Image Denoising

Arthur Leclaire

TELECOM Paris

IP PARIS

Master MVA
November, 10th, 2026

Plan

Bayesian Imaging
- Bayesian Image Restoration
- Bayesian Image Denoising
- Metrics for Inverse Problems

Properties of MAP/MMSE Denoisers

Denoising with Deep Neural Networks

Denoisers for Restoration and Generation
- Plug-and-Play Image Restoration
- Diffusion Generative Models
- Diffusion Posterior Sampling

Bayesian Modeling for Inverse Problems

We consider an inverse problem of the form

$$y = Ax_{GT} + w$$

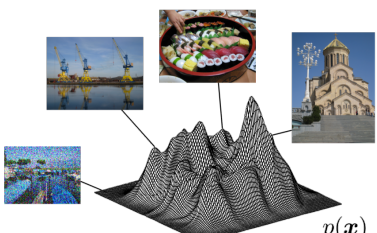
where $x_{GT} \in \mathbf{R}^d$ is the clean image to recover, $A \in \mathbf{R}^{m \times d}$ and $w \sim \mathcal{N}(0, \sigma^2 \mathrm{Id})$.

- The observation model defines the likelihood function

$$p(y|x) \propto \exp\left(-\frac{1}{2\sigma^2}\|y - Ax\|^2\right).$$

- We choose a prior $p(x)$ that favors clean images.
- Bayes formula then gives the posterior distribution

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \propto p(y|x)p(x).$$



From Bayesian Modeling to an Optimization Problem

- The likelihood is linked to the data-fidelity $f = f_y$: $\quad p(y|x) \propto e^{-f_y(x)}$.
- The prior is linked to a regularization $g = \lambda R$: $\quad p(x) \propto e^{-\lambda R(x)}$ (with $\lambda \geqslant 0$).

We can write the neg-log-posterior distribution as

$$-\log p(x|y) = \text{Cst} + f_y(x) + \lambda R(x) = \text{Cst} + \frac{1}{2\sigma^2}\|y - Ax\|^2 + \lambda R(x).$$

From Bayesian Modeling to an Optimization Problem

- The likelihood is linked to the data-fidelity $f = f_y$: $\quad p(y|x) \propto e^{-f_y(x)}$.
- The prior is linked to a regularization $g = \lambda R$: $\quad p(x) \propto e^{-\lambda R(x)}$ (with $\lambda \geqslant 0$).

We can write the neg-log-posterior distribution as

$$-\log p(x|y) = \text{Cst} + f_y(x) + \lambda R(x) = \text{Cst} + \frac{1}{2\sigma^2}\|y - Ax\|^2 + \lambda R(x).$$

We may try to do many things with this posterior distribution
(punctual estimation, Bayesian credibility regions, ...)

In particular, there are two Bayesian problems that concentrate the attention:

- **Maximum A-Posteriori (MAP)** estimation
- **Minimum-Mean Square Error (MMSE)** estimation

MAP and MMSE estimators

Maximum A-Posteriori (MAP) estimation consists in solving

$$\underset{x}{\text{Argmax}}\, p(x|y) = \underset{x}{\text{Argmin}} - \log p(x|y) = \underset{x}{\text{Argmin}}\, f(x) + g(x)$$

This is an optimization problem.

MAP and MMSE estimators

Maximum A-Posteriori (MAP) estimation consists in solving

$$\underset{x}{\text{Argmax}}\, p(x|y) = \underset{x}{\text{Argmin}} - \log p(x|y) = \underset{x}{\text{Argmin}}\, f(x) + g(x)$$

This is an optimization problem.

Minimum-Mean Square Error (MMSE) estimation consists in computing

$$\mathbb{E}[X|Y] = \int_{\mathbf{R}^d} x\, p(x|y) dx.$$

This is an integration problem.

MAP and MMSE estimators

Maximum A-Posteriori (MAP) estimation consists in solving

$$\underset{x}{\text{Argmax}}\, p(x|y) = \underset{x}{\text{Argmin}} - \log p(x|y) = \underset{x}{\text{Argmin}}\, f(x) + g(x)$$

This is an optimization problem.

Minimum-Mean Square Error (MMSE) estimation consists in computing

$$\mathbb{E}[X|Y] = \int_{\mathbf{R}^d} x\, p(x|y) dx.$$

This is an integration problem.
$\mathbb{E}[X|Y]$ is the measurable function $\varphi(Y)$ that minimizes the mean-square error prediction

$$\mathbb{E}[\|X - \varphi(Y)\|^2].$$

MAP and MMSE estimators

Maximum A-Posteriori (MAP) estimation consists in solving

$$\underset{x}{\text{Argmax}}\, p(x|y) = \underset{x}{\text{Argmin}} - \log p(x|y) = \underset{x}{\text{Argmin}}\, f(x) + g(x)$$

This is an optimization problem.

Minimum-Mean Square Error (MMSE) estimation consists in computing

$$\mathbb{E}[X|Y] = \int_{\mathbf{R}^d} x\, p(x|y) dx.$$

This is an integration problem.
$\mathbb{E}[X|Y]$ is the measurable function $\varphi(Y)$ that minimizes the mean-square error prediction

$$\mathbb{E}[\|X - \varphi(Y)\|^2].$$

Both problems are in general very difficult (because x lives in a very high-dimensional space).

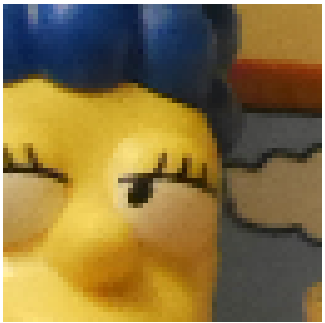
NB: In the simple case of a Gaussian posterior $p(x|y)$, then MAP and MMSE coincide.

Example of Gaussian denoising

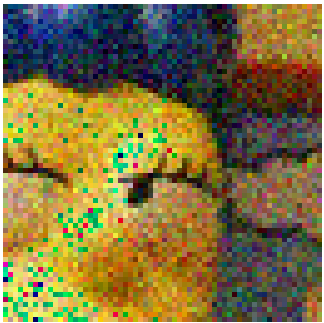
The simplest image inverse problem is **image denoising** (for which $A = \mathrm{Id}$):

$$y = x_{GT} + \xi \quad \text{where} \quad \xi \sim \mathcal{N}(0, \sigma^2 \mathrm{Id}).$$

We want to estimate x_{GT} from a single realization of y...



x_{GT}



y

Maximum likelihood for Gaussian denoising?

For Gaussian denoising, the likelihood function expresses as

$$p(y|x) \propto \exp\left(-\frac{1}{2\sigma^2}\|y-x\|^2\right)$$

The solution to maximizing the likelihood is thus $\underset{x}{\text{Argmax}}\, p(y|x) = y$ ✗ This is useless!

Maximum likelihood for Gaussian denoising?

For Gaussian denoising, the likelihood function expresses as

$$p(y|x) \propto \exp\left(-\frac{1}{2\sigma^2}\|y-x\|^2\right)$$

The solution to maximizing the likelihood is thus $\underset{x}{\text{Argmax}}\, p(y|x) = y$ ✗ This is useless!

If we had several independent realizations $y_1, \ldots, y_K$, we could use the joint likelihood

$$p(y_1, \ldots, y_K|x) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{k=1}^{K}\|y_k-x\|^2\right).$$

Maximum likelihood for Gaussian denoising?

For Gaussian denoising, the likelihood function expresses as

$$p(y|x) \propto \exp\left(-\frac{1}{2\sigma^2}\|y-x\|^2\right)$$

The solution to maximizing the likelihood is thus $\underset{x}{\text{Argmax}}\, p(y|x) = y$ ✗ This is useless!

If we had several independent realizations $y_1, \ldots, y_K$, we could use the joint likelihood

$$p(y_1, \ldots, y_K|x) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{k=1}^{K}\|y_k-x\|^2\right).$$

Then maximizing the likelihood would lead to

$$\underset{x}{\text{Argmax}}\, p(y_1, \ldots, y_K|x) = \underset{x}{\text{Argmin}} \sum_{k=1}^{K}\|y_k-x\|^2 = \frac{1}{K}\sum_{k=1}^{K} y_k.$$

✗ But unfortunately, in practice, we usually have only one noisy input.

Denoising Examples

In a previous session, we already commented on Tychonov/TV_ϵ denoising.



Noisy
PSNR = 19.93



Tychonov denoising
PSNR = 25.89



TV_ε denoising
PSNR = 27.21

Denoising with Various Priors

Several solutions have been proposed for image denoising:

- Total variation denoising [Rudin et al., 1992, Chambolle, 2004]
- Wavelet denoising [Mallat, 2009]
- Patch-based denoising [Buades et al., 2005, Zoran and Weiss, 2011, Yu et al., 2011]
- Neural denoising [Burger et al., 2012, Zhang et al., 2017, Zhang et al., 2021]

Denoising with Various Priors

Several solutions have been proposed for image denoising:

- Total variation denoising [Rudin et al., 1992, Chambolle, 2004]
- Wavelet denoising [Mallat, 2009]
- Patch-based denoising [Buades et al., 2005, Zoran and Weiss, 2011, Yu et al., 2011]
- Neural denoising [Burger et al., 2012, Zhang et al., 2017, Zhang et al., 2021]

Each of these solutions somehow imposes some regularity prior on the image.

This has led to the idea of “Regularization by Denoising” (RED) [Romano et al., 2017].

The RED paradigm allows one to imagine that there is a regularization hidden behind the denoiser.

Denoising with Various Priors

Several solutions have been proposed for image denoising:

- Total variation denoising [Rudin et al., 1992, Chambolle, 2004]
- Wavelet denoising [Mallat, 2009]
- Patch-based denoising [Buades et al., 2005, Zoran and Weiss, 2011, Yu et al., 2011]
- Neural denoising [Burger et al., 2012, Zhang et al., 2017, Zhang et al., 2021]

Each of these solutions somehow imposes some regularity prior on the image.

This has led to the idea of “Regularization by Denoising” (RED) [Romano et al., 2017].

The RED paradigm allows one to imagine that there is a regularization hidden behind the denoiser.

- One can start from a regularizing function and derive the associated denoiser.
  (We did it for Tychonov, TV_ε and wavelet regularizations.)
- Alternately, one can build a very good denoiser, and use it as prior for other inverse problems.

Denoising methods in the pre-deep era



[Donoho and Johnstone, 1994]
[Portilla et al., 2003]

[Geman and Geman, 1984]
[Perona and Malik, 1990]
[Rudin et al., 1992]



Sparcifying transforms (wavelets, dictionnaries)

Variational / Markovian Approaches

[Aharon et al., 2006]
[Dabov et al., 2007]
[Mairal et al., 2009]
[Chatterjee et al., 2011]



[Gilboa and Osher, 2007]
[Peyré et al., 2008]

Sparsity and non-locality

Non-local Total Variation



Patch-based methods

[Buades et al., 2005]
[Awate and Whitaker, 2006]

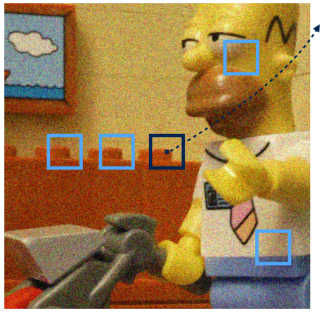


Patch-based approaches perform best (see review of [Katkovnik et al., 2010])

Non-local Means, Patch-based denoising

[Buades et al., 2005]

- NL-means exploits the redundancy of local structures in an image.
- Principle: if we find K copies of the same structure in the image we can average those K copies to reduce noise (by a factor $\sqrt{K}$).
- Actually, it is enough to do **averages weighted by the patch similarity**.



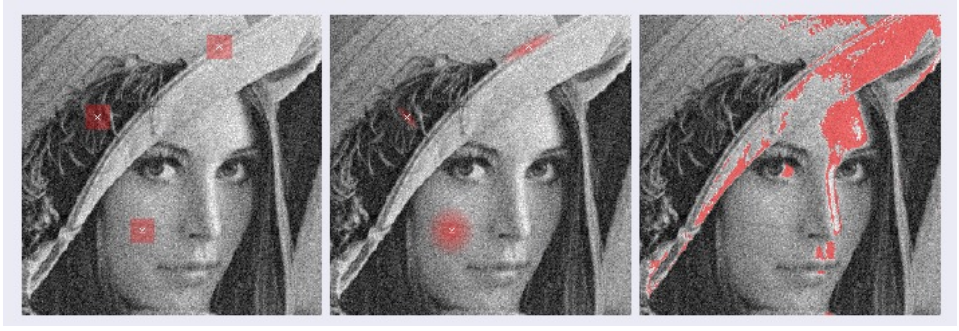
The patch at pixel i can be restored using other patches and weighting by the patch similarity:

$$\hat{x}(i) = \frac{\sum_k w_{i,k} x_k}{\sum_k w_{i,k}}$$

where $w_{i,k} \propto e^{-c\|P_i - P_k\|^2}$ is high when the patches P_i, P_k around i, k are close.

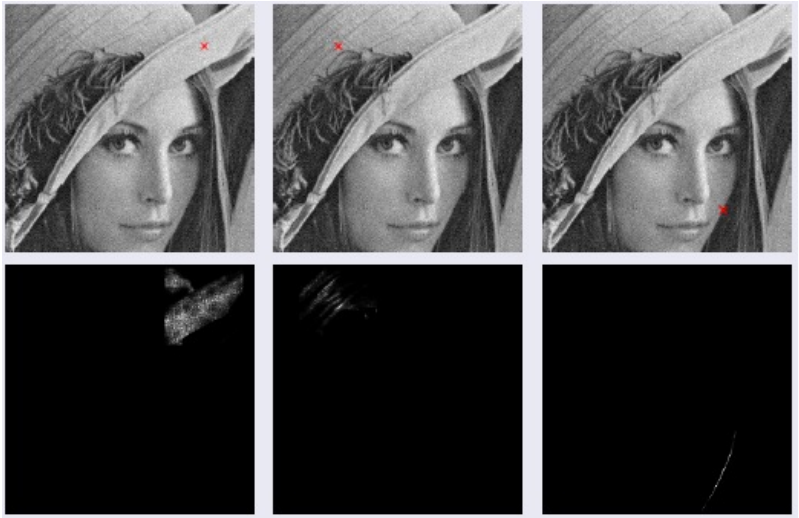
Where to find the good information?

In order to denoise, we need to do some averages, but taking values at the right places.
Local vs. non-local...



Left: simple filter
Middle: anisotropic filter
Right: non-local filter

Non-local Methods - Weight maps



Denoising example with NL-means



Denoising example with NL-means



Patch-based Models and Visual Dictionaries

Several authors made the connection between patch-based denoising and posterior estimation.
The cornerstone is to adopt a Gaussian Mixture Model (GMM) prior at the patch level.

The authors of [Zoran and Weiss, 2011] proposed to use the Expected Patch Log-Likelihood (EPLL):

$$\log p(x) = \sum_i \log p(P_i x)$$

where $P_i x$ is the patch of x at position i.
Then they choose to use a Gaussian mixture model (GMM) as patch prior.
The GMM is estimated on a collection of $2M$ clean patches from an image database.

Interestingly, such a GMM prior can also be learned directly on the degraded patches [Yu et al., 2011].

Euclidean metrics

- Given two images u and v of size $H \times W$ with graylevels between 0 and 255.
- Denote $\Omega = \{0, \dots, H-1\} \times \{0, \dots, W-1\}$ the pixel domain
- Mean Square Error $\downarrow$:

$$\mathrm{MSE}(u, v) = \frac{1}{HW} \sum_{\mathbf{x} \in \Omega} (u(\mathbf{x}) - v(\mathbf{x}))^2$$

- Root Mean Square Error $\downarrow$:

$$\mathrm{RMSE}(u, v) = \left(\frac{1}{HW} \sum_{\mathbf{x} \in \Omega} (u(\mathbf{x}) - v(\mathbf{x}))^2 \right)^{\frac{1}{2}}$$

- Peak Signal to Noise Ratio $\uparrow$:

$$\mathrm{PSNR}(u, v) = 20 \log_{10} \left(\frac{\mathrm{MAX}}{\mathrm{RMSE}(u, v)} \right) \quad (\text{where } \mathrm{MAX} = 255)$$

- Useful for inverse problems such as denoising.
- Not ideal when one hopes to generate new content. And not well correlated to Human perception!

Structural similarity index measure (SSIM ↑) [Wang et al., 2004]

Between patches:

- Given two patches x, y (typically of size 8×8 or 11×11 with a Gaussian windowing)

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \in [-1,1]$$

  with:
  - μ_x the pixel sample mean of x
  - μ_y the pixel sample mean of y
  - σ_x^2 the variance of x
  - σ_y^2 the variance of y
  - σ_{xy} the covariance of x and y
  - $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator, with the range $L = 255$ or 1 and $k_1 = 0.01$ and $k_2 = 0.03$ by default.

- $\mathrm{SSIM}(x,y)$ is the product of three terms:

$$\underset{\text{Luminance}}{l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}} \qquad \underset{\text{Contrast}}{c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}} \qquad \underset{\text{Structure}}{s(x,y) = \frac{\sigma_{xy} + c_2/2}{\sigma_x\sigma_y + c_2/2}}$$

Structural similarity index measure (SSIM ↑) [Wang et al., 2004]

Between images:

- Given two images u and v of size $H \times W$ with graylevels between 0 and 255, define the Mean-SSIM by averaging over all patches:

$$(\mathrm{M})\mathrm{SSIM}(u, v) = \mathrm{mean}(\{\mathrm{SSIM}(P_i u, P_i v)\})$$

  where $P_i u$ is the patch of u at position i.
- There are also multiscale variants.
- SSIM is not a distance, its range is $[-1, 1]$.
- SSIM is closer to a perceptual distance, especially regarding local textures.

Learned Perceptual Image Patch Similarity (LPIPS ↓) [Zhang et al., 2018b]

- Previous works on texture synthesis [Gatys et al., 2015] and style transfer [Gatys et al., 2016] are based on features computed with the VGG network [Simonyan and Zisserman, 2015].
- [Johnson et al., 2016] showed the importance of VGG features for perceptual similarity.
- This means that intermediate features of classification CNN are useful in their own: "**a good feature is a good feature**. Features that are good at semantic tasks are also good at self-supervised and unsupervised tasks, and also provide good models of both human perceptual behavior and macaque neural activity."

LPIPS model: Define a perceptual distance between 64×64 patches by computing a Euclidean norm between features:

$$\text{LPIPS}(u,v)^2 = \sum_{\text{layers } \ell} \frac{1}{H_\ell W_\ell} \sum_{i,j} \| w_\ell \odot (F^\ell(u)_{i,j} - F^\ell(v)_{i,j}) \|_2^2$$

where for each layer ℓ, the neural response $F^\ell(u)$ is weighted by channel weights $w_\ell \in \mathbf{R}^{C_\ell}$ that are learned to reproduce human evaluation of distortion between patches.

Plan

Bayesian Imaging
- Bayesian Image Restoration
- Bayesian Image Denoising
- Metrics for Inverse Problems

Properties of MAP/MMSE Denoisers

Denoising with Deep Neural Networks

Denoisers for Restoration and Generation
- Plug-and-Play Image Restoration
- Diffusion Generative Models
- Diffusion Posterior Sampling

MAP Denoiser as a Proximal Operator

Let us fix a prior p on $\mathbf{R}^d$.

If the prior is a probability density function $p : \mathbf{R}^d \to \mathbf{R}_+$, the maximum a-posteriori (MAP) denoiser can be written as

$$D_\sigma^{\text{MAP}}(y) = \operatorname*{Argmax}_x p(x|y) = \operatorname*{Argmin}_x \frac{1}{2\sigma^2}\|y - x\|^2 - \log p(x),$$

with the convention that $\log 0 = -\infty$.

MAP Denoiser as a Proximal Operator

Let us fix a prior p on $\mathbf{R}^d$.

If the prior is a probability density function $p : \mathbf{R}^d \to \mathbf{R}_+$, the maximum a-posteriori (MAP) denoiser can be written as

$$D_\sigma^{\mathrm{MAP}}(y) = \operatorname*{Argmax}_x p(x|y) = \operatorname*{Argmin}_x \frac{1}{2\sigma^2}\|y-x\|^2 - \log p(x),$$

with the convention that $\log 0 = -\infty$.
Therefore,

$$D_\sigma^{\mathrm{MAP}}(y) = \mathrm{Prox}_{-\sigma^2 \log p}(y).$$

Remarks:

- If $-\log p$ is a convex proper l.s.c. function, then D^{MAP} is uniquely defined.
- We have seen examples where D^{MAP} can be computed explicitly.
- When p has no density, we may adapt the above formula (e.g. when p has finite support).

MMSE Denoiser as a Gradient-Step

Theorem (Tweedie-Miyasawa identity)

Let X, Z be two independent random vectors with $X \in L^1$ and $Z \sim \mathcal{N}(0, \mathrm{Id})$.
For $\sigma > 0$, let $Y = X + \sigma Z$, and denote by p_Y the density of Y.
Then

$$\mathbb{E}[X|Y] = Y + \sigma^2 \nabla \log p_Y(Y), \quad \textit{or equivalently,} \quad \mathbb{E}[Z|Y] = -\sigma \nabla \log p_Y(Y).$$

MMSE Denoiser as a Gradient-Step

Theorem (Tweedie-Miyasawa identity)

Let X, Z be two independent random vectors with $X \in L^1$ and $Z \sim \mathcal{N}(0, \mathrm{Id})$.
For $\sigma > 0$, let $Y = X + \sigma Z$, and denote by p_Y the density of Y.
Then

$$\mathbb{E}[X|Y] = Y + \sigma^2 \nabla \log p_Y(Y), \quad \textit{or equivalently,} \quad \mathbb{E}[Z|Y] = -\sigma \nabla \log p_Y(Y).$$

Interpretation:

The Gaussian MMSE denoiser is a gradient ascent step on the log-density of noisy data.

MMSE Denoiser as a Gradient-Step

Theorem (Tweedie-Miyasawa identity)

Let X, Z be two independent random vectors with $X \in L^1$ and $Z \sim \mathcal{N}(0, \mathrm{Id})$.
For $\sigma > 0$, let $Y = X + \sigma Z$, and denote by p_Y the density of Y.
Then

$$\mathbb{E}[X|Y] = Y + \sigma^2 \nabla \log p_Y(Y), \quad \textit{or equivalently,} \quad \mathbb{E}[Z|Y] = -\sigma \nabla \log p_Y(Y).$$

Interpretation:

The Gaussian MMSE denoiser is a gradient ascent step on the log-density of noisy data.

Recall that the density p_σ of $Y = X + \sigma Z$ is the convolution of p with a Gaussian:

$$p_\sigma(y) = \int_{\mathbf{R}^d} p(x) g_\sigma(y - x) dx \quad \text{where} \quad g_\sigma(x) = (2\pi\sigma^2)^{-d/2} e^{-\frac{\|x\|^2}{2\sigma^2}}.$$

Therefore,

$$D_\sigma^{\mathrm{MMSE}}(y) = y + \sigma^2 \nabla \log p_\sigma(y).$$

In particular D_σ^{MMSE} is smooth on $\mathbf{R}^d$.

One Property of the MMSE Denoiser

Theorem ([Gribonval, 2011])

The function $p_\sigma = p * g_\sigma$ is such that $\frac{\|\cdot\|^2}{2} + \sigma^2 \log p_\sigma$ is strictly convex.

Remark: The proof is based on the study of $\nabla^2 \log p_\sigma$ (i.e. second-order Tweedie identity).

Consequences:

- The MMSE denoiser is the gradient of a strictly convex function.
- $D_\sigma : \mathbf{R}^d \to \mathrm{Im}(D_\sigma)$ is a bijection.

Empirical MMSE denoiser

Assume that the prior p is the empirical distribution associated to n clean images:

$$p = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}.$$

Then the MMSE denoiser can be written with a closed-form formula:

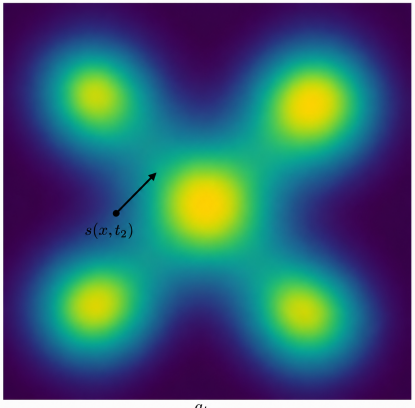
$$D_\sigma^{\text{MMSE}}(y) = \frac{\sum_{i=1}^{n} x_i \; g_\sigma(y - x_i)}{\sum_{i=1}^{n} g_\sigma(y - x_i)} = \frac{\sum_{i=1}^{n} x_i \; e^{-\frac{\|y-x_i\|^2}{\sigma^2}}}{\sum_{i=1}^{n} e^{-\frac{\|y-x_i\|^2}{\sigma^2}}}.$$

Remark: This can be seen as a smoothed nearest neighbor assignment!
Actually, it is the MAP denoiser which directly performs nearest neighbor projection:

$$D_\sigma^{\text{MAP}}(y) = \operatorname*{Argmax}_{x_i} p_\sigma(x_i|y) = \operatorname*{Argmax}_{x_i} p_\sigma(y|x_i) = \operatorname*{Argmin}_{x_i} \|y - x_i\|^2.$$

Illustration of the empirical MMSE denoiser

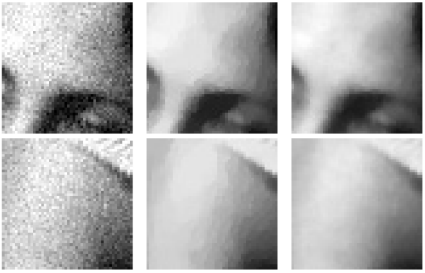
We consider here a simple 2D prior p: uniform on five points in $\mathbf{R}^2$.



(made with ChatGPT... What do you think of the arrow? :-))

MAP vs MMSE

- Choosing between MAP and MMSE (or sth else!) is in general not an obvious choice.
- If we optimize L^2-based scores (e.g. PSNR), MMSE should be favored.
- MAP denoising can sometimes be computed with efficient optimization algorithms.
- MMSE denoising, since it is based on averaging, tends to smooth out some artifacts. For example, with a TV prior, the MMSE denoiser has no staircasing effect:



Left: Noisy image. Middle: TV-MAP denoising. Right: TV-MMSE denoising.
Fig. 4 from [Louchet and Moisan, 2013].

Plan

Bayesian Imaging
- Bayesian Image Restoration
- Bayesian Image Denoising
- Metrics for Inverse Problems

Properties of MAP/MMSE Denoisers

Denoising with Deep Neural Networks

Denoisers for Restoration and Generation
- Plug-and-Play Image Restoration
- Diffusion Generative Models
- Diffusion Posterior Sampling

Denoising Neural Networks

- Nowadays, many denoising algorithms are based on a neural network.
- A neural network is a composition of several parametric functions (layers).
  The parameters of all layers minimize a certain *cost function* related to some task.

Denoising Neural Networks

- Nowadays, many denoising algorithms are based on a neural network.
- A neural network is a composition of several parametric functions (layers).
  The parameters of all layers minimize a certain *cost function* related to some task.
- One can train a neural network denoiser $D_\sigma : \mathbf{R}^d \to \mathbf{R}^d$ by minimizing the MSE loss:

$$\underset{\text{Param}(D_\sigma)}{\text{Argmin}}\ \mathbb{E}\Big[\|D_\sigma(X+\xi) - X\|^2\Big]$$

  where $X \sim p_X$ and $\xi \sim \mathcal{N}(0, \sigma^2 \text{Id})$ are independent, p_X is a data distribution of clean images.
  We can also take the expectation w.r.t. σ (chosen e.g. randomly in $[0, \frac{50}{255}]$).
- Optimization is performed by stochastic gradient descent.
  At each iteration, the loss function is computed only on a batch $(x_b)_{1 \leq b \leq B}$ of data points.
- For certain denoising architectures, the noise level σ is given as input to the network.

Remark: For training, several authors use a L^1 loss instead of the squared L^2 loss.

Neural Network architecture

Neural networks can have various types of layers:

- Fully connected (FC) layers
- Upsampling (interpolation) or Subsampling (max/average pooling) layers
- Convolution/Transposed convolution (with stride)
- Activation functions: RELU, leakyRELU, sigmoid, tanh, etc
- BatchNorm
- Skip connections
- Attention layers
- ...

Convolutional Neural Networks (CNN) are networks that are mainly built on convolutional layers. (CNN were shown to be effective for image recognition in [Simonyan and Zisserman, 2015].)

NB: When speaking of one layer, people often concatenates one connected operator and the following activation function.

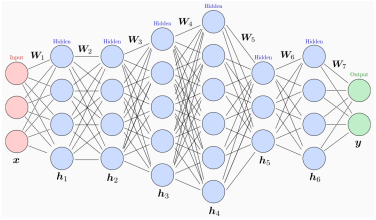
Fully Connected Layers

A fully connected layer $L : \mathbf{R}^C \to \mathbf{R}^{C'}$ is defined by

$$L(x) = \varphi(Wx + b)$$

where

- $W \in \mathbf{R}^{C' \times C}$
- $b \in \mathbf{R}^{C'}$ (bias vector)
- φ is the activation (1D function applied componentwise)



A multilayer perceptron (MLP): network made of 7 FC layers (Credits: C. Deledalle)

Convolution

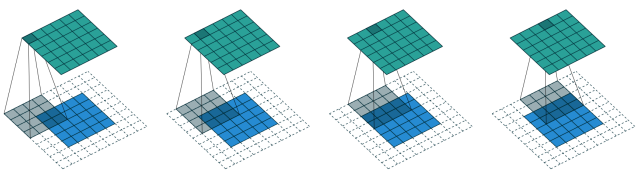
Let $u : \Omega \to \mathbf{R}^C$ be defined on $\Omega = [0 : H-1] \times [0 : W-1]$.

Let $w : \omega \to \mathbf{R}^{C' \times C}$ be defined on a small $\omega \subset \mathbb{Z}^2$. (Often, $\omega = [-k, k]^2$)

Definition

The convolution $w * u$ of the image u with kernel w is defined by

$$w * u(x) = \sum_{y \in \omega} w(y) u(x-y) = \sum_{z \in -\omega} \tilde{w}(z) u(x+z) \quad \text{where} \quad \tilde{w}(z) = w(-z).$$



NB: There are several possible border conditions (restriction, constant padding, periodic, ...)

Convolution and Transposed convolution

Notice that

- The transpose of a convolution with a $k \times k$ kernel is a convolution with a $k \times k$ kernel
- The transpose of a border crop is zero-padding the borders.
- The transpose of a crude subsampling is zero-inserting.

Strided convolutions:

- A "convolution with stride" is a convolution followed by subsampling.
- Called `Conv2d` in PyTorch

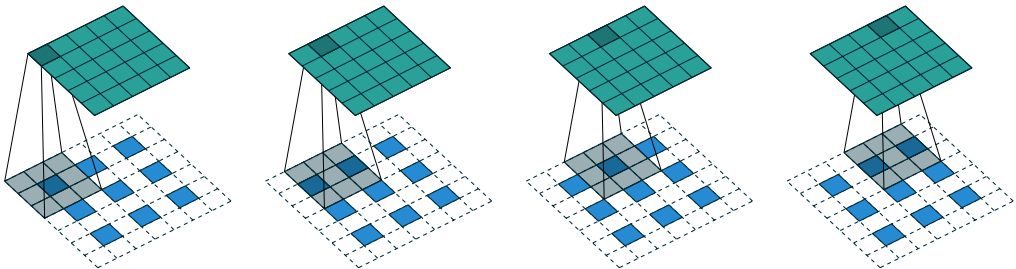
Fractionally strided convolutions:

- This is the transpose operator of convolution with stride.
- Called `ConvTranspose2d` in PyTorch

Useful webpage to compute the size of output after a Conv2D layer:

`https://madebyollin.github.io/convnet-calculator/`

One Example from [Dumoulin and Visin, 2016]



"The transpose of convolving a 3×3 kernel over a 5×5 input padded with a 1×1 border of zeros using 2×2 strides (i.e., $i = 5$, $k = 3$, $s = 2$ and $p = 1$). It is equivalent to convolving a 3×3 kernel over a 3×3 input (with 1 zero inserted between inputs) padded with a 1×1 border of zeros using unit strides (i.e., $i' = 3$, $\tilde{i}' = 5$, $k' = k$, $s' = 1$ and $p' = 1$)."

Batch Normalization

In order to make training more stable, networks often include "BatchNorm" layers. Principles:

- Consider a batch $(x_n)_{1 \leq n \leq N}$ of N responses to a neural layer with C features.
- For each n, $x_{n,i} \in \mathbf{R}^{W \times H}$ is the i-th feature map of the n-th image.
- Batch normalization consists in computing for any n, i

$$y_{n,i} = \gamma_i z_{n,i} + \beta_i \quad \text{with} \quad z_{n,i} = \frac{x_{n,i} - m_i}{\sqrt{\sigma_i^2 + \varepsilon}}$$

  where m_i, σ_i are the mean and std of the gathered feature maps $(x_{n,i})_{1 \leq n \leq N}$.
  (In other words, m_i, σ_i contains averages over N and spatial dimensions H, W.)
- γ_i, β_i are trainable parameters.
- Implemented in `BatchNorm2d` in PyTorch.

At inference: normalization is done with $m_i, \sigma_i, \gamma_i, \beta_i$ learned during training.
Switch to inference mode with `model.eval()`.

Different Kinds of Normalization

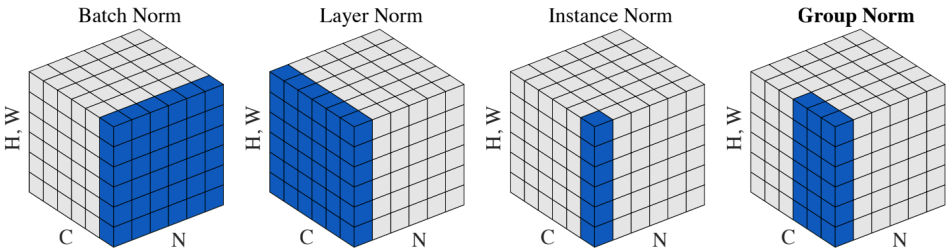


Diagram from [Wu and He, 2018]

- H, W: spatial dimensions
- C: channel dimension
- N: batch dimension

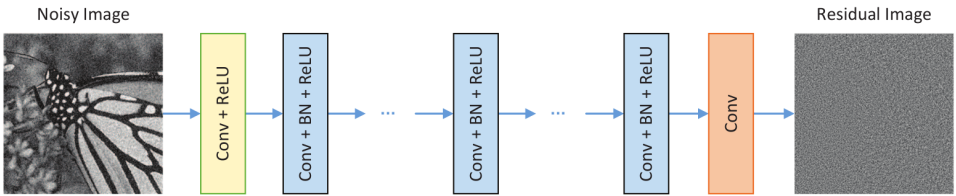
(See the formula for InstanceNorm in [Ulyanov et al., 2017])

A non-exhaustive list of denoising neural networks

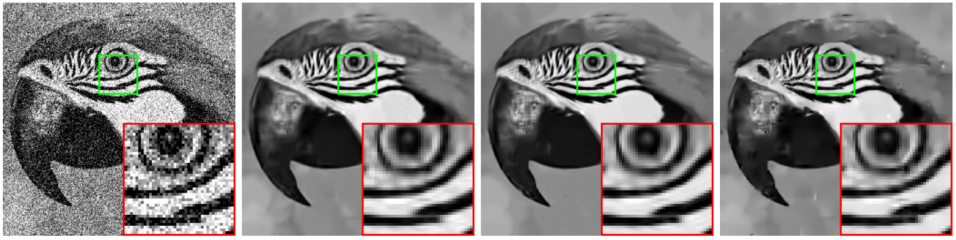
- [Jain and Seung, 2008]: natural image denoising with CNN
- [Burger et al., 2012]: a MLP can compete with BM3D on image denoising. Patches are denoised by a MLP, and then averaged on the overlap.
- RBDN [Santhanam et al., 2017] (CNN with recursive branches)
- DnCNN [Zhang et al., 2017] (blind CNN)
- FFDNet [Zhang et al., 2018a] (non-blind CNN)
- DRUNet [Zhang et al., 2021] (non-blind convolutional UNet)
- ...

DnCNN [Zhang et al., 2017]

- DnCNN is a deep convolutional neural network for denoising [Zhang et al., 2017].
- It is based on residual learning: $D(y) = y - R(y)$ where R is the network.
  Replace the loss $\mathbb{E}[\|D(X+\xi) - X\|^2]$ by the residual loss $\mathbb{E}[\|R(Y) - \xi\|^2]$.
- R has 20 layers of hidden dimension 64 (3×3 convolutions, BatchNorm, ReLU)
- It is trained on noise levels $\sigma \in [0, 50]$ and can be applied blindly (without σ).



DnCNN Results [Zhang et al., 2017]

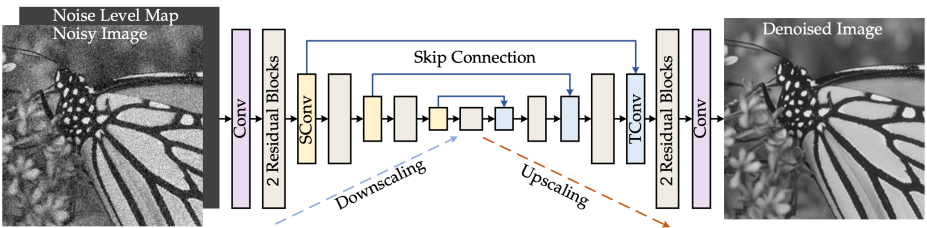


(a) Noisy / 15.00dB (b) BM3D / 25.90dB (c) WNNM / 26.14dB (d) EPLL / 25.95dB

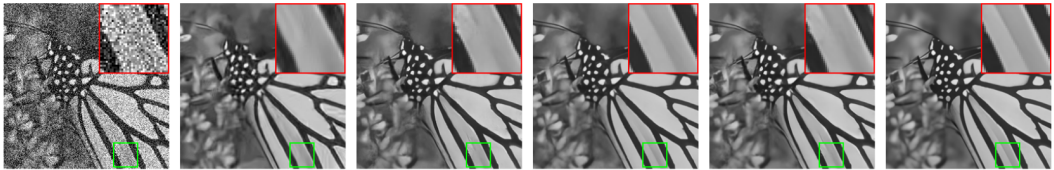
(e) MLP / 26.12dB (f) TNRD / 26.16dB (g) DnCNN-S / 26.48dB (h) DnCNN-B / 26.48dB

DRUNet [Zhang et al., 2021]

- DRUNet is a deep convolutional neural network for denoising [Zhang et al., 2021].
- It is a UNet that includes residual blocks, convolutions (bias-free!), and skip connections.
- The UNet has 4 "scales" of dimensions $64, 128, 256, 512$.
  This forces some multi-resolution processing.
- It is trained on noise levels $\sigma \in [0, 50]$ and takes a noise level map as input.



DRUNet Results [Zhang et al., 2017]



(a) Noisy (14.78dB) (b) BM3D (25.82dB) (c) DnCNN (26.83dB) (d) RNAN (27.18dB) (e) FFDNet (26.92dB) (f) DRUNet (27.31dB)

Fig. 2. Grayscale image denoising results of different methods on image "*Monarch*" from Set12 dataset with noise level 50.



(a) Noisy (14.99dB) (b) BM3D (28.36dB) (c) DnCNN (28.68dB) (d) FFDNet (28.75dB) (e) IRCNN (28.69dB) (f) DRUNet (29.28dB)

Fig. 3. Color image denoising results of different methods on image "*163085*" from CBSD68 dataset with noise level 50.

DRUNet Results [Zhang et al., 2017]

TABLE 2
Average PSNR(dB) results of different methods for noise levels 15, 25 and 50 on CBSD68 [3], [44], [49], Kodak24 and McMaster datasets. The best and second best results are highlighted in red and blue colors, respectively.

Datasets	Noise Level	CBM3D	DnCNN	IRCNN	FFDNet	DRUNet
CBSD68	15	33.52	33.90	33.86	33.87	34.30
	25	30.71	31.24	31.16	31.21	31.69
	50	27.38	27.95	27.86	27.96	28.51
Kodak24	15	34.28	34.60	34.69	34.63	35.31
	25	32.15	32.14	32.18	32.13	32.89
	50	28.46	28.95	28.93	28.98	29.86
McMaster	15	34.06	33.45	34.58	34.66	35.40
	25	31.66	31.52	32.18	32.35	33.14
	50	28.51	28.62	28.91	29.18	30.08

Plan

Bayesian Imaging
- Bayesian Image Restoration
- Bayesian Image Denoising
- Metrics for Inverse Problems

Properties of MAP/MMSE Denoisers

Denoising with Deep Neural Networks

Denoisers for Restoration and Generation
- Plug-and-Play Image Restoration
- Diffusion Generative Models
- Diffusion Posterior Sampling

Plug-and-Play Imaging

Let us consider the inverse problem

$$y = Ax_{GT} + w\,, \quad \text{where} \quad w \sim \mathcal{N}(0, \nu^2 \mathrm{Id}).$$

- PnP algorithms are based on proximal splitting schemes to minimize

$$f + g \quad \text{where} \quad f(x) = -\log p(y|x) \quad \text{and} \quad g(x) = -\log p(x).$$

- For example, PGD alternates implicit and explicit descents on each term.

Plug-and-Play Imaging

Let us consider the inverse problem

$$y = Ax_{GT} + w\,, \quad \text{where} \quad w \sim \mathcal{N}(0, \nu^2 \mathrm{Id}).$$

- PnP algorithms are based on proximal splitting schemes to minimize

$$f + g \quad \text{where} \quad f(x) = -\log p(y|x) \quad \text{and} \quad g(x) = -\log p(x).$$

- For example, PGD alternates implicit and explicit descents on each term.
- Besides, we have seen that
  · MAP denoising can be seen as $\mathrm{Prox}_{-\sigma^2 \log p}$
  · MMSE denoising can be seen as $\mathrm{Id} + \sigma^2 \nabla \log p_\sigma$

Plug-and-Play Imaging

Let us consider the inverse problem

$$y = Ax_{GT} + w\,, \quad \text{where} \quad w \sim \mathcal{N}(0, \nu^2 \mathrm{Id}).$$

- PnP algorithms are based on proximal splitting schemes to minimize

$$f + g \quad \text{where} \quad f(x) = -\log p(y|x) \quad \text{and} \quad g(x) = -\log p(x).$$

- For example, PGD alternates implicit and explicit descents on each term.
- Besides, we have seen that
  · MAP denoising can be seen as $\mathrm{Prox}_{-\sigma^2 \log p}$
  · MMSE denoising can be seen as $\mathrm{Id} + \sigma^2 \nabla \log p_\sigma$
- Therefore, if we use the ideal MAP/MMSE denoiser associated to some prior, we may interpret PnP-PGD as a true minimization algorithm [Xu et al., 2020].

✗ But in practice, a denoising network does not realize exactly the ideal MAP or MMSE. Therefore, convergence is not guaranteed at first sight.

PnP with MAP denoisers

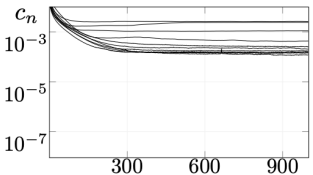
- Recall the PnP-PGD algorithm:

$$x_{k+1} = D_\sigma \circ (\mathrm{Id} - \tau \nabla f)(x_k) \qquad \text{(PnP-PGD)}$$

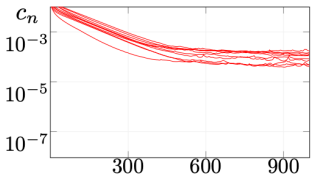
- We wish D_σ to be close to a proximal operator, i.e. related to a MAP denoising problem.
- One can show that if g is convex proper l.s.c., then $T = \mathrm{Prox}_g$ is "firmly non-expansive", which is equivalent to: $2T - \mathrm{Id}$ is 1-Lipschitz.
- Thus the idea to used "constrained denoising networks" to improve PnP convergence.

PnP with Constrained Denoiser [Pesquet et al., 2021]

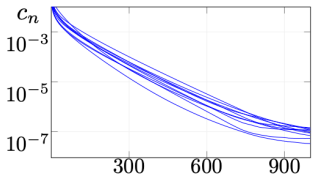
- Terris et al. proposed to train a DnCNN with a Lip($2D_\sigma - \mathrm{Id}$) penalty.
- The benefit is that the corresponding algorithm has better convergence properties.
- This can be checked by plotting the evolution of the residuals $r_k = \frac{\|x_k - x_{k-1}\|}{\|x_0\|}$, as below.
- Here is one deblurring example based on PnP-PGD.



(a) BM3D

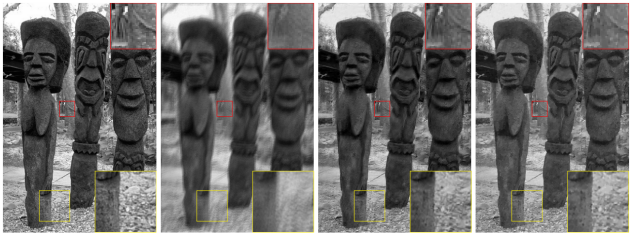


(b) RealSN



(c) Proposed

PnP with Constrained Denoiser [Pesquet et al., 2021]



(a) Groundtruth | (b) Observation (20.48, 0.387) | (c) $\text{prox}_{\mu\|\Psi^\dagger\cdot\|_1}$ (26.13, 0.775) | (d) $\text{prox}_{\mu\|\cdot\|_{\text{TV}}}$ (26.57, 0.787)

(e) BM3D (26.09, 0.732) | (f) RealSN (24.68, 0.726) | (g) DnCNN (26.12, 0.643) | (h) Proposed (**27.09**, **0.789**)

Gradient-Step Plug-and-Play

- Another way to get convergent PnP is to do PGD in the other way around:

$$x_{k+1} = \text{Prox}_{\tau f} \circ (\text{Id} - \tau\lambda\nabla g)(x_k).$$

- For that, it is possible to build a **"Gradient Step" (GS)** denoiser:

$$D_\sigma = \text{Id} - \nabla g_\sigma \quad \text{with} \quad g_\sigma(x) = \frac{1}{2}\|x - N_\sigma(x)\|^2$$

where $N_\sigma : \mathbb{R}^n \to \mathbb{R}^n$ is a $\mathcal{C}^1$ neural network.

Gradient-Step Plug-and-Play

- Another way to get convergent PnP is to do PGD in the other way around:

$$x_{k+1} = \text{Prox}_{\tau f} \circ (\text{Id} - \tau\lambda\nabla g)(x_k).$$

- For that, it is possible to build a **"Gradient Step" (GS)** denoiser:

$$D_\sigma = \text{Id} - \nabla g_\sigma \quad \text{with} \quad g_\sigma(x) = \frac{1}{2}\|x - N_\sigma(x)\|^2$$

  where $N_\sigma : \mathbb{R}^n \to \mathbb{R}^n$ is a $\mathcal{C}^1$ neural network.
- This has been proposed simultaneously in [Hurault et al., 2021], [Cohen et al., 2021].

Gradient-Step Plug-and-Play

- Another way to get convergent PnP is to do PGD in the other way around:

$$x_{k+1} = \text{Prox}_{\tau f} \circ (\text{Id} - \tau\lambda\nabla g)(x_k).$$

- For that, it is possible to build a **"Gradient Step" (GS)** denoiser:

$$D_\sigma = \text{Id} - \nabla g_\sigma \quad \text{with} \quad g_\sigma(x) = \frac{1}{2}\|x - N_\sigma(x)\|^2$$

  where $N_\sigma : \mathbb{R}^n \to \mathbb{R}^n$ is a $\mathcal{C}^1$ neural network.
- This has been proposed simultaneously in [Hurault et al., 2021], [Cohen et al., 2021].
- This led to the "Gradient-Step PnP" algorithm

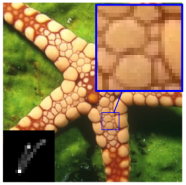
$$x_{k+1} = \text{Prox}_{\tau f} \circ ((1 - \tau\lambda)\text{Id} + \tau\lambda D_\sigma) \qquad \text{(GSPnP)}$$

  which combined high-performance restoration and realistic convergence guarantees.

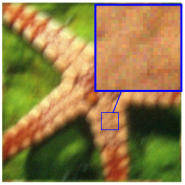
Actually, we also designed some "proximal denoiser" [Hurault et al., 2022] that we won't have time to discuss today. :-)

Deblurring Example [Hurault et al., 2022]

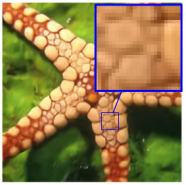
Deblurring with motion kernel and Gaussian noise std $\nu = 0.03$



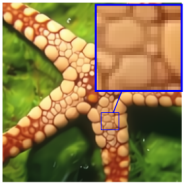
(a) Clean



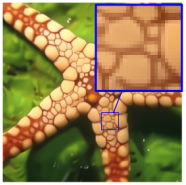
(b) Observed



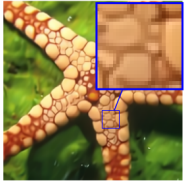
(c) IRCNN (28.66dB)



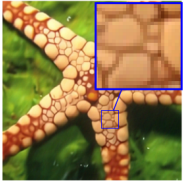
(d) DPIR (29.76dB)



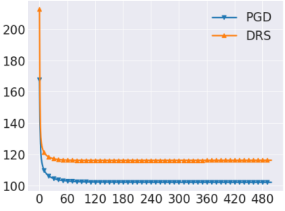
(e) GSPnP-HQS (29.90dB)



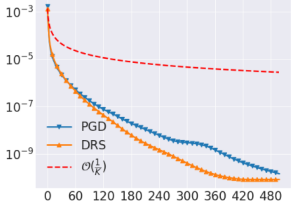
(f) Prox-PnP-PGD (29.41dB)



(g) Prox-PnP-DRS (29.65dB)



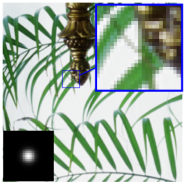
(h) $F_{\lambda,\sigma}(x_k)$



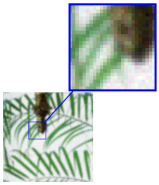
(i) $\min_{i \leq k} ||x_{i+1} - x_i||^2$

Super-resolution Example [Hurault et al., 2022]

Super-resolution with scale 2, Gaussian blur kernel and Gaussian noise std $\nu = 0.01$



(a) Clean



(b) Observed



(c) IRCNN (22.82dB)



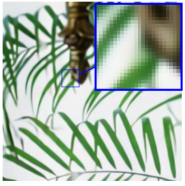
(d) DPIR (23.97dB)



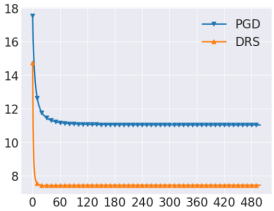
(e) GSPnP-HQS (24.81dB)



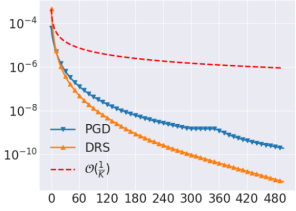
(f) Prox-PnP-PGD (23.96dB)



(g) Prox-PnP-DRS (24.36dB)



(h) $F_{\lambda,\sigma(x_k)}$



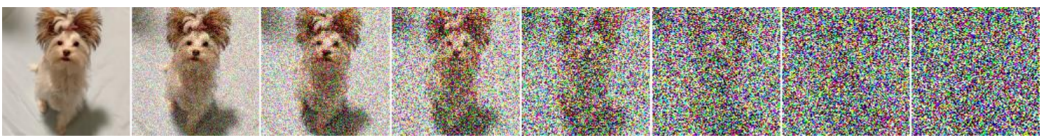
(i) $\min_{i \leq k} ||x_{i+1} - x_i||^2$

Remarks on PnP Algorithms

- These algorithms are easy to implement, provided that we have access to a good denoiser.
- The algorithm's choice is driven by the possibility to compute ∇f or $\text{Prox}_{\tau f}$.
- Some of the algorithms have better convergence properties.
- Also, the parameters should be set carefully (for convergence AND performance):
  σ: strength of the denoiser
  λ: regularization weight
  τ: step size
- Notice that σ may not be equal to the input's noise level ν!

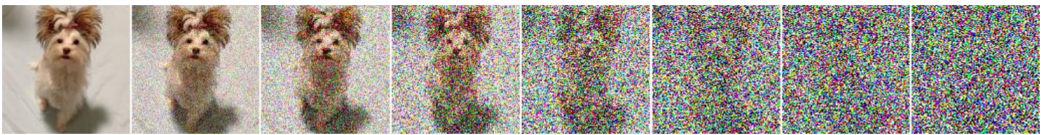
Generative Modeling with Diffusion

- Goal: design a generative model, i.e. a samplable probability distribution $p_\theta(x_0)$, that fits a given some data distribution $q_0(x_0)$ (e.g. an empirical distribution on N data points).
- Starting remark: if $X_0 \sim q_0$ and $Z \sim \mathcal{N}(0, \mathrm{Id})$ are independent, then $X_t = X_0 + \sigma_t Z$ looks noisier and noisier if $\sigma_t \to +\infty$.
  Also, the distribution q_t of X_t "resembles more and more" to a large Gaussian $\mathcal{N}(0, \sigma_t\ \mathrm{Id})$.



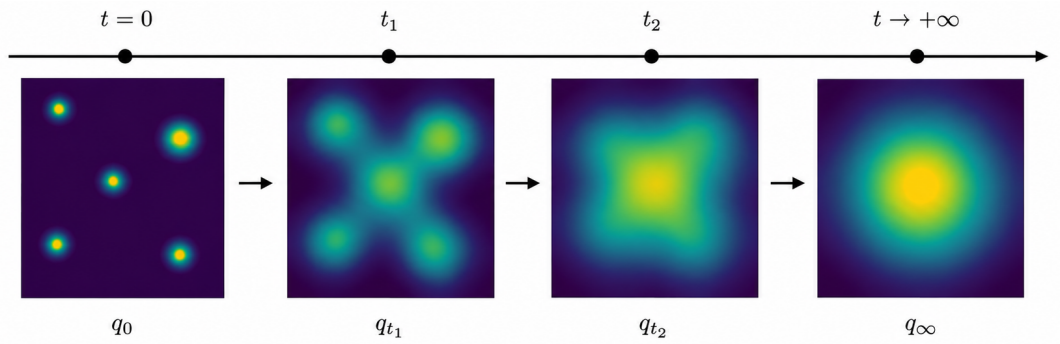
Generative Modeling with Diffusion

- Goal: design a generative model, i.e. a samplable probability distribution $p_\theta(x_0)$, that fits a given some data distribution $q_0(x_0)$ (e.g. an empirical distribution on N data points).
- Starting remark: if $X_0 \sim q_0$ and $Z \sim \mathcal{N}(0, \mathrm{Id})$ are independent, then $X_t = X_0 + \sigma_t Z$ looks noisier and noisier if $\sigma_t \to +\infty$.
  Also, the distribution q_t of X_t "resembles more and more" to a large Gaussian $\mathcal{N}(0, \sigma_t\ \mathrm{Id})$.



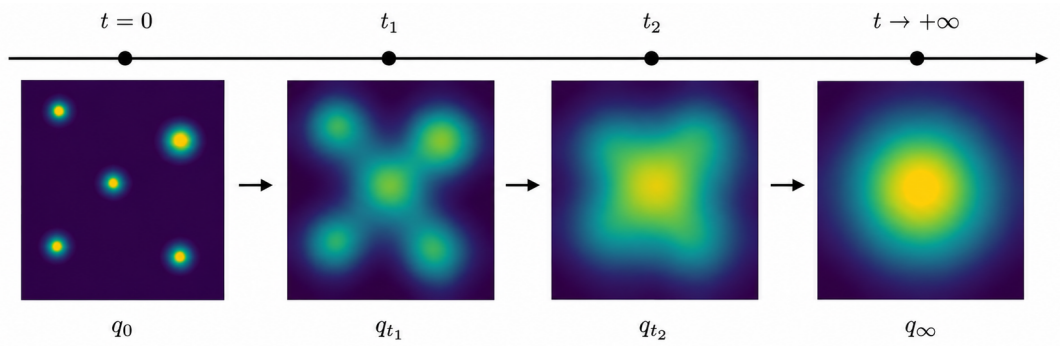
- Thus we can try to sample $\mathbf{X}_T \sim \mathcal{N}(0, T\mathrm{Id})$ for a large $T > 0$, and reverse the noising process to get back $\mathbf{X}_0$ with law $\approx q_0$.

Illustration on the Distribution q_t of X_t



(made with ChatGPT)

Illustration on the Distribution q_t of X_t



(made with ChatGPT)

Recall that the p.d.f. of $Y = X + \sigma Z$ is $p_\sigma = p_X * g_\sigma$.

When adding noise to the samples, we blur the distribution!

Forward and Backward Diffusion

- Backward Diffusion can be seen as a randomized gradient ascent on $\log p_\sigma$ with decreasing σ.
- Convergence guarantees [De Bortoli, 2022] are based on a clever control of the discretization of some stochastic differential equation (Langevin steps).
- Practical algorithms like DDPM [Ho et al., 2020] are based on discrete-time processes.
  For $(\alpha_t), (\beta_t)$ sequences of parameters, we draw $X_0 \sim q_0$ and set recursively

$$X_t = \sqrt{\alpha_t} X_{t-1} + \sqrt{\beta_t} Z_t, \quad \text{where} \quad (Z_t)_{t \geq 1} \text{ are i.i.d. } \mathcal{N}(0, \mathrm{Id}).$$

  or equivalently

$$X_t = \sqrt{\overline{\alpha}_t} X_0 + \sqrt{\overline{\beta}_t} E_t \quad \sim \quad E_t \sim \mathcal{N}(0, \mathrm{Id})$$

  for some $\overline{\alpha}_t, \overline{\beta}_t$ which can be computed explicitly.
- Parameters are adjusted so that $\overline{\alpha}_t \to 0, \overline{\beta}_t \to 1$ when $t \to T$.
- Backward diffusion then tries to emulate the backward process $X_T \to X_{T-1} \to \ldots \to X_0$.

Algorithm for DDPM Backward Sampling

Parameters:

- noise schedule defined by sequences $(\alpha_t), (\beta_t)$.
- score neural network $\varepsilon_\theta(\mathbf{X}_t, t)$: pretrained to estimate the "normalized noise" in $\mathbf{X}_t$.

The sequences $(\overline{\alpha}_t), (\overline{\beta}_t)$ are precomputed.

One can then use the following algorithm to get one backward DDPM sample.

- $\mathbf{X}_T \sim \mathcal{N}(0, (\overline{\alpha}_T + \overline{\beta}_T)\mathrm{Id})$
- For all $t = T, \dots, 1$,

$$\mathbf{X}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{X}_t - \frac{\beta_t}{\sqrt{\overline{\beta}_t}} \varepsilon_\theta(\mathbf{X}_t, t) \right) + \sqrt{\tilde{\beta}_t} \mathbf{Z}_t \quad \text{with } \mathbf{Z}_t \sim \mathcal{N}(0, \mathrm{Id})$$

- Return the sample $\mathbf{X}_0$.

DDPM Algorithm

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

DDPM as iterated Noising-Denoising

[Leclaire et al., 2025]

- Actually, backward sampling can be interpreted quite simply.
- For example, the DDPM algorithm [Ho et al., 2020] can be written with a denoiser D_σ.

$$\mathbf{U}_{t-1} = \big((1-\rho_t)\mathrm{Id} + \rho_t D_{\sigma_t}\big)(\mathbf{U}_t) + \gamma_{t-1}\mathbf{Z}_{t-1}.$$

$\Rightarrow$ **DDPM backward process consists in alternating relaxed noising and denoising.**

DDPM as iterated Noising-Denoising

[Leclaire et al., 2025]

- Actually, backward sampling can be interpreted quite simply.
- For example, the DDPM algorithm [Ho et al., 2020] can be written with a denoiser D_σ.

$$\mathbf{U}_{t-1} = \big((1-\rho_t)\mathrm{Id} + \rho_t D_{\sigma_t}\big)(\mathbf{U}_t) + \gamma_{t-1}\mathbf{Z}_{t-1}.$$

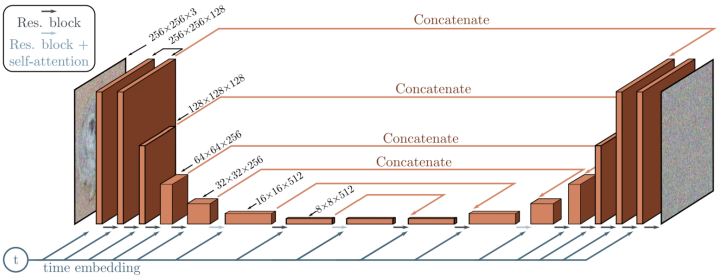
$\Rightarrow$ **DDPM backward process consists in alternating relaxed noising and denoising.**

The noise schedule is actually set to ensure that noise level stays coherent along the process. Indeed, one can check that the choice of $(\sigma_t), (\rho_t), (\gamma_t)$ in [Ho et al., 2020] satisfies

$$\sigma_{t-1}^2 = \quad (1-\rho_t)^2\sigma_t^2 \quad + \quad \gamma_{t-1}^2.$$

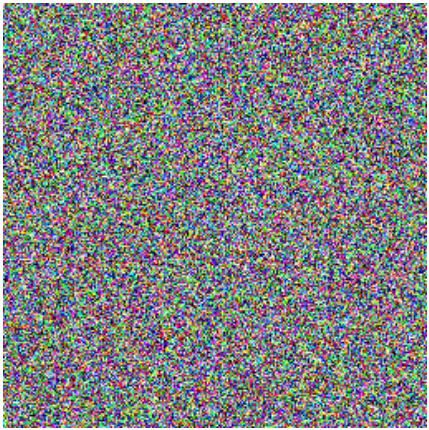
UNet for Diffusion [Dhariwal and Nichol, 2021]

- The network ε_θ used for the score approximation resembles denoising UNets.
- All layers depend on time, through a particular "time embedding".
- Some attention layers are used. (We won't explain that today.)
- But the range of noise level σ must be taken much larger than in classical image denoising.



Source: learnopencv.com

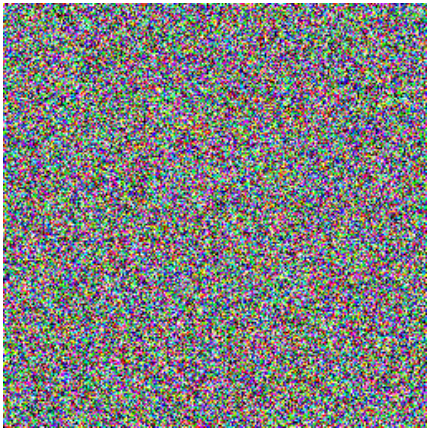
Example Backward Sampling with a DDPM trained on FFHQ



$\mathbf{X}_t$

$t = 999$

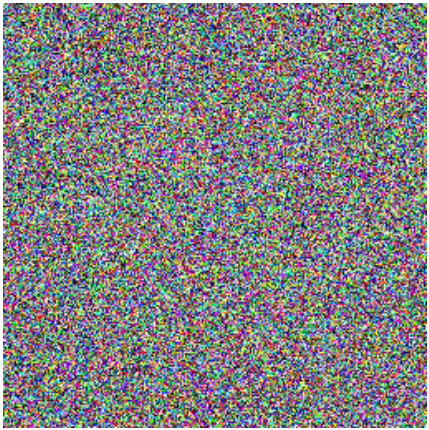
Example Backward Sampling with a DDPM trained on FFHQ



$\mathbf{X}_t$

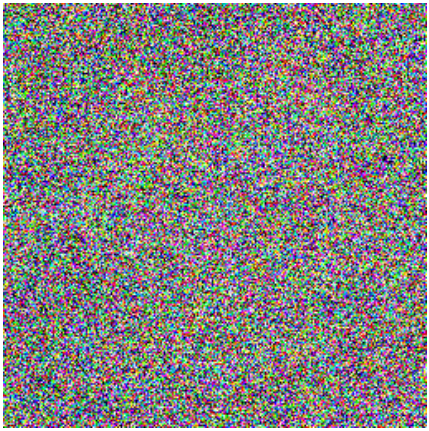
$t = 900$

Example Backward Sampling with a DDPM trained on FFHQ



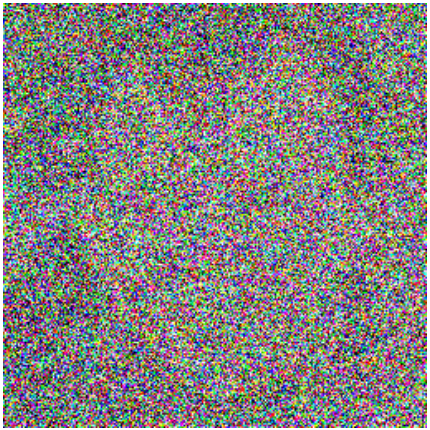
$\mathbf{X}_t$
$t = 800$

Example Backward Sampling with a DDPM trained on FFHQ



$\mathbf{x}_t$
$t = 700$

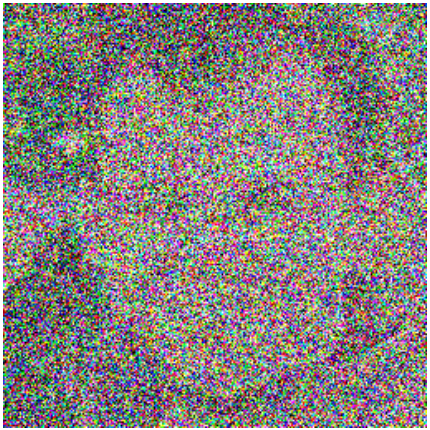
Example Backward Sampling with a DDPM trained on FFHQ



$\mathbf{X}_t$

$t = 600$

Example Backward Sampling with a DDPM trained on FFHQ



$\mathbf{X}_t$

$t = 500$

Example Backward Sampling with a DDPM trained on FFHQ



$\mathbf{X}_t$
$t = 400$

Example Backward Sampling with a DDPM trained on FFHQ



$\mathbf{X}_t$

$t = 300$

Example Backward Sampling with a DDPM trained on FFHQ



$\mathbf{X}_t$
$t = 200$

Example Backward Sampling with a DDPM trained on FFHQ



$\mathbf{x}_t$
$t = 100$

Example Backward Sampling with a DDPM trained on FFHQ



$\mathbf{x}_t$
$t = 0$

Diffusion Posterior Sampling (DPS)

[Chung et al., 2023]

- A Bayesian inverse problem may be solved with a prior that comes from a diffusion process.
- DPS consists in conditioning the diffusion backward process by the degraded observation y.
- We then use a conditional score $\nabla \log p_t(x_t|y)$.
- Bayes formula gives

$$\nabla \log p_t(x_t|y) = \nabla \log p_t(y|x_t) + \nabla \log p_t(x_t).$$

Diffusion Posterior Sampling (DPS)

[Chung et al., 2023]

- A Bayesian inverse problem may be solved with a prior that comes from a diffusion process.
- DPS consists in conditioning the diffusion backward process by the degraded observation y.
- We then use a conditional score $\nabla \log p_t(x_t|y)$.
- Bayes formula gives

$$\nabla \log p_t(x_t|y) = \nabla \log p_t(y|x_t) + \nabla \log p_t(x_t).$$

- The second term is the usual score function.
- The first term is, unfortunately, intractable.

Diffusion Posterior Sampling (DPS)

[Chung et al., 2023]

- A Bayesian inverse problem may be solved with a prior that comes from a diffusion process.
- DPS consists in conditioning the diffusion backward process by the degraded observation y.
- We then use a conditional score $\nabla \log p_t(x_t|y)$.
- Bayes formula gives

$$\nabla \log p_t(x_t|y) = \nabla \log p_t(y|x_t) + \nabla \log p_t(x_t).$$

- The second term is the usual score function.
- The first term is, unfortunately, intractable.
- Chung et al. propose to approximate $p_t(y|x_t) \approx p_t(y|\hat{x}_0(x_t, t))$, where $\hat{x}_0(x_t, t)$ is the estimation of x_0 obtained from x_t (with the learned network).
- Since the Gaussian likelihood is tractable: $p(y|x_0) \propto \exp\left(-\frac{\|y - Ax_0\|^2}{2\sigma^2}\right)$,
  we can approximate the conditional score as $\nabla \log p_t(y|x_t) = -\frac{1}{2\sigma^2} \nabla_{x_t} \|y - A\hat{x}_0(x_t, t)\|^2$.

DPS algorithm

[Chung et al., 2023]

Algorithm 1 DPS - Gaussian

Require: N, $\boldsymbol{y}$, $\{\zeta_i\}_{i=1}^N$, $\{\tilde{\sigma}_i\}_{i=1}^N$

$\boldsymbol{x}_N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

for $i = N-1$ **to** 0 **do**

  $\hat{\boldsymbol{s}} \leftarrow \boldsymbol{s}_\theta(\boldsymbol{x}_i, i)$

  $\hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\boldsymbol{x}_i + (1-\bar{\alpha}_i)\hat{\boldsymbol{s}})$

  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

  $\boldsymbol{x}'_{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\hat{\boldsymbol{x}}_0 + \tilde{\sigma}_i \boldsymbol{z}$

  $\boldsymbol{x}_{i-1} \leftarrow \boldsymbol{x}'_{i-1} - \zeta_i \nabla_{\boldsymbol{x}_i} \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{x}}_0)\|_2^2$

end for

return $\hat{\mathbf{x}}_0$

DPS inpainting results



x_c y $\hat{x}_0$ x_t

$t = 999$

DPS inpainting results



x_c y $\hat{x}_0$ x_t

$t = 900$

DPS inpainting results



x_c y $\hat{x}_0$ x_t

$t = 800$

DPS inpainting results



x_c y $\hat{x}_0$ x_t

$t = 700$

DPS inpainting results



x_c y $\hat{x}_0$ x_t

$t = 600$

DPS inpainting results



x_c y $\hat{x}_0$ x_t

$t = 500$

DPS inpainting results



x_c y $\hat{x}_0$ x_t

$t = 400$

DPS inpainting results



x_c y $\hat{x}_0$ x_t

$t = 300$

DPS inpainting results



x_c y $\hat{x}_0$ x_t

$t = 200$

DPS inpainting results



x_c y $\hat{x}_0$ x_t

$t = 100$

DPS inpainting results



x_c y $\hat{x}_0$ x_t

$t = 0$

Take-home Messages

- We have defined Bayesian concepts for image restoration/denoising.
- MAP/MMSE denoising are related to proximal/gradient-descent step.
- The choice of a prior (or regularization) is linked to the choice of a denoiser.
- Most of the state-of-the-art denoisers are neural networks.
- They are usually trained with an MMSE loss on a large dataset of clean images.
- Notice that many concepts from the pre-deep era actually underly the construction of denoising neural networks (multi-resolution, residual learning, patch processing, nearest neighbor...)
- PnP algorithms allow to use some denoiser to tackle more difficult inverse problems.
- We have several PnP algorithmic variants that may be carefully used to get convergence.
- Diffusion generative models are also based on denoisers (with other training recipes).

THANK YOU FOR YOUR ATTENTION!

References I


Buades, A., Coll, B., and Morel, J.-M. (2005).
A non-local algorithm for image denoising.
In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE.


Burger, H. C., Schuler, C. J., and Harmeling, S. (2012).
Image denoising: Can plain neural networks compete with bm3d?
In *2012 IEEE conference on computer vision and pattern recognition*, pages 2392–2399. IEEE.


Chambolle, A. (2004).
An algorithm for total variation minimization and applications.
Journal of Mathematical imaging and vision, 20(1):89–97.


Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. (2023).
Diffusion posterior sampling for general noisy inverse problems.
In *The Eleventh International Conference on Learning Representations*.


Cohen, R., Blau, Y., Freedman, D., and Rivlin, E. (2021).
It has potential: Gradient-driven denoisers for convergent solutions to inverse problems.
Advances in Neural Information Processing Systems, 34.

References II


De Bortoli, V. (2022).
Convergence of denoising diffusion models under the manifold hypothesis.
Transactions on Machine Learning Research.


Dhariwal, P. and Nichol, A. (2021).
Diffusion models beat gans on image synthesis.
Advances in neural information processing systems, 34:8780–8794.


Dumoulin, V. and Visin, F. (2016).
A guide to convolution arithmetic for deep learning.
ArXiv e-prints.


Gatys, L. A., Ecker, A. S., and Bethge, M. (2015).
Texture synthesis using convolutional neural networks.
In *Advances in Neural Information Processing Systems*, pages 262–270.


Gatys, L. A., Ecker, A. S., and Bethge, M. (2016).
Image style transfer using convolutional neural networks.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423.

References III


Gribonval, R. (2011).
Should penalized least squares regression be interpreted as maximum a posteriori estimation?
IEEE Transactions on Signal Processing, 59(5):2405–2410.


Ho, J., Jain, A., and Abbeel, P. (2020).
Denoising diffusion probabilistic models.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851.


Hurault, S., Leclaire, A., and Papadakis, N. (2021).
Gradient step denoiser for convergent plug-and-play.
In *International Conference on Learning Representations*.


Hurault, S., Leclaire, A., and Papadakis, N. (2022).
Proximal denoiser for convergent plug-and-play optimization with nonconvex regularization.
In *International Conference on Machine Learning*, pages 9483–9505. PMLR.


Jain, V. and Seung, S. (2008).
Natural image denoising with convolutional networks.
Advances in neural information processing systems, 21.

References IV


Johnson, J., Alahi, A., and Fei-Fei, L. (2016).
Perceptual losses for real-time style transfer and super-resolution.
In *Computer Vision – ECCV 2016*, pages 694–711, Cham. Springer International Publishing.


Leclaire, A., Guez, E., and Galerne, B. (2025).
Backward Diffusion iterates Noising-Relaxed Denoising.
In *Actes du GRETSI*.


Louchet, C. and Moisan, L. (2013).
Posterior expectation of the total variation model: Properties and experiments.
SIAM Journal on Imaging Sciences, 6(4):2640–2684.


Mallat, S. (2009).
A Wavelet Tour of Signal Processing, The Sparse Way.
Academic Press, Elsevier, 3rd edition edition.


Pesquet, J.-C., Repetti, A., Terris, M., and Wiaux, Y. (2021).
Learning maximally monotone operators for image recovery.
SIAM Journal on Imaging Sciences, 14(3):1206–1237.

References V


Romano, Y., Elad, M., and Milanfar, P. (2017).
The little engine that could: Regularization by denoising (RED).
SIAM Journal on Imaging Sciences, 10(4):1804–1844.


Rudin, L. I., Osher, S., and Fatemi, E. (1992).
Nonlinear total variation based noise removal algorithms.
Physica D: nonlinear phenomena, 60(1-4):259–268.


Santhanam, V., Morariu, V. I., and Davis, L. S. (2017).
Generalized deep image to image regression.
In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5395–5405. IEEE.


Simonyan, K. and Zisserman, A. (2015).
Very deep convolutional networks for large-scale image recognition.
In Bengio, Y. and LeCun, Y., editors, *Proceedings of the International Conference on Learning Representations*.

References VI


Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2017).
Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6924–6932.


Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004).
Image quality assessment: from error visibility to structural similarity.
IEEE transactions on image processing, 13(4):600–612.


Wu, Y. and He, K. (2018).
Group normalization.
In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.


Xu, X., Sun, Y., Liu, J., Wohlberg, B., and Kamilov, U. S. (2020).
Provable convergence of plug-and-play priors with mmse denoisers.
arXiv preprint arXiv:2005.07685.

References VII


Yu, G., Sapiro, G., and Mallat, S. (2011).
Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity.
IEEE Transactions on Image Processing, 21(5):2481–2499.


Zhang, K., Li, Y., Zuo, W., Zhang, L., Van Gool, L., and Timofte, R. (2021).
Plug-and-play image restoration with deep denoiser prior.
IEEE Transactions on Pattern Analysis and Machine Intelligence.


Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. (2017).
Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising.
IEEE Transactions on Image Processing, 26(7):3142–3155.


Zhang, K., Zuo, W., and Zhang, L. (2018a).
Ffdnet: Toward a fast and flexible solution for cnn-based image denoising.
IEEE Transactions on Image Processing, 27(9):4608–4622.


Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018b).
The unreasonable effectiveness of deep features as a perceptual metric.
In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

References VIII


Zoran, D. and Weiss, Y. (2011).
From learning models of natural image patches to whole image restoration.
In *2011 International Conference on Computer Vision*, pages 479–486. IEEE.