Universal Inverse Problem - Repo Summary

What it is

Research code for the paper ’Solving Linear Inverse Problems Using the Prior Implicit in a Denoiser’ that provides pretrained universal CNN denoisers and an inverse problem solver. Includes a demo notebook for sampling from the implicit prior and solving linear inverse problems without additional training.

Who it’s for

Researchers or practitioners working on image denoising and linear inverse problems who want to study denoiser priors or reproduce the paper’s experiments.

What it does

- Provides pretrained "blind" CNN denoisers (BF_CNN variants) for Gaussian noise removal.
- Loads denoiser weights trained on BSD300, BSD400, and MNIST across noise ranges.
- Samples from the implicit image prior using the denoiser-defined gradient.
- Solves linear inverse problems: inpainting, deblurring, super-resolution, random missing pixels, compressive sensing.
- Includes utilities for test image loading, visualization, and PSNR/SSIM evaluation.
- Runs on CPU or GPU via PyTorch (CUDA if available).

How it works

- Demo.ipynb orchestrates experiments and calls helper functions.
- code/Utils_inverse_prob.py loads pretrained weights from denoisers/ and test images from test_images/, and provides plotting and metrics.
- code/network.py defines the BF_CNN denoiser model used by the solver.
- code/algorithm_inv_prob.py implements univ_inv_sol: iteratively updates an image y using the denoiser and a task object with M and M_T measurement operators to produce a reconstruction.

How to run

- Install Python 3.7.6 and required packages: numpy 1.19.4, skimage 0.17.2, matplotlib 1.19.4, PyTorch 1.7.0, plus argparse, os, time, sys, gzip (per README).
- Open Demo.ipynb in Jupyter and run the notebook to load a denoiser and run sampling/inverse problems.
- Exact environment setup and Jupyter launch commands: Not found in repo.