# Score-based diffusion models in practice

## Implementations and results

**Agnimitra Dasgupta** and Assad A Oberai

Aerospace and Mechanical Engineering Department
University of Southern California

2023 Uncertainty Quantification Summer School
August 4, 2023

# Implementation of score-based generative models

<u>**Problem statement:**</u> Given realization $\left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{N_{\text{train}}} \sim P_{\text{data}}(\boldsymbol{x})$, generate new realizations of $\left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{N_{\text{new}}} \sim P_{\text{data}}(\boldsymbol{x})$.

- ▷ Score-based diffusion models generate samples using the score function $\nabla_{\boldsymbol{x}} \log P_{\text{data}}(\boldsymbol{x})$

Key references:

- Song & Ermon, *Generative modeling by estimating gradients of the data distribution*, NeuRIPs 2019.
- Song & Ermon, *Improved techniques for training score-based generative models*, NeuRIPs 2020.
- Github repository: `https://github.com/ermongroup/ncsnv2/tree/master`
- Lin et al., *Refinenet: Multi-path refinement networks for high-resolution semantic segmentation*, CVPR 2017.

# Outline

# Table of Contents

# Implementation of score-based diffusion networks

Denoising score matching:

$$L(\boldsymbol{\theta}, \{\sigma_i\}_{i=1}^{\ell}) = \sum_{i=1}^{\ell} \lambda(\sigma_i) L_i(\boldsymbol{\theta}; \sigma_i), \tag{1}$$

where

$$L_i(\boldsymbol{\theta}; \sigma_i) = \frac{1}{2} \mathbb{E}_{P_{\text{data}}(\boldsymbol{x})} \left[ \mathbb{E}_{\tilde{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{x}, \sigma_i^2)} \left[ \left\| \boldsymbol{s}(\tilde{\boldsymbol{x}}; \sigma_i) + \frac{\tilde{\boldsymbol{x}} - \boldsymbol{x}}{\sigma^2} \right\|_2^2 \right] \right] \tag{2}$$

- Sequence of noise scales $\sigma_1 > \sigma_2 > \cdots > \sigma_\ell > 0$
- In practice, $\|\boldsymbol{s}(\tilde{\boldsymbol{x}}; \sigma_i)\|_2 \propto 1/\sigma_i$ so choose $\lambda(\sigma_i) = \sigma_i^2$
- This ensures that the order of magnitude of $\lambda(\sigma_i) L_i(\boldsymbol{\theta}; \sigma_i)$ does not depend on the $\sigma_i$

# Implementation of score-based diffusion networks

Practical implementation of denoising score matching:

- Mini-batch of data: $\left\{\boldsymbol{x}^{(j)}\right\}_{j=1}^{N_{\text{batch}}}$
- Sample perturbation noise standard deviation $\left\{\sigma^{(j)}\right\}_{j=1}^{N_{\text{batch}}}$, where $\sigma^{(j)}$ are sampled uniformly from $\{\sigma_i\}_{i=1}^{\ell}$ with replacement
- Sample perturbations $\boldsymbol{\epsilon}^{(j)} = \sigma^{(j)}\boldsymbol{z}^{(j)}$, where $\boldsymbol{z}^{(j)} \sim \mathcal{N}(0, \mathbb{I})$
- Perturb the training data $\tilde{\boldsymbol{x}}^{(j)} = \boldsymbol{x}^{(j)} + \boldsymbol{\epsilon}^{(j)}$
- Evaluate score network $\boldsymbol{s}^{(j)} = \boldsymbol{s}(\tilde{\boldsymbol{x}}^{(j)}; \sigma^{(j)})$
- Compute $L_j = \frac{1}{2}\|\boldsymbol{s}^{(j)} - \boldsymbol{\epsilon}^{(j)}/\sigma^{(j)}\|_2^2$
- Mini-batch loss $L = \frac{1}{N_{\text{batch}}}\sum_{j=1}^{N_{\text{batch}}}\{\sigma^{(j)}\}^2 \cdot L_j$

# Table of Contents

# Implementation of score-based diffusion networks

**Algorithm 1** Sampling score-based diffusion models using annealed Langevin dynamics

**Require:** Noise scales $\sigma_1 > \sigma_2 \ldots > \sigma_\ell$,
step size $\epsilon_{\mathrm{s}}$, total number of steps $T$,
score function $\boldsymbol{s}(\boldsymbol{x}, \sigma; \boldsymbol{\theta})$

```
Initialize x_0
for i = 1 to ℓ do
    α_i = ε_s σ_i^2 / σ_ℓ^2
    for t = 1 to T do
        Sample z_t ~ N(0, I)
        x_t = x_{t-1} + α_i s(x_{t-1}, σ_i; θ) + sqrt(2 α_i) z_t
    end for
end for
return x_T = x_T + σ_T^2 s(x_T, σ_ℓ; θ) {Denoising step}
```

Initialize $\boldsymbol{x}_0$

**for** $i = 1$ to $\ell$ **do**

  $\alpha_i = \epsilon_{\mathrm{s}} \sigma_i^2 / \sigma_\ell^2$

  **for** $t = 1$ to $T$ **do**

    Sample $\boldsymbol{z}_t \sim \mathcal{N}(0, \mathbb{I})$

    $\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \alpha_i \boldsymbol{s}(\boldsymbol{x}_{t-1}, \sigma_i; \boldsymbol{\theta}) + \sqrt{2\alpha_i} \boldsymbol{z}_t$

  **end for**

**end for**

**return** $\boldsymbol{x}_T = \boldsymbol{x}_T + \sigma_T^2 \boldsymbol{s}(\boldsymbol{x}_T, \sigma_\ell; \boldsymbol{\theta})$ {Denoising step}

# Implementation of score-based diffusion networks

Important aspects:

- Building a good model for the score function that works across different noise scales.
- What are good choices for the noise scales $\{\sigma_i\}_{i=1}^{\ell}$?
- What are good choices for key algorithmic parameters: number of steps $T$ and step size $\epsilon_s$?

# Table of Contents

# Implementation of score-based diffusion networks

**Instance normalization++**: For the $k^{\text{th}}$ channel of each training data point $\boldsymbol{x}$ and the $i^{\text{th}}$ noise level

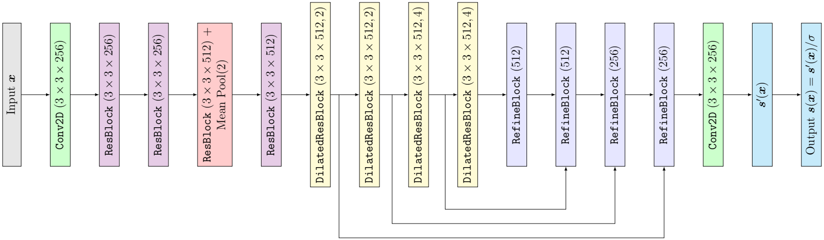
$$\boldsymbol{z}[:,k,:,:] \;=\; \underbrace{\boldsymbol{\gamma}_k \frac{\boldsymbol{x}[:,k,:,:]-\boldsymbol{\mu}[:,k]}{\boldsymbol{\vartheta}[:,k]} + \boldsymbol{\beta}_k}_{\text{Instance normalization}} \;+\; \underbrace{\boldsymbol{\alpha}_k \frac{\boldsymbol{\mu}[:,k]-\boldsymbol{m}}{\boldsymbol{v}}}_{\text{New addition!}} \tag{3}$$

where

- Mean intensity of the $k^{\text{th}}$ channel — $\boldsymbol{\mu}[:,k]$
- Std. dev. of the intensity of the $k^{\text{th}}$ channel — $\boldsymbol{\vartheta}[:,k]$
- Mean intensity across all channels $\boldsymbol{m} = Mean(\boldsymbol{\mu}[:,k])$
- Standard deviation across the mean of every channel $\boldsymbol{v} = \sqrt{Var(\boldsymbol{\mu}) + \varepsilon}$, where $\varepsilon << 1$
- Learnable parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathbb{R}^c$

# Implementation of score-based diffusion networks

## Model for the score network $\boldsymbol{s}(\boldsymbol{x}; \sigma_i)$




- Extensive use of dilated convolutions to capture long-range features.
- All `BatchNorm` is replaced with `InstanceNorm++`.
- Final output is simply scaled by $\sigma$, i.e, $\boldsymbol{s}(\boldsymbol{x}) = \boldsymbol{s}'(\boldsymbol{x})/\sigma$.
- Instead of concatenation (like U-Nets), Refine blocks sum up up-sampled features across resolution scales, and other things ...

# Table of Contents

# Implementation of score-based diffusion networks

## Techniques to improve training/performance

### Technique 1: Choosing the largest noise scale

Let the Gaussian mixture component arounf the $i^{\text{th}}$ data point $P^{(i)}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}^{(i)}, \sigma_1^2 \mathbb{I})$, then

$$\mathbb{E}_{P^{(i)}(\boldsymbol{x})}\left[\frac{P^{(j)}(\boldsymbol{x})}{\sum_j P^{(j)}(\boldsymbol{x})}\right] \leq \boxed{\frac{1}{2}\exp\left(-\frac{\|\boldsymbol{x}^{(i)}-\boldsymbol{x}^{(j)}\|_2^2}{8\sigma_1^2}\right)} \tag{4}$$

This should be large!

*Intuition:* The largest noise scale should smooth out any peaks such that particles can transition between modes.

- Choose $\sigma_1 \sim \mathcal{O}(\max_{i,j}\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|_2)$
- Empirical evidence suggests $\sigma_\ell = 0.01$ is imperceptible.
- With respect to data normalized between [0,1].

# Implementation of score-based diffusion networks

## Techniques to improve training/performance

**Technique 2: Selecting the number of noise scales**

*Intuition:* Samples from $P_{\sigma_{i-1}}(\tilde{\boldsymbol{x}}) = \int P_{\sigma_{i-1}}(\tilde{x}) P_{\text{data}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$ should generate enough samples from high-density regions of $P_{\sigma_i}(\boldsymbol{x})$

- In high-dimensional standard normal spaces, most of the probability density is concentrated around the '*important ring*' of radius $\sqrt{D}$.
- Let $\gamma_i = \sigma_{i-1}/\sigma_i$, then the overlap of the $\pm 3\sigma_{i-1}$ region of the distribution $\mathcal{N}(0, \sigma_{i-1}^2 \mathbb{I})$ with $\mathcal{N}(0, \sigma_i^2 \mathbb{I})$ is:

$$\boxed{\Phi(\sqrt{2D}(\gamma_i - 1) + 3\gamma_i) - \Phi(\sqrt{2D}(\gamma_i - 1) - 3\gamma_i)} \approx 0.5 \tag{5}$$

This shouldn't be small!

- With $\sigma_{i-1}/\sigma_i = \gamma$ choose $L$ that satisfies Eq. 5.

# Implementation of score-based diffusion networks

## Techniques to improve training/performance

**Technique 3: Tuning the total number of annealed steps $T$ and the step size $\epsilon_\mathrm{s}$**

*Intuition:* Particles at step $t = T$ should have variance close to $\sigma_i^2 \mathbb{I}$.
Given $\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \alpha s(\boldsymbol{x};\sigma_i) + \sqrt{2\alpha \boldsymbol{z}_t}$, $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{x}_0, s_T^2)$ and

$$\frac{s_T^2}{\sigma_i^2} = \left(1 - \frac{\epsilon_\mathrm{s}}{\sigma_\ell^2}\right)^{2T} \left(\gamma^2 - \frac{2\epsilon_\mathrm{s}}{\sigma_\ell^2 - \sigma_\ell^2\left(1 - \frac{\epsilon_\mathrm{s}}{\sigma_\ell^2}\right)}\right) + \frac{2\epsilon_\mathrm{s}}{\sigma_\ell^2 - \sigma_\ell^2\left(1 - \frac{\epsilon_\mathrm{s}}{\sigma_\ell^2}\right)} \tag{6}$$

- Choose $T$ depending on the computational budget.
- Choose $\epsilon_\mathrm{s}$ to make Eq. 6 close to 1.

# Implementation of score-based diffusion networks

## Techniques to improve training/performance

**Technique 4: Exponential moving average of network parameters**

After epoch $k$, let the updated weights be $\boldsymbol{\theta}_k$. Then, an independent copy $\boldsymbol{\theta}'$ is updated as

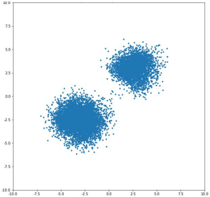
$$\boldsymbol{\theta}' = 0.999 \times \boldsymbol{\theta}' + 0.001 \times \boldsymbol{\theta}_k \tag{7}$$

- Exponential moving average is shown to improve stability of training and better visual quality of generated samples.

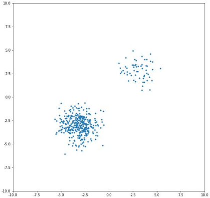
# Table of Contents

# Example 1: Bimodal Gaussian mixture model

Realizations from $P_{\text{data}}(\boldsymbol{x})$



Training data ($N_{\text{train}} = 400$)



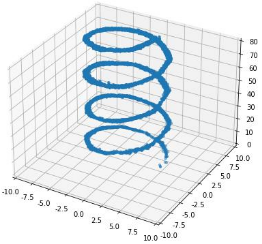
See toy2d.gif

Sample generation

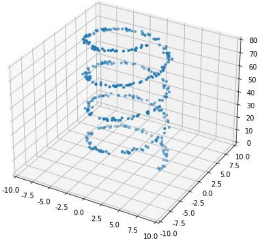
Parameters: $\sigma_1 = 10$, $\sigma_\ell = 0.001$, $\ell = 10$, Langevin steps $T = 100$, step size $\epsilon_s = 1 \times 10^{-6}$
Total training epochs — $10^4$, learning rate — 0.005, batch size — 128

# Example 2: 3-dimensional helix

Realizations from $P_{\text{data}}(\boldsymbol{x})$



Training data ($N_{\text{train}} = 400$)

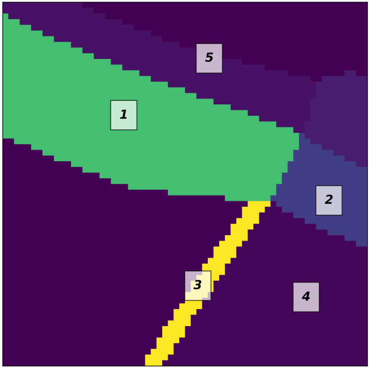


Sample generation

Parameters: $\sigma_1 = 50$, $\sigma_\ell = 0.001$,
$\ell = 10$, Langevin steps $T = 100$,
step size $\epsilon_s = 1 \times 10^{-6}$
Total training epochs — $10^5$,
learning rate — 0.001,
batch size — 128

# Table of Contents

# Example 3: Shear modulus distribution around the Optic Nerve Head (ONH)



;

Figure: Geometry of the optical nerve head. The figure shows (1) sclera, (2) lamina cribrosa, (3) pia matter, (4) optic nerve, and (5) retina

- Specimen size $\sim$ 1.75 mm $\times$ 1.75 mm
- The geometry is parameterized by 10 random variables
- Another 6 random variables control the distribution of the shear modulus in various regions.
- Total 16 random variables, which can be samples to generate different geometries.

# Example 3: Shear modulus distribution around the Optic Nerve Head (ONH)

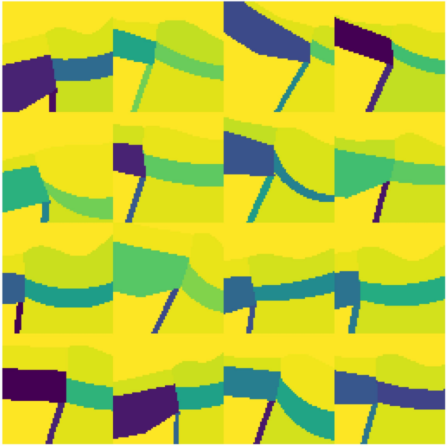


Figure: Training samples

- Synthetically generated 12000 training data points.
- Largest pairwise distance between data points is $\sim 22$ units after normalization.

# Example 3: Shear modulus distribution around the Optic Nerve Head (ONH)

Reverse diffusion process — sample generation

See ONH.gif

Parameters

- $\sigma_1 = 50$
- $\sigma_\ell = 0.01$
- $\ell = 256$
- Langevin steps $T = 100$
- Step size $\epsilon_s = 5.7 \times 10^{-6}$
- Total training epochs — $2 \times 10^5$
- Learning rate — 0.0001
- Batch size — 128

# Example 4: Micro-structure image of uniaxial CFRP

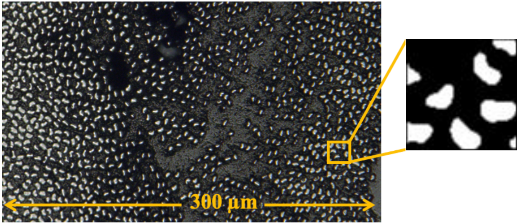



Figure: Optical microscope image of CFRP (NIST data).

Figure: Training samples

- Open-source data with 16000 training points.
- Largest pairwise distance between data points is $\sim 48.131$ units after normalization.

# Example 4: Micro-structure image of uniaxial CFRP

Reverse diffusion process — sample generation

See CFRP.gif

Parameters

- $\sigma_1 = 50$
- $\sigma_\ell = 0.01$
- $\ell = 256$
- Langevin steps $T = 100$
- Step size $\epsilon_s = 5.7 \times 10^{-6}$
- Total training epochs — $2 \times 10^5$
- Learning rate — 0.0001
- Batch size — 128

# Acknowledgments


- We gratefully acknowledge support from ....
- Javier Murgoitio Esandi and Harisankar Ramaswamy provided the ONH and CFRP data, respectively.


Code available at: `https://github.com/adasgupta94/USC_UQ_SummerSchool2023.git`

THANK YOU