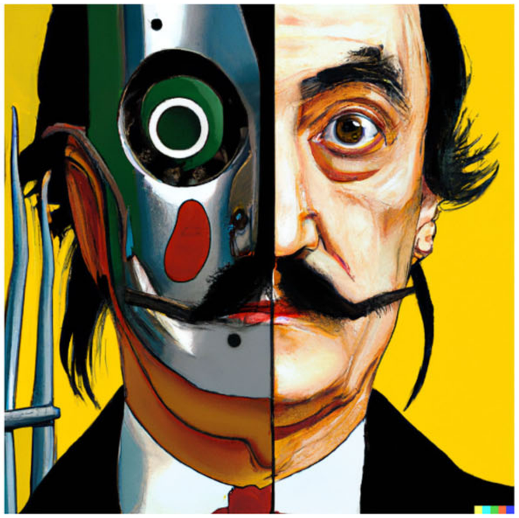
Diffusion Models



vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

an espresso machine that makes coffee from human souls, artstation

panda mad scientist mixing sparkling chemicals, artstation

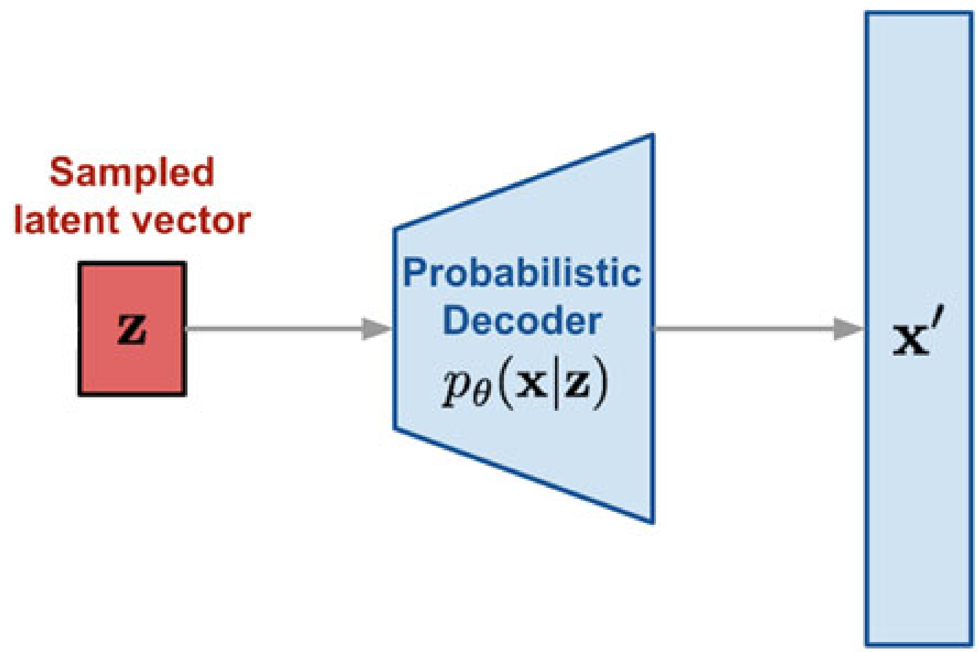
a corgi's head depicted as an explosion of a nebula

Dall-E 2: https://cdn.openai.com/papers/dall-e-2.pdf

Review: **VAEs**

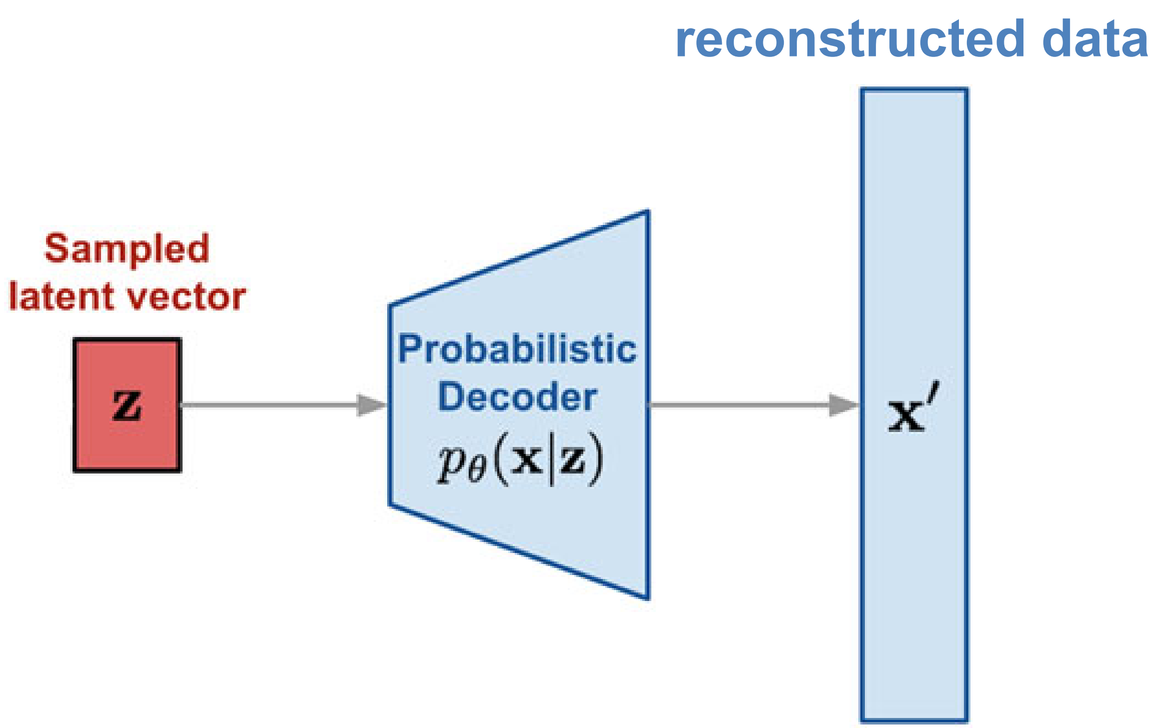
Explicit generative model

P(x, z) = P(z) P(x | z) , where **P(z) is a simple distribution** (unit Gaussian), and **P(x | z) is a neural network decoder**



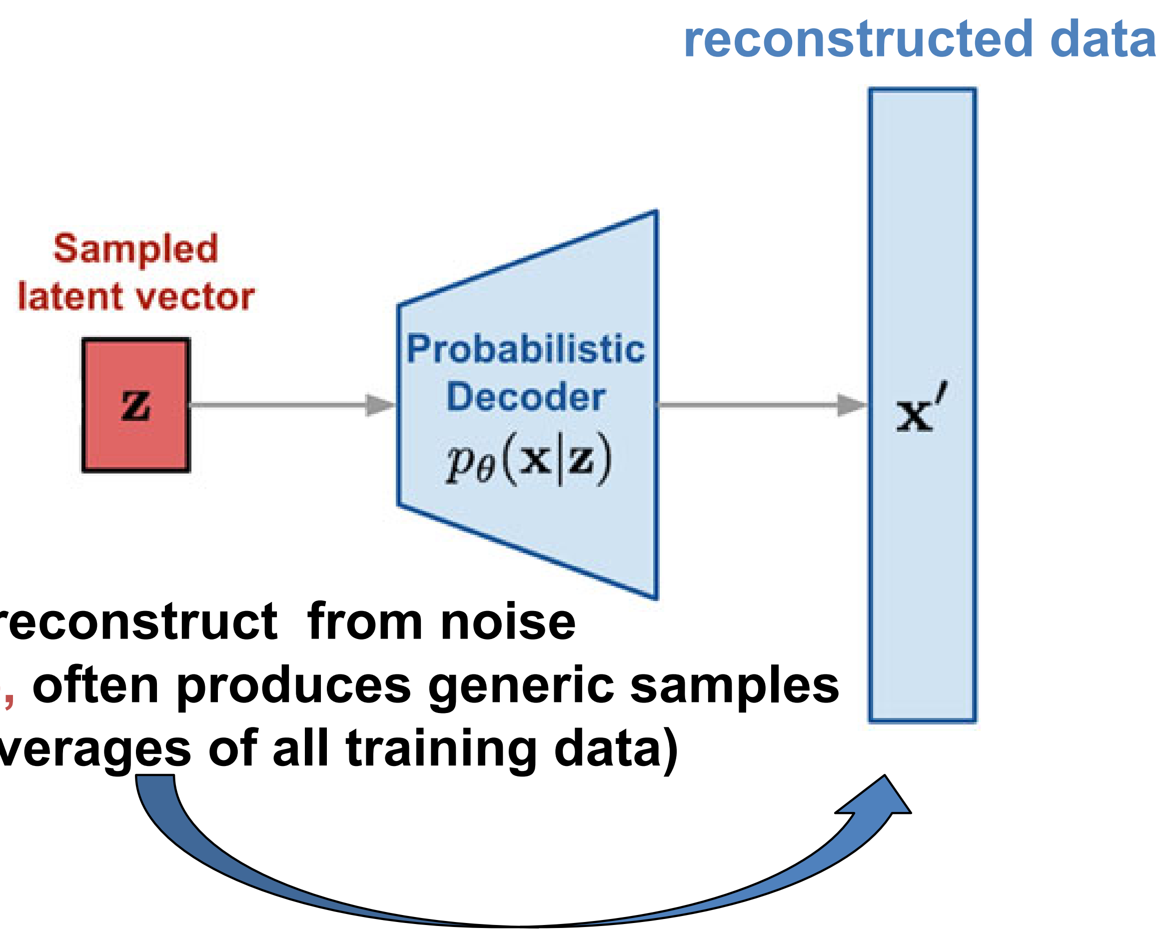
Review: **VAEs**

Many advantages e.g., fast sampling, effective compression of input data, yet poor quality in generated samples



Review: **VAEs**

Many advantages e.g., fast sampling, effective compression of input data, yet poor quality in generated samples



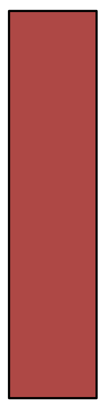
Generative models

(discussed in this course)

- Autoregressive models
- Variational Autoencoders
- **Diffusion Models**
- Generative Adversarial Networks [lower priority, after Easter]

Diffusion models

Follow a more gradual, multi-step reconstruction approach

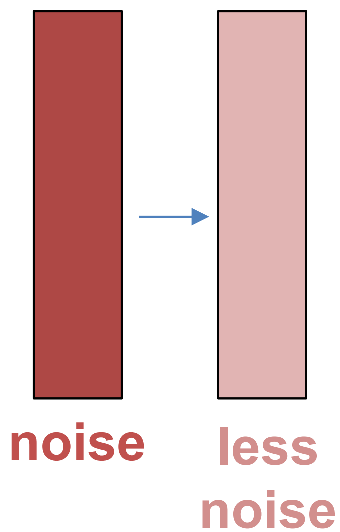


noise

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

Diffusion models

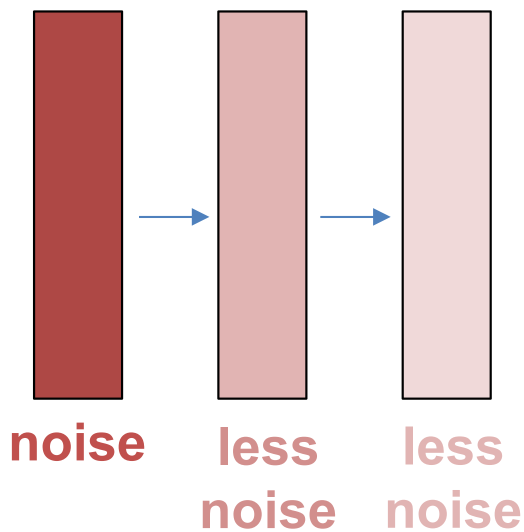
Follow a more gradual, multi-step reconstruction approach



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

Diffusion models

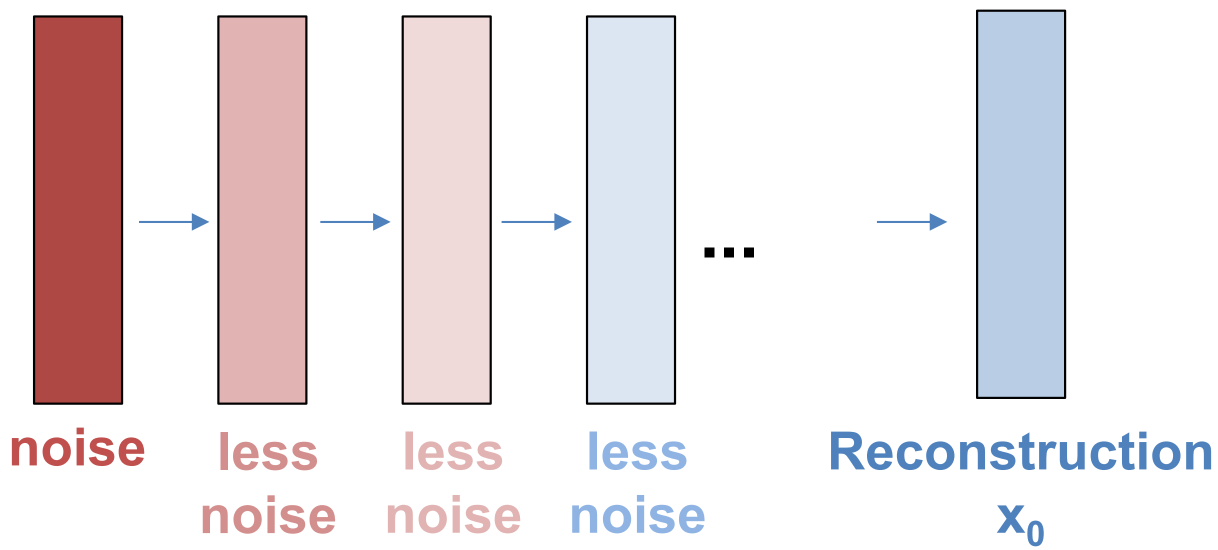
Follow a more gradual, multi-step reconstruction approach



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

Diffusion models

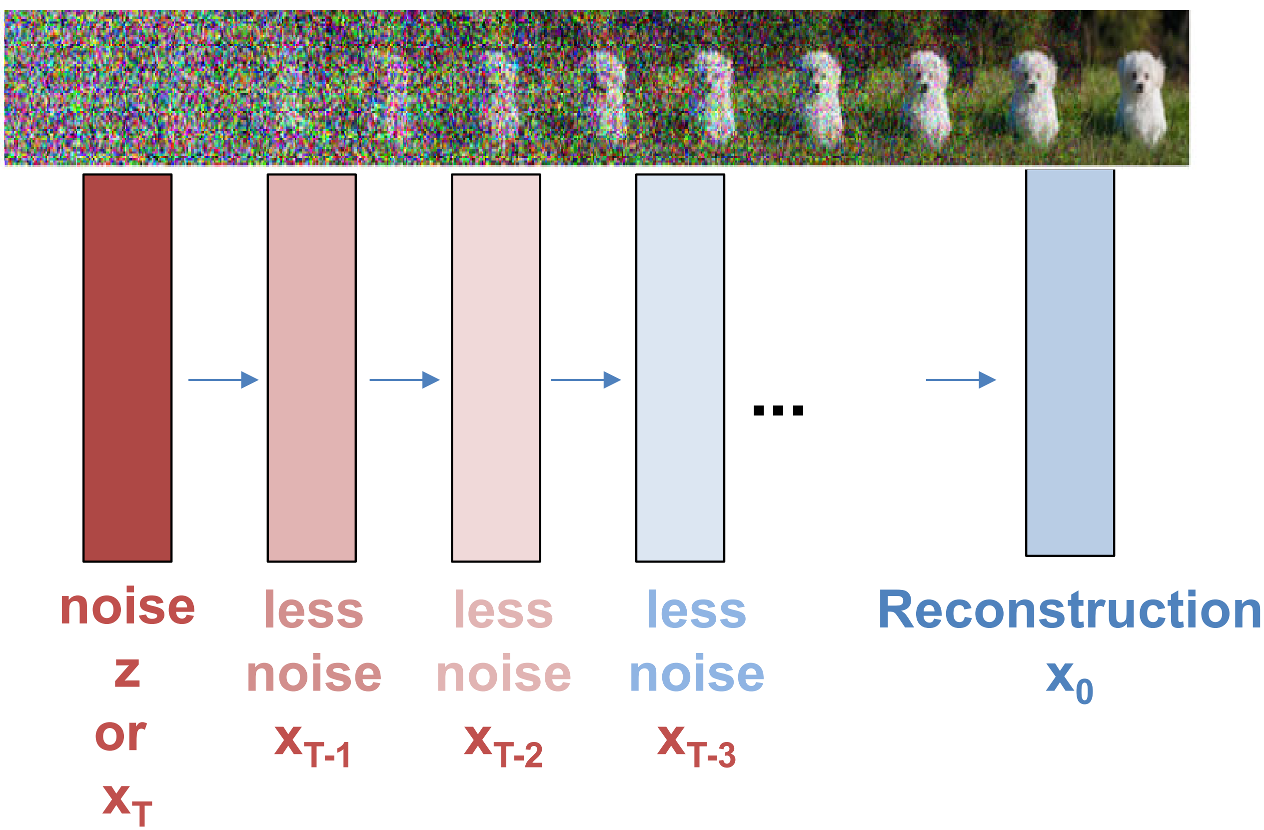
Follow a more gradual, multi-step reconstruction approach



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

Diffusion models

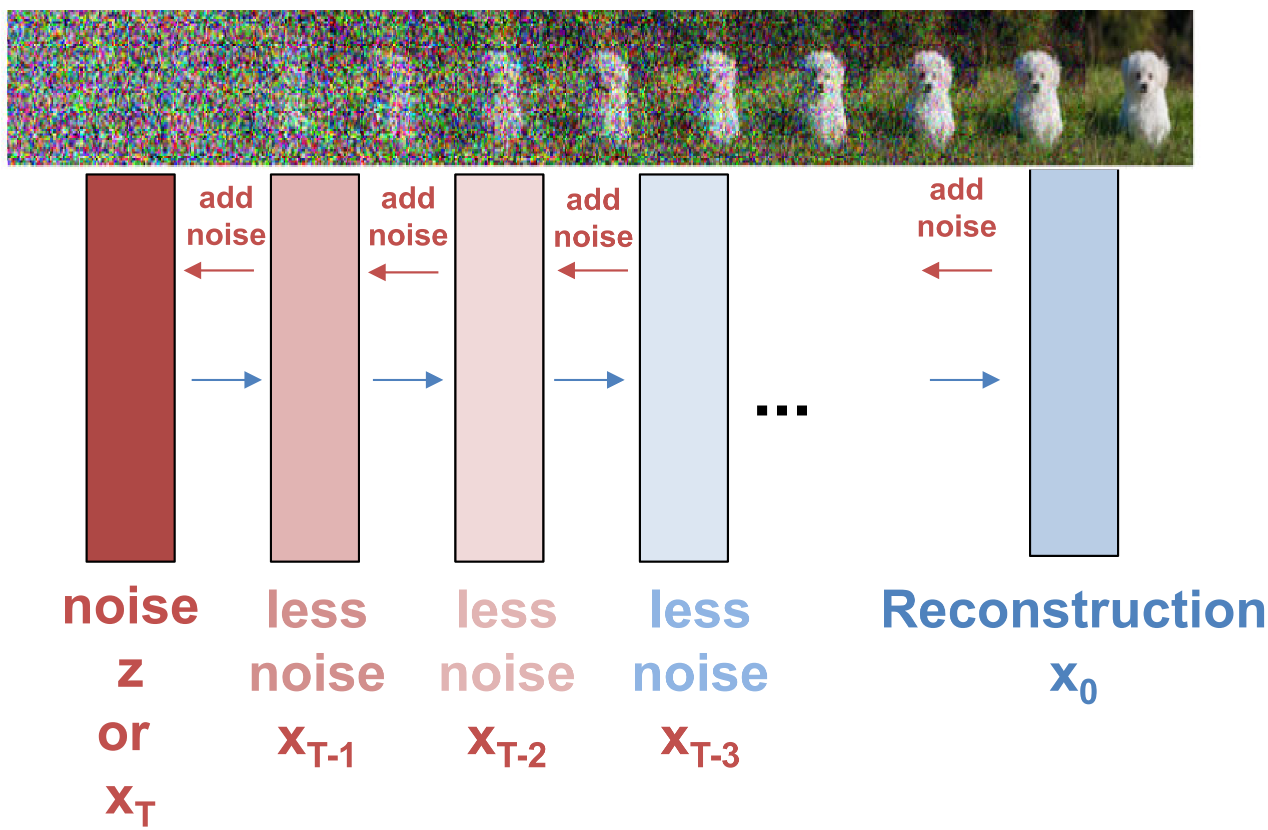
Follow a more gradual, multi-step reconstruction approach



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

Diffusion models - “Forward” process

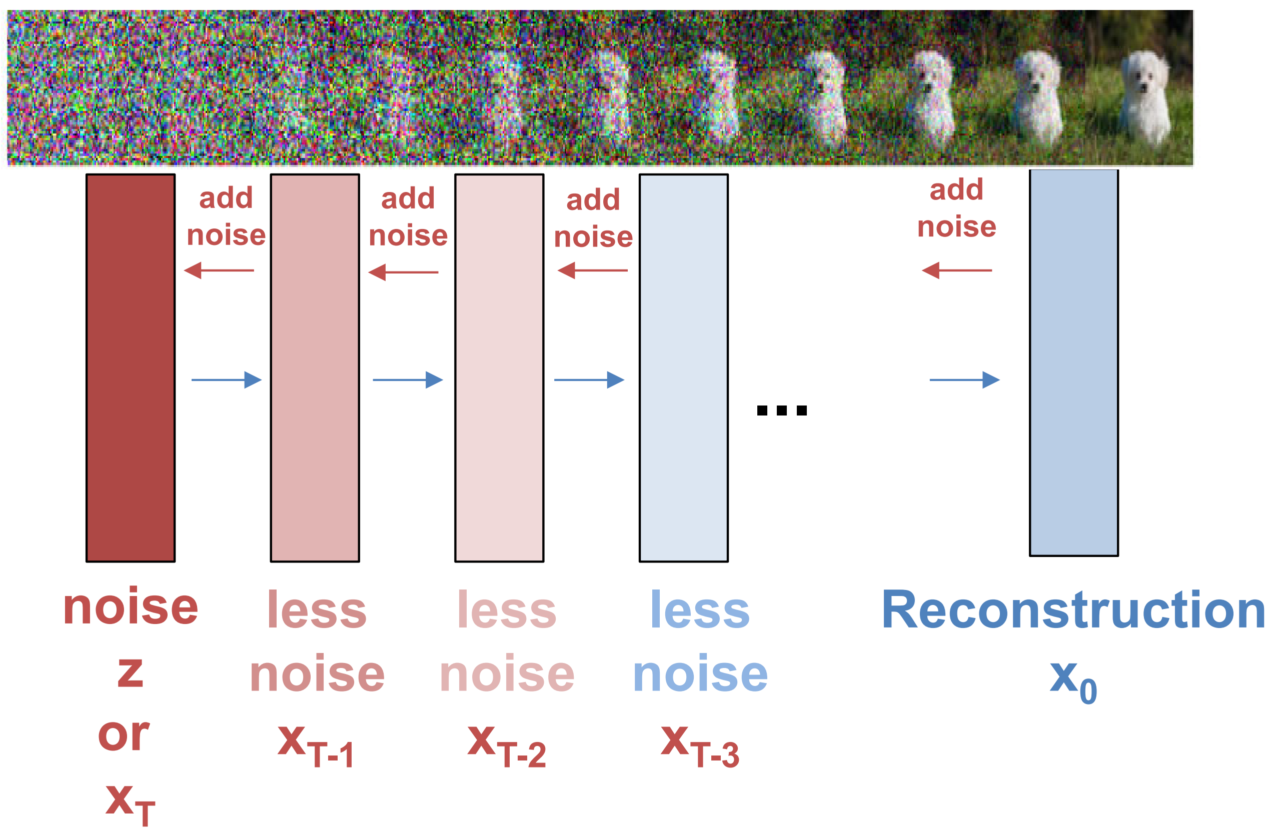
Let’s go from data x_0 to noise **gradually, step-by-step with a simple process: add standard Gaussian noise ε at each step**



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

Diffusion models - "Forward" process

$q(\, x_t \mid x_{t-1}\,) = \textbf{\textit{gaussian}}(\,\textit{previous image, some variance})$

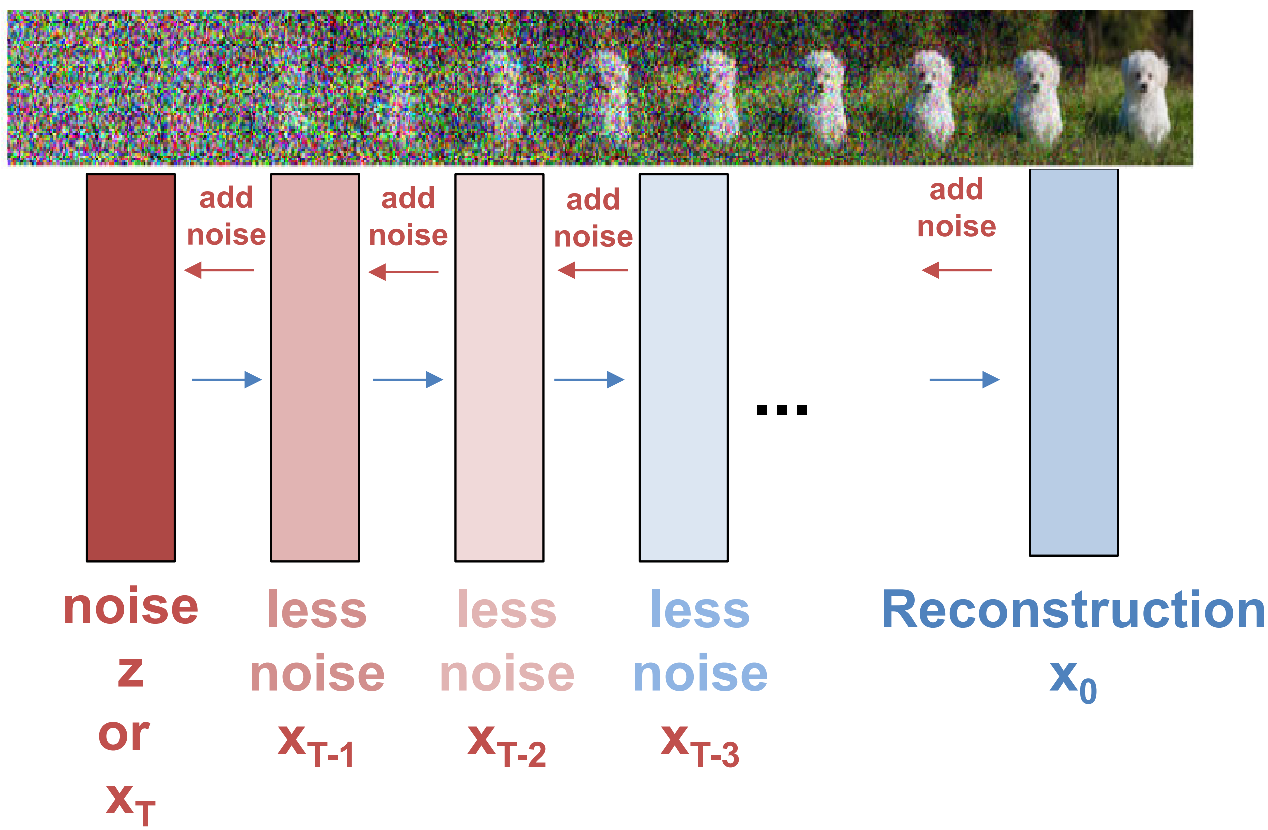


https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion models - "Forward" process

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = N(\mathbf{x}_{t-1}, \beta_t I)$$

(where I is the diagonal matrix, i.e., add noise with diagonal covariance scaled by β_t)

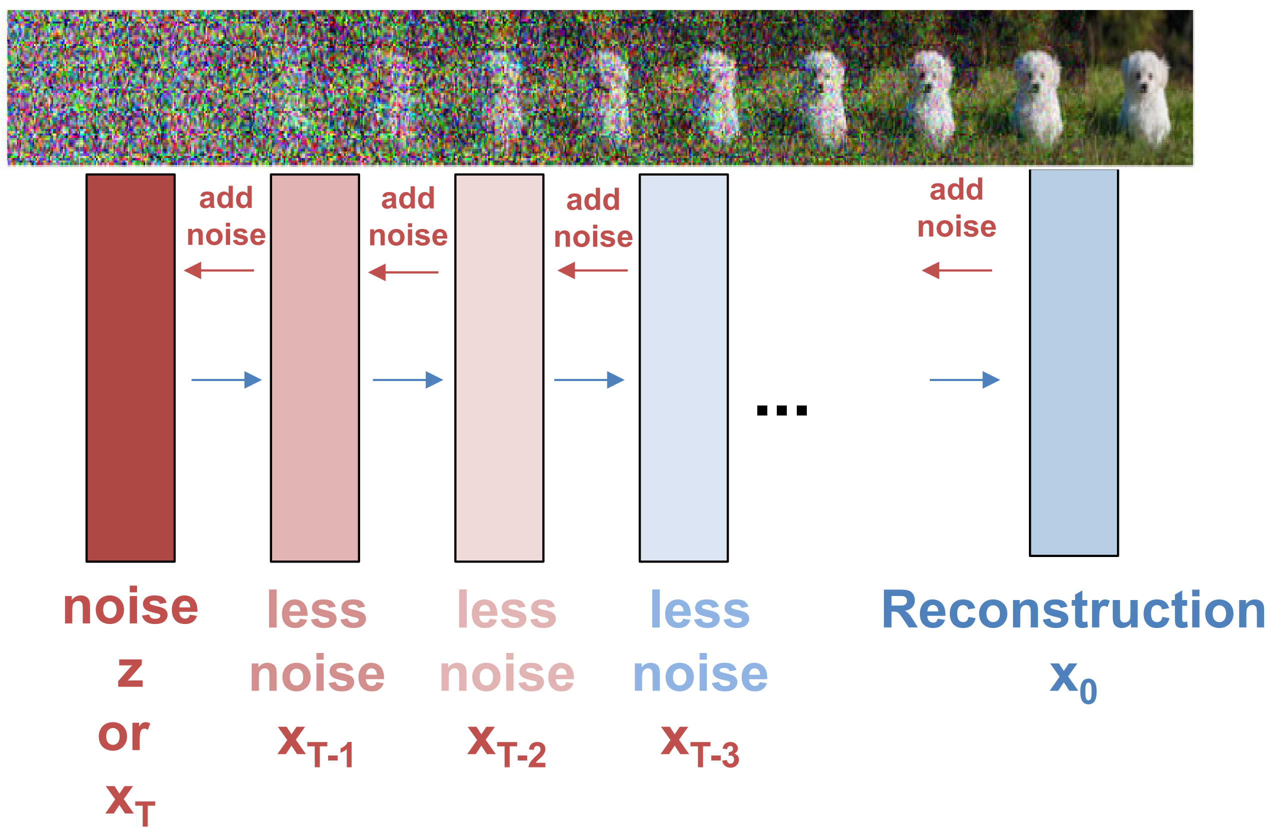


https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion models - “Forward” process

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = N(\sqrt{a_t}\,\mathbf{x}_{t-1}, \beta_t \boldsymbol{I})$$

Scale down input and set: $a_t = 1 - \beta_t$… **Why?**

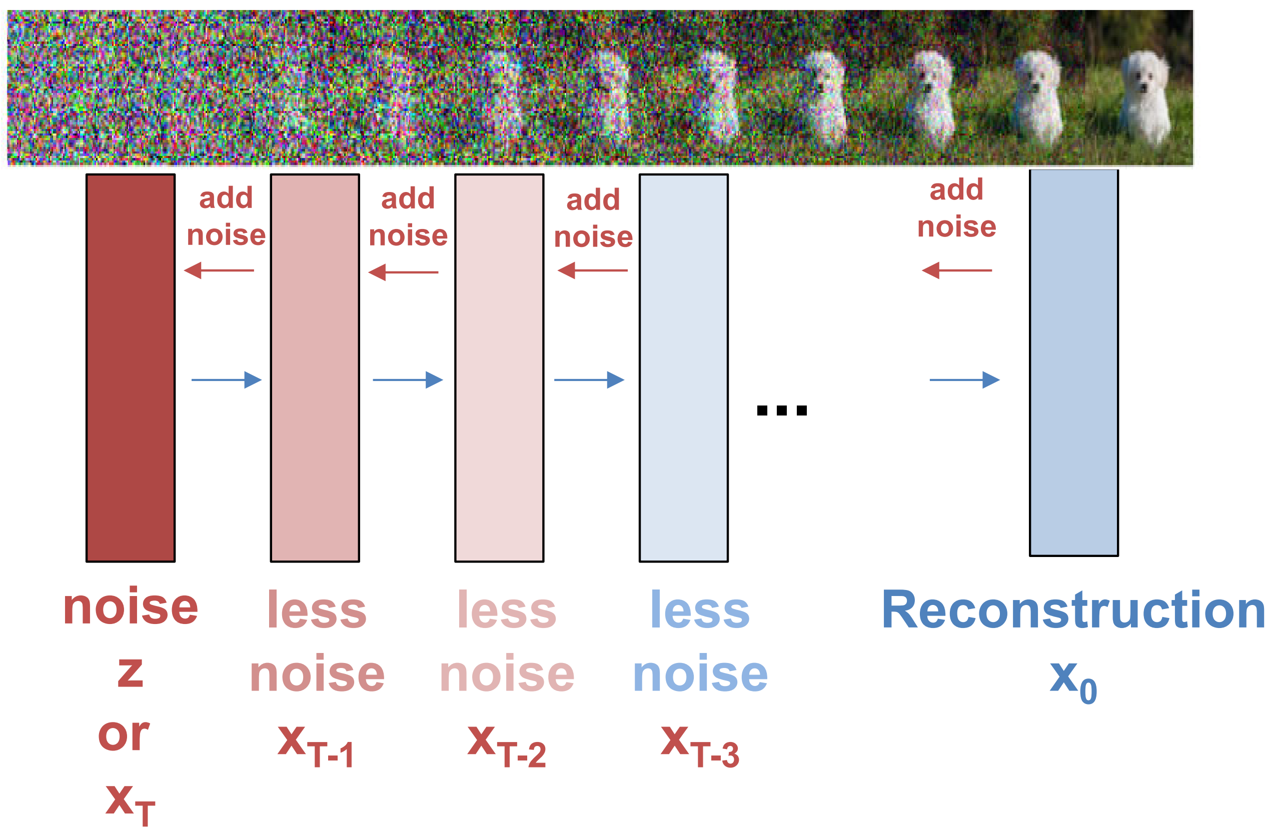


https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion models - “Forward” process

Because it can be shown that in the final step: $\mathbf{z} = \mathbf{x}_T \sim N(0, \mathbf{I})$

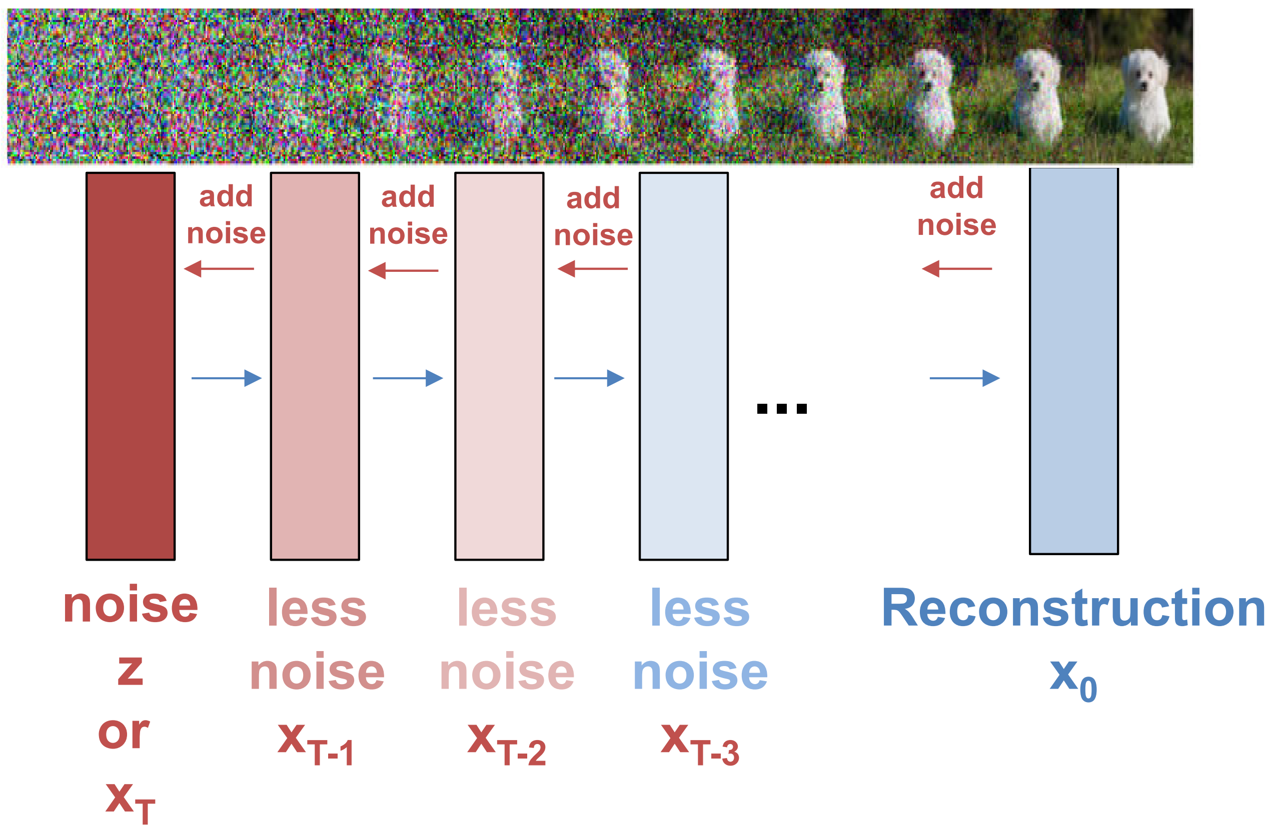
We destroyed the input making it unit Gaussian!



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

Diffusion models - “Reverse” process

We now need to a way to map noise back to the data!

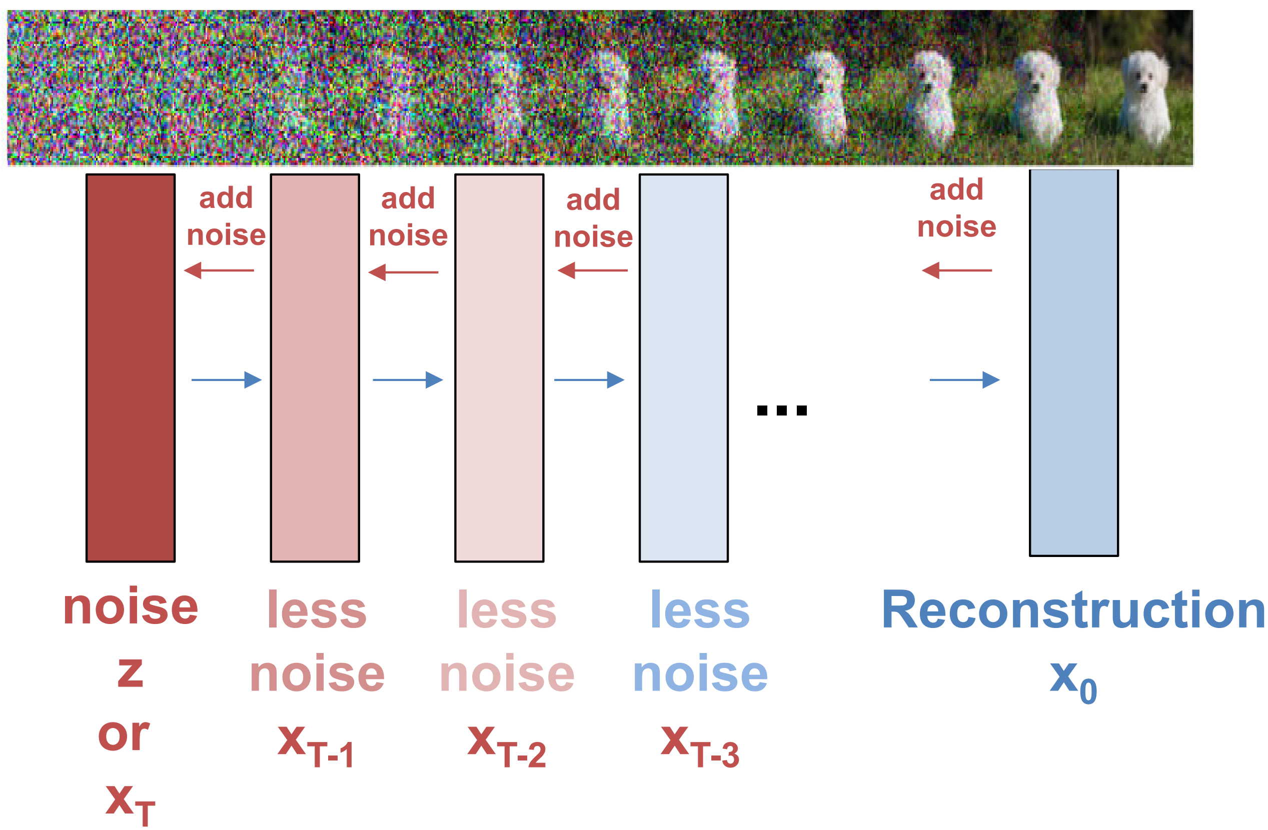


https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion models - “Reverse” process

Remember that the forward process was:

$q(x_t \mid x_{t-1}) = \textit{gaussian}(\textit{ previous image, some variance})$

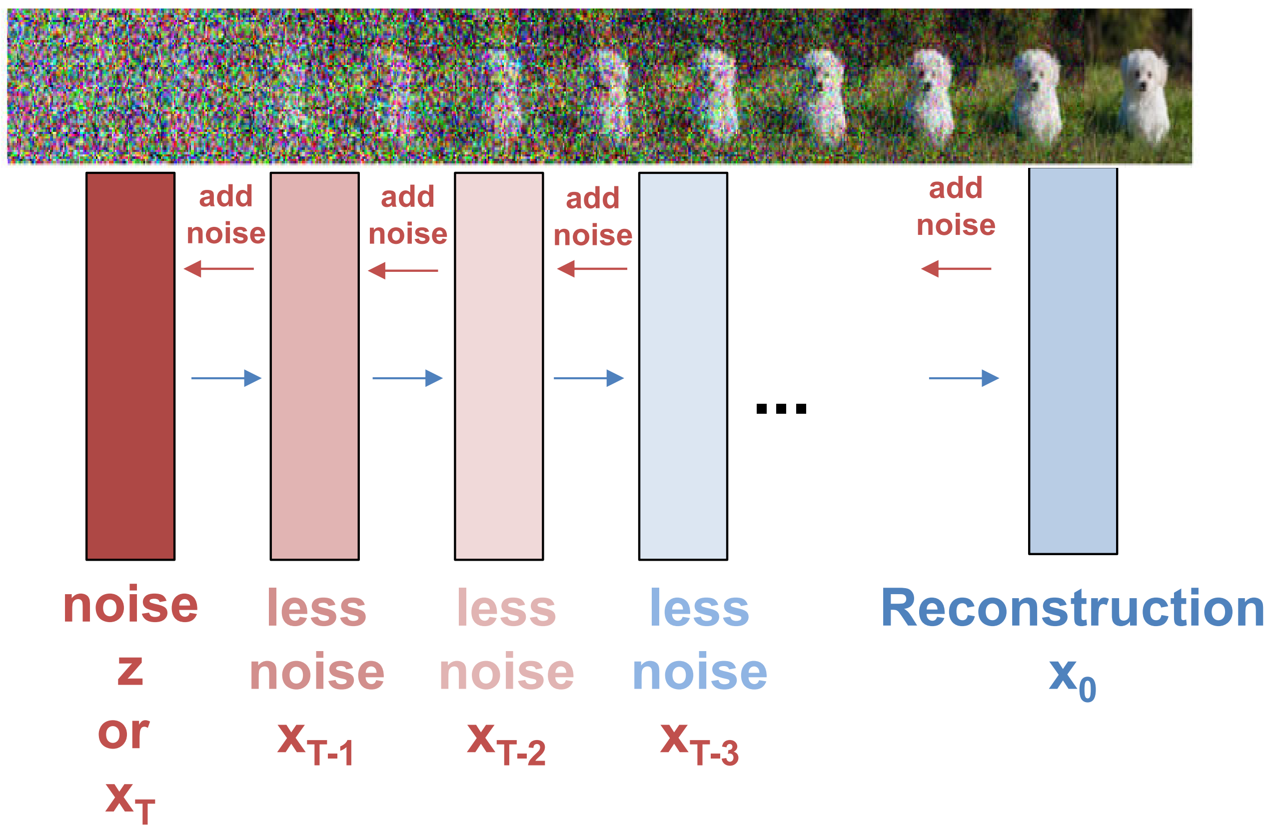


https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion models - “Reverse” process

Reverse the process? Complex... depends on entire dataset!

$q(x_{t-1} \mid x_t)$ = ***not a gaussian!***

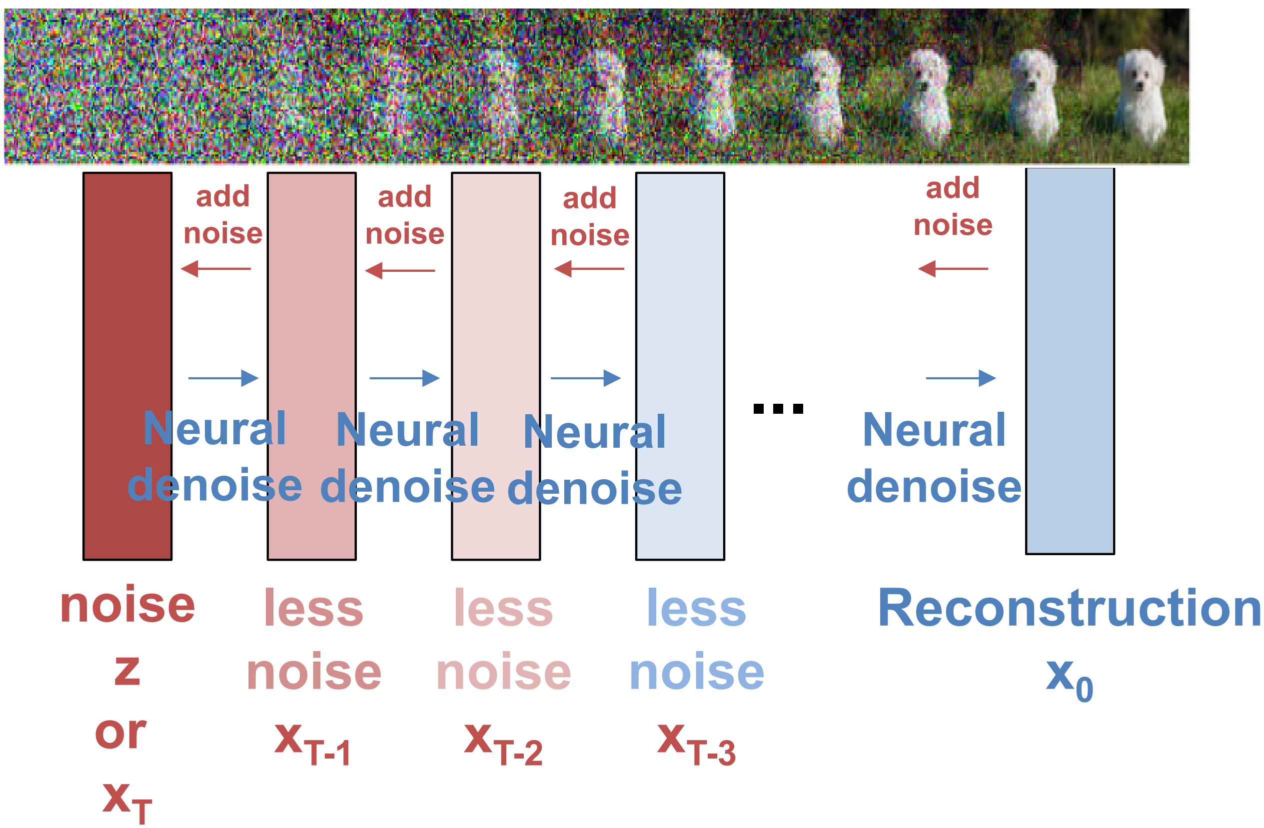


https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion models - "Reverse" process

Use a neural network to approximate it in each small step

$q(x_{t-1} \mid x_t) \approx \textit{gaussian(mean, variance)}$

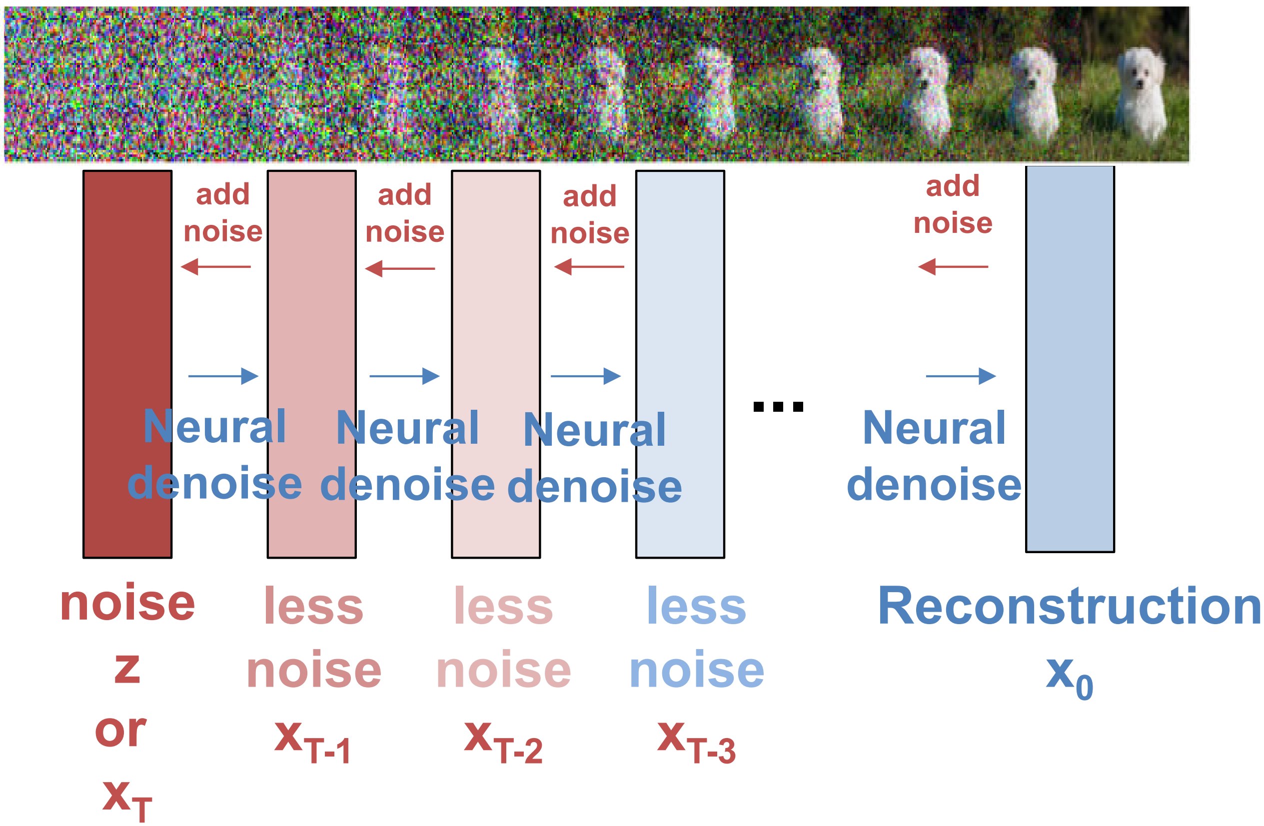


https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion models - "Reverse" process

Given current noisy version $\mathbf{x}_t$ and time $\mathbf{t}$, the network predicts mean & covariance based on learned parameters $\boldsymbol{\theta}$:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \approx N\left(\mu_\theta(\mathbf{x}_t, \mathbf{t}),\ \Sigma_\theta(\mathbf{x}_t, \mathbf{t})\right)$$

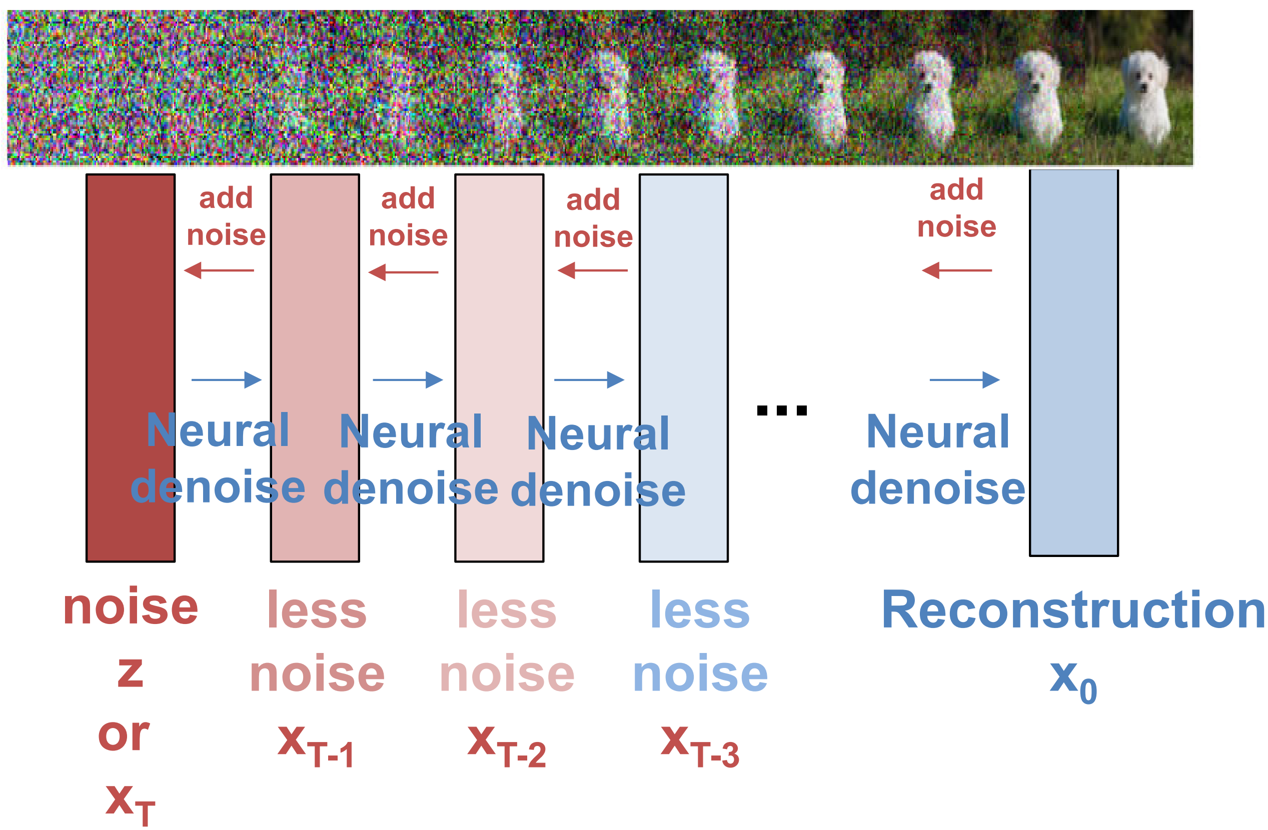


https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion models - “Reverse” process

Need to learn these parameters θ...

$$q(x_{t-1} \mid x_t) \approx N\left(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t) \right)$$

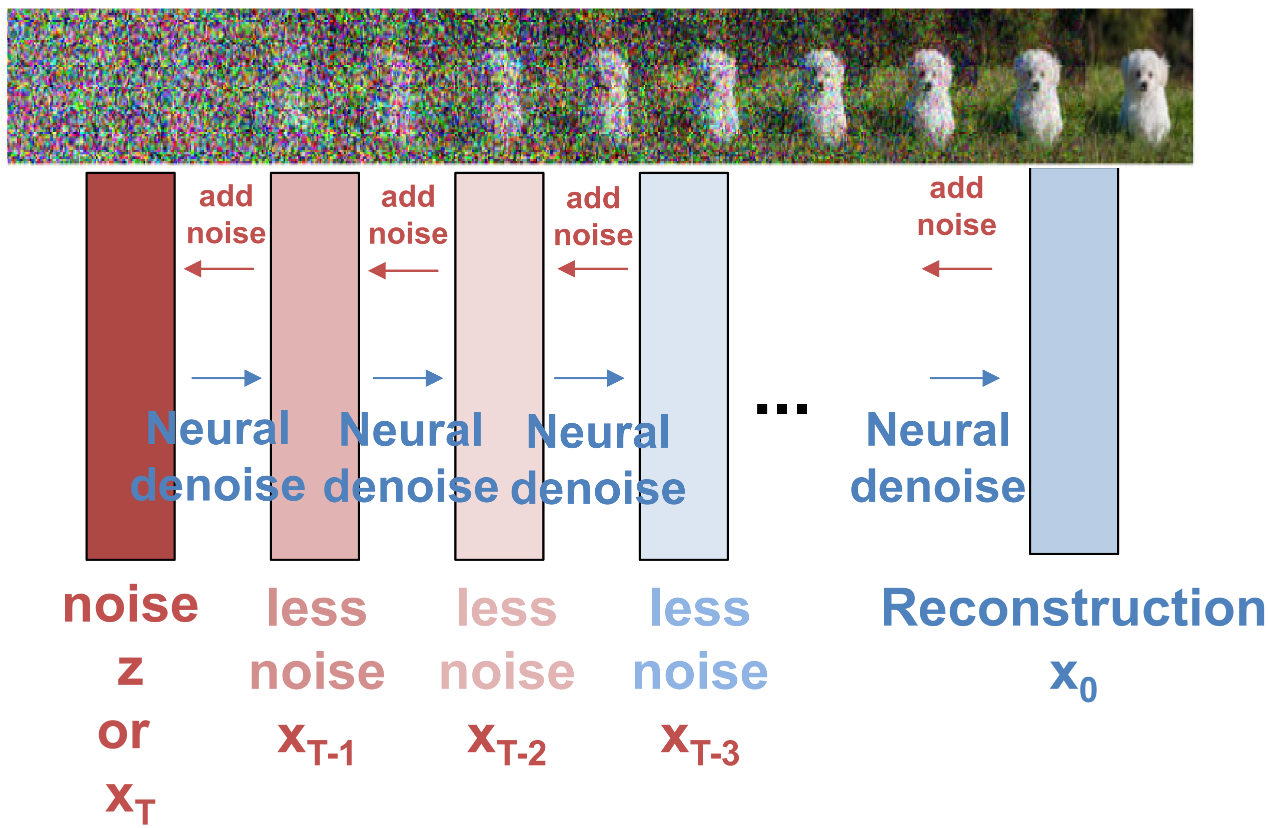


https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion models - “Reverse” process

During training, we observe $\mathbf{x_0}$ **(data to reconstruct)**

$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$

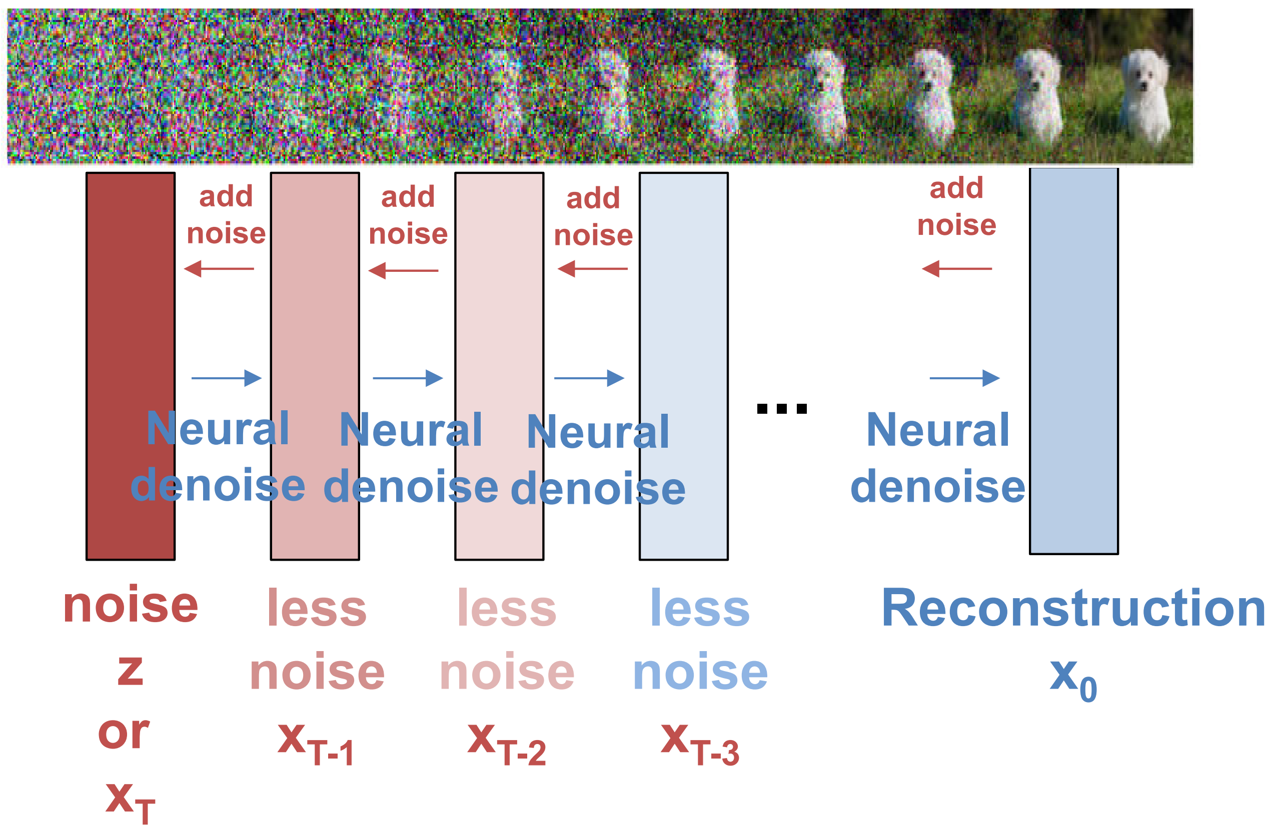


https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion models - "Reverse" process

During training, we observe x_0 **(data to reconstruct)**

$$q(x_{t-1} \mid x_t, x_0) = N(\tilde{\mu}_t, \tilde{\Sigma}_t)$$ <= *computable distribution*



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

Diffusion models - “Reverse” process

During training, we observe $\mathbf{x_0}$ **(data to reconstruct)**

$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = N(\tilde{\mu}_t, \tilde{\Sigma}_t)$ ***<= computable distribution***

Argh...

$$\tilde{\mu}_t = \left(\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t}\mathbf{x}_0\right)/\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_t}\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0\right.$$

where $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$

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

Diffusion models - "Reverse" process

During training, we observe $\mathbf{x_0}$ **(data to reconstruct)**

$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = N\ (\tilde{\mu}_t,\ \tilde{\Sigma}_t\ )$ <= ***computable distribution***

Argh...

$$\tilde{\mu}_t = \left(\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t}\mathbf{x}_0\right)/\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_t}\right. \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0$$

$$\tilde{\Sigma}_t = \tilde{\beta}_t I \quad \textbf{and} \quad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

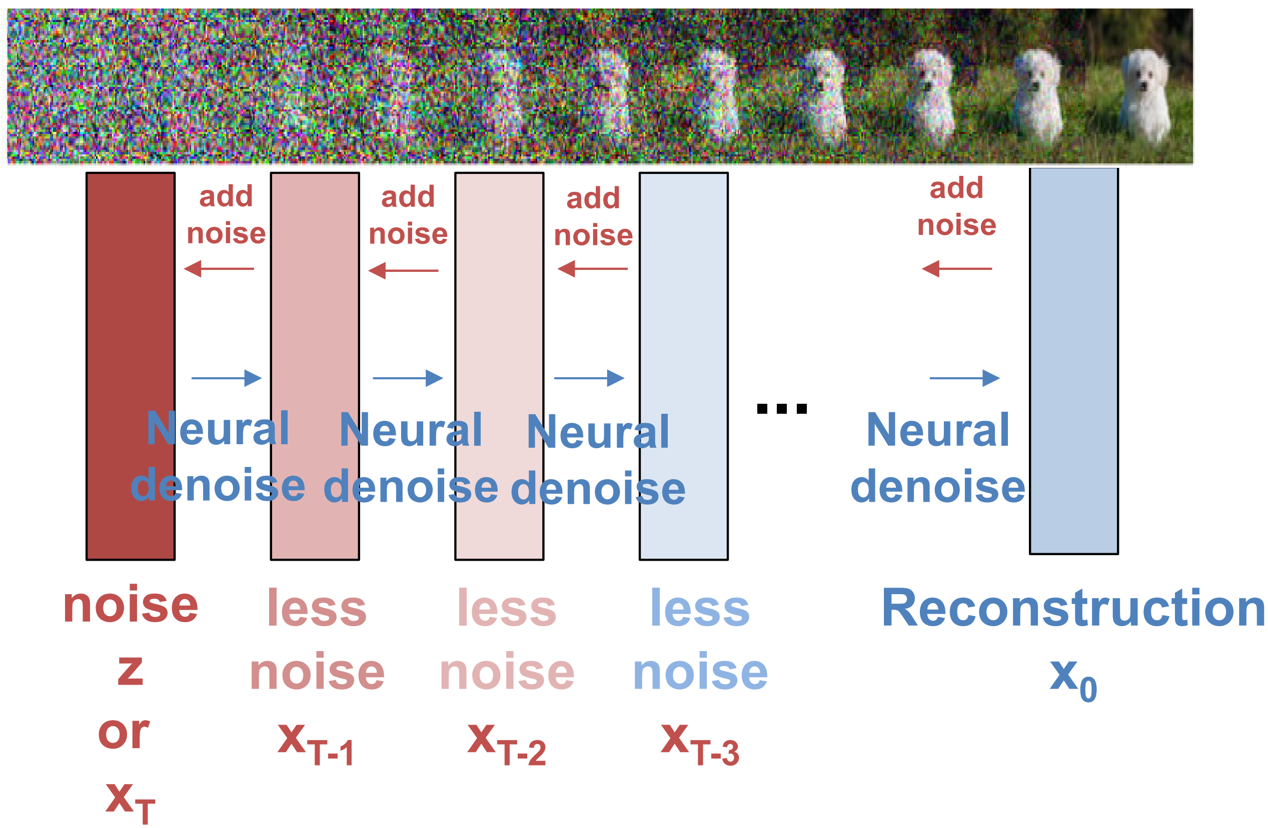
where $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$

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

Diffusion models - "Reverse" process

Basic idea: make the network predict these previous means & covariances as closely as possible using KL divergence...

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \approx \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

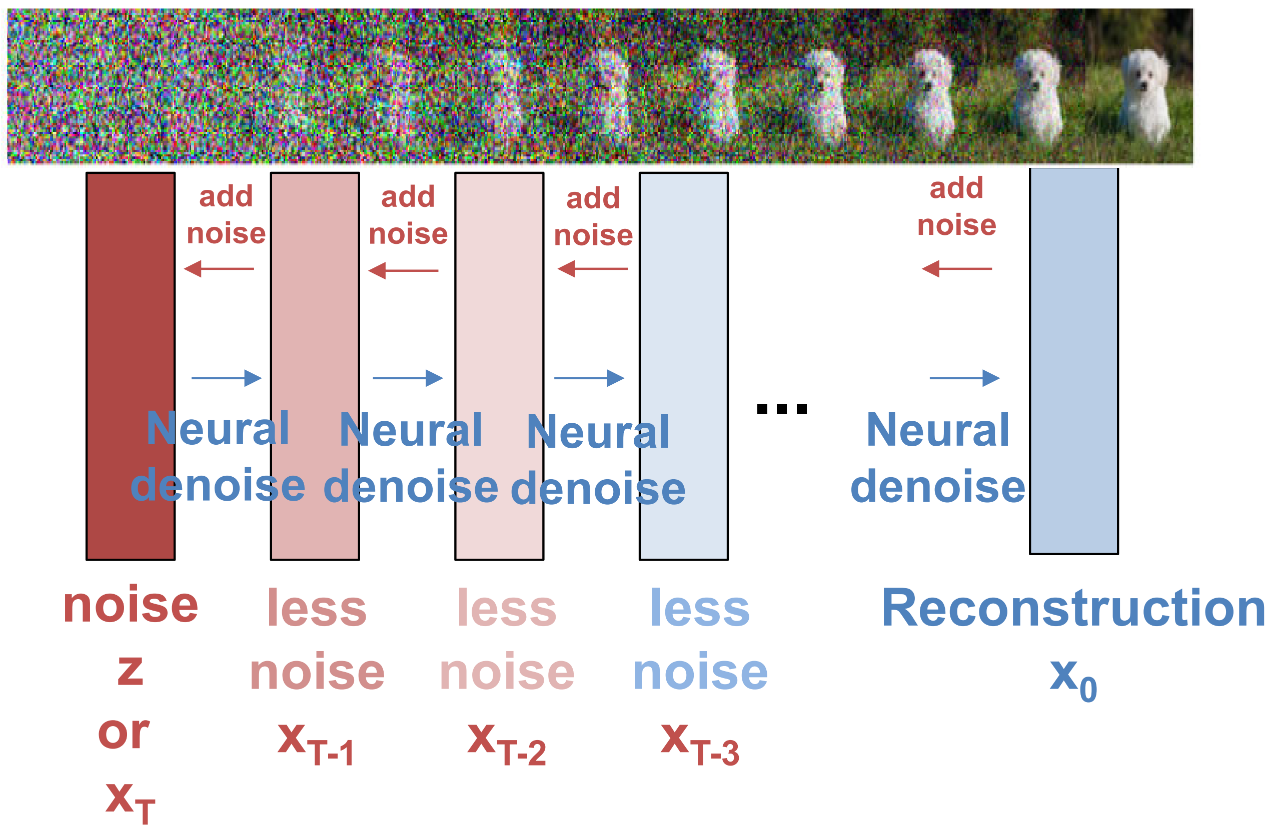


https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion models - “Reverse” process

One more helpful trick. Instead of predicting the **mean...**

$$q(x_{t-1} \mid x_t) = N\left(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right)$$



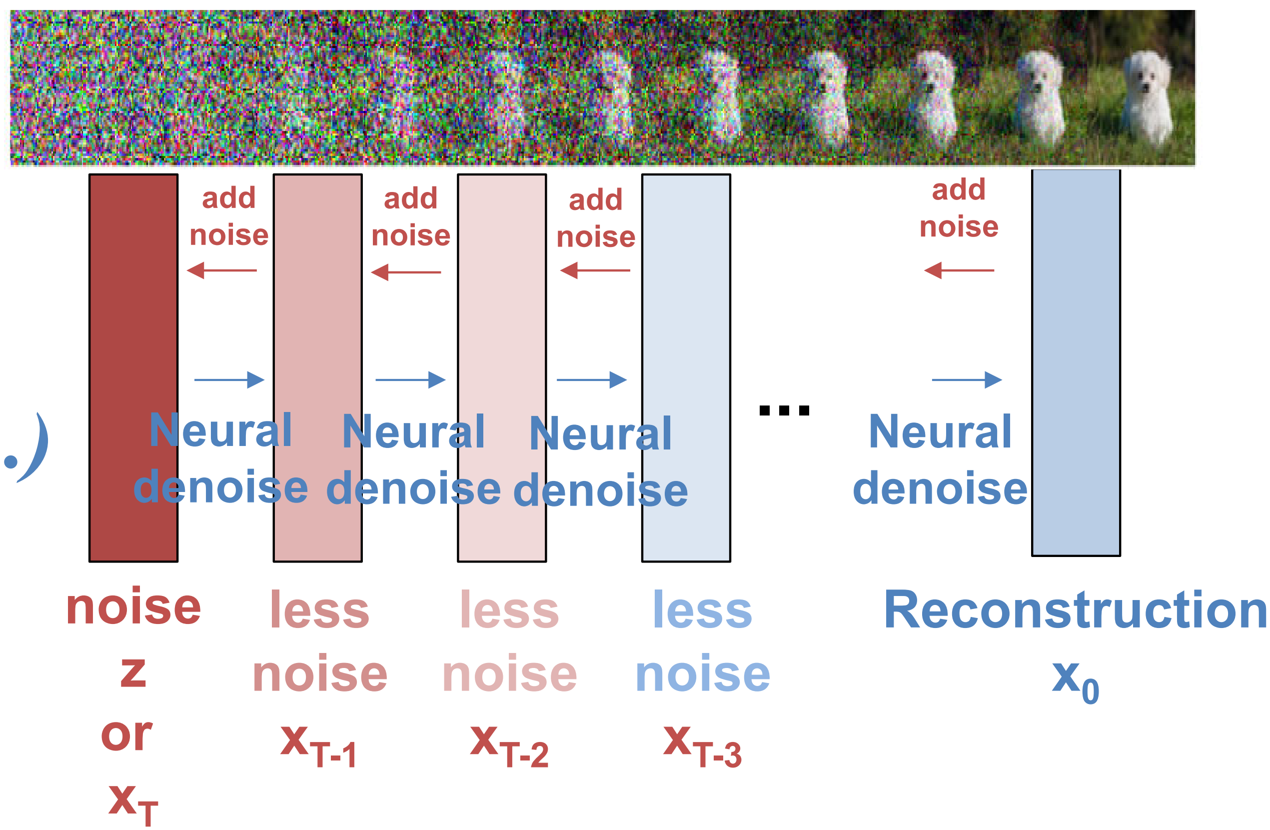
https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion models - "Reverse" process

…predict the noise component (think of it as a residual)

$$q(x_{t-1} \mid x_t) \approx N\left(\alpha_t' x_t - \gamma_t' \varepsilon_\theta(x_t, t),\ \Sigma_\theta(x_t, t) \right)$$

(α_t', γ_t' are scaling factors derived analytically…)



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

Diffusion models - training summary

1. **Sampling step:** generate noisy versions of the input image for a random step

2. **Gradient descent step:** Make the network predict the noise components for that step

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

Conditional Diffusion models

At test time predict the noise component $\boldsymbol{\varepsilon}_{\theta}(\mathbf{x}_t, \mathbf{t}, \mathbf{c})$ conditioned on some input c e.g., class label, text embedding

or…

Conditional Diffusion models

At test time predict the noise component $\boldsymbol{\varepsilon}_{\theta}(\mathbf{x}_t, \mathbf{t}, \mathbf{c})$ conditioned on some input c e.g., class label, text embedding

or...

Predict instead $\boldsymbol{\varepsilon}_{\theta}(\mathbf{x}_t, \mathbf{t}, \mathbf{c}) - \boldsymbol{\varepsilon}_{\theta}(\mathbf{x}_t, \mathbf{t})$ i.e., push the diffusion towards the direction of the input c and away from the direction of input-agnostic noise

GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models
https://www.youtube.com/watch?v=gwI6g1pBD84

GLIDE results

GLIDE (CF Guid.)

“a green train is coming down the tracks”

“a group of skiers are preparing to ski down a mountain.”

“a small kitchen with a low ceiling”

“a group of elephants walking in muddy water.”

“a living area with a television and a table”

“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”

See also **Dall-E 2**: https://cdn.openai.com/papers/dall-e-2.pdf