Data Driven Inverse Problems
Lecture 3 : Unsupervised Learning and Generative Models

Simon R. Arridge[1]

[1]Department of Computer Science, University College London, UK

Data Driven Inverse Problems
Autumn School
Sep 20th – 22nd 2023

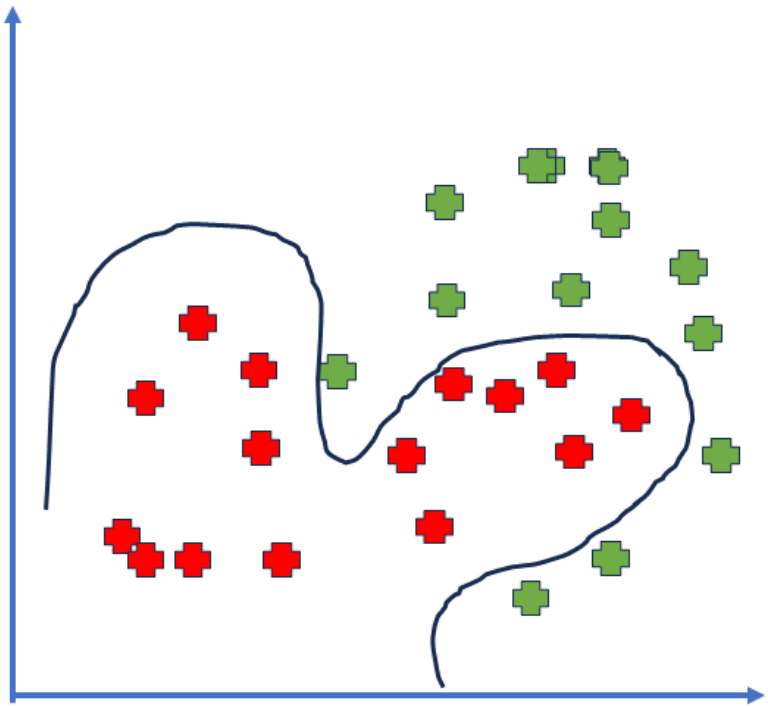
Outline

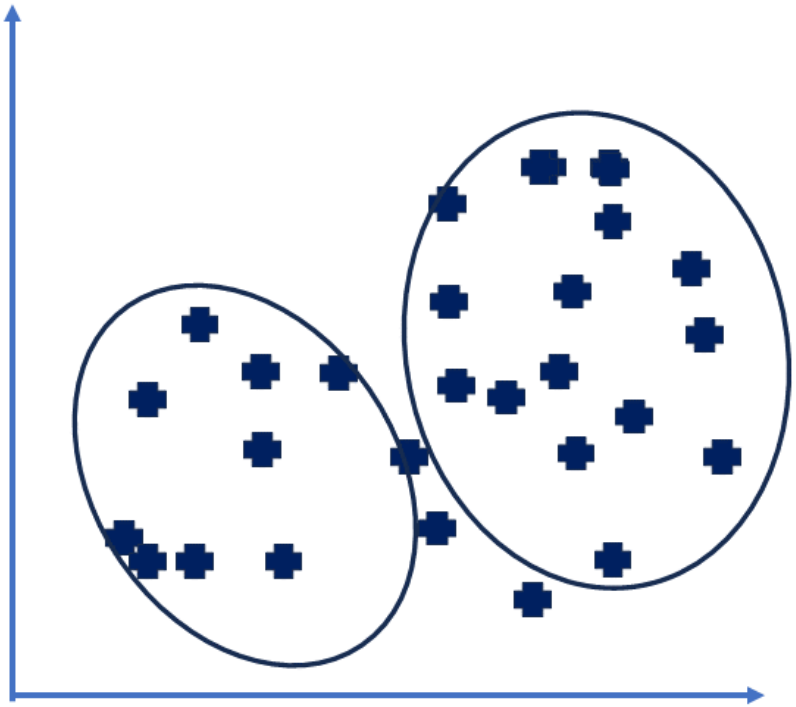
Outline

Introduction

Supervised vs Unsupervised learning

In supervised learning we have data elements together with *labels*. The labels may be a pairing with an element of another space
In unsupervised learning we have no labels and instead seek to identify patterns in the data, e.g. *clusters*





Introduction

Supervised vs Unsupervised learning

- Unsupervised learning can be considered as learning a mapping between spaces X, Y without an explicit pairing in the latter.
- Whereas in supervised learning the pair $(f^{(i)}, y^{(i)})$ is assumed drawn from the joint Probability Density Function $P(f, y)$, in unsupervised learning we only assume a *marginalised* density $P(f)$ and/or $P(y)$
- If one of the spaces is a *Latent Space* Z then the mapped data lie on a manifold which may be lower dimensional and simpler in geometry then its corresponding description in data space.
- The mapping $D : X \rightarrow Z$ (conversely $Y \rightarrow Z$) is called a *discriminator*. A well known example is **k-means** which assigns each data point to a cluster by maximising between class variance and minimising within class variance.
- The mapping $G : Z \rightarrow X$ (conversely $Z \rightarrow Y$) is called a *generator*. It takes samples from a latent space manifold and produces representitive samples in data space

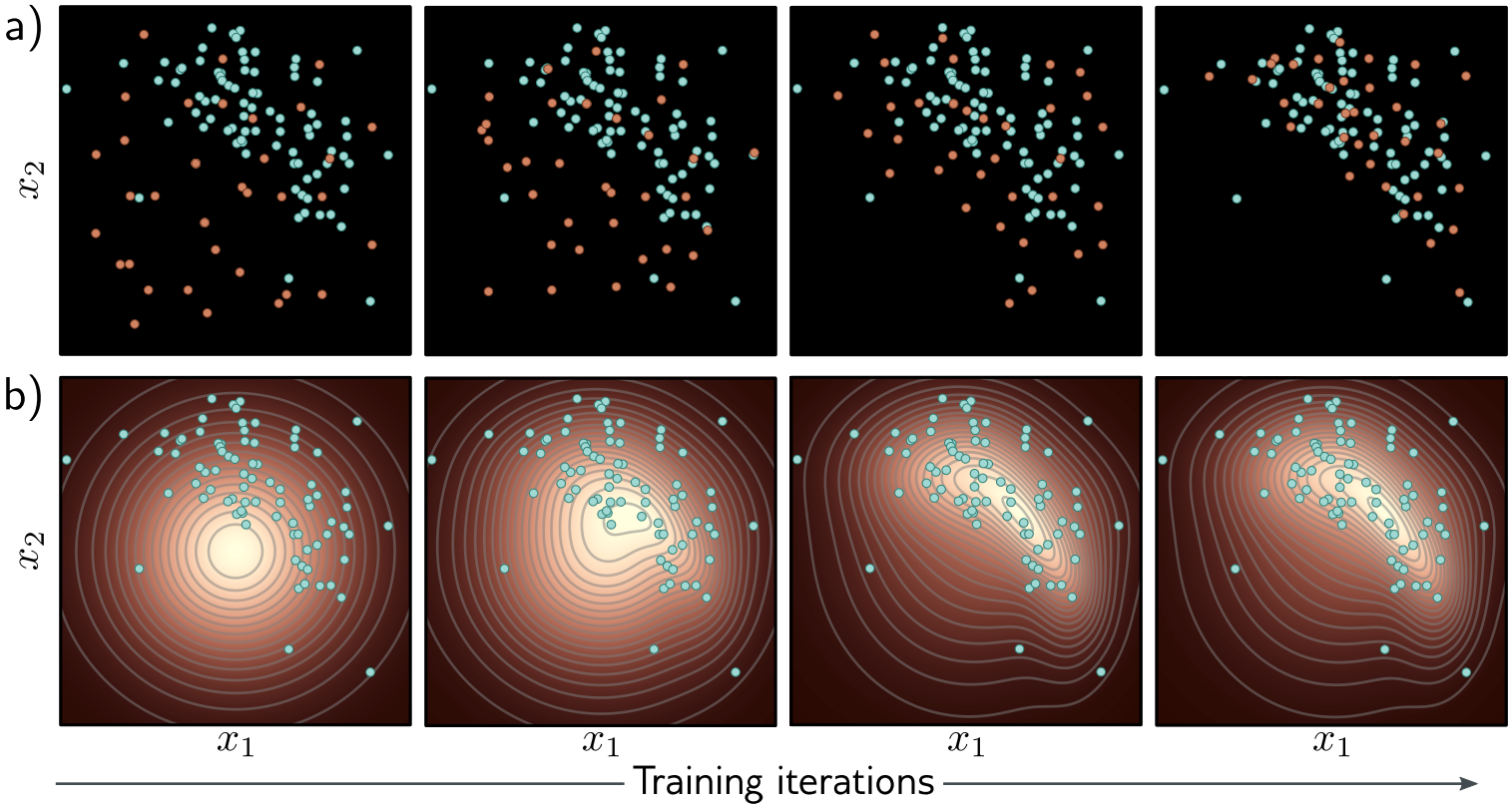
- Because the latent space Z is a more compact and "simpler" representation of the manifold of the data or image distribution, it is natural to think of applying this concept within a reconstruction scheme, eg.

$$\frac{1}{2}\|g - \mathcal{A}\,G(z)\|^2 + \alpha\Psi(z) \rightarrow \min\,, \quad f^\dagger = G(z^\dagger)$$

- The success of these methods depend on how smoothly the distribution $P(f)$ depends on the distribution of $P(z)$
- Generative Adversarial Networks (GANs) learn to generate data examples f from latent variables z, using a loss that encourages the generated examples to be indistinguishable from real examples. In practice they suffer from "mode collapse" which means the learned $P(X)$ tends to be monomodal and a poor representation of the true underlying distribution.
- Probabilistic Generative models learned a parameterised distribution $P(f|\theta)$. Examples include variational autoencoders, normaliszing flow, and score based diffusion models.

Introduction

Generative Models in Inverse Problems



From : [S. Prince "Understanding Deep Learning" 2023[1]] a) During training (left to right), the GAN loss function encourages these samples to become progressively less distinguishable from real data. b) Probabilistic models (including variational autoencoders, normalizing flows, and diffusion models) learn a probability distribution over the training data (left to right).

[1] http://udlbook.com

Outline

Generative Adversarial Networks

Introduction

- Generative Adversarial Networks (GANs) consists of a Generator and Discriminator trained simultaneously.
- The Generator will produce fake samples that will try to fit the distribution of the training dataset. The Discriminator will then produce a probability that a given sample is in the distribution of the training dataset.
- The loss function of the GAN is

$$\min_G \max_D \mathrm{L}(G, D) = \mathbb{E}_x[\log(D(x))] + \mathbb{E}_z[\log[1 - D(G(z))]] \tag{1}$$

- The goal of the Discriminator D is the maximise the probability of assigning the correct label to the training data and samples generated from G. Whereas the goal of the Generator G is to trick the Discriminator into labelling the generated samples with real labels, minimising the Discriminator probability in correctly labelling generated samples and the training data.

Autoencoders

Dimension Reduction : PCA

Let's recall a basic technique in dimension reduction based on *Principle Component Analysis*

- Consider a problem with data $g \in \mathbb{R}^N$. Let X be a $N \times M$ *data matrix* with M training samples each a vector length N.
- Taking the SVD of $\mathrm{X} \rightarrow \mathrm{UWV}$, we project the data into a lower dimensional space by choosing the P highest singular values :

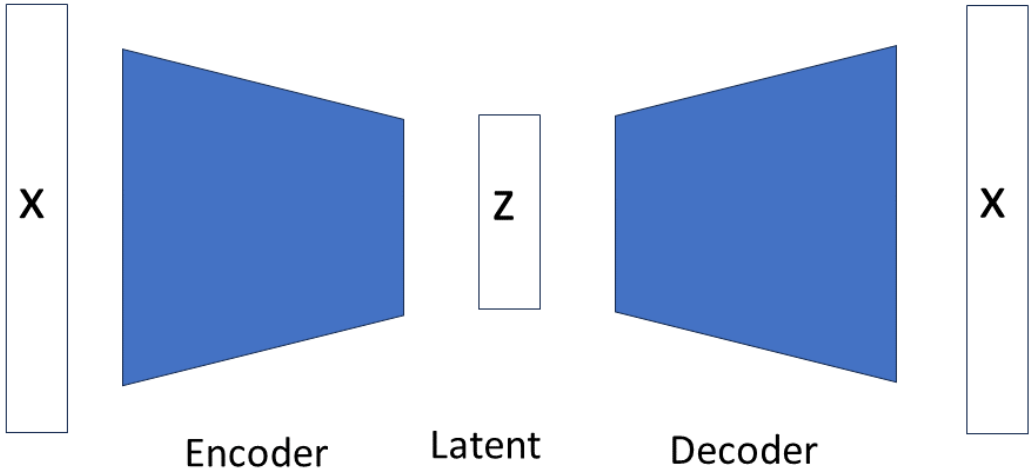
$$\tilde{\mathrm{X}} = \tilde{\mathrm{U}}\tilde{\mathrm{W}}\tilde{\mathrm{V}}$$

- We choose P by looking at the Frobenhuis norm Loss $\mathrm{L} = \|\mathrm{X} - \tilde{\mathrm{X}}\|_F$ (PCA) or Nuclear norm Loss $\mathrm{L} = \|\mathrm{X} - \tilde{\mathrm{X}}\|_*$ (Robust PCA), and choosing the smallest P such that the loss falls below a threshold.
- **Note:** Since we only project in data space we can find the use the eigen decomposition of XX^{T} or the covariance around the mean $(\mathrm{X} - \mu)(\mathrm{X} - \mu)^{\mathrm{T}}$
- Then the mapping $f \rightarrow \tilde{\mathrm{U}}^{\mathrm{T}} f$ is an *encoder* $\mathbb{R}^N \rightarrow \mathbb{R}^P$ and the mapping $\tilde{f} \rightarrow \tilde{\mathrm{U}}\tilde{f}$ is a *decoder* $\mathbb{R}^P \rightarrow \mathbb{R}^N$.

Autoencoders

Learned Dimension Reducion

An autoencoder consists of a *learned* decoder and encoder $\Lambda_\theta = \mathcal{U}_{\theta_D}\mathcal{E}_{\theta_E}$



- The encoder/decoder networks are trained by minimising a loss e.g

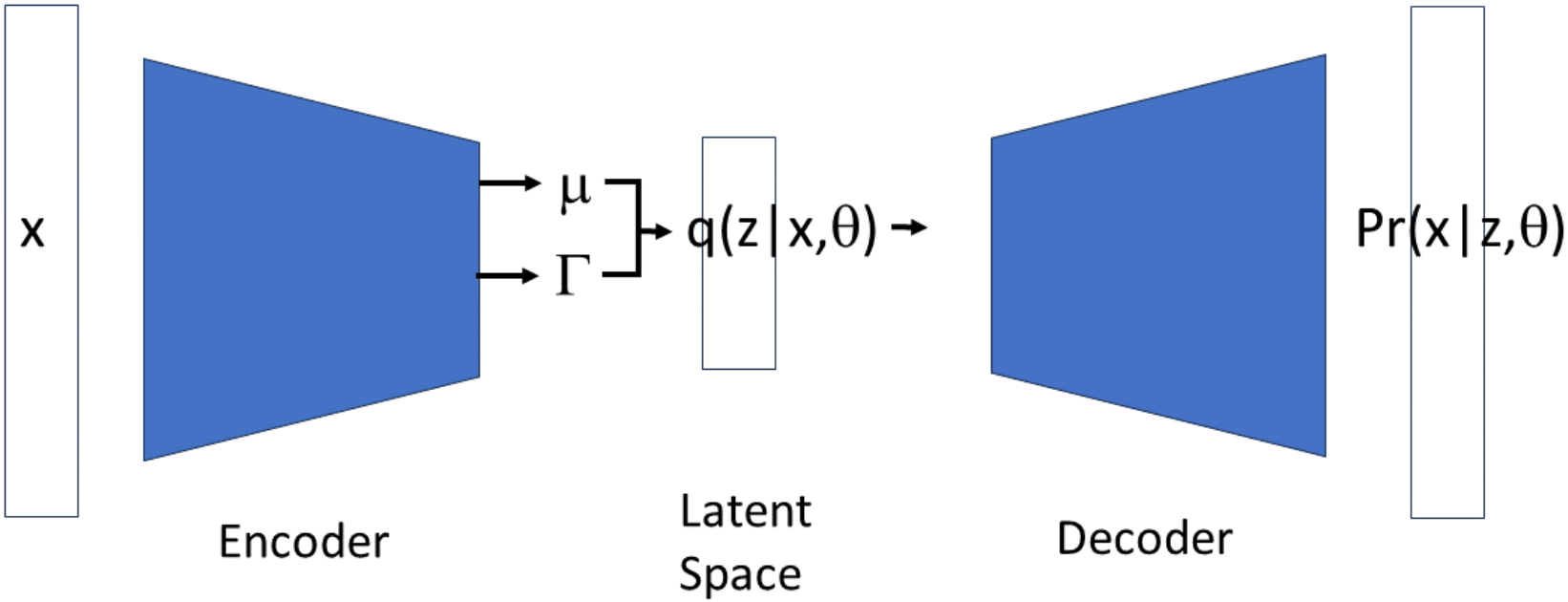
$$\mathrm{L} = \frac{1}{M}\sum_{i=1}^{M} \| f^{(i)} - \mathcal{U}_{\theta_D}\mathcal{E}_{\theta_E} f^{(i)} \|^2$$

  over the parameters $\{\theta_D, \theta_E\}$
- The decoder acts as a generator by inputing random samples.
- However, the structure of the latent space Z is unstructured and there is no particular control over the distribution of the samples obtained

Variational Autoencoders

VAE

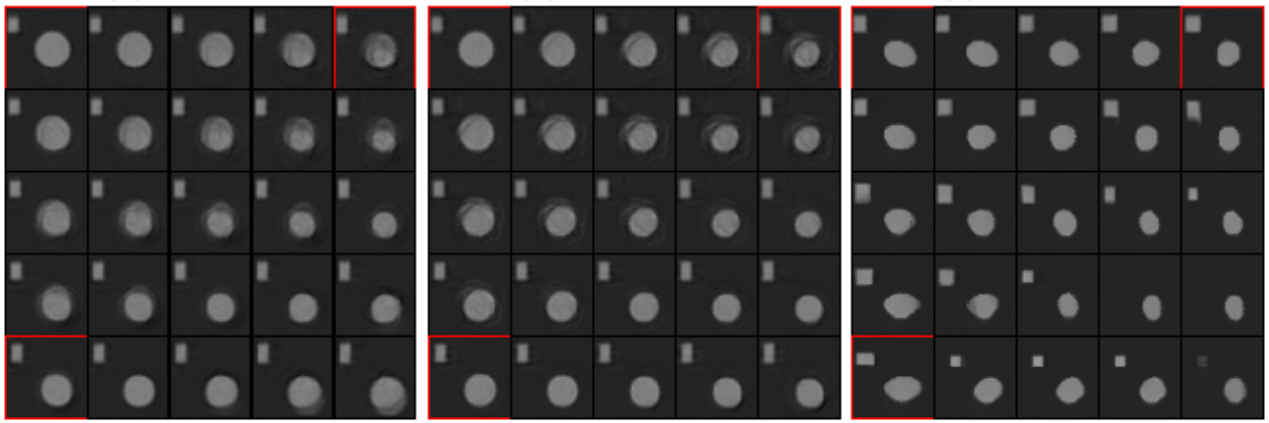
A *Variational Autoencode* (VAE) learns the mean and variance of the data distribution and encourges the encoded distributions to be close to a standard normal distribution



Generative Model Regularisation

Interpolation in Latent Space

(a) MNIST - AE (b) MNIST - VAE (c) MNIST - GAN



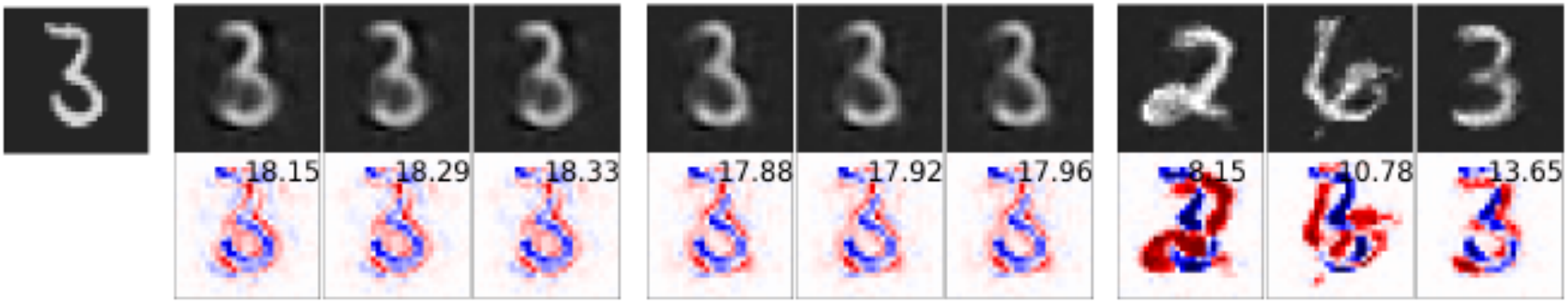
(d) Shapes - AE (e) Shapes - VAE (f) Shapes - GAN

Figure 6: Interpolation ability of an AE, VAE and GAN. The highlighted top left, bottom left and top right latent space values were chosen close to the test dataset and the other images are computed via linear combinations in the latent space.

From : [Duff, Campbell, Ehrhardt, 2022]

Generative Model Regularisation

Deconvolution

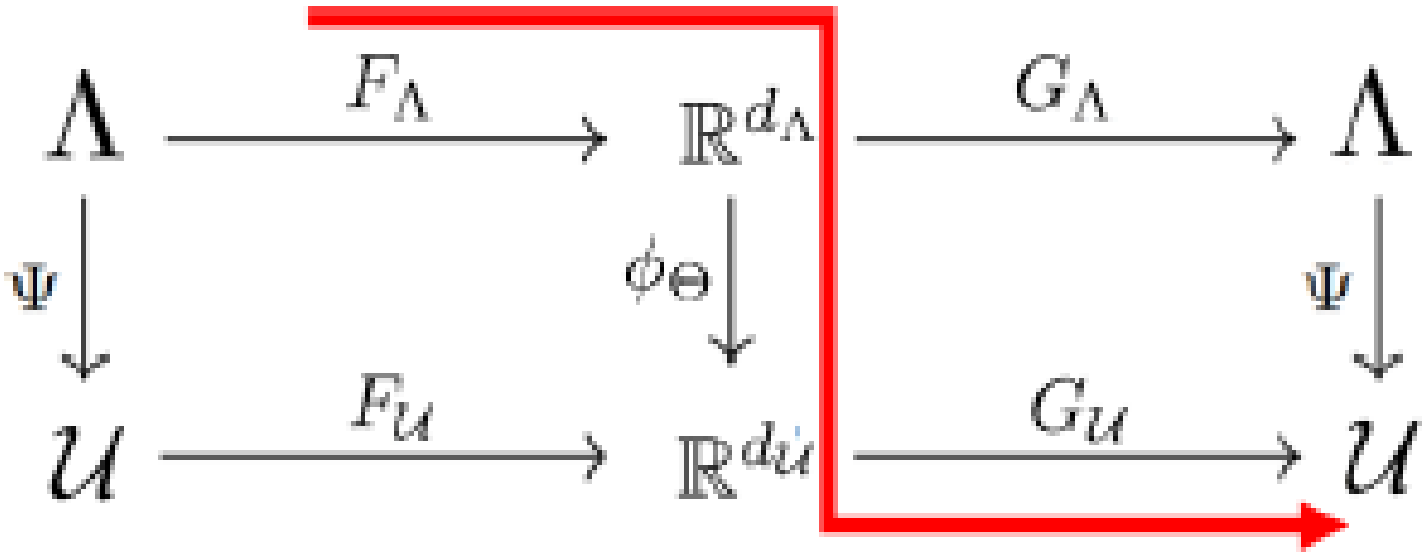


(a) GT (b) AE (c) VAE (d) GAN

Figure 8: Comparisons between the three generators, with eight-dimensional latent space, for the deconvolution problem. Reconstructions use the hard method. The plot shows 3 different initialisations for each generator. The ground truth (GT) is given on the left, the top line shows the reconstruction and the bottom line the residuals with the PSNR values.

From : [Duff, Campbell, Ehrhardt, 2022]

Learned SVD



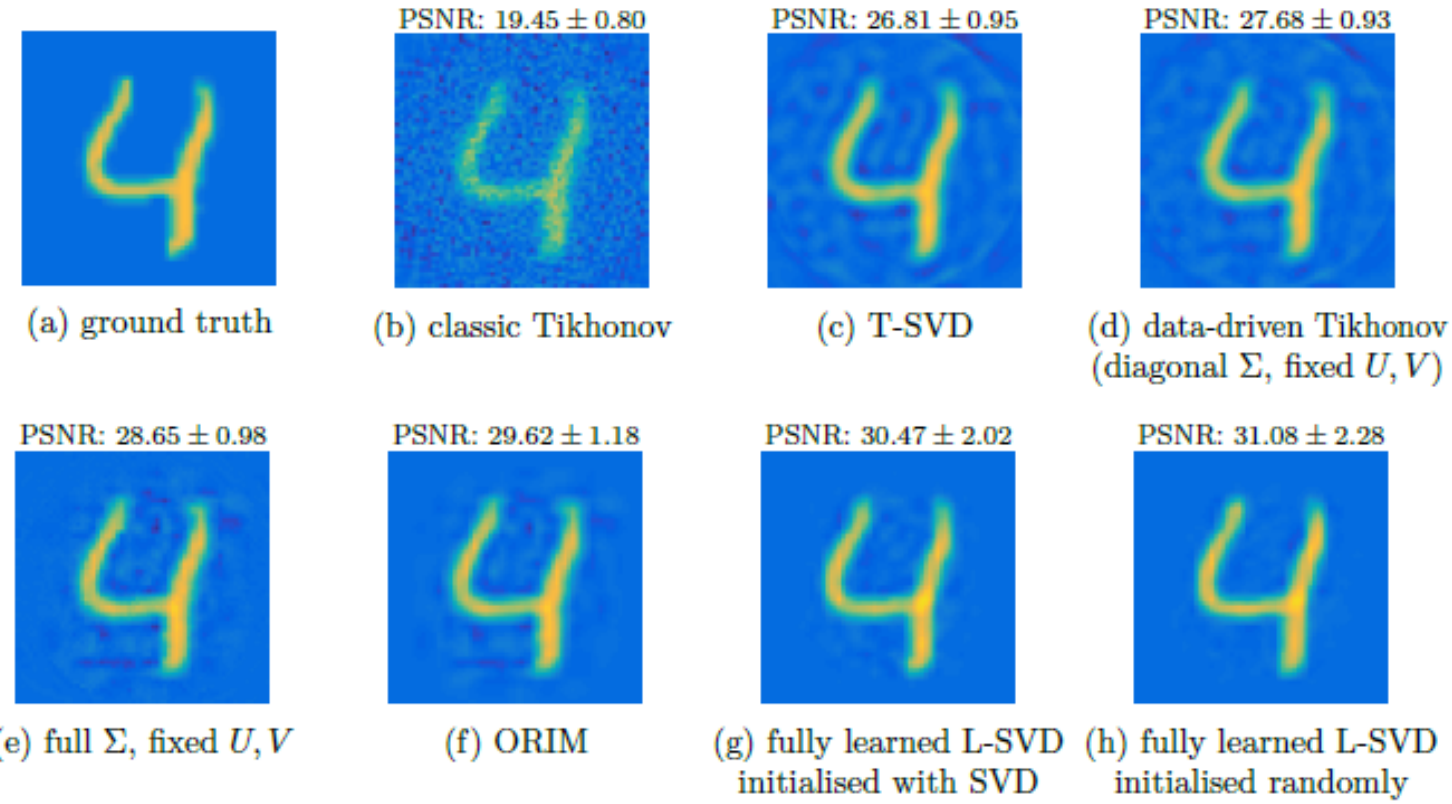
- Train autoencoders (unsupervised) seperately in X, Y.
- Train a mapping between reduced manifolds (supervised)
- Loss function

$$\text{Loss} = \sum_i \mathsf{L}_i(f^{(i)}, \mathcal{U}_x \mathcal{E}_y(g^{(i)})$$

Learned SVD

Results

Tomograhic reconstruction of MNIST images



(a) ground truth (b) classic Tikhonov (c) T-SVD (d) data-driven Tikhonov (diagonal Σ, fixed U, V)

(e) full Σ, fixed U, V (f) ORIM (g) fully learned L-SVD initialised with SVD (h) fully learned L-SVD initialised randomly

From : [Boink and Brune, 2020]

Outline

Score Based Diffusion

Introduction

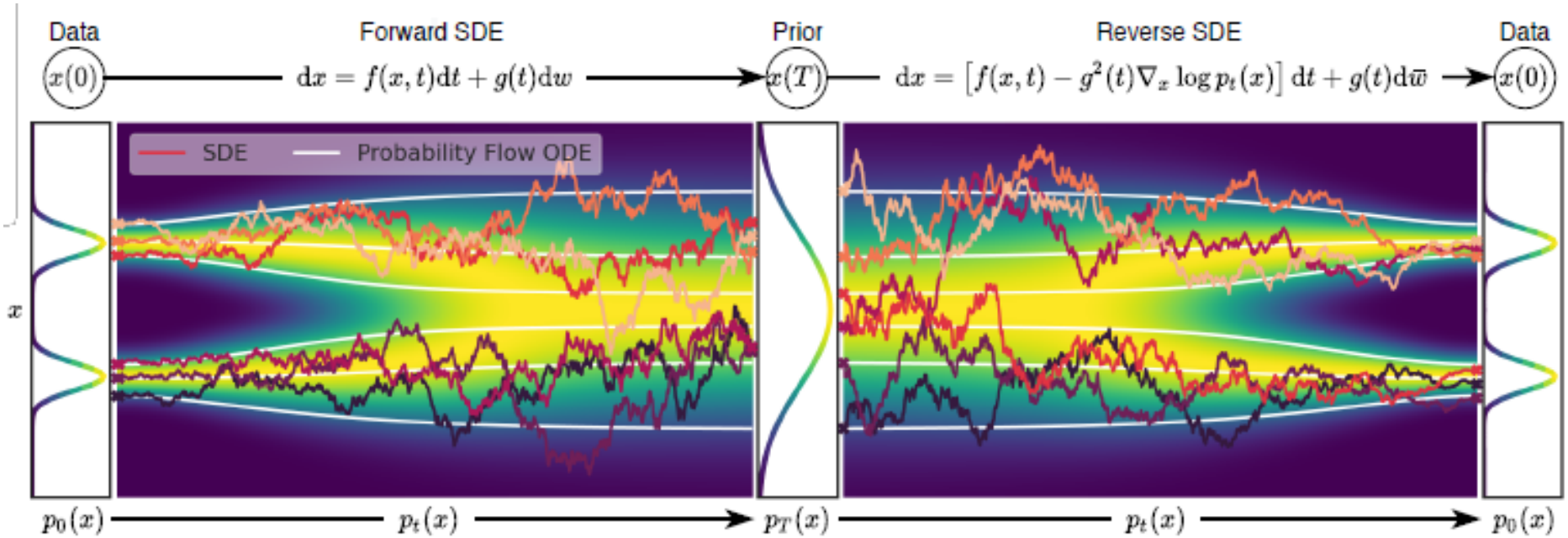
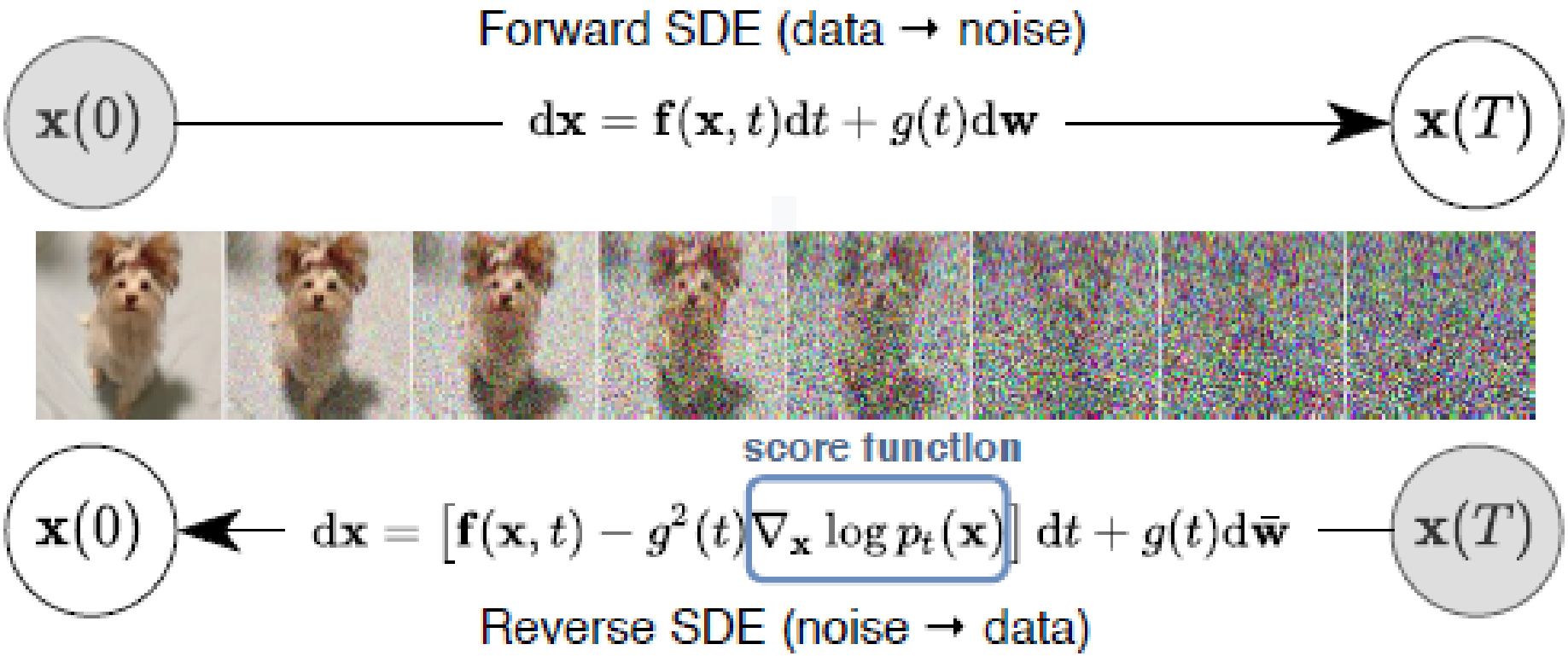


Figure 2: **Overview of score-based generative modeling through SDEs.** We can map data to a noise distribution (the prior) with an SDE (Section 3.1), and reverse this SDE for generative modeling (Section 3.2). We can also reverse the associated probability flow ODE (Section 4.3), which yields a deterministic process that samples from the same distribution as the SDE. Both the reverse-time SDE and probability flow ODE can be obtained by estimating the score $\nabla_{\mathbf{x}} \log p_t(\mathrm{x})$ (Section 3.3).

From : [Song et.al., ICLR 2021]

Score Based Diffusion

Introduction

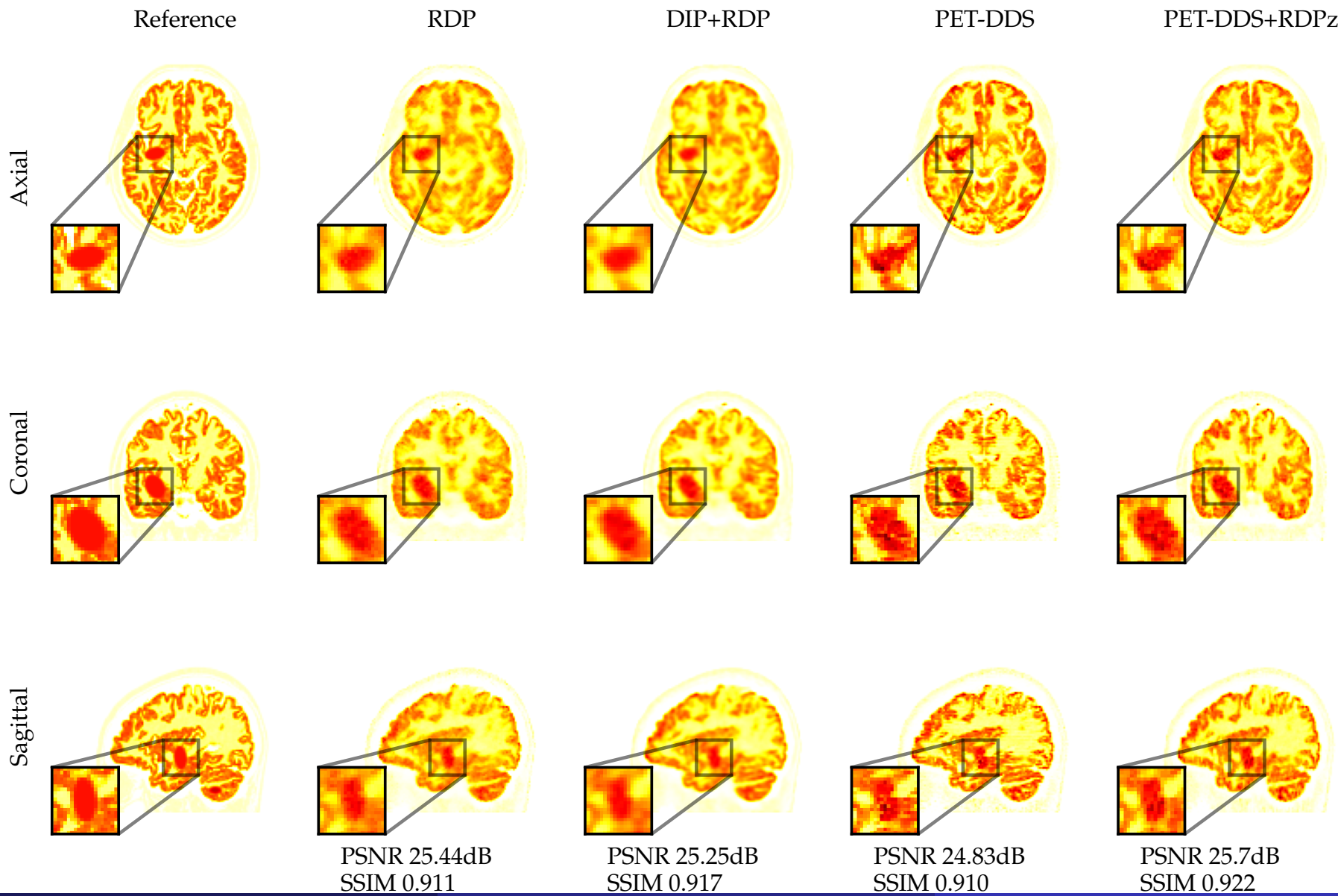


Solving a reverse-time SDE yields a score-based generative model. Transforming data to a simple noise distribution can be accomplished with a continuous-time SDE. This SDE can be reversed if we know the score of the distribution at each intermediate time step, $\nabla_x \log p_t(x)$

From : [Song et.al., ICLR 2021]

Score Based Diffusion

Results



Outline

Deep Image Prior

Introduction

The *Deep Image Prior* (DIP) is written as

$$\theta^{\dagger} = \arg\min_{\theta} \left[\frac{1}{2} \| g - \mathcal{A} F_{\theta} z_0 \|^2 + \alpha \Psi(F_{\theta}) \right], \qquad f^{\dagger} = F_{\theta^{\dagger}} z_0 \tag{2}$$

- The network F_{θ} takes the form of a *decoder* but in fact it is usually implemented as an image-to-image mapping typically Unet or another autoencoder
- The idea of DIP is to act as a solver for the (regulariszed) inverse problem $F_{\theta} : Z \mapsto X$ given a fixed arbitrary initisalisation z_0.
- Obviously, one way to do this would be if F_{θ} was an unrolled network such as LISTA; however this would be trained on many example data pairs $\{g^{(i)}, x^{(i)}\}$
- Instead, DIP solves for a single instantiation of the data.
- The architecture of F_{θ} should have useful properties that allow to reconstruct missing features that are in the null-space of $\mathcal{A}$.

Deep Image Prior

BackPropagation Revisited

Let's revisit back-propagation in the light of non-linear PDEs from lecture 1. Let's write a network as a composition of single layers with a final layer mapping to the range :

$$y = F_\theta z_0 = M \circ \Lambda_\theta z_0 = M \circ \Lambda_\theta^{(N)} \circ \Lambda_\theta^{(N-1)} \circ \cdots \circ \Lambda_\theta^{(1)} z_0$$

then $$\frac{\partial y}{\partial \theta_j} = M \circ \Lambda_\theta z_0 = M \circ \Lambda_\theta^{(N)} \circ \Lambda_\theta^{(N-1)} \circ \cdots \circ \frac{\partial \Lambda^{(j)}}{\partial \theta} \circ \cdots \circ \Lambda_\theta^{(1)} z_0$$

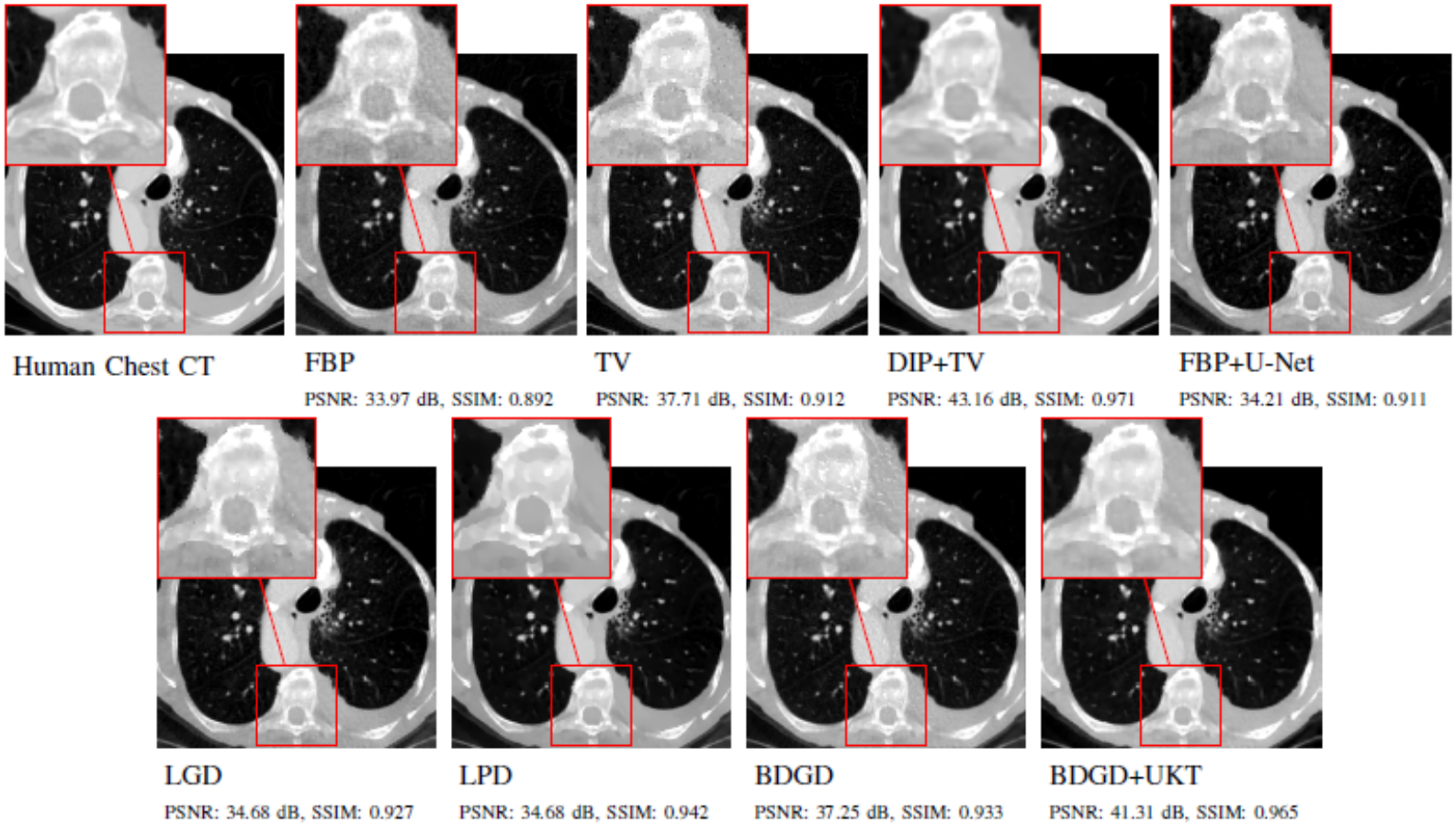
Now for a Loss function $\mathrm{L} = \frac{1}{2}\langle e, e\rangle$ where $e = g - F_\theta z_0$ is the data residual, we have

$$\begin{aligned}
\frac{\partial \mathrm{L}}{\partial \theta} &= -\left\langle e, \frac{\partial y}{\partial \theta} \right\rangle \\
&= -\sum_{j=1}^{N} \langle e, M \circ \Lambda_\theta z_0 \rangle = \left\langle e, M \circ \Lambda_\theta^{(N)} \circ \Lambda_\theta^{(N-1)} \circ \cdots \circ \frac{\partial \Lambda^{(j)}}{\partial \theta} \circ \cdots \circ \Lambda_\theta^{(1)} z_0 \right\rangle \\
&= -\sum_{j=1}^{N} \left\langle \underbrace{\Lambda_\theta^{(N-1-j)} \circ \cdots \circ \Lambda_\theta^{*(N-1)} \circ \Lambda_\theta^{*(N)} \circ M^* e}_{z^{*,N \to N-1-j}}, \frac{\partial \Lambda^{(j)}}{\partial \theta} \underbrace{\Lambda_\theta^{(j-1)} \circ \cdots \circ \Lambda_\theta^{(1)} z_0}_{z^{0 \to j-1}} \right\rangle
\end{aligned}$$

Deep Image Prior

Training

DIP can be *pretrained* in a supervised setting using $\mathcal{A}^* g^{(i)}$ rather than random noise z_0 [Barbano etal. 2020]



Results on low dose CT

Outline

Summary

- Unsupervised Learning and Dimensionality Reduction
- Generative Adversarial Networks
- Autoencoders, VAEs, Learned SVD
- Score Based Diffusion
- Deep Image Prior