

# Parameters Estimation via Unrolling for the Preconditioned Nested Primal Dual Algorithm

Marco Donatelli and Giuseppe Scarlato(✉)

Università dell'Insubria, 22100 Como, CO, Italy
{marco.donatelli,gscarlato}@uninsubria.it

**Abstract.** Variational methods are widely used in inverse problems. Through regularization, they are able to incorporate prior knowledge on the solution and, therefore, to achieve good reconstruction quality. However, those methods typically require careful tuning of the parameters involved in the optimization problem and the method itself, which can be time-consuming and computationally expensive. On the other hand, deep learning methods allow for automatic learning through supervised training at the cost of interpretability, generalization, and inclusion of constraints and physical models. Unrolling techniques have emerged as a bridge between these two approaches, combining the interpretability of deterministic methods with the learning capabilities of deep learning to automatically estimate parameters.

In this work, we propose an unrolling architecture for the *Preconditioned Nested Primal Dual* (PNPD) algorithm. This allows to automatically estimate the parameters involved in the optimization problem and in the numerical method, without the need for manual tuning. Numerical results show the effectiveness of the proposed approach in terms of reconstruction quality and parameter estimation.

**Keywords:** Proximal algorithms · Preconditioning · Unrolling · Deep unfolding · Neural network · Image restoration · Regularization

## 1 Introduction

We consider a linear inverse problem of the form

$$b^\delta = Au + \eta_\delta = b + \eta_\delta, \tag{1}$$

where $A \in \mathbb{R}^{s\times d}$ is a linear operator, $b^\delta \in \mathbb{R}^s$ is the observed data corrupted by noise $\eta_\delta \in \mathbb{R}^s$, and $u \in \mathbb{R}^d$ is the unknown signal we want to recover. This kind of problem arises in many applications, such as image restoration, computed tomography, and many others [5,9]. In particular, we will focus on the case of image deblurring, where $A$ is a convolution operator, $u$ is the original image, and $b^\delta$ is the blurred and noisy observation. The ill-posed nature of the operator

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16763, pp. 156–169, 2027.
https://doi.org/10.1007/978-3-032-30530-5_10

$A$ and the presence of noise makes the problem challenging, and regularization is typically required to achieve a useful reconstruction. A common variational approach to solve Eq. (1) is to consider the following optimization problem

$$\underset{u\in\mathbb{R}^d}{\operatorname{argmin}} \ \frac{1}{2}\|Au - b^\delta\|^2 + h(Wu) = \underset{u\in\mathbb{R}^d}{\operatorname{argmin}} \ f(u) + h(Wu), \tag{2}$$

where, $\|.\|$ denotes the Euclidean norm, $f(u)$ is the data fidelity term, and $h(Wu)$ is the regularization term, with $W \in \mathbb{R}^{d'\times d}$ being a linear operator and $h : \mathbb{R}^{d'} \to \mathbb{R} \cup \{\infty\}$ is a convex and possibly non-smooth function. The balance between the data fidelity and the regularization is typically controlled by a regularization parameter $\lambda > 0$, which here is implicitly included in the definition of $h$. To solve Eq. (2), since $h \circ W$ is non-smooth, a common approach is to leverage the proximity operator of $g = h \circ W$:

$$\operatorname{prox}_g(x) = \underset{u\in\mathbb{R}^d}{\operatorname{argmin}} \ \frac{1}{2}\|u - x\|^2 + g(u).$$

This leads to the use of proximal gradient methods, which are iterative algorithms that alternate between a gradient step on the smooth term $f$ and a proximal step on the convex term $h \circ W$, specifically

$$\begin{cases} u_{n+\frac{1}{2}} = u_n - \alpha\nabla f(u_n) \\ u_{n+1} = \operatorname{prox}_{\alpha h\circ W}(u_{n+\frac{1}{2}}) \end{cases}$$

where $\alpha > 0$ is the step size. Common examples of proximal algorithms include the *Iterative Shrinkage Thresholding Algorithm* (ISTA) [14], the *Fast Iterative Shrinkage Thresholding Algorithm* (FISTA) [4], the *iPiano* algorithm [21], *VMILAn* [7], and many others (for a review, see, for example, [13]). An issue that can occur with this approach is that the proximity operator of $h \circ W$ might not be known in closed form. This is the case, for example, of the Total Variation (TV) regularization. To address this, inexact proximal algorithms have been proposed, which, instead of computing the proximal step exactly, approximate it by solving a subproblem iteratively [12,22]. In particular, we are interested in the *Nested Primal Dual* (NPD) algorithm [8,10], which approximates $\operatorname{prox}_{\alpha h\circ W}$ via a dual sequence that depends on the proximity operator of the Fenchel conjugate of $h$, namely $\operatorname{prox}_{\alpha h^*}$, which in many cases is known in closed form, as in the case of TV regularization. Furthermore, by using a preconditioning strategy, the convergence of the NPD algorithm can be improved. This leads to the *Nested Primal Dual Iterated Tikhonov* (NPDIT) method [1] and the *Preconditioned Nested Primal Dual* (PNPD) algorithm [2], which is the focus of this work.

Variational methods, such as the PNPD algorithm, often depend on several parameters, including the regularization parameter $\lambda$, the step sizes of the algorithm, and the parameters of the preconditioner. Choosing these parameters is challenging, as they interact in nontrivial ways and can significantly affect both the reconstruction quality and the convergence of the algorithm. These parameters are typically chosen manually, but automatic parameter estimation is possible using heuristic rules, such as the discrepancy principle [23], the L-curve

method [18], and generalized cross-validation [16]. These methods, however, can be computationally expensive and may not always provide optimal results. In recent years, deep learning methods have received a lot of attention in the field of inverse problems. They provide a flexible and powerful tool to learn complex relationships from data. A straightforward approach to leverage neural networks for solving inverse problems is to unroll the iterations of a deterministic algorithm, such as the PNPD algorithm, and to learn the parameters of the algorithm through supervised training [20]. With this approach, the method itself becomes a part of a neural network. In this work, we propose an unrolling architecture for the PNPD algorithm, which allows us to automatically estimate the parameters involved in the optimization problem and in the numerical method, without the need for manual tuning.

This paper is organized as follows: Sect. 2 introduces the PNPD algorithm. Section 3 describes the proposed unrolling architecture for the PNPD algorithm and the loss function used for training. Section 4 presents the numerical results for the image deblurring problem. Finally, Sect. 5 concludes the paper and discusses future work.

## 2 Preconditioned Nested Primal Dual Algorithm

In this section, we introduce the NPD algorithm and its preconditioned version, the PNPD algorithm. As shown in [2], the NPDIT method proposed in [1] can also be seen as a preconditioned version of NPD. The difference between the two methods resides in how the preconditioning is applied. The PNPD algorithm corresponds to a left preconditioning, while the NPDIT method corresponds to a right preconditioning strategy. In this work, we focus on the PNPD algorithm, but the proposed unrolling architecture can be adapted to the NPDIT method as well.

### 2.1 Nested Primal-Dual Method (NPD)

The NPD algorithm is an inexact proximal gradient algorithm, and its iterative scheme can be summarized as follows:

$$\begin{cases} \bar{u}_n = u_n + \gamma_n(u_n - u_{n-1}), \\ u_{n+\frac{1}{2}} = \bar{u}_n - \alpha_n \nabla f(\bar{u}_n), \\ u_{n+1} \approx \operatorname{prox}_{\alpha_n h \circ W}(u_{n+\frac{1}{2}}), \end{cases} \tag{3}$$

where $\approx$ indicates an approximation of the proximal-gradient point. Here, $\bar{u}_n$ is usually referred to as the extrapolated point, $\gamma_n \in [0, 1]$ is the extrapolation parameter, and $\alpha_n > 0$ is the step length along the descent direction $-\nabla f(\bar{u}_n)$. For the iterative scheme in Eq. (3), a way to approximate $\operatorname{prox}_{\alpha_n h \circ W}$ is required. This is achieved by computing a dual sequence which, in turn, is used to approximate $\operatorname{prox}_{\alpha_n h \circ W}$. To introduce the dual sequence, we state the following result.

**Lemma 1.** *[8, Theorem 1] Suppose that $h : \mathbb{R}^{d'} \to \mathbb{R} \cup \{\infty\}$ is a proper convex lower semicontinuous function and $W \in \mathbb{R}^{d' \times d}$ is such that there exists $u_0 \in \mathbb{R}^d$ with $Wu_0 \in \operatorname{relint}(\operatorname{dom}(h))$. Given $a \in \mathbb{R}^d$, $\alpha > 0$, $0 < \beta < 2/\|W\|^2$, and $v^0 \in \mathbb{R}^{d'}$, define the sequence*

$$v^{k+1} = \operatorname{prox}_{\beta\alpha^{-1}h^*}(v^k + \beta\alpha^{-1}W(a - \alpha W^T v^k)), \quad \forall k \geq 0. \tag{4}$$

*Then there exists $\hat{v} \in \mathbb{R}^{d'}$ such that:*

*(i)* $\lim_{k\to\infty} v^k = \hat{v}$;
*(ii)* $\operatorname{prox}_{\alpha h \circ W}(a) = a - \alpha W^T \hat{v}$.

Note that by combining items $(i)$ and $(ii)$ of Lemma 1 with Eq. (4), it holds

$$\operatorname{prox}_{\alpha h\circ W}(a) = \lim_{k\to\infty} a - \alpha W^T \operatorname{prox}_{\beta\alpha^{-1}h^*}(v^k + \beta\alpha^{-1}W(a - \alpha W^T v^k)).$$

Using this relationship, it is possible to compute an approximation of $\operatorname{prox}_{\alpha h\circ W}(a)$ by a finite number of steps of the dual sequence (Eq. (4)), provided that the operator $\operatorname{prox}_{\beta\alpha^{-1}h^*}$ and the matrix-vector product with the linear operators $W$ and $W^T$ are easily computable. Let $k_{\max}$ be the maximum number of inner iterations used to compute the dual sequence Eq. (4), and $L$ be the Lipschitz constant of the gradient of $f$, the resulting NPD algorithm is summarized in Algorithm 1.

**Algorithm 1.** NPD

Choose $u_0 \in \mathbb{R}^d$, $0 < \alpha < \frac{2}{L}$, $0 < \beta < \frac{2}{\|W\|^2}$, $k_{\max} \in \mathbb{N}$, $\{\gamma_n\}_{n\in\mathbb{N}} \subseteq \mathbb{R}_{\geq 0}$.
$u_{-1} = u_0$
$v_0^0 = 0$
**for** $n = 0, 1, \ldots$ **do**
  $\bar{u}_n = u_n + \gamma_n(u_n - u_{n-1})$
  $u_{n+\frac{1}{2}} = \bar{u}_n - \alpha\nabla f(\bar{u}_n)$
  **for** $k = 0, 1, \ldots, k_{\max} - 1$ **do**
    $u_n^k = u_{n+\frac{1}{2}} - \alpha W^T v_n^k$
    $v_n^{k+1} = \operatorname{prox}_{\beta\alpha^{-1}h^*}(v_n^k + \beta\alpha^{-1}Wu_n^k)$
  **end for**
  $v_{n+1}^0 = v_n^{k_{\max}}$
  $u_n^{k_{\max}} = u_{n+\frac{1}{2}} - \alpha W^T v_n^{k_{\max}}$
  $u_{n+1} = \frac{1}{k_{\max}} \sum_{j=1}^{k_{\max}} u_n^j$
**end for**

## 2.2 Preconditioned NPD (PNPD)

The main idea of PNPD is to accelerate the convergence of NPD by replacing the gradient descent step inside the proximal evaluation with a preconditioned

iteration. This is achieved by modifying the data fidelity norm with the one induced by an appropriate positive definite preconditioner $S \in \mathbb{R}^{s\times s}$.

Similarly to Eq. (2), given $S \in \mathbb{R}^{s\times s}$ symmetric positive definite (s.p.d.), we consider the optimization problem

$$\operatorname*{argmin}_{u\in\mathbb{R}^d} f_S(u) + h(Wu) = \operatorname*{argmin}_{u\in\mathbb{R}^d} \frac{1}{2}\|Au - b^\delta\|^2_{S^{-1}} + h(Wu). \tag{5}$$

In the data fidelity term

$$f_S(u) = \frac{1}{2}\|S^{-\frac{1}{2}}(Au - b^\delta)\|^2, \tag{6}$$

the linear operator $S^{\frac{1}{2}}$ can be interpreted as a left preconditioner for the linear system $Ax = b^\delta$. If we further assume that there exists $P \in \mathbb{R}^{d\times d}$ s.p.d. such that

$$P^{-1}A^T = A^T S^{-1}, \tag{7}$$

then it holds

$$\nabla f_S(u) = A^T S^{-1}(Au - b^\delta) = P^{-1}\nabla f(u), \tag{8}$$

where $f(u) = \frac{1}{2}\|Au - b^\delta\|^2$ is as in Eq. (2).

Therefore, applying the NPD algorithm to problem (5), under the Assumption (7), we obtain the iterative scheme of PNPD, defined as

$$\begin{cases} \bar{u}_n = u_n + \gamma_n(u_n - u_{n-1}) \\ u_{n+\frac{1}{2}} = \bar{u}_n - \alpha_n P^{-1}\nabla f(\bar{u}_n) \\ u_{n+1} \approx \operatorname{prox}_{\alpha_n h\circ W}(u_{n+\frac{1}{2}}). \end{cases} \tag{9}$$

A reasonable choice for the preconditioner $P$ and the associated matrix $S$, that satisfies the condition (7), is

$$P = A^T A + \nu I, \qquad S = AA^T + \nu I, \tag{10}$$

with $\nu > 0$. This is inspired by the iterated Tikhonov method, which coincides with the Levenberg–Marquardt method when applied to linear problems [15].

Algorithm 2 reports the pseudocode of the PNPD method, where we denote by $L_S$ the Lipschitz constant of $\nabla f_S$, which depends on the choice of the preconditioner $S$, i.e., $P$, due to (7). In particular, thanks to Eq. (8), we have

$$\|\nabla f_S(x) - \nabla f_S(y)\| = \|P^{-1}\nabla f(x) - P^{-1}\nabla f(y)\| \le \|P^{-1}A^T A\|\|x - y\|,$$

and it holds

$$L_S = \|P^{-1}A^T A\| = \|A^T S^{-1} A\|.$$

**Algorithm 2.** PNPD

Choose $u_0 \in \mathbb{R}^d$, $P \in \mathbb{R}^{d\times d}$ s.p.d, $0 < \alpha < \frac{2}{L_S}$, $0 < \beta < \frac{2}{\|W\|^2}$, $k_{\max} \in \mathbb{N}$, $\{\gamma_n\}_{n\in\mathbb{N}} \subseteq \mathbb{R}_{\geq 0}$.
$u_{-1} = u_0$
$v_0^0 = 0$
**for** $n = 0, 1, \ldots$ **do**
  $\bar{u}_n = u_n + \gamma_n(u_n - u_{n-1})$
  $u_{n+\frac{1}{2}} = \bar{u}_n - \alpha P^{-1}\nabla f(\bar{u}_n)$
  **for** $k = 0, 1, \ldots, k_{\max} - 1$ **do**
    $u_n^k = u_{n+\frac{1}{2}} - \alpha W^T v_n^k$
    $v_n^{k+1} = \mathrm{prox}_{\beta\alpha^{-1}h^*}(v_n^k + \beta\alpha^{-1} W u_n^k)$
  **end for**
  $v_{n+1}^0 = v_n^{k_{\max}}$
  $u_n^{k_{\max}} = u_{n+\frac{1}{2}} - \alpha W^T v_n^{k_{\max}}$
  $u_{n+1} = \frac{1}{k_{\max}} \sum_{j=1}^{k_{\max}} u_n^j$
**end for**

## 3 Unrolling Architecture

In this section, we describe the proposed unrolling architecture for the PNPD algorithm and the loss function used for training. The main idea of unrolling is to truncate the iterations of an algorithm to a fixed number $N$. Then, the resulting truncated algorithm is interpreted as a neural network, where the parameters of the algorithm are learned through supervised training (see [20]). A further step is to not just set the parameters of the algorithm as learnable, but to obtain them as the output of a predictor network, which takes as input some information on the current state of the algorithm, such as the current iterate and the observed data. The resulting method will therefore estimate the parameters adaptively, based on the given information, being able, for example, to infer different parameters for different noise levels or different types of images. This is, for example, the approach adopted in [6] for a proximal interior point algorithm, and is the same that we adopt in this work. In particular, let us consider a single iteration of PNPD (Algorithm 2) for which the preconditioner $P$ is defined as in Eq. (10). In this case, one step of the algorithm depends on the choice of the parameters $\lambda, \gamma, \alpha, \beta$, and $\nu$, which are respectively the regularization, the extrapolation, the step size, the dual step size, and the preconditioner parameters. For simplicity, we will not consider the extrapolation step, i.e., we will set $\gamma = 0$. For each parameter $\theta \in \{\lambda, \alpha, \beta, \nu\}$ and for each iteration $i = 0, \ldots, N-1$, we have a convolutional predictor network

$$\mathcal{N}_{\theta_i}(b^\delta, u_i) = \theta_i,$$

which takes as input the observed data $b^\delta$ and the current iterate $u_i$, and outputs the parameter $\theta_i$ to be used in the $i$-th iteration of the algorithm. These predictor networks are then composed with the iterations of the PNPD algorithm to obtain a single iteration of the unrolling architecture, which we denote by $\mathcal{M}_i$ and which

is depicted in Fig. 1. Finally, by composing $N$ times the base network $\mathcal{M}_i$, we obtain the global unrolling architecture, which is depicted in Fig. 2.

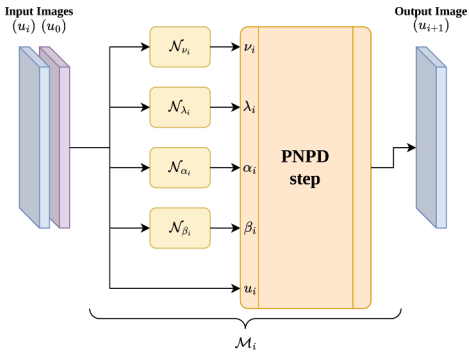


**Fig. 1.** Single iteration of the unrolling architecture $\mathcal{M}_i$ composed by the predictors for each parameter $\mathcal{N}_{\nu_i}, \mathcal{N}_{\lambda_i}, \mathcal{N}_{\alpha_i}, \mathcal{N}_{\beta_i}$ and by one step of the PNPD algorithm.

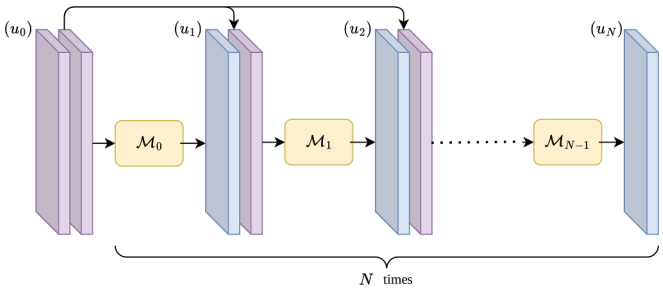


**Fig. 2.** Global unrolling architecture obtained by composing the base network $\mathcal{M}_i$ $N$ times.

If the proposed architecture is used as described above for inference, there is no natural way to extend the number of iterations beyond $N$. For this reason, we set

$$\mathcal{N}_{\theta_i} = \mathcal{N}_{\theta_0}, \quad \forall i = 1, \ldots, N-1.$$

Thus, we force the predictor networks to learn a mapping that is independent of the iteration, based only on the current iterate and on the observed data, which allows us to naturally extend the number of iterations at inference time.

Since each $\mathcal{M}_i$ represents one iteration of the PNPD algorithm, all the intermediate iterates $u_0, u_1, \ldots, u_N$ are available as output of the unrolling architecture. This can be leveraged in the design of the loss function used for training. In

particular, the loss function is defined as a weighted average of the loss computed on each intermediate iterate, i.e.,

$$\mathcal{L}(u_0, \dots, u_{N-1}, u) = \frac{\sum_{i=0}^{N-1} w_i \, \ell(u_i, u)}{\sum_{i=0}^{N-1} w_i}, \tag{11}$$

where $w_i$ is the weight associated with the $i$-th iterate, $\ell$ is a loss function, such as the mean squared error or the structural similarity index measure (SSIM), and $u$ is the ground truth signal. This way of defining the loss is different from the one typically used in unrolling, where the loss is computed only on the final iterate (see, for example, [6] and [17]). The proposed approach aims to reduce the vanishing gradient problem (see [19]), which occurs when training neural networks with a large number of layers via backpropagation. By computing the loss on each iterate, this problem is attenuated, since the gradient can be back-propagated from intermediate iterates, and not only from the final one. Furthermore, this way of defining the loss stabilizes the training using information that would otherwise be ignored. By selecting appropriate weights $w_i$, it is possible to balance between prioritizing the final reconstruction and achieving good reconstruction quality at earlier iterations. This makes the training landscape more favorable for optimization and makes it easier to avoid local minima [11].

## 4 Numerical Results

In this section, we test the proposed unrolling architecture (see Sect. 3) on the image deblurring problem with TV regularization, which corresponds to the optimization problem

$$\operatorname*{argmin}_{u \in \mathbb{R}^d} \frac{1}{2} \|Au - b^\delta\|^2 + \lambda \, \mathrm{TV}(u),$$

where $A \in \mathbb{R}^{d \times d}$ is a blur operator with periodic boundary conditions, $b^\delta \in \mathbb{R}^d$ is the observed blurred and noisy image, $\mathrm{TV}(u)$ is the total variation defined as

$$\mathrm{TV}(u) = \sum_{i=1}^{d} \|(\nabla u)_i\|_2,$$

and $\lambda > 0$ is a regularization parameter.

The algorithm and the tests are implemented in Python 3.14 using PyTorch 2.10. The code is available at a public git repository[1]. The dataset used for training and testing is the BSD500 dataset [3], which consists of 500 natural images subdivided into a training set of 200 images, a validation set of 100 images, and a test set of 200 images. The observed images $b^\delta$ are generated by firstly taking a random crop of size $256 \times 256$ from an image of the dataset,

[1] https://github.com/Giuseppe499/PNPD_unrolling.

then applying a Gaussian blur with standard deviation $\sigma = 2$ pixels, and finally adding white Gaussian noise such that $\|\eta_\delta\| = 0.01\|b\|$.

The predictor networks for the parameters $\lambda, \alpha, \beta$, and $\nu$ are convolutional neural networks with 2 convolutional layers. Both layers have kernel size 5, the first layer has 6 filters and the second layer has 32 filters. Each convolutional layer is followed by a ReLU activation function, and a max pooling layer with kernel size 2 and stride 2. After the convolutional layers, the output is flattened and passed through a fully connected network with a hidden layer of 128 neurons and a ReLU activation function, and an output layer with a single neuron which outputs the parameter to be used in the $i$-th iteration of the algorithm.

The loss function is defined as in Eq. (11) with

$$\ell(u_i, u) = -\,\mathrm{SSIM}(u_i, u),$$

where SSIM is the structural similarity index measure, and with the weights $w_i$ defined as

$$w_i = \frac{1}{N - i},$$

thus the last iterations are more relevant than the earlier ones. The number of steps of the unrolling architecture was set to $N = 18$ for training. The training was performed for 50 epochs using the Adam optimizer with a learning rate of $10^{-4}$ and a batch size of 8. The best performing model on the validation set was selected for testing.

Figure 3 shows the reconstruction of some elements of the test set after 100 iterations of the proposed method. We note that, as described in Sect. 3, the number of iterations can be naturally extended beyond $N$ at inference time, since the predictor networks are shared across iterations. It is also possible to observe that the SSIM metric for the reconstruction is higher than that of the observed image for all the shown examples, without the need for manual tuning of the parameters of the algorithm.

Figure 4 shows the evolution of the parameters and of the SSIM at inference time as a function of the iterations for the same examples shown in Fig. 3. We observe that for all the learned parameters and for a fixed image, the predictor networks tend to output almost constant values across iterations. By varying the image, we observe that the predictor networks tend to always use a step size around $\alpha = 1.5$, a dual step size around $\beta = 0.0625$, and a regularization parameter $\lambda$ of the order of $10^{-3}$. We recall that the theoretically sufficient condition for the convergence of the PNPD algorithm is that the gradient descent step size is $\alpha < 2/L_S = 2$, where $L_S$ is the Lipschitz constant of $\nabla f_S$, while the dual step size is $\beta < \|W\|^{-2} = \frac{1}{8}$, since $W$ is the discrete gradient operator. The fact that the predicted regularization parameter $\lambda$ is almost constant across different images of the test set is coherent with theoretical expectations. In particular, the blurring operator and the noise level are fixed for the test set, and therefore, the optimal regularization parameter is expected to be similar across all examples, slightly varying due to the different content of the images. We also observe that the predictor network for the preconditioner parameter $\nu$ outputs values

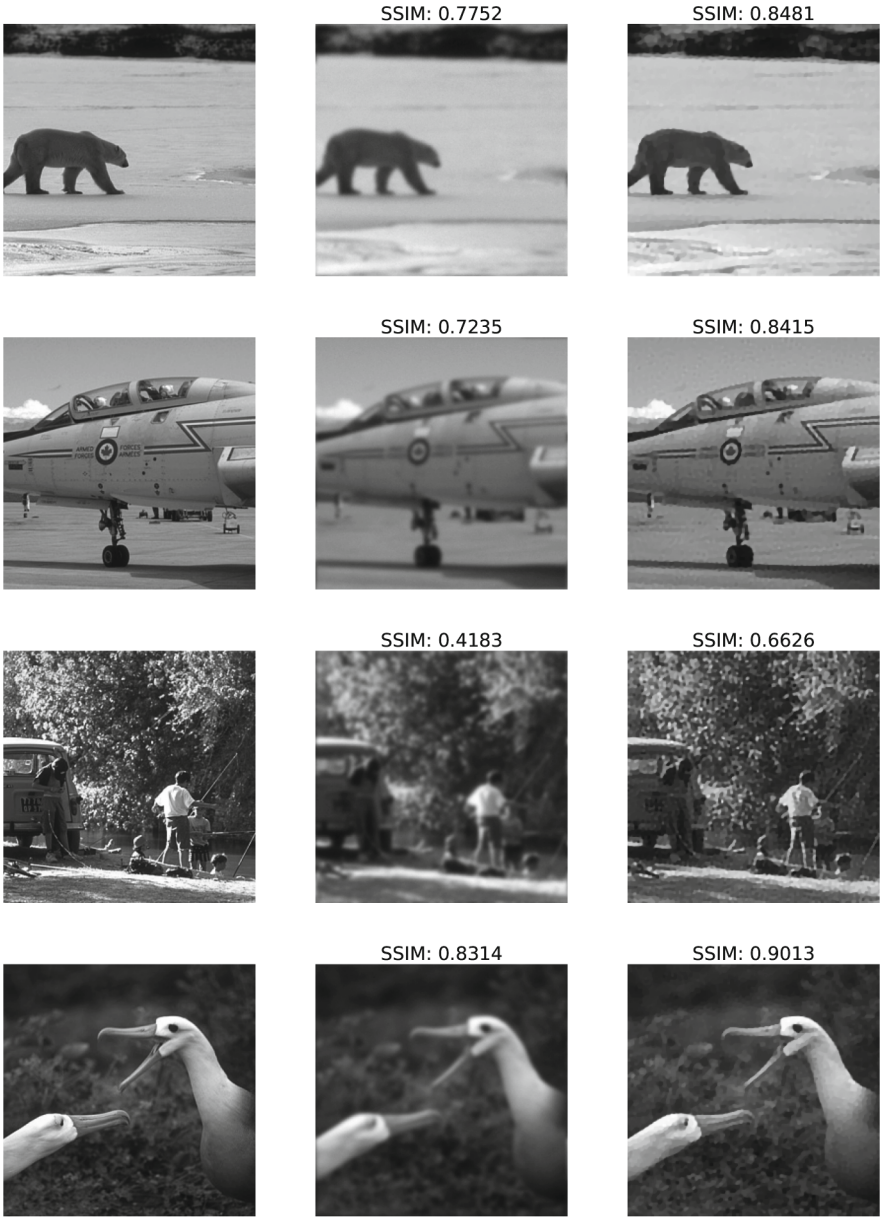


**Fig. 3.** Reconstruction of some images of the test set.

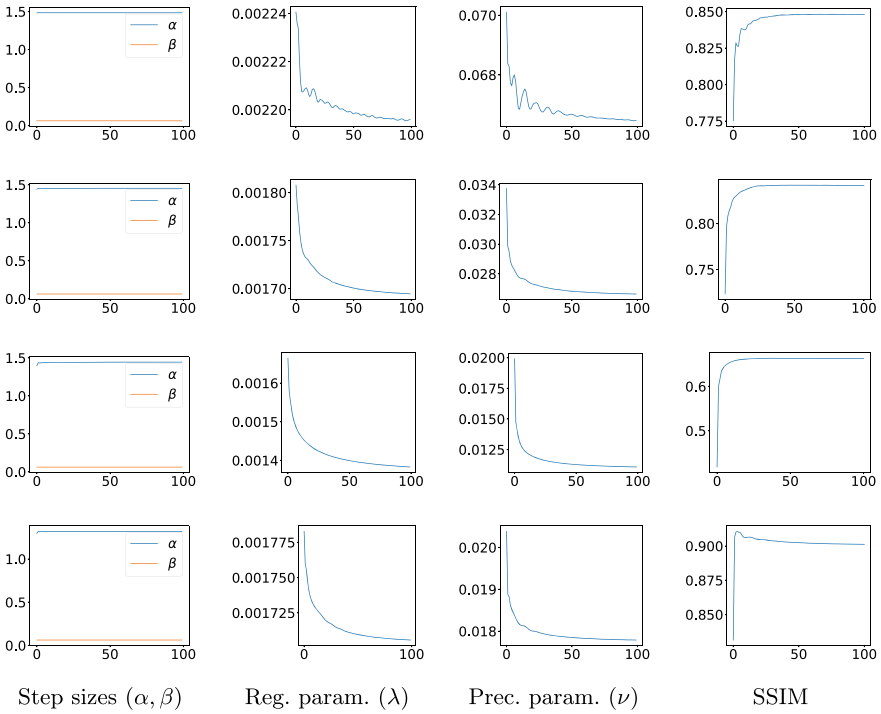


**Fig. 4.** Evolution of the parameters and of the SSIM as a function of the iterations for the images of the test set shown in Fig. 3.

that are in the range $[10^{-2}, 10^{-1}]$. This is consistent with the manual tuning of the parameter $\nu$ for the original PNPD algorithm, cf. [2], which led to fast convergence and good reconstruction quality for values of $\nu$ in the same range, while smaller values of $\nu$ led to instability of the algorithm, and larger values led to slower convergence, comparable to the one of the original NPD algorithm. Figure 4 also shows the evolution of the SSIM across iterations. We observe that the SSIM is monotonically increasing for the second and third images. For the first image, the SSIM exhibits slight oscillations in the initial iterations, but its overall behavior is increasing and stabilizes going on with the iterations. This may be due to $\nu_i$ being too small, which can lead to instability in the early iterations of the algorithm. For the last image, the SSIM reaches a peak at iteration 3 and then stabilizes at a slightly lower value. In this case, early stopping, for example at $i = N = 18$, could yield a slightly better reconstruction quality.

Finally, Fig. 5 shows the evolution of the negative loss function as a function of epochs and of time during training. Since the loss function is defined as the average of the negative SSIM computed on the intermediate iterates, this value can be interpreted as a measure of the reconstruction quality on the training and

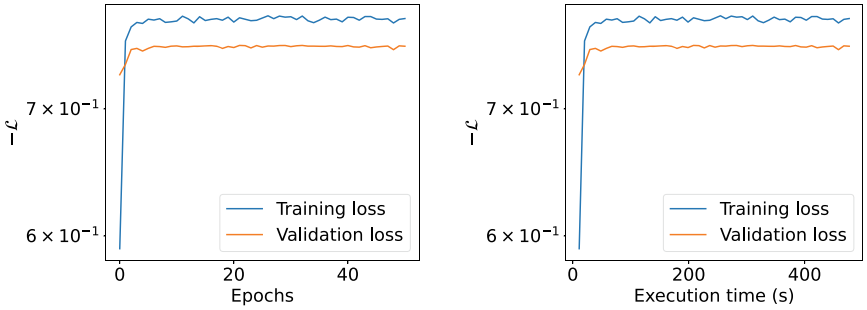


**Fig. 5.** Evolution of the loss function as a function of epochs (left) and of time (right).

validation sets. We observe that the negative loss increases significantly during the first epochs, while it remains nearly constant for the rest of the training.

## 5 Conclusion

In this work, we proposed an unrolled architecture for the PNPD algorithm, which enables the automatic estimation of the parameters involved in both the optimization problem and the algorithm, without the need for manual tuning. Numerical results on the image deblurring problem demonstrate the effectiveness of the proposed approach in terms of both reconstruction quality and parameter estimation.

Some aspects of the proposed method have not been fully explored. One of these concerns the design of the predictor networks, which in this work are simple convolutional networks, as well as the choice of input features, here limited to the observed image and the current iterate. Additional information on the state of the algorithmm, such as the dual iterates, residuals, or other features relevant for parameter estimation, could be incorporated as input to the predictor networks. Another aspect that warrants further investigation is the design of the loss function, which in this work is defined as a weighted average of the negative SSIM computed over the intermediate iterates. Different weighting strategies could be analyzed to see how they affect the training and the reconstruction quality. Finally, the training procedure of the proposed architecture could be further analyzed to improve convergence and stability, ultimately leading to enhanced reconstruction quality and more accurate parameter estimation.

**Acknowledgments.** The authors are partially supported by the Italian Ministry of the University and Research, through the PRIN 2022 project "Inverse Problems in the Imaging Sciences (IPIS)" (No. 2022ANC8HL), and GNCS-INdAM.

## References


1. Aleotti, S., Bonettini, S., Donatelli, M., Prato, M., Rebegoldi, S.: A nested primal-dual iterated Tikhonov method for regularized convex optimization. Comput. Optim. Appl. **91**(2), 357–395 (2025). https://doi.org/10.1007/s10589-024-00613-4
2. Aleotti, S., Donatelli, M., Krause, R., Scarlato, G.: A preconditioned version of a nested primal-dual algorithm for image deblurring. J. Sci. Comput. **103**(3), 85 (2025). https://doi.org/10.1007/s10915-025-02863-8
3. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5), 898–916 (2011). https://doi.org/10.1109/TPAMI.2010.161
4. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. **2**, 183–202 (2009)
5. Bertero, M., Boccacci, P.: Introduction to inverse problems in imaging. Institute of Physics Publishing, Bristol (1998)
6. Bertocchi, C., Chouzenoux, E., Corbineau, M.C., Pesquet, J.C., Prato, M.: Deep unfolding of a proximal interior point method for image restoration. Inverse Prob. **36**(3), 034005 (2020). https://doi.org/10.1088/1361-6420/ab460a
7. Bonettini, S., Loris, I., Porta, F., Prato, M., Rebegoldi, S.: On the convergence of a linesearch based proximal-gradient method for nonconvex optimization. Inverse Prob. **33**(5), 055005 (2017). https://doi.org/10.1088/1361-6420/aa5bfd
8. Bonettini, S., Prato, M., Rebegoldi, S.: A nested primal-dual FISTA-like scheme for composite convex optimization problems. Comput. Optim. Appl. **84**, 85–123 (2023)
9. Chambolle, A., Pock, T.: An introduction to continuous optimization for imaging. Acta Numerica **25**, 161–319 (2016)
10. Chen, J., Loris, I.: On starting and stopping criteria for nested primal-dual iterations. Numer. Algorithms **82**, 605–621 (2019)
11. Chen, Y., Chen, X., Maleki, A., Jalali, S.: Comprehensive examination of unrolled networks for solving linear inverse problems. Entropy **27**(9) (2025). https://doi.org/10.3390/e27090929
12. Chouzenoux, E., Pesquet, J.-C., Repetti, A.: Variable metric forward–backward algorithm for minimizing the sum of a differentiable function and a convex function. J. Optim. Theor. Appl. **162**(1), 107–132 (2013). https://doi.org/10.1007/s10957-013-0465-7
13. Combettes, P.L., Pesquet, J.C.: Proximal splitting methods in signal processing. In: Bauschke, H., Burachik, R., Combettes, P., Elser, V., Luke, D., Wolkowicz, H. (eds.) Fixed-Point Algorithms for Inverse Problems in Science and Engineering. SOIA, vol. 49, pp. 185–212. Springer, New York (2011). https://doi.org/10.1007/978-1-4419-9569-8_10
14. Daubechies, I., Defrise, M., Mol, C.D.: An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Commun. Pure Appl. Math. **57**(11), 1413–1457 (2004)
15. Engl, H.W., Hanke, M., Neubauer, G.: Regularization of Inverse Problems, 1996th edn. Mathematics and Its Applications. Springer, Dordrecht (1996). ISBN: 0-7923-4157-0
16. Golub, G.H., Heath, M., Wahba, G.: Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics **21**(2), 215 (1979)
17. Gregor, K., LeCun, Y.: Learning fast approximations of sparse coding. In: Proceedings of the 27th International Conference on Machine Learning, ICML'10, pp. 399–406. Omnipress, Madison, WI, USA (2010)

18. Hansen, P.C., O'Leary, D.P.: The use of the L-curve in the regularization of discrete ill-posed problems. SIAM J. Sci. Comput. **14**(6), 1487–1503 (1993). https://doi.org/10.1137/0914086
19. Kolen, J.F., Kremer, S.C.: Gradient flow in recurrent nets: the difficulty of learning longterm dependencies, pp. 237–243 (2001). https://doi.org/10.1109/9780470544037.ch14
20. Monga, V., Li, Y., Eldar, Y.C.: Algorithm unrolling: interpretable, efficient deep learning for signal and image processing. IEEE Sig. Process. Mag. **38**(2), 18–44 (2021). https://doi.org/10.1109/MSP.2020.3016905
21. Ochs, P., Chen, Y., Brox, T., Pock, T.: iPiano: inertial proximal algorithm for nonconvex optimization. SIAM J. Imag. Sci. **7**(2), 1388–1419 (2014). https://doi.org/10.1137/130942954
22. Schmidt, M., Roux, N., Bach, F.: Convergence rates of inexact proximal-gradient methods for convex optimization. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (eds.) Advances in Neural Information Processing Systems, vol. 24. Curran Associates, Inc. (2011)
23. Tikhonov, A.N., Goncharsky, A., Stepanov, V.V., Yagola, A.G.: Numerical Methods for the Solution of Ill-Posed Problems, 1995th edn. Mathematics and Its Applications. Springer, Dordrecht, The Netherlands (1995)