Problem Set 4: Denoiser Priors and Self-Supervised Reconstruction

ECE-491 Machine Learning for Medical Imaging
Fall 2026

Assigned: Week 12 **Due:** start of class, Week 14

Instructions. Submit a single PDF containing your derivations, figures, and discussion, together with the code that generated every figure. Use Python, PyTorch, and DeepInverse 0.5. Implement the networks, their training loops, DPIR, ImMAP, and the SSDU loss yourself; use DeepInverse for the MRI physics, the sampling masks, the conjugate-gradient (CG) solver, and the metrics. Training requires a GPU, such as the free tier of Google Colab. Use single precision throughout, and save the weights of every network you train.

Overview. Problem 1 trains a bias-free denoiser on MR images. Problems 2 and 3 use it, without further training, as the prior of a plug-and-play method (DPIR) and of a diffusion method (ImMAP). Problem 4 trains an unrolled network (LPDSNet) with and without fully sampled data (SSDU), and Problem 5 compares the four on noisy, undersampled multicoil brain MRI.

Data and forward model. Use the MoDL brain dataset of Problem Sets 1 and 2, the file `dataset.hdf5` at `https://zenodo.org/records/6481291`. `trnOrg` and `trnCsm` hold 360 slices of four subjects with their 12 coil sensitivity maps, and `tstOrg` and `tstCsm` hold 164 slices of a fifth subject. Scale each slice so that $\max_n |x_n| = 1$. The forward model is $\boldsymbol{A} = \boldsymbol{MFS}$, with $\boldsymbol{S}$ the coil sensitivities, $\boldsymbol{F}$ the unitary 2D DFT of each coil image, and $\boldsymbol{M}$ a mask of phase-encoding lines drawn by `GaussianMaskGenerator` with acceleration $R = 4$ and 8% of the lines at the center. The measurements are $\boldsymbol{y} = \boldsymbol{Ax} + \boldsymbol{\nu}$. We identify a complex vector with the real vector of its stacked real and imaginary parts, and write $N = 2 \cdot 256 \cdot 232$ for the number of real unknowns; $\boldsymbol{A}^{\mathrm{H}}$ then also acts as the transpose of $\boldsymbol{A}$. A noise level σ is the standard deviation of every real component, so $\boldsymbol{\nu} \sim \mathcal{N}(\boldsymbol{0}, \sigma_y^2 \boldsymbol{I})$. Test every method on every eighth test slice (21 slices) at $\sigma_y \in \{0.01, 0.02, 0.05\}$, with the same mask and noise seed per slice for all methods, and report the PSNR and SSIM of the magnitude images.

DeepInverse walkthrough.

```
import h5py, torch, deepinv as dinv
from torch.func import vjp
with h5py.File("dataset.hdf5", "r") as f:
    x = torch.from_numpy(f["trnOrg"][:])          # (360, 256, 232) complex
    s = torch.from_numpy(f["trnCsm"][:])          # (360, 12, 256, 232) complex
x = x / x.abs().amax(dim=(-2, -1), keepdim=True)
x = torch.view_as_real(x).permute(0, 3, 1, 2)     # (360, 2, 256, 232) real
gen = dinv.physics.generator.GaussianMaskGenerator(
          (256, 232), acceleration=4, center_fraction=0.08)
physics = dinv.physics.MultiCoilMRI(img_size=(256, 232),
              noise_model=dinv.physics.GaussianNoise(sigma=0.02))
physics.update(mask=gen.step(batch_size=B, seed=i)["mask"], coil_maps=s[idx])
y = physics(x[idx], sigma=sigma_y, seed=i)  # (B, 2, 12, 256, 232) masked k-space
x0 = physics.A_adjoint(y)              # zero-filled image A^H y
AhA = lambda v: physics.A_adjoint(physics.A(v))
v = dinv.optim.linear.conjugate_gradient(lambda v: AhA(v) + alpha * v, b,
                                         max_iter=10, init=v_prev)
z_hat, pullback = vjp(f, z)            # f(z), and pullback(v)[0] = J^T v
psnr, ssim = dinv.metric.PSNR(complex_abs=True), dinv.metric.SSIM(complex_abs=True)
```

Problem 1 (A Bias-Free Denoiser for MR Images)**.** Train a blind Gaussian denoiser f for complex brain images. Problems 2 and 3 use it as their only prior.

(a) *Tweedie's formula.* Let $\boldsymbol{z} = \boldsymbol{x} + \sigma\boldsymbol{n}$, with $\boldsymbol{x} \sim p_X$ independent of $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and let p_σ be the density of $\boldsymbol{z}$. Show that the minimum mean-squared-error denoiser satisfies

$$\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{z}] = \boldsymbol{z} + \sigma^2 \nabla \log p_\sigma(\boldsymbol{z}).$$

(Hint: differentiate $p_\sigma(\boldsymbol{z}) = \int p_X(\boldsymbol{x})\, g_\sigma(\boldsymbol{z} - \boldsymbol{x})\, d\boldsymbol{x}$, with g_σ the Gaussian density.)

(b) *Bias-free networks.* A network built from bias-free convolutions and transposed convolutions, ReLUs, average pooling, reflection padding, concatenations, and residual connections satisfies $f(\alpha\boldsymbol{z}) = \alpha f(\boldsymbol{z})$ for every $\alpha \geq 0$. Show this, and show that then $f(\boldsymbol{z}) = \boldsymbol{J}_{\boldsymbol{z}}\boldsymbol{z}$, where $\boldsymbol{J}_{\boldsymbol{z}}$ is the Jacobian of f at $\boldsymbol{z}$. Bias-free denoisers generalize to noise levels above their training range (Mohan et al., ICLR 2020); Problem 3 relies on this.

(c) *Implementation.* Implement a bias-free UNet (BF-UNet; Ronneberger et al., MICCAI 2015) with two input and two output channels (the real and imaginary parts), four scales, and $32 \cdot 2^s$ channels at scale $s = 0, \ldots, 3$. Each scale applies two 3×3 conv–ReLU layers. The encoder moves down a scale by 2×2 average pooling. The decoder moves up by a 2×2 transposed convolution with stride 2 that halves the channels, concatenates the result with the encoder features of the same scale, and applies two conv–ReLU layers. A final 1×1 convolution predicts the noise, $f(\boldsymbol{z}) = \boldsymbol{z} - R(\boldsymbol{z})$. Use no biases and no normalization layers, and reflect-pad inputs to a multiple of 8. Report the parameter count, and verify that $f(2\boldsymbol{z}) = 2f(\boldsymbol{z})$ to floating-point precision.

(d) *Training and generalization.* Train on random 128×128 crops of the 360 training slices with random flips, 90° rotations, and a global phase $e^{i\phi}$ with $\phi \sim \mathcal{U}[0, 2\pi)$, batch size 16, a noise level drawn uniformly from $[0, 0.5]$ for each crop, the mean squared error, and Adam with learning rate 10^{-3}, divided by 10 after 60% and after 90% of 20,000 iterations. Plot the training loss and the test PSNR at $\sigma = 0.1$ against iteration. Evaluate on the test slices at $\sigma \in \{0.05, 0.10, \ldots, 1\}$, plot the output PSNR against the input PSNR with the training range shaded, and show one slice denoised at $\sigma = 0.2$ and at $\sigma = 1$. Does the denoiser generalize beyond $\sigma = 0.5$?

Problem 2 (Plug-and-Play: DPIR)**.** DPIR (Zhang et al., IEEE TPAMI 2022) approaches

$$\min_{\boldsymbol{x}} \ \tfrac{1}{2\sigma_y^2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda R(\boldsymbol{x})$$

by half-quadratic splitting with a penalty $\mu_k = \lambda/\sigma_k^2$ that grows as the noise levels $\sigma_1 > \cdots > \sigma_K = \sigma_y$ decrease, and replaces the proximal step on R by a denoiser. From $\boldsymbol{z}_0 = \boldsymbol{A}^{\mathrm{H}}\boldsymbol{y}$, for $k = 1, \ldots, K$,

$$\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \ \tfrac{1}{2\sigma_y^2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \tfrac{\mu_k}{2}\|\boldsymbol{x} - \boldsymbol{z}_{k-1}\|_2^2, \qquad \boldsymbol{z}_k = f(\boldsymbol{x}_k),$$

with output $\boldsymbol{z}_K$ and the σ_k spaced logarithmically.

(a) Show that the $\boldsymbol{x}$-step solves $(\boldsymbol{A}^{\mathrm{H}}\boldsymbol{A} + \alpha_k\boldsymbol{I})\boldsymbol{x} = \boldsymbol{A}^{\mathrm{H}}\boldsymbol{y} + \alpha_k\boldsymbol{z}_{k-1}$ with $\alpha_k = \lambda\sigma_y^2/\sigma_k^2$. Show that the step it replaces, $\arg\min_{\boldsymbol{z}} \frac{\mu_k}{2}\|\boldsymbol{x}_k - \boldsymbol{z}\|_2^2 + \lambda R(\boldsymbol{z})$, is the MAP denoiser of $\boldsymbol{x}_k$ for Gaussian noise of level σ_k when $R = -\log p_X$. Your denoiser is blind and ignores σ_k: what does the schedule still control?

(b) Implement DPIR with $K = 8$, solving each $\boldsymbol{x}$-step with 10 CG iterations warm-started at $\boldsymbol{z}_{k-1}$. Choose λ and σ_1 by a grid search on two training slices at $\sigma_y = 0.02$, and report the grid. On the test slices, report the PSNR, SSIM, and time per slice at each σ_y, and plot the PSNR of $\boldsymbol{z}_k$ against k for one slice.

Problem 3 (Diffusion: ImMAP)**.** Kadkhodaie and Simoncelli (NeurIPS 2021) draw images from the prior implicit in a blind denoiser by the coarse-to-fine stochastic ascent $\boldsymbol{z}_{t+1} = \boldsymbol{z}_t + h_t\big(f(\boldsymbol{z}_t) - \boldsymbol{z}_t\big) + \boldsymbol{\epsilon}_t$, started from noise; the noise level of $\boldsymbol{z}_t$ falls as t grows, as in the reverse process of a diffusion model. ImMAP (Janjušević et al., 2025) adds the likelihood of noisy multicoil measurements to approximate the MAP estimate:

> **ImMAP**$(\boldsymbol{y}, \boldsymbol{A}, \sigma_y, f)$, with $\beta = 0.05$, $h_0 = 0.01$, $\sigma_L = 0.01$. Draw $\boldsymbol{z}_1 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$; set $t = 1$ and $\sigma_1 = 1$. While $\sigma_t > \sigma_L$:
>
> 1. $\hat{\boldsymbol{z}}_t = f(\boldsymbol{z}_t)$, with the vector-Jacobian product $\boldsymbol{v} \mapsto \boldsymbol{J}_t^\top \boldsymbol{v}$ of f at $\boldsymbol{z}_t$;
> 2. $\sigma_t^2 = \|\hat{\boldsymbol{z}}_t - \boldsymbol{z}_t\|_2^2 / N$ and $v_t = \sigma_t^2/(1+\sigma_t^2)$;
> 3. $\boldsymbol{\Sigma}_t = \sigma_y^2 \boldsymbol{I} + v_t \boldsymbol{A}\boldsymbol{A}^{\mathrm{H}}$ and $\boldsymbol{u}_t = -\boldsymbol{J}_t^\top \boldsymbol{A}^{\mathrm{H}} \boldsymbol{\Sigma}_t^{-1}(\boldsymbol{A}\hat{\boldsymbol{z}}_t - \boldsymbol{y})$;
> 4. $h_t = h_0 t / \big(1 + h_0(t-1)\big)$ and $\gamma_t^2 = \sigma_t^2\big[(1-\beta h_t)^2 - (1-h_t)^2\big]$;
> 5. $\boldsymbol{z}_{t+1} = \boldsymbol{z}_t + h_t(\hat{\boldsymbol{z}}_t - \boldsymbol{z}_t + \sigma_t^2 \boldsymbol{u}_t) + \gamma_t \boldsymbol{\epsilon}_t$, with $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $t \leftarrow t+1$.
>
> Return $f(\boldsymbol{z}_t)$.

The paper uses $\beta = 0.01$, which needs more iterations, and returns $\boldsymbol{z}_t$, which still contains noise of level σ_L that one more denoising removes.

(a) *Derivation.*

(i) Suppose $\boldsymbol{z}_t = \boldsymbol{x} + \sigma \boldsymbol{n}$ with $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and f removes the noise. Why does line 2 estimate σ^2?

(ii) Model the clean image given $\boldsymbol{z}_t$ as $\boldsymbol{x} \mid \boldsymbol{z}_t \sim \mathcal{N}(\hat{\boldsymbol{z}}_t, v_t \boldsymbol{I})$ (a Laplace approximation). Show that $\boldsymbol{y} \mid \boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{A}\hat{\boldsymbol{z}}_t, \boldsymbol{\Sigma}_t)$, and that, holding $\boldsymbol{\Sigma}_t$ fixed, $\nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{y} \mid \boldsymbol{z}_t) = \boldsymbol{u}_t$. With Problem 1(a), conclude that for $\gamma_t = 0$ line 5 is a gradient-ascent step of size $h_t \sigma_t^2$ on $\log p_{\sigma_t}(\boldsymbol{z}_t) + \log p(\boldsymbol{y} \mid \boldsymbol{z}_t)$. Explain the limits of v_t as $\sigma_t \to 0$ and $\sigma_t \to \infty$ for images of unit scale.

(iii) Show that $\boldsymbol{A}^{\mathrm{H}}(\sigma_y^2 \boldsymbol{I} + v \boldsymbol{A}\boldsymbol{A}^{\mathrm{H}})^{-1} = (\sigma_y^2 \boldsymbol{I} + v \boldsymbol{A}^{\mathrm{H}}\boldsymbol{A})^{-1} \boldsymbol{A}^{\mathrm{H}}$, so that $\boldsymbol{u}_t$ needs a CG solve with the operator of the DPIR $\boldsymbol{x}$-step. What is α?

(iv) Suppose $\boldsymbol{z}_t = \boldsymbol{x} + \sigma_t \boldsymbol{n}_t$, $\hat{\boldsymbol{z}}_t = \boldsymbol{x}$, and ignore the likelihood. Show that $\boldsymbol{z}_{t+1} = \boldsymbol{x} + \sigma_{t+1} \boldsymbol{n}_{t+1}$ with $\boldsymbol{n}_{t+1} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\sigma_{t+1} = (1 - \beta h_t)\sigma_t$. What do $\beta = 1$ and $\beta \to 0$ mean? Estimate the number of iterations from $\sigma = 1$ to σ_L.

(b) *Implementation.* Implement ImMAP with your BF-UNet, computing $\boldsymbol{u}_t$ by part (a)(iii) with 10 CG iterations warm-started at the previous solution. On the test slices, report the PSNR, SSIM, number of iterations, and time per slice at each σ_y. For one slice at $\sigma_y = 0.02$, plot σ_t against t together with the prediction of part (a)(iv), plot the PSNR of $\boldsymbol{z}_t$ and of $\hat{\boldsymbol{z}}_t$, and show $\boldsymbol{z}_t$ at a few t. Run ImMAP on this slice with four seeds, and show the pixelwise standard deviation of the outputs. On five test slices, compare $\beta \in \{0.01, 0.05, 0.1, 0.5\}$ in PSNR and in the number of iterations.

Problem 4 (LPDSNet: Supervised and Self-Supervised)**.** Primal-dual splitting (PDS) solves $\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{D}^{\mathrm{H}}\boldsymbol{x}\|_1$, with an analysis transform $\boldsymbol{D}^{\mathrm{H}}$ and $\|\cdot\|_1$ the sum of complex magnitudes, by iterating

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \eta\big(\boldsymbol{A}^{\mathrm{H}}(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}) + \boldsymbol{D}\boldsymbol{q}_k\big), \\ \tilde{\boldsymbol{x}}_{k+1} &= \boldsymbol{x}_{k+1} + \theta(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \qquad \boldsymbol{q}_{k+1} = \Pi_\lambda(\boldsymbol{q}_k + \beta \boldsymbol{D}^{\mathrm{H}} \tilde{\boldsymbol{x}}_{k+1}), \end{aligned}$$

where the clipping $\Pi_\lambda(q) = \operatorname{sign}(q)\min(|q|, \lambda)$ acts on the magnitude of each complex entry. LPDSNet (Janjušević et al., ISBI 2025) unrolls K iterations with learned operators. From

$\boldsymbol{x}_0 = \mathbf{0}$ and $\boldsymbol{q}_0 = \mathbf{0}$,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta_k\big(\boldsymbol{A}^{\mathrm{H}}(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}) + \boldsymbol{B}_k\boldsymbol{q}_k\big),$$
$$\tilde{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_{k+1} + \theta_k(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \qquad \boldsymbol{q}_{k+1} = \Pi_{\lambda_k}(\boldsymbol{q}_k + \boldsymbol{C}_k^{\mathrm{H}}\tilde{\boldsymbol{x}}_{k+1}),$$

and $\hat{\boldsymbol{x}} = \boldsymbol{x}_K$. Unlike the paper, we do not subtract mean$(\boldsymbol{A}^{\mathrm{H}}\boldsymbol{y})$ from $\boldsymbol{A}^{\mathrm{H}}\boldsymbol{y}$ and add it back: $\boldsymbol{A}^{\mathrm{H}}\boldsymbol{A}$ vanishes outside the head, where the shifted iteration drifts without bound. Here $\boldsymbol{C}_k^{\mathrm{H}}$ is a stride-2 convolution from one to M complex channels, $\boldsymbol{B}_k$ a stride-2 transposed convolution from M channels to one, both with complex 7×7 kernels, and the clipping levels are noise-adaptive, $\lambda_k = \lambda_{0,k} + \lambda_{1,k}\hat{\sigma} \in \mathbb{R}_+^M$, with $\hat{\sigma} = \sigma_y$ the known noise level.

(a) *Primal-dual splitting.*

(i) Using $\lambda\|\boldsymbol{u}\|_1 = \max_{|q_i|\le\lambda} \mathrm{Re}(\boldsymbol{q}^{\mathrm{H}}\boldsymbol{u})$, write the problem as a saddle-point problem in $(\boldsymbol{x}, \boldsymbol{q})$, and show that PDS alternates a gradient step in $\boldsymbol{x}$ with a projected gradient-ascent step in $\boldsymbol{q}$.

(ii) A sufficient condition for convergence is $(1/\eta - L)/\beta \ge \|\boldsymbol{D}\|_2^2$, with $L = \|\boldsymbol{A}\|_2^2$. Show that $\|\boldsymbol{A}\|_2 \le 1$ when $\sum_c |s_{c,n}|^2 \le 1$ at every pixel n, and check this for the dataset. Show that the initialization in part (c) makes the untrained LPDSNet K iterations of a convergent PDS.

(b) *SSDU* (Yaman et al., MRM 2020). Without fully sampled data, split the sampled k-space locations Ω of each training slice into disjoint sets Ξ (network input) and Λ (loss), and minimize $\sum \mathcal{L}\big(\boldsymbol{y}_\Lambda, \boldsymbol{A}_\Lambda f_\Theta(\boldsymbol{y}_\Xi; \boldsymbol{A}_\Xi)\big)$, drawing a new split at every visit; at test time the network receives all of Ω. For the squared loss and noise independent across k-space samples, show that $\mathbb{E}\|\boldsymbol{y}_\Lambda - \boldsymbol{A}_\Lambda\hat{\boldsymbol{x}}\|_2^2 = \mathbb{E}\|\boldsymbol{A}_\Lambda(\boldsymbol{x} - \hat{\boldsymbol{x}})\|_2^2 + c$, with c independent of the network, when $\hat{\boldsymbol{x}}$ depends only on $\boldsymbol{y}_\Xi$. Why must Ξ and Λ be disjoint?

(c) *Implementation and training.* Implement LPDSNet with $K = 20$ and $M = 64$. Implement a complex convolution with kernel $\boldsymbol{W}_r + i\boldsymbol{W}_i$ as one real convolution from $(\boldsymbol{x}_r, \boldsymbol{x}_i)$ to $(\boldsymbol{W}_r * \boldsymbol{x}_r - \boldsymbol{W}_i * \boldsymbol{x}_i,\ \boldsymbol{W}_r * \boldsymbol{x}_i + \boldsymbol{W}_i * \boldsymbol{x}_r)$, likewise for the transposed convolution, and compute $\Pi_\lambda(q) = q\lambda/\max(|q|, \lambda)$. Initialize $\boldsymbol{C}_k = \boldsymbol{B}_k = \boldsymbol{D}$ with random filters $\boldsymbol{D}$ scaled to $\|\boldsymbol{D}\|_2 = 1$ by power iteration, $\eta_k = 1/2$, $\theta_k = 1$, $\lambda_{0,k} = 10^{-3}$, and $\lambda_{1,k} = 0$; after every step, project η_k, $\lambda_{0,k}$, and $\lambda_{1,k}$ onto $[0, \infty)$ and θ_k onto $[0, 1]$. Give each training slice one fixed mask and one noisy measurement with $\sigma_y \sim \mathcal{U}[0.01, 0.05]$, shared by both networks. Train one network with the supervised loss $\mathcal{L}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2/\|\boldsymbol{x}\|_2 + \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_1/\|\boldsymbol{x}\|_1$ and one with the same loss in k-space under SSDU, with Ξ keeping each sampled location (in all coils) with probability 0.8 and always the central 10×10 locations. Use Adam with a learning rate annealed by a cosine from $5 \cdot 10^{-4}$ to $2 \cdot 10^{-6}$, batch size 4, and 30 epochs. Plot the training losses and the test PSNR against epoch.

(d) *Evaluation.* Report the PSNR and SSIM of both networks and of $\boldsymbol{A}^{\mathrm{H}}\boldsymbol{y}$ on the test slices at each σ_y and at $\sigma_y = 0.1$, outside the training range. Does SSDU match supervised training? Show both reconstructions of one slice.

Problem 5 (Comparison)**.** Compare the four approaches on the same measurements.

(a) Tabulate the PSNR, SSIM, and reconstruction time per slice of $\boldsymbol{A}^{\mathrm{H}}\boldsymbol{y}$, DPIR, ImMAP, and both LPDSNets on the test slices at each σ_y, and at $R = 8$ (4% of the lines at the center) with $\sigma_y = 0.02$, without retraining. Show the five reconstructions of one slice at $R = 4$ and $\sigma_y = 0.02$, with error maps.

(b) For each method, state what training data it needs (fully sampled images, or undersampled k-space only) and what must change when the mask or the noise level changes. Which method would you deploy when no fully sampled scans exist, and which when the acquisition protocol changes often? Why?