# Отчет по воспроизведению статьи "Generative Modeling by Estimating Gradients of the Data Distribution"

Петр Жижин
Факультет Компьютерных наук
ВШЭ
piter.zh@gmail.com

Даяна Савостьянова
Факультет Компьютерных наук
ВШЭ
dayanamuha@gmail.com

## Abstract

Данная работа включает в себя результаты попытки воспроизведения статьи "Generative Modeling by Estimating Gradients of the Data Distribution". Финальной целью является получение схожих со статьей результатов. На данный момент работа содержит проверку начальных экспериментов статьи.

Работа устроена следующим образом, для начала рассмотрим статью, которая является основой для этой работы, далее рассмотрим нашу реализацию определенных частей статьи, подведем результаты о возможности аопспроизведения каждой из частей.

## 1 Обзор статьи

В обозреваемой статье [1] предлагается вариант генеративной модели, использующей динамику Ланжевена основанную на оценке градиентов распределения данных с помощью score matching [1]. Часто выясняется, что данные лежат в низкоразмерных многообразиях, в таких случаях градиент невозможно определить вне данного многообразия. Поэтому в статье данные возмущаются различным шумом из Гауссовского распределения.

### 1.1 Лосс

Для сэмплирования из распределения данных $p_{data}(x)$ при помощи алгоритма Ланжевена, необходимо уметь строить оценку на градиент $s_\theta(x) \approx \nabla_x \log p_{data}(x)$.

Если бы у нас была известна плотность, то её аппроксимацию можно было бы посчитать, прооптимизировав следующий функционал:

$$\mathbb{E}_{p_{data}(x)} \left[ |s_\theta(x) - \nabla_x \log p_{data}(x)|_2^2 \right] \to \min_\theta$$

Однако на практике мы не можем знать плотность распределения, а знаем лишь только некоторую выборку, которая была сгенерирована по данным.

В статье [2] было показано, что эту формулу можно так же представить и в другом виде, который можно использовать для подсчёта.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

Мы будем использовать две переформулировки данного уравнения в разных задачах. Первая из них называется Sliced Score Matching:

$$\mathbb{E}_{p_{data}(x)}\mathbb{E}_{p_v}\left[v^T\nabla_x s_\theta(x)v + \frac{1}{2}|s_\theta(x)|_2^2\right] \tag{1}$$

Для подсчёта этой формулы, мы генерируем случайные вектора $v$. Например, это могут быть стандартные гауссовские вектора.

$\nabla_x s_\theta(x)$ — это гессиан логарифма плотности распределений. Так как напрямую посчитать гессиан для любой достаточно большой модели было бы практически невозможно, то он напрямую не считается. В той же статье было показано, как при помощи любой библиотеки автоматического подсчёта градиента можно посчитать этот лосс, не считая при этом гессиан.

**Algorithm 1** Sliced Score Matching

Require: $\mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}), \mathbf{x}, \mathbf{v}$
1: $\mathbf{v}^\intercal\nabla_{\mathbf{x}}\mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}) \leftarrow \texttt{grad}(\mathbf{v}^\intercal\mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}), \mathbf{x})$
2: $J \leftarrow \frac{1}{2}(\mathbf{v}^\intercal\mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}))^2$ (or $J \leftarrow \frac{1}{2}\|\mathbf{s}_m(\mathbf{x};\boldsymbol{\theta})\|_2^2$)
3: $J \leftarrow J + \mathbf{v}^\intercal\nabla_{\mathbf{x}}\mathbf{s}_m(\mathbf{x};\boldsymbol{\theta})\mathbf{v}$
return $J$

Для данного алгоритма значение $\mathbf{s}_m(\mathbf{x};\boldsymbol{\theta})$ получается оценкой градиента нейронной сетью.

Вторая называется Denoising score matching:

$$\mathbb{E}_{q(\tilde{x}|x)p_{data}(x)}\left[|s_\theta(\tilde{x}) - \nabla_x \log q_\sigma(\tilde{x}|x)|_2^2\right] \tag{2}$$

Тут суть в том, чтобы в данные добавлялся шум $q(\tilde{x}|x)$ с известной плотностью, матожиданием равным $x$ и низкой дисперсией. За счет привнесения шума с известным распределением DSM требует в 4 раза меньше операций, чем SSM согласно [1] .

## 2 Эксперимент на смеси гауссовских распределений

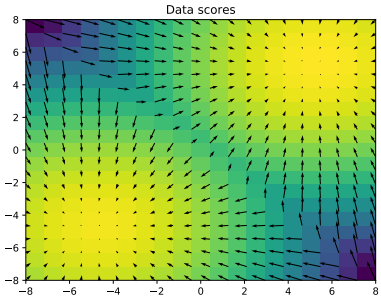
### 2.1 Неверная оценка градиента правдоподобия выборки

Повторим эксперимент из статьи, чтобы показать, что Score Matching лосс адекватно оценивает градиент только в областях высокой плотности. В качестве модели был использован MLP (Multi-Layer Perceptron) с тремя слоями. Размером скрытого слоя 128, функция активации Softplus. Данные генерировались из смеси распределений:

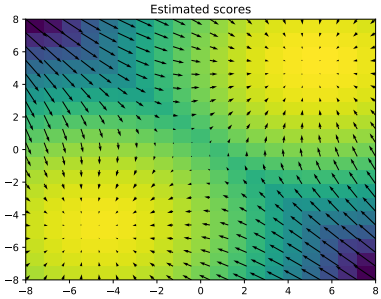
$$\frac{1}{5}\mathcal{N}\left((-5,-5), I\right) + \frac{4}{5}\mathcal{N}\left((5,5), I\right)$$

В качестве оптимизатора был взят Adam с lr = 0.001 с размером батча 128. Модель училась 10000 итераций.

В качестве лосса для Score Matching тут использовался Sliced Score Matching с генерацией только одного случайного вектора $v$ для каждого элемента выборки.




(a) $\nabla_x \log p_{data}(x)$ — истинный градиент логарифма правдоподобия




(b) $s_\theta(x)$ — оценка на $\nabla_x \log p_{data}(x)$

Рис. 1: Сравнение градиента истинного логарифма правдоподобия с его оценкой MLP сетью, красные области соответствуют области с высоким значеием плотности рассматриваемого распределения

На Рис. 1 мы можем видеть, что нейронная сеть адекватно оценивает градиент $\nabla_x \log p_{data}(x)$ только в области вокруг мод распределения. Стрелки указывают в то же направление, что и настоящий градиент плотности.

Значения ожидаемо отличаются в области, где плотность мала.

Данный результат полностью сходится с тем, что было получено в статье.

Стоит отметить, что именно поэтому некорректно было бы использовать обычный алгоритм Ланжевена. Плотность рассматриваемого распределения везде ненулевая. Рассмотрим это в следующем пункте.

### 2.2 Динамика Ланжевена

Хотим посмотреть на разницу в работе обычного Ланжевена и Ланжевена с отжигом. В алгоритме будем использовать для подсчета честные градиенты. Для этого генерировать данные будем как в предыдущем пункте из смеси Гауссовских распределений. Рис. 2a В качестве начальных точек для обеих динамик будем брать равномерно выбранные точки из квадрата $[-8, 8] \times [-8, 8]$. Для обычного Ланжевена возьмем 1000 шагов, $\varepsilon = 0.1$. Рис. 2b. Для Ланжевена с отжигом (алгоритм 2) 100 шагов, $\varepsilon = 0.1$. В статье предлагается взять $\sigma_i$ из геометрической прогрессии, где $\sigma_1 = 10, \sigma_{10} = 0.1$. Отметим, что при данных параметрах предложенный в статье алгоритм динамики Ланжевена с отжигом не сходится. После некоторых экспериментов с параметрами пришли к следующим параметрам: $\sigma_i$ из геометрической прогрессии, где $\sigma_1 = 20, \sigma_{10} = 0.7$. Рис. 2c.

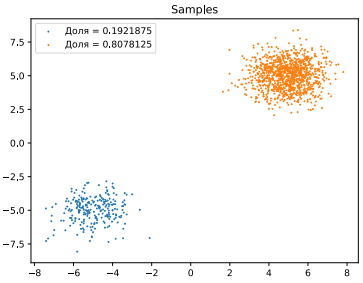
Заметим, что обычный Ланжевен плохо угадывает доли событий по компонентам смеси. Это случается из-за того, что при взятиии градиента $\log p_{data}$ логарифм веса является константным слагаемым, а значит не участвует в финальном выражении.С другой стороны Ланжевен с отжигом смог уловить соотношение между компонентами, в связи с меняющейся дисперсией шума: более весомая часть смеси более устойчива по мере уменьшения дисперсии шума.

Отметим, что качество работы последнего метода зависят от выбора сигмы. Метод

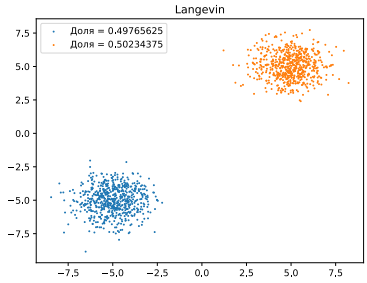
**Algorithm 2** Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$.
Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to $L$ do
  $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ ▷ $\alpha_i$ is the step size.
  for $t \leftarrow 1$ to $T$ do
    Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\, \mathbf{z}_t$
  end for
  $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$

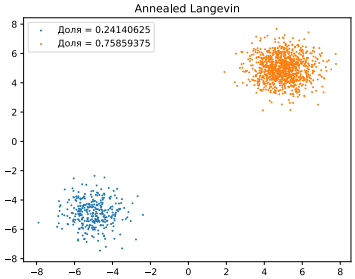
может как не сойтись, так и выдавать
результаты в той же мере неверные, как и обычный Ланжевен.




(a) Сэмплирование




(b) Динамика Ланжевина




(c) Динамика Ланжевена с отжига

Данный результат в целом сходится с тем, что было получено в статье, но стоит отметить, что параметры в статье указаны неподходящие.

## 3 Эксперементы на картинках

### 3.1 Модель

Для оценки градиента правдоподобия данных по картинке используется модель 4-х каскадного RefineNet [3] с модификациями.

К RefineNet были применены следующие модификации:

1. Для снижения разрешения для создания входов блоков RefineNet используется не даунскейлинг, а свёртка с dilation. Для создания входа для второго блока RefineNet дополнительно используется average пулинг с размером ядра 2 и страйдом 2. Параметр dilation равен 1 для входа первого блока RefineNet, для каждого последующего входа dilation умножается на 2.
2. Вместо макс-пулинга в блоке chained residual pooling используется average-пулинг.
3. Для нормализации вместо батч-нормализации используется условная инстанс нормализация. Обуславливание необходимо для того, чтобы учитывать уровень зашумлённости изображения и более подробно будет описано позднее в разделе 3.2
4. Вместо ReLU активаций используется активационная функция ELU со стандартными параметрами.
5. Базовое количество каналов в RefineNet равно 128 для CIFAR и 64 для MNIST. В первом блоке RefineNet используется базовое количество каналов, в остальных — удвоенное количество базовых каналов.

### 3.2 Обуславливание нейронной сети на параметр $\sigma$

Для корректной работы алгоритма Ланжевена с отжигом, необходимо каким-то образом обуславливать модель на уровень добавляемого в данные шума (в алгоритме используется оценка $s_\theta(x, \sigma)$, которая должна зависеть от $\sigma$). Для этого мы повторяем тот же метод обуслаливания, который применялся в статье. Он называется CondInstanceNorm++ и работает следующим образом:

Пусть $x$ — это картинка с $C$ каналов. $\mu_k$ — это среднее значение яркости пикселя в изображении $x$ для канала $k$. $s_k$ — это стандартное отклонение яркости пикселя на канале $k$. Так же считаются $m$ и $v$ — оценки среднего и стандартного $\mu_k$ по всем каналам $k = 1 \dots C$. Обуславливание на уровень шума происходит таким образом, что для разного $\sigma_i$ выбираются разные параметры нормализации. Тогда нормализованное изображение $z_k$ задаётся следующим образом:

$$z_k = \gamma[i, k]\frac{x_k - \mu_k}{s_k} + \beta[i, k] + \alpha[i, k]\frac{\mu_k - m}{v}$$

$i$ — это индекс внедряемого шума. $\gamma \in \mathbb{R}^{L\times C}$, $\beta \in \mathbb{R}^{L\times C}$, $\alpha \in \mathbb{R}^{L\times C}$

$L$ — это количество различных шумов, которые мы внедряем в модель при отжиге.

### 3.3 Лосс для обучения генерации картинок

В качестве лосса для обучения использовался Denoising Score Matching. Зашумление изображений проводилось добавкой к изображению нормального распределения с нулевым матожиданием и стандартным отклонением $\sigma$.

Так как форма шума заранее известна (нормальный шум), то градиент логарифма правдоподобия можно легко выписать, используя формулу 2:

$$\mathbb{E}_{N(\tilde{x}|x,\sigma^2 I)p_{data}(x)}\left[|s_\theta(\tilde{x}, \theta) + \frac{\tilde{x} - x}{\sigma^2}|_2^2\right]$$

Именна эта формула (точнее её оценка по Монте-карло) и используется для обучения модели. Для каждого примера в случайном батче генерируется один зашумлённый пример со случайным уровнем шума $\sigma_i$ (одного из фиксированных).

Далее $s_\theta(x, \sigma_i)$ — это оценка, полученная при помощи нейронной сети. Осталось только подставить это в формулу и получить результат.

### 3.4 Полученные результаты

Модель обучалась на двух датасетах: MNIST и CIFAR. При обучении использовалось 10 уровней шума $\sigma_i$, расположенных в геометрической прогрессии. $\sigma_1 = 1$, $\sigma_{10} = 0.01$. При этом уровень шума 0.01 незаметен человеческому глазу. Для картинок $3\sigma_{10}$ шума эквивалентно изменению интенсивности индивидуального пикселя на 7.68 (если принимать, что интенсивность находится в диапазоне $[0; 256]$.

Модели обучались на одном GPU Tesla V100. MNIST обучался на протяжении 450 эпох, CIFAR учился 230 эпох.

В изначальной статье использовалось два GPU Tesla V100 при обучении на CIFAR. Так как мы использовали только один GPU, то из-за избежания ошибок по переполнению памяти размер батча на CIFAR был снижен до 100 картинок (вместо 128). Размер батча на MNIST не менялся и был равен 128.

Так же в изначальной статье модель на CIFAR училась дольше, там её учили на протяжении около 500 эпох. Так как модель на CIFAR учится гораздо медленнее, а времени к дедлайну дообучить модель у нас не хватило, то резульат был зафиксирован на 230 эпохах вместо 500.

На Рис. 3 можно увидеть лосс при обучении модели на датасетах MNIST и CIFAR-10.

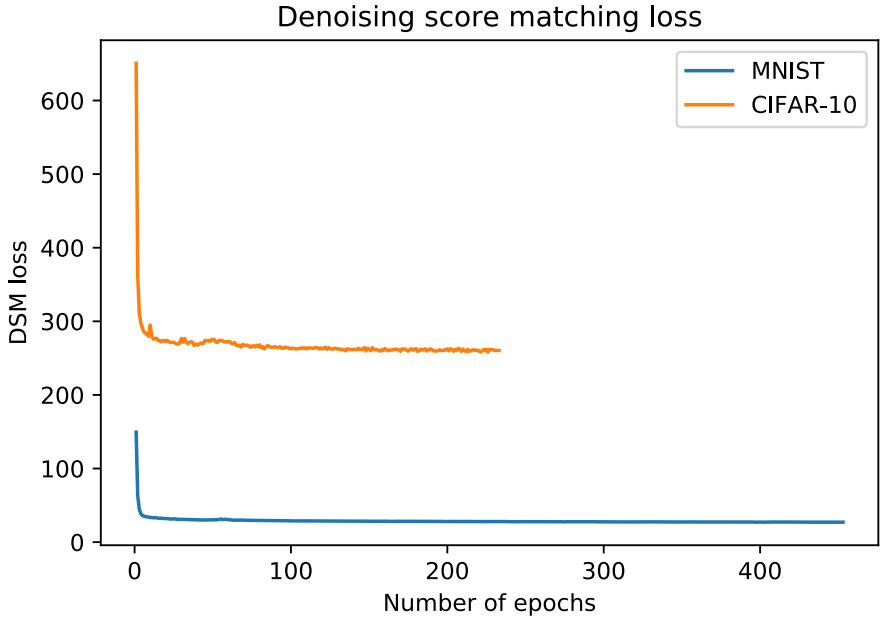
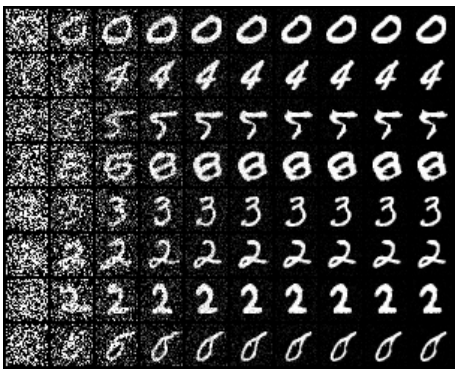



Рис. 3: DSM лосс на MNIST и CIFAR

(a) Сгенерированные картинки



(b) Процесс генерации картинки алгоритмом Ланжевена с отжигом

Рис. 4: Пример генерации картинок на MNIST

### 3.4.1 Результаты генерации на MNIST

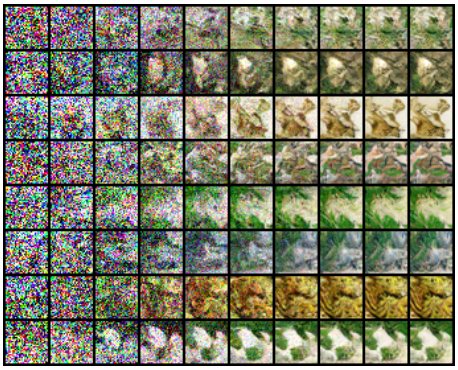
Для генерации картинок используется динамика Ланжевена с отжигом. Генерация начинается со случайной картинки, генерируемой из равномерного шума на отрезке $[0; 1]$.

Learning rate устанавливался равным $5 \cdot 10^{-5}$, количество шагов динамики для одного значения шума $\sigma_i$ было равно 100.

На Рис. 4a можно увидеть пример того, как алгоритм Ланжевена с отжигом генерирует картинки по модели, обученной на MNIST. Данный набор картинок является случайным и не был выбран специально, чтобы показать более качественный результат.

На Рис. 4b можно увидеть процесс генерации картинки. Каждый столбец соответствует финальному результату алгоритма Ланжевена при фиксированном уровне шума $\sigma_i$.

(a) Сгенерированные картинки



(b) Процесс генерации картинки алгоритмом Ланжевена с отжигом

Рис. 5: Пример генерации картинок на CIFAR

| Результат | Inception score | FID score |
|---|---|---|
| Исходная статья | $8.87 \pm .12$ | 25.32 |
| Наша модель | $4.17 \pm .13$ | 110.85 |

Рис. 6: Численная оценка результата генерации

Как можно заметить, картинки генерируются достаточно качественные, но не без ошибок. Иногда можно увидеть кляксы вместо цифр.

### 3.4.2 CIFAR-10

Процесс генерации картинок на CIFAR аналогичен тому же процессу для MNIST. Параметры алгоритма используются одни и те же.

На Рис. 5a и 5b можно увидеть аналогичные результаты на датасете CIFAR.

Как можно заметить, в отличие от MNIST результаты на CIFAR неудовлетворительные. Изображения получились примерно в одной цветовой гамме. Вместо чётко различимых объектов на картинках можно различить только какие-то общие черты.

Мы предполагаем, что это может быть связано с тем, что модель не дообучилась и дальнейшее обучение должно улучшить результат.

Для количественной оценки полученных картинок были подсчитаны Inception score и FID score. Оценки были подсчитаны на выборке из 5000 случайно сгенерированных картинок. На Рис. 6 можно увидеть данные оценки и их сравнение с результатом, представленным в оригинальной статье. Как можно заметить, полученный у нас результат гораздо хуже, чем тот, что был представлен в статье. Это ожидаемо с учётом того, что как было сказано выше, картинки генерируются не лучшего качества.

## 4 Выводы

1. Нами была реализована модель RefineNet и метод генерации картинок Ланжевеном с отжигом.
2. В данной работе мы показали на смеси гауссиан преимущество метода Ланжевена с отжигом против обычного Ланжевена
3. В ходе работы были успешно воспроизведены результаты по генерации картинок на датасете MNIST

4. Также менее успешными получились результаты на датасете CIFAR-10. Полученные результаты свидетельствуют о том, что данной модели необходимо большее время обучения, для генерации более удачных изображений.
5. Реализация воспроизведения статьи была размещена в репозитории `https://github.com/PeterZhizhin/HSE-DeepLearning-NCSN`.

## Список литературы


[1] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. CoRR, abs/1907.05600, 2019.

[2] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. CoRR, abs/1905.07088, 2019.

[3] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. CoRR, abs/1611.06612, 2016.