Data science

Лекция 5. Глубокое обучение и нейросети

2025/2026 учебный год

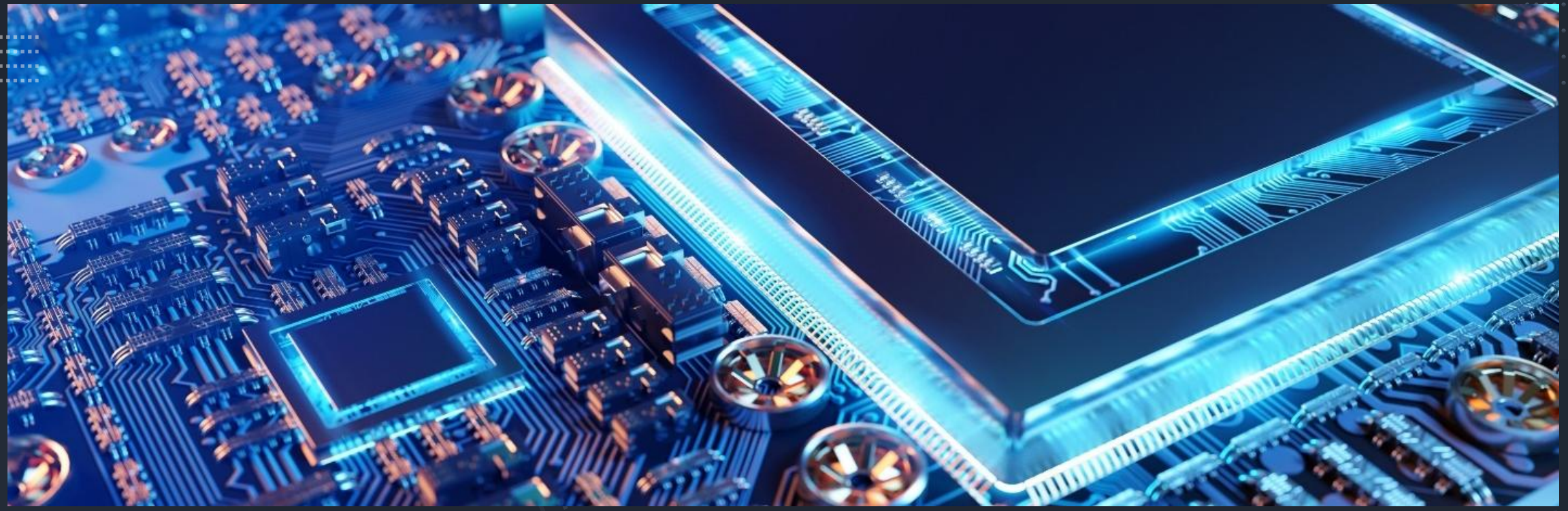
Доцент кафедры MO&MO, Махно В.В.

ИНСТИТУТ
МАТЕМАТИКИ
МЕХАНИКИ
КОМПЬЮТЕРНЫХ
НАУК
имени И.И. Воровича

Нейронные сети в задачах машинного обучения

Качество работы нейронной сети в классификации сопоставимо с качеством бустинга или даже немного уступает, при этом время настройки нейронной сети может быть больше. Поэтому для табличных данных чаще используют бустинг.

А вот классификация изображений (например, что за цветок на фотографии, доволен ли клиент обслуживанием) или текстов (например, категоризация документов) — это уже задача для нейронных сетей

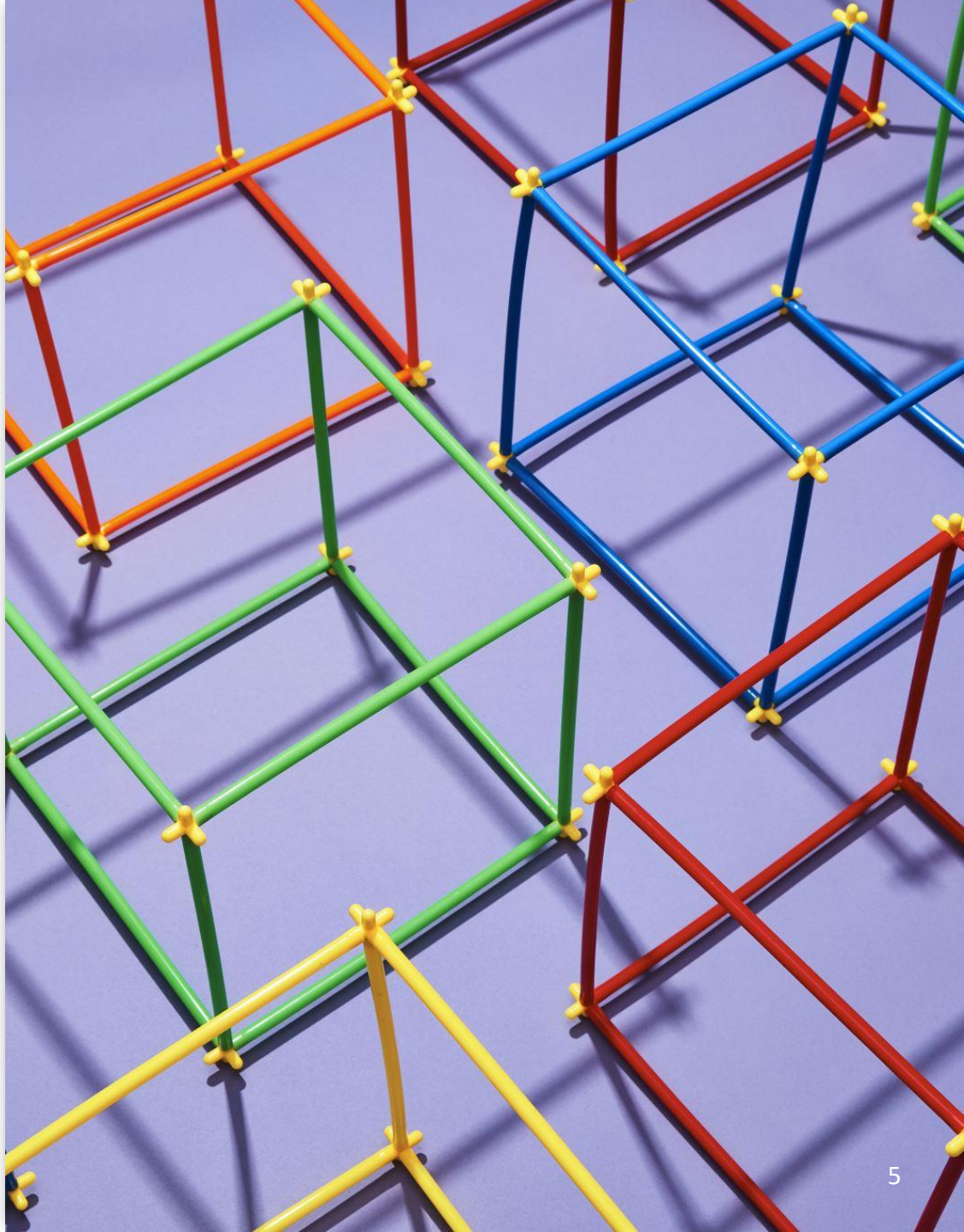


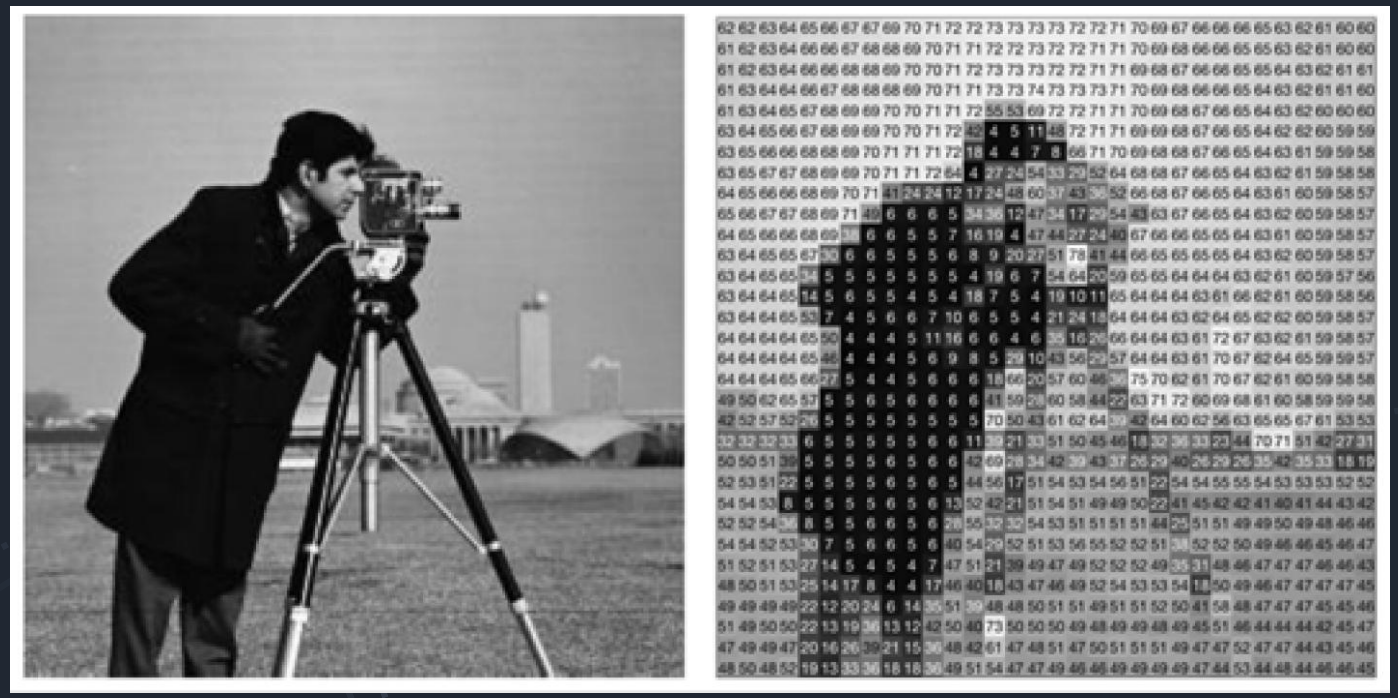
Обучение нейронных сетей вычислительно затратно: для этого используются специальные графические ускорители (Graphics Processing Unit, GPU). Кроме этого, для успешного обучения нейросети необходимо собрать много данных — без этого не получится обучить миллионы параметров нейросети. Появление мощной вычислительной аппаратуры, сбор большого количества данных и разработка особых приемов обучения нейросетей — эти три фактора обусловили прорыв глубинного обучения, произошедший в начале 2010-х годов.

Популярные архитектуры нейронных сетей

Сверточные и рекуррентные нейросети

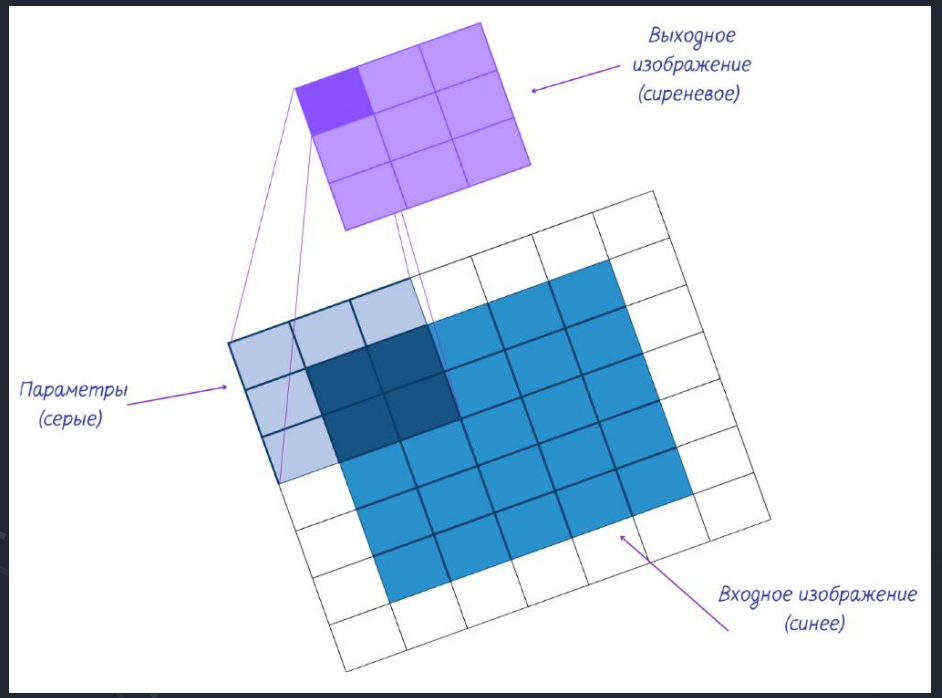
Основная особенность такой нейросети в том, что она не просто преобразует информацию от слоя к слою, а имеет “память”. Каждый нейрон “помнит”, какая информация проходила через него ранее. В итоге эти сети лучше прослеживают связи между блоками информации, даже при небольшом размере самой сети. Поэтому рекуррентные сети лучше других анализируют последовательную информацию – например, человеческую речь или музыку.





Сверточные нейронные сети

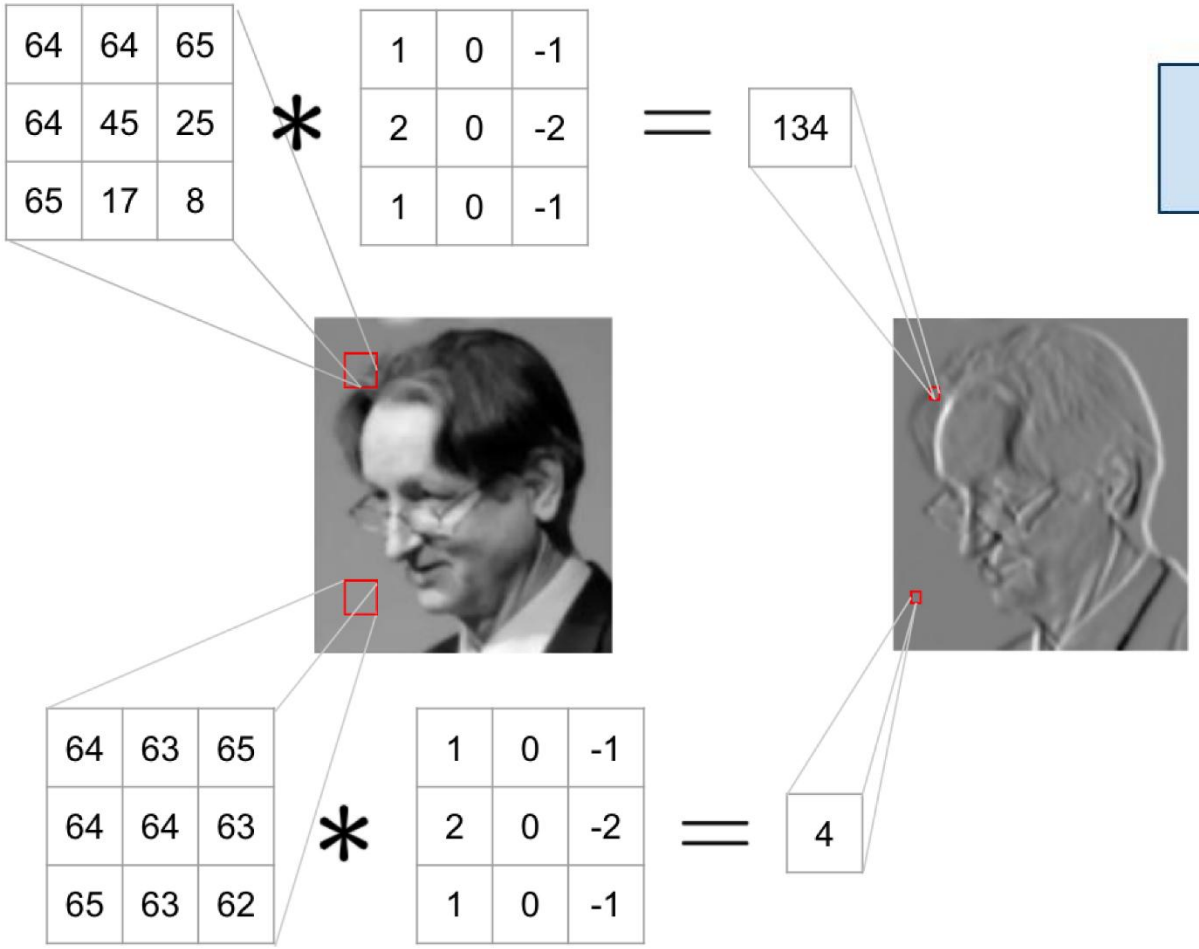
Нейронные сети для анализа изображений

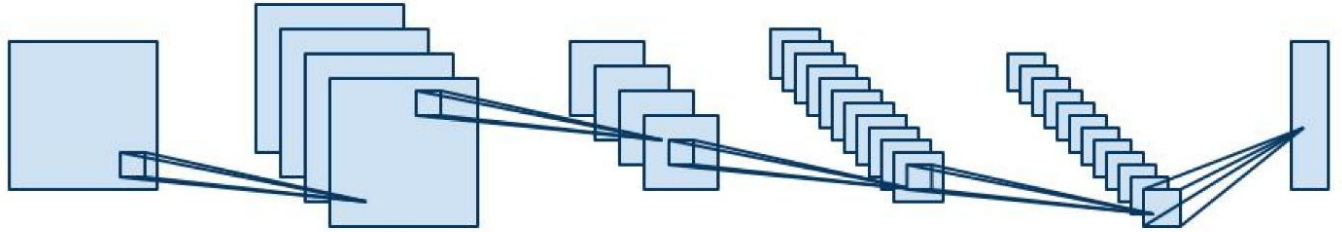


Нейронные сети для анализа изображений

Свертка — это процедура агрегации информации о пикселе и соседних с ним пикселях.

Пример работы фильтра Собеля





MIT: Alexander Amini, 2018 introtodeeplearning.com

Задачи анализа изображений

Задача классификации изображений

Самая популярная задача анализа изображений — задача классификации. В этой задаче на вход нейронной сети подается изображение, а выход — это метка класса. В качестве примеров можно привести:

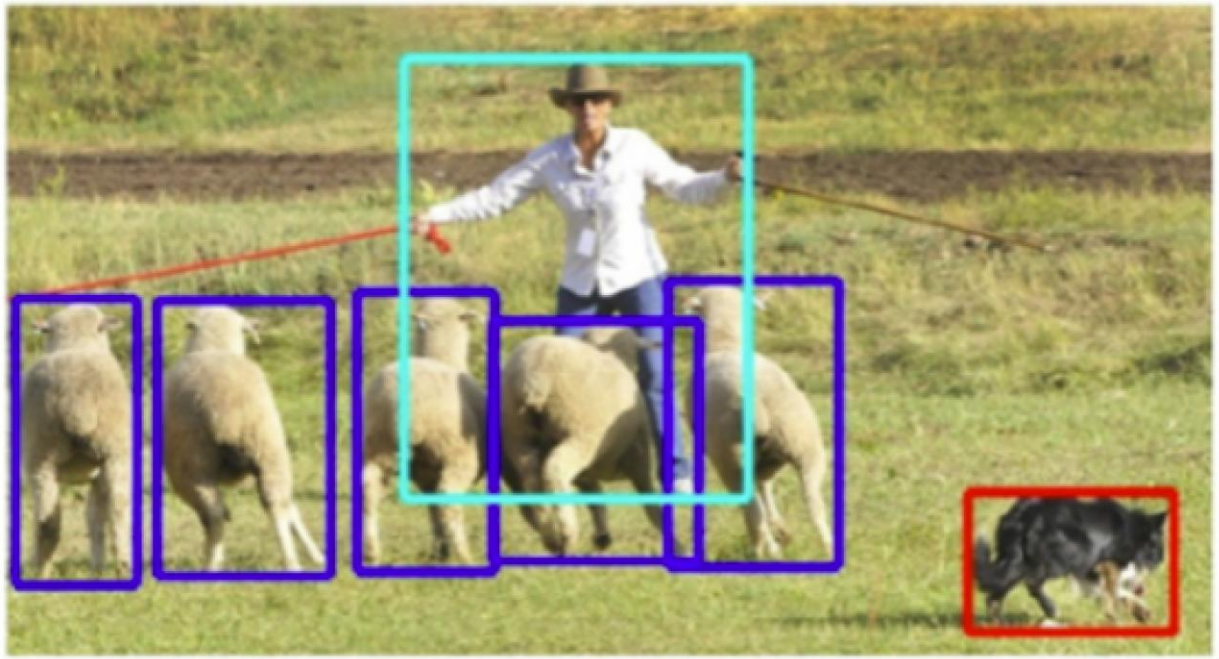
- идентификацию владельца смартфона по фотографии (классы: владелец или не владелец);
- распознавание эмоций клиента при посещении офиса (классы: доволен, возмущен, нейтрален и т. д.);
- распознавание дорожных знаков в беспилотных автомобилях (классы: знаки).

Классификация изображения

Задачи анализа изображений

Задача детекции изображений

Однако классификация — это несколько упрощенная задача: она лишь определяет, какие классы есть на изображении, но не определяет их расположение. Часто также решают задачу детекции: в ней нейронная сеть должна выделить прямоугольниками все объекты, находящиеся на изображении. Как и классификация, детекция — это задача обучения с учителем: для обучения нейронной сети потребуется набор изображений с объектами, выделенными прямоугольниками. Выполнять такую разметку может быть утомительно, часто для этого используют краудсорсинговые сервисы, например Яндекс.Толока.

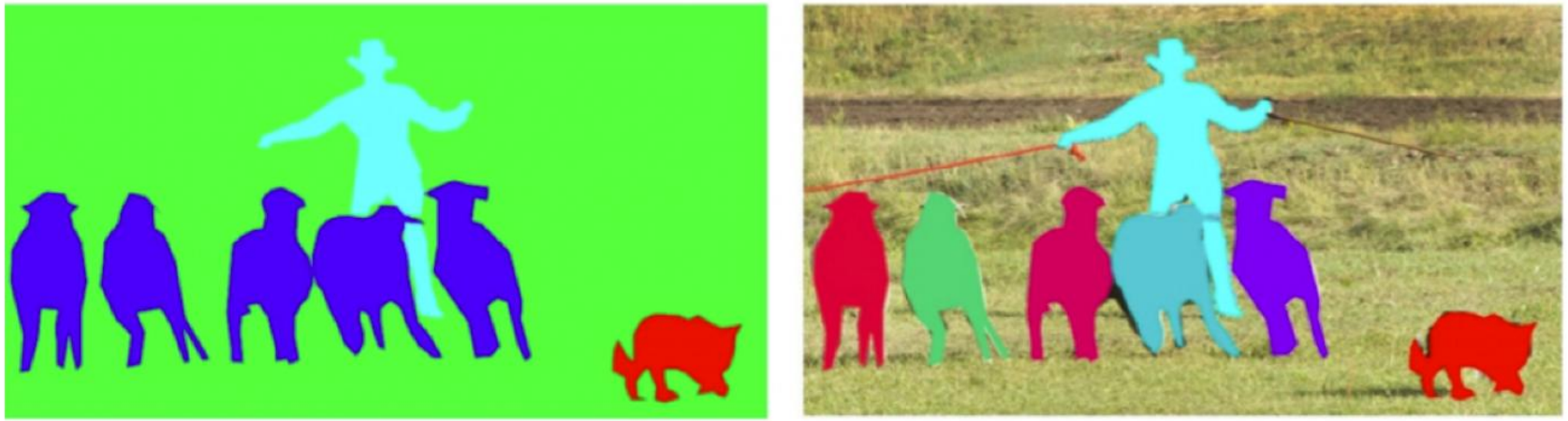


Детекция

Задачи анализа изображений

Задача сегментации изображений

Если нужно проанализировать изображение еще более детально, решают задачу сегментации. В этой задаче нейронная сеть должна для каждого пикселя определить, к какой области он относится (области — это фон и объекты, находящиеся на изображении). Собрать размеченные данные в задаче сегментации еще сложнее, чем в задаче детекции. Сегментация активно применяется в обработке изображений и для составления коллажей: например, можно «вырезать» человека (выделив все пиксели, отнесенные к области «человек») и вставить на другой фон.



Семантическая сегментация

Задача генерации изображений

В 2016 году высокую популярность приобрели приложения, стилизующие изображения под известные картины, например Призма. Научно данная задача называется переносом стиля.

Перенос стиля работает так: на вход нейросети подаются два изображения — то, которое нужно перерисовать, и то, с которого нужно скопировать стиль, а на выход выдается стилизованное изображение.

Задача генерации изображений

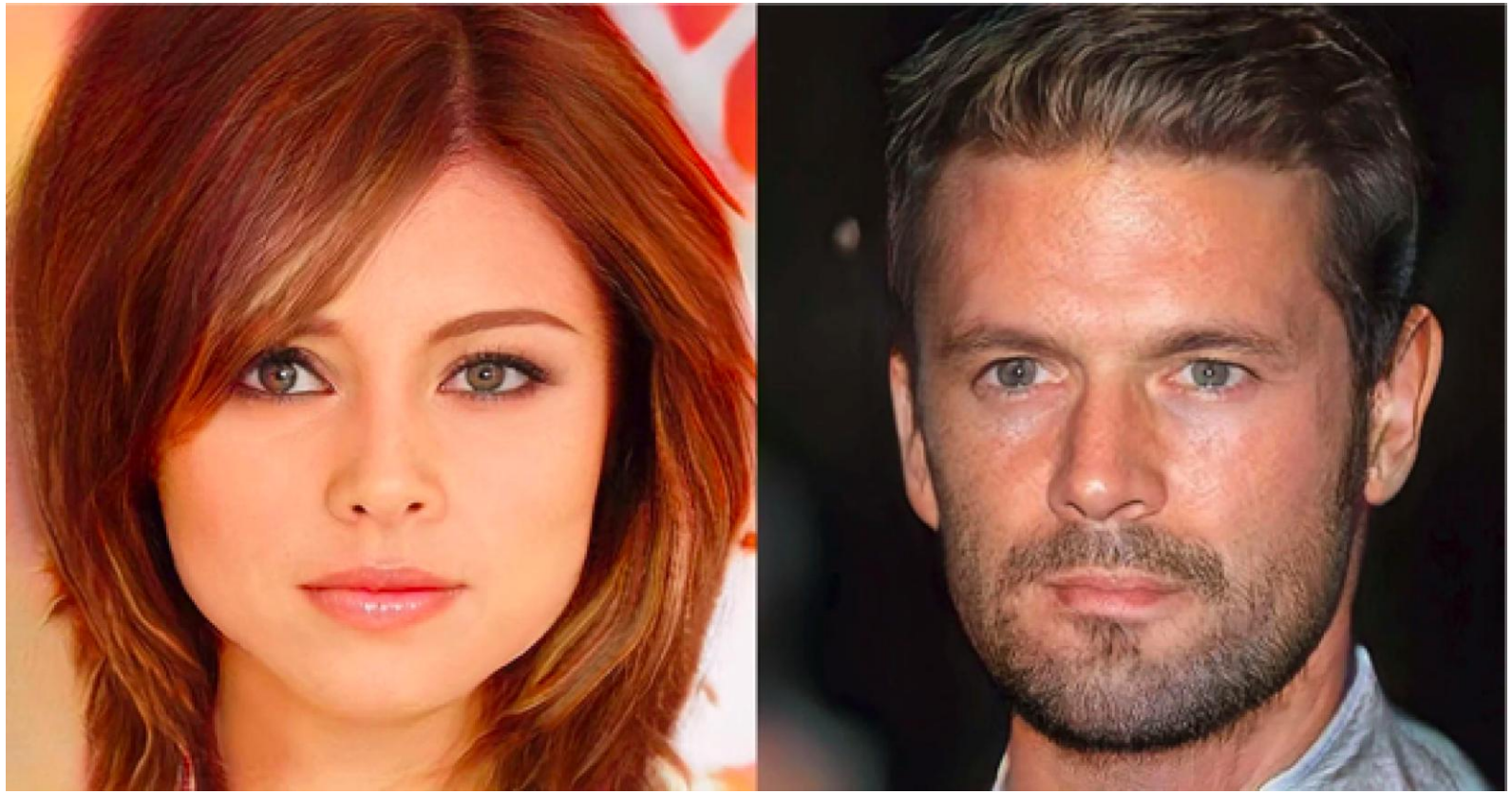
С помощью переноса стиля можно изменить время дня на фотографии.

Чтобы генерировать фотореалистичные изображения, используют так называемые генеративно-состязательные сети (generative adversarial networks, GAN).

GAN состоит из двух нейросетей: первая генерирует изображение, а вторая решает задачу классификации и учится определять, на входе настоящее оно или сгенерированное. Задача первой нейросети — "обмануть" вторую, и она "вынуждена" генерировать реалистичные кадры.

Задача генерации изображений

Примеры лиц знаменитостей, сгенерированных нейросетями.



Задача генерации изображений

В начале 2021 года компания OpenAI представила нейронную сеть DALL·E, которая генерирует изображения по текстовому описанию. Подробнее по ссылке. Например, DALL·E может создавать новые предметы интерьера или даже целые интерьеры. Ниже - примеры, сгенерированные нейросетью по запросу «кресло в форме авокадо»:



Полезные ссылки

- ***"Николай Иронов"***

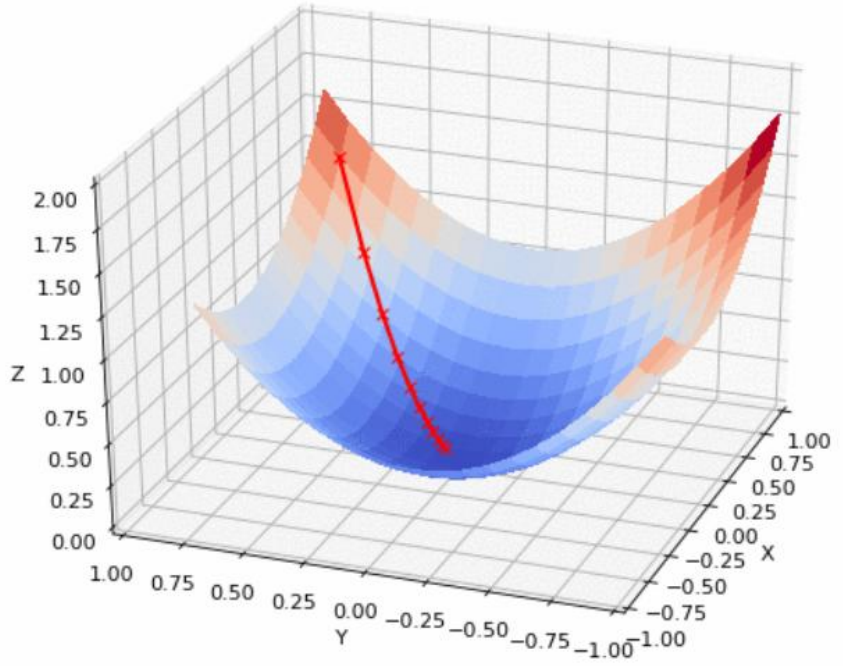
А студия Артемия Лебедева, например, целый год разрабатывала логотипы для реальных заказчиков с помощью искусственного интеллекта Николая Иронова. Клиенты и коллеги «Николая» из студии Лебедева не знали, что работы создаются генеративной моделью. За год модель выполнила более 20 коммерческих проектов.

При этом в Николая заложено много экспертных знаний: о том, как строить композицию, какие цвета комплементарные, как можно трансформировать шрифты. Подробнее об устройстве «искусственного дизайнерского интеллекта» можно почитать по ссылке. (Интервью с создателями нейросети Николай Иронов: как он устроен, в чем вообще фишка и почему все зовут его Колей / Хабр (habr.com))

- Множество интернет-сервисов позволяют в интерактивном режиме познакомиться с нейросетями, генерирующими или анализирующими изображение. К примеру, в англоязычном сервисе https://quickdraw.withgoogle.com/ пользователю предлагается нарисовать рисунок, а нейросеть попытается угадать, что нарисовано. Сервис https://deepart.io/ стилизует изображения пользователя под известные картины. А сервис https://affinelayer.com/pixsrv/ превращает набросок в фотореалистичное изображение.

Обучение нейронной сети

Градиентный спуск (Gradient Descent) — это **итеративный алгоритм оптимизации**, с помощью которого можно **минимизировать функцию потерь**. Это фундамент почти всех алгоритмов обучения: от простой линей регрессии до глубоких нейросетей.



Простыми словами: спуск с холма

- С вершины горы (график ошибки) в тумане добраться в самую низкую точку — минимум функции потерь.
- Карта не видна, но можно ощущать наклон.
- Шаг за шагом в сторону уменьшения наклона — вниз.
- И так постепенно приближаемся к минимуму.
- Это и есть градиентный спуск. На каждом шаге идём в сторону, где ошибка убывает быстрее всего.

Градиентный спуск

Пусть есть функция потерь:

L(w)=ошибка модели от параметров w$L(w)$=ошибка модели от параметров w

Задача — найти такое w, при котором L(w) минимальна.

Обновление параметров с помощью градиентного спуска:

$$w_{\text{new}} = w_{\text{old}} - \eta \cdot \frac{\partial L}{\partial w}$$

где:

- w_{old} – текущее значение параметра
- w_{new} – новое значение после шага
- η– скорость обучения (learning rate): **насколько большой шаг мы делаем**
- $\frac{\partial L}{\partial w}$ – **градиент функции потерь по параметру** `w` (направление наибольшего роста функции потерь)

Градиентный спуск в линейной регрессии

Задача:

Обучить модель линейной регрессии:

$$\hat{y} = w \cdot x + b$$

Найти такие значения `w` и `b`, при которых **ошибка между предсказаниями и настоящими значениями минимальна**.

1) Целевая функция (ошибка) - среднеквадратичная ошибка (MSE) как функцию потерь:

$$L(w, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - (w \cdot x_i + b))^2$$

2) Градиенты — как понять, в какую сторону менять w и b

Чтобы минимизировать функцию потерь, считаем её производные (градиенты):

$$\frac{\partial L}{\partial w} = -\frac{2}{n} \sum x_i \cdot (y_i - \hat{y}_i)$$

$$\frac{\partial L}{\partial b} = -\frac{2}{n} \sum (y_i - \hat{y}_i)$$

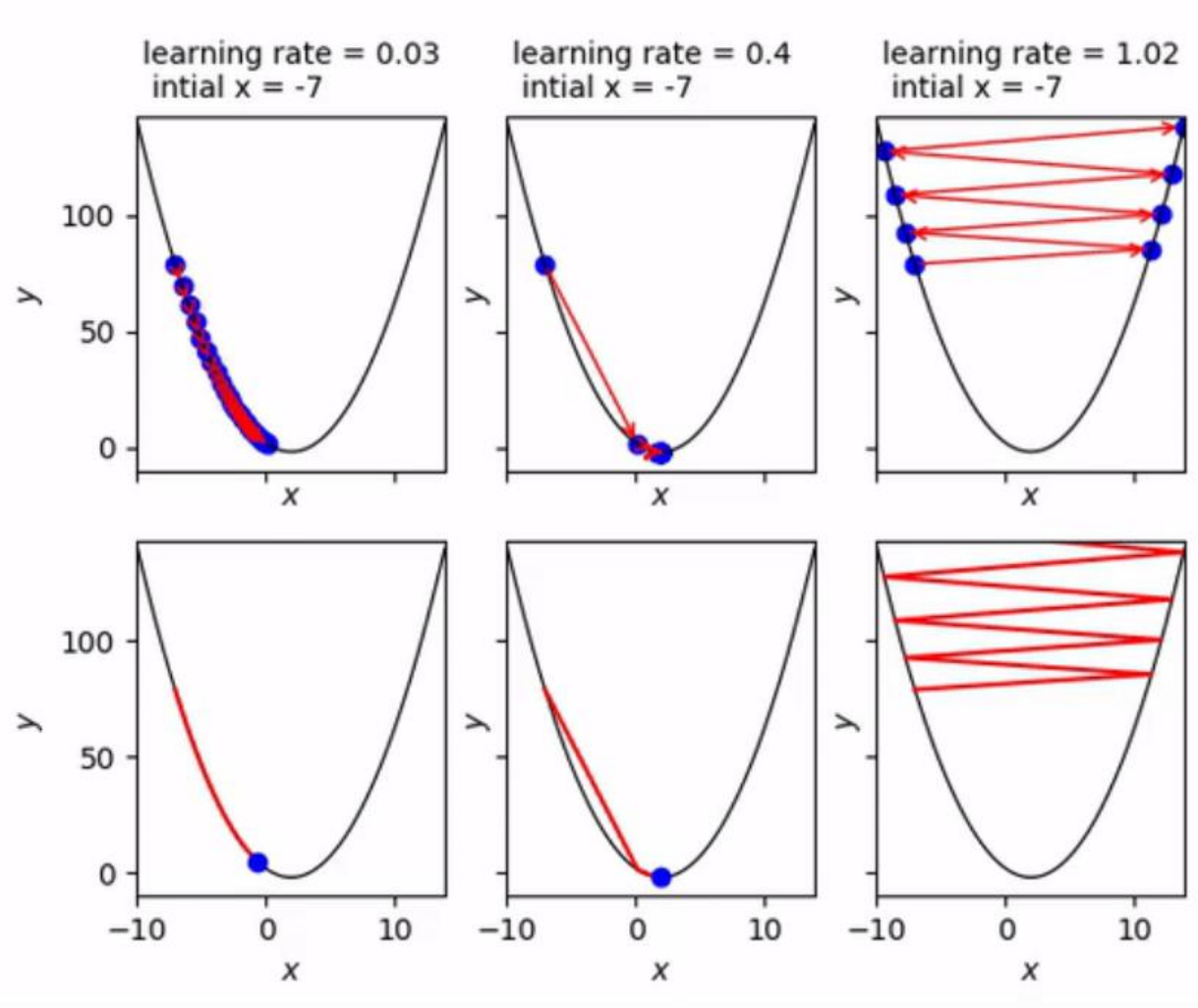
Градиентный спуск в линейной регрессии

3) Обновление параметров

$$w_{\text{new}} = w_{\text{old}} - \eta \cdot \frac{\partial L}{\partial w}$$

$$b_{\text{new}} = b_{\text{old}} - \eta \cdot \frac{\partial L}{\partial b}$$

Скорость обучения (learning rate)
Слишком большой η: перескакиваем минимум, модель не обучается (или «взорвётся»).
Слишком маленький η: обучение идёт очень медленно.
Иногда η адаптируется во времени или по направлению



Градиентный спуск на python

```
import numpy as np

# Данные
X = np.array([1, 2, 3, 4, 5], dtype=float)
y = np.array([3, 5, 7, 9, 11], dtype=float)  # истинная зависимость: y = 2x + 1

# Начальные параметры
w = 0.0
b = 0.0
lr = 0.01  # скорость обучения

# Обучение
for i in range(1000):
    y_pred = w * X + b
    error = y - y_pred

    dw = -2 * np.mean(X * error)
    db = -2 * np.mean(error)

    w -= lr * dw
    b -= lr * db

# Результат
print(f"w = {w:.2f}, b = {b:.2f}")  # Должно быть близко к 2 и 1
```

Виды градиентного спуска. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD)

⚡ **Один пример → один шаг**

📘 Формула:

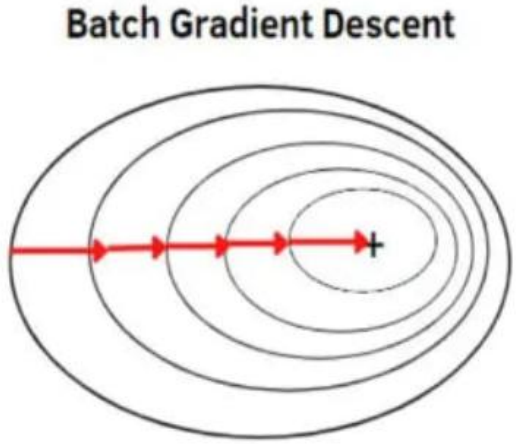
$$w_{new} = w_{old} - \eta \cdot \nabla \ell(x_i, y_i)$$

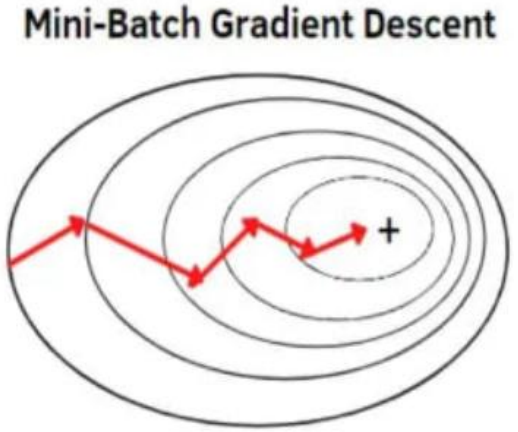
✅ Плюсы:

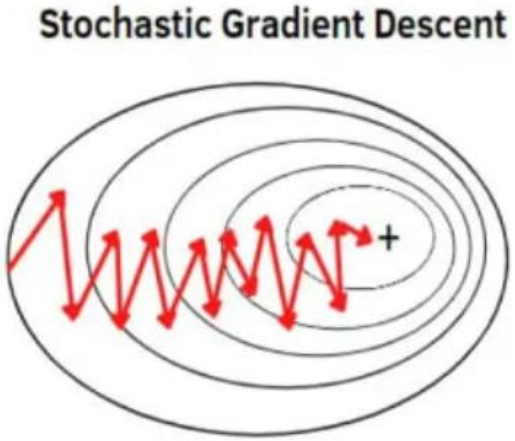
- Быстро
- Может "перепрыгнуть" через локальные минимумы

❌ Минусы:

- Шумное поведение
- Может не стабилизироваться







Модель смотрит на один пример, вычисляет ошибку и сразу обновляет веса.

Виды градиентного спуска. Batch Gradient Descent

📦 **Вся выборка → один шаг**

📘 Формула обновления:

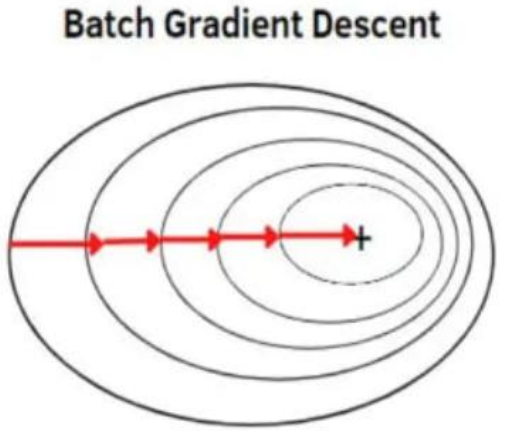
$$w_{new} = w_{old} - \eta \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(x_i, y_i)$$

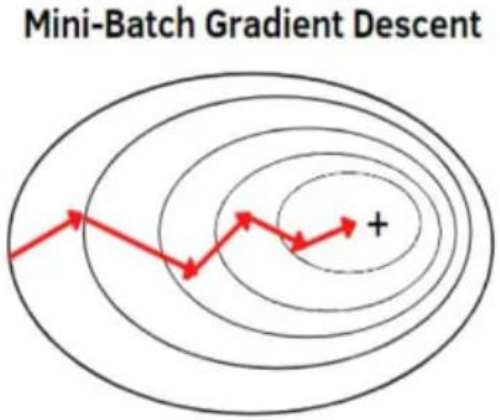
✅ Плюсы:

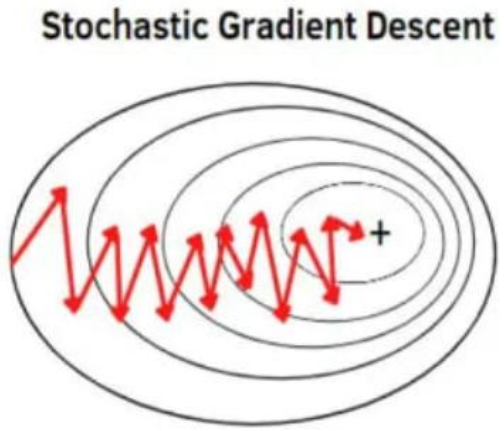
- Плавная и стабильная сходимость
- Хорошо для теоретических моделей

❌ Минусы:

- Медленно при больших данных
- Требует всю выборку в памяти







Модель смотрит на весь датасет целиком, вычисляет суммарную ошибку и только один раз обновляет веса за эпоху.

Виды градиентного спуска. Mini-Batch Gradient Descent

⚖️ **Небольшая подвыборка (обычно 32–128)**

📘 Формула:

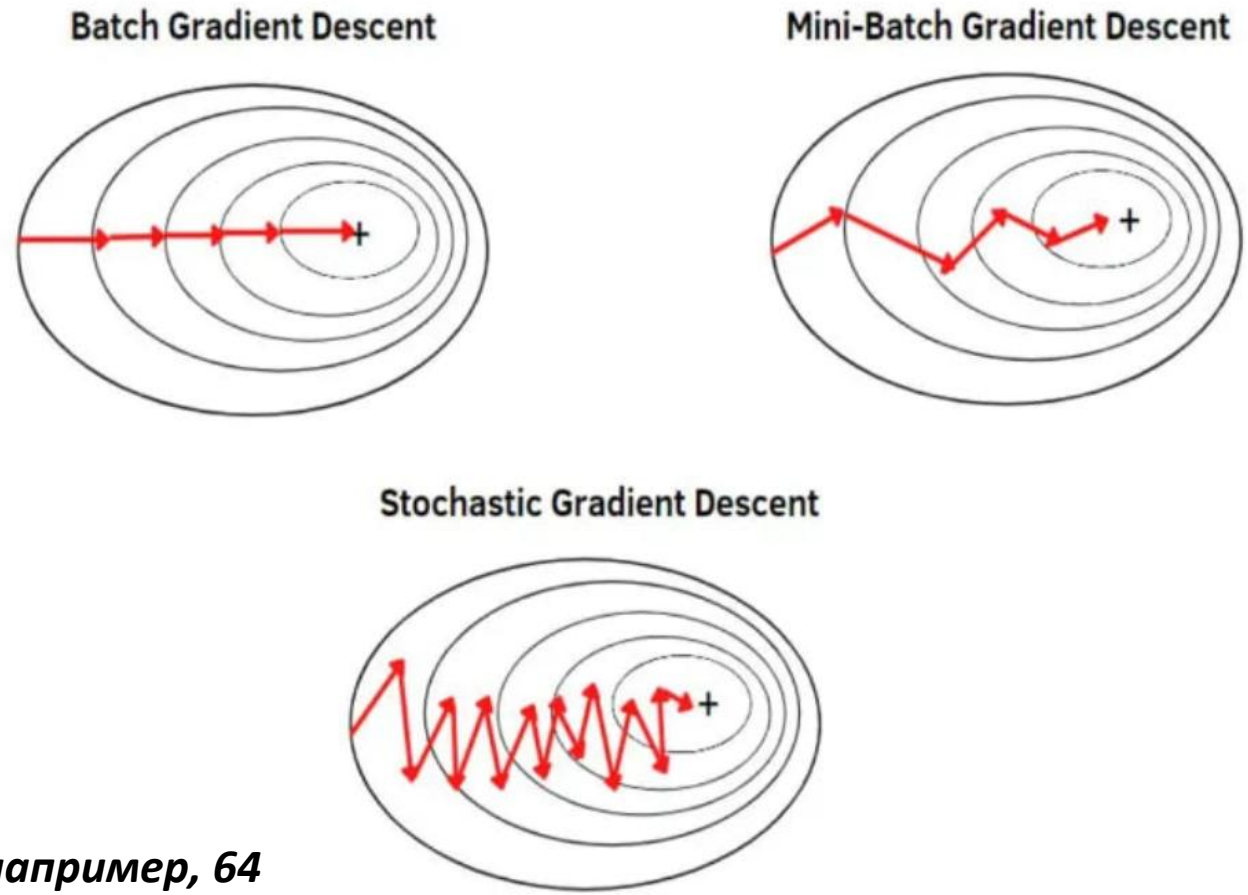
$$w_{new} = w_{old} - \eta \cdot \frac{1}{m} \sum_{j=1}^{m} \nabla \ell(x_j, y_j)$$

✅ Плюсы:

- Баланс между скоростью и стабильностью
- GPU-дружественен

❌ Минусы:

- Нужно настраивать размер батча (`batch_size`)



Модель смотрит на группу примеров (например, 64 картинки), усредняет их ошибку и только потом обновляет веса.