Soft Computing (2025) 29:4153–4174
https://doi.org/10.1007/s00500-025-10642-8

FOCUS

Early stopping strategies in Deep Image Prior

Alessandro Benfenati[1,4] · Ambra Catozzi[2,3,4] · Giorgia Franchini[2,4] · Federica Porta[2,4]

Accepted: 6 February 2025 / Published online: 17 May 2025

Abstract
The so-called Deep Image Prior approach is an unsupervised deep learning methodology which has gained great interest in recent years due to its effectiveness in tackling imaging problems. However, a well known drawback of Deep Image Prior is the need for a proper early stopping technique to prevent undesired corruptions in the reconstructed images. Determining the optimal number of iterations depends on both the specific application and the features of the images being processed. As a consequence, several numerical trials are typically required to decide when to stop the Deep Image Prior procedure, resulting in significant computational costs and time requirements. This paper aims to introduce two early stopping techniques for Deep Image Prior, based on different approaches and different perspectives. The first approach relies on the neural architecture search (NAS) strategy. The aim is to equip the neural network used in Deep Image Prior with hyperparameter configurations that produce high-quality reconstructed images that are (i) comparable to those obtained with the optimally stopped, originally configured Deep Image Prior, and (ii) achieved with significantly fewer iterations. The second proposed early stopping strategy is based on a modified version of the BRISQUE metric, a no-reference image quality measure, and it aims to track the behaviour of the PSNR curve, obtained by applying Deep Image Prior, without knowing the ground truth image. While the NAS-based early stopping technique is particularly suited in those situations where the computational time is limited, this latter one is also relevant when a larger number of iterations is allowed. Several numerical experiments on different denoising applications show a promising performance of Deep Image Prior combined with the suggested early stopping procedures.

Keywords Deep Image Prior · Image denoising · Neural architecture search · Performance predictor · Early stopping · No-reference quality measure · Efficiency · Green AI

1 Introduction

The Deep Image Prior (DIP) (Ulyanov et al. 2018, 2020) is an unsupervised approach tailored for solving a large class of imaging inverse problems. Such approach employs artificial neural networks, particularly U-Nets equipped with skipped connections (Ronneberger et al. 2015): the claim of the original work states that the *inner structure* of this type of networks is capable to capture a great amount of image statistics, without a training on large datasets. In Ulyanov et al. (2018) and Ulyanov et al. (2020) the authors fit an untrained network to a single degraded image, addressing several problems such as image denoising, image restoration, inpainting and super-resolution. Moreover, DIP has found successful practical applications in other imaging domain such as blind deconvolution (Huo et al. 2023; Benfenati et al.

Alessandro Benfenati, Ambra Catozzi, Giorgia Franchini and Federica Porta have contributed equally to this work.

✉ Ambra Catozzi
ambra.catozzi@unipr.it

Alessandro Benfenati
alessandro.benfenati@unimi.it

Giorgia Franchini
giorgia.franchini@unimore.it

Federica Porta
federica.porta@unimore.it

[1] Environmental and Science Policy department, University of Milan, Via Celoria, 2, 20133 Milan, Italy

[2] Department of Physics, Informatics and Mathematics, University of Modena and Reggio Emilia, Via G. Campi, 213/B, 41125 Modena, Italy

[3] Department of Mathematical, Physical and Computer Sciences, University of Parma, Parco Area delle Scienze, 7/A, 43124 Parma, Italy

[4] Gruppo Nazionale Calcolo Scientifico, INDAM, Piazzale Aldo Moro, 5, 00185 Rome, Italy

2023a; Kotera et al. 2021), image segmentation (Burrows et al. 2021; Benfenati et al. 2023b), medical imaging (Yoo et al. 2021; Lin and Huang 2020).

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

In this work we develop two different early stopping strategies under the DIP framework. The first one is based on a Neural Architecture Search (NAS) based approach. NAS is a technique within the field of machine and deep learning that automates the design of neural network architectures (Elsken et al. 2019). NAS methods typically involve exploring the so called *search space* of possible architectures, which can include variations in the number of layers, types of layers, connections between layers, and other architectural hyperparameters. The goal of NAS is to find neural network architectures that achieve high performance on a given task while minimizing the need for human intervention in the design process. Our idea consists of exploiting a NAS strategy to predict an hyperparameters configuration for the DIP net able to halt the learning process before the semi-convergence appears and in a prefixed (typically limited) number N of iterations.

In more detail, once N is fixed, we explore the search space of the original DIP U-Net for generating the most performing net within N iterations. This NAS-based early stopping strategy not only avoids the semi-convergence phenomenon and the resulting tuning of the optimal number of iterations, but also enables the generation of network configurations that yield favorable outcomes within potentially fewer iterations than those required by the original DIP approach, optimally stopped. We additionally note that our search space, besides the architectural ones, also include the hyperparameters related to both the optimization process and the regularizing term in the loss function. As for the *search strategy*, namely the method used for exploring the search space, we exploit a random search walk. Specifically we train a proper performance predictor that, given a test image, is able to provide the best hyperparameter configurations among a pool of different options by observing only the behaviour of the corresponding networks in the very early stage of the training. Moreover, the NAS-based early stopping procedure we consider has been implemented by exploring the search space with two approaches: either keeping fixed the image on which the performance predictor is trained fixed or varying it. In the first case it is possible to evaluate the ability of the performance predictor to generalize to very different test images. With the second approach the performance predictor learns to provide image-dependent hyperparameters configurations. We remark that several NAS approaches have been applied to the DIP framework, but they differ from ours with respect to both the steps for realizing the search and the goal they aim to achieve. In Ho et al. (2021) the author propose a well-designed binary search space and employ as search strategy a genetic algorithm and they prove that the structure of the network is content-style dependent. Chen et al. (2020) restricts the search space to the network components related to the up-sampling part and modifies the original architecture by considering cross-scale residual connections, i.e., the skip connections link blocks of the encoder part with blocks of the decoder part at different spatial dimensions. In Arican et al. (2022) the focus is shifted on the metrics used to evaluate the performance: the authors develop several novel metrics that allow to reduce the costs of NAS procedures. It is worth to notice that, differently from the approach proposed in this paper, in none of the previously cited works on NAS for DIP, a performance predictor has been employed to improve the DIP behaviour. Finally, we emphasize that our main aim in applying NAS to DIP differs from the standard idea behind NAS and found in the cited papers (Ho et al. 2021; Chen et al. 2020; Arican et al. 2022). While those studies aim to find the optimal hyperparameter configurations for specific applications, our focus is on identifying the optimal configurations given a fixed number of iterations, and hence realizing and early stopping technique.

The second early stopping strategy we propose relies on an idea inspired by the Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) (Mittal et al. 2012) metric, a no-reference image quality measure (Mittal et al. 2012). In our case the aim is to track the behaviour of the Peak Signal-to-Noise Ratio (PSNR) curve, obtained by applying the standard DIP, without knowing the ground truth image. In particular, using the hyperparameters settings suggested in the original works (Ulyanov et al. 2018, 2020), we employ a customized version of the BRISQUE metric to estimate the iteration at which the most reliable result can be obtained under the PSNR measure.

Other early stopping procedures for DIP can be found in the literature and are briefly recalled below. The variance of the DIP iterates is used for developing an early stopping method in Wang et al. (2023), which has been successfully applied in DDIPP approach (Benfenati et al. 2023a), where blind deconvolution problems for Poisson data have been addressed under the DIP framework. The paper Cheng et al. (2019) adopts a Bayesian framework and shows that DIP is equivalent to a Gaussian stationary process as soon as the number of input channels goes to infinity: this leads to adopt a Stochastic Gradient Langevin Dynamic algorithm, that avoids the early stopping, but at the cost of a large number

of iterations-between 27000 and 37000-which does not meet Green AI principles (Alshubaily 2021). In Li et al. (2021) the authors propose an early stopping technique based on the loss function values. However, the semi-convergence behaviour typical of DIP is also reflected in the values of the objective function, unless a proper regularizer is considered. Selecting the regularization term and its corresponding parameter, as well as determining the appropriate number of iterations to stop the DIP algorithm, poses a challenging task.

This work is organized as follows. Section 2 recalls the basic notions on DIP framework, while Sect. 3 presents the proposed early stopping strategies. Section 4 is devoted to the numerical validation of the developed techniques. Finally, Sect. 5 draws the conclusion of the presented work.

2 Deep Image Prior

In this section we recall the DIP framework for imaging applications. Particularly we detail the architecture of the neural network employed in the DIP approach and the optimization problem needed to be solved to find the reconstructed image.

2.1 The DIP neural network

The Deep Image Prior approach introduced in Ulyanov et al. (2018) and Ulyanov et al. (2020) shows that generative neural architectures are capable to capture image statistics even in absence of a training phase and without using large datasets. The original work employed U-Net structures for tackling several imaging problems: indeed these architectures have gained interest since they showed to have remarkable performances in several tasks, such as image segmentation (Renuka 2023; Charfi et al. 2024) in medical imaging, in image generation (Torbunov et al. 2023; Benfenati et al. 2022), precipitation nowcasting (Trebing et al. 2021). In this paper we take into account the same structure of Ulyanov et al. (2018) and Ulyanov et al. (2020): for a visual inspection we refer to Fig. 1. A U-Net can be interpreted as an autoencoder: the first part encodes the information by applying several operations (convolutions, batch normalization, leaky ReLU) and each "stage" halves the spatial dimensions of the image. The second part, formally the decoder, doubles the spatial dimensions to retrieve the original image size and reconstruct the information. Moreover, each stage of the encoder is linked to the relative stage of the decoder, acting on the same spatial dimensions, via skip connections: on one hand, this blocks allow to transmit information among the different levels of encoding-decoding, and moreover they seem to improve the learning phase, boosting the search for the minimum (Sun et al. 2020; Li et al. 2018).

In Fig. 1 we depict the architecture used in Ulyanov et al. (2018) and Ulyanov et al. (2020), where both the encoder and the decoder have 5 stages. The first layer of an encoder stage is a convolution operation that halves the spatial dimensions, followed by a batch normalization layer and by a leaky ReLu layer with parameter -0.6. This triplet is repeated, with the sole difference that the second convolution layer does not halve the spatial dimensions. The last stage of the encoder (the lighter bottom block in Fig. 1) ends with an upsampling layer, for having the correct spatial dimensions for using the skip connections. Each decoder stage starts with a depth concatenation layer, in order to combine their input with the output of the skip connections, then 2 triplets of convolution, batch normalization and leaky ReLU operations are combined. The last operation is an upsampling operation for doubling the dimensions, since one wants to recover the original image size. The structure of the skip connections is rather simple: they combine convolution, batch normalization and leaky ReLU layers. In its original form, DIP lacks any form of regularization, neither through explicit terms in the loss function nor through Early Stopping criteria.

2.2 The DIP optimization model

In DIP framework such U-Net is used as a generator: it is fed with a multichannel random Gaussian noise, which is used to recover the image of interest by learning the weights of the net under the minimization of a fit-to-data functional. We briefly recall the linear model for image acquisition. The recorded data g is given by the following model

$$g = \mathcal{P}\left(H * x^\star + b\right) \tag{1}$$

where $x^\star$ is the ground truth, H is a linear operator describing the blur effects-such as out-of-focus, motion blur-of the acquisition system, b a constant background term, $*$ denotes the convolution and $\mathcal{P}$ models the statistical noise affecting the data, which can be signal-dependent or signal-independent. The numerical experiments of Sect. 4 will be carried out on denoising task. In this case H consists in the identity operator. The recorded data can be a B&W or an RGB image, hence $g \in \mathbb{R}^{p\times q\times d}$, with $p \times q$ pixels and $d \in \{1, 3\}$ channels; let $z \sim \mathcal{N}(0, \sigma^2)$, $z \in \mathbb{R}^{p\times q\times h}$ a random Gaussian input with same spatial dimensions and h channels. The U-Net can be described as a function $f : \Theta \times \mathbb{R}^{p\times q\times h} \to \mathbb{R}^{p\times q\times c}$, where Θ is the space of the net weights θ. The DIP approach is based on numerically solving an optimization problem, searching for the optimal weights $\theta^\star$:

$$\theta^\star \in \underset{\theta\in\Theta}{\operatorname{argmin}}\ \mathcal{D}\left(f(\theta, z), g\right) \tag{2}$$

where $\mathcal{D}$ is the discrepancy functional chosen in dependence of the noise affecting the image. Once the optimal weights $\theta^\star$ have been found by applying an iterative algorithm to (2),

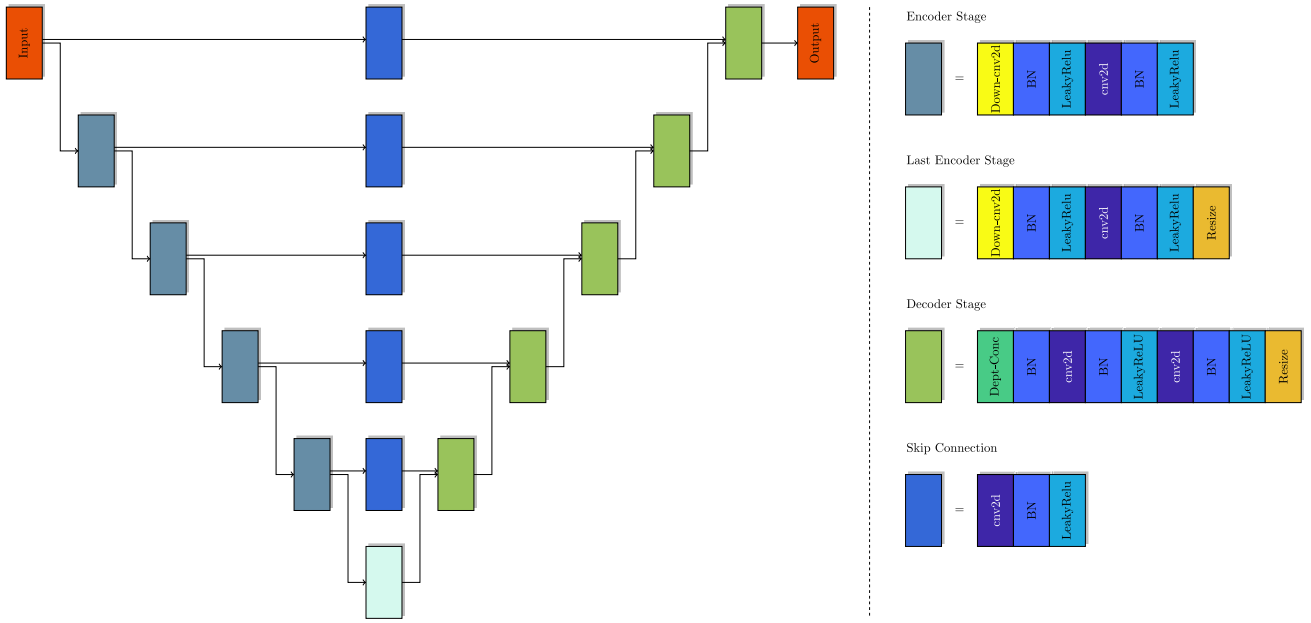


Fig. 1 The Deep Image Prior is characterized by a U-Net architecture. The picture shows the original architecture of Ulyanov et al. (2020), with 5 stages of downsampling and relative upsampling, together with the skip connections. The last stage of the encoder part, which corresponds to the latent feature, is depicted in light blue and differs from the other encoder stage by the presence of a final upsampling layer. We refer to the online version for color references (Colour figure online)

properly stopped before the degraded image starts to be over fitted, the resulting approximation of $x^\star$ can be computed as $f(\theta^\star, z)$. The complete scheme of the DIP approach is reported in Algorithm 1. The network structure used in the original work for the denoising task is depicted in Table 2.

Algorithm 1 Deep Image Prior

Select the network f, initialise the network's weights θ^0, select the discrepancy function $\mathcal{D}$. Set σ^2 for the input's variance. Choose the learning algorithm $Algo$ and its hyperparameters. Select the maximum number of iteration N.
for $k = 0, 1, \ldots, N-1$ **do**
$z \sim \mathcal{N}(0, \sigma^2)$, a realization from a Gaussian probability distribution.
$\ell(\theta^k) \leftarrow \mathcal{D}\left(f(\theta^k, z), g\right)$
$\theta^{k+1} \leftarrow Algo(\theta^k, \nabla\ell(\theta^k))$
end for
Recover the approximation of $x^\star$ as $f(\theta^N, z)$.

We remark that it is possible to add a regularization term to Problem (2), for example the anisotropic Total Variation function:

$$TV(x) = \sum_i \|A_i\, x\|, \tag{3}$$

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

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

$$\underset{\theta\in\Theta}{\operatorname{argmin}}\ \mathcal{D}\left(f(\theta, z), g\right) + \lambda TV(f(\theta, z)), \tag{4}$$

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

3 Proposed early stopping procedures

In this section we detail two distinct early stopping strategies to be combined with the DIP framework. The perspectives of the two proposed procedures are different. The first one aims not only to prevent the semi-convergence behaviour of DIP, but also to get a neural network configuration able to provide effective results in a small number of iterations, accordingly to the Green AI principles. This aspect is very relevant in all those applications where the computational time is limited or a real-time solution is needed, just consider the biomedical or the autonomous driving fields to name a few. On the other hand, the second strategy does not modify the structure of the

neural network on which the original DIP is based, but allows to monitor the generated PSNR behaviour without knowing the ground truth image. The optimization problem we refer to in this section is the one defined in (4).

3.1 NAS-based early stopping

The first approach is an early stopping strategy *de facto*. Given the particular imaging application and, hence, the specific definition of the discrepancy functional in (4), we fix a number of iterations and we search via NAS techniques the architecture that can provide the most reliable results in terms of several performance measures, namely the Peak Signal-to-noise Ratio (PSNR) (Horé and Ziou 2010), the Structure Similarity Index Measure (SSIM), its multiscale version (MS-SSIM) (Horé and Ziou 2010) and the reconstruction relative error (RRE). The search space for the U-Net network of Fig. 1 encompasses different features; we enlist them below:

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

Since the actual search space is potentially infinite, we focus on one of its discrete subsets, considering discrete intervals for each of the above options. We denote this search subspace with $\Omega = \mathcal{I}_1 \times \mathcal{I}_2 \times \cdots \times \mathcal{I}_8$, where $\mathcal{I}_j$ is the discrete interval containing the possible values for the j-th item in the above list. Ω can be considered as a subset of $\mathbb{R}^8$. We refer the reader to Sect. 4 (Tables 2, 7) for the complete list considered in the numerical experiments of this work. It is quite clear that the possible number of configurations to be taken into account is of course too large to be tested in a brute force manner. Testing them all, to find the most effective one for a given number of iterations, is time consuming and goes against the principles of Green AI (Alshubaily 2021). For these reasons, by following a strategy similar to the one suggested in Franchini et al. (2023), we train a performance predictor able to provide effective hyperparameters configurations by knowing only the performance of the networks corresponding to a very small subset of all the possible hyperparameters configurations. Below we detail how to build the performance predictor and to use it for an early stopping procedure. The main steps of this procedure are summarized in Algorithm 2, each step is detailed below.

Algorithm 2 NAS-based early stopping, single image

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

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

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

$$\sum_{q=1}^{Q} \mathcal{D}\left(\rho(h_q, P^{(q)}) - p_N^{(q)}\right).$$

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

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

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

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

called $\{\hat{h}_b\}_{b=1,\dots,B} \subset \{\tilde{h}_m\}_{m=1,\dots,M}$ as predicted by ρ, that will be run the for N iterations.

The aim is to search for the best configuration for recovering images of a set $\mathcal{I}$ sharing common characteristics, such as astronomical images corrupted by Poisson noise (see Sect. 4): hence we select a representative image I, containing the most prominent features of this set.

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

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

$$\sum_{q=1}^{Q} \mathcal{D}\left(\rho(h_q, P^{(q)}), p_N^{(q)}\right),$$

where $\mathcal{D}$ is a suitable distance function. The natural choice is the ℓ_2 norm.

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

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

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

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

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

We also considered a different version of the previously discussed NAS-based early stopping technique where the performance predictor is trained by varying the example image. The aim is to obtain an optimal configuration of hyperparameters that depends on the image itself. In order to reach this goal we need to adjust the dataset on which the predictors are trained. More in detail, besides the values of the hyperparameters and the performance metrics related to the c checkpoints, each sample in the dataset must take into account some peculiar features of the image to be recovered. In order to obtain this image's information we exploit the same 36 features on which the no-reference image quality measure called Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) is based (Mittal et al. 2012). We stress that we are not using the BRISQUE metric at this level, but only the image's features extraction model required to get the BRISQUE values. More details on BRISQUE can be found in Sect. 3.2.

Algorithm 3 reports the scheme of the image dependent NAS-based early stopping procedure. We detail below the main differences between Algorithms 2 and 3.

Set up. The dataset $\mathcal{I}$ of corrupted images is divided into two disjoint subsets: a training set $\mathcal{T}_r$ and a test set $\mathcal{T}_s$. For each image I_f in $\mathcal{T}_r$, with $f = 1, \dots, |\mathcal{T}_r|$, we denote by $F^{(f)} \in \mathbb{R}^{36}$ the vector storing its 36 significant features which have been extracted by following the same strategy employed to train BRISQUE. Moreover, for each image I_f in $\mathcal{T}_r$ we consider Q different hyperparameters configurations $\{h_q^{(f)}\}_{q=1,\dots,Q}$. The value of Q is fixed as in Algorithm 2. In this approach we denote with O the subset of Ω containing the $Q\,|\mathcal{T}_r|$ selected configurations. $N \in \mathbb{N}$ and $M \in \mathbb{N}$ represent again the maximum number of allowed iterations and the number of configurations needed for the **Qualification phase**, respectively. $B \in \mathbb{N}$ denotes instead the number of training images to be identified as the B most similar to a given test image. This parameter is needed for the reconstruction of the corrupted images.

Dataset creation. Algorithm 1 runs Q times for each of the images taken from the training set $\mathcal{T}_r$. We store in the columns of $P^{(f,q)} \in \mathbb{R}^{4\times c}$ the metrics at each checkpoint $t_1, \dots, t_c$ obtained by considering the configuration $h_q^{(f)}$. The vector

Algorithm 3 NAS-based early stopping, multiple images

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

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

Training.
Given the features $\{F^{(f)}\}_{f=1,\dots,|\mathcal{T}_r|}$ of the images considered in the previous step, train ρ on the dataset $\{(F^{(f)}, h_q^{(f)}, P^{(f,q)}), p_N^{(f,q)}\}_{f=1,\dots,|\mathcal{T}_r|;q=1,\dots,Q}$ in order to minimize a distance $\mathcal{D}$:

$$\mathcal{D}\left(\rho(F^{(f)}, h_q, P^{(f,q)}) - p_N^{(f,q)}\right).$$

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

$$\tilde{p}_N^{(f,m)} = \rho(F^{(f)}, \tilde{h}_m^{(f)}, \tilde{P}^{(f,m)}).$$

  end for
end for

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

$p_N^{(f,q)} \in \mathbb{R}^4$ stores the metrics at the end of the run, that is at the N-th iteration.

Training. Differently from Algorithm 2, the predictor ρ is trained on samples that take into account not only the hyperparameters configuration and the metrics in the first checkpoints, but also the features of the image used to get $p_N^{(f,q)}$, $f = 1, \dots, |\mathcal{T}_r|$, $q = 1, \dots, Q$. Particularly, it must hold that

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

Qualification phase. The qualification phase of Algorithm 3 differs from the one of Algorithm 2 only from the fact that it is repeated on M different configurations $\{\tilde{h}_m^{(f)}\}_{m=1,\dots,M}$ for every image belonging to the training set. Recall that these M configurations are different from the ones used during the Training.

Reconstruction of the images from the dataset $\mathcal{I}$. This phase is the most distinctive step of Algorithm 3 compared to Algorithm 2. Given a new image I of the test set $\mathcal{T}_s$, the most similar images belonging to the training set $\mathcal{T}_r$ must be identified. To reach this goal we compare the features of the test image I (stored in a vector called $F \in \mathbb{R}^{36}$) with those of the training images by computing the Mean Squared Error (MSE) between F and $F^{(f)}$, for all $f = 1, \dots, |\mathcal{T}_r|$, namely

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

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

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

We stress that such configuration $h^\star$ possibly varies with the images in the test set.

3.2 BRISQUE-based early stopping

The second early stopping strategy does not aim to modify the structure of the original U-Net architecture considered in the DIP approach and described in Sect. 2. Our goal is to monitor the semiconvergence behaviour of DIP in order to stop the algorithm when the most reliable result can be obtained in

terms of PSNR. Of course this goal must be reached without employing the ground truth image. For this reason, the idea is to exploit a no-reference image quality measure such as the already mentioned BRISQUE metric (Mittal et al. 2012).

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

A naive approach could employ BRISQUE as it is defined in the original paper (Mittal et al. 2012). In more detail, at each iteration k of Algorithm 1, the BRISQUE value related to $f(\theta^{(k)}, z)$ can be computed and once this value starts to increase, Algorithm 1 is stopped. However, as highlighted in Wang et al. (2023) and confirmed in the numerical experiments of the next section, this simple approach is not able to improve standard DIP. For this reason, we consider a customized version of the BRISQUE metric by differently training the prediction algorithm on which BRISQUE itself is built. The details of this approach and its combination with DIP is proposed in Algorithm 4.

The main steps are described below.

Set up. We select a collection $\mathcal{C}$ of corrupted images $\{I_f\}_{f=1,\dots,|\mathcal{C}|}$ with known ground truth $\{G_f\}_{f=1,\dots,|\mathcal{C}|}$, needed to train the customized version of BRISQUE. Moreover all the parameters related to DIP must be selected, particularly the structure of the network f, the initial weights θ^0, the discrepancy function $\mathcal{D}$, the input variance σ^2, the learning algorithm $Algo$ and its hyperparameters and the maximum number N of iterations. We remark that for the numerical experiments, the hyperparameters related to the structure of the network and the optimization algorithm are kept as in the original paper (Ulyanov et al. 2018). Finally, in order to early stop DIP we need a so called patience parameter, denoted by τ.

Training of a customized BRISQUE. The prediction algorithm we consider to train the customized version of BRISQUE is not simply based on a SVR model as in (Mittal et al. 2012). Indeed we consider a PSNR prediction which exploits both SVR and RF methods, as explained hereafter. We train both a SVR and a RF procedures based on a dataset whose samples are the 36 features of the images $\{I_f\}_{f=1,\dots,|\mathcal{C}|}$ and the target values are the related PSNR values. The features of $\{I_f\}_{f=1,\dots,|\mathcal{C}|}$ are stored in the vectors $\{F^{(f)}\}_{f=1,\dots,|\mathcal{C}|}$. In more detail, given the dataset $\{F^{(f)}, \mathrm{PSNR}(I_f)\}_{f=1,\dots,|\mathcal{C}|}$, the SVR and the RF algorithms are trained such that

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

Algorithm 4 BRISQUE-based early stopping

Set up

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

Training of a customized BRISQUE

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

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

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

and

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

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

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

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

Reconstruction of the images from the dataset. $\mathcal{I}$ Given a dataset $\mathcal{I}$ of corrupted images, possibly with unknown ground truth, the goal is to apply the DIP algorithm to recover proper reconstructions by avoiding semi-convergence effect. For this reason we consider a modified version of Algorithm 1 where the $\mathrm{cBRISQUE}_i$, $i = 1, 2, 3$, metrics are employed. Particularly, once the new iterate θ^{k+1} is updated, the $\mathrm{cBRISQUE}_i$ values β_i^{k+1}, $i = 1, 2, 3$, of the corresponding image $f(\theta^{k+1}, z)$ are computed. When all the $\mathrm{cBRISQUE}_i$ metrics are not reduced after a number of successive iterations equal to the value of the patience τ, the DIP procedure is stopped even if the maximum number of iterations is not reached. In this case, the approximation $x^\star$ is computed by means of that vector of weights corresponding to the highest PSNR value predicted by $\mathrm{cBRISQUE}_1$, $\mathrm{cBRISQUE}_2$ and $\mathrm{cBRISQUE}_3$.

4 Numerical experiments

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

Poisson noise is often generated in low-light conditions and when an image is captured by digital devices. It consists in a multi-valued Poisson process: the outcome is influenced by the random incidence of photons on the sensor of the detection instrument (photon counting), with each pixel's value being independent of the others. A further error source, that can be viewed as another Poisson process, is the photon-electron counting, which transforms the light information into the digital one. The intensity of the noise thus depends on the number of photons: as the number of photons increases, the level of noise affecting the image decreases (Bertero et al. 2018; Zanella et al. 2009). The discrepancy functional used for restoring images perturbed by this kind of noise is the generalized Kullback–Leibler function, that for denoising problems reads as:

$$KL(x, g) = \sum_i g_i \ln\left(\frac{g_i}{(x+b)_i}\right) + \sum_i (x + b - g)_i\,, \quad (5)$$

where $g_i \ln g_i = 0$ if $g_i = 0$. Note that the Poisson noise is signal-dependent.

In our experiments, we also examine a second type of noise characterized by impulsive behaviour, known as Cauchy noise. It mainly appears in remote sensing, radar, sonar applications and biomedical image acquisition (Sciacchitano et al. 2015; Mei et al. 2018). The Cauchy noise is characterized by its level γ: the higher the value, the higher the level of perturbation. Moreover, such noise type is signal independent. The discrepancy function employed for denoising images corrupted by Cauchy noise reads as:

$$C(x, g) = \sum_i \log\left(\gamma^2 + (g - x)_i^2\right) \quad (6)$$

The datasets employed in the experiments are the following:

- Top 100 Images of the Hubble telescope (https://esahubble.org/images/archive/top100/; https://www.kaggle.com/datasets/redwankarimsony/top-100-hubble-telescope-images), hereafter called NASA dataset. We preprocessed the images, shrinking them into $N \times M \times 3$ and then we perturbed them with Poisson noise by means of a custom MATLAB function;
- Berkeley Segmentation Dataset and Benchmark (BSDS500, Martin et al. 2001). We selected 49 images and corrupted them via Cauchy noise with parameter $\gamma = 1, 5, 10$ via a custom MATLAB function.

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

4.1 NAS-based early stopping

We performed two distinct numerical experiments to validate Algorithms 2 and 3. In testing Algorithms 2 and 3 we consider different noise affecting the images and different datasets of images.

4.1.1 Evalutation of Algorithm 2

The performance of Algorithm 2 has been evaluated separately on the NASA and the BSDS500 datasets. For the BSDS500 dataset, we consider only Cauchy noise with $\gamma = 5$. Before discussing the implementation details of Algorithm 2, it is important to examine some properties of these datasets. Specifically, Algorithm 2 relies heavily on selecting a single image from the dataset that should be representative of the entire dataset itself.

Datasets properties. The BSDS500 images are considerably more heterogeneous than the NASA images. Indeed, the BSDS500 dataset includes a wide range of RGB natural images, such as depictions of people, animals, landscapes, and inanimate objects. This diversity in colors, compositions, and backgrounds makes the BSDS500 dataset significantly more challenging for Algorithm 2 to process compared to the NASA dataset, whose images are much more uniform. To quantitatively support this observation, we preliminarily computed the 18 main features of the BRISQUE metric for both datasets and present their variance values in Table 1.

As shown in Table 1, the highest variance values among all 18 BRISQUE features are observed in the BSDS500 dataset, highlighting its variability and heterogeneity. For this reason, the representative images for the two datasets were selected differently, as explained in the following paragraph.

Implementation details. In this paragraph, we provide the details related to the implementation of Algorithm 2. Where not specified, the same implementation features were applied to both datasets.

First, we detail the chosen search space underlying Algorithm 2. We discretized the hyperparameters space described in Sect. 3 as reported in Table 2. In details, the set Ω consists of the hyperparameters configurations composed by: the standard deviation σ for the input Gaussian noise and its number of channels C; the number of upsampling and downsampling stages of the network, for the encoder and the decoder, respectively; the number of filters for the skip connections and for the convolutional layer; the regularization parameter λ; the choice of the optimization algorithm for minimizing the loss, such as Stochastic Gradient Descent (SGD), SGD with momentum with $\beta = 0.9$ (SGDM) or Adam; the learning rate lr. This discretized search subspace leads us to engage 145152 samples, see Table 2 for details on the values for each hyperparameter.

For the **Set up** part of Algorithm 2 we decide to sample from Ω the $0,04\%$ ($s = 0.0004$) of all possible configurations, namely $Q = 58$, mimicking the approach in Franchini et al. (2023). We set $M = 200$ and $B = 20$; moreover, we choose $c = 3$ checkpoints, namely we store the PSNR, the SSIM, the MS-SSIM and the RRE at iterations $t_1 = 50$, $t_2 = 100$ and $t_3 = 150$. The total number of iteration is $N = 1000$. The used label for the performance predictor training is the PSNR at the N-th iteration. The performance predictor is based on both the SVR and the Random Forest algorithms. Particularly, we consider as a prediction the averaged values provided by RF and SVR. The representative image for the NASA dataset is the one depicted in Fig. 2: we chose both the characteristics of images with planets (centered object with dark background) and those with nebulae (stars in the background).

Some examples are reported in Fig. 2b–d . On the other hand, to identify the most representative image from the BSDS500 dataset, we computed the centroid of its images. Figure 3 shows the selected image both in its original form and with added Cauchy noise with $\gamma = 5$.

In the **Training** phase of Algorithm 2 we applied Algorithm 1 to the image in Fig. 2a for the NASA dataset and the image in Fig. 3a for the BSDS500 dataset, using the configurations $\{h_q\}_{q=1,\dots,58}$, then we proceeded to the **Qualification Phase** on other $M = 200$ configurations $\{\tilde{h}_m\}_{m=1,\dots,200}$. We chose the best $B = 20$ and, for each of them, Algorithm 1 is run for $N = 1000$ iterations. We emphasize that 1000 iterations represent a relatively low number in this context. Finally, in the **Best Configuration Selection** phase we pick the best configuration $h^\star$ (Tables 3, 4) and we move to the last step (**Reconstruction of the images from the dataset** $\mathcal{I}$) of Algorithm 2: Algorithm 1 is applied to each image of the selected dataset under the related best configuration $h^\star$.

Results for the NASA dataset. First, we analyze the best 20 selected configurations obtained by the **Qualification Phase** for the NASA dataset, summarizing the distributions of the hyperparameters' values in Fig. 4. The hyperparameter values of the vanilla DIP configuration are marked with an asterisk in the labels, while the values of the best configuration, $h^\star$, are highlighted by coloring the corresponding bar in orange. Figure 4 clearly shows that the values of $h^\star$ do not generally correspond to the vanilla DIP hyperparameter configuration and do not always align with the most frequent values among the top 20. From Fig. 4, we can claim that a good configuration does not depend on a single hyperparameter but on an effective combination of all the hyperparameters considered as a whole.

In Fig. 5 we illustrate the results obtained in 1000 iterations by applying Algorithm 1 combined with both the standard DIP hyperparameters configuration and the $h^\star$ one (found by Algorithm 2 and reported in Table 3) on the images Fig. 2b–d. We also report the corresponding PSNR values.

It is evident that within an economic perspective on the number of iterations, that is within 1000 iterations, we achieve a higher PSNR value, indicating a better quality in the recovered image. Our results propose sharper images, without loosing details. Table 5 reports the average PSNR of the reconstructed images from the NASA test set, achieved by the vanilla DIP and Algorithm 1 combined with $h^\star$. The

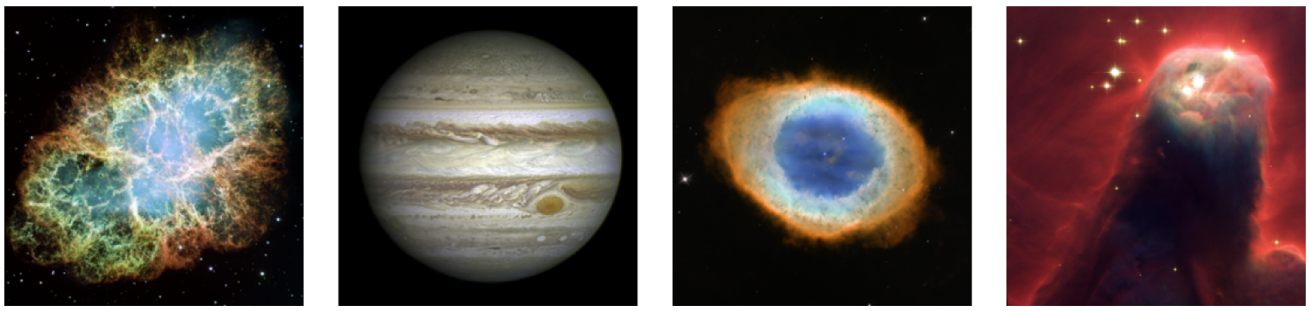
Table 1 Variance of the BRISQUE main coefficients for the NASA and BSDS500 datasets

Data set	Features								
NASA	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
	0.47	0.18	0.1	0.02	0.06	0.06	0.09	0.03	0.05
	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
	0.07	0.09	0.02	0.06	0.06	0.09	0.02	0.06	0.05
BSDS500	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
	0.74	0.38	0.2	0.07	0.12	0.1	0.19	0.07	0.12
	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
	0.1	0.18	0.05	0.11	0.1	0.18	0.05	0.11	0.1

Table 2 Space of the hyperparameters for Algorithm 2

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

The values in bold are the default ones for the vanilla DIP



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

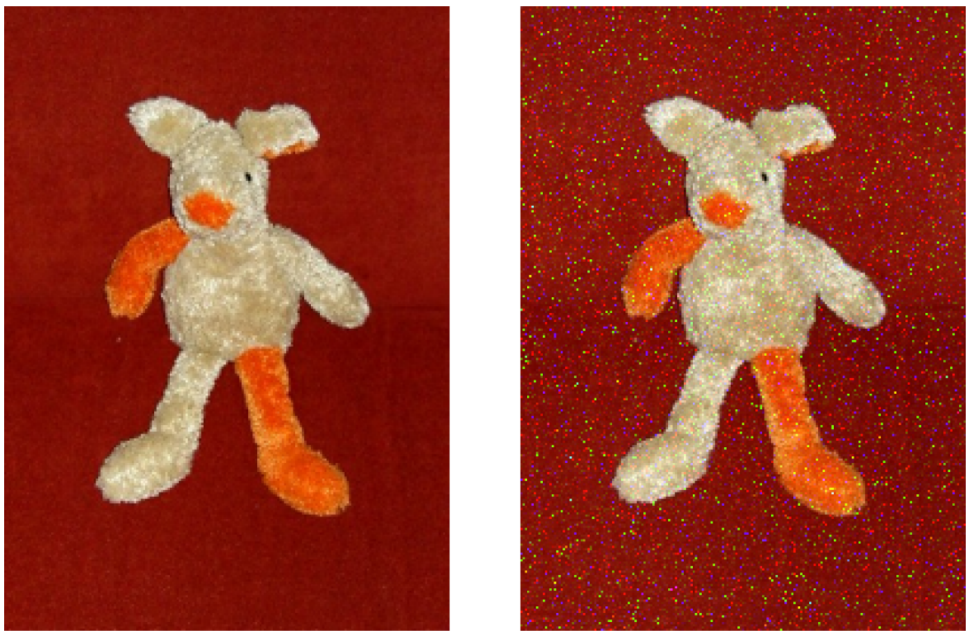
Fig. 2 Images from the NASA dataset. The image in the first panel has been used in the Dataset Creation phase of Algorithm 2 (Colour figure online)

Table 3 Best configuration $h^\star$ obtained by Algorithm 2 on the NASA dataset

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

Table 4 Best configuration $h^\star$ obtained by Algorithm 2 on the BSDS500 dataset

σ	C	# Stages	#filters (skip)	#filters (conv)	λ	Optimizer	lr
0.03	16	6	2	128	0.0001	Adam	0.01



(a) Original image (b) Corrupted image

Fig. 3 Representative image from the BSDS500 dataset. The original image has been used in the Dataset Creation phase of Algorithm 2 (Colour figure online)

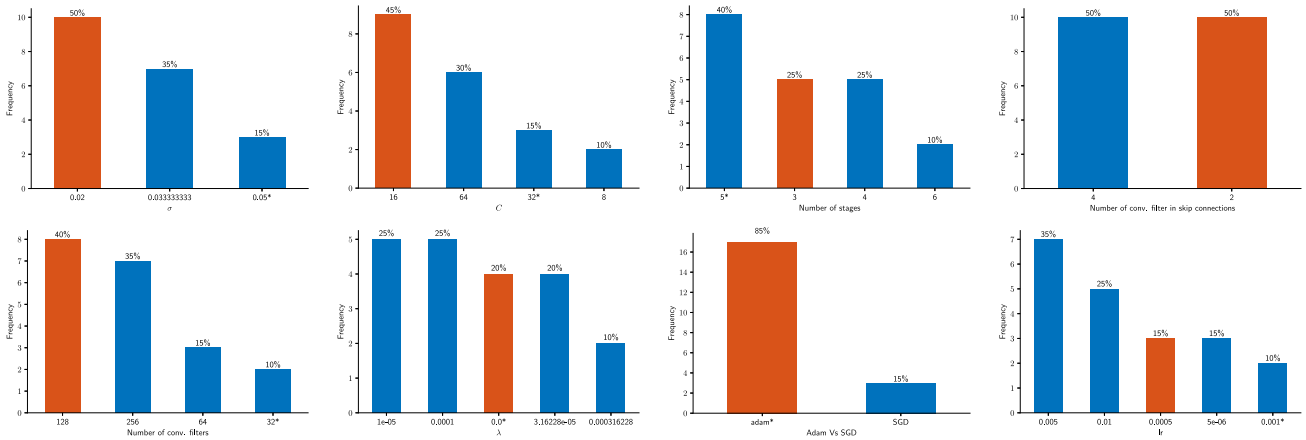


Fig. 4 Histograms of the hyperparameters' distributions among the 20 best configurations. The asterisk * in the labels denotes the setting for the vanilla DIP, while the orange color shows the chosen value by the proposed procedure (Colour figure online)

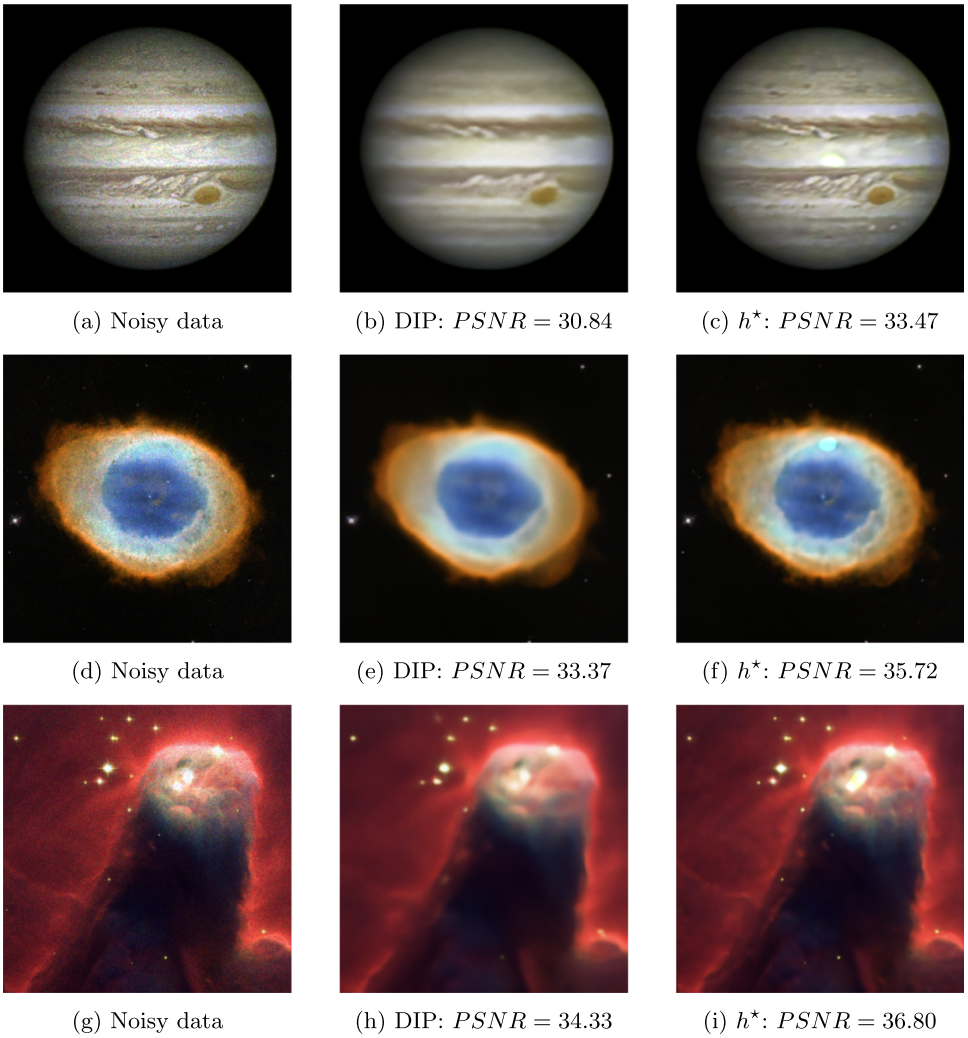
Table 5 Average PSNR and standard deviation for NASA test images recovered by both vanilla DIP and DIP with the optimal configuration obtained by Algorithm 2 after 1000 iteration

Vanilla DIP		Best Config	
Average	Std. dev.	Average	Std. dev.
31.5899	3.2699	34.6645	3.0528

results of Table 5 clearly validate the observations made on Fig. 5.

We remark that the possibility of a semi-convergence behaviour is a priori excluded, as Algorithm 2 systematically explores effective configurations within the given N iterations. Finally, we observe that, for this problem, only 1000 iterations for Algorithm 1 with the standard DIP hyperparameters are not sufficient. Indeed the poor results reported in Fig. 5 for the vanilla DIP method are not due to a semi-convergence effect. As a consequence, one might think that simply increasing the learning rate for the optimization algorithm would suffice to outperform standard DIP by keeping the same number of iterations. However, this thesis is contradicted by Table 3, where it is clear that the learning rate for $h^\star$ is even smaller than that employed in vanilla DIP. This observation highlights the advantage of using a NAS technique to jointly optimize the DIP hyperparameters.

Results for the BSDS500 dataset. In Table 6 we present the average PSNR (and the corresponding standard deviation) obtained in 1000 iterations by applying Algorithm 1



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

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

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

Fig. 5 Results on images from Fig. 2 after 1000 iterations. Left panel: noisy image. Middle panel: vanilla DIP results. Right panel: image obtained via the best configuration $h^\star$ for the hyperparameters reported in Table 3

Table 6 Average PSNR and standard deviation for BSDS500 test images recovered by both vanilla DIP and DIP with the optimal configuration obtained by Algorithm 2 after 1000 iteration

γ	Vanilla DIP		Best config	
	Average	Std. dev.	Average	Std. dev.
5	26.4944	2.5594	25.8300	2.7119

combined with both the standard DIP hyperparameters configuration and the $h^\star$ one (found by Algorithm 2 and reported Table 4) on 9 images of the BSDS500 dataset, different from the representative one. These nine images are the same ones used in the final phase of Algorithm 3, as described in Sect. 4.1.2. This approach enables a direct comparison between Algorithms 2 and 3 on the same dataset.

The results of this experiment indicate that the performance achieved by DIP combined with the best configuration is lower than that of the original DIP. This underperformance is likely due to the high degree of specialization on the selected representative image, which does not sufficiently capture the diversity of the BSDS500 dataset. This experiment confirms that Algorithm 2 is well-suited for those datasets where images share common features, as observed in Sect. 3.1 when the algorithm was introduced. Examples include datasets of astronomical or medical images. However, for the BSDS500 dataset, Algorithm 3 appears more appropriate, as it allows to establish hyperparameters configurations optimized for each image. Indeed Algorithm 3 includes in the search subspace several images from the dataset.

4.1.2 Evaluation of Algorithm 3

The second numerical experiment regards Algorithm 3 and it is carried out on the BSDS500 dataset, whose images have been corrupted by Cauchy noise.

For the **Set up** phase of Algorithm 3 we decided to fix the standard deviation for Gaussian input noise to $\sigma = 1/20$ and the number of channels of the input to $C = 32$ as in

vanilla DIP approach, since the BRISQUE features increase the dimension of the search space Ω. The space of the hyperparameters is reported in Table 7. We set again $N = 1000$, $M = 100$ and $B = 3$, while the training set and the test set are such that $|\mathcal{T}_r| = 120 = 40 \times 3$ and $|\mathcal{T}_s| = 9$. Particularly, the training set is formed by taking 40 images from the BSDS500 dataset; each one is then corrupted by Cauchy noise at levels $\gamma = 1, 5, 10$, yielding then a grand total of 120 images. As for Algorithm 2, $c = 3$ and $t_1 = 50$, $t_2 = 100$, $t_3 = 150$; on the other hand, s is selected such as $Q = 360$.

The **Dataset Creation** phase encompasses the generation of the dataset needed to train the performance predictor. We recall that each row of the dataset refers to a particular image in the training set and a particular hyperparameters configuration. Indeed it consists of the 36 features $F^{(f)}$ of the considered image belonging to $\mathcal{T}_r$, the configuration $h_q^{(f)}$ of 6 hyperparameters from Table 7, the 4 metrics at $c = 3$ checkpoints stored in $P^{(f,q)}$ and the label $p_N^{(f,q)}$, namely the final performance corresponding to $(F^{(f)}, h_q^{(f)}, P^{(f,q)})$.

For the **Training** phase we consider the predictor ρ as in the previous numerical experiment, that is the prediction is the averaged values provided by a RF and a SVR algorithms trained on the dataset $\{(F^{(f)}, h_q^{(f)}, P^{(f,q)}), p_N^{(f,q)}\}_{f=1,\dots,|\mathcal{T}_r|, q=1,\dots,Q}$. The **Qualification Phase** requires, for each image in the test set $\mathcal{T}_s$, to run Algorithm 1 under $M = 100$ different configurations for 150 epochs: we predict thus $\{\tilde{p}_N^{(f,m))}\}_{f=1,\dots,|\mathcal{T}_r|, m=1,\dots,M}$ values, namely the final performance of the 12,000 configurations.

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

Figure 6 shows two of the nine test images of $\mathcal{T}_s$ used to evaluate the performance of Algorithm 3. Figures 7 and 8 depict the achieved results in 1000 iterations with the vanilla DIP and the different best configurations. In detail, the first column reports the noisy images with different level of noise, the second column illustrates the recovered image via the vanilla DIP and the last one the result gained by DIP equipped with the optimal configuration provided by the performance predictor. The best predicted configurations for the images shown in Figs. 7 and 8 and corrupted by different level of Cauchy noise are listed in Table 8. It is evident that classical DIP has a large smoothing behaviour, while an optimal architecture is able to restore more sharp details and visual

Table 9 Average PSNR of the reconstruction for each Cauchy noise level

γ	Vanilla DIP		Best config	
	Average	Std. dev.	Average	Std. dev.
1	28.781111	3.923883	**30.098889**	3.319595
5	26.494444	2.559361	**26.906667**	2.130710
10	24.122222	1.727156	**24.141111**	1.488216

The bolded measures depict the best results

feature of the image. We observe that in those results, the optimal configuration is similar to the vanilla DIP, but the number of stages is lower with a greater number of filters for the convolutional layers.

Finally, in Fig. 9 and in Table 9 we summarize the results obtained for all the test images considered in this second experiment. Particularly, we depict a comparison among the PSNR values obtained by DIP combined either with the original hyperparameters configuration or the best one provided by Algorithm 3. In Fig. 9, the index identifying the test image is reported on the x-axis, the dots correspond to the PSNR values and the lines represent the averaged PSNR value computed on the nine test images. From Fig. 9a, b it is evident that DIP equipped by the the configuration $h^\star$ outperforms the standard DIP in terms of PSNR for all the test images corrupted by Cauchy noise with both $\gamma = 1$ and $\gamma = 5$. For $\gamma = 10$, the results of the two approaches are comparable (see Fig. 9c). Analogous conclusions can be drawn from Table 9. Finally we remark that, comparing Table 6 and the row of Table 9, corresponding to $\gamma = 5$, for the BSDS500 dataset Algorithm 3 is more efficient than Algorithm 2 in equipping DIP with an appropriate hyperparameters configuration.

4.2 BRISQUE-based early stopping

This section is devoted to evaluate the effectiveness of the BRISQUE-based early stopping technique, described in Algorithm 4, in preventing the semiconvergence behaviour typical of the original DIP approach. The collection $\mathcal{C}$ needed in Algorithm 4 has been built starting from both the NASA and the BSDS500 datasets considered in the previous numerical experiments. In particular we take into account 8 images from the NASA dataset, corrupted by Poisson noise, and 40 images from the BSDS500 dataset, each one perturbed by three different noise levels, i.e. $\gamma = 1, 5, 10$. For each of these images, Algorithm 1 has been applied with $N = 10{,}000$ and, every 50 iterations, the PSNR value of the current reconstruction and the corresponding features vector $F^{(f)}$ have been stored. By means of this phase the dataset $\{F^{(f)}, \text{PSNR}(I_f)\}_{f=1,\dots,|\mathcal{C}|}$ required to train cBRISQUE has been created. We remark that we consider the collection $\mathcal{C}$ as described before because we have already available the

Table 7 Space of the hyperparameters for Algorithm 3

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

The values in bold are the default ones for the vanilla DIP

(a) (b)

Fig. 6 Test images from the BSDS500 dataset

Table 8 Best configurations provided by Algorithm 3 with $N = 1000$ iterations for the test images in Fig. 6 corrupted by Cauchy noise of several levels

Test image	lr	Optimizer	#filters (conv)	#filters (skip)	#stages	λ
Fig. 6a, $\gamma = 1$	0.001	Adam	256	8	3	0
Fig. 6a, $\gamma = 5$	0.001	Adam	128	4	3	0
Fig. 6a, $\gamma = 10$	0.001	Adam	128	4	3	0
Fig. 6b, $\gamma = 1; 5$	0.001	Adam	256	2	3	0
Fig. 6b, $\gamma = 10$	0.001	Adam	256	8	3	0

checkpoint images obtained by running Algorithm 1 for the 8 NASA images and the 120 BSDS500 images due to the previous numerical experiments. However, the collection $\mathcal{C}$ can be created on other available couples of corrupted images and related ground truths. We also stress that the customized BRISQUE metrics we trained can now be employed to properly stop Algorithm 1 when used to reconstruct any other image. In particular, the trained customized BRISQUE metrics (cBRISQUE$_i$, $i = 1, 2, 3$) will be made available upon suitable request to the authors.

For the last phase of Algorithm 4, we select the dataset $\mathcal{I}$ as the one consisting of the images of the NASA and BSDS500 datasets excluded from the creation of $\mathcal{C}$ and corrupted by Poisson noise, and Cauchy noise with $\gamma = 1, 5, 10$, respectively. Before presenting the results obtained with Algorithm 4, we first show the results achieved using the original DIP scheme (Algorithm 1), both without early stopping and with a naive early stopping based on the standard BRISQUE metric.

In Table 10 we report the averaged value of the PSNR obtained by Algorithm 1 on the test dataset $\mathcal{I}$ at different number of iterations. The semiconvergence effect of DIP is clearly highlighted by the results shown in Table 10. A good performance of Algorithm 1 can be achieved between 1800 and 3000 iterations. Of course this evaluation can be only deduced a posteriori, once the PSNR has been computed with respect to the ground truths.

In Table 11 we report the results obtained by Algorithm 1 stopped by means of a criterion based on the original BRISQUE. In more detail Algorithm 1 has been stopped when the BRISQUE value of the approximated images does not improve for τ successive iterations.

Based on the results presented in Table 11, we can conclude that an early stopping procedure based on the original BRISQUE metric is not effective. Indeed, regardless the value of the patience, Algorithm 1 is stopped too early, without approaching the optimal PSNR window.

Table 12 show the results obtained by applying Algorithm 4 with different values of the patience τ. The average PSNR

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

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

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

Fig. 7 Results achieved on Fig. 6a corrupted by Cauchy noise for $\gamma = 1$ (top row), $\gamma = 5$ (middle row) and $\gamma = 10$ (bottom row) via standard DIP (second column) and DIP with the best configuration reported in Table 8 (third column)

Table 10 Average PSNR and standard deviation achieved by Algorithm 1 on the test images employed to evaluate Algorithm 4

Number of iterations	Average PSNR	PSNR standard deviation
1000	27.04	3.78
1800	28.18	4.27
3000	28.28	5.25
5000	26.92	6.28
7000	25.50	5.74
10,000	24.82	5.21

Table 11 Results achieved by Algorithm 1 stopped by means of a criterion based on the original BRISQUE approach on the test set employed to evaluate Algorithm 4

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

τ is the value of the patience and $\bar{N}$ is the average number of iterations

values do not significantly vary with τ, by showing a very robust behaviour of Algorithm 4 with respect to the patience. Moreover it is possible to appreciate that Algorithm 4 allows to stop the DIP procedure in a number of iterations very close to the optimal one suggested from the results of Table 10. Indeed, the average PSNR value provided by Algorithm 4 are in line with the optimal ones reported in Table 10. Moreover, we compare the achieved results with the one obtained via the Windowed Moving Variance (WMV) technique presented in Wang et al. (2023), where the maximum number of iterations is set to 4000. The results provided by the WMV method are presented in Table 12. The proposed modified BRISQUE strategy with a patience of $\tau = 5$ provides an average PSNR similar to the one achieved by WMV in a comparable average number of iterations .

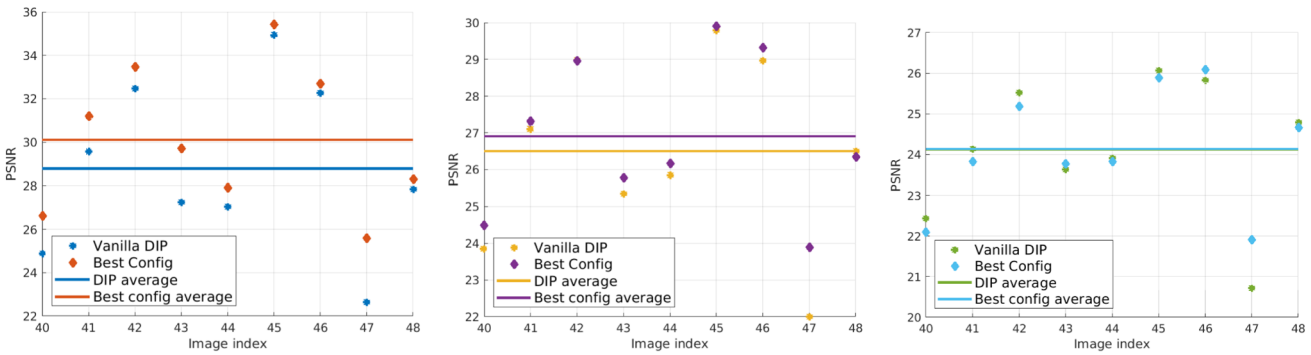
In Fig. 10 we report the PSNR curves predicted by cBRISQUE_1, cBRISQUE_2 and cBRISQUE_3 when running the DIP procedure on four different images of the test dataset $\mathcal{I}$ corrupted by different type and level of noise. Both real

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

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

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

Fig. 8 Results achieved on Fig. 6b corrupted by Cauchy noise for $\gamma = 1$ (top row), $\gamma = 5$ (middle row) and $\gamma = 10$ (bottom row) via standard DIP (second column) and DIP with the best configuration reported in Table 8 (third column)



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

Fig. 9 Results on test images from BSDS500 dataset with Cauchy noise

PSNR and BRISQUE values corresponding to the different iterates have been also shown. All the three customized PSNR predictors follow a trend very similar to that of the original PSNR, differently from the BRISQUE metric.

Finally in Fig. 11 we present the reconstruction of one of the corrupted images of the dataset $\mathcal{I}$ obtained by applying both Algorithm 1 with varying number of iterations and Algorithm 4. In terms of PSNR, the best approximation has been provided by Algorithm 4 which performed 1900 iterations. A very similar image has been provided by Algorithm 1 in 1800 iterations. This example shows a very promising behaviour of Algorithm 4: indeed the DIP approach has been optimally stopped without knowing the ground truth.

5 Conclusions

The selection of proper values for the hyperparameters, including the number of iterations or epochs, in the context of training efficiently an artificial neural network is crucial.

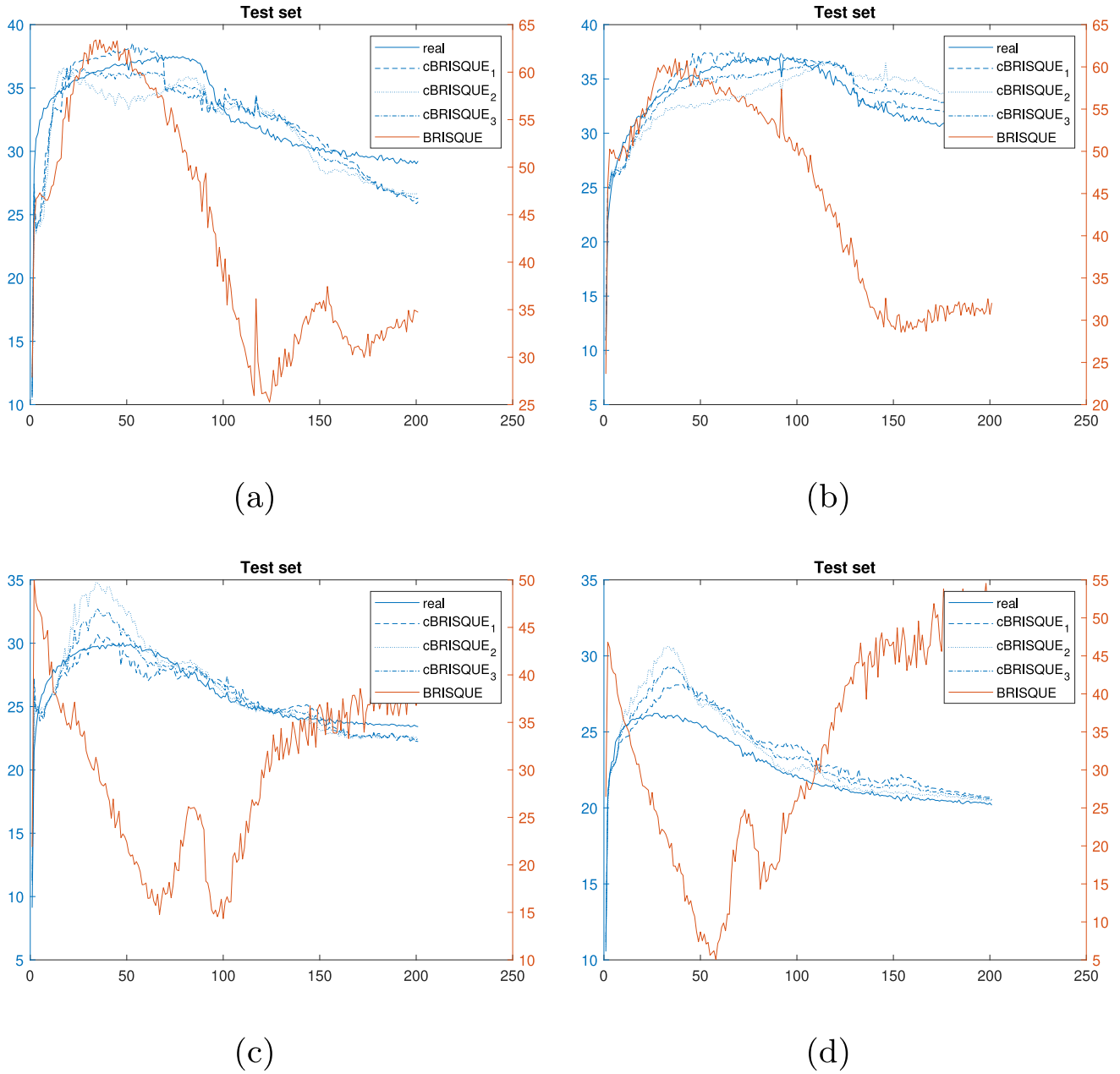
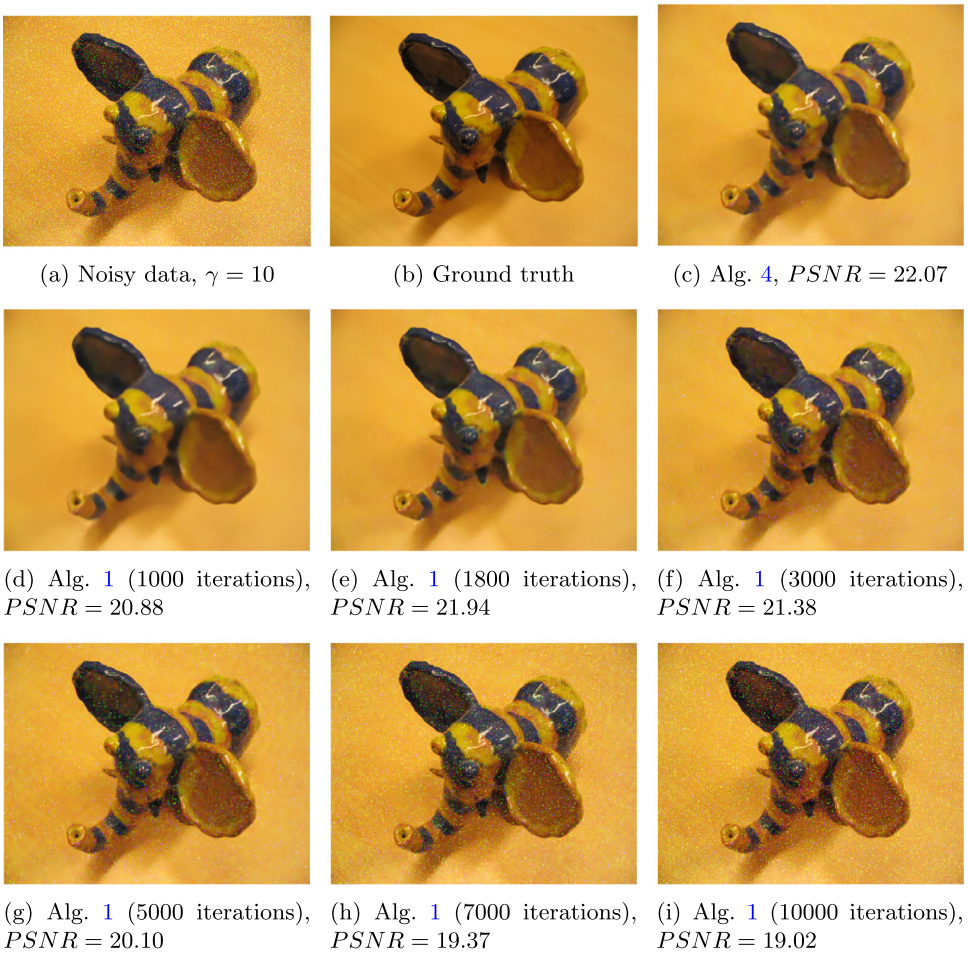


Fig. 10 PSNR curves predicted by cBRISQUE_1, cBRISQUE_2 and cBRISQUE_3 compared to the real PSNR and the BRISQUE values on four different images of the test dataset

This usually meets a very expensive numerical investigation, so we proposed two early stopping procedures for the Deep Image Prior framework for the image denoising task. The results demonstrate the effectiveness of both the approaches. The NAS-based strategy through the performance prediction shows indeed how to properly set both the network architecture (number of layers, filters, …), the training algorithm and also the regularization parameter for the loss function when DIP is applied under an early stopping framewok. The second approach, namely the customized BRISQUE metric, shows how to early stop the training of the vanilla DIP network, i.e. the network with the structure presented in the original work. This approach can be easily generalized to other network architectures and not only to U-net structures.

A Details on BRISQUE

In this section we give some details on the feratures extracted to compute the BRISQUE score as proposed in Mittal et al. (2012). BRISQUE does not rely on features related to particular distortions, such as ringing, blur, or blocking artifacts.



(a) Noisy data, $\gamma = 10$ (b) Ground truth (c) Alg. 4, $PSNR = 22.07$

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

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

Fig. 11 Results achieved by Algorithm 1 (at different iterations) and Algorithm 4 on a test image corrupted by Cauchy noise with $\gamma = 10$. The first row illustrates the input noisy data, the ground truth and the reconstruction reached by Algorithm 4 (1900 iterations). Images from **d**–**i** report the DIP reconstructions. The corresponding PSNR values are also reported

Table 12 Results achieved by Algorithm 4 on the test set

τ	$\bar{N}$	Average PSNR	PSNR standard deviation
5	1600	27.51	4.15
10	3150	27.52	4.85
15	3350	27.29	4.75
20	3500	27.62	4.79
Windowed moving variance (Wang et al. 2023)			
1000	1616	27.55	3.69

τ is the value of the patience and $\bar{N}$ is the average number of iterations. The last row presents the results obtained by the technique in Wang et al. (2023)

Instead, it leverages scene statistics from locally normalized luminance coefficients to assess potential deviations from the image's "naturalness" caused by distortions, resulting in a comprehensive quality measure. The employed features are derived from the empirical distribution of locally normalized luminance values and the products of these values, based on a spatial natural scene statistics model.

Given an image represented as a matrix in $\mathbb{R}^{p\times q}$, the locally normalized luminances can be computed as follows:

$$\hat{g}(i,j) = \frac{g(i,j) - \mu(i,j)}{\sigma(i,j) + C}, \quad i = 1,\dots,p \quad j = 1,\dots,q \tag{7}$$

where $g(i,j)$ represents the intensity of the image associated with the pixel pair (i,j), C is a nonzero constant added to avoid computational instabilities that can arise if the denominator becomes very close to zero, μ is the local mean and σ is the local variance, respectively defined as

$$\mu(i,j) = \sum_{k=-K}^{K}\sum_{l=-L}^{L} w_{k,l} g_{k,l}(i,j),$$

$$\sigma(i,j) = \sum_{k=-K}^{K}\sum_{l=-L}^{L} w_{k,l} (I_{k,l}(i,j) - \mu(i,j))^2.$$

Here $w = \{w_{k,l} \,|\, k = -K,\dots,K, l = -L,\dots,L\}$ represents a 2D circularly-symmetric Gaussian weighting function sampled out to 3 standard deviations and rescaled to unit

Table 13 18 features of the BRISQUE procedure

Feature	Description	Calculation procedure
f_1, f_2	σ, α	Optimal parameters for GGD from MSCN
$f_3 - f_6$	$\sigma_l, \sigma_r, \nu, \eta$	Optimal parameters for AGGD from pairwise product H
$f_7 - f_{10}$	$\sigma_l, \sigma_r, \nu, \eta$	Optimal parameters for AGGD from pairwise product V
$f_{11} - f_{14}$	$\sigma_l, \sigma_r, \nu, \eta$	Optimal parameters for AGGD from pairwise product D1
$f_{15} - f_{18}$	$\sigma_l, \sigma_r, \nu, \eta$	Optimal parameters for AGGD from pairwise product D2

volume. The luminances $\hat{g}(i, j)$ defined in (7) will be hereafter referred to as Mean Subtracted Contrast Normalized (MSNC) coefficients.

The MSNC values strongly tend towards a unit normal Gaussian characteristic for undistorted natural images (Ruderman 1994). On the other hand, each distortion modifies the statistics in its own characteristic way and the corresponding MSCN coefficients locally exhibit different distributions. In Mittal et al. (2012), the authors claimed that a generalized Gaussian distribution (GGD) can be used to effectively capture a broader spectrum of distorted image statistics. The probability density function of the GGD with zero mean is defined as:

$$f(x; \alpha, \sigma^2) = \frac{\alpha}{2\beta\Gamma(1/\alpha)} \exp\left(-\left|\frac{x-\mu}{\beta}\right|^{\alpha}\right), \tag{8}$$

where $\beta = \sigma\sqrt{\frac{\Gamma(1/\alpha)}{\Gamma(3/\alpha)}}$ is the scale parameter and $\Gamma(\cdot)$ represents the gamma function defined as $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1}e^{-t}dt$ for $\alpha > 0$. The parameter α controls the shape of the distribution while σ^2 controls the variance. The zero-mean distribution is chosen due to the typical symmetry observed in MSCN coefficient distributions. The parameters of the GGD, α and σ^2, are estimated using a moment-matching approach, as proposed in Sharifi and Leon-Garcia (1995), in order to fit the empirical distributions of the MSCN coefficients. The parameters α and σ^2 constitute the first set of features used to characterize image distortion.

To further quantify the loss of image naturalness, the statistical relationships between neighboring pixels can be also captured. In undistorted images the signs of adjacent MSCN coefficients follow a regular pattern that is instead disrupted by distortions. In Mittal et al. (2012), this pattern is modeled using the empirical distributions of pairwise products of neighboring MSCN coefficients, considering four orientations: horizontal (H), vertical (V), main-diagonal (D1), and secondary-diagonal (D2). Particularly,

$$\begin{aligned} H(i, j) &= \hat{g}(i, j)\hat{g}(i, j+1) \\ V(i, j) &= \hat{g}(i, j)\hat{g}(i+1, j) \\ D1(i, j) &= \hat{g}(i, j)\hat{g}(i+1, j+1) \\ D2(i, j) &= \hat{g}(i, j)\hat{g}(i+1, j-1) \end{aligned}$$

for $i = 1, \ldots, p$ and $j = 1, \ldots, q$. Assuming that MSCN coefficients follow a Gaussian model with zero mean and unit variance, in the absence of distortions, the pairwise products defined above follow the following distribution (Nuttall 1983):

$$f(x, \rho) = \frac{e^{\frac{|x|\rho}{1-\rho^2}} K_0\left(\frac{|x|}{1-\rho^2}\right)}{\pi\sqrt{1-\rho^2}},$$

where f is an asymmetric probability density function, ρ denotes the correlation coefficient of adjacent coefficients, and K_0 is the modified Bessel function of the second kind. Since this density function does not provide a good fit for the coefficient product histograms from distorted images, in Mittal et al. (2012) the authors adopt the model of Asymmetric Generalized Gaussian Distribution (AGGD). The probability density function of AGGD with zero mode is given by:

$$f(x; \nu, \sigma_l^2, \sigma_r^2) = \begin{cases} \frac{\nu}{(\beta_l + \beta_r)\Gamma\left(\frac{1}{\nu}\right)} \exp\left(-\left(\frac{-x}{\beta_l}\right)^{\nu}\right), & \text{if } x < 0, \\ \frac{\nu}{(\beta_l + \beta_r)\Gamma\left(\frac{1}{\nu}\right)} \exp\left(-\left(\frac{x}{\beta_r}\right)^{\nu}\right), & \text{if } x \geq 0. \end{cases}$$

where $\beta_l = \sigma_l\sqrt{\frac{\Gamma(1/\nu)}{\Gamma(3/\nu)}}$ and $\beta_r = \sigma_r\sqrt{\frac{\Gamma(1/\nu)}{\Gamma(3/\nu)}}$.

The shape parameter ν determines the overall shape of the distribution, while σ_l^2 and σ_r^2 are scale parameters that regulate the spread on the left and right sides of the mode, respectively. The parameters of the AGGD, namely $\nu, \sigma_l^2, \sigma_r^2$, are estimated using the moment-matching based approach proposed in Lasmar et al. (2009). Together with these parameters, the authors also consider the additional parameter η given by

$$\eta = (\beta_r - \beta_l)\frac{\Gamma\left(\frac{2}{\nu}\right)}{\Gamma\left(\frac{1}{\nu}\right)}.$$

The parameters $(\sigma_l, \sigma_r, \nu, \eta)$ of the AGGD are computed for each of the four directions of pairwise products—$H, V, D1, D2$—yielding a total of 16 parameters. Adding the two parameters derived from the GGD brings the total to 18 features, as summarized in Table 13.

Finally, to define the BRISQUE metric in Mittal et al. (2012), the authors suggest to extract all features listed in Table 13 at two scales: the original image resolution and a reduced resolution, achieved by applying a low-pass filter and downsampling by a factor of 2. Hence, a total of 36 features are employed to detect distortions and obtain the no-reference image quality metric.

Acknowledgements F. Porta is supported by the the Italian MUR through the PRIN 2022 Project "Numerical Optimization withAdaptive Accuracy and Applications to Machine Learning", project code: 2022N3ZNAX (CUP E53D23007700006), under the National Recovery and Resilience Plan (PNRR), Italy, Mission 04 Component 2 Investment 1.1 funded by the European Commission - NextGeneration EU programme. F. Porta is supported by the the Italian MUR through the PRIN 2022 PNRR Project "Advanced optimization METhods for automated central veIn Sign detection in multiple sclerosis from magneTic resonAnce imaging (AMETISTA)", project code: P2022J9SNP (CUP E53D23017980001), under the National Recovery and Resilience Plan (PNRR), Italy, Mission 04 Component 2 Investment 1.1 funded by the European Commission - NextGeneration EU programme. A. Benfenati is supported by the Italian MUR through the PRIN 2022 Project "Sustainable Tomographic Imaging with Learning and rEgularization (STILE)", project code: 20225STXSB (CUP E53D23005480006). This work is partially supported by the GNCS project "Deep Variational Learning: un approccio combinato per la ricostruzione di immagini", project code: CUP E53C23001670001.

Author Contributions All authors contributed to the study conception, design, preparation and writing. All authors read and approved the final manuscript.

Funding Open access funding provided by Università degli Studi di Modena e Reggio Emilia within the CRUI-CARE Agreement. F. Porta is supported by the the Italian MUR through the PRIN 2022 Project "Numerical Optimization with Adaptive Accuracy and Applications to Machine Learning", project code: 2022N3ZNAX (CUP E53D23007700006), under the National Recovery and Resilience Plan (PNRR), Italy, Mission 04 Component 2 Investment 1.1 funded by the European Commission - NextGeneration EU programme. F. Porta is supported by the the Italian MUR through the PRIN 2022 PNRR Project "Advanced optimization METhods for automated central veIn Sign detection in multiple sclerosis from magneTic resonAnce imaging (AMETISTA)", project code: P2022J9SNP (CUP E53D23017980001), under the National Recovery and Resilience Plan (PNRR), Italy, Mission 04 Component 2 Investment 1.1 funded by the European Commission - NextGeneration EU programme. A. Benfenati is supported by the Italian MUR through the PRIN 2022 Project "Sustainable Tomographic Imaging with Learning and rEgularization (STILE)", project code: 20225STXSB (CUP E53D23005480006). This work is partially supported by the GNCS project "Deep Variational Learning: un approccio combinato per la ricostruzione di immagini", project code: CUP E53C23001670001.

Data availability Not applicable.

Code availability The code will be made available upon suitable request to the authors.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose. The authors have no Conflict of interest to declare that are relevant to the content of this article.

Ethical approval and consent to participate Not applicable.

References

Alshubaily I (2021) Efficient neural architecture search with performance prediction. arXiv:2108.01854

Arican ME, Kara O, Bredell G, Konukoglu E (2022) ISNAN-DIP: image-specific neural architecture search for deep image prior. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1960–1968

Benfenati A, Bolzi D, Causin P, Oberti R (2022) A deep learning generative model approach for image synthesis of plant leaves. PLoS ONE 17(11):0276972

Benfenati A, Catozzi A, Ruggiero V (2023a) Neural blind deconvolution with Poisson data. Inverse Probl 39:054003

Benfenati A, Catozzi A, Franchini G, Porta F (2023b) Piece-wise constant image segmentation with a deep image prior approach. In: International conference on scale space and variational methods in computer vision. Springer, pp 352–362

Bertero M, Boccacci P, Ruggiero V (2018) Inverse imaging with Poisson data: from cells to galaxies. IOP Publishing

Burrows L, Chen K, Torella F (2021) A deep image prior learning algorithm for joint selective segmentation and registration. In: International conference on scale space and variational methods in computer vision. Springer, pp 411–422

Charfi S, Ansari ME, Koutti L, Ellahyani A, Eljaafari I (2024) Modified residual attention network for abnormalities segmentation and detection in WCE images. Soft Comput 28:6923–6936

Chen Y-C, Gao C, Robb E, Huang J-B (2020) NAS-DIP: learning deep image prior with neural architecture search. In: Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII, vol 16. Springer, pp 442–459

Cheng Z, Gadelha M, Maji S, Sheldon D (2019) A Bayesian perspective on the deep image prior. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Elsken T, Metzen JH, Hutter F (2019) Neural architecture search: a survey. J Mach Learn Res 20(55):1–21

Franchini G, Ruggiero V, Porta F, Zanni L (2023) Neural architecture search via standard machine learning methodologies. Math Eng 5(1):1–21

Ho K, Gilbert A, Jin H, Collomosse J (2021) Neural architecture search for deep image prior. Comput Graph 98:188–196

Horé A, Ziou D (2010) Image quality metrics: PSNR vs. SSIM. In: 2010 20th international conference on pattern recognition, pp 2366–2369

Huo D, Masoumzadeh A, Kushol R, Yang Y-H (2023) Blind image deconvolution using variational deep image prior. IEEE Trans Pattern Anal Mach Intell 45(10):11472–11483

Kotera J, Šroubek F, Šmídl V (2021) Improving neural blind deconvolution. In: 2021 IEEE International Conference on Image Processing (ICIP). IEEE, pp 1954–1958

Lasmar NE, Stitou Y, Berthoumieu Y (2009) Multiscale skewed heavy tailed model for texture analysis. In: Proc. IEEE Int. Conf. Image Process., pp 2281–2284

Lin Y-C, Huang H-M (2020) Denoising of multi b-value diffusion-weighted MR images using deep image prior. Phys Med Biol 65(10):105003

Li H, Xu Z, Taylor G, Studer C, Goldstein T (2018) Visualizing the loss landscape of neural nets. In: Advances in neural information processing systems, vol 31

Li T, Zhuang Z, Liang H, Peng L, Wang H, Sun J (2021) Self-validation: early stopping for single-instance deep generative priors. In: 32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22–25, 2021. BMVA Press, p 108

Martin D, Fowlkes C, Tal D, Malik J (2001) A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. 8th Int'l Conf. Computer Vision, vol 2, pp 416–423

Mei J-J, Dong Y, Huang T-Z, Yin W (2018) Cauchy noise removal by nonconvex ADMM with convergence guarantees. J Sci Comput 74:743–766

Mittal A, Moorthy AK, Bovik AC (2012) No-reference image quality assessment in the spatial domain. IEEE Trans Image Process 21(12):4695–4708

Nuttall AH (1983) Accurate efficient evaluation of cumulative or exceedance probability distributions directly from characteristic functions. Naval Underwater Systems Center, New London, CT, Tech. Rep. ADA133703

Renuka N (2023) Semantic segmentation-based skin cancer detection. Soft Comput 27(16):11895–11903

Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-assisted intervention—MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III, vol 18. Springer, pp 234–241

Ruderman DL (1994) The statistics of natural images. Netw Comput Neural Syst 5(4):517–548

Sciacchitano F, Dong Y, Zeng T (2015) Variational approach for restoring blurred images with Cauchy noise. SIAM J Imaging Sci 8(3):1894–1922

Sharifi K, Leon-Garcia A (1995) Estimation of shape parameter for generalized gaussian distributions in subband decompositions of video. IEEE Trans Circuits Syst Video Technol 5(1):52–56

Sun R, Li D, Liang S, Ding T, Srikant R (2020) The global landscape of neural networks: an overview. IEEE Signal Process Mag 37(5):95–108

Top 100 Hubble Telescope Images. https://www.kaggle.com/datasets/redwankarimsony/top-100-hubble-telescope-images

Top 100 images ESA-Hubble. https://esahubble.org/images/archive/top100/

Torbunov D, Huang Y, Yu H, Huang J, Yoo S, Lin M, Viren B, Ren Y (2023) Uvcgan: Unet vision transformer cycle-consistent gan for unpaired image-to-image translation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp 702–712

Trebing K, Stańczyk T, Mehrkanoon S (2021) SmaAt-UNet: precipitation nowcasting using a small attention-UNet architecture. Pattern Recognit Lett 145:178–186

Ulyanov D, Vedaldi A, Lempitsky V (2018) Deep image prior. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 9446–9454

Ulyanov D, Vedaldi A, Lempitsky V (2020) Deep Image Prior. Int J Comput Vis 128:1867–1888

Wang H, Li T, Zhuang Z, Chen T, Liang H, Sun J (2023) Early stopping for deep image prior. Trans Mach Learn Res 2023. https://www.scopus.com/record/display.uri?eid=2-s2.0-86000639497&origin=inward&txGid=69e3899f7221f3143b89f40954ba7e44

Yoo J, Jin KH, Gupta H, Yerly J, Stuber M, Unser M (2021) Time-dependent deep image prior for dynamic MRI. IEEE Trans Med Imaging 40(12):3337–3348

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