SPyRiT: an open source package for single-pixel imaging based on deep learning

Juan F P J Abascal, Thomas Baudier, Romain Phan, Audrey Repetti, Nicolas Ducros

SPyRiT: an open source package for single-pixel imaging based on deep learning

J. Abascal[1], T. Baudier[1], R. Phan[1], A. Repetti[2], N. Ducros[1, 3, *]

[1] *Univ Lyon, INSA-Lyon, UniversitÃľ Claude Bernard Lyon 1, UJM-Saint Etienne, CNRS, Inserm, CREATIS UMR 5220, U1294, F-69621, LYON, France*
[2] *MACS & EPS, Heriot-Watt University, Maxwell Institute for Mathematical Sciences, Edinburgh, UK*
[3] *Institut universitaire de France (IUF), France*
[*] *nicolas.ducros@creatis.insa-lyon.fr*

Abstract: Single-pixel imaging is able to acquire an image from a few point measurements thanks to dedicated reconstruction algorithms. In recent years, reconstruction approaches based on deep learning have outperformed most alternatives. However, computational experiments and data-driven methods have become difficult, if not impossible, to reproduce. The development of tools enabling reproducibility and benchmarking is therefore now essential. This paper describes SPyRiT, an open source PyTorch-based toolbox capable of handling various simulation configurations and reconstruction methods based on deep learning. In particular, we compare several supervised and plug-and-play methods, including post-processing and iterative strategies. Our results demonstrate that supervised methods trained on simulated data can be successfully applied to experimental data when the signal-to-noise ratio of the measurements is higher or equal to that of the training phase. On the other hand, the hyperparameter of the plug-and-play methods can be tuned to manage lower signal-to-noise ratios. The modularity of SPyRiT enables the evaluation of various configurations and the rigorous benchmarking of reconstructions based on deep learning in single-pixel imaging, as well as in related fields such as ghost imaging.

1. Introduction

Single-pixel imaging is a technique that enables an image to be acquired from a set of point measurements using dedicated reconstruction algorithms [1, 2]. The technique can be traced back to the concept of Hadamard spectroscopy [3], that introduced Hadamard-modulated measurements to enhance the signal-to-noise ratio (SNR) of images reconstructed using the least squares method. Single-pixel imaging has gained renewed interest with the advent of compressed sensing theory, providing a theoretical framework to ensure perfect reconstruction of an image acquired from few measurements obtained with random modulations [4]. The many different applications of the technique include imaging in the visible and infrared domains [5], in addition to the mid-infrared [6], ultraviolet [7], terahertz [8] and X-ray [9] ranges, as well as through scattering media [10], fluorescence lifetime imaging [11], 3D time-of-flight imaging [12], time-of-flight LiDAR [13] and light-field microscopy [14].

Deep learning has revolutionized the field of image reconstruction [15] and greatly advanced computational optics [16]. Since the first application of deep learning to single-pixel imaging [17], a great variety of data-driven reconstruction methods have been designed for the field. Supervised methods train the free parameters of a reconstruction algorithm from image-measurement pairs. Simple yet efficient reconstruction algorithms are obtained by post-processing a first (e.g., linear) reconstruction by an image-domain neural network (e.g., a U-Net [18] or an attention network [19]). The first reconstruction step can be learned from the data in an end-to-end [17] or two-step [20] manner, chosen as the pseudo-inverse solution [21, 22], or the minimum mean squares error solution [23]. Following the trend of algorithm unrolling [24, 25], an empirical expectation maximisation network was designed in [26] while [27] considered unrolling the iterative hard thresholding for expanders. Adversarial training has also been considered [28, 29].

To improve the generalization ability of supervised methods, one strategy involves fine-tuning using a physics-based loss [30]. An alternative is to consider plug-and-play (PnP) methods that combine a (deep) denoiser prior with a model-based reconstruction algorithm. Multiple configurations (e.g., sampling ratio, noise level, type of modulation) can, therefore, be considered using the same denoiser, whereas supervised methods require retraining. Among the possible model-based reconstruction algorithms that can be used for PnP, the proximal gradient descent [31] or the alternating direction method of multipliers algorithm [32] have been widely used in imaging. Different types of denoisers can be considered, such as traditional (e.g., BM3D [33]) or U-Net based denoisers (e.g., U-Net [18], DR-UNet [34]), or auto-regressive models [35]. More recently, untrained networks have emerged that do not require pretraining [36, 37].

In this new data-driven deep learning era, reproducibility is a major challenge. Indeed, despite huge progress in both the practical and theoretical aspects, most computational experiments and data-driven methods have become difficult, if not impossible, to reproduce by an independent researcher [38], an issue referred to as the *reproducibility crisis*. Open source platforms with reference datasets and reconstruction algorithms are therefore required in order to ensure the reproducibility of results and fair benchmarking needed to establish reliable conclusions.

This work introduces SPyRiT, the first, to the best of our knowledge, package dedicated to single-pixel imaging reconstruction based on deep learning. Though packages for solving general inverse problems using deep learning are available (e.g., Pixu [39] or DeepInv [40]), dedicated packages are usually preferred when targeting specific applications (e.g., RTK [41], ASTRA [42] or TIGRE [43] in computed tomography; BART [44] in magnetic resonance imaging). SPyRiT, on the other hand, provides a platform for the simulation and reconstruction of single-pixel measurements, integrated within the OpenSpyrit ecosystem that provides data and hardware control software [45]. The modular structure of the Python-based package allows users to simulate measurements from images, simulate corrupted noisy measurements, process noisy measurements prior to reconstruction, define data-driven reconstruction algorithms and train the associated networks.

To illustrate the capabilities of SPyRiT, we implement and compare six different data-driven reconstruction methods, belonging to the supervised and PnP families of algorithms. All methods are assessed in a controlled manner by evaluation of the images reconstructed by SPyRiT from the ImageNet dataset and SPIHIM experimental dataset. While SPyRiT has been designed for single-pixel imaging, it may also benefit other computational optical imaging modalities, e.g., ghost imaging which is based on the same imaging principle.

2. Theory

2.1. Single-pixel imaging forward model

Single-pixel imaging aims to recover an unknown image $x \in \mathbb{R}^N$ from a few noisy observations

$$m \approx Hx, \tag{1}$$

where $H\colon \mathbb{R}^N \to \mathbb{R}^M$ is a linear measurement operator. In the case when $M = N$, it may be desirable to work with orthogonal bases, such as Hadamard or Fourier [46, 47] to benefit from Fellgett's advantage for improved SNR [3]. This implies patterns with negative values[1]. In practice, however, measurements are obtained by uploading a set of light patterns onto a spatial light modulator (e.g., a digital micromirror device (DMD), see Fig. 1). Therefore, only positive patterns can be implemented. We model the actual acquisition process as

$$y = \mathcal{N}(Ax), \tag{2}$$

[1]An orthogonal matrix with non-negative entries is a permutation matrix, i.e., a matrix that has only one non zero entry (equal to one) in each row. This case corresponds to measuring each pixel in the scene independently

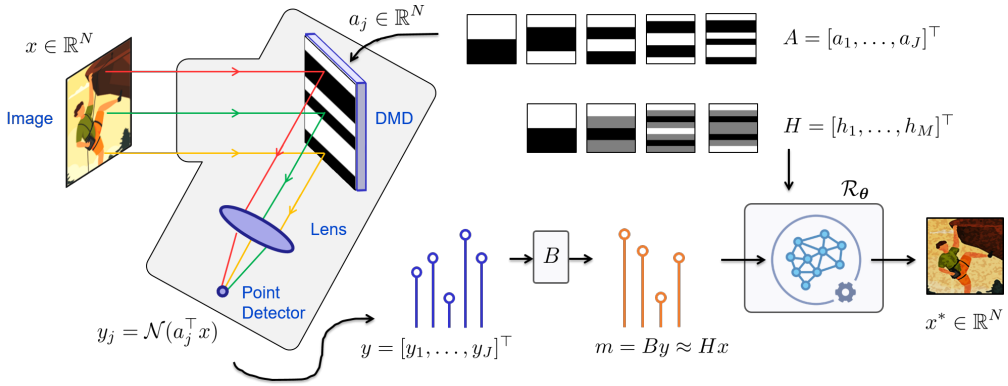

Fig. 1. Principle of single-pixel imaging. A single-pixel camera measures a noisy version of the scalar product of some image $x \in \mathbb{R}^N$ and a DMD pattern $a_j \in \mathbb{R}^N$. The measurement vector $y \in \mathbb{R}^J$, that collects all measurements obtained for all DMD patterns, may be preprocessed, leading to the preprocessed measurement vector $m \in \mathbb{R}^M$. A reconstruction algorithm $\mathcal{R}_\theta$ is necessary to estimate the unknown image x from the preprocessed measurements.

where $\mathcal{N}\colon \mathbb{R}^J \to \mathbb{R}^J$ represents a (not necessarily linear) noise operator (e.g., Poisson or Poisson-Gaussian), $A\colon \mathbb{R}^N \to \mathbb{R}^J$ is the actual acquisition operator that models the (positive) DMD patterns, and J is the number of DMD patterns.

2.2. Handling non negativity with preprocessing

We may consider preprocessing techniques that transform the actual model Eq. (2) into the target model Eq. (1)

$$m = By \approx Hx, \tag{3}$$

where $B\colon \mathbb{R}^J \to \mathbb{R}^M$ is the (linear) preprocessing operator chosen such that $BA = H$. Note that the noise of the preprocessed measurements $m = By$ is not the same as that of the actual measurements y.

In the simple case when H has only non-negative entries, we can choose $A = H$ and B is the identity matrix. There exist multiple variants to build the preprocessing operator such that $BA = H$ when H has negative entries (e.g., see [48]). A simple workaround consists in splitting the positive and negative entries of the matrix. Alternatively, one can add an appropriate constant to the patterns, which usually leads to increased noise.

2.3. Data-driven reconstruction

In the presence of noise and when fewer measurements than unknowns are available (i.e., $M < N$), regularized approaches must be adopted (see Appendix 5). Data-driven methods based on deep learning are a modern regularized approach. They aim to find an estimate $x^* \in \mathbb{R}^N$ of the unknown image x from the preprocessed measurements By (see Eq. (3)), using a reconstruction operator $\mathcal{R}_{\theta^*}\colon \mathbb{R}^M \to \mathbb{R}^N$. Specifically,

$$\mathcal{R}_{\theta^*}(m) = x^* \approx x, \tag{4}$$

where θ^* represents the parameters learned during a training procedure.

In the case of supervised learning, it is assumed that a training dataset $\{x_i, y_i\}_{1 \leq i \leq I}$ of pairs of ground truth images and measurements is available. θ^* is then obtained by solving

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{I} \mathcal{L}\big(x_i, \mathcal{R}_\theta(y_i)\big), \tag{5}$$

where $\mathcal{L}$ is the training loss (e.g., squared error). It should be noted that if only a dataset of ground truth images $\{x_i\}_{1 \leq i \leq I}$ is available, the associated set of measurements can be obtained by defining $y_i = \mathcal{N}(Ax_i)$, $1 \leq i \leq I$.

The design of the reconstruction operator $\mathcal{R}_\theta$ has been the topic of extensive research over the last decade. A simple yet efficient method consists in correcting a traditional (e.g. linear) reconstruction by a data-driven nonlinear step [49]. In this context,

$$\mathcal{R}_\theta = \mathcal{G}_\theta \circ \mathcal{R}, \tag{6}$$

where $\mathcal{R} \colon \mathbb{R}^M \to \mathbb{R}^N$ is a traditional hand-crafted (e.g., regularized) reconstruction operator and $\mathcal{G}_\theta \colon \mathbb{R}^N \to \mathbb{R}^N$ is a nonlinear neural network that acts in the image domain. Algorithm unfolding can be seen as a generalization of Eq. (6), which consists in defining $\mathcal{R}_\theta$ from an iterative scheme

$$\mathcal{R}_\theta = \mathcal{R}_{\theta_K} \circ \ldots \circ \mathcal{R}_{\theta_1}, \tag{7}$$

where $\mathcal{R}_{\theta_k}$ can be interpreted as the computation of the k-th iteration of the iterative scheme and $\theta = \bigcup_k \theta_k$. Many variants of unfolded algorithms have been proposed in the literature [25]. The choice of R_{θ_k} depends on the choice of the algorithm that is unrolled. It generally involves one or several analytical steps that rely on A or H, their adjoint operators (or pseudo-inverse) as well as one or several neural networks. Some standard architectures for $\mathcal{R}_{\theta_k}$ are described in Section 2.4. In particular, the formulation of Eq. (7) encompasses PnP and regularization by denoising methods, for which the neural networks/denoisers involved in $\mathcal{R}_{\theta_k}$ are trained beforehand, independently of A or H.

2.4. Standard data-driven architectures

A simple choice for the first step of the post-processing approach defined by Eq. (6) is to use the pseudo-inverse of H, i.e.,

$$\mathcal{R}(m) = H^\dagger m. \tag{8}$$

An alternative is to use the minimum mean square estimator [50], i.e.,

$$\mathcal{R}(m) = \Sigma H^\top \left(H \Sigma H^\top + \Gamma\right)^{-1} m, \tag{9}$$

where $\Sigma \in \mathbb{R}^{N \times N}$ represents the image covariance and $\Gamma \in \mathbb{R}^{M \times M}$ the noise covariance. These two methods are further detailed in Appendix A.1 and Appendix A.2.

A simple choice for the iterative method described in Eq. (7) is the proximal gradient descent [51] for solving the regularized least-squares problem (see Appendix A.1),

$$x_{k+1} = \mathcal{R}_{\theta_k}(x_k) = \mathcal{G}_{\theta_k}\left(x_k - \gamma_k H^\top (H x_k - m)\right), \tag{10}$$

where the proximal denoiser has been replaced by a neural network $\mathcal{G}_{\theta_k} : \mathbb{R}^N \to \mathbb{R}^N$ acting in the image domain and $\gamma_k > 0$ is the step size.

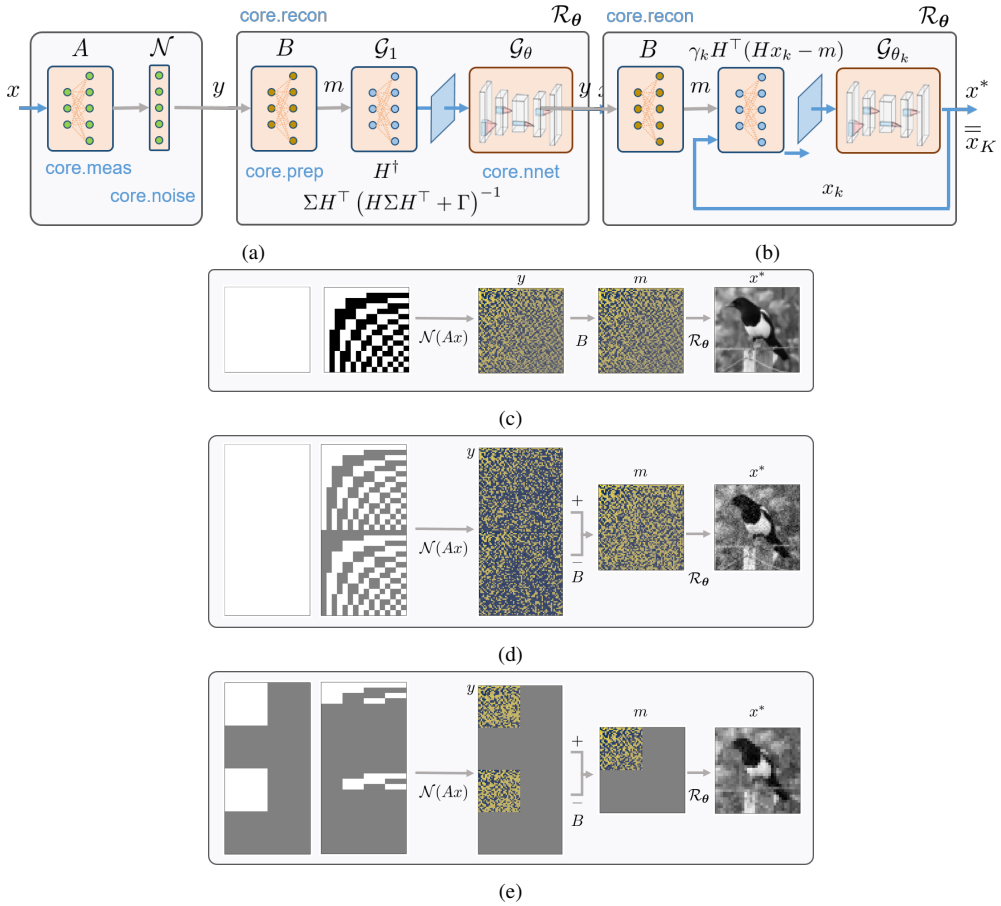

Fig. 2. Typical SPyRiT pipelines involving input of an image $x \in \mathbb{R}^N$, simulation of modulated measurements Ax, application of a noise model $\mathcal{N}$ which leads to the raw measurements $y \in \mathbb{R}^J$. From these, the reconstruction operator $\mathcal{R}_\theta$ outputs an estimate x^* of x. **(a)** Post-processing reconstruction approaches corresponding to Eq. (6). All steps are clearly separated and defined independently in the submodules `core.meas`, `core.noise`, and `core.recon`. The reconstruction algorithm can integrate preprocessing from `core.prep`, analytical steps and neural networks from `core.nnet`. **(b)** Iterative reconstruction approaches corresponding to Eq. (7). **(c)** Hadamard acquisitions which cannot be implemented in practice due to negative values in the target acquisition matrix. **(d)** Split Hadamard patterns defined by Eq. (12) with $M = N$ and the corresponding preprocessing $B = [I_N, -I_N]$. Poisson noise is simulated with $\alpha = 100$. **(e)** Square subsampling for split Hadamard acquisition, i.e., Eq. (12) with $M < N$ and the corresponding preprocessing $B = [I_M, -I_M]$. No noise is simulated. **(c-e)** The images on the left correspond to sampling masks. White masks correspond to the case $M = N$ where all measurements are kept, while grey pixels appear in the case of accelerated acquisitions where $M < N$. A sampling mask is a graphical representation of the permutation matrix Q in Eq. (13). Each white pixel of a sampling mask corresponds to the nonzero entry of each of the first M rows of Q. For simplicity, the $N = 64 \times 64$ acquisition matrices used are presented here as $N = 16 \times 16$.

3. Method

3.1. Overview of the SPyRiT package

SPyRiT has a modular structure with the core functionalities organized in six submodules (see Fig. 2) that provide classes that all inherit from PyTorch `nn.Module`. The module `spyrit.core.torch` provides additional auxiliary PyTorch-based functions used throughout the SPyRiT package.

- `spyrit.core.meas` provides measurement operators that compute linear measurements corresponding to A in Eq. (2). It also provides the adjoint and pseudo-inverse of A, which are at the basis of any reconstruction algorithm.
- `spyrit.core.noise` provides noise operators corresponding to $\mathcal{N}$ in Eq. (2).
- `spyrit.core.prep` provides preprocessing operators for the operator B introduced in Eq. (3). These preprocessing operators can also implement additional processing steps, such as the normalization of Poisson-corrupted images that vary across a wide range of intensities.
- `spyrit.core.nnet` provides neural network operators that correspond to $\mathcal{G}$ in Eq. (6) or Eq. (10).
- `spyrit.core.recon` provides the reconstruction operator that correspond to $\mathcal{R}_\theta$.
- `spyrit.core.train` provides the functionalities to solve the minimization problem of Eq. (5).

3.2. Poisson-corrupted split Hadamard measurements

To illustrate the capabilities of SPyRiT, we consider Poisson-corrupted split Hadamard measurements

$$\mathrm{y}_\alpha \sim \mathcal{P}(\alpha A x), \tag{11}$$

where α (in photons) represents the intensity of the image $x \in [0,1]^N$. The higher the intensity, the higher the SNR of the measurements. We chose the acquisition matrix $A \in \mathbb{R}_+^{2M\times N}$, $M \leq N$, such that the positive and negative components of the target matrix $H \in \mathbb{R}^{M\times N}$ are measured independently

$$A = \begin{bmatrix} H_+ \\ H_- \end{bmatrix}, \tag{12}$$

where $H_\pm = \max(0, \mp H) \in \mathbb{R}_+^{M\times N}$. Choosing $B = [I_N, -I_N]$, we recover $BA = H$. In the case of accelerated acquisitions for which $M < N$, we choose H as a subsampled (i.e., row-decimated) Hadamard matrix

$$H = [I_M, 0] Q H_\uparrow, \tag{13}$$

where $H_\uparrow \in \mathbb{R}^{N\times N}$ represents the full Hadamard matrix, $I_M \in \mathbb{R}^{M\times M}$ the identity matrix, and $Q \in \{0,1\}^{N\times N}$ a permutation matrix that ranks the Hadamard coefficients by significance. The entire process is illustrated in Fig. 2d and Fig. 2e. The acquisition matrix can be defined using the `meas.HadamSplit` class, while Poisson noise is handled via the `noise.Poisson` class.

3.3. Reconstruction algorithms

We consider algorithms that fall into different reconstruction strategies (e.g., optimization-based, post-processing and iterative methods, supervised or PnP methods). Given the modularity of SPyRiT, all methods can be easily defined by exploiting the acquisition and denoising operators. The selected methods are:

- Pseudo-inverse (Pinv), which corresponds to Eq. (8) with $\mathcal{G}_\theta$ being the identity.
- Pseudo-inverse with supervised denoising (Pinv-Net) [21], which corresponds to Eq. (8) where $\mathcal{G}_\theta$ is trained using Eq. (5).
- Pseudo-inverse with PnP denoising (Pinv-PnP), which corresponds to Eq. (8) where $\mathcal{G}_\theta$ is trained independently as a denoiser.
- Supervised denoised completion (DC-Net) [50], which corresponds to Eq. (9) where $\mathcal{G}_\theta$ is trained using Eq. (5).
- Learned proximal gradient descent (LPGD), which corresponds to Eq. (10) where $\mathcal{G}_\theta$ is trained using Eq. (5).
- Proximal gradient descent with PnP denoising (DPGD-PnP), which corresponds to Eq. (10), where $\mathcal{G}_\theta$ is trained independently as a denoiser.

The Pinv, Pinv-Net and Pinv-PnP methods are all instances of the same class `core.recon.PinvNet`, where the only difference is the denoising layer (the identity for Pinv, UNet [18] trained in a supervised manner for Pinv-Net, and pretrained DR-UNet [34] for Pinv-PnP). The LPGD and DPGD-PnP are also two instances of the same class `recon.LearnedPGD`. All algorithm details are provided in Section A.1. Note that both PnP methods (i.e., Pinv-PnP, DPGD-PnP) rely on a hyperparameter (ν for Pinv-PnP and μ for DPGD-PnP) that allows the denoising/regularization level to be manually tuned (see Appendix and Sec. 1.1 and 1.2 of Supplement 1).

3.4. Example code

The code required to run both the simulation and reconstruction with the `pinvNet` network is shown below.

```python
# Data simulation
import math
import torch
from spyrit.core.meas import HadamSplit
from spyrit.core.noise import Poisson

alpha = 100.0
h = math.sqrt(N)
meas_op = HadamSplit(M, h, Ord_rec)
noise_op = Poisson(meas_op, alpha)
torch.manual_seed(0)
y = noise_op(x) # x is vectorized image

# Image reconstruction
from spyrit.core.recon import PinvNet
from spyrit.core.nnet import Unet

prep_op = SplitPoisson(alpha, meas_op)
denoi_net = Unet()
full_op = PinvNet(noise_op, prep_op, denoi_net)
x_rec = full_op.reconstruct(y)

# Simulation and reconstruction
```

```
torch.manual_seed(0)
x_rec_2 = full_op(x)

(x_rec == x_rec_2).all() # True
```

The code that generates the masks, acquisition matrices and images in Fig. 2 is provided in the last section of Supplement 1. Multiple examples are provided in the tutorials accessible in [52].

3.5. Training

All supervised methods (i.e., Pinv-Net, DC-Net, LPGD) are trained by minimizing the mean squared error loss of Eq. (5) over the test dataset (100k images) of the ImageNet ILSVRC2012 database [53]. Color images are subject to grayscale transformation, with random crops of size 128×128 pixels and normalization to $[-1, 1]$ applied to all images. Measurements are simulated according to Eq. (11) for an image intensity $\alpha = 10$ photons. The minimization problem is solved using the Adam optimizer [54] with a learning rate of 10^{-3}, a batch size of 128 for Pinv-Net and LPGD and 256 for DC-Net, and 30 epochs. The learning rate is decreased by a factor of 0.5 every 10 epochs. The weight decay regularization parameter is set to $10^{\cdot -7}$. The parameters of the optimizer were optimized previously for DC-Net and Pinv-Net [23]. For the PnP methods (i.e., Pinv-PnP and DPGD-PnP), the data driven layers are trained independently as denoisers, see [34] for Pinv-PnP and [55] for DPGD-PnP. The methods adapt to different noise levels by adjustment of a hyperparameter that controls the amount of regularization/denoising (see Sec. 1.1 and 1.2 of Supplement 1).

4. Results

4.1. Simulated data

We simulate Hadamard split measurements corrupted by Poisson noise according to Eq. (11) and Eq. (12) for three different image intensities α equal to 2, 10 and 50 photons, which correspond to increasing SNR. Methods are evaluated using the entire evaluation set (50k images) of the ImageNet ILSVRC2012 database [53]. All images are cropped to $N = 128 \times 128$ pixels. The number of measurements is set to $M = 4096$ and a $\times 4$ square subsampling strategy is used (i.e., the measurements correspond to a full acquisition for 64×64 Hadamard patterns). We compare all methods in terms of root mean square error (RMSE), structural similarity index measure (SSIM), and by visual inspection. The hyperparameter of the PnP methods is chosen for each noise level to minimize RMSE and maximize SSIM across a subset of 384 images of the training set of the ImageNet ILSVRC2012 database. When RMSE and SSIM disagree, the hyperparameter was selected by visual inspection, as described in Sections 1.1 and 1.2 of Supplement 1.

Table 1 reports RMSE and SSIM for the three noise levels for all methods. For the noise level used for training ($\alpha = 10$ photons), all supervised methods provide similar results in terms of RMSE and SSIM, which are slightly better than for PnP methods. Among PnP methods, Pinv-PnP yields better metrics than DPGD-PnP. Deep learning methods reduce RMSE by up to half compared to the traditional method Pinv. Comparing the results for higher SNR ($\alpha = 50$ photons), slightly improved metrics (lower RMSE and higher SSIM) are achieved by deep learning methods with respect to $\alpha = 10$ photons. For lower SNR ($\alpha = 2$ photons), supervised methods (except for DC-Net) approximately double RMSE compared to PnP methods.

Figure 3 presents a number of ground truth and reconstructed images. For $\alpha = 10$ photons, all deep learning methods are able to remove noise quite efficiently. Supervised methods lead to sharper images than PnP methods. Pinv-Net and LPGD are more consistent across images than DC-Net, which yields sharper images in some cases and more blurred results in others. LPGD leads to sharper results than Pinv-Net for some images. Pinv-PnP provides a natural texture with

low noise but images are less sharp. DC-Net provides the best results in terms of sharpness. For other image intensities, the best results are obtained using PnP methods and DC-Net, with more detail or less noise. For $\alpha = 2$ photons (i.e., lower SNR than training), supervised methods result in noisy images equivalent to Pinv, except for DC-Net which achieves similar results to PnP methods. For $\alpha = 50$ photons (i.e., higher SNR than training) DC-Net performs well. More results can be found in Sec. 1.3 of Supplement 1, including measurements simulated from the human brain image dataset [56].

4.2. Experimental data

Experimental data is used from the SPIHIM dataset [45], an open-access collection of single-pixel acquisitions that fulfil the FAIR principles. [57]. In particular, we consider the Star Sector resolution target (Thorlabs, R1L1S2P, see top row of Fig. 4) and a tomato slice (see bottom row of Fig. 4), which has a smother texture and no symmetry. As in Section 4.1, the measurements correspond to a full acquisition with $M = 4096$ split Hadamard patterns of size 64×64, which correspond to a $\times 4$ square subsampling strategy for a reconstruction with $N = 128 \times 128$ pixels. Figure 4 shows the reconstruction of the Star Sector resolution target and tomato slice. As for the simulated data, Pinv results in noisy images while the data-driven methods efficiently remove noise. For the resolution target, Pinv-Net, LPGD and DPGD-PnP effectively remove noise but lead to artefacts that are especially visible close to the central white disk. DC-Net leads to a more natural texture but slightly blurred. Pinv-PnP yields almost perfect reconstruction. For the tomato slice, supervised methods Pinv-Net and LPGD result in the sharpest images, while the other methods yield images that are excessively blurred. These findings are corroborated by the reconstruction results of other images, which are reported in Section 2 of Supplement 1.

5. Conclusion

To the best of our knowledge, this is the first time a PyTorch package has been presented to enable reproducible research and benchmarking in the field of single-pixel imaging. The package includes a number of supervised and plug-and-play methods, including post-processing and iterative strategies.

We observe that all data-driven methods perform well provided that the image intensity is similar or higher to the training phase. However, most of the data-driven methods are not robust if the intensity is lower than the training phase, which is often the case in practice. This suggests that comparative studies should carefully address this kind of discrepancy. In this regard, the versatility of PnP methods is an advantage via manual tuning of the hyperparameter with respect to the noise level. The hyperparameter, however, has a significant impact on reconstruction quality and therefore requires careful selection. Among the supervised methods, DC-Net appears to be robust to deviations in the noise level, almost comparable to PnP methods and without hyperparameter selection. This is due to the fact that DC-Net includes a denoising step that removes noise prior to reconstruction. We also observe that the data-driven methods trained on simulations perform very well on experimental data, provided that the noise level corresponds to that of the training phase, which indicates that there is not a large model mismatch between simulations and experiments.

The modularity and versatility of SPyRiT make it suitable for further studies beyond this work, such as the comparison or learning of acquisition matrices, noise models, or reconstruction algorithms. Current developments include hyperspectral imaging and dynamic imaging [58]. While SPyRiT has been designed for single-pixel imaging, it may also benefit other modalities, in particular in the field of computational optics where the formation models are often in the form of Eq. (2). Computational ghost imaging [59] is of particular interest due to its equivalence, from an optical perspective, to single-pixel imaging.

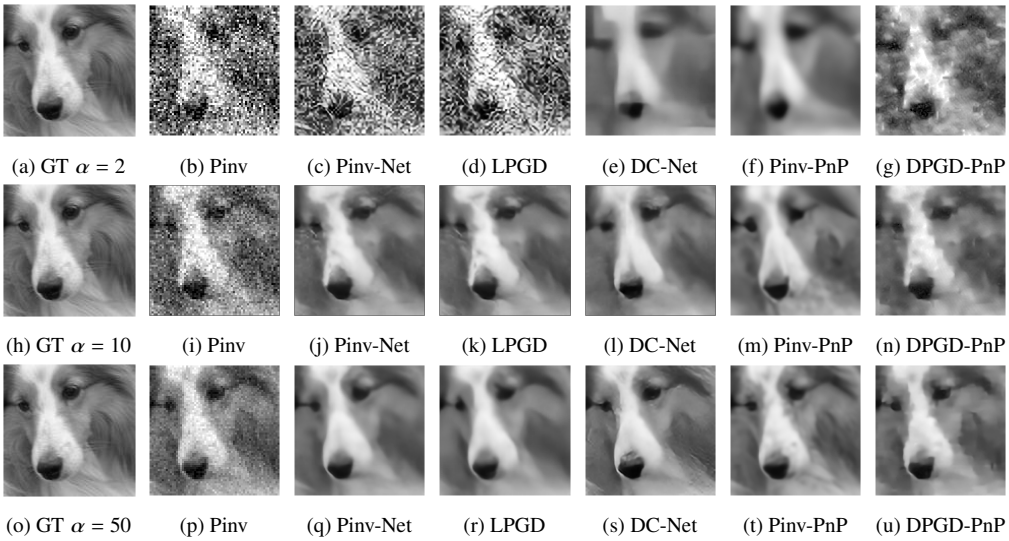

Fig. 3. Ground truth (GT) and reconstructed images obtained from measurements simulated from the ImageNet ILSVRC2012 evaluation set. All supervised methods are trained with an image intensity $\alpha = 10$ photons and tested for $\alpha = 2$ (first row), $\alpha = 10$ (second row), and $\alpha = 50$ (third row) photons. The measurements correspond to Hadamard split acquisitions with a $\times 4$ square subsampling strategy. The images are reconstructed from $M = 4096$ measurements, with the number of pixels set to $N = 128 \times 128$. The Pinv-PnP hyperparameter is set to $\nu = 115$ for $\alpha = 2$; $\nu = 45$ for $\alpha = 10$; and $\nu = 20$ for $\alpha = 50$. The DPGD-PnP hyperparameter is set to $\mu = 6000$ for $\alpha = 2$; $\mu = 3500$ for $\alpha = 10$; and $\mu = 1500$ for $\alpha = 50$.

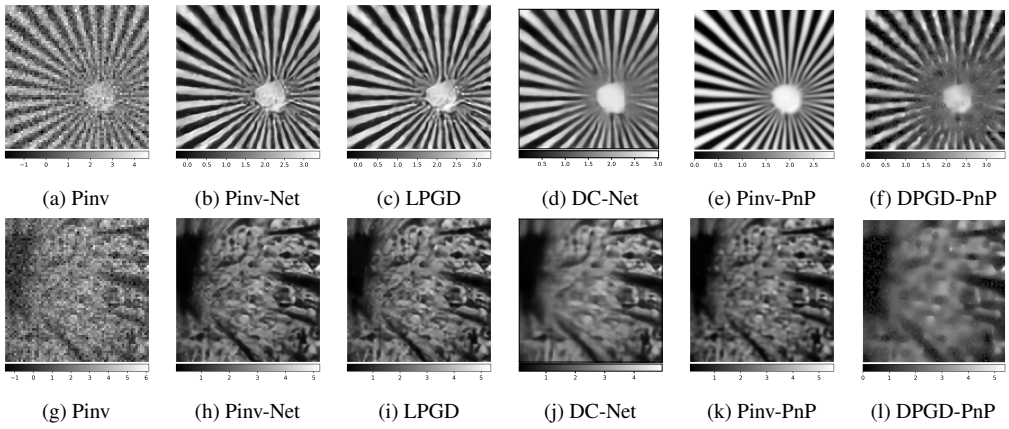

Fig. 4. Experimental reconstruction results for the Star Sector resolution target and tomato slice. The measurements correspond to Hadamard split acquisitions using a $\times 4$ square subsampling strategy. The images are reconstructed from $M = 4096$ measurements, with the number of pixels set to $N = 128 \times 128$. The Pinv-PnP hyperparameter is set to $\nu = 55$ for the resolution target; and $\nu = 35$ for the tomato slice. The DPGD-PnP hyperparameter is set to $\mu = 4000$ for the resolution target; and $\mu = 4000$ for the tomato slice.

Table 1. Reconstruction metrics obtained for measurements simulated using the evaluation set of the ImageNet ILSVRC2012 database for three image intensities α. All data-driven reconstruction methods were trained considering $\alpha = 10$ photons. Values correspond to mean (standard deviations) across images. For DPGD-PnP, standard deviations are not shown because metrics were obtained from only 384 images, given its higher computational cost. Pinv leads to highly corrupted images for some cases, resulting in NaN values for SSIM.

Methods	RMSE	SSIM
$\alpha = 2$		
Pinv	1.09 (0.64)	NaN
Pinv-Net	0.81 (0.51)	0.19 (0.14)
LPGD	0.84 (0.51)	0.18 (0.12)
DC-Net	0.41 (0.19)	0.41 (0.19)
Pinv-PnP ($\nu = 115$)	0.34 (0.16)	0.42 (0.19)
DPGD-PnP ($\mu = 6000$)	0.45 (-)	0.26 (-)
$\alpha = 10$ (as for training)		
Pinv	0.53 (0.29)	NaN
Pinv-Net	0.25 (0.14)	0.61 (0.16)
LPGD	0.26 (0.14)	0.61 (0.16)
DC-Net	0.26 (0.14)	0.61 (0.17)
Pinv-PnP ($\nu = 50$)	0.27 (0.14)	0.56 (0.18)
DPGD-PnP ($\mu = 3500$)	0.30 (-)	0.46 (-)
$\alpha = 50$		
Pinv	0.31 (0.15)	NaN
Pinv-Net	0.22 (0.14)	0.68 (0.17)
LPGD	0.23 (0.14)	0.66 (0.18)
DC-Net	0.25 (0.15)	0.71 (0.14)
Pinv-PnP ($\nu = 20$)	0.23 (0.13)	0.68 (0.15)
DPGD-PnP ($\mu = 1500$)	0.22 (-)	0.67 (-)

Appendix

A. Algorithm details

A.1. Reconstruction problem

Regularized methods consist in defining the estimate as a solution to the minimization reconstruction problem

$$\min_x \mathcal{D}(m, Hx) + g_\mu(x), \tag{14}$$

where $\mathcal{D}$ is the data fidelity term and g_μ is a regularization term that depends on the hyperparameter (e.g., regularization parameter) $\mu > 0$. We choose the weighted least-squares data fidelity term

$$\mathcal{D}(m, Hx) = \frac{1}{2}\|m - Hx\|^2_{\Gamma^{-1}}, \tag{15}$$

where $\Gamma \in \mathbb{R}^{M\times M}$ represents the noise covariance, which is assumed to be diagonal for independent measurements, i.e., $\Gamma = \mathsf{diag}\left(\sigma_1^2, \ldots, \sigma_M^2\right)$ with $\sigma_i^2 > 0$ the variance of the i-th measurement. For the Poisson-corrupted split measurements described in Section 3.2, the variance can be approximated as

$$\sigma_i^2 = \frac{1}{4}(m_i^+ + m_i^-), \tag{16}$$

where m_i^+ and m_i^- represent, respectively, the raw measurements obtained from the positive and negative component of the i-th target pattern [23].

A.2. Closed-forms: pseudo-inverse and Tikhonov regularization

Without regularization (i.e., when $g_\mu(x) = 0$) and in the absence of noise model (i.e., choosing $\Gamma = I$), the solution to the problem in Eq. (14) is given by the pseudo-inverse approach defined by Eq. (8).

When $g_\mu(x) = \|x\|^2_{\Sigma^{-1}}$, where Σ represents the image covariance, the solution to the problem in Eq. (14) admits the closed form solution given by Eq. (9). The DC-Net method is of the form of Eq. (6) with $\mathcal{R}$ given in Eq. (9). In this approach, the image covariance Σ, which has been computed on the ImageNet ILSVRC2012 database (see [45, Sec. 3.3]), and the learnable parameters of $\mathcal{G}_\theta$ have been downloaded from [60].

A.3. Proximal gradient descent

A common iterative method used to solve the minimization problem in Eq. (14) with data-fidelity term given in Eq. (15) is the proximal gradient descent algorithm (a.k.a. forward-backward splitting algorithm) [61]. For this problem, iterations are given by

$$x_{k+1} = \mathrm{prox}_{\gamma g_\mu}(x_k - \gamma H^\top \Gamma^{-1}(Hx_k - m)), \tag{17}$$

where $\mathrm{prox}_{\gamma g}$ is the proximity operator of g (see [61]) and γ is a step size. The sequence $(x_k)_{k\in\mathbb{N}}$ generated by Eq. (17) converges with rate $O(1/k)$ using a fixed step size $\gamma \in (0, \beta^{-1})$, where

$$\beta = \|\Gamma^{-1/2} H H^\top \Gamma^{-1/2}\|_2 = \|H^\top \Gamma^{-1} H\|_2, \tag{18}$$

with $\|\cdot\|_2$ the spectral norm (i.e., the largest singular value of its argument).

A.4. Step size for a Hadamard matrix

To compute the step-size γ in Eq. (17), it is necessary to compute β. In the case when H is the Hadamard matrix, we have $H^\top H = N I_N$. Hence, using Eq. (18), we have $\beta = N\|\Gamma^{-1}\|_S = N \max_i \sigma_i^{-1}$, so

$$\beta^{-1} = \frac{\min_i \sigma_i^2}{N}. \tag{19}$$

A.5. Learned proximal gradient descent

The LPGD method considers the proximal iteration of Eq. (17), where the proximity operator of g_μ is replaced by a neural network $\mathcal{G}_{\theta_k}$ [51], i.e.,

$$x_{k+1} = \mathcal{G}_{\theta_k}\left(x_k - \gamma H^\top (Hx_k - m)\right). \tag{20}$$

To limit the number of parameters, the weights of the neural network are shared across iterations, i.e., for every k, $\mathcal{G}_{\theta_k} \equiv \mathcal{G}_\theta$. Since the neural networks of LPGD are learned end-to-end, we only unroll Eq. (20) over a fixed number of iterations K.

In this context, two hyperparameters need to be tuned: the number of iterations K and the step size γ. The number of iterations is commonly chosen between 3 and 10, as a trade-off between performance and computational cost. We have tested both fixed and decreasing step sizes, i.e., $\gamma_k = \tau\gamma_{k-1}$ with a decaying multiplicative factor $0 < \tau \leq 1$. The step size has been initialized to the inverse of the Lipschitz constant given in Eq. (19), as suggested by optimization theory (see Appendix A.4). We have tested different values for the number of iterations (3, 6 and 10) and different decaying factors (0.1, 0.5, 0.9, and 1). The best results were obtained for $\tau = 0.9$, and as little difference was found in terms of number of iterations, we selected $K = 3$ (results not shown).

A.6. Dual PGD network

We consider the LPGD iteration given in Eq. (20), where $\mathcal{G}_{\theta_k}$ is trained as a denoiser independent from the measurement model and is itself an unfolded algorithm that mimics the proximity operator of g_μ. Specifically, we consider the case when $g_\mu = \iota_{[-1,1]} + \lambda\|L\cdot\|_1$ with $L\colon \mathbb{R}^N \to \mathbb{R}^S$ a linear sparsifying operator. In this case the proximity operator of g_μ can be computed using the dual PGD algorithm [61]. Within a deep learning framework, only a fixed number of iterations of the dual PGD algorithm are unrolled, and the operator L can be learned (and changed for each layer), leading to a Dual PGD network dubbed DPGD-PnP (see [55, 62]).

The DPGD-PnP network is composed of 20 layers, and the linearity associated with each layer corresponds to a convolution with 64 features. We trained the network using the ImageNet test dataset.

Since this approach is based on the LPGD iteration from Eq. (20), the step size is chosen as per Appendix A.4. The main advantage compared to the supervised approaches is that the training is independent of the measurement model. However, a large number of iterations must be considered. We choose $K_{\max} = 101$ with a stopping criterion $\|x_{k+1} - x_k\| < 10^{-4}\|x_k\|$ in case convergence is reached before $K_{\max}$ iterations.

Funding. Institut Universitaire de France; Agence Nationale de la Recherche (ANR-11-LABX-0063, ANR-22-CE19-0030-01).

Disclosures. The authors declare no conflicts of interest.

Data Availability Statement. The code and data are provided in [63].

Supplemental document. See Supplement 1 for supporting content.

References

1. M. P. Edgar, G. M. Gibson, and M. J. Padgett, "Principles and prospects for single-pixel imaging," Nat. Photonics **13**, 13–20 (2019).
2. G. M. Gibson, G. M. Gibson, S. D. Johnson, *et al.*, "Single-pixel imaging 12 years on: A review," Opt. Express **28**, 28190–28208 (2020).
3. M. Harwit and N. J. A. Sloane, *Hadamard Transform Optics* (Academic Press, 1979).
4. M. Duarte, M. Davenport, D. Takhar, *et al.*, "Single-Pixel Imaging via Compressive Sampling," Signal Process. Mag. IEEE **25**, 83–91 (2008).
5. N. Radwell, K. J. Mitchell, G. M. Gibson, *et al.*, "Single-pixel infrared and visible microscope," Optica **1**, 285–289 (2014).

6. Y. Wang, K. Huang, J. Fang, *et al.*, "Mid-infrared single-pixel imaging at the single-photon level," Nat. Commun. **14**, 1073 (2023).
7. J.-T. Ye, C. Yu, W. Li, *et al.*, "Ultraviolet photon-counting single-pixel imaging," Appl. Phys. Lett. **123**, 024005 (2023).
8. W. L. Chan, K. Charan, D. Takhar, *et al.*, "A single-pixel terahertz imaging system based on compressed sensing," Appl. Phys. Lett. **93**, 121105 (2008).
9. Y. Klein, A. Schori, I. P. Dolbnya, *et al.*, "X-ray computational ghost imaging with single-pixel detector," Opt. Express **27**, 3284–3293 (2019).
10. Y. Jauregui-Sánchez, P. Clemente, J. Lancis, and E. Tajahuerce, "Single-pixel imaging with Fourier filtering: Application to vision through scattering media," Opt. Lett. **44**, 679–682 (2019).
11. F. Rousset, N. Ducros, F. Peyrin, *et al.*, "Time-resolved multispectral imaging based on an adaptive single-pixel camera," Opt. Express **26**, 10550–10558 (2018).
12. M.-J. Sun and J.-M. Zhang, "Single-Pixel Imaging and Its Application in Three-Dimensional Reconstruction: A Brief Review," Sensors **19**, 732 (2019).
13. N. Radwell, S. D. Johnson, M. P. Edgar, *et al.*, "Deep learning optimized single-pixel LiDAR," Appl. Phys. Lett. **115**, 231101 (2019).
14. M. Yao, J. Cheng, Z. Huang, *et al.*, "Reflection light-field microscope with a digitally tunable aperture by single-pixel imaging," Opt. Express **27**, 33040–33050 (2019).
15. S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, "Solving inverse problems using data-driven models," Acta Numer. **28**, 1–174 (2019).
16. G. Barbastathis, A. Ozcan, and G. Situ, "On the use of deep learning for computational imaging," Optica **6**, 921–943 (2019).
17. C. F. Higham, R. Murray-Smith, M. J. Padgett, and M. P. Edgar, "Deep learning for real-time single-pixel video," Sci. Reports **8**, 2369 (2018).
18. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention,* (Springer, 2015), pp. 234–241.
19. S. Yang, H. Qin, X. Yan, *et al.*, "Deep spatial-spectral prior with an adaptive dual attention network for single-pixel hyperspectral reconstruction," Opt. Express **30**, 29621–29638 (2022).
20. R. Shang, K. Hoffer-Hawlik, F. Wang, *et al.*, "Two-step training deep learning framework for computational imaging without physics priors," Opt. Express **29**, 15239–15254 (2021).
21. S. Rizvi, J. Cao, K. Zhang, and Q. Hao, "Improving Imaging Quality of Real-time Fourier Single-pixel Imaging via Deep Learning," Sensors **19**, 4190 (2019).
22. N. Ducros, A. Lorente Mur, and F. Peyrin, "A completion network for reconstruction from compressed acquisition," in *IEEE 17th International Symposium on Biomedical Imaging (ISBI),* (IEEE, Iowa City, United States, 2020), pp. 619–623.
23. A. Lorente Mur, P. Leclerc, F. Peyrin, and N. Ducros, "Single-pixel image reconstruction from experimental data using neural networks," Opt. Express **29**, 17097–17110 (2021).
24. V. Monga, Y. Li, and Y. C. Eldar, "Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing," IEEE Signal Process. Mag. **38**, 18–44 (2021).
25. J. Zhang, B. Chen, R. Xiong, and Y. Zhang, "Physics-Inspired Compressive Sensing: Beyond deep unrolling," IEEE Signal Process. Mag. **40**, 58–72 (2023).
26. A. Lorente Mur, F. Peyrin, and N. Ducros, "Deep Expectation-Maximization for Single-Pixel Image Reconstruction With Signal-Dependent Noise," IEEE Trans. on Comput. Imaging **8**, 759–769 (2022).
27. J. Sauder, M. Genzel, and P. Jung, "Learning Structured Sparse Matrices for Signal Recovery via Unrolled Optimization," in *NeurIPS 2021 Workshop on Deep Learning and Inverse Problems,* (2021).
28. J. Li, Y. Li, J. Li, *et al.*, "Single-pixel compressive optical image hiding based on conditional generative adversarial network," Opt. Express **28**, 22992–23002 (2020).
29. P. Jiang, J. Liu, L. Wu, *et al.*, "Fourier single pixel imaging reconstruction method based on the U-net and attention mechanism at a low sampling rate," Opt. Express **30**, 18638–18654 (2022).
30. F. Wang, C. Wang, C. Deng, *et al.*, "Single-pixel imaging using physics enhanced deep learning," Photonics Res. **10**, 104 (2022).
31. G. Qu, P. Wang, and X. Yuan, "Dual-Scale Transformer for Large-Scale Single-Pixel Imaging," in *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR),* (2024).
32. Y. Tian, Y. Fu, and J. Zhang, "Plug-and-play algorithms for single-pixel imaging," Opt. Lasers Eng. **154**, 106970 (2022).
33. Y. Mäkinen, L. Azzari, and A. Foi, "Collaborative Filtering of Correlated Noise: Exact Transform-Domain Variance for Improved Shrinkage and Patch Matching," IEEE Trans. on Image Process. **29**, 8339–8354 (2020).
34. K. Zhang, Y. Li, W. Zuo, *et al.*, "Plug-and-Play Image Restoration With Deep Denoiser Prior," IEEE Trans. on Pattern Anal. Mach. Intell. **44**, 6360–6376 (2022).
35. A. Dave, A. K. Vadathya, R. Subramanyam, *et al.*, "Solving Inverse Computational Imaging Problems Using Deep Pixel-Level Prior," IEEE Trans. on Comput. Imaging **5**, 37–51 (2019).
36. F. Wang, C. Wang, M. Chen, *et al.*, "Far-field super-resolution ghost imaging with a deep neural network constraint,"

Light. Sci. & Appl. **11**, 1 (2022).
37. J. Li, B. Wu, T. Liu, and Q. Zhang, "URNet: High-quality single-pixel imaging with untrained reconstruction network," Opt. Lasers Eng. **166**, 107580 (2023).
38. J. Shenouda and W. U. Bajwa, "A Guide to Computational Reproducibility in Signal Processing and Machine Learning [Tips & Tricks]," IEEE Signal Process. Mag. **40**, 141–151 (2023).
39. M. Simeoni, S. Kashani, J. Rué-Queralt, and P. Developers, "Pyxu-org/pyxu: Pyxu," Zenodo.
40. J. Tachella, "Deepinv/deepinv: PyTorch library for solving imaging inverse problems using deep learning," .
41. S. Rit, M. V. Oliva, S. Brousmiche, *et al.*, "The Reconstruction Toolkit (RTK), an open-source cone-beam CT reconstruction toolkit based on the Insight Toolkit (ITK)," J. Physics: Conf. Ser. **489**, 012079 (2014).
42. W. van Aarle, W. J. Palenstijn, J. Cant, *et al.*, "Fast and flexible X-ray tomography using the ASTRA toolbox," Opt. Express **24**, 25129–25147 (2016).
43. A. Biguri, M. Dosanjh, S. Hancock, and M. Soleimani, "TIGRE: A MATLAB-GPU toolbox for CBCT image reconstruction," Biomed. Phys. & Eng. Express **2**, 055010 (2016).
44. "BART Toolbox for Computational Magnetic Resonance Imaging," .
45. G. Beneti Martins, L. Mahieu-Williame, T. Baudier, and N. Ducros, "OpenSpyrit: An ecosystem for open single-pixel hyperspectral imaging," Opt. Express **31**, 15599 (2023).
46. Z. Zhang, X. Ma, and J. Zhong, "Single-pixel imaging by means of Fourier spectrum acquisition," Nat. communications **6** (2015).
47. Z. Zhang, X. Wang, G. Zheng, and J. Zhong, "Hadamard single-pixel imaging versus Fourier single-pixel imaging," Opt. Express **25**, 19619–19639 (2017).
48. A. Lorente Mur, M. Ochoa, J. E. Cohen, *et al.*, "Handling negative patterns for fast single-pixel lifetime imaging," in *Proc. SPIE 10862,* vol. Molecular-Guided Surgery: Molecules, Devices, and Applications V (2019).
49. K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep Convolutional Neural Network for Inverse Problems in Imaging," IEEE Trans. on Image Process. **26**, 4509–4522 (2017).
50. A. L. Mur, B. Montcel, F. Peyrin, and N. Ducros, "Deep neural networks for single-pixel compressive video reconstruction," in *Unconventional Optical Imaging II,* vol. 11351 (International Society for Optics and Photonics, 2020), p. 113510S.
51. M. Mardani, Q. Sun, S. Vasawanala, *et al.*, "Neural proximal gradient descent for compressive imaging," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems,* (Curran Associates Inc., Red Hook, NY, USA, 2018), NIPS'18, pp. 9596–9606.
52. "https://spyrit.readthedocs.io/en/master/gallery/index.html," .
53. O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," Int. J. Comput. Vis. **115**, 211–252 (2015).
54. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings,* Y. Bengio and Y. LeCun, eds. (2015).
55. H. T. V. Le, A. Repetti, and N. Pustelnik, "PNN: From proximal algorithms to robust unfolded image denoising networks and Plug-and-Play methods," (2023).
56. H. Fabelo, M. Halicek, S. Ortega, *et al.*, "Deep Learning-Based Framework for In Vivo Identification of Glioblastoma Tumor using Hyperspectral Images of Human Brain," Sensors **19**, 920 (2019).
57. M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, *et al.*, "The FAIR Guiding Principles for scientific data management and stewardship," Sci. Data **3**, 160018 (2016).
58. T. Maitre, E. Bretin, L. Mahieu-Williame, *et al.*, "Hybrid single-pixel camera for dynamic hyperspectral imaging," (2023).
59. J. H. Shapiro, "Computational ghost imaging," Phys. Rev. A **78**, 061802 (2008).
60. "https://github.com/openspyrit/spyrit-examples/tree/master/2022_OE_spyrit2," .
61. P. L. Combettes and J.-C. Pesquet, "Proximal Splitting Methods in Signal Processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering,* H. H. Bauschke, R. S. Burachik, P. L. Combettes, *et al.*, eds. (Springer, New York, NY, 2011), pp. 185–212.
62. A. Repetti, M. Terris, Y. Wiaux, and J.-C. Pesquet, "Dual forward-backward unfolded network for flexible plug-and-play," in *2022 30th European Signal Processing Conference (EUSIPCO),* (IEEE, 2022), pp. 957–961.
63. "https://github.com/openspyrit/spyrit-examples/tree/master/dev/2024_Optica," .