Deep Learning for Computer Vision

Andrii Liubonko

Grammarly*

* The opinions expressed in this presentation and on the following slides are solely those of the presenter and not necessarily those of Grammarly

Logistics

4 units

3 types of homework:

- paper review [20 % of FINAL SCORE]
- notebooks [30 % of FINAL SCORE]
- mini-project [50 % of FINAL SCORE]

important dates:

31 January, 23:59

course repo:

https://github.com/lyubonko/ucu2022cv

Overview of the course

Unit I

[T] Intro
[P] pytorch

Unit II

[T] CNNs in depth, Object Detection
[P] simple nets

Unit III

[T] Attention in CV, Transformers
[P] classification

Unit IV

[T] Generative models, Diffusion Models
[P] project structure, stable diffusion

“The sculpture is already complete within the marble block, before I start my work. It is already there, I just have to chisel away the superfluous material.”
© Michelangelo

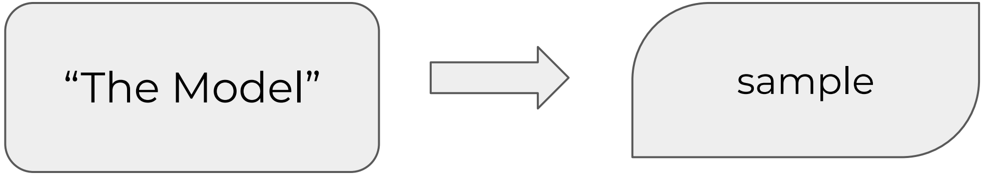
“Creating noise from data is easy;
creating data from noise is generative modeling”
© 2011.13456

Content of today’s lecture

- Generative modeling
  - appetiser
  - approaches
- Diffusion models
  - intro
  - tale of marvelous Gaussians
  - more details
- Stable Diffusion

Generative modeling

- uncontrollable (unconditional) generation:



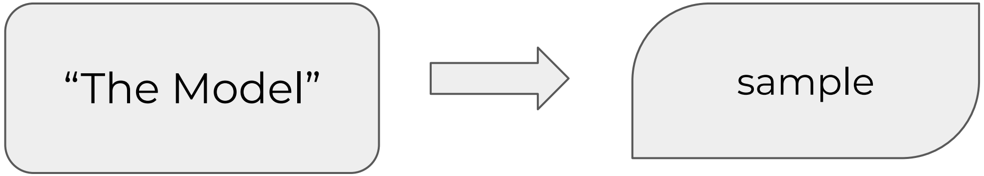
“The Model”

sample

- text
- visual
- sound
- code
- …

Generative modeling

- uncontrollable (unconditional) generation:

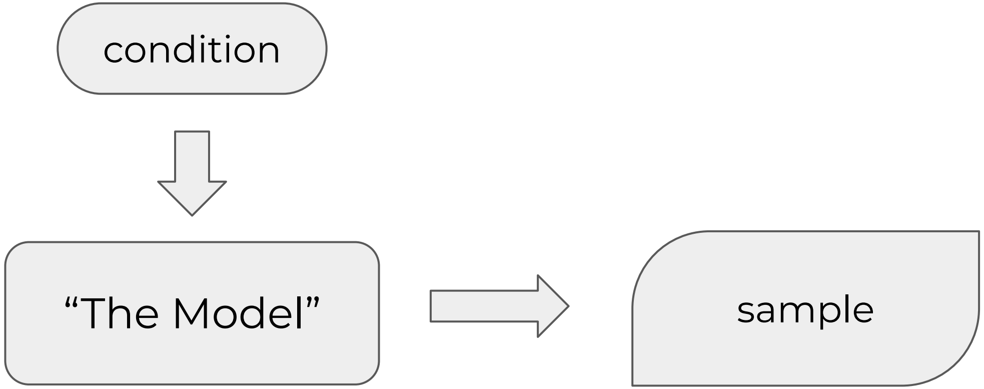


“The Model”

sample

- text
- visual
- sound
- code
- …

- controllable (conditional) generation:

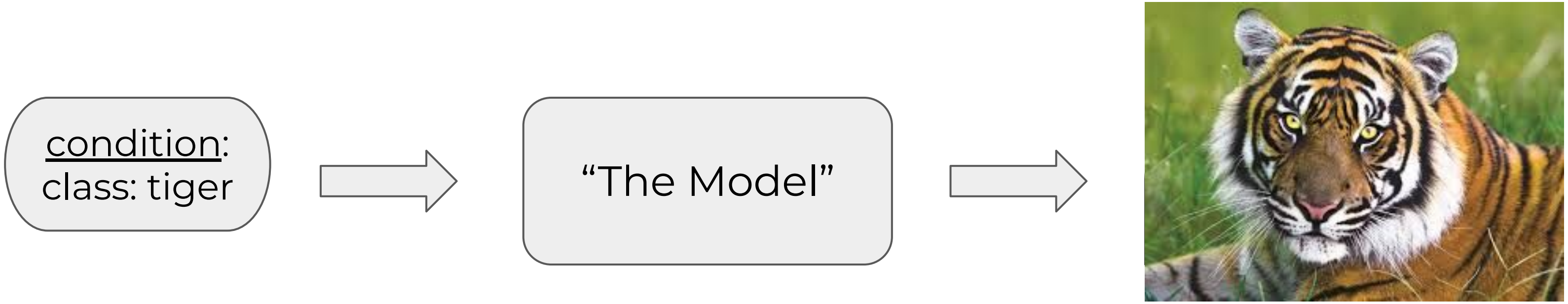


condition

“The Model”

sample

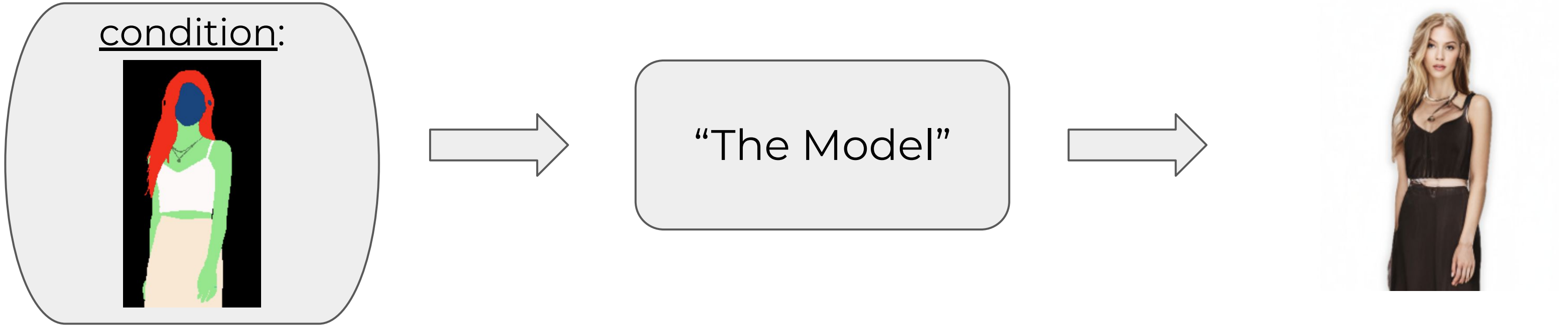
Generative modeling



condition:
class: tiger

“The Model”

Generative modeling



condition:

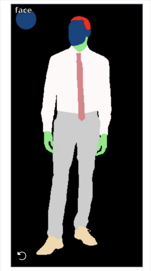
“The Model”

https://huggingface.co/spaces/CVPR/drawings-to-human

Generative modeling

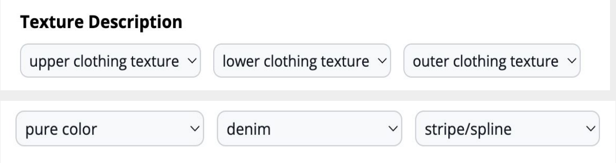
Control over multiple conditions:

condition:



+

condition:



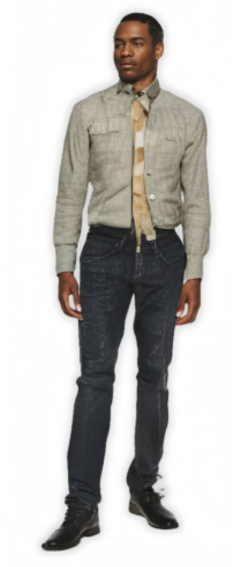
Texture Description

upper clothing texture: pure color

lower clothing texture: denim

outer clothing texture: stripe/spline

“The Model”

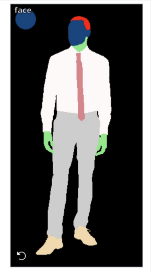


https://huggingface.co/spaces/CVPR/drawings-to-human

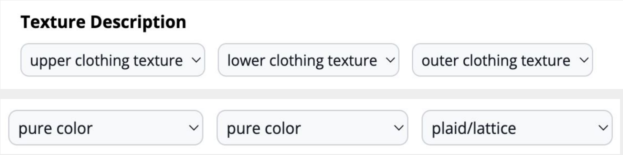
Generative modeling

Control over multiple conditions:

condition:



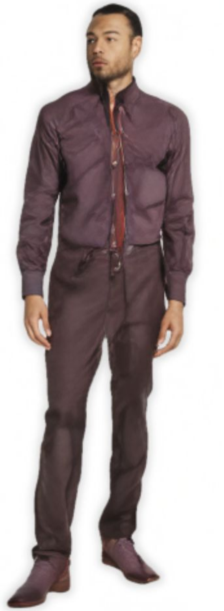
condition:



Texture Description

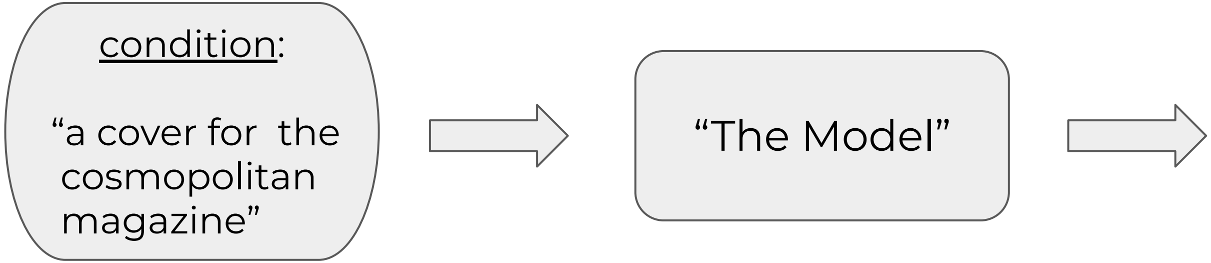
upper clothing texture	lower clothing texture	outer clothing texture
pure color	pure color	plaid/lattice

“The Model”



https://huggingface.co/spaces/CVPR/drawings-to-human

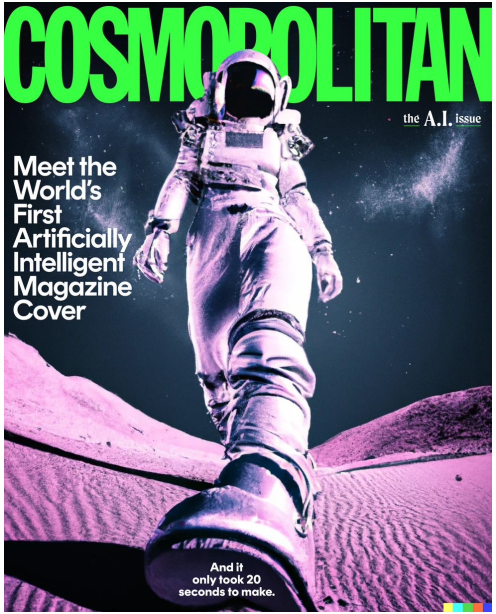
Generative modeling



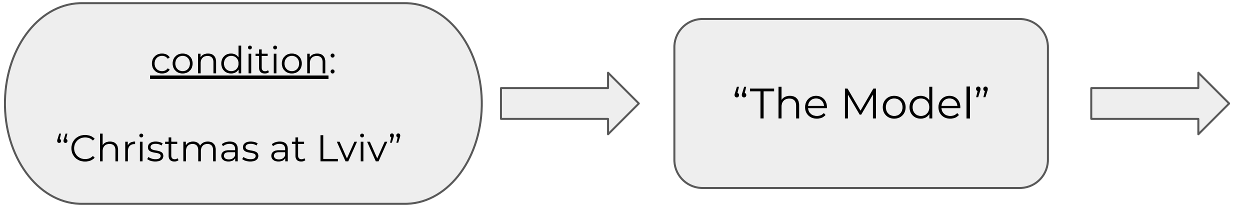
condition:

"a cover for the cosmopolitan magazine"

"The Model"


COSMOPOLITAN
the A.I. issue
Meet the World's First Artificially Intelligent Magazine Cover
And it only took 20 seconds to make.

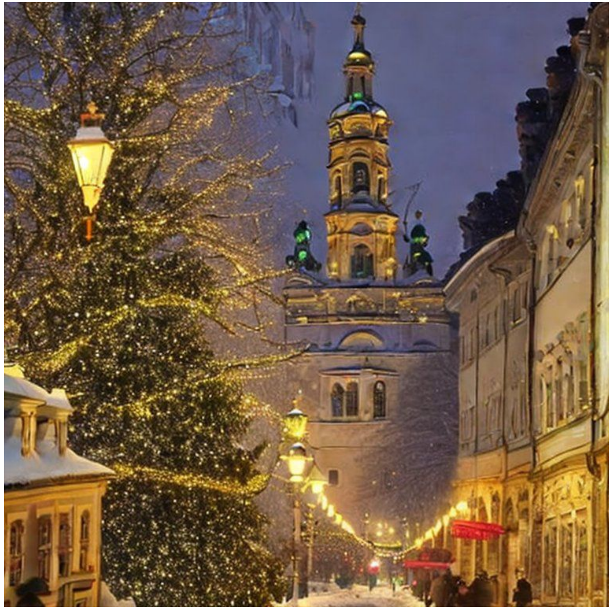
Generative modeling



condition:

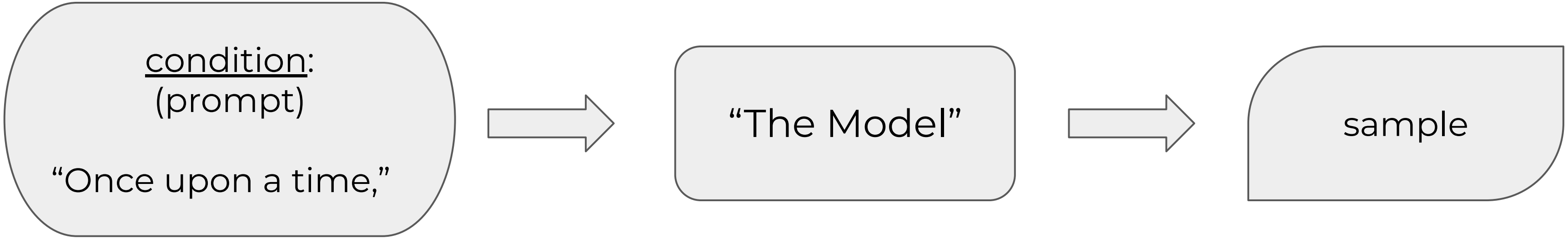
“Christmas at Lviv”

“The Model”



https://huggingface.co/spaces/stabilityai/stable-diffusion

Generative modeling [for text]

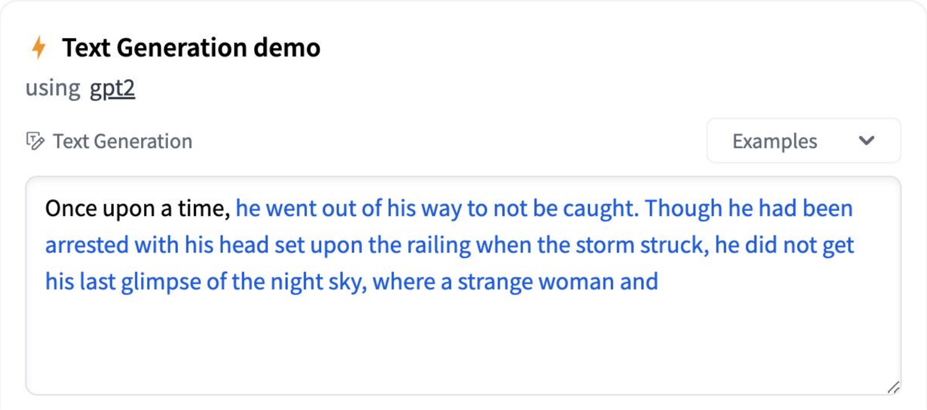


condition:
(prompt)

“Once upon a time,”

“The Model”

sample

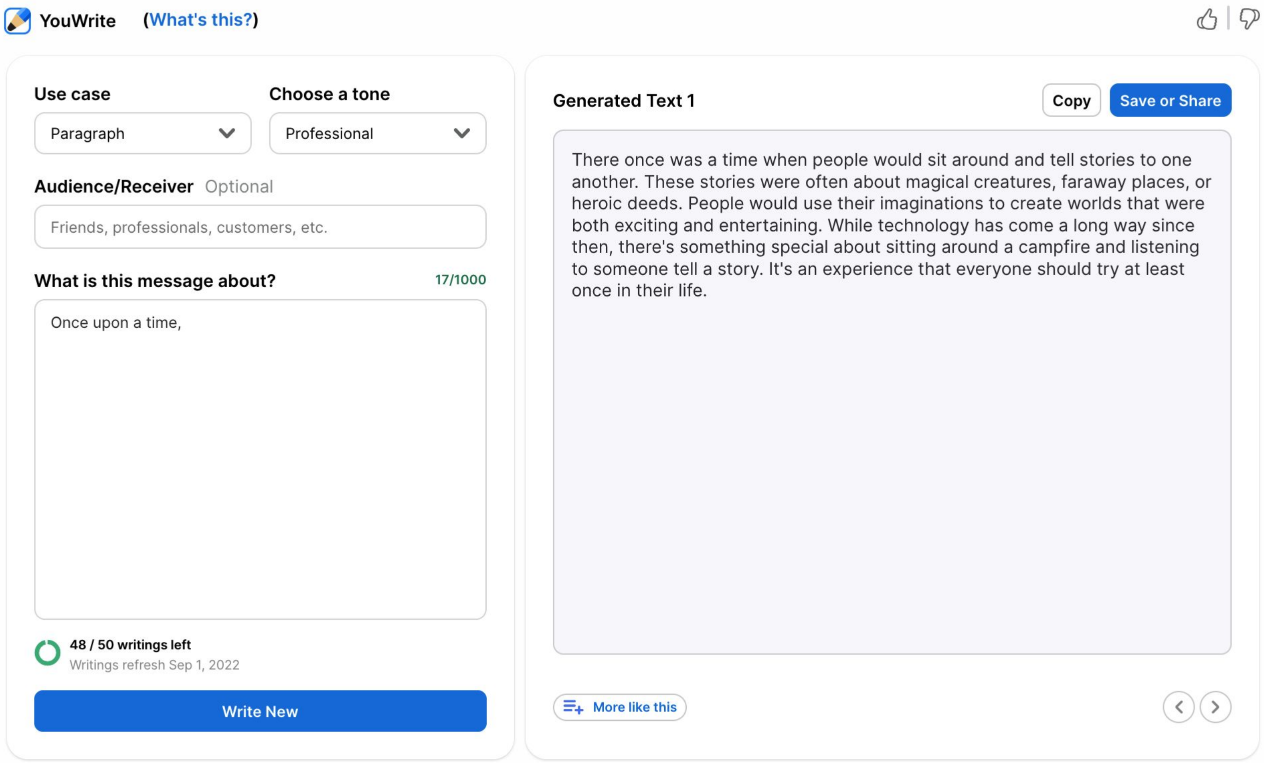


Once upon a time, he went out of his way to not be caught. Though he had been arrested with his head set upon the railing when the storm struck, he did not get his last glimpse of the night sky, where a strange woman and

https://huggingface.co/tasks/text-generation

Generative modeling [for text]

- controllable (conditional) generation:



Use case: Paragraph

Choose a tone: Professional

Audience/Receiver Optional

Friends, professionals, customers, etc.

What is this message about? 17/1000

Once upon a time,

48 / 50 writings left

Writings refresh Sep 1, 2022

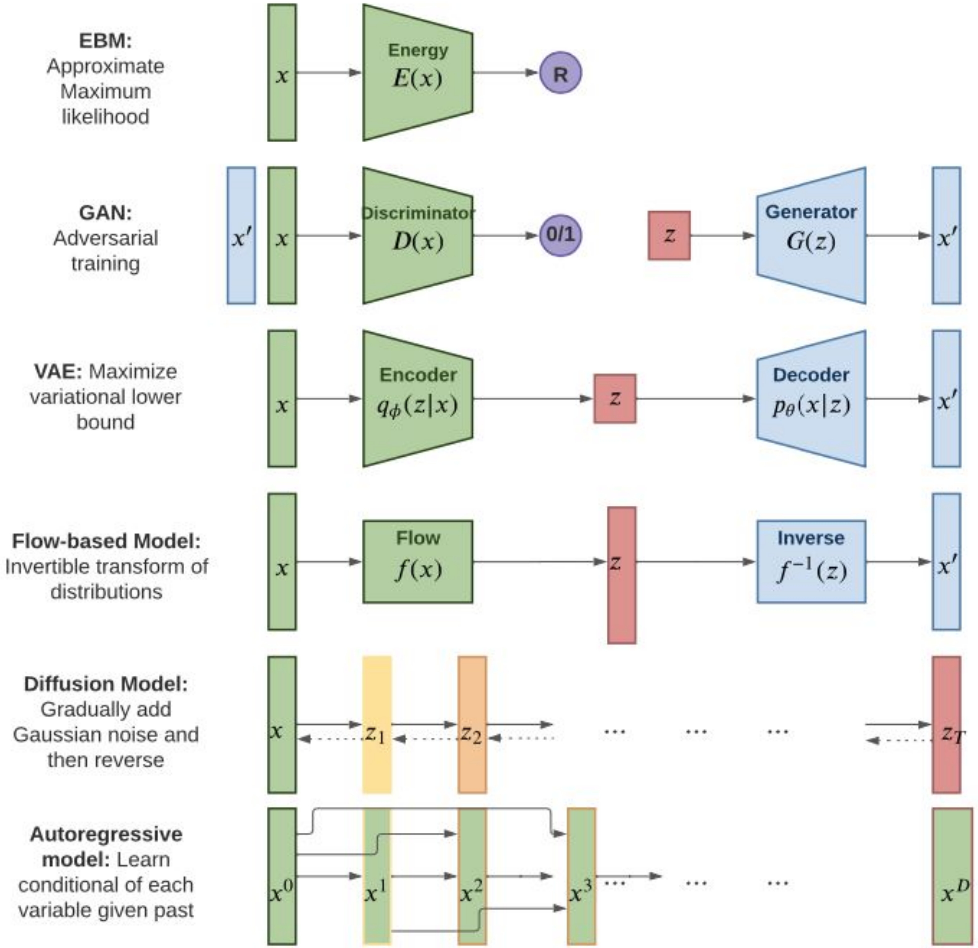
Generated Text 1

There once was a time when people would sit around and tell stories to one another. These stories were often about magical creatures, faraway places, or heroic deeds. People would use their imaginations to create worlds that were both exciting and entertaining. While technology has come a long way since then, there's something special about sitting around a campfire and listening to someone tell a story. It's an experience that everyone should try at least once in their life.

YouWrite

Content of today’s lecture

- Generative modeling
  - appetiser
  - approaches
- Diffusion models
  - intro
  - tale of marvelous Gaussians
  - more details
- Stable Diffusion



EBM: Approximate Maximum likelihood

GAN: Adversarial training

VAE: Maximize variational lower bound

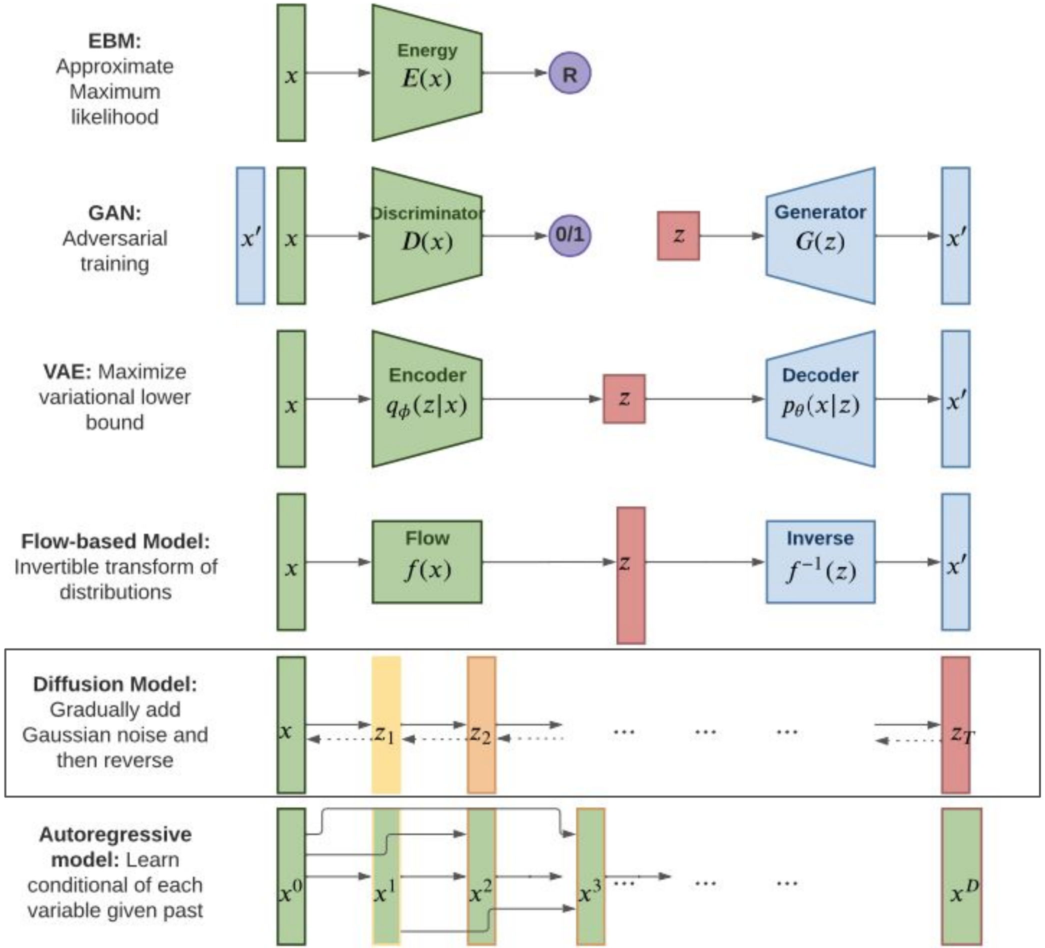
Flow-based Model: Invertible transform of distributions

Diffusion Model: Gradually add Gaussian noise and then reverse

Autoregressive model: Learn conditional of each variable given past

[Kevin P. Murphy “Probabilistic Machine Learning: Advanced Topics”, 2023]

originally based on
[What are Diffusion Models? | Lil'Log]

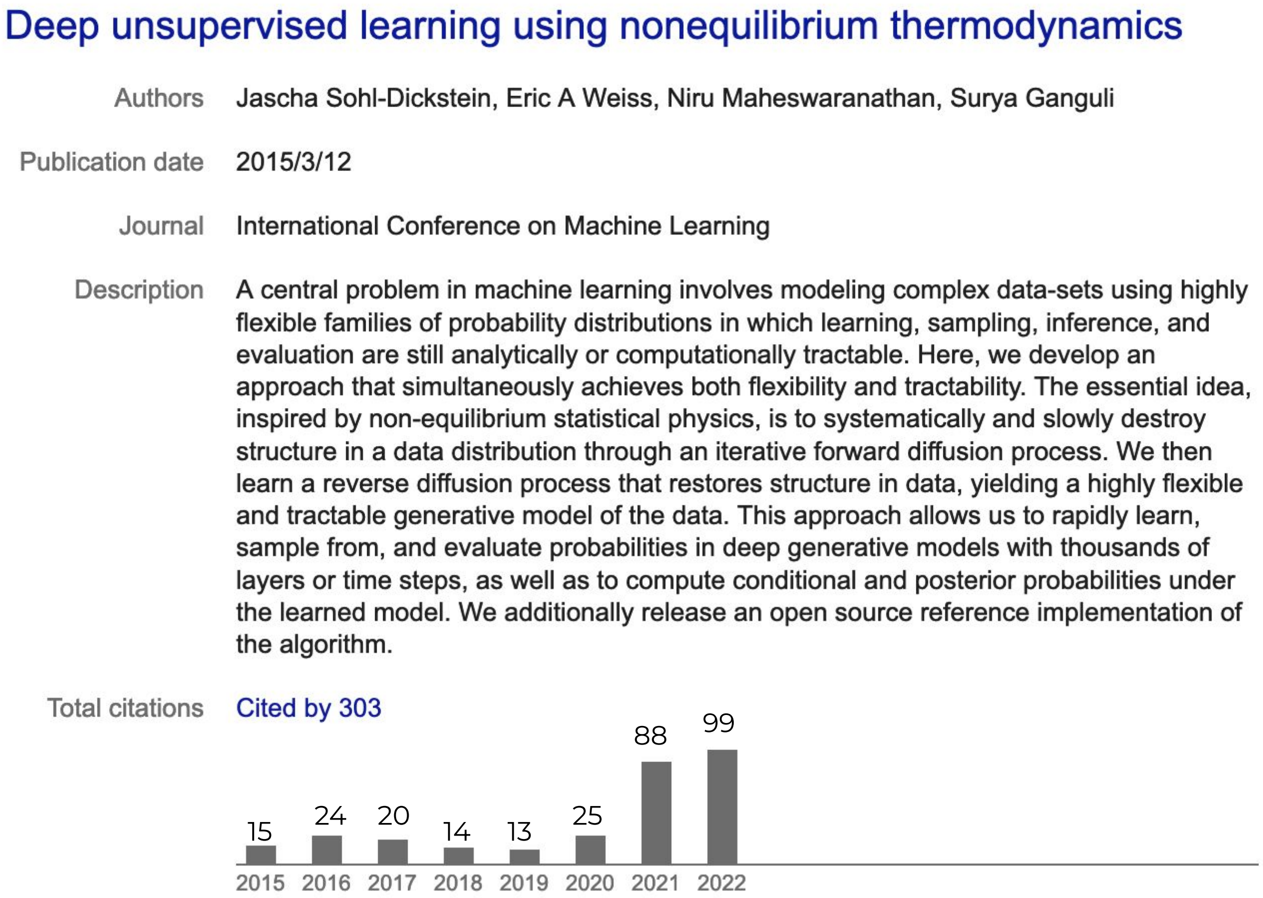


[Kevin P. Murphy “Probabilistic Machine Learning: Advanced Topics”, 2023]

originally based on
[What are Diffusion Models? | Lil'Log]

Content of today's lecture

- Generative modeling
  - appetiser
  - approaches
- Diffusion models
  - intro
  - tale of marvelous Gaussians
  - more details
- Stable Diffusion



Deep unsupervised learning using nonequilibrium thermodynamics

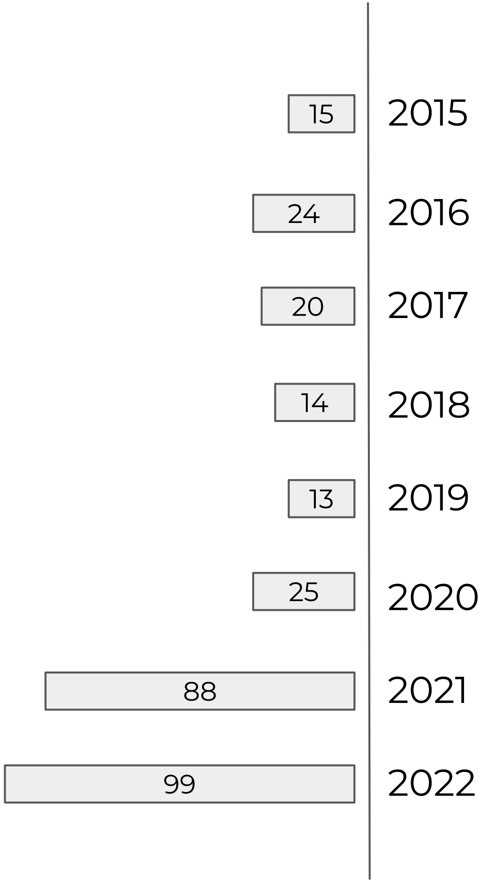
Authors Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, Surya Ganguli

Publication date 2015/3/12

Journal International Conference on Machine Learning

Description 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 are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

Total citations Cited by 303



[1503.03585] Deep Unsupervised Learning using Nonequilibrium Thermodynamics
#foundational, #images,
Jascha Sohl-Dickstein [Stanford University]

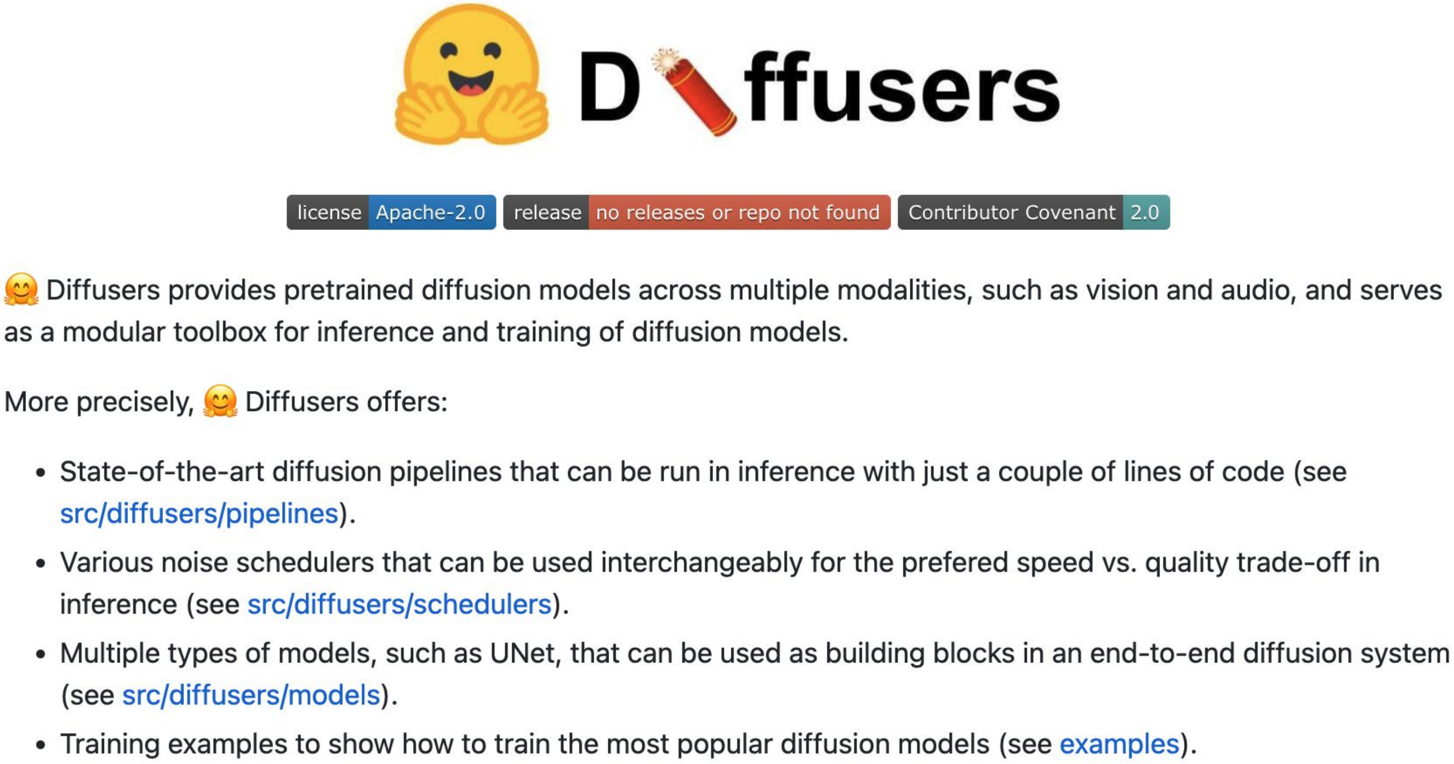
[2006.11239] Denoising Diffusion Probabilistic Models #foundational, #images,
#connecting score based method (1907.05600)
Jonathan Ho [UC Berkeley]

[2011.13456] Score-Based Generative Modeling through Stochastic Differential Equations code blog #foundational, #extending to Stochastic stuff
Yang Song, Jascha Sohl-Dickstein [Stanford University + Google Brain]

[2105.05233] Diffusion Models Beat GANs on Image Synthesis **code**
#images, [openAI]
Yang Song, Jascha Sohl-Dickstein [Stanford University + Google Brain]

[2205.14217] Diffusion-LM Improves Controllable Text Generation
#language, #Stanford

[2208.04202] Analog Bits: Generating Discrete Data using Diffusion Models with Self-Conditioning
#images, #language, #Google
Hinton



🤗 D🧨ffusers

license Apache-2.0 | release no releases or repo not found | Contributor Covenant 2.0

🤗 Diffusers provides pretrained diffusion models across multiple modalities, such as vision and audio, and serves as a modular toolbox for inference and training of diffusion models.

More precisely, 🤗 Diffusers offers:

- State-of-the-art diffusion pipelines that can be run in inference with just a couple of lines of code (see src/diffusers/pipelines).
- Various noise schedulers that can be used interchangeably for the prefered speed vs. quality trade-off in inference (see src/diffusers/schedulers).
- Multiple types of models, such as UNet, that can be used as building blocks in an end-to-end diffusion system (see src/diffusers/models).
- Training examples to show how to train the most popular diffusion models (see examples).

https://github.com/huggingface/diffusers

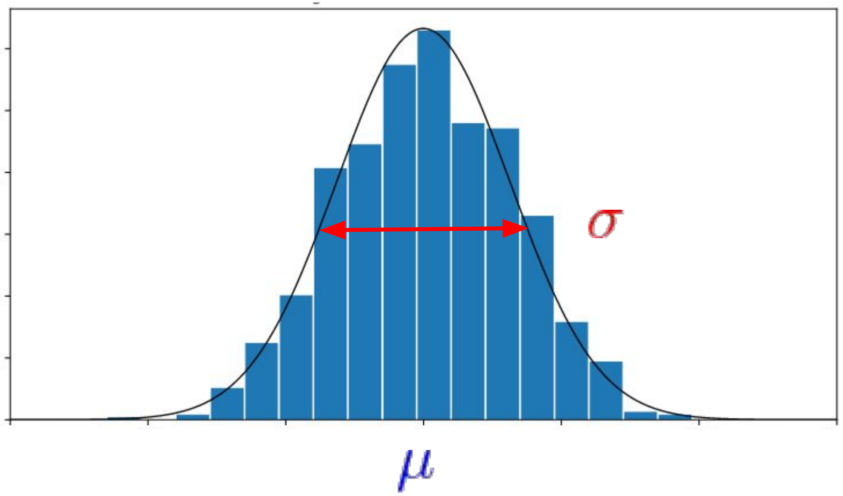
- Generative modeling

- Diffusion models
    - intro
    - **tale of marvelous Gaussians**
    - details

- Application to NLP domain
    - [2205.14217] Diffusion-LM Improves Controllable Text Generation
    - [2208.04202] Analog Bits: Generating Discrete Data using Diffusion Models with Self-Conditioning
    - [2208.00638] Composable Text Control Operations in Latent Space with Ordinary Differential Equations

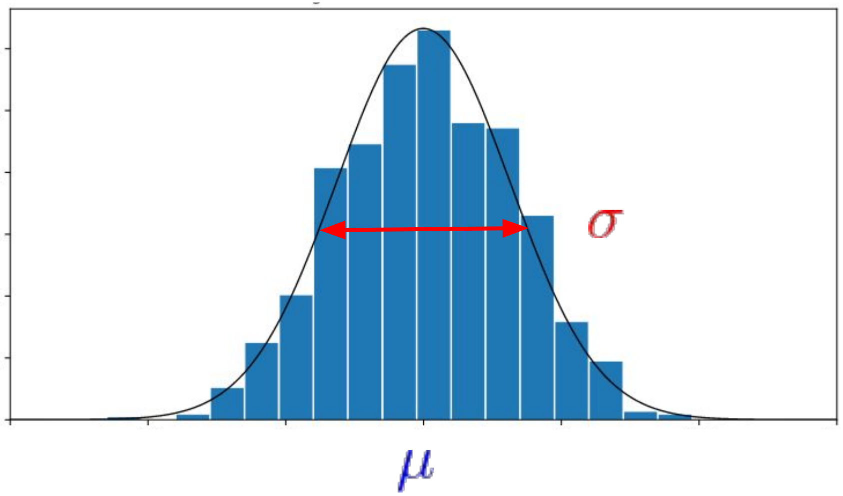
Gaussian distribution

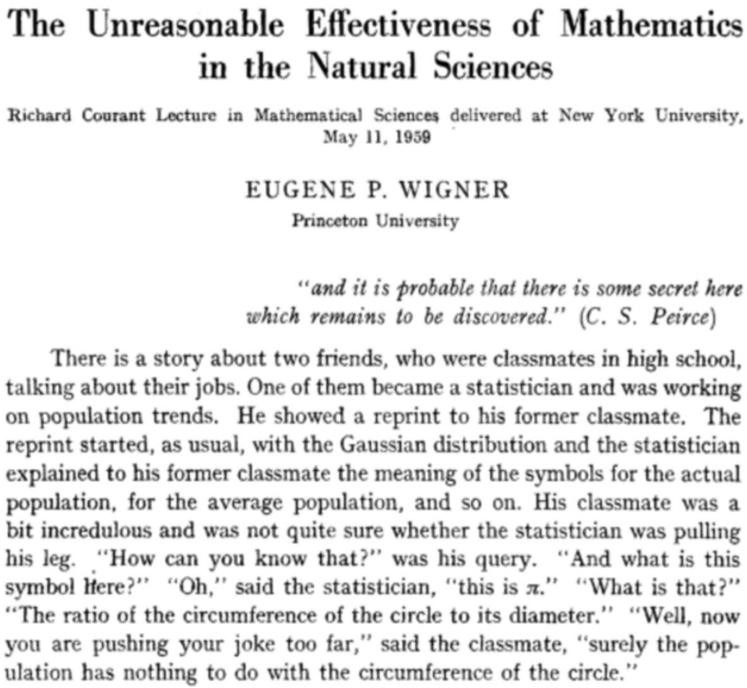
$$\mathcal{N}(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$



Gaussian distribution

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





The Unreasonable Effectiveness of Mathematics in the Natural Sciences

Richard Courant Lecture in Mathematical Sciences delivered at New York University, May 11, 1959

EUGENE P. WIGNER
Princeton University

> *"and it is probable that there is some secret here which remains to be discovered."* (*C. S. Peirce*)

There is a story about two friends, who were classmates in high school, talking about their jobs. One of them became a statistician and was working on population trends. He showed a reprint to his former classmate. The reprint started, as usual, with the Gaussian distribution and the statistician explained to his former classmate the meaning of the symbols for the actual population, for the average population, and so on. His classmate was a bit incredulous and was not quite sure whether the statistician was pulling his leg. "How can you know that?" was his query. "And what is this symbol here?" "Oh," said the statistician, "this is π." "What is that?" "The ratio of the circumference of the circle to its diameter." "Well, now you are pushing your joke too far," said the classmate, "surely the population has nothing to do with the circumference of the circle."

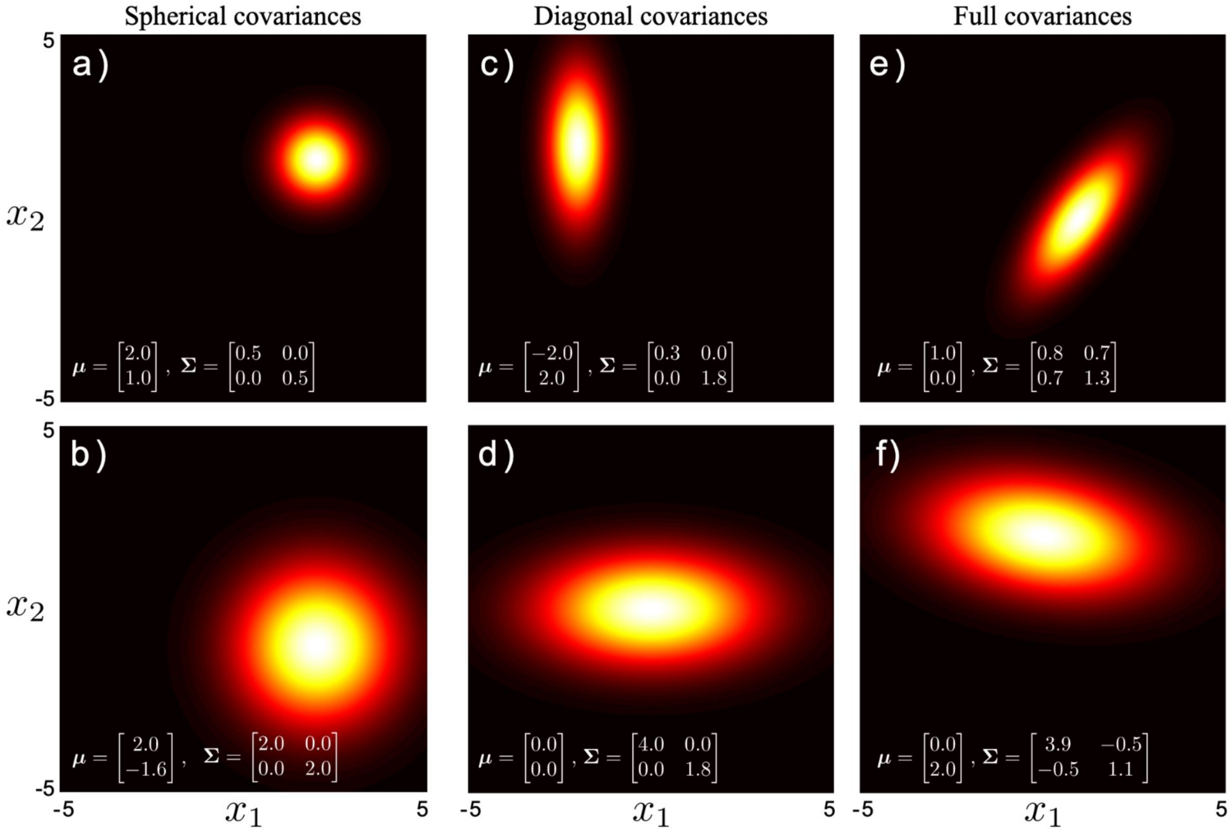
Multivariate Gaussian distribution

$$\mathcal{N}(\mathbf{x}; \mu, \Sigma) = \frac{1}{|\Sigma|^{1/2} (2\pi)^{D/2}} \exp\left[-\frac{1}{2} (\mathbf{x} - \mu)^T \cdot \Sigma^{-1} \cdot (\mathbf{x} - \mu)\right]$$

μ is *[D x 1]* vector that describes the position of the distribution

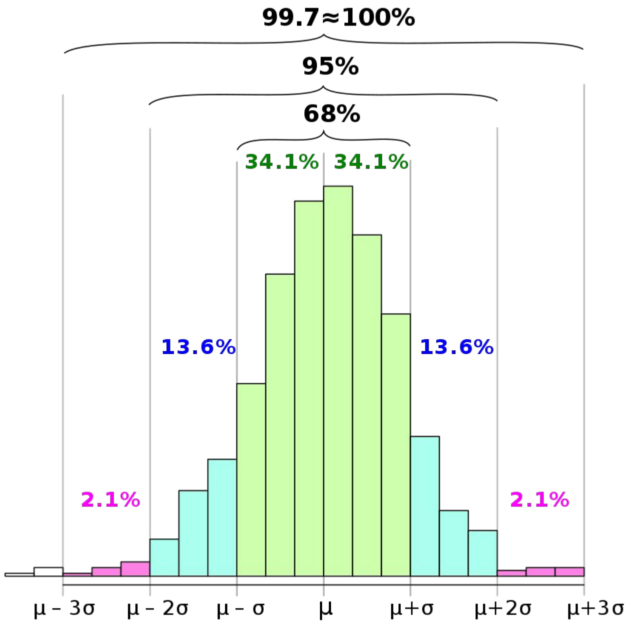
Σ is *[D x D]* positive definite matrix
(implying that $\boldsymbol{z}^{\boldsymbol{T}} \cdot \boldsymbol{\Sigma} \cdot \boldsymbol{z}$ is positive for any real vector $\boldsymbol{z}$)
and describes the shape of the distribution

$$\mathbf{\Sigma}_{spher} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \quad \mathbf{\Sigma}_{diag} = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \quad \mathbf{\Sigma}_{full} = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 \end{bmatrix}$$



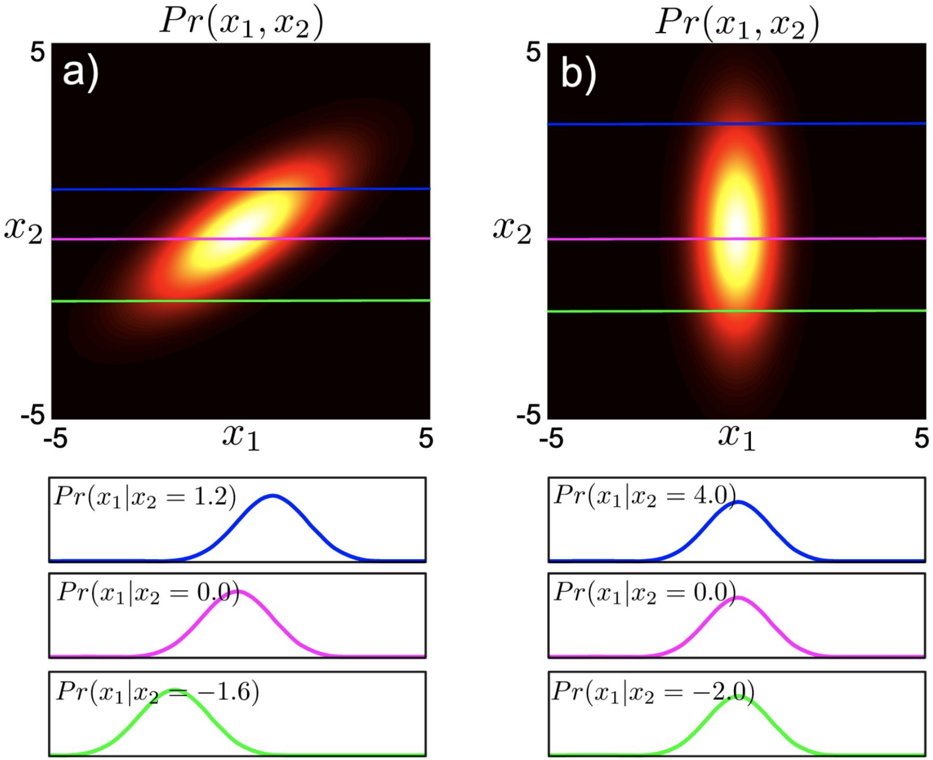
Simon J.D. Prince, “Computer Vision: Models, Learning, and Inference”

Multivariate Gaussian distribution



n = 1	99.73 %
n = 10	97.33 %
n = 100	76.31 %
n = 300	44.44 %
n = 1000	6.70 %

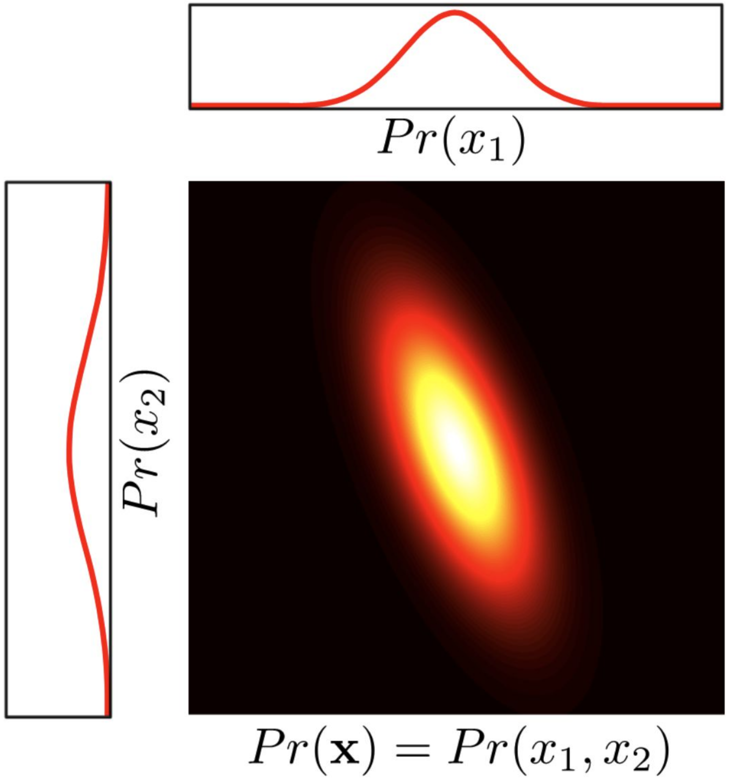
Conditional distributions (of multivariate Gaussian)



If we take any multivariate normal distribution, fix a subset of the variables, and look at the distribution of the remaining variables, this distribution will also take the form of a normal. The mean of this new normal depends on the values that we fixed the subset to, but the covariance is always the same.

Simon J.D. Prince, “Computer Vision: Models, Learning, and Inference”

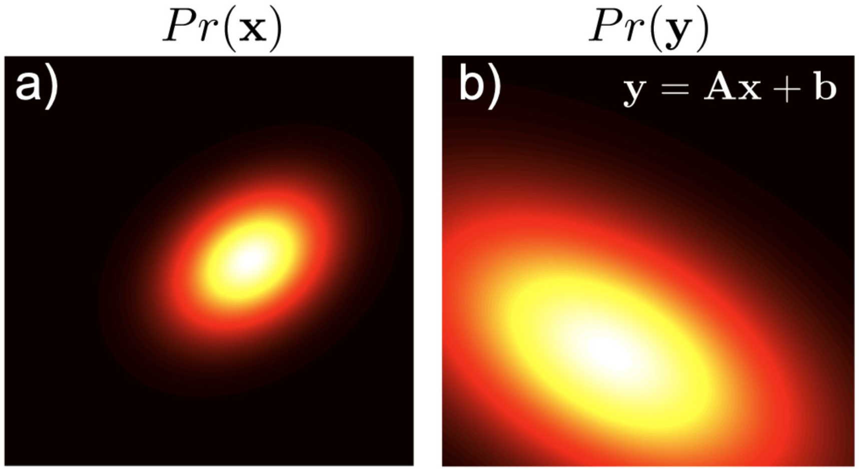
Marginal distributions (of multivariate Gaussian)



The marginal distribution of any subset of variables in a normal distribution is also normally distributed. In other words, if we sum over the distribution in any direction, the remaining quantity is also normally distributed. To find the mean and the covariance of the new distribution, we can simply extract the relevant entries from the original mean and covariance matrix.

Simon J.D. Prince, “Computer Vision: Models, Learning, and Inference”

Linear transformations of variables



The form of the multivariate normal is preserved under linear transformations

y = Ax + b

If the original distribution was

$$P(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mu, \mathbf{\Sigma})$$

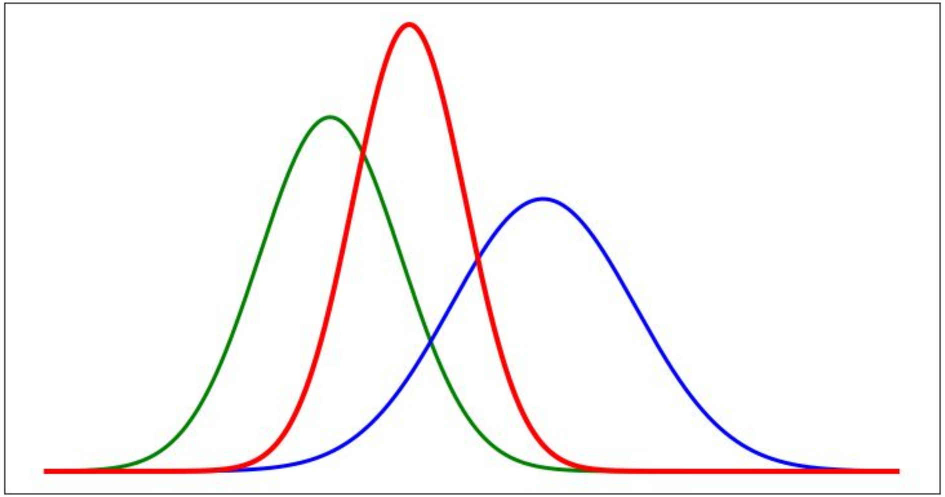
then the transformed variable **y** is distributed as:

$$Pr(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mathbf{A}\mu + \mathbf{b}, \mathbf{A\Sigma A}^T)$$

This relationship provides a simple method to draw samples from a normal distribution with mean μ and covariance Σ. We first draw a sample x from a standard normal distribution (μ = 0 and Σ = I) and then apply the transform y = $\Sigma^{1/2}$x + μ [“*Reparameterization Trick*”]

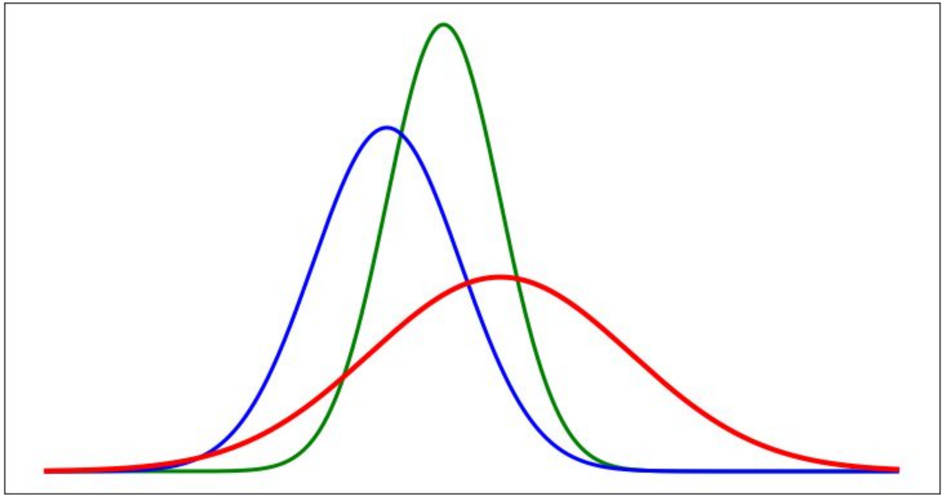
Simon J.D. Prince, “Computer Vision: Models, Learning, and Inference”

Product of two Gaussians



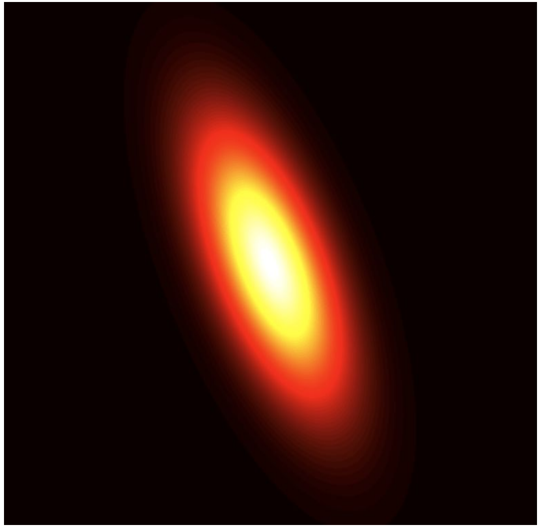
The product of any two normals **N1** and **N2** is proportional to a third normal **N3** distribution, with a mean between the two original means and a variance that is smaller than either of the original distributions.

Sum of two Gaussians



The sum of any two normals **N1** and **N2** is proportional to a third normal **N3** distribution, with a mean as a sum of two original means and a variance that is also sum of original variances

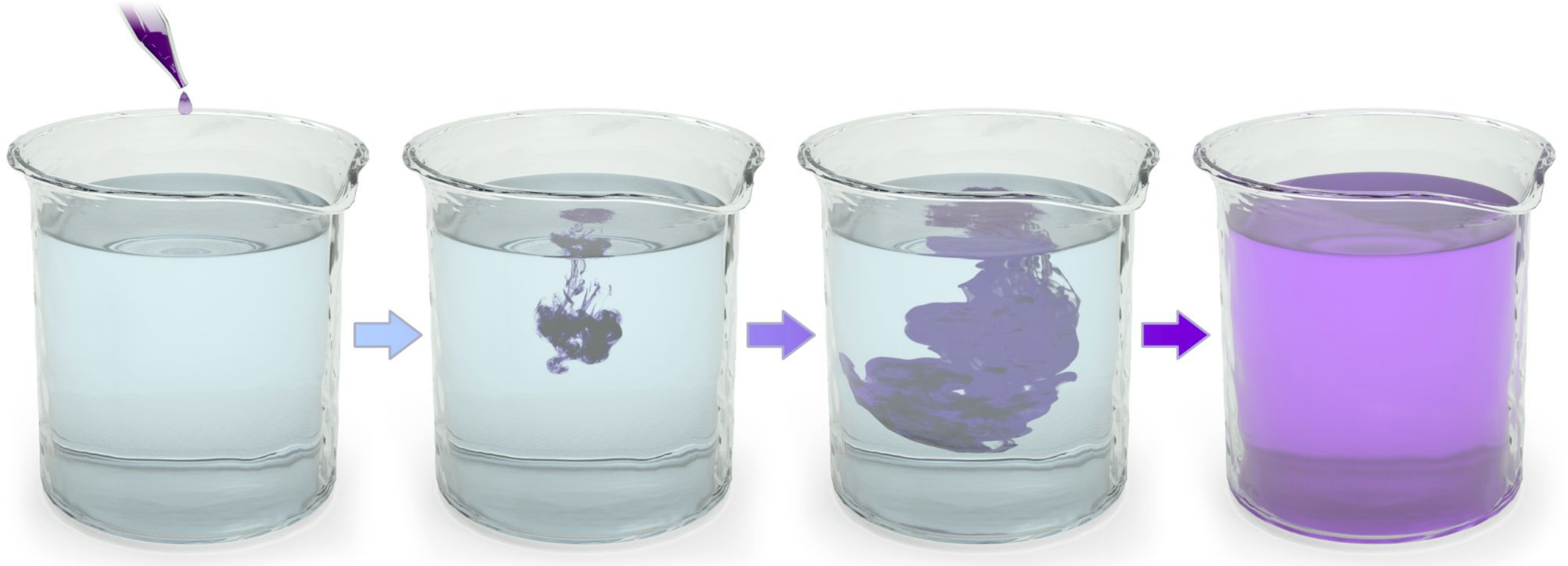
Other nice properties



- KL divergence between two Gaussians has a nice form;
- the (inverse)Fourier transform of a Gaussian is another Gaussian
- the convolution of two Gaussians is another Gaussian

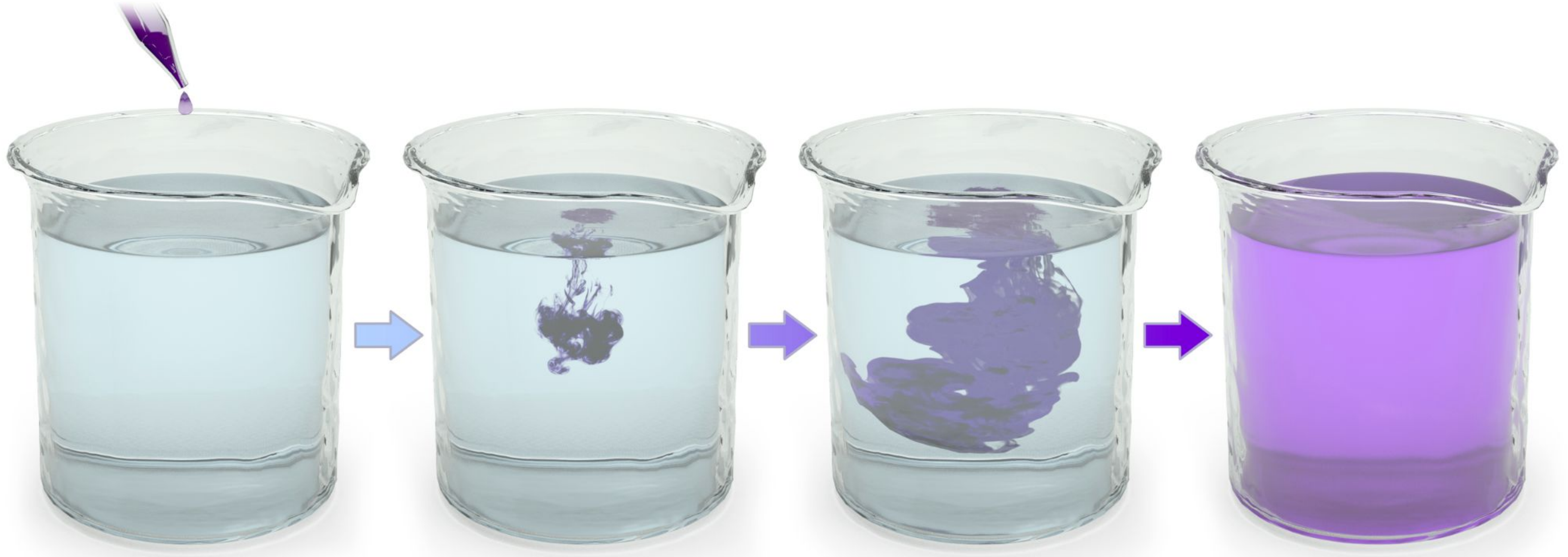
Content of today's lecture

- Generative modeling
  - appetiser
  - approaches
- Diffusion models
  - intro
  - tale of marvelous Gaussians
  - more details
- Stable Diffusion



Structure ⟶ Uniform distribution

https://en.wikipedia.org/wiki/Diffusion

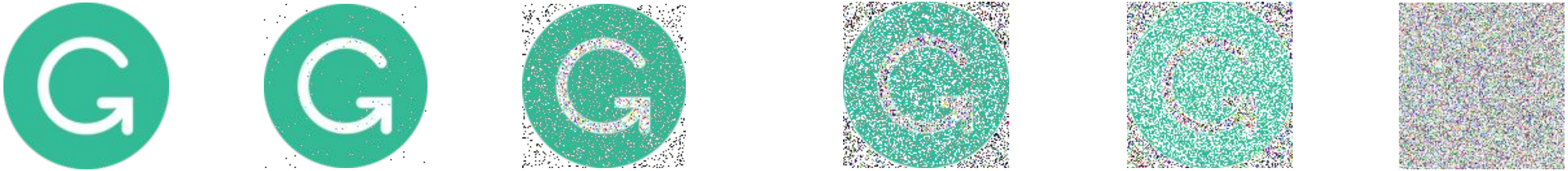


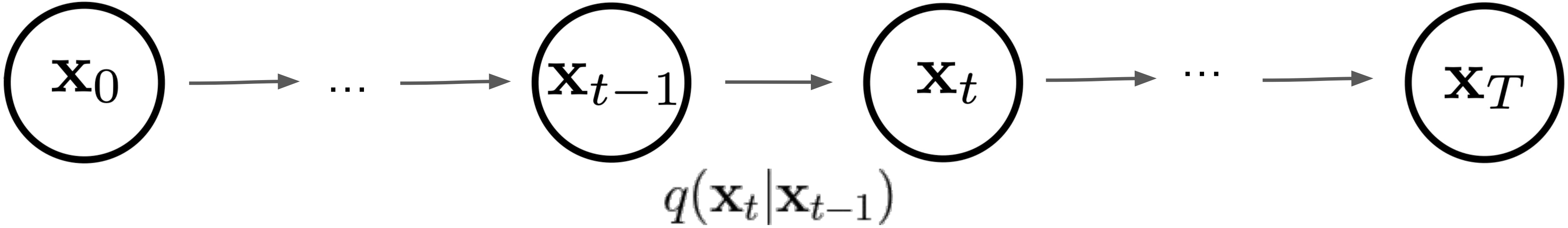
Structure ← Uniform distribution

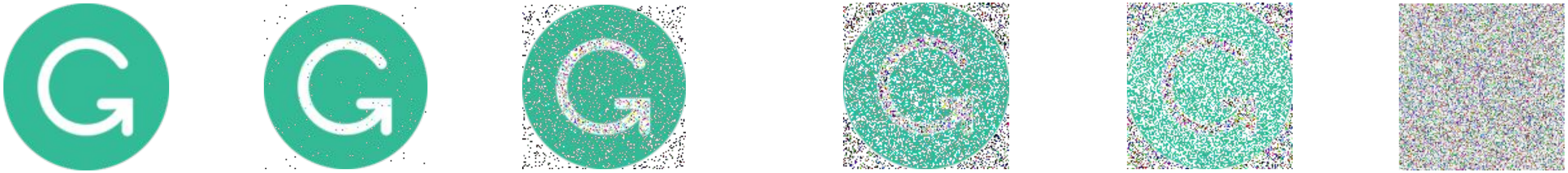
https://en.wikipedia.org/wiki/Diffusion

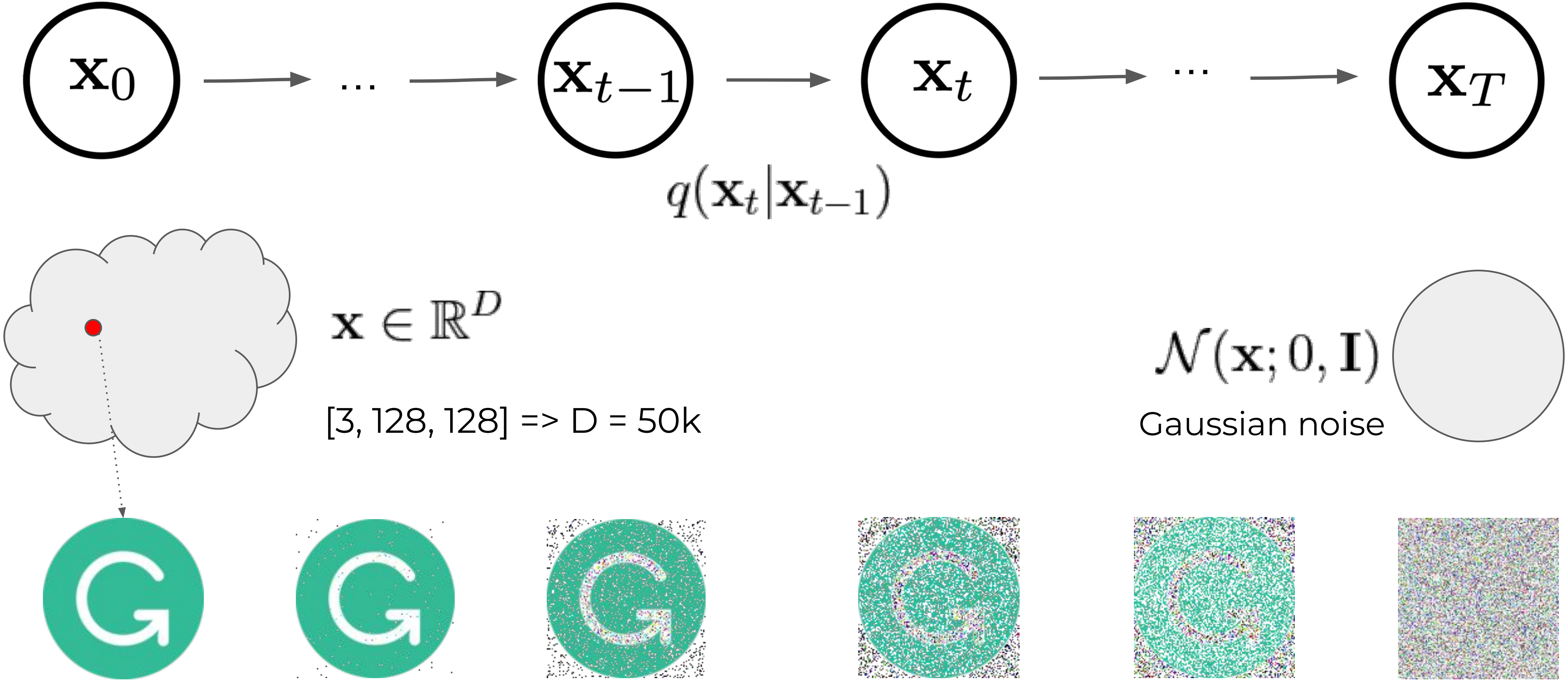
- Destroy all structure in data distribution using diffusion process
- Learn reversal of diffusion process
- *Reverse diffusion process* is the *model* of the data

MIT 6.S192 - Lecture 22: Diffusion Probabilistic Models, Jascha Sohl-Dickstein




$\mathbf{x}_0$
$\cdots$
$\mathbf{x}_{t-1}$
$\mathbf{x}_t$
$\cdots$
$\mathbf{x}_T$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$





$\mathbf{x}_0 \rightarrow \cdots \rightarrow \mathbf{x}_{t-1} \rightarrow \mathbf{x}_t \rightarrow \cdots \rightarrow \mathbf{x}_T$

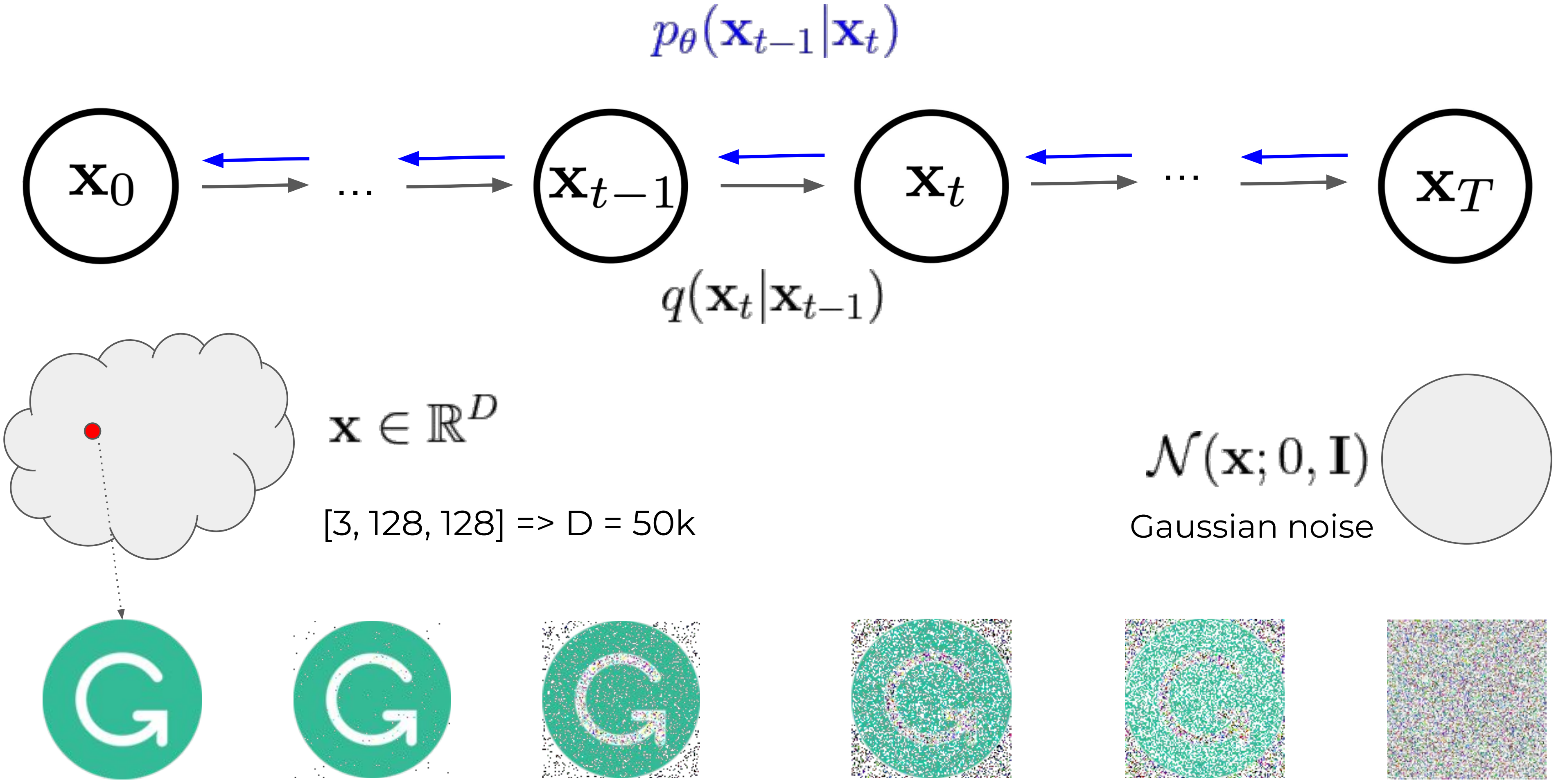
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$\mathbf{x} \in \mathbb{R}^D$

[3, 128, 128] => D = 50k

$\mathcal{N}(\mathbf{x}; 0, \mathbf{I})$

Gaussian noise



$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

$\mathbf{x}_0$ ... $\mathbf{x}_{t-1}$ $\mathbf{x}_t$ ... $\mathbf{x}_T$

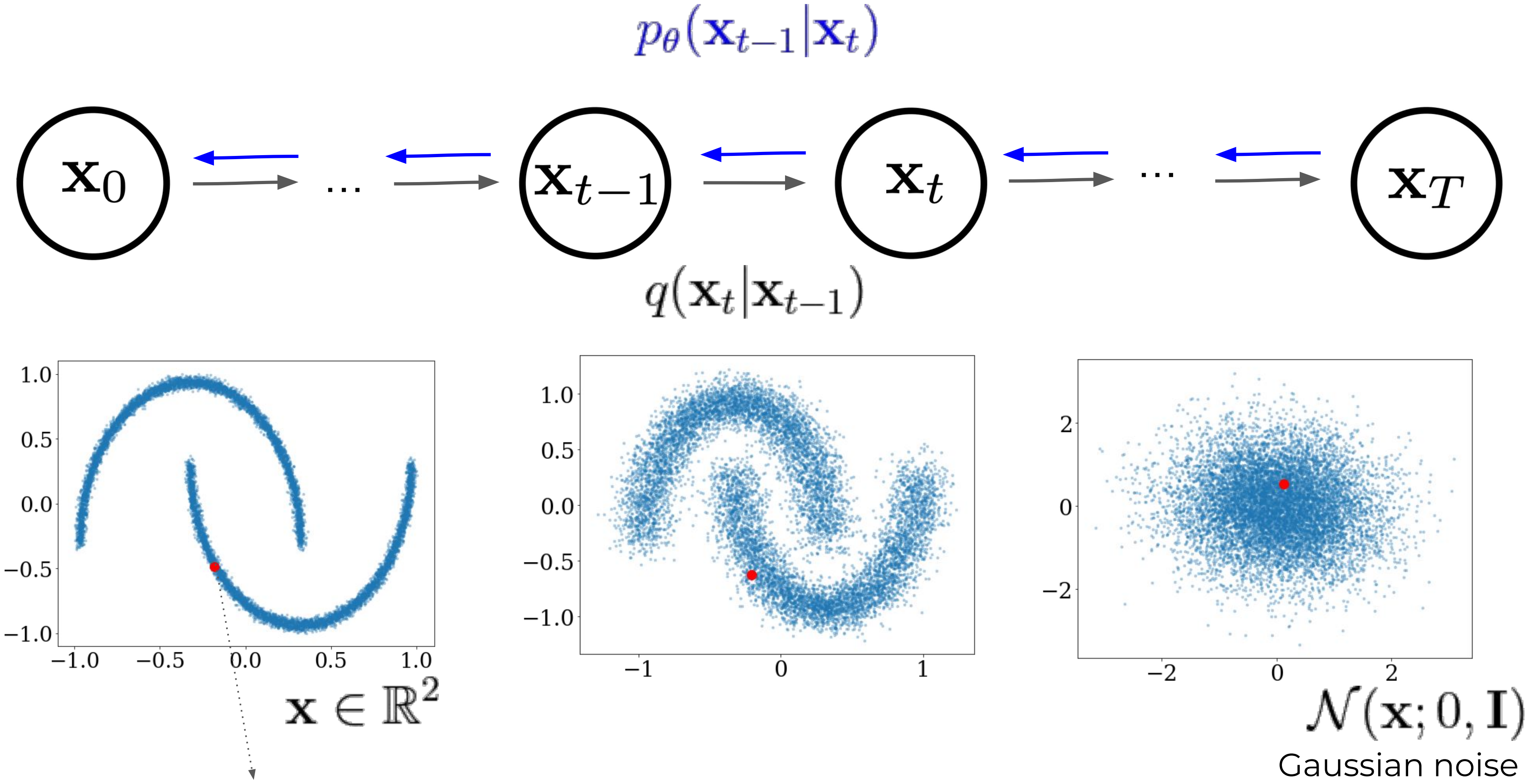
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$\mathbf{x} \in \mathbb{R}^D$

[3, 128, 128] => D = 50k

$\mathcal{N}(\mathbf{x}; 0, \mathbf{I})$

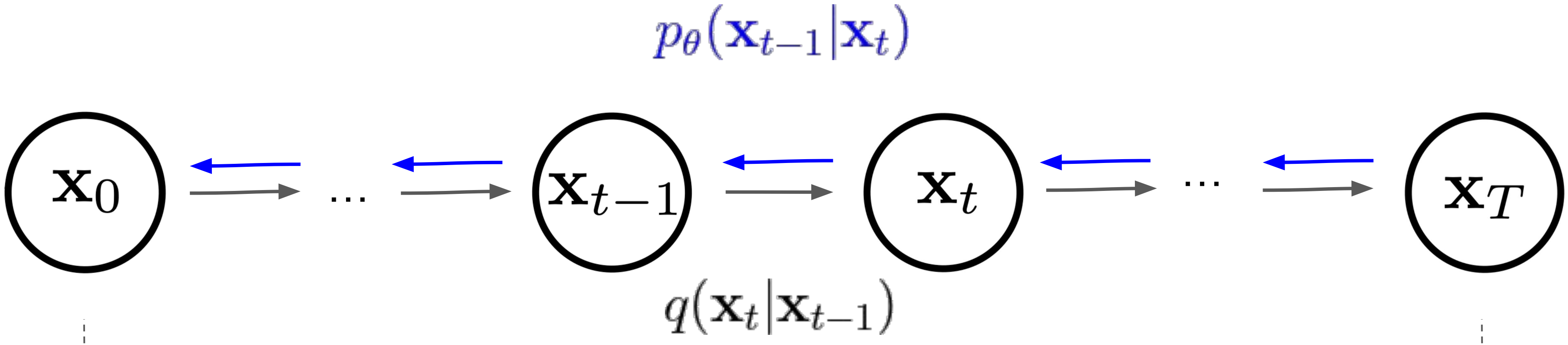
Gaussian noise


$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_0$
...
$\mathbf{x}_{t-1}$
$\mathbf{x}_t$
...
$\mathbf{x}_T$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
$\mathbf{x} \in \mathbb{R}^2$
1.0
0.5
0.0
−0.5
−1.0
−1
0
1
2
0
−2
−2
0
2
$\mathcal{N}(\mathbf{x}; 0, \mathbf{I})$
Gaussian noise

[-0.18, -0.49]

[-0.20, -0.62]

[0.12, 0.54]

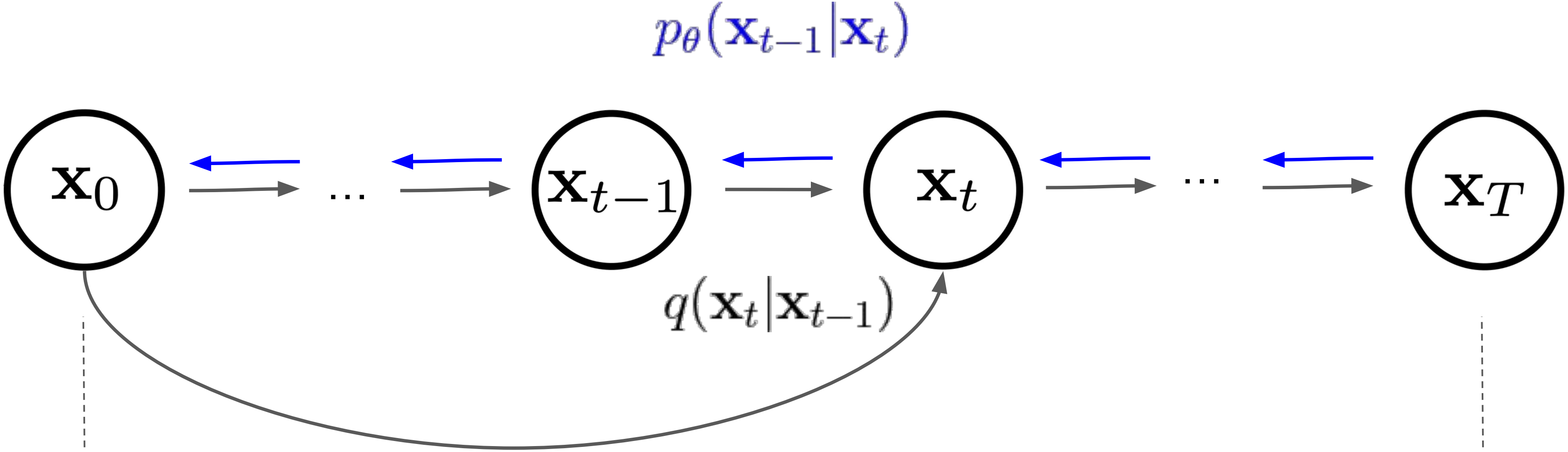

$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_0$
...
$\mathbf{x}_{t-1}$
$\mathbf{x}_t$
...
$\mathbf{x}_T$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$

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

$$0 < \beta_1 < \beta_2 < \ldots < \beta_T < 1$$

$\mathcal{N}(\mathbf{x}; \mathbf{x}_0, \mathbf{0})$ limiting cases $\mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{I})$

Gaussian noise

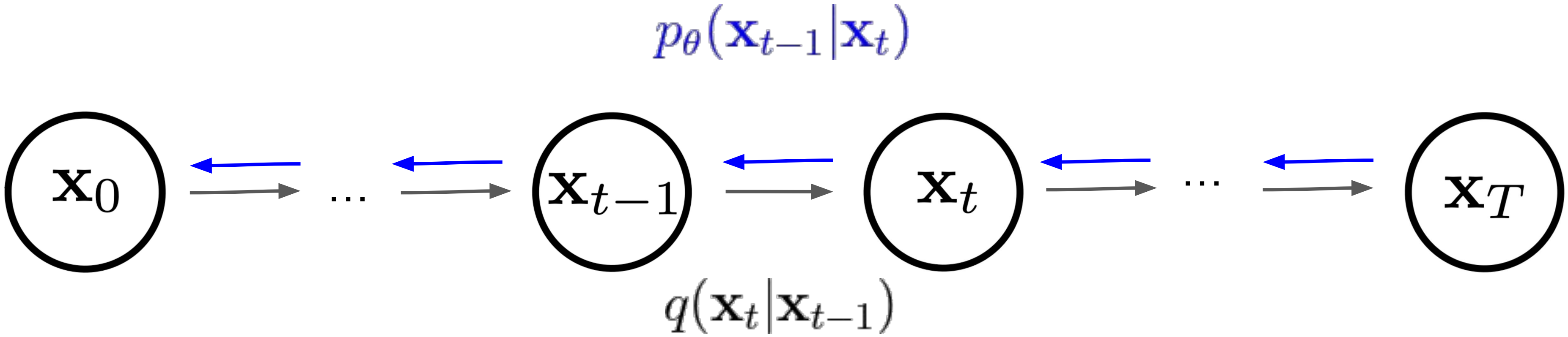


$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

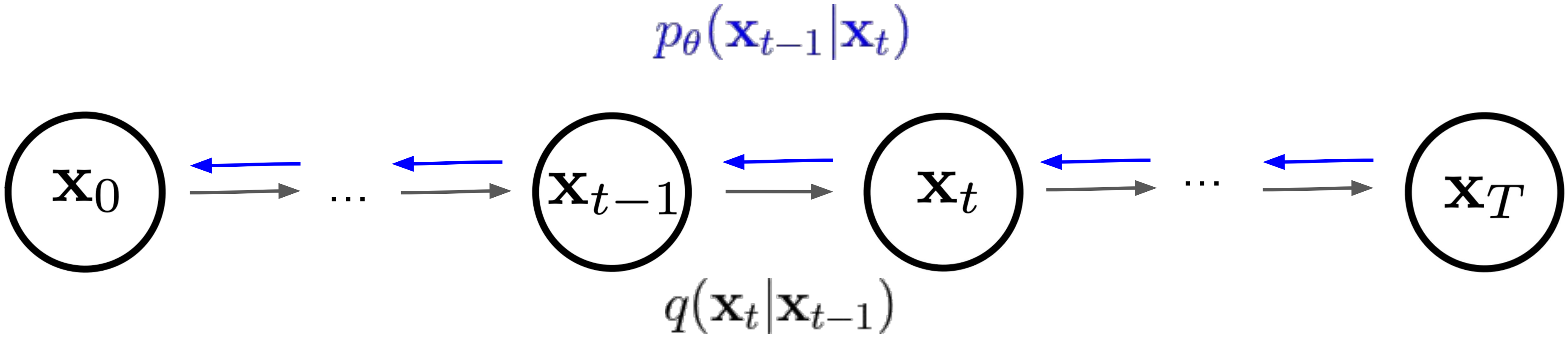
$$\bar{\alpha}_t := \prod_{s=1}^{T} \alpha_s \quad \alpha_t := 1 - \beta_t$$

$\mathcal{N}(\mathbf{x}; \mathbf{x}_0, \mathbf{0})$ limiting cases $\mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{I})$

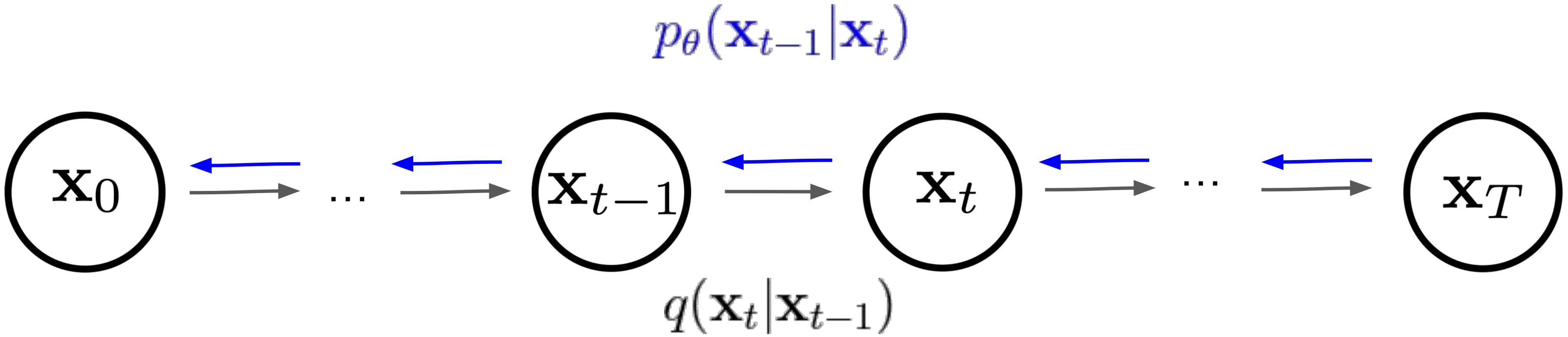
Gaussian noise



$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

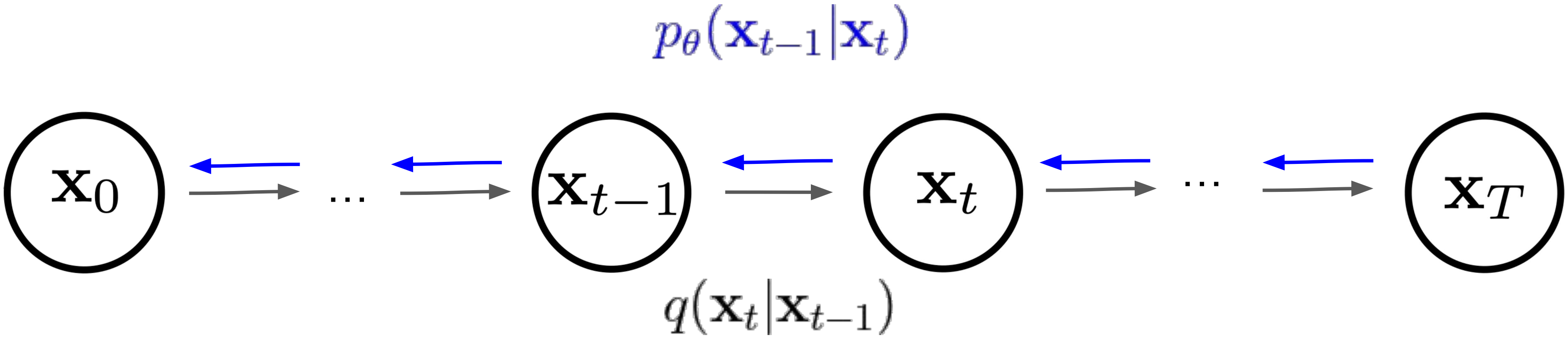


$$\begin{aligned} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \\ &= \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}) \end{aligned}$$



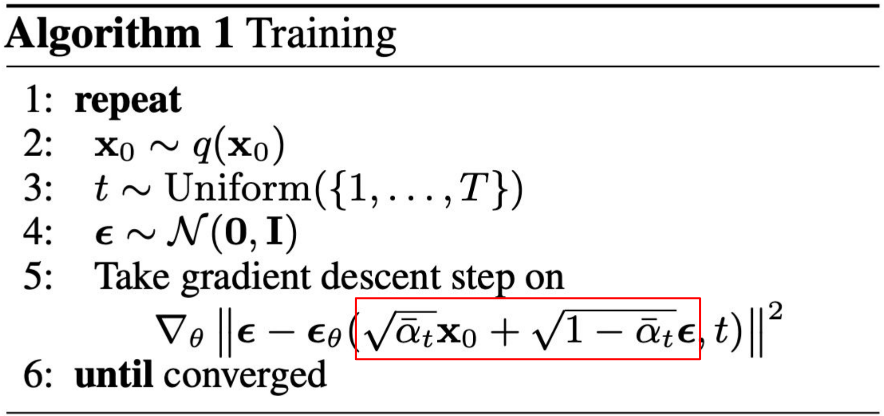
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_0$
...
$\mathbf{x}_{t-1}$
$\mathbf{x}_t$
...
$\mathbf{x}_T$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$$\begin{aligned} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \\ &= \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\text{Trainable network}}, \sigma_t^2 \mathbf{I}) \end{aligned}$$



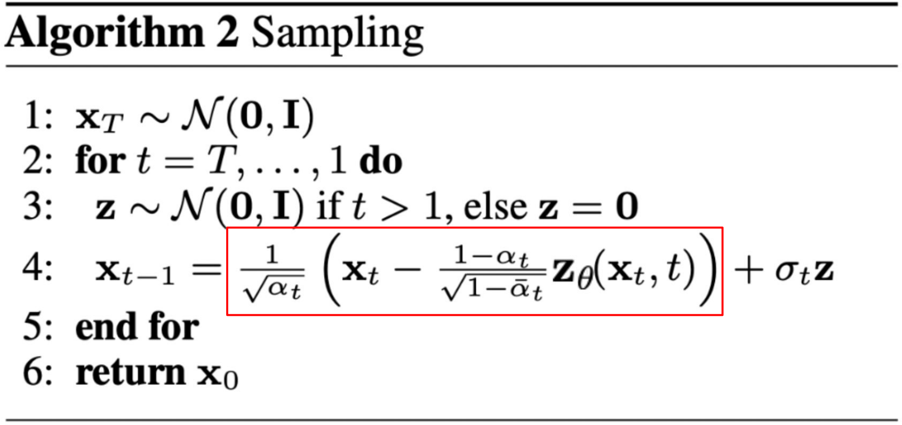
$$\begin{aligned} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \\ &= \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}) \end{aligned}$$

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\alpha_t}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \underbrace{\epsilon(\mathbf{x}_t, t)}_{\text{Trainable network}}\right)$$



Algorithm 1 Training

```
repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
      $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boxed{\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}}, t) \right\|^2$
until converged
```

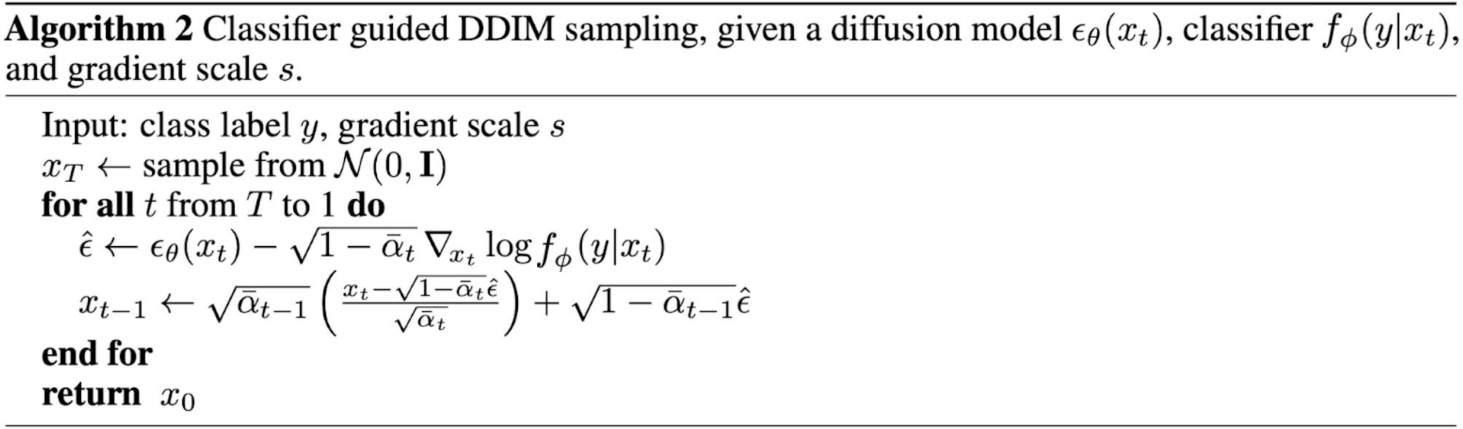


Algorithm 2 Sampling

```
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ do
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \boxed{\frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \mathbf{z}_\theta(\mathbf{x}_t, t) \right)} + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$
```

Conditional generation

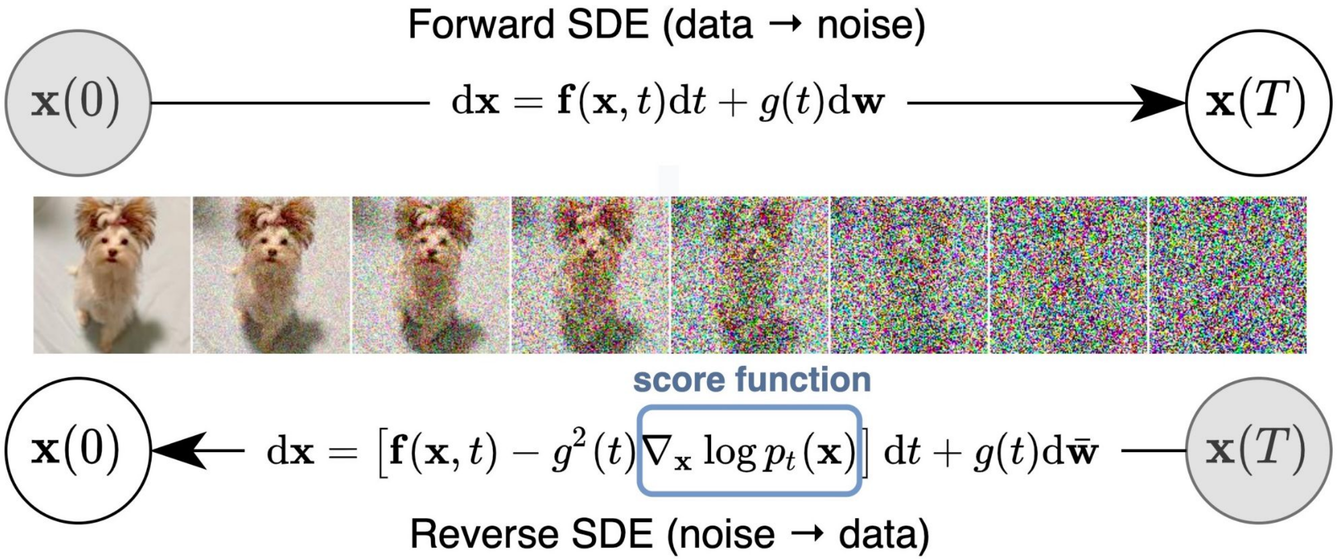
To explicit incorporate class information into the diffusion process, Dhariwal & Nichol (2021) trained a classifier $f_\phi(y|\mathbf{x}_t, t)$ on noisy image and use gradients $\nabla_\mathbf{x} \log f_\phi(y|\mathbf{x}_t, t)$ to guide the diffusion sampling process toward the target class label .



Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $f_\phi(y|x_t)$, and gradient scale s.

```
Input: class label $y$, gradient scale $s$
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all $t$ from $T$ to 1 do
    $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1 - \bar{\alpha}_t}\, \nabla_{x_t} \log f_\phi(y|x_t)$
    $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left( \frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1 - \bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return $x_0$
```

Score-based formulation



Forward SDE (data → noise)

$\mathbf{x}(0)$

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$$

$\mathbf{x}(T)$

score function

$\mathbf{x}(0)$

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$$

$\mathbf{x}(T)$

Reverse SDE (noise → data)

Generative Modeling by Estimating Gradients of the Data Distribution | Yang Song

Content of today’s lecture

- Generative modeling
  - appetiser
  - approaches
- Diffusion models
  - intro
  - tale of marvelous Gaussians
  - more details
- Stable Diffusion

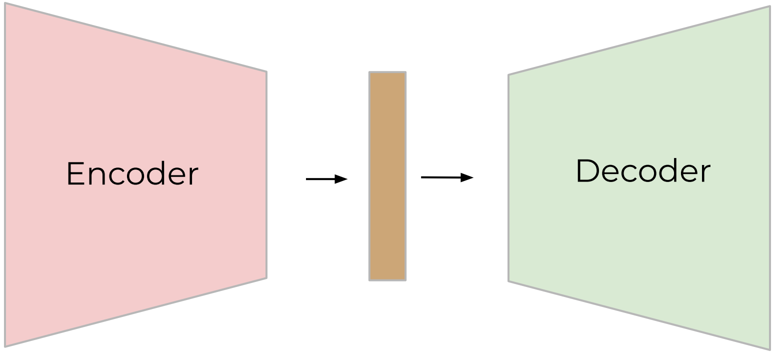
Stable Diffusion

[2112.10752] High-Resolution Image Synthesis with Latent Diffusion Models

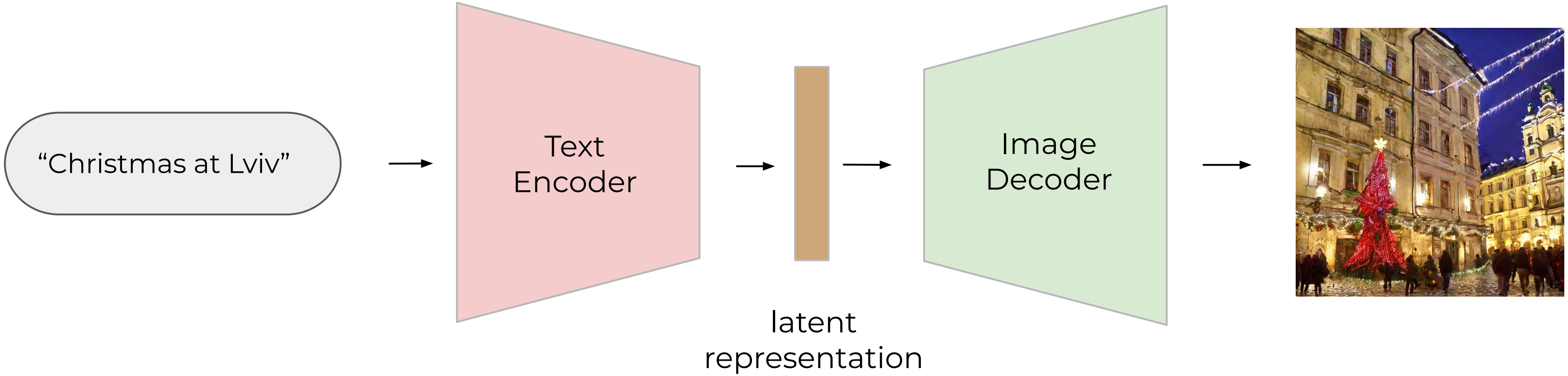
GitHub - CompVis/stable-diffusion: A latent text-to-image diffusion model

Official Announcement:
Stable Diffusion launch announcement — Stability AI

Stable Diffusion



Stable Diffusion



“Christmas at Lviv”

Text Encoder

latent representation

Image Decoder

Stable Diffusion

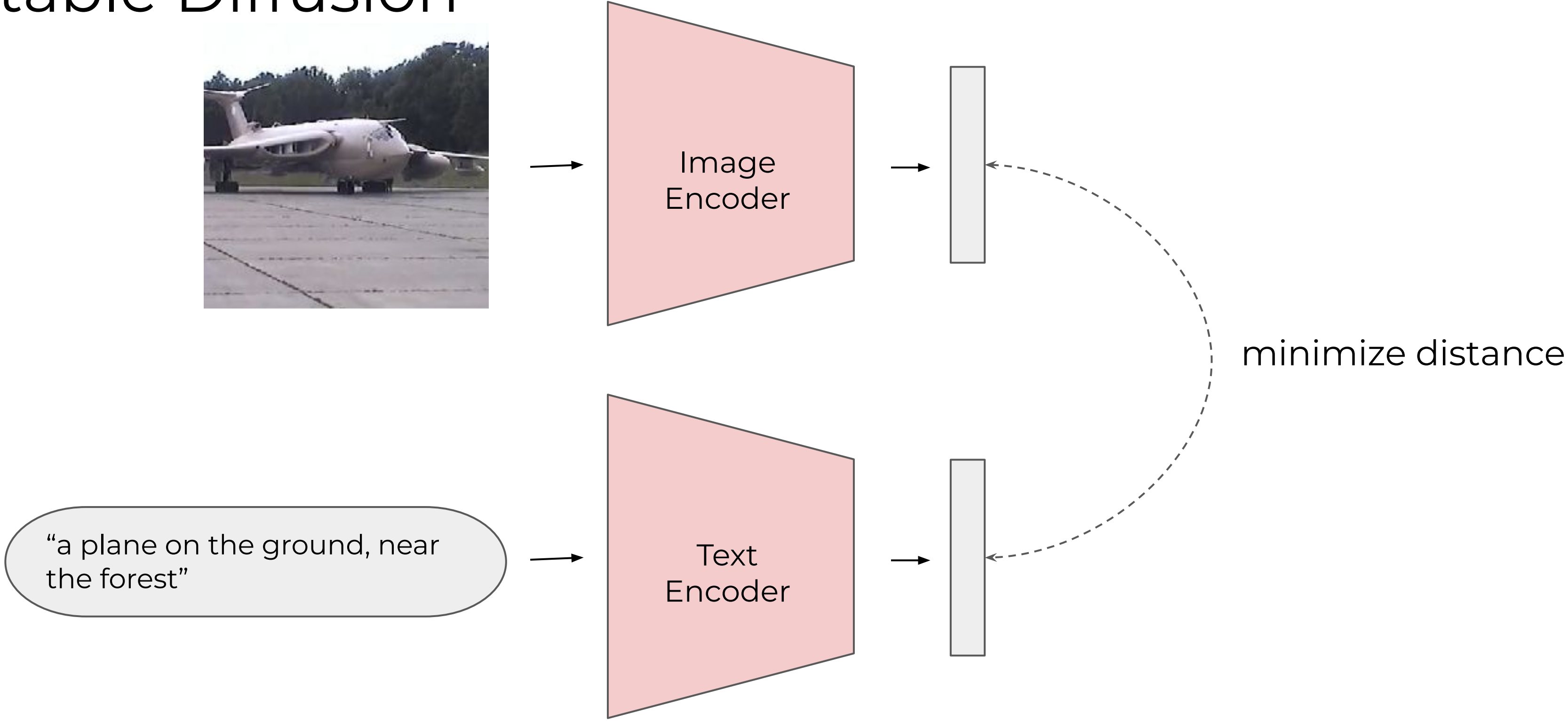


Image
Encoder

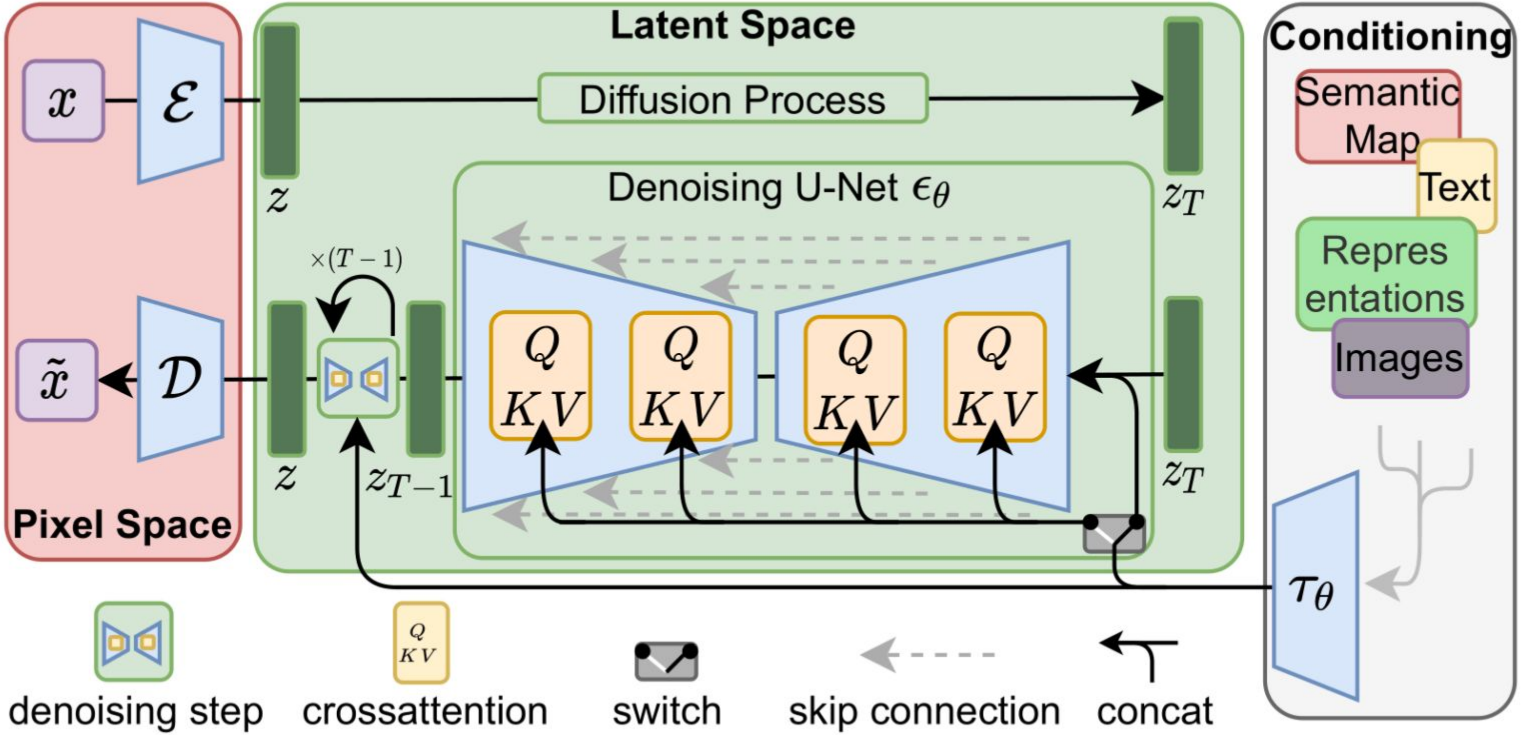
minimize distance

“a plane on the ground, near the forest”

Text
Encoder

[GitHub - openai/CLIP: Contrastive Language-Image Pretraining](#)
[GitHub - mlfoundations/open_clip: An open source implementation of CLIP.](#)

Stable Diffusion



[2112.10752] High-Resolution Image Synthesis with Latent Diffusion Models

Summary

- We are witnessing a revolution in computer-assisted creativity.

- Generation won't only be limited to images, already now there are video, 3D generation, and more is coming.