Generative AI with Diffusion Models

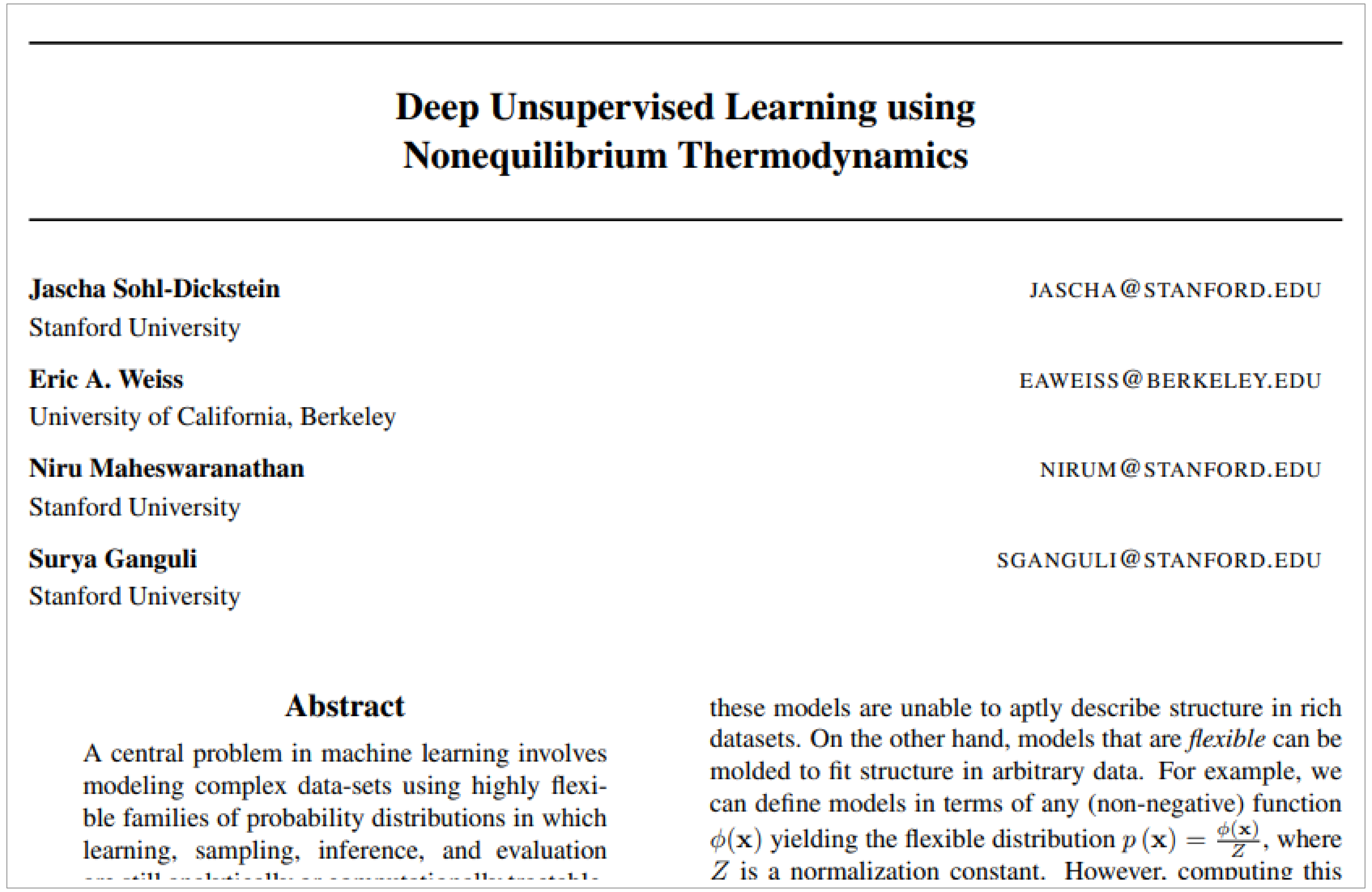
Part 2: Denoising Diffusion Probabilistic Models

Agenda

- Part 1: From U-Nets to Diffusion
- Part 2: Denoising Diffusion Probabilistic Models
- Part 3: Optimizations
- Part 4: Classifier-Free Diffusion Guidance
- Part 5: CLIP
- Part 6: Wrap-up & Assessment

Diffusion Inspiration

Thermodynamic Origins



Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation

these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this

Green food coloring dissolving in a glass of water

Thermodynamic Origins

Denoising Diffusion Probabilistic Models

Getting Lost in the Noise

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

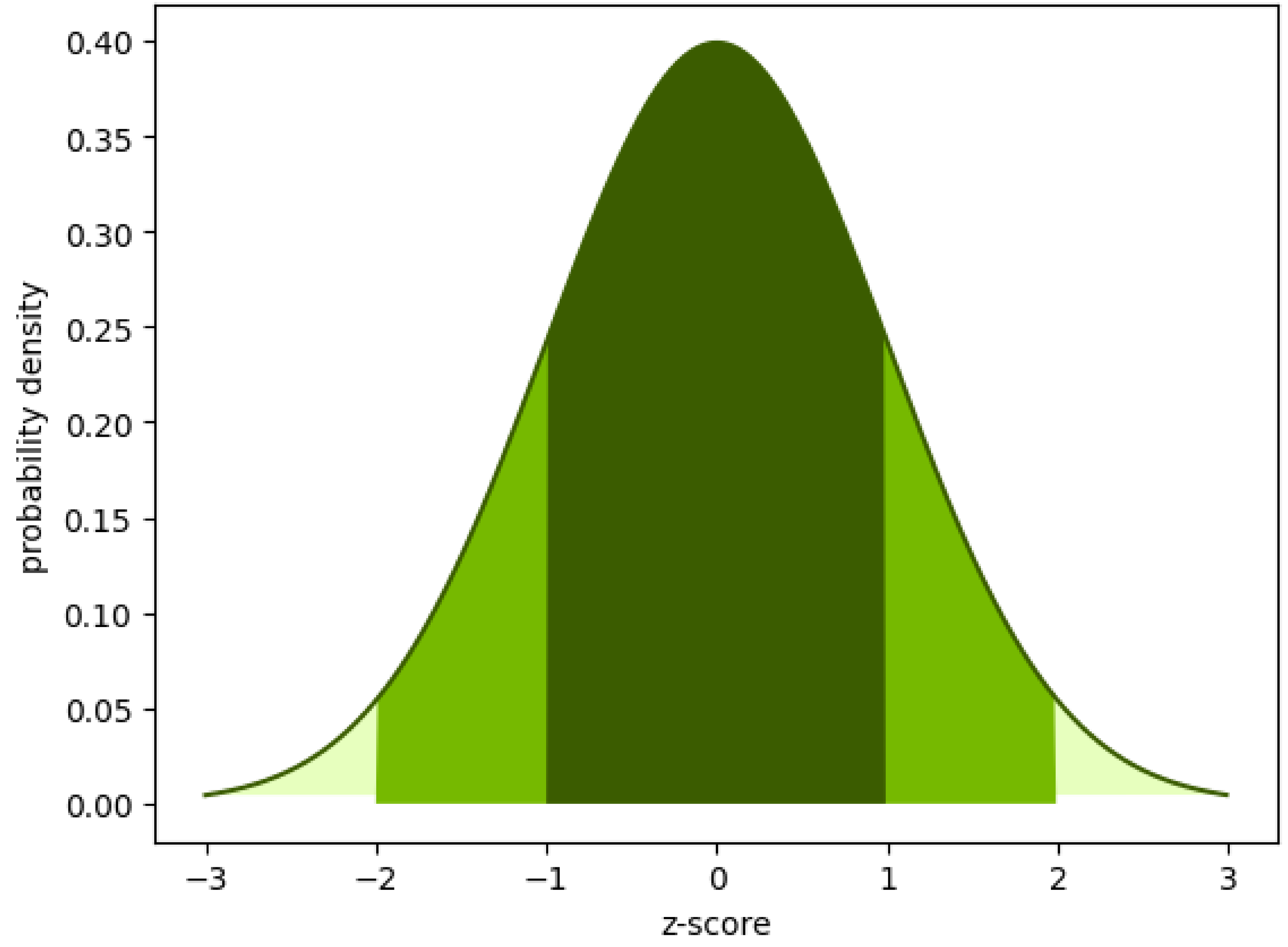
Abstract

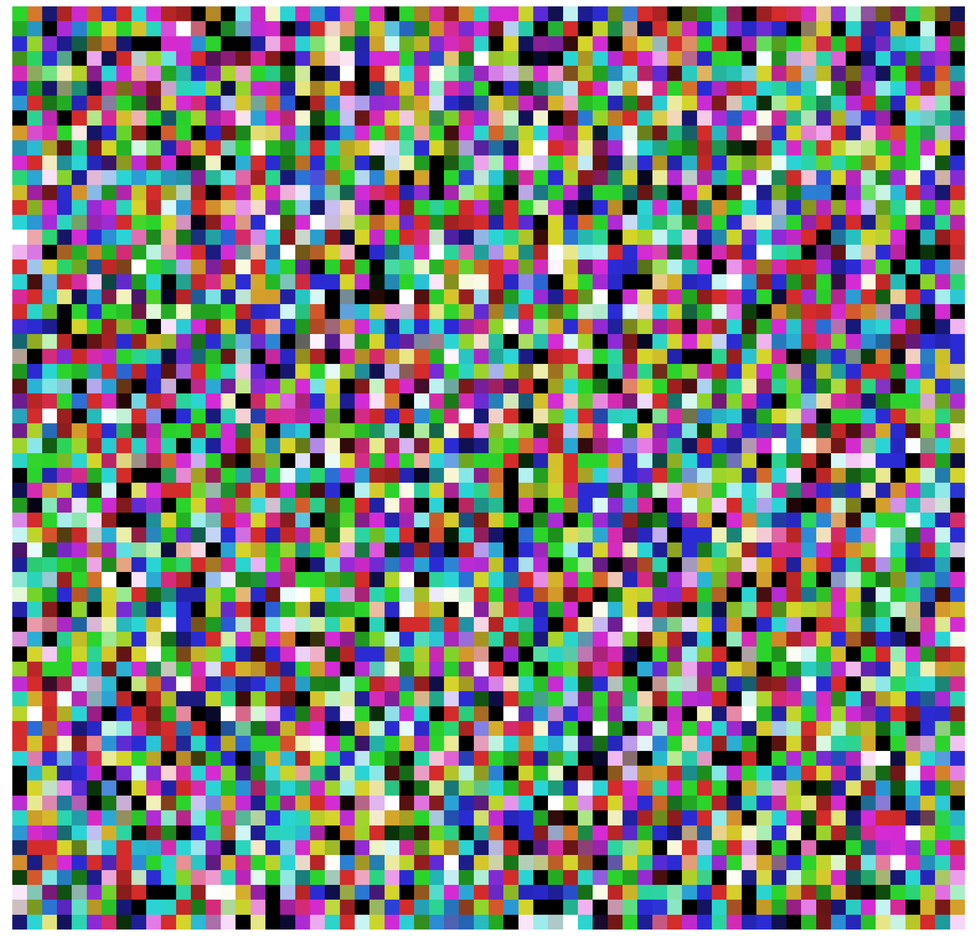
We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a

Forward Diffusion

Getting Lost in the Noise

Forward Diffusion





$$N(\mathrm{x}; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Getting Lost in the Noise

Forward Diffusion

$$\beta = [\underset{t=0}{0.0001}, \underset{t=1}{0.0023}, \underset{t=3}{0.0045}, \underset{t=4}{0.0067}, \ldots, \underset{t=10}{0.0200}]$$

$$T = 10$$

Total number of timesteps

$$\alpha = 1 - \beta$$

Convenient shorthand

Getting Lost in the Noise

Forward Diffusion

t = 0 t = 1 t = 2 t = 3 t = 4 t = 5 t = 6 t = 7 t = 8 t = 9 t = 10

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = N(\mathbf{x}_t; \sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1}, \beta_t \cdot \mathbf{I})$$

Code:

```
noise = torch.randn_like(x_t)
x_t = torch.sqrt(1 - B[t]) * x_t
  + torch.sqrt(B[t]) * noise
```

Step 1:

- Generate noise from a standard normal distribution
- Multiply result by $\sqrt{\beta_t}$

Step 2:

- Multiply image at previous step by $\sqrt{1-\beta_t}$
- Add it to the result from step 1

Reverse Diffusion

Getting Lost in the Noise

Reverse Diffusion

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = N(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

Difficult to calculate

Approximate average used to create $\mathbf{x}_t$

Getting Lost in the Noise

Reverse Diffusion

Can generate "evidence" of p using q

Getting Lost in the Noise

Reverse Diffusion

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = N(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = N(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \cdot \mathbf{I})$$

$$\tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t$$

Use Bayes' Rule and ...

Getting Lost in the Noise

Reverse Diffusion

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = N(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = N(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \cdot \mathbf{I})$$

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}}\boxed{\epsilon_t}\right)$$

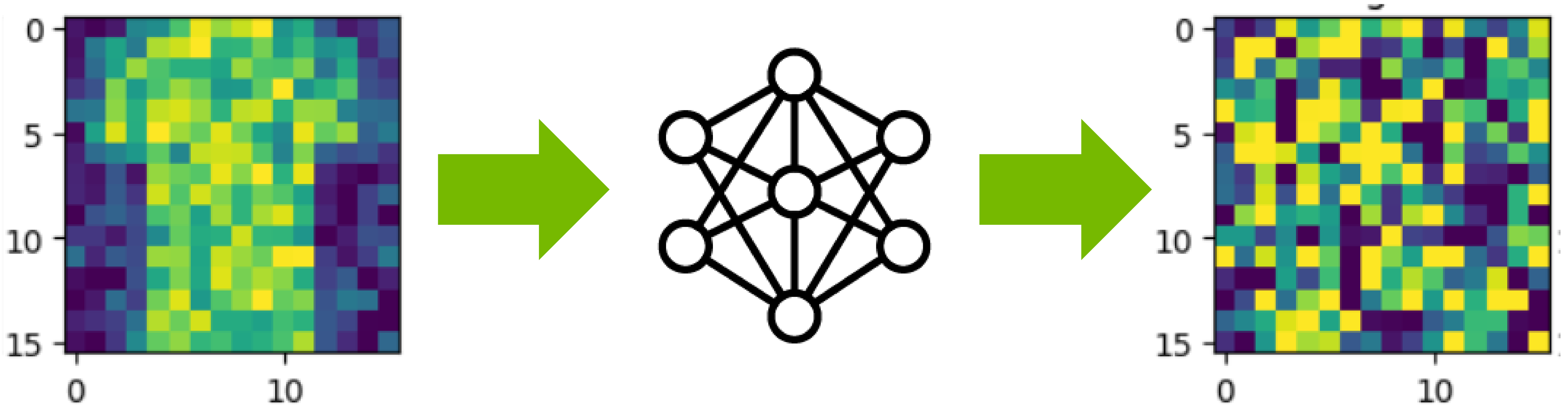
The noise added at time t
This is what the neural network will estimate

Model Training

Model Training

Training Data

Noisy image at time t

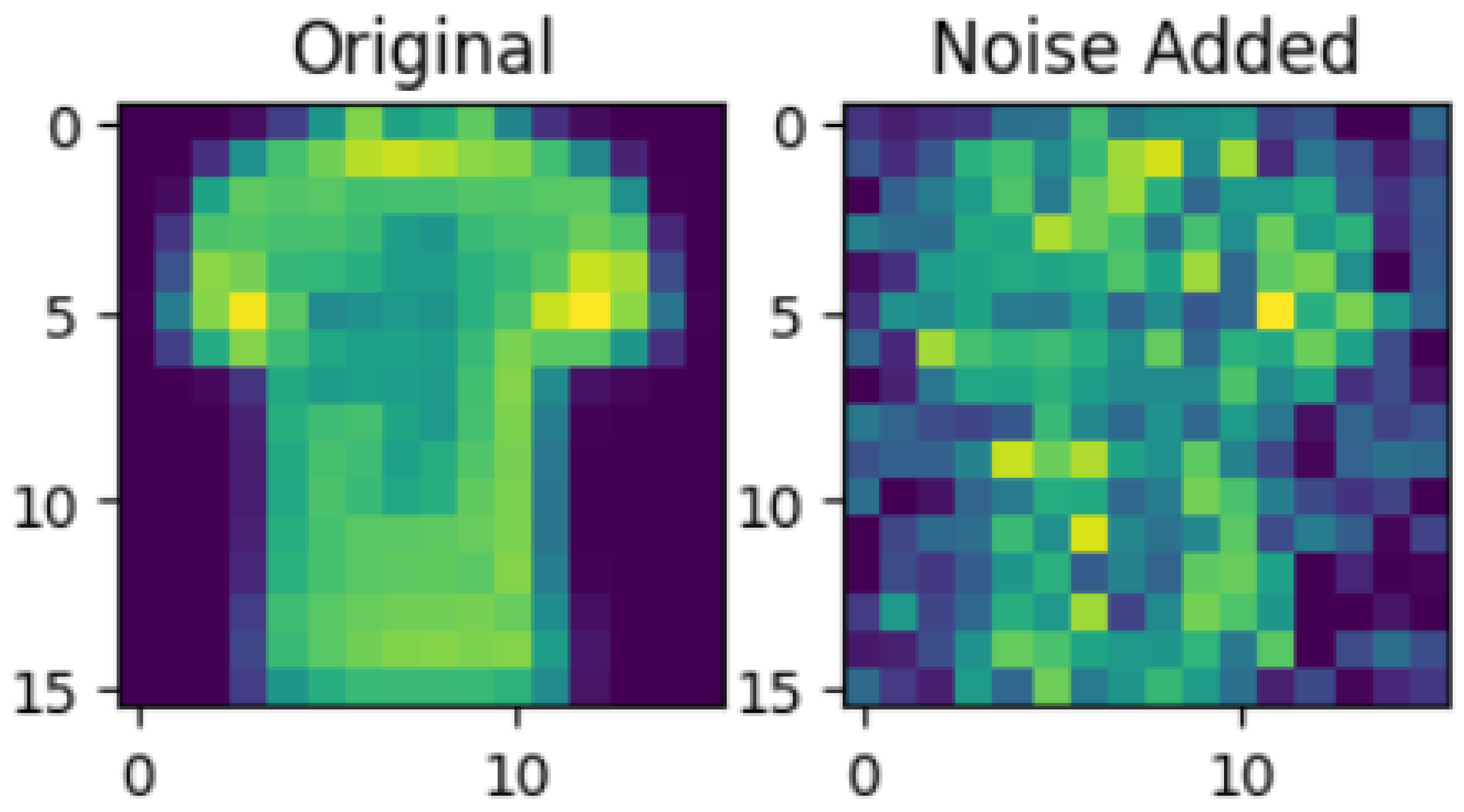


Noise added at time t

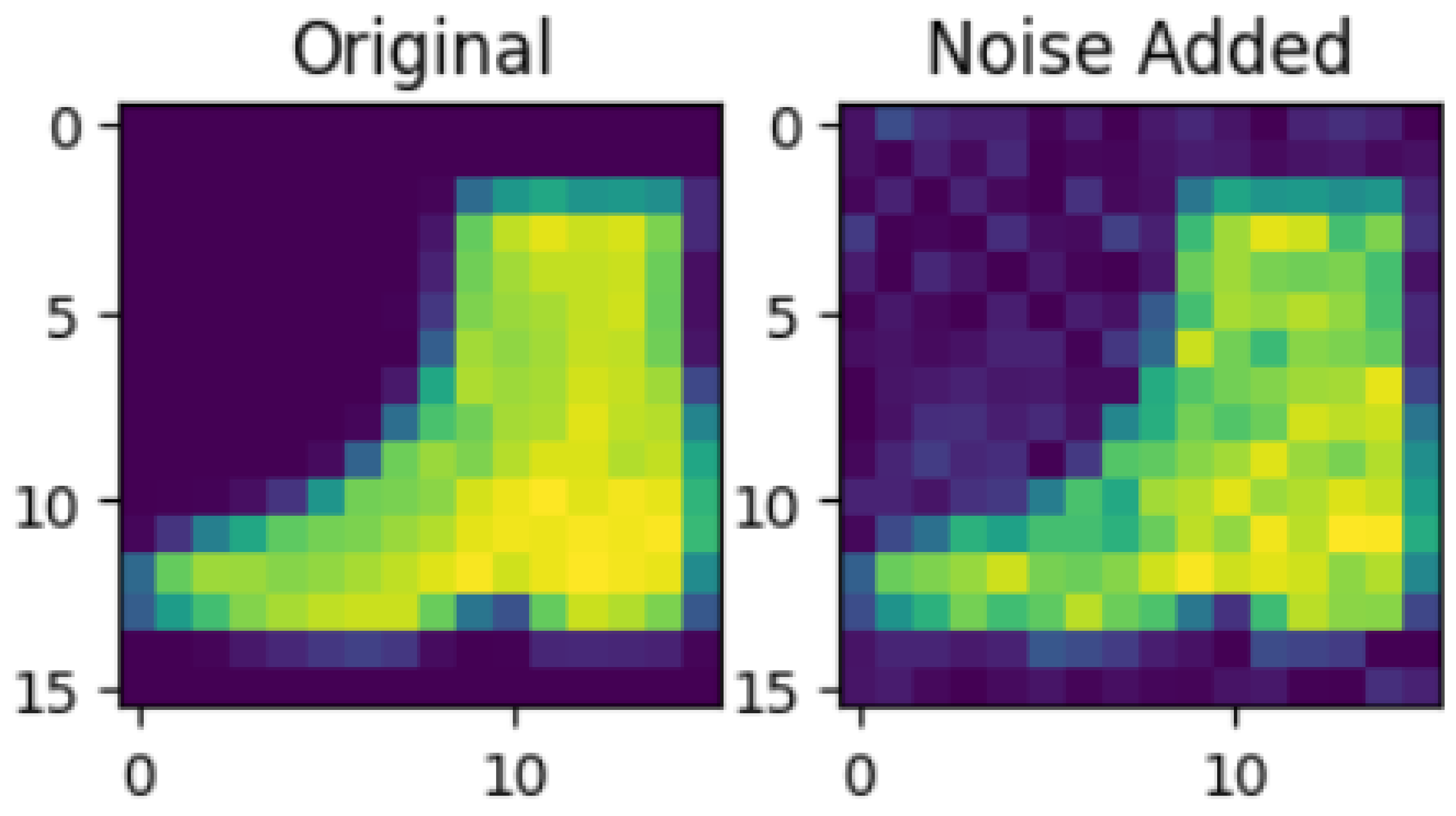
$$q(\mathbf{x}_t|\mathbf{x}_0) = N(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0, (1 - \bar{\alpha}_t) \cdot \mathbf{I})$$

"Skip ahead" function

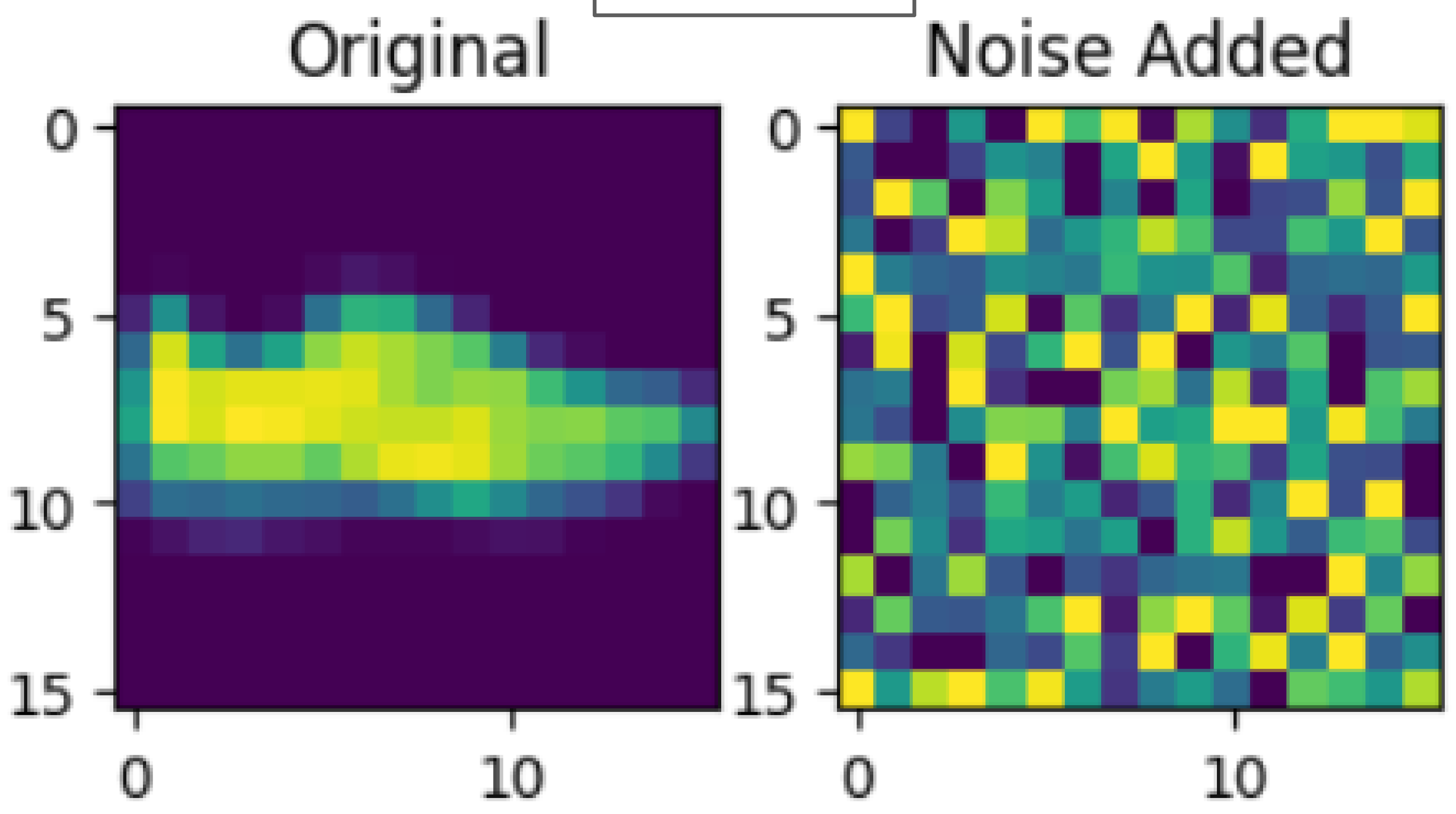
$t = 5$



$t = 25$



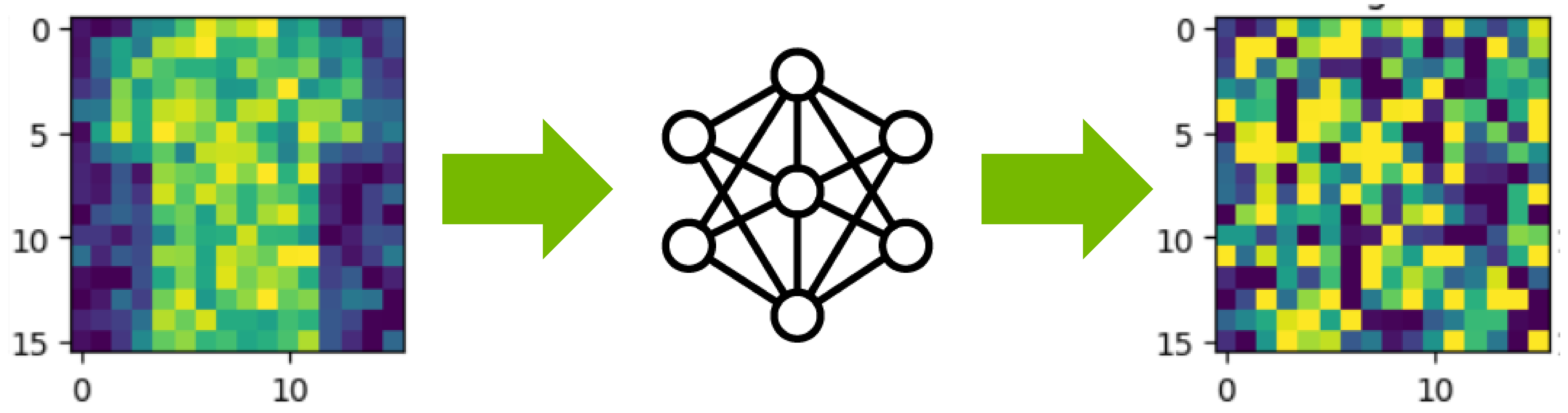
$t = 100$



Model Training

ELBO Loss

Noisy image at time t



Noise added at time t

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Model Architecture

The U-Net Architecture

Previous Lab

The U-Net Architecture

Next Lab

It’s About Time

Adding Time

It’s About Time

Broadcasting

$$\cancel{5 + \begin{bmatrix}1 & 0 & 1\\ & & \\ 1 & 0 & 1\end{bmatrix} = ?}$$

$$5 \quad \begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\\ 1 & 0 & 1\end{bmatrix}$$

0 or 1

$3\ x\ 3$

Dimensions

It’s About Time

Broadcasting

$$[[5]] + \begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\\ 1 & 0 & 1\end{bmatrix}$$

$1 \; x \; 1$

Each dimension of 1 will repeat its element to align with destination

$$\begin{bmatrix}5 & 5 & 5\\ 5 & 5 & 5\\ 5 & 5 & 5\end{bmatrix} + \begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\\ 1 & 0 & 1\end{bmatrix} = \begin{bmatrix}6 & 5 & 6\\ 5 & 6 & 5\\ 6 & 5 & 6\end{bmatrix}$$

$3 \; x \; 3$ | $3 \; x \; 3$ | $3 \; x \; 3$

It’s About Time

Broadcasting

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$3\ x\ 1$

Each dimension of 1 will repeat its element to align with destination

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 2 \\ 2 & 3 & 2 \\ 4 & 3 & 4 \end{bmatrix}$$

$3\ x\ 3$ $\quad$ $3\ x\ 3$ $\quad$ $3\ x\ 3$

Let’s get started!

NVIDIA