Diffusion Models

Tianyi’s learning & review

DDPM official Tutorial with

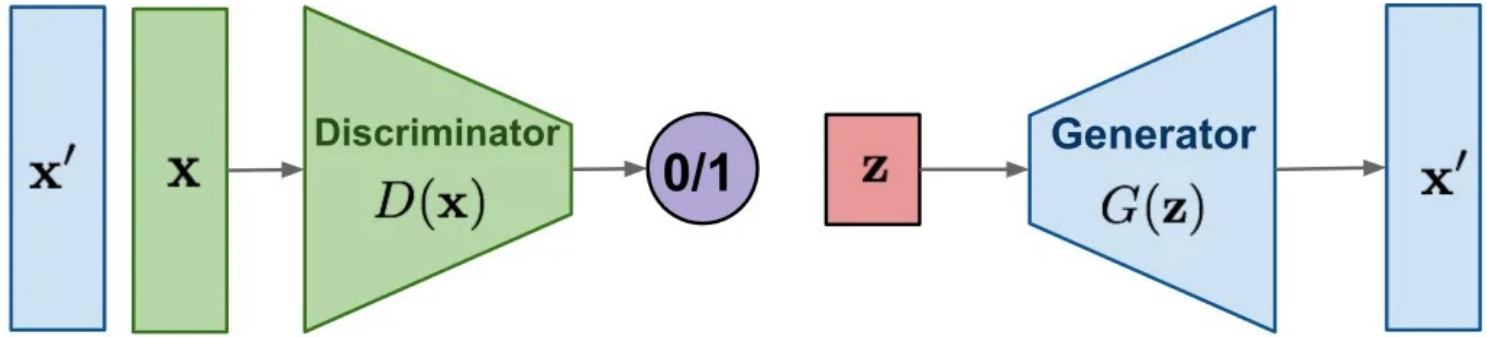
Math
https://nn.labml.ai/diffusion/ddpm/index.html

Coding
https://github.com/labmlai/annotated_deep_learning_paper_implementations/tree/master/labml_nn/diffusion/ddpm

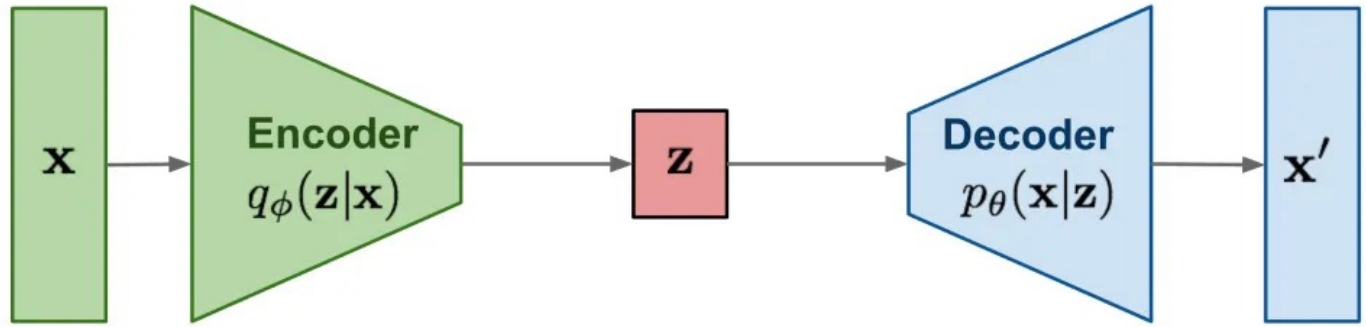
Many slides here are taken from:

https://web.cs.ucla.edu/~patricia.xiao/files/Win2023_Math_Reading_Group_Stable_Diffusion.pdf

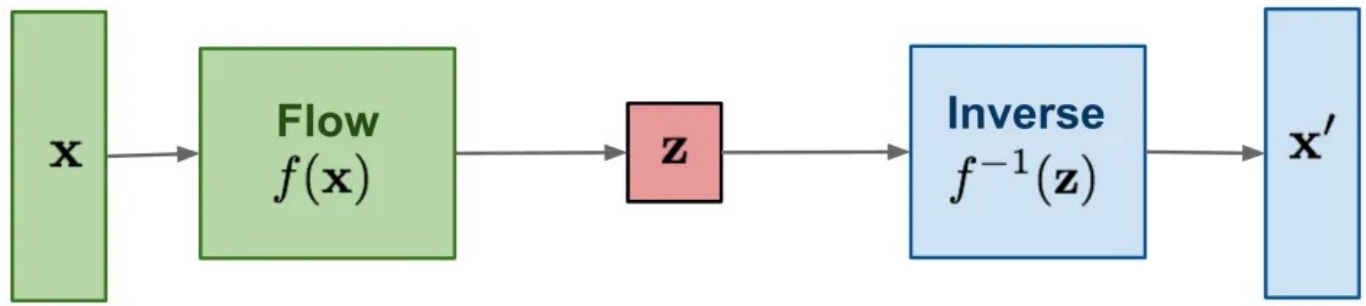
GAN: Adversarial training



VAE: maximize variational lower bound



Flow-based models: Invertible transform of distributions



Diffusion models: Gradually add Gaussian noise and then reverse

Table: Advantage & Disadvantage of the Generative Models

Model	likelihood-based?	Good At	Not Good At
GAN	NO	efficient sampling; perceptual quality	optimize; capture data distribution
VAE/ Flow-based	YES	capture data distribution; optimize	perceptual quality
DMs	YES	capture data distribution; perceptual quality	computation cost

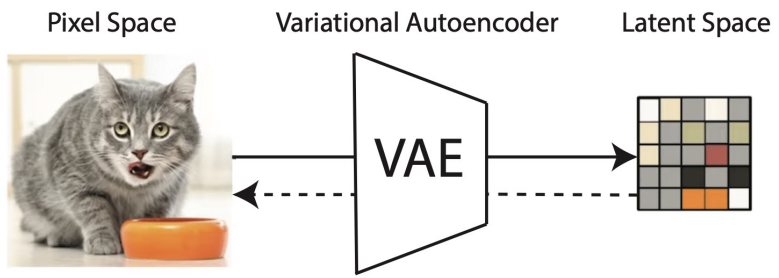


Figure: The role autoencoder plays in the model. Note: we actually use **separated** encoder $\mathcal{E}$ and decoder $\mathcal{D}$. But this detail is not illustrated in this figure precisely for simplicity concern.

Important papers

ICML2015 (Original Diffusion paper) Deep Unsupervised Learning using Nonequilibrium Thermodynamics

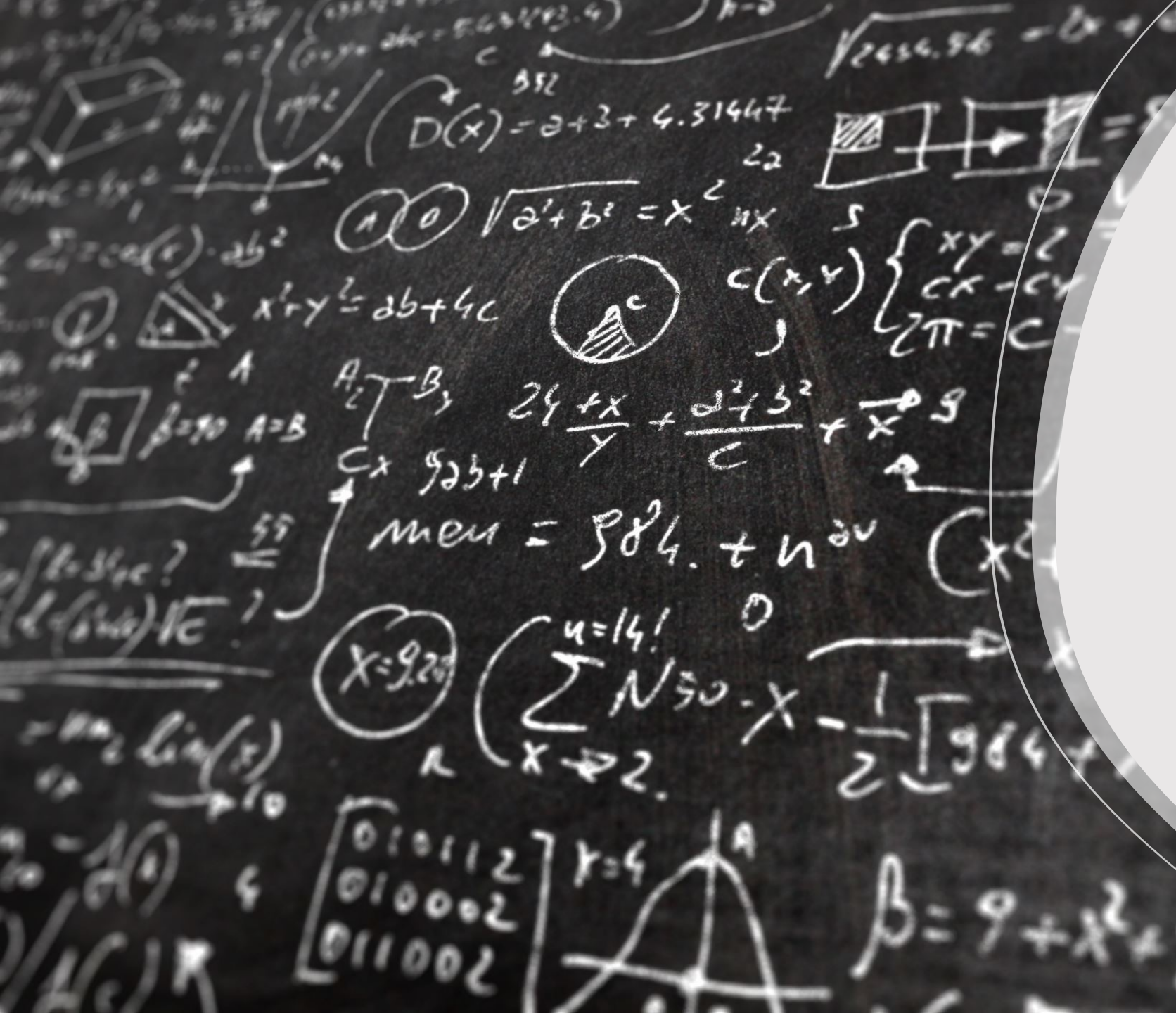
2006 DDPM （Most Important DM paper)

2112 High-Resolution Image Synthesis with Latent Diffusion Models
(CVPR 2022: Latent Diffusion Models, LDMs)

2105 Diffusion Models Beat GANs on Image Synthesis (guided diffusion)

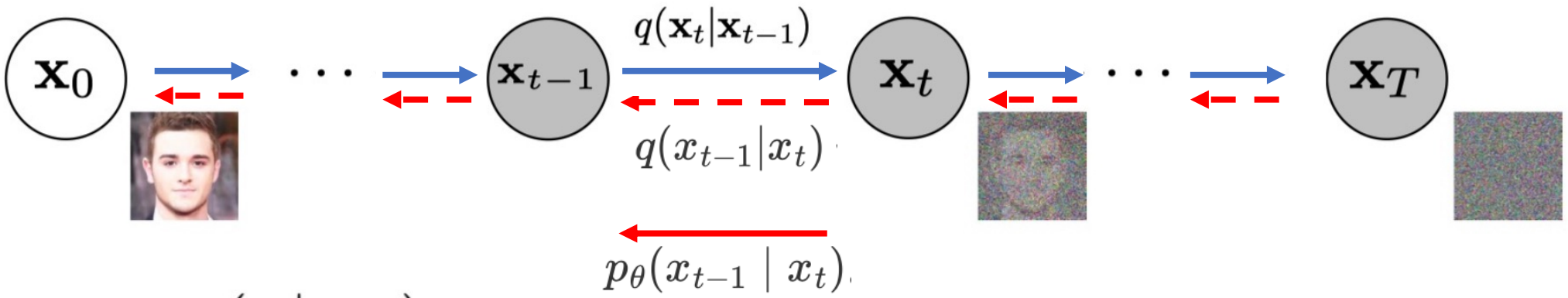
2301 Understanding the diffusion models by conditional expectations

2302 Adding Conditional Control to Text-to-Image Diffusion Models



Process of Diffuse and Reverse

Key concept / difinations



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

add noise (dim dim dim + mask mask mask)

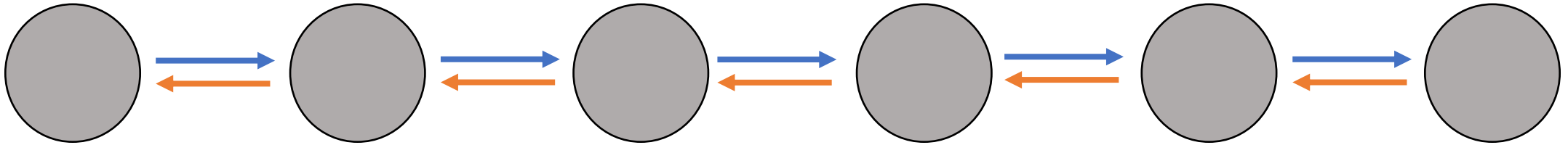
Reverse-Diffuse : $q(x_{t-1}|x_t)$

The reverse process of diffuse. (can't be calculated but can be estimated)

Denoise : $p_\theta(x_{t-1} \mid x_t)$

The estimated process projecting denoised samples.

Markov Chain and backward



MC:

Forward: each state is only based on previous state and the transmition operation

For gradient backward

Backward: If each step's transmition function is differetiable, then we can have gradient.

Reparametermization trick

$$z \sim \mathcal{N}(z; \mu_\theta, \sigma_\theta^2 \mathbf{I})$$

$$z = \mu_\theta + \sigma_\theta \odot \epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

Gaussian distribution understandings

Add two normal distribution

$$\mathcal{N}(0, \sigma_1^2 \mathbf{I}) + \mathcal{N}(0, \sigma_2^2 \mathbf{I}) \sim \mathcal{N}(0, (\sigma_1^2 + \sigma_2^2)\mathbf{I})$$

Change the miu and theta is move and rescale

$$\sqrt{a_t(1-\alpha_{t-1})}z_2 \sim \mathcal{N}(0, a_t(1-\alpha_{t-1})\mathbf{I})$$
$$\sqrt{1-\alpha_t}z_1 \sim \mathcal{N}(0, (1-\alpha_t)\mathbf{I})$$
$$\sqrt{a_t(1-\alpha_{t-1})}z_2 + \sqrt{1-\alpha_t}z_1 \sim \mathcal{N}(0, [\alpha_t(1-\alpha_{t-1}) + (1-\alpha_t)]\mathbf{I})$$
$$= \mathcal{N}(0, (1-\alpha_t\alpha_{t-1})\mathbf{I}).$$

Prove of the relationship on X0 and beta, and XT ~N(0,1)

Let: $\alpha_t = 1 - \beta_t \quad \bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$

$$
\begin{aligned}
x_t &= \sqrt{a_t} x_{t-1} + \sqrt{1-\alpha_t} z_1 \quad \text{where} \quad z_1, z_2, \ldots \sim \mathcal{N}(0, \mathbf{I}); \\
&= \sqrt{a_t}\left(\sqrt{a_{t-1}} x_{t-2} + \sqrt{1-\alpha_{t-1}} z_2\right) + \sqrt{1-\alpha_t} z_1 \\
&= \sqrt{a_t a_{t-1}} x_{t-2} + \left(\sqrt{a_t(1-\alpha_{t-1})} z_2 + \sqrt{1-\alpha_t} z_1\right) \\
&= \sqrt{a_t a_{t-1}} x_{t-2} + \sqrt{1-\alpha_t \alpha_{t-1}}\, \bar{z}_2 \quad \text{where} \quad \bar{z}_2 \sim \mathcal{N}(0, \mathbf{I}); \\
&= \ldots \\
&= \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\, \bar{z}_t .
\end{aligned}
$$

Put Xt-1 into xt

$T \to \infty$, $x_T \sim \mathcal{N}(0, \mathbf{I})$

$$x_0 = \frac{1}{\sqrt{\bar{a}_t}}\left(x_t - \sqrt{1-\bar{a}_t}\, \bar{z}_t\right)$$

Prove that both $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $q(x_{t-1}|x_t)$ are normal distribution when beta is small

Define $q(x_{t-1}|x_t)$ as a normal distribution

$$p_\theta(X_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t);$$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)).$$

虽然我们无法得到逆转后的分布 $q(x_{t-1}|x_t)$，但是如果知道 x_0 ，是可以通过贝叶斯公式得到 $q(x_{t-1}|x_t, x_0)$ 为：

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t\mathbf{I}) \tag{6}$$

The bayessian rules and piorer estimation on $q(x_{t-1}|x_t)$

$$
\begin{aligned}
q(x_{t-1}|x_t, x_0) &= q(x_t|x_{t-1}, x_0)\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{(x_t-\sqrt{\alpha_t}x_{t-1})^2}{\beta_t}+\frac{(x_{t-1}-\sqrt{\bar{\alpha}_{t-1}}x_0)^2}{1-\bar{a}_{t-1}}-\frac{(x_t-\sqrt{\bar{\alpha}_t}x_0)^2}{1-\bar{a}_t}\right)\right) \\
&= \exp\left(-\frac{1}{2}\Big(\underbrace{\left(\frac{\alpha_t}{\beta_t}+\frac{1}{1-\bar{\alpha}_{t-1}}\right)x_{t-1}^2}_{x_{t-1}\text{方差}}-\underbrace{\left(\frac{2\sqrt{\alpha_t}}{\beta_t}x_t+\frac{2\sqrt{\bar{a}_{t-1}}}{1-\bar{\alpha}_{t-1}}x_0\right)x_{t-1}}_{x_{t-1}\text{均值}}+\underbrace{C(x_t,x_0)}_{\text{与}x_{t-1}\text{无关}}\Big)\right)
\end{aligned}
$$

Exp of gaussian

$$
\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)=\exp\left(-\frac{1}{2}\left(\frac{1}{\sigma^2}x^2-\frac{2\mu}{\sigma^2}x+\frac{\mu^2}{\sigma^2}\right)\right)
$$

Prove that $q(x_{t-1}|x_t, x_0)$ can be numerically estimated

$$\frac{1}{\sigma^2} = \frac{1}{\tilde{\beta}_t} = (\frac{\alpha_t}{\beta_t} + \frac{1}{1-\overline{\alpha}_{t-1}}); \quad \tilde{\beta}_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t} \cdot \beta_t$$

$$\frac{2\mu}{\sigma^2} = \frac{2\tilde{\mu}_t(x_t, x_0)}{\tilde{\beta}_t} = (\frac{2\sqrt{\alpha_t}}{\beta_t} x_t + \frac{2\sqrt{\overline{a}_{t-1}}}{1-\overline{\alpha}_{t-1}} x_0);$$

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{a_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t} x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t} x_0.$$

As we already have:

$$x_0 = \frac{1}{\sqrt{\overline{a}_t}}(x_t - \sqrt{1-\overline{a}_t}\overline{z}_t)$$

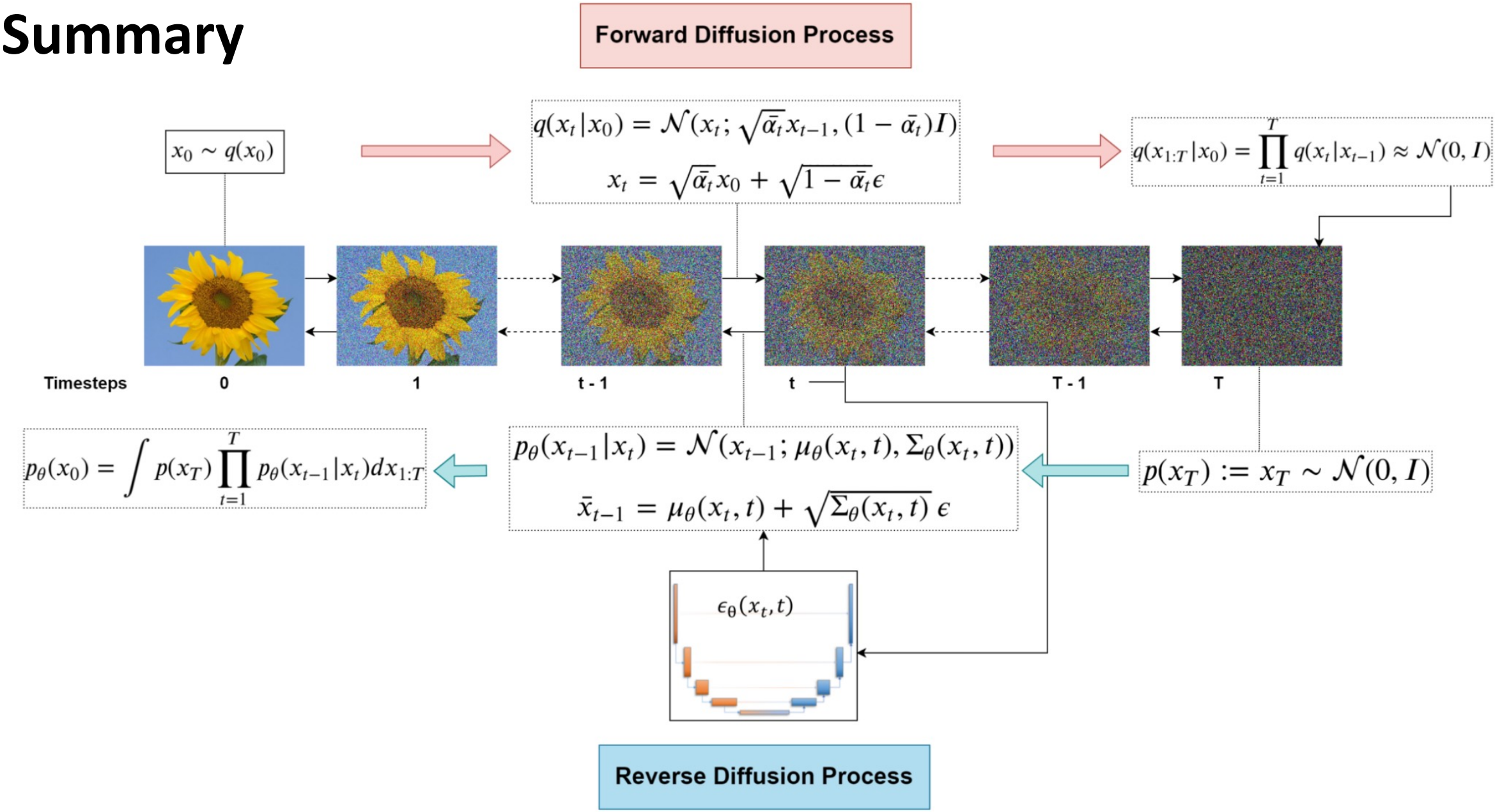
So

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{a_t}}(x_t - \frac{\beta_t}{\sqrt{1-\overline{a}_t}} z_\theta(x_t, t))$$

$$\tilde{\mu}_t = \frac{1}{\sqrt{a_t}}(x_t - \frac{\beta_t}{\sqrt{1-\overline{a}_t}} \overline{z}_t)$$

Which means it can be numerically estimated.

Summary



All equations related to the forward and reverse diffusion processes.

Loss Design

For the process of getting X(t-1) based on X(t):

The Reverse q:

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

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

We need the P-theta (denoising) to estimate reverse q.

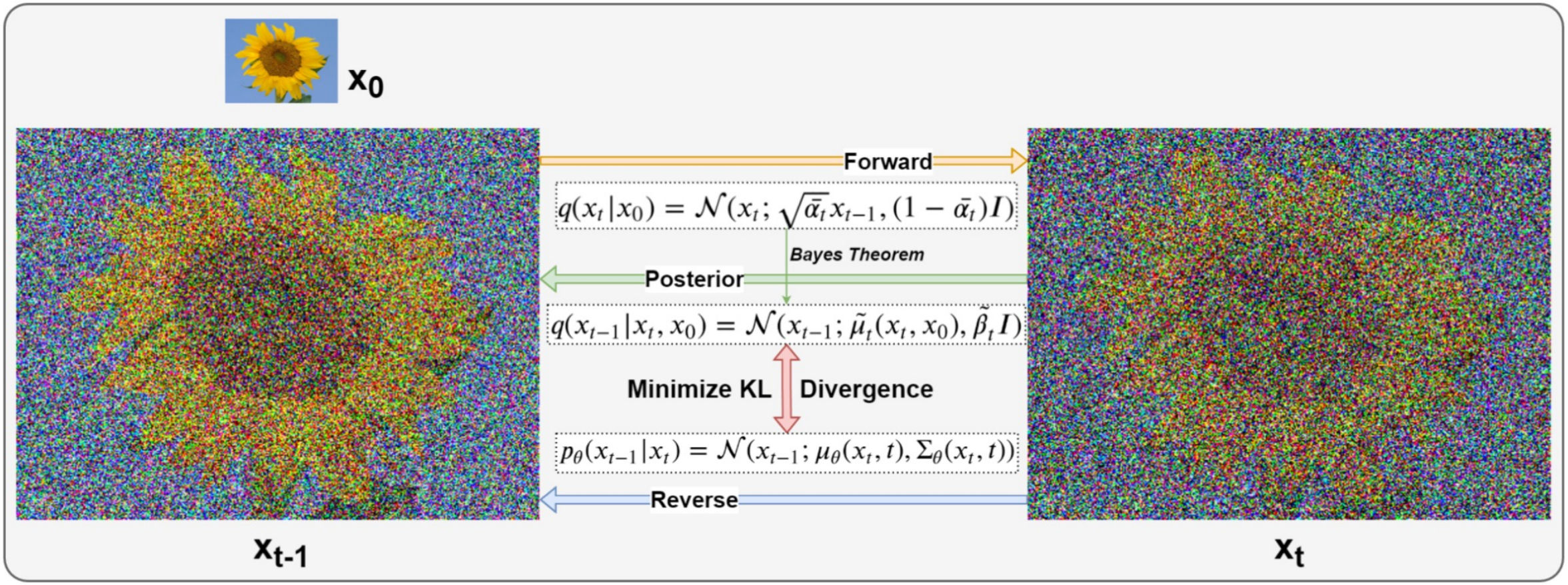
To further simplify the task of the model, ***the authors decided to fix the variance to a constant β_t.***

Now, the model only needs to learn to predict the above equation. And the reverse diffusion kernel gets modified to:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

$$\downarrow$$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma^2 I)$$



To measure the difference of ‘reverse q’ and ‘p theta’:

KL-divergence (P and Q)

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x)\log(\frac{p(x)}{q(x)})dx$$

Background: information theory and entropy

To optimize the P based on theta, MLE(maximal likelihood estimation)

Maximum likelyhood estimation (MLE)

how we know we can calculate $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ ← Variance of p_θ

but how to get the correct θ so the value works?

MLE: $\mathcal{L}$ is a function about θ; when we have the given p denoise

output of x, the value of $\mathcal{L}$ is possibility of obtain x under θ:

$\mathcal{L}(\theta|x) = p(X=x|\theta)$ we want the chance to be the highest (maximumazation)

$\Rightarrow \min -\mathcal{L} \Rightarrow \min -\mathcal{L}(\theta|x) = \min -p(X=x|\theta)$ also as $p_\theta(x)$

How to build this $\mathcal{L}$?

$p_\theta(x_0) := \int p_\theta(x_{0:T})\, dx_{1:T}$ by right this is the mathmatic Target def.

search for the minimal negative log Expectation

$\mathcal{L} = E_q(-\log p_\theta(x_0))$ 最小化负对数似然 $0 \to x_0$ $0 \to T$

With the D(kL) >= 0, we can create this:

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0) \| p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\Big] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\Big] \\
&= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \\
\text{Let } L_{\mathrm{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0)
\end{aligned}
$$

$$
\begin{aligned}
L_{\mathrm{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \\
&= \mathbb{E}_q\Big[\log \frac{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T)\prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \Big(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}\Big) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\Big] \\
&= \mathbb{E}_q[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^T \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]
\end{aligned}
$$

$$\mathcal{L}_{VLB} = L_T + L_{T-1} + \ldots + L_0$$

$$L_T = D_{KL}(q(x_T|x_0)||p_\theta(x_T))$$

X0 is known, cannot be optimized

$$L_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t,t)\|^2\right] + C$$

Next page

$$L_0 = -\log p_\theta(x_0|x_1)$$

cannot understand why its not helpful yet (future exploration)

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

Now, there are **three approaches** we can take here:

1. Directly predict $\mathbf{x_0}$ and find $\bar{\boldsymbol{\mu}}$ using it in the posterior function.
2. Predict the entire $\bar{\boldsymbol{\mu}}$ term.
3. **Predict the noise at each timestep.** This is done by writing the $\mathbf{x_0}$ in $\bar{\boldsymbol{\mu}}$ in terms of $\mathbf{x_t}$ using the reparameterization trick.

By using the third option, and after some simplification, $\bar{\boldsymbol{\mu}}$ can be expressed as:

$$\bar{\mu}(x_t, x_0) = \frac{1}{\sqrt{\bar{x}_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t \right)$$

Similarly, the formulation for $\boldsymbol{\mu_\theta}(x_t, t)$ is set to:

$$\mu_\theta(x_t, x_0) = \frac{1}{\sqrt{\bar{x}_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right)$$

At training and inference time, we know the **β**'s, **α**'s, and $\mathbf{x_t}$. So our **model only needs to predict the noise at each timestep.** The simplified (after ignoring some weighting terms) loss function used in the *Denoising Diffusion Probabilistic Models* is as follows:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t, \mathbf{x}_0, \boldsymbol{\epsilon}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2 \right]$$

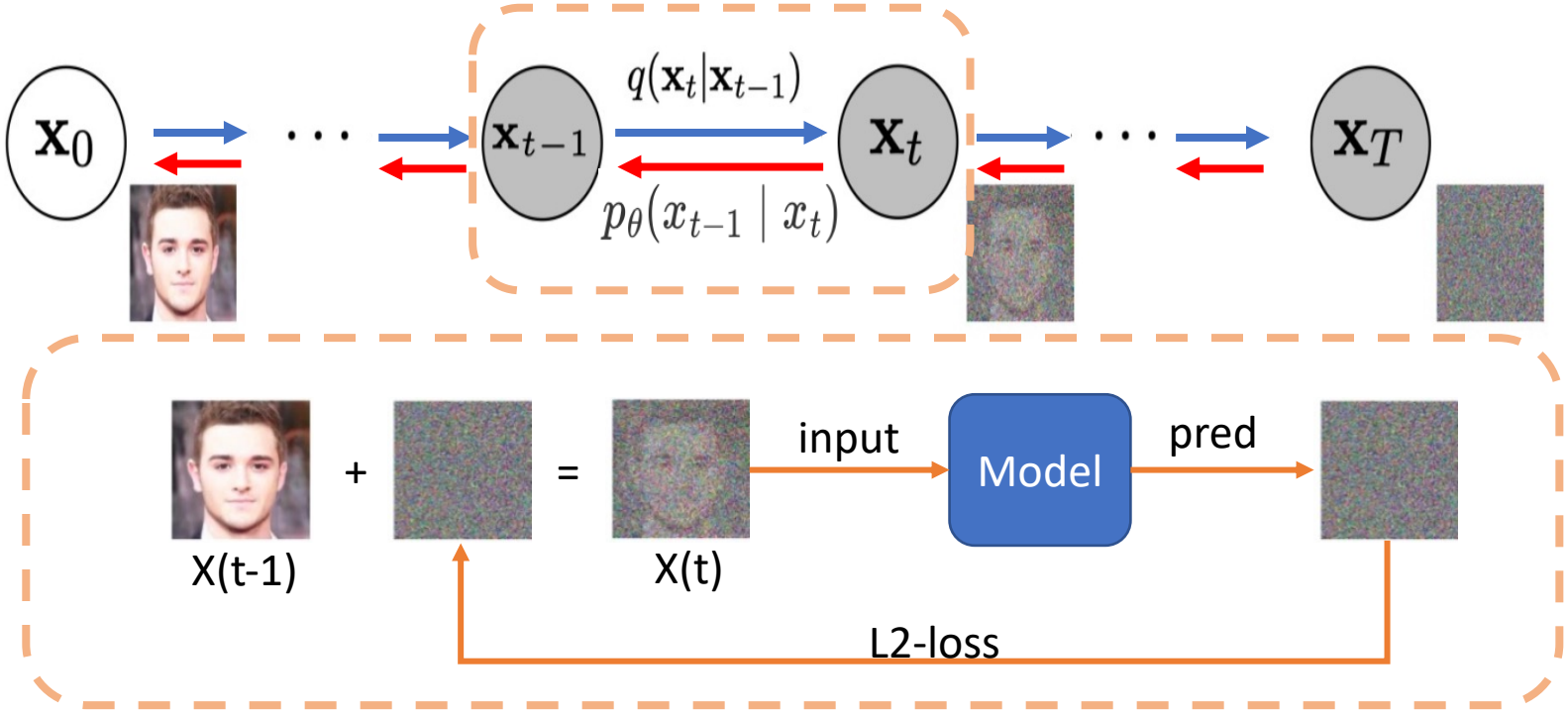
Comparing just the noise.

Which is basically:

$$L_{\text{simple}} = E_{t,x_0,\epsilon}\left[||\epsilon - \epsilon_\theta(x_t, t)||^2\right]$$

This is the final loss function we use to train DDPMs, which is just a "Mean Squared Error" between the noise added in the forward process and the noise predicted by the model. This is the most impactful contribution of the paper Denoising Diffusion Probabilistic Models.

It's awesome because, beginning from those scary-looking ELBO terms, we ended up with the simplest loss function in the entire machine learning domain.



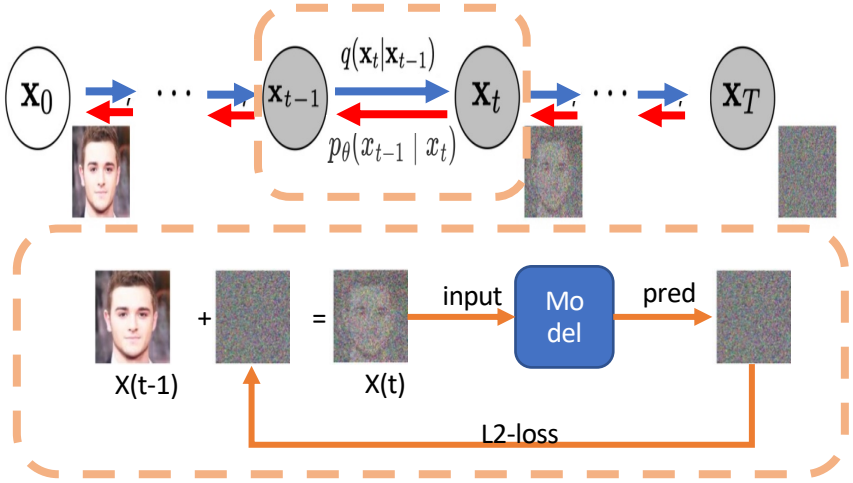
Belong to the class of likelihood-based probabilistic models.

Learn a data distribution $p(x)$ by gradually denoising a normally-distributed variable x_T, i.e. learning the **reverse process** of a fixed Markov Chain of length T.

$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right],$$

where t is uniformly sampled from $\{1, 2, \ldots T\}$, and the models can be interpreted as an equally-weighted sequence of denoising autoencoders $\epsilon_\theta(x_t, t)$ (ϵ_θ is typically implemented as **U-Net**), trained to predict the denoised version of x_t, namely x. *The above version works well in* ***image*** *settings.*

Diffusion models are capable of modeling conditional distributions $p(x|y)$ via using a a conditional denoising autoencoder $\epsilon_\theta(x_t, t, y)$.

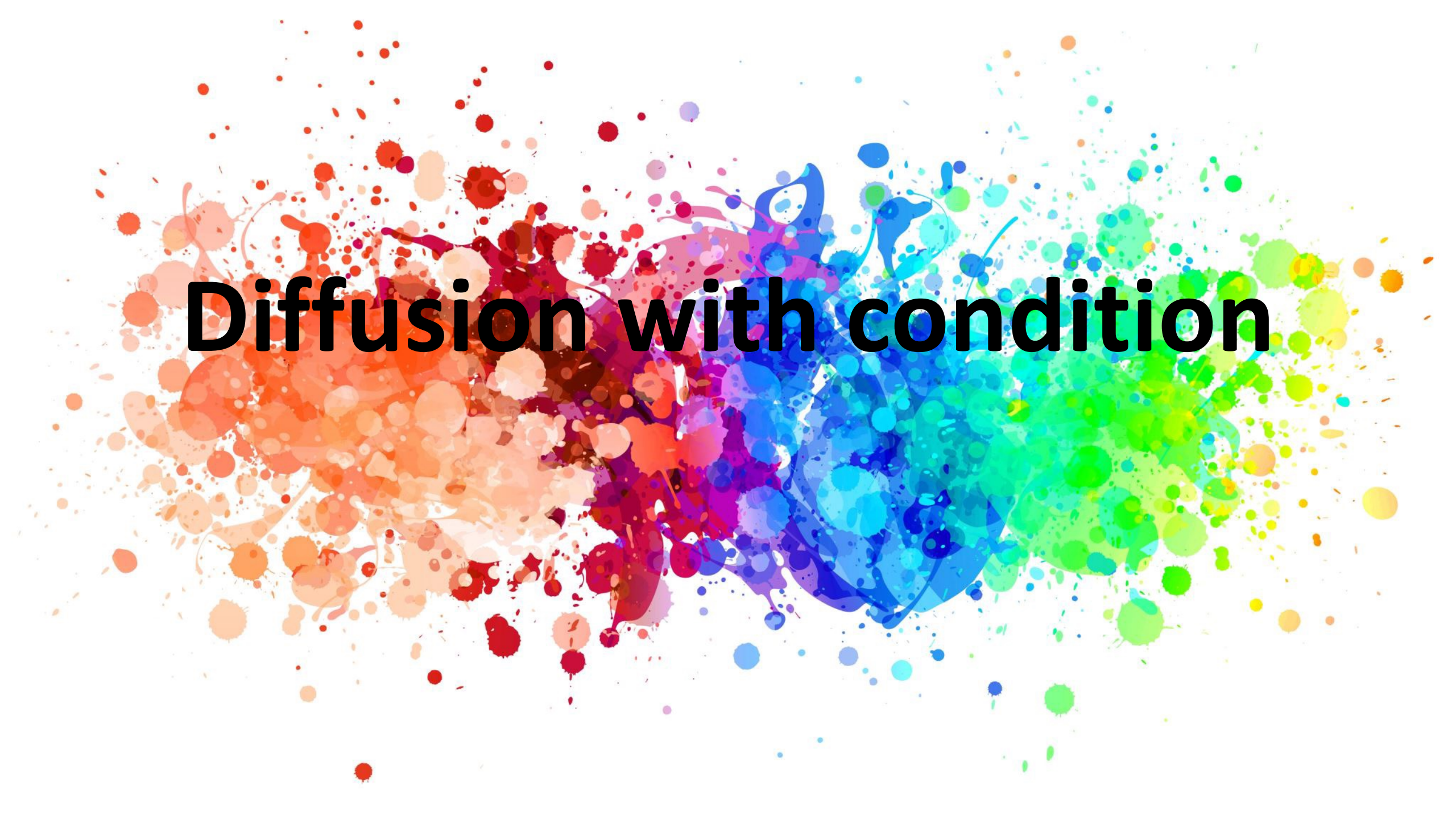


$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right]$$

The most powerful DMs are often computationally demanding.

- Costly Training: UNet has typically $\approx 800M$ parameters; the model takes hundreds of GPU days to train, prone to spend excessive amounts of capacity on modeling imperceptible details; [2]
- Costly Evaluation: cost a lot of time and memory, must run the same architecture sequentially for many of steps.
- e.g. Diffusion Models Beat GANs on Image Synthesis (NeurIPS'21) takes 150 - 1000 *V100 days* to train, 25 - 1000 steps to evaluate.


Training loop
and Coding


Diffusion with condition

Quick Recap: Mode-Covering Behavior

Minimize difference between $Q(x)$ and $P(x)$ at data distribution $P(x) > 0$:

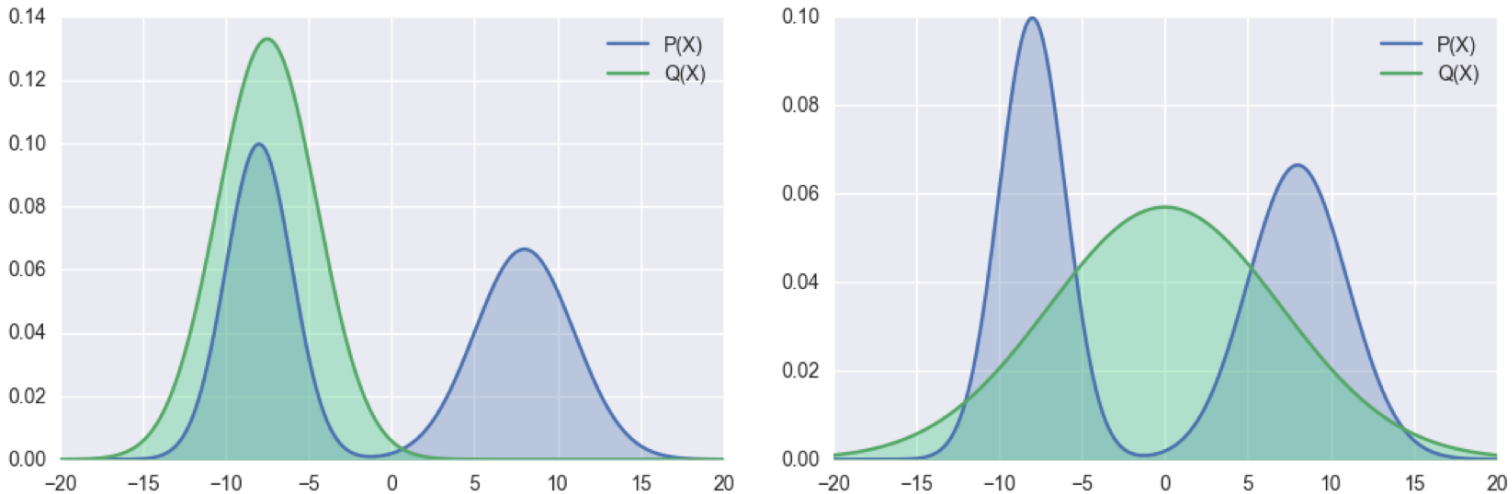


Figure: Bad v.s. Good mode-covering. On the left (bad) example, the right hand side **mode** is not covered by $Q(x)$, but it is the case that $P(x) > 0$. (from agustinus.kristia.de)

This behavior makes the DMs costly.

Latent Diffusion Models (LDMs): Motivation

Previous solutions of the cost:

- Weighted importance of steps. (still expensive)

$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\lambda(t)\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right]$$

- Having an extra model to learn upsampling & sharpening of images. (still working on image space) e.g. GLIDE.

Observation: Most bits of a digital image correspond to imperceptible details, but we still need to train and evaluate on all pixels if we work on pixel spaces.

Latent Diffusion Models (LDMs): Novelty

Highlighted Novelty: Do Diffusion on **Latent** Space, and accept more general types of conditions.

- Operating on latent space (perceptually equivalent space) of powerful pre-trained auto-encoders, instead of directly on pixel space *(compared with standard Diffusion Models)*.
- **Less Costly**: Fast sampling, efficient training, one-step decoding to image space.
- **More Flexibility**: More general conditions. (Besides, operation on latent space makes it easier to add other signals.)

Three Major Components, trained **separately**:

- Autoencoder: Implemented as Variational Autoencoder (VAE); Handling perceptual image compression.
  - Encoder $\mathcal{E}$, Decoder $\mathcal{D}$
  - $z = \mathcal{E}(x)$ where the RGB image $x \in \mathbb{R}^{H \times W \times 3}$ turns into latent representation $z \in \mathbb{R}^{h \times w \times c}$, while $\tilde{x} = \mathcal{D}(z)$ tries to reconstruct x.
  - Some regularization terms applied (e.g. KL-penalty towards a standard normal) to avoid arbitrarily high-variance.
- Denoiser: **Latent Diffusion Models**

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t)\|_2^2\right],$$

  where the neural backbone ϵ_θ of LDM is realized as a time-conditional attention UNet.
- Conditioning Encoder: can be arbitrary encoder that produces a sequence of tokens.

Latent Diffusion Models (LDMs): Conditioning

DMs are capable of modeling conditional distribution $p(z|y)$, by using a a conditional denoiser $\epsilon_\theta(z_t, t, y)$.

LDMs propose to augment the UNet backbone implementing ϵ_θ with the cross-attention mechanism. Q, K, V are projection of $\phi_i(z_t)$, $\tau_\theta(y)$, $\tau_\theta(y)$ respectively, where $\phi_i(z_t)$ is the flattened intermediate representation of the U-Net implementing ϵ_θ.

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\|_2^2\right],$$

τ_θ is a domain specific encoder used to project y, e.g. τ_θ can be transformers when y are text prompts.


Diffusion with condition

Look back at VAE

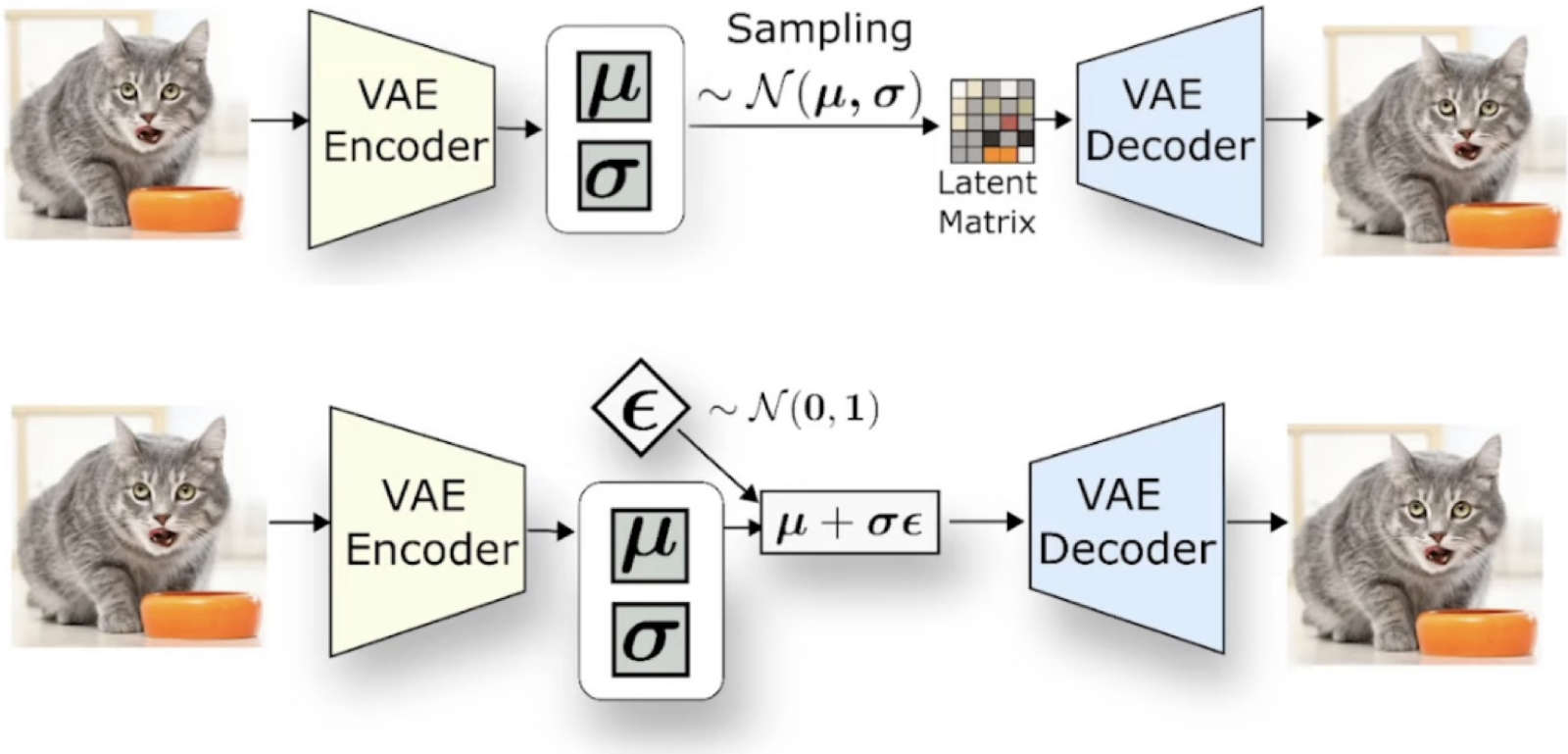


Figure: from Lightning AI video. Re-parameterization trick.

Look back at VAE

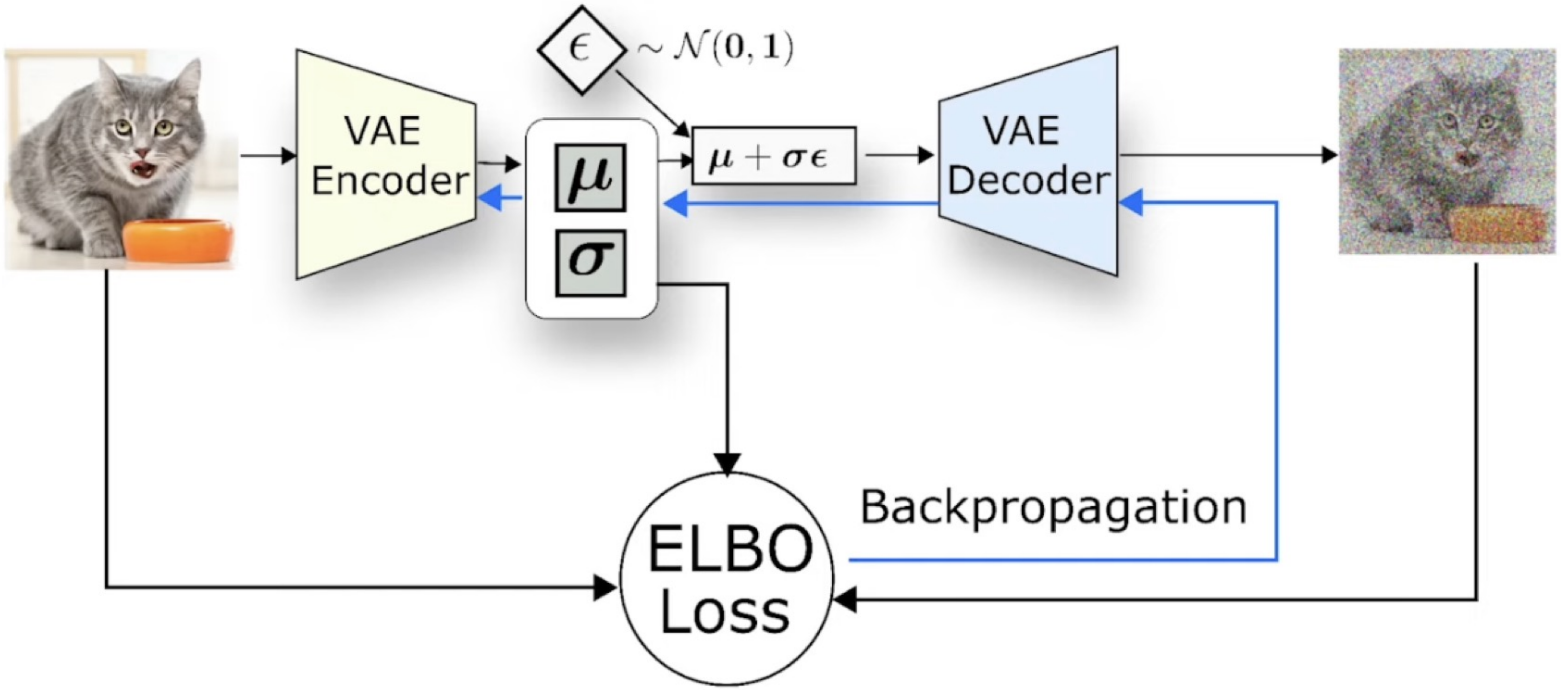


Figure: from Lightning AI video. Evidence Lower Bound Objective (ELBO): $likelihood(\text{input} - \text{output}) - KLD(\mathcal{N}(\mu, \sigma)||\mathcal{N}(0,1))$

Latent Diffusion Models (LDMs): Architecture

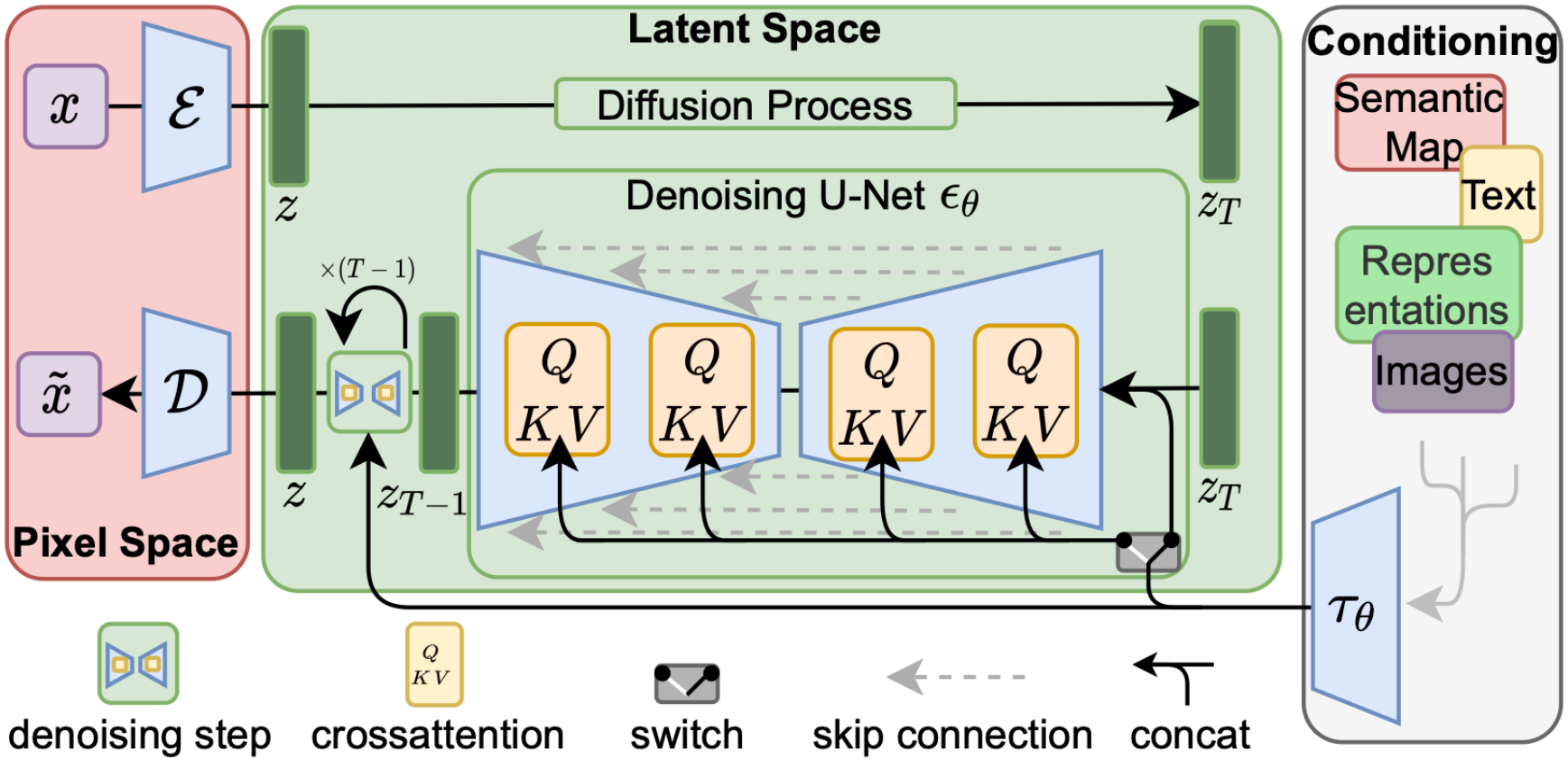
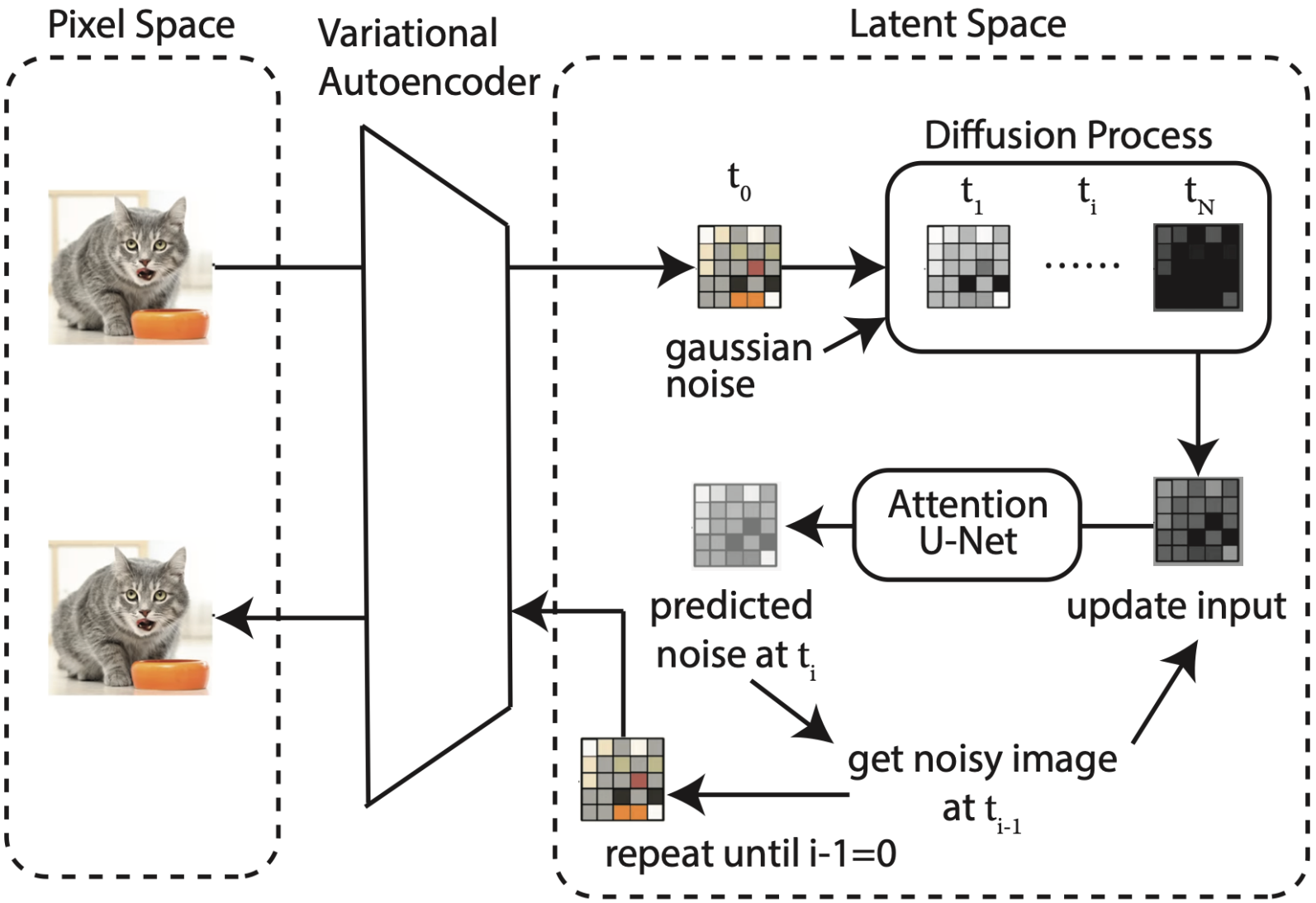
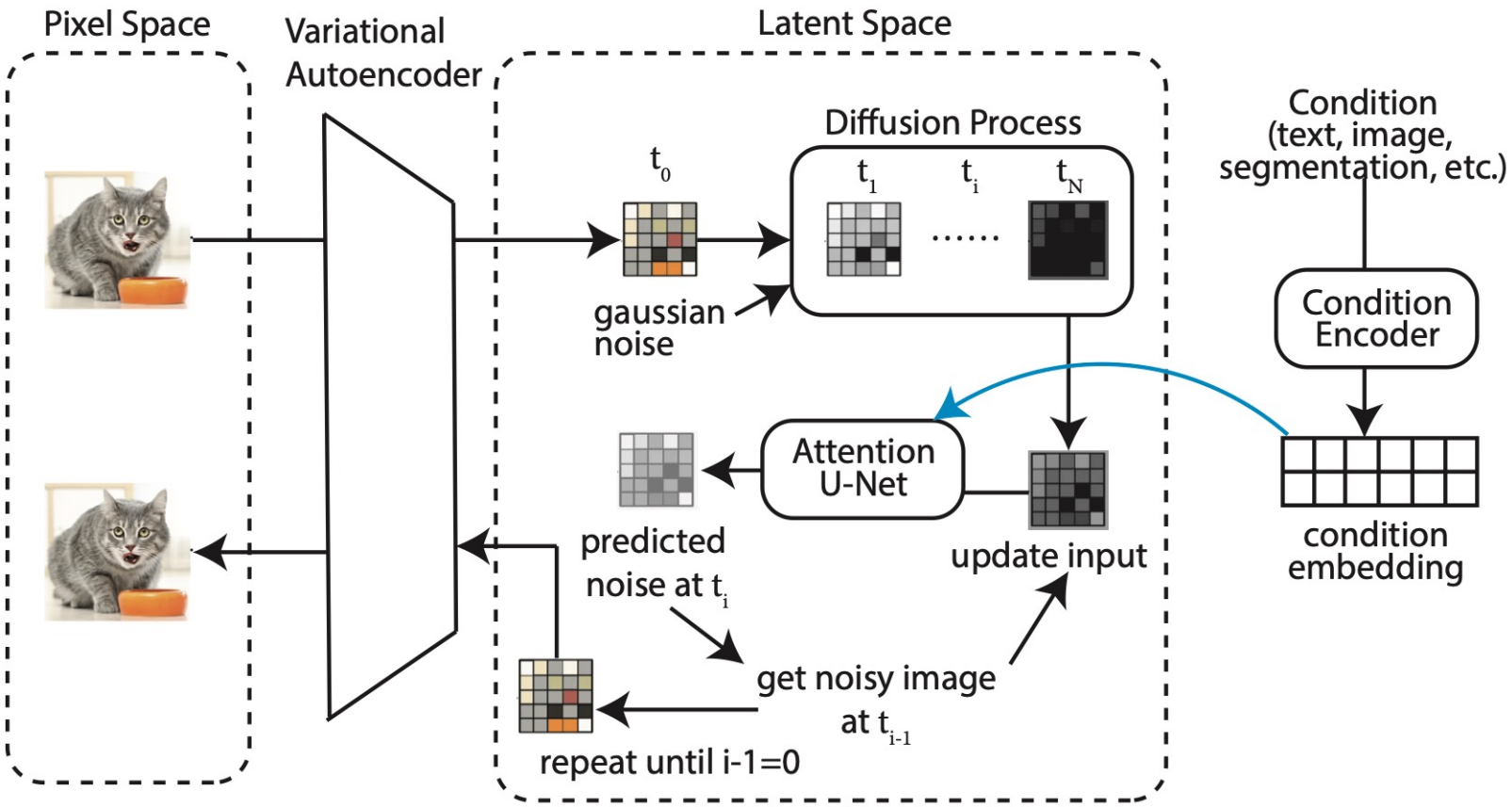


Figure: Figure 3 in the original paper.

LDMs: Autoencoder & Denoiser



LDMs: Autoencoder & Denoiser & Conditioning



LDMs: New-Image Generation

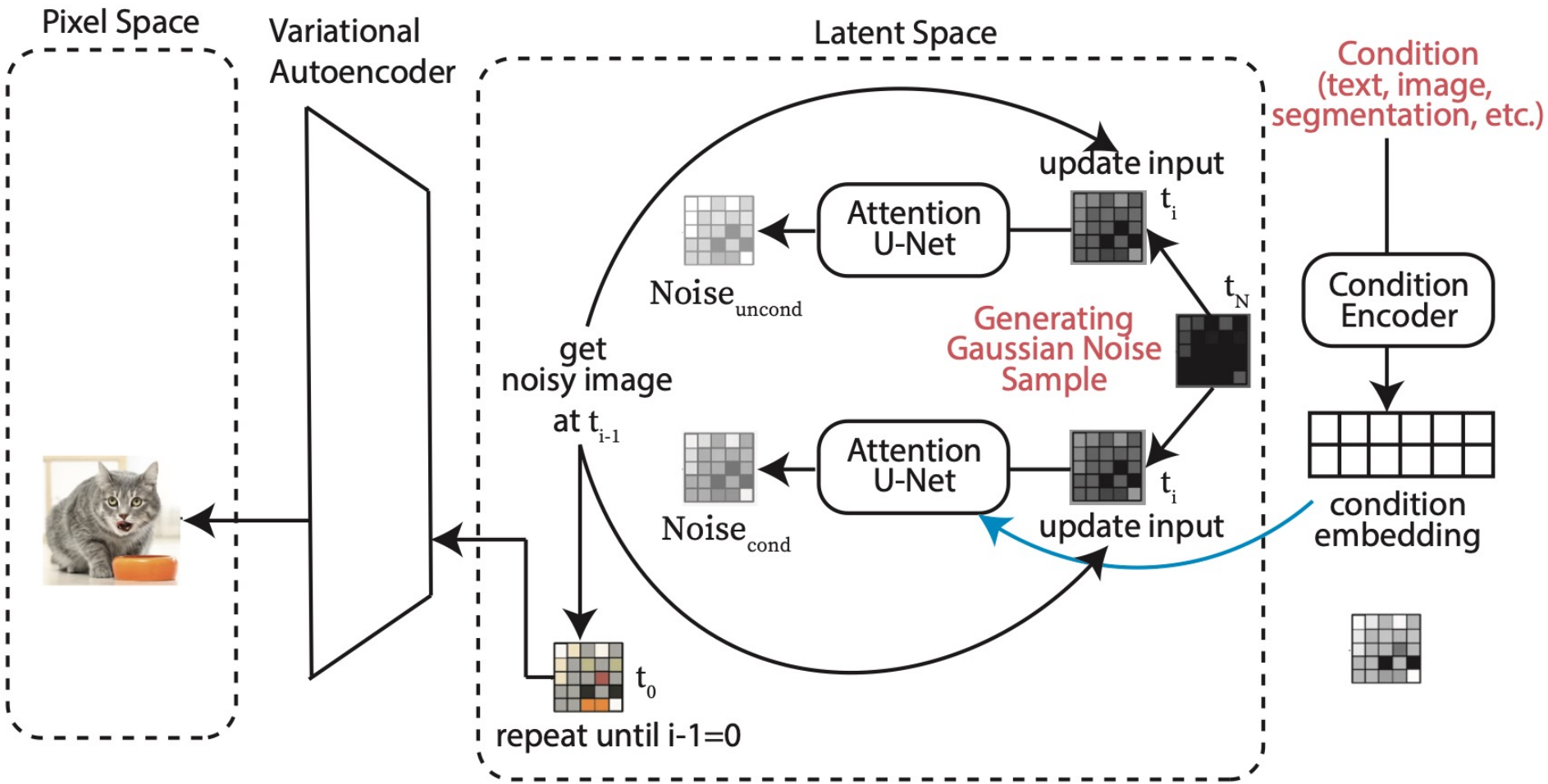
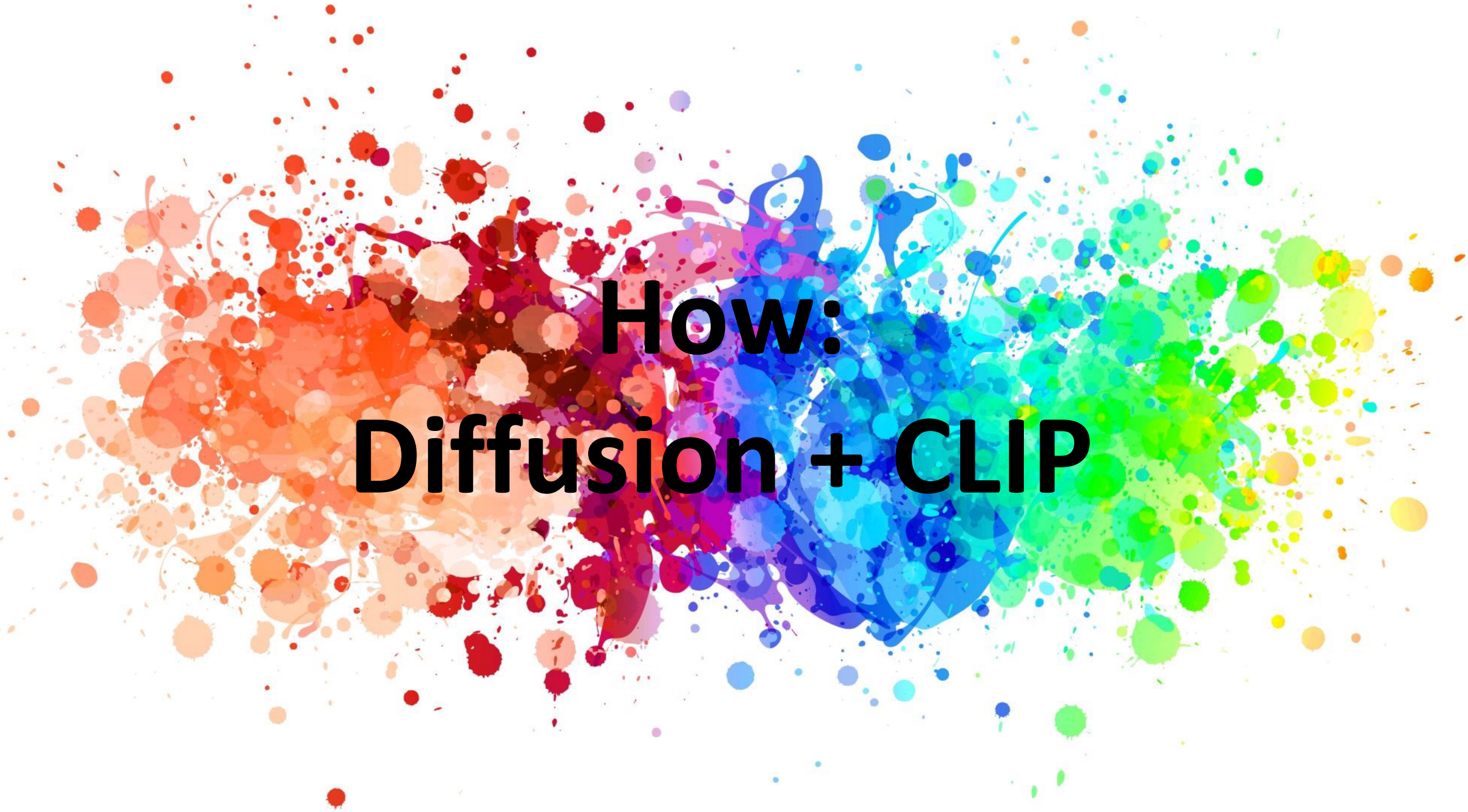


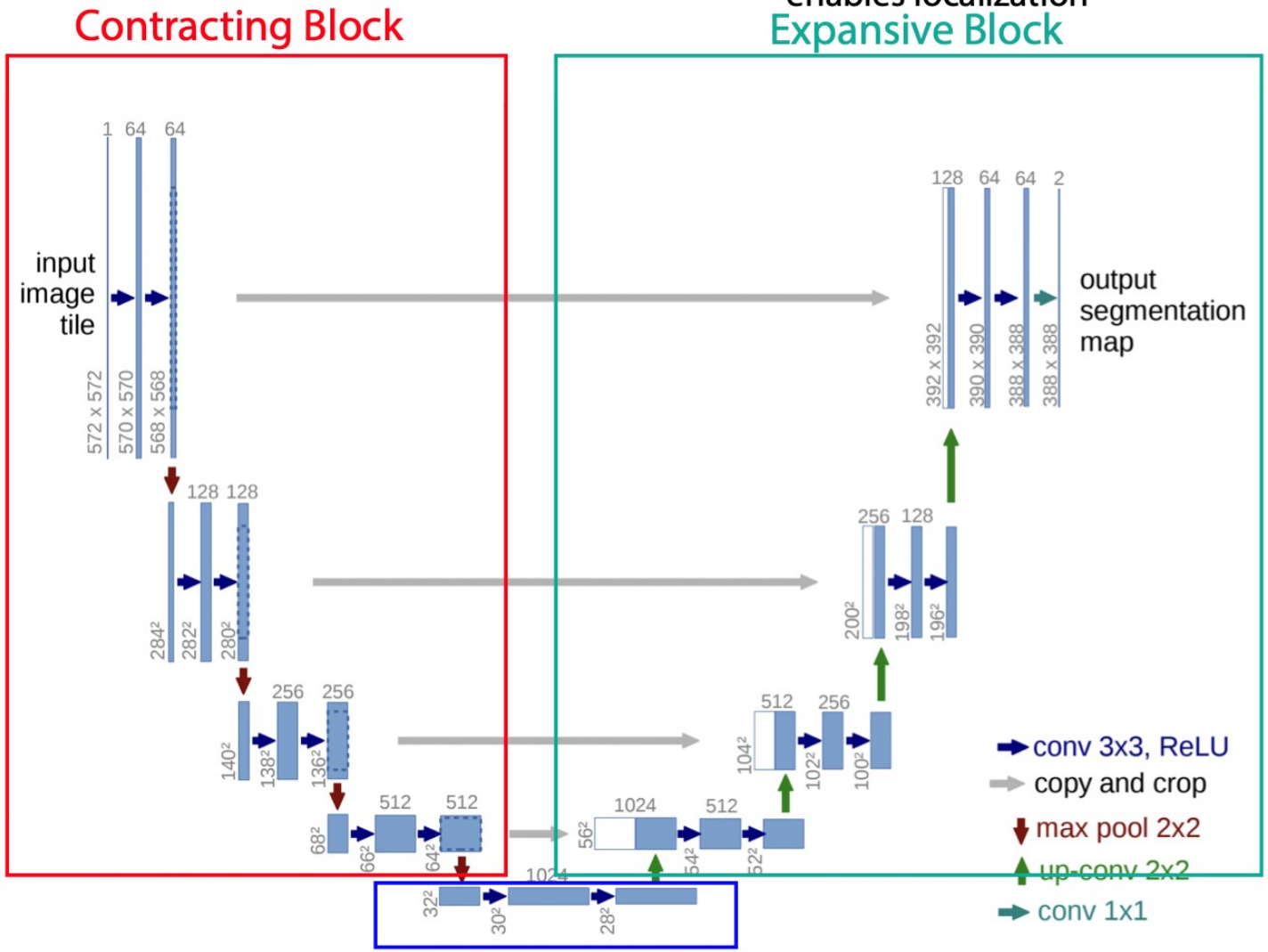
Figure: $Noise = Noise_{uncond} + \beta(Noise_{cond} - Noise_{uncond})$


How:
Diffusion + CLIP

U-Net

decreasing size
increasing feature
capture context

increasing size
decreasing feature
combining contextual information
enables localization



keep the same size
increasing feature
capture context

Attention U-Net

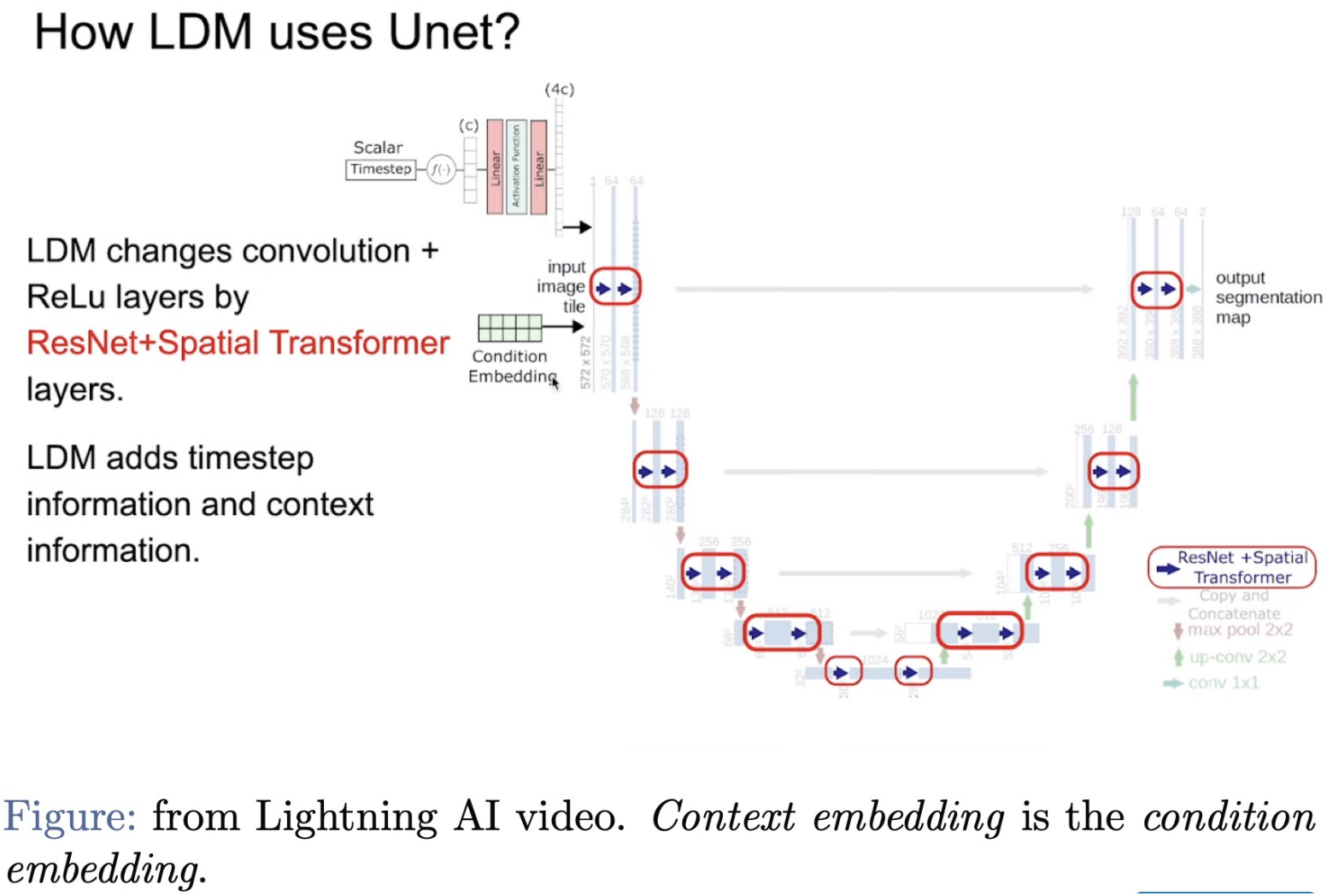


Figure: from Lightning AI video. *Context embedding* is the *condition embedding*.

ResNet block + timestep embedding

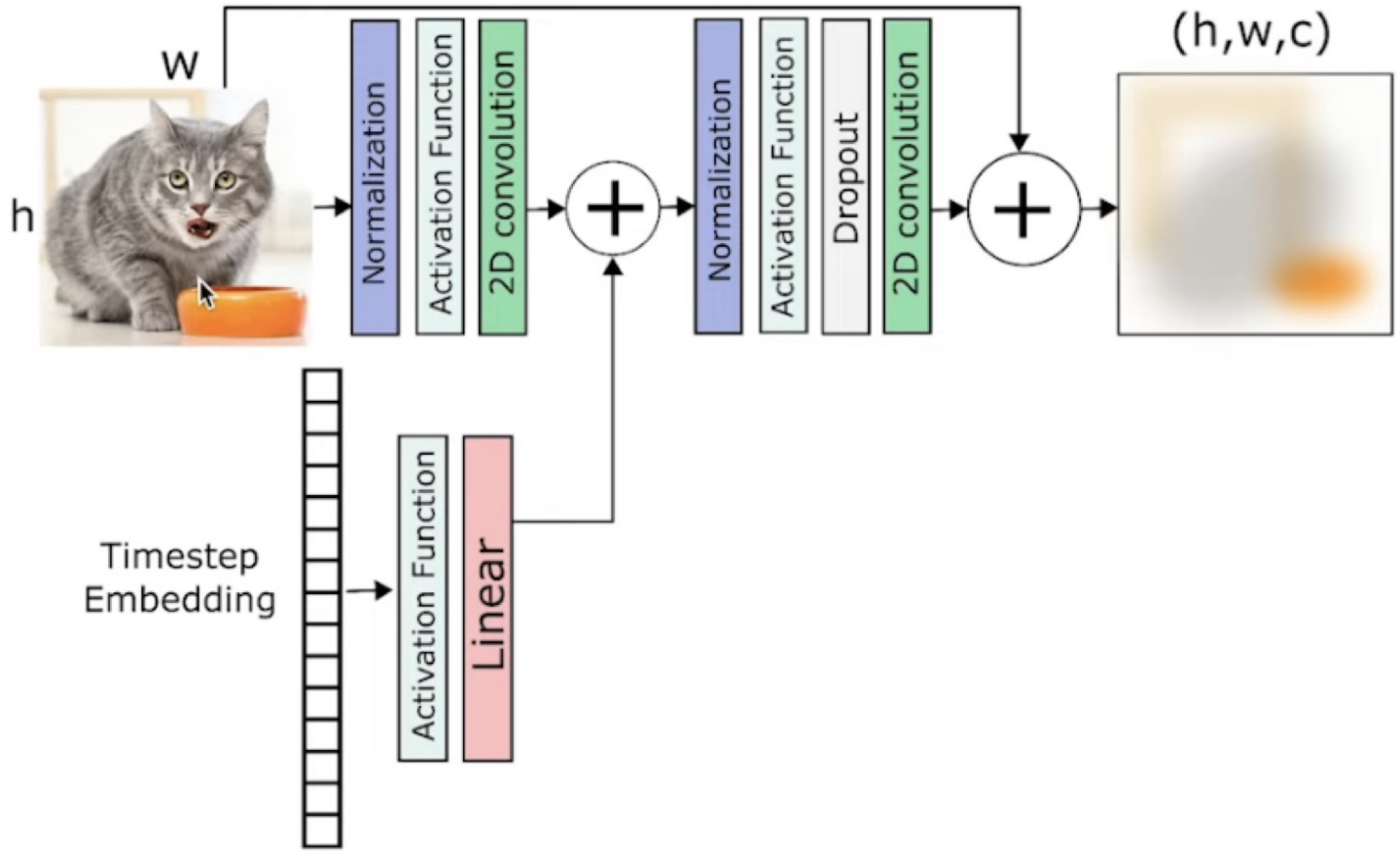


Figure: from Lightning AI video. Takes in time-step embedding.

Spacial Transformer

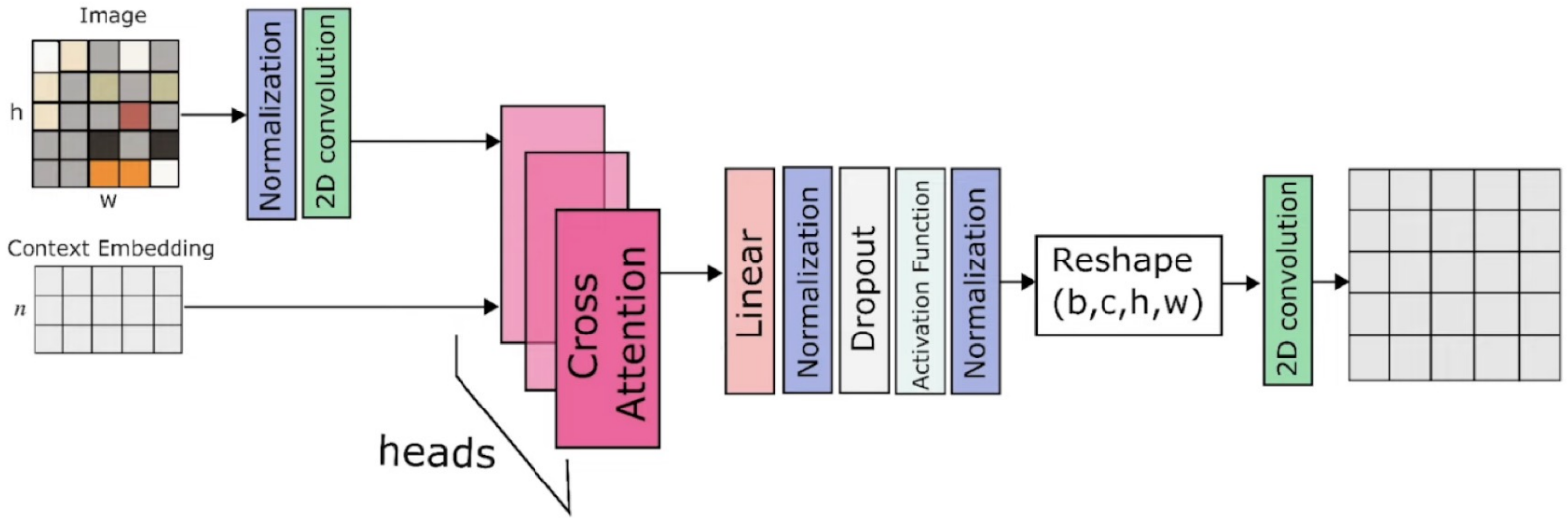
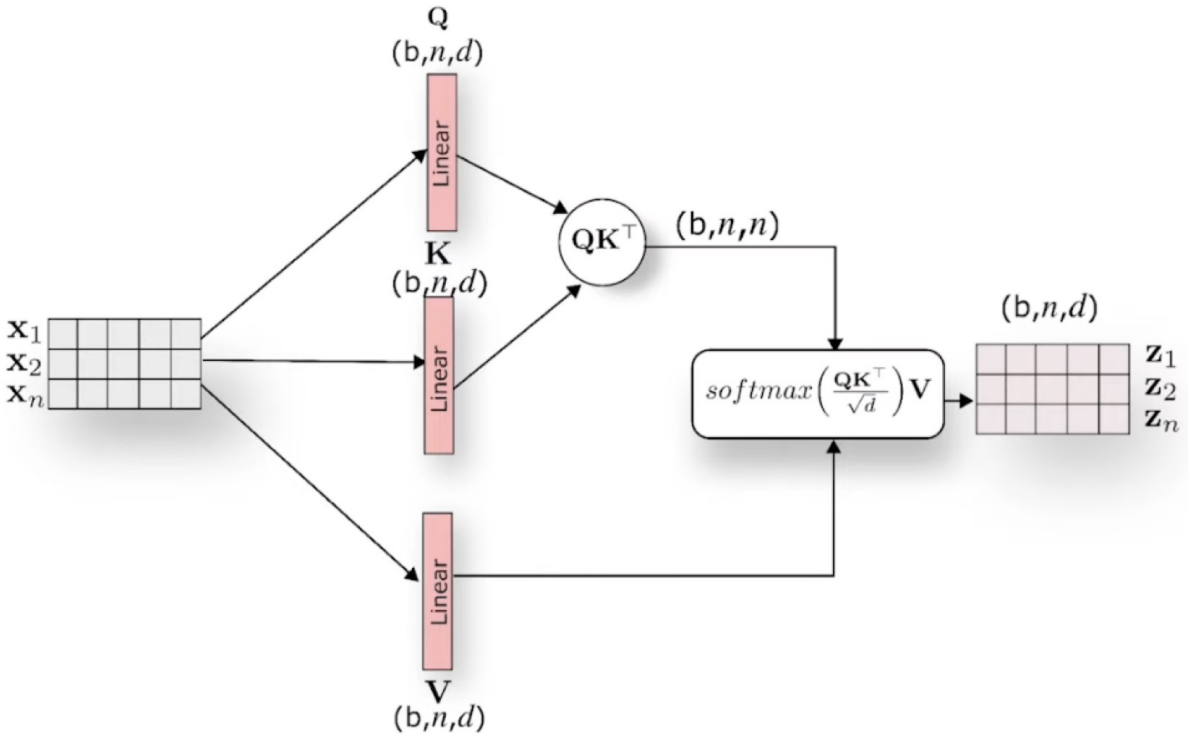
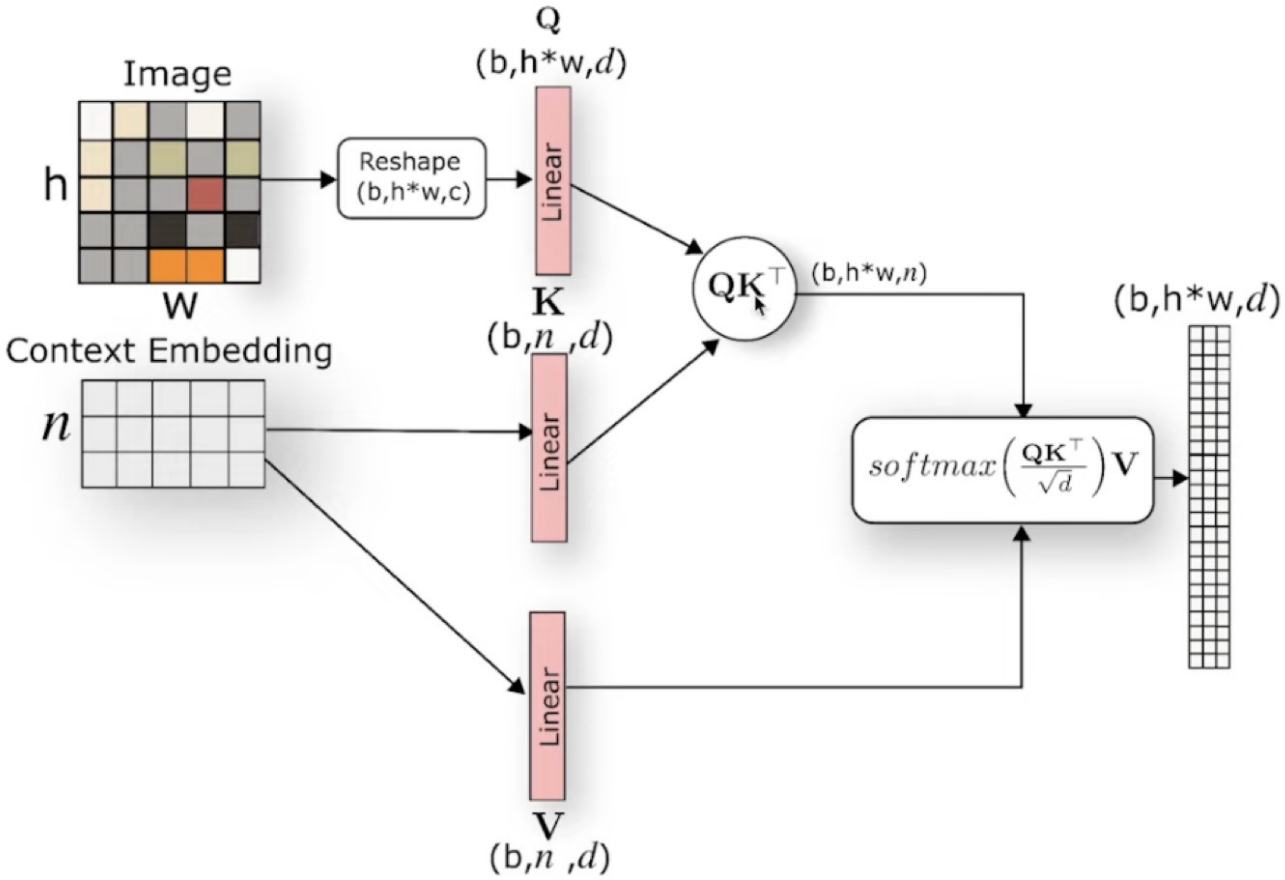


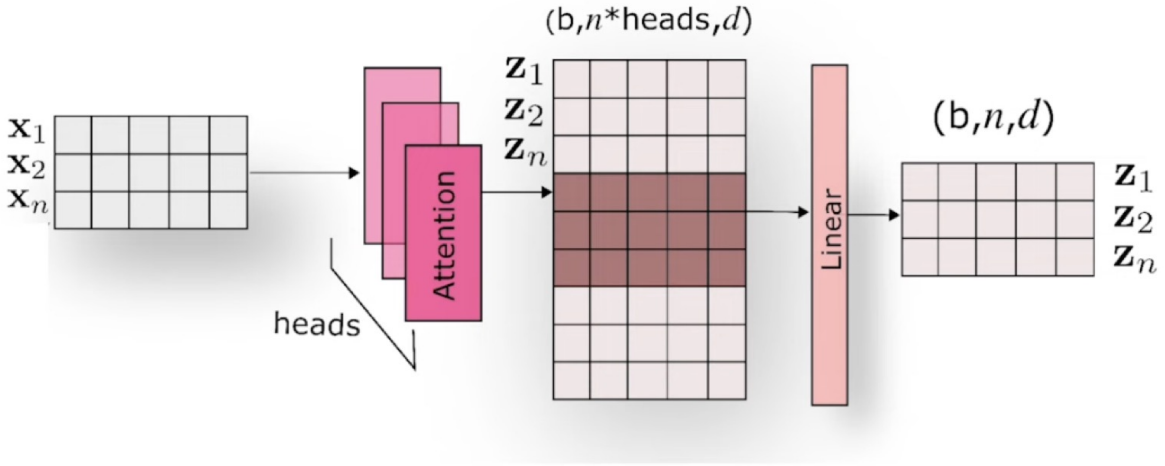
Figure: from Lightning AI video. Takes in context (condition) embedding.

Attention and cross-attention



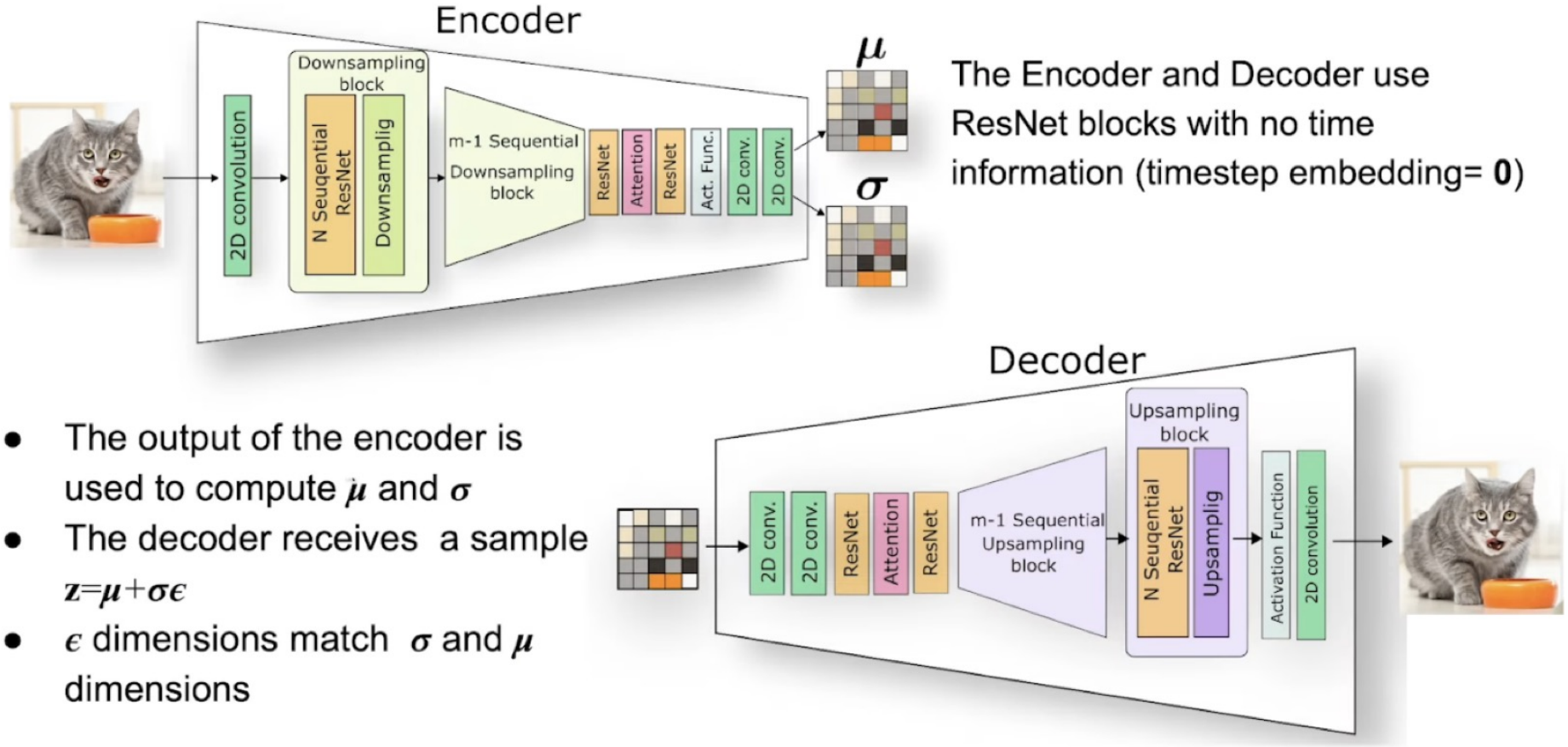


Multi-head attention



Q from image embedding, K and V both from context (condition) embedding.

Component Details: Variational Autoencoder



- The output of the encoder is used to compute μ and σ
- The decoder receives a sample $\mathbf{z}=\mu+\sigma\epsilon$
- ϵ dimensions match σ and μ dimensions

Figure: from Lightning AI video. Note: μ and σ can be vector instead of matrices, making implementation easier.

Component Details: Attention in VAE

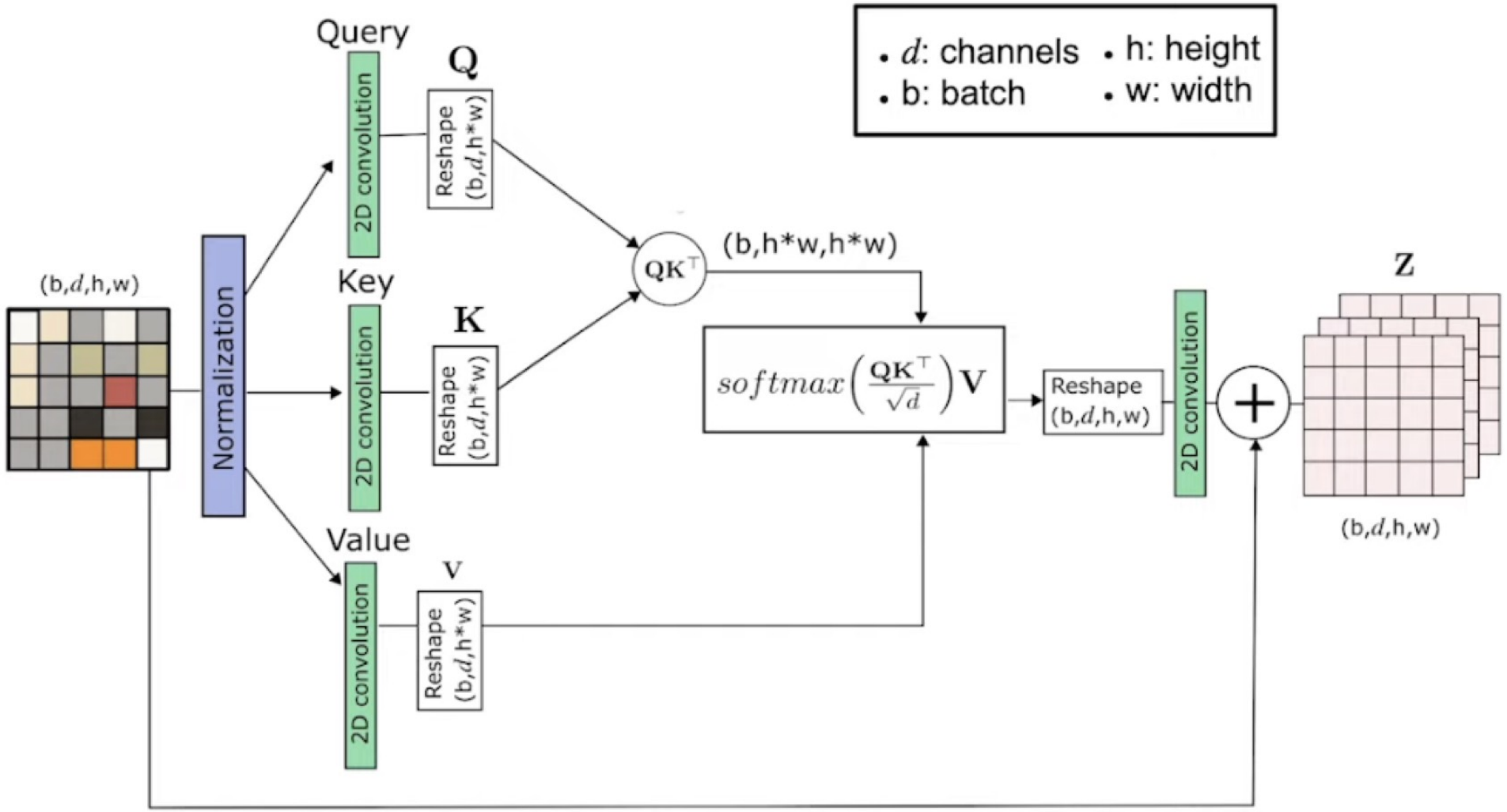


Figure: from Lightning AI video. Attention mechanism in the VAE.

Denoiser (U-Net) Training

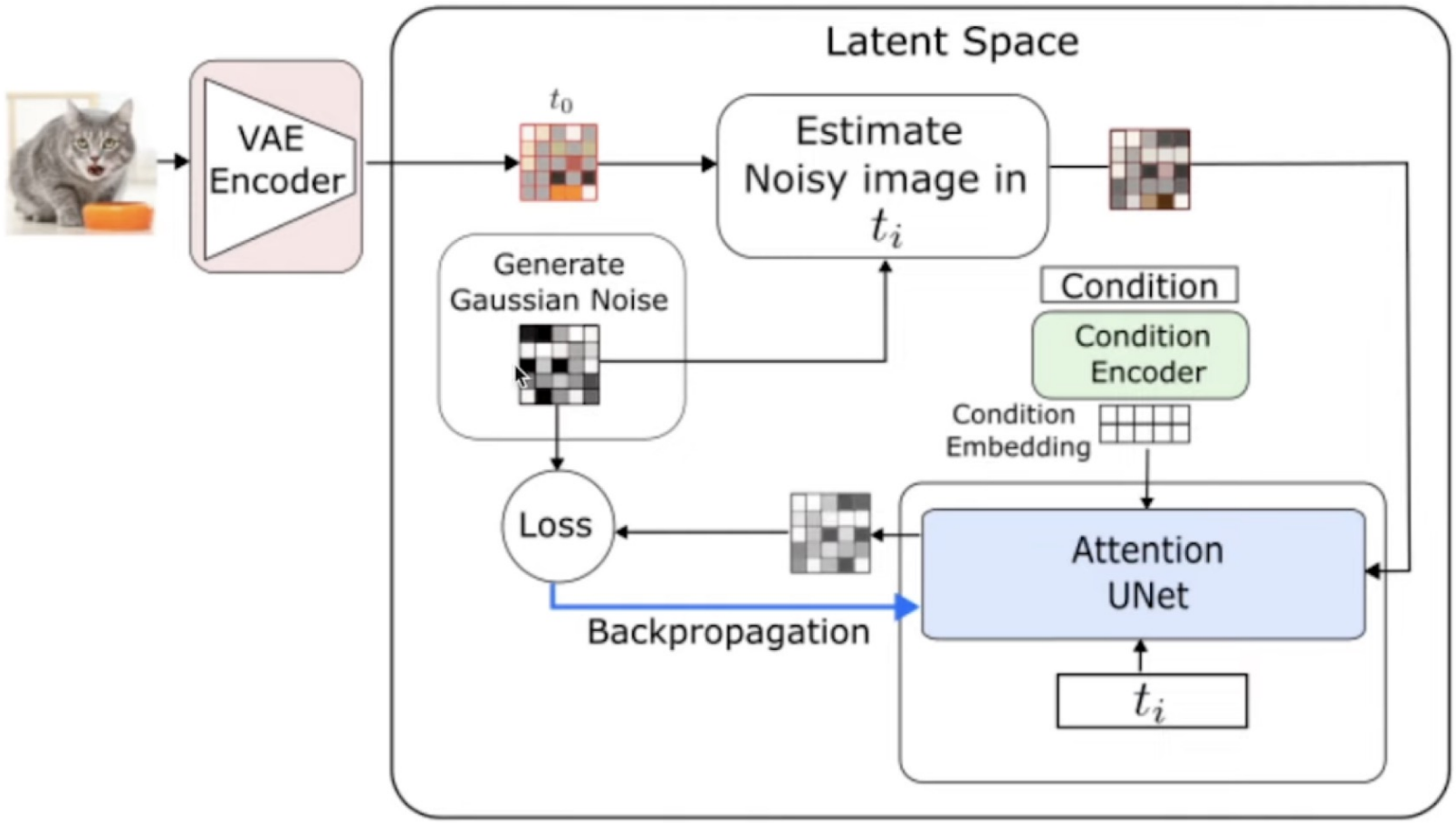


Figure: from Lightning AI video. Loss can be ℓ_p norm.

Condition Encoder (CLIP) Training

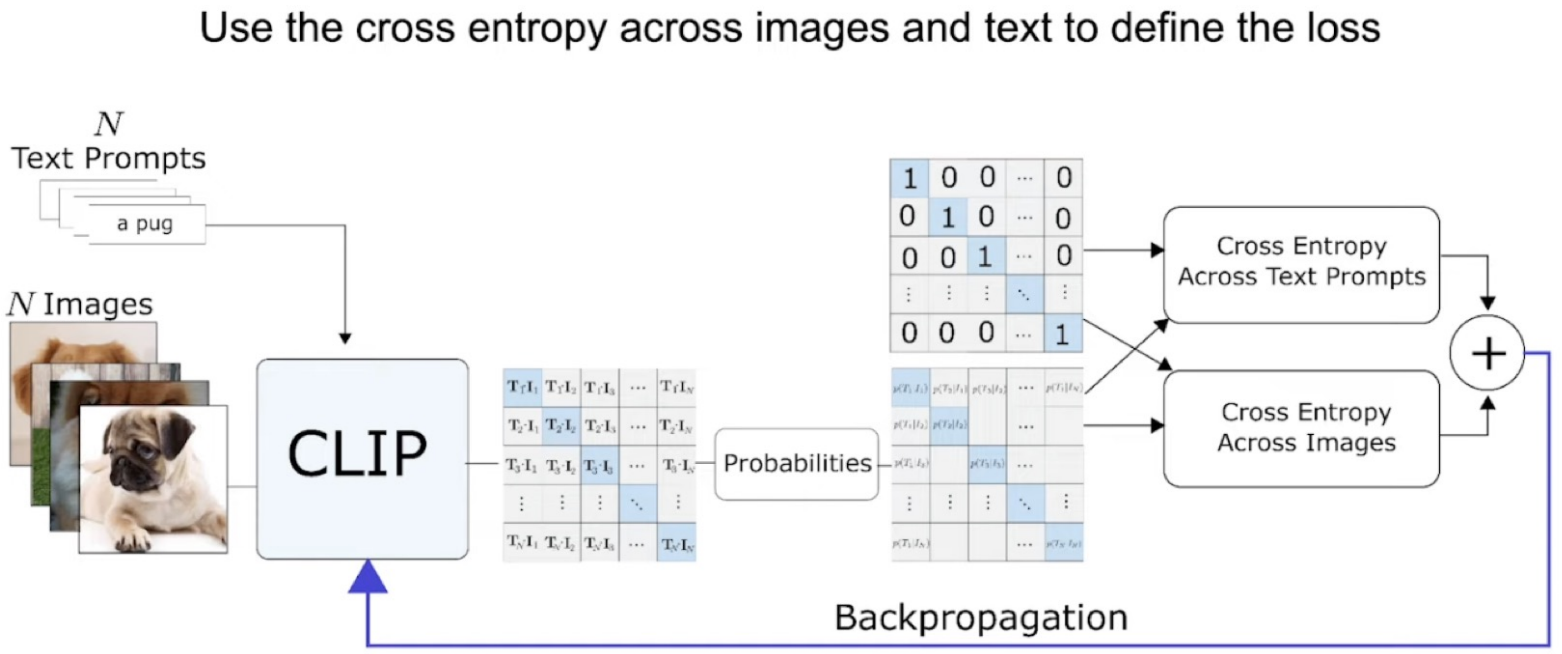


Figure: from Lightning AI video. Sum the loss across rows and columns (i.e., across text and across images) as loss.

Applications: Personalize

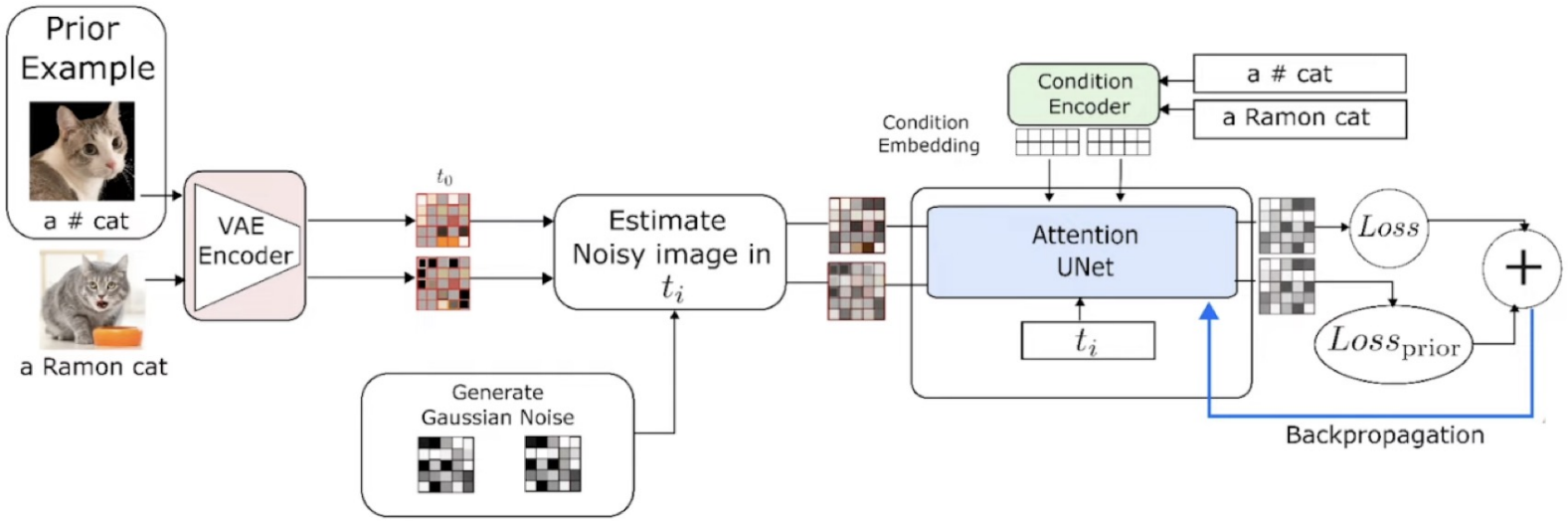
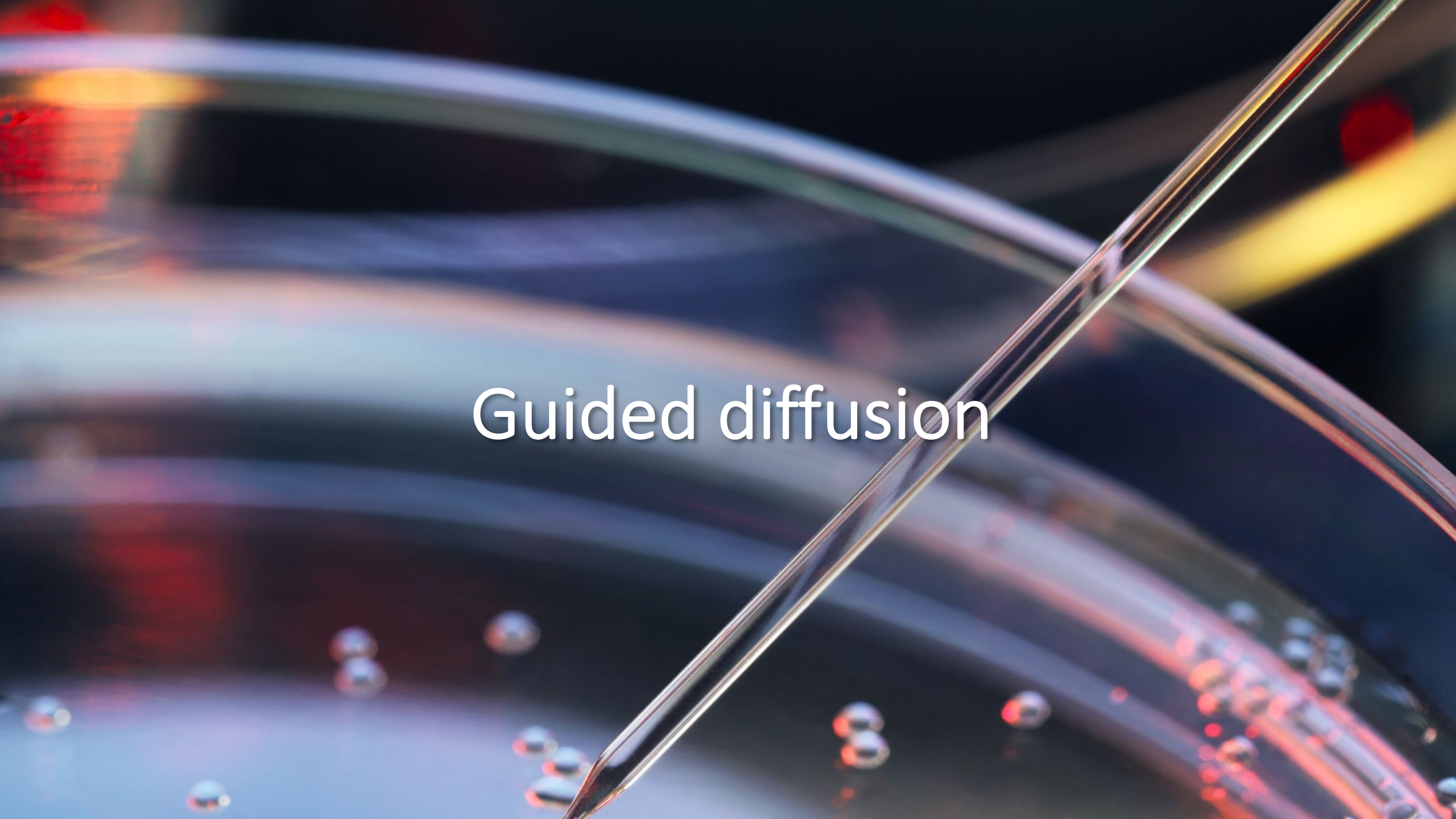


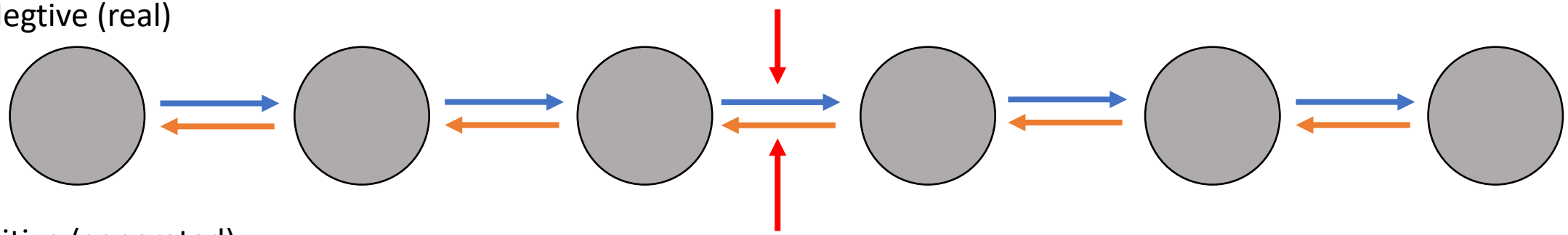
Figure: from Lightning AI video. In this example, a more general sample is provided to avoid language drift. An example of fine-tuning. A few-shot learning scenario.


Guided diffusion

Guided diffusion

Condition / guidance

Negtive (real)



Positive (generated)

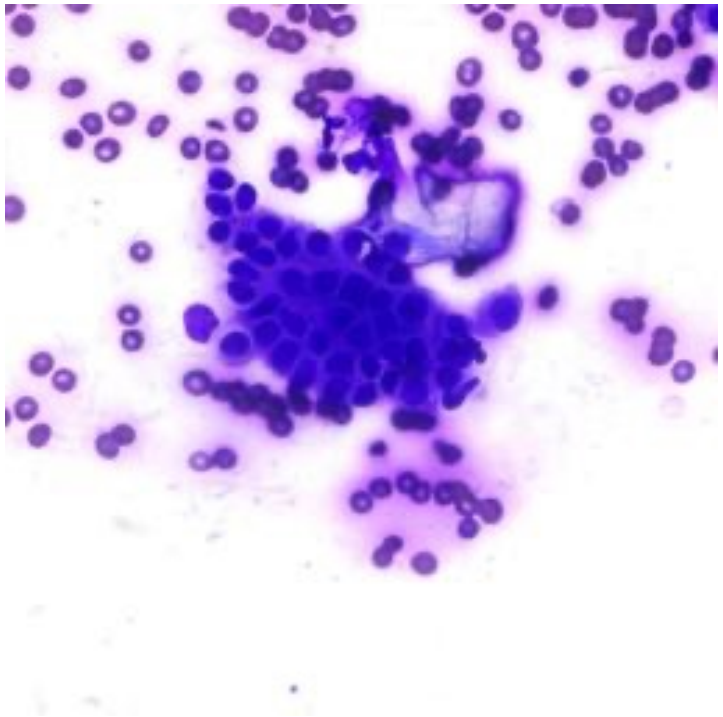
$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_t),$$

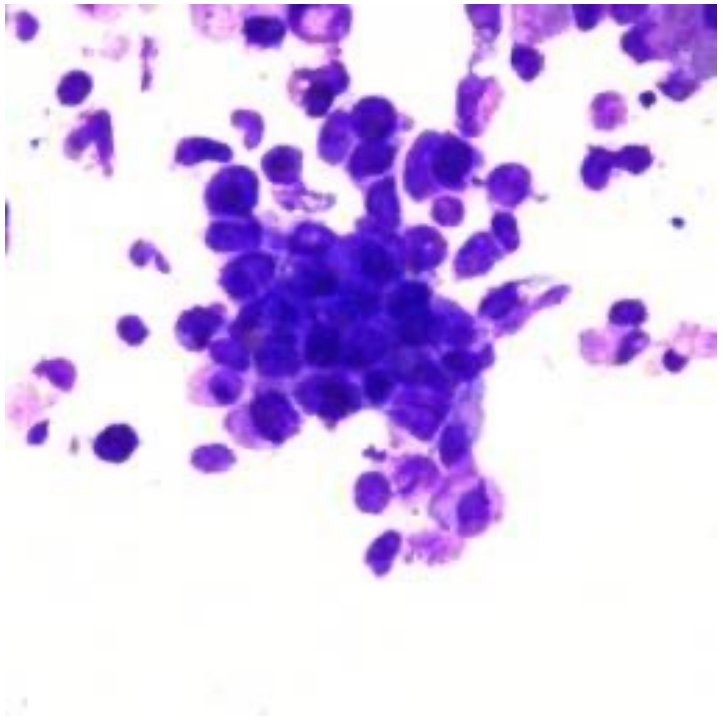
$$\tilde{\beta}_t = 1/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})$$

$$p(\boldsymbol{x}_{\tau_{i-1}}|\boldsymbol{x}_{\tau_i}) \approx \mathcal{N}\left(\boldsymbol{x}_{\tau_{i-1}}; \frac{\bar{\alpha}_{\tau_{i-1}}}{\bar{\alpha}_{\tau_i}}\left(\boldsymbol{x}_{\tau_i} - \left(\bar{\beta}_{\tau_i} - \frac{\bar{\alpha}_{\tau_i}}{\bar{\alpha}_{\tau_{i-1}}}\sqrt{\bar{\beta}^2_{\tau_{i-1}} - \tilde{\sigma}^2_{\tau_i}}\right)\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\tau_i}, \tau_i)\right), \tilde{\sigma}^2_{\tau_i}\boldsymbol{I}\right)$$

$$\sigma_t = \frac{\bar{\beta}_{t-1}\beta_t}{\bar{\beta}_t} = \frac{\bar{\beta}_{t-1}}{\bar{\beta}_t}\sqrt{1-\frac{\bar{\alpha}^2_t}{\bar{\alpha}^2_{t-1}}} \quad \rightarrow \quad \frac{\bar{\beta}_{\tau_{i-1}}}{\bar{\beta}_{\tau_i}}\sqrt{1-\frac{\bar{\alpha}^2_{\tau_i}}{\bar{\alpha}^2_{\tau_{i-1}}}} = \tilde{\sigma}_{\tau_i}$$

Guided diffusion results

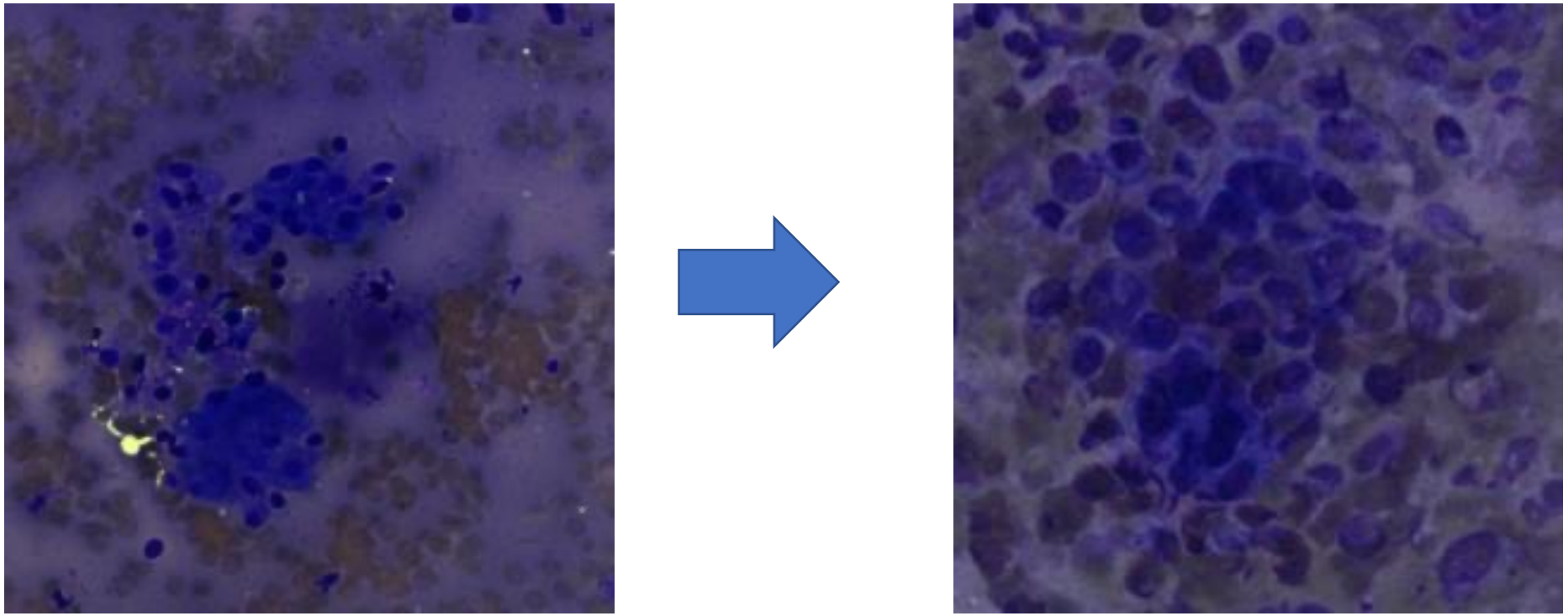




Negtive (real)

Positive (generated)

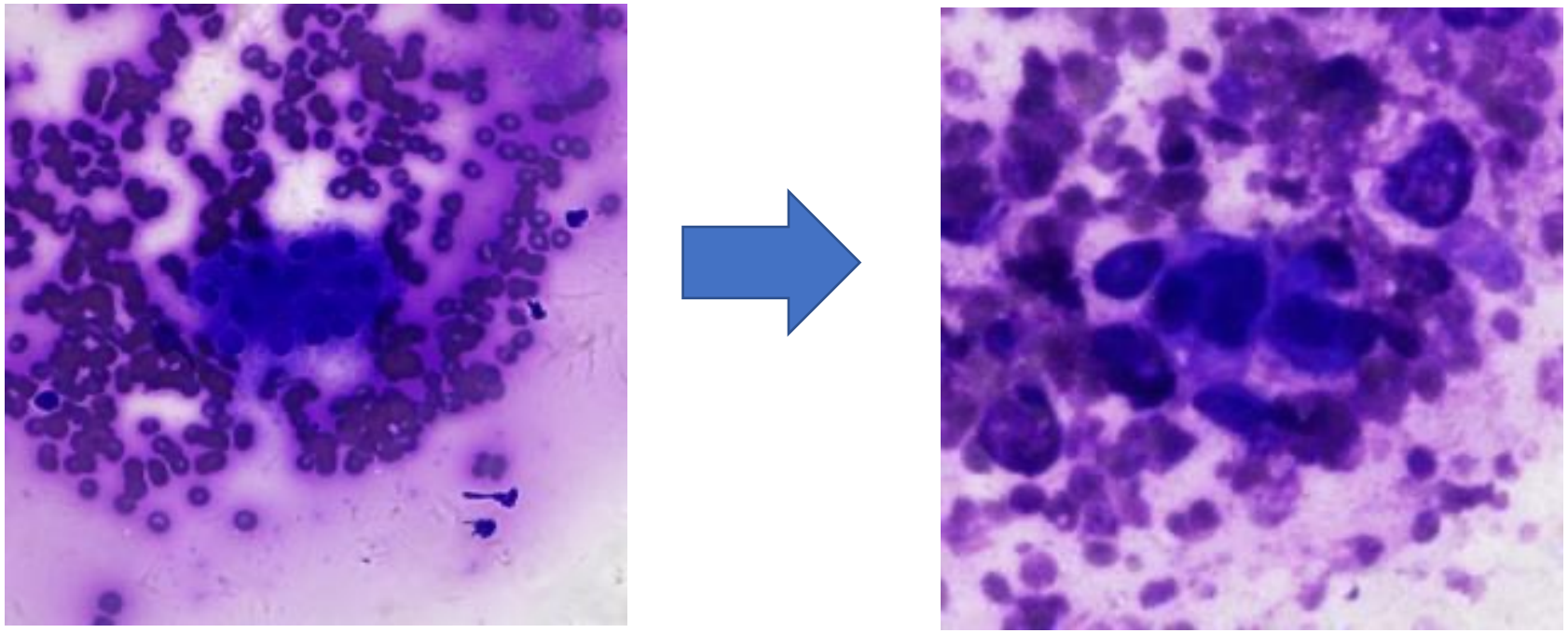
Guided diffusion results



Negtive (real)

Positive (generated)

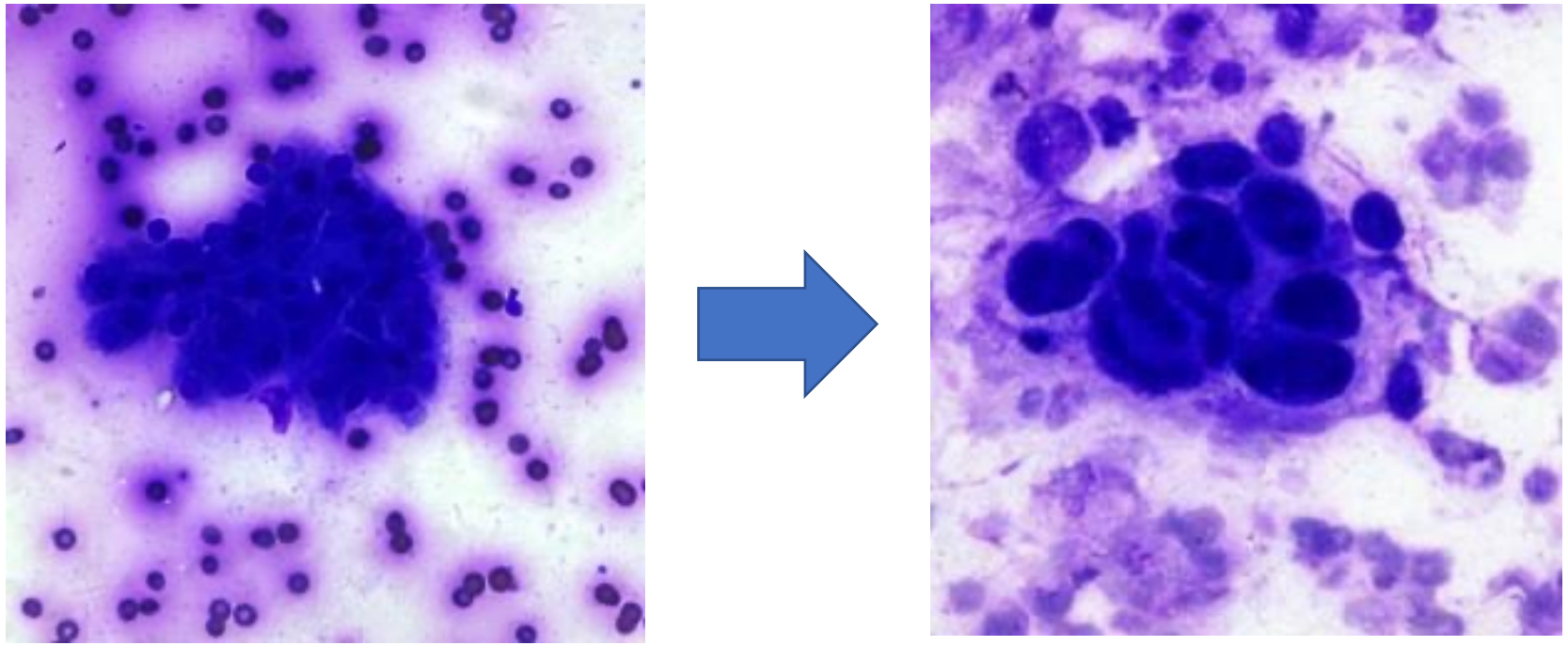
Guided diffusion results



Negtive (real)

Positive (generated)

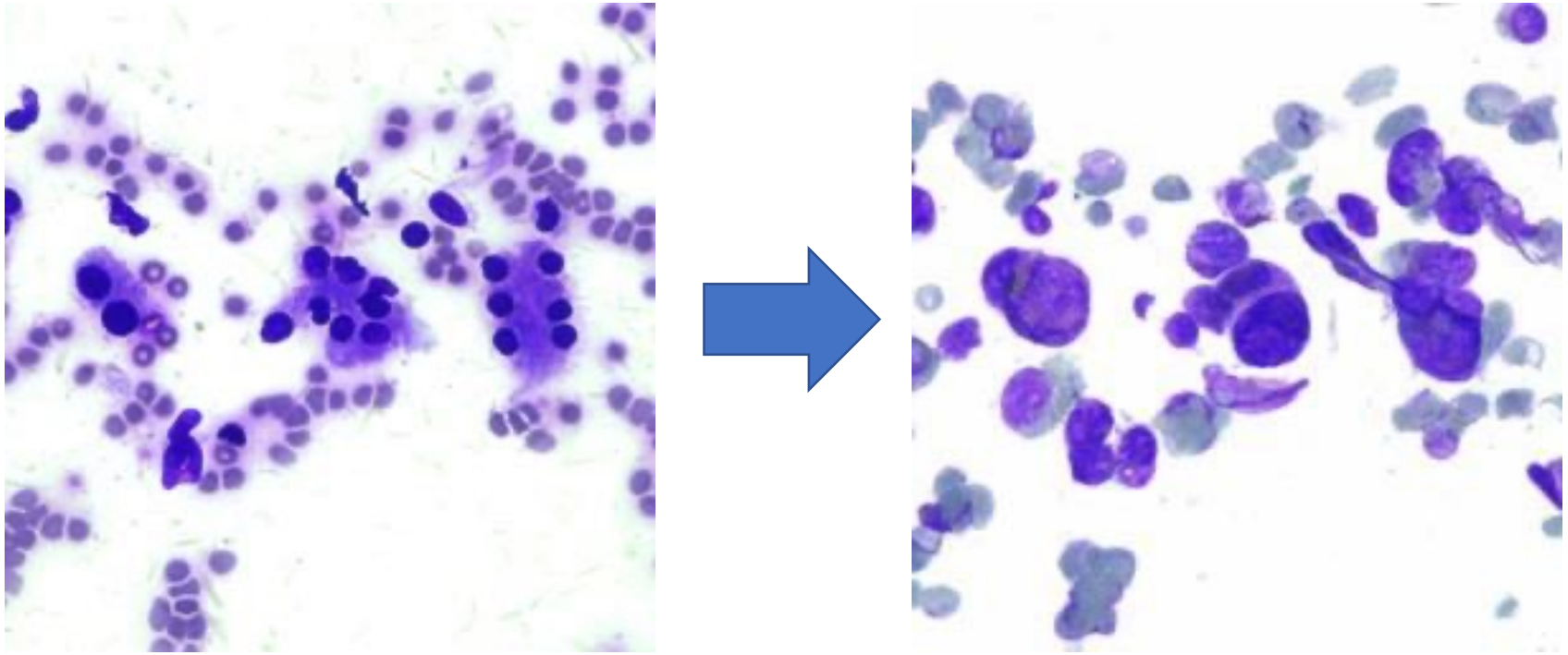
Guided diffusion results



Negtive (real)

Positive (generated)

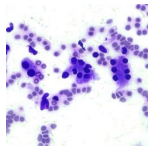
Guided diffusion results



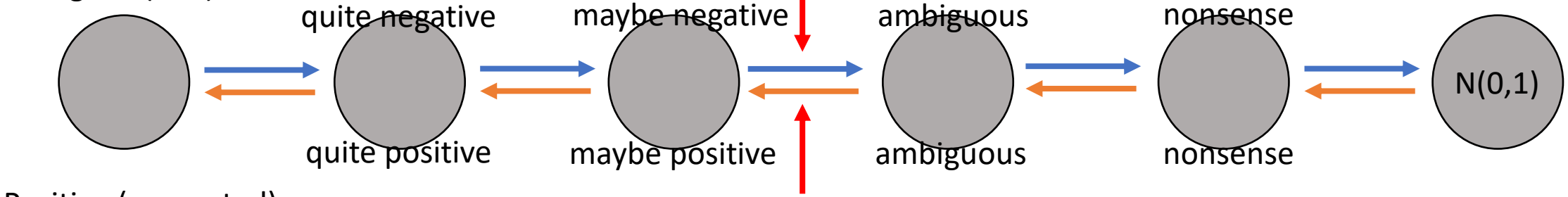
Negtive (real)

Positive (generated)

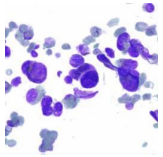
A better supervision



Negtive (real)



Positive (generated)

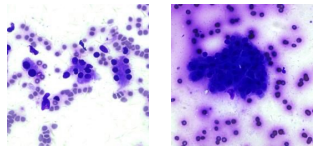


By undersdanding how to turn possitive into negative (turn negative to possitive),

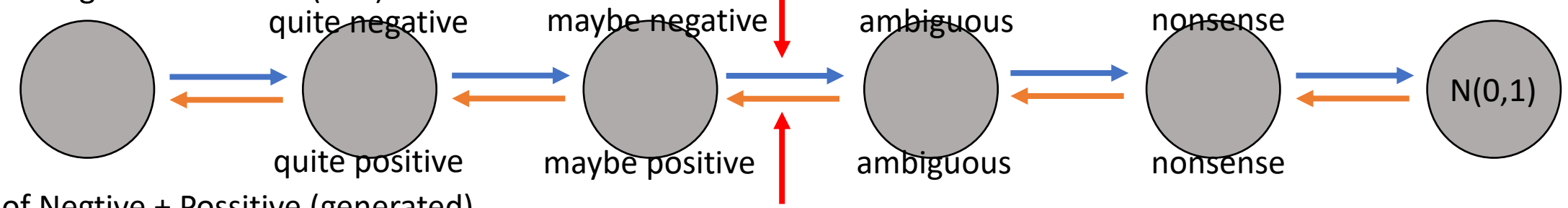
the model can be guided understand the essence of ‘what’s negative, what’s possitive, what’s invarient part’ ?

The process may not gurantee our ambition, that calls.

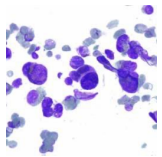
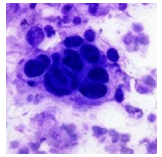
Guided diffusion process



A bag of Negtive + Possitive (real)



A bag of Negtive + Possitive (generated)

Every time we control the mixture of the tokens forming the bag.
To make mode learning the difference in distribution ?