ДИФФУЗИОННЫЕ МОДЕЛИ

Сергей Николенко

СПбГУ — Санкт-Петербург

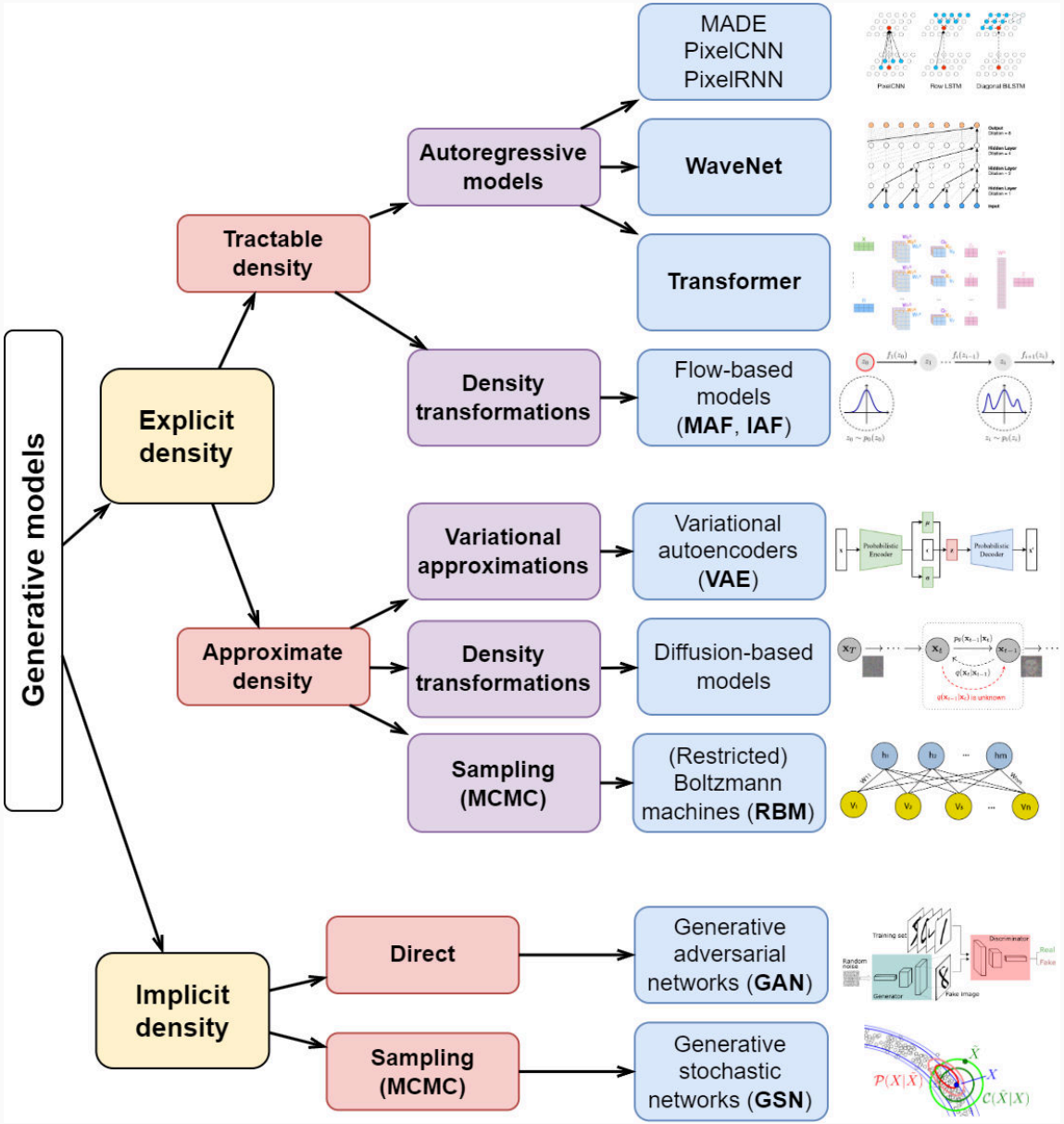
17 ноября 2025 г.

Random facts:

- 17 ноября — Международный день студентов (в память об аресте >1200 пражских студентов нацистами 17 ноября 1939 года); кроме того, 17 ноября — Международный день недоношенных детей, а в России — День участковых уполномоченных полиции
- 17 ноября 1875 г. Елена Блаватская и Генри Стил Олкотт основали в Нью-Йорке Теософское общество, в которое входили Томас Эдисон, отец Джавахарлала Неру Мотилал, Уильям Батлер Йейтс и другие; общество пришло в упадок после смерти Блаватской в 1891 г., но созданный в 1907 г. российский филиал процветал до 1918-го
- 17 ноября 1959 г. южноафриканская компания De Beers начала производство искусственных алмазов
- 17 ноября 1853 г. произошло первое в истории сражение пароходофрегатов: русский «Владимир» победил турецкий «Перваз-Бахри»
- 17 ноября 2018 г. американский миссионер Джон Чау погиб от рук аборигенов Андаманских островов; Чау считал их последним оплотом Сатаны на Земле, но вскоре после высадки был застрелен из лука, посмертно получив за это премию Дарвина
- 17 ноября 2019 г. в Китае выявили первого инфицированного COVID-19

Порождающие модели в глубоком обучении

Глубокие порождающие модели



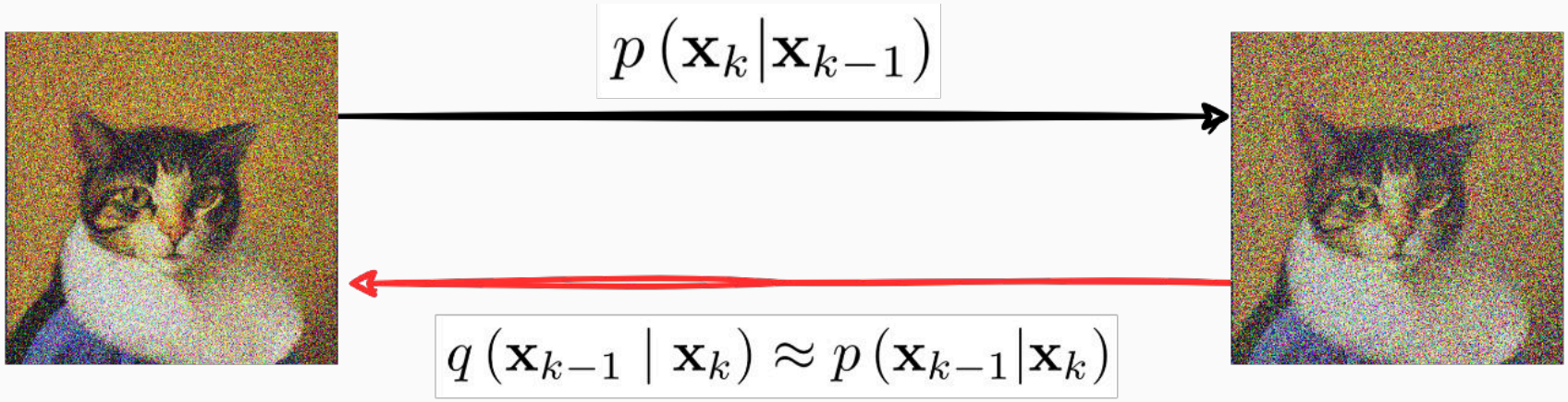
Диффузионные модели

Диффузионные модели

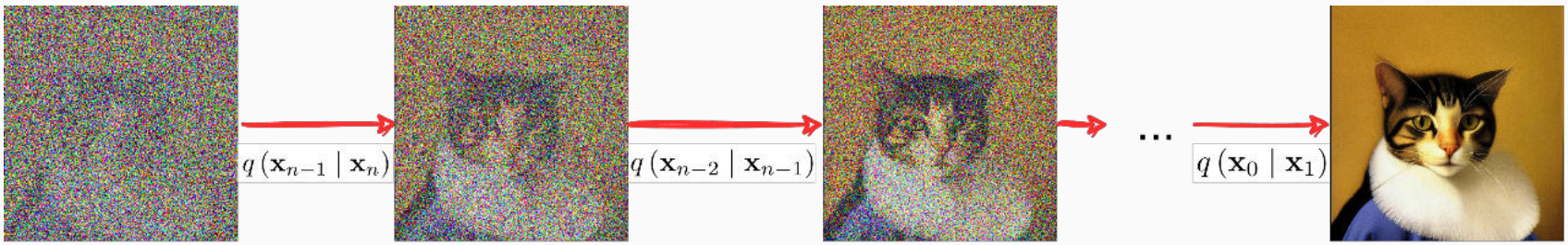
- (Sohl-Dickstein et al., 2015): Deep Unsupervised Learning using Nonequilibrium Thermodynamics
- Идея из статистической физики: давайте обучать марковскую цепь, которая постепенно сделает из случайного шума нужное распределение
- Для этого начнём с цепи, которая сделает из нужного распределения случайный шум:

Диффузионные модели

- А потом попробуем обратить каждый шаг:



- Если получится, то потом мы можем начать со случайного шума и двигаться обратно:



Диффузионные модели

- Forward diffusion: начнём с $p(\mathbf{x}_0)$ и будем добавлять гауссовский шум:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = N\left(\mathbf{x}_t \middle| \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right).$$

- Тогда постепенно $\mathbf{x}_0$ превращается в шум, скажем, за T шагов:

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}).$$

Диффузионные модели

- И для каждого t мы можем просэмплировать $\mathbf{x}_t$ в замкнутой форме через reparametrization trick: пусть $\epsilon \sim N(0, \mathbf{I})$ и обозначим $\alpha_t = 1 - \beta_t$, $A_t = \prod_{i=1}^{T} \alpha_i$; тогда

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon = \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon = \ldots$$

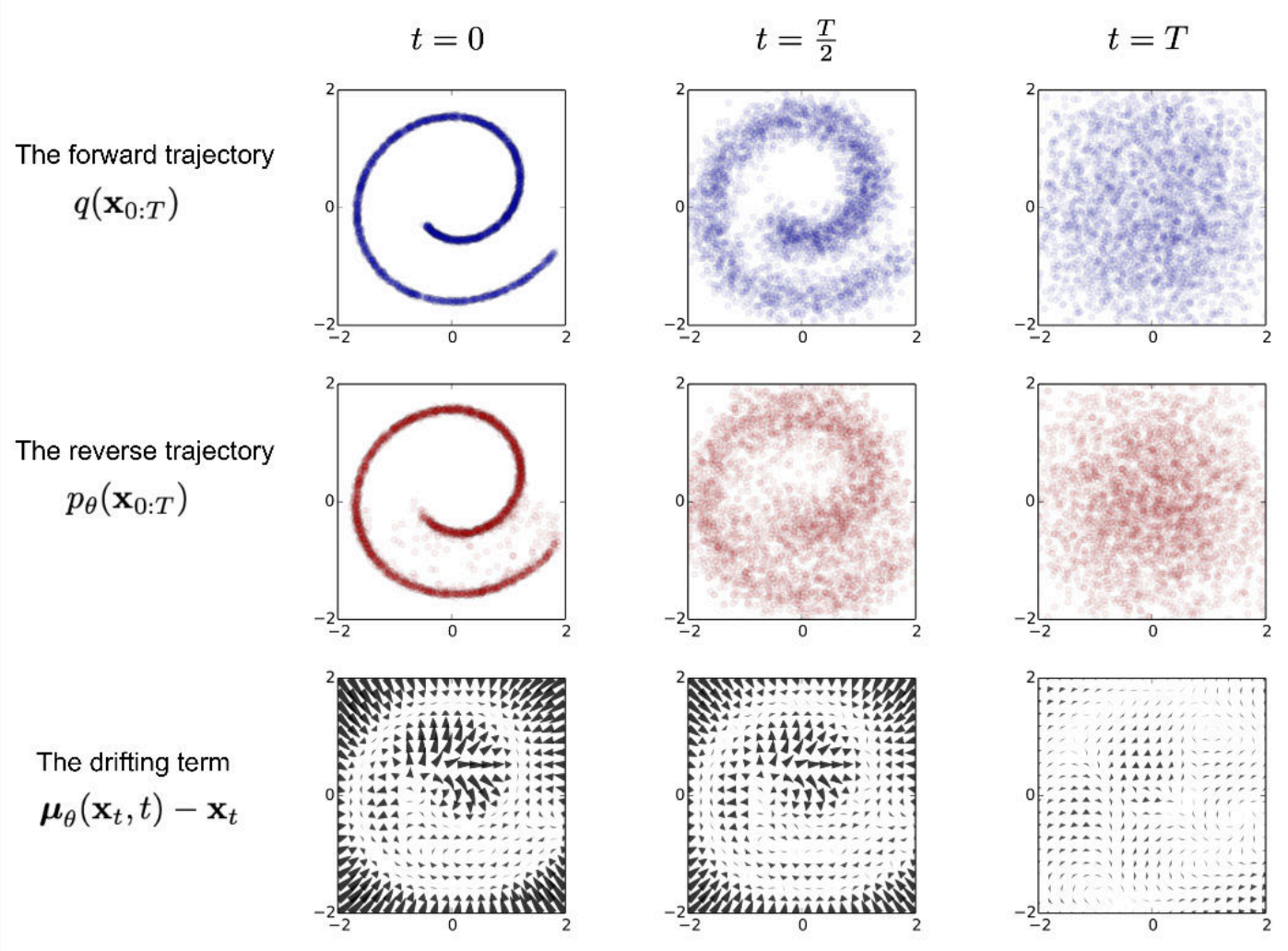
$$\ldots = \sqrt{A_t}\mathbf{x}_0 + \sqrt{1-A_t}\epsilon, \text{ то есть } q(\mathbf{x}_t \mid \mathbf{x}_0) = N\left(\mathbf{x}_t \middle| \sqrt{A_t}\mathbf{x}_0, (1-A_t)\mathbf{I}\right)$$

потому что когда мы складываем два гауссиана $N(0, \sigma_1^2\mathbf{I})$ и $N(0, \sigma_1^2\mathbf{I})$, получается гауссиан $N(0, (\sigma_1^2 + \sigma_2^2)\mathbf{I})$, и у нас это

$$(1-\alpha_t) + \alpha_t(1-\alpha_t) = 1 - \alpha_t\alpha_{t-1}.$$

Диффузионные модели

- А теперь задача, казалось бы, в том, чтобы обратить этот процесс, научиться сэмплировать из $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)$



Диффузионные модели

- Для маленьких β_t обратные распределения тоже будут гауссовскими, но параметры их теперь уже зависят от точек данных.
- Reverse diffusion: теперь уже надо обучать распределения

$$p_\theta\left(\mathbf{x}_{0:T}\right) = p_\theta\left(\mathbf{x}_T\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right), \quad \text{где}$$

$$p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) = N\left(\mathbf{x}_{t-1} \middle| \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)\right).$$

Диффузионные модели

- Если бы мы знали $\mathbf{x}_0$, то обратить было бы можно аналитически, кстати:

$$q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right) \frac{q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_0\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)} \propto$$

$$\propto e^{-\frac{1}{2}\left(\frac{\left(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1}\right)^2}{\beta_t} + \frac{\left(\mathbf{x}_{t-1} - \sqrt{A_{t-1}}\mathbf{x}_0\right)^2}{1-A_{t-1}} + \frac{\left(\mathbf{x}_t - \sqrt{A_t}\mathbf{x}_0\right)^2}{1-A_t}\right)},$$

и в этой формуле теперь надо выделить полный квадрат по $\mathbf{x}_{t-1}$...

Диффузионные модели

- Получится

$$q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = N\left(\mathbf{x}_{t-1} \middle| \tilde{\mu}\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right), \quad \text{где}$$

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

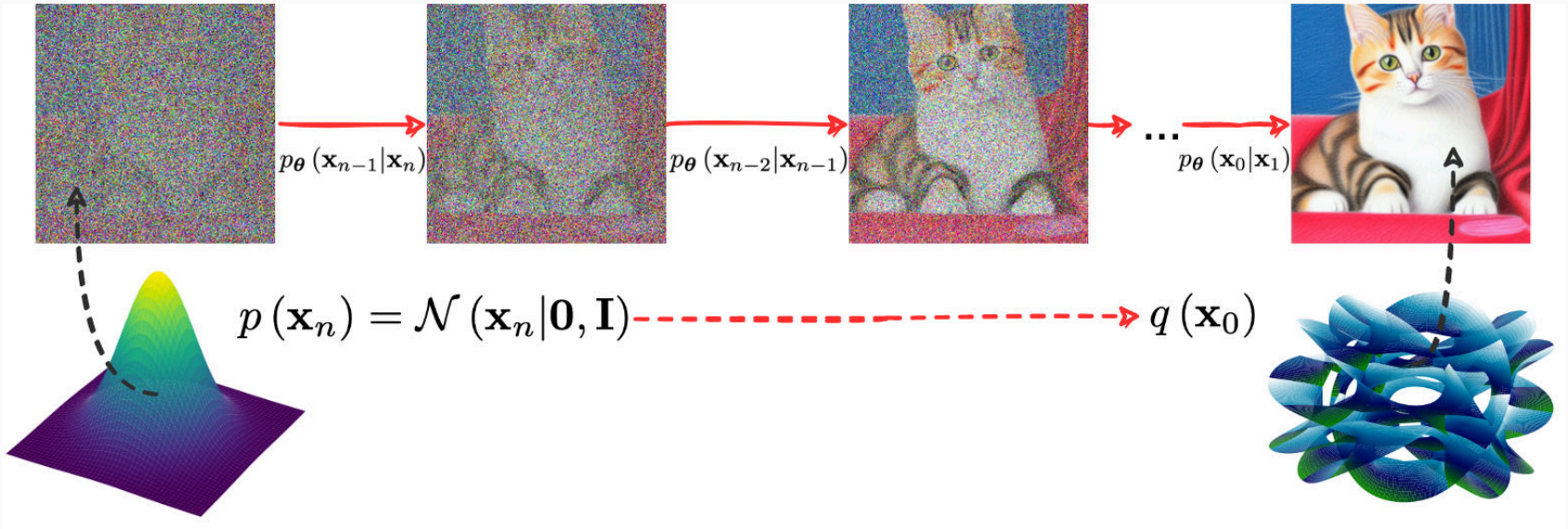
$$\tilde{\mu}\left(\mathbf{x}_t, \mathbf{x}_0\right) = \frac{\sqrt{\alpha_t}\left(1 - A_{t-1}\right)}{1 - A_t}\mathbf{x}_t + \frac{\sqrt{A_{t-1}}\beta_t}{1 - A_t}\mathbf{x}_0,$$

и поскольку мы знаем, что $\mathbf{x}_t = \sqrt{A_t}\mathbf{x}_0 + \sqrt{1 - A_t}\epsilon$,

$$\tilde{\mu}\left(\mathbf{x}_t, \mathbf{x}_0\right) = \frac{1}{\sqrt{A_t}}\left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - A_t}}\epsilon_t\right).$$

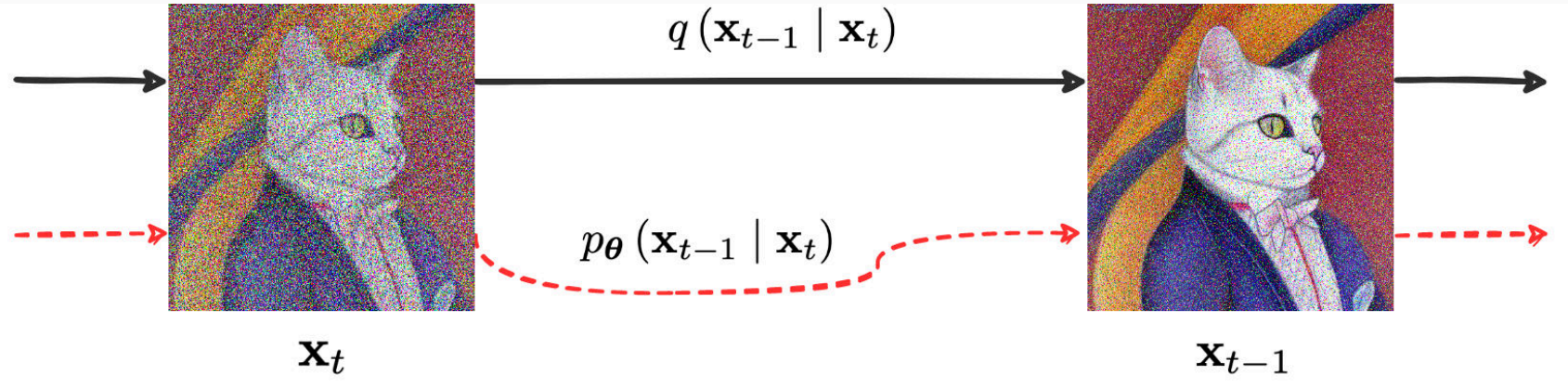
Диффузионные модели

- В конечном счёте хочется, чтобы обратная диффузия реконструировала $q(\mathbf{x}_0)$ из стандартного входа в $\mathbf{x}_n$; примерно так:



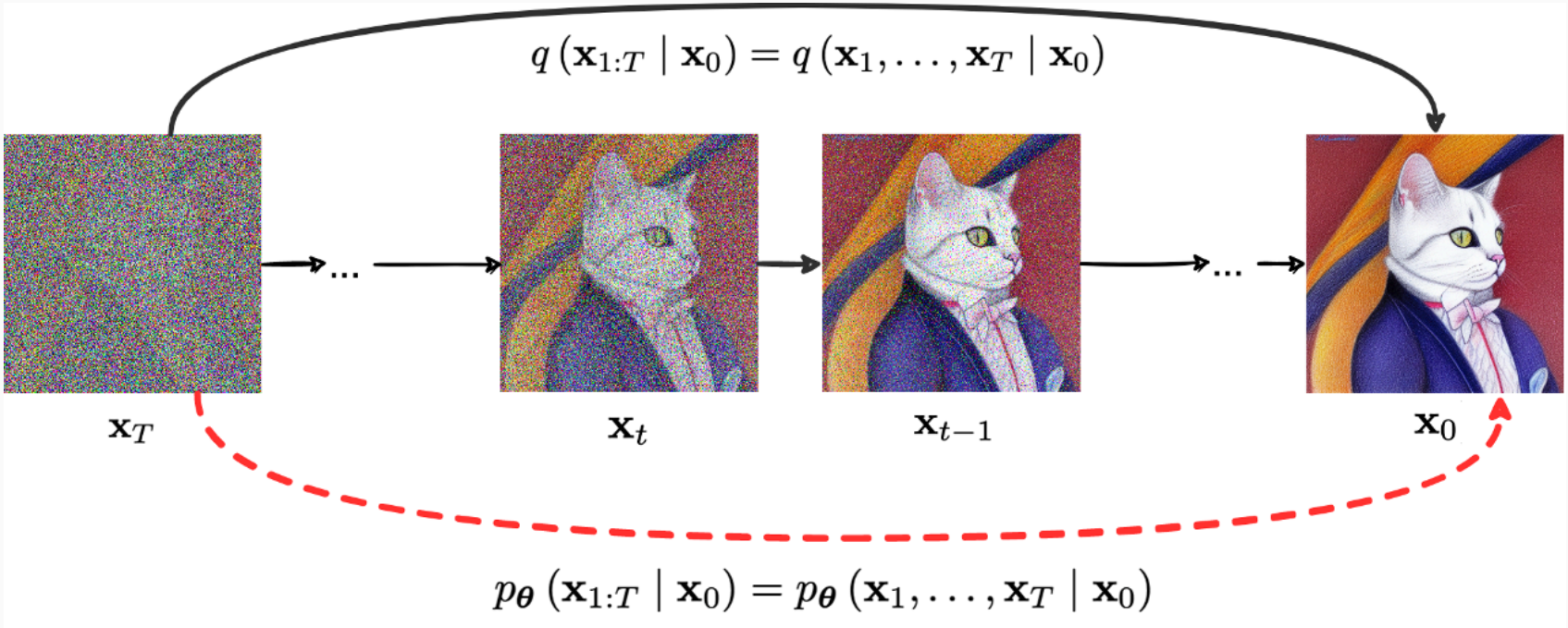
Диффузионные модели

- Что делать? Как всегда в байесовском выводе, приближать!
- На каждом шаге модель должна стать хорошим приближением к $q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)$, без условия на неизвестное $\mathbf{x}_0$:



Диффузионные модели

- Чтобы получить эту аппроксимацию, нужна вариационная нижняя оценка, похожая на ту, которая была в VAE и DALL-E
- Начинаем с оценки на глобальное распределение $q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right) = q\left(\mathbf{x}_1, \ldots, \mathbf{x}_T \mid \mathbf{x}_0\right)$:



- А потом она разложится на оценки для отдельных шагов диффузионного процесса

Диффузионные модели

- Вспомним идею вариационных приближений:

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x})\, p(\mathbf{z}|\mathbf{x}),$$

$$\log p(\mathbf{x}) = \log p(\mathbf{x}, \mathbf{z}) - \log p(\mathbf{z}|\mathbf{x}),$$

возьмём ожидание по $q(\mathbf{z})$:

$$\mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x})] = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z})] - \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{z}|\mathbf{x})],$$

а затем сделаем стандартные преобразования:

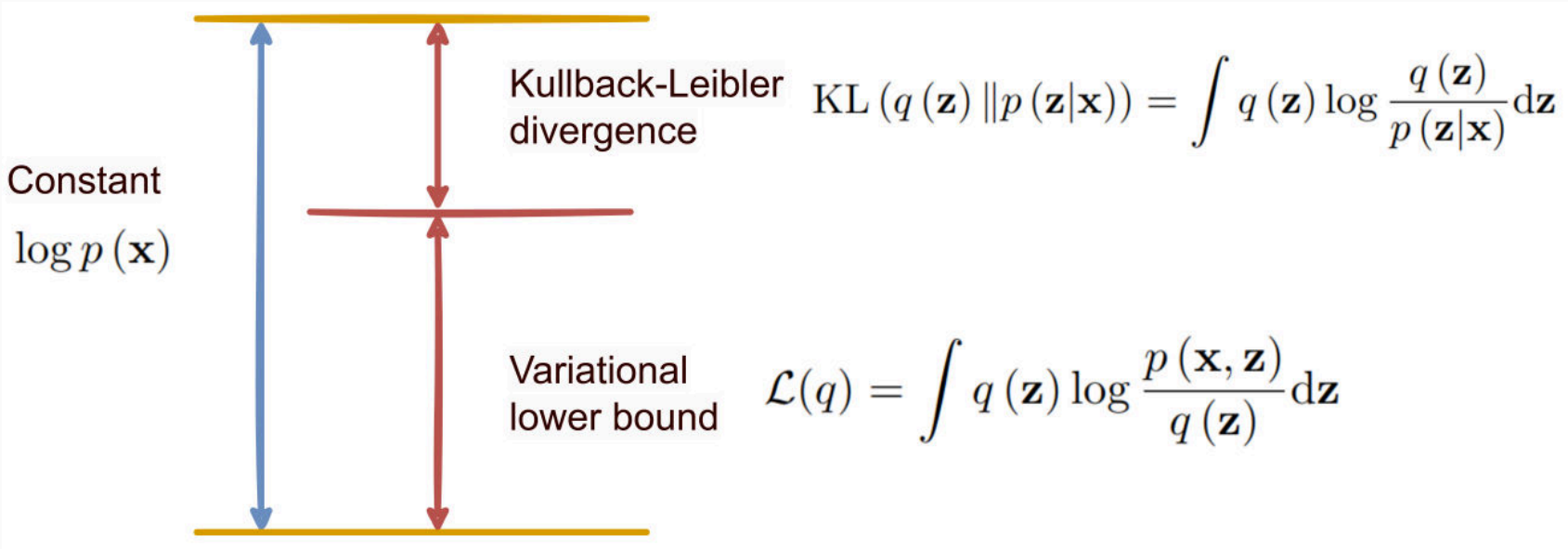
$$\begin{aligned}\log p(\mathbf{x}) =& \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] + \\ &+ \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] - \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{z}|\mathbf{x})], \\ \log p(\mathbf{x}) =& \int q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} d\mathbf{z} + \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}.\end{aligned}$$

Диффузионные модели

- Итого получается, что

$$\log p\left(\mathbf{x}\right) = L(q) + \mathrm{KL}\left(q\left(\mathbf{z}\right) \| p\left(\mathbf{z}|\mathbf{x}\right)\right), \quad \text{где}$$

$$L(q) = \int q\left(\mathbf{z}\right) \log \frac{p\left(\mathbf{x}, \mathbf{z}\right)}{q\left(\mathbf{z}\right)} \mathrm{d}\mathbf{z}.$$



Диффузионные модели

- Давайте для диффузионной модели выведем вариационную оценку снова из первых принципов:

$$\log p_\theta(\mathbf{x}_0) = \log p_\theta(\mathbf{x}_0, \dots, \mathbf{x}_T) - \log p_\theta(\mathbf{x}_1, \dots, \mathbf{x}_T \mid \mathbf{x}_0),$$

$$\mathbb{E}_{q(\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0)] = \mathbb{E}_{q(\mathbf{x}_{0:T})}[\log p_\theta(\mathbf{x}_{0:T})] - \mathbb{E}_{q(\mathbf{x}_{0:T})}[\log p_\theta(\mathbf{x}_{1:T} \mid \mathbf{x}_0)] =$$
$$= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}\right] + \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{p_\theta(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}\right].$$

- Здесь второе слагаемое — это KL-дивергенция, и мы получаем функцию ошибки как минус вариационную нижнюю оценку:

$$L = \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \geq -\mathbb{E}_{q(\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0)].$$

Диффузионные модели

- Эту оценку можно подсчитать как сумму:

$$
\begin{aligned}
L =& \mathbb{E}_q \left[\log \frac{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_{0:T}\right)} \right] = \mathbb{E}_q \left[\log \frac{\prod_{t=1}^{T} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)}{p_\theta\left(\mathbf{x}_T\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)} \right] \\
=& \mathbb{E}_q \left[-\log p_\theta\left(\mathbf{x}_T\right) + \sum_{t=1}^{T} \log \frac{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)}{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)} \right] \\
=& \mathbb{E}_q \left[-\log p_\theta\left(\mathbf{x}_T\right) + \sum_{t=2}^{T} \log \frac{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)}{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)} + \log \frac{q\left(\mathbf{x}_1 \mid \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)} \right] \\
=& \mathbb{E}_q \left[-\log p_\theta\left(\mathbf{x}_T\right) + \sum_{t=2}^{T} \log \left(\frac{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)} \frac{q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)}{q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_0\right)} \right) + \log \frac{q\left(\mathbf{x}_1 \mid \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)} \right] \\
=& \mathbb{E}_q \left[-\log p_\theta\left(\mathbf{x}_T\right) + \sum_{t=2}^{T} \log \frac{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)} + \log \frac{q\left(\mathbf{x}_T \mid \mathbf{x}_0\right)}{q\left(\mathbf{x}_1 \mid \mathbf{x}_0\right)} + \log \frac{q\left(\mathbf{x}_1 \mid \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)} \right] \\
=& \mathbb{E}_q \left[\log \frac{q\left(\mathbf{x}_T \mid \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_T\right)} + \sum_{t=2}^{T} \log \frac{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)} - \log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right) \right].
\end{aligned}
$$

Диффузионные модели

- Итого получается

$$
\begin{aligned}
L =& L_T + L_{T-1} + \ldots + L_0, \quad \text{where} \\
L_T =& \mathrm{KL}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_T\right)\right), \\
L_t =& \mathrm{KL}\left(q\left(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_t \mid \mathbf{x}_{t+1}\right)\right), \quad t = 1, \ldots, T-1, \\
L_0 =& -\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right).
\end{aligned}
$$

Диффузионные модели

- Всё это KL-дивергенции между гауссианами, их можно подсчитать и подставить в функцию ошибки
- Например, в L_t мы используем гауссовскую параметризацию

$$p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) = N\left(\mathbf{x}_{t-1} \middle| \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)\right)$$

и пытаемся её параметры совместить с $q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right)$. Для среднего, например,

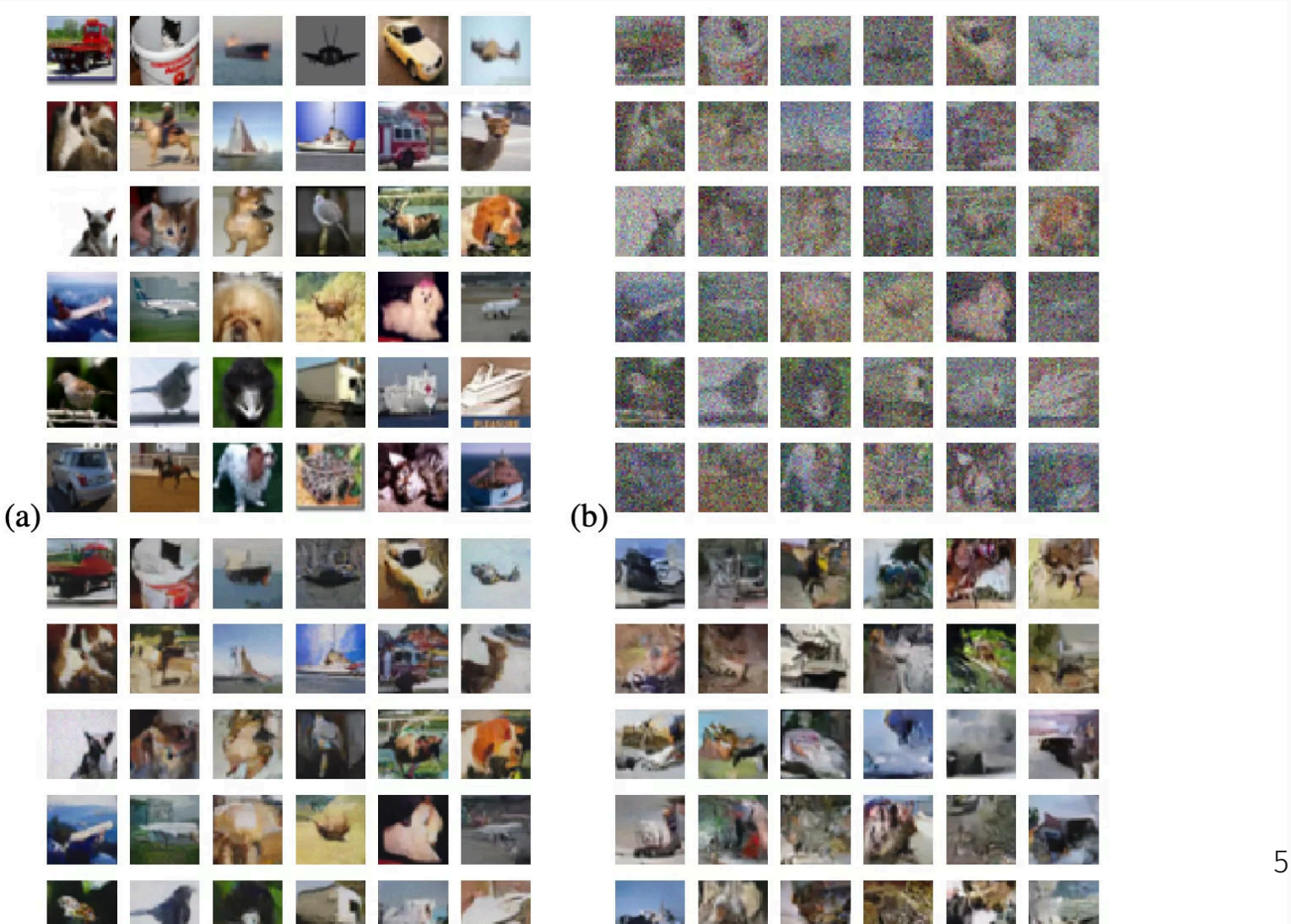
$$\mu_\theta(\mathbf{x}_t, t) \approx \tilde{\mu}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) = \frac{1}{\sqrt{A_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-A_t}}\epsilon_t\right),$$

и поскольку мы знаем $\mathbf{x}_t$ при обучении, мы можем параметризовать шум напрямую:

$$p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) = N\left(\mathbf{x}_{t-1} \middle| \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-A_t}}\epsilon_\theta(\mathbf{x}_t, t)\right), \Sigma_\theta(\mathbf{x}_t, t)\right).$$

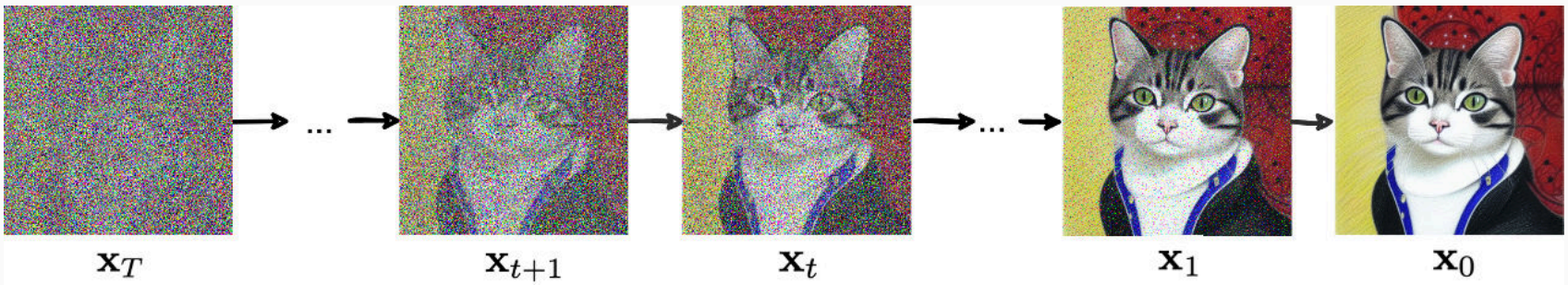
Диффузионные модели

- (Sohl-Dickstein et al., 2015) само по себе не слишком хорошо работало, разве что на CIFAR:



DDPM

- Следующий шаг: “Denoising Diffusion Probabilistic Models” (DDPM; Ho et al., 2020)
- Та же основная идея, та же структура функции ошибки:



DDPM

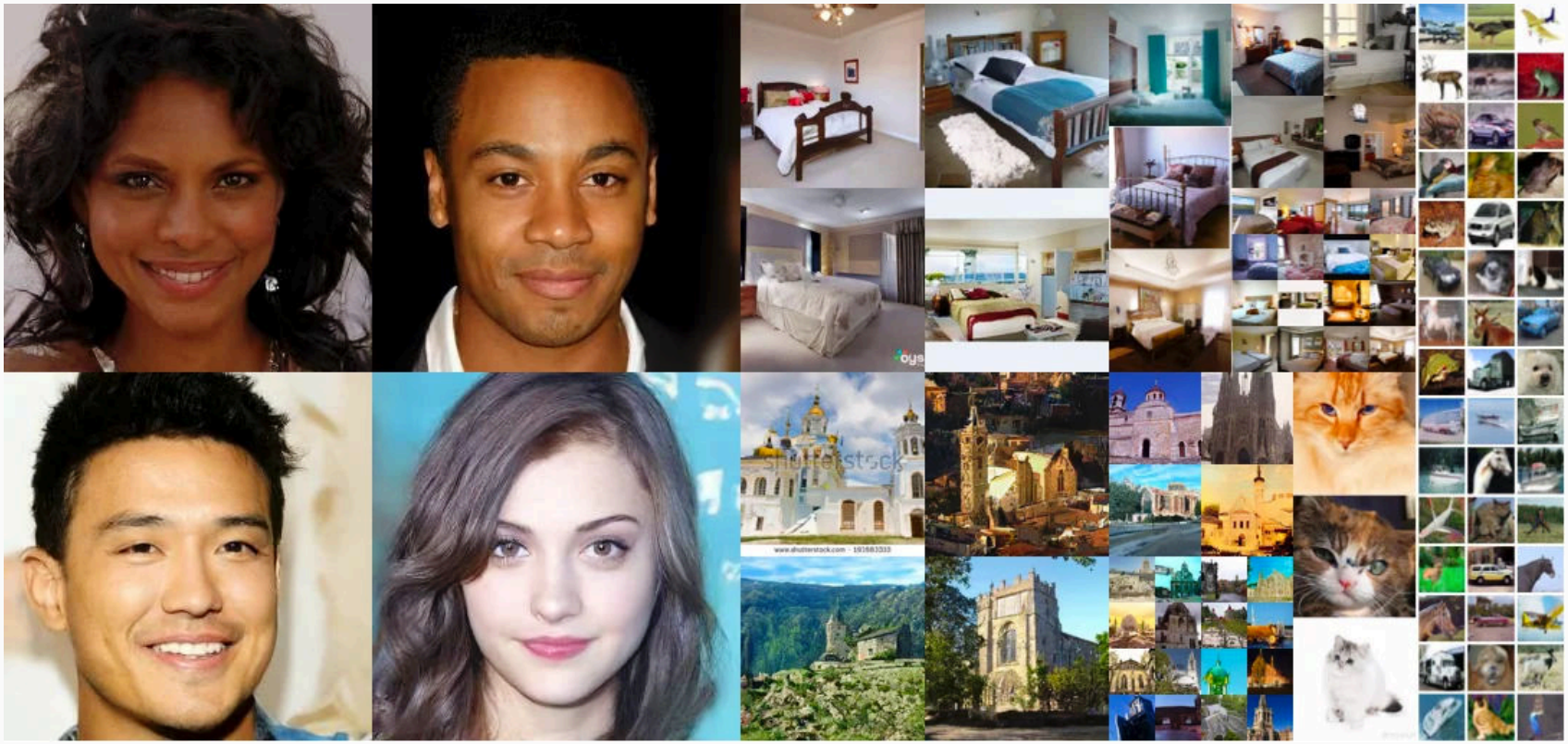
- Но дальше DDPM делает следующие замечания:
  - дисперсии прямой диффузии β_t — постоянные гиперпараметры, их не обучают, так что все q не обучаются; т.к. $p_\theta(\mathbf{x}_T)$ — фиксированное распределение, из которого мы будем сэмплировать, L_T является константой;
  - для промежуточных шагов дисперсии в $p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1})$ тоже не обучаются, и можно найти простую замкнутую форму для L_t;
  - главное — давайте введём отдельный дискретный декодер для L_0; если данные состоят из целых чисел от 0 до 255, отмасштабированных в $[-1, 1]$ (это естественное представление для картинок), DDPM моделирует

$$p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1) = \prod_{i=1}^{D} \int_{\delta_-(x_0, i)}^{\delta_+(x_0, i)} N\left(x \middle| \mu_{\theta,i}(\mathbf{x}_1), \sigma_1^2\right) \mathrm{d}x,$$

    где i идёт по пикселям, $\mu_\theta(\mathbf{x}_1)$ — декодер, пределы интегрирования $\frac{1}{255}$ в каждую сторону от $x_{0,i}$
- Т.е. можно подставить другую модель на последнем шаге и использовать $\mu_\theta(\mathbf{x}_1)$ без шума во время сэмплирования

DDPM

- DDPM уже давала отличное порождение, сравнимое с лучшими GAN'ами того времени:

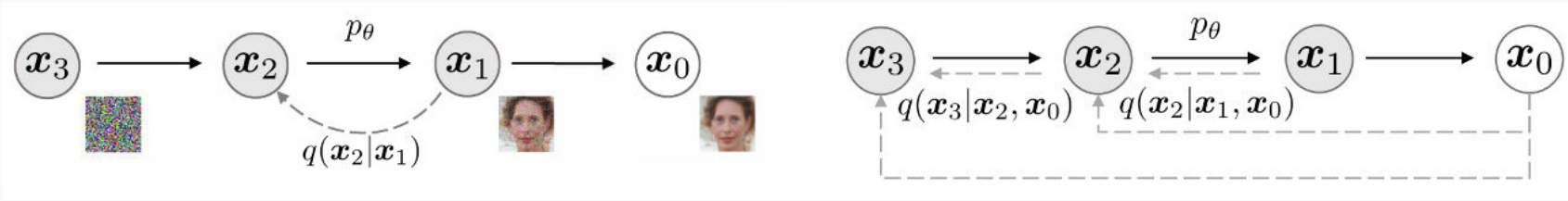


DDIM

- Следующий шаг пришёл очень скоро: “Denoising Diffusion Implicit Models” (DDIM; Song et al., 2020)
- Важный недостаток всех диффузионных моделей пока что в том, что они *очень медленные*
- Порождение должно пройти по всем шагам диффузии, а их буквально тысячи, и они идут последовательно, не параллелизуются
- Song et al. упоминают, что сэмплирование из обученного GAN примерно в 1000 раз быстрее, чем сэмплирование из DDPM для того же размера картинок

DDIM

- Как можно ускорить диффузионные модели?
- Song et al. обобщают диффузионные модели и в частности DDPM
- Функция ошибки L не зависит напрямую от совместного распределения $q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)$, а только от маргиналов $q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)$
- Это значит, что мы можем переиспользовать ту же функцию ошибки для другого совместного распределения, если у него те же маргиналы
- В частности, обобщить можно даже на немарковские диффузионные процессы, если для них получится найти обратную марковскую цепь:



DDIM

- Для DDIM мы определим диффузионный процесс в терминах его обратной формы:

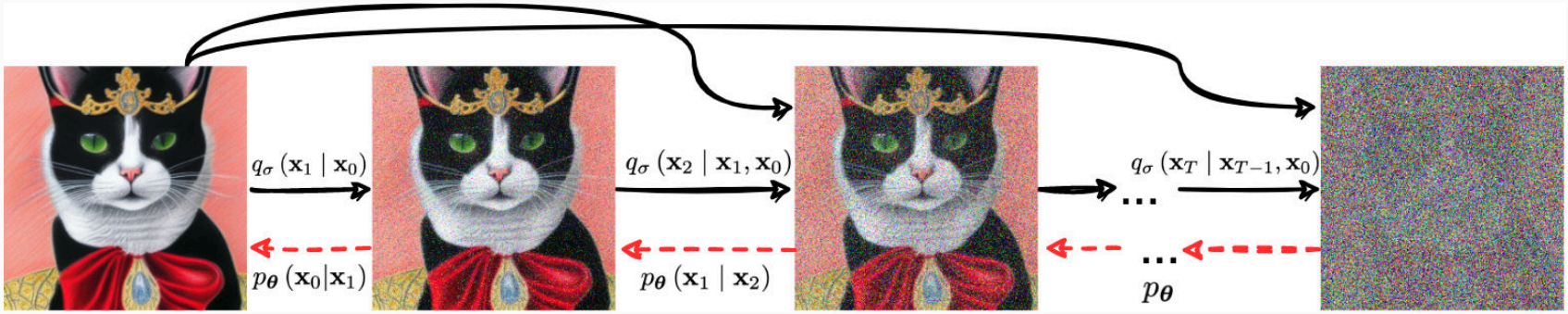
$$q_\sigma\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right) = q_\sigma\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \prod_{t=2}^{T} q_\sigma\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right).$$

- Теперь можно выразить распределения прямой диффузии по теореме Байеса:

$$q_\sigma\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right) = \frac{q_\sigma\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) q_\sigma\left(\mathbf{x}_t \mid \mathbf{x}_0\right)}{q_\sigma\left(\mathbf{x}_{t-1} \mid \mathbf{x}_0\right)}.$$

DDIM

- И теперь можно показать, что у полученного процесса ровно те же маргиналы, так что обратную диффузию можно обучать с той же функцией ошибки, и она будет марковской цепью:

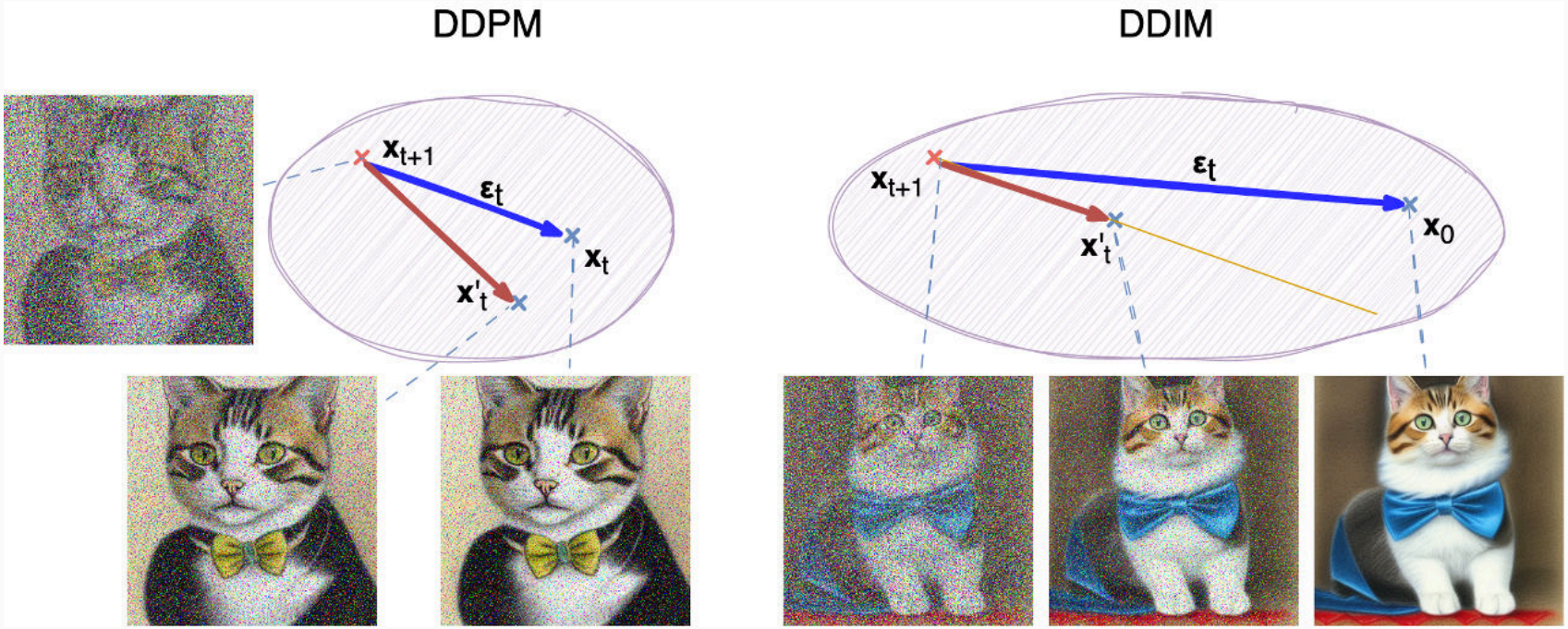


DDIM

- Пока это не очень помогает: мы расширили класс распределений для прямой диффузии, но сэмплирование новой картинки всё равно должно пройти через все шаги обратной диффузии
- Но теперь, вместо того чтобы приближать случайный шум ϵ_t, который ведёт от $\mathbf{x}_t$ к $\mathbf{x}_{t+1}$, можно приближать случайный шум ϵ_t, который приведёт сразу от $\mathbf{x}_0$ к $\mathbf{x}_{t+1}$!
- И когда мы идём в обратном направлении, мы приближаем направление не к $\mathbf{x}_t$, а сразу к $\mathbf{x}_0$, и двигаться в эту сторону можно быстрее

DDIM

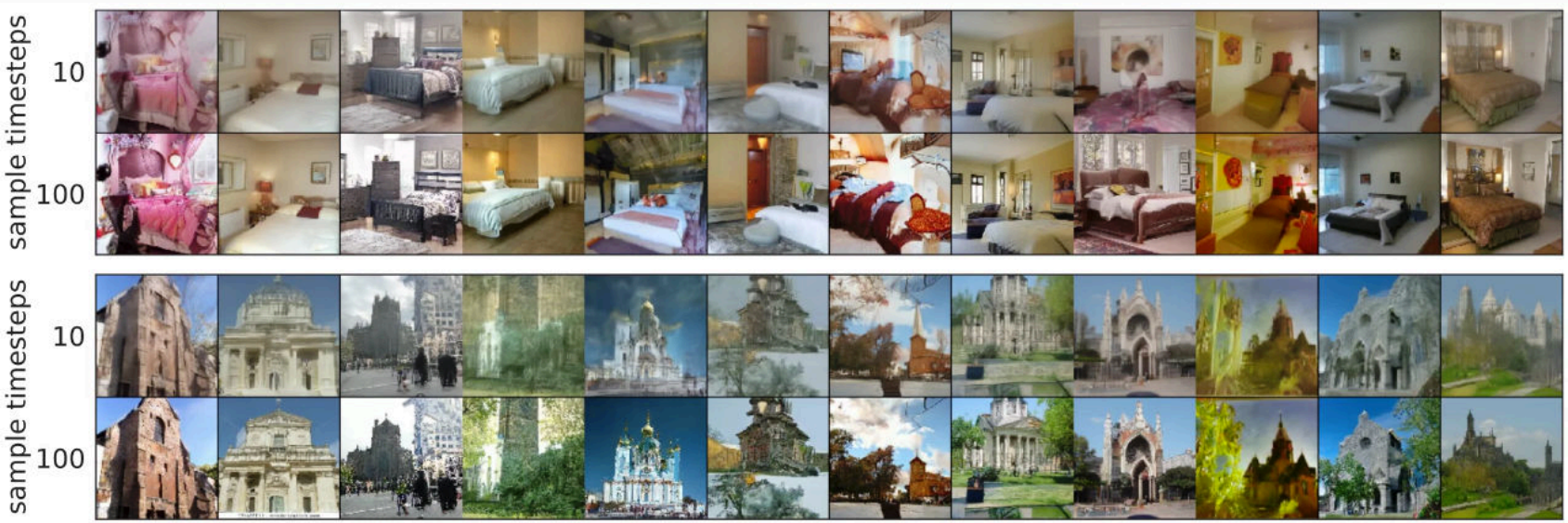
- Иллюстрация:



- Сложно сказать, какое приближение лучше, но теперь мы разделили зависимости ϵ_t от $\mathbf{x}_0$, и мы можем перепрыгнуть на несколько шагов сразу, двигаясь от $\mathbf{x}_t$ к $\mathbf{x}_{t+k}$ за один шаг с увеличенным ϵ!

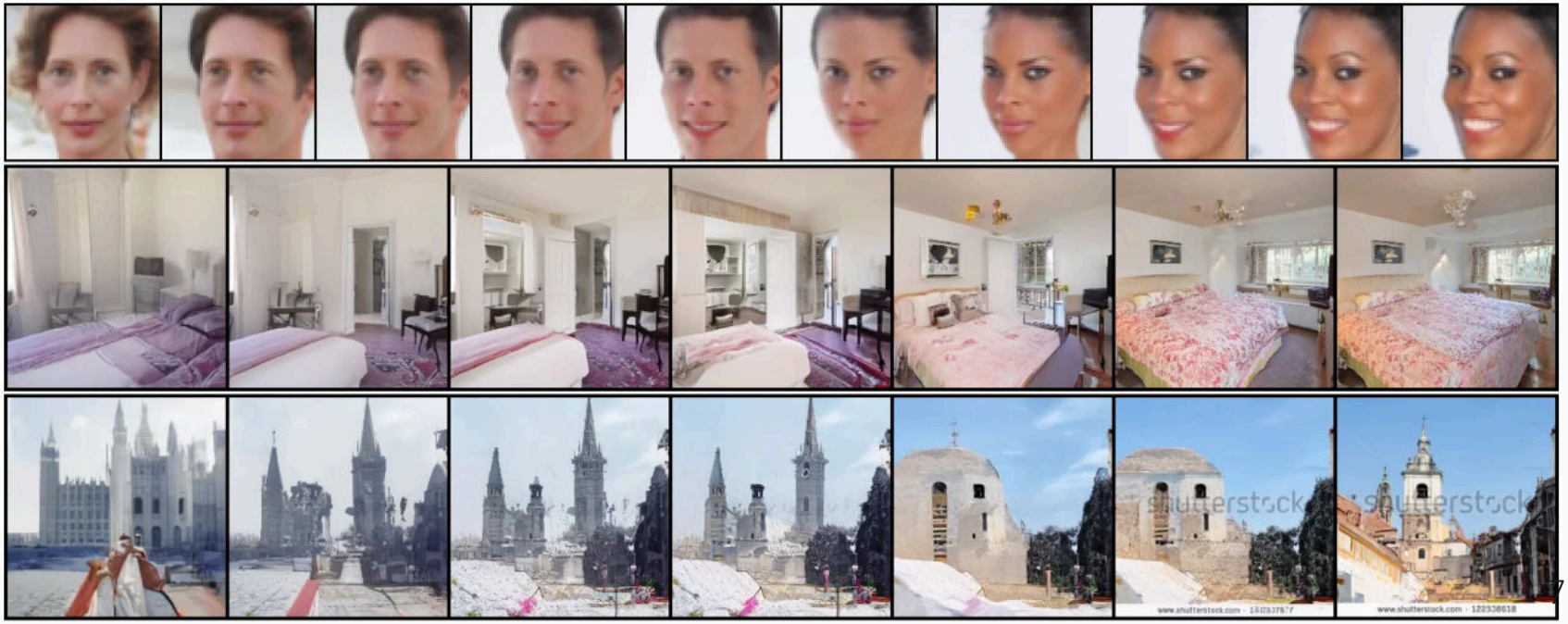
DDIM

- Это привело к 10x–100x ускорению по сравнению с DDPM без потери качества:



DDIM

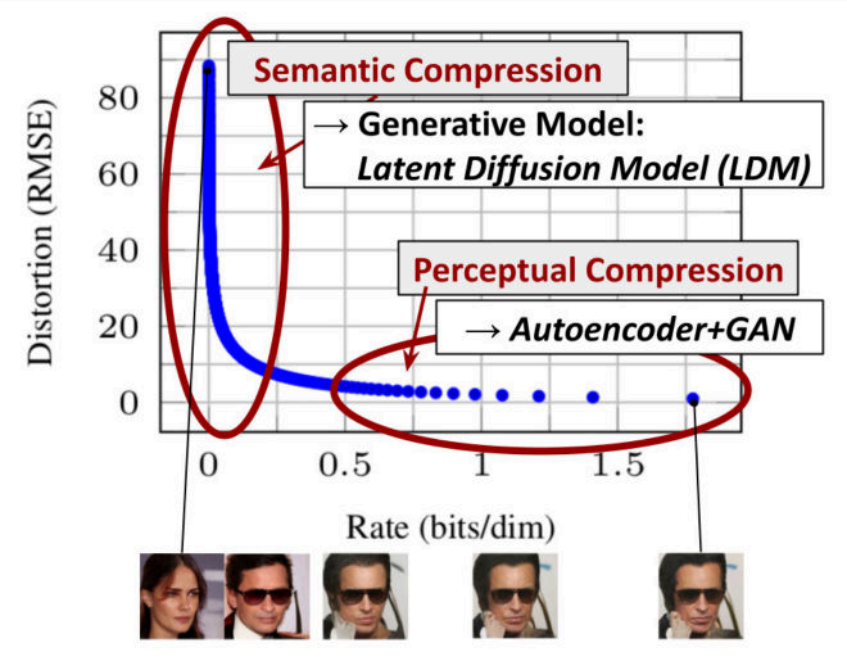
- Порождение в DDIMs тоже не обязано быть стохастическим; можно дисперсию обратной диффузии установить в ноль при порождении
- И теперь латентный код в пространстве $\mathbf{x}_T$ соответствует ровно одной картинке, а DDIM — хорошая модель для латентных представлений; вот, например, интерполяции в латентном пространстве:



Stable Diffusion

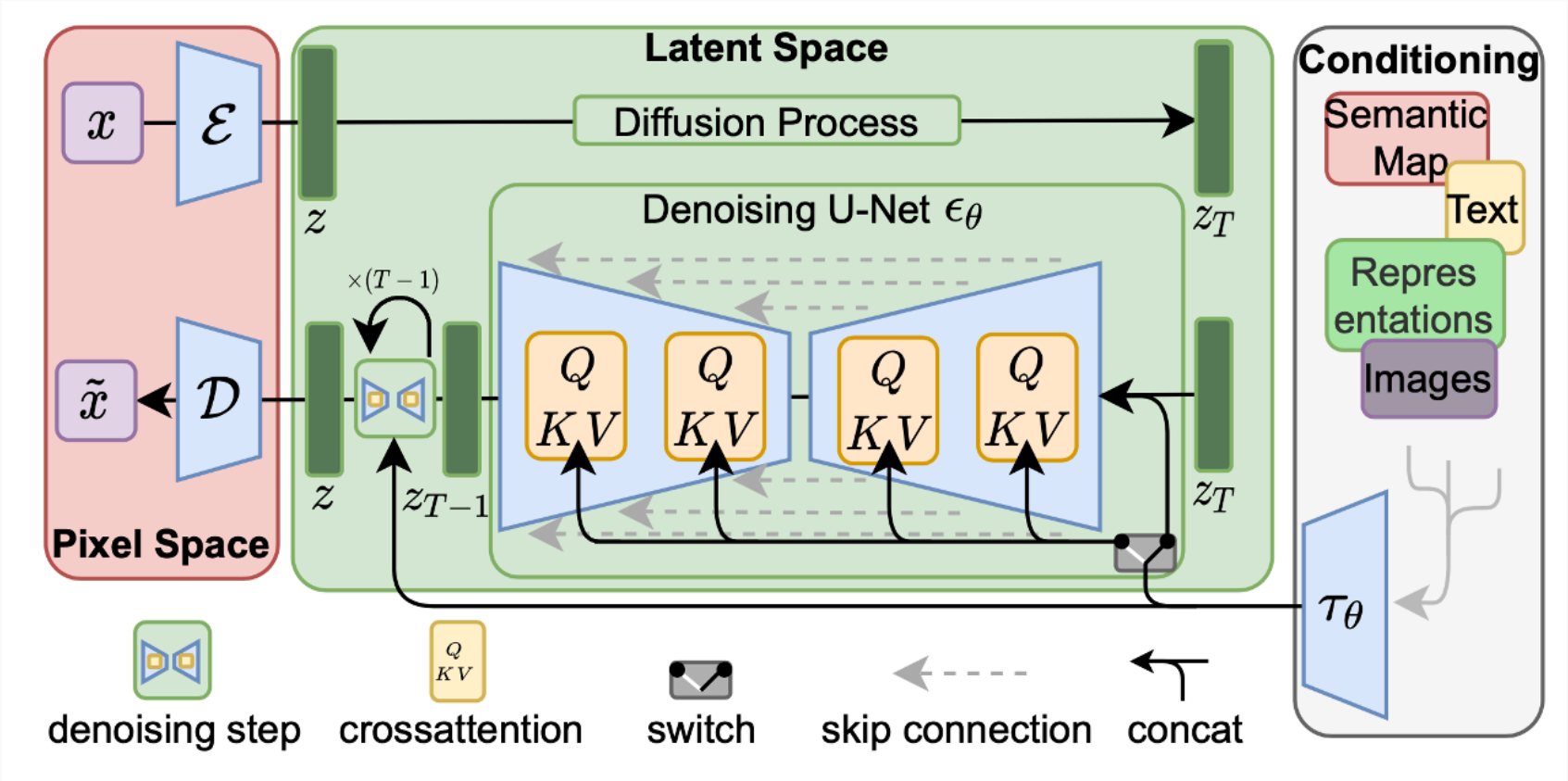
Stable Diffusion

- Ещё один прорыв пришёл тогда, когда стали использовать этот диффузионный процесс *в латентном пространстве*
- Stable Diffusion (Rombach, Blattmann et al., 2022):



Stable Diffusion

- Stable Diffusion (Rombach, Blattmann et al., 2022):

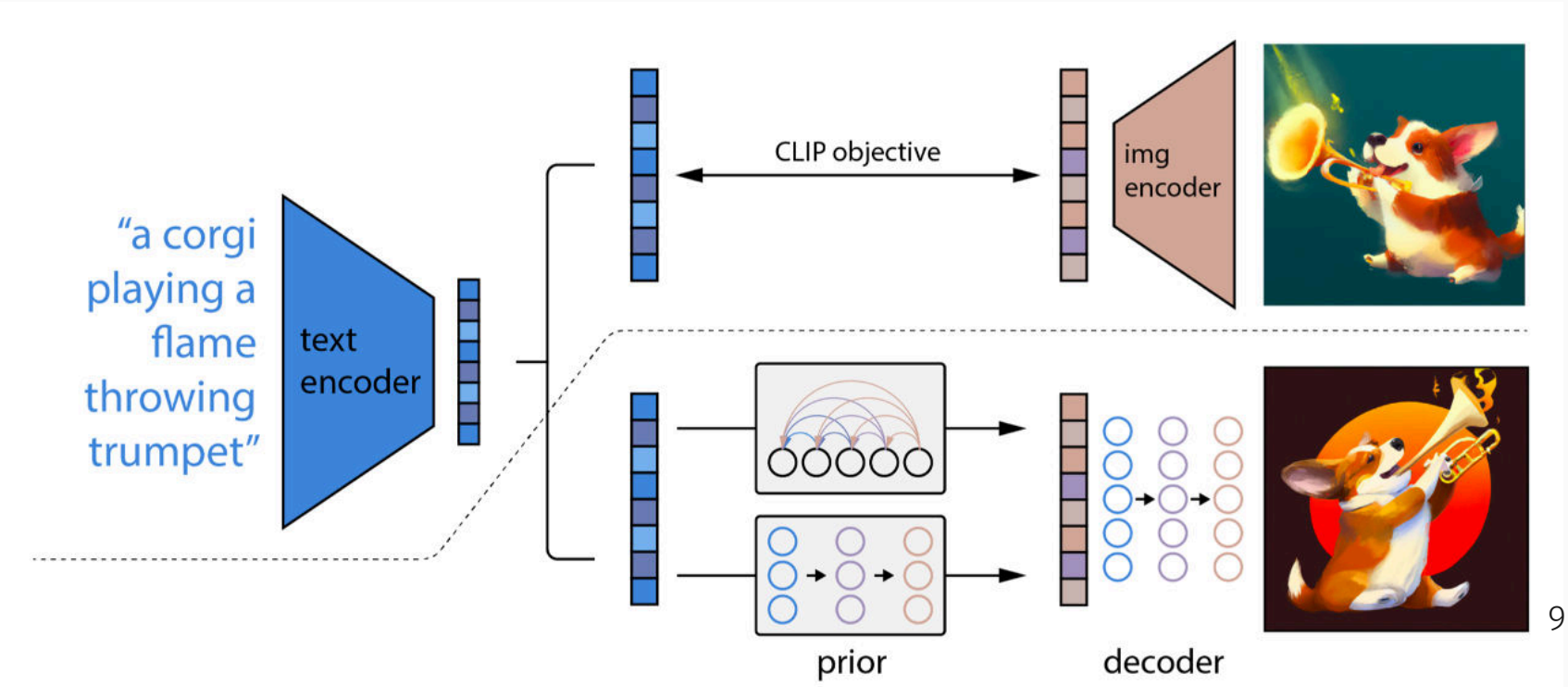


Stable Diffusion

- unCLIP, он же DALL-E 2 (Ramesh et al., 2022), обучает

$$p\left(\mathbf{x}|y\right) = p\left(\mathbf{x}|\mathbf{c}, y\right) p\left(\mathbf{c}|y\right),$$

где prior model $p\left(\mathbf{c}|y\right)$ строит CLIP embedding по тексту, а в $p\left(\mathbf{x}|\mathbf{c}, y\right)$ сначала порождается «шум» (prior) для картинки, а потом по ней диффузионная модель рисует саму картинку:



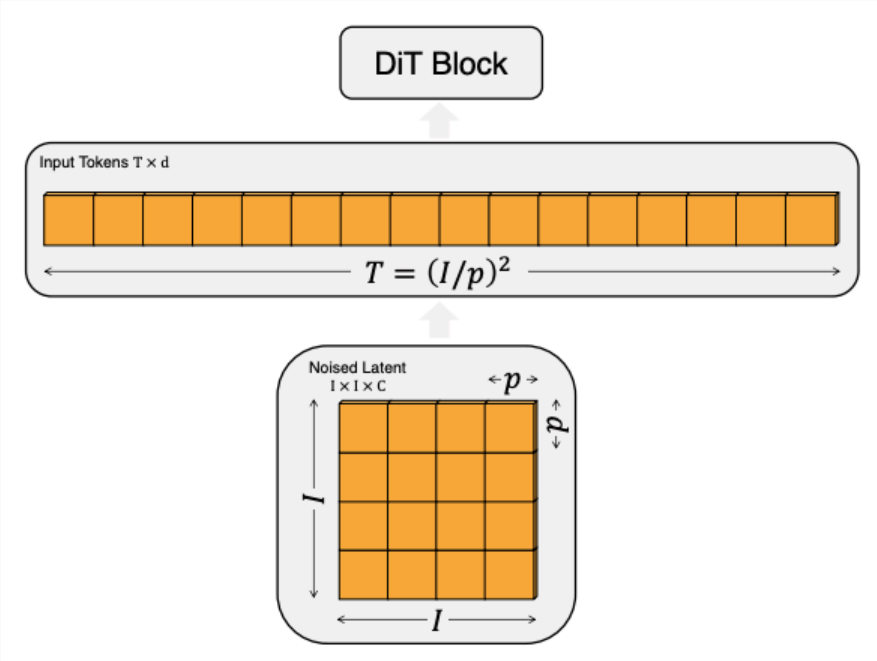
DALL-E 2

Midjourney

Diffusion Transformers

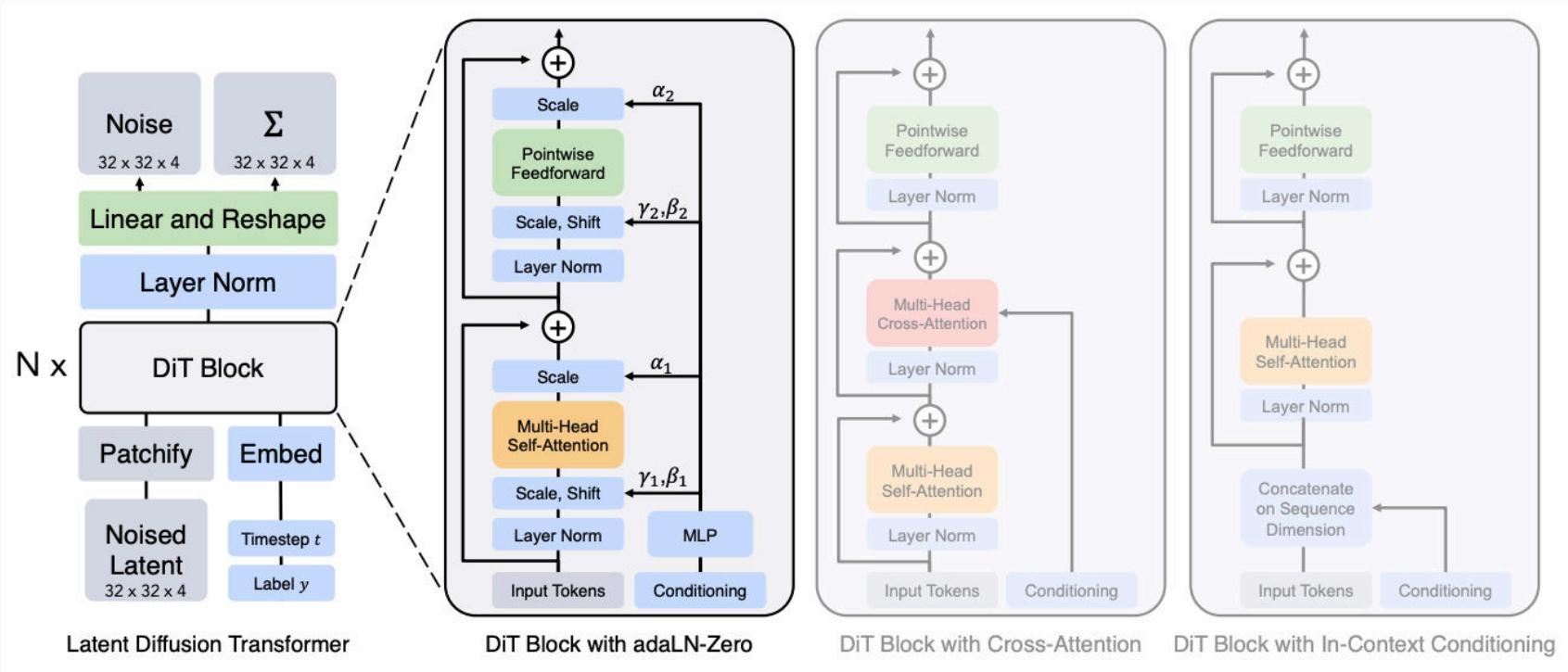
Diffusion Transformers

- Уже вполне современная архитектура – DiT (Peebles, Xie, 2022)
- Это модель латентной диффузии, но вместо U-Net-подобной архитектуры для латентных кодов используют трансформер
- Вход трансформера – тензор $I \times I \times C$, "patchified" в последовательность $p \times p$ патчей длины $T = (I/p)^2$; p – гиперпараметр, квадратично влияющий на сложность



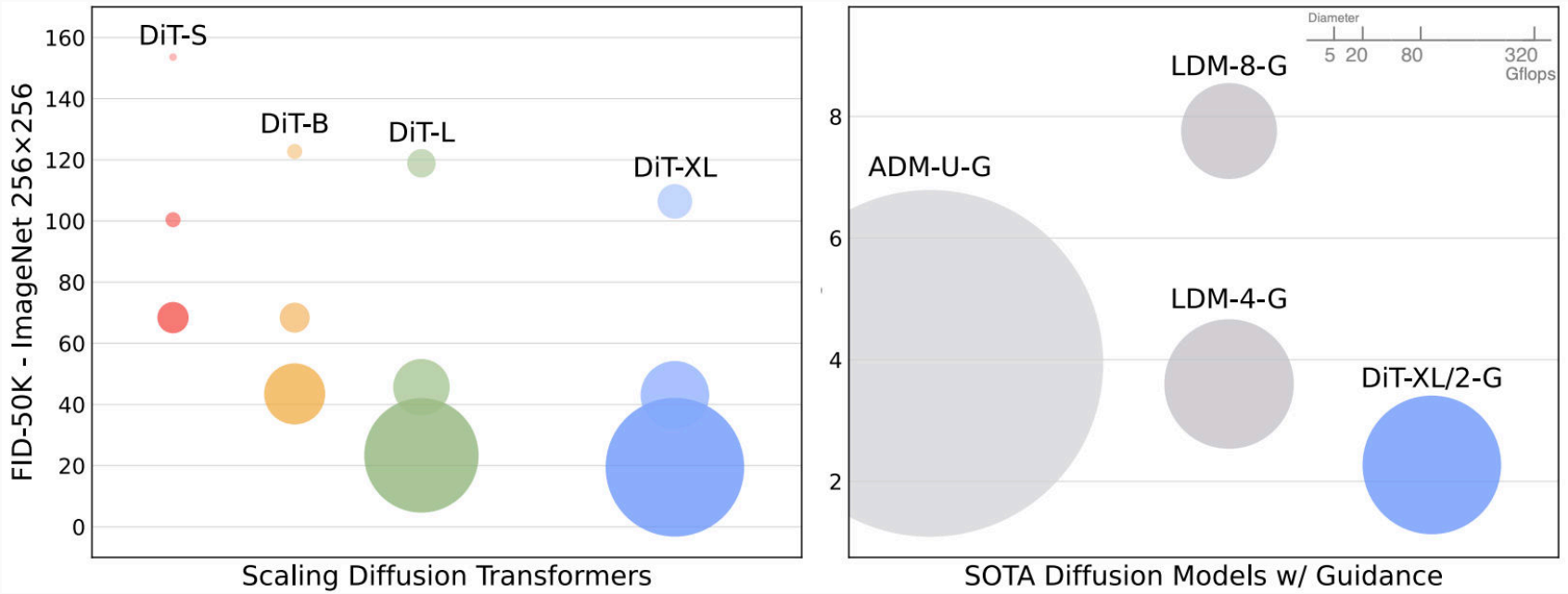
Diffusion Transformers

- Затем его обрабатывают блоками DiT; интересная часть здесь – adaptive layer norm с параметрами, взятыми из условия



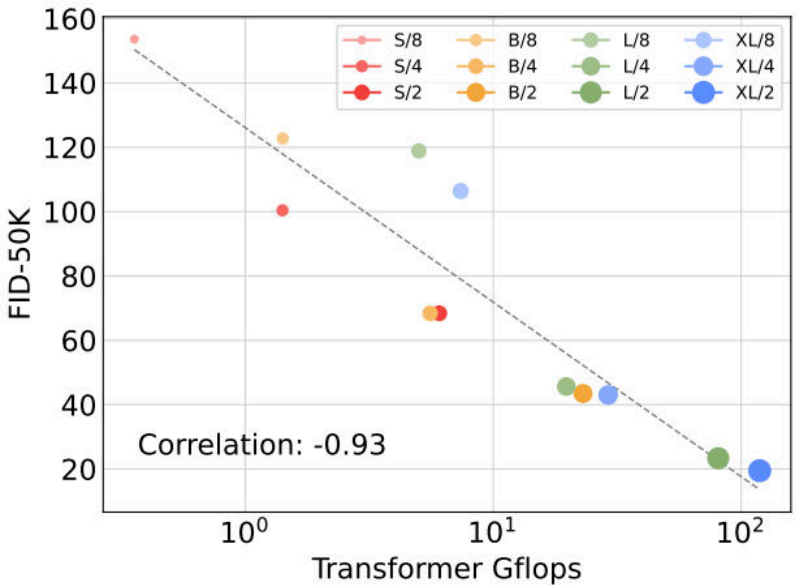
Diffusion Transformers

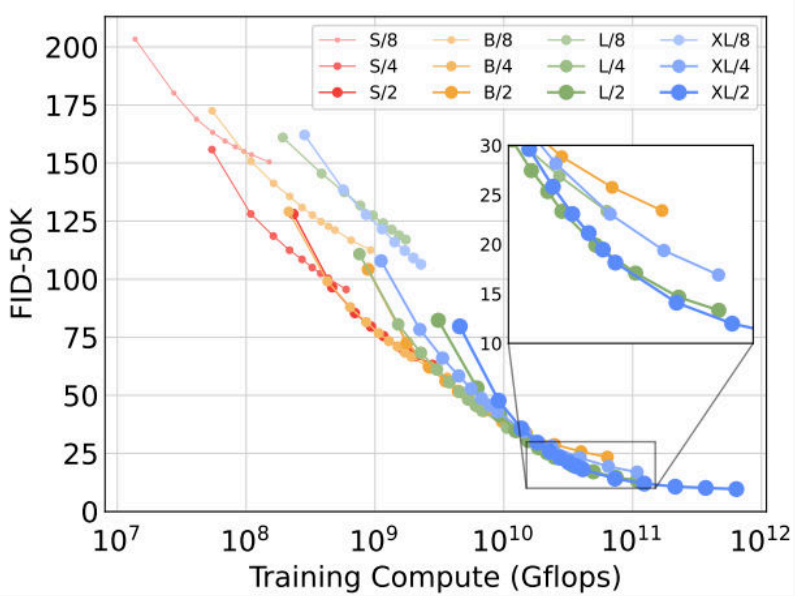
- Получается, что DiT куда более вычислительно эффективны:



Diffusion Transformers

- Transformer GFlops сильно скоррелированы с качеством, а большие модели DiT используют вычисления более эффективно:





Diffusion Transformers

- Хорошие сэмплы:

Flux

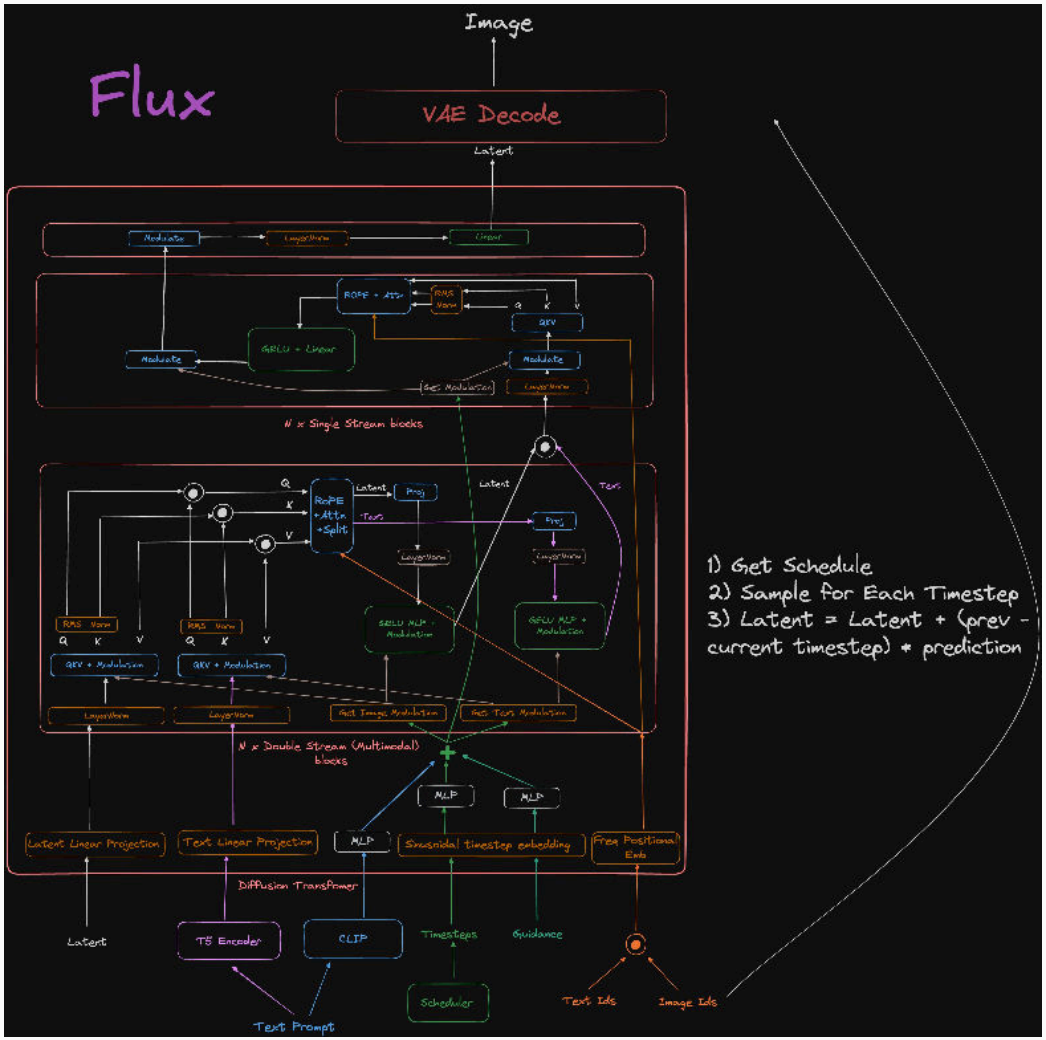
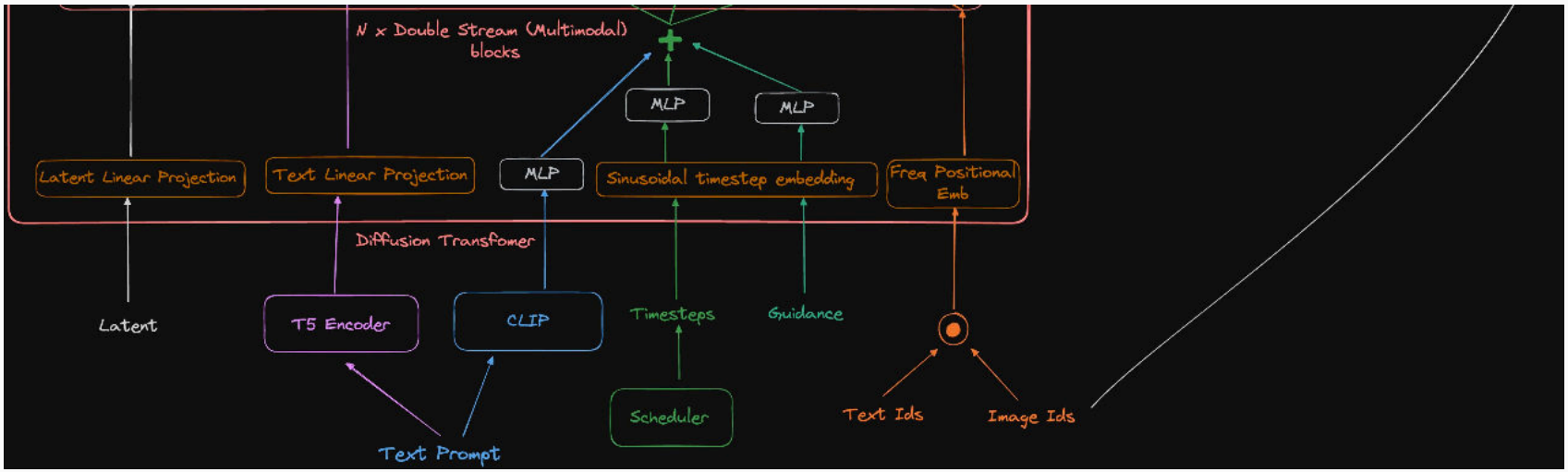
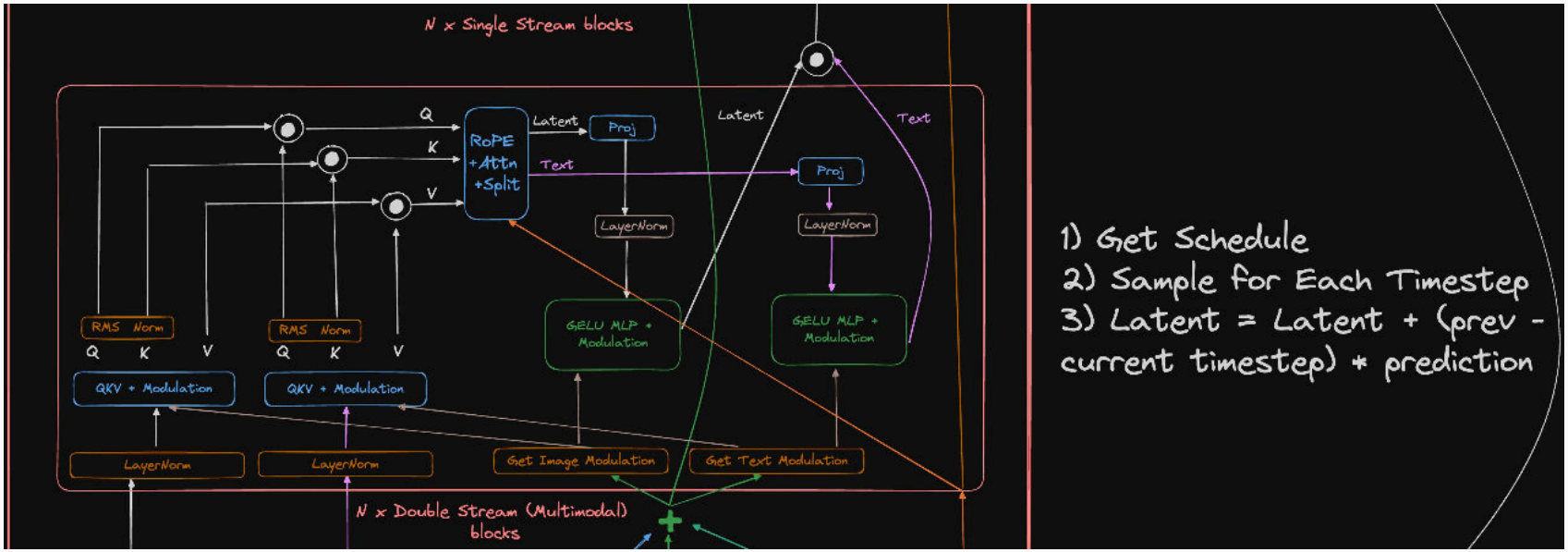


Image
Flux
VAE Decode
Latent
N x Single Stream blocks
RoPE +Attn +Split
Latent
Text
N x Double Stream (Multimodal) blocks
MLP
Latent Linear Projection
Text Linear Projection
Sinusoidal timestep embedding
Diffusion Transformer
Latent
T5 Encoder
CLIP
Timesteps
Guidance
Scheduler
Text Prompt
Text Ids
Image Ids
1) Get Schedule
2) Sample for Each Timestep
3) Latent = Latent + (prev - current timestep) * prediction

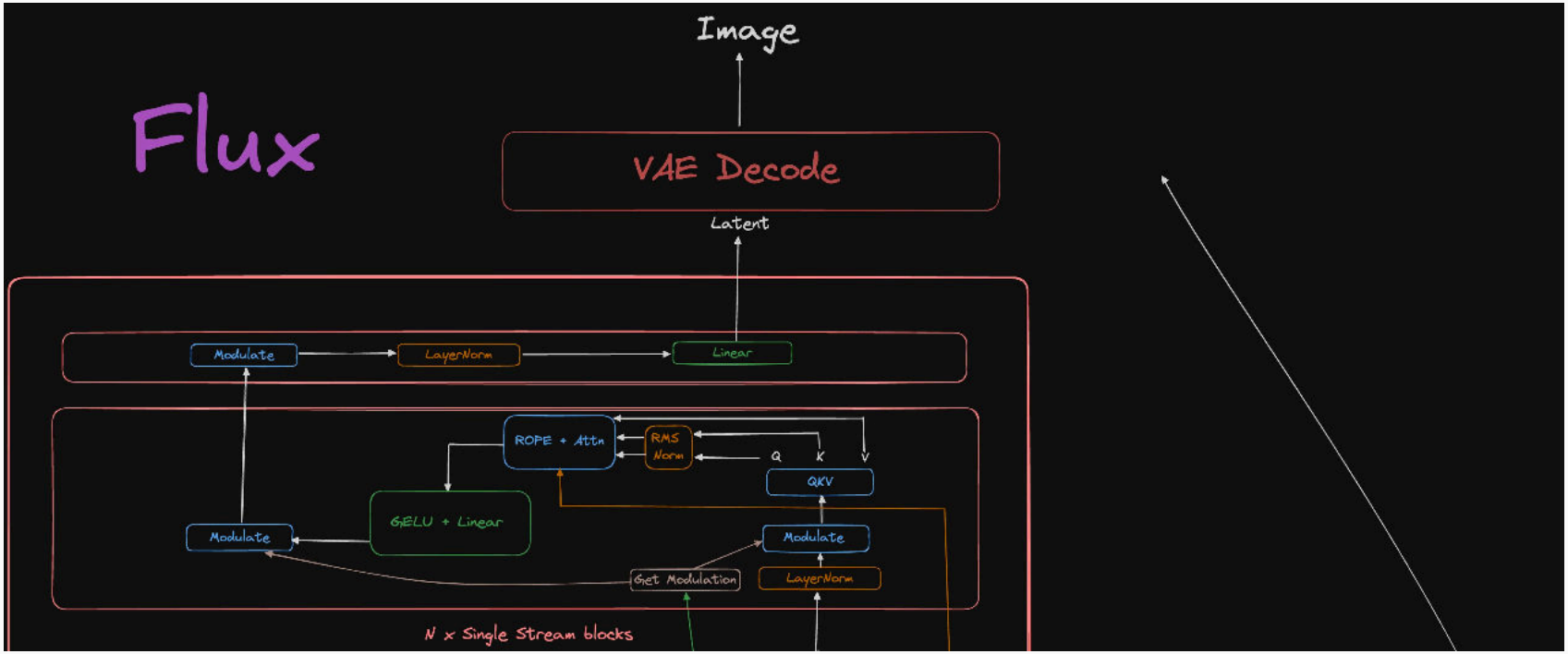
Flux


N x Double Stream (Multimodal) blocks
MLP
MLP
Latent Linear Projection
Text Linear Projection
MLP
Sinusoidal timestep embedding
Freq Positional Emb
Diffusion Transfomer
Latent
T5 Encoder
CLIP
Timesteps
Guidance
Scheduler
Text Ids
Image Ids
Text Prompt

Flux


N x Single Stream blocks
Q
K
V
RoPE
+Attn
+Split
Latent
Proj
Text
Proj
LayerNorm
LayerNorm
Latent
Text
RMS Norm
Q K V
RMS Norm
Q K V
GELU MLP + Modulation
GELU MLP + Modulation
QKV + Modulation
QKV + Modulation
LayerNorm
LayerNorm
Get Image Modulation
Get Text Modulation
N x Double Stream (Multimodal) blocks
1) Get Schedule
2) Sample for Each Timestep
3) Latent = Latent + (prev - current timestep) * prediction

Flux


Flux
Image
VAE Decode
Latent
Modulate
LayerNorm
Linear
ROPE + Attn
RMS Norm
Q
K
V
QKV
GELU + Linear
Modulate
Modulate
Get Modulation
LayerNorm
N x Single Stream blocks

Flux

Flux

Flux

Flux

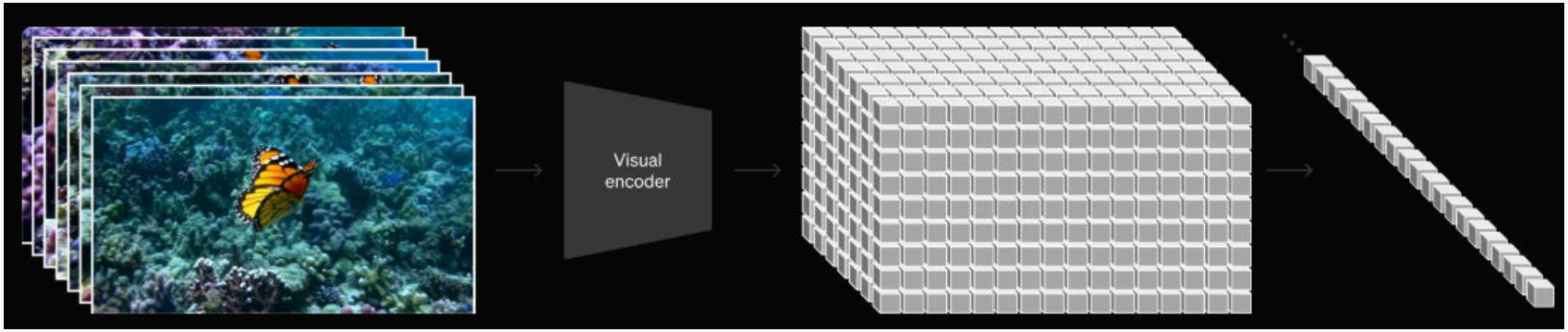
Flux

OpenAI Sora

- Текущее применение DiT: OpenAI Sora (Feb 2024, и вот ещё совсем недавно)

OpenAI Sora

- OpenAI всех деталей не раскрывает, но там точно есть DiT:



Спасибо за внимание!