Stable Diffusion

links

https://jalammar.github.io/illustrated-stable-diffusion/

https://medium.com/@steinsfu/stable-diffusion-clearly-explained-ed008044e07e

https://www.youtube.com/watch?v=ltLNYA3lWAQ

https://github.com/castorini/daam

https://www.youtube.com/watch?v=J2WtkA1Xfew

https://vaclavkosar.com/ml/cross-attention-in-transformer-architecture

https://arxiv.org/abs/2112.10752

https://sander.ai/2022/05/26/guidance.html

https://stable-diffusion-art.com/how-stable-diffusion-work/#How_training_is_done

https://theaisummer.com/diffusion-models/

https://education.yandex.ru/handbook/ml/article/diffuzionnye-modeli

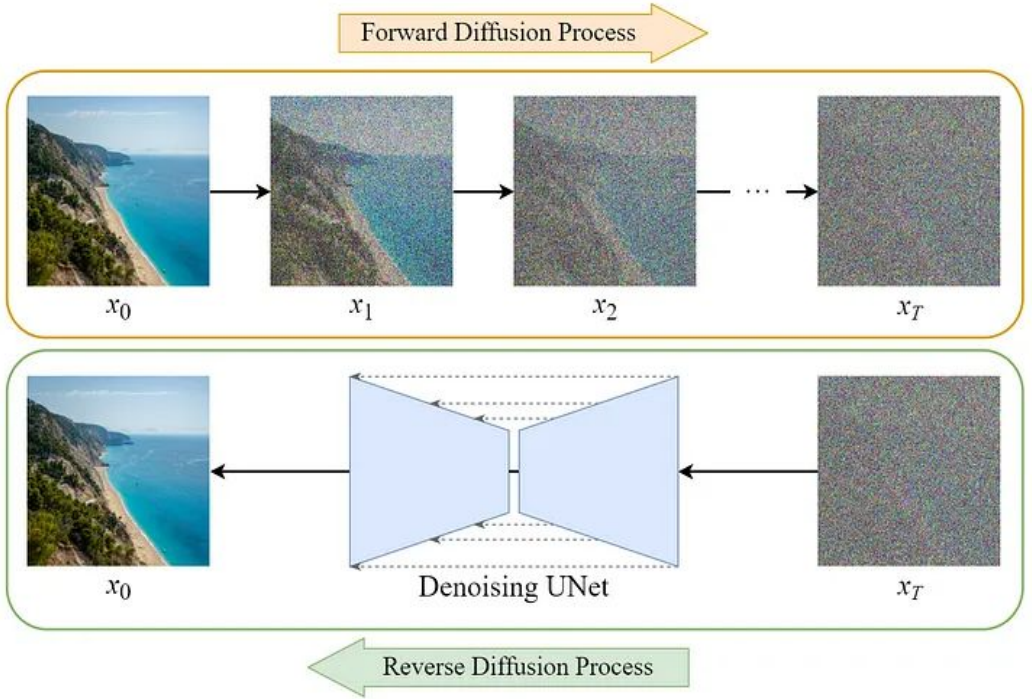
https://arxiv.org/pdf/2403.18103.pdf

https://strikingloo.github.io/wiki/ddim

https://nn.labml.ai/diffusion/stable_diffusion/index.html

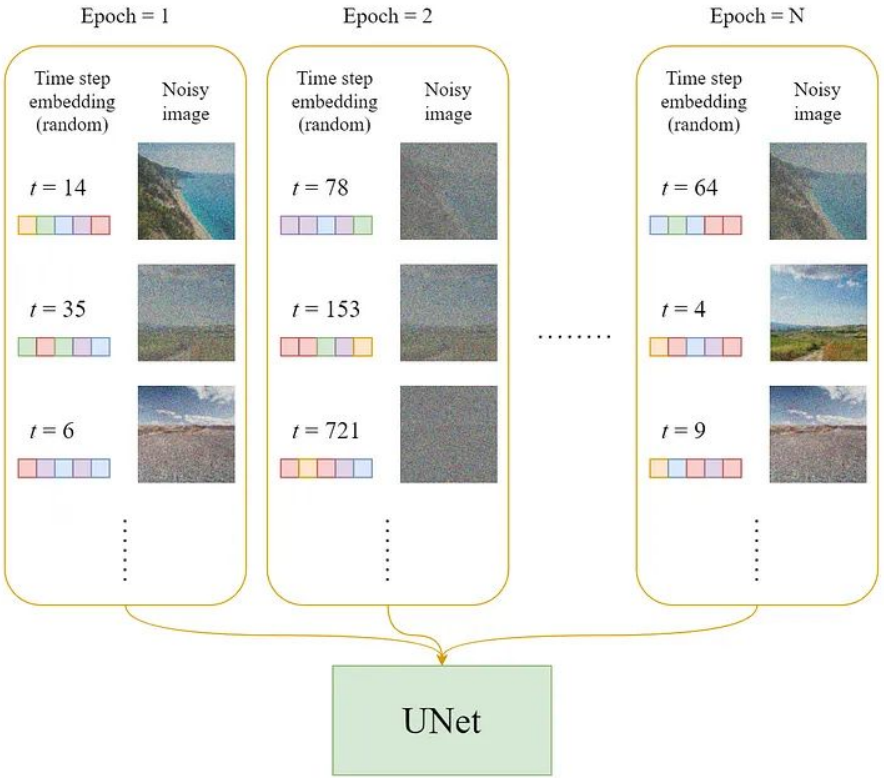
DDPM (Denoising Diffusion Probabilistic Models)

1. Forward Diffusion Process → добавить шум к изображению.
2. Reverse Diffusion Process →убрать шум из изображения .



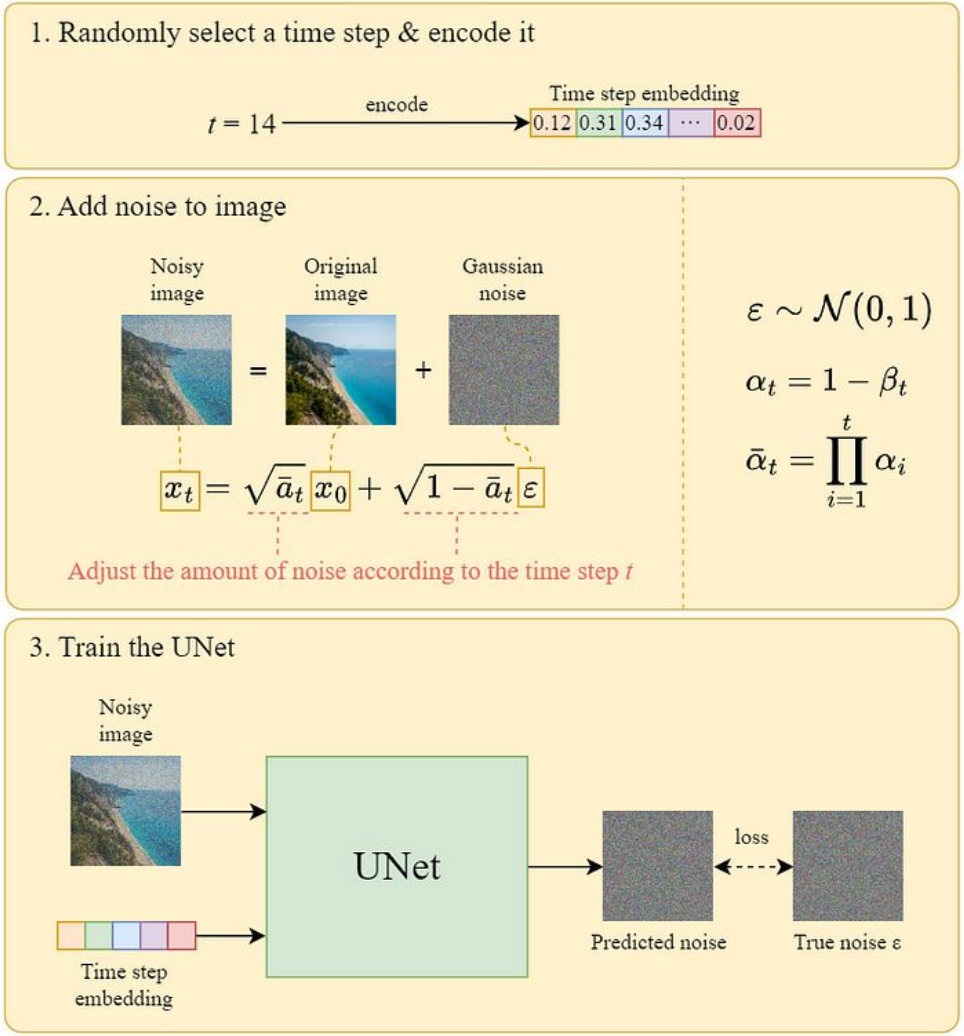
Training In each epoch:

1. Для каждого изображения будет выбран случайный временной шаг t.
2. Применен гауссовский шум (соответствующий t) к каждому изображению.
3. временные шаги t преобразуются в embedding вектор.



Training DDPM

1. Рандомно выбираем t
2. Генерируем шум и прибавляем его к изображению (количество шума регулируется t)
3. Тренируем UNet предсказывать шум для t



Training step

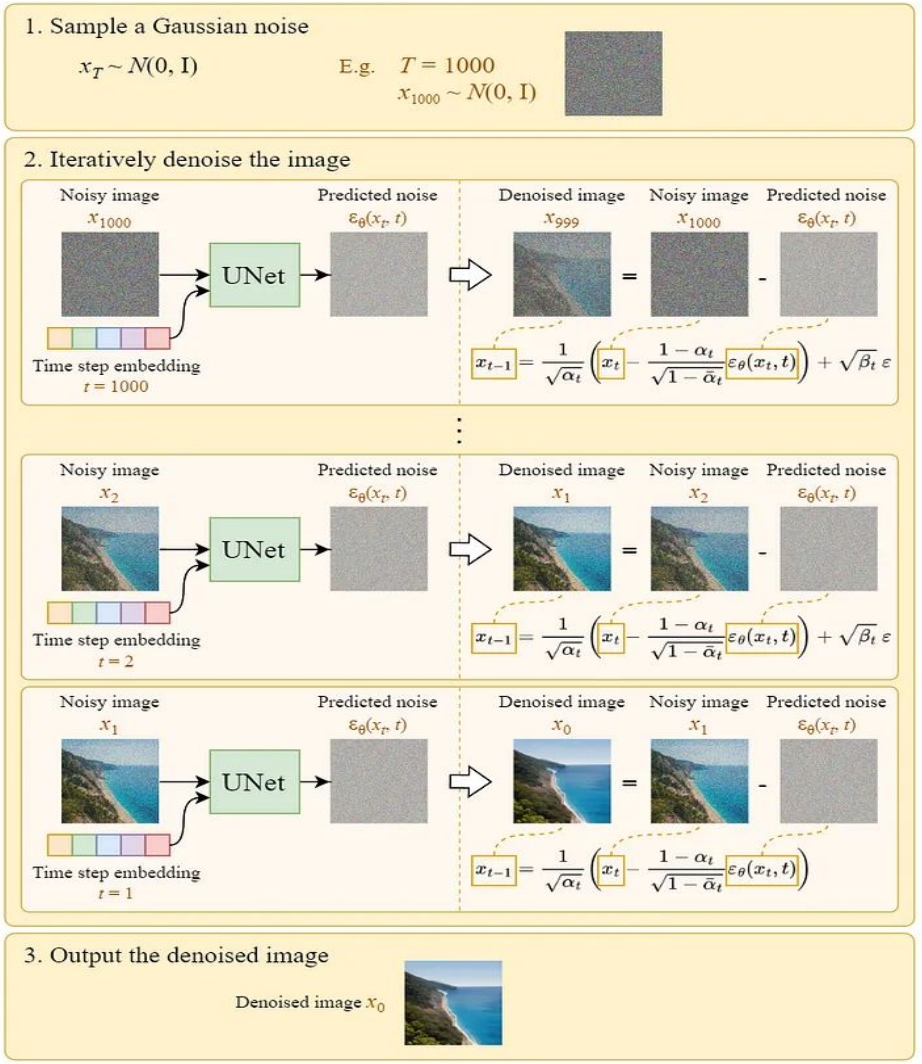
Генерируем гауссовский шум

Итеративно избавляемся от шума пока не получим исходное изображение

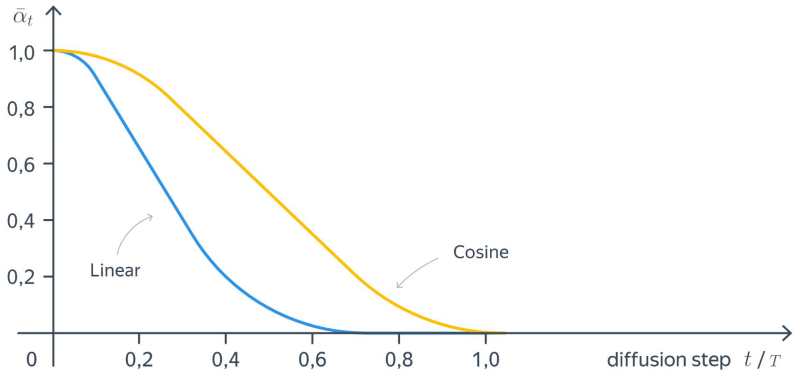
Недостатки:

Работа с целым изображением вычислительно очень медленная и дорогостоящая операция

DDPM - генерация занимает много времени и хотелось генерировать быстрее чем за 1000 шагов



Improved Denoising Diffusion Probabilistic Models



https://arxiv.org/abs/2102.09672

DDIM (Denoising Diffusion Implicit Models)

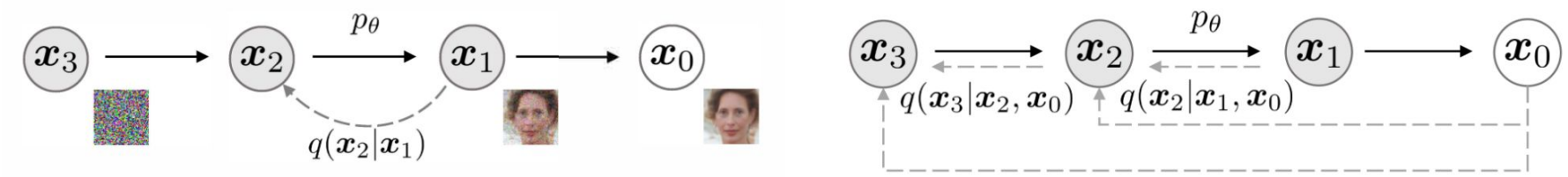


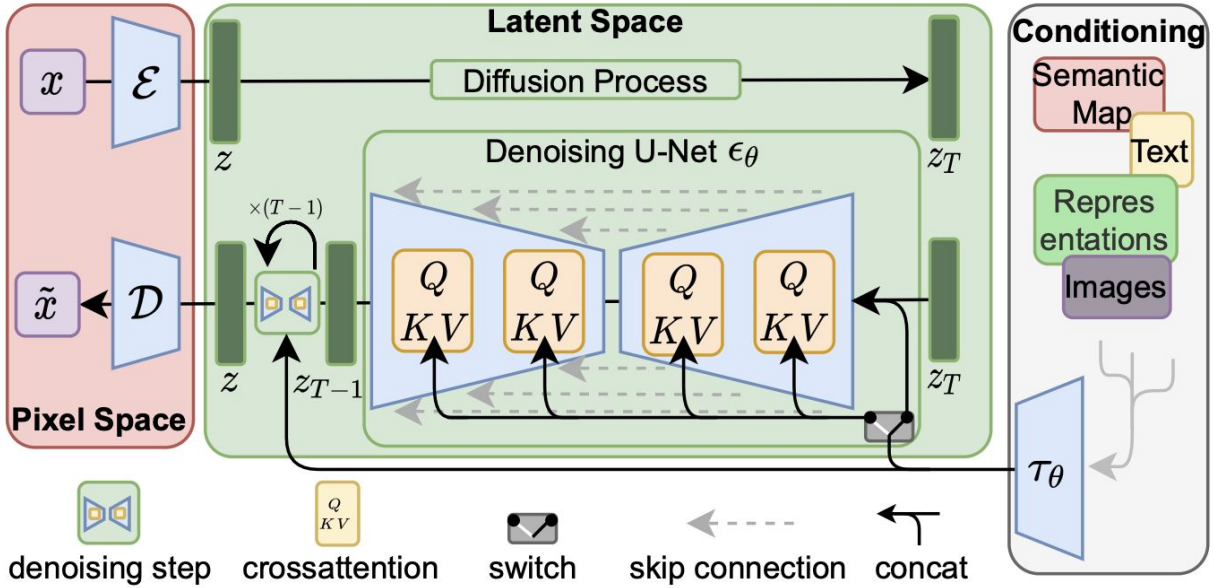
Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

Авторы предопределили Backward process

https://arxiv.org/pdf/2010.02502.pdf

Latent Diffusion Models

- Encoder/Decoder
- Latent Space
- UNet Scheduler
- Conditioning

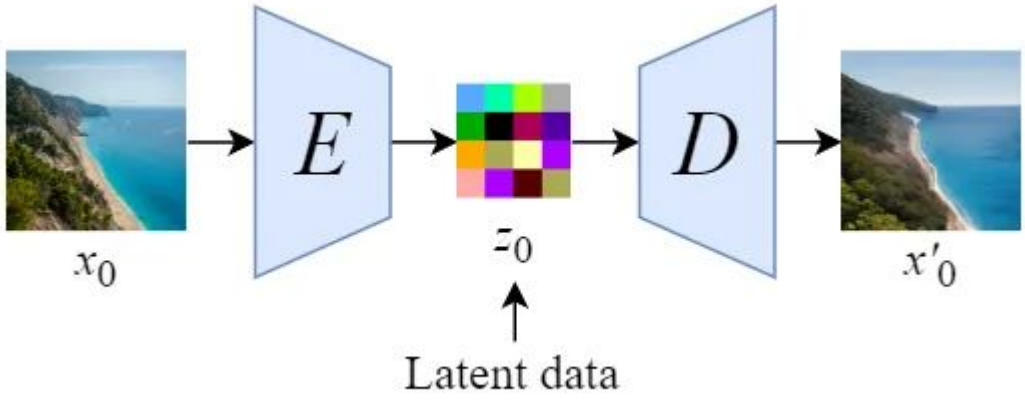


Latent Space

процесс диффузии теперь выполняется в латентном пространстве

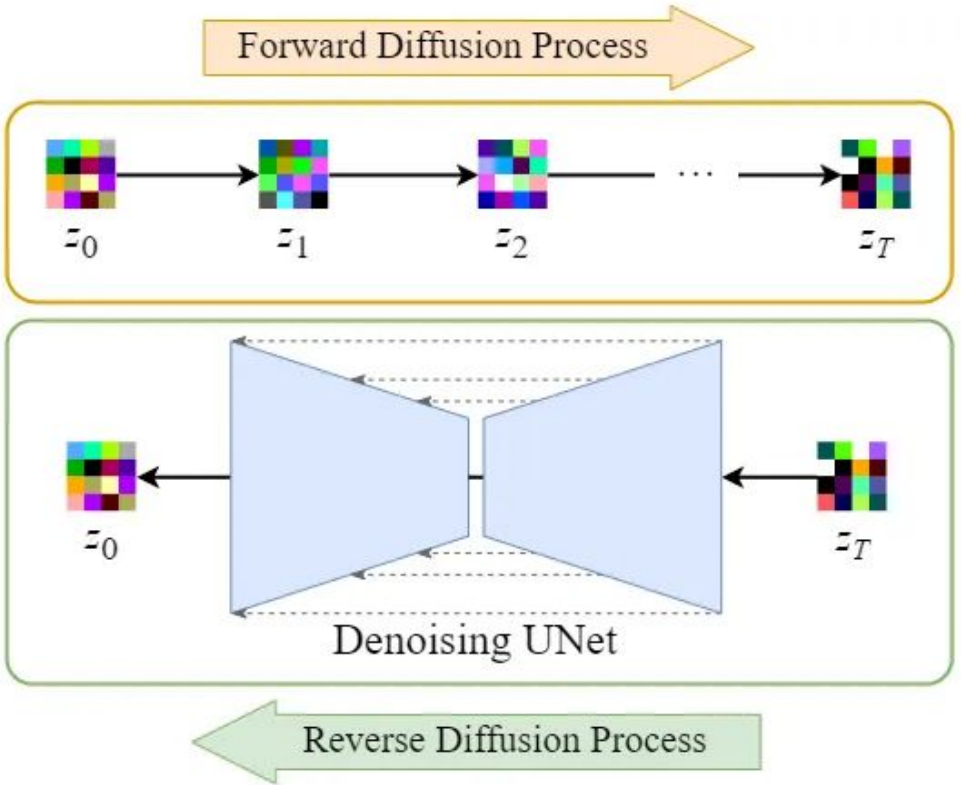
- encoder ***E*** кодирует изображение
- decoder **D** декодирует

изображение(3, 512, 512) -> латент(4,64,64)



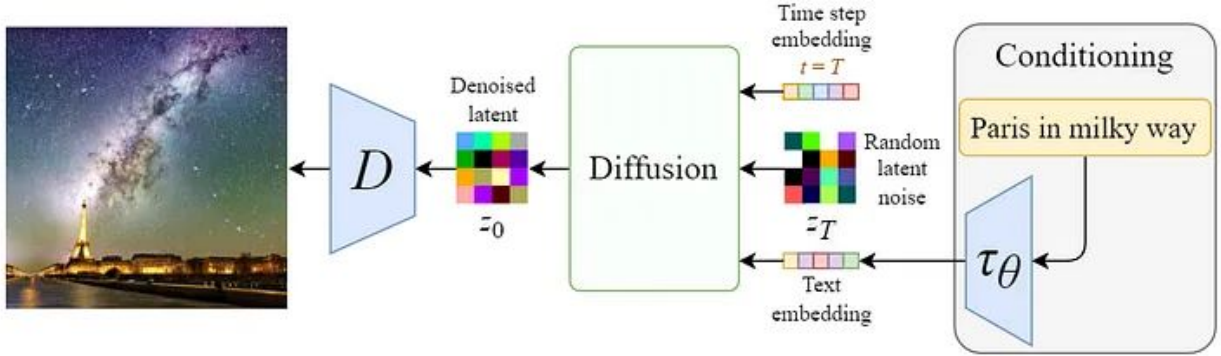
Latent Diffusion

1. Forward Diffusion Process → добавить шум к латенту.
2. Reverse Diffusion Process →убрать шум из латента.



Conditioning

Добавление в процесс диффузии (условия) в данном случае текста с использованием механизма **cross-attention** Для генерации новых изображений по текстовому описанию



Classifier-free guidance

Все, что нам нужно, чтобы превратить модель безусловной диффузии в условную, — это классификатор!

В отличие от вероятностных моделей на прямую аппроксимирующие p(x), диффузионная модель предсказывает score function

$$\nabla_x \log p(x \mid y).$$

Classifier guidance

$$\nabla_x \log p_\gamma(x \mid y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y \mid x).$$

Classifier-free guidance

$$\nabla_x \log p_\gamma(x \mid y) = \nabla_x \log p(x) + \gamma \left(\nabla_x \log p(x \mid y) - \nabla_x \log p(x)\right),$$

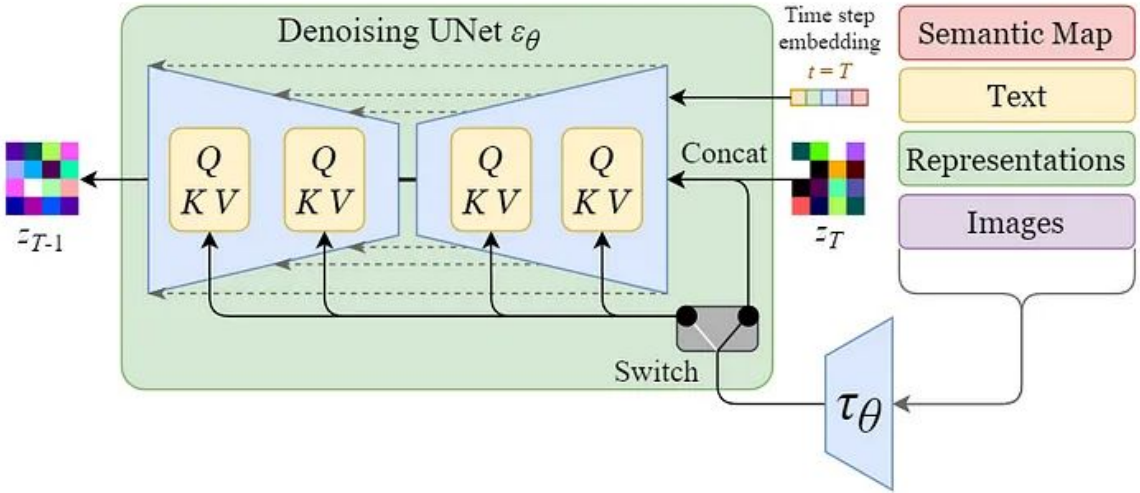
https://sander.ai/2022/05/26/guidance.html

Conditioning

switch нужен для использование разного типа “условий” на вход

для текста используется языковая модель для получения вектора (BERT, CLIP) и с помощью Attention мапится в UNet

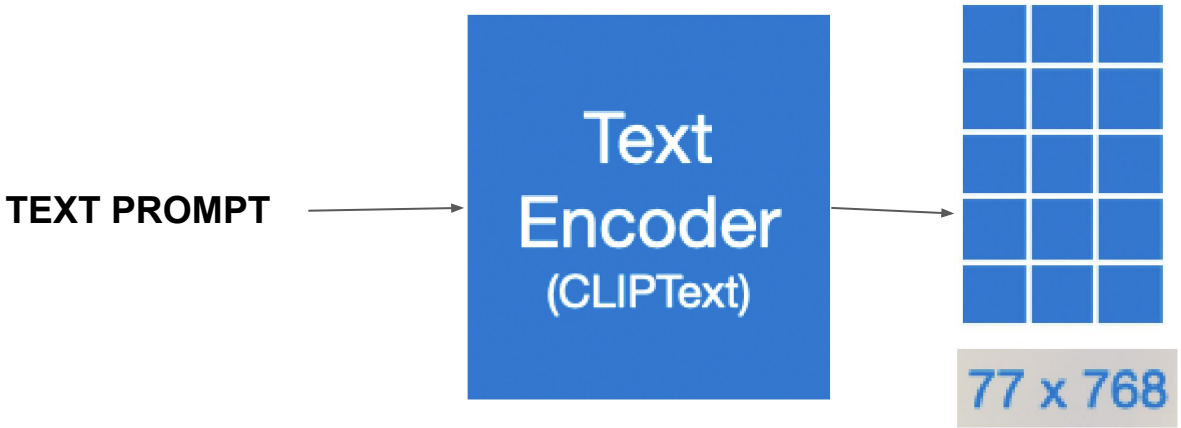
Для другого рода входов (семантические маски, изображения) может использоваться конкатинация



ClipText

Input: text.

Output: 77 token embeddings vectors, each in 768 dimensions.



Loss function

$$x_t = \sqrt{\bar{a}_t}\, x_0 + \sqrt{1 - \bar{a}_t}\, \varepsilon$$

$$L_{\text{simple}} = \mathbb{E}_{t, x_0, \boldsymbol{\varepsilon}} \left[\| \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(x_t, t) \|^2 \right]$$

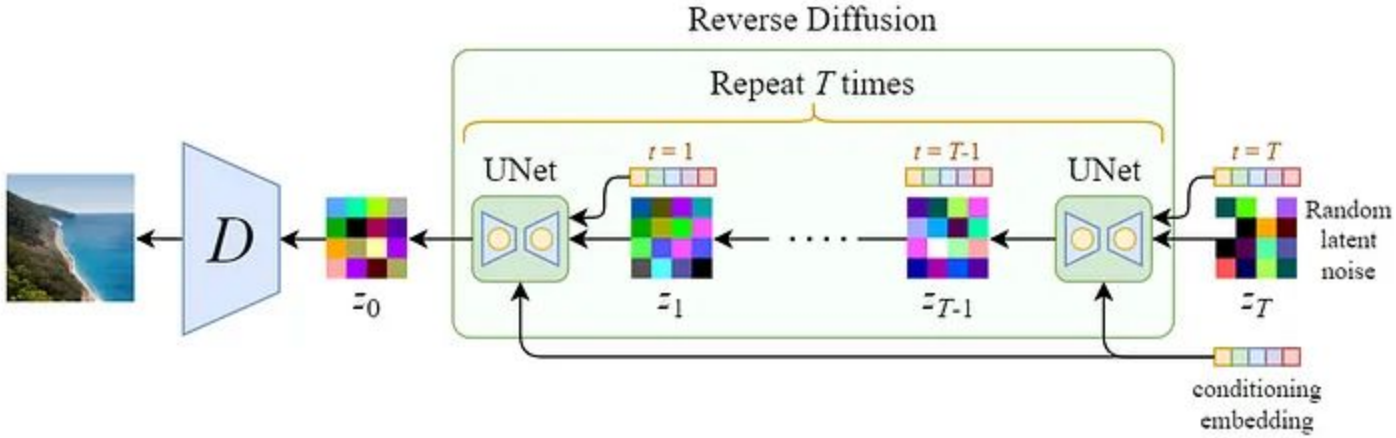
$$z_0 = E(x_0)$$

$$z_t = \sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon$$

$$L_{\text{LDM}} = \mathbb{E}_{t, z_0, \boldsymbol{\varepsilon}, y} \left[\| \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(z_t, t, \tau_\theta(y)) \|^2 \right]$$

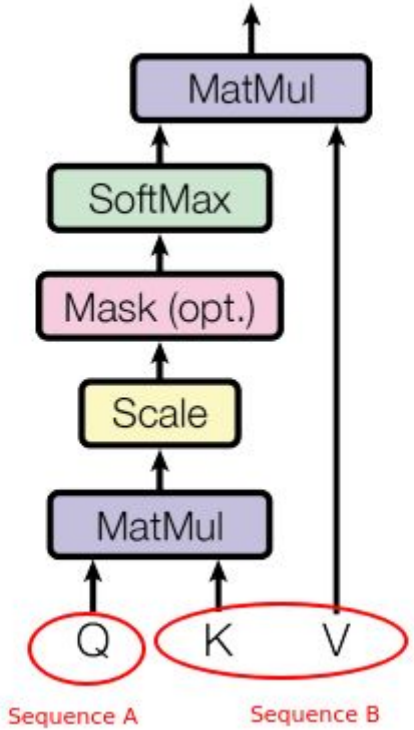
Conditioning

Sampling

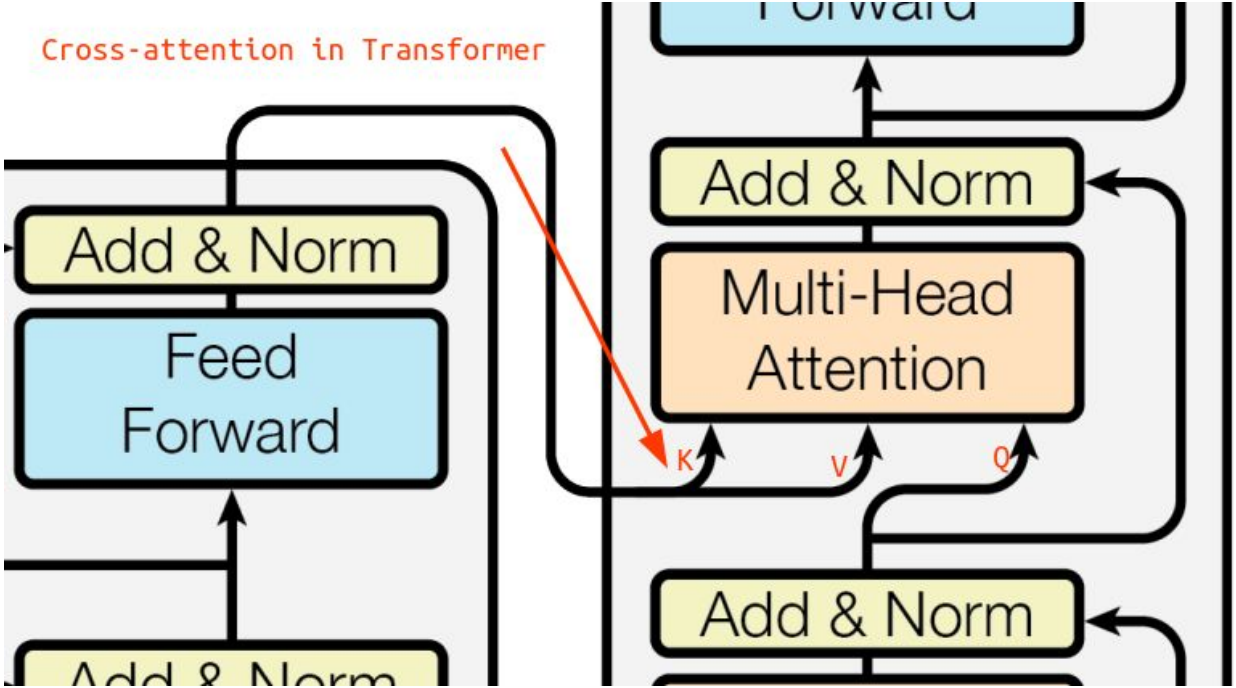

Reverse Diffusion
Repeat T times
UNet
UNet
$t = 1$
$t = T\text{-}1$
$t = T$
D
z_0
z_1
z_{T-1}
z_T
Random latent noise
conditioning embedding

Cross attention

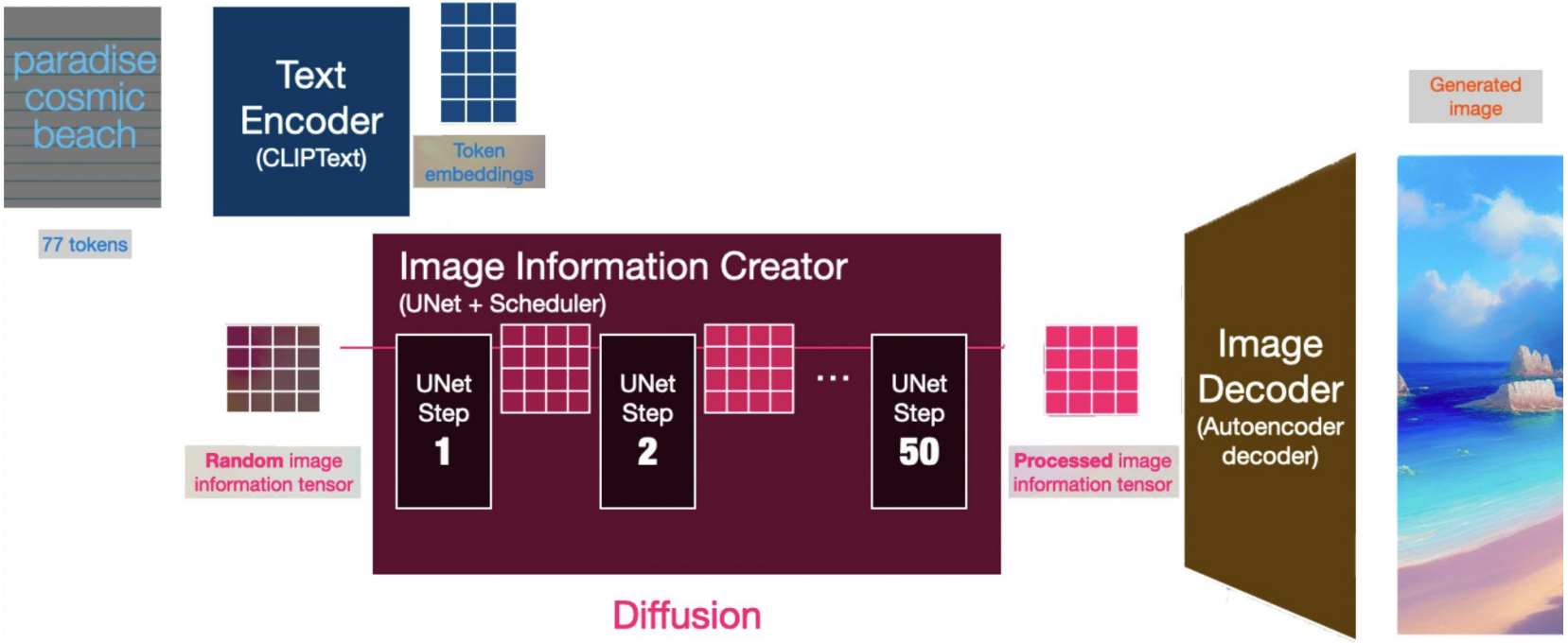
- механизм внимания в архитектуре Transformer, который смешивает две разные последовательности
- две последовательности должны иметь одинаковую размерность
- две последовательности могут иметь разную модальность (например, текст, изображение, звук)
- одна из последовательностей определяет длину вывода, так как она играет роль **query**
- другая последовательность является источником векторов **key value**



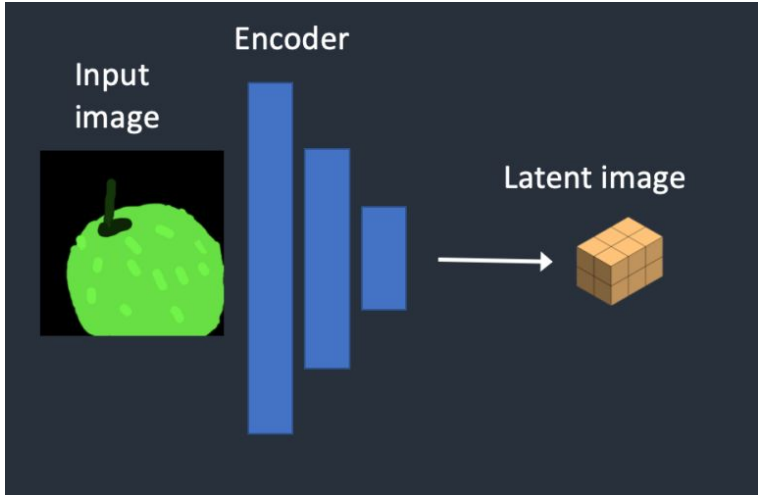
Cross-Attention in Transformer Decoder

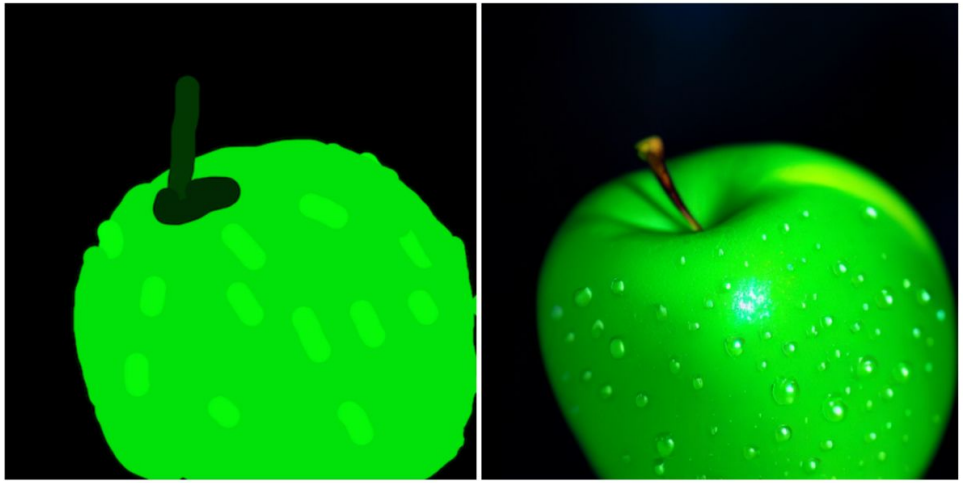


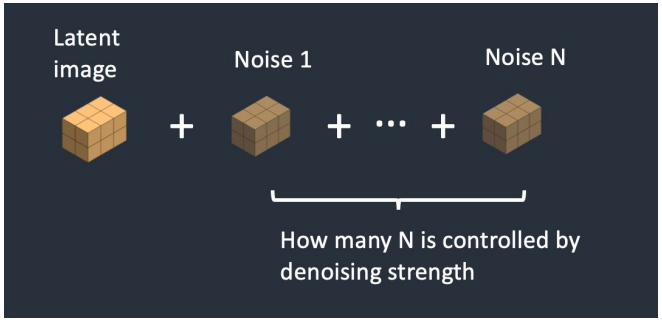
text2img


paradise
cosmic
beach
77 tokens
Text
Encoder
(CLIPText)
Token
embeddings
Image Information Creator
(UNet + Scheduler)
UNet
Step
1
UNet
Step
2
UNet
Step
50
Random image
information tensor
Processed image
information tensor
Image
Decoder
(Autoencoder
decoder)
Generated
image
Diffusion

Img2img


Encoder
Input image
Latent image




Latent image
Noise 1
Noise N
How many N is controlled by denoising strength