ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Лекция 14

Диффузионные генеративные модели

DDPM, стохастические процессы, обучение предсказанию шума и генерация изображений

Преподаватель: Аршын Алтыбай

Кафедра компьютерной инженерии,
Алматинский технологический университет, Алматы, Казахстан

Учебный год: 2026

> **Место в курсе.** В лекциях 11–13 рассматривались автоэнкодеры, Transformer и GAN. Теперь изучается другой принцип генерации: случайное постепенное разрушение данных гауссовским шумом и обучение нейронной сети обращать этот процесс. Базовая работа по DDPM — Ho, Jain и Abbeel [11], а более ранняя постановка диффузионного моделирования — Sohl-Dickstein и соавторы [10].

1. Цель лекции и планируемые результаты

Цель — последовательно вывести вероятностную модель диффузии, объяснить вариационное обучение и построить простой генератор изображений на PyTorch. Изложение предполагает знание нормального распределения, многомерной линейной алгебры, обратного распространения ошибки и основ сверточных сетей [6, 7].

После изучения студент умеет:

1. различать генерацию при помощи GAN, VAE и диффузионных моделей;
2. вычислять закон прямого процесса $q(x_t \mid x_0)$ без моделирования всех промежуточных шагов;
3. выводить апостериорное распределение $q(x_{t-1} \mid x_t, x_0)$;
4. формулировать вариационную нижнюю границу (ELBO) и функцию потерь DDPM;
5. объяснять параметризации ϵ, x_0 и v;
6. реализовывать прямую диффузию, простую сеть предсказания шума и обратный процесс;

7. корректно тестировать качество генерации, фиксировать случайные начальные условия;
8. сравнивать DDPM, DDIM, условную генерацию и латентную диффузию.

Обозначения $x_0 \in \mathbb{R}^d$ — исходный объект; x_t — его зашумленная версия на шаге t; T — число шагов; $\beta_t \in (0,1)$ — дисперсия шума; $\alpha_t = 1 - \beta_t$; $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$; $\epsilon \sim \mathcal{N}(0, I)$ — стандартный гауссовский шум; ϵ_θ — обучаемый предсказатель шума. Индекс t является дискретным временем алгоритма, а не физическим временем.

2. Почему возникли диффузионные модели

Генеративная модель должна строить новые объекты, похожие на наблюдаемые данные. Математически пусть $p_{\text{data}}(x)$ обозначает неизвестный закон распределения реальных изображений. Наша цель — параметризовать $p_\theta(x)$ так, чтобы новые сэмплы модели имели близкую статистику и содержательную структуру.

В GAN совместно обучаются две модели, что может приводить к неустойчивой динамике и коллапсу мод [8]. В VAE вводится латентная переменная и оптимизируется вариационная граница, причём конкретные архитектуры и функции потерь определяют свойства реконструкции [9]. В диффузионных моделях обучение удобно сводится к предсказанию специально добавленного шума: на каждом шаге задача контролируется известной случайной величиной.

Семейство	Ключевая идея	Обучение	Особенность генерации
GAN	Состязание генератора и дискриминатора	Минимакс / две функции потерь	Нередко один проход сети
VAE	Латентное вероятностное представление	ELBO, реконструкция и KL	Выборка z, затем декодирование
DDPM	Последовательное удаление шума	Предсказание шума / вариационная цель	Много последовательных шагов

Важная оговорка. Ни одна из моделей не гарантирует истинность или безопасность сгенерированного содержимого. Генерация изображения —

статистическая задача, а не проверка фактов.

3. Вероятностная база: гауссовские распределения

Стандартный многомерный нормальный вектор $\epsilon \sim \mathcal{N}(0, I_d)$ имеет среднее 0 и ковариационную матрицу I_d. Для фиксированного вектора μ и матрицы A верно

$$\mu + A\epsilon \sim \mathcal{N}(\mu, AA^\top). \tag{1}$$

В частности, если $A = \sigma I_d$, то $x = \mu + \sigma\epsilon$ имеет распределение $\mathcal{N}(\mu, \sigma^2 I_d)$.

> **Почему именно гауссовский шум?** Сумма независимых гауссовских векторов опять имеет гауссовское распределение. Это позволяет аналитически вычислять закон зашумленного объекта на любом шаге и существенно ускоряет обучение DDPM [11].

Плотность $\mathcal{N}(\mu, \sigma^2 I_d)$ равна

$$p(x) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{\|x-\mu\|_2^2}{2\sigma^2}\right). \tag{2}$$

При двух распределениях с одинаковой ковариацией $\sigma^2 I_d$ их KL-дивергенция принимает форму

$$D_{\mathrm{KL}}\left(\mathcal{N}(\mu_1, \sigma^2 I_d) \,\|\, \mathcal{N}(\mu_2, \sigma^2 I_d)\right) = \frac{\|\mu_1 - \mu_2\|_2^2}{2\sigma^2}. \tag{3}$$

Это равенство объясняет, почему регрессия среднего обратного процесса приводит к квадратичной ошибке предсказания шума.

4. Прямой диффузионный процесс

4.1 Марковская цепь постепенного зашумления

Выберем возрастающий или иной допустимый набор коэффициентов $\beta_1, \dots, \beta_T \in (0,1)$. Условный закон шага прямого процесса задается формулой

$$q(x_t \mid x_{t-1}) = \mathcal{N}\big(\sqrt{1-\beta_t}\, x_{t-1}, \beta_t I_d\big) = \mathcal{N}(\sqrt{\alpha_t}\, x_{t-1}, (1-\alpha_t) I_d). \tag{4}$$

Соответствующее правило выборки:

$$x_t = \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t, \qquad \epsilon_t \overset{\text{iid}}{\sim} \mathcal{N}(0, I_d). \tag{5}$$

Цепь называется марковской, поскольку x_t при фиксированном x_{t-1} не зависит от более ранних состояний. Полное условное распределение раскладывается:

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}). \tag{6}$$

4.2 Непосредственная выборка состояния на любом шаге

Наивная реализация вычисляла бы последовательно $x_1, x_2, \dots, x_t$. Но для обучения это необязательно. Подставим формулу для x_1 в формулу для x_2:

$$x_1 = \sqrt{\alpha_1} x_0 + \sqrt{1-\alpha_1}\, \epsilon_1, \tag{7}$$

$$x_2 = \sqrt{\alpha_2 \alpha_1} x_0 + \sqrt{\alpha_2(1-\alpha_1)}\, \epsilon_1 + \sqrt{1-\alpha_2}\, \epsilon_2. \tag{8}$$

Случайная часть справа — сумма независимых нормальных векторов. Её ковариация:

$$\big[\alpha_2(1-\alpha_1) + (1-\alpha_2)\big] I_d = (1-\alpha_1\alpha_2) I_d. \tag{9}$$

По индукции получаем один из основных результатов DDPM:

$$\boxed{q(x_t \mid x_0) = \mathcal{N}\big(\sqrt{\bar\alpha_t}\, x_0, (1-\bar\alpha_t) I_d\big).} \tag{10}$$

Или, в форме репараметризации,

$$\boxed{x_t = \sqrt{\bar\alpha_t}\, x_0 + \sqrt{1-\bar\alpha_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I_d).} \tag{11}$$

Эта формула позволяет выбрать случайное t, получить x_t за одно действие и обучить сеть восстанавливать ϵ [11].

> **Доказательство по индукции** Предположим, что $x_{t-1} \mid x_0 \sim \mathcal{N}(\sqrt{\bar\alpha_{t-1}} x_0, (1-\bar\alpha_{t-1}) I_d)$. Тогда из (5) среднее x_t равно $\sqrt{\alpha_t \bar\alpha_{t-1}} x_0 = \sqrt{\bar\alpha_t} x_0$, а ковариация равна $[\alpha_t(1-\bar\alpha_{t-1}) + (1-\alpha_t)] I_d = (1-\bar\alpha_t) I_d$. Начальное значение $t=1$ следует непосредственно из определения.

4.3 Интерпретация сигнала и шума

В формуле (11) множитель $\sqrt{\bar{\alpha}_t}$ управляет долей сигнала, а $\sqrt{1-\bar{\alpha}_t}$ — долей шума. При $\bar{\alpha}_t \approx 1$ сохраняется почти исходное изображение; при $\bar{\alpha}_t \approx 0$ состояние близко к стандартному гауссовскому шуму. Последнее утверждение является приближенным: для конечного T точное совпадение с $\mathcal{N}(0, I_d)$ обычно не достигается.

Можно ввести отношение сигнал/шум $\mathrm{SNR}(t) = \bar{\alpha}_t/(1-\bar{\alpha}_t)$. Эта величина помогает понимать вклад различных уровней зашумления.

5. Численные примеры прямой диффузии

5.1 Пример для одномерного сигнала

Пусть $x_0 = 2$, $\beta_1 = 0.1$, $\beta_2 = 0.2$, $\epsilon = -0.5$. Тогда $\alpha_1 = 0.9$, $\alpha_2 = 0.8$, $\bar{\alpha}_2 = 0.72$ и

$$x_2 = \sqrt{0.72}\cdot 2 + \sqrt{0.28}\cdot(-0.5) \tag{12}$$

$$\approx 1.69706 - 0.26458 = 1.43248. \tag{13}$$

Условное среднее равно 1.69706, а условная дисперсия — 0.28. Важно: выбранное значение ϵ — лишь одна реализация; фактическое x_2 случайно.

5.2 Пример с двумерным вектором

Возьмем $x_0 = (1, -1)^\top$, $\bar{\alpha}_t = 0.64$ и $\epsilon = (0.5, -2)^\top$. Тогда

$$x_t = 0.8(1,-1)^\top + 0.6(0.5,-2)^\top = (1.1, -2)^\top. \tag{14}$$

Для изображений та же операция применяется покомпонентно к тензору формы (C, H, W).

Пример 1. Прямое зашумление массива на NumPy

```python
import numpy as np

rng = np.random.default_rng(42)
x0 = np.array([1.0, -1.0], dtype=float)
abar = 0.64
eps = np.array([0.5, -2.0])
xt = np.sqrt(abar) * x0 + np.sqrt(1 - abar) * eps
print(xt)
# [ 1.1 -2. ]
```

6. График шума: выбор расписания β_t

6.1 Линейное расписание

Простейший вариант: $\beta_t = \beta_{\min} + (\beta_{\max} - \beta_{\min})(t-1)/(T-1)$ при $T > 1$. Его удобно понимать и реализовать, но подходящие границы зависят от количества шагов, масштаба данных и параметризации модели.

6.2 Косинусное расписание

Nichol и Dhariwal [12] предложили задавать накопленное сохранение сигнала через косинусную функцию. Пусть

$$f(t) = \cos^2\left(\frac{t/T + s}{1+s}\frac{\pi}{2}\right), \quad \bar{\alpha}_t = \frac{f(t)}{f(0)}, \tag{15}$$

где $s > 0$ — небольшой сдвиг. Для вычислений $\beta_t = 1 - \bar{\alpha}_t/\bar{\alpha}_{t-1}$ дополнительно ограничивают сверху, чтобы избежать численных особенностей на последнем шаге.

> **Типичная ошибка** Не следует путать α_t с $\bar{\alpha}_t$. Первая величина характеризует один шаг, вторая — весь накопленный процесс. Неправильное использование этих коэффициентов приводит к генерации шума вместо объектов.

7. Обратный процесс: задача генерации

В обучении прямой процесс известен. При генерации исходного x_0 нет: доступен только случайный $x_T \sim \mathcal{N}(0, I_d)$. Нужно постепенно получить $x_{T-1}, \ldots, x_0$. Обратная модель параметризуется нейронной сетью:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t), \quad p(x_T) = \mathcal{N}(0, I_d). \tag{16}$$

В DDPM обратный переход выбирают гауссовским:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\big). \tag{17}$$

Почему это разумно? Для достаточно малых шагов обратный переход имеет удобное локальное приближение; тем не менее точное обратное условное распределение $q(x_{t-1} \mid x_t)$ после усреднения по данным вообще не обязано быть одним гауссовским распределением. Гауссовская модель — выбранное семейство аппроксимаций.

8. Апостериорный закон одного шага

8.1 Точное распределение при известном x_0

Чтобы вывести формулы обучения, важно рассмотреть условное распределение $q(x_{t-1} \mid x_t, x_0)$. Поскольку $q(x_t \mid x_{t-1})$ и $q(x_{t-1} \mid x_0)$ гауссовские, произведение их плотностей как функции x_{t-1} снова пропорционально гауссовской плотности:

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\big(\widetilde{\mu}_t(x_t, x_0), \widetilde{\beta}_t I_d\big), \qquad t \geq 2, \tag{18}$$

где

$$\widetilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\, \beta_t, \tag{19}$$

$$\widetilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\, \beta_t}{1 - \bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} x_t. \tag{20}$$

При $t = 1$ условное распределение $q(x_0 \mid x_1, x_0)$ вырождено в точке x_0, поэтому в вариационном разложении этот случай выделяется отдельно.

8.2 Вывод методом выделения полного квадрата

Из формулы Байеса с точностью до множителя, не зависящего от x_{t-1}, имеем

$$q(x_{t-1} \mid x_t, x_0) \propto \exp\left[-\frac{\|x_t - \sqrt{\alpha_t}x_{t-1}\|^2}{2\beta_t} - \frac{\|x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0\|^2}{2(1-\bar{\alpha}_{t-1})}\right]. \tag{21}$$

Коэффициент при $\|x_{t-1}\|^2/2$ составляет

$$\widetilde{\beta}_t^{-1} = \frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}} = \frac{1-\bar{\alpha}_t}{\beta_t(1-\bar{\alpha}_{t-1})}. \tag{22}$$

Коэффициент при линейной части, умноженный на $\widetilde{\beta}_t$, даёт (20). Эта алгебра имеет практический смысл: она позволяет поставить обучение обратного перехода как задачу предсказания известного условного среднего [11].

9. Параметризация обратного среднего через шум

Из (11) формально восстановим исходный объект по шуму:

$$x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon}{\sqrt{\bar{\alpha}_t}}. \tag{23}$$

Подставляя в (20) и упрощая, получаем

$$\widetilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right). \tag{24}$$

Во время генерации истинный ϵ неизвестен. На его место подставляют предсказание нейронной сети $\epsilon_\theta(x_t, t)$:

$$\boxed{\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right).} \tag{25}$$

Именно эта формула — основа стандартного шага генерации DDPM. Для фиксированной дисперсии обратного процесса далее можно положить $\Sigma_\theta(x_t, t) = \sigma_t^2 I_d$; варианты σ_t^2 обсуждаются ниже.

> **Проверка размерностей** x_t и $\epsilon_\theta(x_t, t)$ имеют одинаковую форму, например $(B, 3, 64, 64)$ для пакета RGB-изображений. Коэффициенты $\alpha_t, \bar{\alpha}_t, \beta_t$ по каждому элементу пакета должны корректно транслироваться (broadcasting) по каналам и координатам.

10. Вероятностная цель обучения: ELBO

10.1 От правдоподобия к вариационной границе

Хотелось бы максимизировать $\log p_\theta(x_0)$, где скрытые состояния $x_{1:T}$ интегрируются. Введём известное распределение прямой диффузии $q(x_{1:T} \mid x_0)$ и применим неравенство Йенсена:

$$\log p_\theta(x_0) = \log \int q(x_{1:T} \mid x_0) \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\, dx_{1:T} \tag{26}$$

$$\geq \mathbb{E}_q \left[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0)\right] =: \mathcal{L}_{\text{ELBO}}(x_0). \tag{27}$$

Следовательно, $-\mathcal{L}_{\text{ELBO}}$ — верхняя оценка отрицательного логарифмического правдоподобия, а не произвольная эвристическая функция потерь [9, 11].

10.2 Разложение по шагам

Используем марковские факторизации и тождество Байеса:

$$\begin{aligned} -\mathcal{L}_{\text{ELBO}}(x_0) = {} & D_{\text{KL}}\big(q(x_T \mid x_0) \,\|\, p(x_T)\big) \\ & + \sum_{t=2}^{T} \mathbb{E}_{q(x_t \mid x_0)} D_{\text{KL}}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\big) \\ & - \mathbb{E}_{q(x_1 \mid x_0)} \log p_\theta(x_0 \mid x_1). \end{aligned} \tag{28}$$

Первое слагаемое связано с согласованием последнего зашумленного состояния и стандартного нормального prior. Средние слагаемые измеряют качество отдельных обратных переходов. Последнее слагаемое отвечает за декодирование из x_1 в исходное наблюдение.

10.3 Почему возникает квадратичная потеря

Если при $t \geq 2$ обе нормальные плотности имеют одинаковую, фиксированную ковариацию $\sigma_t^2 I_d$, то из (3) следует

$$L_{t-1} = \mathbb{E}\left[\frac{\|\widetilde{\mu}_t - \mu_\theta\|_2^2}{2\sigma_t^2}\right] + \text{const.} \tag{29}$$

Из (24)–(25):

$$L_{t-1} = \mathbb{E}\left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right] + \text{const.} \tag{30}$$

Таким образом, квадратичная ошибка в предсказании шума математически связана с вариационной границей, но её коэффициенты зависят от шага t. Популярная упрощенная потеря DDPM отбрасывает эти веса: это отдельный выбор цели обучения, а не буквальное тождество с полной ELBO.

11. Упрощенная функция потерь DDPM

На практике часто минимизируют

$$\boxed{L_{\text{simple}}(\theta) = \mathbb{E}_{x_0 \sim p_{\text{data}},\, t \sim U\{1,\dots,T\},\, \epsilon \sim \mathcal{N}(0,I)}\left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|_2^2\right].} \tag{31}$$

Если применяется `MSELoss(reduction="mean")`, ошибка дополнительно нормируется числом элементов тензора; при фиксированных размерах это меняет масштаб градиента, но не точку минимума данной функции потерь.

1. Берём случайное реальное изображение x_0.
2. Независимо выбираем шаг t равномерно из $1, \dots, T$.
3. Генерируем $\epsilon \sim \mathcal{N}(0, I)$.
4. Строим x_t по закрытой формуле (11).
5. Передаем (x_t, t) в нейросеть.
6. Сравниваем ответ $\epsilon_\theta(x_t, t)$ с известным ϵ.
7. Обновляем параметры θ градиентным оптимизатором.

> **Почему нельзя подавать модели только x_t?** Одинаковая визуальная структура может соответствовать разной степени зашумления. Поэтому временная переменная t — обязательный вход сети либо напрямую, либо через кодирование времени. Она помогает различать, какой тип шума требуется удалить.

12. Архитектура нейросети предсказания шума

12.1 От MLP к U-Net

Для небольших двумерных точек $x_0 \in \mathbb{R}^2$ может хватить многослойного перцептрона, получающего пару (x_t, t). Для изображений обычно применяют сверточные модели U-Net с несколькими пространственными масштабами [18, 11]. Сеть должна вернуть тензор той же формы, что и входное изображение.

Типичный U-Net содержит нисходящую ветвь с уменьшением пространственного разрешения, центральный блок и восходящую ветвь; связи между соответствующими уровнями позволяют сохранить локальные детали. Residual blocks и attention-блоки улучшают представление информации на разных масштабах [15].

12.2 Временные эмбеддинги

Простейшая инженерная реализация использует нормализованное время t/T. В более развитых архитектурах применяют синусоидальные эмбеддинги:

$$e_{2i}(t) = \sin\left(t/10000^{2i/d_e}\right), \tag{32}$$

$$e_{2i+1}(t) = \cos\left(t/10000^{2i/d_e}\right), \tag{33}$$

где d_e — размерность кодирования. Полученные признаки затем преобразуются полносвязными слоями и добавляются к промежуточным картам признаков.

12.3 Сравнение вычислительной сложности

Для изображения формы (C, H, W) один прямой проход сети имеет сложность, определяемую ее архитектурой. Обучение DDPM обычно требует один вызов предсказателя шума на выбранном случайном шаге; стандартная генерация DDPM требует порядка T вызовов, что объясняет важность ускоренных алгоритмов выборки.

13. Алгоритм генерации DDPM

Инициализируем $x_T \sim \mathcal{N}(0, I_d)$. Для $t = T, T-1, \ldots, 1$ вычисляем

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z, \tag{34}$$

где $z \sim \mathcal{N}(0, I_d)$ при $t > 1$, а при последнем шаге берём $z = 0$. Один из классических вариантов — $\sigma_t^2 = \widetilde{\beta}_t$ для $t \geq 2$; также рассматриваются другие схемы фиксированной и обучаемой дисперсии [11, 12].

> **Процесс генерации не является простым удалением случайного шума** На каждом шаге нейросеть предсказывает структуру добавленного шума, соответствующую текущему состоянию и текущему времени. Затем алгоритм корректирует условное среднее и, кроме последнего шага, может добавлять новую случайную компоненту. Именно повторение этого правила приводит к формированию нового изображения.

13.1 Контроль правильности

Перед генерацией следует убедиться, что порядок индексов времени при обучении и генерации совпадает; величины $\bar{\alpha}_t$ соответствуют тем же β_t; форма z совпадает с формой x_t; последний шаг не содержит дополнительного случайного шума.

14. Пример на PyTorch: подготовка расписания и зашумление

Пример 2. Коэффициенты DDPM и прямой процесс

```python
import torch

T = 200
betas = torch.linspace(1e-4, 0.02, T)
alphas = 1.0 - betas
alpha_bar = torch.cumprod(alphas, dim=0)

def gather_to_image(arr, t, x):
    # arr: (T,), t: (B,), x: (B,C,H,W)
    return arr[t].view(-1, 1, 1, 1).to(x.device)

```

Пример 2 (продолжение)

```
def q_sample(x0, t, noise=None):
    if noise is None:
        noise = torch.randn_like(x0)
    abar = gather_to_image(alpha_bar, t, x0)
    xt = abar.sqrt() * x0 + (1.0 - abar).sqrt() * noise
    return xt, noise

x0 = torch.zeros(4, 1, 28, 28)
t = torch.randint(0, T, (x0.shape[0],))
xt, true_noise = q_sample(x0, t)
print(xt.shape, true_noise.shape)
# torch.Size([4, 1, 28, 28]) torch.Size([4, 1, 28, 28])
```

В коде временной индекс $t = 0, \ldots, T-1$ соответствует математическому шагу $t+1$. Это распространённое соглашение в программных библиотеках. Функция `gather_to_image` обеспечивает корректное широковещательное умножение каждого изображения пакета на собственный коэффициент расписания.

15. Минимальная нейросеть шумопредсказания для MNIST

Следующий пример намеренно использует небольшую CNN для учебных целей. Он показывает интерфейс $\epsilon_\theta(x_t, t)$, но не претендует на качество больших U-Net. Нормализация пикселей в интервал $[-1, 1]$ согласована с диапазоном обрабатываемых тензоров.

Пример 3. Компактная модель предсказания шума

```
import torch
from torch import nn

class NoiseCNN(nn.Module):
    def __init__(self, hidden=64):
        super().__init__()
        self.net = nn.Sequential(
            nn.Conv2d(2, hidden, 3, padding=1),
            nn.SiLU(),
            nn.Conv2d(hidden, hidden, 3, padding=1),
            nn.SiLU(),
            nn.Conv2d(hidden, hidden, 3, padding=1),
            nn.SiLU(),
            nn.Conv2d(hidden, 1, 3, padding=1),
        )

    def forward(self, x, t):
        B, _, H, W = x.shape
```

Пример 3 (продолжение)

```
        tchan = (t.float() / (T - 1)).view(B, 1, 1, 1)
        tchan = tchan.expand(B, 1, H, W)
        return self.net(torch.cat([x, tchan], dim=1))

model = NoiseCNN()
x = torch.randn(8, 1, 28, 28)
t = torch.randint(0, T, (8,))
print(model(x, t).shape)
# torch.Size([8, 1, 28, 28])
```

15.1 Ограничения компактной сети

Сеть из примера обрабатывает локальные признаки и получает временной канал. Но она имеет ограниченное рецептивное поле, не содержит многошкальной обработки и для сложных изображений обычно существенно уступает полноценному U-Net. Улучшение архитектуры — самостоятельная задача студента, а не гарантированный результат исходного примера.

16. Обучение на реальных изображениях

Пример 4. Учебный цикл обучения на MNIST

```
import torch
from torch import nn
from torch.utils.data import DataLoader
from torchvision import datasets, transforms

# Функции q_sample и класс NoiseCNN определены выше.
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
trans = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,)),
])
dataset = datasets.MNIST(root='./data', train=True,
                         download=True, transform=trans)
loader = DataLoader(dataset, batch_size=128, shuffle=True)
model = NoiseCNN().to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

# Для GPU расписание переносим на тот же device.
betas = betas.to(device)
alphas = alphas.to(device)
alpha_bar = alpha_bar.to(device)

for epoch in range(5):
```

Пример 4 (продолжение)

```
    total = 0.0
    for x0, _ in loader:
        x0 = x0.to(device)
        t = torch.randint(0, T, (x0.size(0),), device=device)
        noise = torch.randn_like(x0)
        xt, _ = q_sample(x0, t, noise)
        estimate = model(xt, t)
        loss = nn.functional.mse_loss(estimate, noise)
        optimizer.zero_grad(set_to_none=True)
        loss.backward()
        optimizer.step()
        total += loss.item() * x0.size(0)
    print(f'epoch={epoch+1}, mse={total/len(dataset):.5f}')
```

Комментарии. Данные MNIST загружаются из сети при первом запуске. Скорость работы зависит от CPU/GPU. Пять эпох приведены для демонстрации процедуры, а не как универсально достаточное количество для качественной генерации. Нельзя утверждать, что сеть научилась генерировать корректные цифры, пока не исследованы сохранённые сэмплы и динамика метрик.

17. Реализация обратной выборки на PyTorch

В следующем примере используется дисперсия $\widetilde{\beta}_t$ для шагов $t \geq 2$. При переходе к нулевому программному индексу промежуточное значение $\bar{\alpha}_{t-1}$ удобно брать из предварительно вычисленного массива со значением 1 для отсутствующего предыдущего шага.

Пример 5. Генерация после обучения

```
@torch.no_grad()
def generate(model, n=16, channels=1, size=28):
    model.eval()
    x = torch.randn(n, channels, size, size, device=device)
    abar_prev = torch.cat([torch.ones(1, device=device),
                           alpha_bar[:-1]])
    for i in reversed(range(T)):
        t = torch.full((n,), i, device=device, dtype=torch.long)
        eps_pred = model(x, t)
        mean = (x - betas[i] / torch.sqrt(1 - alpha_bar[i])
                * eps_pred) / torch.sqrt(alphas[i])
        if i > 0:
            beta_tilde = betas[i] * (1 - abar_prev[i]) / (
                1 - alpha_bar[i])
            x = mean + beta_tilde.sqrt() * torch.randn_like(x)
        else:
            x = mean
```

Пример 5 (продолжение)

```
    return x.clamp(-1, 1)

samples = generate(model, n=16)
print(samples.shape)  # torch.Size([16, 1, 28, 28])
```

Важно для воспроизводимости В коде `clamp(-1,1)` применяется к окончательному изображению для визуализации, а не на каждом переходе. Преждевременное ограничение значений изменяет задаваемый стохастический процесс. Сохраняйте начальное случайное состояние, версию PyTorch, расписание коэффициентов и веса модели.

18. Сравнение предсказания шума, исходного изображения и скорости

Существуют разные эквивалентные по представлению выхода параметризации модели при фиксированном времени и расписании:

1. **Предсказание шума** ϵ**.** Сеть оценивает гауссовскую компоненту в (11); это классическая форма DDPM.
2. **Предсказание чистого объекта** x_0**.** Сеть непосредственно приближает исходное изображение. Из такой оценки шум можно восстановить алгебраически.
3. **Предсказание** v**.** Часто определяют $v_t = \sqrt{\bar{\alpha}_t}\,\epsilon - \sqrt{1-\bar{\alpha}_t}\,x_0$. При таком соглашении из пары (x_t, v_t) получаем

$$x_0 = \sqrt{\bar{\alpha}_t}\,x_t - \sqrt{1-\bar{\alpha}_t}\,v_t, \tag{35}$$

$$\epsilon = \sqrt{1-\bar{\alpha}_t}\,x_t + \sqrt{\bar{\alpha}_t}\,v_t. \tag{36}$$

Разные параметризации приводят к разным весам ошибок и практическим свойствам оптимизации. Их нельзя безоговорочно назвать «одинаковыми» после выбора конкретной функции потерь, архитектуры и схемы выборки.

19. DDIM: ускоренная выборка

Стандартная DDPM-процедура использует последовательные стохастические переходы. В DDIM [13] рассматривают другую конструкцию немарковских прямых процессов, согласованных с теми же маргиналами, и получают семейство ускоренных алгоритмов. Для перехода от текущего шага t к более раннему $s < t$ можно записать

$$\widehat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}, \tag{37}$$

$$x_s = \sqrt{\bar{\alpha}_s}\,\widehat{x}_0 + \sqrt{1-\bar{\alpha}_s-\sigma_{t\to s}^2}\,\epsilon_\theta(x_t, t) + \sigma_{t\to s} z. \tag{38}$$

Параметр $\sigma_{t\to s}$ задаёт степень стохастичности перехода. При $\sigma_{t\to s} = 0$ выборка детерминирована после выбора начального x_T; при других значениях добавляется случайность. Формула требует неотрицательности выражения под корнем и согласованного расписания шагов.

19.1 Что означает «ускорение»?

Если для генерации используется подмножество $T' > 0$ временных шагов вместо всех T, число прямых проходов нейронной сети снижается примерно с T до T'. При этом компромисс между скоростью и качеством зависит от обученной модели, схемы отбора шагов и метода интегрирования; меньший T' не гарантирует сохранение качества.

20. Условные диффузионные модели

В простейшей постановке сеть обучается на неразмеченных изображениях. Но можно генерировать объект с дополнительным условием c: названием класса, текстовым описанием, другой картинкой или числовым атрибутом. В таком случае модель предсказывает $\epsilon_\theta(x_t, t, c)$ и приближает условное распределение $p_\theta(x_0 \mid c)$.

20.1 Classifier Guidance

При доступном классификаторе $p_\phi(c \mid x_t)$ его градиент по изображению может корректировать направление выборки [15]. Это даёт более сильное соответствие условию, но может менять разнообразие и устойчивость генерации.

20.2 Classifier-Free Guidance

Подход Ho и Salimans [16] строит условное и безусловное предсказание одной моделью, обучаемой с периодическим удалением условия. На этапе генерации применяют

$$\widehat{\epsilon}_\theta(x_t, t, c) = \epsilon_\theta(x_t, t, \varnothing) + w\big[\epsilon_\theta(x_t, t, c) - \epsilon_\theta(x_t, t, \varnothing)\big]. \tag{39}$$

При $w = 1$ получаем условное предсказание, при $w = 0$ — безусловное. Выбор $w > 1$ усиливает влияние условия, но слишком большое значение может приводить к перенасыщенности, артефактам и снижению разнообразия.

21. Латентная диффузия и генерация по тексту

Главное затруднение пиксельной диффузии — высокая стоимость обработки изображений большого разрешения. В Latent Diffusion Models [17] сначала обучают сжимающую модель, отображающую изображение x в компактный латентный код $z = E(x)$, после чего диффузионный процесс работает в пространстве z, а декодер восстанавливает изображение $\widehat{x} = D(z)$.

Это не то же самое, что просто уменьшить размер картинки: автоэнкодер обучается сохранять визуально значимую информацию. В текстоусловных моделях текст кодируется отдельным энкодером; модель генерации использует cross-attention для согласования пространственных признаков и текстовых представлений [20, 17].

Критерий	Пиксельная DDPM	Латентная диффузия
Пространств процесса	Пиксели изображения	Сжатые представления
Вход сети	Пиксельный тензор	Латентный тензор
Требование к декодеру	Обычно не нужен отдельный	Требуется для восстановления
Потенциальн компромисс	Высокая стоимость вычислений	Погрешность сжатия и качество декодера

22. Диффузия, непрерывное время и score-based models

Непрерывные стохастические дифференциальные уравнения дают более общую точку зрения. Например, дискретная диффузия может рассматриваться как приближение процесса с постепенно меняющейся интенсивностью шума. В score-based моделях сеть оценивает поле

$$s_\theta(x, t) \approx \nabla_x \log p_t(x), \tag{40}$$

называемое *score* распределения на уровне шума t. Для условного гауссовского $q(x_t \mid x_0)$ из (10) имеем

$$\nabla_{x_t} \log q(x_t \mid x_0) = -\frac{x_t - \sqrt{\bar{\alpha}_t} x_0}{1 - \bar{\alpha}_t} = -\frac{\epsilon}{\sqrt{1 - \bar{\alpha}_t}}. \tag{41}$$

Поэтому оценка шума тесно связана с оценкой score-функции. Непрерывная теория и методы решения обратных SDE/ODE развиваются в работе Song и соавторов [14]; здесь они приводятся как расширение базового курса, без отождествления всех непрерывных схем с конкретной дискретной DDPM.

23. Оценивание качества и диагностика результатов

23.1 Визуальный контроль

Для MNIST полезно фиксировать начальный шум и сохранять сетку из 16 или 64 изображений после каждой эпохи. Нужно проверять читаемость цифр, разнообразие классов, частоту одинаковых образцов и наличие артефактов. Но одного визуального контроля недостаточно.

23.2 FID и метрики

Fréchet Inception Distance (FID) сравнивает средние и ковариационные матрицы эмбеддингов реальных и синтетических изображений [19]:

$$\mathrm{FID} = \|m_r - m_g\|_2^2 + \mathrm{Tr}\big(C_r + C_g - 2(C_r^{1/2} C_g C_r^{1/2})^{1/2}\big). \tag{42}$$

Эта запись соответствует стандартной матричной форме при положительно полуопределённых ковариационных матрицах. Оценка зависит от выбранного извлекателя признаков, количества изображений, размеров входа и предобработки; её нельзя сравнивать между экспериментами с несовместимыми настройками.

23.3 Экспериментальная дисциплина

Нужно фиксировать разделение выборки, начальные состояния генераторов случайных чисел, версии зависимостей, тип GPU, количество обучающих шагов, шаги выборки, размер пакета и метод оценки. Желательно сообщать средние значения по нескольким запускам, а не только лучший результат.

24. Распространённые ошибки и их исправление

1. **Путаница индексов** t**.** Проверяйте соответствие математического $1, \dots, T$ программному $0, \dots, T-1$.
2. **Несоответствие диапазонов изображений.** Если обучение идет на $[-1, 1]$, не следует подавать сети изображения из $[0, 255]$ без нормализации.
3. **Отсутствие эмбеддинга времени.** Сеть плохо различает уровни шума.
4. **Неправильная форма коэффициентов.** Вектор $(B,)$ нужно преобразовать к $(B, 1, 1, 1)$.
5. **Добавление шума после последнего шага.** Последний переход должен использовать выбранное корректное правило декодирования.
6. **Несогласованность расписаний обучения и генерации.** Параметры β_t должны совпадать.
7. **Слишком мало обучающих итераций.** Убывающая MSE не гарантирует визуально убедительных образцов.
8. **Смешение метрик.** Низкая ошибка предсказания шума не равносильна лучшему FID.
9. **Оценка на тренировочных данных.** Для количественных сравнений нужен корректный отдельный набор реальных изображений.
10. **Отсутствие контроля лицензий и данных.** Учитывайте право использования обучающих данных и риск генерации недостоверного контента.

25. Практикум: сравнение расписаний и числа шагов

Постановка: обучите компактную модель на MNIST с линейным расписанием и сохраните генерации после фиксированного числа оптимизационных шагов. Затем измените лишь число шагов генерации, если используете совместимый ускоренный сэмплер, и сравните время и визуальное качество. Не сравнивайте результаты различных моделей, обученных разное количество итераций, как будто это эффект только метода генерации.

Требуемые результаты:

1. графики β_t, $\bar{\alpha}_t$, $\mathrm{SNR}(t)$;
2. несколько зашумленных вариантов одной цифры при разных t;
3. график средней MSE обучения по эпохам;
4. сетка генераций с зафиксированным случайным началом;
5. таблица с архитектурой, временем обучения, числом шагов сэмплинга;
6. обсуждение наблюдаемых артефактов и ограничений экспериментальной оценки.

26. Расчётные задачи с решениями

Задача 1. Накопленное сохранение сигнала

Пусть $\beta_1 = 0.1$, $\beta_2 = 0.2$, $\beta_3 = 0.25$. Найдите $\bar{\alpha}_3$ и дисперсию $q(x_3 \mid x_0)$.

Решение. $\bar{\alpha}_3 = (0.9)(0.8)(0.75) = 0.54$, поэтому ковариация равна $0.46 I_d$; среднее — $\sqrt{0.54}x_0$.

Задача 2. Апостериорная дисперсия

При $\beta_2 = 0.2$ и $\bar{\alpha}_1 = 0.9$ получаем $\bar{\alpha}_2 = 0.72$ и

$$\widetilde{\beta}_2 = \frac{1 - 0.9}{1 - 0.72} \cdot 0.2 = \frac{0.02}{0.28} = \frac{1}{14} \approx 0.07143. \tag{43}$$

Задача 3. Восстановление x_0 по известному шуму

Пусть $x_t = 1.1$, $\bar{\alpha}_t = 0.64$, $\epsilon = 0.5$. Тогда $x_0 = (1.1 - 0.6 \cdot 0.5)/0.8 = 1.0$. Проверьте результат подстановкой в (11).

Задача 4. Ошибка предсказания шума

Если $\epsilon = (1, -1)$ и $\epsilon_\theta = (0.7, -0.8)$, квадратичная сумма ошибок равна $(0.3)^2 + (-0.2)^2 = 0.13$, а покомпонентная средняя квадратичная ошибка равна 0.065.

Задача 5. Количество вызовов модели

Стандартный сэмплер на $T = 1000$ шагов и альтернативный совместимый сэмплер на 50 шагах требуют соответственно 1000 и 50 вызовов шумопредсказателя на один пакет, если каждый шаг использует один вызов. Отношение составляет 20, но реальное ускорение по времени зависит от накладных расходов и вычислительной платформы.

27. Контрольные вопросы

1. Что означает генеративное моделирование данных?
2. Как отличается принцип генерации GAN, VAE и DDPM?
3. Какие преимущества даёт предсказание заранее известного шума?
4. Что такое прямой диффузионный процесс?
5. Почему цепь $x_0, \dots, x_T$ является марковской?
6. Как определить α_t и $\bar{\alpha}_t$?
7. Почему ковариация прямого шага равна $\beta_t I$?
8. Докажите формулу $q(x_t \mid x_0)$.
9. Какова роль формулы репараметризации?
10. Как меняется SNR при уменьшении $\bar{\alpha}_t$?
11. Чем линейное расписание отличается от косинусного?
12. Почему при конечном T x_T не всегда строго нормален?
13. Что представляет собой обратная диффузия?
14. Зачем аппроксимировать условное распределение гауссовским?
15. Почему рассматривают $q(x_{t-1} \mid x_t, x_0)$?
16. Выведите формулу апостериорной дисперсии.
17. Выведите формулу апостериорного среднего.
18. Как выразить x_0 через x_t и шум?
19. Почему обратное среднее можно параметризовать через ϵ_θ?
20. Что такое ELBO?
21. На каком шаге вывода используется неравенство Йенсена?
22. На какие члены раскладывается вариационная цель DDPM?
23. Как KL двух гауссианов связан с MSE?
24. Почему L_{simple} не тождествен полной ELBO?
25. Как обучить модель на случайно выбранном шаге времени?
26. Почему необходимо передавать время t на вход модели?
27. Чем U-Net отличается от простой CNN?
28. Какую роль выполняют временные эмбеддинги?
29. Сколько проходов сети требует стандартный DDPM-сэмплер?
30. Почему на последнем шаге не добавляется обычный гауссовский шум?
31. Чем параметризация x_0 отличается от ϵ-параметризации?
32. Как определяется v-параметризация?
33. В чём идея DDIM?
34. Что означает детерминированная обратная выборка?
35. Что такое условная диффузионная модель?
36. Как устроен Classifier-Free Guidance?

37. Почему слишком большой guidance scale ухудшает разнообразие?
38. Что такое латентная диффузия?
39. Как применяется Cross-Attention в текстовой генерации изображений?
40. Что такое score-функция распределения?
41. Как связаны score и предсказание шума?
42. Что измеряет FID и какие у него ограничения?
43. Почему важно фиксировать генератор случайных чисел?
44. Какие ошибки возникают при неправильном broadcasting коэффициентов?
45. Как организовать корректное сравнение двух диффузионных моделей?

28. Задания для самостоятельной работы

1. Докажите формулу (10) для произвольного t по индукции.
2. Выведите (19) методом выделения полного квадрата.
3. Проверьте алгебраически переход от (20) к (24).
4. Реализуйте функции линейного и косинусного расписания на NumPy; постройте графики.
5. Изобразите одну и ту же цифру MNIST при десяти значениях t.
6. Реализуйте предсказание шума для синтетических 2D-точек небольшой сетью MLP.
7. Добавьте в NoiseCNN синусоидальное кодирование времени и сравните ошибки.
8. Постройте U-Net для изображений 28×28; проверьте размерности промежуточных тензоров.
9. Сравните два значения learning rate при фиксированном seed и одинаковых эпохах.
10. Реализуйте сохранение контрольных генераций после каждой эпохи.
11. Измените число шагов выборки с помощью корректного DDIM-сэмплера.
12. Сравните предсказание ϵ и x_0 на одной архитектуре.
13. Объясните отличие условного и безусловного генератора на задаче MNIST.
14. Исследуйте влияние веса classifier-free guidance на качество и разнообразие.
15. Подготовьте отчёт с архитектурой, гиперпараметрами, результатами и анализом ошибок.

29. Итоги лекции

Главная математическая идея. Прямой процесс задаёт аналитически вычислимые условные нормальные распределения. Нейронная сеть учится предсказывать шум, а обратный процесс использует это предсказание для постепенного перехода от нормального шума к новым наблюдениям. DDPM опирается на вариационный принцип, но популярная упрощенная MSE-потеря является специальным практическим выбором. Архитектура сети, расписание шума и сэмплер совместно определяют качество и стоимость генерации.

Библиография

Список литературы

[1] Trask A. *Grokking Deep Learning*. Manning, 2019.

[2] Rashid T. *Make Your Own Neural Network*. CreateSpace, 2016.

[3] Géron A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd ed., O'Reilly, 2022.

[4] Chollet F. *Deep Learning with Python*. 2nd ed., Manning, 2021.

[5] Stevens E., Antiga L., Viehmann T. *Deep Learning with PyTorch*. Manning, 2020.

[6] Goodfellow I., Bengio Y., Courville A. *Deep Learning*. MIT Press, 2016. https://www.deeplearningbook.org/

[7] Nielsen M. *Neural Networks and Deep Learning*. Determination Press, 2015. http://neuralnetworksanddeeplearning.com/

[8] Goodfellow I. et al. Generative Adversarial Nets. *NeurIPS*, 2014. https://arxiv.org/abs/1406.2661

[9] Kingma D. P., Welling M. Auto-Encoding Variational Bayes. *ICLR*, 2014. https://arxiv.org/abs/1312.6114

[10] Sohl-Dickstein J. et al. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. *ICML*, 2015. https://arxiv.org/abs/1503.03585

[11] Ho J., Jain A., Abbeel P. Denoising Diffusion Probabilistic Models. *NeurIPS*, 2020. https://arxiv.org/abs/2006.11239

[12] Nichol A. Q., Dhariwal P. Improved Denoising Diffusion Probabilistic Models. *ICML*, 2021. https://arxiv.org/abs/2102.09672

[13] Song J., Meng C., Ermon S. Denoising Diffusion Implicit Models. *ICLR*, 2021. https://arxiv.org/abs/2010.02502

[14] Song Y. et al. Score-Based Generative Modeling through Stochastic Differential Equations. *ICLR*, 2021. https://arxiv.org/abs/2011.13456

[15] Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis. *NeurIPS*, 2021. https://arxiv.org/abs/2105.05233

[16] Ho J., Salimans T. Classifier-Free Diffusion Guidance. 2022. https://arxiv.org/abs/2207.12598

[17] Rombach R. et al. High-Resolution Image Synthesis with Latent Diffusion Models. *CVPR*, 2022. https://arxiv.org/abs/2112.10752

[18] Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *MICCAI*, 2015. https://arxiv.org/abs/1505.04597

[19] Heusel M. et al. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *NeurIPS*, 2017. https://arxiv.org/abs/1706.08500

[20] Vaswani A. et al. Attention Is All You Need. *NeurIPS*, 2017. https://arxiv.org/abs/1706.03762