Генеративные модели

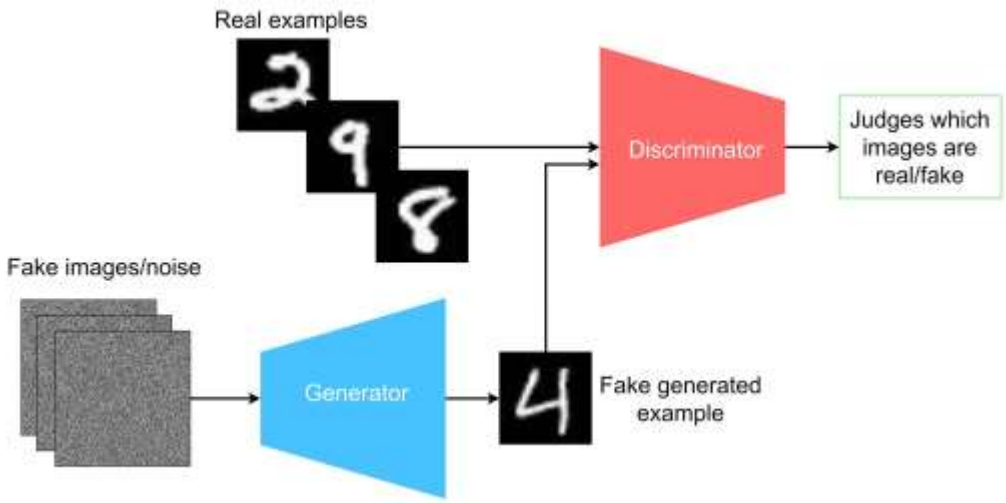
Лекция 7: *Score Matching*

Повторение

Generative Adversarial Networks

Обучение $GANs$ реализуется через состязание двух нейросетей*:*

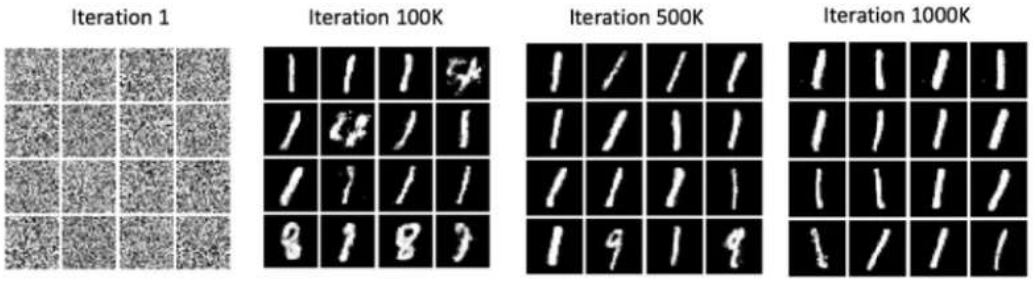
- ***Генератор*** $\boldsymbol{G}$: создаёт реалистичные данные $\mathbf{x} = G(\boldsymbol{z})$ из вектора $\boldsymbol{z} \sim p(\boldsymbol{z})$
- ***Дискриминатор*** $\boldsymbol{D}$: бинарный классификатор, который получает на вход объект $\mathbf{x}$ и выдает вероятность $D(\mathbf{x})$, что $\mathbf{x}-$ реальный



Mode Collapse

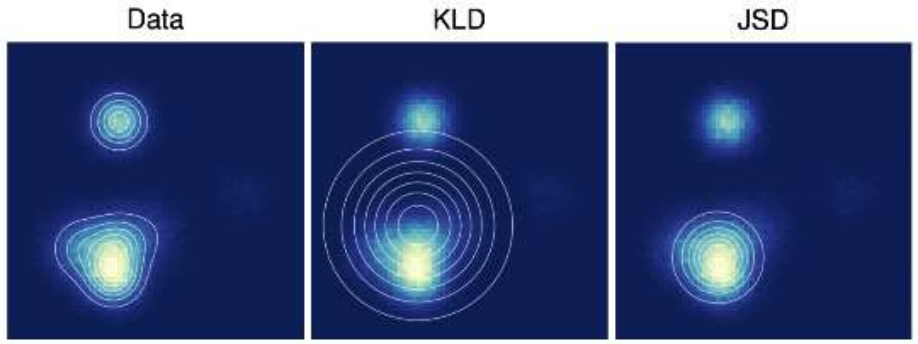
Коллапс мод (***mode collapse***) – главная практическая проблема *GANs*

Генератор «схлопывается» и начинает генерировать только одну или несколько мод распределения



Причина:

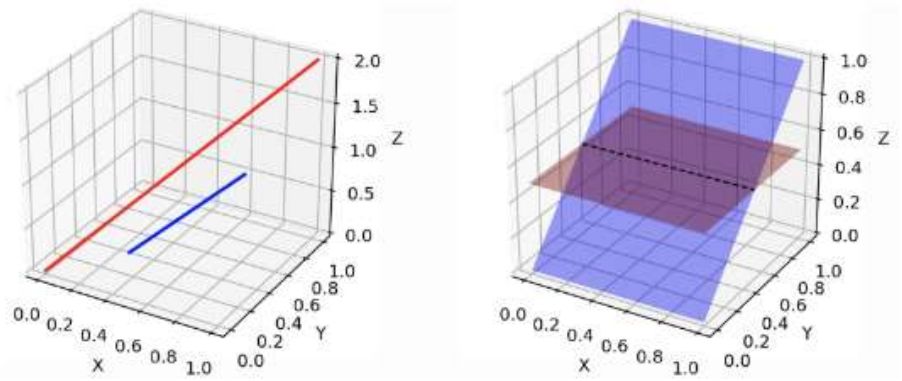
- Стандартный *GAN* минимизирует *JSD*, которая склонна к ***поиску мод*** (***mode seeking***)



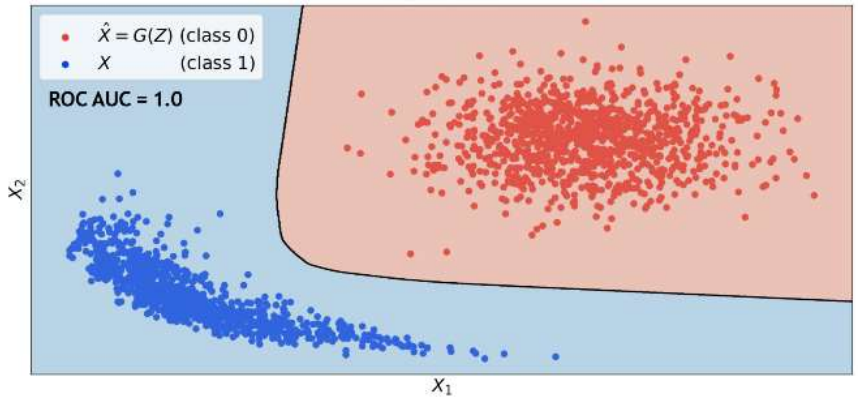
Генератору проще найти одну моду и идеально ее воспроизводить

Низкоразмерное многообразие

- В пространстве высокой размерности эти два многообразия почти гарантированно не пересекаются (***непересекающиеся носители, disjoint supports***)



- Поскольку эти многообразия не пересекаются, то дискриминатору очень легко найти разделяющую плоскость и стать идеальным классификатором



Низкоразмерное многообразие

Forward KL:

- В случае непересекающихся носителей в области, где $p_{data}(\mathbf{x}) > 0$, мы имеем $p_{\boldsymbol{\theta}}(\mathbf{x}) = 0$:

$$KL(p_{data}||p_{\boldsymbol{\theta}}) = \int p_{data}(\mathbf{x}) \log \frac{p_{data}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})} d\mathbf{x} \to \infty$$

Reverse KL:

- В случае непересекающихся носителей в области, где $p_{\boldsymbol{\theta}}(\mathbf{x}) > 0$, мы имеем $p_{data}(\mathbf{x}) = 0$:

$$KL(p_{\boldsymbol{\theta}}||p_{data}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x})}{p_{data}(\mathbf{x})} d\mathbf{x} \to \infty$$

Низкоразмерное многообразие

Jensen – Shannon:

$$JSD(p_{data} \,||\, p_{\boldsymbol{\theta}}) = \frac{1}{2} KL\left(p_{data}(\mathbf{x}) \,||\, \frac{p_{data}(\mathbf{x}) + p_{\boldsymbol{\theta}}(\mathbf{x})}{2}\right) + \frac{1}{2} KL\left(p_{\boldsymbol{\theta}}(\mathbf{x}) || \, \frac{p_{data}(\mathbf{x}) + p_{\boldsymbol{\theta}}(\mathbf{x})}{2}\right) =$$

$$\frac{1}{2}\int p_{data}(\mathbf{x}) \log \frac{p_{data}(\mathbf{x})}{p_m(\mathbf{x})} d\mathbf{x} + \frac{1}{2}\int p_{\boldsymbol{\theta}}(\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x})}{p_m(\mathbf{x})} d\mathbf{x} = \frac{1}{2}\int p_{data}(\mathbf{x}) \log \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x})/2} d\mathbf{x} +$$

$$+ \frac{1}{2}\int p_{\boldsymbol{\theta}}(\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})/2} d\mathbf{x} = \frac{1}{2}\log 2 + \frac{1}{2}\log 2 = \log 2$$

- Как только дискриминатор становиться идеальным, JSD дивергенция становится константой
- Обучение останавливается, поскольку градиенты становятся равными нулю

$Wasserstain\ GAN$

Мы не можем найти идеальную f, поэтому будем аппроксимировать ее с помощью нейросети критика $f_{\boldsymbol{\phi}}$ ($\boldsymbol{Critic}$)

Обеспечение Липшицевости:

- Будем принудительно обрезать (***clamped***) все веса в диапазон $[-c, c]$ после каждого шага градиентного спуска
- Это гарантирует, что функция будет K −Липшицевой

Множество наших нейросетей $f_{\boldsymbol{\phi}}$ − это подмножество всех K −Липшицевых функций f:

$$K \cdot W(\pi, p) = \max_{||f||_L \leq K} \left[\mathbb{E}_{\pi(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{p(\mathbf{x})} f(\mathbf{x})\right] \geq \max_{\boldsymbol{\phi} \in \boldsymbol{\Phi}} \left[\mathbb{E}_{\pi(\mathbf{x})} f_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbb{E}_{p(\mathbf{x})} f_{\boldsymbol{\phi}}(\mathbf{x})\right]$$

$Wasserstain\ GAN$

Функция потерь $\boldsymbol{GAN}$:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \left(\mathbb{E}_{p_{data}(\mathbf{x})} \log D_{\boldsymbol{\phi}}(\mathbf{x}) + \mathbb{E}_{p(\boldsymbol{z})} \log \left(1 - D_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z})) \right) \right)$$

Функция потерь $\boldsymbol{WGAN}$:

$$\min_{\boldsymbol{\theta}} W(\pi, p) \approx \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi} \in \boldsymbol{\Phi}} \left[\mathbb{E}_{\pi(\mathbf{x})} f_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbb{E}_{p(\boldsymbol{z})} f_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z})) \right]$$

Основные отличия:

- Дискриминатор $D_{\boldsymbol{\phi}}$ → Критик $f_{\boldsymbol{\phi}}$ выдает любое число (оценку)
- Параметры критика $\boldsymbol{\phi}$ ограничены

$WGAN - GP$

Weight Clipping – очень плохой способ заставить критика быть K –Липшицевым

Идея $\boldsymbol{WGAN - GP}$:

Вместо того, чтобы обрезать веса $\boldsymbol{\phi}$, будем наказывать критика, если он нарушает 1 –Липшицево ограничение

1 –Липшицевость означает, что норма градиента критика $||\nabla f_{\boldsymbol{\phi}}(\mathbf{x})|| \leq 1$ ***в любой точке***

- Мы не можем проверять градиент везде
- Авторы доказали, что достаточно проверить его на случайных точках между реальными x_{data} и поддельными x_{fake} данными

$WGAN - GP$

Gradient Penalty:

- Создаем случайный объект с помощью интерполяции:

$$\hat{\mathbf{x}} = \epsilon \cdot \mathbf{x}_{data} + (1 - \epsilon) \cdot \mathbf{x}_{fake}$$

- Вычисляем градиент критика $f_{\boldsymbol{\phi}}$ в этой точке $\hat{\mathbf{x}}$ и штрафуем его если $||\nabla f_{\boldsymbol{\phi}}(\hat{\mathbf{x}})||_2 \geq 1$

Функция потерь $\boldsymbol{WGAN - GP}$:

$$\mathcal{L} = \underbrace{\left[\mathbb{E}_{\pi(\mathbf{x})} f_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbb{E}_{p(\boldsymbol{z})} f_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))\right]}_{WGAN\ Loss} + \underbrace{\lambda \cdot \mathbb{E}_{\hat{\mathbf{x}}}\left[\left(||\nabla f_{\boldsymbol{\phi}}(\hat{\mathbf{x}})||_2 - 1\right)^2\right]}_{Gradient\ Penalty}$$

$StyleGAN$

Идея:

- Возьмем за основу $PG-GAN$ и научим модель делать контролируемую генерацию

Проблема:

- В обычных $GANs$ латентный вектор $\boldsymbol{z}-$ это черный ящик
- Хотим сделать распутанное пространство, где мы смогли бы контролировать конкретные черты изображения

В $StyleGAN$ предложили разделить черты на 3 уровня:

- ***Грубые*** $(\boldsymbol{Coarse})$ $(4\times4, 8\times8)-$ поза, форма лица
- ***Средние*** $(\boldsymbol{Middle})$ $(16\times16, 32\times32)-$ черты лица, детали прически
- ***Мелкие*** $(\boldsymbol{Fine})$ $(64\times64\rightarrow1024\times1024)-$ микро-детали, цвет глаз, морщины

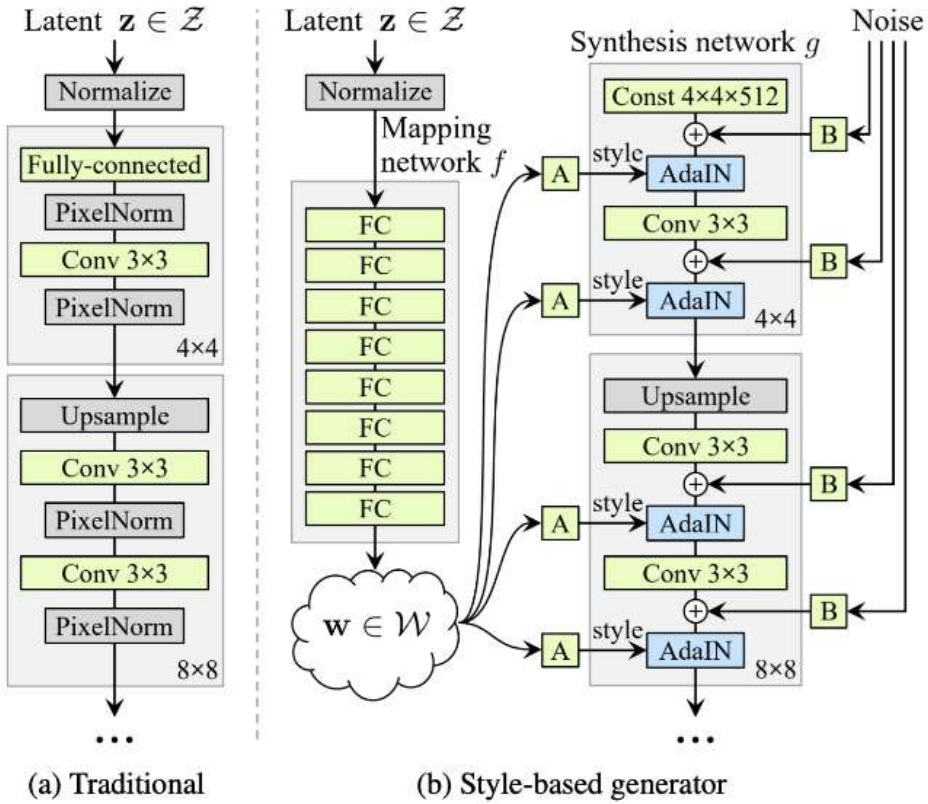
StyleGAN

Обычный GAN:

- Латентный вектор $\boldsymbol{z}$ подается на вход первому слою и проходит через всю сеть перемешиваясь

Style – based GAN:

- Генератор разделён на 2 сети
- ***Mapping Network*** – 8 слойный *MLP*, который распутывает $\boldsymbol{z}$ и превращает его в стилевой вектор $\boldsymbol{w}$
- ***Synthesis Network*** – сверточная сеть, которая рисует изображение, используя $\boldsymbol{w}$ как инструкцию



План

- *Energy Based Models*
- *Score Matching*
- *Denoising Score Matching*
- *Noise Conditioned Score Network*

$Energy - Based\ Models$

Проблемы генеративных моделей

Ограничения генеративных моделей сдерживают их выразительную мощность:

- ***Авторегрессионные модели*** требуют заданного порядка при генерации
- ***Нормализующие потоки*** требуют обратимые преобразования в скрытое пространство
- ***Вариационные автокодировщики*** более гибкие, но оптимизируют $ELBO$

Введем функцию энергии $E_{\boldsymbol{\theta}}(\mathbf{x})$, которая будет принимать на вход объект $\mathbf{x}$ и выдавать одно число – энергию

$Energy - Based\ Models$

Функция $E_{\boldsymbol{\theta}}(\mathbf{x})$ определяет ландшафт в пространстве данных*:*

- Низкая энергия (впадины) – наиболее вероятные данные
- Высокая энергия (холмы) – маловероятные данные

- Будем учить нейросеть $E_{\boldsymbol{\theta}}(\mathbf{x})$ так, чтобы впадины были там, где находятся реальные данные

EBM решает проблемы других моделей*:*

- $E_{\boldsymbol{\theta}}(\mathbf{x})$ не обязана быть обратимой
- $E_{\boldsymbol{\theta}}(\mathbf{x})$ не требует заданного порядка

$Energy - Based\ Models$

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

$Energy - Based\ Models$

Найдем градиент логарифмма плотности EBM:

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

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = -\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}$$

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

$$\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} = \frac{1}{Z_{\boldsymbol{\theta}}} \nabla_{\boldsymbol{\theta}} Z_{\boldsymbol{\theta}} = \frac{1}{Z_{\boldsymbol{\theta}}} \nabla_{\boldsymbol{\theta}} \int e^{-E_{\boldsymbol{\theta}}(\mathbf{x})}\, d\mathbf{x} = \frac{1}{Z_{\boldsymbol{\theta}}} \int \nabla_{\boldsymbol{\theta}} e^{-E_{\boldsymbol{\theta}}(\mathbf{x})}\, d\mathbf{x}$$

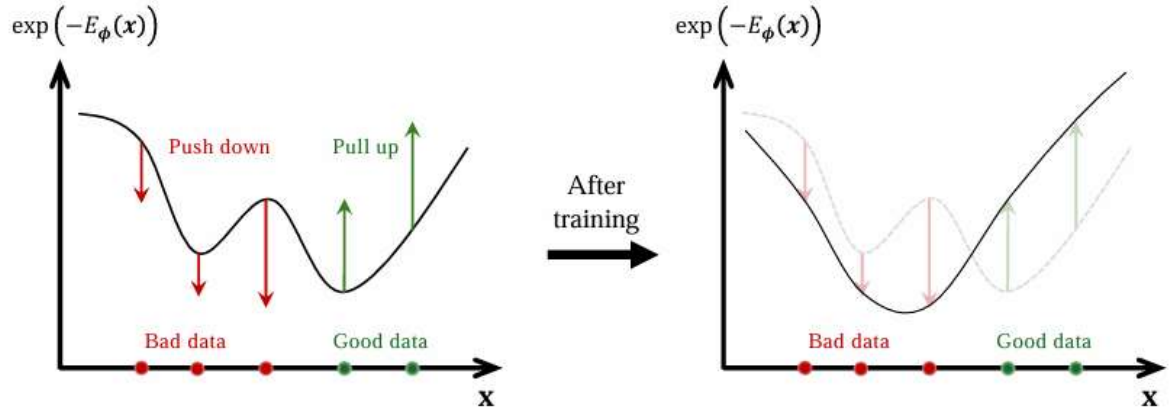
$$= \frac{1}{Z_{\boldsymbol{\theta}}} \int e^{-E_{\boldsymbol{\theta}}(\mathbf{x})} \cdot (-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x}))\, d\mathbf{x} = \int \underbrace{\frac{e^{-E_{\boldsymbol{\theta}}(\mathbf{x})}}{Z_{\boldsymbol{\theta}}}}_{p_{\boldsymbol{\theta}}(\mathbf{x})} \cdot (-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x}))\, d\mathbf{x} = -\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})} \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})$$

$Energy - Based\ Models$

В итоге мы получаем:

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

- $-\mathbb{E}_{p_{data}(\mathbf{x})} \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})$ работает с реальными данными, уменьшает энергию
- $\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})} \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})$ увеличивает энергию



$Energy - Based\ Models$

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

- 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***):

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

Найдем $score\ function$ для EBM:

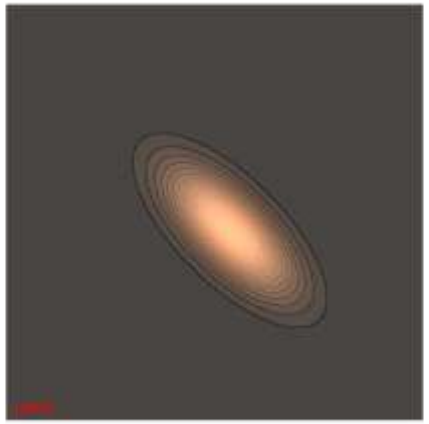
$$s_{\boldsymbol{\theta}}(\mathbf{x}) = \nabla_{\mathbf{x}} \log \frac{e^{-E_{\boldsymbol{\theta}}(\mathbf{x})}}{Z_{\boldsymbol{\theta}}} = \nabla_{\mathbf{x}} \log e^{-E_{\boldsymbol{\theta}}(\mathbf{x})} - \nabla_{\mathbf{x}} \log Z_{\boldsymbol{\theta}} = -\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x})$$

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

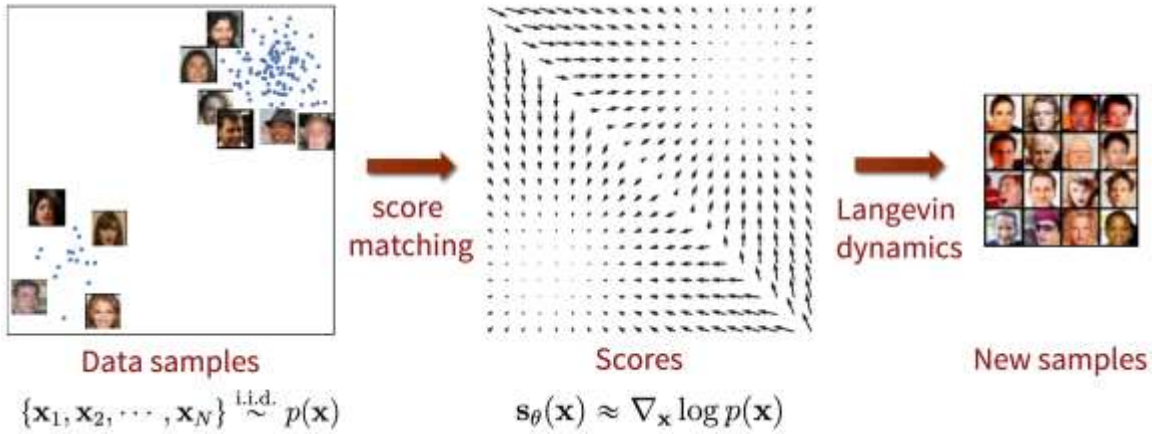
Score Matching

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

Факт:

- Если *score*-функции двух распределений $q(x)$ и $p(x)$ равны, то и сами распределения равны (с точностью до константы)

Дано: $\nabla_{\mathbf{x}} \log p(\mathbf{x}) = \nabla_{\mathbf{x}} \log q(\mathbf{x})$

Доказательство:

Проинтегрируем обе части выражения:

$$\log p(\mathbf{x}) = \log q(\mathbf{x}) + C$$

$$p(\mathbf{x}) = e^{C} \cdot e^{\log q(\mathbf{x})} = K \cdot q(\mathbf{x})$$

Score Matching

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

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

$$= \frac{1}{2}\int p_{data}(\mathbf{x})\,||\nabla_{\mathbf{x}}\log p_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}}\log p_{data}(\mathbf{x})||^2$$

- 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 $s_{data}(\mathbf{x})$

Score Matching

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

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

- $tr\left(\nabla_{\mathbf{x}} s_{\boldsymbol{\theta}}(\mathbf{x})\right)$ – след матрицы Якоби функции $s_{\boldsymbol{\theta}}(\mathbf{x})$

Доказательство $(1D)$:

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

$\frac{1}{2}s_{data}(x)^2 = const$ относительно параметров $\boldsymbol{\theta}$

Score Matching

Преобразуем удвоенное произведение:

$$\mathbb{E}_{p_{data}(\mathbf{x})}[s_{\boldsymbol{\theta}}(x) \cdot s_{data}(x)] = \int p_{data}(x) \cdot s_{\boldsymbol{\theta}}(x) \cdot \nabla_x \log p_{data}(x)\, dx =$$

$$= \int p_{data}(x) \cdot s_{\boldsymbol{\theta}}(x) \cdot \frac{1}{p_{data}(x)} \cdot \nabla_x\, p_{data}(x) dx = \int s_{\boldsymbol{\theta}}(x) \cdot \nabla_x\, p_{data}(x) dx =$$

$$= s_{\boldsymbol{\theta}}(x) \cdot p_{data}\,(x)\Big|_{-\infty}^{\infty} - \int \nabla_x\, s_{\boldsymbol{\theta}}(x) \cdot p_{data}\,(x) dx = - \int \nabla_x\, s_{\boldsymbol{\theta}}(x) \cdot p_{data}\,(x) dx =$$

$$= \mathbb{E}_{p_{data}(\mathbf{x})}\,[\nabla_x\, s_{\boldsymbol{\theta}}(x)]$$

- $s_{\boldsymbol{\theta}}(x) \cdot p_{data}\,(x)|_{-\infty}^{\infty} \to 0$

Score Matching

Мы показали*:*

$$D_F\left(p_{data}(x)||p_{\boldsymbol{\theta}}(x)\right) = \mathbb{E}_{p_{data}(x)}\left[\frac{1}{2}s_{\boldsymbol{\theta}}(x)^2 + \nabla_x\, s_{\boldsymbol{\theta}}(x)\right] + C$$

В многомерном случае:

- $s_{\boldsymbol{\theta}}(x)^2 = (\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}))^2 \rightarrow \left|\left|\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x})\right|\right|^2$
- $\nabla_x\, s_{\boldsymbol{\theta}}(x) = \nabla_{\mathbf{x}}^2 \log p_{\boldsymbol{\theta}}(\mathbf{x}) \rightarrow tr(\nabla_{\mathbf{x}}^2 \log p_{\boldsymbol{\theta}}(\mathbf{x}))$

В итоге*:*

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

Подход, использующий такую функцию потерь был назвал ***Implicit Score Matching***, поскольку избегает вычисления истинной $score$-функции $s_{data}(\mathbf{x})$

Denoising Score Matching

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 $q(\mathbf{x}_{\boldsymbol{\sigma}})$:

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

Проблема:

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 $p_{data}(\mathbf{x})$:

Теорема *Denoising Score Matching*

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

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

Доказательство:

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

$\nabla_{\mathbf{x}_{\boldsymbol{\sigma}}} \log q(\mathbf{x}_{\boldsymbol{\sigma}}) = const$ относительно параметров $\boldsymbol{\theta}$

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:

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

Подставим в нашу функцию потерь*:*

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

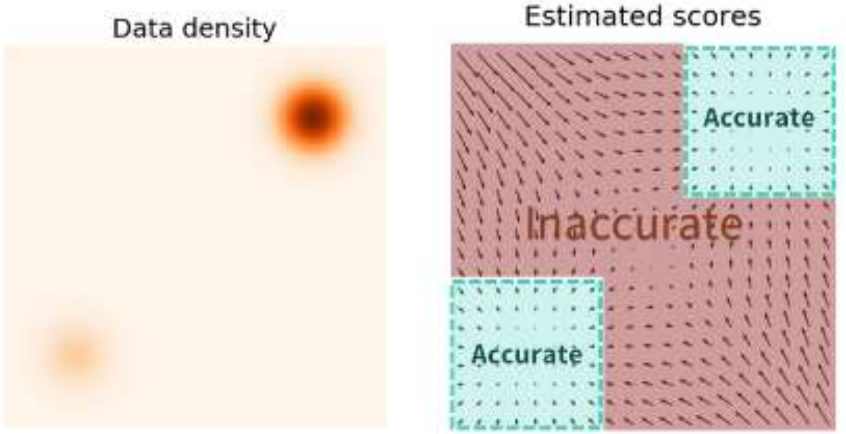
- Нейросеть $s_{\boldsymbol{\theta}}(\mathbf{x}_{\boldsymbol{\sigma}})$ учится напрямую предсказывать шум $\boldsymbol{\epsilon}$, который был добавлен к $\mathbf{x}$

Noise Conditioned Score Network

Проблема *Denoising Score Matching*

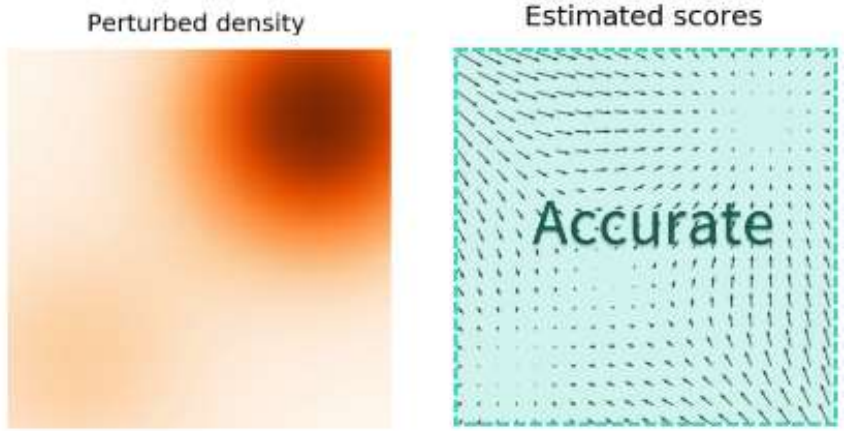
Если σ маленький:

- $q(\mathbf{x}_{\boldsymbol{\sigma}}) \approx p_{data}(\mathbf{x})$
- 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

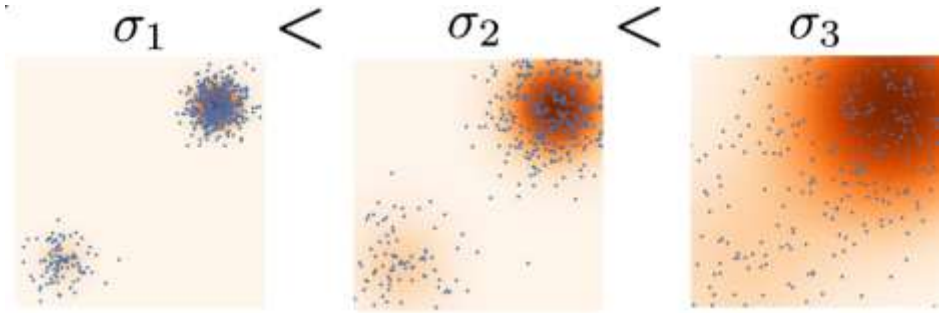
Идея:

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 $\frac{1}{\sigma_t^2}$

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

Обучение $NCSN$

Алгоритм обучения:

- Берем сэмпл $\mathbf{x}_0 \sim p_{data}(\mathbf{x})$
- Случайно выбираем уровень шума $t \sim U(1, T)$
- Сэмплируем шум $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$
- Создаем зашумленный сэмпл $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \cdot \boldsymbol{\epsilon}$
- Вычисляем лосс $\mathcal{L}_{\boldsymbol{\theta},\, t} = \sigma_t^2 \left\| s_{\boldsymbol{\theta},\sigma_t}(\mathbf{x}_t) + \frac{\boldsymbol{\epsilon}}{\sigma_t} \right\|^2$

Annealed Langevin Dynamics

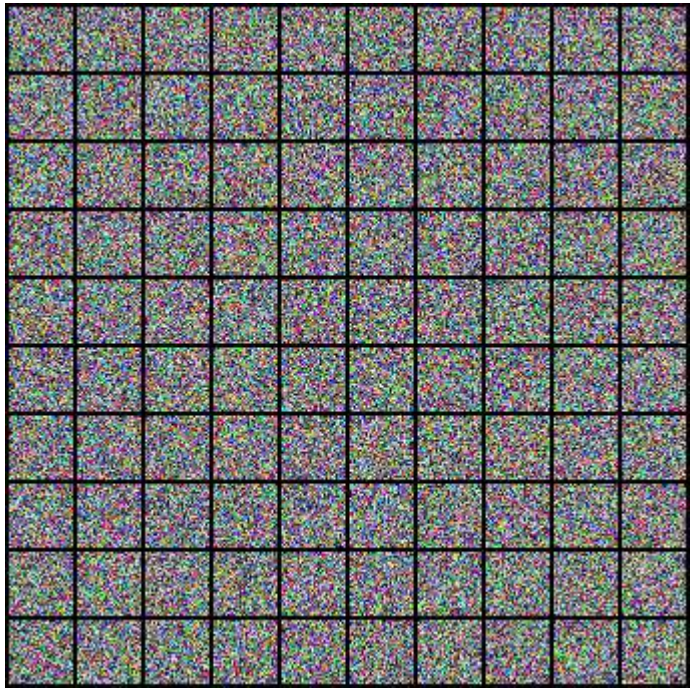
Для сэмплирования из модели используют ***динамику Ланжевена с отжигом*** (***Annealed Langevin Dynamics***):

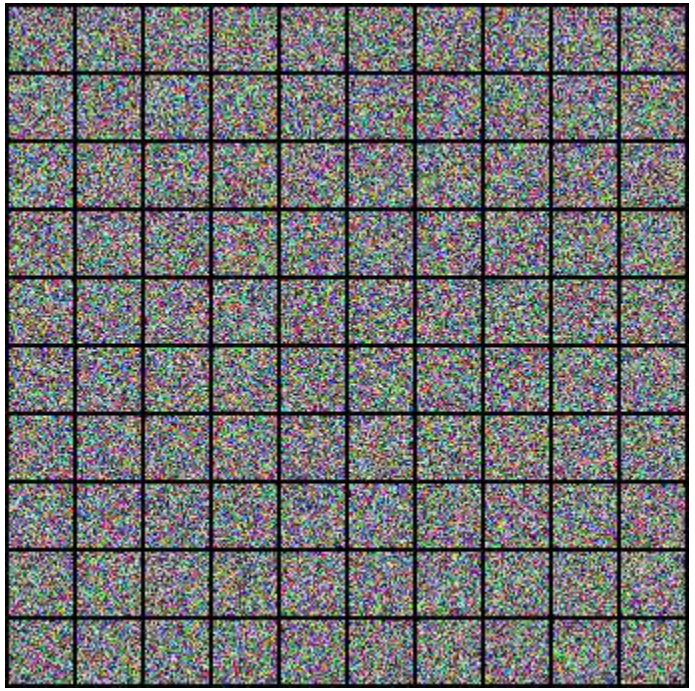
- Начинаем с шума $\mathbf{x}_T \sim \mathcal{N}(0, \sigma_T^2 \boldsymbol{I})$
- Для текущего шума σ_t применяем L шагов динамики Ланжевена*:*

$$\mathbf{x}_l = \mathbf{x}_{l-1} + \frac{\eta}{2} s_{\boldsymbol{\theta},\sigma_t}(\mathbf{x}_{l-1}) + \sqrt{\eta_t}\boldsymbol{\epsilon}_l$$

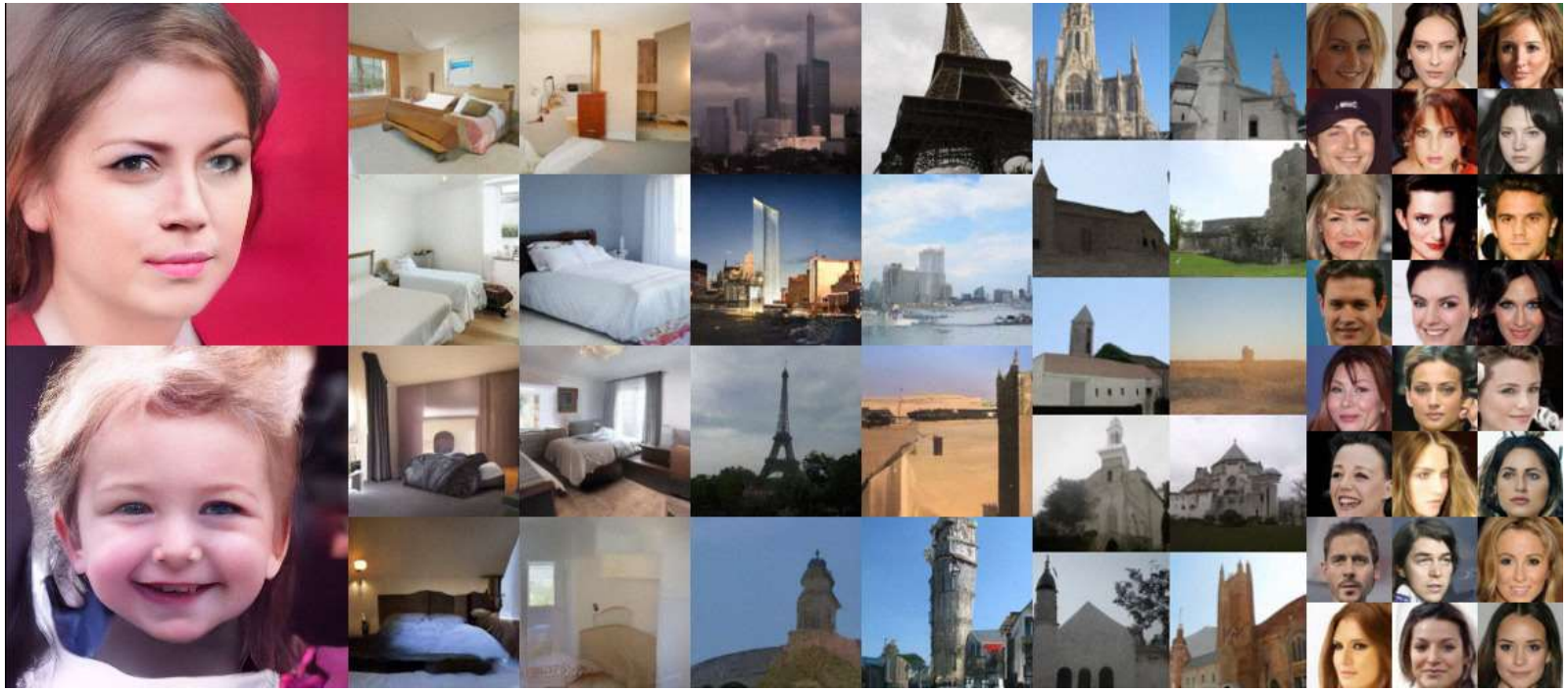
- После L шагов выход $\mathbf{x}_L$ становится инициализацией $\mathbf{x}_{t-1}$ для следующего уровня шума σ_{t-1}

Annealed Langevin Dynamics for NCSN





NCSN Samples



Спасибо за внимание!