Applications of Generative Models

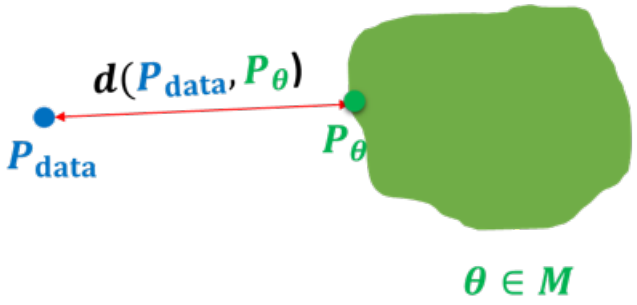
Stefano Ermon, Yang Song

Stanford University

Lecture 19

Generative models

$x_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$

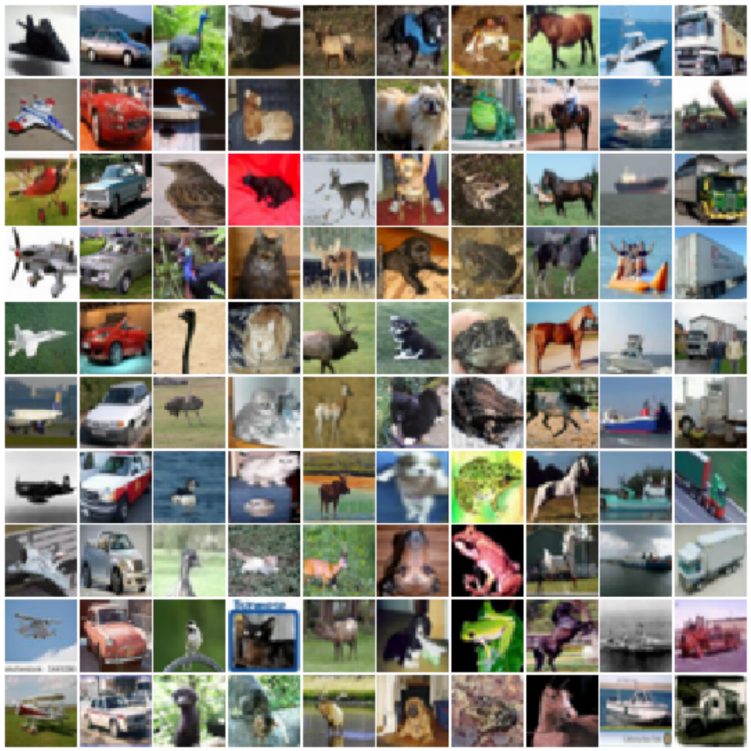


- Representing probability distributions
  - **Probability density/mass functions**: autoregressive models, flow models, variational autoencoders, energy-based models.
  - **Sampling process**: Generative adversarial networks.
  - **Score function**: Score-based generative models
- Distances between distributions
  - **Kullback-Leibler (KL) divergence**: maximum likelihood training.
  - f-**divergences** and **Wasserstein distance**: GANs.
  - **Fisher divergence**: score matching, denoising score matching, sliced score matching.
  - Noise contrastive estimation.

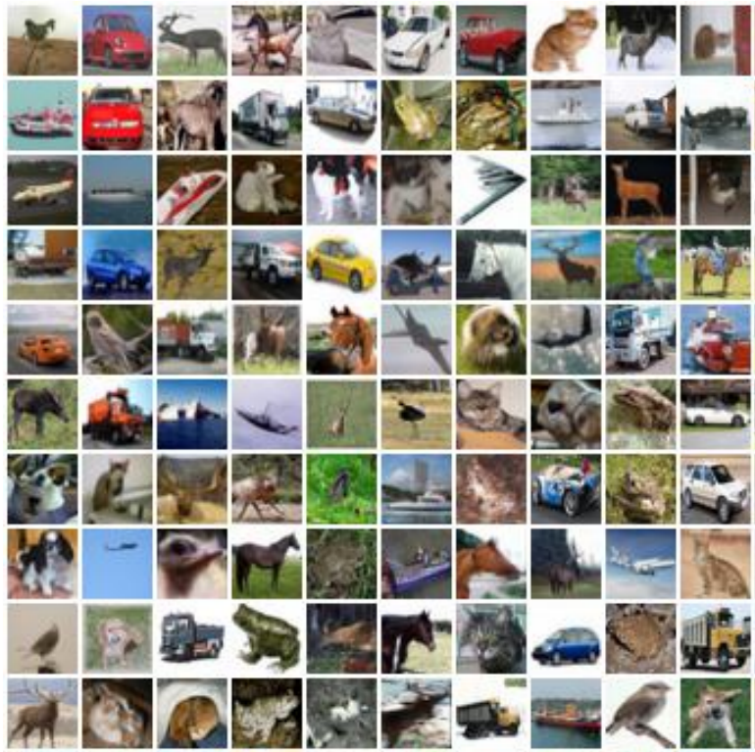
Applications

Why are generative models useful?

- Data generation



Training data



Samples (Score SDE)

Potential applications: Data augmentation, dataset synthesis, art creation, code generation, text generation, audio synthesis, etc.

Art creation

Metadata

Flaming Tarot: Death #2

SOLD OUT

Metadata

Flaming Tarot: Death #3

SOLD OUT

Metadata

Flaming Tarot: Death #4

SOLD OUT

Metadata

Flaming Tarot: The Lovers #1

SOLD OUT

Metadata

Flaming Tarot: The Lovers #2

SOLD OUT

Metadata

Flaming Tarot: The Lovers #3

SOLD OUT

Metadata

Flaming Tarot: The Lovers #4

SOLD OUT

Metadata

Landscape Tarot: Death #1

SOLD OUT

https://chainbreakers.kath.io/

- Synthesized by score-based generative models.
- Sold as non-fungible tokens (NFTs).

Text generation

(Prompt) Students need to learn generative models, because they are more powerful, they allow for better human-in-the-loop training of neural networks, they allow for better interpretability, and they allow for better generalization.

- Generated by GPT-3, an autoregressive model based on transformers.
- Text generation encompasses a multitude of other tasks! https://beta.openai.com/examples

Audio synthesis

Text-to-speech synthesis: WaveGrad (score-based model)
Lyrics-to-singing synthesis: Jukebox (VQ-VAE)

Text-to-image generation

Text-to-image generation: DALL-E (autoregressive model + VAE):

TEXT PROMPT an armchair in the shape of an avocado. . . .

AI-GENERATED IMAGES

Edit prompt or view more images↓

TEXT PROMPT a store front that has the word 'openai' written on it. . . .

AI-GENERATED IMAGES

Edit prompt or view more images↓

Text-to-image generation

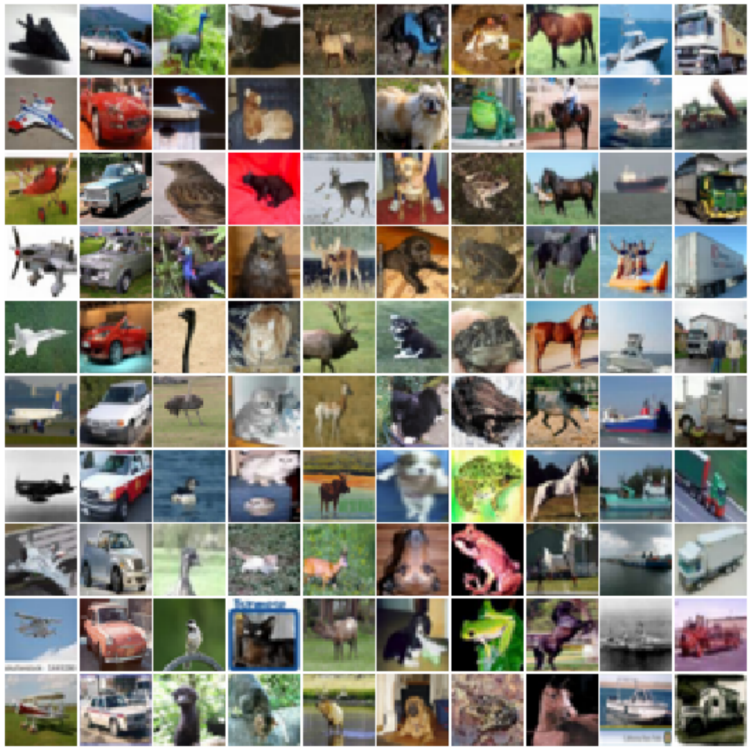
Text-to-image generation: CLIP-guided diffusion (score-based model):
(Prompt): Treehouse in the style of Studio Ghibli animation

image source: @danielrussruss
See also @RiversHaveWings

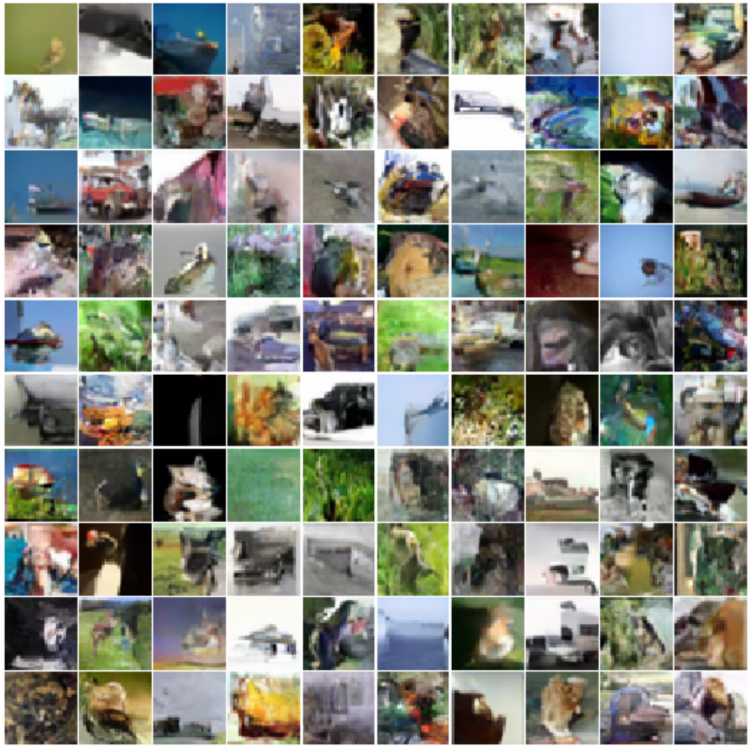
Applications

Why are generative models useful?

- Data generation



Training data



Samples (PixelCNN++)

Caveat: Humans are very sensitive to the sample quality.

Applications

Why are generative models useful?

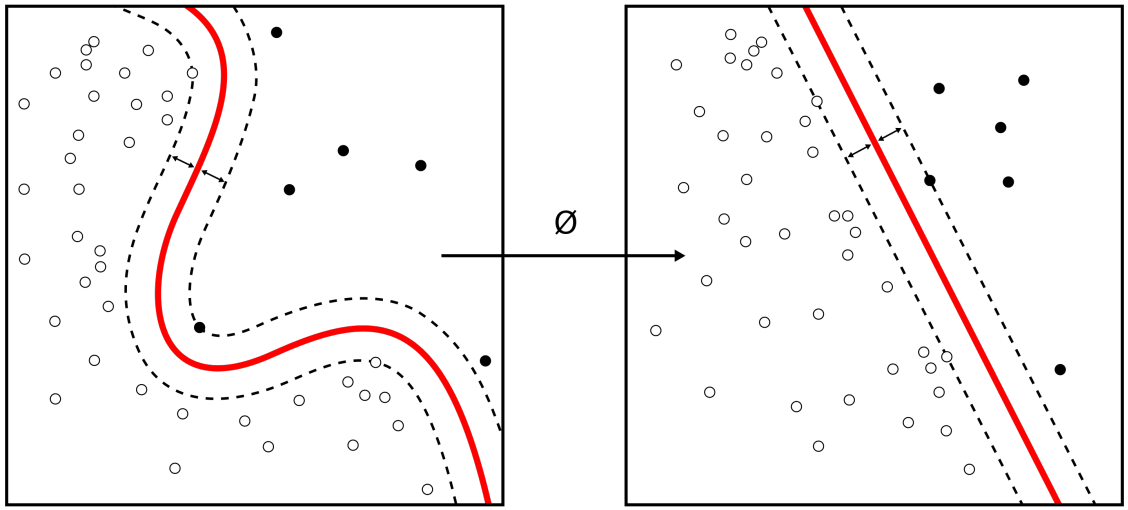
- Measuring the plausibility of data.

Examples of applications:

- Anomaly detection.
- Lossless compression.
- Classification.
- Solving inverse problems in science and engineering.

Anomaly detection

Detecting anomalous inputs or outliers in a dataset.



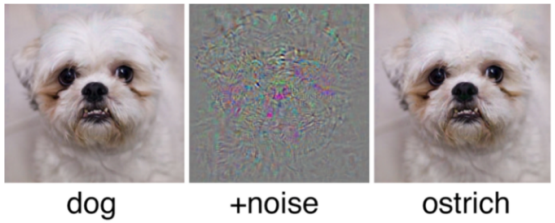
Suppose $\mathbf{x}$ is a normal datapoint, and $\mathbf{x}'$ is an outlier, it is natural to expect that

$$p_{\text{data}}(\mathbf{x}) > p_{\text{data}}(\mathbf{x}')$$

Available techniques: autoregressive models, normalizing flow models, variational autoencoders, energy-based models, score-based models (SDE).

Adversarial examples are anomalies

Adversarial examples are anomalies that fool classifiers but not humans.



Generating an adversarial example:

- $p_\phi(y \mid \mathbf{x})$ is a classifier. Suppose $\mathbf{x}_0$ is a normal image with label y_0.
- Find an adversarial noise perturbation $\mathbf{n}^*$ by solving

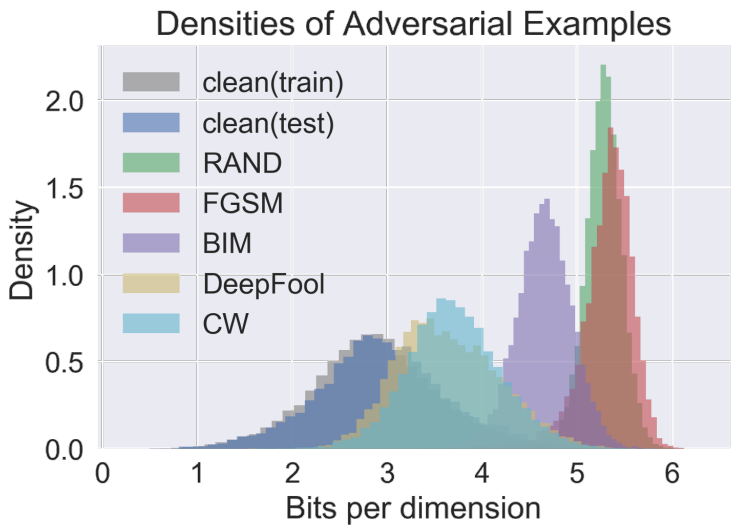
$$\mathbf{n}^* = \underset{\|\mathbf{n}\|_\infty < \epsilon}{\operatorname{argmin}} \log p_\phi(y_0 \mid \mathbf{x}_0 + \mathbf{n})$$

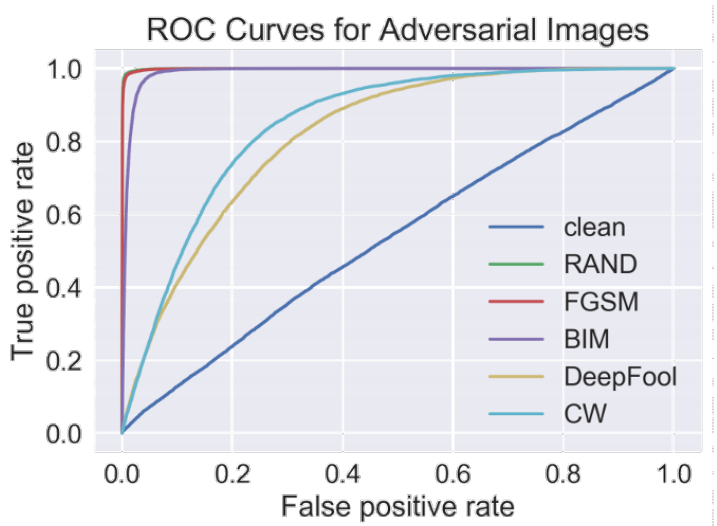
- Adversarial example: $\mathbf{x}_0' = \mathbf{x}_0 + \mathbf{n}^*$.

PixelDefend

PixelDefend (Song et. al. 2018): a generative approach to detecting and mitigating adversarial examples.

1. Train a PixelCNN model (a classical autoregressive model) $p_\theta(\mathbf{x})$ on normal images, such that $p_\theta(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$.
2. Choose a threshold η.
3. If $\log p_\theta(\mathbf{x}) < \eta$, $\mathbf{x}$ is likely to be an adversarial example; otherwise $\mathbf{x}$ is normal.





Randomly perturbed images are anomalies, but not adversarial examples!

A hypothesis testing framework for anomaly detection

Two-sample test:

- Training images $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N \overset{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$.
- A query image $\mathbf{x}' \sim q(\mathbf{x})$.
- Null hypothesis $H_0 : q = p_{\text{data}}$.
- Alternative hypothesis $H_1 : q \neq p_{\text{data}}$.
- If the null hypothesis holds, $\mathbf{x}'$ is not an anomaly; otherwise $\mathbf{x}'$ is likely to be an anomaly.

Goal: find a test statistic $T(\mathbf{x}', \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N)$ for rejecting the null hypothesis.

- Control the type-I error rate: $p(\text{reject} \mid H_0) < \alpha$.
- Maximize the test power: $p(\text{reject} \mid H_1)$.

PixelDefend test statistic (permutation test):

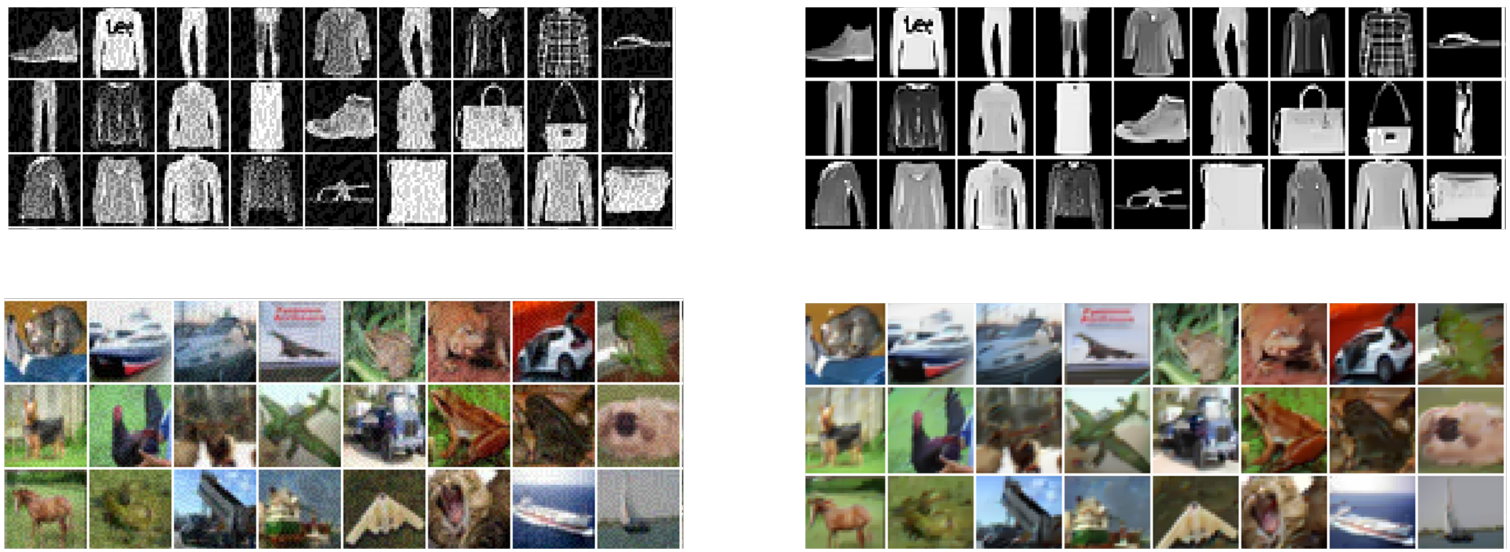
$$T(\mathbf{x}', \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N) = \frac{1}{N+1}\left(\sum_{i=1}^{N} \mathbb{I}[p_\theta(\mathbf{x}_i) \leq p_\theta(\mathbf{x}')] + 1\right)$$

Type-I error: $p(T \leq \alpha \mid H_0) = \alpha$, where $\alpha \in \{\frac{1}{N+1}, \frac{2}{N+1}, \cdots, 1\}$.

Derisking adversarial examples with PixelDefend

Purifying an adversarial example $\mathbf{x}'$:

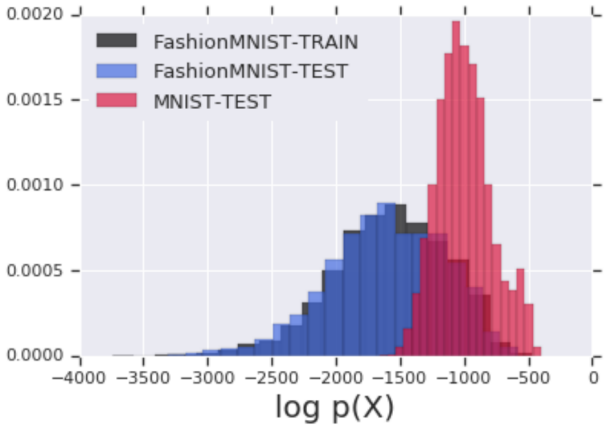
$$\mathbf{x}^* = \text{argmax } \|\mathbf{x} - \mathbf{x}'\|_\infty < \epsilon \log p_\theta(\mathbf{x}).$$



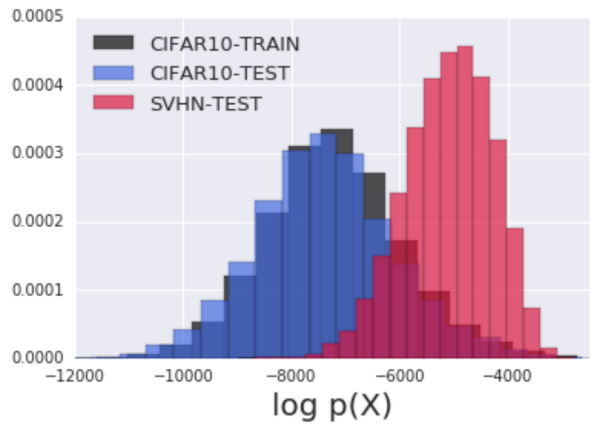
Adversarial images (left) and purified images after PixelDefend (right)

Counterexamples to generative anomaly detection

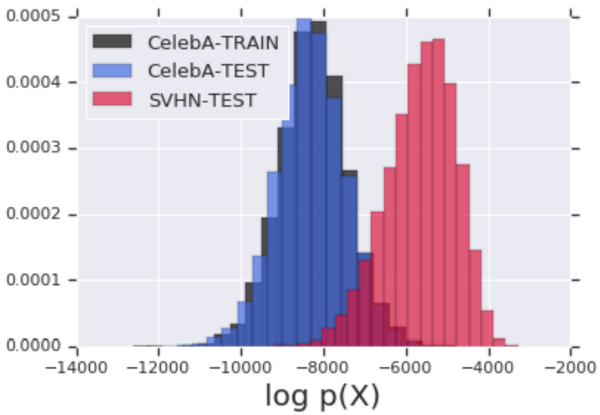
Do deep generative models know what they don't know? (Nalisnick et al. 2019)



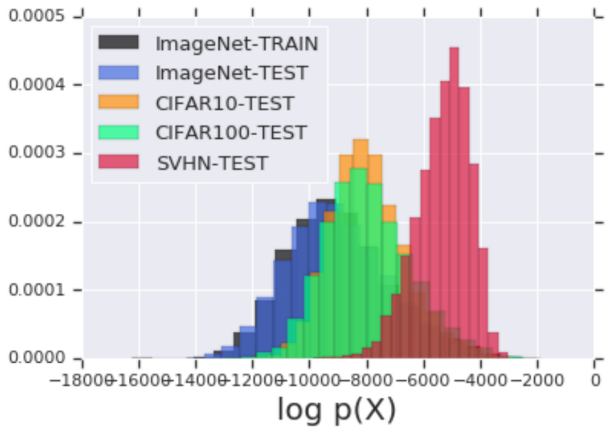
(a) Train on FashionMNIST, Test on MNIST



(b) Train on CIFAR-10, Test on SVHN



(c) Train on CelebA, Test on SVHN



(d) Train on ImageNet,
Test on CIFAR-10 / CIFAR-100 / SVHN

Lossless compression

- A string of characters is normally represented with a fixed number of bits for each character.
- We can use fewer bits to represent more frequent characters, and more bits for less frequent characters in order to compress data.

Theorem (Shannon's source coding theorem)

Suppose $x \in \mathcal{X}$ and $\mathcal{X}$ is a finite alphabet. For any coding function $C(x)$, the smallest possible expected code length is the entropy of x. That is,

$$E[|C(x)|] = \sum_{x \in \mathcal{X}} |C(x)| p_{data}(x) \geq -\sum_{x \in \mathcal{X}} p_{data}(x) \log_2 p_{data}(x) = H(x)$$

Given a generative model $p_\theta(x)$, we can find a code of length $-\log_2 p_\theta(x)$ for x. The compression rate is near optimal if $p_\theta(x) \approx p_{\text{data}}(x)$.
Entropy coding: Huffman coding, arithmetic coding, asymmetric numeral systems (ANS), etc.

Lossless compression with deep generative models

Sender and receiver share an autoregressive model $\prod_{i=1}^{n} p_\theta(x_i \mid x_{<i})$:

- Encode/decode $\mathbf{x}$ by going through $x_1, x_2, \cdots, x_n$ sequentially using an entropy coding method with $p_\theta(x_i \mid x_{<i})$.

Sender and receiver share a VAE with the decoder $p_\theta(\mathbf{x} \mid \mathbf{z})$, the encoder $q_\phi(\mathbf{z} \mid \mathbf{x})$, and the prior $p(\mathbf{z})$: **bits-back coding**.

- Assume the sender has some extra bits to communicate.
- The sender decodes these extra bits according to $q_\phi(\mathbf{z} \mid \mathbf{x})$, yielding a sample $\mathbf{z}$.
- The sender encodes both $\mathbf{z}$ and $\mathbf{x}$ according to $p(\mathbf{z})$ and $p_\theta(\mathbf{x} \mid \mathbf{z})$ respectively, and send the codes to the receiver.
- The receiver decodes $\mathbf{z}$ according to $p(\mathbf{z})$.
- The receiver decodes $\mathbf{x}$ according to $p_\theta(\mathbf{x} \mid \mathbf{z})$.
- The receiver encodes $\mathbf{z}$ with $q_\phi(\mathbf{z} \mid \mathbf{x})$ to obtain the extra bits.

Bits-back coding

- Assume the sender has some extra bits to communicate.
- The sender decodes these extra bits according to $q_\phi(\mathbf{z} \mid \mathbf{x})$, yielding a sample $\mathbf{z}$.
- The sender encodes both $\mathbf{z}$ and $\mathbf{x}$ according to $p(\mathbf{z})$ and $p_\theta(\mathbf{x} \mid \mathbf{z})$ respectively, and send the codes over.
- The receiver decodes $\mathbf{z}$ according to $p(\mathbf{z})$.
- The receiver decodes $\mathbf{x}$ according to $p_\theta(\mathbf{x} \mid \mathbf{z})$.
- The receiver encodes $\mathbf{z}$ with $q_\phi(\mathbf{z} \mid \mathbf{x})$ to recover the extra bits.

What's the effective code length on average?

$$E_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log p(\mathbf{z}) - \log p_\theta(\mathbf{x} \mid \mathbf{z}) + \log q_\phi(\mathbf{z} \mid \mathbf{x})]$$

$$= -E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})}\right] \geq -\log p_\theta(\mathbf{x}) = -\log \int p(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}) d\mathbf{z}$$

Effective code length is the negative evidence lower bound.

Classification

Given a conditional generative model $p_\theta(\mathbf{x})$, the Bayes' rule gives

$$p_\theta(\mathbf{y} \mid \mathbf{x}) \propto p_\theta(\mathbf{x} \mid \mathbf{y}) p(\mathbf{y}).$$

Model approach	Accuracy [%] ↑	NLL [bits/dim.] ↓
Invertible Network (Mackowiak et al. [27])	67.30	4.34
GLOW (Fetaya et al. [8])	84.00	3.53
Normalizing flow (Ardizzone et al. [1])	89.73	5.25
Energy model (Grathwohl et al. [13])	92.90	N/A
SBGC (ours)	**95.04**	**3.11**
WideResNet-28-12 (Targ et al. [44])	95.42	N/A

Score-Based Generative Classifiers (Zimmermann et al. 2021)

Solving inverse problems

Definition

An inverse problem seeks to recover an unknown signal $\mathbf{x}$ from an observation $\mathbf{y}$, given the forward model $p(\mathbf{y} \mid \mathbf{x})$.

Numerous applications in science and engineering:

- Image inpainting.
- Image superresolution.
- Image colorization.
- computed tomography (CT) and magnetic resonance imaging (MRI).
- geophysics, oceanography, astronomy, remote sensing, etc.

Compressed sensing

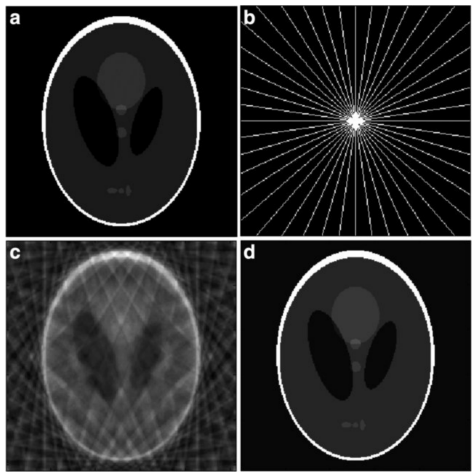
Problem definition of compressed sensing:

- $\mathbf{x} \in \mathbb{R}^n$.
- $\mathbf{y} \in \mathbb{R}^m$, $m < n$.
- $p(\mathbf{y} \mid \mathbf{x}) = \mathcal{N}(\mathbf{y} \mid A\mathbf{x}, \sigma^2 I)$, $A \in \mathbb{R}^{m \times n}$, σ is known.

Compressed sensing is an ill-posed inverse problem, and requires additional assumption on $\mathbf{x}$ to solve.

Traditional method assumes $\mathbf{x}$ is sparse, and solves the inverse problem via

$$\mathbf{x}^* = \text{argmin} \, \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1$$



Compressed sensing with generative models

Key idea: assume $\mathbf{x}$ is generated from a GAN model $g_\theta(\mathbf{z})$.

- $\mathbf{z}^* = \text{argmin}_{\mathbf{z}} \|\mathbf{y} - Ag_\theta(\mathbf{z})\|_2^2$.
- $\mathbf{x}^* = g_\theta(\mathbf{z}^*)$.

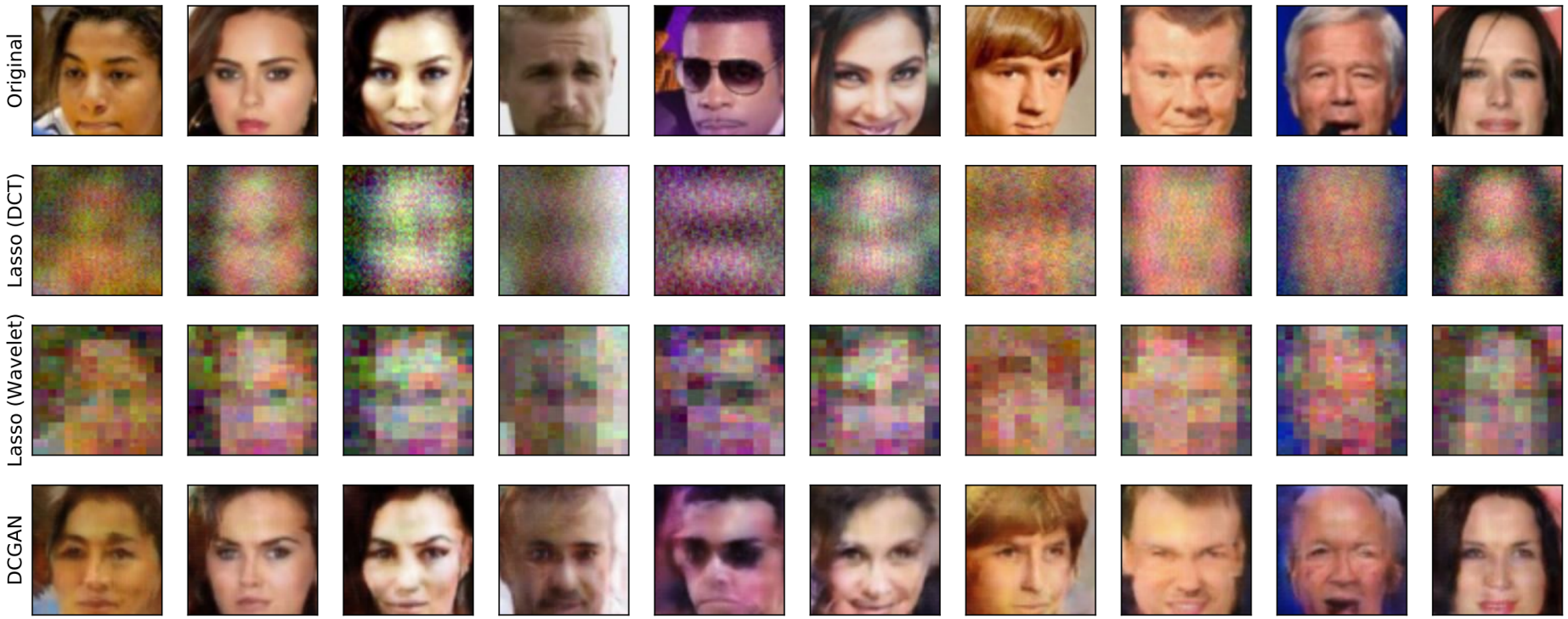


Image source: Bora et al. 2017

Inverse problem solving with score-based models

Key idea:

- Solve the inverse problem by sampling from the posterior $p(\mathbf{x} \mid \mathbf{y})$ with Langevin dynamics

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) + \sqrt{2\epsilon}\mathbf{z}^{(t)},$$

  where $\mathbf{z}^{(t)} \sim \mathcal{N}(0, I)$.
- Bayes' rule gives

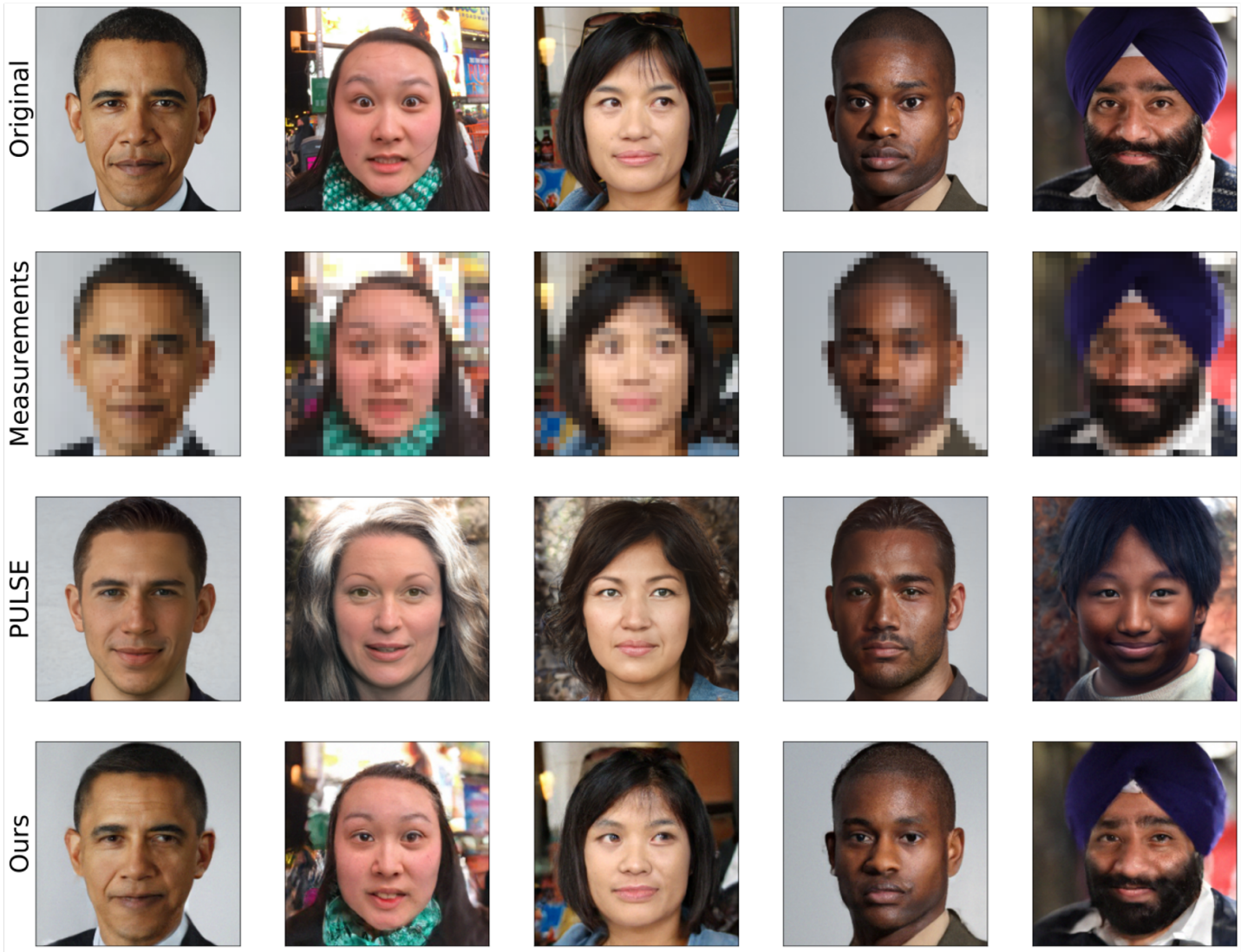
$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{x}).$$

- Train a generative model on data, $p_\theta(\mathbf{x}) \approx p(\mathbf{x})$.
- $\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) \approx \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$.
- With score-based generative models, we have

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) \approx s_\theta(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$$

.

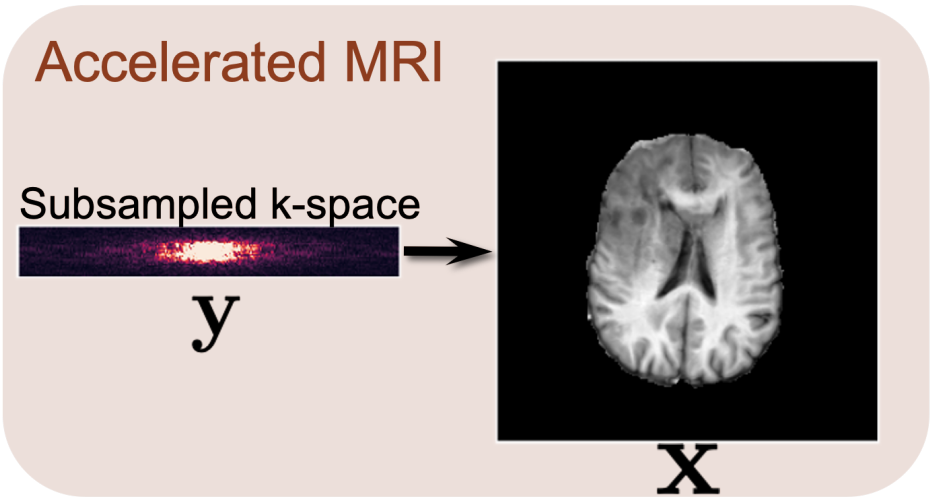
Inverse problem solving with score-based models

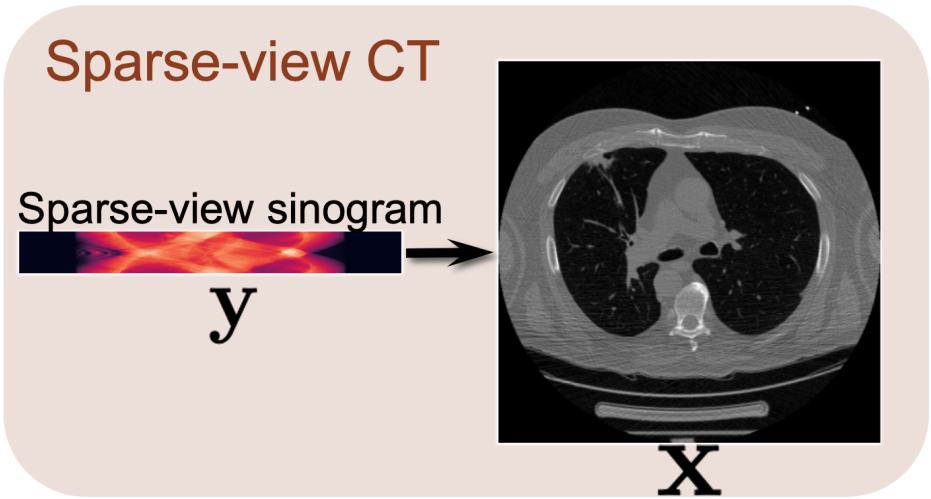


Jalal, Karmalkar, Hoffmann, Dimakis, Price, “Fairness for Image Generation with Uncertain Sensitive Attributes”, ICML 2021

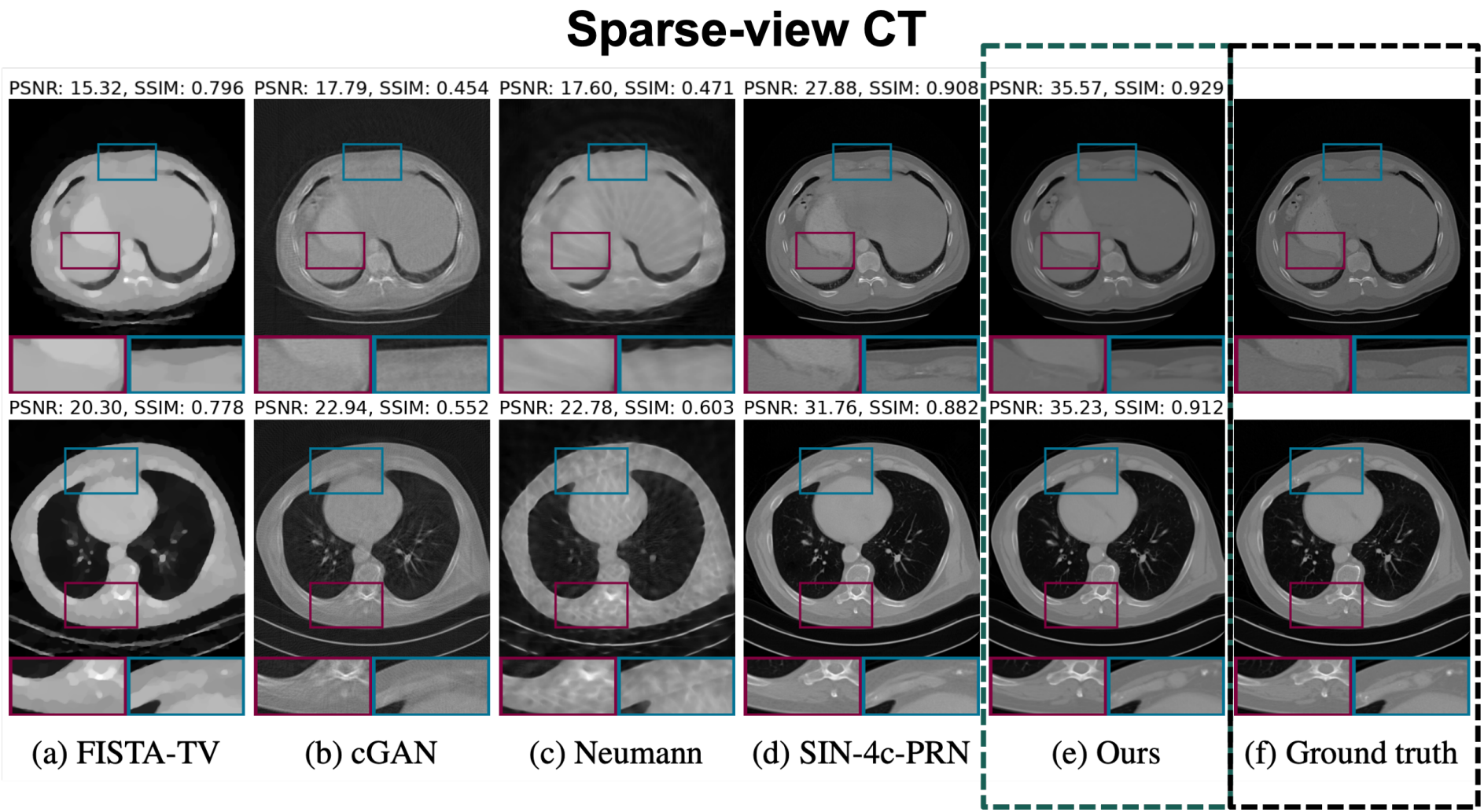
Inverse problem solving with score-based models

Sparse-view computed tomography (CT) and accelerated magnetic resonance imaging (MRI)





Inverse problem solving with score-based models



Score-based models beat bespoke deep learning competitors without requiring labeling information (Song et al., 2021).

Summary

Deep generative models have numerous applications:

- **Data generation**: images, texts, audio, molecules, graphs, material structures, text-to-speech, lyric-to-singing, text-to-image, trajectories in imitation learning (GAIL), etc.
- **Measuring the plausibility of data**:
  - Anomaly detection.
  - Compression.
  - Classification.
  - Solving inverse problems.

THE END.