Генерация изображений

Генерация изображений

Задачи генерации:

- **генерация изображения (generation)**
- условная генерация изображения (conditional generation)
- генерация изображения по текстовому описанию (text-to-image)
- …

Задача генерации изображений

Распределение данных $p(x)$

Дано: датасет D ~ $p(x)$

Задача генерации: научиться сэмплить $x \sim p(x)$

Задача условной генерации: научиться сэмплить $x \sim p(x|y)$

Оценка качества генерации

Проблема: нет groundtruth

Используемые метрики:

- Inception score (IS)
- Fréchet Inception distance (FID)
- Precision & Recall
- text-to-image
  - CLIPScore

Что будем оценивать?

[F] качество (fidelity)

low

high

[D] разнообразие (diversity)

low

high

Inception Score

Inception score (IS): paper (чем больше, тем лучше)

Как это можно оценить?
Интуиция:
Возьмем хороший классификатор Inception

- [F] Inception уверенно распознает класс по изображению
- [D] генерируются все классы

С точки зрения распределений

- [F] p(y|x) имеет низкую энтропию
- [D] p(y) имеет высокую энтропию

Формула: $IS(G) = \exp\{\mathbb{E}_{x \sim p_G(x)} D_{KL}(p(y|x)||p(y))\}$
G - порождающая модель
D_{KL} - дивергенция Кульбака-Лейблера
exp - экспонента для удобства восприятия людьми

Inception Score

Особенность IS:
Inception подходит только для естественных изображений. Для другого домена надо взять <domain classifier model> score.

Проблемы IS:

- легко “обмануть”: запомнить обучающий [сбалансированный] датасет
- игнорирование настоящих данных: сгенерированные изображения сравниваются не с ними, а с Inception
- проблема с классами:
  - а что если на картинке несколько объектов разных классов?
  - а что если на картинке объект класса не из ImageNet?

Fréchet Inception distance

Fréchet Inception distance (FID): paper (чем меньше, тем лучше)

Что хотим оценить: схожесть распределения признаков настоящих и сгенерированных изображений

Как это можно оценить?

- возьмем хороший классификатор Inception
- сделаем из него feature extractor (отбросив последний полносвязный слой)
- сравним распределения извлеченных признаков у настоящих и сгенерированных изображений
- посчитаем расстояние Fréchet между распределениями

Fréchet Inception distance

Как считается расстояние Fréchet?

Note: для простоты вычислений делается допущение о нормальности распределений $X \sim \mathcal{N}(\mu_X, \Sigma_X), Y \sim \mathcal{N}(\mu_Y, \Sigma_Y)$

Note: параметры распределений берутся из статистик

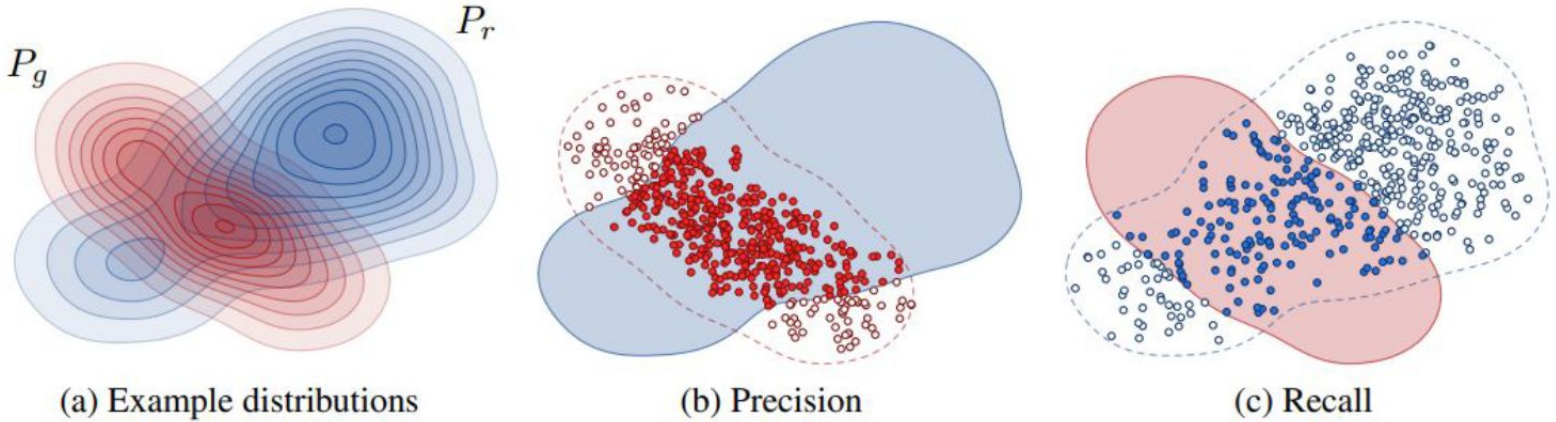
Формула:

$$FID = ||\mu_X - \mu_Y||^2 + Tr(\Sigma_X + \Sigma_Y - 2\sqrt{\Sigma_X \Sigma_Y})$$

Precision & Recall

Precision & Recall [paper]

- [F] precision
- [D] recall



(a) Example distributions (b) Precision (c) Recall

Precision & Recall

Даны настоящие и сгенерированные данные: $X_r \sim P_r, X_g \sim P_g$

Возьмем энкодер из хорошего классификатора

Рассмотрим векторы признаков ϕ_r, ϕ_g

Множества всех векторов $\mathbf{\Phi}_r, \mathbf{\Phi}_g$

$$f(\phi, \mathbf{\Phi}) = \begin{cases} 1, \text{ if } \left\|\phi - \phi'\right\|_2 \leq \left\|\phi' - \mathrm{NN}_k\left(\phi', \mathbf{\Phi}\right)\right\|_2 \text{ for at least one } \phi' \in \mathbf{\Phi} \\ 0, \text{ otherwise,} \end{cases}$$

$$\mathrm{precision}(\mathbf{\Phi}_r, \mathbf{\Phi}_g) = \frac{1}{|\mathbf{\Phi}_g|} \sum_{\phi_g \in \mathbf{\Phi}_g} f(\phi_g, \mathbf{\Phi}_r) \qquad \mathrm{recall}(\mathbf{\Phi}_r, \mathbf{\Phi}_g) = \frac{1}{|\mathbf{\Phi}_r|} \sum_{\phi_r \in \mathbf{\Phi}_r} f(\phi_r, \mathbf{\Phi}_g)$$

CLIP Score

CLIPScore paper

Проверяет соответствие изображения *i* текстовому описанию *t* (с масштабом *w*):

$$\text{CLIPScore}(i, t) = w * \max\{0, \cos(\text{CLIP}(i), \text{CLIP}(t))\}$$

Таксономия генеративных моделей

2 семейства

1. Explicit density models
    - оценка плотности
    - сэмплирование
2. Implicit density models
    - ~~оценка плотности~~
    - сэмплирование

Таксономия генеративных моделей

- explicit density models
  - tractable density models
    - autoregressive models
    - flow-based models
  - approximate density models
    - VAEs
    - diffusion models
- implicit density models
  - GANs

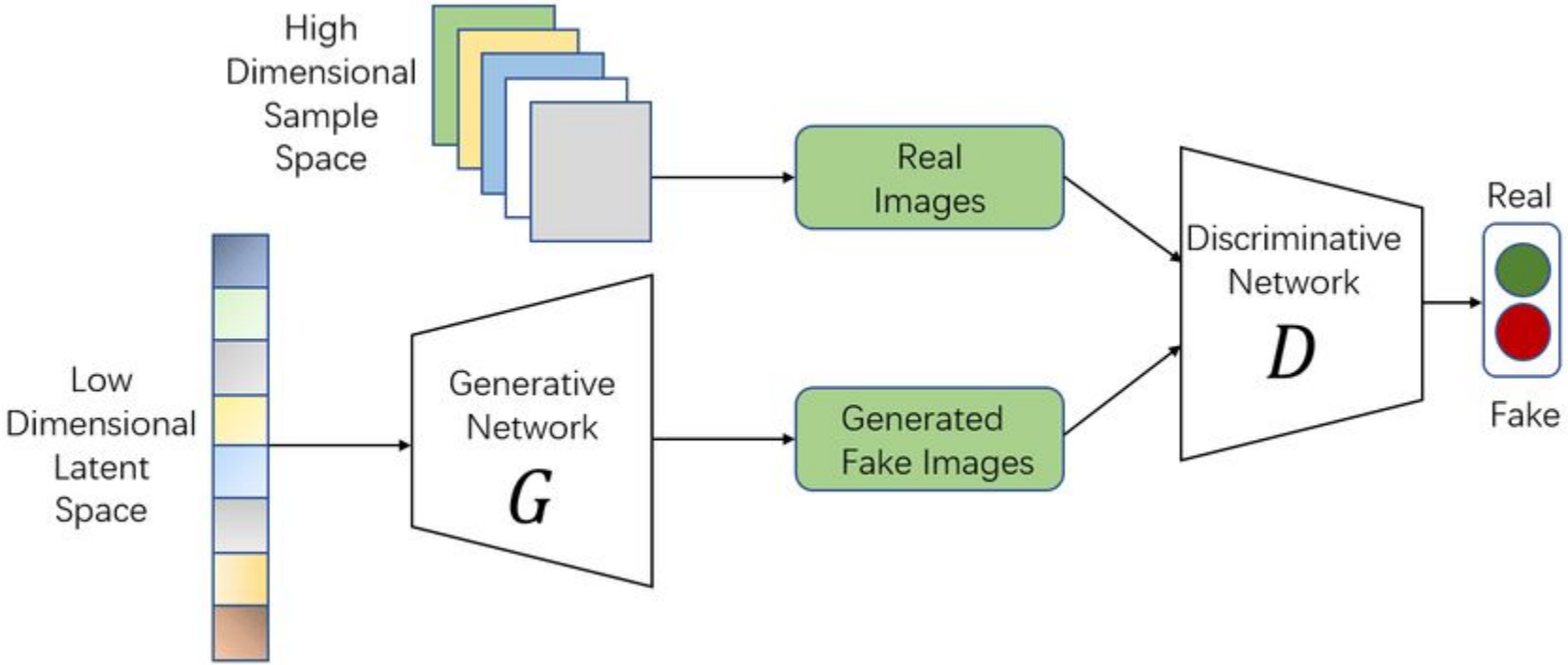
GAN

GAN

Генеративно-состязательная сеть (Generative Adversarial Network, GAN) - алгоритм обучения порождающей модели без учителя

- Как сгенерировать x ~ p(x)?
  Идея: z ~ N(0, I) -> generator model -> x ~ p(x)
- Как научить модель model конвертировать z в x с нужным распределением?
  Идея GAN: использовать модель-дискриминатор, распознающую подделки

Архитектура GAN



Обучение GAN

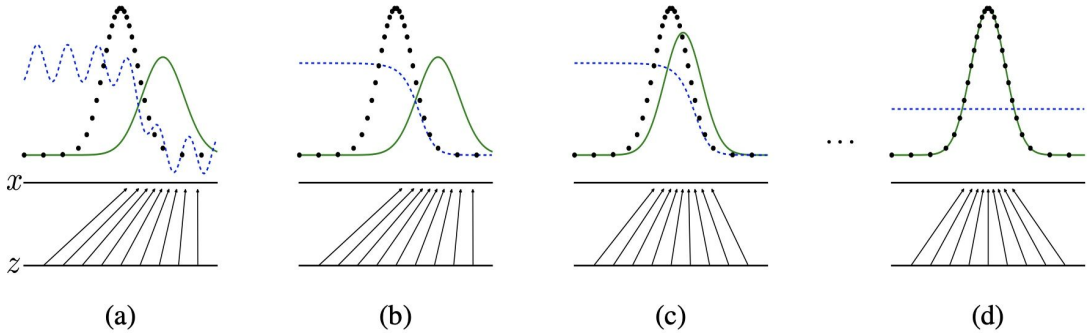
Оптимизация $\min_G \max_D L(G, D)$

где $L(G, D) = \mathbb{E}_{x \sim p(x)}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(Z)))]$

Note: На практике обучают G и D поочередно

Пример обучения:

- реальные данные
- генерируемые данные
- разделяющая кривая дискриминатора



Когда использовать GAN?

Плюсы:

- высокое качество
- быстрая генерация

Минусы:

- малое разнообразие генерируемого
- сложно обучать

Проблемы GAN

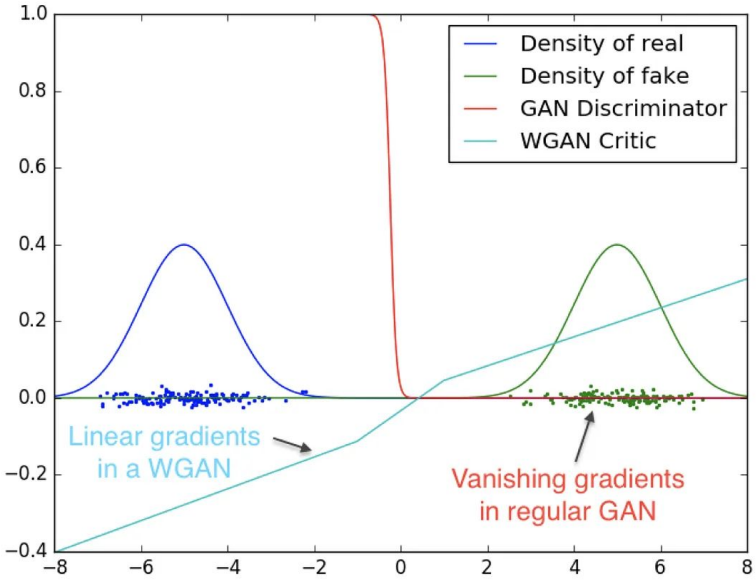
- Затухание градиента (vanishing gradient)
- Схлопывание мод распределения (mode collapse)
- Проблема стабильности обучения (convergence failure)
- Непонимание глобальной информации

Vanishing Gradient

Причины:

генератор еще плохой
 => дискриминатору легко различать
 => генератор получает нулевую производную
 => генератор не может обучиться

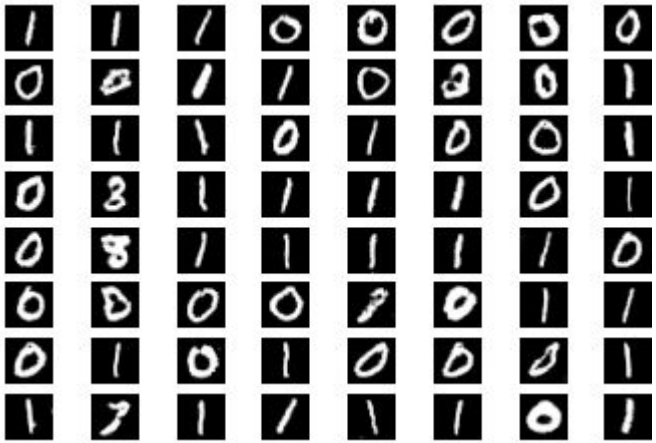
Решение: лосс получше (напр., Wasserstein Loss)

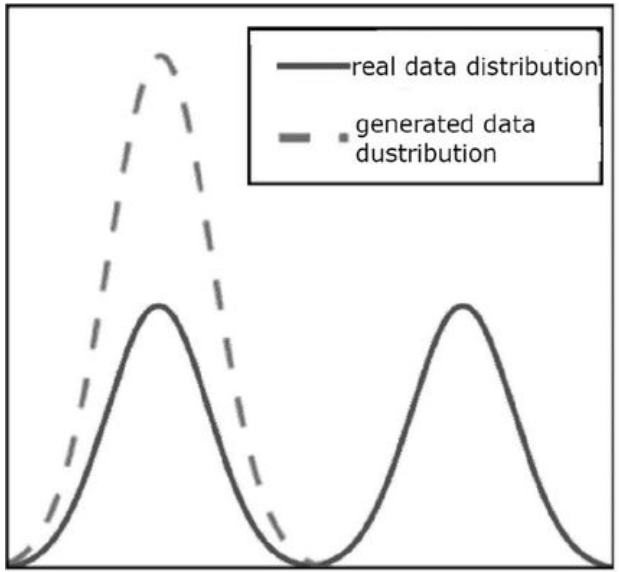


Mode Collapse

Проблема: сэмплит только из некоторых мод

Решение: WGAN





Convergence Failure

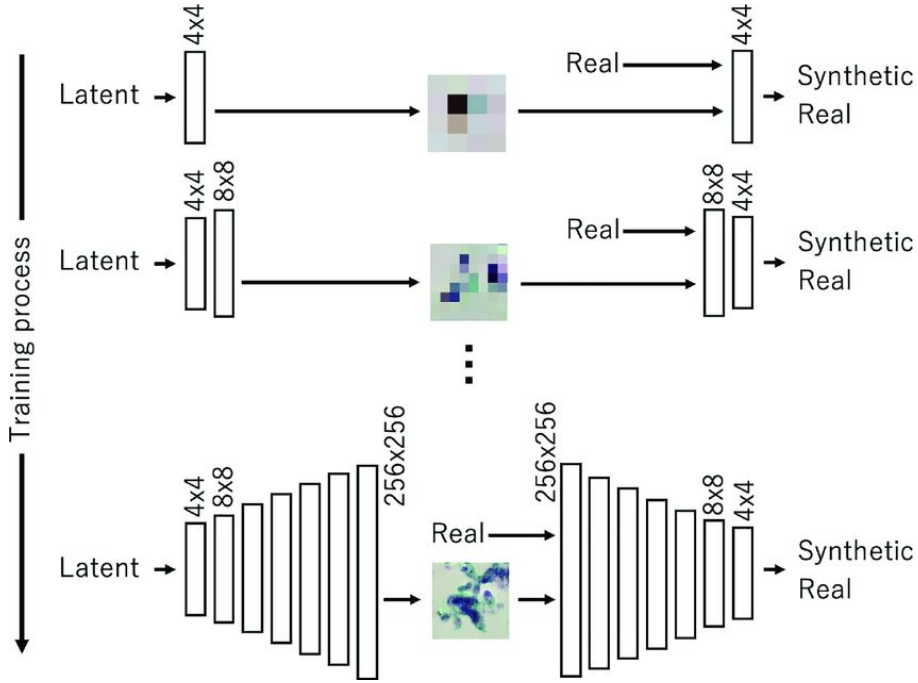
Note: Сходимость не гарантирована

Пример: осциллирование между модами

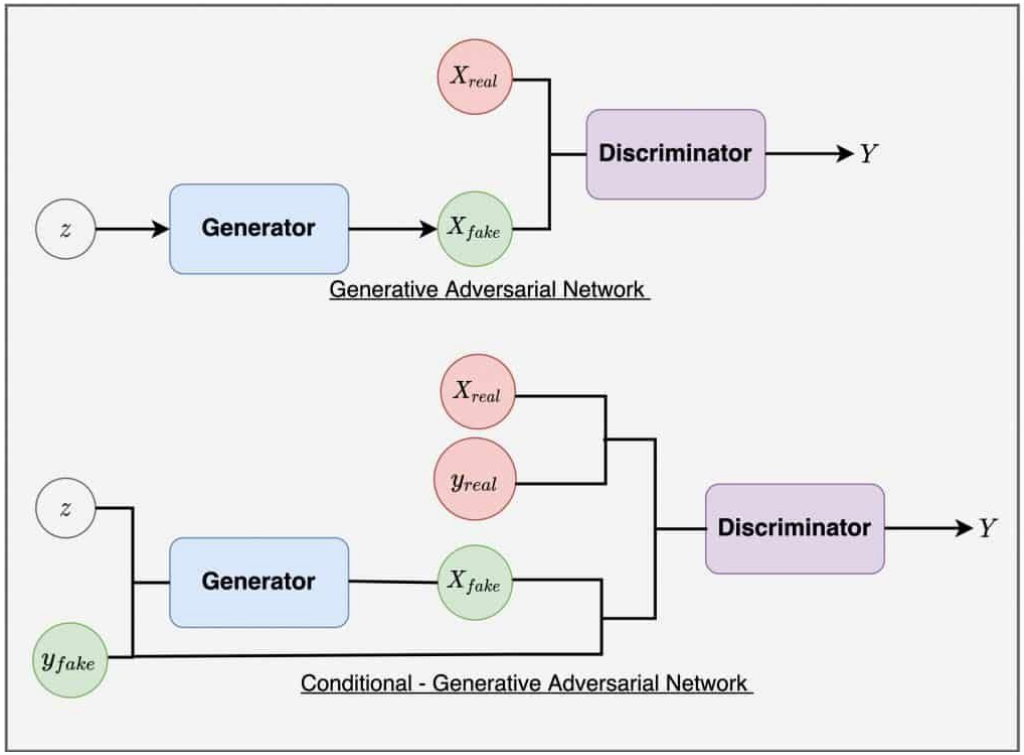
Непонимание глобальной информации

Проблема: на верхних слоях нет глобальной информации

Решение: ProGAN



Conditional GAN



Авторегрессионные модели

Авторегрессионные модели

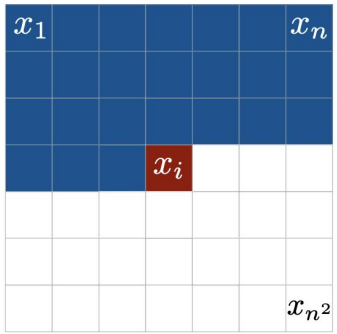
Авторегрессионные модели (autoregressive models)

Идея: генерим пиксель за пикселем (при условии предыдущих) $p(x_n|x_{n-1},...x_{n-k})$

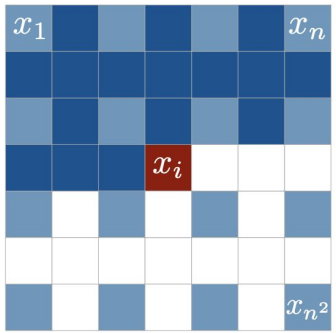
Проблема: очень долгий инференс

Примеры:

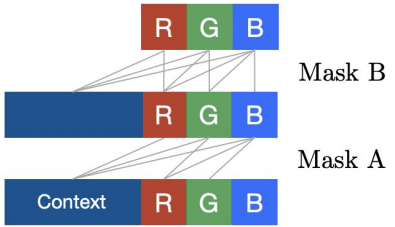
- Pixel RNN
- Pixel CNN
- Autoregressive Flows



Context

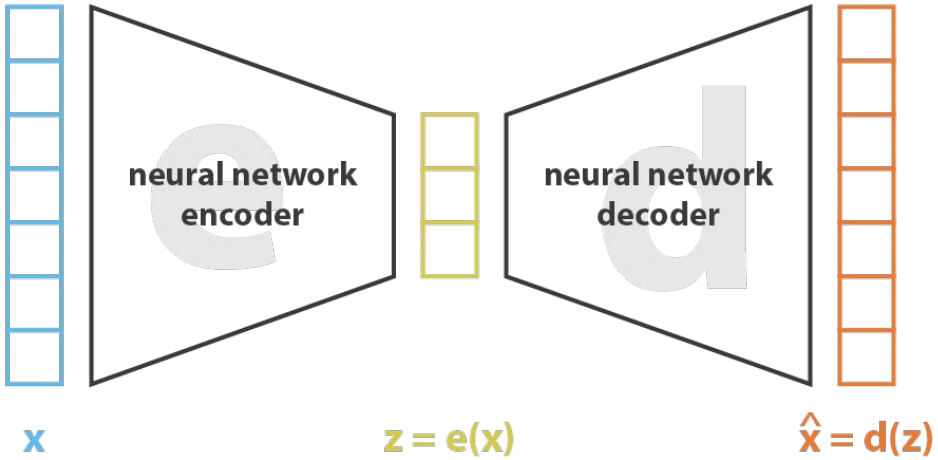


Multi-scale context



VAE

Автоэнкодер

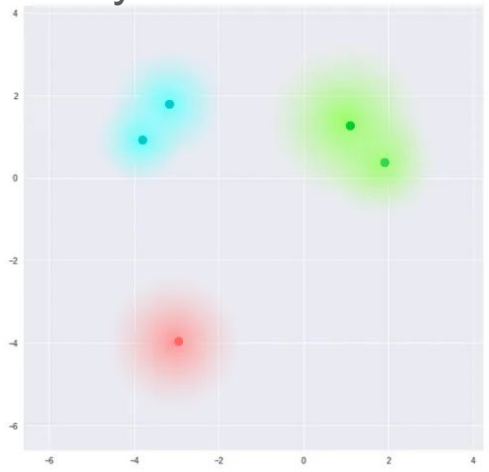


$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

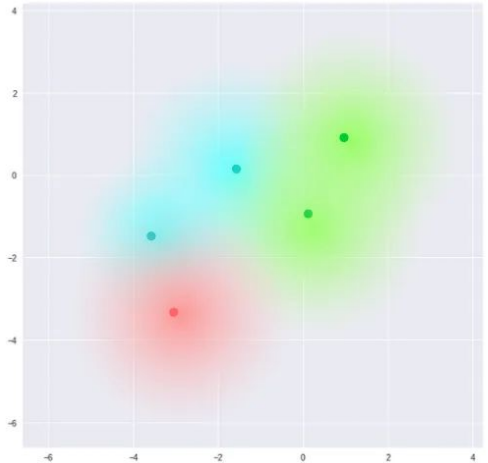
Проблема автоэнкодера

Проблема: дискретность латентных представлений

получим



хотим



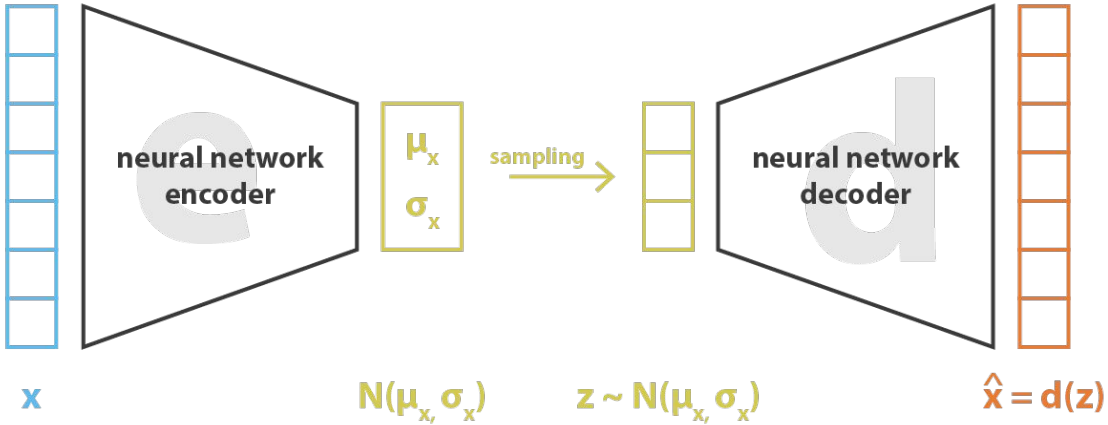
VAE

Вариационный автокодировщик (Variational Autoencoder, VAE)

Как обеспечить непрерывность латентных представлений:

1. энкодер отображает объект *x* не в одну точку *z*, а в некоторое латентное распределение на *z*
2. регуляризация латентных распределений к *N(0, I)*

Архитектура VAE



$$\text{loss} = \| x - \hat{x} \|^2 + KL[\, N(\mu_x, \sigma_x), N(0, I) \,] = \| x - d(z) \|^2 + KL[\, N(\mu_x, \sigma_x), N(0, I) \,]$$

D_{KL} между нормальными

Легко считается по формуле

Одномерное нормальное распределение

$$D_{\mathrm{KL}}(p \parallel q) = \log \frac{\sigma_1}{\sigma_0} + \frac{\sigma_0^2 + (\mu_0 - \mu_1)^2}{2\sigma_1^2} - \frac{1}{2}$$

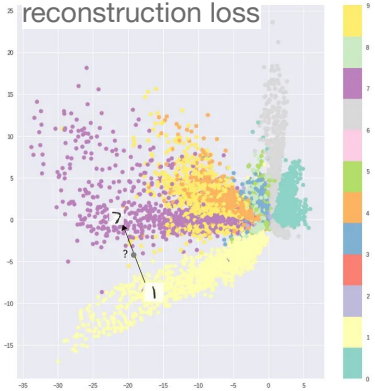
Многомерное нормальное распределение

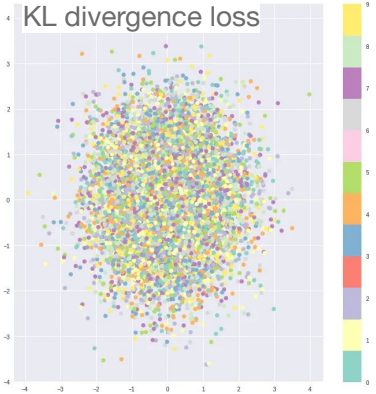
$$D_{\mathrm{KL}}(\mathcal{N}_0 \parallel \mathcal{N}_1) = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right) - k + (\mu_1 - \mu_0)^{\mathsf{T}}\Sigma_1^{-1}(\mu_1 - \mu_0) + \ln\left(\frac{\det\Sigma_1}{\det\Sigma_0}\right)\right).$$

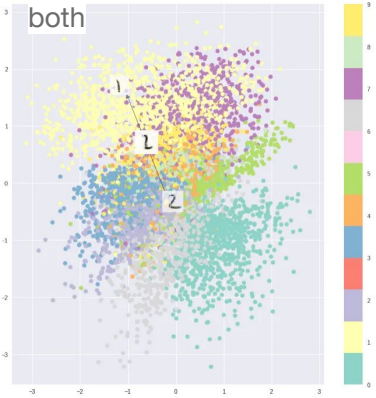
Функция потерь VAE

$$L_{VAE} = MSE(x, \hat{x}) + D_{KL}(\mathcal{N}(\mu_x, \sigma_x^2)||\mathcal{N}(0, I))$$

- reconstruction loss: $MSE(x, \hat{x})$
- KL divergence loss: $D_{KL}(\mathcal{N}(\mu_x, \sigma_x^2)||\mathcal{N}(0, I)) = -\frac{1}{2}(1 + \log \sigma_x^2 - \mu_x^2 - \sigma_x^2)$





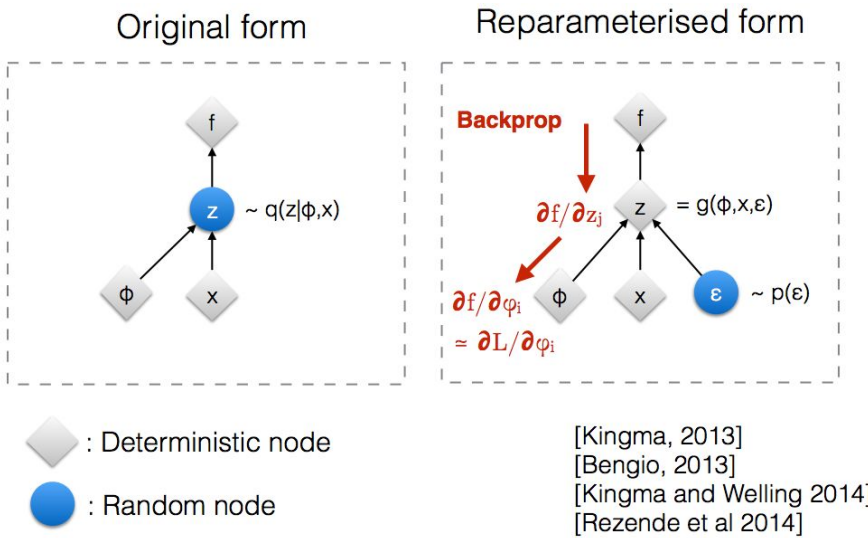


Reparametrization Trick

Проблема: сэмплирование $z \sim N(\mu, \sigma^2)$ мешает обратному распространению

Note: $N(\mu, \sigma^2) \sim \mu + \sigma * N(0, I)$

Трюк репараметризации
(Reparametrization trick):
Вместо сэмплирования $z \sim N(\mu, \sigma^2)$
Посемплируем заранее $\epsilon \sim N(0, I)$
и посчитаем детерминированно $z = \mu + \sigma * \epsilon$

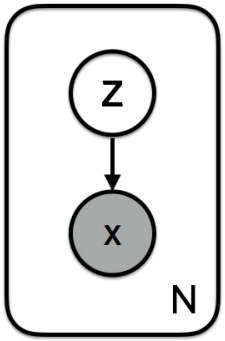


[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

VAE

VAE - вероятностная графовая модель
вариационный вывод
ELBO

Где почитать: link



Когда использовать VAE?

Плюсы:

- высокое разнообразие генерируемого
- быстрая генерация
- легко обучать

Минусы:

- низкое качество (размытость генерации)

VQ-VAE

Решение проблемы размытости - Vector Quantized VAE (VQ-VAE) [paper]

Идея: генерация из дискретного словаря латентных представлений (codebook)

Реализация:

- зафиксируем словарь векторов латентного пространства $e_1 \ldots e_n$
- маппим $z_e(x)$ на ближайший $e_k =: z_q(x)$ прежде чем декодировать

Как провести градиент через квантизацию: рассчитываем градиент по z_q, применяем его к z_e

Как выбрать словарь: обучаем вместе с энкодером (похоже на k-means)

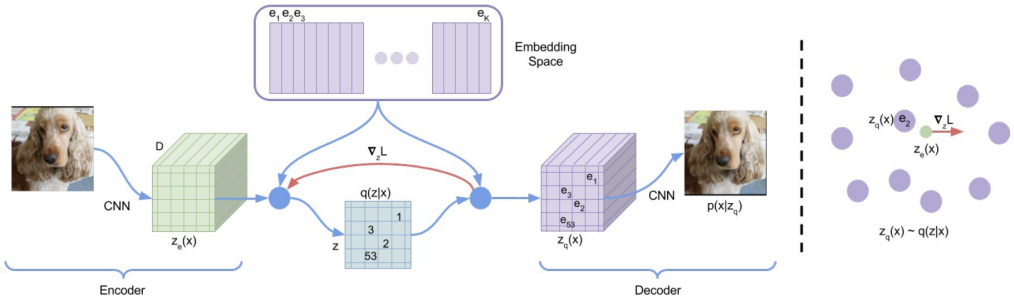


Figure 1: Left: A figure describing the VQ-VAE. Right: Visualisation of the embedding space. The output of the encoder $z(x)$ is mapped to the nearest point e_2. The gradient $\nabla_z L$ (in red) will push the encoder to change its output, which could alter the configuration in the next forward pass.

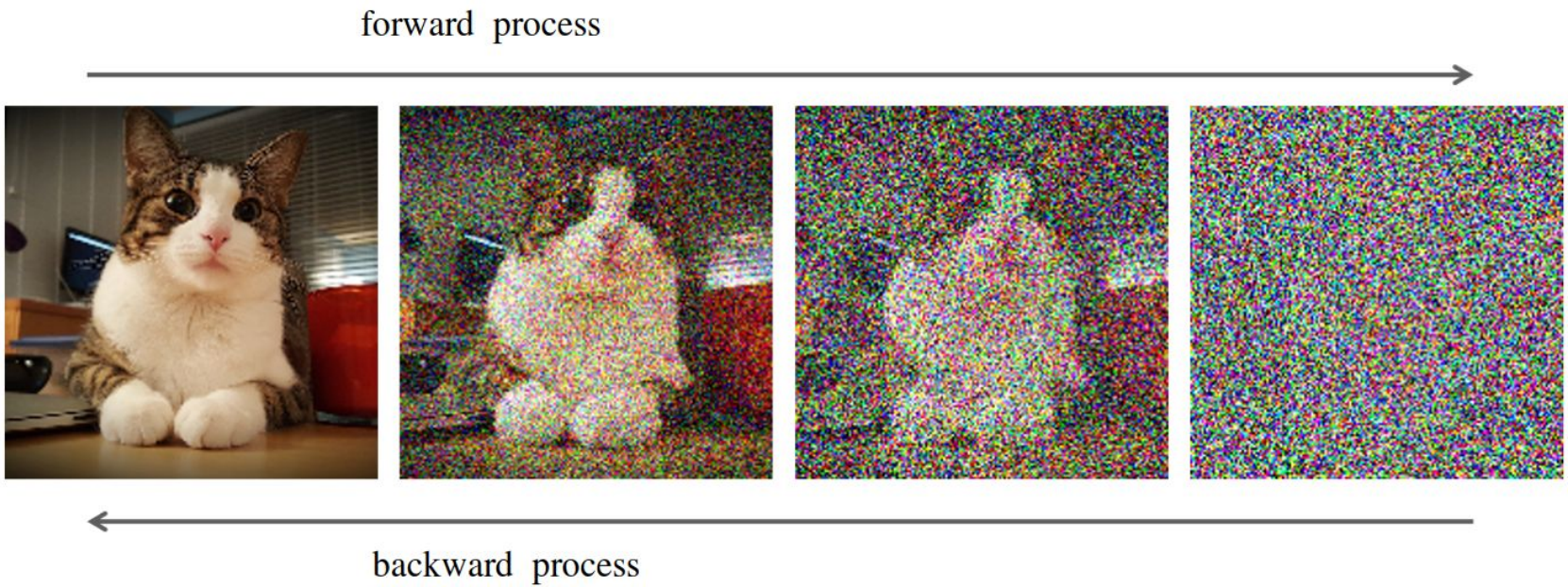
Лосс: sg[.] - stop gradient оператор

$$L_{VQ-VAE} = L_{VAE} + ||sg[z_e(x)] - z_q(x)||^2 + \beta||z_e(x) - sg[z_q(x)]||^2$$

Диффузия

Диффузия

Диффузия (diffusion)



Диффузия: forward process

Выбираем модель зашумления: марковская цепь

$$x_0 \to x_1 \to \ldots \to x_t \to \ldots \to x_T$$

$$x_0 \sim q(x_0),\ q(x_t|x_{t-1}) = \mathcal{N}(x_t|\sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I)$$

α_t берут < 1, по какому-то расписанию:

- убывающие
- cosine annealing

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

Диффузия: forward process

$$q(x_t|x_0) = ?$$

$$x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\varepsilon_{t-1} =$$

$$= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\varepsilon_{t-2} + \sqrt{1-\alpha_t}\varepsilon_{t-1} =$$

$$= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\varepsilon'_{t-2} =$$

$$= \ldots = \sqrt{A_t} x_0 + \sqrt{1-A_t}\varepsilon, \; A_t = \prod_{i=1}^{t} \alpha_i$$

Note: в пределе стандартное нормальное

Диффузия: backward process

Хотим: обратить шаги диффузии

Проблема: $q(x_{t-1}|x_t)$ не посчитать, т.к. зависит от $q(x_0)$

Решение: будем аппроксимировать $p_\theta(x_0) \sim q(x_0)$

Диффузия: backward process

Хотим: найти $p_\theta(x_0) \sim q(x_0)$

Для этого решаем задачу оптимизации: $D_{KL}(q(x_0)||p_\theta(x_0)) \underset{\theta}{\rightarrow} \min$

Похоже на VAE приходим к: $\mathbb{E}_q L_{VLB} \underset{\theta}{\rightarrow} \min$

где $L_{\text{VLB}} = L_T + L_{T-1} + \cdots + L_0$

$$L_T = D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))$$

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

$$L_0 = -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)$$

Диффузия: backward process

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

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

$$\tilde{\beta}_t = \frac{1 - A_{t-1}}{1 - A_t} \cdot \beta_t$$

$$\mu(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1 - A_t}}\varepsilon_t),$$

$$\text{где } x_t = \sqrt{A_t}x_0 + \sqrt{1 - A_t}\varepsilon_t$$

Будем искать $$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \tilde{\beta}_t I)$$

Диффузия: backward process

Лучше предсказывать не матожидание, а шум:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-A_t}}\varepsilon_\theta(x_t, t))$$

Соответственно поменяем L_t:

$$\begin{aligned} L_t^{\text{simple}} &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\varepsilon}_t}\left[\|\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\|^2\right] \\ &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\varepsilon}_t}\left[\|\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}_t, t)\|^2\right] \end{aligned}$$

Диффузия: алгоритм

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

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

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** $\mathbf{x}_0$

Условная генерация

Добавляем везде |y

Когда использовать диффузию?

Плюсы:

- высокое разнообразие генерируемого
- высокое качество
- легко обучать

Минусы:

- медленная генерация

Text-2-Image

Name	Year	Authors	Method	Link
DALL-E	2021	OpenAI	autoregressive	link
GLIDE	2021	OpenAI	diffusion	link
LDM	2021	Runway ML	diffusion	link
DALL-E 2	2022	OpenAI	diffusion	link
Stable Diffusion	2022	Runway ML	diffusion	link
Imagen	2022	Google	diffusion	link
Parti	2022	Google	autoregressive	link
Muse	2023	Google	transformers	link
StyleGAN-T	2023	NVIDIA	GAN	link
DALL-E 3	2023	OpenAI	diffusion	link

DALL-E

Обучающий датасет: пары image+text

Авторегрессионная модель:

- GPT-3 кодирует текстовые токены
- dVAE-энкодер сжимает изображение в токены
- трансформер-декодер авторегрессионно генерит по ним токены нового изображения
- dVAE-декодер преобразует токены в изображение

Дополнительно: выбирать лучшую генерацию используя CLIP

DALL-E

Достижения:

креативное

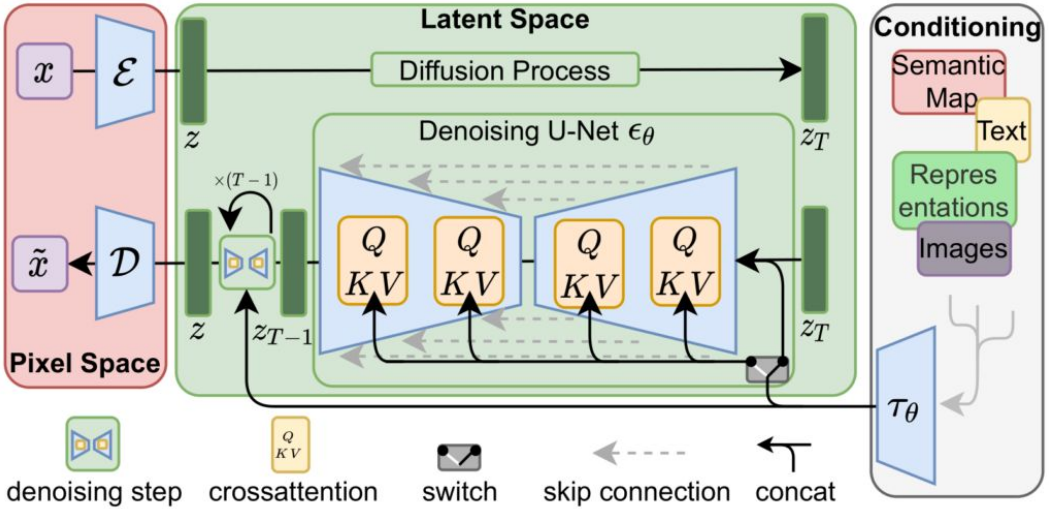
текст

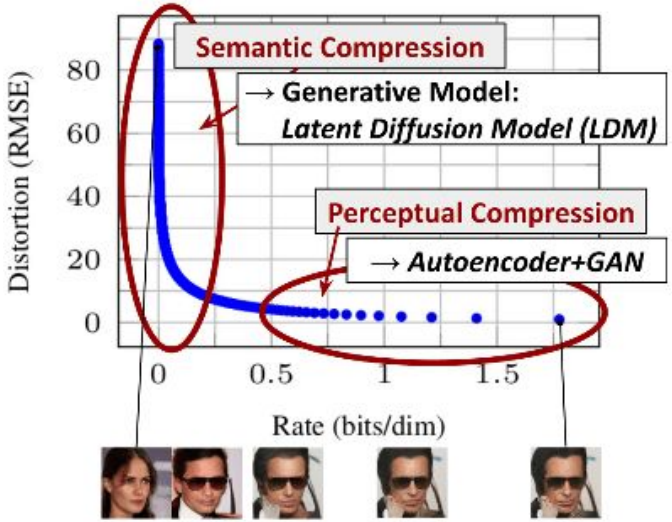
стили

LDM

Прорывная идея:

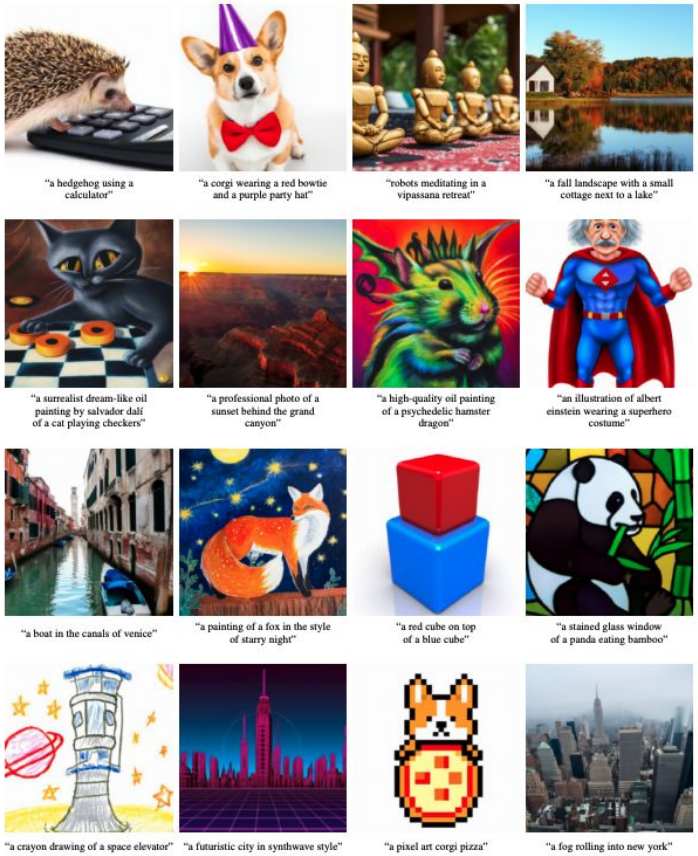
- использовать VAE на низком уровне (сжатие, генерация деталей)
- а диффузию только для генерации латентных представлений (семантики)





GLIDE

- диффузия
- Unet + глобальный аттеншн
- CLIP кодирует текст
- усиление влияния текста на генерацию
  - CLIP guidance
  - classifier-free guidance



Classifier Guided Diffusion

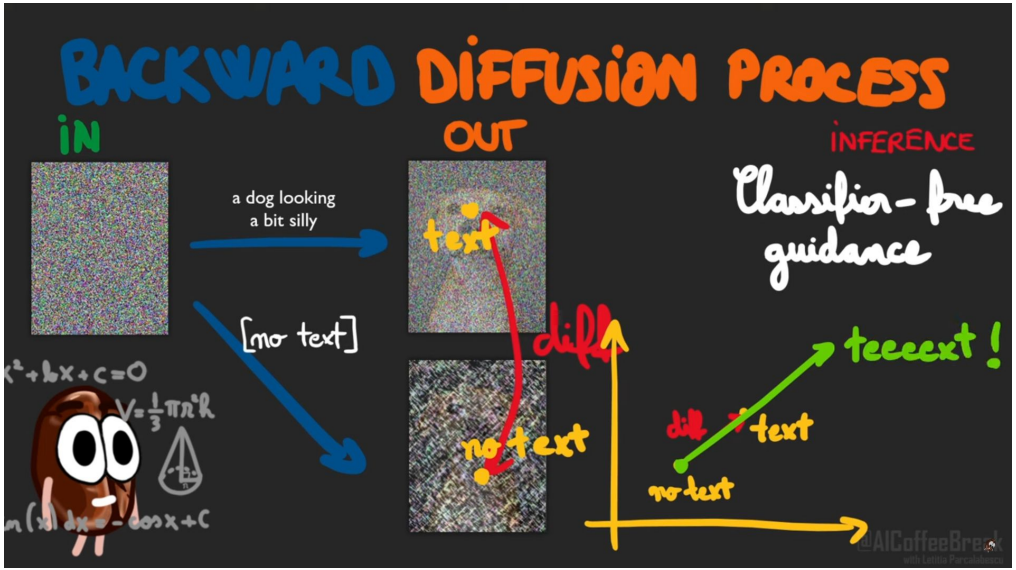
- обучить классификатор $f_\phi(y|\mathbf{x}_t, t)$
- сдвигать шаги генерации в сторону моды классификатора

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $f_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log f_\phi(y|x_t), \Sigma)$
end for
return x_0

Classifier-Free Guidance

- обучить условную диффузию
- обучить безусловную диффузию
- при сэмплинге сдвигать сильнее по вектору разности



source: https://www.youtube.com/watch?v=344w5h24-h8

Stable Diffusion

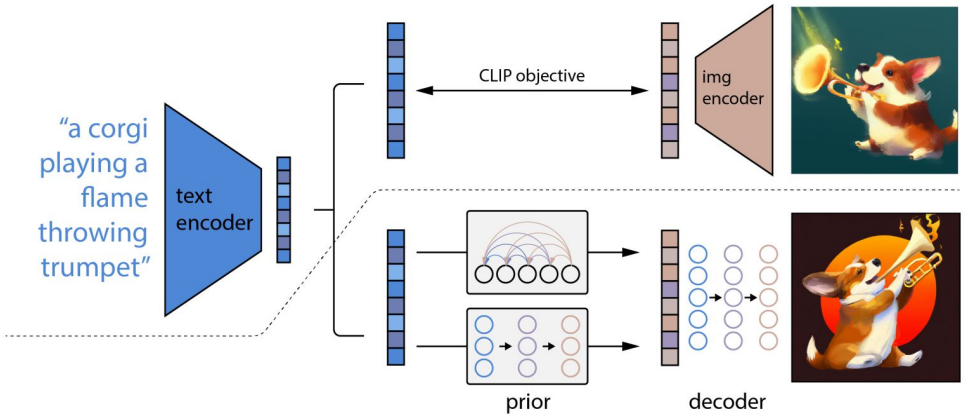
Та же LDM, но обученная на улучшенном датасете:
LAION-Aesthetics - подмножество особо красивых картинок из LAION-5B

DALL-E 2

Похоже на GLIDE

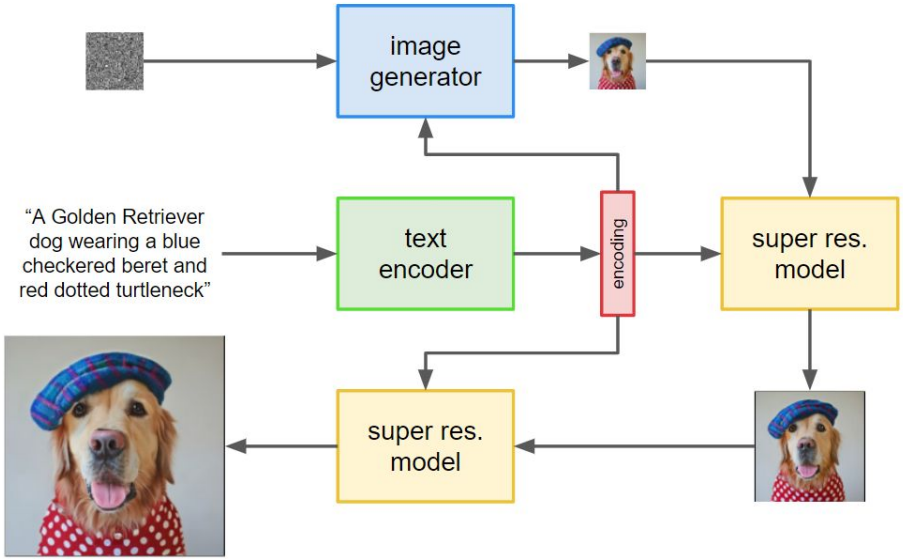
- CLIP: text-промпт -> text-эмбеддинг *y*
- text-эмбеддинг *y* -> image эмбеддинг *z*
  a. авторегрессионно
  b. диффузия

  } равно хороши
- диффузия: image эмбеддинг *z* -> изображение *x*



Imagen

- Похоже на GLIDE, но улучшили текстовые промпты:
  - GLIDE обучал свой текстовый энкодер
  - Imagen взял предобученную LLM T5-XXL
    (ее заморозили чтобы избежать оверфиттинга на тексты из обучающего датасета и сохранить генерализацию)
- super-resolution diffusion



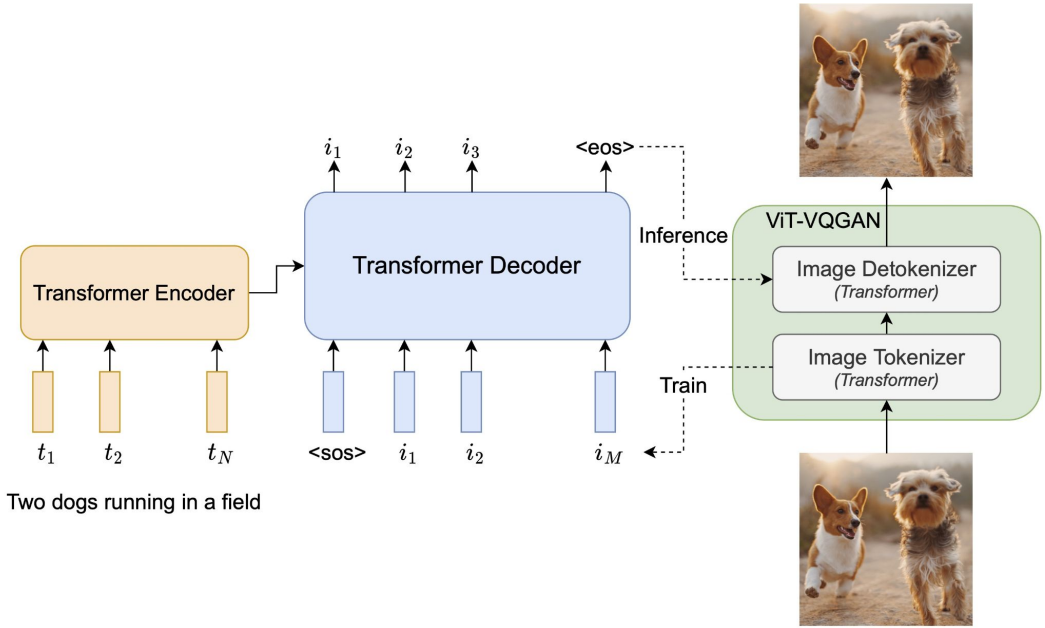
Imagen

Предложили бенчмарк из 200 промптов для тестирования разных аспектов **Drawbench**:

- подсчет объектов
- редкие слова
- длинные промпты
- креативные промпты
- …

Parti

- авторегрессионная модель из 2022
- похожа на dall-e (v1)
- image токенизация - ViT-VQGAN



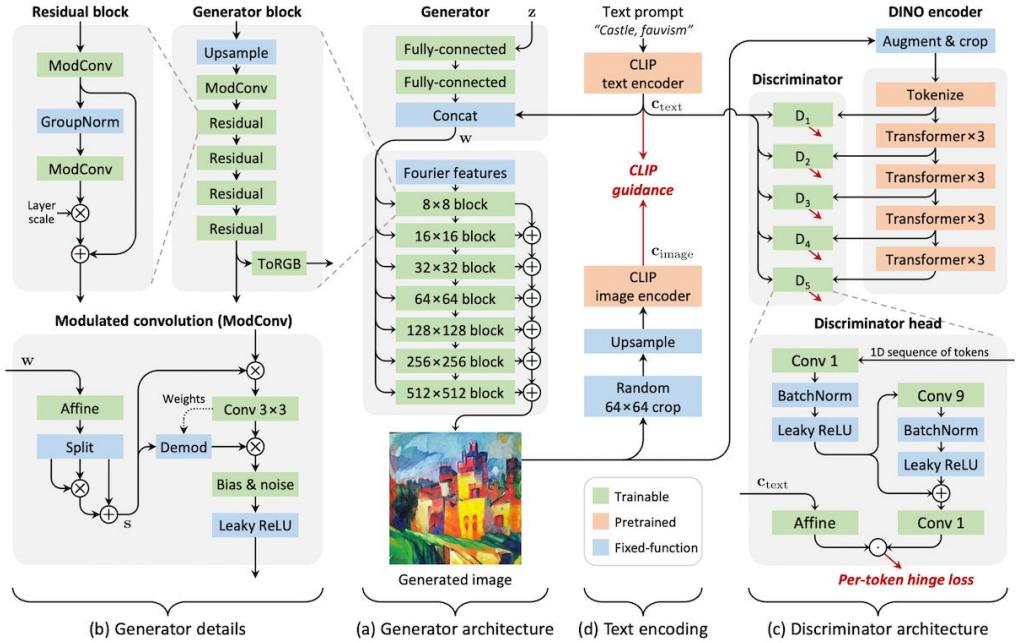
2023 - не только диффузии

StyleGAN-T: большой сложный GAN

Muse: трансформер

StyleGAN-T

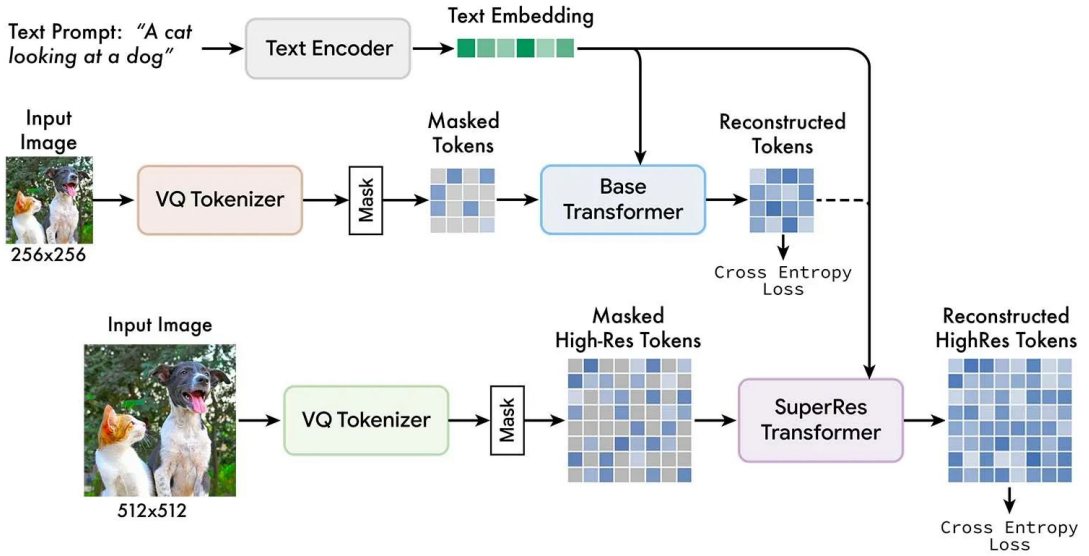
GANы все еще актуальны



Muse

Архитектура:

- T5-XXL: text -> text-токены
- VQ-GAN:
  - image -> image-токены
  - сгенеренные токены -> image
- трансформер:
  text-токены & image-токены ->
  сгенеренные токены



Обучение:

- часть входных image-токенов заменяется на [MASK]
- цель обучения - восстановить замаскированные токены

Инференс: все входные image-токены заменяются на [MASK]

DALL-E 3

Проблема: плохо генерируем по детализированным описаниям

Причина: датасеты из пар image + caption, caption - короткие тексты - т.е. нет подходящих обучающих данных

Решение: сгенерируем синтетические детализированные описания картинок и на этом обучим генеративную модель
(для этого потьюнили image captioning модель делать такие длинные описания - это проще, чем обучать генерацию картинок)

Note: из NLP есть подозрения, что у обучения на синтетике тоже есть потолок

Дополнительные материалы

Вводный курс по GANам от deeplearning.ai на coursera

VAE как вероятностная графовая модель link

Материалы по диффузии github

Посты Lilian Weng GAN VAE диффузия потоки

Спасибо за внимание!