits from a reduced execution time compared to **HQ-SC**, as illustrated in Table 3. For the sake of comparison, we also display the test time of the second best performing method, **U-HQ-FixN** (i.e., **U-HQ** without noise estimation layer). One can notice that this architecture has similar test time than **U-HQ**, which shows that introducing data-driven fully connected layers (see Fig. 7) does not increase the network complexity. On top of its good qualitative and computational performance, the training stability of **U-HQ**, and the absence of overfitting phenomenon can be assessed on the training loss curves on Dataset 1 given by Fig. 5. An example of restored signal in Dataset 1 is displayed in Fig. 6. We refer the reader to Section S5 of our supplementary file for similar plots on Datasets 2 and 3.

*Ablation study on **U-HQ**.* Third, the proposed architecture **U-HQ** appears superior to all its simplified variants, namely **U-HQ-DE**, **U-HQ-FixS**, **U-HQ-FixN**, and **U-HQ-FixN-OverP**. This illustrates the importance of the learnable stepsize and hyper-parameter sequences, as well as the benefits of overparametrization. A visual representation of learnt parameters is given in Fig. 7 for Dataset 1 (see Section S5 of our supplementary file for the other datasets). Interestingly, several regimes can be observed, namely boosting phases with higher regularization parameters and higher stepsizes, and stabilization phases where all parameters stabilize to constant values.

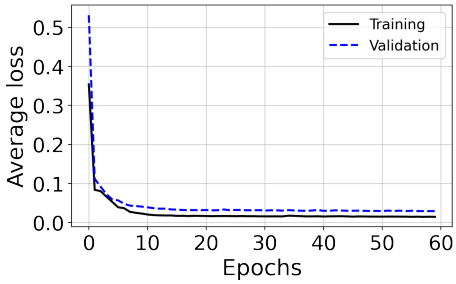

Figure 5: Evolution of training loss (solid black) and validation loss (dashed blue) along epochs for **U-HQ** for Dataset 1.

	Dataset 1	Dataset 2	Dataset 3
HQ-SC			
Iterations (averaged)	6.38	8.33	10.4
CPU time (s)	5.47	7.26	8.47
GPU time (s)	0.29	0.37	0.47
U-HQ-FixN			
Number of layers	8	8	8
GPU time (s)	0.17	0.17	0.17
U-HQ-FixN-OverP			
Number of layers	8	8	8
GPU time (s)	0.17	0.17	0.17
U-HQ			
Number of layers	8	8	8
GPU time (s)	0.17	0.17	0.17
FCNet			
GPU time (s)	3.42×10^{-4}	3.43×10^{-4}	3.28×10^{-4}
AE			
GPU time (s)	6.53×10^{-4}	7.89×10^{-4}	6.67×10^{-4}
ResUNet			
GPU time (s)	4.72×10^{-3}	4.55×10^{-3}	4.36×10^{-3}

Table 3: Averaged iteration/layer numbers and execution time on GPU per signal, for inference on test set on Dataset 1 to Dataset 3, using **HQ-SC**,**U-HQ-FixN**, **U-HQ-FixN-OverP**, **U-HQ** and DL methods (**FCNet**, **AE**, and **ResUNet**, respectively). For the former, we indicate also CPU times, for the sake of comparison.

Finally, it can be seen that, like for optimization-based methods, employing both penalty branches in the architecture of **U-HQ** has clear benefit on the performance when compared with the use of only the convex or non-convex term. The choice of good penalty terms is crucial to get high performance. A large class of alternative penalties, such as the Tikhonov-based penalty used in NMR spectroscopy in [12] for instance, can be easily encompassed in our framework. However, as our datasets are made of sparse signals, such penalty

is not well adapted, and considerably degrades the quality of the results, as it can be seen in the table. The aforementioned observations confirm our choices for the final **U-HQ** architecture with a hybrid convex/non-convex penalty model.

U-HQ *vs state-of-the-art benchmarks.* The proposed **U-HQ** appears superior, in terms of both qualitative metrics, with respect to its competitors **U-PD** and **U-ISTA**, which again shows the benefit of considering a flexible and learnable regularization strategy in this applicative context. Furthermore, as shown in the bottom lines of Table 4, the deep learning models **FCNet** and **AE** perform very poorly on this task. Our approach displays consistently better restoration scores compared to with **ResUNet**. In terms of inference time, DL methods are however faster than the other competitors. The main reason is that **U-HQ** and its variants, require at each layer (i.e., each iteration in its iterative form), the inversion of an $n \times n$ matrix to build the operator W_k.

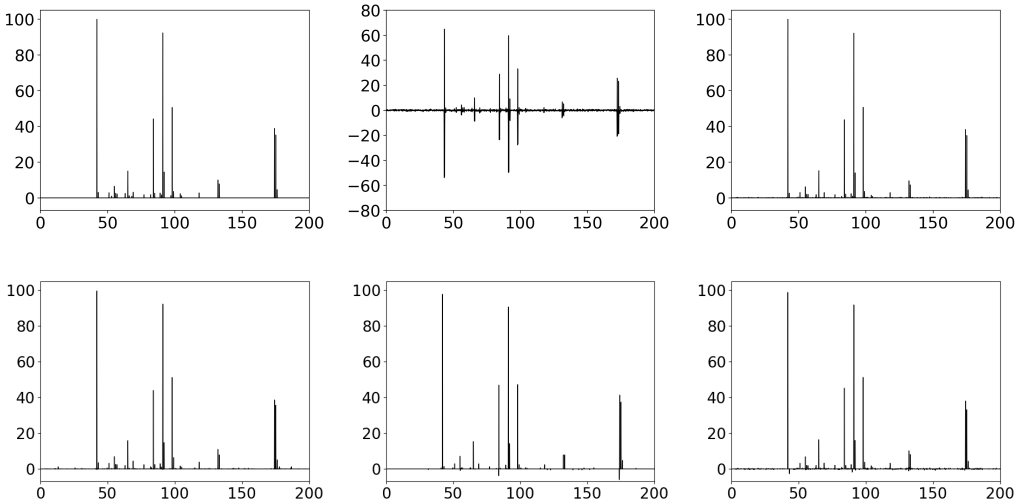

Figure 6: Groundtruth signal $\overline{x}$ (top left), degraded observation y (top middle) and reconstruction $\widehat{x}$ using proposed method **U-HQ** (top right), **ResUNet** (bottom left), **U-ISTA** (bottom middle) and **U-PD** (bottom right) for one example from the test set of Dataset 1. x-axis indicates the mass-to-charge (m/z) ratio of chemical compounds and y-axis the abundance of the corresponding compound. SNR/TSNR scores of restored signal (in dB) are 31.75/34.96, 29.97/31.84, 23.08/25.50 and 26.13/29.50 for **U-HQ**, **ResUNet**, **U-ISTA**, and **U-PD**, respectively.

Penalty	Dataset 1	Dataset 2	Dataset 3
HQ-SC			
Convex	28.28 (7.37)/34.48 (7.18)	19.93 (3.79)/25.36 (5.27)	26.20 (5.49)/30.85 (5.01)
Non-convex	27.99 (6.26)/36.39 (7.36)	22.13 (4.12)/28.07 (5.62)	30.45 (5.04)/31.20 (5.02)
Hybrid	28.77 (6.49)/35.96 (7.22)	22.58 (4.13)/27.85 (5.62)	30.41 (4.96)/31.07 (4.95)
HQ-ES			
Convex	28.23 (7.79)/34.52 (7.79)	18.86 (3.64)/25.89 (5.35)	25.40 (5.81)/30.03 (6.00)
Non-convex	28.04 (6.33)/36.56 (7.55)	22.16 (4.10)/28.07 (5.77)	30.18 (5.22)/30.78 (5.36)
Hybrid	28.27 (6.47)/36.48 (7.58)	22.20 (4.12)/28.05 (5.78)	30.18 (5.22)/30.78 (5.36)
U-ISTA			
Convex	20.55 (6.89)/21.25 (6.60)	21.62 (5.85)/23.03 (5.83)	17.03 (6.39)/17.92 (5.72)
U-PD			
Convex	22.13 (6.76)/25.16 (6.42)	21.55 (4.87)/22.88 (4.80)	24.42 (4.60)/26.21 (4.61)
U-HQ-DE			
Hybrid	28.16 (5.57)/34.43 (6.34)	22.26 (3.46)/27.95 (27.95)	31.66 (5.85)/33.04 (5.63)
U-HQ-FixS			
Convex	28.27 (6.41)/34.26 (6.78)	20.64 (3.57)/26.66 (4.72)	26.72 (5.96)/30.25 (5.97)
Non-convex	29.44 (4.98)/32.96 (5.62)	24.34 (3.47)/26.67 (4.29)	32.22 (6.76)/34.43 (6.12)
Hybrid	29.44 (4.98)/32.99 (5.62)	24.50 (3.52)/26.40 (4.31)	32.34 (5.60)/33.41 (5.33)
U-HQ-FixN			
Convex	28.88 (6.70)/36.27 (7.74)	22.56 (3.47)/26.10(4.26)	26.51 (6.39)/31.84 (6.25)
Non-convex	29.89 (5.03)/33.19 (5.65)	25.08 (3.81)/26.46 (4.25)	32.17 (5.88)/33.53 (5.54)
Hybrid	30.07 (4.86)/32.78 (5.40)	25.22 (3.91)/26.57 (4.34)	32.36 (5.73)/33.65 (5.41)
U-HQ-FixN-OverP			
Convex	30.26 (7.23)/36.13 (7.47)	24.52 (3.70)/27.08 (4.41)	31.07 (5.53)/33.22 (5.30)
Non-convex	29.96 (5.24)/33.44 (5.78)	25.03 (3.85)/26.25 (4.23)	32.17 (5.22)/32.88 (4.97)
Hybrid	30.38 (4.88)/32.30 (5.28)	25.18 (3.87)/26.52 (4.27)	32.48 (5.41)/33.29 (5.11)
U-HQ			
Tikhonov	7.24 (3.86)/15.64 (7.97)	6.93 (3.27)/15.56 (7.89)	2.14 (1.01)/4.34 (3.30)
Convex	28.83 (5.40)/33.75 (6.06)	19.98 (3.14)/25.36 (4.94)	25.42 (5.65)/29.87 (5.19)
Non-convex	31.13 (5.37)/32.89 (5.26)	24.27 (3.19)/27.17 (4.67)	32.19 (5.67)/33.91 (5.79)
Hybrid	**31.56 (5.37)/34.63 (5.26)**	**25.27 (3.57)/27.04 (4.65)**	**33.75 (7.28)/35.87 (7.65)**
DL			
FCNet	1.97 (1.83)/2.29 (2.11)	1.90 (1.93)/2.17 (2.26)	1.83 (1.71)/2.11 (1.91)
AE	0.32 (0.43)/0.49 (0.56)	0.31 (0.45) /0.46 (0.59)	0.35 (0.45)/0.50 (0.55)
ResUNet	29.97 (6.13)/31.84 (6.36)	25.05 (5.28)/ 26.23 (5.60)	28.67 (3.67)/29.83 (3.71)

Table 4: Summary of results in terms of mean (std) of SNR/TSNR metrics, in dB, computed on test sets (see acronyms definitions in Table 2). Best scores are highlighted in bold. Second best scores are underlined.

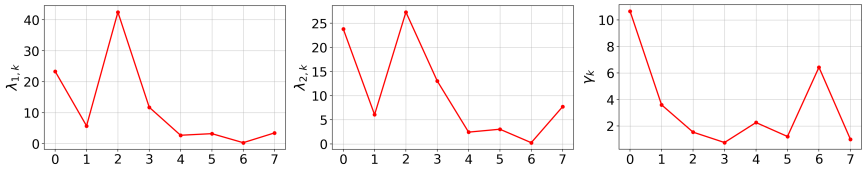

Figure 7: Learnt hyperparameter sequences, averaged on test set, $(\lambda_{1,k})_{0\leq k\leq K-1}$ (left), $(\lambda_{2,k})_{0\leq k\leq K-1}$ (middle) and $(\gamma_k)_{0\leq k\leq K-1}$ (right) with respect to layer index k (x-axis) for the proposed method **U-HQ**, for Dataset 1.

4.5. Discussion

Let us now take a step back, to discuss the strengths of the proposed unrolled architecture **U-HQ**, and the remaining ways for improvement. A key feature of unrolled architectures is that they are model-based. Specifically, the proposed approach **U-HQ** explicitly integrates the knowledge of the degradation operator H and of the statistical noise model, as well as the prior knowledge on the sought signals. This is clearly beneficial, as it brings interpretability, flexibility, and good performance to our model, compared to the DL methods. **U-HQ** is able to perform consistently well, even though our datasets are very challenging, with varying shapes for H, varying signal sparsity ratios, and varying noise levels, and a relatively low number of training samples. First, an avenue for future research is to go one step further, by considering an even more flexible architecture, for instance able to learn the degradation operator H (i.e., blind deconvolution context), or to cope with non-Gaussian noise. This would certainly require complexifying the problem formulation and the associated iterative algorithm method, for instance as in [86], and larger datasets for training. Second, as illustrated in the experiments, the performance of **U-HQ** depend on the choice for the potential functions retained, to build the activations R_1, R_2, as well as the linear block W_k. The hybrid strategy allows to combine optimally two classes of penalties, and thus a wide class of priors on the sought signal. In a future work, we plan to investigate a higher number of penalty branches, with the aim to encompass a broader class of datasets encountereed in other applications of signal processing (e.g., seismic data [59]). Third, as discussed in the introduction, the unrolling paradigm paves the way for a thorough stability analysis, using advanced tools from fixed point analysis [70]. Such analysis

remains an open question for **U-HQ**, and would be an interesting future work perspective.

5. Conclusion

In this work, we propose a novel approach to solve sparse recovery problems, motivated by a class of inverse problems arising in chemistry. The proposed method relies on merging a model-based half-quadratic iterative method and a data driven supervised approach, by adopting the unrolling paradigm. Unrolling the half-quadratic algorithm leads to an original interpretable deep network architecture. Leveraging on automatic differentiation allows us to efficiently tune the algorithm hyperparameters. Furthermore, the resulting architecture turns out to be efficient in terms of reconstruction quality and computational budget, both of which are important features in any experimental setting. Our simulations illustrate the capability of our unrolling technique on various realistic databases of mass spectrometry spectra degraded by different blur kernels and noise levels.

Acknowledgments

This work has been supported by the ITN-ETN project TraDE-OPT funded by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 861137 and by the European Research Council Starting Grant MAJORIS ERC-2019-STG850925.

References

[1] L. Heaney, D. Jones, T. Suzuki, Mass spectrometry in medicine: a technology for the future?, Future Science OA 3 (Jun 2017).

[2] R. R. Ernst, G. Bodenhausen, A. Wokaun, Principles of Nuclear Magnetic Resonance in One and Two Dimensions, 2nd ed., ser. International Series of Monographs on Chemistry. Oxford, NY: Oxford University Press, 19, 1987.

[3] A. Cherni, E. Chouzenoux, M.-A. Delsuc, PALMA, an improved algorithm for DOSY signal processing, Analyst 142 (5) (2016) 772–779.

[4] E. Chouzenoux, S. Moussaoui, J. Idier, F. Mariette, Optimization of a maximum entropy criterion for 2D Nuclear Magnetic Resonance reconstruction, in: Proceedings of the 35th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2010), 2010, pp. 4154–4157.

[5] A. Cherni, E. Chouzenoux, L. Duval, J. C. Pesquet, SPOQ ℓ_p-over-ℓ_q regularization for sparse signal recovery applied to mass spectrometry, IEEE Transactions on Signal Processing 68 (2020) 6070–6084.

[6] H. Liu, T. Zhang, L. Yan, H. Fang, Y. Chang, A map-based algorithm for spectroscopic semi-blind deconvolution, Analyst 137 (2012) 3862–3873.

[7] E. Lin, V.-V. Telkki, X. Lin, C. Huang, H. Zhan, Y. Yang, Y. Huang, Z. Chen, High-resolution reconstruction for multidimensional Laplace NMR, The Journal of Physical Chemistry Letters 12 (2021) 5085–5090.

[8] A. Mohammad-Djafari, J.-F. Giovannelli, G. Demoment, J. Idier, Regularization, maximum entropy and probabilistic methods in mass spectrometry data processing problems, International Journal of Mass Spectrometry 215 (1) (2002) 175–193.

[9] E. Chouzenoux, S. Moussaoui, J. Idier, F. Mariette, Efficient maximum entropy reconstruction of Nuclear Magnetic Resonance $T_1 - T_2$ spectra, IEEE Transactions on Signal Processing 58 (12) (2010) 6040–6051.

[10] Y.-Q. Song, L. Venkataramanan, M. D. Hürlimann, M. Flaum, P. Frulla, C. Straley, $T_1 - T_2$ correlation spectra obtained using a fast two-dimensional Laplace inversion., Journal of Magnetic Resonance 154 2 (2002) 261–8.

[11] C. Rondeau-Mouro, R. Kovrlija, E. Van Steenberge, S. Moussaoui, Two dimensional IR-FID-CPMG acquisition and adaptation of a maximum entropy reconstruction, Journal of Magnetic Resonance 265 (2016) 16–24.

[12] L. Venkataramanan, Y.-Q. Song, M. Hurlimann, Solving Fredholm integrals of the first kind with tensor product structure in 2 and 2.5 dimensions, IEEE Transactions on Signal Processing 50 (5) (2002) 1017–1026.

[13] X. Qu, Y. Huang, H. Lu, T. Qiu, D. Guo, T. Agback, V. Orekhov, Z. Chen, Accelerated Nuclear Magnetic Resonance spectroscopy with deep learning, Angewandte Chemie (International ed. in English) 59 (26) (2020) 10297—10300.

[14] C. Kim, D. Park, H.-N. Lee, Compressive sensing spectroscopy using a residual convolutional neural network, Sensors 20 (2020) 594.

[15] B. Debus, H. Parastar, P. Harrington, D. Kirsanov, Deep learning in analytical chemistry, TrAC Trends in Analytical Chemistry 145 (2021) 116459.

[16] D. Chen, Z. Wang, D. Guo, V. Orekhov, X. Qu, Review and prospect: deep learning in Nuclear Magnetic Resonance spectroscopy, Chemistry–A European Journal 26 (46) (2020) 10391–10401.

[17] G. Karunanithy, D. F. Hansen, FID-Net: A versatile deep neural network architecture for NMR spectral reconstruction and virtual decoupling, Journal of Biomolecular NMR 75 (2021) 179–191.

[18] D. F. Hansen, Using deep neural networks to reconstruct non-uniformly sampled NMR spectra, Journal of biomolecular NMR 73 (10-11) (2019) 577–585.

[19] K. Wu, J. Luo, Q. Zeng, X. Dong, J. Chen, C. Zhan, Z. Chen, Y. Lin, Improvement in signal-to-noise ratio of liquid-state NMR spectroscopy via a deep neural network DN-Unet, Analytical Chemistry 93 (3) (2020) 1377–1382.

[20] C. C. Horgan, M. Jensen, A. Nagelkerke, J.-P. St-Pierre, T. Vercauteren, M. M. Stevens, M. S. Bergholt, High-Throughput molecular imaging via deep-learning-enabled Raman spectroscopy, Analytical chemistry 93 (48) (2021) 15850–15860.

[21] J. Brandt, K. Mattsson, M. Hassellov, Deep learning for reconstructing low-quality FTIR and Raman spectra a case study in microplastic analyses, Analytical chemistry 93 (49) (2021) 16360–16368.

[22] L. V. Haar, T. Elvira, O. Ochoa, An analysis of explainability methods for convolutional neural networks, Engineering Applications of Artificial Intelligence 117 (2023) 105606.

[23] A. Neacsu, R. Ciubotaru, J.-C. Pesquet, C. Burileanu, Design of robust complex-valued feed-forward neural networks, in: Proceedings of the 30th European Signal Processing Conference (EUSIPCO 2022), 2022, pp. 1596–1600.

[24] K. Gupta, F. Kaakai, B. Pesquet-Popescu, J.-C. Pesquet, F. D. Malliaros, Multivariate lipschitz analysis of the stability of neural networks, Frontiers in Signal Processing 2 (2022).

[25] V. Antun, F. Renna, C. Poon, B. Adcock, A. Hansen, On instabilities of deep learning in image reconstruction and the potential costs of AI, Proceedings of the National Academy of Sciences 117 (2020) 30088–30095.

[26] P. Combettes, J.-C. Pesquet, Lipschitz certificates for layered network structures driven by averaged activation operators, SIAM Journal on Mathematics of Data Science 2 (2020) 529–557.

[27] M. Genzel, J. Macdonald, M. März, Solving inverse problems with deep neural networks – robustness included?, IEEE Transactions on Pattern Analysis and Machine Intelligence 45 (1) (2023) 1119–1134.

[28] J. Hershey, J. Le Roux, F. Weninger, Deep unfolding: Model-based inspiration of novel deep architectures, Tech. rep., https://arxiv.org/abs/1409.2574 (Sep. 2014).

[29] V. Monga, Y. Li, Y. C. Eldar, Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing, IEEE Signal Processing Magazine 38 (2) (2021) 18–44.

[30] C. Bertocchi, E. Chouzenoux, M. Corbineau, J.-C. Pesquet, M. Prato, Deep Unfolding of a Proximal Interior Point Method for Image Restoration, Inverse Problems 36 (3) (Feb. 2020.) 034005.

[31] D. Ito, S. Takabe, T. Wadayama, Trainable ISTA for sparse signal recovery, IEEE Transactions on Signal Processing 67 (12) (2019) 3113–3125.

[32] M. Savanier, E. Chouzenoux, J.-C. Pesquet, C. Riddell, Deep Unfolding of the DBFB Algorithm With Application to ROI CT Imaging With Limited Angular Density, IEEE Transactions on Computational Imaging 9 (2023) 502–516.

[33] J. Sun, Z. Xu, Color image denoising via discriminatively learned iterative shrinkage, IEEE Transactions on Image Processing 24 (2015) 4148–59.

[34] H. T. V. Le, N. Pustelnik, M. Foare, The faster proximal algorithm, the better unfolded deep learning architecture ? the study case of image denoising, in: Proceedings of the 30th European Signal Processing Conference (EUSIPCO 2022), 2022, pp. 947–951.

[35] Y. Sun, P. Babu, D. Palomar, Majorization-Minimization algorithms in signal processing, communications, and machine learning, IEEE Transactions on Signal Processing 65 (2017) 794–816.

[36] P. Charbonnier, L. Blanc-Feraud, G. Aubert, M. Barlaud, Deterministic edge-preserving regularization in computed imaging, IEEE Transactions on Image Processing 6 (2) (1997) 298–311.

[37] D. Geman, C. Yang, Nonlinear image recovery with half-quadratic regularization, IEEE Transactions on Image Processing 4 (7) (1995) 932–946.

[38] V. Mazet, D. Brie, J. Idier, Baseline spectrum estimation using half-quadratic minimization, in: Proceedings of the 12th European Signal Processing Conference (EUSIPCO 2004), 2004, pp. 305–308.

[39] J. Liu, J. Liu, J. Sun, X. Huang, G. Li, B. Liu, Goldindec: A novel algorithm for Raman spectrum baseline correction, Applied Spectroscopy 69 (7) (2015).

[40] E. Chouzenoux, J. Idier, S. Moussaoui, A majorize–minimize strategy for subspace optimization applied to image restoration, IEEE Transactions on Image Processing 20 (6) (2011) 1517–1528.

[41] E. Chouzenoux, A. Jezierska, J.-C. Pesquet, H. Talbot, A Majorize-Minimize subspace approach for ℓ_2-ℓ_0 image regularization, SIAM Journal on Imaging Sciences 6 (1) (2013) 563–591.

[42] S. Cadoni, E. Chouzenoux, J.-C. Pesquet, C. Chaux, A block parallel Majorize-Minimize memory gradient algorithm, in: Proceedings of the 23rd IEEE International Conference on Image Processing (ICIP 2016), 2016, pp. 3194–3198.

[43] M. Chalvidal, E. Chouzenoux, Block distributed 3MG algorithm and its application to 3D image restoration, in: Proceedings of the 27th IEEE International Conference on Image Processing (ICIP 2020), 2020, pp. 938–942.

[44] M. Hong, M. Razaviyayn, Z.-Q. Luo, J.-S. Pang, A unified algorithmic framework for block-structured optimization involving big data: With applications in machine learning and signal processing, IEEE Signal Processing Magazine 33 (1) (2016) 57–77.

[45] A. Breloy, Y. Sun, P. Babu, D. Palomar, Block Majorization-Minimization algorithms for low-rank clutter subspace estimation, in: Proceedings of the 24th European Signal Processing Conference (EUSIPCO 2016), 2016, pp. 2186–2190.

[46] J. Bobin, J. Xu, A. de Vismes Ott, C. Bobin, Learning to unmix from Poisson measurements with application to γ-spectroscopy, in: Proceedings of the Signal Processing with Adaptive Sparse structured Representations workshop (SPARS 2019), Toulouse, France, 2019.

[47] S. Ahmadi, L. Kästner, J. C. Hauffen, P. Jung, M. Ziegler, Photothermal-SR-Net: A customized deep unfolding neural network for photothermal super resolution imaging, IEEE Transactions on Instrumentation and Measurement 71 (2022) 1–9.

[48] Z. Wang, D. Guo, Z. Tu, Y. Huang, Y. Zhou, J. Wang, L. Feng, D. Lin, Y. You, T. Agback, et al., A sparse model-inspired deep thresholding network for exponential signal reconstruction–application in fast biological spectroscopy, IEEE Transactions on Neural Networks and Learning Systems 34 (2022) 7578–7592.

[49] Y. Yang, P. Xiao, B. Liao, N. Deligiannis, A robust deep unfolded network for sparse signal recovery from noisy binary measurements, in: Proceedings of the 28th European Signal Processing Conference (EUSIPCO 2020), 2021, pp. 2060–2064.

[50] S. Li, W. Zhang, Y. Cui, Jointly sparse signal recovery via deep autoencoder and parallel coordinate descent unrolling, in: Proceedings of the 21st IEEE Wireless Communications and Networking Conference (WCNC 2020), 2020, pp. 1–6.

[51] I. A. Huijben, B. S. Veeling, R. J. van Sloun, Learning sampling and model-based signal recovery for compressed sensing MRI, in: Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020), 2020, pp. 8906–8910.

[52] Y. Li, M.-R. Tofighi, V. Monga, Y. C. Eldar, An algorithm unrolling approach to deep image deblurring, Proceedings of the 44th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019) (2019) 7675–7679.

[53] T. Alt, J. Weickert, P. Peter, Translating diffusion, wavelets, and regularisation into residual networks, Tech. rep., https://arxiv.org/abs/2002.02753 (June 2020).

[54] R. A. Willoughby, Solutions of ill-posed problems (A. N. tikhonov and V. Y. arsenin), SIAM Reviews 21 (2) (1979) 266–267.

[55] M. Elad, P. Milanfar, R. Rubinstein, Analysis versus synthesis in signal priors, in: Proceedings of the 14th European Signal Processing Conference (EUSIPCO 2006), 2006, pp. 1–5.

[56] T. Park, G. Casella, The Bayesian Lasso, Journal of the American Statistical Association 103 (482) (2008) 681–686.

[57] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, Journal of the Royal Statistical Society, Series B 67 (2005) 301–320.

[58] A. Guitton, W. W. Symes, Robust inversion of seismic data using the Huber norm, Geophysics 68 (4) (2003) 1310–1319.

[59] A. Repetti, M. Q. Pham, L. Duval, E. Chouzenoux, J. C. Pesquet, Euclid in a Taxicab: Sparse blind deconvolution with smoothed ℓ_1/ℓ_2 regularization, IEEE Signal Processing Letters 22 (5) (2015) 539–543.

[60] E. Soubies, L. Blanc-Féraud, G. Aubert, A continuous exact ℓ_0 penalty (CEL0) for least-squares regularized problem, SIAM Journal on Imaging Science 8 (3) (2015) 1574–1606.

[61] H. Mohimani, M. Babaie-Zadeh, C. Jutten, A fast approach for overcomplete sparse decomposition based on smoothed ℓ_0 norm, IEEE Transactions on Signal Processing 57 (1) (2009) 289–301.

[62] E. Chouzenoux, J.-C. Pesquet, A stochastic Majorize-Minimize subspace algorithm for online penalized least squares estimation, IEEE Transactions on Signal Processing 65 (18) (2017) 4770–4783.

[63] M. Nikolova, M. K. Ng, Analysis of half-quadratic minimization methods for signal and image recovery, SIAM journal on Scientific Computing 27 (2005) 937–966.

[64] Z. Zhang, J. Kwok, D.-Y. Yeung, Surrogate maximization/minimization algorithms and extensions, Machine Learning 69 (2007) 1–33.

[65] D. R. Hunter, K. Lange, A tutorial on MM algorithms, The American Statistician 58 (1) (2004) 30–37.

[66] J. Mairal, Incremental majorization-minimization optimization with application to large-scale machine learning, SIAM Journal on Optimization 25 (02 2014).

[67] Y. Sun, P. Babu, D. Palomar, Majorization-Minimization algorithms in signal processing, communications, and machine learning, IEEE Transactions on Signal Processing 65 (2017) 794–816.

[68] M. Allain, J. Idier, Y. Goussard, On global and local convergence of half-quadratic algorithms, Image Processing, IEEE Transactions on 15 (2006) 1130 – 1142.

[69] P. L. Combettes, J.-C. Pesquet, Deep neural network structures solving variational inequalities, Set-Valued and Variational Analysis 28 (2020) 491–518.

[70] C. de Valle, E. Centofanti, E. Chouzenoux, J.-C. Pesquet, Stability of unfolded forward-backward to perturbations in observed data, in: Proceedings of the 31st European Signal Processing Conference (EUSIPCO 2023), 2023, pp. 865–869.

[71] H. H. Bauschke, P. L. Combettes, Convex Analysis and Monotone Operator Theory in Hilbert Spaces, 2nd Edition, Springer Publishing Company, Incorporated, 2017.

[72] P. L. Combettes, J.-C. Pesquet, Fixed point strategies in data science, IEEE Transactions on Signal Processing 69 (2021) 3878–3905.

[73] J.-P. Vial, Strong and weak convexity of sets and functions, Mathematics of Operations Research 8 (2) (1983) 231–259.

[74] F. Bauer, M. A. Lukas, Comparing parameter choice methods for regularization of ill-posed problems, Mathematics and Computers in Simulation 81 (9) (2011) 1795–1841.

[75] M. D. Zeiler, M. Ranzato, R. Monga, M. Z. Mao, K. Yang, Q. V. Le, P. Nguyen, A. W. Senior, V. Vanhoucke, J. Dean, G. E. Hinton, On rectified linear units for speech processing, Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2013) (2013) 3517–3521.

[76] S. Bell-Kligler, A. Shocher, M. Irani, Blind super-resolution kernel estimation using an internal-GAN, in: Proceedings of the Conference on Neural Information Processing Systems 2019 (NEURIPS 2019), Vancouver, Canada, 2019.

[77] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, Tech. rep., https://arxiv.org/abs/1412.6980 (2015).

[78] `https://massbank.eu/MassBank/`.

[79] N. H. Ricker, The form and nature of seismic waves and the structure of seismograms, Geophysics 5 (4) (1940) 348–366.

[80] A. Felinger, Data analysis and signal processing in chromatography, Elsevier, 1998.

[81] D. Gilton, G. Ongie, R. Willett, Deep equilibrium architectures for inverse problems in imaging, IEEE Transactions on Computational Imaging 7 (2021) 1123–1133.

[82] M. Gharbi, S. Villa, E. Chouzenoux, J.-C. Pesquet, Unrolled primal-dual deep network for sparse signal restoration, Tech. rep., https://hal.inria.fr/hal-03988686 (2022).

[83] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm with application to wavelet-based image deblurring, in: Proceedings of the 34th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009), IEEE, 2009, pp. 693–696.

[84] A. Chambolle, T. Pock, A first-order primal-dual algorithm for convex problems with applications to imaging, Journal of Mathematical Imaging and Vision 40 (2011) 120–145.

[85] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: Proceedings of 18th International Conference Medical Image Computing and Computer Assisted Intervention (MICCAI 2015), Springer, 2015, pp. 234–241.

[86] Y. Huang, E. Chouzenoux, J.-C. Pesquet, Unrolled variational Bayesian algorithm for image blind deconvolution, IEEE Transactions on Image Processing 32 (2022) 430–445.

An Unrolled Half-Quadratic Approach for Sparse Signal Recovery in Spectroscopy (supplementary)

Mouna Gharbi[a], Emilie Chouzenoux[a], Jean-Christophe Pesquet[a]

[a] *CVN, CentraleSupélec, Inria Saclay, University Paris Saclay, Gif-sur-Yvette, France*

Abstract

This supplementary file is organized as follows. In Section S1, we present our proof for Proposition 1. In Section S2, we provide the calculation details for α-averaging constants of the activation functions ϱ of Table 1. Then, in Section S3, we derive the expression of the convex function φ which was introduced in Proposition 1. We finally provide our training settings in Section S4 and display in Section S5 additional figures, complementing the Section 4 of the paper.

S1. Proof of Proposition 1

- If ϱ is 1-Lipschitz then there exists $\alpha \in [\frac{1}{2}, 1]$ and a 1-Lipschitz function ϑ such that
$$(\forall x \in \mathbb{R}) \quad \varrho(x) = (1-\alpha)x + \alpha\vartheta(x). \tag{E1}$$
(In the worst case, one can simply choose $\alpha = 1$.) A function ϱ satisfying (E1) is said to be α-averaged. The existence of $\varphi \in \Gamma_0(\mathbb{R})$ such that (17) holds then follows from [R2, Proposition 3.1(ii)].

  Since ψ is even by assumption, its derivative ϱ is odd, and we deduce from (17) that prox_φ is odd too. For every $p \in \mathbb{R}$, let $u \in \partial\varphi(p)$ and let $x = u + p$. Then
$$\begin{aligned} & x - p \in \partial\varphi(p) \\ \Leftrightarrow \quad & p = \mathrm{prox}_\varphi(x) \\ \Leftrightarrow \quad & -p = \mathrm{prox}_\varphi(-x) \\ \Leftrightarrow \quad & -u = -x + p \in \partial\varphi(-p), \end{aligned} \tag{E2}$$

where $\partial\varphi$ denotes the subdifferential of φ (see [R3]). This shows that $\partial\varphi$ is odd.

According to [R3, Proposition 16.17],

$$\varnothing \neq \operatorname{int}(\operatorname{dom}\varphi) \subset \operatorname{dom}\partial\varphi, \tag{E3}$$

where

$$\operatorname{dom}\varphi = \{x \in \mathbb{R} \mid \varphi(x) < +\infty\} \tag{E4}$$

$$\operatorname{dom}\partial\varphi = \{x \in \mathbb{R} \mid \partial\varphi(x) \neq \varnothing\}, \tag{E5}$$

and $\operatorname{int}(S)$ denotes the interior of a set S. There thus exists $\overline{p} \in \mathbb{R}$ and $\epsilon > 0$ such that

$$\begin{aligned} \overline{p} \in \operatorname{int}(\operatorname{dom}\varphi) &\subset \operatorname{dom}\partial\varphi \\ &= \operatorname{dom}\partial\check{\varphi} && \text{(E6)} \\ &\subset \operatorname{dom}\check{\varphi}, && \text{(E7)} \end{aligned}$$

where $\check{\varphi}\colon x \mapsto \varphi(-x)$. Equality in (E6) follows from the fact that, for every $p \in \mathbb{R}$, $\partial\check{\varphi}(p) = -\partial\varphi(-p) = \partial\varphi(p)$ since $\partial\varphi$ is odd. Then, we deduce from (E7) that $\overline{p} \in \operatorname{int}(\operatorname{dom}\varphi) \cap \operatorname{dom}\check{\varphi}$.

Let us consider the even part φ_{e} of φ defined as

$$\varphi_{\mathrm{e}} = \frac{1}{2}(\varphi + \check{\varphi}). \tag{E8}$$

Since we have shown that $\operatorname{int}(\operatorname{dom}\varphi) \cap \operatorname{dom}\check{\varphi}$ is nonempty, it follows from [R3, Theorem 16.47] that

$$\partial\varphi_{\mathrm{e}} = \frac{1}{2}\big(\partial\varphi + \partial\check{\varphi}\big) = \partial\varphi. \tag{E9}$$

As φ_{e} and φ have the same subdifferential, $\operatorname{prox}_{\varphi_{\mathrm{e}}} = \operatorname{prox}_{\varphi}$. This shows that, in (17), we can assume that φ is even.

- It follows from [R3, Corollary 24.5] that $\widetilde{\varphi}^*$ is a differentiable function such that

$$\operatorname{prox}_{\varphi} = (\widetilde{\varphi}^*)' \tag{E10}$$

where $(\cdot)'$ denotes the derivative of the function in argument. We deduce from (17) that

$$(\forall x \in \mathbb{R}) \quad \psi'(x) = \varrho(x) = (1-2\alpha)x + 2\alpha(\widetilde{\varphi}^*)'(x). \tag{E11}$$

By integrating, this yields

$$(\forall x \in \mathbb{R}) \quad \psi(x) = (1-2\alpha)\frac{x^2}{2} + 2\alpha\widetilde{\varphi}^*(x) + C, \tag{E12}$$

where $C \in \mathbb{R}$.

- If ψ is convex, its derivative ϱ is increasing. Since ϱ is defined on $\mathbb{R}$, it follows from [R2, Proposition 3.1(iii)] that $\alpha = 1/2$.

S2. α-averaging constants of activation functions

In this section, we present calculation details for α-averaging constants of the activation functions ϱ of Table 1, as defined in Proposition 1. Firmly nonexpansive activations ϱ have an α constant equal to $\frac{1}{2}$. This applies to the first five rows of Table 1. Let us now calculate α for the non-monotonic activation functions ϱ listed in lines $5-10$ of Table 1. It follows from (E1) that $\alpha \in]1/2, 1]$ is the smallest constant such that $\vartheta = \mathrm{Id} + \alpha^{-1}(\varrho - \mathrm{Id})$ is 1-Lipschitz, i.e., it satifies the condition

$$\sup_{t\in\mathbb{R}} |\vartheta'(t)| = 1 \tag{E13}$$

- **Welsch:**

It follows from the expression of Welsch activation function that (E13) is equivalent to

$$\sup_{t\in\mathbb{R}} \left| 1 + \alpha^{-1}\Big(\exp\left(\frac{-t^2}{2\delta^2}\right)\left(1 - \frac{t^2}{\delta^2}\right) - 1\Big) \right| = 1. \tag{E14}$$

The zeros of the derivative of
ϑ' are

$$t = 0, \quad t = \sqrt{3}\delta, \quad \text{and} \quad t = -\sqrt{3}\delta. \tag{E15}$$

Condition (E14) is then equivalent to checking the latter two values, namely

$$\frac{2\exp(-3/2) + 1)}{\alpha} - 1 \leq 1 \tag{E16}$$

The smallest averaging constant is thus $\alpha = \exp(-3/2) + 1/2$.

- **Geman-McClure:**

From the expression of Geman-McClure activation function (E13) reads

$$\sup_{t\in\mathbb{R}}\left|1+\alpha^{-1}\Big(\frac{4\delta^4(2\delta^2-3t^2)}{(2\delta^2+t^2)^3}-1\Big)\right|=1. \tag{E17}$$

The zeros of the derivative of ϑ' are

$$t=0,\quad t=\sqrt{2}\delta,\quad \text{and}\quad t=-\sqrt{2}\delta$$

Thus, the maximum in (E17) is reached for the latter two values, which yield the minimum value of the averaging constant $\alpha = 5/8$.

- **Tukey biweight**:

By using the piecewise form of the activation function, plugging in the expression of ϱ on $[-\sqrt{6}\delta, \sqrt{6}\delta]$, and deriving ϑ, we obtain the equivalent formulation of (E13):

$$\sup_{t\in[-\sqrt{6}\delta,\sqrt{6}\delta]}\left|1+\alpha^{-1}\Big(\left(1-\frac{t^2}{6\delta^2}\right)\left(1-\frac{5t^2}{6\delta^2}\right)-1\Big)\right|=1 \tag{E18}$$

The maximum is reached for $|t| = 3\sqrt{\frac{2}{5}}\delta$ and the averaging constant of Tukey biweight activation function is equal to $\alpha = 9/10$.

- **Cauchy:**

Condition (E13) reads

$$\sup_{t\in\mathbb{R}}\left|1+\alpha^{-1}\Big(\frac{\delta^2(\delta^2-t^2)}{(\delta^2+t^2)^2}-1\Big)\right|=1,$$

the maximum being reached when $|t| = \sqrt{3}\delta$. This results in $\alpha = \frac{9}{16}$.

- **Hyperbolic Tangent:**

Condition (E13) is equivalent to

$$\sup_{t\in\mathbb{R}}\left|1+\alpha^{-1}\Big(\frac{\delta^2\cosh(\frac{t^2}{2\delta^2})-2t^2\sinh(\frac{t^2}{2\delta^2})}{\delta^2\cosh^3(\frac{t^2}{2\delta^2})}-1\Big)\right|=1. \tag{E19}$$

The zeros of the derivative of ϑ' satisfy

$$2t\Big(3t^2\sinh^2(\frac{t^2}{2\delta^2}) \;-\; 3\delta^2\cosh(\frac{t^2}{2\delta^2})\sinh(\frac{t^2}{2\delta^2}) \;-\; t^2\cosh^2(\frac{t^2}{2\delta^2})\Big) \quad = \quad 0.$$

The maximum in (E13) is obtained for the nonzero solutions to this equation. To determine the latter values, a numerical resolution is required. By using Newton-Raphson's method, we obtain numerically that the solutions are $t/\delta \simeq \pm 1.5355$. We deduce that $\alpha \simeq 0.9581$.

S3. Associated convex functions

In this appendix, we derive the expression of the convex function φ which was introduced in Proposition 1.

- **Fair potential**:

By using (20) and (E10), we have

$$(\forall t \in \mathbb{R}) \quad \varrho(t) = (\widetilde{\varphi}^*)'(t) = \frac{\delta t}{|t| + \delta}. \tag{E20}$$

This function can be seen as a scaled version of the Elliot activation function investigated in [R1, Example 2.15]. We have thus

$$(\forall t \in \mathbb{R}) \quad \widetilde{\varphi}^*(t) = \delta(|t| - \delta\log(|t| + \delta)) + \eta \tag{E21}$$

with $\eta \in \mathbb{R}$. Since $\widetilde{\varphi} \in \Gamma_0(\mathbb{R})$,

$$(\forall t \in \mathbb{R}) \quad \widetilde{\varphi}(t) = \widetilde{\varphi}^{**}(t) = \sup_{u\in\mathbb{R}} tu - \widetilde{\varphi}^*(u). \tag{E22}$$

For every $t \in \mathbb{R}$, let us define

$$\begin{aligned}(\forall u \in \mathbb{R}) \quad g(u) &= tu - \widetilde{\varphi}^*(u)\\ &= tu - \delta(|u| - \delta\log(|u| + \delta)) - \eta.\end{aligned} \tag{E23}$$

If $t \geq \delta$, then $g(u) \underset{u\to+\infty}{\longrightarrow} +\infty$, and if $t \leq \delta$ then $g(u) \underset{u\to-\infty}{\longrightarrow} +\infty$. For $t \in]-\delta, \delta[$, g reaches its maximum at u such that

$$t = \frac{\delta u}{|u| + \delta} \quad \Leftrightarrow \quad u = \frac{\delta t}{\delta - |t|}. \tag{E24}$$

For this value of u, by setting $\eta = 2\delta^2 \log \delta$,

$$g(u) = -\delta|t| - \delta^2 \log(\delta - |t|). \tag{E25}$$

Therefore, we deduce from (18) that, for every $t \in \mathbb{R}$,

$$\varphi(t) = \begin{cases} -\delta|t| - \delta^2 \log(\delta - |t|) - \frac{t^2}{2} & \text{if } |t| < \delta \\ +\infty & \text{otherwise} \end{cases} \tag{E26}$$

- **Huber**:

$$\begin{aligned} (\forall t \in \mathbb{R}) \quad \varrho(t) &= \begin{cases} t & \text{if } |t| < \delta \\ \delta \operatorname{sign}(t) & \text{otherwise} \end{cases} \\ &= \operatorname{proj}_{[-\delta,\delta]}(t) \end{aligned} \tag{E27}$$

where proj_S denotes the projection onto a nonempty closed convex set S. Thus

$$(\forall t \in \mathbb{R}) \quad \varphi(t) = \iota_{[-\delta,\delta]}(t) = \begin{cases} 0 & \text{if } t \in [-\delta, \delta] \\ +\infty & \text{otherwise.} \end{cases} \tag{E28}$$

- **Green**:

$$(\forall t \in \mathbb{R}) \quad \varrho(t) = \tanh(t). \tag{E29}$$

This activation function was already studied in [1,Example 2.12] where it was proved that, for every $t \in \mathbb{R}$,

$$\varphi(t) = \begin{cases} \frac{(1+t)\log(1+t)+(1-t)\log(1-t)-t^2}{2} & \text{if } |t| < 1 \\ \log 2 - 1/2 & \text{if } |t| = 1 \\ +\infty & \text{otherwise.} \end{cases} \tag{E30}$$

- **Hyperbolic**:

We will just consider the cases when $\kappa \in \{1, 2\}$ for which closed form expressions can be obtained.

- If $\kappa = 1$, then

$$(\forall t \in \mathbb{R}) \quad \varrho(t) = \frac{t}{\sqrt{1 + \frac{t^2}{\delta^2}}}. \tag{E31}$$

We can thus set

$$(\forall t \in \mathbb{R}) \quad \widetilde{\varphi}^*(t) = \delta^2 \sqrt{1 + \frac{t^2}{\delta^2}}. \tag{E32}$$

By defining g as in (E23), if $t \geq \delta$, then $g(u) \underset{u \to +\infty}{\longrightarrow} +\infty$ and, if $t \leq \delta$, then $g(u) \underset{u \to -\infty}{\longrightarrow} +\infty$. For $t \in]-\delta, \delta[$, the maximum of g is reached at u such that

$$t = \frac{u}{\sqrt{1 + \frac{u^2}{\delta^2}}} \quad \Leftrightarrow \quad u = \frac{t}{\sqrt{1 - \frac{t^2}{\delta^2}}}. \tag{E33}$$

We deduce that, for every $t \in \mathbb{R}$,

$$\varphi(t) = \begin{cases} -\delta^2 \sqrt{1 - \frac{t^2}{\delta^2}} - \frac{t^2}{2} & \text{if } |t| < \delta \\ +\infty & \text{otherwise.} \end{cases} \tag{E34}$$

- If $\kappa = 2$, then ϱ reduces to the identity function and $\varphi = 0$.

S4. Training settings for Section 4

We summarize in the Table T1 the training settings used for the unrolled methods experimented in the paper.

S5. Additional figures for Section 4

We display in Figure F1 the training loss curves on Dataset 2 and 3, for the proposed architecture **U-HQ**. Examples of restored signals in Datasets 2 and 3, are displayed in Figure F3. We provide in Figure F4 a visual representation of learnt parameters by the proposed architecture **U-HQ**, for Datasets 2 and 3.

S6. Settings of DL methods

In this section, we display the **ResUNet** architecture used for the sake of comparison on Fig. F5. A detailed overview of each of this network's blocks is provided in Table. T2. It is important to note that each residual convolutional block has a skip connection parallel to the convolutional branch. Moreover, we display the details of both **AE** and **FCNet** architectures in Table T3. Finally, we provide the DL methods training settings in Table T4.

Penalty	Dataset 1			Dataset 2			Dataset 3		
	K	p	(lr,bst,bsv)	K	p	(lr,bst,bsv)	K	p	(lr,bst,bsv)
U-ISTA									
Convex	16	$2K$	$10^{-2}, 100, 10$	16	$2K$	$10^{-3}, 150, 10$	16	$2K$	$10^{-3}, 150, 10$
U-PD									
Convex	16	$3K$	$10^{-3}, 15, 10$	16	$3K$	$10^{-3}, 15, 10$	16	$3K$	$10^{-3}, 15, 10$
U-HQ-DE									
Hybrid	8	2	$10^{-2}, 15, 10$	8	2	$10^{-1}, 10, 10$	8	2	$10^{-1}, 5, 5$
U-HQ-FixS									
Convex	8	K	$10^{-3}, 15, 10$	8	K	$10^{-2}, 15, 10$	8	K	$10^{-1}, 15, 10$
Non-convex	8	K	$10^{-2}, 15, 10$	8	K	$10^{-2}, 15, 10$	8	K	$10^{-1}, 15, 10$
Hybrid	8	$2K$	$10^{-2}, 15, 10$	8	$2K$	$10^{-1}, 15, 10$	8	$2K$	$10^{-1}, 15, 10$
U-HQ-FixN									
Convex	8	$2K$	$10^{-2}, 15, 10$	8	$2K$	$10^{-2}, 15, 10$	8	$2K$	$10^{-2}, 15, 10$
Non-convex	8	$2K$	$10^{-2}, 15, 10$	8	$2K$	$10^{-2}, 15, 10$	8	$2K$	$10^{-2}, 15, 10$
Hybrid	8	$3K$	$10^{-2}, 15, 10$	8	$3K$	$10^{-2}, 15, 10$	8	$3K$	$10^{-1}, 5, 5$
U-HQ-FixN-OverP									
Convex	8	$2KN$	$10^{-2}, 15, 10$	8	$2KN$	$10^{-2}, 15, 10$	8	$2KN$	$10^{-2}, 15, 10$
Non-convex	8	$2KN$	$10^{-1}, 15, 10$	8	$2KN$	$10^{-2}, 15, 10$	8	$2KN$	$10^{-2}, 15, 10$
Hybrid	8	$3KN$	$10^{-2}, 15, 10$	8	$3KN$	$10^{-2}, 15, 10$	8	$3KN$	$10^{-2}, 15, 10$
U-HQ									
Tikhonov	8	$K(m+N)$	$10^{-5}, 5, 5$	8	$K(m+N)$	$10^{-5}, 5, 5$	8	$K(m+N)$	$10^{-6}, 5, 5$
			$10^{-2}, 5, 5$			$10^{-2}, 5, 5$			$10^{-2}, 5, 5$
Convex	8	$K(m+N)$	$10^{-5}, 5, 5$	8	$K(m+N)$	$10^{-5}, 5, 5$	8	$K(m+N)$	$10^{-6}, 5, 5$
			$10^{-2}, 5, 5$			$10^{-2}, 5, 5$			$10^{-2}, 5, 5$
Non-convex	8	$K(m+N)$	$10^{-5}, 15, 10$	8	$K(m+N)$	$10^{-5}, 15, 10$	8	$K(m+N)$	$10^{-6}, 5, 5$
			$10^{-2}, 15, 10$			$10^{-2}, 15, 10$			$10^{-2}, 5, 5$
Hybrid	8	$2K(m+N)$	$10^{-6}, 15, 10$	8	$2K(m+N)$	$10^{-6}, 15, 10$	8	$2K(m+N)$	$10^{-6}, 5, 5$
			$10^{-2}, 15, 10$			$10^{-2}, 15, 10$			$10^{-2}, 5, 5$

Table T1: Summary of training settings for experimented deep unrolled architectures: number of layers K, number of learnt parameters p, optimizer, learning rate (lr) and training bst and validation bsv batch sizes respectively. For **U-HQ**, we indicate lr_λ (top) and lr_γ (bottom).

References

[R1]. P.L. Combettes and J.-C. Pesquet. Deep neural network structures solving variational inequalities. Set-Valued and Variational Analysis, 28:491-518, 2020.

[R2]. P.L. Combettes and J.-C. Pesquet. Lipschitz Certificates for Layered Network Structures Driven by Averaged Activation Operators. SIAM Journal on Mathematics of Data Science, 2:529-557, Jun. 2020.

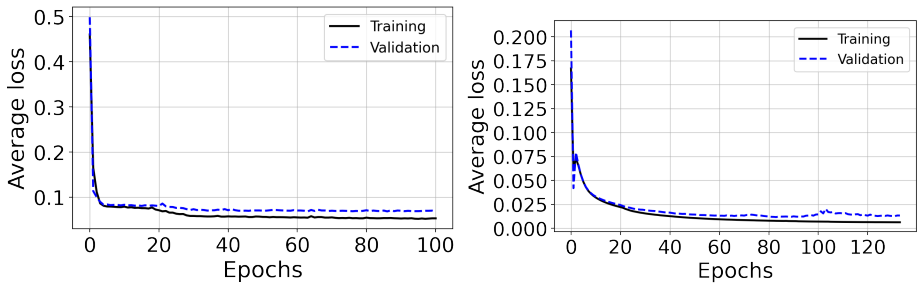

Figure F1: Evolution of training loss (solid black) and validation loss (dashed blue) along epochs for **U-HQ** for Dataset 2 (left) and Dataset 3 (right).

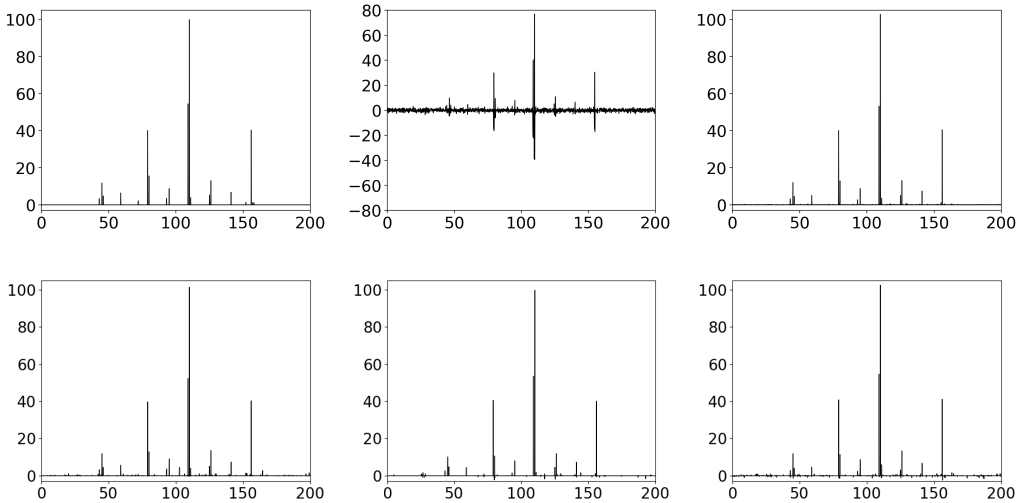

Figure F2: Groundtruth signal $\overline{x}$ (left up), degraded observation y (middle up) and reconstruction $\widehat{x}$ (right up) using proposed method **U-HQ**, **ResUNet** (left bottom), **U-ISTA** (middle bottom) and **U-PD** (right bottom), for an example from Dataset 2. x-axis indicates the mass-to-charge (m/z) ratio of chemical compounds and y-axis the abundance of corresponding compound. SNR/TSNR scores of restored signals (in dB) are 26.26/28.09, 25.05/26.23, 22.92/25.21, and 22.11/25.83 for the respective methods.

[R3]. H. H. Bauschke, P. L. Combettes. Convex Analysis and Monotone Operator Theory in Hilbert Spaces, 2nd Edition, Springer Publishing Company, Incorporated, 2017.

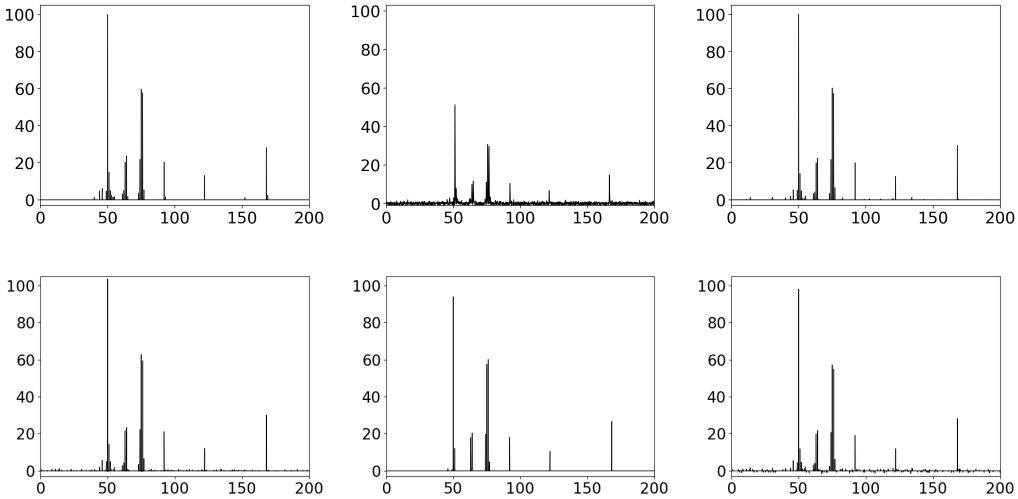

Figure F3: Groundtruth signals $\overline{x}$ (left), degraded observations y (middle) and reconstruction $\widehat{x}$ (right) using proposed method **U-HQ**, **ResUNet** (left bottom), **U-ISTA** (middle bottom) and **U-PD** (right bottom) for an example from Dataset 3. x-axis indicates the mass-to-charge (m/z) ratio of chemical compounds and y-axis the abundance of corresponding compound. SNR/TSNR scores of restored signals (in dB) are 26.85/28.03, 28.67/29.83, 17.17/19.04 and 21.35/25.19 respectively.

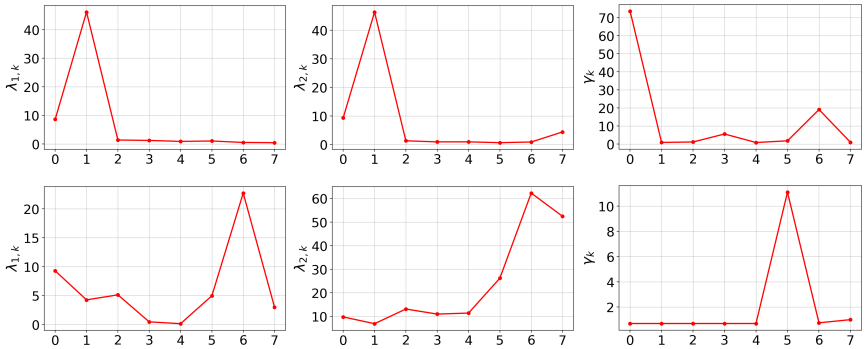

Figure F4: Learnt hyperparameter sequences $(\lambda_{1,k})_{0\leq k\leq K-1}$, $(\lambda_{2,k})_{0\leq k\leq K-1}$ and $(\gamma_k)_{0\leq k\leq K-1}$ with respect to layer index k (x-axis) for the proposed method **U-HQ**, for Dataset 2 (top) and Dataset 3 (bottom). Learnt parameters are averaged on test set.

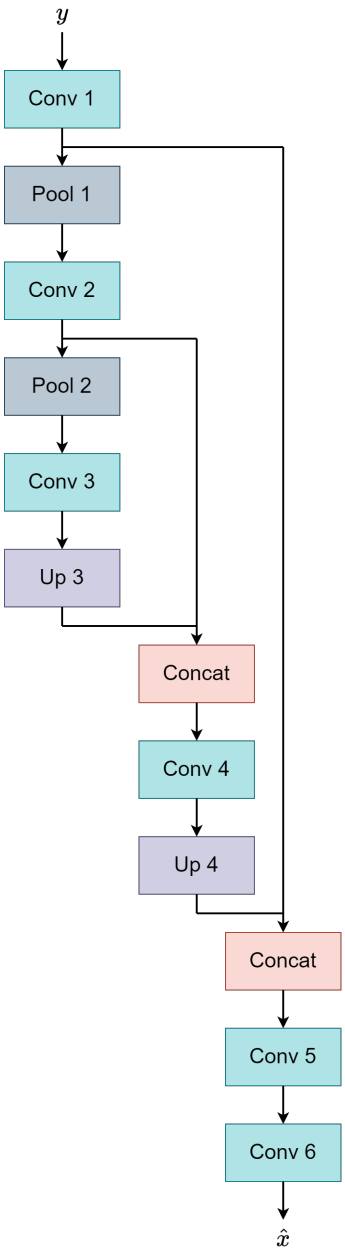

Figure F5: **ResUNet** architecture. Borders of the input signal y are truncated, to reach an input size of $n = 2000$.

Layer Name	Function	Details
Conv 1	Basic Convolution	$1 \times \begin{cases} \text{1D Conv}(1, 64, 3, 1, 1)^1 \\ \text{PReLU} \end{cases}$
	Residual Convolution	$3 \times \begin{cases} \text{1D Conv}(64, 64, 3, 1, 1) \\ \text{PReLU} \end{cases}$
Pool 1	Maxpooling	Maxpool1D(2)[2]
Conv 2	Basic Convolution	$1 \times \begin{cases} \text{1D Conv}(64, 128, 3, 1, 1) \\ \text{PReLU} \end{cases}$
	Residual Convolution	$3 \times \begin{cases} \text{1D Conv}(128, 128, 3, 1, 1) \\ \text{PReLU} \end{cases}$
Pool 2	Maxpooling	Maxpool1D(2)
Conv 3	Basic Convolution	$1 \times \begin{cases} \text{1D Conv}(128, 256, 3, 1, 1) \\ \text{PReLU} \end{cases}$
	Residual Convolution	$3 \times \begin{cases} \text{1D Conv}(256, 256, 3, 1, 1) \\ \text{PReLU} \end{cases}$
	Basic Convolution	$1 \times \begin{cases} \text{1D Conv}(256, 128, 3, 1, 1) \\ \text{PReLU} \end{cases}$
Up 3	Upsampling	UpSample(2)[3]
Conv 4	Basic Convolution	$1 \times \begin{cases} \text{1D Conv}(256, 128, 3, 1, 1) \\ \text{PReLU} \end{cases}$
	Residual Convolution	$3 \times \begin{cases} \text{1D Conv}(128, 128, 3, 1, 1) \\ \text{PReLU} \end{cases}$
	Basic Convolution	$1 \times \begin{cases} \text{1D Conv}(128, 64, 3, 1, 1) \\ \text{PReLU} \end{cases}$
Up 4	Upsampling	UpSample(2)
Conv 5	Basic Convolution	$1 \times \begin{cases} \text{1D Conv}(128, 64, 3, 1, 1) \\ \text{PReLU} \end{cases}$
	Residual Convolution	$3 \times \begin{cases} \text{1D Conv}(64, 64, 3, 1, 1) \\ \text{PReLU} \end{cases}$
Conv 6	Basic Convolution	$1 \times \begin{cases} \text{1D Conv}(64, 1, 3, 1, 1) \\ \text{PReLU} \end{cases}$

[1] (in-channels,out-channels,kernel-size,stride,padding)
[2] (kernel-size)
[3] (scale-factor)

Table T2: ResUNet Description.

Method	Architecture Description
FCNet	Layer 1 : $\begin{cases} \text{Fully Connected}^{1}(m,n) \\ \text{ReLU} \end{cases}$ Layer 2 : $\begin{cases} \text{Fully Connected}(n,n) \\ \text{ReLU} \end{cases}$ Layer 3 : $\begin{cases} \text{Fully Connected}(n,n) \\ \text{ReLU} \end{cases}$ Layer 4 : $\begin{cases} \text{Fully Connected}(n,n) \\ \text{ReLU} \end{cases}$
AE	Encoder $\begin{cases} \text{Layer1 : } \begin{cases} \text{Fully Connected}(m, m \operatorname{div} 2) \\ \text{ReLU} \end{cases} \\ \text{Layer2 : } \begin{cases} \text{Fully Connected}(m \operatorname{div} 2, m \operatorname{div} 4) \\ \text{ReLU} \end{cases} \\ \text{Layer3 : } \begin{cases} \text{Fully Connected}(m \operatorname{div} 4, m \operatorname{div} 8) \\ \text{ReLU} \end{cases} \\ \text{Layer4 : } \begin{cases} \text{Fully Connected}(m \operatorname{div} 8, m \operatorname{div} 16) \\ \text{ReLU} \end{cases} \end{cases}$ Decoder $\begin{cases} \text{Layer1 : } \begin{cases} \text{Fully Connected}(m \operatorname{div} 16, m \operatorname{div} 8) \\ \text{ReLU} \end{cases} \\ \text{Layer2 : } \begin{cases} \text{Fully Connected}(m \operatorname{div} 8, m \operatorname{div} 4) \\ \text{ReLU} \end{cases} \\ \text{Layer3 : } \begin{cases} \text{Fully Connected}(m \operatorname{div} 4, m \operatorname{div} 2) \\ \text{ReLU} \end{cases} \\ \text{Layer4 : } \begin{cases} \text{Fully Connected}(m \operatorname{div} 2, n) \\ \text{ReLU} \end{cases} \end{cases}$

[1] (in-features,out-features)

Table T3: **FCNet** and **AE** Description. $m = 2049$, $n = 2000$

Number of layer	Number of parameters	lr, decay, bst,bsv
FCNet		
$K = 4$	p $= mn + 3n^2$	lr $= 10^{-4}, 50, 50$
AE		
$K_{\text{enc}} = 4$, $K_{\text{dec}} = 4$	p $= m^2/2 + mn + 21/64m$	lr $= 10^{-4}, 50, 50$
ResUNet		
	p $= 1330264$	lr $= 5 \times 10^{-4}$, d $= 10^{-2}, 50, 50$

Table T4: Summary of training settings for experimented Deep Learning methods. All settings are shared between three datasets. For **FC**, K indicates the number of (linear+non-linear) layers. For **AE**, K_{enc} and K_{dec} indicate, respectively, the number of (linear+non-linear) layers in the encoder and decoder. p, lr, d, bst, bvt are, respectively, the number of parameters, learning rate, weight decay (if used), training batch size, and validation batch size.