Generative Models*

- Inference vs Generation
- Monte Carlo vs Generator Methods
- Gibbs Distributions
- Monte Carlo Markov Chains

*See this helpful blog for an overview: Yang Song, “Generative Modeling by Estimating Gradients of the Data Distribution," web blog post, May 5, 2021, https://yang-song.net/blog/2021/score .

Inference vs Generation

- Two primary goals in deep learning
  - Inference Model: Learn a function
  - Generative Model: Learn to sample from a distribution
- Key issues for generative models:
  - How can we learn the distribution from sample data?
  - How to generate random vectors with a desired distribution?

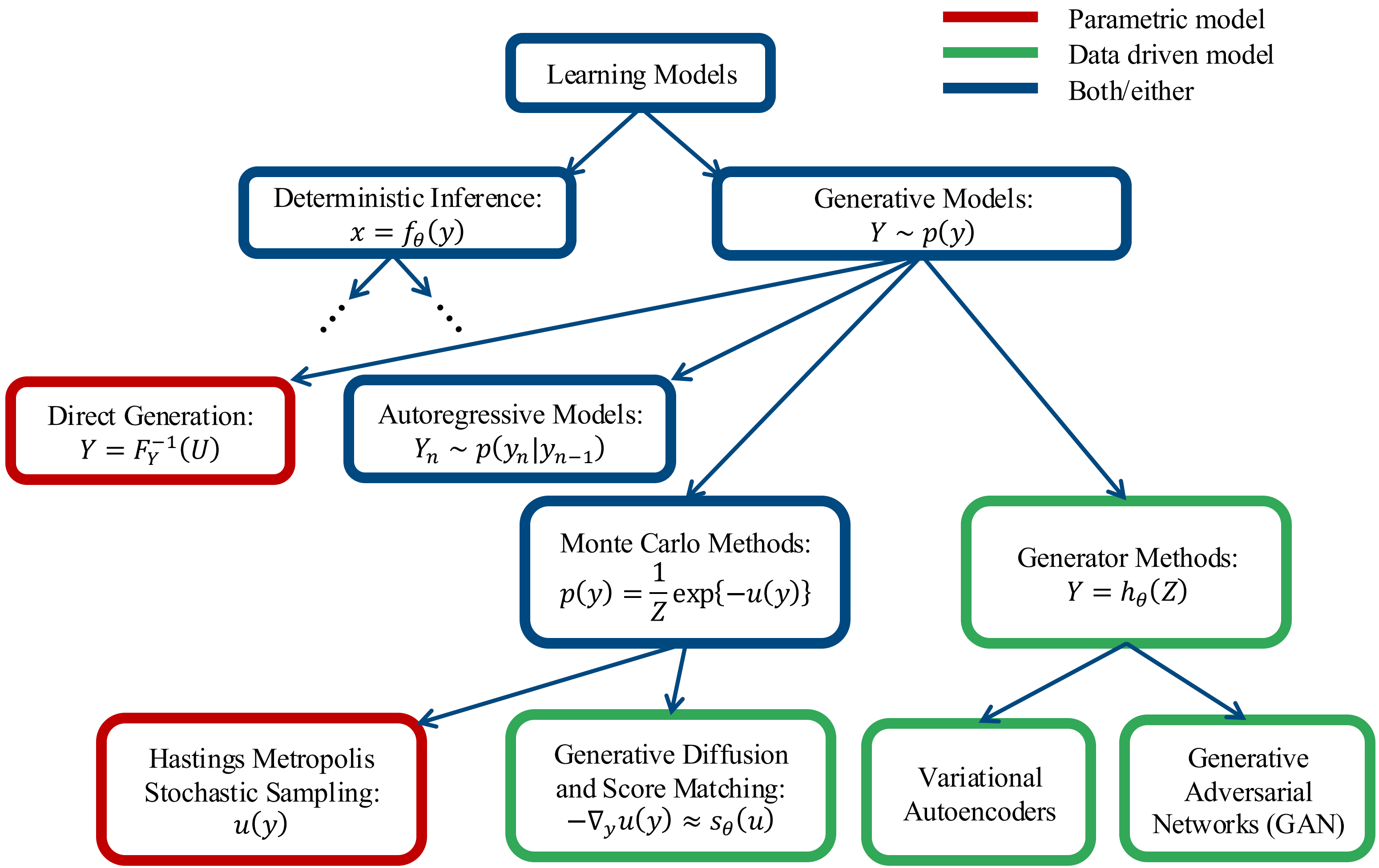
Inference Model:
$x = f_\theta(y)$

Goal: Predict unknown quantity.

Generative Model:
$Y \sim p_\theta(y)$

Goal: Generate random vectors.

Taxonomy of Learning Models



Gibbs Distribution

- Let $X \sim p(x)$ be a random object (i.e., image, video, speech).
- Typically, X is assumed to have a Gibbs distribution given by

$$p(x) = \frac{1}{z} \exp\{-u(x)\}$$

  - where $u(x)$ is the energy function, and z is the partition function given by $z = E[\exp\{-u(X)\}]$.

- Facts:
  - $u(x) = -\log p(x)$ always exists as long as $p(x) > 0$.
  - z is usually intractable to compute, but that's OK.
  - $u(x)$ increases $\Rightarrow p(x)$ decreases
  - $u(x)$ decreases $\Rightarrow p(x)$ increases

- From Thermodynamics:
  - Also known as Boltzmann distribution
  - The distribution of any system in thermodynamic equilibrium

Monte Carlo Markov Chains

- Metropolis algorithm:
  - Uses a symmetric proposal distribution $q(w|x) = q(x|w)$.

Initialize X_0; $n \leftarrow 0$

Repeat:

  Generate a proposal $W \sim q(w|X_n)$

  $\Delta E \leftarrow u(W) - u(X_n)$

  $p \leftarrow \min\{\exp\{-\Delta E\}, 1\}$

  With probability p:

    $X_{n+1} \leftarrow W$

  else

    $X_{n+1} \leftarrow X_n$

- Result:
  - X_n is a homogeneous Markov Chain.
  - X_n is a reversible, ergodic, MC with stationary distribution $p(x) = \frac{1}{z}\exp\{-u(x)\}$.
  - This is a way to sample from any Gibbs distribution!

Hastings Metropolis Algorithm

- Hastings Metropolis algorithm:
  - Uses proposal distribution $q(w|x)$.

Initialize X_0; $n \leftarrow 0$

Repeat:

Generate a proposal $W \sim q(w|X_n)$

$\Delta E \leftarrow u(W) - u(X_n)$

$p \leftarrow \min\left\{\frac{q(X_n|W)}{q(W|X_n)} \exp\{-\Delta E\}, 1\right\}$

With probability p:

$X_{n+1} \leftarrow W$

else

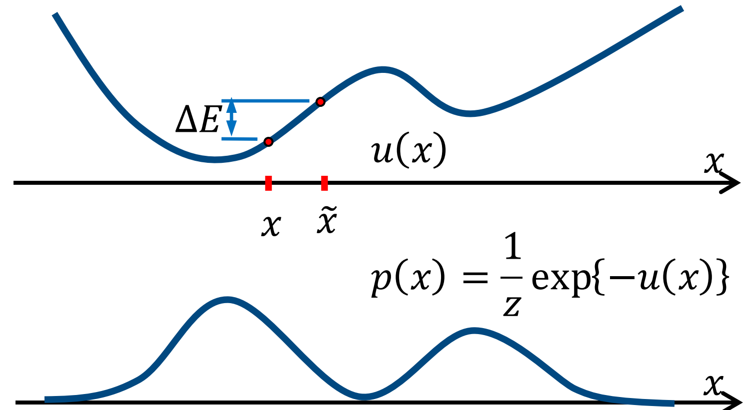
$X_{n+1} \leftarrow X_n$

- Result:
  - Generates a homogeneous, reversible, ergodic Markov Chain with stationary distribution

$$p(x) = \frac{1}{z} \exp\{-u(x)\}$$

Stochastic Sample of Gibbs Distribution

- Gibbs distribution
  - $u(x)$: Energy function
  - $p(x)$: Probability density



- Interpretation
  - Proposals that reduce energy are always accepted
  - Proposals that increase energy are sometimes accepted.
- Problem:
  - Requires a parametric expression for $u(x)$.

Data Driven Stochastic Sampling?

- Two approaches to modeling:
  - Parametric model (traditional):
    - Human design; small number of parameters; often a physics model
    - Example: $u_\theta(x) = \sum_{\{i,j\}} \theta_{i,j} |x_i - x_j|$
  - Data Driven model (proposed):

$$\left\{ \begin{matrix} \{X_0, \cdots, X_{K-1}\} \\ \text{training samples} \end{matrix} \right\} \Rightarrow u_\theta(x)$$

deep neural network

- Great idea, but…
  - How do we train a DNN to fit the $u(x)$ that describes training data?
  - We don't even know $u(x)$!
  - This reduces are problem to an inference problem.
  - But what loss function should we use?
- Solution: <u>Score Matching</u>

Score Matching

- The Score
- Denoising Score Matching
- Geometric Interpretation

Defining the Score[†]

- Let $X \sim p(x)$ be a random object, then we define
  - Log probability is given by[†]:
    $$l(x) = \log p(x) = -u(x) + \mathrm{c}$$
  - The score is given by[†]:
    $$s(x) = \nabla_x \log p(x) = -\nabla_x u(x)$$
- Important ideas:
  - If you know $s(x)$, then you know $u(x)$.
  - $s(x)$ is a conservative vector field $\Leftrightarrow [\nabla_x s(x)]^t = \nabla_x s(x)$

[†]Definitions are given assuming a Bayesian estimation framework. The more traditional Frequentist framework uses slightly different definitions and terminology.

Score Matching

- Let $X \sim p(x) = \frac{1}{z}\exp\{-u(x)\}$:
  - Then we can learn the score, $s_\theta(x)$, from data via

$$\hat{\theta} = \arg\min_{\theta} L_{SM}(\theta)$$

  - where

$$L_{SM}(\theta) = E\left[\frac{1}{2}\|s(X) - s_\theta(X)\|^2\right]$$

- Then we have that:
  - $s_{\hat{\theta}}(x)$ is an estimate of the score
  - But it may not be a conservative vector field.
- Important Question: Where do we get $\mathrm{s(x)} = \nabla_x u(x)$?

Denoising Score Matching: Theorem*

- Theorem (Vincent):
  - $X \sim p(x) = \frac{1}{z}\exp\{-u(x)\}$ Gibbs distribution of X
  - $\tilde{X}|X \sim q_\sigma(\tilde{x}|x)$ Proposal distribution†
  - $\tilde{X} \sim p_\sigma(\tilde{x}) = \frac{1}{z}\exp\{-u_\sigma(x)\}$ Gibbs distribution of $\tilde{X}$
  - $s_\sigma(\tilde{x}) = -\nabla_{\tilde{x}}\, u_\sigma(\tilde{x})$ Score of $\tilde{X}$

  and define:
  - $L_{SM}(\theta;\sigma) = E\left[\frac{1}{2}\left\|s_\sigma(\tilde{X}) - s_\theta(\tilde{X})\right\|^2\right]$
  - $L_{DSM}(\theta;\sigma) = E\left[\frac{1}{2}\left\|\nabla_{\tilde{x}} \log q_\sigma(\tilde{X}|X) - s_\theta(\tilde{X})\right\|^2\right]$.

  Then

$$L_{SM}(\theta;\sigma) = L_{DSM}(\theta;\sigma) + C$$

Proof: Clever but straight forward. See reference.

*P. Vincent. A connection between score matching and denoising autoencoders. Neural Computation, 23(7):1661–1674, 2011.

†We assume the technical conditions that $q_\sigma(\tilde{x}|x)$ is continuously differentiable w.r.t. $\tilde{x}$ and $\forall x, \tilde{x}$, $q_\sigma(\tilde{x}|x) > 0$.

Proof of Denoising Score Matching Theorem*

Appendix

Proof that $J_{ESMq_\sigma} \smile J_{DSMq_\sigma}$ (11)

The explicit score matching criterion using the Parzen density estimator is defined in Eq. 7 as

$$J_{ESMq_\sigma}(\theta) = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})}\left[\frac{1}{2}\left\|\psi(\tilde{\mathbf{x}};\theta) - \frac{\partial \log q_\sigma(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}}\right\|^2\right]$$

which we can develop as

$$J_{ESMq_\sigma}(\theta) = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})}\left[\frac{1}{2}\left\|\psi(\tilde{\mathbf{x}};\theta)\right\|^2\right] - S(\theta) + C_2 \qquad (16)$$

where $C_2 = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})}\left[\frac{1}{2}\left\|\frac{\partial \log q_\sigma(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}}\right\|^2\right]$ is a constant that does not depend on θ, and

$$\begin{aligned}
S(\theta) &= \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})}\left[\left\langle \psi(\tilde{\mathbf{x}};\theta), \frac{\partial \log q_\sigma(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}}\right\rangle\right] \\
&= \int_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}}) \left\langle \psi(\tilde{\mathbf{x}};\theta), \frac{\partial \log q_\sigma(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}}\right\rangle d\tilde{\mathbf{x}} \\
&= \int_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}}) \left\langle \psi(\tilde{\mathbf{x}};\theta), \frac{\frac{\partial}{\partial \tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})}{q_\sigma(\tilde{\mathbf{x}})}\right\rangle d\tilde{\mathbf{x}} \\
&= \int_{\tilde{\mathbf{x}}} \left\langle \psi(\tilde{\mathbf{x}};\theta), \frac{\partial}{\partial \tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})\right\rangle d\tilde{\mathbf{x}} \\
&= \int_{\tilde{\mathbf{x}}} \left\langle \psi(\tilde{\mathbf{x}};\theta), \frac{\partial}{\partial \tilde{\mathbf{x}}} \int_{\mathbf{x}} q_0(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) d\mathbf{x}\right\rangle d\tilde{\mathbf{x}} \\
&= \int_{\tilde{\mathbf{x}}} \left\langle \psi(\tilde{\mathbf{x}};\theta), \int_{\mathbf{x}} q_0(\mathbf{x}) \frac{\partial q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}} d\mathbf{x}\right\rangle d\tilde{\mathbf{x}} \\
&= \int_{\tilde{\mathbf{x}}} \left\langle \psi(\tilde{\mathbf{x}};\theta), \int_{\mathbf{x}} q_0(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) \frac{\partial \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}} d\mathbf{x}\right\rangle d\tilde{\mathbf{x}} \\
&= \int_{\tilde{\mathbf{x}}} \int_{\mathbf{x}} q_0(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) \left\langle \psi(\tilde{\mathbf{x}};\theta), \frac{\partial \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}}\right\rangle d\mathbf{x} d\tilde{\mathbf{x}} \\
&= \int_{\tilde{\mathbf{x}}} \int_{\mathbf{x}} q_\sigma(\tilde{\mathbf{x}}, \mathbf{x}) \left\langle \psi(\tilde{\mathbf{x}};\theta), \frac{\partial \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}}\right\rangle d\mathbf{x} d\tilde{\mathbf{x}} \\
&= \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}},\mathbf{x})}\left[\left\langle \psi(\tilde{\mathbf{x}};\theta), \frac{\partial \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}}\right\rangle\right].
\end{aligned}$$

Substituting this expression for $S(\theta)$ in Eq. 16 yields

$$\begin{aligned}
J_{ESMq_\sigma}(\theta) &= \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})}\left[\frac{1}{2}\left\|\psi(\tilde{\mathbf{x}};\theta)\right\|^2\right] \\
&\quad - \mathbb{E}_{q_\sigma(\mathbf{x},\tilde{\mathbf{x}})}\left[\left\langle \psi(\tilde{\mathbf{x}};\theta), \frac{\partial \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}}\right\rangle\right] + C_2. \qquad (17)
\end{aligned}$$

12

We also have defined in Eq. 9,

$$J_{DSMq_\sigma}(\theta) = \mathbb{E}_{q_\sigma(\mathbf{x},\tilde{\mathbf{x}})}\left[\frac{1}{2}\left\|\psi(\tilde{\mathbf{x}};\theta) - \frac{\partial \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}}\right\|^2\right],$$

which we can develop as

$$\begin{aligned}
J_{DSMq_\sigma}(\theta) &= \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})}\left[\frac{1}{2}\left\|\psi(\tilde{\mathbf{x}};\theta)\right\|^2\right] \\
&\quad - \mathbb{E}_{q_\sigma(\mathbf{x},\tilde{\mathbf{x}})}\left[\left\langle \psi(\tilde{\mathbf{x}};\theta), \frac{\partial \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}}\right\rangle\right] + C_3 \qquad (18)
\end{aligned}$$

where $C_3 = \mathbb{E}_{q_\sigma(\mathbf{x},\tilde{\mathbf{x}})}\left[\frac{1}{2}\left\|\frac{\partial \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}}\right\|^2\right]$ is a constant that does not depend on θ.

Looking at equations 17 and 18 we see that $J_{ESMq_\sigma}(\theta) = J_{DSMq_\sigma}(\theta) + C_2 - C_3$. We have thus shown that the two optimization objectives are equivalent.

13

*Reproduced from P. Vincent. A connection between score matching and denoising autoencoders. Neural Computation, 23(7):1661–1674, 2011.

DSM with Additive White Gaussian Noise

- Take the proposal distribution to be

$$\tilde{X} = X + \sigma W \text{ where } W \sim N(0, I)$$

  – Then we have that

$$q_\sigma(\tilde{x}|x) = \frac{1}{(2\pi\sigma^2)^{\frac{p}{2}}} \exp\left\{-\frac{1}{2\sigma^2}\|\tilde{x} - x\|^2\right\}$$

$$\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x) = \frac{1}{\sigma^2}(x - \tilde{x})$$

  – So, then the DSM loss function is*

$$L_{DSM}(\theta;\sigma) = E\left[\frac{1}{2}\left\|\frac{1}{\sigma^2}(X - \tilde{X}) - s_\theta(\tilde{X})\right\|^2\right]$$

Score for distribution of $\tilde{X}$

noise-less image

noisy image

We can compute this!

*[Yang Song, Y. and Stefan Ermon, "Improved Techniques for Training Score-Based Generative Models", Neural Information Processing Systems 2020.]

The DSM with AWGN: Loss Function

- Goal: Formulate loss function from training data
  - $\{x_0, \cdots, x_{K-1}\}$ - training samples from desired distribution
  - For $k = 0, \cdots, K-1$, create noisy sample:

$$\tilde{x}_k = x_k + \sigma w_k \text{ where } w \sim N(0, I)$$

- Practical loss function is

$$\theta_\sigma = \arg\min_\theta \sum_{k=0}^{K-1} \frac{1}{2} \left\| \frac{1}{\sigma^2}(x_k - \tilde{x}_k) - s_\theta(\tilde{x}_k) \right\|^2$$

Score for distribution of $\tilde{X}$

ground truth image

noisy image

Tweedie's Formula

- Define:
  - $\tilde{X} = X + \sigma W$ where $W \sim N(0, I)$
  - $s_\sigma(\tilde{x}) = -\log p_\sigma(\tilde{x})$
- Then from the DSM theorem, we then know that

$$\mathrm{E}\left[\frac{1}{\sigma^2}(X - \tilde{X}) | \tilde{X}\right] = s_\sigma(\tilde{X})$$

- This results in Tweedie's Formula

$$\mathrm{E}\left[X | \tilde{X}\right] = \mathrm{denoise}(\tilde{X}; \sigma^2) = \tilde{X} + \sigma^2 s_\sigma\left(\tilde{X}\right)$$

- Interpretation:
  - The MMSE denoiser can be implemented by adding the scaled score.
  - The score is of the noisy image, not the clean one.

Tweedie's Formula Interpretation

- Tweedie's Formula:

$$\text{Denoise}(\tilde{X};\sigma^2) = E[X|\tilde{X}] = \tilde{X} + \sigma^2 s_{\theta_\sigma}(\tilde{X})$$

  – or equivalently that

$$s_{\theta_\sigma}(\tilde{X}) = \frac{1}{\sigma^2}\left[\text{Denoise}(\tilde{X};\sigma^2) - \tilde{X}\right]$$

- Interpretation:
  – $\text{Denoise}(\tilde{X};\sigma^2)$ is a MMSE denoiser
  – $\sigma s_{\theta_\sigma}(\tilde{X})$ estimates the negative noise.
  – This is just residual training for an image denoiser.
  – As $\sigma \to 0$, then $s_{\theta_\sigma}(x) \to s(x)$

DSM with AWGN: Graphical Interpretation

- Take the proposal distribution to be

$$\tilde{X} = X + \sigma W \text{ where } W \sim N(0, I)$$

- If we first define

$$\tilde{L}_{DSM}(\theta, \tilde{x}; \sigma) = E\left[\frac{1}{2}\left\|\frac{1}{\sigma^2}(X - \tilde{x}) - s_\theta(\tilde{x})\right\|^2 \middle| \tilde{X} = \tilde{x}\right]$$

$$= \int_{\Re^N} \frac{1}{2}\left\|\frac{1}{\sigma^2}(x - \tilde{x}) - s_\theta(\tilde{x})\right\|^2 p_{\sigma^2}(x|\tilde{x})\,dx$$

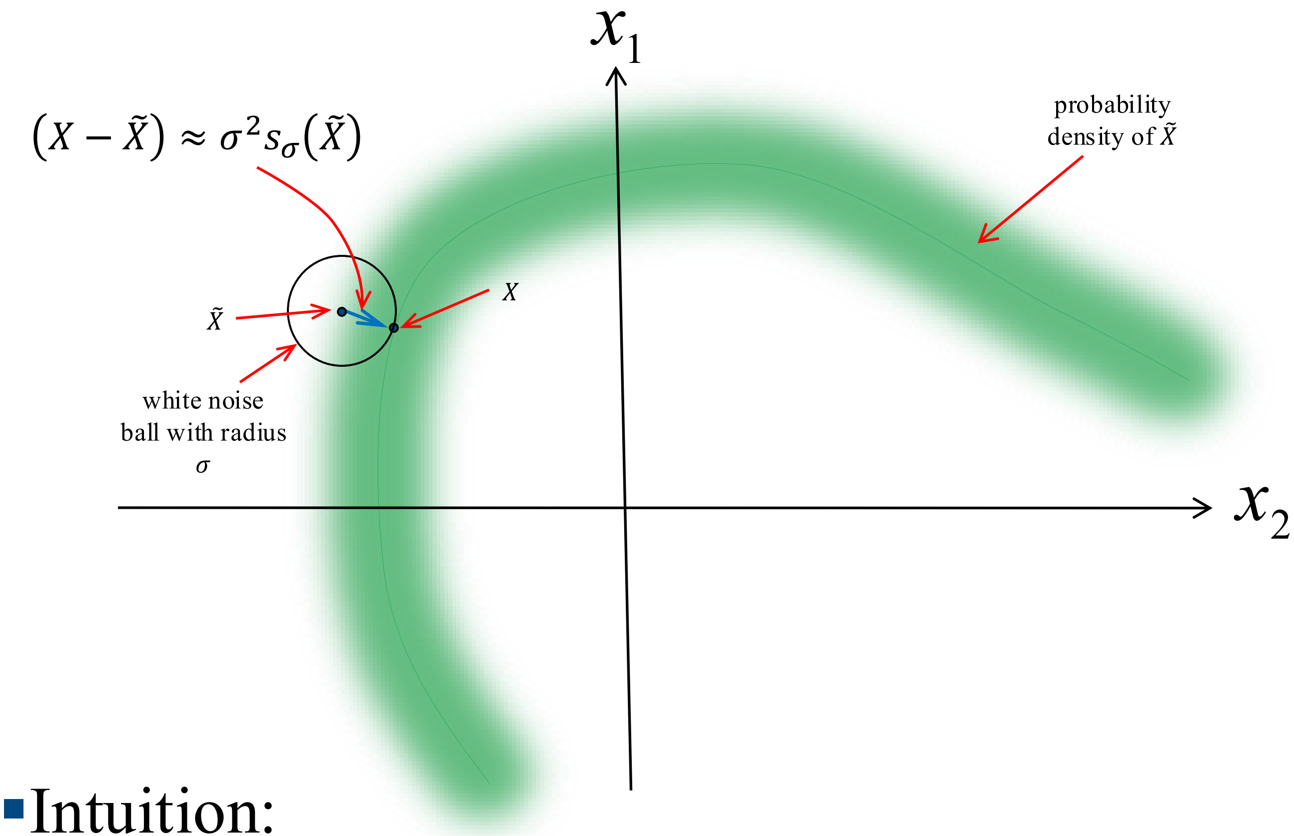
Posterior distribution of noiseless image given noisy image

– Then we have that

$$L_{DSM}(\theta; \sigma) = E\left[\tilde{L}_{DSM}(\theta, \tilde{X}; \sigma)\right]$$

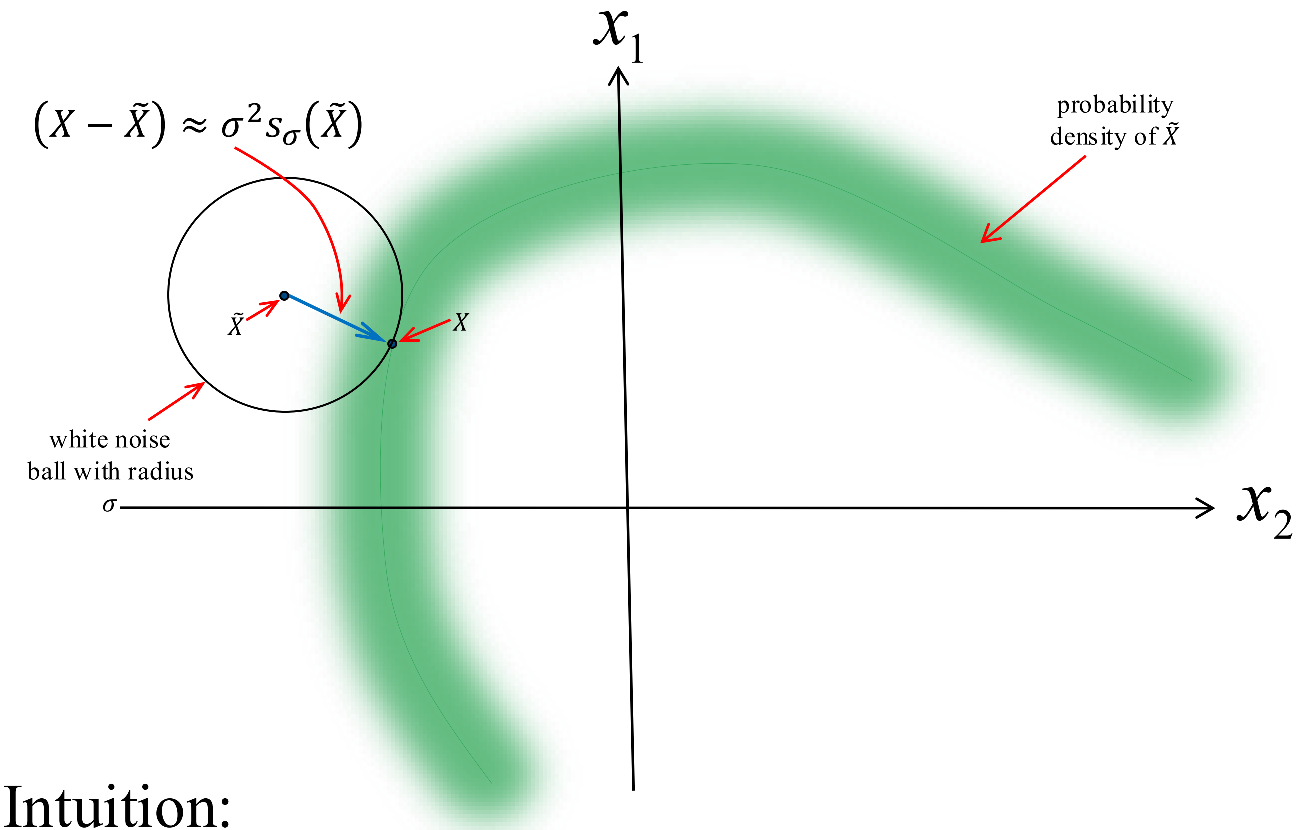
$$= \int_{\Re^N} \tilde{L}_{DSM}(\theta, \tilde{x}; \sigma)\, p_{\sigma^2}(\tilde{x})\,d\tilde{x}$$

Interpretation of Denoising Score Matching



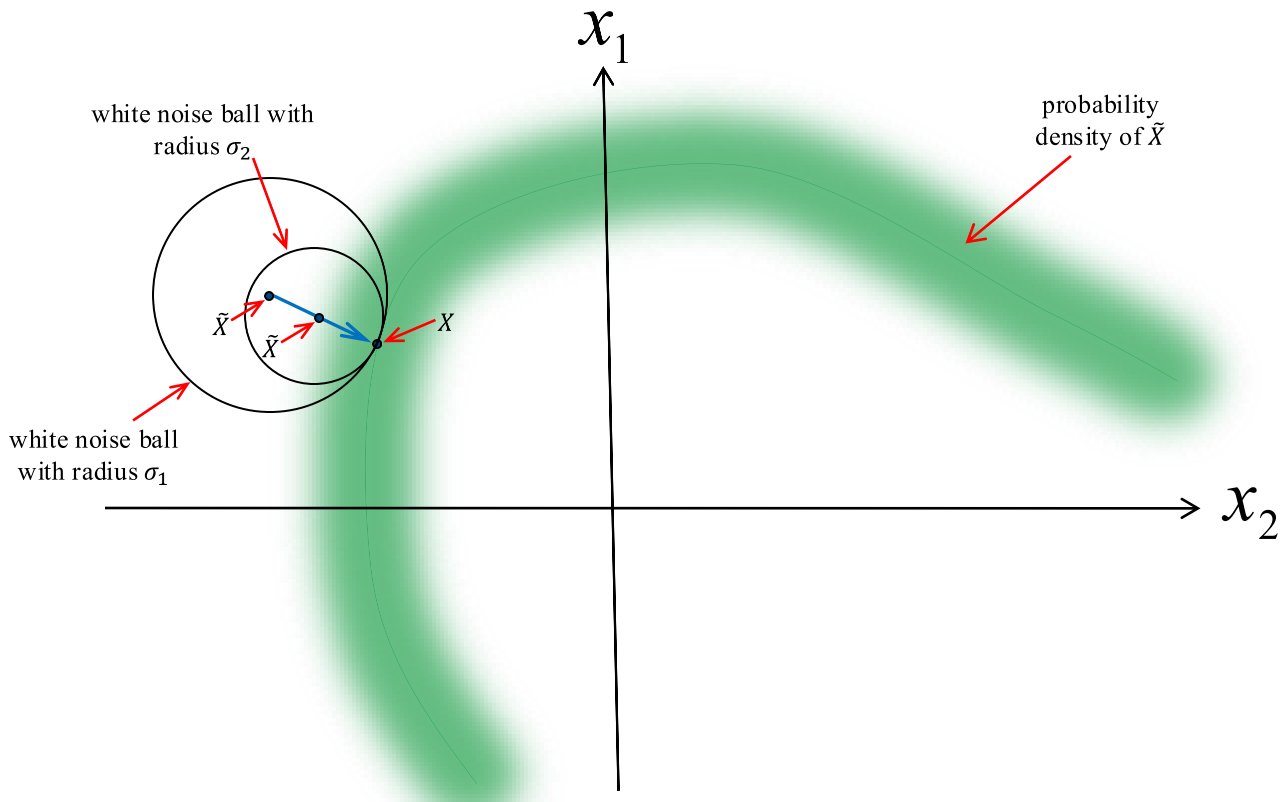
- Intuition:
  - Denoiser moves towards larger probability
  - Expected change approximates score

Interpretation of DSM with larger σ



- Intuition:
  - Samples further from the peak of the distribution
  - Allows for sample in low probability regions
  - Speeds convergence of MCMC

DSM with Descreasing σ



- Intuition:
  - Large σ samples far from the peak ⇒ used early in the simulation
  - Small σ samples close to the peak ⇒ used late in the simulation

Generative PnP (GPnP)*

- Proximal generators
- Markov chains
- Intuition behind GPnP

* Charles A. Bouman and Gregery T. Buzzard, "Generative Plug and Play: Posterior Sampling for Inverse Problems," 2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 2023, pp. 1-7, doi: 10.1109/Allerton58177.2023.10313413

Posterior Distribution

- The posterior distribution is given by

$$p(x|y) = \frac{1}{Z} \exp\{-u_1(x) - u_0(x)\}$$

where and

$$u_1(x) = -\log p(y|x)$$

$$u_0(x) = -\log p(x)$$

- Strategy:
  - Create Markov chain
  - Proximal generators: create sequential random samples
  - Modular implementation

Prior Proximal Generator

- Proximal Map

$$\bar{F}_0(v) = \arg\min_x \left\{ u_0(x) + \frac{1}{2\gamma^2} \|x - v\|^2 \right\}$$

- Proximal distribution

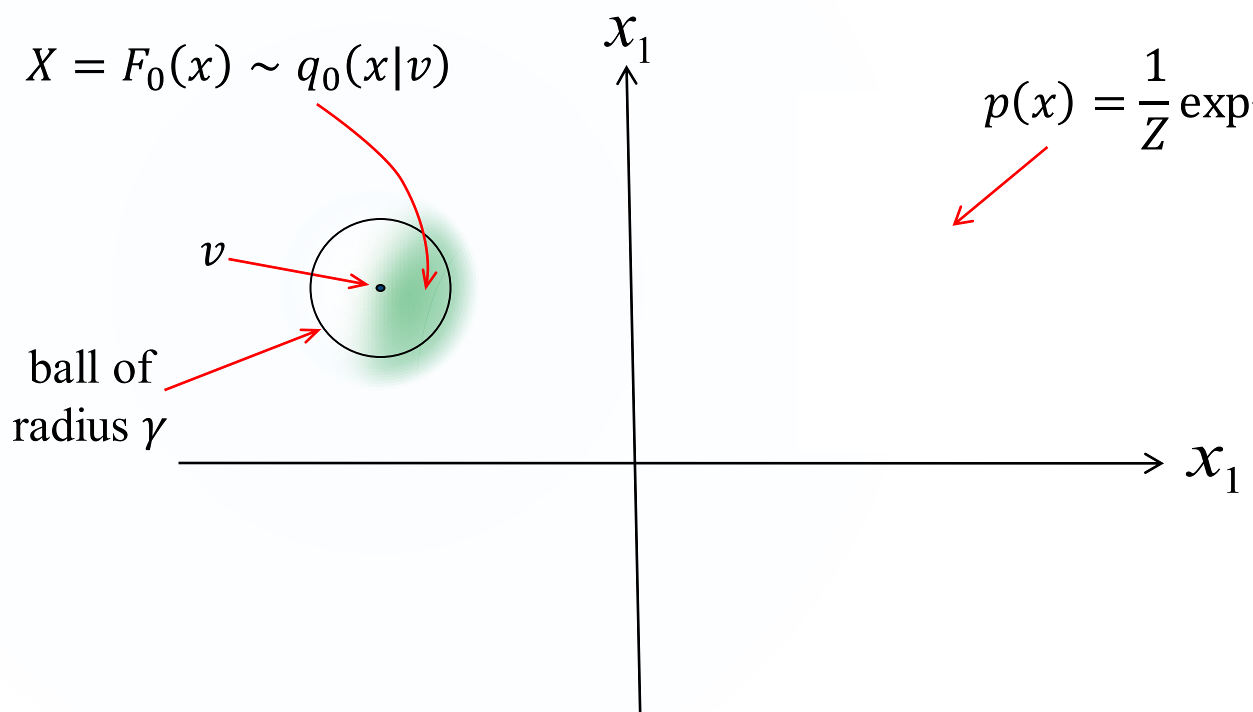
$$q_0(x|v) = \frac{1}{Z} \exp\left\{ -u_0(x) - \frac{1}{2\gamma^2} \|x - v\|^2 \right\}$$

- Proximal Generator

$$X = F_0(v) \sim q_0(x|v)$$

Generates a sample from the proximal distribution

Interpretation of Proximal Generator



- Intuition:
  - Locally samples from the prior distribution
  - Expected change approximates score

Forward Proximal Generator

- Proximal Map

$$\bar{F}_1(v) = \arg\min_x \left\{ u_1(x) + \frac{1}{2\gamma^2} \|x - v\|^2 \right\}$$

- Proximal distribution

$$q_1(x|v) = \frac{1}{Z} \exp\left\{ -u_1(x) - \frac{1}{2\gamma^2} \|x - v\|^2 \right\}$$

- Proximal Generator

$$X = F_1(v) \sim q_1(x|v)$$

Generates a sample from the proximal distribution

Generative PnP

Initialize $X = \text{Random}(0, I) + {}^1/_2$
Repeat {

$X \leftarrow F_0(X)$ // Prior Model Proximal Generator

$X \leftarrow F_1(X)$ // Forward Model Proximal Generator

}
$\text{Return}(x)$

- Observations/questions:
  - This is a Markov chain
  - Does it converge to a stationary distribution?
  - If so, then what is the stationary distribution?

GPnP Theorem

Theorem: Consider $X_n = F_1\left(F_0(X_{n-1})\right)$, then

- X_n is a reversible Markov chain
- X_n has a stationary distribution given by

$$\tilde{p}(x|y) = \frac{1}{Z}\exp\{-u_1(x) - \tilde{u}_0(x;\gamma^2)\}$$

  – where $\tilde{u}_0(x;\gamma\ )$ is $u_0(x)$ blurred with a Gaussian of variance γ^2.

- Bottom line:
  – Sequential application of F_0 and F_1 converges to "desired" distribution.
  – But GPnP introduces AWGN with variance γ^2 to the prior distribution!

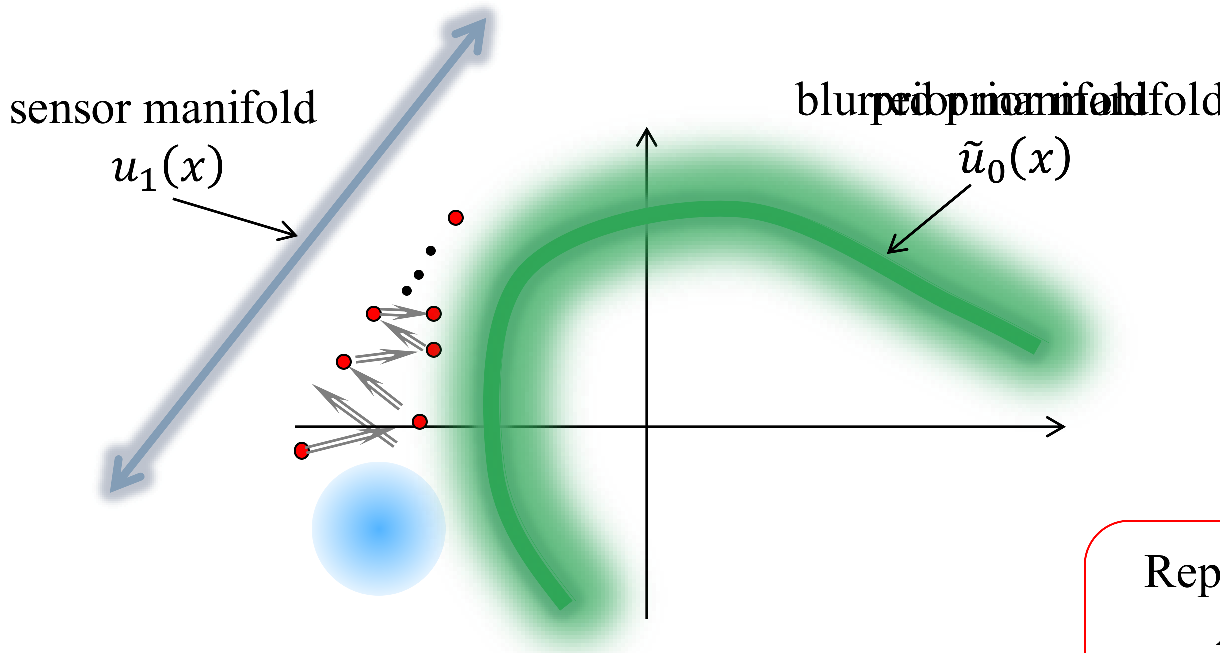
* Charles A. Bouman and Gregery T. Buzzard, "Generative Plug and Play: Posterior Sampling for Inverse Problems," 2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 2023, pp. 1-7, doi: 10.1109/Allerton58177.2023.10313413

Blurred Energy Function

- Definition of blurred energy function:
  - Let $\tilde{u}_0(x;\gamma)$ denote the blurring of energy function $u_o(x)$ with parameter $\gamma > 0$. Then
$$\tilde{u}_0(x;\gamma) = -\log\left(\exp\{-u_0(x)\} * g_\gamma(x)\right)$$
  - where $*$ denotes convolution and
$$g_\gamma(x) = \frac{1}{(2\pi\gamma^2)^{\frac{p}{2}}} \exp\left\{-\frac{1}{2\gamma^2}\|x\|^2\right\}.$$

- Notice that:
  - As $\lim_{\gamma\to 0} \tilde{u}_0(x;\gamma) = u_o(x)$
  - If $\tilde{X} \sim \tilde{u}_0(x;\gamma)$ and $X \sim u_o(x)$, then $\left(\tilde{X} - X\right) \sim N(0, \gamma I)$.
  - So $\tilde{X}$ is a noisy version of X.

Generative Plug-and-Play Intuition



Repeat {
$X \leftarrow F_0(X)$
$X \leftarrow F_1(X)$
}

Implementing Proximal Generators:

- Prior model proximal generator
- Forward model proximal generator
- GPnP Psuedo-code

Denoising Score Matching (Vincent 2011)*

- Tweedie's Formula:
  - The AWGN denoiser provides the score of the blurred distribution

$$s_\sigma(x) = -\nabla \tilde{u}_0(x;\sigma^2) = \frac{1}{\sigma^2}[\mathrm{Denoise}(x;\sigma) - x]$$

MMSE denoiser for AWGN

  - Exactly true for any σ

- But….
  - $\tilde{u}_0(x;\sigma^2)$ is the energy function for the blurred/noisy prior
  - So, we have the exact solution, but for a noisy prior

*P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Computation*, 2011.

Prior Proximal Generator Derivation

- Proximal distribution

regularization factor

$$q_0(x|v) = \frac{1}{Z}\exp\left\{-ru_0(x) - \frac{1}{2\gamma^2}\|x-v\|^2\right\}$$

- Instead, use the proximal generator of the blurred distribution

$$\tilde{q}_0(x|v) = \frac{1}{Z}\exp\left\{-r\tilde{u}_0(x;\sigma) - \frac{1}{2\gamma^2}\|x-v\|^2\right\}$$
$$\approx \frac{1}{Z}\exp\left\{-\frac{1}{2\gamma^2}\|x-[v+r\gamma^2 s_\sigma(v)]\|^2\right\}$$

where $-\tilde{u}_0(x;\sigma) \approx (x-v)s_\sigma(v)$ and $\gamma = \sqrt{\beta}\sigma$ where $\beta \ll 1$.

- Resulting proximal generator

$$X \sim \tilde{F}_0(v) = (1-r\beta)v + r\beta\text{Denoise}(v;\sigma) + \sqrt{\beta}W$$

  – where $W \sim N(0, I)$ and $\beta \ll 1$.

Prior Proximal Generator

- Prior proximal generator:
  - Approximation using score matching is:

$$\tilde{F}_0(x;\beta,\sigma) \approx (1 - r\beta)x + r\beta\text{Denoise}(x;\sigma) + \sqrt{\beta}\sigma W$$

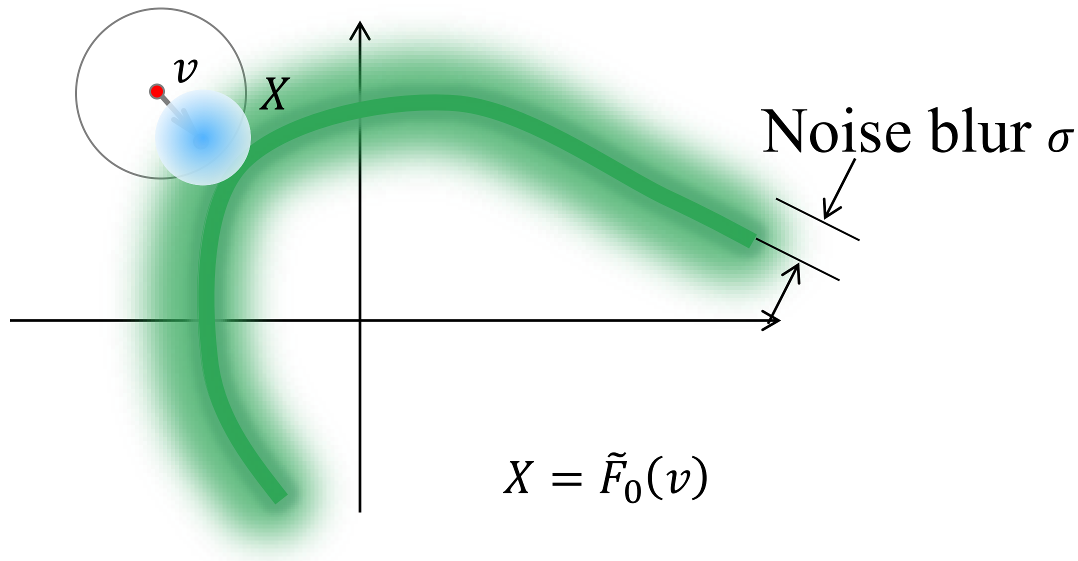
where:

  - $W \sim N(0, I)$ is AWGN
  - $\tilde{F}_0$ is the proximal generator for the blurred/noisy prior

- Parameters:
  - σ = prior blur – Typically varied from large to small
  - β = step size – $\beta = \frac{1}{4}$ works well
  - r = regularization factor – Typically $r = 1.3$ works well.

Prior Model Proximal Generator

$$\tilde{F}_0(v;\beta,\sigma) \approx (1-r\beta)v + r\beta\text{Denoise}(v;\sigma) + \sqrt{\beta}\sigma W$$



- Prior blurred by σ
- Step size = β
- Regularization = r

Forward Model Proximal Generator Derivation

- Proximal distribution

$$q_1(x|v) = \frac{1}{Z}\exp\left\{-u_1(x) - \frac{1}{2\gamma^2}\|x-v\|^2\right\}$$
$$\approx \frac{1}{Z'}\exp\left\{-\frac{1}{2\gamma^2}\|x-\bar{F}_1(v,\gamma)\|^2\right\}$$

where

$$\bar{F}_1(v) = \arg\min_x\left\{u_1(x) + \frac{1}{2\gamma^2}\|x-v\|^2\right\}$$

- Setting $\gamma = \sqrt{\beta}\sigma$, results in the forward model proximal generator:

$$F_1(v) \approx \bar{F}_1\left(v;\sqrt{\beta}\sigma\right) + \sqrt{\beta}\sigma W$$

  - where $W \sim N(0,I)$ and $\beta \ll 1$.

Forward Model Proximal Generator

- For γ small, just add white noise!

$$F_1(v) = \bar{F}_1(v; \sqrt{\beta}\sigma) + \sqrt{\beta}\sigma W$$

Proximal generator

Ordinary proximal map

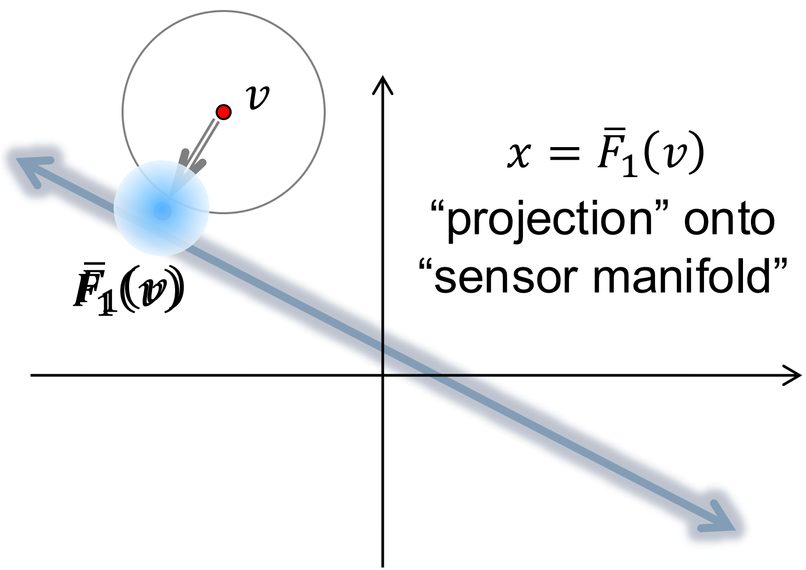
Proximal map parameter

additive white Gaussian noise

Forward Model Proximal Generator

- For small γ,

$$F_1(v) = \bar{F}_1(v) + \sqrt{\beta}\sigma W$$



GPnP Algorithm

$\beta = {}^1/_4;\ \sigma_{\max} = 2;\ r = 1.3;$

Initialize $X = \text{Random}(0, I) + {}^1/_2$

Repeat {

$X \leftarrow (1 - r\beta)X + r\beta\text{Denoise}(X; \sigma) + \sqrt{\beta}\sigma\text{RandN}(0, \text{I})$

$X \leftarrow \bar{F}_1(X) + \sqrt{\beta}\sigma\text{RandN}(0, \text{I})$

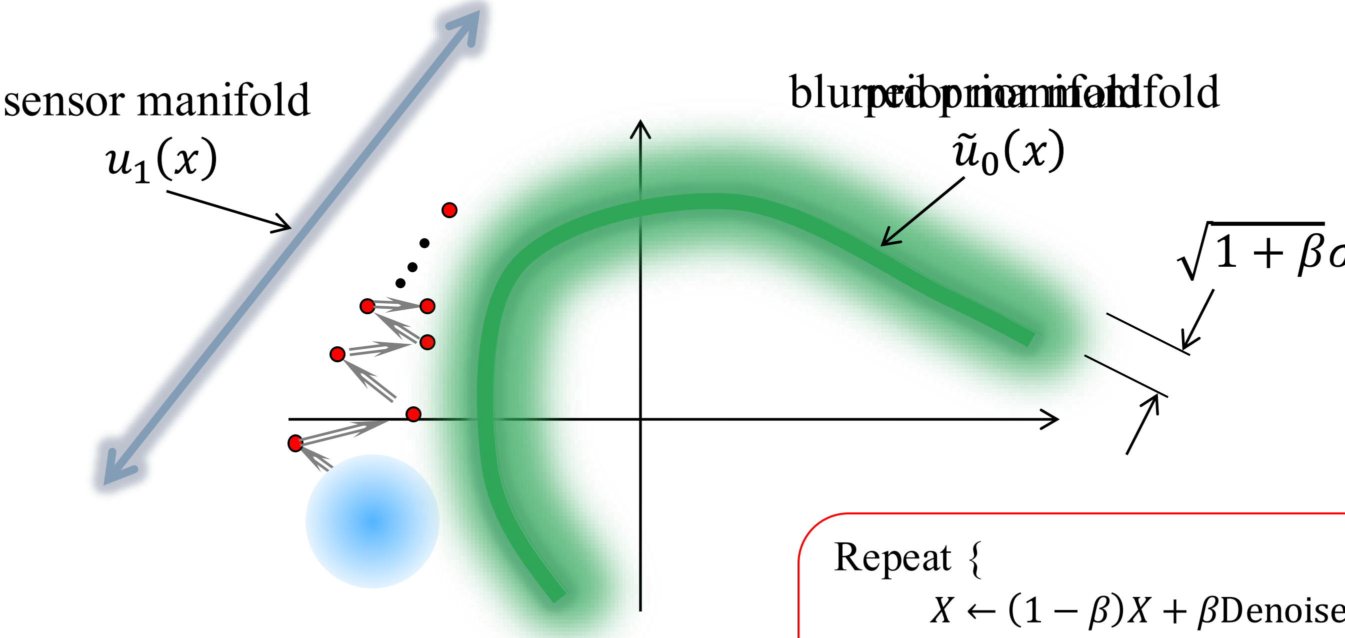
$\sigma \leftarrow \text{Reduce}(\sigma)$

}

Return(x)

- Parameters:
  - σ = prior blur
  - β = step size
  - r = regularization factor
  - Denoise($X; \sigma$) - MMSE denoiser trained for AWGN with variance σ^2.

GPnP Interations



Repeat {

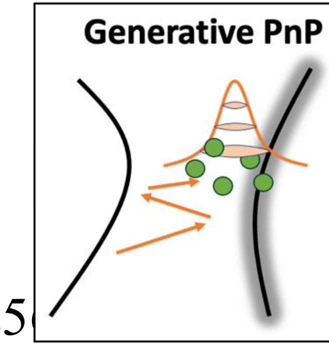
$X \leftarrow (1-\beta)X + \beta\text{Denoise}(X; \alpha\sigma) + \sqrt{\beta}\sigma\text{W}$

$X \leftarrow \bar{F}_1(X) + \sqrt{\beta}\sigma\text{W}$

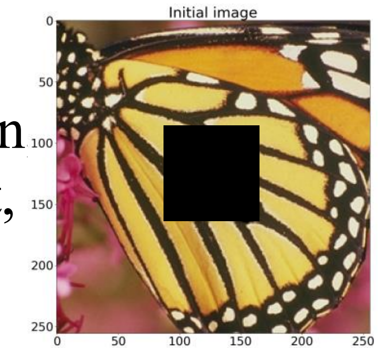
$\sigma \leftarrow \text{Reduce}(\sigma)$

}

GPnP: Effect of the prior (denoiser)



- Experiment:
  - Prior proximal generator (denoisers):
    - BM3D, DRUNet*, DDPM denoiser trained on CelebAHQ-25
  - Forward model: interpolation with missing rectangle.
  - Same parameters work for different problems (interpolation tomography, …) and different denoisers (BM3D, DRUNet,



* Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte, "Plug-and-Play Image Restoration With Deep Denoiser Prior," TPAMI 2022.

** J. Ho, A. Jain, P. Abbeel, "Denoising Diffusion Probabilistic Models", arxiv:2006.11239, 2020.

Baseline: denoising only

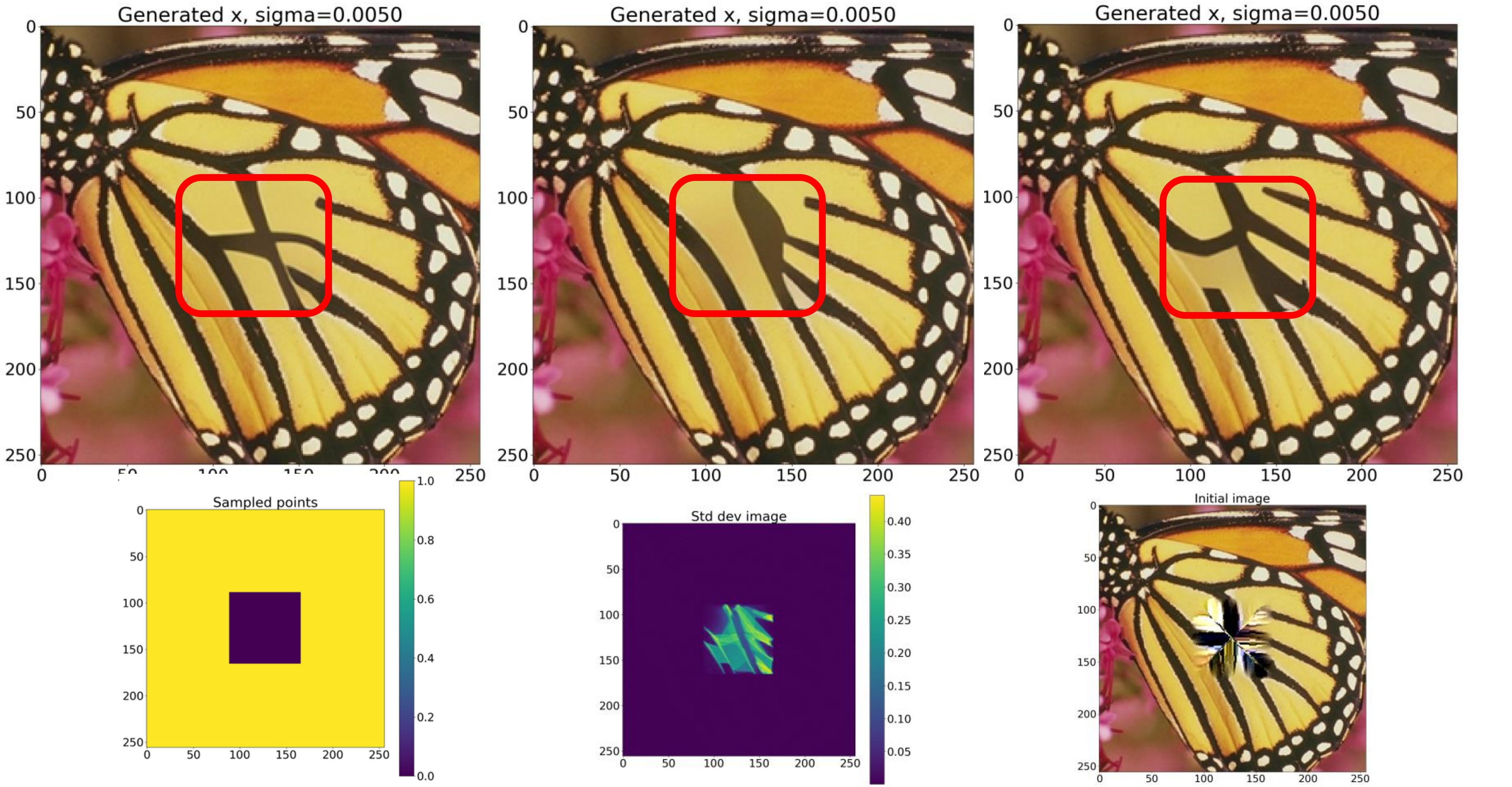
Noisy
NRMSE= 0.554

DRUNet
NRMSE= 0.074

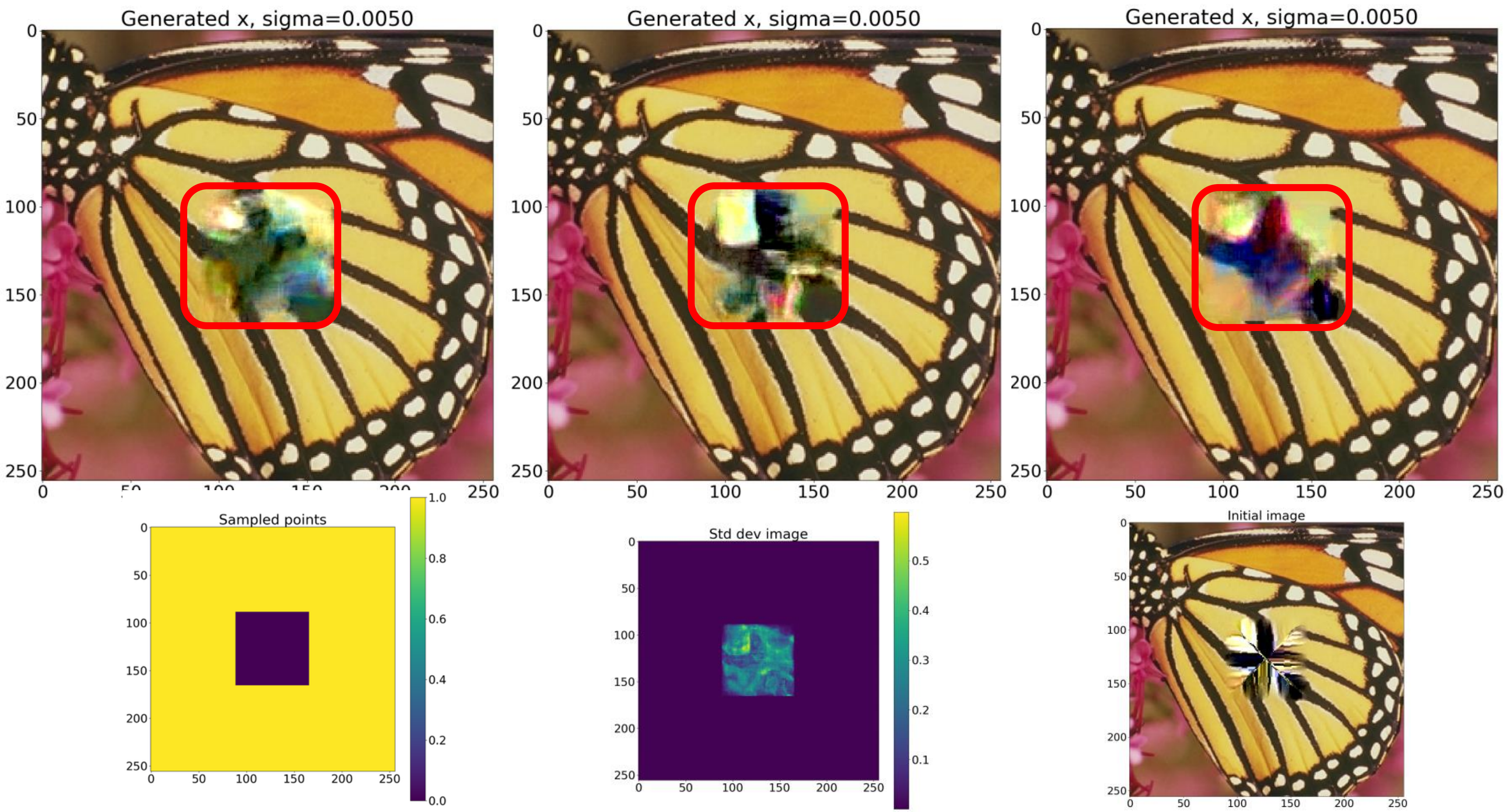
DDPM-CelebA
NRMSE= 0.090

BM3D
NRMSE=0.091

GPnP Inpainting: Center rectangle omitted - 3 samples, DRUNet prior

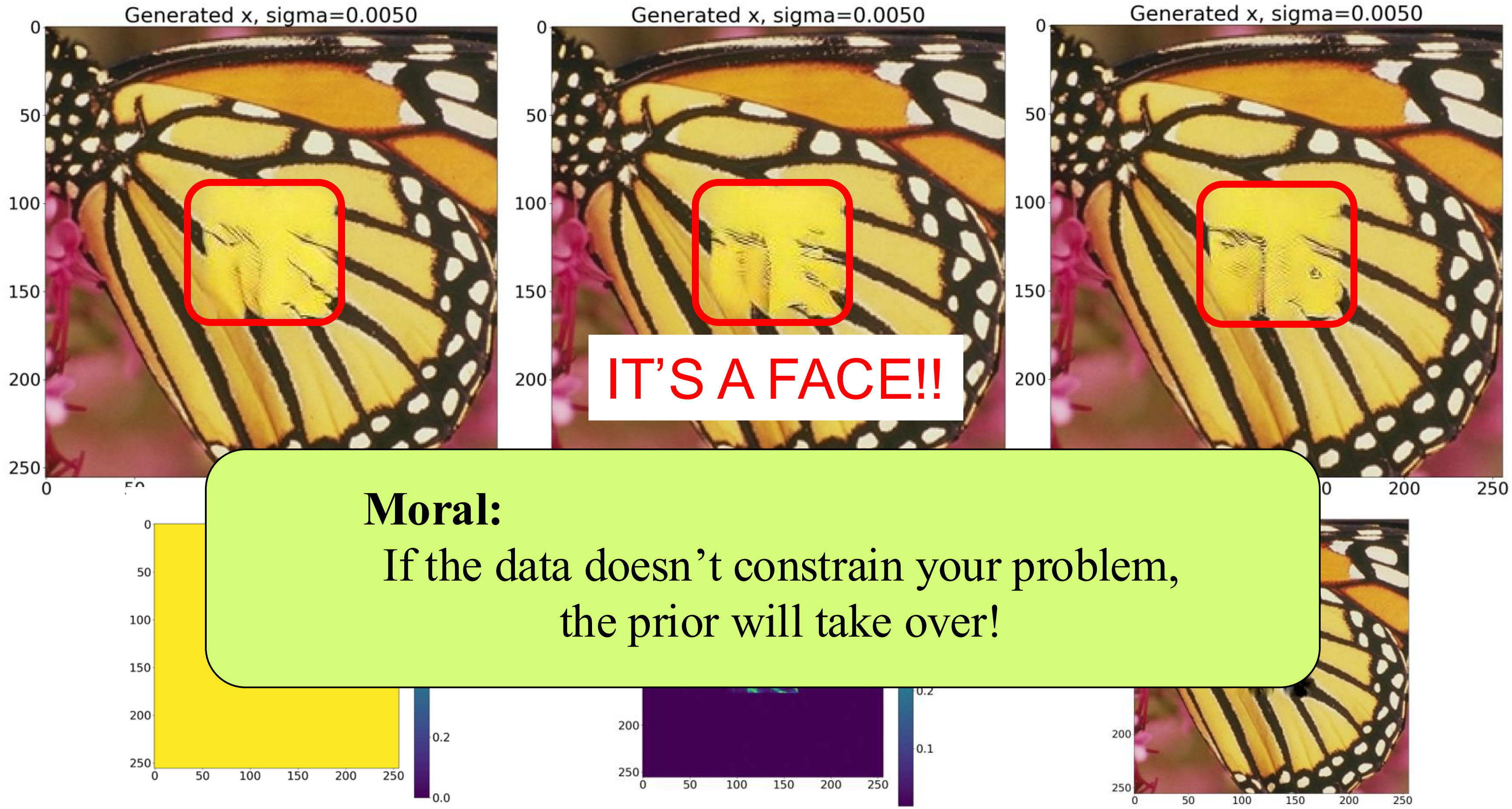


GPnP Inpainting: Center rectangle omitted - 3 samples, BM3D prior



GPnP Inpainting:

Center rectangle omitted - 3 samples,
DDPM denoiser trained on CelebAHQ-256 prior



Moral:
If the data doesn't constrain your problem,
the prior will take over!

GPnP-EM for Blind Deconvolution

- Joint MAP/ML Estimation of Blur Kernel
- ML Estimate of Blur Kernal
- Required Functions for SEM-GPnP
- GPnP-EM Algorithm

Can GPnP adapt to model mismatch?

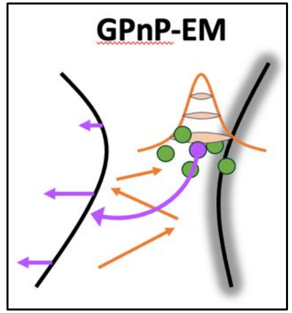
- Problem: Forward model with parameters ϕ may not match reality
  - Now we need to find
  - $(\hat{x}, \hat{\phi}) = \text{argmin}_{x,\phi} \left\{ \frac{1}{2} \left\| y - A_\phi x \right\|^2 + f_0(x) \right\}$
    - Nonlinear dependence: y depends on ϕ and x through $A_\phi x$
    - Typically, x is high-dimensional, ϕ is low-dimensional
    - Direct joint minimization is difficult
    - Alternating minimization often gets stuck in local minima
  - Examples: blind deblurring, CT geometry parameters

- Question: Can we estimate ϕ without x?

GPnP-EM: Expectation-Maximization

- Goal: Compute ML estimate of ϕ

$$\hat{\phi} = \text{argmin}_\phi\{-\log p_\phi(y) + s(\phi)\}$$
$$= \text{argmin}_\phi\left\{-\log \int_{\Re^N} p_\phi(y,x)dx + s(\phi)\right\}$$
$$= \text{argmin}_\phi\left\{\int \frac{1}{Z}\exp\left\{-\frac{1}{2}\left\|y - A_\phi x\right\|^2 + f_0(x)\right\} dx + s(\phi)\right\}$$



- Problem: High-dimensional integral is not practical

- Solution: Expectation-Maximization
  - Expectation: Use GPnP to get posterior samples
  - Maximization: Estimate A_ϕ from these samples: low-dimensional optimization
  - Iterate and decrease σ

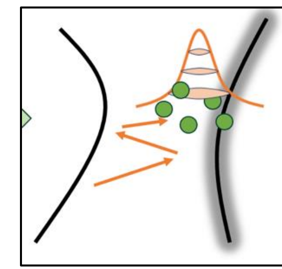
- Key point:
  - Estimating (ϕ, x) together is ill-conditioned, but the use of sample expectation provides implicit regularization.

GPnP-EM Implementation

Goal: $\hat{\phi} = \text{argmin}_\phi\{-\log p_\phi(y) + s(\phi)\}$

- **E-step:**
  - Use GPnP to generate samples from the posterior using previous ϕ_n:

$$X_1, \dots, X_K \sim \exp\left(-\frac{1}{2}\left\|y - A_{\phi_n}x\right\|^2 - f_0(x)\right)$$

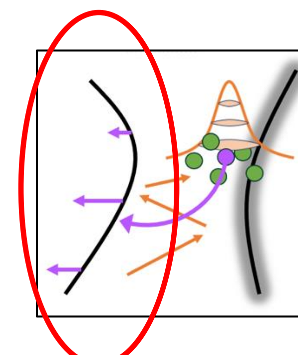


  - Define sample expectation of NLL as a function of ϕ:

$$Q_n(\phi) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{2}\left\|y - A_\phi X_k\right\|^2$$

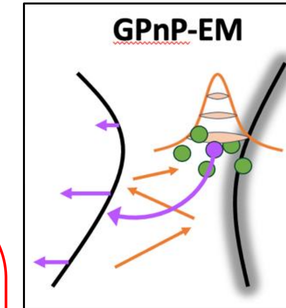
- **M-step:**

$$\phi_{n+1} = \text{argmin}_\phi\{Q_n(\phi) + s(\phi)\}$$



  - $s(\phi)$ is additional regularization on ϕ.

GPnP-EM algorithm



GPnP-EM

Repeat

for $k = 1, \dots, K$ # Get K samples

$$X \leftarrow (1 - r\beta)X_k + r\beta\text{Denoise}(X_\text{k};\ \sigma) + \sqrt{\beta}\sigma\ \text{RandN}(0, \text{I})$$

$$X_{k+1} \leftarrow F_1(X; \phi) + \sqrt{\beta}\sigma\ \text{RandN}(0, \text{I})$$

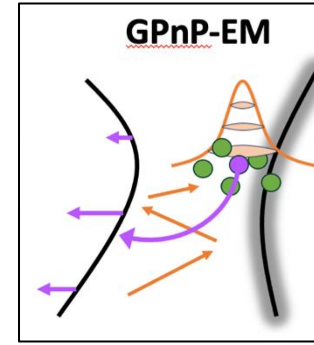
$\phi \leftarrow \text{argmin}_{\phi'}\{Q_n(\phi'; y, X_1, \dots, X_K) + s(\phi')\}$ # Estimate ϕ

$\sigma \leftarrow \text{Reduce}(\sigma)$

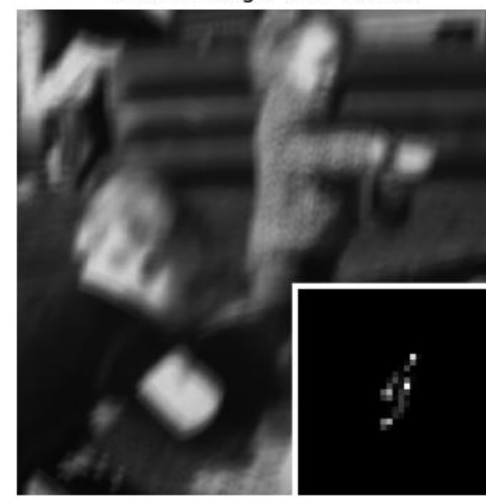
- Natural extension of GPnP: sample from the posterior to run the EM algorithm
- MMSE estimate of X can be obtained by averaging the $X_n's$.

GPnP-EM

- Blind deblurring:
  - Input a blurry image
  - Estimate the clean image and the blur kernel from the blurry image alone.
  - Use images from Levin, et al* with known blur kernels.



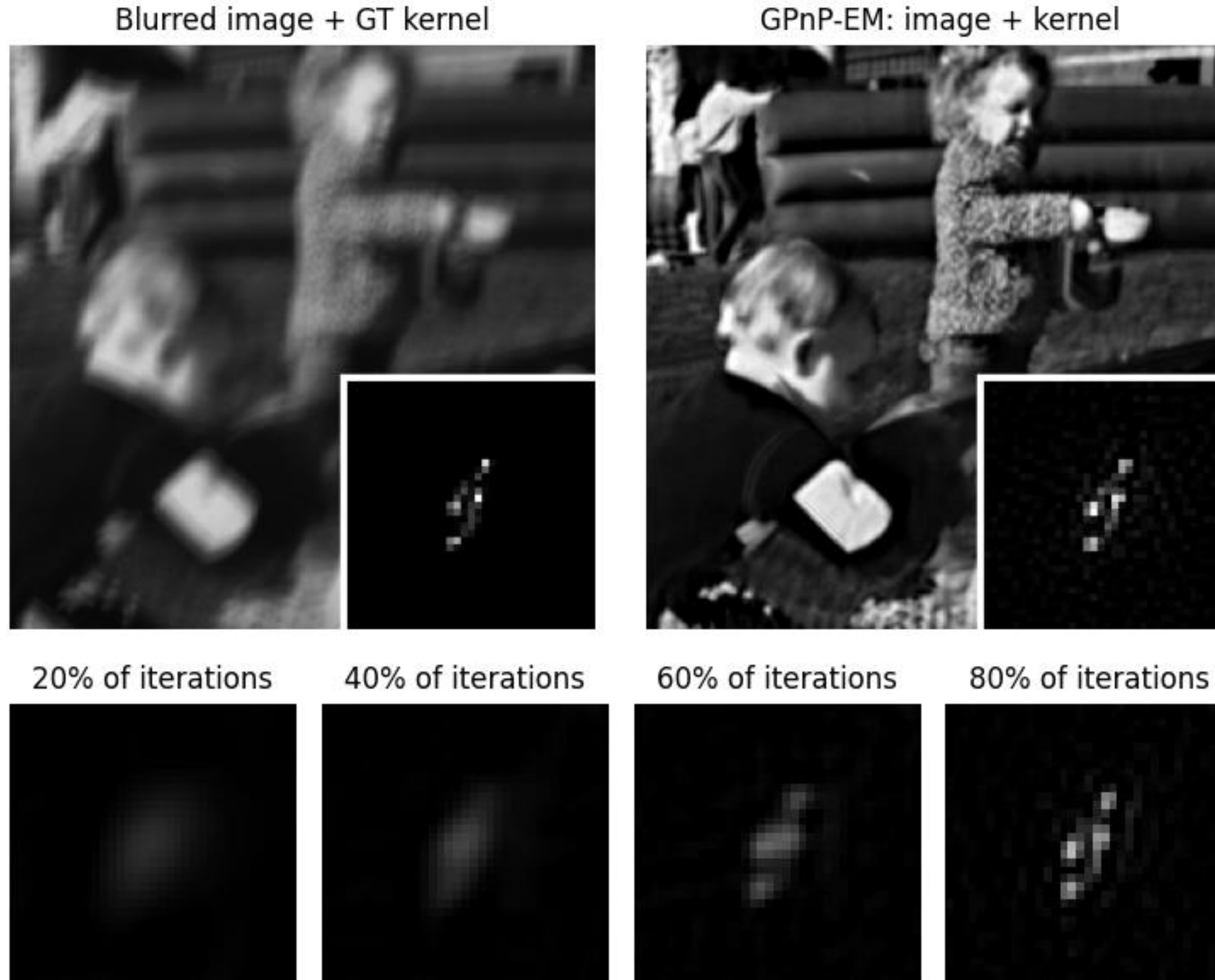
Blurred image + GT kernel



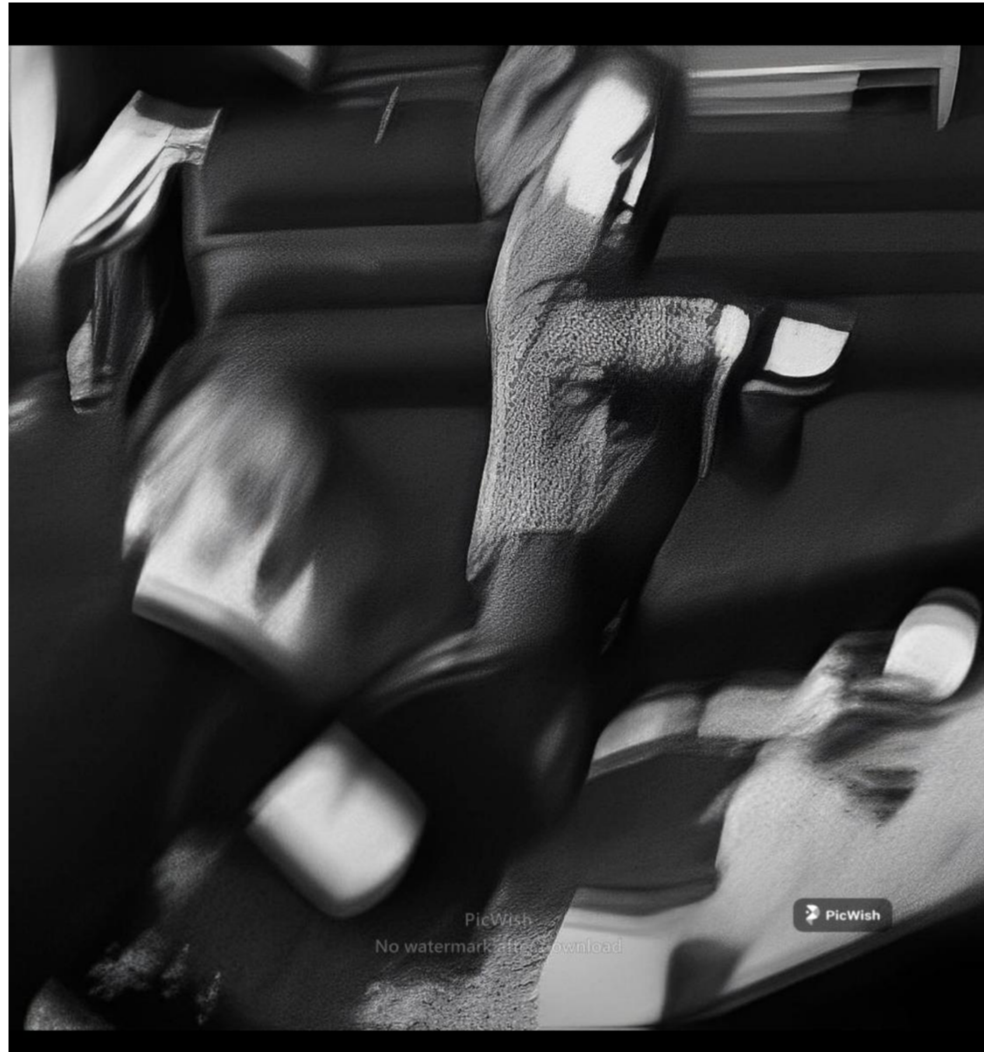
* A Levin, Y Weiss, F Durand, and WT Freeman, "Understanding blind deconvolution algorithms," IEEE TPAMI, 2011

GPnP-EM: Blind deblurring

Ground Truth Image

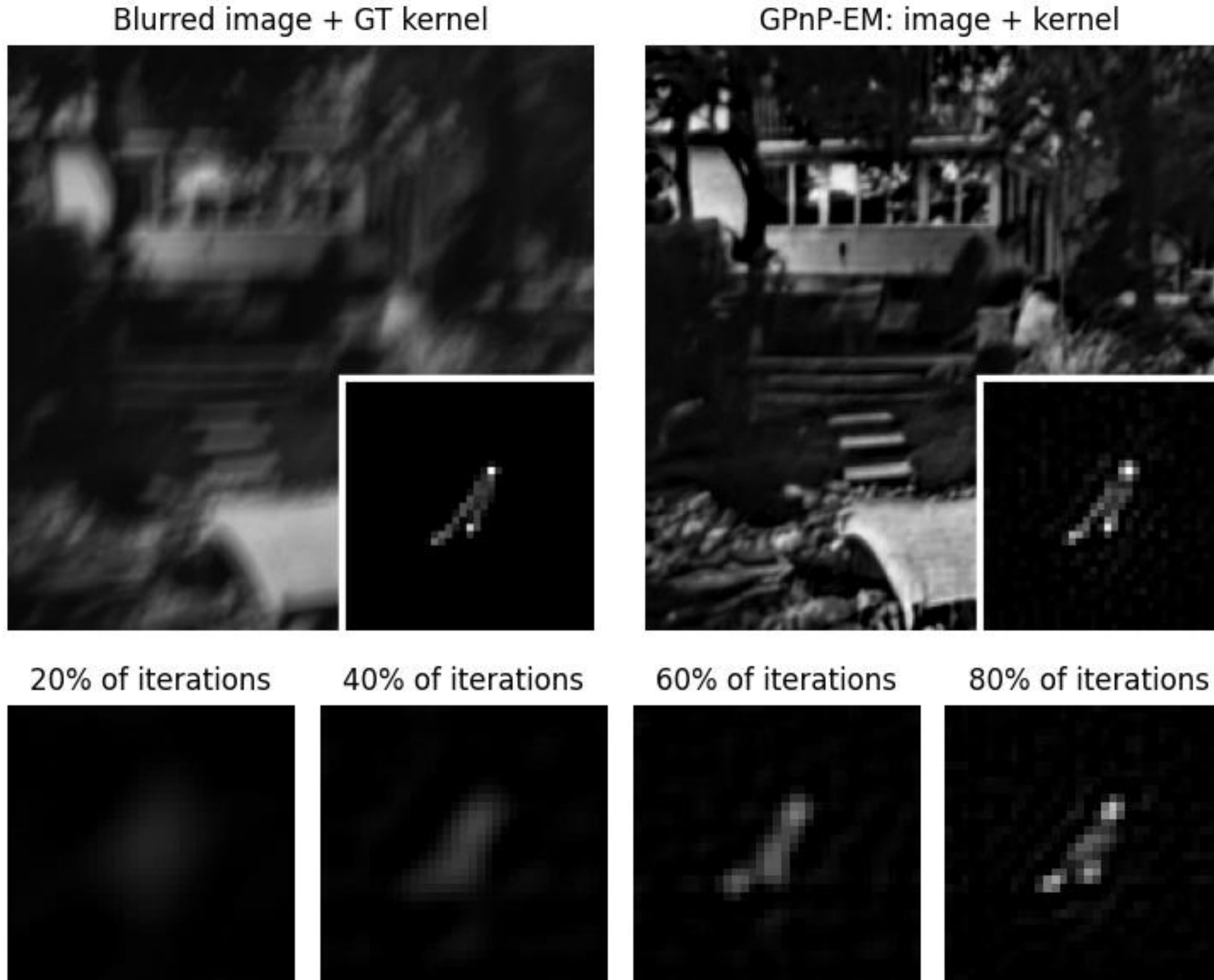


Typical online NN deblurring



GPnP-EM: Blind deblurring

Ground Truth Image



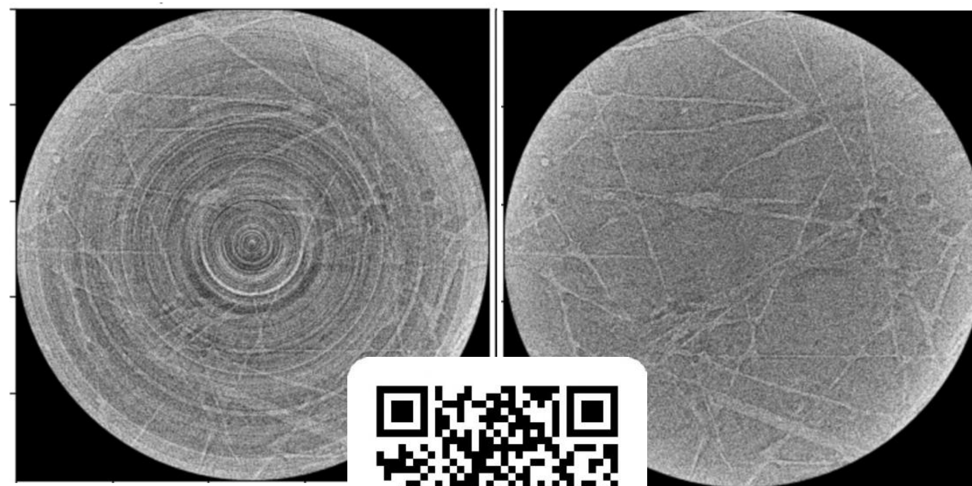
Typical online NN deblurring

More applications

- CT Imaging:
  - Center of rotation
  - Detector bias
  - Beam hardening
  - Material decomposition
  - ….

Original Reconstruction

Reconstruction with Ring Removal



MBIRJAX

Generative Diffusion Models*†

- Langevin dynamics

*Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole, "Score-Based Generative Modeling Through Stochastic Differential Equations" ICLR 2021.

†Yang Song, "Generative Modeling by Estimating Gradients of the Data Distribution," web blog post, May 5, 2021, https://yang-song.net/blog/2021/score .

Langevin Dynamics*

- How can you use the score to generate samples from the Gibbs distribution?
- Langevin dynamics:

$$X_n = X_{n-1} + \epsilon \nabla_x u(X_{n-1}) + \sqrt{2\epsilon}\, W_n$$

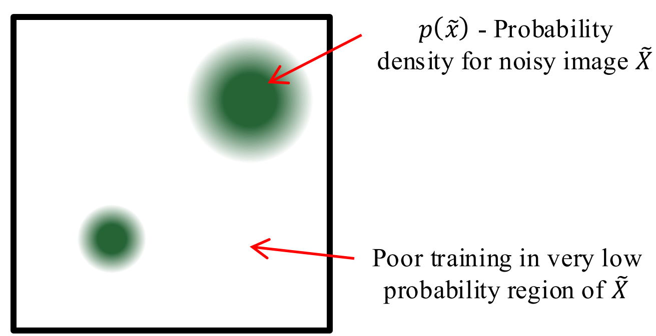
  – We can use our estimate of the score to generate

$$X_n = X_{n-1} + \epsilon s_{\theta_\sigma}(X_{n-1}) + \sqrt{2\epsilon}\, W_n$$

Score learns the gradient of the log probability.

White noise, $W_n \sim N(0, I)$.

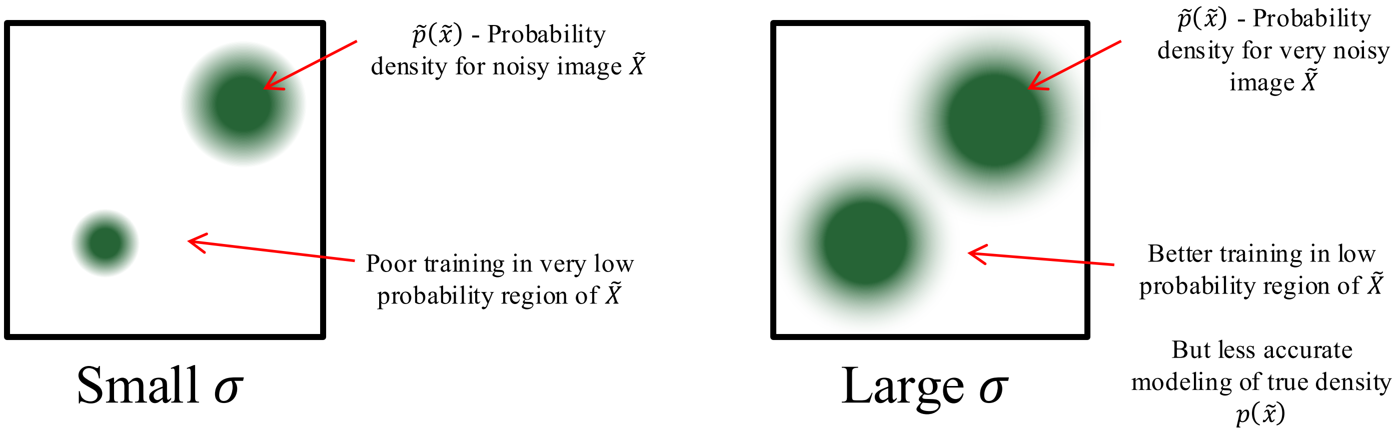
- Problem: Takes too long to converge



*Ulf Grenander and Michael Miller, "Representations of Knowledge in Complex Systems," J. Royal Statistical Society, vol. 56, no. 4, 1994.

Annealed Langevin Dynamics*

- Key idea: Increase σ to get better estimation in low density regions
  - Small vs Large values of σ



- Annealed Langevin dynamics:
  - Pick ϵ_o and let $\sigma_1 > \sigma_2 > \cdots > \sigma_N$

For $n = 1$ to N {

$$\epsilon_n \leftarrow \epsilon_o \frac{\sigma_n}{\sigma_L}$$

$$X_n \leftarrow X_{n-1} + \epsilon_n s_{\theta_{\sigma_n}}(X_{n-1}) + \sqrt{2\epsilon_n}\, W_n$$

}

*Yang Song and Stefano Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution," NeurIPS 2019.

Practical Recommendations: Annealed*†

- Annealed Langevin dynamics:
  - Pick ϵ_o and let $\sigma_1 > \sigma_2 > \cdots > \sigma_N$

$\epsilon_o \leftarrow$ init; $\sigma_{\min} \leftarrow$ init; $\sigma_{\max} \leftarrow$ init;

$\alpha \leftarrow \left(\frac{\sigma_{\min}}{\sigma_{\max}}\right)^{\frac{1}{N-1}}$;

For $n = 0$ to $N-1$ {

$\sigma_n \leftarrow \alpha^n \sigma_{\max}$

$\epsilon_n \leftarrow \epsilon_o \frac{\sigma_n}{\sigma_{\max}}$

$X_n \leftarrow X_{n-1} + \epsilon_n s_{\theta_{\sigma_n}}(X_{n-1}) + \sqrt{2\epsilon_n}\, W_n$

}

Annealed Langevin Dynamics

- Practical considerations
  - Geometric sequence for σ_n
  - $\sigma_{\max} = \max_{i,j} \text{RMS}(X_i - X_j)$ where X_i and X_j are training images.
  - Use a U-net (RefineNet) with skipped connections for score modeling.
  - Apply exponential moving average on the weights of the score-based model when used at test time.

*Yang Song, "Generative Modeling by Estimating Gradients of the Data Distribution," web blog post, May 5, 2021, https://yang-song.net/blog/2021/score
.

†Yang Song, Y. and Stefan Ermon, "Improved Techniques for Training Score-Based Generative Models", Neural Information Processing Systems 2020.

Langevin: Denoising Interpretation

- Annealed Langevin dynamics:

$$X_n = X_{n-1} + \epsilon_n s_{\theta_{\sigma_n}}(X_{n-1}) + \sqrt{2\epsilon_n}\, W_n$$

  - where

$$s_{\theta_\sigma}(x) = \frac{1}{\sigma^2}[\text{Denoise}(x; \sigma^2) - x]$$

- If we set $\epsilon_n = \sigma^2$, then we get

$$X_n = \text{Denoise}(X_{n-1}; \sigma^2) + \sqrt{2}\sigma\, W_n$$

  - where $W_n \sim N(0, I)$

- Interpretation:
  - Remove noise with variance σ^2, then add AWGN with variance $2\sigma^2$.
  - As $\sigma \to 0$, this iteration generates samples from the distribution $p(x)$.

Denoising Interpretation of Langevin

- Annealed Langevin dynamics:

$\sigma_{\min} \leftarrow \text{init}; \sigma_{\max} \leftarrow \text{init};$

$\alpha \leftarrow \left(\frac{\sigma_{\min}}{\sigma_{\max}}\right)^{\frac{1}{N-1}};$

For $n = 0$ to $N - 1$ {

$\sigma_n \leftarrow \alpha^n \sigma_{\max}$

$X_n \leftarrow \text{Denoise}(X_{n-1}; \sigma_n^2) + \sqrt{2}\sigma_n W_n$

}

Annealed Langevin Dynamics:
Denoising Interpretation

- Interpretation:
  - Remove noise with variance σ^2, then add back AWGN with variance $2\sigma^2$.
  - Denoiser trained using MMSE loss on samples from $p(x)$ with AWGN of variance σ^2.
  - As $\sigma \to 0$, this iteration generates samples from the distribution $p(x)$.