SPyRiT: A Python Toolbox for Deep Single-Pixel Image Reconstruction — supplemental document

1. SIMULATION RESULTS

All the simulation results presented in this work consider the square subsampling strategy with a $\times 4$ reduction for 128×128 DMD patterns (i.e. $M = 4096$, $N = 16384$). The selection of the hyperparameters is based on the computation of the MSE and SSIM of reconstructions of measurements simulated from 384 images (to reduce computational time) from the training set of the ImageNet ILSVRC2012 database, as well as visual inspection of five images from the validation set of the ImageNet ILSVRC2012 database, and one image from the human brain dataset [1].

A. Choice of parameters for PnP methods

A.1. Pinv-PnP: selection of the noise level

We use a DR-UNet which takes as input a noisy image as well as a noise level map. We choose the noise level map as a constant image with value $\nu > 0$ [2]. In Table S1, we report the RMSE and SSIM of the images reconstructed by Pinv-PnP for various noise levels. We consider three different image intensities, which correspond to different signal-to-noise ratios of the measurements. We display some reconstructed images in Fig. S1.

Best results in terms of both RMSE and SSIM are obtained using $\nu = 45$ for $\alpha = 10$ photons and $\nu = 20$ for $\alpha = 50$ photons. For $\alpha = 2$ photons (the lowest SNR), best results are obtained for $\nu = 100$ (RMSE) and $\nu = 110$ (SSIM). We selected $\nu = 100$ as images are excessively blurred for $\nu = 110$. As expected the lower the image intensity, the higher the noise level of Pinv-PnP.

A.2. DPGD-PnP: selection of the regularization parameter

The DPGD-PnP method requires tuning the regularization parameter μ (see Eq. (14) of the main document). In Table S2, we report the RMSE and SSIM of the images reconstructed by DPGD-PnP for various regularization parameters. We consider three different image intensities, which correspond to different signal-to-noise ratios of the measurements. We display some reconstructed images in Fig. S2.

The choice of the best regularization parameter depends on the metric. Optimal values of RMSE require higher μ values (more regularization) than optimal SSIM. Hence, there is trade-off between noise reduction and detail preservation. Optimal ranges for μ are 2500–3000 for $\alpha = 10$ photons, 1200–1500 for $\alpha = 50$ photons, and 4500–6000 for $\alpha = 2$ photons. For $\alpha = 10$ photons, we chose $\mu = 3000$ by visual assessment. Most details are preserved, although it presents some cartoon-like artifacts; larger values of μ could be considered for a more natural texture. For $\alpha = 50$ photons, we retain $\mu = 1500$ that presents a good trade off between noise reduction and detail preservation. For $\alpha = 2$ photons, all images are very noisy, we select $\mu = 6000$.

B. Influence of the image intensity

We compare all the methods introduced in Sec. 3.3 of the main document for three different image intensities. All supervised methods (Pinv-Net, DC-Net, LPGD) were trained from simulations considering an image intensity of 10 photons, while the hyperparameter of Pinv-PnP and DPGD-PnP was selected according to the procedure described in Section A.1 and Section A.2, respectively. For consistency, we use the same U-Net for Pinv-Net, DC-Net and LPGD. The original U-Net architecture has been described in [3], we use the modified architecture described in [4].

We display the reconstruction of five images from the validation set of the ImageNet ILSVRC2012 database, and one image from the human brain dataset [1] in Fig. S3 ($\alpha = 10$ photons, same intensity as training), Fig. S4 ($\alpha = 50$ photons, higher SNR than training), and Fig. S5 ($\alpha = 2$ photons, lower SNR than training).

For $\alpha = 10$ photons, supervised methods (Pinv-Net, DC-Net, LPGD) provide similar results while PnP methods (Pinv-PnP, DPGD-PnP) are more prone to over-smoothing or artefacts.

(a) GT $\alpha = 2$ **(b)** $\nu = 80$ **(c)** $\nu = 90$ **(d)** $\nu = 100$ **(e)** $\nu = 110$ **(f)** $\nu = 115$

(g) GT $\alpha = 2$ **(h)** $\nu = 80$ **(i)** $\nu = 90$ **(j)** $\nu = 100$ **(k)** $\nu = 110$ **(l)** $\nu = 115$

(m) GT $\alpha = 10$ **(n)** $\nu = 20$ **(o)** $\nu = 25$ **(p)** $\nu = 30$ **(q)** $\nu = 40$ **(r)** $\nu = 50$

(s) GT $\alpha = 10$ **(t)** $\nu = 20$ **(u)** $\nu = 25$ **(v)** $\nu = 30$ **(w)** $\nu = 40$ **(x)** $\nu = 50$

(y) GT $\alpha = 50$ **(z)** $\nu = 10$ **(aa)** $\nu = 15$ **(ab)** $\nu = 20$ **(ac)** $\nu = 25$ **(ad)** $\nu = 30$

(ae) GT $\alpha = 50$ **(af)** $\nu = 10$ **(ag)** $\nu = 15$ **(ah)** $\nu = 20$ **(ai)** $\nu = 25$ **(aj)** $\nu = 30$

Fig. S1. Images reconstructed by Pinv-PnP for different noise levels ν and image intensities α. From top to bottom, we consider $\alpha = 2$, 10, and 50 photons. The simulations consider the square subsampling strategy with $\times 4$ reduction for 128×128 images.

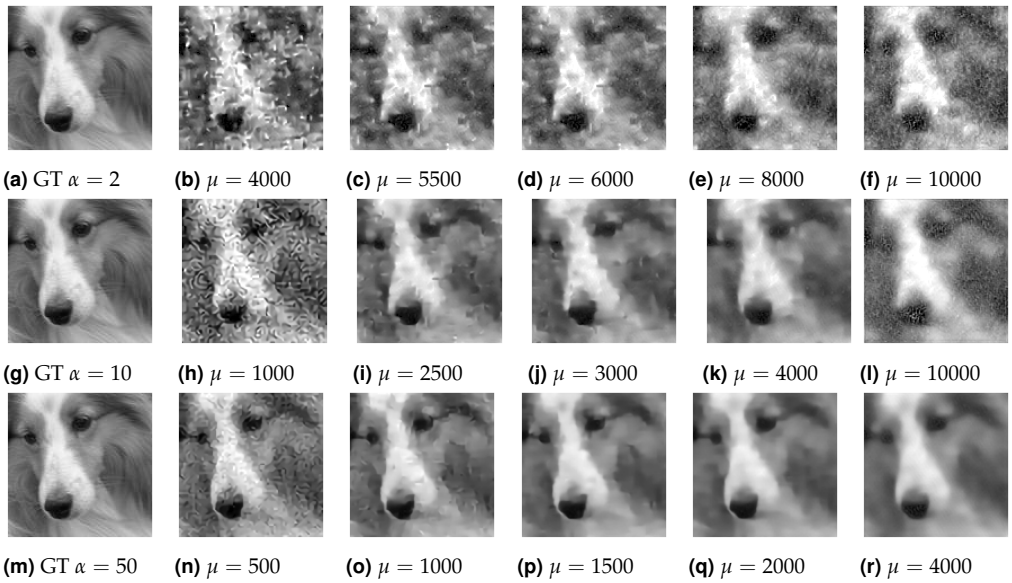


Fig. S2. Images reconstructed by DPGD-PnP for different regularization parameters μ and image intensities α. From top to bottom, we consider $\alpha = 2$, 10, and 50 photons. The simulations consider the square subsampling strategy with $\times 4$ reduction for 128×128 images.

Among supervised methods, Pinv-Net and LPGD behave similarly irrespective of the image; DC-Net provides sharper images in some cases and more blurred in others. LPGD leads to similar or slightly sharper results than Pinv-Net. As the image intensity is similar to training, similar image quality was expected from the supervised methods.

For $\alpha = 50$ photons (i.e., a higher SNR than training), supervised methods generate blurred images, except DC-Net that reconstructs sharp images. Plug-and-play methods lead to sharper images than most supervised methods, especially for the brain image. For $\alpha = 2$ photons (i.e., a lower SNR than training), supervised methods yield excessively noisy images, except for DC-Net. Here again, the best results are achieved by DC-Net and by the PnP methods whose hyperparameter can be adapted to the SNR of the measurements.

2. EXPERIMENTAL RESULTS

We compare all reconstruction methods for four experimental acquisitions (see Fig. S6). We investigate the choice of the hyperparameter of Pinv-PnP and DPGD-PnP in Fig. S7 and Fig. S8, respectively. We have chosen a wide range of hyperparameters to show their impact on the two methods. Note that PnP methods require selecting the hyperparameter for each experimental image carefully.

All methods provide similar results for the USAF target, which contains a high spatial frequency content in the horizontal and vertical directions. For the star sector target, whose spatial frequency content is distributed in all direction, DC-Net and Pinv-PnP provide the best results. For the tomato slices, which are smoother that the resolution target and exhibit a natural texture, Pinv-Net and LPGD provide the sharpest results, with low noise and many details, while DC-Net leads to smoother results and DPGD-PnP to blurred images.

3. EXAMPLE CODE

We provide below the code that generates the images in Figure 2 of the main document. It has been run using SPyRiT version 2.3.3. We split the code in three blocks: (i) import of required libraries/modules and definition of general parameters, (ii) main code for image generation and plot, (iii) auxiliary functions. It is available via the gitHub repository [**?**].

This first code block executes all the library imports necessary for the code to run, and defines the parameters used in the script.

```
import numpy as np
```

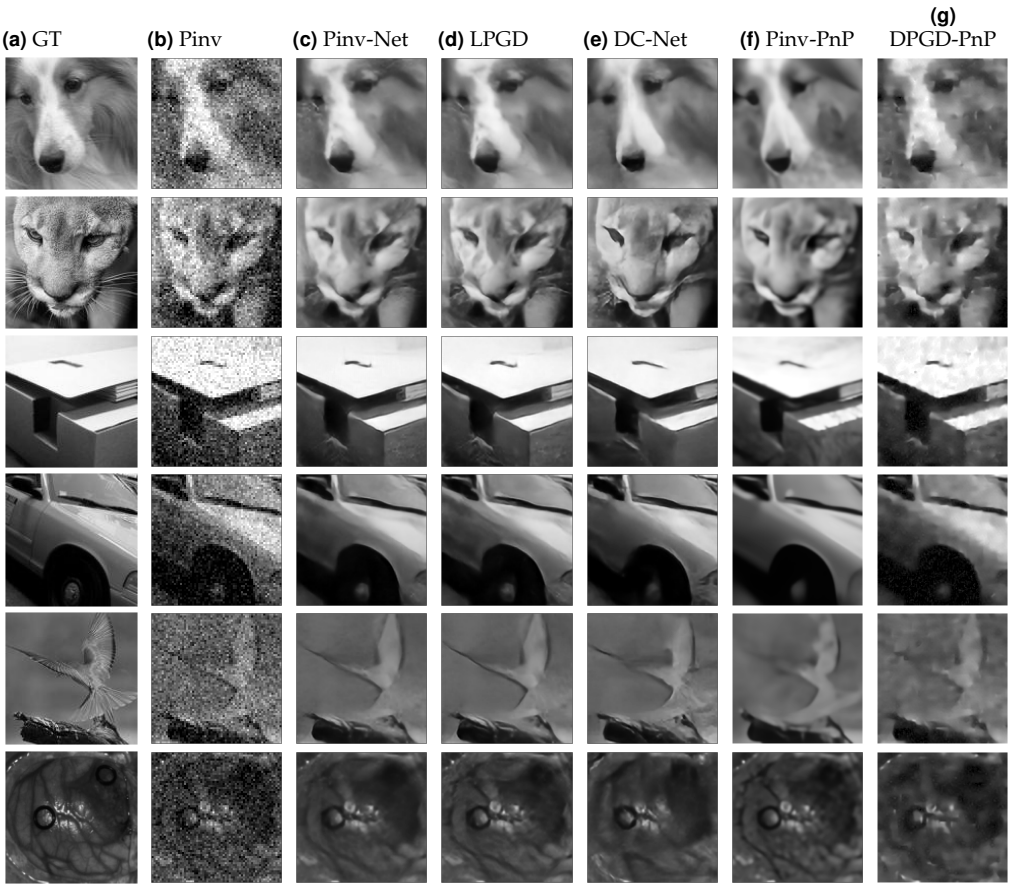


Fig. S3. Reconstructions from simulated measurements corresponding to an image intensity of $\alpha = 10$ photons. The top five rows correspond to five images from the validation set of ImageNet ILSVRC2012 database; the bottom row to an image from the human brain dataset [1]. The hyperparameter for Pinv-PnP is set to $\nu = 45$ for all rows except for the last where $\nu = 25$. The hyperparameter for DPGD-PnP is set to $\mu = 3000$ for all rows except the last where it is set to $\mu = 1800$.

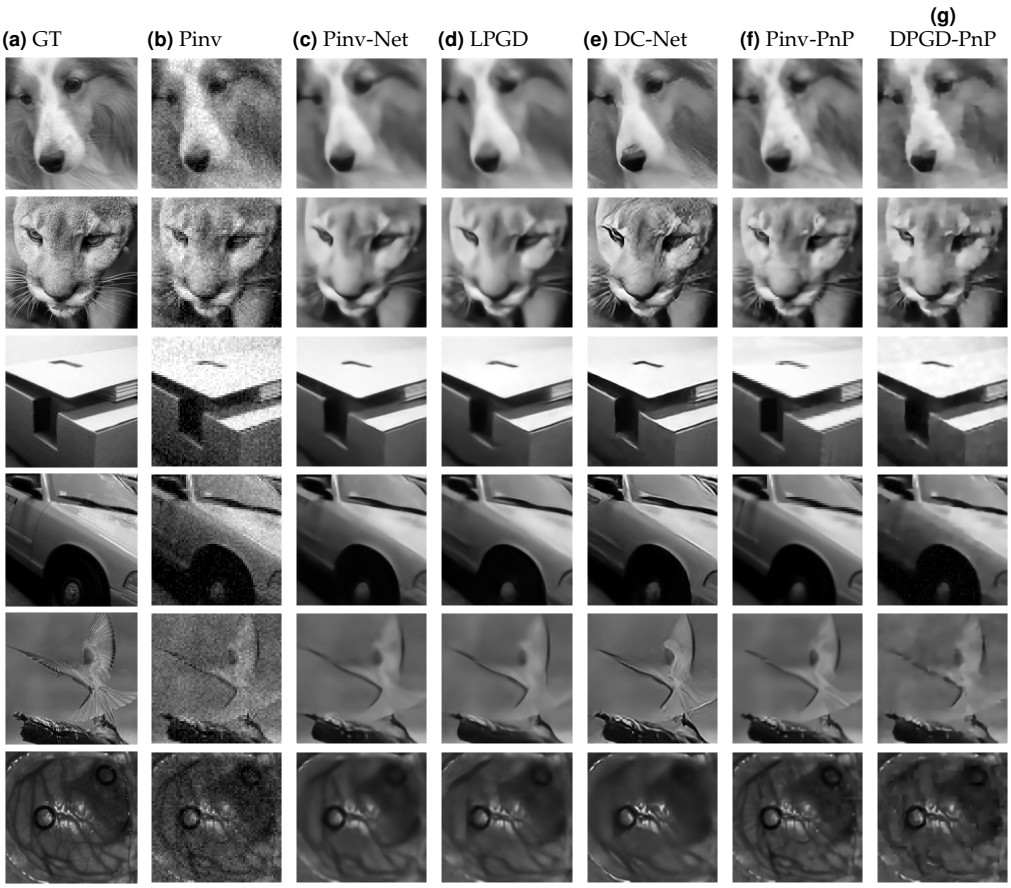


Fig. S4. Reconstructions from simulated measurements corresponding to an image intensity of $\alpha = 50$ photons. The top five rows correspond to five images from the validation set of ImageNet ILSVRC2012 database; the bottom row to an image from the human brain dataset [1]. The hyperparameter for Pinv-PnP is set to $\nu = 20$ for all rows except for the last where $\nu = 10$. The hyperparameter for DPGD-PnP is set to $\mu = 1500$ for all rows except the last where it is set to $\mu = 700$.

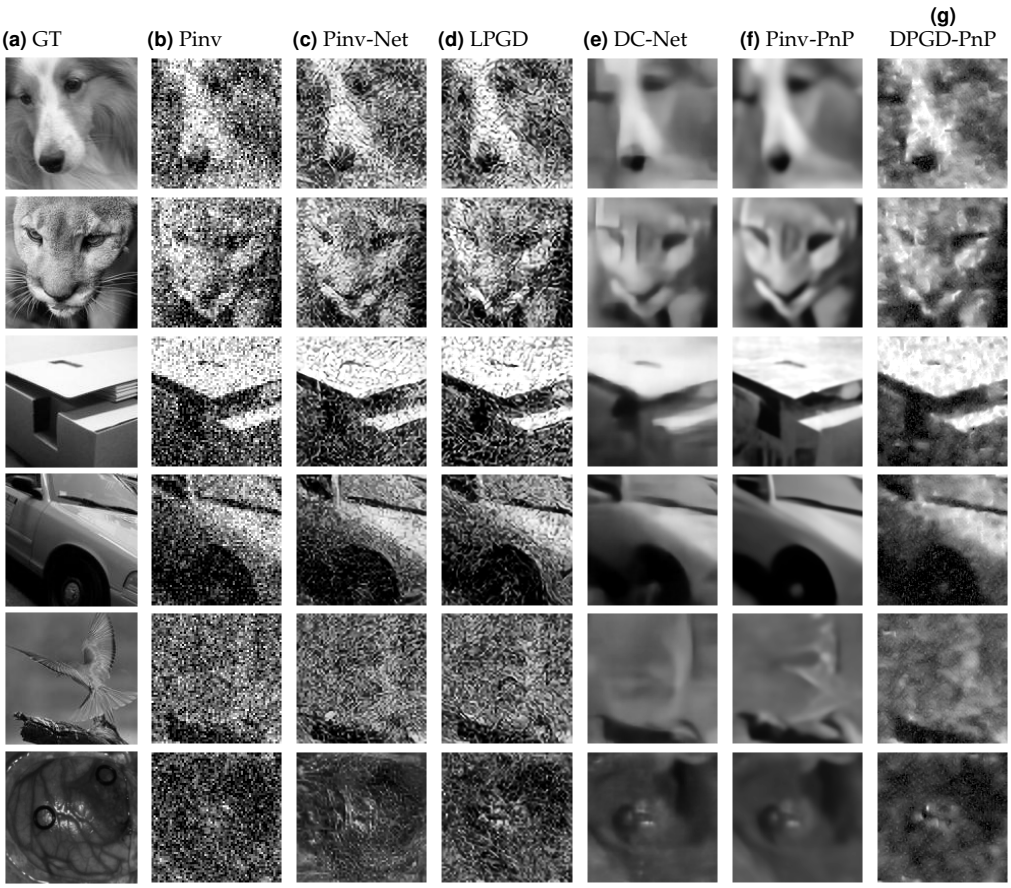


Fig. S5. Reconstructions from simulated measurements corresponding to an image intensity of $\alpha = 2$ photons. The top five rows correspond to five images from the validation set of ImageNet ILSVRC2012 database; the bottom row to an image from the human brain dataset [1]. The hyperparameter for Pinv-PnP is set to $\nu = 115$ for the first two rows, $\nu = 100$ for the next three rows, and $\nu = 70$ for the last row. The hyperparameter for DPGD-PnP is set to $\mu = 6000$ for all rows except the last where it is set to $\mu = 4000$.

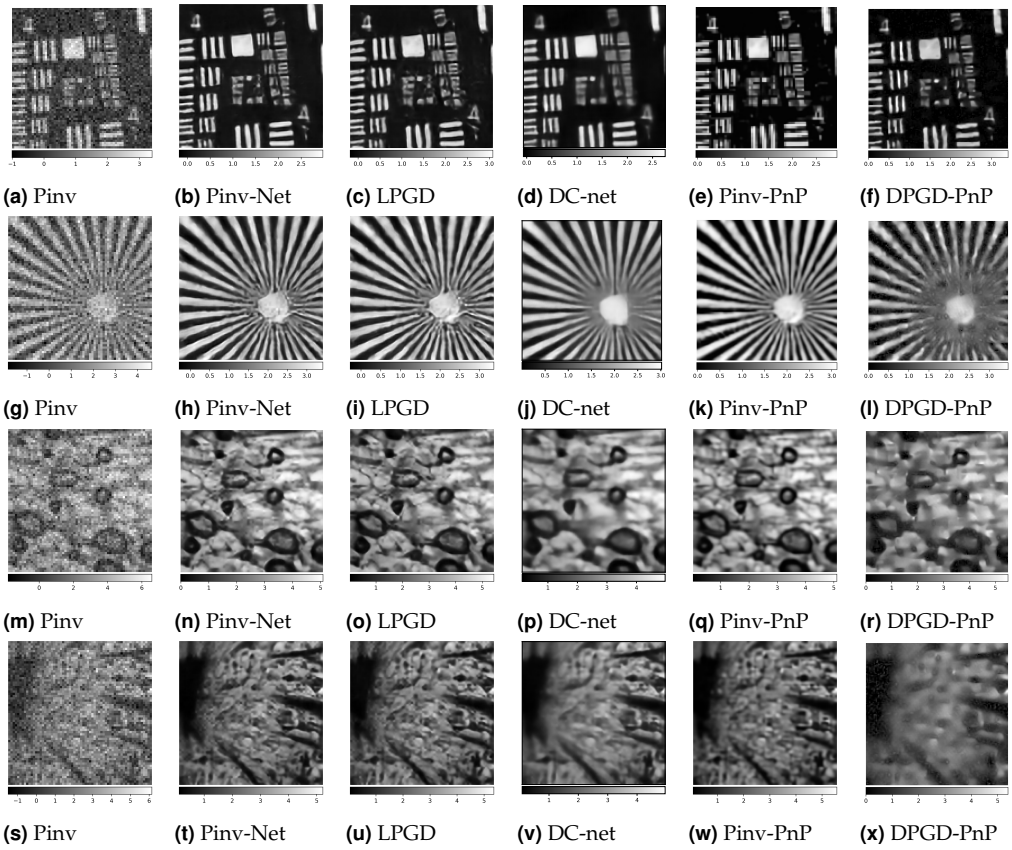


Fig. S6. Reconstruction of four experimental datasets. The acquisitions correspond to the square subsampling strategy with a $\times 4$ reduction considering 128×128 DMD patterns. **(a-f)** USAF target; Pinv-PnP with $\nu = 30$, DPGD-PnP with $\mu = 2000$. **(g-l)** star sector; Pinv-PnP with $\nu = 45$, DPGD-PnP with $\mu = 4000$. **(m-r)** tomato slice $\times 12$ zoom; Pinv-PnP with $\nu = 40$, DPGD-PnP with $\mu = 3000$. **(s-x)** tomato slice $\times 2$ zoom; Pinv-PnP with $\nu = 35$, DPGD-PnP with $\mu = 4000$.

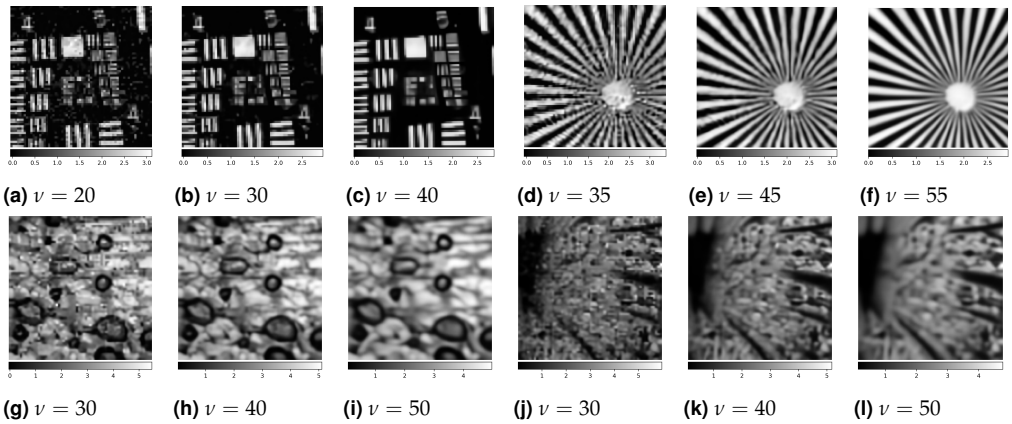
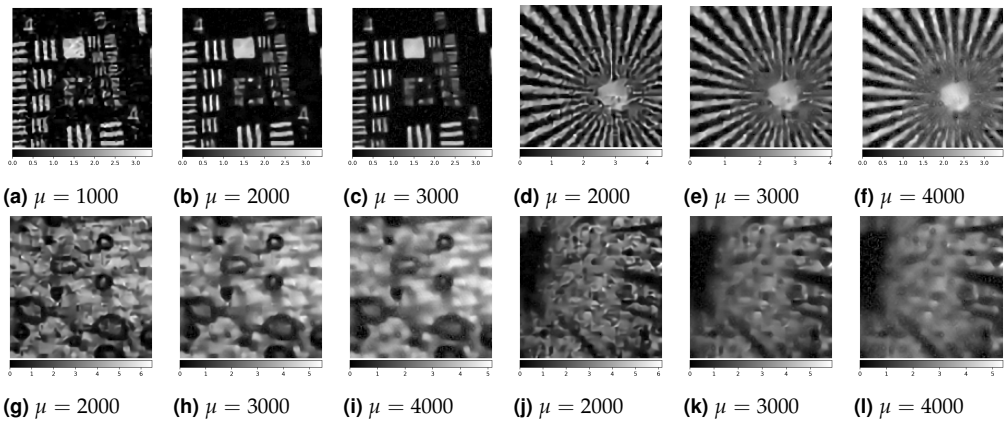


Fig. S7. Influence of the noise level for the reconstruction of experimental acquisitions using Pinv-PnP. We consider the same measurements as in Fig. S6. **(a-c)** USAF target; **(d-f)** star sector; **(g-i)** tomato slice $\times 12$ zoom; **(j-l)** tomato slice $\times 2$ zoom.



(a) $\mu = 1000$ **(b)** $\mu = 2000$ **(c)** $\mu = 3000$ **(d)** $\mu = 2000$ **(e)** $\mu = 3000$ **(f)** $\mu = 4000$

(g) $\mu = 2000$ **(h)** $\mu = 3000$ **(i)** $\mu = 4000$ **(j)** $\mu = 2000$ **(k)** $\mu = 3000$ **(l)** $\mu = 4000$

Fig. S8. Influence of the regularization parameter for the reconstruction of experimental acquisitions using DPGD-PnP. We consider the same measurements as in Fig. S6. **(a-c)** USAF target; **(d-f)** star sector; **(g-i)** tomato slice $\times 12$ zoom; **(j-l)** tomato slice $\times 2$ zoom.

```python
import torch
import torchvision
import matplotlib.ticker as ticker
import matplotlib.pyplot as plt

import spyrit.core.meas as meas
import spyrit.core.noise as noise
import spyrit.core.prep as prep
import spyrit.core.torch as spytorch
import spyrit.misc.statistics as stats
import spyrit.misc.sampling as samp
from spyrit.misc.load_data import download_girder

# Generic parameters for the script
# ---------------------------------

# Image size
h = 64
# matrix size for display
h_disp = 16
# number of measurements
M = h**2
# subsampling factor, when used
sub = 4
# noise parameter (poisson distribution)
alpha = 100

# figure size (in inches)
figsize = (5, 5)
figsize_dbl = (5, 10)
# dots per inch (resolution)
dpi = 300
# measurement domain colormap
cmap_meas = plt.cm.cividis

# set the seed for reproducibility
seed = 404
```

This code block simulates the noisy measurements and reconstructs them. The visualization of the results is based on the three custom functions provided in a separate block. We choose a symmetric log scale to improve the visualization of the measurements.

```
# Get image
# ---------

# download image from girder server
url = "https://tomoradio-warehouse.creatis." +
    "insa-lyon.fr/api/v1"
dataID = "668e986a7d138728d4806d7a"
local_folder = "./image_sample/"
class_folder = "test/"
data_name = "ILSVRC2012_test_00000002.jpeg"
image_abs_path = download_girder(url, dataID,
    local_folder+class_folder, data_name)

# Transform to greyscale, resize to h
transform = stats.transform_gray_norm(img_size=h)
# Create dataset and loader
dataset = torchvision.datasets.ImageFolder(
    root=local_folder, transform=transform)
dataloader = torch.utils.data.DataLoader(
    dataset, batch_size=7)

x, _ = next(iter(dataloader))
x = x[0, :, :, :]
print(f"Shape of input image: {x.shape}")

# Select image
x = x.detach().clone()
c, height, width = x.shape

# Plot Ground-truth image
x_plot = x.view(h, h).cpu().numpy()
orig_minmax = (-1, 1)
imagesc_mod(x_plot, figsize=figsize, dpi=dpi,
    minmax=orig_minmax)

# SQUARE SAMPLING MAP
# -------------------
Sampling_map = np.ones((h, h))
Sampling_map[:, h//sub_x:] = 0
Sampling_map[h//sub_y:, :] = 0
Sampling_map = torch.from_numpy(Sampling_map)


# MEASUREMENT OPERATORS
# ---------------------
H = spytorch.walsh2_matrix(h)
meas_op1 = meas.Linear(H)
meas_op2 = meas.HadamSplit(M, h)
meas_op3 = meas.HadamSplit(M//sub, h, Ord=Sampling_map)

# plot the mask matrices
mask1 = (meas_op1.indices.argsort() <
    meas_op1.M).reshape(h, h)
mask2 = (meas_op2.indices.argsort() <
    meas_op2.M).reshape(h, h)
mask3 = (meas_op3.indices.argsort() <
    meas_op3.M).reshape(h, h)

# op2 & op3 are split, we double the masks
mask2 = torch.cat((mask2, mask2), dim=0)
mask3 = torch.cat((mask3, mask3), dim=0)

imagesc_mod(mask1.cpu().numpy(),
    figsize=figsize, dpi=dpi, minmax=(-1, 1))
imagesc_mod(mask2.cpu().numpy(),
    figsize=figsize_dbl, dpi=dpi, minmax=(-1, 1))
imagesc_mod(mask3.cpu().numpy(),
    figsize=figsize_dbl, dpi=dpi, minmax=(-1, 1))


# MATRICES H
# ----------

# show 1D walsh-ordered hadamard matrices
H_disp1 = spytorch.walsh_matrix(h_disp)
H_disp2 = torch.cat((H_disp1, -H_disp1), dim=0)
```

```
H_disp2[H_disp2 == -1] = 0 # take positive part

order = [0, 1, 7, 6] # subsampling custom order
zer = torch.zeros((h_disp-len(order), h_disp))
H_disp3 = torch.cat(
    (H_disp1[order], zer, -H_disp1[order], zer),
    dim=0)
H_disp3[H_disp3 == -1] = 0 # take positive part

imagesc_mod(H_disp1, figsize=figsize,
    dpi=dpi, minmax=(-1, 1))
imagesc_mod(H_disp2, figsize=figsize_dbl,
    dpi=dpi, minmax=(-1, 1))
imagesc_mod(H_disp3, figsize=figsize_dbl,
    dpi=dpi, minmax=(-1, 1))


# MEASUREMENTS
# ------------

# set seed for reproducibility in Poisson noise
torch.manual_seed(seed)

# corrupt using poisson noise
noise_op1 = noise.NoNoise(meas_op1)
noise_op2 = noise.Poisson(meas_op2, alpha)
noise_op3 = noise.Poisson(meas_op3, alpha)

# vectorize image for measurements
x = x.view(-1, h*h)
# measure
y1 = noise_op1(x)
y2 = noise_op2(x)
y3 = noise_op3(x)

# plots
y1_plot = y1.reshape(h, h)
y2_plot = split_meas2img(y2, meas_op2)
y3_plot = split_meas2img(y3, meas_op3)

# for split ops, center measurements
y2_plot = center_measurements(y2_plot)
y3_plot = center_measurements(y3_plot)

norm = 'symlog'
imagesc_mod(y1_plot, figsize=figsize, dpi=dpi,
    norm=norm, colormap=cmap_meas)
imagesc_mod(y2_plot, figsize=figsize_dbl, dpi=dpi,
    norm=norm, colormap=cmap_meas)
imagesc_mod(y3_plot, figsize=figsize_dbl, dpi=dpi,
    norm=norm, colormap=cmap_meas)


# PREPROCESSING
# -------------

# preprocess measurements
prep_op1 = prep.DirectPoisson(1, meas_op1)
prep_op2 = prep.SplitPoisson(alpha, meas_op2)
prep_op3 = prep.SplitPoisson(alpha, meas_op3)

# Generate measurements using the prep operators
m1 = prep_op1(y1)
m2 = prep_op2(y2)
m3 = prep_op3(y3)

# reorder measurements to mathc shown mask
m2_plot = torch.from_numpy(
    samp.meas2img(m2, meas_op2.Ord.numpy()))
m2_plot[m2_plot == 0] = torch.tensor(float('nan'))

m3_plot = torch.from_numpy(
    samp.meas2img(m3, meas_op3.Ord.numpy()))
m3_plot[m3_plot == 0] = torch.tensor(float('nan'))

# plot the preprocessed measurements
imagesc_mod(
    m1.view(h, h).cpu(), figsize=figsize,
    dpi=dpi, norm=norm, colormap=cmap_meas)
imagesc_mod(
    m2_plot.reshape(h, h), figsize=figsize,
```

```
    dpi=dpi, norm=norm, colormap=cmap_meas)
imagesc_mod(
    m3_plot.reshape(h, h), figsize=figsize,
    dpi=dpi, norm=norm, colormap=cmap_meas)


# RECONSTRUCTION
# --------------

z1 = meas_op1.pinv(m1)
z2 = meas_op2.pinv(m2)
z3 = meas_op3.pinv(m3)

imagesc_mod(z1.view(h, h).cpu().numpy(),
    figsize=figsize, dpi=dpi, minmax=orig_minmax)
imagesc_mod(z2.view(h, h).cpu().numpy(),
    figsize=figsize, dpi=dpi, minmax=orig_minmax)
imagesc_mod(z3.view(h, h).cpu().numpy(),
    figsize=figsize, dpi=dpi, minmax=orig_minmax)
```

This last code block contains the three auxiliary functions necessary to display the results computed in the previous code block.

```
def imagesc_mod(img,
                title='',
                figsize=(5, 5),
                colormap=plt.cm.gray,
                title_fontsize=16,
                dpi=100,
                minmax=None,
                showscale=False,
                **kwargs):
    fig = plt.figure(figsize=figsize, dpi=dpi)
    ax = fig.add_subplot(1, 1, 1)
    # clean the axes
    ax.xaxis.set_major_locator(
        ticker.NullLocator()
    )
    ax.yaxis.set_major_locator(
        ticker.NullLocator()
    )
    # set min max for the colormap
    if minmax is None:
        minmax = (
            img[~img.isnan()].min(),
            img[~img.isnan()].max()
        )
    # define the color for 'nan' values
    colormap.set_bad(color='grey')
    plt.imshow(img, cmap=colormap,
        vmin=minmax[0], vmax=minmax[1], **kwargs)
    plt.title(title, fontsize=title_fontsize)
    if showscale:
        plt.colorbar(orientation="vertical")
    plt.show()

def split_meas2img(measurements, meas_operator):
    M = meas_operator.M
    N = meas_operator.N
    h,w = meas_operator.meas_shape
    # using 'nan' for custom color
    img_pos = torch.full(
        (N,),
        torch.tensor(float('nan'))
    ) # even rows
    img_neg = torch.full(
        (N,),
        torch.tensor(float('nan'))
    ) # odd rows

    # split the measurements in pos/neg
    # then apply meas2img to each
    meas = measurements.view(2*M)
    meas_pos = meas[0::2]
    meas_neg = meas[1::2]

    # fill pos/neg with the measurements
    img_pos[meas_operator.indices[:M]] = meas_pos
    img_neg[meas_operator.indices[:M]] = meas_neg

    # concatenate and reshape the images
    img = torch.cat(
        (   img_pos.reshape(h,w),
            img_neg.reshape(h,w)
        ),
        dim=0)

    return img

def center_measurements(measurements):
    max_val = measurements[
        ~measurements.isnan()
    ].max()
    min_val = measurements[
        ~measurements.isnan()
    ].min()
    return measurements - (max_val + min_val) / 2
```

REFERENCES

1. H. Fabelo, S. Ortega, A. Szolna, *et al.*, "In-Vivo Hyperspectral Human Brain Image Database for Brain Cancer Detection," IEEE Access **7**, 39098–39116 (2019).
2. 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).
3. 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.
4. 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).

Table S1. Reconstruction metrics for different noise levels ν of the Pinv-PnP methods. We computed the RMSE and SSIM considering 384 images of the ImageNet reconstruction set for three image intensities ($\alpha = 2$, $\alpha = 10$, $\alpha = 50$ photons). The simulations consider the square subsampling strategy with $\times 4$ reduction for 128×128 images. The reported results correspond to mean (standard deviation) of the metrics across images.

Methods	RMSE	SSIM
$\alpha = 2$		
$\nu = 70$	0.514 (0.095)	0.327 (0.031)
$\nu = 80$	0.456 (0.073)	0.362 (0.029)
$\nu = 90$	0.386 (0.037)	0.403 (0.032)
$\nu = 95$	0.364 (0.028)	0.410 (0.031)
$\nu = 100$	0.363 (0.035)	0.413 (0.031)
$\nu = 105$	0.349 (0.024)	0.420 (0.031)
$\nu = 110$	0.339 (0.024)	0.435 (0.032)
$\alpha = 10$		
$\nu = 30$	0.321 (0.028)	0.492 (0.025)
$\nu = 35$	0.303 (0.026)	0.526 (0.025)
$\nu = 40$	0.275 (0.019)	0.559 (0.024)
$\nu = 45$	0.266 (0.018)	0.564 (0.025)
$\nu = 50$	0.268 (0.019)	0.563 (0.029)
$\nu = 55$	0.269 (0.019)	0.555 (0.032)
$\nu = 60$	0.271 (0.021)	0.558 (0.035)
$\alpha = 50$		
$\nu = 10$	0.256 (0.017)	0.612 (0.017)
$\nu = 15$	0.241 (0.016)	0.663 (0.018)
$\nu = 20$	0.224 (0.014)	0.689 (0.018)
$\nu = 25$	0.227 (0.015)	0.666 (0.023)
$\nu = 30$	0.235 (0.017)	0.642 (0.028)
$\nu = 35$	0.240 (0.017)	0.621 (0.030)

Table S2. Reconstruction metrics for different regularization parameters μ of the DPGD-PnP method. We computed the RMSE and SSIM considering 384 images of the ImageNet reconstruction set for three image intensities ($\alpha = 2, \alpha = 10, \alpha = 50$ photons). The simulations consider the square subsampling strategy with $\times 4$ reduction for 128×128 images. The reported results correspond to mean (standard deviation) of the metrics across images.

Scenario	RMSE	SSIM
$\alpha = 2$		
$\mu = 2000$	0.914 (0.048)	0.179 (0.015)
$\mu = 4000$	0.531 (0.094)	0.274 (0.011)
$\mu = 4500$	0.504 (0.099)	0.284 (0.009)
$\mu = 5000$	0.465 (0.055)	0.282 (0.007)
$\mu = 5500$	0.452 (0.047)	0.273 (0.008)
$\mu = 6000$	0.449 (0.042)	0.263 (0.007)
$\mu = 8000$	0.492 (0.077)	0.210 (0.006)
$\mu = 10000$	0.556 (0.036)	0.165 (0.007)
$\alpha = 10$		
$\mu = 1000$	0.481 (0.105)	0.370 (0.024)
$\mu = 2000$	0.318 (0.027)	0.500 (0.016)
$\mu = 2500$	0.293 (0.020)	0.504 (0.016)
$\mu = 3000$	0.287 (0.023)	0.500 (0.015)
$\mu = 3500$	0.299 (0.017)	0.474 (0.015)
$\mu = 4000$	0.305 (0.018)	0.442 (0.018)
$\mu = 5000$	0.324 (0.019)	0.395 (0.018)
$\alpha = 50$		
$\mu = 500$	0.280 (0.026)	0.586 (0.020)
$\mu = 1000$	0.229 (0.017)	0.669 (0.016)
$\mu = 1200$	0.217 (0.012)	0.682 (0.016)
$\mu = 1500$	0.216 (0.013)	0.666 (0.017)
$\mu = 1800$	0.224 (0.015)	0.633 (0.022)
$\mu = 2000$	0.234 (0.019)	0.619 (0.026)
$\mu = 3000$	0.263 (0.020)	0.490 (0.030)
$\mu = 4000$	0.286 (0.021)	0.460 (0.030)