

Разбор статьи “Elucidating the Design Space of Diffusion-Based Generative Models”

Алишер Асланов [†]

Содержание

1 Введение	1
2 Общий фреймворк	4
3 Детерминированная генерация	7
4 Стохастическая генерация	10
5 Предобуславливание и обучение	11
6 Заключение	13
Приложения	15

1 Введение

В данном тексте мы разберем основные идеи и результаты из статьи “Elucidating the Design Space of Diffusion-Based Generative Models” [1]. Статья была принята на NeurIPS 2022, где получила награду “Outstanding Paper Award”. Авторы — исследователи из NVIDIA.

В этом разделе мы даем краткое введение в диффузионные модели и их интерпретации; в последующих — разбираем соответствующие разделы из статьи. В конце мы выделяем открытые вопросы, поднятые авторами статьи, и предлагаем возможные направления дальнейших исследований диффузионных моделей.

Рассмотрим задачу генерации объектов из неизвестного распределения p_{data} . Давайте построим процесс, “переводящий” это распределение в какое-то известное, например, стандартное нормальное. Если мы обратим этот процесс, то исходная задача сведется к генерации из $\mathcal{N}(0, I)$, поскольку тогда будет достаточно прогнать сгенерированный сэмпл через “обратный процесс”, и мы получим объект из p_{data} . Это основная идея диффузионных моделей. Существует несколько различных взглядов на эту идею, приведем некоторые из них.

Начнем с того, как именно модифицировать распределение данных. Рассмотрим марковскую цепь вида

$$x_{t+1} = \sqrt{1 - \beta_t} x_t + \sqrt{\beta_t} \varepsilon_t, \quad (1)$$

[†]Выпускник ФКН ПМИ, студент 1 курса магистратуры МИЭМ.

где $\varepsilon_t \sim \mathcal{N}(0, I)$ и $\beta_t \in (0, 1)$. Это т.н. *процесс с сохраняющейся дисперсией* (*VP, variance preserving*). В самом деле, если $\mathbb{D}x_0 = I$, то и $\forall t \mathbb{D}x_t = I$. Также справедливо

$$q(x_t | x_0) = \mathcal{N}(x_t | \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I), \quad (2)$$

где $\bar{\alpha}_t = \prod_{s=1}^t (1 - \beta_s)$. Имеет место сходимость распределений $q(x_t | x_0) \rightarrow \mathcal{N}(x_t | 0, I)$ для любого x_0 . Таким образом, при достаточно больших T можно считать, что $p_T(x_T) \approx \mathcal{N}(x_T | 0, I)$, как мы и хотели.

Также можно рассмотреть *процесс со взрывающейся дисперсией* (*VE, variance exploding*):

$$x_{t+1} = x_t + g(t)\varepsilon_t, \quad (3)$$

где $\varepsilon_t \sim \mathcal{N}(0, I)$. Тогда $q(x_t | x_0) = \mathcal{N}(x_t | x_0, \sigma_t^2 I)$, где $\sigma_t^2 = \sum_{s=1}^t g^2(s)$. Если дисперсия σ_T^2 достаточно велика, то $p_T(x_T) \approx \mathcal{N}(x_T | 0, \sigma_T^2 I)$.

По сути оба этих процесса соответствуют постепенному вспрыскиванию нормального шума в исходный объект, и начиная с какого-то момента, он становится неотличим от чистого шума. Но как мы будем обращать этот процесс, т.е. удалять шум?

Байесовский вывод. В работе [2] эта задача была рассмотрена с точки зрения байесовской статистики. А именно, предлагается формализовать описанную выше идею через модель $p_\theta(x_0, x_1, \dots, x_T)$ со скрытыми переменными x_1, x_2, \dots, x_T . Параметры θ настраиваются методом максимального правдоподобия. Логарифм неполного правдоподобия $\log p_\theta(X)$ оценивается снизу той же техникой, что и при выводе ЕМ-алгоритма (но в данном случае оценка будет не вариационная, т.к. распределение $q(x_1, x_2, \dots, x_T)$ на латентные переменные жестко задано процессом (1)).

И оказывается, что вместо аппроксимации “честных” обратных распределений $q(x_{t-1} | x_t)$ (невычислимых на практике) достаточно приближать условные распределения $q(x_{t-1} | x_t, x_0)$ (можно показать, что они все гауссовские). Модель p_θ зададим как марковскую цепь:

$$p_\theta(x_0, x_1, \dots, x_T) = p(x_T)p_\theta(x_{T-1} | x_T) \dots p_\theta(x_0 | x_1), \quad (4)$$

где $p(x_T) = \mathcal{N}(x_T | 0, I)$, $p_\theta(x_{t-1} | x_t) = \mathcal{N}(x_{t-1} | \mu_\theta(x_t, t), \tilde{\beta}_t I)$ и $\mu_\theta(x_t, t)$ — обучаемая нейросеть. В таком виде она предсказывает среднее распределения $q(x_{t-1} | x_t, x_0)$, но удобнее сделать репараметризацию и предсказывать исходный сигнал $x_0 \approx x_\theta(x_t, t)$ или добавленный в него шум $\varepsilon \approx \varepsilon_\theta(x_t, t)$, где $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$ и $\varepsilon \sim \mathcal{N}(0, I)$. Все эти задачи эквивалентны, но в оригинальной работе используют последний вариант и оптимизируют функционал

$$\mathcal{L}(\theta) = \mathbb{E}_{t, x_0, \varepsilon} \left[\|\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon, t)\|^2 \right]. \quad (5)$$

Генерация осуществляется последовательно из распределений $p(x_T)$ и $p_\theta(x_{t-1} | x_t)$, $t = T, \dots, 1$. На практике обычно $T \sim 1000$, и в случае с изображениями это работает довольно медленно, по сравнению с другими семействами генеративных моделей. Далее мы увидим, как можно ускорить процесс генерации.

Стochastic differential equations. Авторы работы [3] обобщили предыдущие подходы к диффузионным моделям, сделав процессы (1) и (3) непрерывными через *стochastic differential equations* (*SDE*). Это уравнения вида

$$dx_t = f(x_t, t) dt + g(t) dw_t, \quad (6)$$

где $f(\cdot, t): \mathbb{R}^d \rightarrow \mathbb{R}^d$ и $g: \mathbb{R} \rightarrow \mathbb{R}$ — заданные функции и dw_t — дифференциал винеровского процесса. По сути это бесконечно малое приращение $w_{t+h} - w_t \sim \mathcal{N}(0, hI)$, означающее непрерывное впрыскивание нормального шума с магнитудой $g(t)$. Также обычно задается распределение в начальный момент времени: $x_0 \sim p_0$.

Прежде чем возвращаться к диффузионным моделям, рассмотрим некоторые классические результаты из теории SDE. Оказывается, несмотря на то, что траектории решения уравнения (6) случайны, плотность x_t меняется детерминированно. Эволюция плотности p_t описывается *уравнением Фоккера-Планка(-Колмогорова)*:

$$\frac{\partial}{\partial t} p_t(x_t) = -\operatorname{div}\left(f(x_t, t)p_t(x_t)\right) + \frac{g^2(t)}{2}\Delta_x p_t(x_t), \quad (7)$$

где div обозначает дивергенцию и Δ_x — оператор Лапласа. Путем несложных манипуляций с этим уравнением можно получить т.н. *probability flow ODE* — обыкновенное дифференциальное уравнение

$$dx_t = \left(f(x_t, t) - \frac{g^2(t)}{2}\nabla_x \log p_t(x_t)\right) dt, \quad (8)$$

которое вместе с начальным условием $x_0 \sim p_0$ описывает ту же эволюцию плотности исходного SDE. ODE легко обращать по времени — достаточно поставить минус перед функцией при dt . Таким образом, можно считать, что у нас уже есть обратный процесс. Но можно пойти еще дальше и получить обратное SDE

$$dx_t = \left(f(x_t, t) - g^2(t)\nabla_x \log p_t(x_t)\right) dt + g(t) d\bar{w}_t, \quad (9)$$

где dt и $d\bar{w}_t$ — отрицательные бесконечно малые приращения. При некоторых условиях регулярности, SDE (9) вместе с начальным условием $x_T \sim p_T$ описывает на отрезке $[0, T]$ обратную эволюцию плотности $p_T \rightarrow p_0$ [4].

Выпишем SDE, соответствующие описанным ранее диффузионным процессам. Для VP-процесса получим уравнение $dx_t = -\frac{\beta_t}{2}x_t dt + \sqrt{\beta_t} dw_t$, для VE-процесса — $dx_t = g(t) dw_t$. Изначально введенные процессы (1) и (3) являются дискретизациями соответствующих SDE.

Теперь задача генерации свелась к вычислению $\nabla_x \log p_t(x)$ и решению либо обратного ODE (8), либо SDE (9). Простейшая схема численного решения SDE на отрезке $[0, T]$ — *метод Эйлера-Маруямы*. Выберем сетку $0 = t_0 < t_1 < \dots < t_N = T$ и заменим все дифференциалы в (6) на соответствующие разности. Получим итеративный процесс

$$x_{t_{i+1}} = x_{t_i} + f(x_{t_i}, t_i)h_i + g(t_i)\varepsilon_i, \quad (10)$$

где $h_i = t_{i+1} - t_i$ и $\varepsilon_i \sim \mathcal{N}(0, h_i I)$. Более плотная сетка соответствует более точной аппроксимации, но требует пропорционально больше вычислений. Также отметим, что численно решать ODE проще, чем SDE (в т.ч. можно использовать продвинутые аддитивные методы).

Score matching. Осталось понять как оценить функцию $\nabla_x \log p_t(x)$, называемую *score функцией*. Аналитически она не считается, т.к. $p_t(x_t) = \int q(x_t | x_0)p_0(x_0) dx_0$, где p_0 — неизвестное нам распределение данных. Имеет место равенство

$$\nabla_x \log p_t(x) = \mathbb{E} \left[\nabla_x \log q(x_t | x_0) \mid x_t = x \right], \quad (11)$$

где $x_0 \sim p_{0|t}(\cdot | x)$. И поскольку $\mathbb{E}[Y | X = x] = \arg \min_g \mathbb{E}\|g(X) - Y\|^2$, то задача сводится к вычислению наилучшего среднеквадратичного приближения функции $\nabla_x \log q(x_t | x_0)$, которую

уже легко посчитать аналитически. Например, в случае VE-процесса $x_t = x_0 + \sigma_t \varepsilon$, $\varepsilon \sim \mathcal{N}(0, I)$, $q(x_t | x_0) = \mathcal{N}(x_t | x_0, \sigma_t^2 I)$ и

$$\nabla_x \log q(x_t | x_0) = \frac{x_0 - x_t}{\sigma_t^2} = -\frac{\varepsilon}{\sigma_t}. \quad (12)$$

Получили, что задача снова свелась к denoising, т.е. предсказанию исходного сигнала x_0 или добавленного шума ε по x_t и t . То же самое справедливо и для VP-процесса. В частности, отсюда видно, что оптимизация функционала (5) — это не что иное, как обучение score функции с точностью до отрицательной константы. Более того, можно показать, что процедура генерации из работы [2] — это дискретизация соответствующего обратного SDE методом Эйлера-Маруямы [3, прил. E].

Итак, взгляд на диффузионные модели через теорию SDE открывает новые возможности для построения различных модификаций и улучшений. Например, детерминированная генерация через probability flow ODE (8) и исследование его интегральных кривых с целью ускорения генерации без потери качества. Это и другие улучшения предлагаются в статье [1], непосредственно к разбору которой мы сейчас приступим.

2 Общий фреймворк

Прежде всего, какова мотивация данного исследования? Все предшествующие работы по диффузионным моделям имеют хорошую теоретическую базу, но в то же время предлагаемые модели могут показаться сильно ограниченными с точки зрения возможности модификации отдельных компонент. Вдруг все сломается, если изменить диффузионный процесс и/или процедуру обучения? Авторы рассмотрели работы [3, 5, 6] и определили полный набор отдельных компонент дизайна процедур обучения и генерации, которые можно независимо друг от друга модифицировать, не испортив при этом корректность модели. Для каждой из этих компонент они находят наилучший (в каком-то смысле) вариант. В результате удалось достичь state-of-the-art (на момент публикации) FID на CIFAR-10 и ImageNet-64, значительно ускорив при этом генерацию (35 и 511 вызовов нейросети на изображение соответственно).

В диффузионных моделях есть два возможных источника ошибок:

1. Нейросеть плохо оценила score функцию;
2. Плохая аппроксимация траектории решения ODE/SDE при генерации.

Утверждается, что эти два источника независимы и должны анализироваться отдельно. Это логично, поскольку на этапе обучения мы по сути просто оцениваем score функцию, а при генерации используем выученный denoiser как черный ящик. Этую гипотезу можно подтвердить эмпирически (см. секцию 3).

В работе прямой процесс обобщается через функции $\sigma(t)$ и $s(t)$, задающие расписание зашумления и шкалирования соответственно:

$$\hat{x} = x_0 + \sigma(t)\varepsilon, \quad x = s(t)\hat{x}. \quad (13)$$

Здесь $x_0 \sim p_{\text{data}}$ и $\varepsilon \sim \mathcal{N}(0, I)$. Распределение \hat{x} авторы обозначают $p(\hat{x}; \sigma)$,

$$p(\hat{x}; \sigma) = [p_{\text{data}} * \mathcal{N}(0, \sigma^2 I)](\hat{x}), \quad (14)$$

где $\sigma = \sigma(t)$ и $*$ обозначает свертку. Переходное распределение для такого процесса есть

$$p_{t|0}(x | x_0) = \mathcal{N}(x | s(t)x_0, s^2(t)\sigma^2(t)I). \quad (15)$$

Можно показать [1, прил. B.2], что шкалированный объект x имеет в момент времени t плотность

$$p_t(x) = \int_{\mathbb{R}^d} p_{t|0}(x | x_0) p_{\text{data}}(x_0) dx_0 = \frac{1}{s^d(t)} p\left(\frac{x}{s(t)}; \sigma(t)\right), \quad (16)$$

где d — размерность данных.

Хочется понять, как $\sigma(t)$ и $s(t)$ влияют на процесс генерации. Для этого авторы предлагают переписать probability flow ODE (8) в терминах этих функций. Как это сделать? Рассмотрим SDE общего вида, у которого $f(x_t, t) = f(t)x_t$, где $f: \mathbb{R} \rightarrow \mathbb{R}$,

$$dx_t = f(t)x_t dt + g(t) dw_t. \quad (17)$$

Для процесса x_t , являющегося решением этого SDE, можно в явном виде выписать переходные распределения $p_{t|0}(x_t | x_0)$ (см. детали в прил. A). В то же время мы хотим, чтобы это SDE описывало наш прямой процесс зашумления, то есть переходные распределения должны иметь вид (15). Из этого требования вытекает, что функции $\sigma(t)$ и $s(t)$ однозначно определяются функциями f и g :

$$s(t) = \exp\left(\int_0^t f(\xi) d\xi\right), \quad \sigma(t) = \sqrt{\int_0^t \frac{g^2(\xi)}{s^2(\xi)} d\xi}. \quad (18)$$

И наоборот [1, прил. B.2]:

$$f(t) = \frac{\dot{s}(t)}{s(t)}, \quad g(t) = s(t) \sqrt{2\dot{\sigma}(t)\sigma(t)}, \quad (19)$$

где точка обозначает производную по времени. Подставляя выражения (16) и (19) в (8), получим

$$dx_t = \left[\frac{\dot{s}(t)}{s(t)} x_t - s^2(t) \dot{\sigma}(t) \sigma(t) \nabla_x \log p\left(\frac{x_t}{s(t)}; \sigma(t)\right) \right] dt. \quad (20)$$

В частности, если $s(t) = 1$ (VE-процесс, без шкалирования), то имеем

$$dx_t = -\dot{\sigma}(t) \sigma(t) \nabla_x \log p(x_t; \sigma(t)) dt. \quad (21)$$

Чем полезна такая форма записи? Она позволяет напрямую исследовать зависимость траекторий решений probability flow ODE от параметров $\sigma(t)$ и $s(t)$ прямого процесса. При этом, за счет отсутствия стохастичности, все траектории детерминированы и обратимы при заданном начальном условии. Это позволяет прогонять объекты как вперед по времени, так и назад.

Существенной проблемой при генерации является кривизна траекторий. Метод Эйлера численного интегрирования ODE (процесс (10) без последнего слагаемого) по сути строит кусочно-линейную аппроксимацию траектории, что, очевидно, приводит к высокой ошибке, если она сильно искривлена. Иллюстрация этой проблемы в одномерном случае приведена на рис. 1. На эффективность решения этой проблемы влияют 3 компоненты дизайна процедуры генерации:

1. **Сетка времен** $\{t_i\}_{i=0}^N$. Чем больше число шагов N , тем точнее решение, но тем медленнее генерация. Хотелось бы делать короткие шаги только в областях с высокой кривизной;
2. **Расписания $\sigma(t)$ и $s(t)$** . Хочется подобрать их так, чтобы траектории решений probability flow ODE были как можно более прямыми. Это позволит делать длинные шаги, тем самым ускоряя генерацию;
3. **Метод численного интегрирования**. Имеет смысл использовать методы высших порядков, чтобы повысить точность и, следовательно, качество генерации.

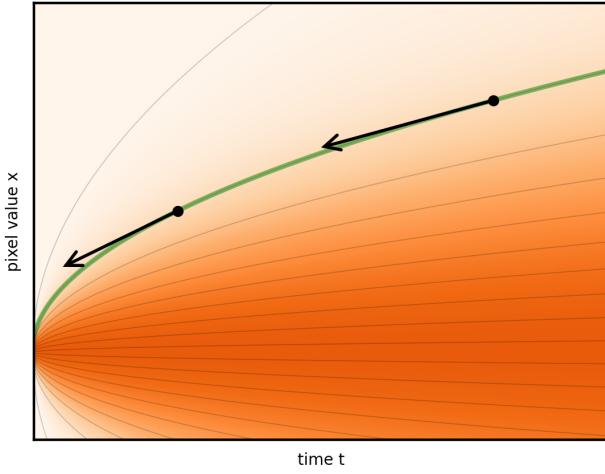


Рис. 1: Шаги вдоль касательной (метод Эйлера) дают плохую аппроксимацию при высокой кривизне траектории. Источник: [7].

Стохастическую генерацию и ее параметры авторы рассматривают отдельно (см. секцию 4).

Еще одна важная проблема возникает при обучении диффузационной модели. Как было упомянуто ранее, обучение сводится к denoising score matching. Если $D(x; \sigma)$ — идеальный denoiser, т.е. решение задачи оптимизации

$$\mathbb{E}_{y \sim p_{\text{data}}} \mathbb{E}_{n \sim \mathcal{N}(0, \sigma^2 I)} \|D(y + n; \sigma) - y\|_2^2 \rightarrow \min \quad (22)$$

для заданного σ , то score функция представляется в виде [1, прил. B.3]

$$\nabla_x \log p(x; \sigma) = \frac{D(x; \sigma) - x}{\sigma^2}. \quad (23)$$

На практике $D(x; \sigma)$ заменяется нейросетью $D_\theta(x; \sigma)$, которая обучается на взвешенный по всем σ функционал (22). Параметры θ общие для всех σ , а вход $x = y + n$ имеет большую дисперсию за счет различных уровней шума, что может привести к численной нестабильности обучения. Как можно решить эту проблему? Предлагается ввести **предобусловливатели** $c_{\text{skip}}(\sigma)$, $c_{\text{in}}(\sigma)$, $c_{\text{out}}(\sigma)$ и $c_{\text{noise}}(\sigma)$ и представить D_θ в виде

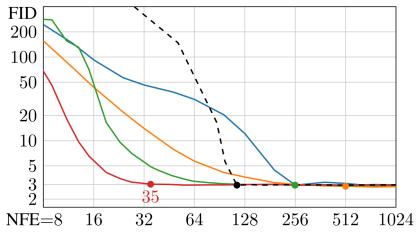
$$D_\theta(x; \sigma) = c_{\text{skip}}(\sigma)x + c_{\text{out}}(\sigma)F_\theta(c_{\text{in}}(\sigma)x; c_{\text{noise}}(\sigma)), \quad (24)$$

где F_θ — обучаемая модель. Эти функции подбираются так, чтобы дисперсия входа и выхода F_θ была единичной, а также чтобы минимизировать вклад ошибки предсказания. Это можно сделать аналитически вне зависимости от модели (см. секцию 5).

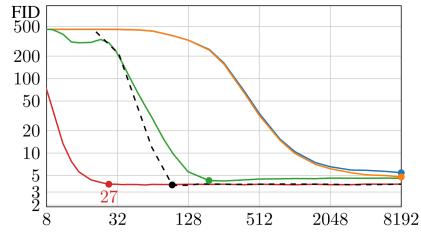
Еще две компоненты дизайна процедуры обучения — **распределение шумов и веса в функционале ошибки**. Их подбор тоже обсуждается в секции 5. Отметим также, что влияние конкретной архитектуры F_θ в работе не исследуется.

Итак, выше выделены все компоненты дизайна генерации и обучения для широкого класса диффузационных моделей¹. Авторы показывают [1, табл. 1, прил. C], что предшествующие методы [3, 5, 6] можно переформулировать в рамках их фреймворка (= явно выписать для них расписания, предобусловливатели и все остальное). То есть несмотря на то, что эти методы

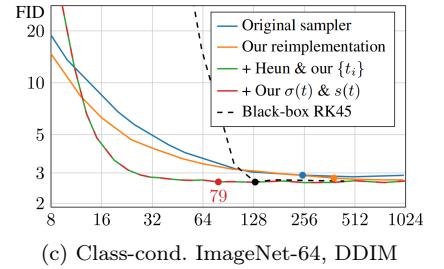
¹Существуют методы, не вписывающиеся в этот фреймворк. Например, диффузационные модели с обучаемой дисперсией [5] или с не гауссовским зашумлением [8].



(a) Uncond. CIFAR-10, VP



(b) Uncond. CIFAR-10, VE



(c) Class-cond. ImageNet-64, DDIM

Рис. 2: Сравнение методов детерминированной генерации для трех предобученных моделей. Для каждой кривой точка обозначает наименьшее значение NFE, для которого FID отличается от минимального не более чем на 3%.

были выведены через различные теоретические основания, и их части выглядят тесно связанными между собой, их можно разбить на независимые компоненты², полностью определяющие каждый метод. Причем оказывается, что выбор многих из этих компонент сделан неявно и субоптимально. Давайте теперь разберем, как именно авторы предлагают делать оптимальный выбор.

3 Детерминированная генерация

Как было отмечено выше, процедура обучения D_θ не должна влиять на выбор $\sigma(t)$, $s(t)$ и $\{t_i\}$, и наоборот. Авторы подтверждают это эмпирически, модифицировав процедуру генерации для предобученных моделей из предыдущих статей и добившись лучшего качества и скорости. Результаты эксперимента приведены на рис. 2. Рассматривались модели для VP- и VE-процесса из работы [3] (CIFAR-10, безусловная генерация) и DDIM [6] (ImageNet-64, условная генерация)³. В результате получилось уменьшить число вызовов D_θ на изображение (NFE, neural function evaluations), необходимое для высокого качества, в 7.3, 300 и 3.2 раза для VP, VE и DDIM соответственно.

Численное интегрирование ODE. Метод Эйлера является методом первого порядка с локальной ошибкой $\mathcal{O}(h^2)$, где h — длина шага⁴. В качестве альтернативы ему авторы рассматривают семейство методов Рунге-Кутты второго порядка. Они дают более высокую точность (локальная ошибка $\mathcal{O}(h^3)$) взамен на еще одно вычисление D_θ за шаг. Это семейство параметризовано числом α , и чтобы найти его оптимальное значение, авторы провели серию отдельных экспериментов. Они показали, что значение $\alpha = 1$, соответствующее *методу Гойна* (рис. 3), дает наилучший компромисс между точностью и скоростью (NFE) [1, прил. D.2].

Суть метода в следующем. Сделаем из текущей точки один шаг по Эйлеру. Из полученной точки делаем еще один такой шаг. Результирующим шагом из исходной точки будет среднее двух предыдущих (рис. 4). Так мы будем куда более точно следовать истинной траектории решения.

Сетка времен $\{t_i\}$. Чтобы подобрать оптимальную дискретизацию, авторы исследуют поведение локальных ошибок при интегрировании probability flow ODE на различных уровнях шума. Результаты показывают, что при постоянной длине шага локальная ошибка на малых σ существенно выше, чем на больших σ . Это означает, что наиболее существенное искривление

²Отметим, что независимость здесь довольно условная: при изменении одной компоненты не обязательно менять другие — в теории модель всегда сходится к данным в *пределе*. Но понятно, что какие-то комбинации компонент на практике работают лучше, чем другие.

³Везде использовались вариации архитектуры U-Net. Детали приведены в [1, прил. F.4, табл. 8].

⁴Локальная ошибка есть $|x_i^* - x_i|$, где x_i — численная аппроксимация на шаге i истинного значения x_i^* . h в данном случае есть $|t_i - t_{i-1}|$.

Algorithm 1 Deterministic sampling using Heun's 2nd order method with arbitrary $\sigma(t)$ and $s(t)$.

```

1: procedure HEUNSMPLER( $D_\theta(\mathbf{x}; \sigma)$ ,  $\sigma(t)$ ,  $s(t)$ ,  $t_{i \in \{0, \dots, N\}}$ )
2:   sample  $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \sigma^2(t_0) s^2(t_0) \mathbf{I})$                                  $\triangleright$  Generate initial sample at  $t_0$ 
3:   for  $i \in \{0, \dots, N - 1\}$  do                                               $\triangleright$  Solve Eq. 4 over  $N$  time steps
4:      $\mathbf{d}_i \leftarrow \left( \frac{\dot{\sigma}(t_i)}{\sigma(t_i)} + \frac{\dot{s}(t_i)}{s(t_i)} \right) \mathbf{x}_i - \frac{\dot{\sigma}(t_i)s(t_i)}{\sigma(t_i)} D_\theta \left( \frac{\mathbf{x}_i}{s(t_i)}; \sigma(t_i) \right)$        $\triangleright$  Evaluate  $d\mathbf{x}/dt$  at  $t_i$ 
5:      $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + (t_{i+1} - t_i) \mathbf{d}_i$                                           $\triangleright$  Take Euler step from  $t_i$  to  $t_{i+1}$ 
6:     if  $\sigma(t_{i+1}) \neq 0$  then                                      $\triangleright$  Apply 2nd order correction unless  $\sigma$  goes to zero
7:        $\mathbf{d}'_i \leftarrow \left( \frac{\dot{\sigma}(t_{i+1})}{\sigma(t_{i+1})} + \frac{\dot{s}(t_{i+1})}{s(t_{i+1})} \right) \mathbf{x}_{i+1} - \frac{\dot{\sigma}(t_{i+1})s(t_{i+1})}{\sigma(t_{i+1})} D_\theta \left( \frac{\mathbf{x}_{i+1}}{s(t_{i+1})}; \sigma(t_{i+1}) \right)$   $\triangleright$  Eval.  $d\mathbf{x}/dt$  at  $t_{i+1}$ 
8:        $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + (t_{i+1} - t_i) \left( \frac{1}{2} \mathbf{d}_i + \frac{1}{2} \mathbf{d}'_i \right)$            $\triangleright$  Explicit trapezoidal rule at  $t_{i+1}$ 
9:   return  $\mathbf{x}_N$                                           $\triangleright$  Return noise-free sample at  $t_N$ 

```

Рис. 3: Метод Гойна для численного решения ОДЕ (20).

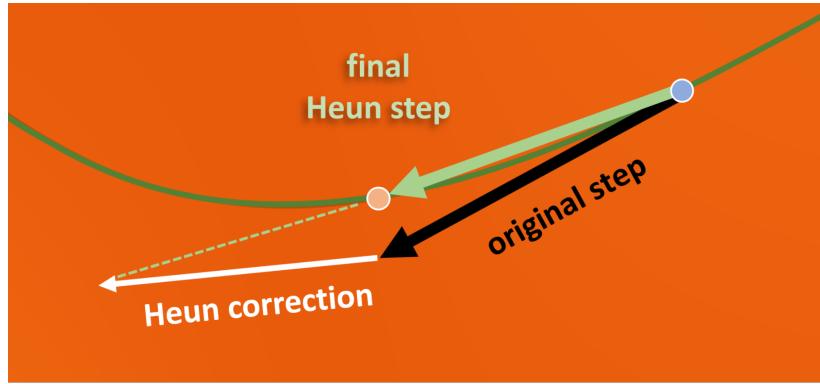


Рис. 4: Геометрическая иллюстрация одного шага метода Гойна. Источник: [7].

траекторий происходит около многообразия данных. Соответственно, имеет смысл монотонно уменьшать длину шага с уменьшением σ [1, прил. D.1].

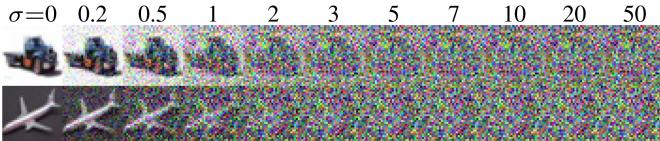
Авторы задают непосредственно уровни шума $\{\sigma_i\}$; соответствующие времена выражаются как $t_i = \sigma^{-1}(\sigma_i)$. Предлагается положить $\sigma_{i < N} = (Ai + B)^\rho$ и подобрать числа A и B так, чтобы $\sigma_0 = \sigma_{\max}$ и $\sigma_{N-1} = \sigma_{\min}$ ⁵, т.е.

$$\sigma_{i < N} = \left(\sigma_{\max}^{\frac{1}{\rho}} + \frac{i}{N-1} \left(\sigma_{\min}^{\frac{1}{\rho}} - \sigma_{\max}^{\frac{1}{\rho}} \right) \right)^\rho \quad \text{и} \quad \sigma_N = 0. \quad (25)$$

Параметр ρ отвечает за плотность сетки около σ_{\min} , т.е. чем больше ρ , тем короче шаги на маленьких уровнях шума и тем они длиннее на больших. $\rho = 1$ соответствует равномерной дискретизации.

Чтобы подобрать оптимальное ρ , авторы исследуют, как этот параметр влияет на FID для различных моделей и значений N . Во всех случаях лучший результат показало $\rho = 7$, и далее авторы используют его. Также они смотрят на распределение локальных ошибок для различных ρ . Примечательно, что стандартные отклонения этих ошибок для всех уровней шума σ и всех ρ примерно равны нулю [1, рис. 13]. Отсюда вывод, что адаптивный подбор длины шага в зависимости от конкретного изображения на практике не дает никаких преимуществ.

⁵На практике σ_{\max} и σ_{\min} — максимальный и минимальный больший нуля уровни шума — задаются в зависимости от конкретной модели. Также отметим, что в нотации авторов чем больше номер шага i , тем меньше шум σ_i (т.е. время уже обращено).

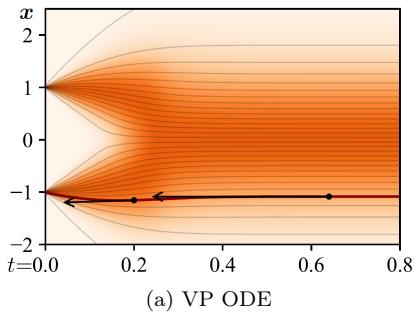


(a) Сэмплы из $p(x; \sigma)$

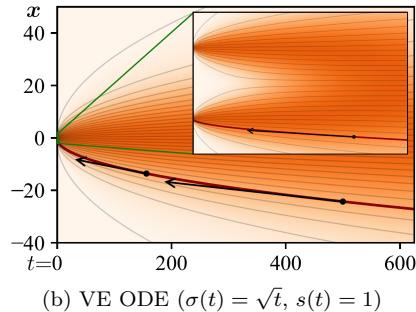


(b) Выходы идеального denoiser'a $D(x; \sigma)$

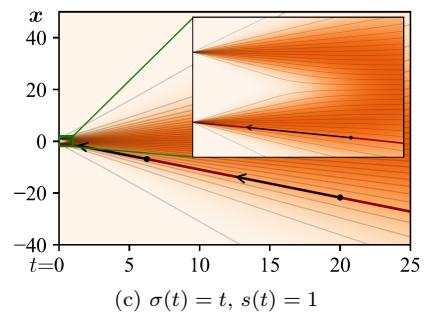
Рис. 5: Denoising score matching на CIFAR-10.



(a) VP ODE



(b) VE ODE ($\sigma(t) = \sqrt{t}$, $s(t) = 1$)



(c) $\sigma(t) = t$, $s(t) = 1$

Рис. 6: Траектории probability flow ODE в одномерном случае для различных расписаний. $p_{\text{data}}(x) = \frac{1}{2}\delta(x - 1) + \frac{1}{2}\delta(x + 1)$. На всех графиках $\sigma \in [0, 25]$, в приближении — $\sigma \in [0, 1]$. Наибольшее искривление заметно ближе к $t = 0$. Предложенное авторами расписание дает наиболее прямые траектории.

Расписания $\sigma(t)$ и $s(t)$. Авторы утверждают, что лучший выбор — $\sigma(t) = t$ и $s(t) = 1$. Почему так? Подставляя эти расписания и выражение для score функции (23) в (20), получим ODE

$$dx_t = \frac{x_t - D(x_t; t)}{t} dt. \quad (26)$$

Заметим, что для любых t и x_t один шаг метода Эйлера в $t = 0$ даст

$$x_0 = x_t - \left(\frac{x_t - D_\theta(x_t; t)}{t} \right) t = D_\theta(x_t; t). \quad (27)$$

Метод Эйлера соответствует движению вдоль касательной \Rightarrow касательная к траектории любого решения ODE (26) в любой точке имеет в нуле значение $D_\theta(x_t; t)$.

Естественно ожидать, что $D_\theta(x_t; t)$ меняется медленно по t — процесс расщумления представляется плавным. И это действительно так, в некоторой степени (см. рис. 5). Если бы выход denoiser'a не менялся по t вообще, отсюда немедленно бы следовало, что все траектории (26) линейны⁶. Соответственно, при плавном изменении значения касательной в нуле траектории не сильно будут отличаться от прямых⁷ — это ровно то, что мы изначально и хотели. Этую интуицию подтверждает модельный пример на рис. 6c — почти везде траектории прямые, за исключением лишь узкой области возле данных. То же самое видно и на реальных данных на рис. 5b — относительно резкое изменение выхода denoiser'a происходит лишь в узком интервале шумов.

⁶В самом деле, пусть $h \in C^1([0, T])$ — скалярная функция и $\forall t h_t(0) = a$, где h_t — касательная в точке t . Тогда $h(t) - h'(t)t = a$. Решая дифференциальное уравнение, получаем $h(t) = Ct + a$, $C \in \mathbb{R}$.

⁷Контрапозиция: если бы траекторию “колбасило”, то значение касательной в нуле “колбасило” бы еще сильнее.

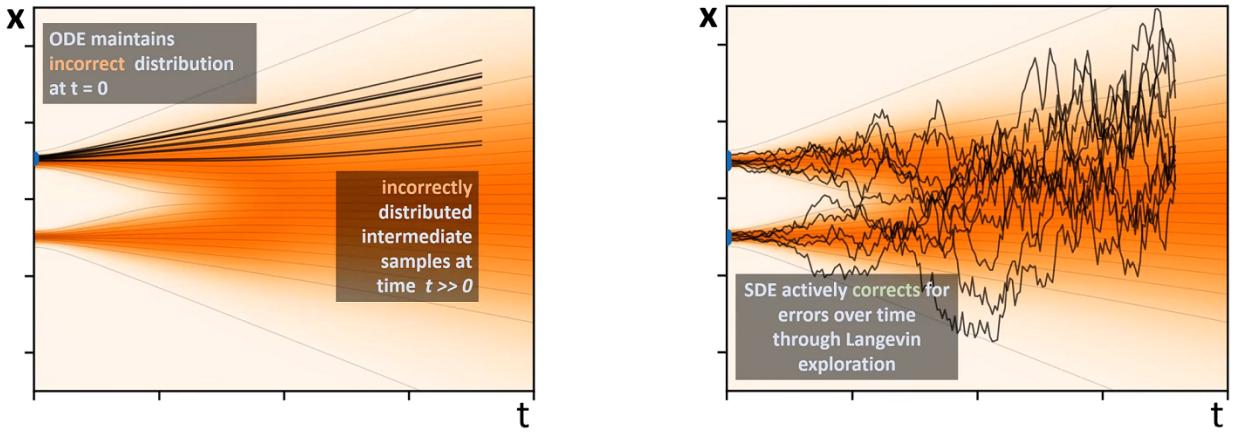


Рис. 7: Пусть имеются некорректно распределенные сэмплы в момент времени t . Если следовать детерминированной траектории, то распределение может так и остаться плохим (слева, покрыта только одна мода). Если же добавить стохастичности, то сэмплы постепенно “забывают”, откуда они пришли, и с более высокой вероятностью станут корректно распределенными к моменту времени 0 (справа, покрыты обе моды). Источник: <https://youtu.be/T0Qxzf0eaio&t=1956>.

4 Стохастическая генерация

Теперь обсудим предложенные улучшения для стохастической генерации. Для начала запишем SDE, дающее нужные нам маргинальные распределения (16):

$$dx_t^\pm = \underbrace{-\dot{\sigma}(t)\sigma(t)\nabla_x \log p(x_t; \sigma(t)) dt}_{\text{probability flow ODE (21)}} \pm \underbrace{\beta(t)\sigma^2(t)\nabla_x \log p(x_t; \sigma(t)) dt + \sqrt{2\beta(t)}\sigma(t) dw_t}_{\text{SDE Ланжевена}}. \quad (28)$$

Для упрощения записи считаем, что $s(t) = 1$. Здесь dx_t^+ и dx_t^- — прямое и обратное SDE соответственно. Это обобщение probability flow ODE с добавкой в виде *уравнения Ланжевена*⁸, которое по сути является случайным блужданием, сохраняющим распределение. Т.е. теперь движение по детерминированным траекториям, которое было у нас ранее, разбавляется случайным шумом, а $\beta(t)$ отвечает за интенсивность впрыскивания этого шума⁹. В частности, $\beta(t) = 0$ дает ODE, а $\beta(t) = \frac{\dot{\sigma}(t)}{\sigma(t)}$ — VE SDE.

Заметим, что probability flow ODE (21) и SDE (28) в теории описывают одну и ту же динамику плотности p_t . В чем же тогда состоит роль стохастичности на практике? Оказывается, что случайное блуждание в уравнении Ланжевена помогает корректировать ошибки в распределении (рис. 7). За счет этого стохастическая генерация может в каких-то случаях работать лучше, чем детерминированная.

Но возникают две проблемы: аппроксимация траекторий уравнения Ланжевена сама по себе тоже вносит ошибку; и непонятно, как выбирать $\beta(t)$. Поэтому вместо методов численного интегрирования SDE общего назначения авторы предлагают скомбинировать метод Гойна для probability flow ODE с явным добавлением и удалением шума, а количество шума определять

⁸Общий вид уравнения Ланжевена: $dx_t = \frac{g^2(t)}{2}\nabla_x \log p_t(x_t) dt + g(t) dw_t$. Его основное свойство: если $x_0 \sim p_0$, то $\forall t x_t \sim p_0$.

⁹Здесь можно заподозрить противоречие: откуда мог взяться свободный параметр $\beta(t)$, если мы ранее установили, что коэффициенты SDE f и g однозначно выражаются через $\sigma(t)$ уравнениями (19)? Получается, что для заданного $\sigma(t)$ есть лишь одно SDE, дающее такой процесс зашумления? Вообще говоря, нет. Уравнения (19) верны только в предположении, что $f(x_t, t) = f(t)x_t$. А SDE (28) выводится из более общих соображений [1, прил. B.5]. Оказывается, что нам также подходит целое семейство коэффициентов f вида $f(x_t, t) = \left(\frac{g^2(t)}{2} - \dot{\sigma}(t)\sigma(t)\right)\nabla_x \log p(x_t; \sigma(t))$, где $g(t)$ — любая. Параметризуя $g(t) = \sqrt{2\beta(t)}\sigma(t)$, получаем (28).

Algorithm 2 Our stochastic sampler with $\sigma(t) = t$ and $s(t) = 1$.

```

1: procedure STOCHASTICSAMPLER( $D_\theta(\mathbf{x}; \sigma)$ ,  $t_{i \in \{0, \dots, N\}}$ ,  $\gamma_{i \in \{0, \dots, N-1\}}$ ,  $S_{\text{noise}}$ )
2:   sample  $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, t_0^2 \mathbf{I})$ 
3:   for  $i \in \{0, \dots, N-1\}$  do
4:     sample  $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, S_{\text{noise}}^2 \mathbf{I})$ 
5:      $\hat{t}_i \leftarrow t_i + \gamma_i t_i$ 
6:      $\hat{\mathbf{x}}_i \leftarrow \mathbf{x}_i + \sqrt{\hat{t}_i^2 - t_i^2} \boldsymbol{\epsilon}_i$ 
7:      $\mathbf{d}_i \leftarrow (\hat{\mathbf{x}}_i - D_\theta(\hat{\mathbf{x}}_i; \hat{t}_i)) / \hat{t}_i$ 
8:      $\mathbf{x}_{i+1} \leftarrow \hat{\mathbf{x}}_i + (t_{i+1} - \hat{t}_i) \mathbf{d}_i$ 
9:     if  $t_{i+1} \neq 0$  then
10:       $\mathbf{d}'_i \leftarrow (\mathbf{x}_{i+1} - D_\theta(\mathbf{x}_{i+1}; t_{i+1})) / t_{i+1}$ 
11:       $\mathbf{x}_{i+1} \leftarrow \hat{\mathbf{x}}_i + (t_{i+1} - \hat{t}_i) (\frac{1}{2} \mathbf{d}_i + \frac{1}{2} \mathbf{d}'_i)$ 
12:   return  $\mathbf{x}_N$ 

```

Рис. 8: Процедура стохастической генерации, предложенная авторами.

эмпирически. Алгоритм для $\sigma(t) = t$ приведен на рис. 8.

На очередной итерации сэмпл x_i сперва дополнительно зашумляется до уровня $\hat{t}_i = t_i + \gamma_i t_i$ ($\gamma_i \geq 0$), и затем для полученного сэмпла \hat{x}_i делается шаг метода Гойна из \hat{t}_i в t_{i+1} . Процедура корректна, потому что в каждый момент времени сохраняется нужное распределение. Основное отличие от метода Эйлера-Маруямы (процесс (10)) в том, что последний делает шаг из $x_i; t_i$ и потом уже добавляет шум, в то время как метод авторов делает шаг уже после зашумления — из $\hat{x}_i; \hat{t}_i$.

Функцию $\beta(t)$ по сути заменяют 4 параметра — $\{S_{\text{churn}}, S_{\text{tmin}}, S_{\text{tmax}}, S_{\text{noise}}\}$. S_{churn} определяет общее количество шума ($\gamma_i = S_{\text{churn}}/N$); если $t_i \notin [S_{\text{tmin}}, S_{\text{tmax}}]$, то шум не добавляется ($\gamma_i = 0$); S_{noise} задает стандартное отклонение добавляемого шума. Все параметры подбираются по сетке для конкретной модели и данных, что является существенным недостатком [1, прил. E.2].

Применяя предложенный метод к предобученным моделям из предыдущей секции, авторы получили лучшие результаты, по сравнению с детерминированной генерацией, однако с большим NFE [1, рис. 4]. В частности, в задаче условной генерации на ImageNet-64 удалось выбрать FID 1.55 (при NFE = 1024), что было очень близко к SOTA на момент публикации работы. Но, как мы увидим далее, стохастическая генерация не всегда работает лучше детерминированной.

5 Предобусловливание и обучение

До этого момента мы предполагали, что denoiser D_θ фиксирован и обучен за нас, и обсуждали улучшения только для процедуры генерации. Теперь разберем предлагаемые авторами практики для обучения.

D_θ принимает на вход зашумленный объект $x = y + n$, где $y \sim p_{\text{data}}$ и $n \sim \mathcal{N}(0, \sigma^2 I)$, и предсказывает y . При этом шум σ может быть как около-нулевым, так и очень большим \Rightarrow дисперсия x велика, что может плохо сказаться на обучении. В идеале, вход и выход нейросети должен иметь единичную дисперсию. В предыдущих работах использовалась параметризация $D_\theta(x; \sigma) = x - \sigma F_\theta(\cdot)$; в таком виде F_θ предсказывает стандартный нормальный шум. На практике же происходит следующее:

- При малых σ все хорошо: x — это практически чистый сигнал, а предсказание F_θ доминируется на маленьком числе \Rightarrow вклад ошибки нейросети очень мал;

- При больших σ все наоборот: x — это практически чистый шум, а выход F_θ домножается на большое число, что кратно увеличивает ошибку предсказания. Таким образом, мы вычитаем из чистого шума чистый шум и надеемся получить исходный сигнал — очевидно, не самая эффективная стратегия.

Стало быть, при больших σ проще предсказывать исходный сигнал, а при малых — добавленный шум. Этот подход можно непрерывно интерполировать на все σ , что дает параметризацию

$$D_\theta(x; \sigma) = c_{\text{skip}}(\sigma)x + c_{\text{out}}(\sigma)F_\theta(c_{\text{in}}(\sigma)x; c_{\text{noise}}(\sigma)). \quad (29)$$

Здесь $c_{\text{skip}}(\sigma)$ контролирует вклад “skip connection”, $c_{\text{in}}(\sigma)$ и $c_{\text{out}}(\sigma)$ шкалируют вход и выход F_θ и $c_{\text{noise}}(\sigma)$ шкалирует шум σ для F_θ . Функционал ошибки задается как

$$\mathbb{E}_{\sigma, y, n} [\lambda(\sigma) \|D_\theta(y + n; \sigma) - y\|_2^2], \quad (30)$$

где $\sigma \sim p_{\text{train}}$, $y \sim p_{\text{data}}$ и $n \sim \mathcal{N}(0, \sigma^2 I)$. Подставляя (29) в (30) и упрощая, получаем функционал ошибки для F_θ :

$$\mathbb{E}_{\sigma, y, n} \left[\underbrace{\lambda(\sigma) c_{\text{out}}(\sigma)^2}_{\text{weight}} \left\| F_\theta \left(\underbrace{c_{\text{in}}(\sigma)(y + n)}_{\text{input}}; c_{\text{noise}}(\sigma) \right) - \underbrace{\frac{1}{c_{\text{out}}(\sigma)} (y - c_{\text{skip}}(\sigma)(y + n))}_{\text{target}} \right\|_2^2 \right]. \quad (31)$$

Осталось подобрать предобуславливатели, веса $\lambda(\sigma)$ и распределение p_{train} . Вход должен иметь единичную дисперсию:

$$\mathbb{D}_{y, n} [c_{\text{in}}(\sigma)(y + n)] = 1 \iff c_{\text{in}}(\sigma) = 1 / \sqrt{\sigma^2 + \sigma_{\text{data}}^2}. \quad (32)$$

Здесь σ_{data}^2 — оценка дисперсии распределения p_{data} ¹⁰.

Обозначим target из (31) через $F_{\text{target}}(y, n; \sigma)$. Он также должен иметь единичную дисперсию. При этом, как видно из (29), коэффициент $c_{\text{out}}(\sigma)$ при F_θ должен быть минимален, чтобы не “раздувать” ошибку предсказания:

$$\begin{cases} \mathbb{D}_{y, n} [F_{\text{target}}(y, n; \sigma)] = 1 \\ c_{\text{skip}}(\sigma) = \arg \min_{c_{\text{skip}}(\sigma)} c_{\text{out}}(\sigma) \end{cases} \iff \begin{cases} c_{\text{out}}(\sigma) = \sigma \cdot \sigma_{\text{data}} / \sqrt{\sigma^2 + \sigma_{\text{data}}^2} \\ c_{\text{skip}}(\sigma) = \sigma_{\text{data}}^2 / (\sigma^2 + \sigma_{\text{data}}^2) \end{cases}. \quad (33)$$

Эти формулы согласуются с интуицией: если $\sigma \approx 0$, то $c_{\text{skip}}(\sigma) \approx 1$ и $F_{\text{target}}(y, n; \sigma) \propto n$. Если же $\sigma \rightarrow \infty$, то $c_{\text{skip}}(\sigma) \rightarrow 0$ и $F_{\text{target}}(y, n; \sigma) \rightarrow y$, т.е. при малых σ нейросеть предсказывает шум, а при больших — исходный сигнал, как мы изначально и хотели.

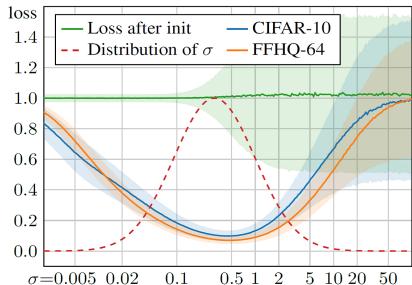
$\lambda(\sigma)$ подбирается так, чтобы все σ имели единичный вес в функционале (31):

$$\lambda(\sigma) c_{\text{out}}(\sigma)^2 = 1 \iff \lambda(\sigma) = (\sigma^2 + \sigma_{\text{data}}^2) / (\sigma \cdot \sigma_{\text{data}})^2. \quad (34)$$

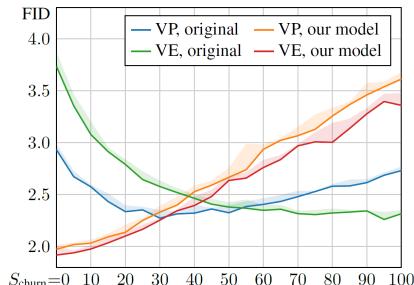
Это уравновешивает норму градиентов по всем уровням шума, что делает обучение более стабильным. Подробный вывод всех формул выше см. в [1, прил. B.6].

Чтобы подобрать $p_{\text{train}}(\sigma)$, авторы смотрят на значение функционала ошибки в зависимости от σ (рис. 9a). Видно, что значимо уменьшить ошибку возможно только на “промежуточных” уровнях шума в логарифмической шкале. Поэтому разумно во время обучения сэмплировать

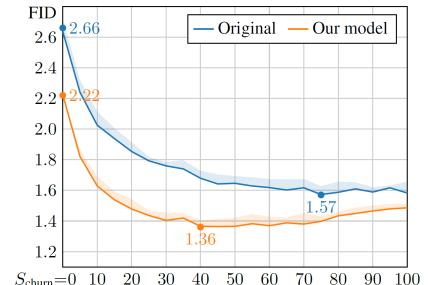
¹⁰ $\mathbb{D}y \approx \sigma_{\text{data}}^2 I$. В реальности, вообще говоря, $\text{cov}(y_i, y_j) \neq 0$ и $\mathbb{D}y_i \neq \mathbb{D}y_j$, поэтому оценка очень грубая. В (32) мы немного оскорбляем нотацию, неявно подразумевая, что матрицы ковариаций y и n имеют вид αI , и записывая уравнение для диагонального элемента.



(a) Распределение ошибок и шумов



(b) Стохастичность на CIFAR-10



(c) Стохастичность на ImageNet-64

Рис. 9: (a) Значение функционала ошибки после инициализации (зеленый график) и после обучения (синий и оранжевый) для различных σ (шкала логарифмическая). Красный график — плотность распределения $p_{\text{train}}(\sigma)$. (b, c) FID для стохастической генерации в зависимости от S_{churn} ($\text{NFE} = 511$).

σ из логнормального распределения: $\ln(\sigma) \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$, параметры подбираются эмпирически ¹¹.

Также модели на вход подается информация об уровне шума $c_{\text{noise}}(\sigma)$. Авторы кладут $c_{\text{noise}}(\sigma) = \frac{1}{4} \ln(\sigma)$ — так вход будет иметь нормальное распределение.

Со всеми предложенными улучшениями авторы обучают модели из [3] на CIFAR-10 ¹². С детерминированным сэмплером и $\text{NFE} = 35$ удалось выбрать FID 1.79 и 1.97 в задачах условной и безусловной генерации соответственно — SOTA на момент публикации. При этом занятно, что стохастический сэмплер (для любого $S_{\text{churn}} > 0$) только ухудшает качество (рис. 9b, красный и оранжевый графики).

Наконец, обучая с нуля модель для условной генерации на ImageNet-64, авторы выбивают FID 1.36 — также SOTA на момент публикации ($\text{NFE} = 511$). На этот раз, как можно видеть из рис. 9c, стохастичность оказалась полезна — без нее FID почти на единицу выше. Отсюда гипотеза, что стохастическая генерация релевантна скорее для больших и разнообразных данных.

6 Заключение

Итак, основной вклад работы [1] — объединение широкого класса диффузионных моделей под общий фреймворк, состоящий из независимых компонент. Авторы обоснованно сделали оптимальный выбор для каждой компоненты. Улучшения только процедуры генерации для предобученных моделей уже дают впечатляющее качество, а обучение с нуля со всеми улучшениями дало в свое время SOTA FID на CIFAR-10 и ImageNet-64.

Ограничения работы и открытые вопросы. Авторы не рассматривали, например,

- Различные вариации для архитектуры denoiser'a;
- Расширение пайплайна для генерации в высоком разрешении (super-resolution);
- Применение к large-scale моделям/данным.

¹¹Авторы задают $P_{\text{mean}} = -1.2$ и $P_{\text{std}} = 1.2$.

¹²Чтобы избежать переобучения, они также добавляют аугментации. При обучении в F_θ подается информация о типе аугментации, а во время генерации этот параметр выставляется в 0 — так модель генерирует изображения только из исходного распределения [1, прил. F.2].

Также для будущей работы были оставлены следующие вопросы:

- Применение методов Рунге-Кутты второго порядка с адаптивным подбором α для детерминированной генерации;
- Более оптимальный подбор параметров стохастической генерации и их связь с процедурой обучения модели;
- Адаптация фреймворка под методы с обучаемой дисперсией в обратном процессе [5].

Идеи для исследования. Автору разбора, в свою очередь, кажутся интересными следующие идеи:

- Аргументы в пользу расписаний $\sigma(t) = t$ и $s(t) = 1$ довольно убедительны, но можно ли более подробно и формально исследовать кривизну траекторий probability flow ODE? Имеет ли смысл подбирать расписания в зависимости от score функции? Формально кривизна определяется матрицей Гессе — может, попробовать как-то исследовать ее, чтобы понять, какие расписания лучше выпрямляют траектории?
- На диффузионную модель можно смотреть как на модель со скрытыми переменными. Однако распределение q на эти переменные фиксировано. А что если сделать его обучаемым (как в VAE)? Тогда в теории можно придать модели особые свойства: например, приоритетное зашумление отдельных частей изображения или упрощенный обратный процесс.

Благодарности. Автор выражает благодарность Дмитрию Петровичу Ветрову за лекции по диффузионным моделям на курсе “Байесовские методы в машинном обучении”, а также Денису Ракитину за приглашенные доклады на курсе “Стохастический анализ” и публикацию в открытый доступ видеозаписей занятий по курсу “Генеративные модели на основе диффузии”.

Список литературы

- [1] Tero Karras, Miika Aittala, Timo Aila и Samuli Laine. “Elucidating the Design Space of Diffusion-Based Generative Models”. B: *Proc. NeurIPS*. 2022.
- [2] Jonathan Ho, Ajay Jain и Pieter Abbeel. “Denoising Diffusion Probabilistic Models”. B: *arXiv preprint arxiv:2006.11239* (2020).
- [3] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma и др. “Score-Based Generative Modeling through Stochastic Differential Equations”. B: *Proc. ICLR* (2021). URL: <https://openreview.net/forum?id=PxTIG12RRHS>.
- [4] Brian D.O. Anderson. “Reverse-time diffusion equation models”. B: *Stochastic Processes and their Applications* 12(3) (1982), c. 313—326.
- [5] Alex Nichol и Prafulla Dhariwal. “Improved Denoising Diffusion Probabilistic Models”. B: *Proc. ICML* 139 (2021), c. 8162—8171.
- [6] Jiaming Song, Chenlin Meng и Stefano Ermon. “Denoising Diffusion Implicit Models”. B: *Proc. ICLR* (2021).
- [7] Miika Aittala. *Generative AI Research Spotlight: Demystifying Diffusion-Based Models*. 2023. URL: <https://developer.nvidia.com/blog/generative-ai-research-spotlight-demystifying-diffusion-based-models/> (дата обр. 06.11.2024).
- [8] Andrey Okhotin, Dmitry Molchanov, Vladimir Arkhipkin и др. “Star-Shaped Denoising Diffusion Probabilistic Models”. B: *Advances in Neural Information Processing Systems*. Т. 36. 2023.

Приложения

A Вывод формул (18)

Формулы (18), выражающие $s(t)$ и $\sigma(t)$ через коэффициенты SDE f и g в случае $f(x_t, t) = f(t)x_t$, приведены в [1, прил. B.1], но без вывода, поэтому мы приводим его тут.

Рассмотрим SDE $dx_t = f(t)x_t dt + g(t)dw_t$ и его дискретизацию с шагом h :

$$x_{t+h} = x_t + f(t)x_t h + g(t)\sqrt{h}\varepsilon_t, \quad (35)$$

где $\varepsilon_t \sim \mathcal{N}(0, 1)$. Для простоты работаем в одномерном случае. Хотим для любого t найти переходное распределение $p_{t|0}(x_t | x_0)$. Если в уравнении (35) выразить x_t через x_{t-h} по той же схеме, и так далее до x_0 , то получим, что x_{t+h} есть сумма независимых нормальных величин ε_t с какими-то коэффициентами плюс константа, зависящая от x_0 . Таким образом, $p_{t|0}(x_t | x_0) = \mathcal{N}(x_t | m_t, v_t)$ ¹³. Осталось вычислить $m_t = \mathbb{E}[x_t | x_0]$ и $v_t = \mathbb{D}[x_t | x_0]$.

Навешивая условные ожидания на обе части уравнения (35), получаем

$$m_{t+h} = m_t + f(t)hm_t \iff \frac{m_{t+h} - m_t}{h} = f(t)m_t. \quad (36)$$

Устремляя $h \rightarrow 0$, получаем задачу Коши

$$\begin{cases} \dot{m}_t = f(t)m_t \\ m_0 = x_0 \end{cases} \iff m_t = x_0 \cdot \exp\left(\int_0^t f(\xi) d\xi\right) \quad (37)$$

Теперь вынесем за скобку x_t в (35) и навесим условную дисперсию:

$$v_{t+h} = (1 + f(t)h)^2 v_t + g^2(t)h \iff \quad (38)$$

$$v_{t+h} = (1 + 2f(t)h)v_t + g^2(t)h + o(h) \iff \quad (39)$$

$$\frac{v_{t+h} - v_t}{h} = 2f(t)v_t + g^2(t) + o(1). \quad (40)$$

Устремляя $h \rightarrow 0$, получим

$$\dot{v}_t = 2f(t)v_t + g^2(t) \iff \dot{v}_t - 2f(t)v_t = g^2(t). \quad (41)$$

Теперь домножим обе части последнего равенства на $a_t = \exp\left(-\int_0^t 2f(\xi) d\xi\right)$. Легко убедиться, что $\dot{v}_t a_t - 2f(t)v_t a_t = (v_t a_t)'$, поэтому наше дифференциальное уравнение эквивалентно

$$(v_t a_t)' = g^2(t)a_t. \quad (42)$$

По формуле Ньютона-Лейбница, $v_t a_t - v_0 a_0 = \int_0^t g^2(\xi) a_\xi d\xi$, и $v_0 = 0$, как дисперсия константы.

Отсюда

$$v_t = a_t^{-1} \int_0^t g^2(\xi) a_\xi d\xi \quad (43)$$

¹³К сожалению, автор не владеет достаточным математическим аппаратом, чтобы доказать нормальность строго, поэтому приходится через дискретизацию.

Итак, мы нашли переходное распределение $p_{t|0}(x_t | x_0) = \mathcal{N}(x_t | m_t, v_t)$ для исходного SDE. От этого SDE мы требуем, чтобы оно описывало процесс зашумления. То есть с другой стороны, переходные распределения должны иметь вид (15):

$$p_{t|0}(x_t | x_0) = \mathcal{N}(x_t | s(t)x_0, s^2(t)\sigma^2(t)). \quad (44)$$

Соответственно, мат. ожидания и дисперсии обязаны совпасть. Получаем

$$s(t) = \exp \left(\int_0^t f(\xi) d\xi \right). \quad (45)$$

Замечая, что $a_t^{-1} = s^2(t)$, выражаем $\sigma(t)$:

$$s^2(t)\sigma^2(t) = a_t^{-1} \int_0^t g^2(\xi)a_\xi d\xi \iff \quad (46)$$

$$s^2(t)\sigma^2(t) = s^2(t) \int_0^t \frac{g^2(\xi)}{s^2(\xi)} d\xi \iff \quad (47)$$

$$\sigma(t) = \sqrt{\int_0^t \frac{g^2(\xi)}{s^2(\xi)} d\xi}. \quad (48)$$

В частности, для VE SDE $dx_t = g(t) dw_t$ получаем $s(t) = 1$ и

$$\sigma^2(t) = \int_0^t g^2(\xi) d\xi \iff g(t) = \sqrt{\frac{d[\sigma^2(t)]}{dt}}. \quad (49)$$

В такой форме VE SDE приводится в работе [3, урав. 9]. Применяя chain rule, можно упростить $g(t) = \sqrt{2\dot{\sigma}(t)\sigma(t)}$ и получить еще одну эквивалентную форму: $dx_t = \sqrt{2\dot{\sigma}(t)\sigma(t)} dw_t$. Именно она получается подстановкой $\beta(t) = \frac{\dot{\sigma}(t)}{\sigma(t)}$ в (28).

Также отметим, что изначальное предположение $f(x_t, t) = f(t)x_t$ критично, т.к. без него не получилось бы составить уравнения на m_t и v_t .

B Дополнительные соображения и источники

Автора давно интересовало, почему в диффузионных моделях, вообще говоря, нельзя зашумлять и удалять шум за один шаг. Это можно объяснить с разных сторон:

- С точки зрения ODE/SDE: если траектория не прямая, то очевидно, что большой шаг сразу в $t = 0$ вряд ли даст что-то хорошее;
- С точки зрения байесовского вывода: теоретический оптимум в задаче (22) есть условное ожидание $D(x_t; t) = \mathbb{E}[x_0 | x_t]$. Если мы в прямом процессе за один шаг превращаем объект x_0 в чистый шум x_1 , то $D(x_1; 1) = \mathbb{E}[x_0 | x_1] \approx \mathbb{E}x_0$, т.к. x_0 и x_1 практически независимы. То есть лучшее, что может выучить denoiser — среднее по выборке (рис. 5b);
- С точки зрения Annealed Langevin dynamics ¹⁴: соседние распределения должны быть похожи друг на друга, чтобы после каждой внешней итерации получалось хорошее начальное приближение для следующей.

Однако недавний метод InstaFlow (ICLR 2024) ¹⁵ показывает, что качественная генерация всего за один шаг возможна. Там как раз используется идея о выпрямлении траекторий ODE.

Источники. При написании разбора автору сильно помогли следующие материалы в открытом доступе:

- Miika Aittala: Elucidating the Design Space of Diffusion-Based Generative Models, <https://youtu.be/T0Qxzf0eaio>;
- Стохастический анализ, Лекция 18: Приглашенный доклад (Диффузионные модели), <https://www.youtube.com/live/sf-Tx9fDe2I>;
- Стохастический анализ, Лекция 18+: Приглашенный доклад (Диффузионные модели), <https://www.youtube.com/live/b04PKXXppu8>;
- http://wiki.cs.hse.ru/Генеративные_модели_на_основе_диффузии.

Официальная имплементация: <https://github.com/NVlabs/edm>.

¹⁴Generative Modeling by Estimating Gradients of the Data Distribution, <https://arxiv.org/abs/1907.05600>.

¹⁵InstaFlow: One Step is Enough for High-Quality Diffusion-Based Text-to-Image Generation, <https://arxiv.org/abs/2309.06380>.