Learning Maximally Monotone Operators for Image Recovery

Jean-Christophe Pesquet, Audrey Repetti, Matthieu Terris, Yves Wiaux

LEARNING MAXIMALLY MONOTONE OPERATORS FOR IMAGE RECOVERY*

JEAN-CHRISTOPHE PESQUET†, AUDREY REPETTI‡, MATTHIEU TERRIS§, AND YVES WIAUX¶

Abstract. We introduce a new paradigm for solving regularized variational problems. These are typically formulated to address ill-posed inverse problems encountered in signal and image processing. The objective function is traditionally defined by adding a regularization function to a data fit term, which is subsequently minimized by using iterative optimization algorithms. Recently, several works have proposed to replace the operator related to the regularization by a more sophisticated denoiser. These approaches, known as plug-and-play (PnP) methods, have shown excellent performance. Although it has been noticed that, under some Lipschitz properties on the denoisers, the convergence of the resulting algorithm is guaranteed, little is known about characterizing the asymptotically delivered solution. In the current article, we propose to address this limitation. More specifically, instead of employing a functional regularization, we perform an operator regularization, where a maximally monotone operator (MMO) is learned in a supervised manner. This formulation is flexible as it allows the solution to be characterized through a broad range of variational inequalities, and it includes convex regularizations as special cases. From an algorithmic standpoint, the proposed approach consists in replacing the resolvent of the MMO by a neural network (NN). We present a universal approximation theorem proving that nonexpansive NNs are suitable models for the resolvent of a wide class of MMOs. The proposed approach thus provides a sound theoretical framework for analyzing the asymptotic behavior of first-order PnP algorithms. In addition, we propose a numerical strategy to train NNs corresponding to resolvents of MMOs. We apply our approach to image restoration problems and demonstrate its validity in terms of both convergence and quality.

Key words. Monotone operators, neural networks, convex optimization, plug-and-play methods, inverse problems, computational imaging, nonlinear approximation

AMS subject classifications. 47H05, 90C25, 90C59, 65K10, 49M27, 68T07, 68U10, 94A08.

1. Introduction.

In many problems in data science, in particular when dealing with inverse problems, a variational approach is adopted which amounts to

$$\underset{x\in\mathcal{H}}{\text{minimize}} \;\; f(x)+g(x) \tag{1.1}$$

where $\mathcal{H}$ is the underlying data space, here assumed to be a real Hilbert space, $f\colon \mathcal{H} \to]-\infty,+\infty]$ is a data fit (or data fidelity) term related to some available data z (observations), and $g\colon \mathcal{H} \to]-\infty,+\infty]$ is some regularization function. The data fit term is often derived from statistical considerations on the observation model through the maximum likelihood principle. For many standard noise distributions,

*Submitted to the editors 12/23/2020.

Funding: M. Terris would like to thank Heriot-Watt University for the PhD funding. This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant EP/T028270/1. The work of J.-C. Pesquet was supported by Institut Universitaire de France and the ANR Chair in AI BRIGEABLE.

† Université Paris-Saclay, Inria, Center for Visual Computing, Gif sur Yvette, France (jean-christophe.pesquet@centralesupelec.fr).

‡Second corresponding author. School of Mathematics and Computer Sciences and School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh, UK. Maxwell Institute for Mathematical Sciences, Bayes Centre, Edinburgh, UK (a.repetti@hw.ac.uk).

§First corresponding author. School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh, UK (mt114@hw.ac.uk).

¶School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh, UK (y.wiaux@hw.ac.uk).

the negative log-likelihood corresponds to a smooth function (e.g. Gaussian, Poisson-Gauss, or logistic distributions). The regularization term is often necessary to avoid overfitting or to overcome ill-posedness problems. A vast literature has been developed on the choice of this term. It often tends to promote the smoothness of the solution or to enforce its sparsity by adopting a functional analysis viewpoint. Good examples of such regularization functions are the total variation semi-norm [53] and its various extensions [12, 26], and penalizations based on wavelet (or "x-let") frame representations [27]. Alternatively, a Bayesian approach can be followed where this regularization is viewed as the negative-log of some prior distribution, in which case the minimizer of the objective function in (1.1) can be understood as a Maximum A Posteriori (MAP) estimator. In any case, the choice of this regularization introduces two main roadblocks. First, the function g has to be chosen so that the minimization problem in (1.1) be tractable, which limits its choice to relatively simple forms. Secondly, the definition of this function involves some parameters which need to be set. The simplest case consists of a single scaling parameter usually called the regularization factor, the choice of which is often very sensitive on the quality of the results. Note that, in some works, this regularization function is the indicator function of some set encoding some smoothness or sparsity constraint. For example, it can model an upper bound on some functional of the discrete gradient of the sought signal, this bound playing then a role equivalent to a regularization parameter [19]. Using an indicator function can also model standard constraints in some image restoration problems, where the image values are bounded [1, 10]. In order to solve such problems, a wide range of proximal splitting algorithms have been developed in the past decades [20].

By denoting by $\Gamma_0(\mathcal{H})$ the class of lower-semicontinuous convex functions from $\mathcal{H}$ to $]-\infty,+\infty]$ with a nonempty domain, let us now assume that both f and g belong to $\Gamma_0(\mathcal{H})$. The Moreau subdifferentials of these functions will be denoted by ∂f and ∂g, respectively. Under these convexity assumptions, if

$$0 \in \partial f(x) + \partial g(x), \tag{1.2}$$

then x is a solution to the minimization problem (1.1). Actually, under mild qualification conditions the sets of solutions to (1.1) and (1.2) coincide [7]. By reformulating the original optimization problem under the latter form, we have moved to the field of variational inequalities. Interestingly, it is a well-established fact that the subdifferential of a function in $\Gamma_0(\mathcal{H})$ is a maximally monotone operator (MMO), which means that (1.2) is a special case of the following monotone inclusion problem:

$$\text{Find } x \in \mathcal{H} \text{ such that } 0 \in \partial f(x) + A(x), \tag{1.3}$$

where A is an MMO. We recall that a multivalued operator A defined on $\mathcal{H}$ is maximally monotone if and only if, for every $(x_1, u_1) \in \mathcal{H}^2$,

$$u_1 \in Ax_1 \quad \Leftrightarrow \quad (\forall x_2 \in \mathcal{H})(\forall u_2 \in Ax_2)\langle x_1 - x_2 \mid u_1 - u_2\rangle \geqslant 0, \tag{1.4}$$

see e.g. [7, 23] for a background on monotone operator theory. Actually the class of monotone inclusion problems is much wider than the class of convex optimization problems and, in particular, includes saddle point problems and game theory equilibria [18]. It is also worth noticing that many existing algorithms for solving convex optimization problems are particular cases of algorithms designed for solving monotone inclusion problems. The latter often leverage the resolvent operator of A

in (1.3), which reduces to a proximal operator when $A = \partial g$ as in (1.2) [7]. This suggests that it is more flexible, and probably more efficient, to substitute (1.3) for (1.1) in problems encountered in data science. In other words, instead of performing a functional regularization, we can introduce an operator regularization through the maximally monotone mapping A. Although this extension of (1.1) may appear both natural and elegant, it induces a high degree of freedom in the choice of the regularization strategy. However, if we except the standard case when $A = \partial g$, it is hard to have a good intuition about how to make a relevant choice for A. To circumvent this difficulty, our proposed approach will consist in learning A in a supervised manner by using some available dataset in the targeted application.

Since a MMO is fully characterized by its resolvent, we propose in this paper to learn the resolvent of A. We show that this can be formulated as learning a denoiser with an appropriate 1-Lipschitz condition. Thus, our approach enters into the family of so-called plug-and-play (PnP) methods, where one replaces the proximity operator in an optimization algorithm with a denoiser, e.g. a denoising neural network (NN) [69]. It is worth mentioning that by doing so, any algorithm whose proof is based on MMO theory, e.g., Forward-Backward (FB), Douglas-Rachford, Peaceman-Rachford, primal-dual approaches, and more [7, 20, 25, 37, 60] can be turned into a PnP algorithm. To ensure the convergence of such PnP algorithms, according to fixed point theory (under mild conditions) it is sufficient for the denoiser to be firmly nonexpansive [23]. In this context, (1.3) offers an elegant characterization of the solution.

Unfortunately, most pre-defined denoisers do not satisfy this assumption, and learning a firmly nonexpansive denoiser remains challenging [54, 58]. The main bottleneck is the ability to tightly constrain the Lipschitz constant of a NN. During the last years, several works proposed to control the Lipschitz constant (see e.g. [5, 9, 16, 32, 47, 54, 56, 58, 65]). Nevertheless, only few of them are accurate enough to ensure the convergence of the associated PnP algorithm and often come at the price of strong computational and architectural restrictions (e.g., absence of residual skip connections) [9, 32, 54, 58]. The method proposed in [9] allows a tight control of convolutional layers, but in order to ensure the nonexpansiveness of the resulting architecture, one cannot use residual skip connections, despite their wide use in NNs for denoising applications. In [32], the authors propose to train an averaged NN by projecting the full convolutional layers on the Stiefel manifold and showcase the usage of their network in a PnP algorithm. Yet, the architecture proposed by the authors remains constrained by proximal calculus rules. In our previous work [58], we proposed a method to build firmly nonexpansive convolutional NNs; to the best of our knowledge, this was the first method ensuring the firm nonexpansiveness of a denoising NN. However, the resulting architecture was strongly constrained and did not improve over the state-of-the-art. Since building firmly nonexpansive denoisers is difficult, many works on PnP methods leverage ADMM algorithm which may appear easier to handle in practice [54]. However, the convergence of ADMM requires restrictive conditions on the involved linear operators [37].

Alternative approaches have been proposed in the literature to circumvent these difficulties. Ryu et al. [54] proposed the first convergent NN-based PnP algorithm in a more general framework, namely RealSN. Nevertheless, in this work, the authors rely on [54, Assumption A], which may potentially lead to expansive networks (see [54, Lemma 9 & 10]), and unstable PnP algorithms. The regularization by denoising (RED) approach [3, 17] investigates situations when the PnP algorithm converges to a solution to a minimization problem. In particular, the minimum mean square

error (MMSE) denoiser can be employed [3, 64]. However, as underlined by the authors, the denoising NN is only an approximation to the MMSE regressor. The authors of [17] proposed RED-PRO, an extension of RED within the framework of fixed point theory. Under a demi-contractivity assumption, the authors show that the PnP algorithm converges to a solution to a constrained minimization problem.

The contribution of this paper is twofold. We first provide a universal approximation theorem for a wide class of monotone operators. In particular, we show that one can approximate the resolvent of a *stationary* MMO as closely as desired with a firmly nonexpansive NN. We then propose a practical framework to impose the firm nonexpansiveness of our NN. To do so, we regularize the training loss of the NN with the spectral norm of the Jacobian of a suitable nonlinear mapping. Although the resulting NN could be plugged into a variety of iterative algorithms, our work is focused on the standard FB algorithm. We illustrate the convergence of the corresponding PnP scheme in an image restoration problem. We show that our method compares positively in terms of quality to both state-of-the-art PnP methods and regularized optimization approaches.

This article is organized as follows. In section 2, we recall how MMOs can be mathematically characterized and explain how their resolvent can be modeled by an averaged residual neural network. We also establish that NNs are generic models for a wide class of MMOs. In section 3, we show the usefulness of learning MMOs in the context of plug-and-play (PnP) first-order algorithms employed for solving inverse problems. We also describe the training approach which has been adopted. In section 4, we provide illustrative results for the restoration of monochromatic and color images. Finally, some concluding remarks are made in section 5.

Notation: Throughout the article, we will denote by $\|\cdot\|$ the norm endowing any real Hilbert space $\mathcal{H}$. The same notation (being clear from the context) will be used to denote the norm of a bounded linear operator L from $\mathcal{H}$ to some real Hilbert space $\mathcal{G}$, that is $\|L\| = \sup_{x\in\mathcal{H}\setminus\{0\}} \|Lx\|/\|x\|$. The inner product of $\mathcal{H}$ associated to $\|\cdot\|$ will be denoted by $\langle\cdot \mid \cdot\rangle$, here again without making explicit the associated space. Let D be a subset of $\mathcal{H}$ and $T\colon D \to \mathcal{H}$. The operator T is *μ-Lipschitzian* for $\mu > 0$ if, for every $(x, y) \in D^2$, $\|Tx - Ty\| \leqslant \mu\|x - y\|$. If T is 1-Lipschitzian, its is said to be *nonexpansive*. The operator T is *firmly nonexpansive* if, for every $(x, y) \in D^2$, $\|Tx - Ty\|^2 \leqslant \langle x - y \mid Tx - Ty\rangle$. We denote by $2^{\mathcal{H}}$ be the power set of $\mathcal{H}$, that is the set of all subsets of $\mathcal{H}$. Let $A\colon \mathcal{H} \to 2^{\mathcal{H}}$ be a multivariate operator, i.e., for every $x \in \mathcal{H}$, $A(x)$ is a subset of $\mathcal{H}$. The *graph* of A is defined as $\operatorname{gra} A = \{(x, u) \in \mathcal{H}^2 \mid u \in Ax\}$. The operator $A\colon \mathcal{H} \to 2^{\mathcal{H}}$ is *monotone* if, for every $(x, u) \in \operatorname{gra} A$ and $(y, v) \in \operatorname{gra} A$, $\langle x - y \mid u - v\rangle \geqslant 0$, and *maximally-monotone* if (1.4) holds, for every $(x_1, u_1) \in \mathcal{H}^2$. The resolvent of A is $J_A = (\mathrm{Id} + A)^{-1}$, where the inverse is here defined in the sense of the inversion of the graph of the operator. For further details on monotone operator theory, we refer the reader to [7, 23].

2. Neural network models for maximally monotone operators.

2.1. A property of maximally monotone operators.

We first introduce a main property that will link explicitly MMOs to nonexpansive operators, noticing that any multivalued operator on $\mathcal{H}$ is fully characterized by its resolvent.

PROPOSITION 2.1. *Let $A\colon \mathcal{H} \to 2^{\mathcal{H}}$. A is a MMO if and only if there exists a nonexpansive (i.e. 1-Lipschitzian) operator $Q\colon \mathcal{H} \to \mathcal{H}$ such that*

$$J_A\colon \mathcal{H} \to \mathcal{H}\colon x \mapsto \frac{x + Q(x)}{2}, \tag{2.1}$$

that is

$$A = 2(\mathrm{Id} + Q)^{-1} - \mathrm{Id}\,. \tag{2.2}$$

Proof. This result is a direct consequence of Minty's theorem and the fact that any firmly nonexpansive operator can be expressed as the arithmetic mean of the identity operator and some nonexpansive operator Q (see [7, Proposition 23.8]). (2.2) is deduced by inverting (2.1). ∎

The above result means that the class of MMOs can be derived from the class of nonexpansive mappings. The focus should therefore turn on how to model operators in the latter class with neural networks.

2.2. Nonexpansive neural networks. Our objective will next be to derive a parametric model for the nonexpansive operator Q in (2.1). Due to their oustanding approximation capabilities, neural networks appear as good choices for building such models. We will restrict our attention to feedforward NNs.

MODEL 2.2. *Let $(\mathcal{H}_m)_{0\leqslant m\leqslant M}$ be real Hilbert spaces such that $\mathcal{H}_0 = \mathcal{H}_M = \mathcal{H}$. A feedforward NN having M layer and both input and ouput in $\mathcal{H}$ can be seen as a composition of operators:*

$$Q = T_M \cdots T_1, \tag{2.3}$$

where

$$(\forall m \in \{1,\dots,M\}) \quad T_m \colon \mathcal{H}_{m-1} \to \mathcal{H}_m \colon x \mapsto R_m(W_m x + b_m). \tag{2.4}$$

At each layer $m \in \{1,\dots,M\}$, $R_m \colon \mathcal{H}_m \to \mathcal{H}_m$ is a nonlinear activation operator, $W_m \colon \mathcal{H}_{m-1} \to \mathcal{H}_m$ is a bounded linear operator corresponding to the weights of the network, and $b_m \in \mathcal{H}_m$ is a bias parameter vector.

In the remainder, we will use the following notation:

NOTATION 2.3. *Let V and V' be nonempty subsets of some Euclidean space and let $\mathcal{N}_{\mathcal{F}}(V, V')$ denote the class of nonexpansive feedforward NNs with inputs in V and outputs in V' built from a given dictionary $\mathcal{F}$ of allowable activation operators.*

Also, we will make the following assumption:

ASSUMPTION 2.4. *The identity operator as well as the sorting operator performed on blocks of size 2 belong to dictionary $\mathcal{F}$.*

In other words, a network in $\mathcal{N}_{\mathcal{F}}(V, V')$ can be linear, or it can be built by using max-pooling with blocksize 2 and any other kind of activation function, say some given function $\rho \colon \mathbb{R} \to \mathbb{R}$, operating componentwise in some of its layers, provided that the resulting structure is 1-Lipschitzian.

The main difficulty is to design such a feedforward NN so that Q in (2.3) has a Lipschitz constant smaller or equal to 1. An extensive literature has been devoted to the estimation of Lipschitz constants of NNs [5, 55, 57], but the main goal was different from ours since these works were motivated by robustness issues in the presence of adversarial perturbations [29, 38, 50, 57].

In this work, we propose to enforce the nonexpansiveness of Q by adding a regularization during the training of the NN as explained in section 3. Other conditions ensuring the firm nonexpansiveness of Model 2.2 can be found in Appendix A.

2.3. Stationary maximally monotone operators. The one-to-one correspondence between a MMO and a nonexpansive operator from Proposition 2.1 raises the following question: can one approximate the resolvent of a MMO as closely as desired by a firmly nonexpansive neural network structure? The remainder of this work aims to answer this question in the case of a subclass of MMOs, namely those which are stationary, as defined hereunder.

DEFINITION 2.5. *Let $(\mathcal{H}_k)_{1\leqslant k\leqslant K}$ be real Hilbert spaces. An operator A defined on the product space space $\mathcal{H} = \mathcal{H}_1 \times \cdots \times \mathcal{H}_K$ will be said to be a* stationary *MMO if its resolvent J_A is an operator from $\mathcal{H}$ to $\mathcal{H}$ such that, for every $k \in \{1, \ldots, K\}$, there exists a bounded linear operator $\Pi_k \colon \mathcal{H} \to \mathcal{H}_k$ and a self-adjoint nonnegative operator $\Omega_k \colon \mathcal{H} \to \mathcal{H}$ such that*

$$(\forall (x, y) \in \mathcal{H}^2) \quad \|\Pi_k\big(2J_A(x) - x - 2J_A(y) + y\big)\|^2 \leqslant \langle x - y \mid \Omega_k(x - y)\rangle \tag{2.5}$$

with

$$\sum_{k=1}^{K} \Pi_k^* \Pi_k = \mathrm{Id} \tag{2.6}$$

$$\Big\| \sum_{k=1}^{K} \Omega_k \Big\| \leqslant 1. \tag{2.7}$$

Immediate consequences of this definition are given below. In particular, we will see that stationary MMOs define a subclass of the set of MMOs.

PROPOSITION 2.6. *Let $(\mathcal{H}_k)_{1\leqslant k\leqslant K}$ be real Hilbert spaces and let $\mathcal{H} = \mathcal{H}_1 \times \cdots \times \mathcal{H}_K$. Let $A \colon \mathcal{H} \to 2^{\mathcal{H}}$.*

(i) *If A is a stationary MMO on $\mathcal{H}$, then it is maximally monotone.*

(ii) *Assume that (2.6) is satisfied where, for every $k \in \{1, \ldots, K\}$, $\Pi_k \colon \mathcal{H} \to \mathcal{H}_k$ is a bounded linear operator. If $\operatorname{ran}(A + \mathrm{Id}) = \mathcal{H}$ and*

$$(\forall (p, q) \in \mathcal{H}^2)(\forall p' \in A(p))(\forall q' \in A(q)) \ \ \langle \Pi_k(p - q) \mid \Pi_k(p' - q')\rangle \geqslant 0, \tag{2.8}$$

then A is a stationary MMO.

Proof. (i): Let A be a stationary MMO defined on $\mathcal{H}$. Summing over k in (2.5) yields, for every $(x, y) \in \mathcal{H}^2$,

$$\begin{aligned} \Big\langle 2J_A(x) - x - 2J_A(y) + y \mid \Big(\sum_{k=1}^{K} \Pi_k^* \Pi_k\Big)\big(2J_A(x) - x - 2J_A(y) + y\big)\Big\rangle \\ \leqslant \Big\langle x - y \mid \sum_{k=1}^{K} \Omega_k(x - y)\Big\rangle. \end{aligned} \tag{2.9}$$

It thus follows from (2.6), (2.7), and the nonnegativity of $(\Omega_k)_{1\leqslant k\leqslant K}$ that

$$\|2J_A(x) - x - 2J_A(y) + y\|^2 \leqslant \Big\| \sum_{k=1}^{K} \Omega_k \Big\| \|x - y\|^2 \leqslant \|x - y\|^2. \tag{2.10}$$

This shows that $2J_A - \mathrm{Id}$ is a nonexpansive operator. Hence, based on Proposition A.1, A is an MMO.

(ii): Let k be an arbitrary integer in $\{1,\dots,K\}$. (2.8) can be reexpressed as

$$(\forall (p,q)\in\mathcal{H}^2)(\forall p'\in A(p))(\forall q'\in A(q)) \quad \langle \Pi_k^*\Pi_k(p-q)\mid p'-q'+p-q\rangle \geqslant \langle \Pi_k^*\Pi_k(p-q)\mid p-q\rangle. \tag{2.11}$$

In particular, this inequality holds if $p\in J_A(x)$ and $q\in J_A(y)$ where x and y are arbitrary elements of $\mathcal{H}$. Then, by definition of J_A, we have $x-p\in A(p)$, $y-q\in A(q)$, and (2.11) yields

$$\langle \Pi_k^*\Pi_k(p-q)\mid x-y\rangle \geqslant \langle \Pi_k^*\Pi_k(p-q)\mid p-q\rangle. \tag{2.12}$$

By summing over k and using (2.6), it follows that J_A is firmly nonexpansive and it is thus single valued. (2.12) is then equivalent to

$$\|\Pi_k\big(2J_A(x)-x-2J_A(y)+y\big)\|^2 \leqslant \langle x-y\mid \Pi_k^*\Pi_k(x-y)\rangle. \tag{2.13}$$

This shows that Inequality (2.5) holds with $\Omega_k=\Pi_k^*\Pi_k$. Since (2.7) is then obviously satisfied, A is a stationary MMO. ∎

In order to demonstrate the versatility of stationary MMOs, we feature a few examples of such operators. The validity of these examples is proved in Appendix B.

EXAMPLE 2.7. *For every $k\in\{1,\dots,K\}$, let B_k be an MMO defined on a real Hilbert space $\mathcal{H}_k$ and let B be the operator defined as*

$$(\forall x=(x^{(k)})_{1\leqslant k\leqslant K}\in\mathcal{H}=\mathcal{H}_1\times\cdots\times\mathcal{H}_K)\quad B(x)=B_1(x^{(1)})\times\cdots\times B_K(x^{(K)}). \tag{2.14}$$

Let $U\colon\mathcal{H}\to\mathcal{H}$ be a unitary linear operator. Then $A=U^*BU$ is a stationary MMO.

EXAMPLE 2.8. *For every $k\in\{1,\dots,K\}$, let $\varphi_k\in\Gamma_0(\mathbb{R})$, and let the function g be defined as*

$$(\forall x=(x^{(k)})_{1\leqslant k\leqslant K}\in\mathbb{R}^K)\quad g(x)=\sum_{k=1}^{K}\varphi_k(x^{(k)}). \tag{2.15}$$

Let $U\in\mathbb{R}^{K\times K}$ be an orthogonal matrix. Then the subdifferential of $g\circ U$ is a stationary MMO.

EXAMPLE 2.9. *Let $(\mathcal{H}_k)_{1\leqslant k\leqslant K}$ be real Hilbert spaces and let B be a bounded linear operator from $\mathcal{H}=\mathcal{H}_1\times\cdots\times\mathcal{H}_K$ to $\mathcal{H}$ such that one of the following conditions holds:*

(i) *$B+B^*$ is nonnegative*
(ii) *B is skewed*
(iii) *B is cocoercive.*

Let $c\in\mathcal{H}$. Then the affine operator $A\colon\mathcal{H}\to\mathcal{H}\colon x\mapsto Bx+c$ is a stationary MMO.

EXAMPLE 2.10. *Let $(\mathcal{H}_k)_{1\leqslant k\leqslant K}$ be real Hilbert spaces, let $\mathcal{H}=\mathcal{H}_1\times\cdots\times\mathcal{H}_K$, and let $A\colon\mathcal{H}\to 2^{\mathcal{H}}$ be a stationary MMO. Then its inverse A^{-1} is a stationary MMO.*

EXAMPLE 2.11. *Let $(\mathcal{H}_k)_{1\leqslant k\leqslant K}$ be real Hilbert spaces, let $\mathcal{H}=\mathcal{H}_1\times\cdots\times\mathcal{H}_K$, and let $A\colon\mathcal{H}\to 2^{\mathcal{H}}$ be a stationary MMO. Then, for every $\rho\in\mathbb{R}\smallsetminus\{0\}$, $\rho A(\cdot/\rho)$ is a stationary MMO.*

2.4. Universal approximation theorem. In this section we provide one of the main contributions of this article, consisting in a universal approximation theorem for stationary MMOs defined on $\mathcal{H}=\mathbb{R}^K$. To this aim, we first need to introduce useful results, starting by recalling the definition of a lattice.

DEFINITION 2.12. *A set $\mathcal{L}_E$ of functions from a set E to $\mathbb{R}$ is said to be a* lattice *if, for every $(h^{(1)}, h^{(2)}) \in \mathcal{L}_E^2$, $\min\{h^{(1)}, h^{(2)}\}$ and $\max\{h^{(1)}, h^{(2)}\}$ belong to $\mathcal{L}_E$. A sub-lattice of $\mathcal{L}_E$ is a lattice included in $\mathcal{L}_E$.*

This notion of lattice is essential in the variant of the Stone-Weierstrass theorem provided below.

PROPOSITION 2.13. [5] *Let (E, d) be a compact metric space with at least two distinct points. Let $\mathcal{L}_E$ be a sub-lattice of $\mathrm{Lip}_1(E, \mathbb{R})$, the class of 1-Lipschtzian (i.e. nonexpansive) functions from E to $\mathbb{R}$. Assume that, for every $(u, v) \in E^2$ with $u \neq v$ and, for every $(\zeta, \eta) \in \mathbb{R}^2$ such that $|\zeta - \eta| \leqslant d(u, v)$, there exists a function $h \in \mathcal{L}_E$ such that $h(u) = \zeta$ and $h(v) = \eta$. Then $\mathcal{L}_E$ is dense in $\mathrm{Lip}_1(E, \mathbb{R})$ for the uniform norm.*

This allows us to derive the following approximation result that will be instrumental to prove our main result.

COROLLARY 2.14. *Let V be a subspace of $\mathbb{R}^K$ and let $h \in \mathrm{Lip}_1(V, \mathbb{R})$. Let E be a compact subset of V. Then, for every $\varepsilon \in]0, +\infty[$, there exists $h_\varepsilon \in \mathcal{N}_\mathcal{F}(V, \mathbb{R})$, where $\mathcal{F}$ is any dictionary of activation function satisfying Assumption 2.4, such that*

$$(\forall x \in E) \quad |h(x) - h_\varepsilon(x)| \leqslant \varepsilon. \tag{2.16}$$

Proof. First note that $\mathcal{N}_\mathcal{F}(V, \mathbb{R})$ is a lattice. Indeed, if $h^{(1)} \colon V \to \mathbb{R}$ and $h^{(2)} \colon V \to \mathbb{R}$ are 1-Lipschitzian, then $\min\{h^{(1)}, h^{(2)}\}$ and $\max\{h^{(1)}, h^{(2)}\}$ are 1-Lipschitzian. In addition, if $h^{(1)}$ and $h^{(2)}$ are elements in $\mathcal{N}_\mathcal{F}(V, \mathbb{R})$, then by applying sorting operations on the two outputs of these two networks, $\min\{h^{(1)}, h^{(2)}\}$ and $\max\{h^{(1)}, h^{(2)}\}$ are generated. Each of these outputs can be further selected by applying weight matrices either equal to $[1 \;\; 0]$ or $[0 \;\; 1]$ as a last operation, so leading to a NN in $\mathcal{N}_\mathcal{F}(V, \mathbb{R})$.

Let E be a compact subset of V. Assume that E has at least two distinct points. Since $\mathcal{N}_\mathcal{F}(V, \mathbb{R})$ is a lattice, the set of restrictions to E of elements in $\mathcal{N}_\mathcal{F}(V, \mathbb{R})$ is a sub-lattice $\mathcal{L}_E$ of $\mathrm{Lip}_1(E, \mathbb{R})$. In addition, let $(u, v) \in E^2$ with $u \neq v$ and let $(\zeta, \eta) \in \mathbb{R}^2$ be such that $|\zeta - \eta| \leqslant \|u - v\|$. Set $h \colon V \to \mathbb{R} \colon x \mapsto w^\top (x - v) + \eta$ where $w = (\zeta - \eta)(u - v)/\|u - v\|^2$. Since $\|w\| = |\zeta - \eta|/\|u - v\| \leqslant 1$, h is a linear network in $\mathcal{N}_\mathcal{F}(V, \mathbb{R})$ and we have $h(u) = \zeta$ and $h(v) = \eta$. This shows that the restriction of h to E is an element of $\mathcal{L}_E$ satisfying the assumptions of Proposition 2.13. It can thus be deduced from this proposition that (2.16) holds.

The inequality also trivially holds if E reduces to a single point x since it is always possible to find a linear network in $\mathcal{N}_\mathcal{F}(V, \mathbb{R})$ whose output equals $h(x)$. ∎

REMARK 2.15. *This result is valid whatever the norm used on V.*

We are now able to state a universal approximation theorem for MMOs defined on $\mathcal{H} = \mathbb{R}^K$ (i.e., for every $k \in \{1, \ldots, K\}$, $\mathcal{H}_k = \mathbb{R}$ in Definition 2.5).

THEOREM 2.16. *Let $\mathcal{H} = \mathbb{R}^K$. Let $A \colon \mathcal{H} \to 2^\mathcal{H}$ be a stationary MMO. For every compact set $S \subset \mathcal{H}$ and every $\epsilon \in]0, +\infty[$, there exists a NN $Q_\epsilon \in \mathcal{N}_\mathcal{F}(\mathcal{H}, \mathcal{H})$, where $\mathcal{F}$ is any dictionary of activation function satisfying Assumption 2.4, such that $A_\epsilon = 2(\mathrm{Id} + Q_\epsilon)^{-1} - \mathrm{Id}$ satisfies the following properties.*

(i) *For every $x \in S$, $\|J_A(x) - J_{A_\epsilon}(x)\| \leqslant \epsilon$.*

(ii) *Let $x \in \mathcal{H}$ and let $y \in A(x)$ be such that $x + y \in S$. Then, there exists $x_\epsilon \in \mathcal{H}$ and $y_\epsilon \in A_\epsilon(x_\epsilon)$ such that $\|x - x_\epsilon\| \leqslant \epsilon$ and $\|y - y_\epsilon\| \leqslant \epsilon$.*

Proof. (i): If $A \colon \mathbb{R}^K \to 2^{\mathbb{R}^K}$ is a stationary MMO then it follows from Propositions 2.1 and 2.6(i) that there exists a nonexpansive operator $Q \colon \mathbb{R}^K \to \mathbb{R}^K$ such

that $J_A = (\mathrm{Id}+Q)/2$. In addition, according to Definition 2.5, there exist vectors $(p_k)_{1\leqslant k\leqslant K}$ in $\mathbb{R}^K$ such that, for every $k \in \{1,\ldots,K\}$,

$$(\forall (x,y)\in\mathcal{H}^2)\quad |\langle p_k \mid Q(x)-Q(y)\rangle|^2 \leqslant \langle x-y \mid \Omega_k(x-y)\rangle \tag{2.17}$$

where

$$\sum_{k=1}^{K} p_k p_k^\top = \mathrm{Id} \tag{2.18}$$

and $(\Omega_k)_{1\leqslant k\leqslant K}$ are positive semidefinite matrices in $\mathbb{R}^{K\times K}$ satisfying (2.7).

Set $k \in \{1,\ldots,K\}$ and define $h_k\colon x \mapsto \langle p_k \mid Q(x)\rangle$. Let V_k be the nullspace of Ω_k and let $V_k^\perp$ be its orthogonal space. We distinguish the cases when $V_k^\perp \neq \{0\}$ and when $V_k^\perp = \{0\}$. Assume that $V_k^\perp \neq \{0\}$. It follows from (2.17) that, for every $x \in V_k^\perp$ and $(y,z)\in V_k^2$,

$$h_k(x+y) = h_k(x+z) = \widetilde{h}_k(x) \tag{2.19}$$

where $\widetilde{h}_k\colon V_k^\perp \to \mathbb{R}$ is such that

$$(\forall (x,x')\in (V_k^\perp)^2)\quad |\widetilde{h}_k(x)-\widetilde{h}_k(x')| \leqslant \|x-x'\|_{\Omega_k} \tag{2.20}$$

and $(\forall x\in\mathbb{R}^K)\ \|x\|_{\Omega_k} = \langle x \mid \Omega_k x\rangle^{1/2}$. $\|\cdot\|_{\Omega_k}$ defines a norm on $V_k^\perp$. Inequality (2.20) shows that $\widetilde{h}_k$ is 1-Lipschitzian on $V_k^\perp$ equipped with this norm. Let S be a compact subset of $\mathbb{R}^K$ and let $\mathrm{proj}_{V_k^\perp}$ be the orthogonal projection onto $V_k^\perp$. $E_k = \mathrm{proj}_{V_k^\perp}(S)$ is a compact set and, in view of Corollary 2.14, for every $\epsilon\in\mathbb{R}$, there exists $\widetilde{h}_{k,\epsilon} \in \mathcal{N}_{\mathcal{F}}(V_k^\perp,\mathbb{R})$ such that

$$(\forall x\in E_k)\quad |\widetilde{h}_k(x)-\widetilde{h}_{k,\epsilon}(x)| \leqslant \frac{2\epsilon}{\sqrt{K}}. \tag{2.21}$$

Set now $h_{k,\epsilon} = \widetilde{h}_{k,\epsilon}\circ \mathrm{proj}_{V_k^\perp}$. According to (2.19) and (2.21), we have

$$\begin{aligned}(\forall x\in S)\quad & |h_k(x)-h_{k,\epsilon}(x)|\\ &= |h_k(\mathrm{proj}_{V_k}(x)+\mathrm{proj}_{V_k^\perp}(x)) - h_{k,\epsilon}(\mathrm{proj}_{V_k}(x)+\mathrm{proj}_{V_k^\perp}(x))|\\ &= |\widetilde{h}_k(\mathrm{proj}_{V_k^\perp}(x)) - \widetilde{h}_{k,\epsilon}(\mathrm{proj}_{V_k^\perp}(x))|\\ &\leqslant \frac{2\epsilon}{\sqrt{K}}.\end{aligned} \tag{2.22}$$

In addition, by using the Lipschitz property of $\widetilde{h}_{k,\epsilon}$ with respect to norm $\|\cdot\|_{\Omega_k}$, for every $(x,x')\in\mathbb{R}^K$,

$$\begin{aligned}&\big(h_{k,\epsilon}(x)-h_{k,\epsilon}(x')\big)^2\\ &= \big(\widetilde{h}_{k,\epsilon}(\mathrm{proj}_{V_k^\perp}(x)) - \widetilde{h}_{k,\epsilon}(\mathrm{proj}_{V_k^\perp}(x'))\big)^2\\ &\leqslant \|\mathrm{proj}_{V_k^\perp}(x)-\mathrm{proj}_{V_k^\perp}(x')\|_{\Omega_k}^2\\ &= \Big\langle \mathrm{proj}_{V_k^\perp}(x-x') \mid \Omega_k \mathrm{proj}_{V_k^\perp}(x-x')\Big\rangle\\ &= \Big\langle \Omega_k^{1/2}\mathrm{proj}_{V_k^\perp}(x-x') \mid \Omega_k^{1/2}\mathrm{proj}_{V_k^\perp}(x-x')\Big\rangle\\ &= \langle x-x' \mid \Omega_k(x-x')\rangle.\end{aligned} \tag{2.23}$$

If $V_k^\perp = \{0\}$, then it follows from (2.17) that $h_k = 0$. Therefore, (2.22) holds with $h_{k,\epsilon} = 0$, which belongs to $\mathcal{N}_\mathcal{F}(V_k^\perp, \mathbb{R})$ and obviously satisfies (2.23).

Condition (2.18) means that $(p_k)_{1\leqslant k\leqslant K}$ is an orthornormal basis of $\mathbb{R}^K$ in the standard Euclidean metric. This implies that

$$(\forall x \in \mathbb{R}^K) \quad Q(x) = \sum_{k=1}^{K} h_k(x)\, p_k. \tag{2.24}$$

Set

$$(\forall x \in \mathbb{R}^K) \quad Q_\epsilon(x) = \sum_{k=1}^{K} h_{k,\epsilon}(x)\, p_k. \tag{2.25}$$

It follows from (2.23) and (2.7) that, for every $(x, x') \in (\mathbb{R}^K)^2$,

$$\begin{aligned} &\|Q_\epsilon(x) - Q_\epsilon(x')\|^2 \\ &= \sum_{k=1}^{K} \big(h_{k,\epsilon}(x) - h_{k,\epsilon}(x')\big)^2 \\ &\leqslant \sum_{k=1}^{K} \langle x - x' \mid \Omega_k(x - x')\rangle \\ &\leqslant \|x - x'\|^2, \end{aligned} \tag{2.26}$$

which shows that $Q_\epsilon \in \mathrm{Lip}_1(\mathbb{R}^K, \mathbb{R}^K)$. In addition since, for every $x \in \mathbb{R}^K$,

$$Q_\epsilon(x) = W[h_{1,\epsilon}(x), \ldots, h_{K,\epsilon}(x)]^\top \tag{2.27}$$

with $W = [p_1, \ldots, p_K]$ and, for every $k \in \mathbb{N}$, $h_{k,\epsilon} \in \mathcal{N}_\mathcal{F}(\mathbb{R}^K, \mathbb{R})$, Q_ϵ belongs to $\mathcal{N}_\mathcal{F}(\mathbb{R}^K, \mathbb{R}^K)$. Let $A_\epsilon = 2(\mathrm{Id} + Q_\epsilon)^{-1} - \mathrm{Id}$. We finally deduce from (2.22) that, for every $x \in S$,

$$\begin{aligned} &\|J_A(x) - J_{A_\epsilon}(x)\|^2 \\ &= \Big\| \frac{x + Q(x)}{2} - \frac{x + Q_\epsilon(x)}{2} \Big\|^2 \\ &= \frac{1}{4} \sum_{k=1}^{K} \big(h_k(x) - h_{k,\epsilon}(x)\big)^2 \leqslant \epsilon^2. \end{aligned} \tag{2.28}$$

(ii): Let $(x, y) \in (\mathbb{R}^K)^2$. We have

$$y \in A(x) \quad \Leftrightarrow \quad x = J_A(x + y). \tag{2.29}$$

Assume that $x + y \in S$. It follows from (i) that there exists $x_\epsilon \in \mathbb{R}^K$ such that $x_\epsilon = J_{A_\epsilon}(x + y)$ and $\|x - x_\epsilon\| \leqslant \epsilon$. Let $y_\epsilon = x - x_\epsilon + y$. We have $x_\epsilon = J_{A_\epsilon}(x_\epsilon + y_\epsilon)$, that is $y_\epsilon \in A_\epsilon(x_\epsilon)$. In addition, $\|y - y_\epsilon\| = \|x - x_\epsilon\| \leqslant \epsilon$. ∎

We will next show that Theorem 2.16 extends to a wider class of MMOs.

Corollary 2.17. *Let $\mathcal{H} = \mathbb{R}^K$. Let $(\omega_i)_{1\leqslant i\leqslant I} \in]0,1]^I$ be such that $\sum_{i=1}^I \omega_i = 1$. For every $i \in \{1, \ldots, I\}$, let $A_i \colon \mathcal{H} \to 2^\mathcal{H}$ be a stationary MMO. Then the same properties as in Theorem 2.16 hold if $A \colon \mathcal{H} \to 2^\mathcal{H}$ is the MMO with resolvent $J_A = \sum_{i=1}^I \omega_i J_{A_i}$.*

Proof. First note that $J_A\colon \mathcal{H} \to \mathcal{H}$ is firmly nonexpansive [7, Proposition 4.6]), hence A is indeed an MMO. As a consequence of Theorem 2.16, for every compact set $S \subset \mathcal{H}$ and every $\epsilon \in]0,+\infty[$, there exist NNs $(Q_{i,\epsilon})_{1\leqslant i\leqslant I}$ in $\mathcal{N}_{\mathcal{F}}(\mathcal{H},\mathcal{H})$ such that $(A_{i,\epsilon})_{1\leqslant i\leqslant I} = \big(2(\mathrm{Id}+Q_{i,\epsilon})^{-1} - \mathrm{Id}\,\big)_{1\leqslant i\leqslant I}$ satisfy:

$$(\forall i \in \{1,\dots,Q\})(\forall x \in S) \quad \|J_{A_i}(x) - J_{A_{i,\epsilon}}(x)\| \leqslant \epsilon. \tag{2.30}$$

Let $Q_\epsilon = \sum_{i=1}^I \omega_i Q_{i,\epsilon}$. Then $Q_\epsilon \in \mathrm{Lip}_1(\mathbb{R}^K,\mathbb{R}^K)$ and, since it is built from a linear combination of the outputs of I NNs in $\mathcal{N}_{\mathcal{F}}(\mathcal{H},\mathcal{H})$ driven with the same input, it belongs to $\mathcal{N}_{\mathcal{F}}(\mathcal{H},\mathcal{H})$. In addition, $A_\epsilon = 2(\mathrm{Id}+Q_\epsilon)^{-1} - \mathrm{Id}$ is such that

$$J_{A_\epsilon} = \frac{1}{2}\Big(\sum_{i=1}^I \omega_i Q_{i,\epsilon} + \mathrm{Id}\Big) = \sum_{i=1}^I \omega_i J_{A_{i,\epsilon}}, \tag{2.31}$$

which allows us to deduce from (2.30) that

$$(\forall x \in S) \quad \|J_A(x) - J_{A_\epsilon}(x)\| \leqslant \sum_{i=1}^I \omega_i \|J_{A_i}(x) - J_{A_{i,\epsilon}}(x)\| \leqslant \epsilon. \tag{2.32}$$

The rest of the proof follows the same line as for Theorem 2.16. ∎

Remark 2.18. *The above results are less accurate than standard universal approximations ones which, for example, guarantee an arbitrary close approximation to any continuous function with a network having only one hidden layer [34, 40]. Indeed, the requirement that the resolvent of a MMO must be firmly nonexpansive induces some significant increase of the difficulty of the mathematical problem. Nonetheless, the firm nonexpansiveness will enable us to build convergent PnP algorithms described in the next sections.*

3. Proposed algorithm.

3.1. Forward-backward algorithm. Let us now come back to problems of the form (1.3). Such monotone inclusion problems can be tackled by a number of algorithms [18, 23], which are all grounded on the use of the resolvent of A (or a scaled version of this operator). For simplicity, let us assume that f is a smooth function. In this case, a famous algorithm for solving (1.3) is the forward-backward (FB) algorithm [14, 24], which is expressed as

$$(\forall n \in \mathbb{N}) \quad x_{n+1} = J_{\gamma A}\big(x_n - \gamma \nabla f(x_n)\big) \tag{3.1}$$

where $\gamma > 0$. If a neural network $\widetilde{J}$ is used to approximate $J_{\gamma A}$, then a natural substitute for (3.1) is

$$(\forall n \in \mathbb{N}) \quad x_{n+1} = \widetilde{J}\big(x_n - \gamma \nabla f(x_n)\big). \tag{3.2}$$

The following convergence result then straightforwardly follows from standard asymptotic properties of the FB algorithm [24].

Proposition 3.1. *Let $\mu \in]0,+\infty[$ and let $\gamma \in]0, 2/\mu[$. Let $f\colon \mathcal{H} \to \mathbb{R}$ be a convex differentiable function with μ-Lipschitzian gradient. Let $\widetilde{J}$ be a neural network such that $\widetilde{J}$ is $1/2$-averaged as in (2.1). Let $\widetilde{A}$ be the maximally monotone operator equal to $(\widetilde{J}^{-1} - \mathrm{Id})$. Assume that the set $\mathcal{S}_\gamma$ of zeros of $\nabla f + \gamma^{-1}\widetilde{A}$ is nonempty. Then,*

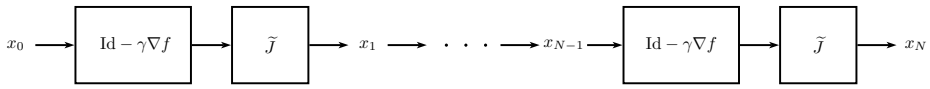


Fig. 1: Unfolded FB algorithm over N iterations.

the sequence $(x_n)_{n\in\mathbb{N}}$ *generated by iteration* (3.2) *converges (weakly) to* $\widehat{x} \in \mathcal{S}_\gamma$, *i.e.,* $\widehat{x}$ *satisfies*

$$0 \in \nabla f(\widehat{x}) + \gamma^{-1}\widetilde{A}(\widehat{x}). \tag{3.3}$$

Remark 3.2.

(i) For a given choice of $\widetilde{A}$, *the stepsize* γ *in the proposed PnP-FB does not only act on the convergence profile, but also on the set of solutions, as shown in* (3.3).

(ii) The technical assumption $\mathcal{S}_\gamma \neq \varnothing$ *can be waived if* $\nabla f + \gamma^{-1}\widetilde{A}$ *is strongly monotone [7, Corollary 23.37]). Then there exists a unique solution to* (3.3). *This is achieved if* f *is strongly convex or if* $\widetilde{J} = (\mathrm{Id} + Q)/2$ *where* $Q = \widetilde{Q}/(1+\delta)$ *with* $\delta \in]0, +\infty[$ *and* $\widetilde{Q}\colon \mathcal{H} \to \mathcal{H}$ *nonexpansive. In the latter case,* $\widetilde{A} = B + \delta(2+\delta)^{-1}\,\mathrm{Id}$ *where* $B\colon \mathcal{H} \to 2^{\mathcal{H}}$ *is maximally monotone. An example of such a strongly monotone operator* $\widetilde{A}$ *is encountered in elastic net regularization.*

(iii) A classical result by Rockafellar states that $\widetilde{A}$ *is the subdifferential of some convex lower-semincontinuous function if and only if* $\widetilde{A}$ *is maximally* cyclically *monotone [18, 52]. As we only enforce the maximal monotonicity,* (3.3) *does not necessarily correspond to a minimization problem in general.*

Algorithm (3.2) is common to other frameworks, such as regularization-by-denoising (RED) and unfolded networks. As underlined by the authors of [17], the RED-PRO algorithm [17, Algorithm 4.1] can be written as a PnP-FB algorithm of the same form as Algorithm (3.2), where $\widetilde{J} = (1-\alpha)\,\mathrm{Id} + \alpha Q$, with $\alpha \in]0, 1/2[$ and Q being d-demicontractive for $d \in [0, 1[$. So, although [17, Algorithm 4.1] and Algorithm (3.2) have the same structure (considering $\alpha = 1/2$ and that nonexpansiveness implies 0-demicontractive), they also exhibit several differences. First, in [17] the authors train Q as a denoiser, while we train $\widetilde{J}$. In practice, our approach is closer to classical PnP algorithms where a proximity operator is replaced by a denoiser (and not a relaxed version of the denoiser). Second, the characterization of the limit point differs since, in [17, Theorems 4.3 & 4.4], it is shown that RED-PRO converges to a minimizer of f, where the solution is a fixed point of $\widetilde{J}$.

If a finite number N of iterations of Algorithm (3.2) are performed, unfolding the FB algorithm results in the NN architecture given in Figure 1. If $\gamma < 2/\mu$, the gradient operator $(\mathrm{Id} - \gamma\nabla f)$ is a $\gamma\mu/2$-averaged operator. It can thus be interpreted as an activation operator [22]. This activation operator is however non standard both because of its form and its dependence on the observed data z. A special case arises when f corresponds to a least squares data fit term, i.e.,

$$(\forall x \in \mathcal{H}) \quad f(x) = \frac{1}{2}\|Hx - z\|^2, \tag{3.4}$$

where z belongs to some real Hilbert space $\mathcal{G}$ and H is a bounded operator from $\mathcal{H}$ to $\mathcal{G}$ modelling some underlying linear observation process (e.g. a degradation

operator in image recovery). Then, $\nabla f \colon x \mapsto H^*(Hx - z)$ where H^* denotes the adjoint of H and $\mu = \|H\|^2$. Hence, $\mathrm{Id} - \gamma \nabla f$ is an affine operator involving a self-adjoint weight operator $\mathrm{Id} - \gamma H^* H$ and a bias $\gamma H^* z$. The unfolded network has thus a structure similar to a residual network where groups of layers are identically repeated and the bias introduced in the gradient operator depends on z. A parallel could also be drawn with a recurrent neural network driven with a stationary input, which would here correspond to z. It is worth pointing out that, under the assumptions of Proposition 3.1, the unfolded network in Figure 1 is robust to adversarial input perturbations, since it is globally nonexpansive. Note finally that, in the case when f is given by (3.4), allowing the parameter γ and the operator $\widetilde{J}$ to be dependent on $n \in \{1, \ldots, N\}$ in Figure 1 would yield an extension of ISTA-net [66]. However, as shown in [15], convergence of such a scheme requires specific assumptions on the target signal model. Other works have also proposed NN architectures inspired from primal dual algorithms [2, 6, 35].

3.2. Training. A standard way of training a NN operating on $\mathcal{H} = \mathbb{R}^K$ for PnP algorithms is to train a denoiser for data corrupted with Gaussian noise [70]. Let $\overline{\boldsymbol{x}} = (\overline{x}_\ell)_{1 \leqslant \ell \leqslant L}$ be training set of L images of $\mathcal{H}$ and let

$$(\forall \ell \in \{1, \ldots, L\}) \qquad y_\ell = \overline{x}_\ell + \sigma_\ell w_\ell \tag{3.5}$$

be a noisy observation of $\overline{x}_\ell$, where $\sigma_\ell \in \left]0, +\infty\right[$ and $(w_\ell)_{1 \leqslant \ell \leqslant L}$ are assumed to be realizations of standard normal i.i.d. random variables. In practice, either $\sigma_\ell \equiv \sigma > 0$ is chosen to be constant during training [69], or σ_ℓ is chosen to be a realization of a random variable with uniform distribution in $[0, \sigma]$, for $\sigma \in \left]0, +\infty\right[$ [71].

The NN $\widetilde{J}$ described in the previous section will be optimally chosen within a family $\{\widetilde{J}_\theta \mid \theta \in \mathbb{R}^P\}$ of NNs. For example, the parameter vector θ will account for the convolutional kernels and biases of a given network architecture. An optimal value $\widehat{\theta}$ of the parameter vector is thus a solution to the following problem:

$$\underset{\theta}{\text{minimize}} \ \sum_{\ell=1}^{L} \|\widetilde{J}_\theta(y_\ell) - \overline{x}_\ell\|^2 \quad \text{s.t.} \quad Q_\theta = 2\widetilde{J}_\theta - \mathrm{Id} \quad \text{is nonexpansive.} \tag{3.6}$$

(The squared ℓ_2 norm in (3.6) can be replaced by another cost function, e.g., an ℓ_1 norm [67].) The main difficulty with respect to a standard training procedure is the nonexpansiveness constraint stemming from Proposition 2.1 which is crucial to ensure the convergence of the overall PnP algorithm. In this context, the tight sufficient conditions described in Proposition A.1 for building the associated nonexpansive operator Q_θ are however difficult to enforce. For example, the maximum value of the left-hand side in inequality (A.1) is NP-hard to compute [59] and estimating an accurate estimate of the Lipschitz constant of a NN requires some additional assumptions [50] or some techniques which do not scale well to high-dimensional data [29]. In turn, by assuming that, for every $\theta \in \mathbb{R}^P$ Q_θ is differentiable, we leverage on the fact that Q_θ is nonexpansive if and only if its Jacobian $\boldsymbol{\nabla} Q_\theta$ satisfies

$$(\forall x \in \mathcal{H}) \qquad \|\boldsymbol{\nabla} Q_\theta(x)\| \leqslant 1. \tag{3.7}$$

In practice, one cannot enforce the constraint in (3.7) for all $x \in \mathcal{H}$. We therefore propose to impose this constraint on every segment $[\overline{x}_\ell, \widetilde{J}_\theta(y_\ell)]$ with $\ell \in \{1, \ldots, L\}$, or more precisely at points

$$\widetilde{x}_\ell = \varrho_\ell \overline{x}_\ell + (1 - \varrho_\ell)\widetilde{J}_\theta(y_\ell), \tag{3.8}$$

where ϱ_ℓ is a realization of a random variable with uniform distribution on [0,1]. To cope with the resulting constraints, instead of using projection techniques which might be slow [58] and raise convergence issues when embedded in existing training algorithms [4], we propose to employ an exterior penalty approach. The final optimization problem thus reads

$$\underset{\theta}{\text{minimize}} \quad \sum_{\ell=1}^{L} \Phi_\ell(\theta), \tag{3.9}$$

where, for every $\ell \in \{1, \ldots, L\}$,

$$\Phi_\ell(\theta) = \|\widetilde{J}_\theta(y_\ell) - \overline{x}_\ell\|^2 + \lambda \max\left\{\|\boldsymbol{\nabla} Q_\theta(\widetilde{x}_\ell)\|^2, 1-\varepsilon\right\}, \tag{3.10}$$

$\lambda \in]0, +\infty[$ is a penalization parameter, and $\varepsilon \in]0,1[$ is a parameter allowing us to control the constraints. Standard results concerning penalization methods [42, Section 13.1], guarantee that, if $\widehat{\theta}_\lambda$ is a solution to (3.9) for $\lambda \in]0, +\infty[$, then $(\forall \ell \in \{1, \ldots, L\})$ $\lim_{\lambda \to +\infty} \|\boldsymbol{\nabla} Q_{\widehat{\theta}_\lambda}(\tilde{x}_\ell)\|^2 \leqslant 1-\varepsilon$. Then, since $\varepsilon > 0$, there exists $\overline{\lambda} \in]0, +\infty[$ such that, for every $\lambda \in [\overline{\lambda}, +\infty[$ and every $\ell \in \{1, \ldots, L\}$, $\|\boldsymbol{\nabla} Q_{\widehat{\theta}_\lambda}(\tilde{x}_\ell)\| \leqslant 1$.

Remark 3.3. *Hereabove, we have made the assumptions that the network is differentiable. Automatic differentiation tools however are applicable to networks which contain nonsmooth linearities such as ReLU (see [11] for a theoretical justification for this fact).*

To solve (3.9) numerically, we resort to the Adam optimizer [71] as described in Algorithm 3.1. This algorithm uses a fixed number of iterations $N \in \mathbb{N}^*$ and relies on approximations to the gradient of $\sum_\ell \Phi_\ell$ computed on randomly sampled batches of size D, selected from the training set of images $(\overline{x}_\ell)_{1 \leqslant \ell \leqslant L}$. More precisely, at each iteration $t \in \{1, \ldots, N\}$, we build the approximated gradient $\frac{1}{D}\sum_{d=1}^{D} g_d$ (see lines 3-9), followed by an Adam update (line 10) consisting in a gradient step on θ_d with adaptive moment [36]. Then the approximated gradient is computed as follows. For every $d \in \{1, \ldots, D\}$, we select randomly an image from the training set (line 4), we draw at random a realization of a normal i.i.d. noise that we use to build a noisy observation y_d (line 5-6). We then build $\widetilde{x}_d$ as in (3.8) (lines 5-7) and compute the gradient g_d of the loss Φ_d w.r.t. to the parameter vector at its current estimate θ_n (line 8). Note that any other gradient-based algorithm, such as SGD or RMSprop [49] could be used to solve (3.9).

In order to compute the spectral norm $\|\boldsymbol{\nabla} Q_\theta(x)\|$ for a given image $x \in \mathcal{H}$, we use the power iterative method (see details in Appendix C).

Remark 3.4.

(i) *Other works in the GAN literature have investigated similar regularizations [31, 51, 62] to constrain the gradient norm of the discriminator (recall that, the disciminator being a function with values in $\mathbb{R}$, its gradient is well defined). In our work, we aim at constraining the Lipschitz constant of $Q : \mathbb{R}^N \to \mathbb{R}^N$, the gradient of which is not defined (only its Jacobian is), hence the need of a more involved method for computing the spectral norm.*

(ii) *A related approach was presented in [33], where the loss is regularized with the Froebenius norm of the Jacobian. The latter is not enough to ensure convergence of the PnP method (3.2) which requires to constrain the spectral norm $\|\cdot\|$ of the Jacobian.*

Algorithm 3.1 Adam algorithm to solve (3.9)

Let $D \in \mathbb{N}^*$ be the batch size, and $N \in \mathbb{N}^*$ be the number of training iterations.
for $n = 1, \ldots, N$ **do**
  for $d = 1, \ldots, D$ **do**
    Select randomly $\ell \in \{1, \ldots, L\}$;
    Draw at random $w_d \sim \mathcal{N}(0,1)$ and $\varrho_d \sim \mathcal{U}([0,1])$;
    $y_d = \overline{x}_\ell + \sigma w_d$;
    $\widetilde{x}_d = \varrho_d \overline{x}_\ell + (1-\varrho_d)\widetilde{J}_{\theta_n}(y_d)$;
    $g_d = \nabla_\theta \Phi_d(\theta_n)$;
  end for
  $\theta_{n+1} = \mathrm{Adam}(\frac{1}{D}\sum_{d=1}^{D} g_d, \theta_n)$;
end for
return $\widetilde{J}_{\theta_N}$

(iii) *The power iterative method has been used in previous works [47, 54, 65]. In particular, RealSN [54] relies on a power iteration to compute the Lipschitz constant of each convolutional layer. In our case, we compute the spectral norm of the Jacobian for the full network Q.*

4. Simulations and results.

4.1. Experimental setting.

Inverse Problem. We focus on inverse deblurring imaging problems, where the objective is to find an estimate $\widehat{x} \in \mathbb{R}^K$ of an original unknown image $\overline{x} \in \mathbb{R}^K$, from degraded measurements $z \in \mathbb{R}^K$ given by

$$z = H\overline{x} + e, \tag{4.1}$$

where $H \colon \mathbb{R}^K \to \mathbb{R}^K$ is a blur operator and $e \in \mathbb{R}^K$ is a realization of an additive white Gaussian random noise with zero-mean and standard deviation $\nu \in]0, +\infty[$. In this context, a standard choice for the data-fidelity term is given by (3.4). In our simulations, H models a blurring operator implemented as a circular convolution with impulse response h. We will consider different kernels h taken from [41] and [8], see Figure 2 for an illustration. The considered kernels are normalized such that the Lipschitz constant μ of the gradient of f is equal to 1.

Datasets. Our training dataset consists of 50000 test images from the ImageNet dataset [28] that we randomly split in 98% for training and 2% for validation. In the case of grayscale images, we investigate the behavior of our method either on the full BSD68 dataset [45] or on a subset of 10 images, which we refer to as the BSD10 set. For color images, we consider both the BSD500 test set [45] and the Flickr30 test set [63].[1] Eventually, when some fine-tuning is required, we employ the Set12 and Set18 datasets [69] for grayscale and color images, respectively.

Network architecture and pretraining. In existing PnP algorithms involving NNs (see e.g. [39, 68, 69, 71]), the NN architecture $\widetilde{J}$ often relies on residual skip connections. This is equivalent, in (2.3), to set $Q = \mathrm{Id} + \widetilde{T}_M \ldots \widetilde{T}_1$ where, for every $m \in \{1, \ldots, M\}$, $\widetilde{T}_m$ is standard neural network layer (affine operator followed by activation operator). More specifically, the architecture we consider for $\widetilde{J}$ is such

[1] We consider normalised images, where the coefficient values are in $[0,1]$ (resp. $[0,1]^3$) for grayscale (resp. color) images.

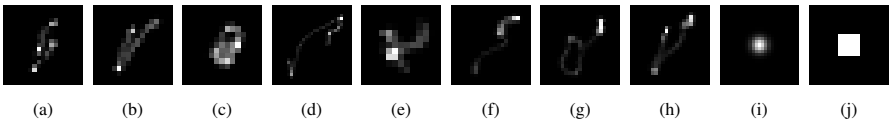


Fig. 2: Blur kernels used in our simulations. (a)-(h) are kernels 1-8 from [41] respectively while (i) is the kernel from the GaussianA setup and (j) from the Square setup in [8].

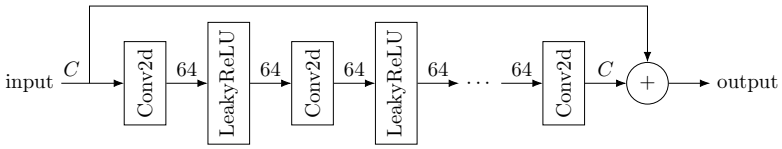


Fig. 3: Proposed DnCNN architecture of $\widetilde{J}$, with a total of 20 convolutional layers. It corresponds to a modified version of the DnCNN-B architecture [68]. The number of channels C is indicated above arrows ($C = 1$ for grayscale images and $C = 3$ for color ones).

that $M = 20$. It is derived from the DnCNN-B architecture [68] from which we have removed batch normalization layers and where we have replaced ReLUs with LeakyReLUs (see Figure 3).

We first pretrain the model $\widetilde{J}$ in order to perform a denoising task for a variable level of noise, without any Jacobian regularization. For each training batch, we generate randomly sampled patches of size 50×50 from images that are randomly rescaled and flipped. More precisely, we consider Problem (3.9)-(3.10) with $\lambda = 0$, and $(\sigma_\ell)_{1 \leqslant \ell \leqslant L}$ chosen to be realizations of i.i.d. random variable with uniform distribution in $[0, 0.1]$ for each patch. We use the Adam optimizer [36] to pretrain the network with learning rate 10^{-4}, and considering 150 epochs, each consisting of 490 iterations of the optimizer. The learning rate is divided by 10 after 100 epochs and we clip gradient norms at 10^{-2} for an improved stability during training. This pretrained network will serve as a basis for our subsequent studies. The details regarding the training of our networks will be given on a case-by-case basis in the following sections.

All models are trained on 2 Nvidia Tesla 32 Gb V100 GPUs and experiments are performed in PyTorch[2].

Goal. We aim to study the PnP-FB algorithm (3.2) where $\widetilde{J}$, chosen according to the architecture given in Figure 3, has been trained in order to solve (3.10). We will first study the impact of the choice of the different parameters appearing in the training loss (3.10) on the convergence of the PnP-FB algorithm and on the reconstruction quality. Then, we will compare the proposed method to state-of-the-art iterative algorithms either based on purely variational or PnP methods.

We evaluate the reconstruction quality with Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) metrics [61]. The PSNR between an image $x \in \mathbb{R}^K$ and the ground truth $\overline{x} \in \mathbb{R}^K$ is defined as

$$\mathrm{PSNR}(x, \overline{x}) = 20 \log_{10} \left(\frac{\sqrt{K}\, \max_{1 \leqslant \ell \leqslant L} \overline{x}_\ell}{\|x - \overline{x}_\ell\|} \right), \tag{4.2}$$

[2]Code publicly available at https://github.com/basp-group/PnP-MMO-imaging

where, in our case, we have $\max_{1\leqslant \ell\leqslant L} \overline{x}_\ell = 1$. The SSIM is given by

$$\mathrm{SSIM}(x,\overline{x}) = \frac{(2\mu_x\mu_{\overline{x}}+\vartheta_1)(2\sigma_{x\overline{x}}+\vartheta_2)}{(\mu_x^2+\mu_{\overline{x}}^2+\vartheta_1)(\sigma_x^2+\sigma_{\overline{x}}^2+\vartheta_2)}, \tag{4.3}$$

where (μ_x,σ_x) and $(\mu_{\overline{x}},\sigma_{\overline{x}})$ are the mean and the variance of x and $\overline{x}$ respectively, $\sigma_{x\overline{x}}$ is the cross-covariance between x and $\overline{x}$, and $(\vartheta_1,\vartheta_2) = (10^{-4}, 9\times 10^{-4})$.

4.2. Choice of the parameters. In this section, we study the influence of the parameters[3] (λ,σ,γ) on the results of the PnP-FB algorithm (3.2) applied to the NN in Figure 3. We recall that λ is the parameter acting on the Jacobian regularization, σ is the noise level for which the denoiser is trained, and γ is the stepsize in the PnP-FB algorithm (3.2).

Simulation settings. We consider problem (4.1) with H associated with the kernels shown in Figure 2(a)-(h), and $\nu = 0.01$. In this section, we consider the grayscale images from the BSD68 dataset.

To investigate the convergence behavior of the PnP-FB algorithm, we consider the quantity defined at iteration $n\in\mathbb{N}\setminus\{0\}$ as

$$c_n = \|x_n - x_{n-1}\|/\|x_0\|, \tag{4.4}$$

where $(x_n)_{n\in\mathbb{N}}$ is the sequence generated by the PnP-FB algorithm (3.2). Note that the quantity $(c_n)_{n\in\mathbb{N}}$ is known to be monotonically decreasing if the network $\widetilde{J}$ is firmly nonexpansive [7].

Influence of the Jacobian penalization. We study the influence of λ on the convergence behavior of the PnP-FB algorithm (3.2). In particular we consider $\lambda\in\{5\times 10^{-7}, 10^{-6}, 2\times10^{-6}, 5\times10^{-6}, 10^{-5}, 2\times10^{-5}, 4\times10^{-5}, 1.6\times10^{-4}, 3.2\times10^{-4}, 6.4\times10^{-4}\}$.

After pretraining, we train our DnCNN by considering the loss given in (3.10), in which we set $\varepsilon = 5\times10^{-2}$ and $\sigma = 0.01$. The batches are built as in the pretraining setting. The network is trained for 100 epochs and the learning rate is divided by 10 at epoch 80. The training is performed with Algorithm 3.1 where $D = 100$ and $N = 4.9\times10^4$. For Adam's parameters, we set the learning rate to 10^{-4} and the remaining parameters to the default values provided in [36].

To verify that our training loss enables the firm nonexpansiveness of our NN $\widetilde{J}$, we evaluate the norm of the Jacobian $\|\boldsymbol{\nabla} Q(y_\ell)\|$ on a set of noisy images $(y_\ell)_{1\leqslant\ell\leqslant 68}$, obtained from the BSD68 test set considering the denoising problem (3.5). The maximum of these values is given in Table 1 for the different considered values of λ. We observe that the norm of the Jacobian decreases as λ increases and is smaller than 1 for $\lambda\geqslant 10^{-5}$.

We now investigate the convergence behavior of the PnP-FB algorithm, depending on λ, considering BSD10 (a subset of BSD68). In our simulations, we set $\gamma = 1/\mu = 1$. In Figure 4 we show the values $(c_n)_{1\leqslant n\leqslant 1000}$ for 1000 iterations, considering kernel (a) from Figure 2 for the different values of λ. The case $\lambda = 0$ corresponds to training a DnCNN without the Jacobian regularization. We observe that the stability of the PnP-FB algorithm greatly improves as λ increases: for $\lambda\geqslant 10^{-5}$, all curves are monotonically decreasing. These observations are in line with the metrics from Table 1 showing that $\|\boldsymbol{\nabla} Q(y_\ell)\|\leqslant 1$ for $\lambda\geqslant 10^{-5}$. The case when $\|\boldsymbol{\nabla} Q\|$ is slightly larger than 1 is of interest. In particular, when $\lambda = 5\times10^{-6}$, $\|\boldsymbol{\nabla} Q\|^2\approx 1.03$ (see Table 1) one observes that most curves from Figure 4 (e) are monotonically decreasing,

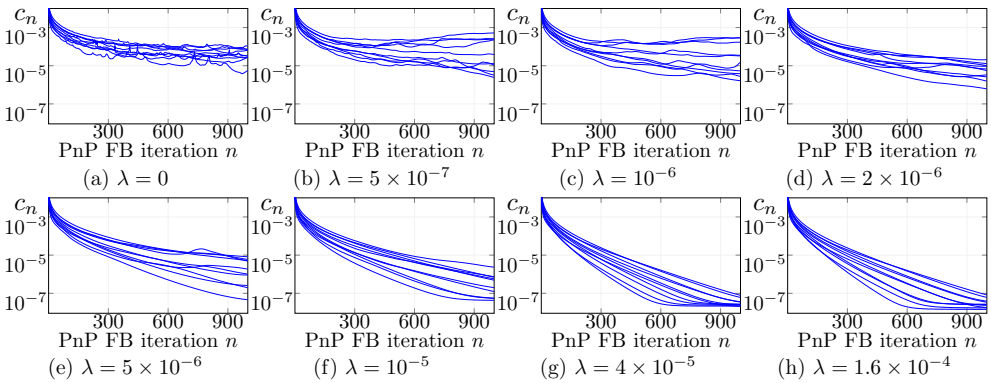
[3] We noticed that the influence of $\varepsilon > 0$ in (3.10) was negligible compared with that of (λ,σ,γ).

but some are not. When $\lambda = 10^{-6}$, $\|\boldsymbol{\nabla} Q\|^2 \approx 1.35$ (see Table 1), none of the curves from Figure 4 (c) are monotonically decreasing, but we observed convergence in some case for a larger number of iterations.

These results confirm that by choosing an appropriate value of λ, one can ensure Q to be 1-Lipschitz, i.e. $\widetilde{J}$ to be firmly nonexpansive, and consequently we secure the convergence of the PnP-FB algorithm (3.2). It also confirms that $\|\boldsymbol{\nabla} Q\| \leqslant 1$ is a sufficient condition to ensure convergence. Finally, we also emphasize that the average PSNR values obtained with the PnP-FB algorithm, for the different considered λ, has a bell shape with maximum obtained when $\lambda = 10^{-5}$.

λ	0	5×10^{-7}	1×10^{-6}	2×10^{-6}	5×10^{-6}	1×10^{-5}	4×10^{-5}	1.6×10^{-4}	3.2×10^{-4}
$\|\boldsymbol{\nabla} Q\|^2$	31.36	1.65	1.349	1.156	1.028	0.9799	0.9449	0.9440	0.9401

Table 1: Numerical evaluation of the firm nonexpansiveness $\widetilde{J}$ on a denoising problem on the BSD68 test set for different values of λ.



(a) $\lambda = 0$ (b) $\lambda = 5 \times 10^{-7}$ (c) $\lambda = 10^{-6}$ (d) $\lambda = 2 \times 10^{-6}$

(e) $\lambda = 5 \times 10^{-6}$ (f) $\lambda = 10^{-5}$ (g) $\lambda = 4 \times 10^{-5}$ (h) $\lambda = 1.6 \times 10^{-4}$

Fig. 4: Influence of $\lambda \in \{0, 5\times10^{-7}, 10^{-6}, 2\times10^{-6}, 5\times10^{-6}, 10^{-5}, 4\times10^{-5}, 1.6\times10^{-4}\}$ on the stability of the PnP-FB algorithm for the deblurring problem with kernel in Figure 2(a). (a)-(h): On each graph, evolution of the quantity c_n defined in (4.4) for each image of the BSD10 test set, for a value of λ, along the iterations of the PnP-FB algorithm (3.2).

Influence of the stepsize and training noise level. Second, we investigate the influence (σ, γ) on the reconstruction quality of the images restored with the PnP-FB algorithm. Indeed, according to Proposition 3.1, the value of γ acts on the solution, and hence potentially on the reconstruction quality. We train the NN $\widetilde{J}$ given in Figure 3 for $\sigma \in \{0.005, 0.006, 0.007, 0.008, 0.009, 0.01\}$. As per the procedure followed in the study of the parameter λ, after pretraining, we train $\widetilde{J}$ by considering the loss given in (3.10), in which we set $\varepsilon = 5 \times 10^{-2}$ and $\lambda = 10^{-5}$. The batches are built as in the pretraining setting. The network is trained for 100 epochs and the learning rate is divided by 10 at epoch 80. The training is performed with Algorithm 3.1 where $D = 100$ and $N = 4.9 \times 10^4$. For Adam's parameters, we set the learning rate to 10^{-4} and the remaining parameters to the default values provided in [36]. We subsequently plug the trained DnCNN $\widetilde{J}$ in the PnP-FB algorithm (3.2), considering different values for $\gamma \in]0, 2[$. In these simulations, we focus on the case when the blur kernel in Problem (4.1) corresponds to the one shown in Figure 2(a).

Before discussing the simulation results, we present a heuristic argument suggesting that (i) σ should scale linearly with γ, and (ii) the appropriate scaling coefficient is given as $2\nu\|h\|$. Given the choice of the data-fidelity term (3.4), the PnP-FB algorithm (3.2) reads

$$(\forall n \in \mathbb{N}) \quad x_{n+1} = \widetilde{J}\big(x_n - \gamma H^*(Hx_n - H\overline{x} - e)\big). \tag{4.5}$$

We know that, under suitable conditions, the sequence $(x_n)_{n\in\mathbb{N}}$ generated by (4.5) converges to a fixed point $\widehat{x}$, solution to the variational inclusion problem (3.3). We assume that $\widehat{x}$ lies close to $\overline{x}$ up to a random residual $e' = H(\widehat{x}-\overline{x})$, whose components are uncorrelated and with equal standard deviation, typically expected to be bounded from above by the standard deviation ν of the components of the original noise e. Around convergence, (4.5) therefore reads as

$$\widehat{x} = \widetilde{J}\left(\widehat{x} - \gamma H^*\left(e' - e\right)\right), \tag{4.6}$$

suggesting that, $\widetilde{J}$ is acting as a denoiser of $\widehat{x}$ for an effective noise $-\gamma H^*(e' - e)$. If the components of $e' - e$ are uncorrelated, the standard deviation of this noise is bounded by $\gamma\nu_{\text{eff}}$, with $\nu_{\text{eff}} = 2\nu\|h\|$, a value reached when $e' = -e$. This linear function of γ with scaling coefficient ν_{eff} thus provides a strong heuristic for the choice of the standard deviation σ of the training noise. For the considered kernel (shown in Figure 2(a)), we have $\nu_{\text{eff}} = 0.0045$, so the interval $\sigma \in [0.005, 0.01]$ also reads $\sigma \in [1.1\,\nu_{\text{eff}}, 2.2\,\nu_{\text{eff}}]$.

In Figure 5 we provide the average PSNR (left) and SSIM (right) values associated with the solutions to the deblurring problem for the considered simulations as a function of $\sigma/\gamma\nu_{\text{eff}}$. For each sub-figure, the different curves correspond to different values of γ. We observe that, whichever the values of γ, the reconstruction quality is sharply peaked around values of $\sigma/\gamma\nu_{\text{eff}}$ consistently around 1, thus supporting our heuristic argument. We also observe that the peak value increases with γ. We recall that, according to the conditions imposed on γ in Proposition 3.1 to guarantee theoretically the convergence of the sequence generated by PnP-FB algorithm, one has $\gamma < 2$. The values $\gamma = 1.99$ and $\sigma/\gamma\nu_{\text{eff}} = 1$ (resp. $\gamma = 1.99$ and $\sigma/\gamma\nu_{\text{eff}} = 0.9$) gives the best results for the PSNR (resp. SSIM).

In Figure 6 we provide visual results for an image from the BSD10 test set, to the deblurring problem for different values of γ and σ. The original unknown image $\overline{x}$ and the observed blurred noisy image are displayed in Figure 6(a) and (g), respectively. On the top row, we set $\sigma = 2\nu_{\text{eff}}$, while the value of γ varies from 1 to 1.99. We observe that the reconstruction quality improves when γ increases, bringing the ratio $\sigma/\gamma\nu_{\text{eff}}$ closer to unity. Precisely, in addition to the PSNR and SSIM values increasing with γ, we can see that the reconstructed image progressively loses its oversmoothed aspect, showing more details. The best reconstruction for this row is given in Figure 6(f), for $\gamma = 1.99$. On the bottom row, we set $\gamma = 1$ and vary σ from $1.3\,\nu_{\text{eff}}$ to $2.2\,\nu_{\text{eff}}$. We see that sharper details appear in the reconstructed image when σ decreases, again bringing the ratio $\sigma/\gamma\nu_{\text{eff}}$ closer to unity. The best reconstructions for this row are given in Figure 6(h) and (i), corresponding to the cases $\sigma = 1.3\,\nu_{\text{eff}}$ and $\sigma = 1.6\,\nu_{\text{eff}}$, respectively. Overall, as we have already noticed, the best reconstruction is obtained for $\gamma = 1.99$ and $\sigma/\gamma\nu_{\text{eff}} = 1$, for which the associated image is displayed in Figure 6(f). It can also be noticed that for all the considered values of σ, the best quality reconstructions are achieved when $\gamma = 1.99/\mu$. These results further support both our analysis of Figure 5 and our heuristic argument for a linear scaling of σ with

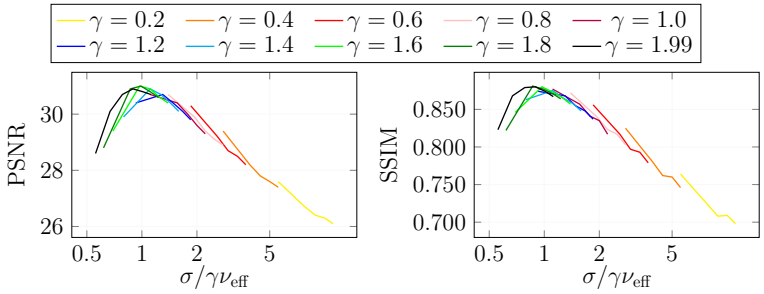


Fig. 5: Influence of $\gamma \in]0, 1.99]$ and $\sigma \in [0.005, 0.01]$ on the reconstruction quality for the deblurring problem with kernel from Figure 2(a) on the BSD10 test set. For this experiment $\nu_{\text{eff}} = 0.0045$. Left: average PSNR, right: average SSIM.

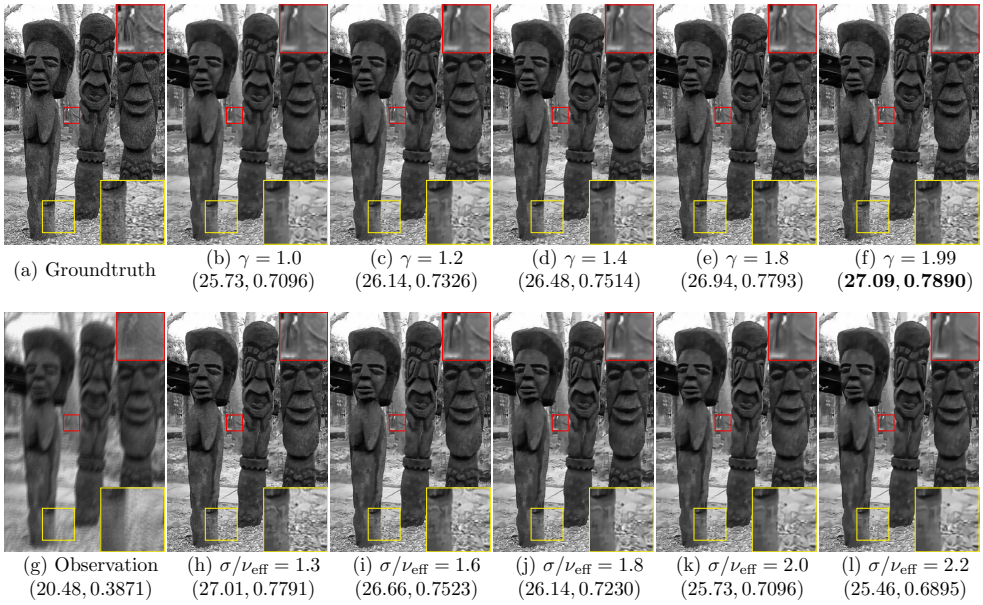
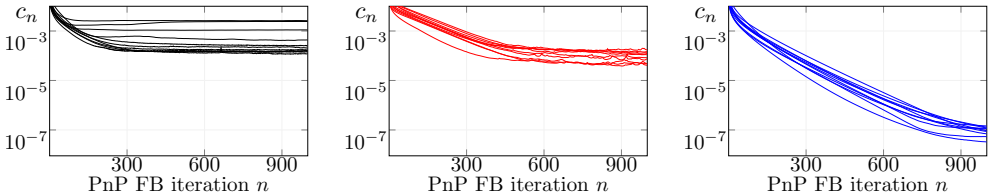


Fig. 6: Reconstructions of an image from the BSD10 test set obtained with the PnP-FB algorithm (3.2) for the deblurring problem with kernel from Figure 2(a) for which $\nu_{\text{eff}} = 0.0045$. Top row: results for $\gamma \in [1, 1.99]$ in algorithm (3.2) and $\sigma/\nu_{\text{eff}} = 2$ (i.e. $\sigma = 0.009$). Bottom row: results for $\sigma/\nu_{\text{eff}} \in [1.3, 2.2]$ during training in (3.10) and $\gamma = 1$ in algorithm (3.2).

γ, with scaling coefficient closely driven by the value ν_{eff}. The best strategy relies on choosing γ the largest as possible, and deduce the value of σ as $\sigma = \gamma\nu_{\text{eff}}$. In other words, our results suggest that the optimal values for both the stepsize and training noise level can be determined as functions of parameters of the data fidelity term f, namely the Lipschitz constant μ of its gradient and the norm of the blurring kernel h, with $\gamma = 1.99/\mu$ and $\sigma = 4\nu\|h\|/\mu$.



(a) BM3D (b) RealSN (c) Proposed

Fig. 7: Convergence profile of the PnP-FB algorithm (3.2) for different denoisers plugged in as $\widetilde{J}$, namely BM3D (a), RealSN (b) and the proposed firmly nonexpansive DnCNN (c). Results are shown for the deblurring problem (4.1) with kernel from Figure 2(a). Each graph shows the evolution of c_n defined in (4.4) for each image of the BSD10 test set.

4.3. Comparison with other PnP methods. In this section we investigate the behavior of the PnP-FB algorithm (3.2) with $\widetilde{J}$ corresponding either to the proposed DnCNN provided in Figure 3, or to other denoisers. In this section, we aim to solve problem (4.1), considering either grayscale or color images.

Grayscale images. We consider the deblurring problem (4.1) with H associated with the kernels from Figure 2(a)-(h), $\nu = 0.01$, evaluated on the BSD10 test set.

We choose the parameters of our method to be the ones leading to the best PSNR values in Figure 5, i.e. $\sigma = 0.009$ and $\gamma = 1.99$ corresponding to $\sigma/\gamma\nu_{\text{eff}} = 1$ for the kernel (a) of Figure 2, and we set $\lambda = 10^{-5}$.

We compare our method with other PnP-FB algorithms, where the denoiser corresponds either to RealSN [54], BM3D [43], DnCNN [68], or standard proximity operators [24, 48]. In our simulations, we consider the proximal operators of the two following functions: (i) the ℓ_1-norm composed with a sparsifying operator consisting in the concatenation of the first eight Daubechies (db) wavelet bases [13, 44], and (ii) the total variation (TV) norm [53]. In both cases, the regularization parameters are fine-tuned on the Set12 dataset [68] to maximize the reconstruction quality. Note that the training process for RealSN has been adapted for the problem of interest.

denoiser	kernel (see Figure 2)								convergence
	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	
Observation	23.36	22.93	23.43	19.49	23.84	19.85	20.75	20.67	
RealSN [54]	26.24	26.25	26.34	25.89	25.08	25.84	24.81	23.92	✓
$\text{prox}_{\mu_{\ell_1}\|\Psi^\dagger\cdot\|_1}$	29.44	29.20	29.31	28.87	30.90	30.81	29.40	29.06	✓
$\text{prox}_{\mu_{\text{TV}}\|\cdot\|_{\text{TV}}}$	29.70	29.35	29.43	29.15	30.67	30.62	29.61	29.23	✓
DnCNN [68]	29.82	29.24	29.26	28.88	30.84	30.95	29.54	29.17	✗
BM3D [43]	30.05	29.53	29.93	29.10	31.08	30.78	29.56	29.41	✗
Proposed	**30.86**	**30.33**	**30.31**	**30.14**	**31.72**	**31.69**	**30.42**	**30.09**	✓

Table 2: Average PSNR values obtained by different denoisers plugged in the PnP-FB algorithm (3.2), to solve the deblurring problem (4.1) with kernels of Figure 2(a)-(h) considering the BSD10 test set. The last row provides the average SSIM values for the observed blurred image y in each experimental setting. Each algorithm is stopped after a fixed number of iterations equal to 1000. The best PSNR values are indicated in bold.

We first check the convergence of the PnP-FB algorithm considering the above-mentioned different denoisers. We study the quantity $(c_n)_{n\in\mathbb{N}}$ defined in (4.4), considering the inverse problem (4.1) with kernel in Figure 2(a). Figure 7 shows the c_n

values with respect to the iterations $n \in \{1, \ldots, 1000\}$ of the PnP-FB algorithm for various denoisers $\widetilde{J}$: BM3D (Figure 7(a)), RealSN (Figure 7(b)), and the proposed firmly nonexpansive DnCNN (Figure 7(c)). On the one hand, as expected, PnP-FB with our network, which has been trained to be firmly nonexpansive, shows a convergent behavior with monotonic decrease of c_n. On the other hand, we notice that the PnP-FB algorithm with BM3D or RealSN does not converge since $(c_n)_{n\in\mathbb{N}}$ does not tend to zero, which confirms that neither BM3D nor RealSN are firmly nonexpansive. Note that, in the case of RealSN [54], nonexpansive-based constraints are also imposed on the network. Nevertheless, in practice these constraints are more restrictive than our approach (see Remark 3.4(iii)), and less accurate (see [54, Assumption A]).

In Table 2 we provide a quantitative analysis of the restoration quality obtained on the BSD10 dataset with the different denoisers. Although DnCNN and BM3D do not benefit from any convergence guarantees, we report the SNR values obtained after 1000 iterations. For all the eight considered kernels, the best PSNR values are delivered by the proposed firmly nonexpansive DnCNN. A possible explanation for the improved performance of our method compared to DnCNN, BM3D and RealSN is the stability of our PnP algorithm induced by the firm nonexpansiveness of our denoiser $\widetilde{J}$.

In Figure 8 we show visual results and associated PSNR and SSIM values obtained with the different methods on the deblurring problem (4.1) with kernel from Figure 2(a). We notice that despite good PSNR and SSIM values, the proximal methods yield reconstructions with strong visual artifacts (wavelet artifacts in Figure 8(c) and cartoon effects in Figure 8(d)). PnP-FB with BM3D provides a smoother image with more appealing visual results, yet some grid-like artifacts appear in some places (see e.g. red boxed zoom in Figure 8(e)). RealSN introduces ripple and dotted artifacts, while DnCNN introduces geometrical artifacts, neither of those corresponding to features in the target image. For this image, we can observe that our method provides better visual results as well as higher PSNR and SSIM values than other methods.

The results presented in this section show that the introduction of the Jacobian regularizer in the training loss (3.10) not only allows to build convergent PnP-FB methods, but also improves the reconstruction quality over both FB algorithms involving standard proximity operators, and existing PnP-FB approaches.

Color images. We now apply our strategy to a color image deblurring problem of the form (4.1), where the noise level and blurring operator are chosen to reproduce the experimental settings of [8], focusing on the four following experiments: First, the Motion A (M. A) setup with blur kernel (h) from Figure 2 and $\nu = 0.01$; second, the Motion B (M. B) setup with blur kernel (c) from Figure 2 and $\nu = 0.01$; third, the Gaussian A (G. A) setup with kernel (i) from Figure 2 and $\nu = 0.008$; finally, the Square (S.) setup with kernel (j) from Figure 2 $\nu = 0.01$. The experiments in this section are run on the Flickr30 dataset and on the test set from BSD500[4]. We compare our method on these problems with the variational method VAR from [8], and three PnP algorithms, namely PDHG [46], and the PnP-FB algorithm combined with the BM3D or DnCNN denoisers. It is worth mentioning that, among the above mentioned methods, only the proposed approach and VAR have convergence guaranties. The results for PDHG and VAR are borrowed from [8].

For the proposed method, we choose $\gamma = 1.99$ in the PnP-FB algorithm (3.2), and we keep the same DnCNN architecture for $\widetilde{J}$ given in Figure 3, only changing

[4] As in [8], we consider 256×256 centered-crop versions of the datasets.

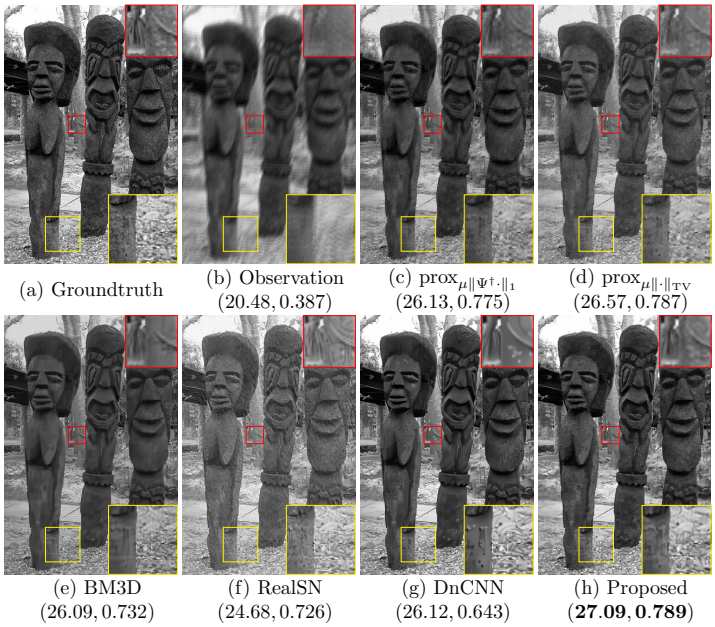


Fig. 8: Reconstructions of an image from the BSD10 test set obtained with the PnP-FB algorithm (3.2), considering different denoisers as $\widetilde{J}$, for the deblurring problem with kernel from Figure 2(a) and $\nu = 0.01$. Associated (PSNR, SSIM) values are indicated below each image, best values are highlighted in bold. Each algorithm is stopped after a fixed number of iteration equal to 1000.

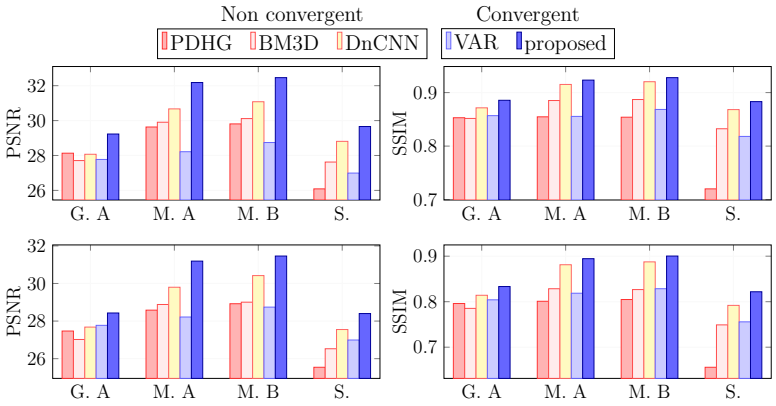


Fig. 9: Average PSNR and SSIM values obtained on the Flickr30 (top) and BSD500 (bottom) test sets using the experimental setups of [8]: G. A, M. A, M. B, and S., for different methods.

the number of input/output channels to $C = 3$. We first pretrain our network as described in subsection 4.1. We then keep on training it considering the loss given in (3.10), in which we set $\varepsilon = 5 \times 10^{-2}$, $\lambda = 10^{-5}$, and $\sigma = 0.007$.

The average PSNR and SSIM values obtained with the different considered reconstruction methods, and for the different experimental settings, are reported in Figure 9. This figure shows that our method significantly improves reconstruction

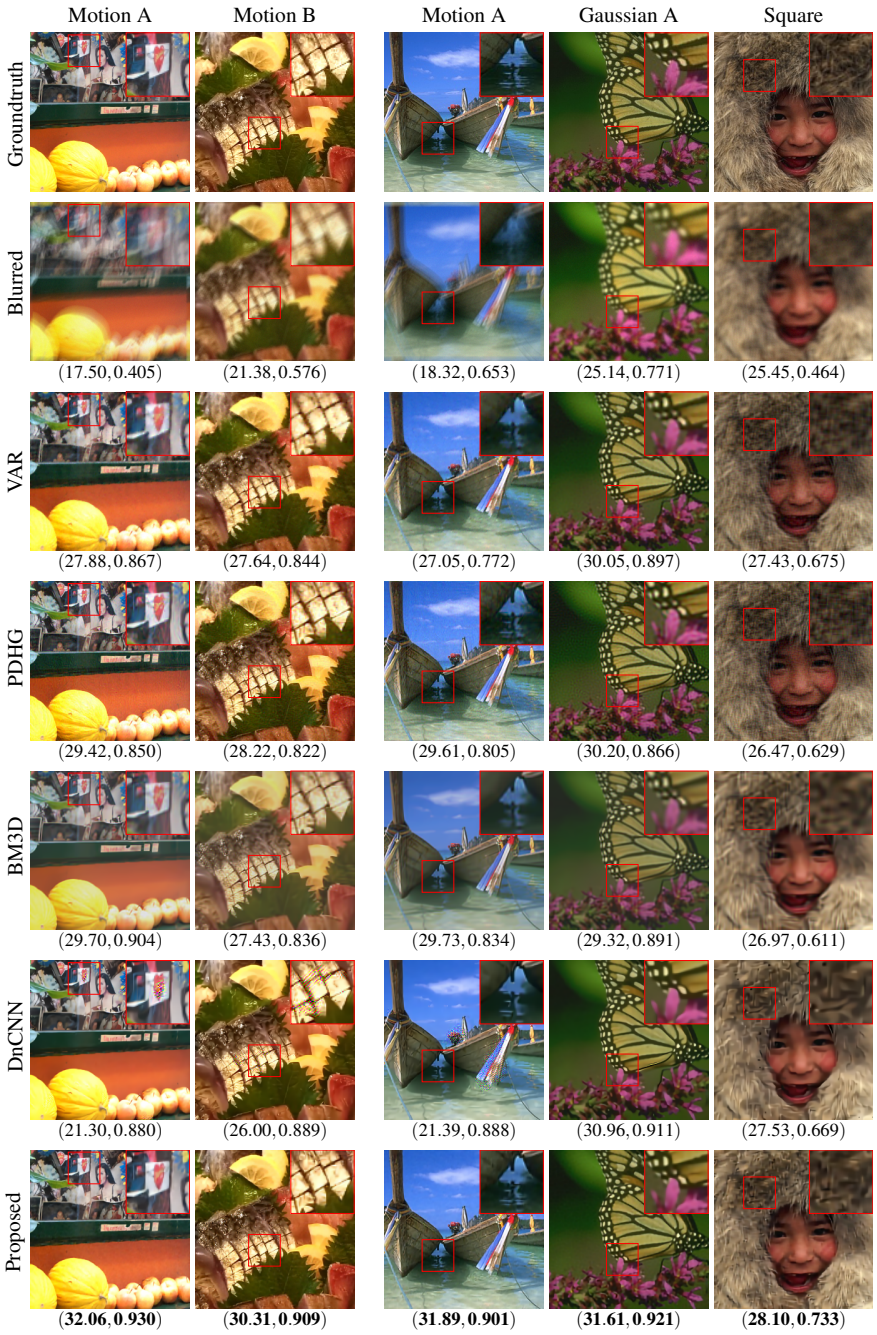


Fig. 10: Visual results on Flickr30 (first two columns) and BSD500 (last three columns) datasets for different methods (VAR, PDHG, PnP-FB with BM3D, PnP-FB with DnCNN and ours) for different deblurring problems with $\nu = 0.01$. Associated (PSNR, SSIM) values are indicated below each image, best values are highlighted in bold. Except PDHG, each algorithm is stopped after 1000 iterations.

quality over the other considered PnP methods.

Visual comparisons are provided in Figure 10 for the different approaches. These results show that our method also yields better visual results. The reconstructed images contain finer details and do not show the oversmoothed appearance of PnP-FB with DnCNN or slightly blurred aspect of PnP-FB with BM3D. Note that, in particular, thanks to its convergence, the proposed method shows a homogeneous performance over all images, unlike PnP-FB with DnCNN that may show some divergence effects (see the boat picture for Motion A, row (f)). One can observe that the improvement obtained with our approach are more noticeable on settings M. A and M. B than on G. A and S.

5. Conclusion. In this paper, we investigated the interplay between PnP algorithms and monotone operator theory, in order to propose a sound mathematical framework yielding both convergence guarantees and a good reconstruction quality in the context of computational imaging.

First, we established a universal approximation theorem for a wide range of MMOs, in particular the new class of stationary MMOs we have introduced. This theorem constitutes the theoretical backbone of our work by proving that the resolvents of these MMOs can be approximated by building nonexpansive NNs. Leveraging this result, we proposed to learn MMOs in a supervised manner for PnP algorithms. A main advantage of this approach is that it allows us to characterize their limit as a solution to a variational inclusion problem.

Second, we proposed a novel training loss to learn the resolvent of an MMO for high dimensional data, by imposing mild conditions on the underlying NN architecture. This loss uses information of the Jacobian of the NN, and can be optimized efficiently using existing training strategies. Finally, we demonstrated that the resulting PnP algorithms grounded on the FB scheme have good convergence properties when trained by leveraging the proposed Jacobian regularization. We showcased our method on an image deblurring problem and showed that the proposed PnP-FB algorithm outperforms both standard variational methods and state-of-the-art PnP algorithms. Interestingly, our results suggest that the optimal values for both the stepsize and training noise level can be determined automatically as functions of parameters of the data fidelity term, namely the Lipschitz constant μ of its gradient and the norm of the blurring kernel.

Note that the ability of approximating resolvents as we did is directly applicable to a much wider class of iterative algorithms than the forward-backward splitting [23]. In addition, we could consider a wider scope of applications than the restoration problems addressed in this work, such as image super-resolution or reconstruction, and future works also include the investigation of automatic methods for choosing λ.

REFERENCES

[1] A. Abdulaziz, A. Dabbech, and Y. Wiaux, *Wideband super-resolution imaging in radio interferometry via low rankness and joint average sparsity models (hypersara)*, Monthly Notices of the Royal Astronomical Society, 489 (2019), pp. 1230–1248.

[2] J. Adler and O. Öktem, *Learned primal-dual reconstruction*, IEEE Trans. on Medical Imaging, 37 (2018), pp. 1322–1332.

[3] R. Ahmad, C. A. Bouman, G. T. Buzzard, S. Chan, S. Liu, E. T. Reehorst, and P. Schniter, *Plug-and-play methods for magnetic resonance imaging: Using denoisers for image recovery*, IEEE Signal Process. Mag., 37 (2020), pp. 105–116.

[4] A. Alacaoglu, Y. Malitsky, and V. Cevher, *Convergence of adaptive algorithms for weakly convex constrained optimization*, arXiv preprint arXiv:2006.06650, (2020).

[5] C. Anil, J. Lucas, and R. Grosse, *Sorting out lipschitz function approximation*, in International Conference on Machine Learning, PMLR, 2019, pp. 291–301.
[6] S. Banert, A. Ringh, J. Adler, J. Karlsson, and O. Oktem, *Data-driven nonsmooth optimization*, SIAM J. on Optimization, 30 (2020), pp. 102–131.
[7] H. H. Bauschke and P. L. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*, Springer, 2017.
[8] C. Bertocchi, E. Chouzenoux, M.-C. Corbineau, J.-C. Pesquet, and M. Prato, *Deep unfolding of a proximal interior point method for image restoration*, Inverse Problems, 36 (2020), p. 034005.
[9] A. Bibi, B. Ghanem, V. Koltun, and R. Ranftl, *Deep layers as stochastic solvers*, in International Conference on Learning Representations, 2019.
[10] J. Birdi, A. Repetti, and Y. Wiaux, *Sparse interferometric stokes imaging under the polarization constraint (polarized sara)*, Monthly Notices of the Royal Astronomical Society, 478 (2018), pp. 4442–4463.
[11] J. Bolte and E. Pauwels, *Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning*, Mathematical Programming, (2020), pp. 1–33.
[12] K. Bredies, K. Kunisch, and T. Pock, *Total generalized variation*, SIAM J. on Imaging Sciences, 3 (2010), pp. 492–526.
[13] R. E. Carrillo, J. D. McEwen, D. Van De Ville, J.-P. Thiran, and Y. Wiaux, *Sparsity averaging for compressive imaging*, IEEE Signal Process. Lett., 20 (2013), pp. 591–594.
[14] G. H.-G. Chen and R. T. Rockafellar, *Convergence rates in forward-backward splitting*, SIAM J. Optim, 7 (1997), pp. 421–444.
[15] X. Chen, J. Liu, Z. Wang, and W. Yin, *Theoretical linear convergence of unfolded ista and its practical weights and thresholds*, in Advances in Neural Information Processing Systems, 2018, pp. 9061–9071.
[16] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, *Parseval networks: Improving robustness to adversarial examples*, in Proceedings of the 34th International Conference on Machine Learning, JMLR, 2017, pp. 854–863.
[17] R. Cohen, M. Elad, and P. Milanfar, *Regularization by denoising via fixed-point projection (RED-PRO)*, arXiv preprint arXiv:2008.00226, (2020).
[18] P. L. Combettes, *Monotone operator theory in convex optimization*, Math. Program., 170 (2018), pp. 177–206.
[19] P. L. Combettes and J.-C. Pesquet, *Image restoration subject to a total variation constraint*, IEEE Trans. Image Process., 13 (2004), pp. 1213–1222.
[20] P. L. Combettes and J.-C. Pesquet, *Proximal splitting methods in signal processing*, in Fixed-point algorithms for inverse problems in science and engineering, Springer, 2011, pp. 185–212.
[21] P. L. Combettes and J.-C. Pesquet, *Deep neural network structures solving variational inequalities*, Set-Valued and Variational Analysis, (2020), pp. 1–28.
[22] P. L. Combettes and J.-C. Pesquet, *Lipschitz certificates for layered network structures driven by averaged activation operators*, SIAM J. on Mathematics of Data Science, 2 (2020), pp. 529–557.
[23] P. L. Combettes and J. C. Pesquet, *Fixed point strategies in data science*, IEEE Trans. on Signal Processing, (2021), pp. 1–1, https://doi.org/10.1109/TSP.2021.3069677.
[24] P. L. Combettes and V. R. Wajs, *Signal recovery by proximal forward-backward splitting;*, Multiscale Model. Simul, 4 (2005), pp. 1168–1200.
[25] L. Condat, *A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms*, Journal of optimization theory and applications, 158 (2013), pp. 460–479.
[26] L. Condat, *Semi-local total variation for regularization of inverse problems*, in Proceedings of the IEEE European Signal Processing Conference, 2014, pp. 1806–1810.
[27] I. Daubechies, M. Defrise, and C. DeMol, *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint*, Comm. Pure Appl. Math, 57 (2004), pp. 1413–1457.
[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.
[29] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas, *Efficient and accurate estimation of Lipschitz constants for deep neural networks*, in Advances in Neural Information Processing Systems, 2019, pp. 11427–11438.
[30] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3, JHU press, 2013.
[31] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, *Improved*

training of wasserstein gans, in Advances in Neural Information Processing Systems, 2017, pp. 5767–5777.

[32] J. Hertrich, S. Neumayer, and G. Steidl, *Convolutional proximal neural networksand plug-and-play algorithms*, arXiv preprint arXiv:2011.02281, (2020).

[33] J. Hoffman, D. A. Roberts, and S. Yaida, *Robust learning with jacobian regularization*, arXiv preprint arXiv:1908.02729, (2019).

[34] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators*, Neural networks, 2 (1989), pp. 359–366.

[35] M. Jiu and N. Pustelnik, *A deep primal-dual proximal network for image restoration*, arXiv preprint arXiv:2007.00959, (2020).

[36] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in International Conference on Learning Representations, 2015.

[37] N. Komodakis and J.-C. Pesquet, *Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems*, IEEE Signal Process. Mag., 32 (2015), pp. 31–54.

[38] F. Latorre, P. Rolland, and V. Cevher, *Lipschitz constant estimation of neural networks via sparse polynomial optimization*, in International Conference on Learning Representations, 2020.

[39] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., *Photo-realistic single image super-resolution using a generative adversarial network*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4681–4690.

[40] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, p, (1993), pp. 861–867.

[41] A. Levin, Y. Weiss, F. Durand, and W. Freeman, *Understanding and evaluating blind deconvolution algorithms*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[42] D. G. Luenberger, Y. Ye, et al., *Linear and nonlinear programming, Fourth Edition*, vol. 228, Springer, 2016.

[43] Y. Mäkinen, L. Azzari, and A. Foi, *Collaborative filtering of correlated noise: Exact transform-domain variance for improved shrinkage and patch matching*, IEEE Trans. Image Process., 29 (2020), pp. 8339–8354.

[44] S. Mallat, *A wavelet tour of signal processing: the sparse way*, Academic press, 2008.

[45] D. Martin, C. Fowlkes, D. Tal, and J. Malik, *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*, in Proceedings of the IEEE International Conference on Computer Vision, vol. 2, 2001, pp. 416–423.

[46] T. Meinhardt, M. Moller, C. Hazirbas, and D. Cremers, *Learning proximal operators: Using denoising networks for regularizing inverse imaging problems*, in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1781–1790.

[47] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, *Spectral normalization for generative adversarial networks*, in International Conference on Learning Representations, 2018.

[48] J.-J. Moreau, *Proximité et dualité dans un espace hilbertien*, Bulletin de la Société mathématique de France, 93 (1965), pp. 273–299.

[49] M. C. Mukkamala and M. Hein, *Variants of rmsprop and adagrad with logarithmic regret bounds*, in International Conference on Machine Learning, JMLR, 2017, pp. 2545–2553.

[50] A. Neacsu, J.-C. Pesquet, and C. Burileanu, *Accuracy-robustness trade-off for positively weighted neural networks*, in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2020, pp. 8389–8393.

[51] H. Petzka, A. Fischer, and D. Lukovnikov, *On the regularization of wasserstein gans*, in International Conference on Learning Representations, 2018.

[52] R. Rockafellar, *Characterization of the subdifferentials of convex functions*, Pacific J. Math., 17 (1966), pp. 497–510.

[53] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D: nonlinear phenomena, 60 (1992), pp. 259–268.

[54] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, *Plug-and-play methods provably converge with properly trained denoisers*, in International Conference on Machine Learning, PMLR, 2019, pp. 5546–5557.

[55] K. Scaman and A. Virmaux, *Lipschitz regularity of deep neural networks: Analysis and efficient estimation*, Adv. Neural Inform. Process. Syst, 31 (2018), pp. 3839–3848.

[56] H. Sedghi, V. Gupta, and P. M. Long, *The singular values of convolutional layers*, arXiv preprint arXiv:1805.10408, (2018).

[57] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, *Intriguing properties of neural networks*, in International Conference on Learning Representations, 2014.
[58] M. Terris, A. Repetti, J.-C. Pesquet, and Y. Wiaux, *Building firmly nonexpansive convolutional neural networks*, in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2020, pp. 8658–8662.
[59] A. Virmaux and K. Scaman, *Lipschitz regularity of deep neural networks: analysis and efficient estimation*, in Advances in Neural Information Processing Systems, 2018, pp. 3835–3844.
[60] B. C. Vũ, *A splitting algorithm for dual monotone inclusions involving cocoercive operators*, Advances in Computational Mathematics, 38 (2013), pp. 667–681.
[61] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, *Image quality assessment: from error visibility to structural similarity*, IEEE Trans. Image Process., 13 (2004), pp. 600–612.
[62] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang, *Improving the improved training of wasserstein gans: A consistency term and its dual effect*, in International Conference on Learning Representations, 2018.
[63] L. Xu, J. S. Ren, C. Liu, and J. Jia, *Deep convolutional neural network for image deconvolution*, in Advances in Neural Information Processing Systems, 2014, pp. 1790–1798.
[64] X. Xu, Y. Sun, J. Liu, B. Wohlberg, and U. S. Kamilov, *Provable convergence of plug-and-play priors with MMSE denoisers*, arXiv preprint arXiv:2005.07685, (2020).
[65] Y. Yoshida and T. Miyato, *Spectral norm regularization for improving the generalizability of deep learning*, arXiv preprint arXiv:1705.10941, (2017).
[66] J. Zhang and B. Ghanem, *ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1828–1837.
[67] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, *Plug-and-play image restoration with deep denoiser prior*, arXiv preprint arXiv:2008.13751, (2020).
[68] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, *Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising*, IEEE Trans. Image Process., 26 (2017), pp. 3142–3155.
[69] K. Zhang, W. Zuo, S. Gu, and L. Zhang, *Learning deep CNN denoiser prior for image restoration*, in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2017, pp. 3929–3938.
[70] K. Zhang, W. Zuo, and L. Zhang, *Learning a single convolutional super-resolution network for multiple degradations*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3262–3271.
[71] K. Zhang, W. Zuo, and L. Zhang, *Deep plug-and-play super-resolution for arbitrary blur kernels*, in IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 1671–1681.

Appendix A. Nonexpansive networks: additional results.

Based on the results in [22], useful sufficient conditions for a NN to be nonexpansive are given below:

Proposition A.1. *Let Q be a feedforward NN as defined in Model 2.2. Assume that, for every $m \in \{1,\dots,M\}$, R_m is α_m-averaged with $\alpha_m \in [0,1]$. Then Q is nonexpansive if one of the following conditions holds:*

(i) $\|W_1\| \cdots \|W_M\| \leqslant 1$;

(ii) *for every $m \in \{1,\dots,M-1\}$, $\mathcal{H}_m = \mathbb{R}^{K_m}$ with $K_m \in \mathbb{N} \setminus \{0\}$, R_m is a separable activation operator, in the sense that there exist real-valued one-variable functions $(\rho_{m,k})_{1\leqslant k\leqslant K_m}$ such that, for every $x = (\xi_k)_{1\leqslant k\leqslant K_m} \in \mathcal{H}_m$, $R_m(x) = (\rho_{m,k}(\xi_k))_{1\leqslant k\leqslant K_m}$, and*

$$(\forall \Lambda_1 \in \mathcal{D}_{1,\{1-2\alpha_1,1\}}) \dots (\forall \Lambda_{M-1} \in \mathcal{D}_{M-1,\{1-2\alpha_{M-1},1\}}) \quad \|W_M \Lambda_{M-1} \cdots \Lambda_1 W_1\| \leqslant 1, \tag{A.1}$$

where, for every $m \in \{1,\dots,M-1\}$, $\mathcal{D}_{m,\{1-2\alpha_m,1\}}$ denotes the set of diagonal $K_m \times K_m$ matrices with diagonal terms equal either to $1-2\alpha_m$ or 1;

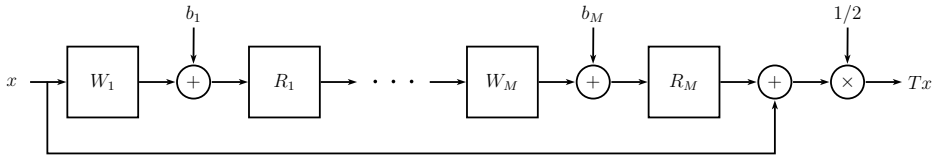


Fig. 11: Neural network modelling the resolvent of a maximally monotone operator. The weight operators $(W_m)_{1\leqslant m\leqslant M}$ have to be set according to the conditions provided in Proposition A.1.

(iii) *for every* $m \in \{1,\ldots,M\}$, $\mathcal{H}_m = \mathbb{R}^{K_m}$ *with* $K_m \in \mathbb{N}\backslash\{0\}$ *and* W_m *is a matrix with nonnegative elements,* $(R_m)_{1\leqslant m\leqslant M-1}$ *are separable activation operators, and*

$$\|W_M \cdots W_1\| \leqslant 1. \tag{A.2}$$

Note that the α-averageness assumption on $(R_m)_{1\leqslant m\leqslant M-1}$ means that, for every $m \in \{1,\ldots,M-1\}$, there exists a nonexpansive operator $\widetilde{R}_m\colon \mathcal{H}_m \to \mathcal{H}_m$ such that $R_m = (1-\alpha_m)\,\mathrm{Id} + \alpha_m \widetilde{R}_m$. Actually, most of the activation operators employed in neural networks (ReLU, leaky ReLU, sigmoid, softmax,...) satisfy this assumption with $\alpha_m = 1/2$ [21]. A few others like the sorting operator used in max-pooling correspond to a value of the constant α_m larger than $1/2$ [22]. It is also worth mentioning that, although Condition (i) in Proposition A.1 is obviously the simplest one, it is usually quite restrictive. If equation (A.2) is weaker than Condition (i), Condition (iii) requires yet the network weights to be nonnegative as shown in [22].

By summarizing the results of the previous section, Figure 11 shows a feedforward NN architecture for MMOs, for which Proposition A.1 can be applied. It can be noticed that (2.1) induces the presence of a skip connection in the global structure.

Appendix B. Stationnary MMOs: additional proofs.

Proof of Example 2.7. As B is an MMO and U is surjective, U^*BU is an MMO [7, Corollary 25.6]. We are thus guaranteed that $\mathrm{ran}\,(\mathrm{Id}+A) = \mathcal{H}$ [7, Theorem 21.1]. For every $k \in \{1,\ldots,K\}$, let

$$D_k\colon \mathcal{H} \to \mathcal{H}_k\colon (x^{(\ell)})_{1\leqslant\ell\leqslant K} \mapsto x^{(k)} \tag{B.1}$$

$$\Pi_k = D_k U. \tag{B.2}$$

It can be noticed that

$$\sum_{k=1}^{K} \Pi_k^*\Pi_k = U^*U = \mathrm{Id}\,. \tag{B.3}$$

Let $(p,q) \in \mathcal{H}^2$. Every $(p',q') \in A(p)\times A(q)$ is such

$$p' = U^*r \text{ and } q' = U^*s, \tag{B.4}$$

where $r \in B(Up)$ and $s \in B(Uq)$. Using (B.2) and (B.4) yield, for every $k \in \{1,\ldots,K\}$,

$$\langle \Pi_k(p-q) \mid \Pi_k(p'-q')\rangle = \langle D_kUp - D_kUq \mid D_kr - D_ks\rangle. \tag{B.5}$$

Because of the separable form of B, $D_k r \in B_k(D_k U p)$ and $D_k s \in B_k(D_k U q)$. It then follows from (B.5) and the monotonicity of B_k that

$$\langle \Pi_k(p-q) \mid \Pi_k(p'-q') \rangle \geqslant 0. \tag{B.6}$$

By invoking Proposition 2.6(ii), we conclude that A is a stationary MMO. ∎

Proof of Example 2.8. This corresponds to the special case of Example 2.7 when, for every $k \in \{1,\ldots,K\}$, $\mathcal{H}_k = \mathbb{R}$ (see [7, Theorem 16.47,Corollary 22.23]). ∎

Proof of Example 2.9. If $B+B^*$ is nonnegative, B, hence A, are maximally monotone and $J_A = J_B(\cdot - c)$ is firmly nonexpansive. As a consequence, the reflected resolvent of B, given by $Q = 2(\mathrm{Id}+B)^{-1} - \mathrm{Id}$, is nonexpansive. For every $k \in \{1,\ldots,K\}$, let D_k be the decimation operator defined in (B.1) and let

$$\Pi_k = D_k \tag{B.7}$$

$$\Omega_k = Q^* D_k^* D_k Q \tag{B.8}$$

Π_k satisfies (2.6) and, since

$$\Big\| \sum_{k=1}^{K} \Omega_k \Big\| = \|Q^* Q\| = \|Q\|^2 \leqslant 1, \tag{B.9}$$

(2.7) is also satisfied. In addition, for every $(x,y) \in \mathcal{H}^2$ and, for every $k \in \{1,\ldots,K\}$, we have

$$\begin{aligned} &\|\Pi_k\big(2J_A(x) - x - 2J_A(y) + y\big)\|^2 \\ &= \|\Pi_k\big(2J_B(x-c) - x + c - 2J_B(y-c) + y - c\big)\|^2 \\ &= \langle x - y \mid \Omega_k(x-y) \rangle, \end{aligned} \tag{B.10}$$

which shows that A is a stationary MMO.
Note finally that, if B is skewed or cocoercive linear operator, then $B + B^*$ is nonnegative. ∎

Proof of Example 2.10. The resolvent of A^{-1} is given by $J_{A^{-1}} = \mathrm{Id} - J_A$. In addition, since A is stationary, there exist bounded linear operators $(\Pi_k)_{1\leqslant k\leqslant K}$ and self-adjoint operators $(\Omega_k)_{1\leqslant k\leqslant K}$ satisfying (2.5)-(2.7). For every $k \in \{1,\ldots,K\}$, we have then, for every $(x,y) \in \mathcal{H}^2$,

$$\begin{aligned} \|\Pi_k\big(2J_{A^{-1}}(x) - x - 2J_{A^{-1}}(y) + y\big)\|^2 &= \|\Pi_k\big(2J_A(y) - y - 2J_A(x) + x\big)\|^2 \\ &\leqslant \langle y - x \mid \Omega_k(y-x) \rangle. \end{aligned} \tag{B.11}$$

∎

Proof of Example 2.11. $B = \rho A(\cdot/\rho)$ is maximally monotone and its resolvent reads $J_B = \rho J_A(\cdot/\rho)$ [7, Corollary 23.26]. Using the same notation as previously, for every $k \in \{1,\ldots,K\}$ and for every $(x,y) \in \mathcal{H}^2$,

$$\begin{aligned} \|\Pi_k\big(2J_B(x) - x - 2J_B(y) + y\big)\|^2 &= \rho^2 \left\| \Pi_k\Big(2J_A\Big(\frac{x}{\rho}\Big) - \frac{x}{\rho} - 2J_A\Big(\frac{y}{\rho}\Big) + \frac{y}{\rho}\Big) \right\|^2 \\ &\leqslant \langle y - x \mid \Omega_k(y-x) \rangle. \end{aligned} \tag{B.12}$$

∎

Appendix C. Power iterative method. The power iterative algorithm allows the spectral norm of a matrix B to be computed efficiently [30]. In our case, we apply

this algorithm with $B = \nabla Q(x)$ at some point x. Notice that we do not need to compute or store the full Jacobian: in practice, at each iteration of this algorithm, we just need to apply $\nabla Q(x)$ to some vector and apply $\nabla Q(x)^\top$ to another vector. These products can be computed by automatic differentiation.
Due to memory limitations, all our experiments are performed with 5 iterations of the power method. We nevertheless observe in practice that this is sufficient to ensure convergence of the PnP-FB algorithm.