Генеративные модели

Лекция 8: *Diffusion Models*

Повторение

$Energy - Based\ Models$

Определим плотность $p_{\boldsymbol{\theta}}(\mathbf{x})$ через энергию с помощью распределения Больцмана:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{e^{-E_{\boldsymbol{\theta}}(\mathbf{x})}}{Z_{\boldsymbol{\theta}}}$$

- $E_{\boldsymbol{\theta}}(\mathbf{x})$ – ненормализованная плотность
- $Z_{\boldsymbol{\theta}}$ – нормировочная константа (статистическая сумма)

$Z_{\boldsymbol{\theta}}$ – это интеграл по всему пространству данных:

$$Z_{\boldsymbol{\theta}} = \int e^{-E_{\boldsymbol{\theta}}(\mathbf{x})}\, d\mathbf{x}$$

- Для сложных высокоразмерных данных мы не можем вычислить этот интеграл
- Не зная $Z_{\boldsymbol{\theta}}$, мы не сможем найти $p_{\boldsymbol{\theta}}(\mathbf{x})$ и его градиент для обучения

$Energy - Based\ Models$

$$\mathcal{L}_{\boldsymbol{\theta}} = -\mathbb{E}_{p_{data}(\mathbf{x})}\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x}) + \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})}\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})$$

- Нам нужно уметь оценивать $\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})}[\dots]$
- Будем оценивать этот член с помощью метода Монте-Карло (***Markov Chain Monte Carlo***, ***MCMC***)

Введем понятие ***функции оценки*** (***score function***):

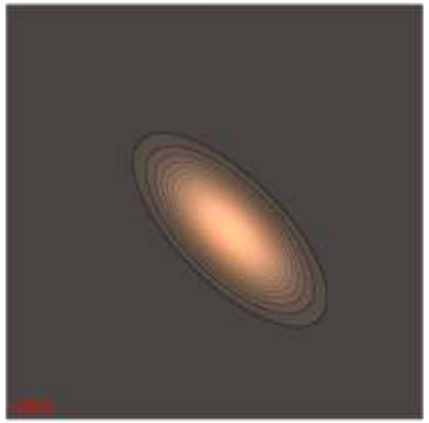
$$s_{\boldsymbol{\theta}}(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$$

Langevin Dynamics

Для сэмплирования из $p_{\boldsymbol{\theta}}(\mathbf{x})$ будем использовать ***динамику Ланжевена*** (***Langevin Dynamics***):

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{\eta}{2}\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \sqrt{\eta}\boldsymbol{\epsilon}_t = \mathbf{x}_t + \frac{\eta}{2} s_{\boldsymbol{\theta}}(\mathbf{x}_t) + \sqrt{\eta}\boldsymbol{\epsilon}_t$$

- $\mathbf{x}_t -$ текущая точка
- $\eta -$ размер шага
- $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) -$ шум



- Член $\frac{\eta}{2}\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t)$ – это градиентный подъем, который толкает нашу точку $\mathbf{x}$ в направлении наибольшего возрастания плотности

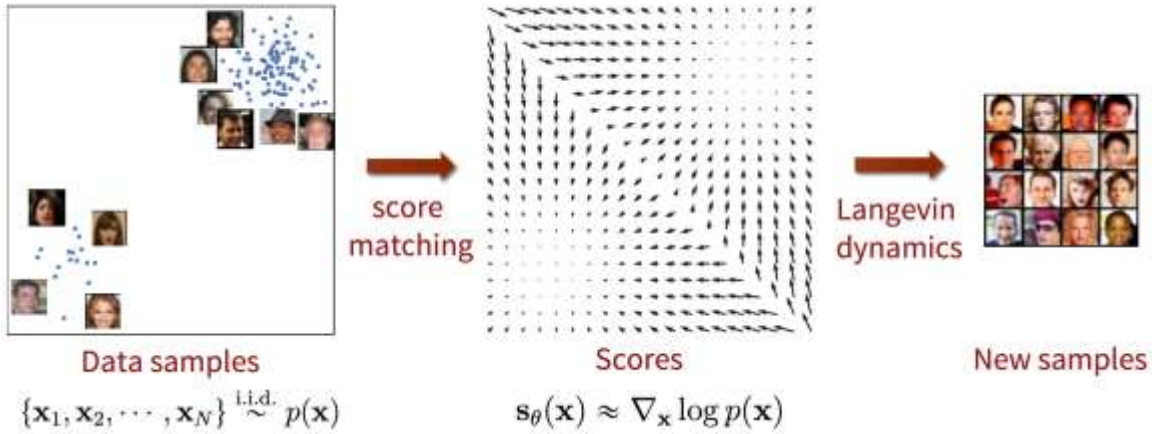
- Член $\sqrt{\eta}\boldsymbol{\epsilon}_t$ добавляет случайности и не дает нашей точке $\mathbf{x}$ застрять в локальном максимуме

Проблемы $MCMC$

- Обучение с использованием $MCMC$ обычно очень медленное, не стабильное и даёт смещенные градиенты

Идея:

- Попробуем обучать $score\ function\ s_{\boldsymbol{\theta}}(\mathbf{x})$ напрямую



Score Matching

Цель ***score matching*** (***Hyvärinen*, 2005**) заключатся в минимизации дивергенции Фишера*:*

$$D_F\left(p_{data}(\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{x})\right) = \mathbb{E}_{p_{data}(\mathbf{x})}\left[\frac{1}{2}||\nabla_{\mathbf{x}}\log p_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}}\log p_{data}(\mathbf{x})||^2\right]$$

- Минимизируя эту функцию потерь, мы заставляем *score*-функцию модели $s_{\boldsymbol{\theta}}(\mathbf{x})$ соответствовать *score*-функции реальных данных $s_{data}(\mathbf{x})$
- Если $s_{\boldsymbol{\theta}}(\mathbf{x}) \approx s_{data}(\mathbf{x})$, то $p_{\boldsymbol{\theta}}(\mathbf{x}) \approx p_{data}(\mathbf{x})$

Мы не знаем *score*-функцию реальных данных $s_{data}(\mathbf{x})$

Hyvärinen показал, что при определенных условиях регулярности функция потерь может быть переписана в виде*:*

$$D_F\left(p_{data}(\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{x})\right) = \mathbb{E}_{p_{data}(\mathbf{x})}\left[\frac{1}{2}||s_{\boldsymbol{\theta}}(\mathbf{x}) - s_{data}(\mathbf{x})||^2\right] = \mathbb{E}_{p_{data}(\mathbf{x})}\left[\frac{1}{2}||s_{\boldsymbol{\theta}}(\mathbf{x})||^2 + tr\left(\nabla_{\mathbf{x}}s_{\boldsymbol{\theta}}(\mathbf{x})\right)\right]$$

Проблемы *Score Matching*

- Классический **score – *matching*** требует, чтобы $p_{data}(\mathbf{x})$ была гладкой и непрерывно дифференцируемой

- Реальные данные часто могут быть дискретны (изображения $0 - 255$)
- Для таких данных $\log p_{data}(\mathbf{x})$ является разрывной функцией

Идея:

Добавим небольшой шум к нашим данным*:*

$$\mathbf{x}_\sigma = \mathbf{x} + \sigma \cdot \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

Процесс добавления шума описывается условной вероятностью:

$$q(\mathbf{x}_{\boldsymbol{\sigma}}|\mathbf{x}) = \mathcal{N}(\mathbf{x}_{\boldsymbol{\sigma}}|\mathbf{x}, \sigma^2 \cdot \boldsymbol{I})$$

Denoising Score Matching

Мы получили новое гладкое распределение, к которому можно применять **score – *matching***:

$$q(\mathbf{x}_{\boldsymbol{\sigma}}) = \int q(\mathbf{x}_{\boldsymbol{\sigma}}|\mathbf{x}) \cdot p_{data}(\mathbf{x})\, d\mathbf{x}$$

Будем минимизировать дивергенцию Фишера между *score*-функцией модели $s_{\boldsymbol{\theta}}(\mathbf{x})$ и *score*-функцией зашумленных данных $q(\mathbf{x}_{\boldsymbol{\sigma}})$:

$$\frac{1}{2}\mathbb{E}_{q(\mathbf{x}_{\boldsymbol{\sigma}})}\left[\left|\left|s_{\boldsymbol{\theta}}(\mathbf{x}_{\boldsymbol{\sigma}}) - \nabla_{\mathbf{x}_{\boldsymbol{\sigma}}} \log q(\mathbf{x}_{\boldsymbol{\sigma}})\right|\right|_2^2\right] \rightarrow \min_{\boldsymbol{\theta}}$$

Мы по-прежнему не знаем $\nabla_{\mathbf{x}_{\boldsymbol{\sigma}}} \log q(\mathbf{x}_{\boldsymbol{\sigma}})$, так как $q(\mathbf{x}_{\boldsymbol{\sigma}})$ зависит от $p_{data}(\mathbf{x})$:

Vincent (2010) показал, что минимизация дивергенции Фишера эквивалентна минимизации новой функции потерь*:*

$$\mathbb{E}_{q(\mathbf{x}_{\boldsymbol{\sigma}})}\left[\left|\left|s_{\boldsymbol{\theta}}(\mathbf{x}_{\boldsymbol{\sigma}}) - \nabla_{\mathbf{x}_{\boldsymbol{\sigma}}} \log q(\mathbf{x}_{\boldsymbol{\sigma}})\right|\right|_2^2\right] = \mathbb{E}_{p_{data}(\mathbf{x})}\mathbb{E}_{q(\mathbf{x}_{\sigma}|\mathbf{x})}\left[\left|\left|s_{\boldsymbol{\theta}}(\mathbf{x}_{\boldsymbol{\sigma}}) - \nabla_{\mathbf{x}_{\boldsymbol{\sigma}}} \log q(\mathbf{x}_{\sigma}|\mathbf{x})\right|\right|^2\right] + const$$

Denoising Score Matching

Мы сами задаем $q(\mathbf{x}_\sigma|\mathbf{x})$:

$$q(\mathbf{x}_\sigma|\mathbf{x}) = \mathcal{N}(\mathbf{x}_\sigma|\mathbf{x}, \sigma^2\boldsymbol{I})$$

В этом случае *score*-функция вычисляется аналитически:

$$\nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x}) = \nabla_{\mathbf{x}_\sigma}\left(-\frac{\left||\mathbf{x}_\sigma - \mathbf{x}|\right|^2}{2\sigma^2}\right) = -\frac{\mathbf{x}_\sigma - \mathbf{x}}{\sigma^2} = -\frac{\boldsymbol{\epsilon}}{\sigma}$$

Подставим в нашу функцию потерь*:*

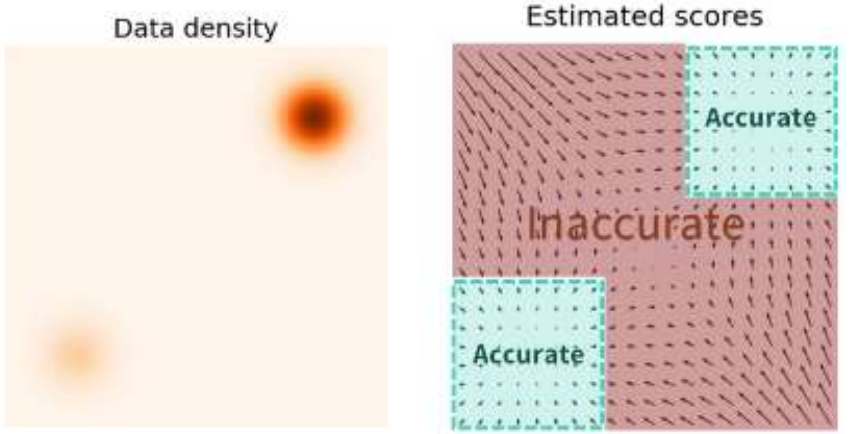
$$\frac{1}{2}\mathbb{E}_{p_{data}(\mathbf{x})}\mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})}\left[\left|\left|s_{\boldsymbol{\theta}}(\mathbf{x}_{\boldsymbol{\sigma}}) - \nabla_{\mathbf{x}_{\boldsymbol{\sigma}}} \log q(\mathbf{x}_\sigma|\mathbf{x})\right|\right|^2\right] = \frac{1}{2}\mathbb{E}_{p_{data}(\mathbf{x})}\mathbb{E}_{\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left|\left|s_{\boldsymbol{\theta}}(\mathbf{x} + \sigma\boldsymbol{\epsilon}) + \frac{\boldsymbol{\epsilon}}{\sigma}\right|\right|^2\right] \to \min_{\boldsymbol{\theta}}$$

- Нейросеть $s_{\boldsymbol{\theta}}(\mathbf{x}_{\boldsymbol{\sigma}})$ учится напрямую предсказывать шум $\boldsymbol{\epsilon}$, который был добавлен к $\mathbf{x}$

Проблема *Denoising Score Matching*

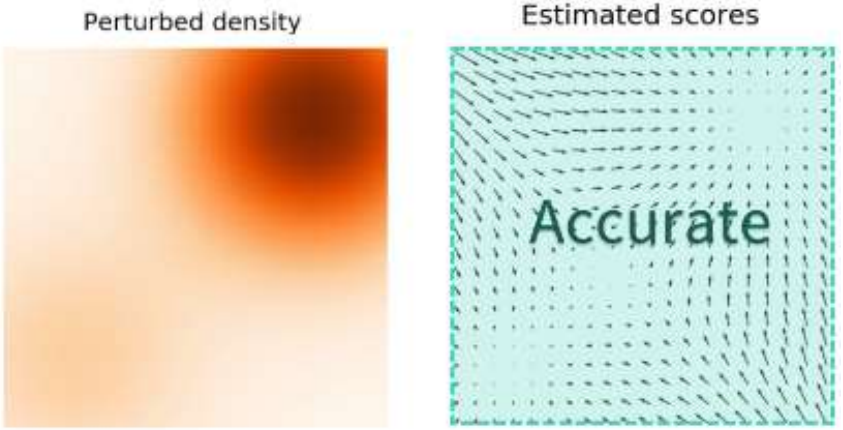
Если σ маленький:

- $q(\mathbf{x}_{\boldsymbol{\sigma}}) \approx p_{data}(\mathbf{x})$
- Функция потерь *DSM* содержит члены $\frac{1}{\sigma}$ и $\frac{1}{\sigma^2}$, что приводит к высокой дисперсии градиентов
- Обучение нестабильно, динамика Ланжевена застревает в модах



Если σ большой:

- Функция потерь *DSM* с низкой дисперсией градиентов
- $q(\mathbf{x}_{\boldsymbol{\sigma}}) \neq p_{data}(\mathbf{x})$ – сильно размытое распределение
- Модель хорошо работает в областях с низкой плотностью, но изучает искаженное распределение



Noise Conditioned Score Network

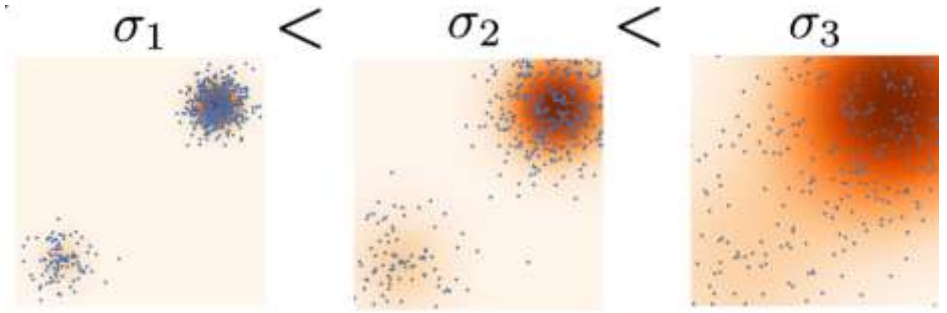
Идея:

Будем одновременно использовать несколько уровней шума

- Определим последовательность уровней шума $\sigma_1 < \sigma_2 < \cdots < \sigma_T$:

- $\sigma_1 -$ небольшой шум, $q(\mathbf{x}_1) \approx p_{data}(\mathbf{x})$
- $\sigma_T -$ большой шум, $q(\mathbf{x}_T) \approx \mathcal{N}(0, \sigma_T^2 \boldsymbol{I})$

- Каждому σ_t будет соответствовать свое распределение $q(\mathbf{x}_t)$



Обучение $NCSN$

Будем обучать $s_{\boldsymbol{\theta},\sigma_t}(\mathbf{x}_t)$ на каждом уровне шума, используя взвешенную функцию потерь DSM:

$$\mathcal{L}_{\boldsymbol{\theta}} = \sum_{t=1}^{T} \lambda(\sigma_t) \cdot \mathbb{E}_{p_{data}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x})} \left[\left|\left| s_{\boldsymbol{\theta},\sigma_t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_{\boldsymbol{t}}|\mathbf{x}) \right|\right|^2 \right],$$

где $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_{\boldsymbol{t}}|\mathbf{x}) = -\frac{\boldsymbol{\epsilon}}{\sigma_t}$

Обычно веса $\lambda(\sigma_t)$ берут в виде σ_t^2, чтобы сбалансировать члены с $\frac{1}{\sigma_t^2}$

$$\mathcal{L}_{\boldsymbol{\theta}} = \sum_{t=1}^{T} \sigma_t^2 \cdot \mathbb{E}_{p_{data}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x})} \left[\left|\left| s_{\boldsymbol{\theta},\sigma_t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_{\boldsymbol{t}}|\mathbf{x}) \right|\right|^2 \right]$$

План

- *Diffusion Models*
- *Forward Diffusion Process*
- *Reverse Diffusion Process*
- *DDPM*

Diffusion Models

Идея диффузионных моделей

- ***Sohl – Dickstein*** (2015) задались вопросом как преобразовать одно распределение в другое

VAE:

- Кодировщик $q_{\phi}(\boldsymbol{z}|\mathbf{x})$ сжимает данные в латентный вектор $\boldsymbol{z}$
- Декодер $p_{\theta}(\boldsymbol{z}|\mathbf{x})$ восстанавливает $\mathbf{x}$ из $\boldsymbol{z}$

Проблема:

Декодер $p_{\theta}(\boldsymbol{z}|\mathbf{x})$ должен за один шаг преобразовать вектор $\boldsymbol{z}$ в сложный объект $\mathbf{x}$

Идея:

Попробуем вместо одного шага построить цепочку из T небольших шагов

Forward Diffusion Process

Forward Diffusion Process

Определим ***прямой диффузионный процесс*** как ***марковскую цепь,*** в которой мы добавляем шум к нашим данным:

$$q(\mathbf{x}_{0:T}) = q(\mathbf{x}_0)\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

- $q(\mathbf{x}_0) -$ распределение реальных данных
- $q(\mathbf{x}_t|\mathbf{x}_{t-1}) -$ переходное распределение

Для описания добавления шума на каждом шаге t будем использовать ***расписание шума*** (***noise schedule***) β_t:

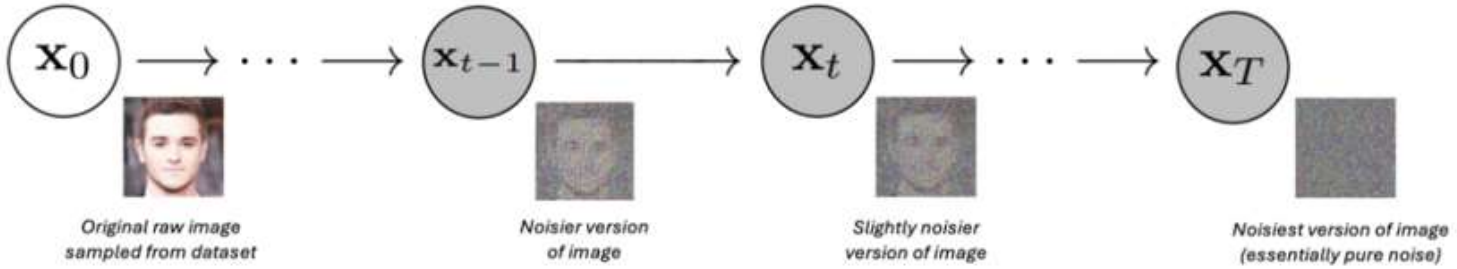
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t|\sqrt{1-\beta_t}\cdot\mathbf{x}_{t-1}, \beta_t\boldsymbol{I})$$

- $\sqrt{1-\beta_t}\cdot\mathbf{x}_{t-1} -$ среднее значение
- $\beta_t\boldsymbol{I} -$ дисперсия

Forward Diffusion Process

Шаг получения $\mathbf{x}_t$ из $\mathbf{x}_{t-1}$ в явном виде*:*

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



Иногда удобнее работать с параметром α_t, который имеет смысл ***коэффициента сохранения сигнала***:

$$\alpha_t = 1 - \beta_t$$

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

Forward Diffusion Process

- Можно ли сразу узнать зашумленное состояние $\mathbf{x}_t$, если нам известно начальное состояние $\mathbf{x}_0$?

Условное распределение $q(\mathbf{x}_t|\mathbf{x}_0)$, которое устанавливает связь между начальным состоянием $\mathbf{x}_0$ и любым последующим $\mathbf{x}_t$ имеет вид:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t\middle|\sqrt{\overline{\alpha}_t}\cdot\mathbf{x}_0, (1-\overline{\alpha}_t)\boldsymbol{I}\right)$$

- $\overline{\alpha}_t = \prod_{i=1}^{t}\alpha_i$

Forward Diffusion Process

Подставим последовательно определение шагов диффузии:

$$\mathbf{x}_t = \sqrt{\alpha_t} \cdot \mathbf{x}_{t-1} + \sqrt{1-\alpha_t} \cdot \boldsymbol{\epsilon}_t$$

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}} \cdot \mathbf{x}_{t-2} + \sqrt{1-\alpha_{t-1}} \cdot \boldsymbol{\epsilon}_{t-1}$$

Получаем:

$$\mathbf{x}_t = \sqrt{\alpha_t} \cdot \left(\sqrt{\alpha_{t-1}} \cdot \mathbf{x}_{t-2} + \sqrt{1-\alpha_{t-1}} \cdot \boldsymbol{\epsilon}_{t-1}\right) + \sqrt{1-\alpha_t} \cdot \boldsymbol{\epsilon}_t =$$

$$= \sqrt{\alpha_t \cdot \alpha_{t-1}} \cdot \mathbf{x}_{t-2} + \underbrace{\sqrt{\alpha_t(1-\alpha_{t-1})} \cdot \boldsymbol{\epsilon}_{t-1} + \sqrt{1-\alpha_t} \cdot \boldsymbol{\epsilon}_t}_{\boldsymbol{w}}$$

$$\mathbb{E}[\boldsymbol{w}] = \mathbb{E}[\boldsymbol{\epsilon}_{t-1}] + \mathbb{E}[\boldsymbol{\epsilon}_t] = 0$$

$$Var[\boldsymbol{w}] = \alpha_t(1-\alpha_{t-1}) \cdot \boldsymbol{I} + (1-\alpha_t) \cdot \boldsymbol{I} = (1-\alpha_t \cdot \alpha_{t-1}) \cdot \boldsymbol{I}$$

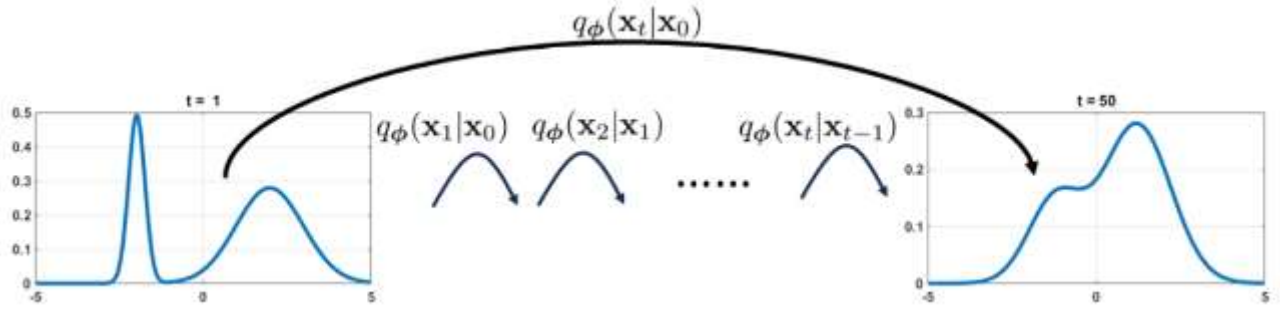
Forward Diffusion Process

Повторяя шаги по индукции, мы получим:

$$\mathbf{x}_t = \sqrt{\alpha_t \alpha_{t-1} \dots \alpha_1} \cdot \mathbf{x}_0 + \sqrt{1 - \alpha_t \alpha_{t-1} \dots \alpha_1} \cdot \boldsymbol{\epsilon}_0 = \sqrt{\overline{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \overline{\alpha}_t} \cdot \boldsymbol{\epsilon}_0$$

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

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\overline{\alpha}_t} \cdot \mathbf{x}_0, (1 - \overline{\alpha}_t)\boldsymbol{I})$$



Рассмотрим предел $t \to \infty$:

$$q(\mathbf{x}_t|\mathbf{x}_0) \to \mathcal{N}\left(\sqrt{0} \cdot \mathbf{x}_0, (1-0)\boldsymbol{I}\right) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

Forward Diffusion Process

Вспомним динамику Ланжевена:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{\eta}{2}\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \sqrt{\eta}\boldsymbol{\epsilon}_t \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

Прямой диффузионный процесс*:*

$$\mathbf{x}_t = \sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}_t$$

Воспользуемся приближением $\sqrt{1-x} \approx 1 - \frac{x}{2}$ из ряда Тейлора для $x \ll 1$:

$$\mathbf{x}_t = \sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_t \approx \left(1 - \frac{\beta_t}{2}\right) \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}_t = \mathbf{x}_{t-1} + \frac{\beta_t}{2}(-\mathbf{x}_{t-1}) + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}_t$$

Forward Diffusion Process

$$\boldsymbol{LD}: \quad \mathbf{x}_{t+1} = \mathbf{x}_t + \frac{\eta}{2} \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \sqrt{\eta}\boldsymbol{\epsilon}_t$$

$$\boldsymbol{FDP}: \quad \mathbf{x}_t \approx \mathbf{x}_{t-1} + \frac{\beta_t}{2}(-\mathbf{x}_{t-1}) + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}_t$$

Видим соответствие:

$$\beta_t = \eta$$

$$\nabla_{\mathbf{x}_{t-1}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}) = -\mathbf{x}_{t-1}$$

Найдем *score* −функцию стандартного нормального распределения:

$$\nabla_{\mathbf{x}} \log \mathcal{N}(\mathbf{0}, \boldsymbol{I}) = \nabla_{\mathbf{x}} \log C e^{-\frac{\mathbf{x}^2}{2}} = \nabla_{\mathbf{x}} \log C + \nabla_{\mathbf{x}} \log e^{-\frac{\mathbf{x}^2}{2}} = -\mathbf{x}$$

Forward Diffusion Process

$$\textbf{LD}: \quad \mathbf{x}_{t+1} = \mathbf{x}_t + \frac{\eta}{2} \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \sqrt{\eta}\boldsymbol{\epsilon}_t$$

$$\textbf{FDP}: \quad \mathbf{x}_t \approx \mathbf{x}_{t-1} + \frac{\beta_t}{2}(-\mathbf{x}_{t-1}) + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}_t$$

Вывод:

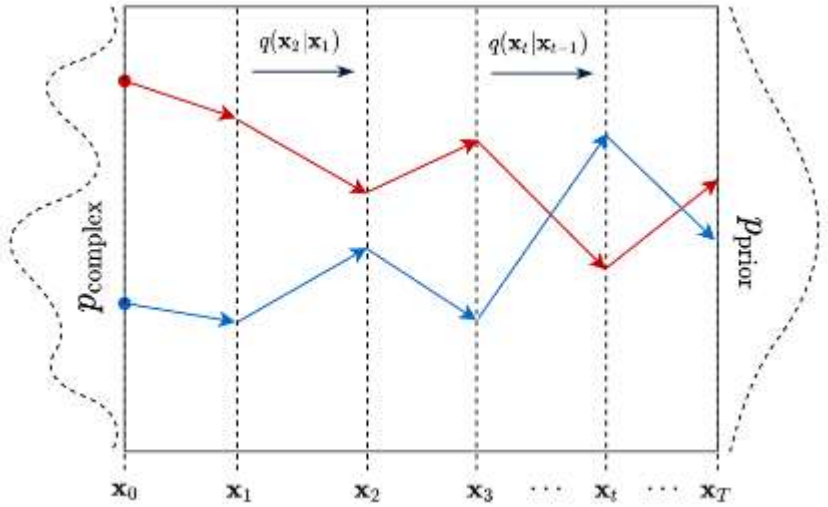
- Прямой диффузионный процесс – это динамика Ланжевена, которая тянет наши данные $\mathbf{x}$ к распределению $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$, используя его *score* −функцию
- Это даёт теоретическое обоснование, что $\mathbf{x}_T$ сойдется к $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$

Математика прямого процесса

Диффузия – это физический процесс перехода частиц из области высокой плотности в область низкой плотности

Прямой процесс:

- Берём $\mathbf{x}_0 \sim p_{data}(\mathbf{x})$
- Добавляем к ней шум $\mathbf{x}_t = \sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}_t$
- После T шагов получим $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$



- Прямой процесс фиксирован и не обучается

- Наша цель – попытаться обратить этот процесс

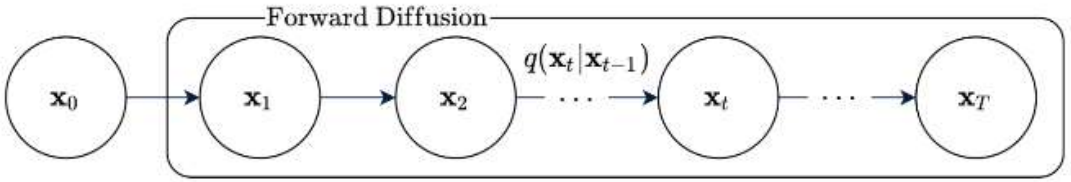
Reverse Diffusion Process

$Diffusion\ Model\ as\ VAE$

- Построим декодер $p_{\boldsymbol{\theta}}$, который будет обращать прямой диффузионный процесс q

Идея:

Рассмотрим весь диффузионный процесс как иерархический VAE:



- $\mathbf{x} = \mathbf{x}_0 -$ наблюдаемые данные
- $\boldsymbol{z} = (\mathbf{x}_1, \dots, \mathbf{x}_T) -$ скрытые переменные

- Кодировщик $q_{\boldsymbol{\phi}} -$ это наш прямой процесс q, который не является обучаемым

- В этом случае вариационное распределение $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$ имеет вид:

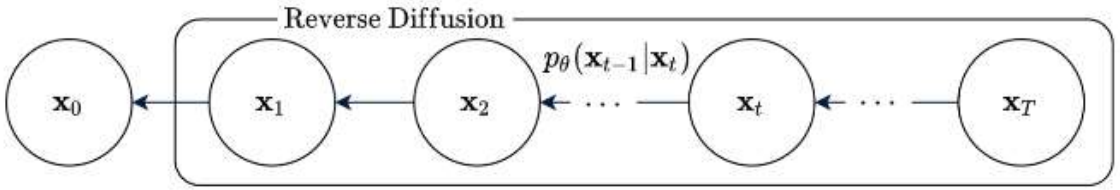
$$q(\boldsymbol{z}|\mathbf{x}) = q(\mathbf{x}_1, \dots \mathbf{x}_T|\mathbf{x}_0) = q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

$Diffusion\ Model\ as\ VAE$

В VAE декодер $p_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{z})$ – это шаг $\boldsymbol{z} \to \mathbf{x}$

В нашем случае $\boldsymbol{z} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$, поэтому $p_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{z})$ – это последний шаг:

$$p_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{z}) = p_{\boldsymbol{\theta}}(\mathbf{x_0}|\mathbf{x_1})$$



Априорное распределение $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ – это вся остальная обратная цепь*:*

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}) = p_{\boldsymbol{\theta}}(\mathbf{x_{1:T}}) = p_{\boldsymbol{\theta}}(\mathbf{x_T}) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\mathbf{x_{t-1}}|\mathbf{x_t})$$

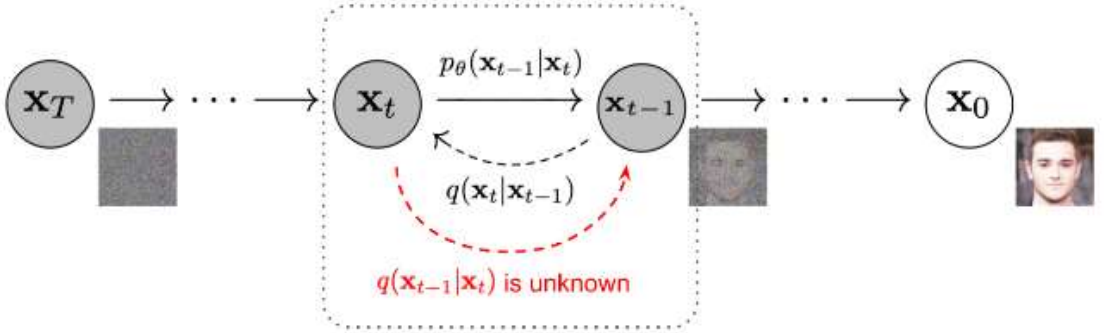
- ***VAE***: простой $prior$ $p(\boldsymbol{z})$ + сложный декодер $p_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{z})$
- ***Diffusion***: сложный $prior$ $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ + простой декодер $p_{\boldsymbol{\theta}}(\mathbf{x_0}|\mathbf{x_1})$

$Diffusion\ Model\ as\ VAE$

Мы хотим обучить $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$, чтобы он был похож на истинный обратный шаг $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$

Можем найти $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ по теореме Байеса:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}) \cdot q(\mathbf{x}_{t-1})}{q(\mathbf{x}_t)}$$



$q(\mathbf{x}_t)$ – маргинальное распределение всех возможных зашумленных изображений на шаге t:

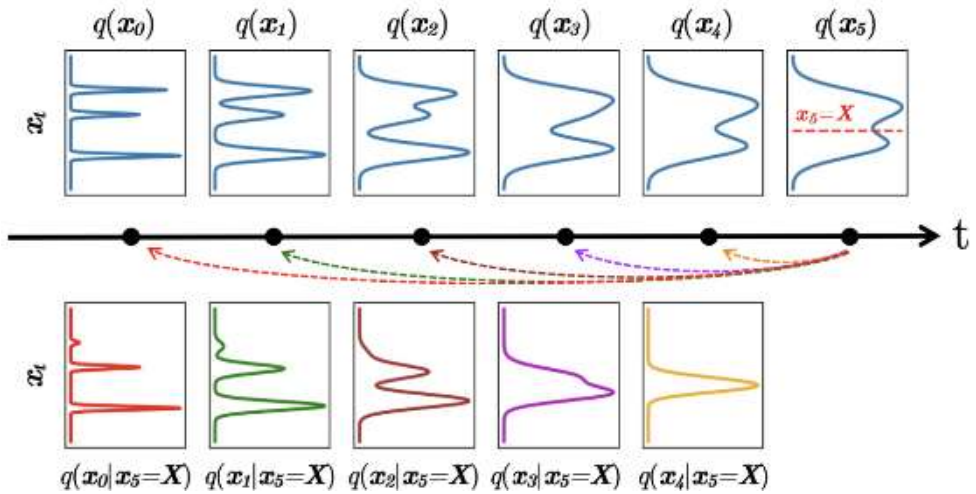
$$q(\mathbf{x}_t) = \int q(\mathbf{x}_t|\mathbf{x}_0) \cdot p_{data}(\mathbf{x}_0) d\mathbf{x}_0$$

Мы не можем вычислить этот интеграл из–за $p_{data}(\mathbf{x}_0)$

Теорема Феллера

Теорема **(*Feller*, 1949)**:

Если β_t на каждом шаге достаточно мал, то обратный процесс $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ тоже будет гауссовским



Это объясняет почему мы должны брать много шагов диффузии ($T \approx 1000$)

Будем учить декодер $p_{\boldsymbol{\theta}}$ предсказывать параметры $\mathbf{x}_{t-1}$, зная $\mathbf{x}_t$ и t:

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\mathbf{x}_t, t))$$

$ELBO\ for\ Diffusion\ Model$

Как и в VAE, будем обучать модель, максимизируя $ELBO$:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathbb{E}_{q(\boldsymbol{z}|\mathbf{x})} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{z})}{q(\boldsymbol{z}|\mathbf{x})} = \mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) \rightarrow \max_{\boldsymbol{\phi},\boldsymbol{\theta}}$$

Подставим наши распределения:

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_0, \mathbf{x}_{1:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} = \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})}$$

- В числителе $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$, в знаменателе $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ – напоминает KL –дивергенцию
- Попробуем перевернуть $q(\mathbf{x}_t|\mathbf{x}_{t-1}) \rightarrow q(\mathbf{x}_{t-1}|\mathbf{x}_t)$

Условное ядро

Рассмотрим условное ядро $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) \cdot q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}$$

- $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = q(\mathbf{x}_t|\mathbf{x}_{t-1}) -$ ядро перехода из $\mathbf{x}_{t-1}$ в $\mathbf{x}_t$ не зависит от $\mathbf{x}_0$
- $q(\mathbf{x}_{t-1}|\mathbf{x}_0) -$ ядро перехода из $\mathbf{x}_0$ в $\mathbf{x}_{t-1}$
- $q(\mathbf{x}_t|\mathbf{x}_0) -$ ядро перехода из $\mathbf{x}_0$ в $\mathbf{x}_t$

Подставим всё известное формулу*:*

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{\mathcal{N}(\sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1}, \beta_t \boldsymbol{I}) \cdot \mathcal{N}(\sqrt{\overline{\alpha}_{t-1}} \cdot \mathbf{x}_0, (1-\overline{\alpha}_{t-1})\boldsymbol{I})}{\mathcal{N}(\sqrt{\overline{\alpha}_t} \cdot \mathbf{x}_0, (1-\overline{\alpha}_t)\boldsymbol{I})} = \mathcal{N}(\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \cdot \boldsymbol{I})$$

Условное ядро

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{\mathcal{N}(\sqrt{1-\beta_t}\cdot\mathbf{x}_{t-1}, \beta_t\boldsymbol{I})\cdot\mathcal{N}(\sqrt{\overline{\alpha}_{t-1}}\cdot\mathbf{x}_0, (1-\overline{\alpha}_{t-1})\boldsymbol{I})}{\mathcal{N}(\sqrt{\overline{\alpha}_t}\cdot\mathbf{x}_0, (1-\overline{\alpha}_t)\boldsymbol{I})} = \mathcal{N}(\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\cdot\boldsymbol{I})$$

Параметры нового распределения:

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\cdot\mathbf{x}_t + \frac{\sqrt{\overline{\alpha}_{t-1}}(1-\alpha_t)}{1-\overline{\alpha}_t}\cdot\mathbf{x}_0$$

$$\tilde{\beta}_t = \frac{(1-\alpha_t)(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}$$

- Теперь у нас есть параметры распределения, которые должен предсказывать декодер $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$

ELBO for Diffusion Model

$$q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\cdot q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}$$

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})} = \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0)} =$$

$$\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T)\cdot p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)\cdot\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_1|\mathbf{x}_0)\cdot\prod_{t=2}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0)} = \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T)\cdot p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)\cdot\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_1|\mathbf{x}_0)\cdot\prod_{t=2}^{T} \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\cdot q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}} =$$

$$\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T)\cdot p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)\cdot\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_T|\mathbf{x}_0)\cdot\prod_{t=2}^{T} q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}$$

$$\prod_{t=2}^{T} \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\cdot q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} = \frac{q(\mathbf{x}_1|\mathbf{x}_2,\mathbf{x}_0)\cdot q(\mathbf{x}_2|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_2|\mathbf{x}_3,\mathbf{x}_0)\cdot q(\mathbf{x}_3|\mathbf{x}_0)}{q(\mathbf{x}_2|\mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_3|\mathbf{x}_4,\mathbf{x}_0)\cdot q(\mathbf{x}_4|\mathbf{x}_0)}{q(\mathbf{x}_3|\mathbf{x}_0)}$$

ELBO for Diffusion Model

$$\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T) \cdot p_\theta(\mathbf{x}_0|\mathbf{x}_1) \cdot \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_T|\mathbf{x}_0) \cdot \prod_{t=2}^{T} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} =$$

$$\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1) + \log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \right] =$$

$$\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1) + \mathbb{E}_{q(\mathbf{x}_T|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} + \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{x}_{t-1},\mathbf{x}_t|\mathbf{x}_0)} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} =$$

$$\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1) - KL(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)) - \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} KL(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t))$$

$$\mathbb{E}_{q(\mathbf{x}_{t-1},\mathbf{x}_t|\mathbf{x}_0)} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\mathbb{E}_{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \right]$$

ELBO for Diffusion Model

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \underbrace{\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)}_{\mathcal{L}_0} - \underbrace{KL(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{\mathcal{L}_T} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} KL(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}_{\mathcal{L}_t}$$

$\mathcal{L}_0$ – единственный шаг декодера $p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)$, отвечает за восстановление изображения $\mathbf{x}_0$ из $\mathbf{x}_1$:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1) = \log \mathcal{N}\left(\mathbf{x}_0|\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_1), \boldsymbol{\sigma}^2_{\boldsymbol{\theta},t}(\mathbf{x}_1)\right), \qquad \mathbf{x}_1 \sim q(\mathbf{x}_1|\mathbf{x}_0)$$

$\mathcal{L}_T$ – KL-дивергенция между двумя необучаемыми распределениями:

$$KL(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)) = KL\left(\mathcal{N}\left(\sqrt{\overline{\alpha}_T} \cdot \mathbf{x}_0, (1 - \overline{\alpha}_T)\right)||\mathcal{N}(\mathbf{0}, \boldsymbol{I})\right) = const$$

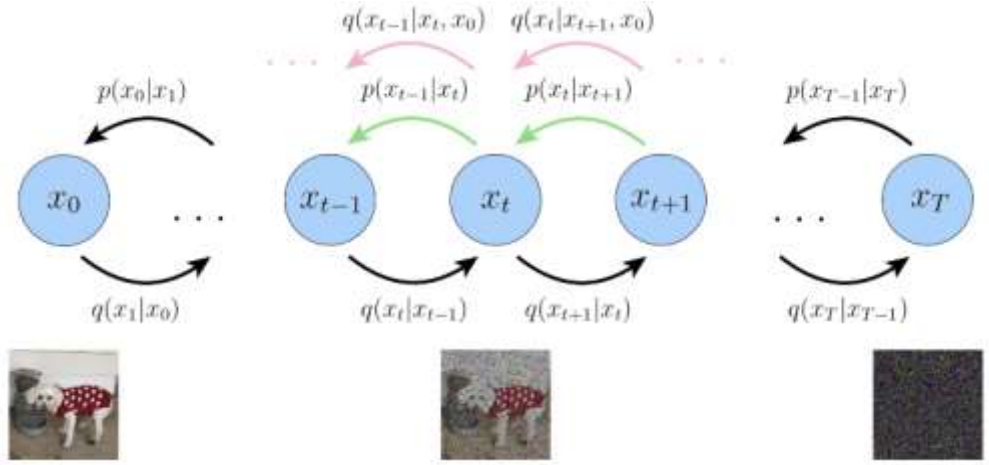
- $\mathcal{L}_t$ – потеря на каждом шаге t
- Каждый член заставляет нейросеть $p_{\boldsymbol{\theta}}$ быть похожим на истинное распределение q

ELBO for Diffusion Model

$$\mathcal{L}_t = \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} KL(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t))$$

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

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1}|\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t), \boldsymbol{\sigma}^2_{\boldsymbol{\theta},t}(\mathbf{x}_t)\right)$$



ELBO for Diffusion Model

- Не будем обучать дисперсию $\boldsymbol{\sigma}^2_{\boldsymbol{\theta},t}$
- Зафиксируем $\boldsymbol{\sigma}^2_{\boldsymbol{\theta},t}$, приравняв к известной дисперсии $\tilde{\beta}$:

$$\sigma^2_{\theta,t} = \tilde{\beta}_t \cdot \boldsymbol{I}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1}|\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t), \sigma^2_{\theta,t}(\mathbf{x}_t)\right) = \mathcal{N}\left(\mathbf{x}_{t-1}|\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t), \tilde{\beta}_t \cdot \boldsymbol{I}\right)$$

KL –дивергенция между двумя гауссианами с одинаковой дисперсией – это MSE между их средними:

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} KL\left(\mathcal{N}\left(\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \boldsymbol{I}\right) || \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t), \tilde{\beta}_t \cdot \boldsymbol{I}\right)\right) =$$

$$= \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\tilde{\beta}_t} \left|\left|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t)\right|\right|^2\right]$$

Training

Обучение:

- Берем объект $\mathbf{x}_0 \sim p_{data}(\mathbf{x})$
- Получаем зашумленный объект $\mathbf{x}_t = \sqrt{\overline{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \overline{\alpha}_t} \cdot \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Вычисляем лосс*:*

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1) - KL(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)) - \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\tilde{\beta}_t} \left|\left| \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t) \right|\right|^2 \right]$$

Inference

Сэмплинг:

- Сэмплируем $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Делаем обратный проход $range(T, 1)$:

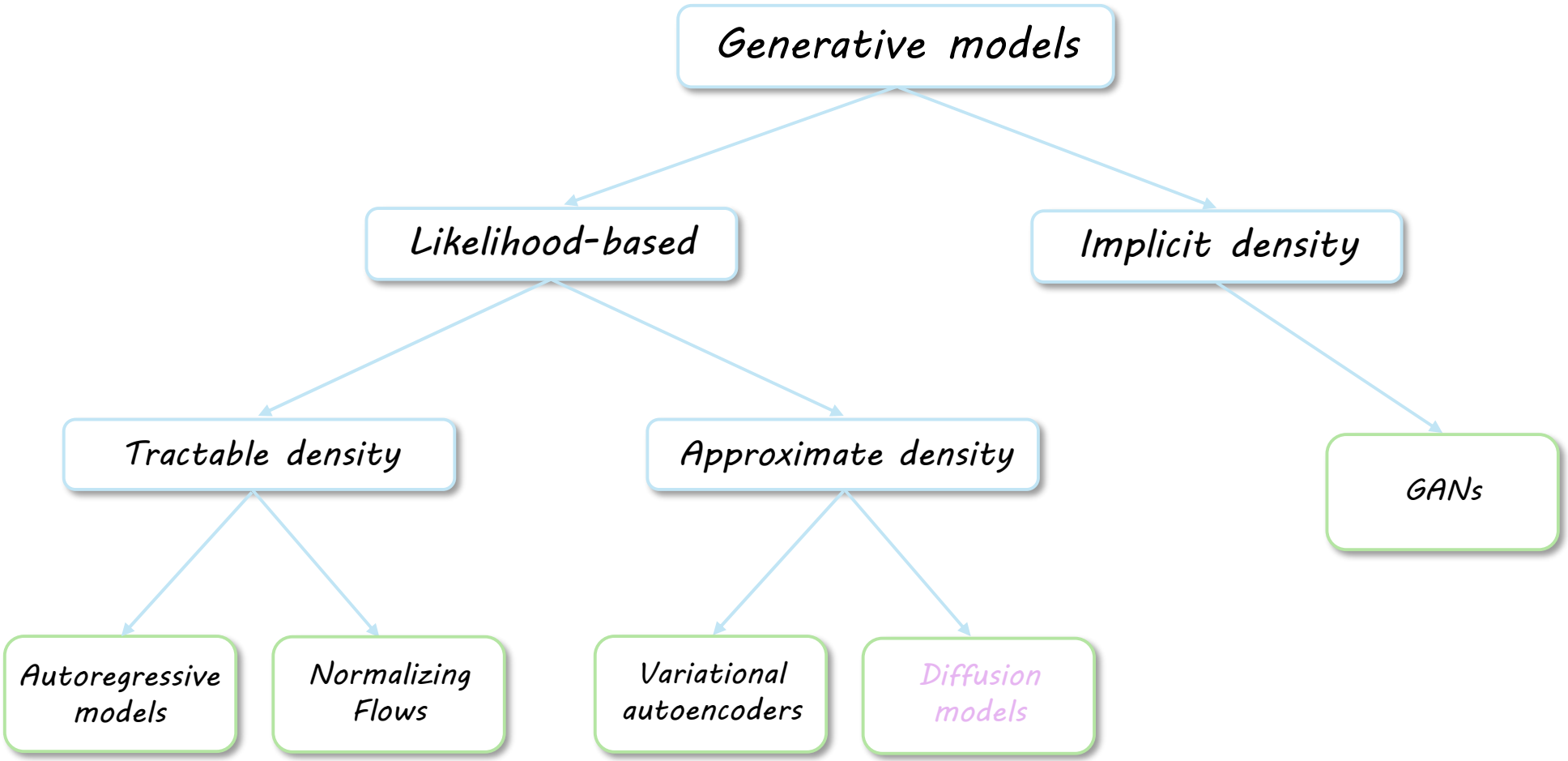
  Подаем $\mathbf{x}_t$ и t в обученную нейросеть**,** получаем $\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t)$

  Сэмплируем $\mathbf{x}_{t-1}$ из обратного процесса $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$:

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \sqrt{\tilde{\beta}_t} \cdot \boldsymbol{\epsilon}$$

Denoising Diffusion Probabilistic Model

Зоопарк генеративных моделей



Reparameterization

Проблема:

Обучать нейросеть предсказывать $\boldsymbol{\mu_\theta}$ – не всегда стабильно

Существует линейная зависимость между $\mathbf{x}_0$, $\mathbf{x}_t$ и $\boldsymbol{\epsilon}$:

$$\mathbf{x}_t = \sqrt{\overline{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \overline{\alpha}_t} \cdot \boldsymbol{\epsilon} \qquad \mathbf{x}_0 = \frac{1}{\sqrt{\overline{\alpha}_t}}(\mathbf{x}_t - \sqrt{1 - \overline{\alpha}_t} \cdot \boldsymbol{\epsilon})$$

Подставим в формулу для $\tilde{\boldsymbol{\mu}}_t$:

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\alpha_t}(1 - \overline{\alpha}_{t-1})}{1 - \overline{\alpha}_t} \cdot \mathbf{x}_\mathrm{t} + \frac{\sqrt{\overline{\alpha}_{t-1}}(1 - \alpha_t)}{1 - \overline{\alpha}_t} \cdot \mathbf{x}_0 =$$

$$\frac{\sqrt{\alpha_t}(1 - \overline{\alpha}_{t-1})}{1 - \overline{\alpha}_t} \cdot \mathbf{x}_\mathrm{t} + \frac{\sqrt{\overline{\alpha}_{t-1}}(1 - \alpha_t)}{1 - \overline{\alpha}_t} \cdot \frac{1}{\sqrt{\overline{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1 - \overline{\alpha}_t} \cdot \boldsymbol{\epsilon}\right) = \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{(1 - \alpha_t)}{\sqrt{\alpha_t(1 - \overline{\alpha}_t)}} \cdot \boldsymbol{\epsilon}$$

Reparameterization

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{(1-\alpha_t)}{\sqrt{\alpha_t(1-\overline{\alpha}_t)}} \cdot \boldsymbol{\epsilon}$$

Сделаем репараметризацию и заставим нейросеть иметь такую же форму:

$$\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t) = \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{(1-\alpha_t)}{\sqrt{\alpha_t(1-\overline{\alpha}_t)}} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{t}}(\mathbf{x}_t)$$

Теперь наша модель будет предсказывать шум $\boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{t}}(\mathbf{x}_t)$, а не среднее $\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t)$:

$$\mathcal{L}_t = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\frac{(1-\alpha_t)^2}{2\tilde{\beta}_t\alpha_t(1-\overline{\alpha}_t)}\left|\left|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{t}}(\mathbf{x}_t)\right|\right|^2\right] =$$

$$\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\frac{(1-\alpha_t)^2}{2\tilde{\beta}_t\alpha_t(1-\overline{\alpha}_t)}\left|\left|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{t}}(\sqrt{\overline{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t} \cdot \boldsymbol{\epsilon})\right|\right|^2\right]$$

Функция потерь $DDPM$

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1) - KL(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)) - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} KL(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}_{\mathcal{L}_t}$$

$$\mathcal{L}_t = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})} \left[\frac{(1-\alpha_t)^2}{2\tilde{\beta}_t \alpha_t (1-\overline{\alpha}_t)} \left|\left| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{t}}(\sqrt{\overline{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t} \cdot \boldsymbol{\epsilon}) \right|\right|^2 \right]$$

Отбросим весовой коэффициент:

$$\mathcal{L}_t = \mathbb{E}_{t\sim U(2,T)} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})} \left[\left|\left| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{t}}(\sqrt{\overline{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t} \cdot \boldsymbol{\epsilon}) \right|\right|^2 \right]$$

Training

Обучение:

- Берем объект $\mathbf{x}_0 \sim p_{data}(\mathbf{x})$
- Выбираем номер шага $t \sim U(2, T)$ и шум $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Получаем зашумленный объект $\mathbf{x}_t = \sqrt{\overline{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \overline{\alpha}_t} \cdot \boldsymbol{\epsilon}$
- Подаем $\mathbf{x}_t$ и t в нейросеть
- Вычисляем MSE между реальным $\boldsymbol{\epsilon}$ и предсказанным шумом $\boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{t}}$

Inference

Сэмплинг:

- Сэмплируем $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Делаем обратный проход $range(T, 1)$:

  Подаем $\mathbf{x}_t$ и t в обученную нейросеть**,** получаем $\boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)$

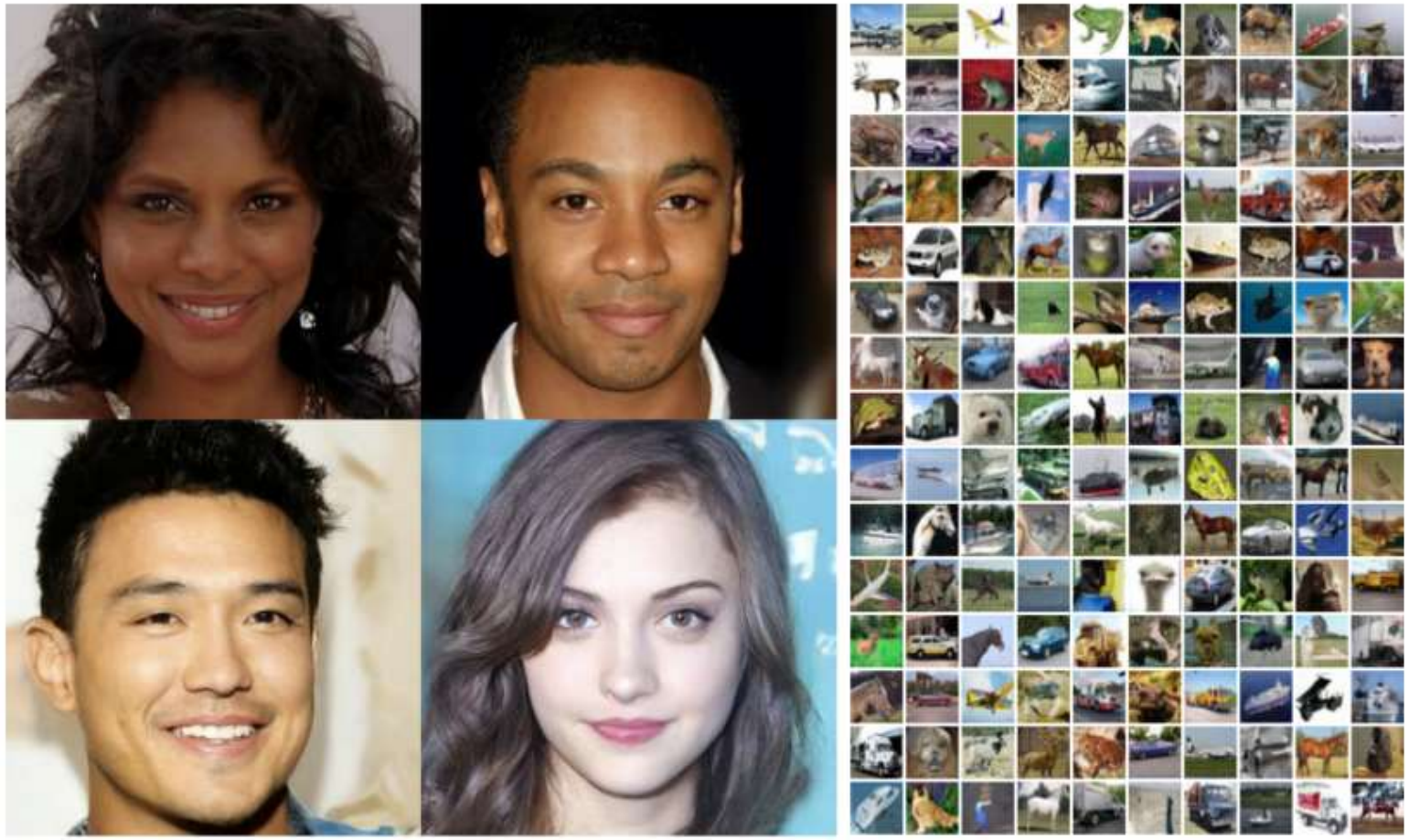
  Вычисляем среднее*:*

$$\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t) = \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{(1-\alpha_t)}{\sqrt{\alpha_t(1-\overline{\alpha}_t)}} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)$$

  Сэмплируем $\mathbf{x}_{t-1}$ из обратного процесса $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$:

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \sqrt{\tilde{\beta}_t} \cdot \boldsymbol{\epsilon}$$

DDPM Samples



Спасибо за внимание!