



# 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



Adobe Firefly

## Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein  
Stanford University

JASCHA@STANFORD.EDU

Eric A. Weiss  
University of California, Berkeley

EAWISS@BERKELEY.EDU

Niru Maheswaranathan  
Stanford University

NIRUM@STANFORD.EDU

Surya Ganguli  
Stanford University

SGANGULI@STANFORD.EDU

### Abstract

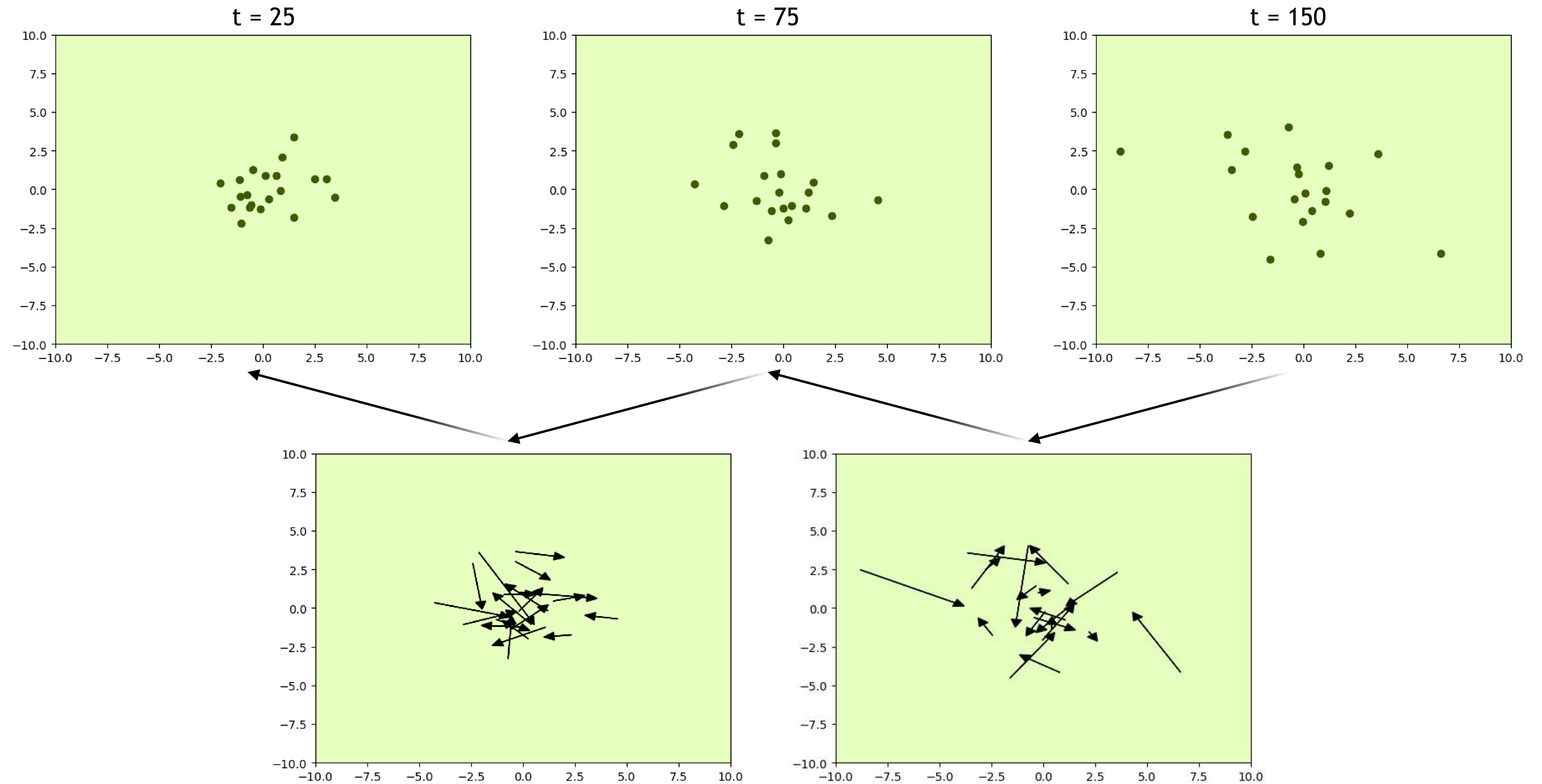
A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation

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





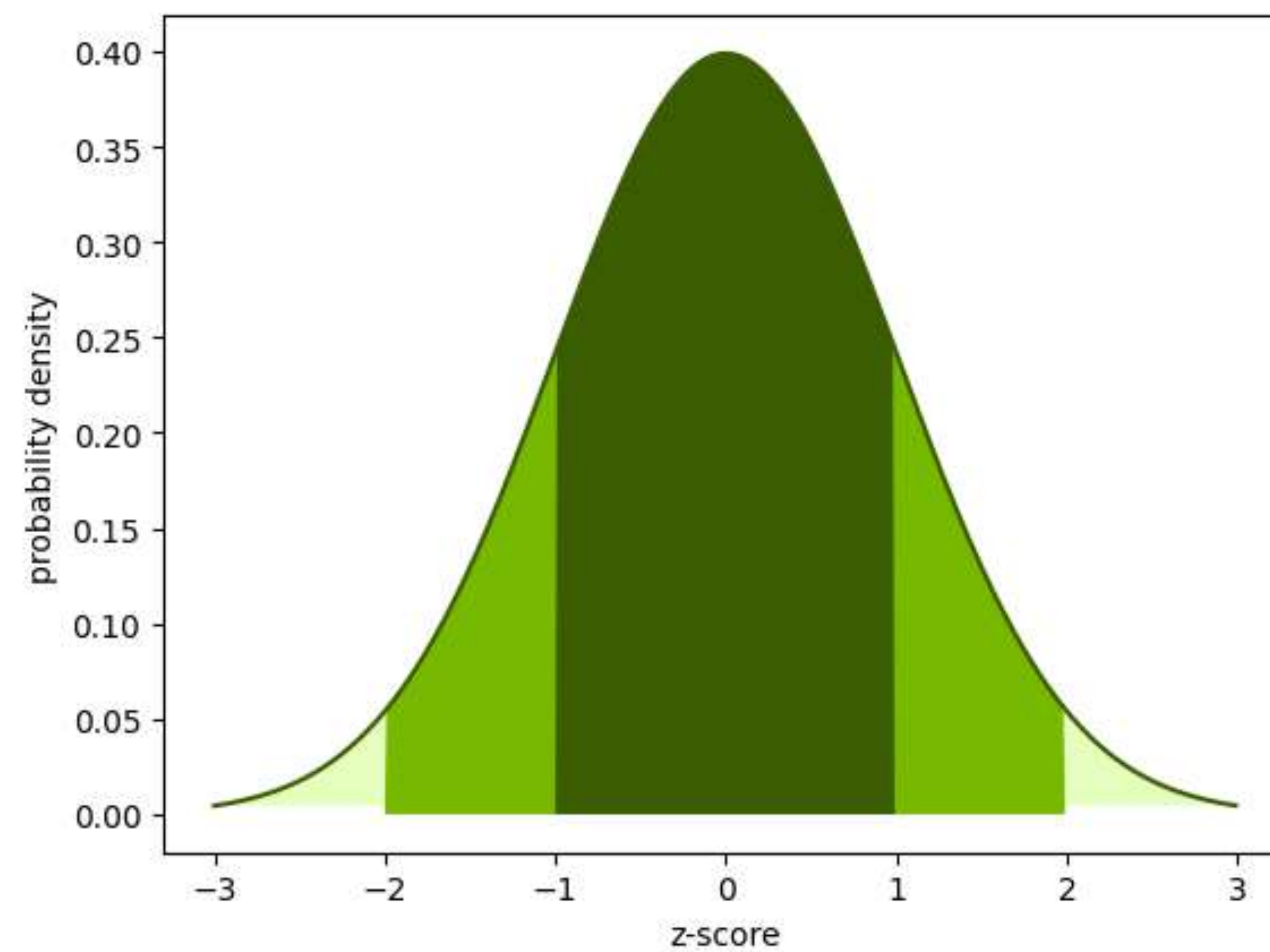
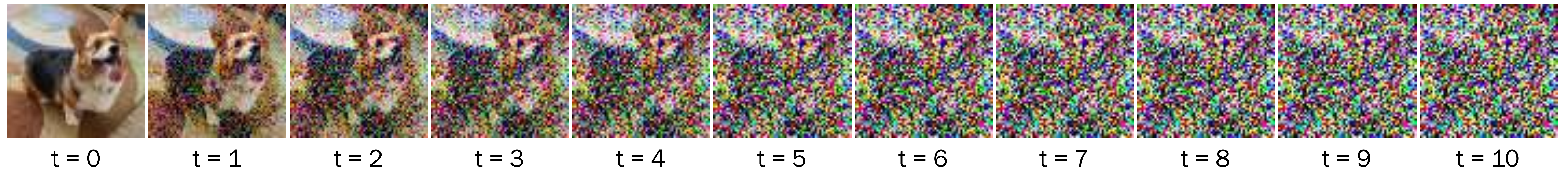


# Forward Diffusion



# Getting Lost in the Noise

Forward Diffusion

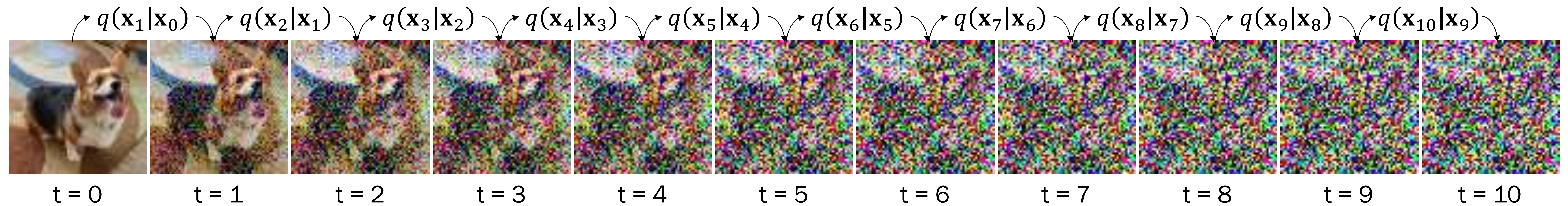


$$N(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



# Getting Lost in the Noise

Forward Diffusion



$$\beta = [0.0001, 0.0023, 0.0045, 0.0067, \dots, 0.0200]$$

$t = 0 \quad t = 1 \quad t = 3 \quad t = 4 \quad t = 10$

$$T = 10$$

Total number of timesteps

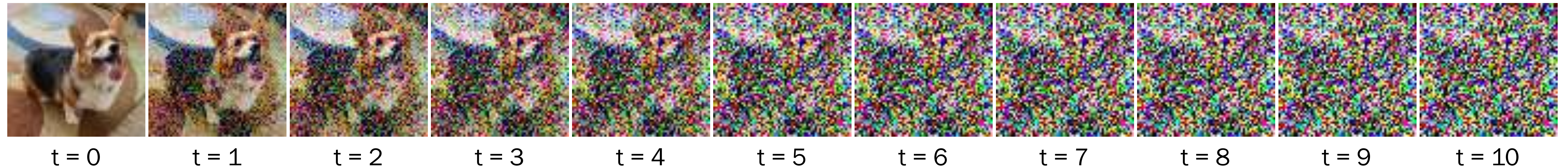
$$\alpha = 1 - \beta$$

Convenient shorthand



# Getting Lost in the Noise

## Forward Diffusion



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

### Code:

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

### Step 2:

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

### Step 1:

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



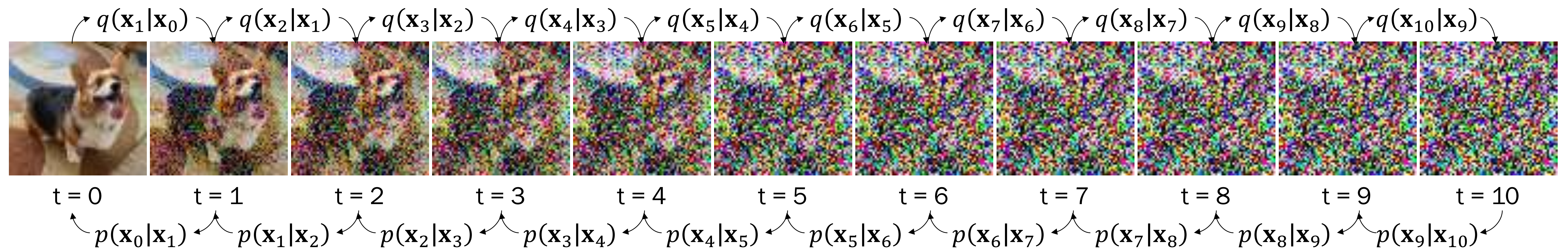


# Reverse Diffusion



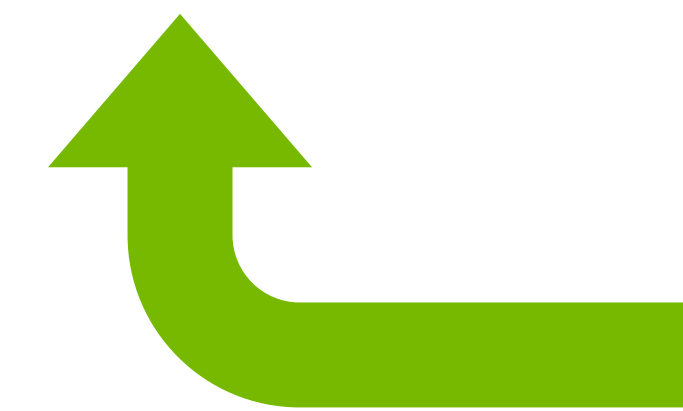
# Getting Lost in the Noise

## 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

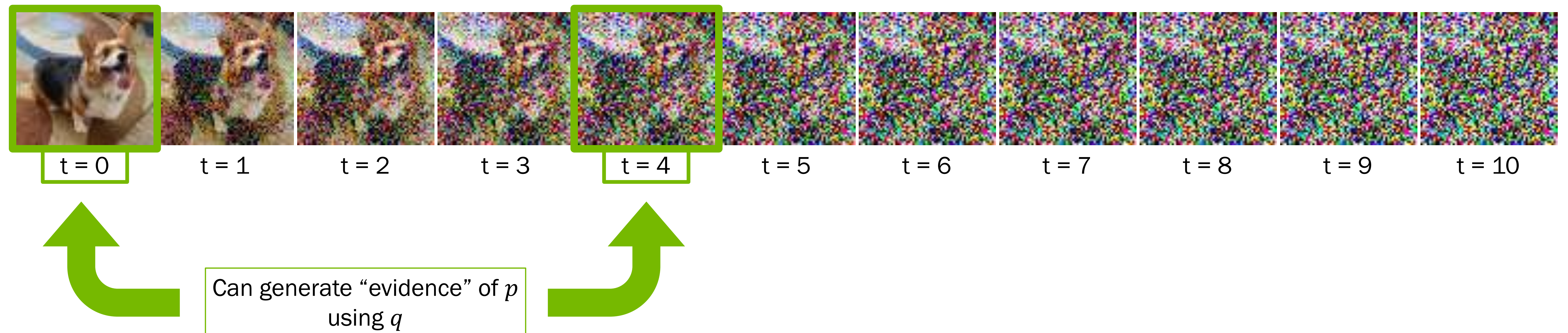


Approximate average used to  
create  $\mathbf{x}_t$



# Getting Lost in the Noise

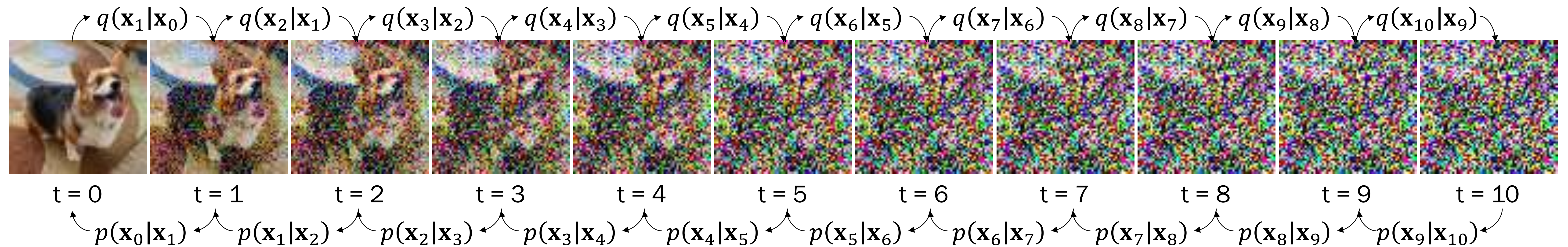
Reverse Diffusion





# Getting Lost in the Noise

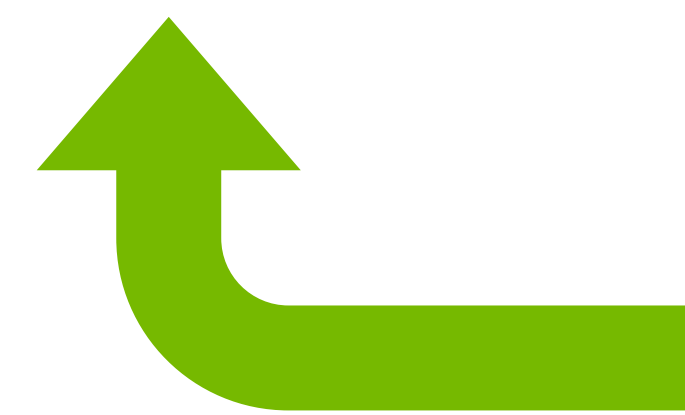
## Reverse Diffusion



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

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

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

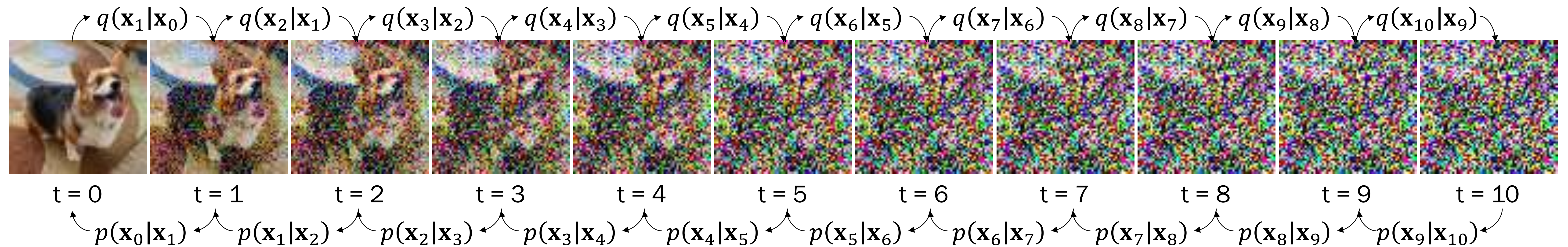


Use Bayes' Rule and ...



# Getting Lost in the Noise

## Reverse Diffusion



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

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

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

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



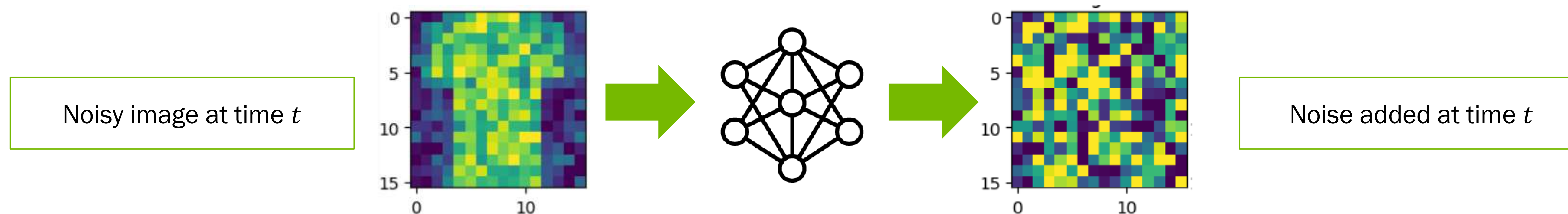


# Model Training



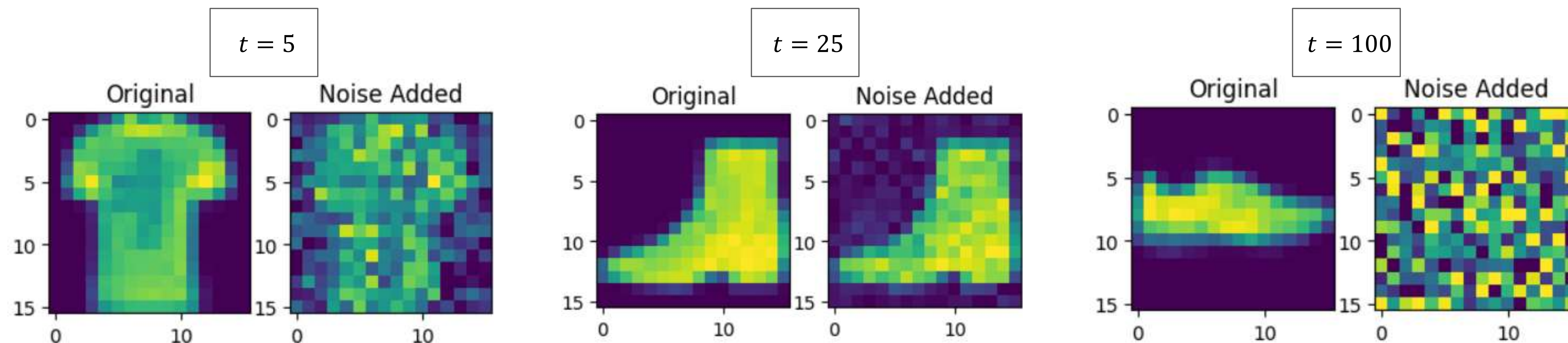
# Model Training

Training Data



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

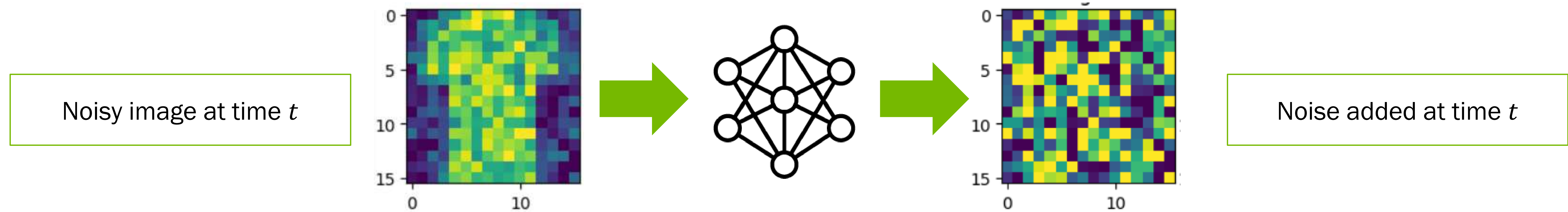
“Skip ahead” function





# Model Training

ELBO Loss



~~$ELBO := \mathbb{E}_{q(\phi)} \left[ \log p(y_i | \phi) \right] - \mathbb{E}_{q(\phi)} \left[ \log q(\phi) \right]$~~

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



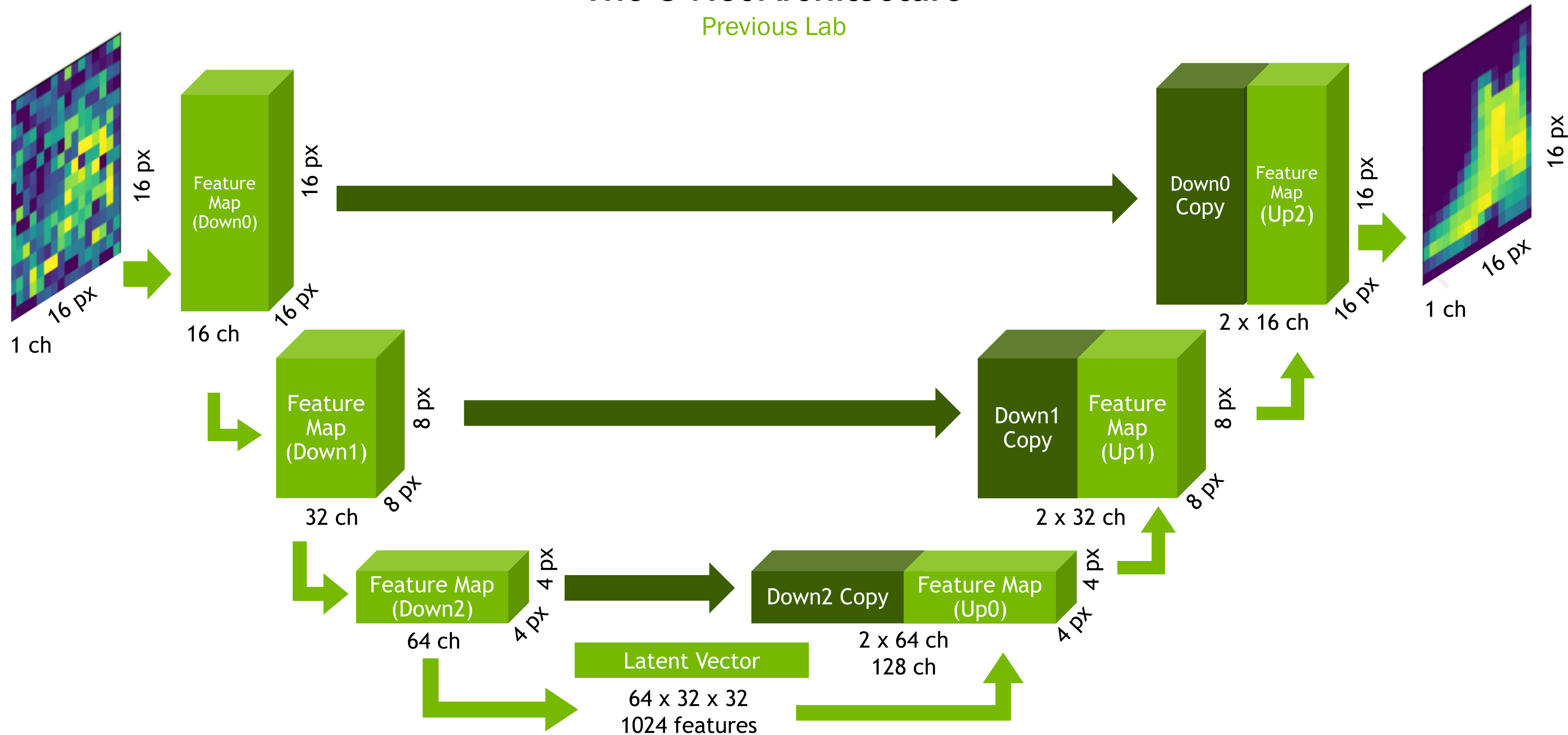


# Model Architecture



# The U-Net Architecture

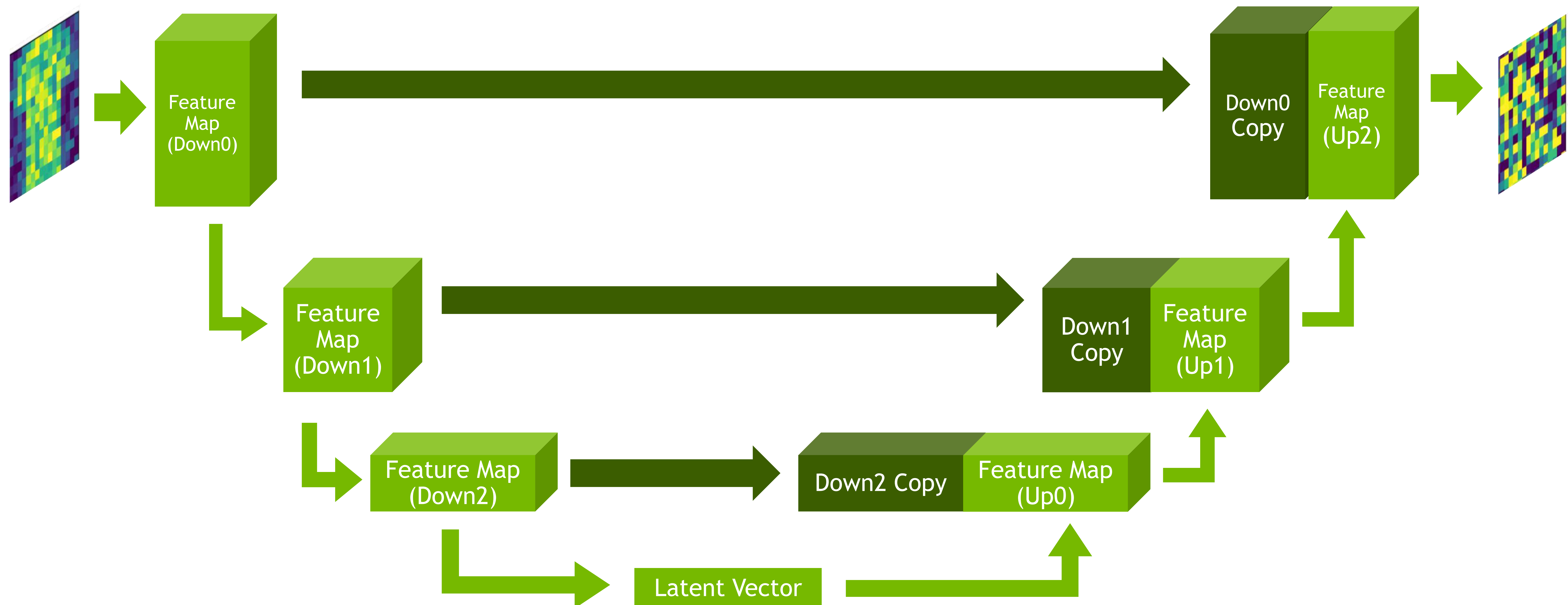
Previous Lab





# The U-Net Architecture

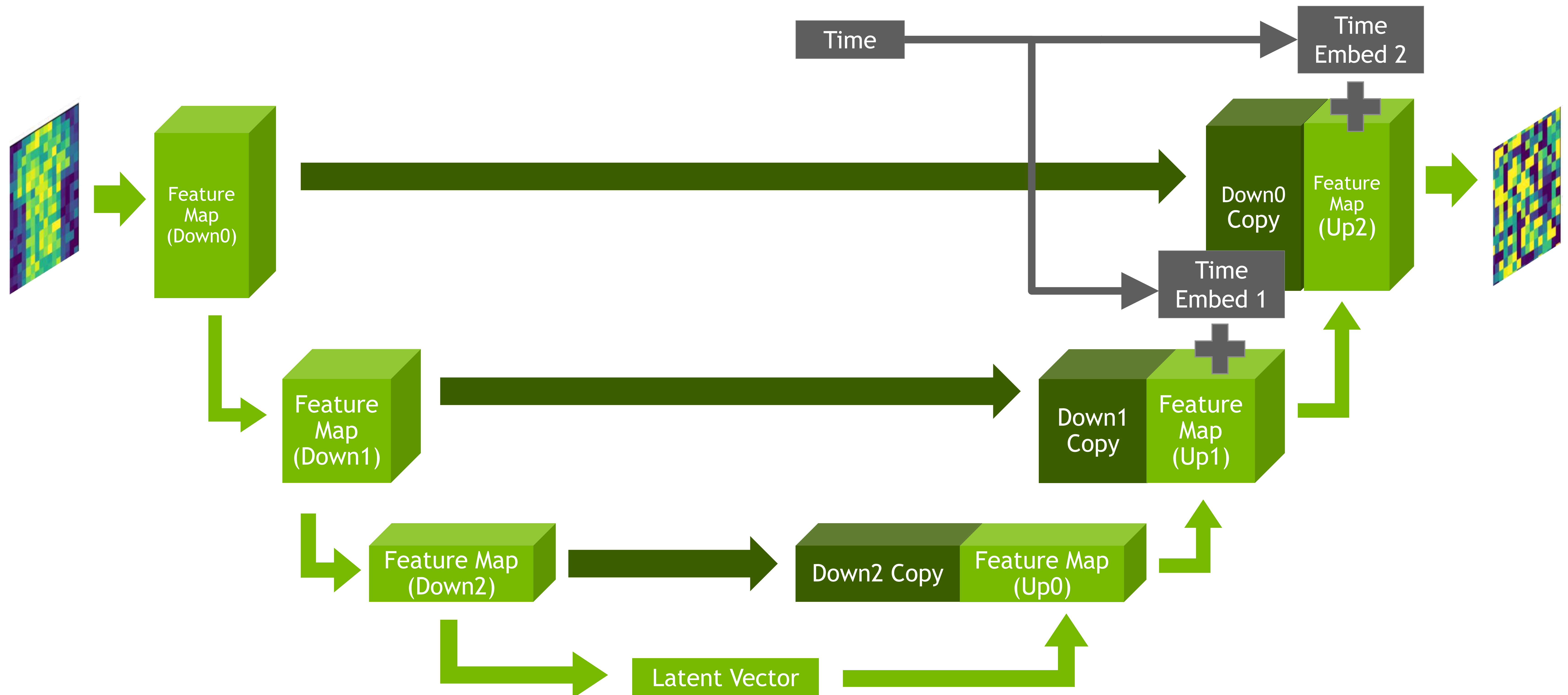
Next Lab





# It's About Time

Adding Time





# It's About Time

Broadcasting

5 + 

1	0	1
1	0	1

 = ?

5 

1	0	1
0	1	0
1	0	1

0 or 1

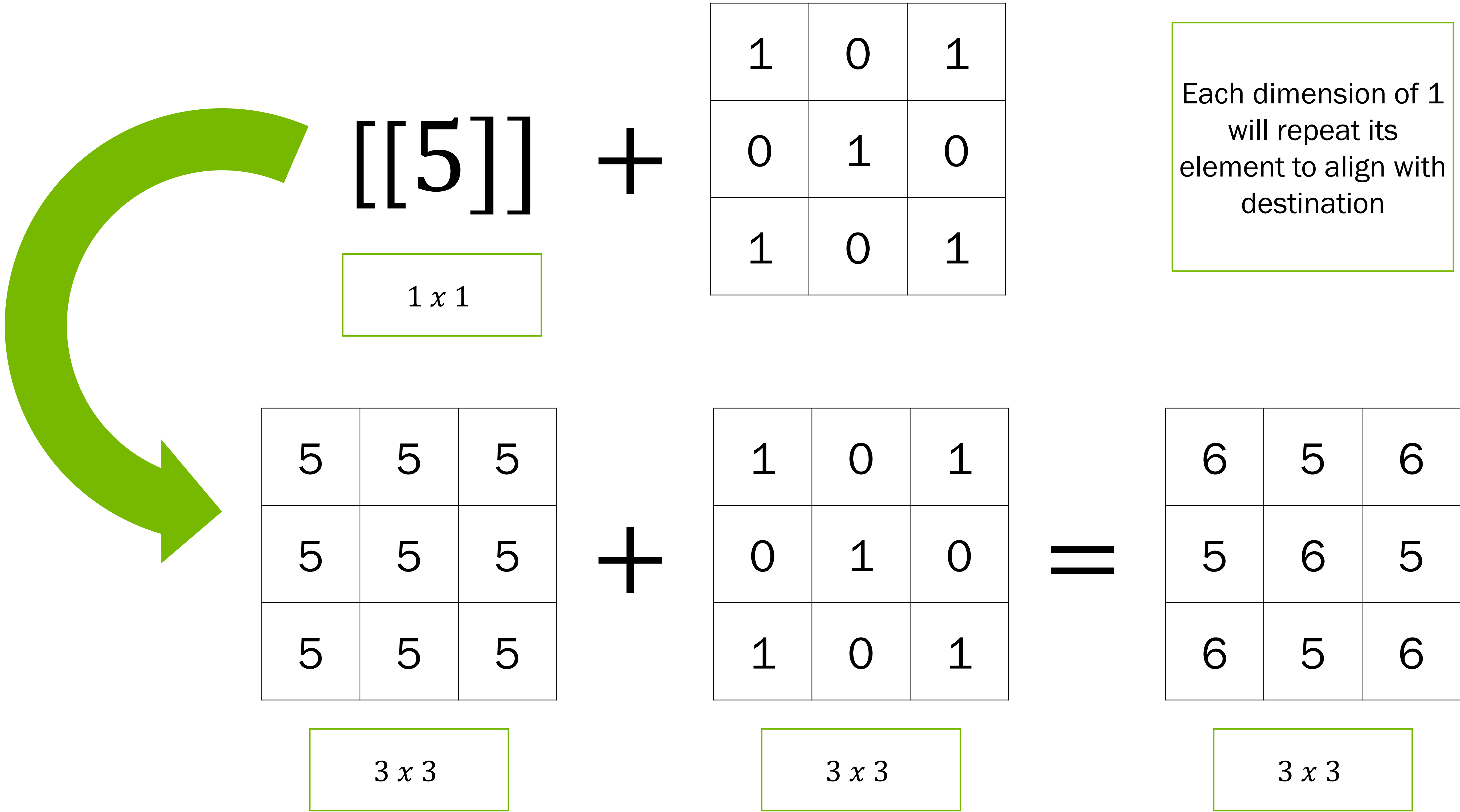
3 x 3

Dimensions



# It's About Time

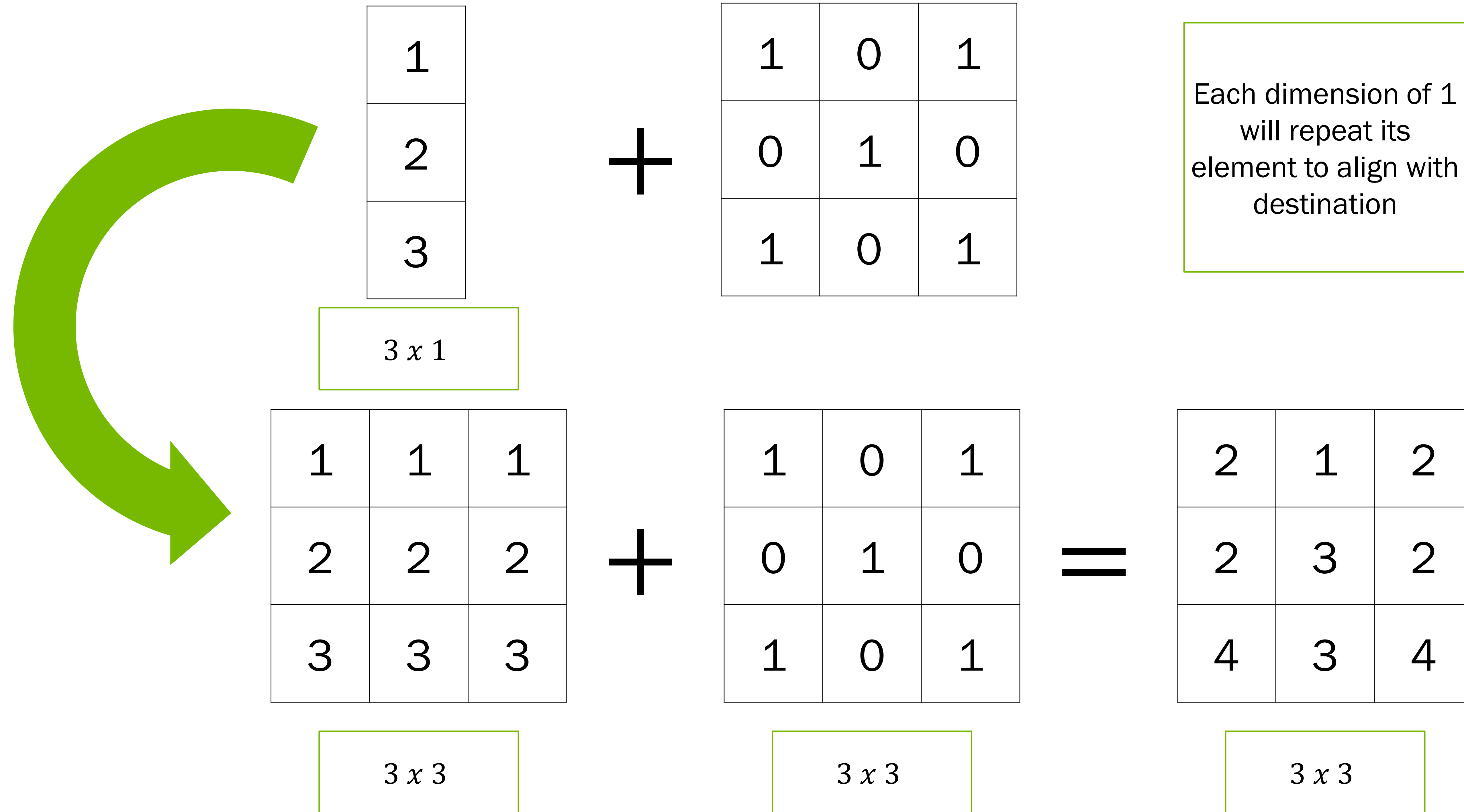
Broadcasting





# It's About Time

## Broadcasting







**Let's get started!**



