

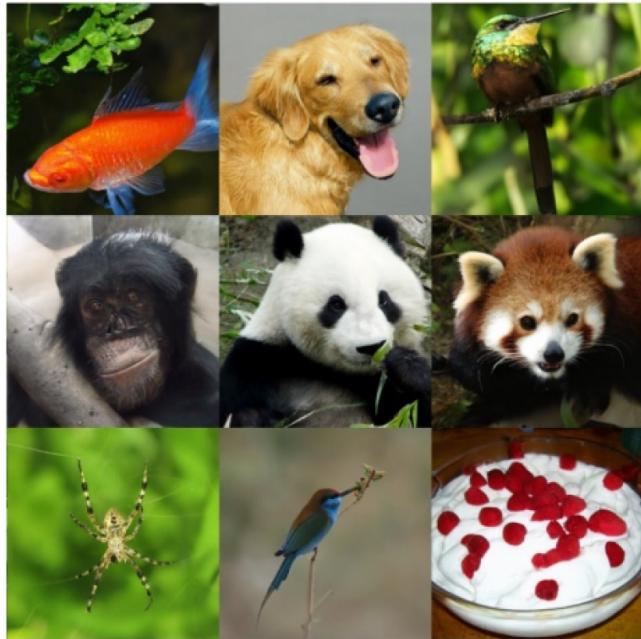
## 5. Diffusion Models

M. Ravasi

AI Summer School @ KAUST

# Diffusion models: what's the hype about?

---



Diffusion Models **Beat GANs** on Image  
Synthesis [Dharwal & Nichol](#),  
[OpenAI, 2021](#)

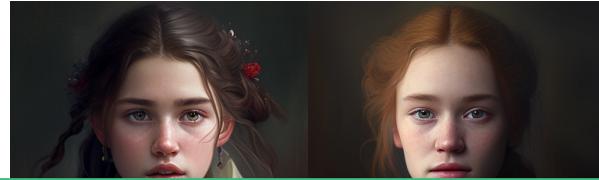
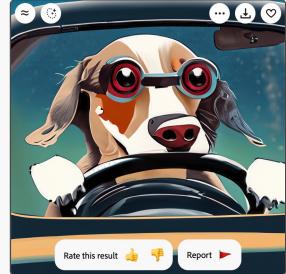
# Diffusion models: what's the hype about?



 **Adobe Firefly (Beta)**



# Diffusion models: what's the hype about?



Text-to-image generative models

Diffusion → image network



 Adobe Firefly (Beta)



# Diffusion models: and the technology?

---



Developer: **OpenAI**

Release: Jan 2021 (v1), Jul 2022 (v2)

Tech: diffusion model conditioned on CLIP image embeddings



Developer: **StabilityAI**

Release: Aug 2022

Tech: latent diffusion model with CLIP ViT-L/14 text encoder



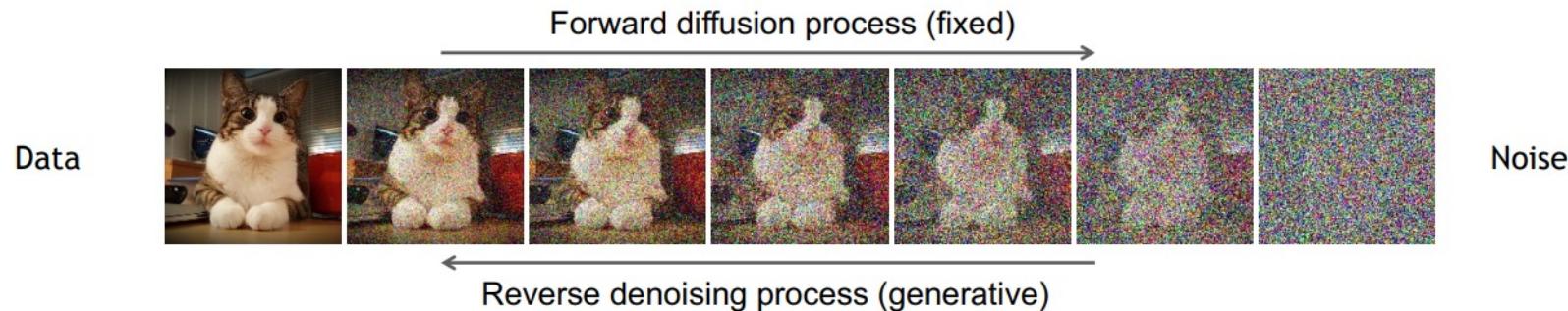
Developer: **Midjourney**

Release: Feb 2022 (v1), Mar 2023 (v5)

Tech: not disclosed fully

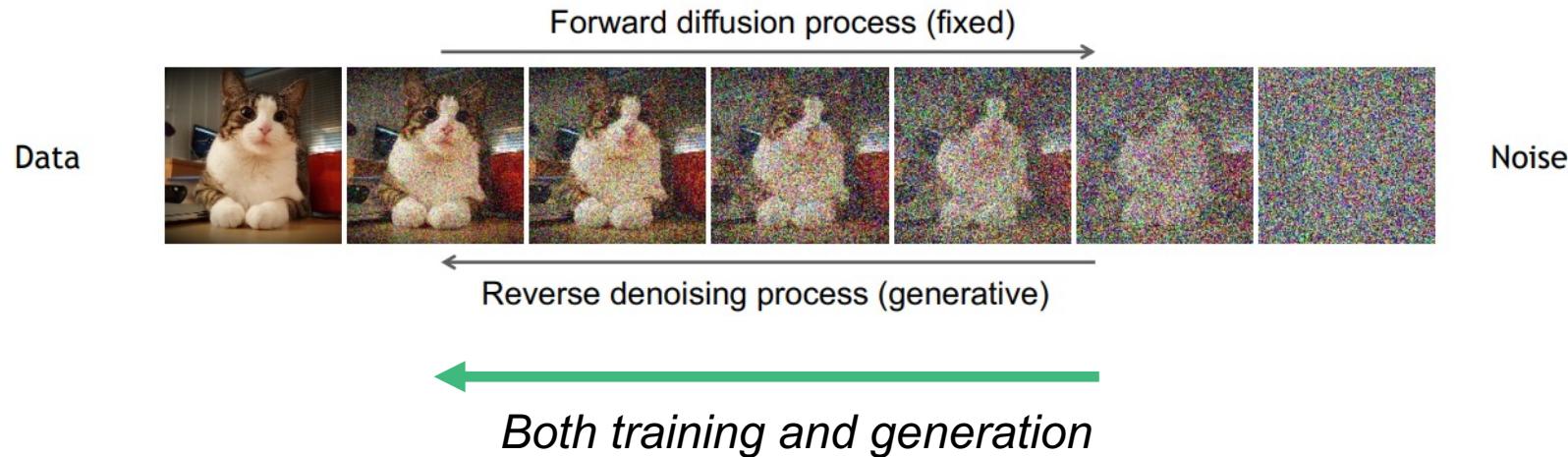
# Diffusion models: intuition

*Family of neural networks that can learn to **generate images** by gradually refining a Gaussian Noise realization and making it more and more into a real image at each timestep*



# Diffusion models: intuition

*Family of neural networks that can learn to **generate images** by gradually refining a Gaussian Noise realization and making it more and more into a real image at each timestep*

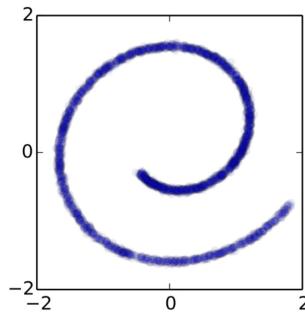


# Normalizing Flows: intuition

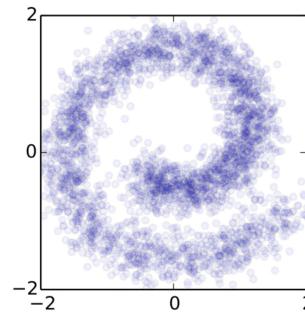
The forward trajectory

$$q(\mathbf{x}_{0:T})$$

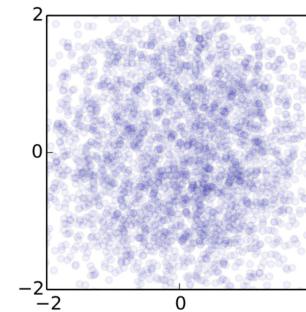

$$t = 0$$



$$t = \frac{T}{2}$$

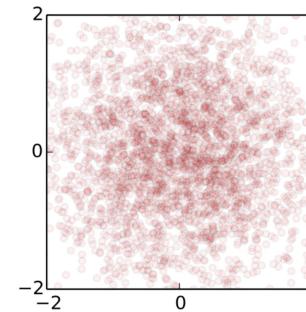
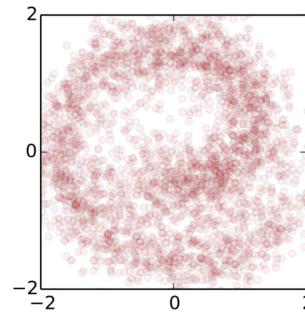
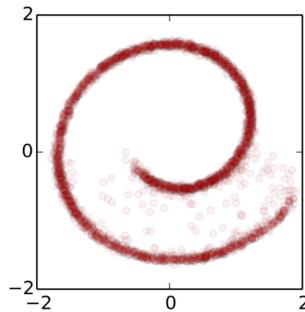


$$t = T$$



The reverse trajectory

$$p_{\theta}(\mathbf{x}_{0:T})$$

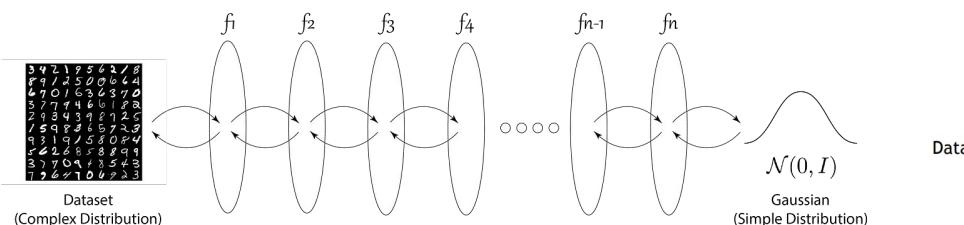



Sohl-Dickstein et al. (2015)

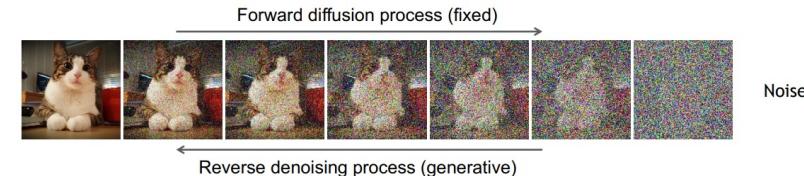
# Diffusion models vs NFs

Diffusion models and NFs look similar at first.

Both operate in **2 directions**: one making something complex (image) into something simple (noise) and one doing the opposite.



NF



Diffusion model

# Diffusion models vs NFs

---

Diffusion models and NFs look similar at first.

Both operate in **2 directions**: one making something complex (image) into something simple (noise) and one doing the opposite.

But there are some **fundamental differences**:

- No need for **invertible transformations** in DMs
- **Simpler training process** for DMs (no log-det of Jacobian)
- **Only sampling** for DMs

# Diffusion models vs NFs

---

Diffusion models and NFs look similar at first.

Both operate in **2 directions**: one making something complex (image) into something simple (noise) and one doing the opposite.

But there are some **fundamental differences**:

- No need for **invertible transformations** in DMs
- **Simpler training process** for DMs (no log-det of Jacobian)
- **Only sampling** for DMs

# Diffusion models: 3 in 1

Different ways to achieve the same goal (unified lately into unique framework)

## DDPMs

(Denoising Diffusion  
Probabilistic Models)

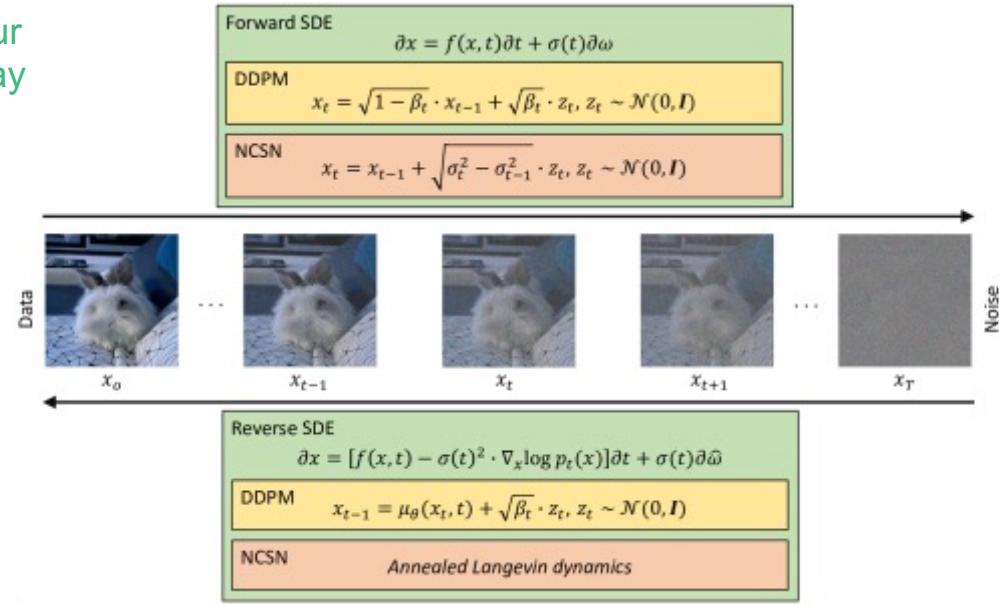
Most of our  
focus today

## NCSNs

(Noise Conditioned  
Score Networks)

## SDEs

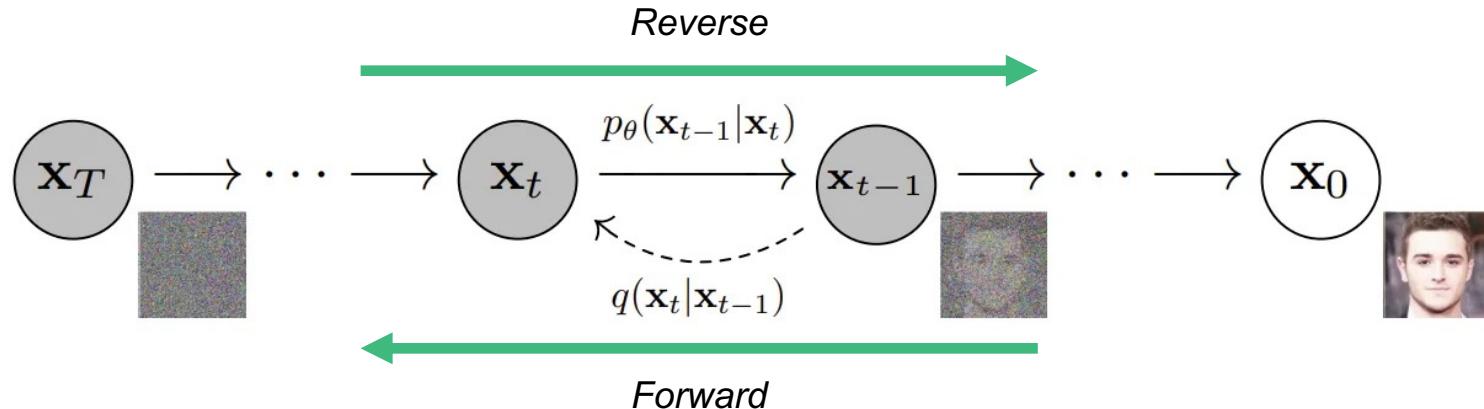
(Stochastic differential  
equations)



# Denoising Diffusion Probabilistic Models

Forward: inspired from non-equilibrium thermodynamics, Markov chain of diffusion steps to **slowly add random noise to data until output is pure noise – fixed**

Reverse: reversed Markov chain to **slowly remove noise** and generate outputs that belong to the distribution of training data - **learned**



## DDPMs: forward

---

Starting from a training sample  $x_0 \sim q(x)$  (subscript: diffusion step), we define the diffusion process as follows (i.e., write  $x_t$  with  $t = 1, 2, \dots, T$  as function of  $x_{t-1}$ ):

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

## DDPMs: forward

Starting from a training sample  $x_0 \sim q(x)$  (subscript: diffusion step), we define the diffusion process as follows (i.e., write  $x_t$  with  $t = 1, 2, \dots, T$  as function of  $x_{t-1}$ ):

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

Normal distribution (since we keep adding Gaussian noise)

Mean and covariance of conditional distribution

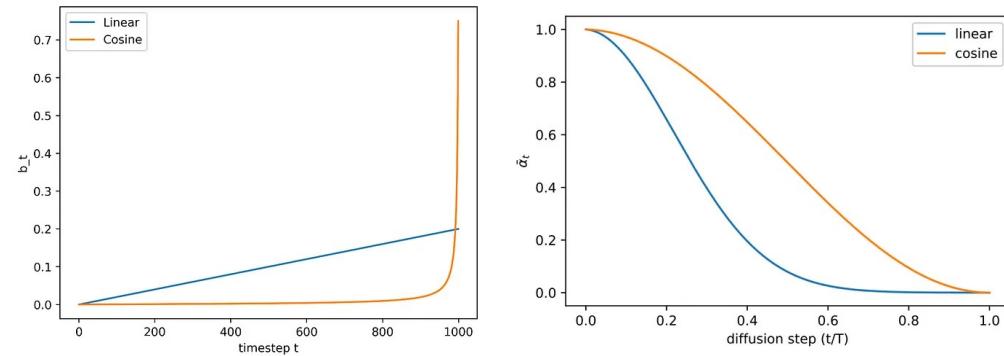
# DDPMs: forward

Noise variance changes over steps (**variance scheduler**):

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

$$\alpha_t := 1 - \beta_t$$

$$\bar{\alpha}_t := \prod_{s=1}^t \alpha_s$$



**DDPM paper (Ho et al., 2020):** linear scheduler with  $\beta_1 = 1e^{-4}$ ,  $\beta_{T=1000} = 0.02$  (always small compared to  $x_0 \in [-1,1]$ )

**Improved DDPM paper (Nichol and Dhariwal, 2021):** cosine-based scheduler (near-linear drop in the middle of the training process and smaller changes at both ends)

## DDPMs: forward

Using the **reparametrization trick** (remember VAEs!), we can write

$$\mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \epsilon \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

More importantly, we can find a **closed-form solution for  $x_t$  directly from  $x_0$**  (much cheaper than generating all steps – T=1000 in DDPM paper!)

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

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + (1 - \bar{\alpha}_t) \epsilon$$

# DDPMs: forward

Derivation:

$$\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_{t-1} \quad ; \text{where } \boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \dots \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$= \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \bar{\boldsymbol{\epsilon}}_{t-2} \quad ; \text{where } \bar{\boldsymbol{\epsilon}}_{t-2} \text{ merges two Gaussians (*).}$$

= ...

$$= \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}$$

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

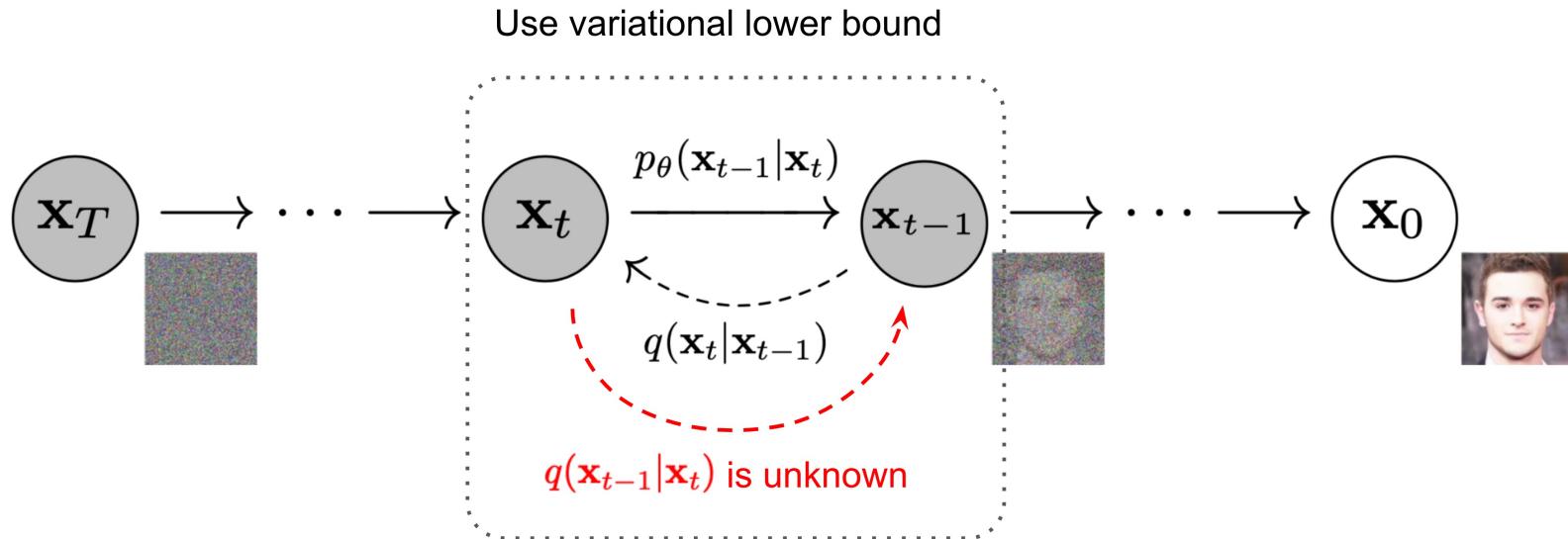
(\*) Recall that when we merge two Gaussians with different variance,  $\mathcal{N}(\mathbf{0}, \sigma_1^2 \mathbf{I})$  and  $\mathcal{N}(\mathbf{0}, \sigma_2^2 \mathbf{I})$ , the new distribution is  $\mathcal{N}(\mathbf{0}, (\sigma_1^2 + \sigma_2^2) \mathbf{I})$ . Here the merged standard deviation is

$$\sqrt{(1 - \alpha_t) + \alpha_t(1 - \alpha_{t-1})} = \sqrt{1 - \alpha_t \alpha_{t-1}}.$$

From <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

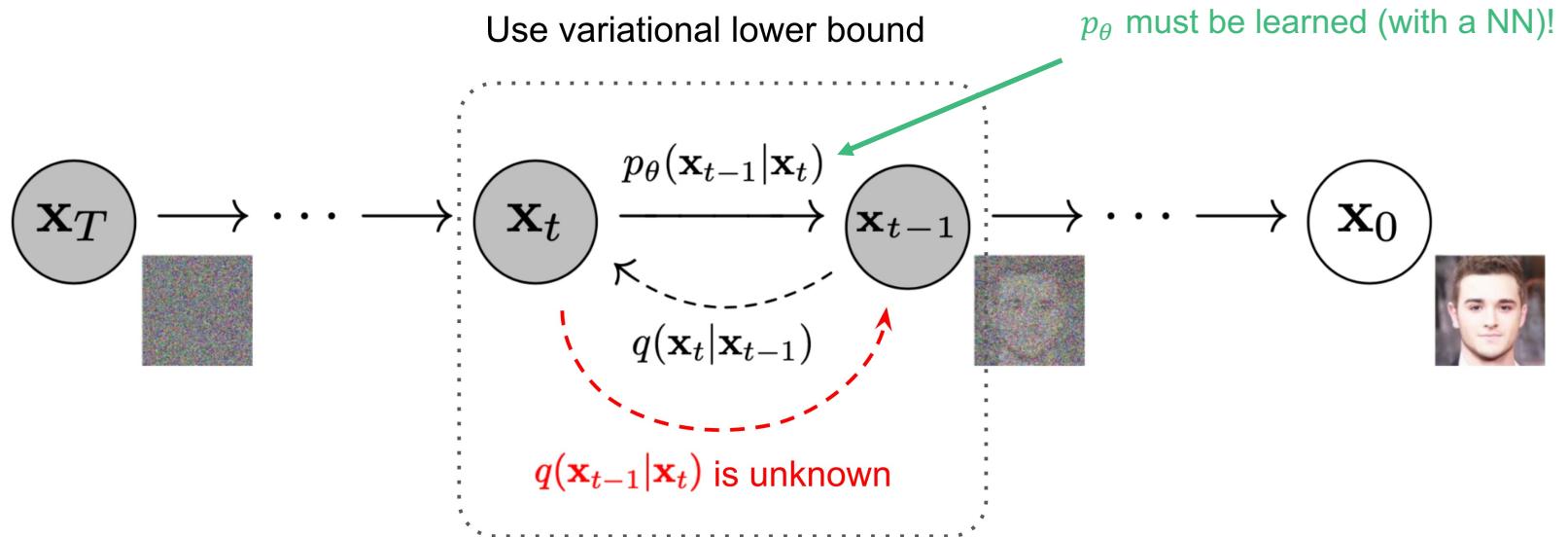
# DDPMs: reverse

Assuming we knew  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ , we could run the reverse process by sampling noise for  $\mathbf{x}_T$  and slowly removing it... **but we do not know this probability!**



# DDPMs: reverse

Assuming we knew  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ , we could run the reverse process by sampling noise for  $\mathbf{x}_T$  and slowly removing it... **but we do not know this probability!**



# DDPMs: reverse

Assuming only that  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ , is **Gaussian** (true for small  $\beta_t$ ) we could just learn its mean and covariance

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


The equation  $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_{\theta}(\mathbf{x}_t, t), \Sigma_{\theta}(\mathbf{x}_t, t))$  is displayed. Three arrows point from text below to specific parts of the equation: one arrow points to  $\mu_{\theta}(\mathbf{x}_t, t)$ , another points to  $\Sigma_{\theta}(\mathbf{x}_t, t)$ , and a third points to  $t$ .

**Parametrized** Mean and covariance of  
conditional distribution

$t$  is additionally fed to the network  
so it can learn to behave differently  
during the reverse process

# DDPMs: reverse

Assuming only that  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ , is **Gaussian** (true for small  $\beta_t$ ) we could just learn its mean and covariance

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

The equation  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$  is displayed. Three arrows point upwards from text descriptions to specific parts of the equation:

- A diagonal arrow points from the text "Parametrized Mean and covariance of conditional distribution" to the term  $\mu_\theta(\mathbf{x}_t, t)$ .
- A horizontal arrow points from the text "t is additionally fed to the network so it can learn to behave differently during the reverse process" to the term  $\Sigma_\theta(\mathbf{x}_t, t)$ .
- A vertical arrow points from the same text to the parameter  $t$  in the term  $\Sigma_\theta(\mathbf{x}_t, t)$ .

**Parametrized** Mean and covariance of  
conditional distribution

**t** is additionally fed to the network  
so it can learn to behave differently  
during the reverse process

**DDPM paper:** fixed covariance  $\Sigma_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$      $\sigma_t^2 = \beta_t$     or     $\sigma_t^2 = \bar{\alpha}_t$

**Improved DDPM paper:** learned covariance

## DDPMs: reverse

---

The combination of  $q(\mathbf{x}_t | \mathbf{x}_{t-1})$  and  $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$  can be seen as **VAE**

→ use the **ELBO** to minimize negative log-likelihood

Overall training loss (since we can sample each step independently, **we can optimize random terms of the loss**):

$$L = L_0 + L_1 + \dots + L_T$$

# DDPMs: reverse

The combination of  $q(\mathbf{x}_t | \mathbf{x}_{t-1})$  and  $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$  can be seen as **VAE**

→ use the **ELBO** to minimize negative log-likelihood

Overall training loss (since we can sample each step independently, **we can optimize random terms of the loss**):

$$L = L_0 + L_1 + \dots + L_T$$


Treated a bit specially...      KL divergence of 2 gaussians → L2 loss!

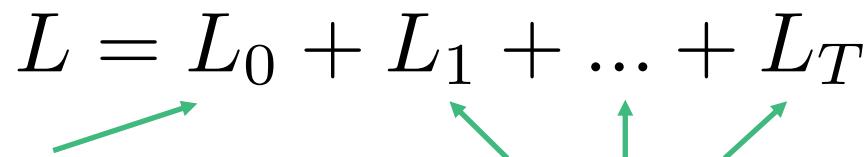
$$\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 = \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{(1 - \bar{\alpha}_t)}\epsilon, t)\|^2$$

# DDPMs: reverse

The combination of  $q(\mathbf{x}_t | \mathbf{x}_{t-1})$  and  $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$  can be seen as **VAE**

→ use the **ELBO** to minimize negative log-likelihood

Overall training loss (since we can sample each step independently, **we can optimize random terms of the loss**):

$$L = L_0 + L_1 + \dots + L_T$$


Treated a bit specially...      KL divergence of 2 gaussians → L2 loss!

$$\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 = \|\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{(1 - \bar{\alpha}_t)} \epsilon, t)}_{\mathbf{x}_t}\|^2$$

# DDPMs: reverse (simplified derivation)

1. Write expression for  $L_t$  ( $t = 1, \dots, T - 1$ ) losses

$$L_t = D_{\text{KL}}(q(\mathbf{x}_t | \mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t | \mathbf{x}_{t+1}))$$

↑  
Extra conditioning on  $x_0$  to make it tractable

Given 2 gaussians, there is closed-form solution:

$$L_t = \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[ \frac{1}{2\|\Sigma_\theta(\mathbf{x}_t, t)\|_2^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right]$$

# DDPMs: reverse (simplified derivation)

2. Need to write an expression for the mean of  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  to know what  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$  has to learn:

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \quad \xrightarrow{\text{Bayes Rule}} \\ &\propto \exp \left( -\frac{1}{2} \left( \frac{(\mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_0)^2}{1 - \bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0)^2}{1 - \bar{\alpha}_t} \right) \right) \\ &= \exp \left( -\frac{1}{2} \left( \frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t} \mathbf{x}_t \mathbf{x}_{t-1} + \alpha_t \mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_0 \mathbf{x}_{t-1} + \bar{\alpha}_{t-1} \mathbf{x}_0^2}{1 - \bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0)^2}{1 - \bar{\alpha}_t} \right) \right) \\ &= \exp \left( -\frac{1}{2} \left( \left( \frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) \mathbf{x}_{t-1}^2 - \left( \frac{2\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) \mathbf{x}_{t-1} + C(\mathbf{x}_t, \mathbf{x}_0) \right) \right) \end{aligned}$$



Write mean out...

# DDPMs: reverse (simplified derivation)

2. Need to write an expression for the mean of  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  to know what  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$  has to learn:

$$\begin{aligned}\tilde{\beta}_t &= 1/\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) = 1/\left(\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t(1-\bar{\alpha}_{t-1})}\right) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \\ \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) &= \left(\frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}} \mathbf{x}_0\right) / \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) \\ &= \left(\frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}} \mathbf{x}_0\right) \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \\ &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \mathbf{x}_0 \quad \leftarrow \text{Using } \mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + (1-\bar{\alpha}_t)\epsilon\end{aligned}$$

$$= \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t \right)$$

# DDPMs: reverse (simplified derivation)

2. Need to write an expression for the mean of  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  to know what  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$  has to learn:

$$\begin{aligned}\tilde{\beta}_t &= 1/\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) = 1/\left(\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t(1-\bar{\alpha}_{t-1})}\right) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \\ \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) &= \left(\frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}} \mathbf{x}_0\right) / \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) \\ &= \left(\frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}} \mathbf{x}_0\right) \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \\ &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \mathbf{x}_0 \quad \leftarrow \text{Using } \mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + (1-\bar{\alpha}_t)\epsilon \\ &= \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t \right) \quad \leftarrow \text{This is why we cannot directly reverse the process directly (outside of training we do not know } \epsilon_t\text{)}\end{aligned}$$

# DDPMs: reverse (simplified derivation)

3. Plug the derived mean of  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  into the loss

$$\begin{aligned} L_t &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[ \frac{1}{2\|\Sigma_\theta(\mathbf{x}_t, t)\|_2^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[ \frac{1}{2\|\Sigma_\theta\|_2^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_t \right) - \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) \right\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[ \frac{(1 - \alpha_t)^2}{2\alpha_t(1 - \bar{\alpha}_t)\|\Sigma_\theta\|_2^2} \|\epsilon_t - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[ \frac{(1 - \alpha_t)^2}{2\alpha_t(1 - \bar{\alpha}_t)\|\Sigma_\theta\|_2^2} \|\epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon_t, t)\|^2 \right] \end{aligned}$$

Empirically found in DDPM paper

# DDPMs: algorithm

---

## Algorithm 1 Training

```
1: repeat
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:   Take gradient descent step on
       $\nabla_{\theta} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t)\|^2$ 
6: until converged
```

## Algorithm 2 Sampling

```
1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$ 
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 
```

In practice done in batches, each sample has different  $t$  (to learn different stages of denoising at same time)

# DDPMs: algorithm

$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)$  : mean of  $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$

---

## Algorithm 1 Training

---

```
1: repeat
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:   Take gradient descent step on
       $\nabla_{\theta} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t)\|^2$ 
6: until converged
```

---

---

## Algorithm 2 Sampling

---

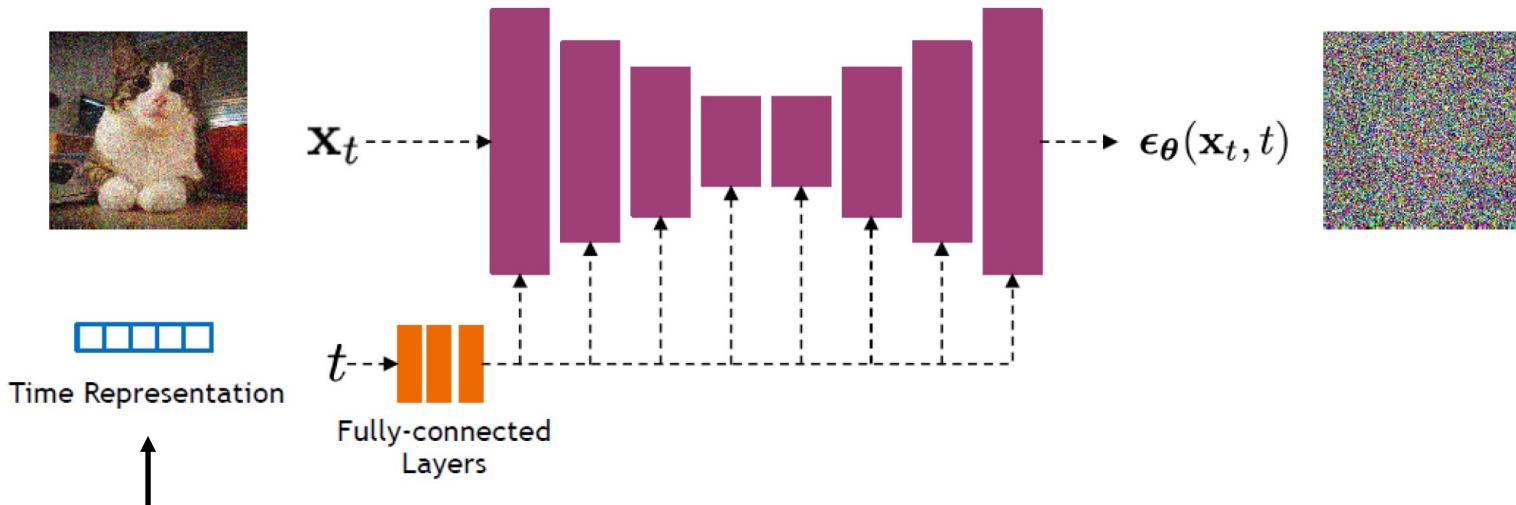
```
1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$ 
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 
```

---

In practice done in batches, each sample has different  $t$  (to learn different stages of denoising at same time)

# DDPMs: network architecture

Whilst this can be anything, a **UNet-like CNN architecture** is usually employed



*Sinusoidal positional embeddings or random Fourier features (added to channels)*

## DDPMs: more details

---

- **Covariance learning:** instead of being fixed, it can be learned to improve the quality of the model (actually a vector  $\mathbf{v}$  mixing  $\beta_t$  and  $\tilde{\beta}_t$  is learned in Nichol & Dhariwal, 2021))

# DDPMs: more details

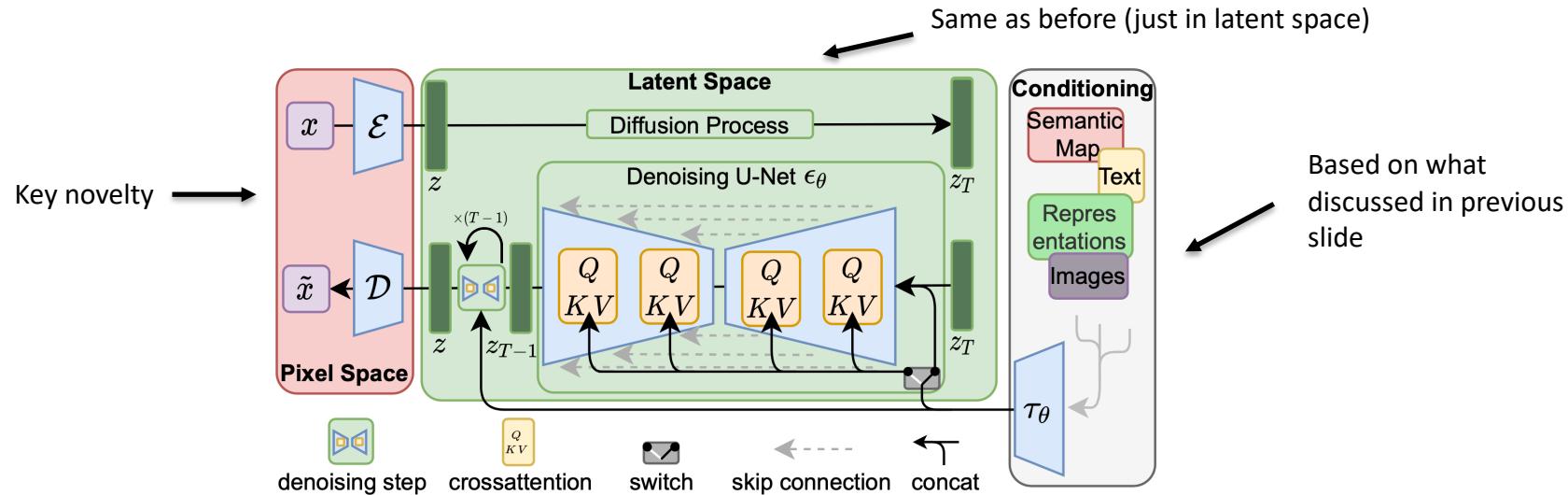
---

- **Covariance learning:** instead of being fixed, it can be learned to improve the quality of the model (actually a vector  $\mathbf{v}$  mixing  $\beta_t$  and  $\tilde{\beta}_t$  is learned in Nichol & Dhariwal, 2021))
- **Conditioning:** can add class (if training with data with labels – eg ImageNet) or text prompt (or any descriptive representation, eg another image)
  - **Classifier guided diffusion:** train classifier on noisy images and use its gradient to guide diffusion reverse process towards class of interest.
  - **Classifier-free guidance:** learn both an unconditional and conditional diffusion process with one network and choose which one to use at generation time

# DDPMs: more details

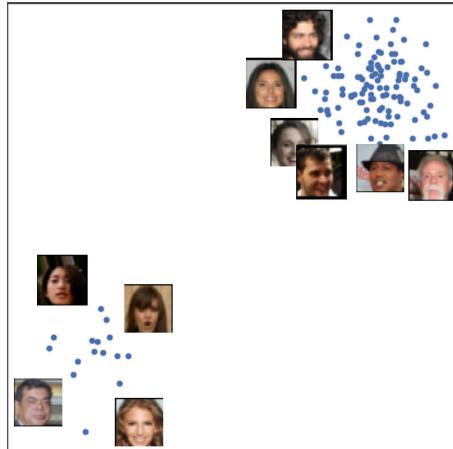
## - Speed-up:

- to sample a subset of steps from T to 0 in generation, one can learn a direct noise map from any t to 0 - DDIMs (Denoising Diffusion Implicit Models)
- can learn the diffusion process in a latent space – LDM (Latent Diffusion Models)



# NCSNs: an alternative approach

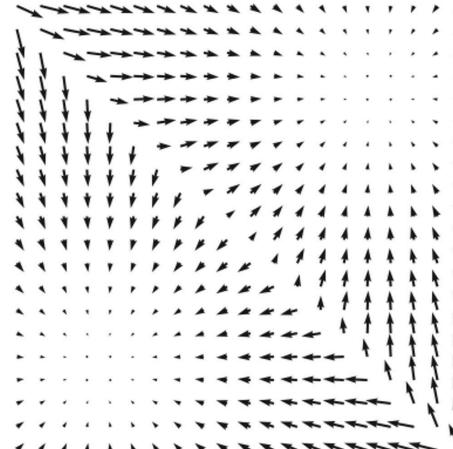
An alternative approach which relies on the **score function** and **Langevin dynamics**.



Data samples

$$\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \stackrel{\text{i.i.d.}}{\sim} p(\mathbf{x})$$

score  
matching



Scores

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$$

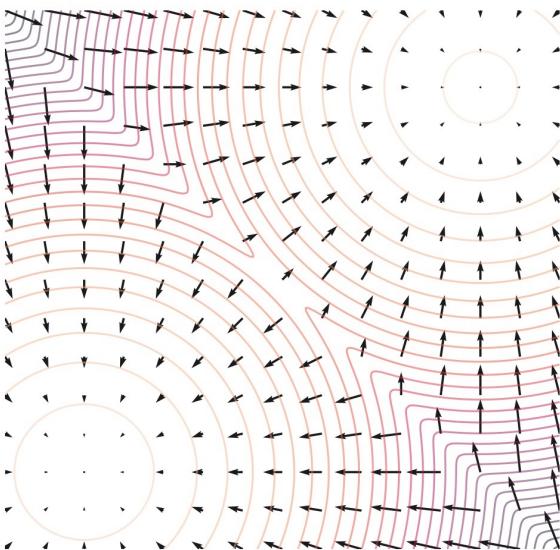
Langevin  
dynamics



New samples

# NCSNs: an alternative approach

**Score function** corresponds to the gradient of log-likelihood



$$\nabla_{\mathbf{x}} \log p(\mathbf{x})$$

# NCSNs: an alternative approach

Ideally we would like to model  $p(\mathbf{x})$  directly (by maximizing its log-likelihood):

$$\operatorname{argmax}_{\theta} \log p(\mathbf{x})$$

We can write it in an exponential form and learn the parameters:

$$p_{\theta}(\mathbf{x}) = \frac{e^{-f_{\theta}(\mathbf{x})}}{Z_{\theta}} \quad \begin{matrix} \text{Normalization factor such as} \\ \int p_{\theta}(\mathbf{x}) d\mathbf{x} = 1 \end{matrix}$$

# NCSNs: an alternative approach

Ideally we would like to model  $p(\mathbf{x})$  directly (by maximizing its log-likelihood):

$$\operatorname{argmax}_{\theta} \log p(\mathbf{x})$$

We can write it in an exponential form and learn the parameters:

$$p_{\theta}(\mathbf{x}) = \frac{e^{-f_{\theta}(\mathbf{x})}}{Z_{\theta}}$$

Normalization factor such as  
 $\int p_{\theta}(\mathbf{x})d\mathbf{x} = 1$

Intractable (must be evaluated in learning process)

# NCSNs: an alternative approach

By learning a score based model, the normalizing factor is side-stepped

$$s_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} \log f_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \cancel{\log Z_\theta} = -\nabla_{\mathbf{x}} \log f_\theta(\mathbf{x})$$

The learning process minimizes the **Fisher divergence** between the model and the data distributions:

$$E_p[\|s_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{x})\|_2^2] \longleftarrow \text{Score matching methods!}$$

# NCSNs: an alternative approach

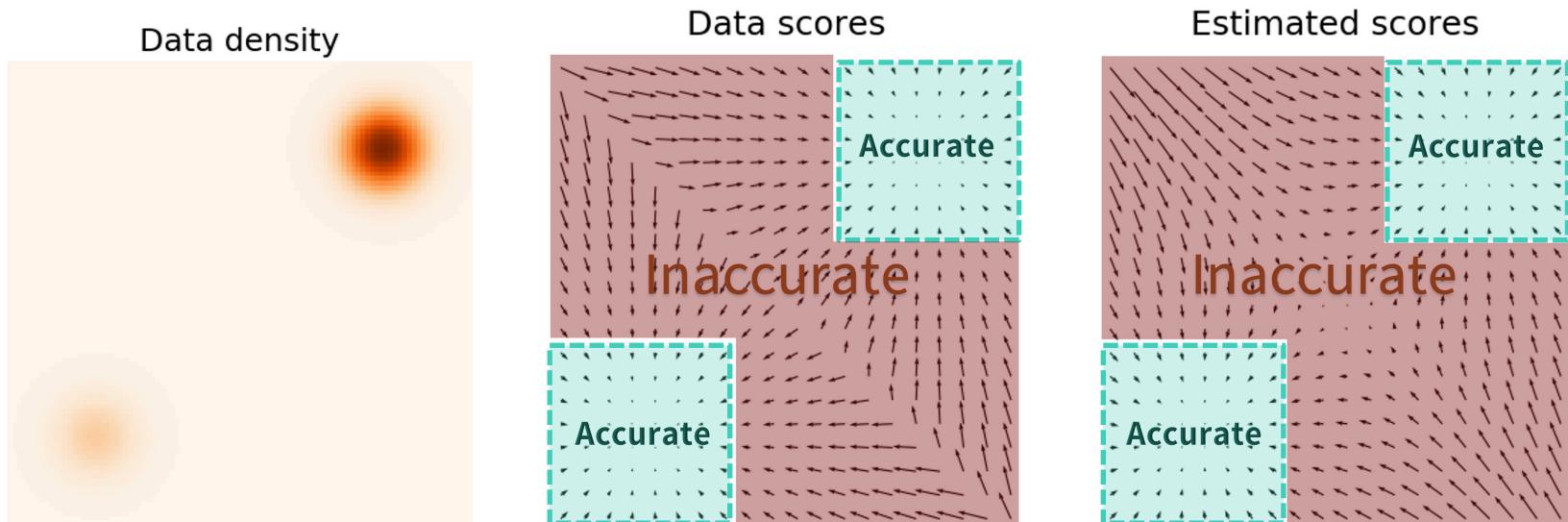
The generation process becomes simple – i.e. gradient ascent on  $p(\mathbf{x})$ :

$$\begin{aligned}\mathbf{x}_{i+1} &= \mathbf{x}_i + \underbrace{\epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x})}_{\approx s_{\theta}(\mathbf{x}_{i+1})} + \sqrt{2\epsilon} \mathbf{z}_i \\ &\quad \text{Indirect/learned access to the score function}\end{aligned}$$

**Note:** this is not pure gradient ascent (would not sample the distribution as it is deterministic) → Langevin dynamics = stochastic version of gradient ascent

# NCSNs: where is the connection with DDPMs?

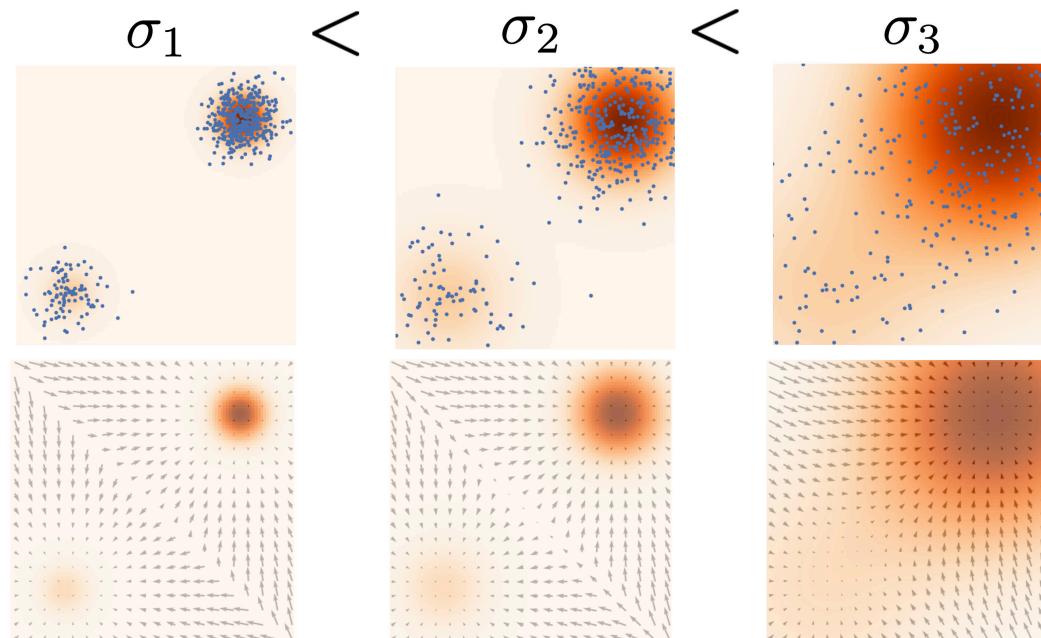
Turns out score functions cannot be stably learned in areas of low probs.



Source: <https://yang-song.net/blog/2021/score/>

# NCSNs: where is the connection with DDPMs?

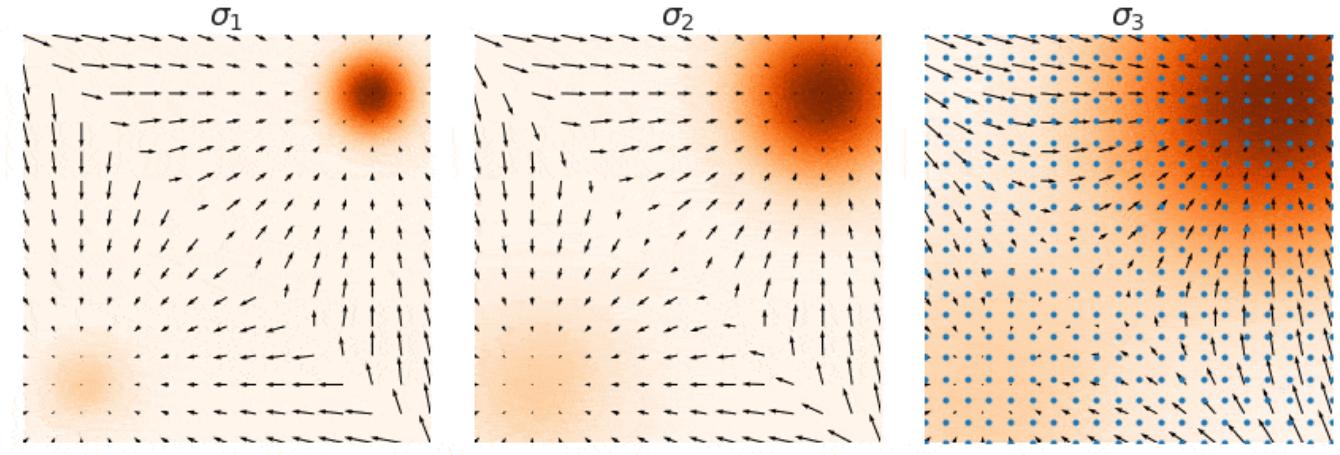
Solution: **perturb training data with different levels of noise** (=convolve/smooth underlying pdf)



Accurate reconstruction at all scales

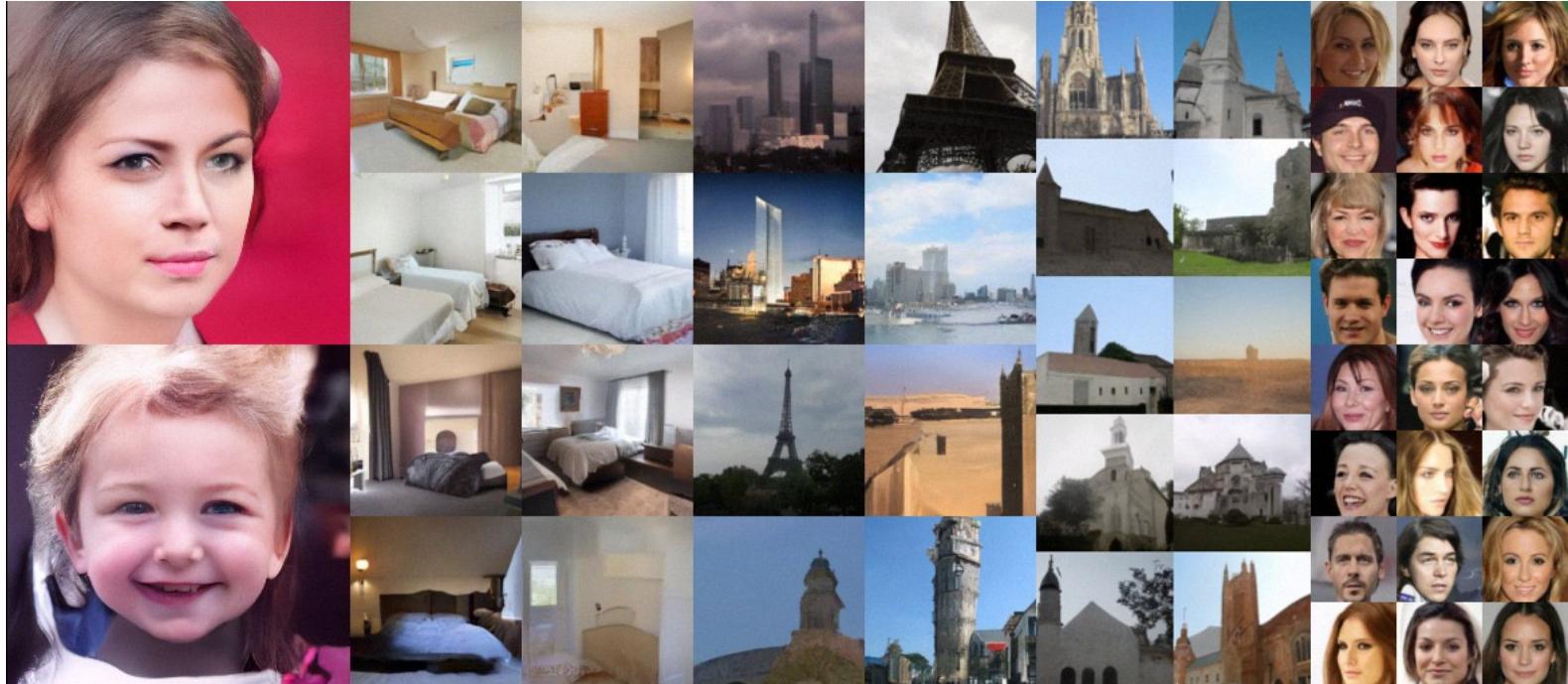
# NCSNs: where is the connection with DDPMs?

Solution: combine a **sequence of Langevin chains** with gradually decreasing noise scales.



Source: <https://yang-song.net/blog/2021/score/>

# NCSNs: an alternative approach



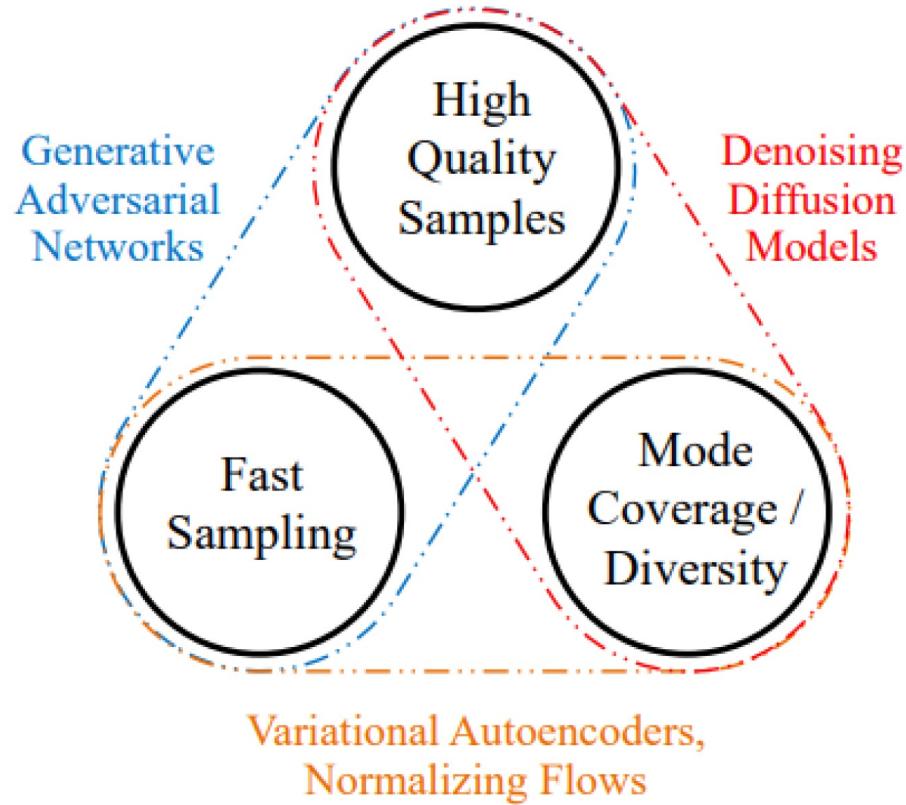
From NCSNv2 model

# Trying out Diffusion models for free

A nice demo of stable diffusion (without code / fee):

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

# Wrapping up the week

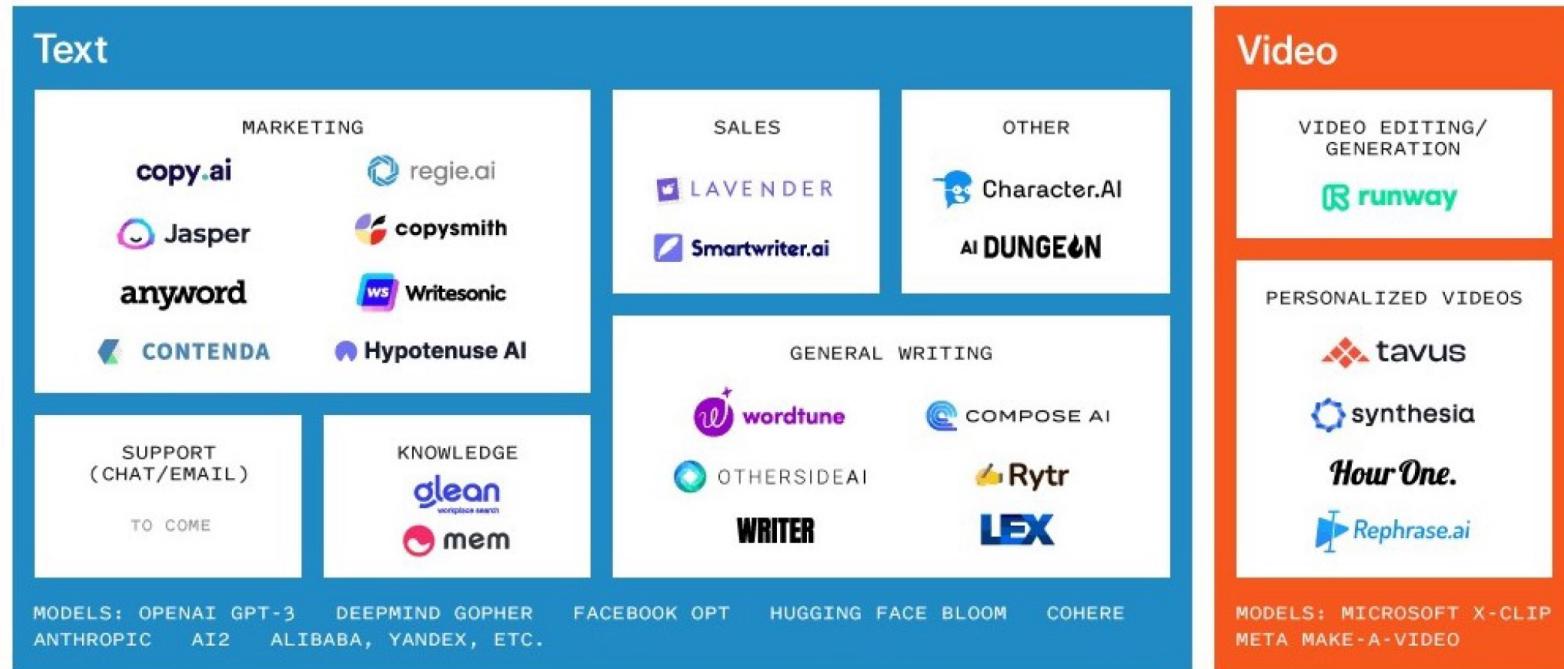


Generative learning trilemma

*From: Tackling the Generative Learning Trilemma with Denoising Diffusion GANs*

# Wrapping up the week

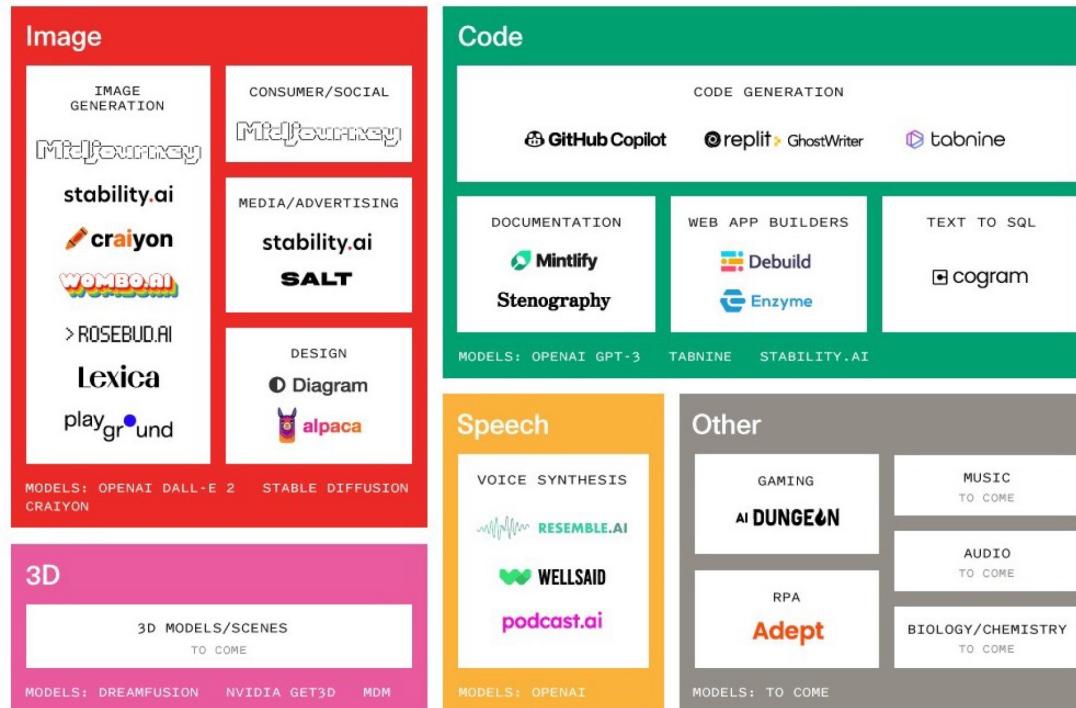
## The Generative AI landscape



From Sonya Huang (@sonyatweetybird)

# Wrapping up the week

## The Generative AI landscape



From Sonya Huang (@sonyatweetybird)