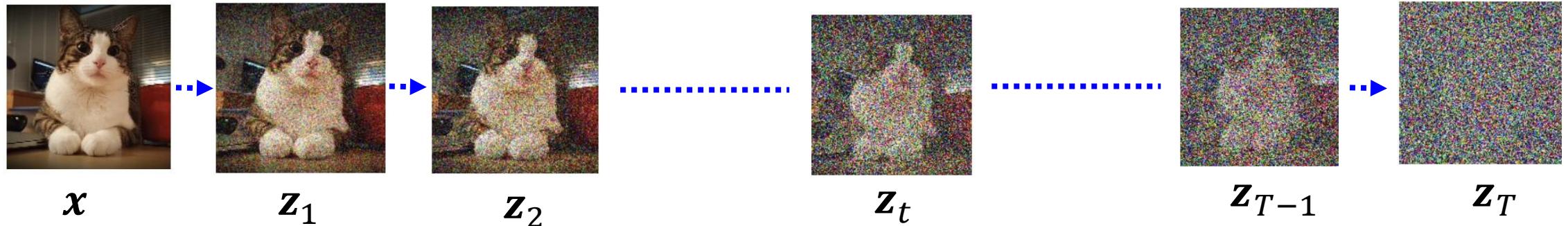


Diffusion Models

Denoising Diffusion Models

- Consider gradually corrupting an image ($\mathbf{z}_0 = \mathbf{x}$) till it becomes **pure noise** (\mathbf{z}_T)



- Each step $\mathbf{z}_{t-1} \rightarrow \mathbf{z}_t$ is a pre-defined Gaussian perturbation (**forward process**)

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

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

$$\beta_t \in (0, 1) \quad \text{and} \quad \beta_1 < \beta_2 < \dots < \beta_{T-1} < \beta_T$$

Usually pre-defined but
can also be learned

Imp: Thus we can also **compute \mathbf{z}_t from \mathbf{x} directly in a single step**

implies

$$q(\mathbf{z}_t | \mathbf{x}) = \mathcal{N}(\mathbf{z}_t | \sqrt{\alpha_t} \mathbf{x}, (1 - \alpha_t) \mathbf{I})$$

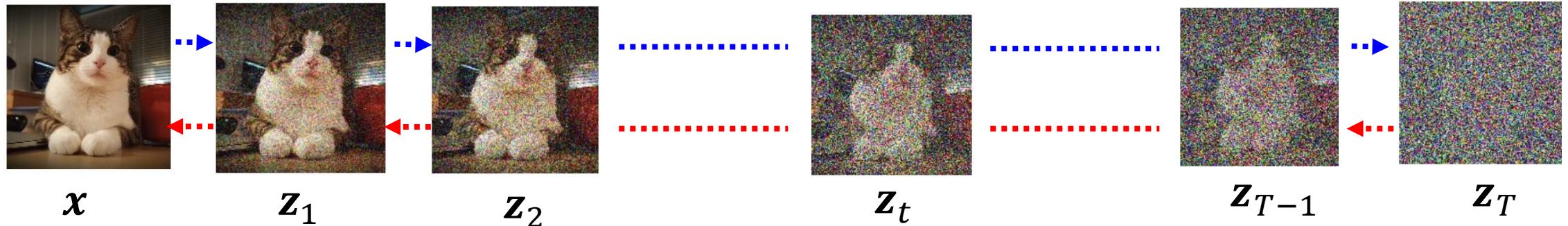
where $\alpha_t = \prod_{\tau=1}^t (1 - \beta_\tau)$

$$\mathbf{z}_t = \sqrt{\alpha_t} \mathbf{x} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}$$

$$q(\mathbf{z}_T | \mathbf{x}) = \mathcal{N}(\mathbf{z}_T | \mathbf{0}, \mathbf{I}) \quad \text{as } T \rightarrow \infty$$

Generating Data by Reversing Diffusion

- Reversing the diffusion (red arrows) would enable generating data from pure noise



- To reverse the diffusion, we need the distribution of z_{t-1} given z_t , i.e., $q(z_{t-1}|z_t)$

$$q(z_{t-1}|z_t) = \frac{q(z_{t-1}) q(z_t|z_{t-1})}{q(z_t)}$$

The denoising distribution

Intractable because $q(z_t)$ and $q(z_{t-1})$ are difficult to compute

Since the true data distribution $p(x)$ is not known, we can't compute this integral

$$q(z_t) = \int q(z_t|x)p(x)dx$$

Towards a Tractable Reverse Diffusion

- Although $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ isn't tractable, the following distribution is tractable

$$\begin{aligned} q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) &= \frac{q(\mathbf{z}_{t-1}|\mathbf{x}) q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x})}{q(\mathbf{z}_t|\mathbf{x})} \\ &= \frac{q(\mathbf{z}_{t-1}|\mathbf{x}) q(\mathbf{z}_t|\mathbf{z}_{t-1})}{q(\mathbf{z}_t|\mathbf{x})} \end{aligned}$$

- Reason: $q(\mathbf{z}_{t-1}|\mathbf{x})$ and $q(\mathbf{z}_t|\mathbf{z}_{t-1})$ are Gaussians, so $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$ is Gaussian

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) = \mathcal{N}(\mathbf{z}_{t-1} | \mathbf{m}(\mathbf{x}, \mathbf{z}_t), \sigma_t^2 \mathbf{I})$$

Using $\mathbf{x} = \frac{1}{\sqrt{\alpha_t}} \mathbf{z}_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}} \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$$\mathbf{m}(\mathbf{x}, \mathbf{z}_t) = \frac{(1 - \alpha_{t-1})\sqrt{1 - \beta_t} \mathbf{z}_t + \sqrt{\alpha_{t-1}}\beta_t \mathbf{x}}{1 - \alpha_t}$$

$$\sigma_t^2 = \frac{\beta_t(1 - \alpha_{t-1})}{1 - \alpha_t}$$

$$= \frac{1}{\sqrt{1 - \beta_t}} \left\{ \mathbf{z}_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \boldsymbol{\epsilon} \right\}$$

Towards a Tractable Reverse Diffusion

- We saw that the reverse diffusion distribution is the Gaussian

$$q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x}) = \mathcal{N}(\mathbf{z}_{t-1} | \mathbf{m}(\mathbf{x}, \mathbf{z}_t), \sigma_t^2 \mathbf{I})$$

where

$$\mathbf{m}(\mathbf{x}, \mathbf{z}_t) = \frac{1}{\sqrt{1 - \beta_t}} \left\{ \mathbf{z}_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \boldsymbol{\epsilon} \right\}$$

Issue: At generation time, we don't have \mathbf{x} (the goal is to generate \mathbf{x} which is only available for training data) so we can't use $\mathbf{m}(\mathbf{x}, \mathbf{z}_t)$ at generation time as $\boldsymbol{\epsilon}$ depends on \mathbf{x}



- Let's approximate $q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})$ by another Gaussian that doesn't depend on \mathbf{x}

$$p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w}) = \mathcal{N}(\mathbf{z}_{t-1} | \mu(\mathbf{z}_t, \mathbf{w}, t), \Sigma(\mathbf{z}_t, \mathbf{w}, t))$$

- Usually, $\Sigma(\mathbf{z}_t, \mathbf{w}, t)$ is chosen to be spherical. A popular choice: $\Sigma(\mathbf{z}_t, \mathbf{w}, t) = \beta_t \mathbf{I}$
- The mean $\mu(\mathbf{z}_t, \mathbf{w}, t)$ is defined to mimic the form of $\mathbf{m}(\mathbf{x}, \mathbf{z}_t)$

$$\mu(\mathbf{z}_t, \mathbf{w}, t) = \frac{1}{\sqrt{1 - \beta_t}} \left\{ \mathbf{z}_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} g(\mathbf{z}_t, \mathbf{w}, t) \right\}$$

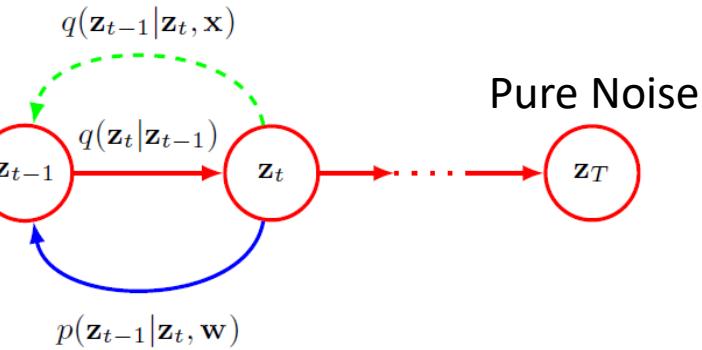
Learn to predict the noise $\boldsymbol{\epsilon}$ using g

Reversing the Diffusion

- The joint distribution of data and latents

$$p(\mathbf{x}, \mathbf{z}_1, \dots, \mathbf{z}_T | \mathbf{w}) = p(\mathbf{z}_T) \left\{ \prod_{t=2}^T p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w}) \right\} p(\mathbf{x} | \mathbf{z}_1, \mathbf{w})$$

- Let's assume $p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w}) = \mathcal{N}(\mathbf{z}_{t-1} | \boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t), \beta_t \mathbf{I})$



- The true joint distribution of the latents given \mathbf{x}

$$q(z_1, z_2, \dots, z_T | x) = q(z_1 | x) \prod_{t=2}^T q(z_t | z_{t-1}, x)$$

- To estimate \mathbf{w} , we can maximize the ELBO defined as

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_q \left[\ln \frac{p(\mathbf{z}_T) \left\{ \prod_{t=2}^T p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w}) \right\} p(\mathbf{x} | \mathbf{z}_1, \mathbf{w})}{q(\mathbf{z}_1 | \mathbf{x}) \prod_{t=2}^T q(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x})} \right] = \mathbb{E}_q \left[\ln p(\mathbf{z}_T) + \sum_{t=2}^T \ln \frac{p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x})} - \ln q(\mathbf{z}_1 | \mathbf{x}) + \ln p(\mathbf{x} | \mathbf{z}_1, \mathbf{w}) \right]$$

From ELBO

$$\mathbb{E}_q \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right]$$

Also note that unlike VI, here we aren't estimating the q distribution

First and third terms don't contain \mathbf{w} so can be ignored when maximizing the ELBO

This term is just like the VAE reconstruction error term (can approximate it using samples of \mathbf{z}_1 from $q(\mathbf{z}_1 | \mathbf{x})$)

ELBO (contd)

- Recall the ELBO for the denoising diffusion model

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_q \left[\ln p(\mathbf{z}_T) + \sum_{t=2}^T \ln \frac{p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x})} - \ln q(\mathbf{z}_1|\mathbf{x}) + \ln p(\mathbf{x}|\mathbf{z}_1, \mathbf{w}) \right]$$

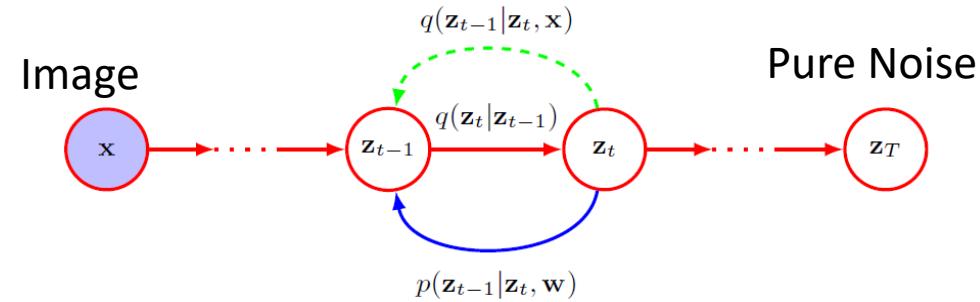
- Ignoring terms that don't depend on \mathbf{w} and using $q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}) = \frac{q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})q(\mathbf{z}_t|\mathbf{x})}{q(\mathbf{z}_{t-1}|\mathbf{x})}$

$$\ln \frac{p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x})} = \ln \frac{p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})} + \ln \frac{q(\mathbf{z}_{t-1}|\mathbf{x})}{q(\mathbf{z}_t|\mathbf{x})} \quad \xrightarrow{\text{blue arrow}} \quad \mathcal{L}(\mathbf{w}) = \mathbb{E}_q \left[\sum_{t=2}^T \ln \frac{p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})} + \ln p(\mathbf{x}|\mathbf{z}_1, \mathbf{w}) \right]$$

- The ELBO becomes $\mathcal{L}(\mathbf{w}) = \underbrace{\int q(\mathbf{z}_1|\mathbf{x}) \ln p(\mathbf{x}|\mathbf{z}_1, \mathbf{w}) d\mathbf{z}_1}_{\text{reconstruction term}} - \underbrace{\sum_{t=2}^T \int \text{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) \| p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})) q(\mathbf{z}_t|\mathbf{x}) d\mathbf{z}_t}_{\text{consistency terms}}$

- Since both distributions in the KL divergence term are Gaussians, it becomes

$$\text{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) \| p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})) = \frac{1}{2\beta_t} \|\mathbf{m}_t(\mathbf{x}, \mathbf{z}_t) - \boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t)\|^2 + \text{const}$$



Predicting the noise

- The KL terms in the ELBO are of the form

$$\text{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})\|p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})) = \frac{1}{2\beta_t} \|\mathbf{m}_t(\mathbf{x}, \mathbf{z}_t) - \boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t)\|^2 + \text{const}$$

Network which gives the mean of the denoised \mathbf{z}_{t-1}

- Note that

$$\mathbf{x} = \frac{1}{\sqrt{\alpha_t}} \mathbf{z}_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}} \boldsymbol{\epsilon}_t \quad \xrightarrow{\text{From the definition of } m_t(x, z_t)} \quad \mathbf{m}_t(\mathbf{x}, \mathbf{z}_t) = \frac{1}{\sqrt{1-\beta_t}} \left\{ \mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \boldsymbol{\epsilon}_t \right\}$$

- Instead of learning $\boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t)$, we will learn a **noise predictor** $\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t)$ s.t.

Using the same form as \mathbf{m}_t with $\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t)$ trying to predict $\boldsymbol{\epsilon}_t$

$$\boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t) = \frac{1}{\sqrt{1-\beta_t}} \left\{ \mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \mathbf{g}(\mathbf{z}_t, \mathbf{w}, t) \right\}$$

- Therefore

$$\begin{aligned} & \text{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})\|p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})) \\ &= \frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)} \|\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t) - \boldsymbol{\epsilon}_t\|^2 + \text{const} \\ &= \frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)} \|\mathbf{g}(\sqrt{\alpha_t}\mathbf{x} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t, \mathbf{w}, t) - \boldsymbol{\epsilon}_t\|^2 + \text{const} \end{aligned}$$

Basically, we are now just predicting the noise $\boldsymbol{\epsilon}_t$ using the neural network $\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t)$

Predicting the noise

- We basically had the following

$$\text{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})\|p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})) = \frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)} \|\mathbf{g}(\sqrt{\alpha_t}\mathbf{x} + \sqrt{1-\alpha_t}\epsilon_t, \mathbf{w}, t) - \epsilon_t\|^2 + \text{const}$$

- The reconstruction error part in the ELBO can also be written as noise prediction

$$\ln p(\mathbf{x}|\mathbf{z}_1, \mathbf{w}) = -\frac{1}{2\beta_1} \|\mathbf{x} - \boldsymbol{\mu}(\mathbf{z}_1, \mathbf{w}, 1)\|^2 + \text{const.} = -\frac{1}{2(1-\beta_t)} \|\mathbf{g}(\mathbf{z}_1, \mathbf{w}, 1) - \epsilon_1\|^2 + \text{const}$$

- Ignoring the constants in front of the squared error terms above, the ELBO becomes

Empirically found to give improved performance

Pick an example \mathbf{x} randomly, generate a corruption \mathbf{z}_t by sampling ϵ_t and make a gradient based update to \mathbf{w}

$$\mathcal{L}(\mathbf{w}) = \underbrace{\int q(\mathbf{z}_1|\mathbf{x}) \ln p(\mathbf{x}|\mathbf{z}_1, \mathbf{w}) d\mathbf{z}_1}_{\text{reconstruction term}} - \underbrace{\sum_{t=2}^T \int \text{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})\|p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})) q(\mathbf{z}_t|\mathbf{x}) d\mathbf{z}_t}_{\text{consistency terms}}$$

$$= - \sum_{t=1}^T \|\mathbf{g}(\sqrt{\alpha_t}\mathbf{x} + \sqrt{1-\alpha_t}\epsilon_t, \mathbf{w}, t) - \epsilon_t\|^2$$

Can optimize using stochastic optimization

Denoising Diffusion Model: The Training Algo

- The overall training algo is as follows

Input: Training data $\mathcal{D} = \{\mathbf{x}_n\}$
 Noise schedule $\{\beta_1, \dots, \beta_T\}$

Output: Network parameters \mathbf{w}

```

for  $t \in \{1, \dots, T\}$  do
|    $\alpha_t \leftarrow \prod_{\tau=1}^t (1 - \beta_\tau)$  // Calculate alphas from betas
end for
repeat
|    $\mathbf{x} \sim \mathcal{D}$  // Sample a data point
|    $t \sim \{1, \dots, T\}$  // Sample a point along the Markov chain
|    $\epsilon \sim \mathcal{N}(\epsilon | \mathbf{0}, \mathbf{I})$  // Sample a noise vector
|    $\mathbf{z}_t \leftarrow \sqrt{\alpha_t} \mathbf{x} + \sqrt{1 - \alpha_t} \epsilon$  // Evaluate noisy latent variable
|    $\mathcal{L}(\mathbf{w}) \leftarrow \|\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t) - \epsilon\|^2$  // Compute loss term
|   Take optimizer step
until converged
return  $\mathbf{w}$ 

```

Denoising Diffusion Model: Generation

- Using the training model, we can now generate data as follows

Input: Trained denoising network $\mathbf{g}(\mathbf{z}, \mathbf{w}, t)$

Noise schedule $\{\beta_1, \dots, \beta_T\}$

Output: Sample vector \mathbf{x} in data space

$\mathbf{z}_T \sim \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$ // Sample from final latent space

for $t \in T, \dots, 2$ **do**

$\alpha_t \leftarrow \prod_{\tau=1}^t (1 - \beta_\tau)$ // Calculate alpha

// Evaluate network output

$\mu(\mathbf{z}_t, \mathbf{w}, t) \leftarrow \frac{1}{\sqrt{1-\beta_t}} \left\{ \mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \mathbf{g}(\mathbf{z}_t, \mathbf{w}, t) \right\}$

$\epsilon \sim \mathcal{N}(\epsilon|\mathbf{0}, \mathbf{I})$ // Sample a noise vector

$\mathbf{z}_{t-1} \leftarrow \mu(\mathbf{z}_t, \mathbf{w}, t) + \sqrt{\beta_t} \epsilon$ // Add scaled noise

end for

$\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}} \left\{ \mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}} \mathbf{g}(\mathbf{z}_1, \mathbf{w}, t) \right\}$ // Final denoising step

return \mathbf{x}

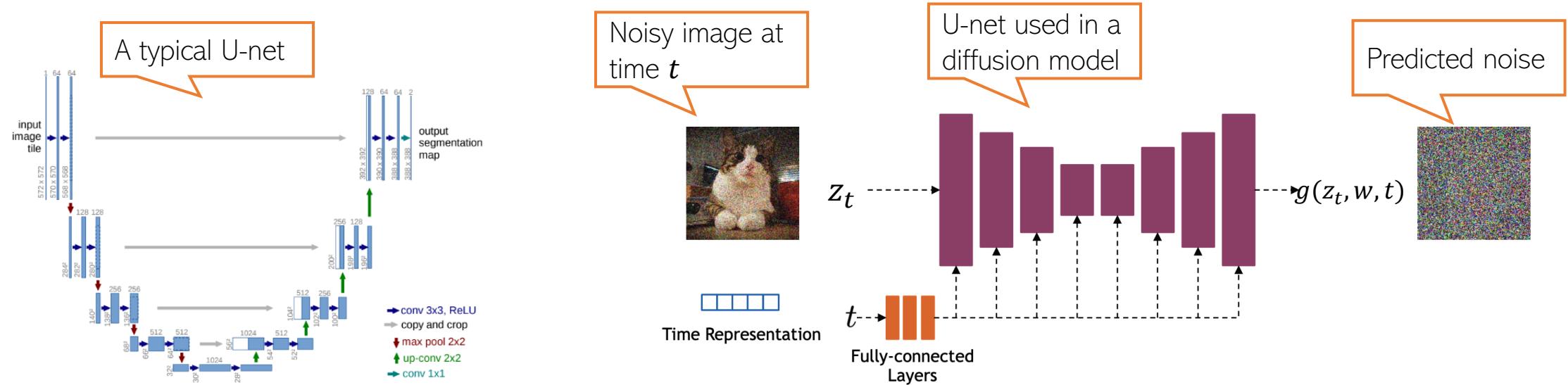
Generation can be slow because it requires several steps

Reducing the number of steps is an active area of research

One such approach is DDIM (denoising diffusion implicit model) which relaxes the Markov assumption in the noise process

Noise Predictor Network

- A “U-net” model (a neural net) is commonly used as the noise predictor network



- An embedding (positional embedding) of the time-step t is fed into the residual blocks of the U-net architecture

How Diffusion Models avoid mode collapse?

- **No adversarial training:** Instead of a competition between generator and discriminator, DDPMs learn directly from data using a well-defined likelihood-based loss function (Mean Squared Error on noise).
- **Denoising is a local correction task:** Each step in diffusion only removes small noise distortions, rather than needing to create an entire image at once.
- **All training samples contribute equally:** GANs rely on a discriminator, which can sometimes ignore certain parts of the data distribution, leading to biased learning. DDPMs, however, learn a direct mapping from noise to image using all training samples.



BigGAN-deep



Denoising diffusion implicit models



Training images from dataset

Dhariwal, P., & Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34, 8780-8794.

Feature	GANs	DDPMs
Mode Collapse?	Yes (generator may produce only a few types of images)	No (learns from full dataset)
Training Stability?	Unstable (adversarial training, hard to balance)	Stable (well-defined loss function)
Loss Function	Hard (min-max game between generator & discriminator)	Simple (predict noise, minimize MSE)
Image Quality	High, but sometimes artifacts	High, often better than GANs
Diversity of Outputs	Limited (mode collapse risk)	High (samples from full data distribution)
Computational Cost	Faster (1-step generation)	Slower (many diffusion steps needed)

Conditional Image Generation: Guided Diffusion

- Incorporate image embeddings into the diffusion in order to "guide" the generation
- Conditioning a prior data distribution $p(\mathbf{x})$ with a condition y to give $p(\mathbf{x}|y)$
 - y is a class label or an image/text embedding
- Unconditional generation reverse process

$$p_{\theta}(\mathbf{x}_{0:T}) = p_{\theta}(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1} | x_t)$$

- Conditional generation reverse process

$$p_{\theta}(\mathbf{x}_{0:T}|y) = p_{\theta}(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1} | x_t, y)$$

- We add conditioning information y at each diffusion step

Guided Diffusion Models

- For any time step t , Guided diffusion models aims to learn

$$\nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t \mid y)$$

- Using Bayes rule

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t \mid y) &= \nabla_{\mathbf{x}_t} \log \left(\frac{p_{\theta}(y \mid \mathbf{x}_t) p_{\theta}(\mathbf{x}_t)}{p_{\theta}(y)} \right) \\ &= \nabla_{\mathbf{x}_t} \log p_{\theta}(y \mid \mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t)\end{aligned}$$

$$\boxed{\nabla_{\mathbf{x}_t} \log p_{\theta}(y) = 0}$$

- Adding scalar guidance s (influence)

$$\nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t \mid y) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t \mid y) + s \cdot \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t)$$

$$p_{\theta}(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \boldsymbol{\mu}_{\theta}(\mathbf{x}_t, t), \Sigma_{\theta}^2(x_t, t))$$

Classifier guidance

- Use another classifier to guide the diffusion toward the target class y
- Train a classifier $f_{\phi}(y \mid \mathbf{x}_t, t)$ on noisy image \mathbf{x}_t to predict its class y
- Use the gradients $\nabla_{\mathbf{x}_t} \log(f_{\phi}(y \mid \mathbf{x}_t))$ to guide the diffusion
- Gradient tells us **how to modify the image \mathbf{x}_t** to increase the likelihood of class y
- Modify predicted mean $\mu_{\theta}(\mathbf{x}_t, t)$ using the classifier gradient

$$\hat{\mu}_{\theta}(\mathbf{x}_t, t, y) = \mu_{\theta}(\mathbf{x}_t, t) + s \cdot \Sigma_{\theta}(\mathbf{x}_t, t) \nabla_{\mathbf{x}_t} \log(f_{\phi}(y \mid \mathbf{x}_t))$$

- s controls how much influence the classifier has on generation
- $\nabla_{\mathbf{x}_t} \log(f_{\phi}(y \mid \mathbf{x}_t))$ – classifier's gradient pushes \mathbf{x}_t towards more probable class-conditioned images.

Why modify input (∇_x) instead of weights (∇_θ)?

Preserves the Pretrained Diffusion Model

- ∇_x modifies the **sample** directly rather than updating the model weights
- ∇_θ implies we are training the diffusion model again
- Classifier's influence is **only added during inference**, not training
- Diffusion model remains **unchanged**, so we don't need to retrain it for each class

More Flexible Control Over Generation

- ∇_x pushes the generated sample towards a particular class during inference
- By changing the classifier, we can **guide the model toward different classes** without modifying the diffusion model

Classifier guidance

- Low s (e.g., 1.0-2.0)
 - Model generates diverse samples but may not strongly follow the class conditioning
- High s (e.g., 5.0+)
 - Model strongly adheres to the class label but may reduce diversity and introduce artifacts.



classifier scale 1.0



classifier scale 10.0

Text embedding as guidance



“a hedgehog using a calculator”



“a corgi wearing a red bowtie and a purple party hat”



“robots meditating in a vipassana retreat”



“a fall landscape with a small cottage next to a lake”



“a surrealist dream-like oil painting by salvador dali of a cat playing checkers”



“a professional photo of a sunset behind the grand canyon”



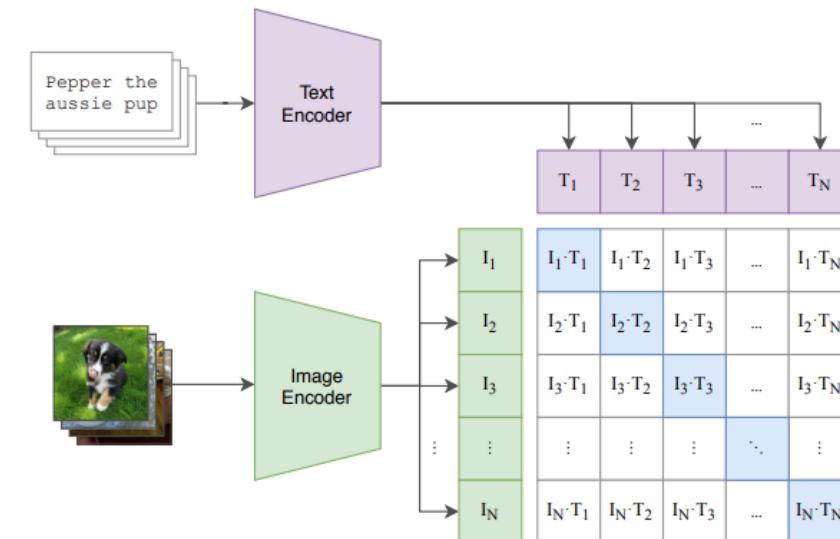
“a high-quality oil painting of a psychedelic hamster dragon”



“an illustration of albert einstein wearing a superhero costume”

Text embedding as guidance

- Use text embedding to guide the diffusion
- CLIP (Contrastive Language Image Pretraining) Model for embedding
 - A pre-trained model for telling you how well a given *text* and a given *image* fit together
- CLIP generates
 - Text embedding - $h(c)$
 - Image embedding – $g(\mathbf{x}_t)$



Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... & Sutskever, I. (2021, July). Learning transferable visual models from natural language supervision. In *International conference on machine learning* (pp. 8748-8763). PMLR.

Nichol, A. Q., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., ... & Chen, M. (2022, June). GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. In *International Conference on Machine Learning* (pp. 16784-16804). PMLR.

Text embedding as guidance

$$\hat{\mu}(\mathbf{x}_t, t, y) = \mu_\theta(\mathbf{x}_t, t) + s \cdot \Sigma_\theta(\mathbf{x}_t, t) \nabla_{\mathbf{x}_t} \log(f_\phi(y \mid \mathbf{x}_t))$$


$$\hat{\mu}(\mathbf{x}_t, t, c) = \mu_\theta(\mathbf{x}_t, t) + s \cdot \Sigma_\theta(\mathbf{x}_t, t) \nabla_{\mathbf{x}_t} \text{sim}(h(c), g(\mathbf{x}_t))$$

$\nabla_{\mathbf{x}_t} \text{sim}(h(c), g(\mathbf{x}_t))$ -- gradient of CLIP similarity between text and image embeddings

Intuition

- If $g(\mathbf{x}_t)$ and $h(c)$ are not aligned, gradient $\nabla_{\mathbf{x}_t} \text{sim}(h(c), g(\mathbf{x}_t))$ nudges \mathbf{x}_t in a direction that increases similarity
- s adjusts how aggressively similarity adjustment is applied
- Using CLIP's semantic understanding instead of a separate classification model

Image inpainting with text embedding guidance



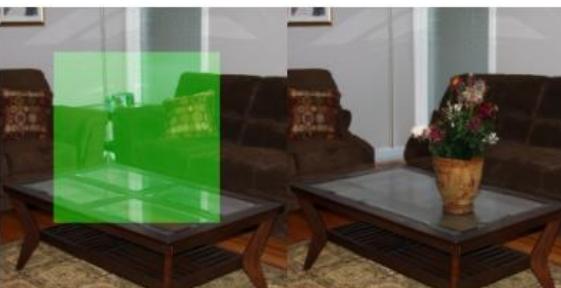
"zebras roaming in the field"



"a girl hugging a corgi on a pedestal"



"a man with red hair"



"a vase of flowers"



"an old car in a snowy forest"



"a man wearing a white hat"

Green region is erased, and the model fills it in conditioned on the given prompt.

Classifier-free guidance

No external classifier

- Train diffusion model in two modes
 - Unconditional mode
 - Ignoring conditioning by setting $y = \emptyset$
 - with probability $\mathbf{p}_{dropout}$
 - Model predicts $\epsilon_{\theta}(\mathbf{x}_t, t)$
 - Conditional mode
 - Using conditioning, e.g., a text prompt or class label
 - Model predicts $\epsilon_{\theta}(\mathbf{x}_t, t, y)$
- At inference, noise is prediction is modified
$$\tilde{\epsilon}_{\theta} = w \cdot \epsilon_{\theta}(\mathbf{x}_t, t, y) + (1 - w)\epsilon_{\theta}(\mathbf{x}_t, t)$$

Why Classifier-Free Guidance Works?

Balancing Control and Diversity

- Low w (e.g., 1-3)
 - More diverse generations, but weaker conditioning (image may not match the text prompt exactly).
- Medium w (e.g., 5-7.5)
 - Good balance between **adherence to condition** and **image diversity**
- High w (e.g., 10+)
 - Very strong conditioning, but may introduce **artifacts** or reduce diversity

More Stable than Classifier Guidance

- No need for a separate classifier, reducing complexity
- No reliance on noisy-image classifiers, avoiding unstable gradients

Why Classifier-Free Guidance Works?

More interpretable control, as w directly adjusts generation adherence

- Computationally Efficient
- Uses same neural network for both conditional and unconditional predictions
- Works seamlessly for text-to-image, class-conditional, and other conditional generation tasks



A wall in a royal castle. There are two paintings on the wall. The one on the left a detailed oil painting of the royal raccoon king. The one on the right a detailed oil painting of the royal raccoon queen.

A group of teddy bears in suit in a corporate office are celebrating the birthday of their friend.

A chrome-plated duck with a golden beak arguing with an angry turtle in a forest.



A squirrel is inside a giant bright shiny crystal ball in the sky. There are few clouds on the surface of blue ocean.



A bald eagle made of chocolate powder, mango, and whipped cream.



A marble statue of a Koala DJ in front of a marble statue of a turntable. The Koala is wearing large marble headphones.

Scaling up diffusion models: Issues

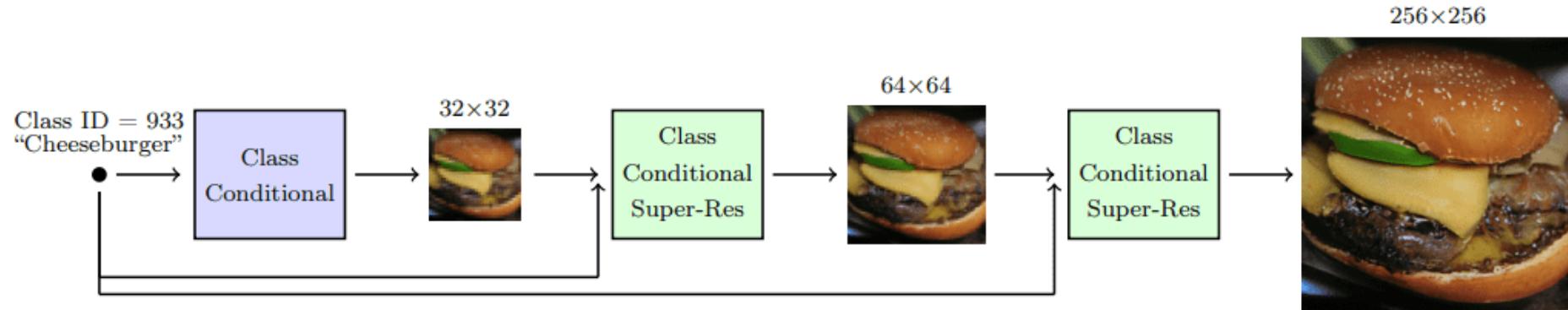
Expensive Training

- Large diffusion models require massive compute resources (e.g., Imagen, Stable Diffusion, DALL·E 2)
- Training involves thousands of forward and reverse passes over large datasets, making it computationally expensive compared to GANs
- Scaling up resolution (e.g., 1024×1024 vs. 256×256) significantly increases memory and FLOPs

Slow Inference (Many Sampling Steps)

- Diffusion models require hundreds or thousands of denoising steps to generate high-quality images.
- Sampling is slow compared to GANs, which generate an image in a single pass.

Cascaded Diffusion Model



- Base Diffusion Model
 - Trained to generate a **low-resolution image** (e.g., 32×32).
 - Captures **global structure** but lacks fine details
- First Super-Resolution Model
 - Upscales the image from 32×32 to 64×64
 - **Conditional diffusion model** to refine textures conditioned on the low-res image to **preserve content**
$$p_{\theta}(x_{t-1} | x_t, x_{low-res})$$
- Second Super-Resolution Model
 - Further **upsamples** from 64×64 to 256×256

Cascaded Diffusion Model (CDM)

- CDMs use classifier-free guidance to improve the quality of super-resolution

$$\tilde{\epsilon}_\theta = w \cdot \epsilon_\theta(\mathbf{x}_t, t, \mathbf{x}_{low-res}) + (1 - w)\epsilon_\theta(\mathbf{x}_t, t)$$

- w is the **guidance scale** controlling adherence to the lower-resolution image
- CDMs can integrate **text embeddings** (e.g., from CLIP or T5) at each stage

To train super-resolution models

- Noise is added to high-resolution images before downsampling
- Model learns to denoise while restoring high-frequency details
- Prevents the model from simply memorizing bicubic upscaling

Latent diffusion models

Issue

Traditional DMs suppress semantically meaningless information but still need to evaluate gradient on all pixels

Aim

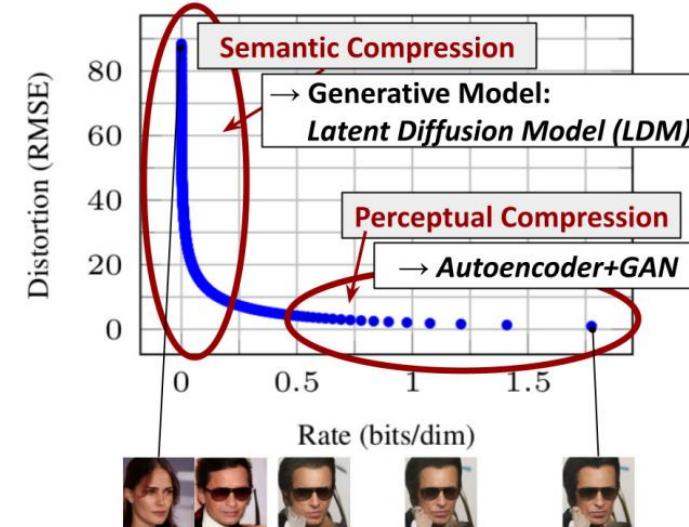
Find a perceptually equivalent, but computationally more suitable space

Perceptual Compression (Encoder)

- Compress the image into a lower-dimensional latent space, keeping only the important high-level features

Semantic Compression

- Diffusion Model stores features that affect human perception (sharpness, texture, colors)
- Decoder reconstructs a visually realistic image



Latent diffusion models (LDMs) are an effective generative model with a separate mild compression stage that only eliminates imperceptible details

Stable Diffusion: Latent diffusion models

- Encoder f_{enc} (CNN) that maps image to a latent vector

$$\mathbf{z}_0 = f_{enc}(\mathbf{x}_0)$$

- Forward process

$$q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t; \mu_t = \sqrt{1 - \beta_t} \mathbf{z}_{t-1}, \Sigma_t = \beta_t \mathbf{I})$$

- Reverse process

$$p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t) = \mathcal{N}(\mathbf{z}_{t-1}; \mu_\theta(\mathbf{z}_t, t), \Sigma_\theta^2(\mathbf{z}_t, t))$$

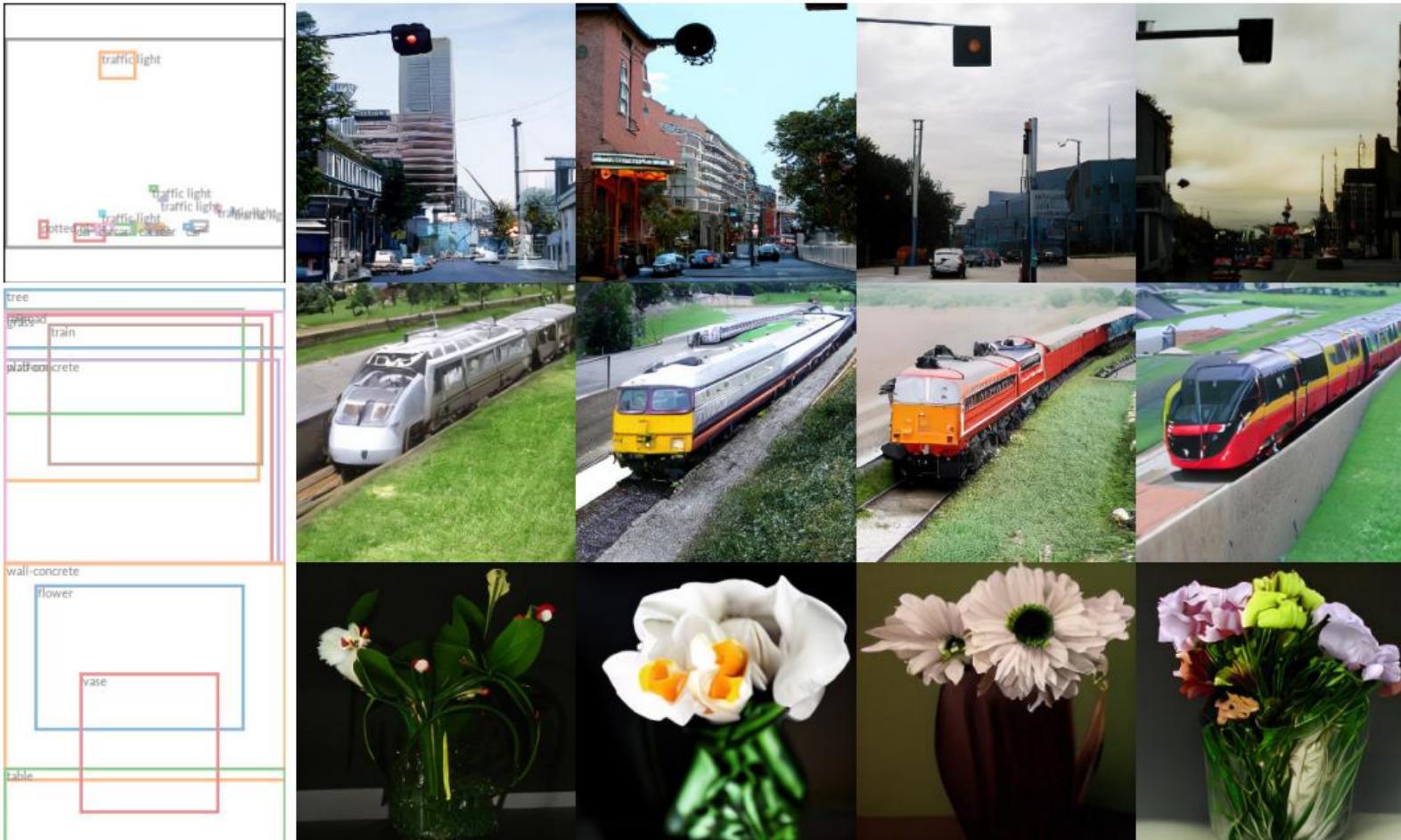
- Loss

$$L_{denoise} = \mathbb{E}_q[\|\epsilon - \epsilon_\theta(z_t, t)\|_2^2]$$

- Decoder (symmetric CNN to Encoder) reconstructs latent vector to image

$$\hat{\mathbf{x}}_0 = f_{dec}(\hat{\mathbf{z}}_0)$$

Layout to Synthesis with LDM



Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10684-10695).

Summary

- Diffusion Models (denoising diffusion models, score based models, etc) are currently the best performing methods
- A lot of ongoing work on diffusion models, e.g.,
 - Improving quality of generation
 - Speeding-up generation
 - Combining them with other generative models (e.g., large language models)

References

- Chapter 20, Christopher N. Bishop, Deep Learning
- Chapter 25, Kevin Murphy, [Probabilistic Machine Learning: Advanced Topics](#)