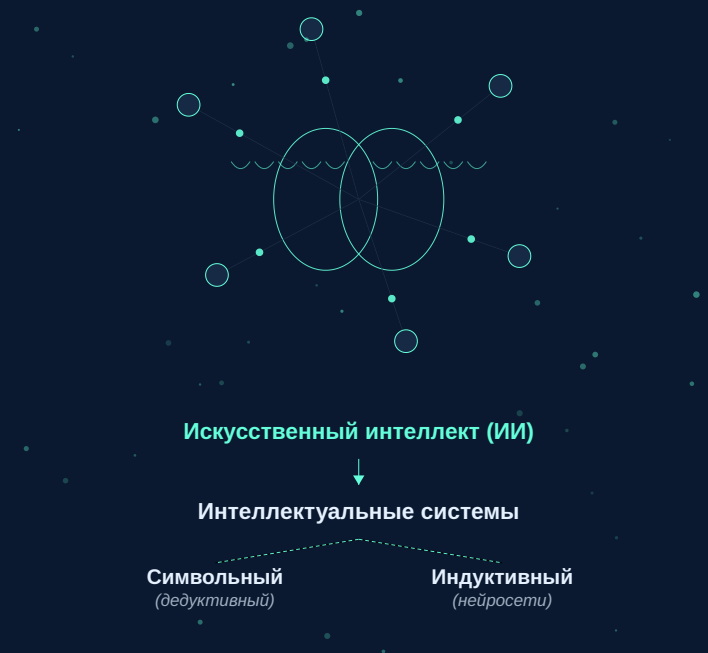


Интеллектуальные системы и искусственный интеллект (1)

Искусственный интеллект (ИИ) — свойство интеллектуальных систем решать задачи, требующие для своего решения человеческого интеллекта (обучение, рассуждение, восприятие, принятие решений) в условиях неопределенности.

Интеллектуальная система — программно-аппаратный комплекс, реализующий функции интеллектуального поведения (анализ, синтез, обобщение) при решении задач в условиях неполноты или противоречивости исходной информации.

Интеллектуальные системы и искусственный интеллект



Академические основы и архитектурная эволюция

План лекции: ключевые направления ИИ (2)

- Компьютерное зрение: детекция, сегментация, генерация изображений
- Обработка естественного языка: трансформеры, языковые модели
- Обучение с подкреплением: робототехника, оптимизация решений
- Генеративные модели: GAN, диффузия, автоэнкодеры
- Эволюция архитектур: от RNN до Mamba
- Будущее ИИ: эффективность, интерпретируемость, этика



Компьютерное зрение
детекция, сегментация,
генерация изображений



Обработка языка
трансформеры,
языковые модели



Обучение с подкреплением
робототехника,
оптимизация решений

Генеративные модели
GAN, диффузия,
автоэнкодеры



Эволюция архитектур
от RNN до Mamba,
линейная сложность



Будущее ИИ
эффективность,
интерпретируемость, этика



Шесть фундаментальных направлений современного ИИ.

Основные определения (3)

Искусственный интеллект (ИИ) — свойство интеллектуальных систем решать задачи, требующие человеческого интеллекта (обучение, рассуждение, восприятие) в условиях неопределенности.

Интеллектуальная система — программно-аппаратный комплекс, реализующий функции интеллектуального поведения при решении задач в условиях неполноты или противоречивости информации.

Иерархия ИИ

Искусственный интеллект (ИИ)



Интеллектуальные системы



Символьные системы

(дедуктивный подход)

- Экспертные системы
- Логическое программирование



Машинное обучение

(индуктивный подход)

- Нейросети
- Алгоритмы обучения



Гибридные системы

(комбинированный подход)

- Нейро-символьные
- Мультимодальные

Каждый уровень — подмножество вышестоящего

Классификация систем по степени интеллектуальности (4)

Тип системы	Основа функционирования	Адаптация	Примеры
Автоматическая	Жёсткие алгоритмы	Отсутствует	Термостат, конвейер
Автоматизированная	Человек + инструменты	Человеком	CAD-системы
Интеллектуализированная	Эвристики + правила	Ограниченная	Системы поддержки решений
Интеллектуальная	Моделирование интеллекта	Автономная	Нейросети, агенты с обучением

Ключевой критерий: способность к автономному обучению и адаптации без вмешательства человека.



Ключевой критерий интеллектуальности:
способность к автономному обучению и адаптации
без вмешательства человека

Статические и динамические интеллектуальные системы (5)

Статические системы (интеллектуализированные)

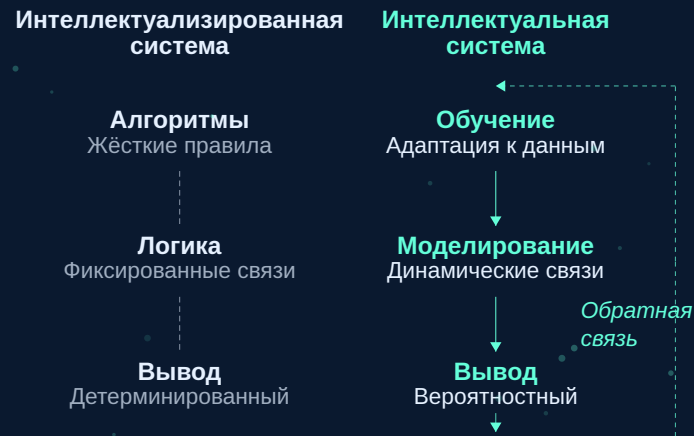
- Однократная обработка входных данных
- Фиксированный алгоритм преобразования
- Отсутствие памяти и обратной связи
- Пример: экспертный вывод, компилятор

Динамические системы (интеллектуальные)

- Непрерывное взаимодействие со средой
- Адаптация на основе обратной связи
- Наличие внутреннего состояния и памяти
- Пример: агенты с обучением, рекуррентные сети

Фундаментальное различие: наличие механизма обратной связи и способность к изменению состояния на основе опыта.

Два уровня «интеллекта»

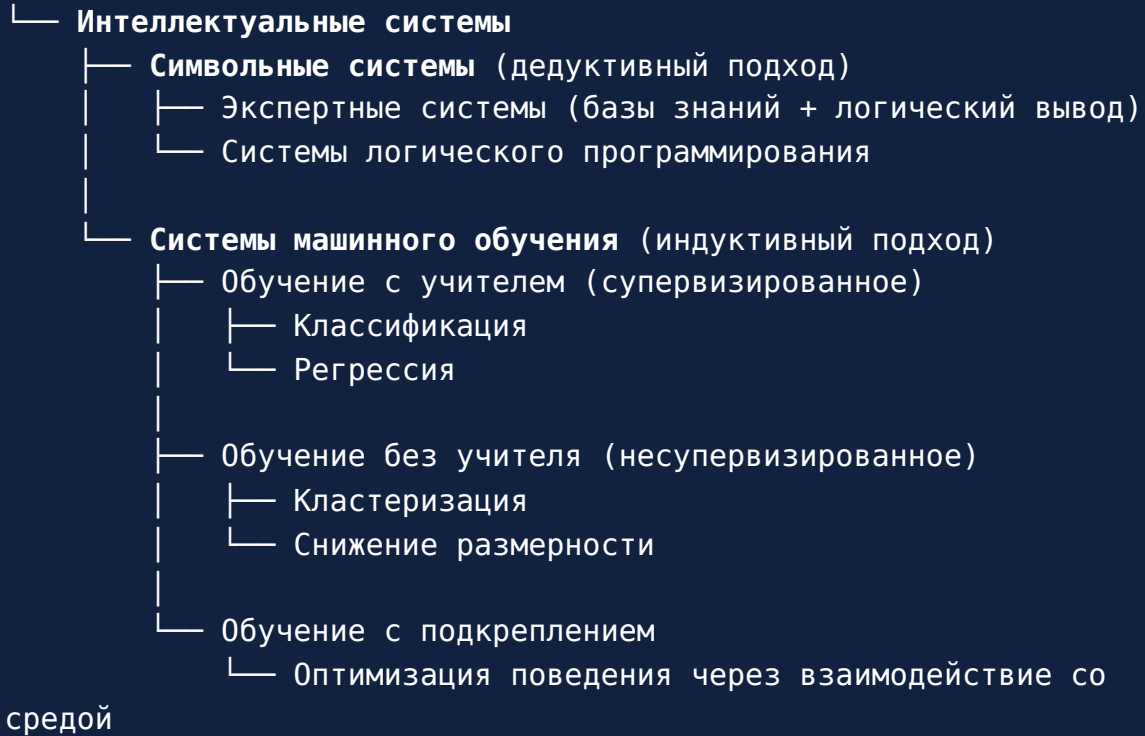


Динамическая система: $s_{t+1} = f(s_t, \mathbf{x}_t)$, $\mathbf{y}_t = g(s_t, \mathbf{x}_t)$

Фундаментальное различие:
наличие обратной связи и адаптация на опыте

Иерархия искусственного интеллекта (6)

Искусственный интеллект (ИИ)



Классификация интеллектуальных систем



Дедуктивный подход: экспертные системы (7)

Экспертная система — программа, использующая базу знаний и механизм логического вывода для решения задач в узкой предметной области на уровне эксперта.

Архитектура:

- База знаний: факты и правила («если-то»)
- Механизм вывода: прямой/обратный вывод
- Интерфейс с пользователем

Ограничения:

- Требуется явное формализованное знание
- Не способна к обучению на данных
- Проблема приобретения знаний

Пример: MYCIN (диагностика бактериальных инфекций, 1970-е).

Архитектура экспертной системы



Индуктивный подход: машинное обучение (8)

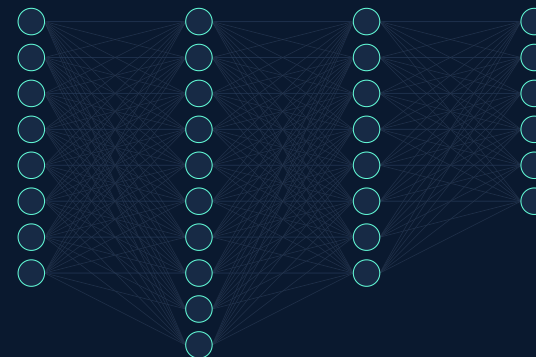
Машинное обучение (МО) — подраздел ИИ, изучающий алгоритмы, способные обучаться на данных без явного программирования правил.

Фундаментальная идея: при наличии достаточного объёма данных алгоритм МО автоматически извлекает закономерности и строит функцию отображения вход \rightarrow выход.

Преимущество перед дедуктивными системами: способность к обобщению на новые данные без изменения архитектуры системы.

Пример: нейронные сети как универсальные аппроксиматоры (теорема Цыбенко).

Нейронная сеть



Теоремы универсальной аппроксимации (9)

Теорема Колмогорова-Арнольда (1957):

Любая многомерная непрерывная функция представима как суперпозиция функций одной переменной.

$$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \phi_{q,p}(x_p) \right)$$

Теорема Цыбенко (1989):

Сеть с одним скрытым слоем и сигмоидальной активацией аппроксимирует любую непрерывную функцию на компакте.

$$\forall f \in C(I_n), \forall \epsilon > 0, \exists F(\mathbf{x}) = \sum_{j=1}^N \alpha_j \sigma(\mathbf{w}_j^T \mathbf{x} + b_j)$$

$$\sup_{\mathbf{x} \in I_n} |F(\mathbf{x}) - f(\mathbf{x})| < \epsilon$$

Практическое ограничение: Теоремы гарантируют существование, но не эффективность — число нейронов может быть экспоненциально большим.

Сравнение архитектур нейронных сетей

Однослойная сеть
(Shallow Network)



Сложность: $O(2^n)$
экспоненциальный рост

Многослойная сеть
(Deep Network)



Сложность: $O(n \cdot k)$
линейный рост

Глубокие сети эффективнее для сложных функций
иерархическое извлечение признаков с меньшими затратами

Преимущество глубины архитектур (10)

Экспоненциальная сложность плоских сетей:

Для представления иерархических функций однослойной сети требуется экспоненциальное число нейронов.

$$N_{shallow} \sim O(2^n)$$

Линейная сложность глубоких сетей:

Глубокая архитектура решает ту же задачу с полиномиальным числом параметров.

$$N_{deep} \sim O(n \cdot k)$$

где n — размерность входа, k — глубина сети.

Примеры глубоких архитектур:

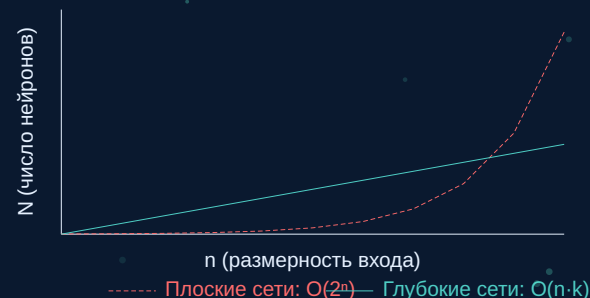
- **ResNet-152**: 152 слоя для классификации изображений (ImageNet)
- **BERT**: 12-24 слоя трансформера для обработки языка
- **GPT-3**: 96 слоёв, 175 млрд параметров

Вывод: Глубина позволяет выделять иерархические признаки с меньшей вычислительной стоимостью.

Глубина решает: экспонента vs полином

Плоские сети
Экспоненциальная сложность:
 $N_{shallow} \sim O(2^n)$

Глубокие сети
Линейная сложность:
 $N_{deep} \sim O(n \cdot k)$
 n — размерность входа, k — глубина



Примеры глубоких архитектур:

ResNet-152	BERT	GPT-3
152 слоя	12-24 слоя	96 слоёв
Классификация ImageNet	Обработка языка	175 млрд параметров

Глубина позволяет выделять иерархические признаки с меньшей вычислительной стоимостью

Теорема Эльдана-Шамира о глубине (11)

Теорема (Eldan & Shamir, 2016):

Существуют функции, которые глубокая сеть с 3 слоями может представить точно, а сеть с 2 слоями требует экспоненциального числа нейронов.

Математическая формулировка:

Существует распределение $p(\mathbf{x})$ и функция $f : \mathbb{R}^d \rightarrow \mathbb{R}$ такая что:

$$\exists \text{ 3-layer network : } N_3 = O(d)$$

$$\forall \text{ 2-layer network : } N_2 = \Omega(e^d)$$

Интуиция:

Глубокие сети эффективно представляют композицию функций $f = f_L \circ f_{L-1} \circ \dots \circ f_1$

Практическое значение:

- Объясняет эффективность глубоких архитектур в реальных задачах
- Глубина \neq просто больше параметров, а качественное изменение выразительности
- Основа для проектирования архитектур ResNet, DenseNet, Transformer

Теорема Эльдана-Шамира о глубине (2016)

Существуют функции, которые глубокая сеть с 3 слоями может представить точно, а сеть с 2 слоями требует экспоненциального числа нейронов.

Математическая формулировка:

Существует распределение $p(\mathbf{x})$ и функция $f : \mathbb{R}^d \rightarrow \mathbb{R}$ такая

$$\exists \text{ 3-layer network : } N_3 = O(d)$$

$$\forall \text{ 2-layer network : } N_2 = \Omega(e^d)$$

Визуальное сравнение:



2 слоя: $N_2 = \Omega(e^d)$



3 слоя: $N_3 = O(d)$

Интуиция:

Глубокие сети эффективно представляют композицию функций

$$f = f_L \circ f_{L-1} \circ \dots \circ f_1$$

$$f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow f_4 \rightarrow f_5$$

Практическое значение:

- Объясняет эффективность глубоких архитектур в реальных задачах
- Глубина \neq просто больше параметров, а качественное изменение выразительности
- Основа для проектирования архитектур ResNet, DenseNet

Три парадигмы машинного обучения (12)

Обучение с учителем

- Обучающая выборка: пары (\mathbf{x}_i, y_i)
- Цель: найти функцию $f : \mathcal{X} \rightarrow \mathcal{Y}$, минимизирующую ошибку
- Примеры: классификация изображений, предсказание цен

Обучение без учителя

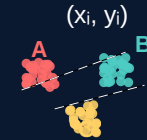
- Обучающая выборка: только $\{\mathbf{x}_i\}_{i=1}^N$
- Цель: выявление скрытой структуры данных (кластеры, многообразия)
- Примеры: кластеризация клиентов, снижение размерности

Обучение с подкреплением

- Взаимодействие агента со средой через действия a_t
- Обратная связь: скалярная награда r_{t+1}
- Цель: максимизация дисконтированной совокупной награды

Three ML Paradigms

Supervised



Unsupervised



Reinforcement Learning



$$s_t \rightarrow a_t \rightarrow r_{t+1}, s_{t+1}$$

Обучение с учителем: регрессия и классификация (13)

Регрессия

- Выходное пространство: $\mathcal{Y} \subseteq \mathbb{R}^d$ (непрерывное)
- Функция потерь: среднеквадратичная ошибка (MSE)
- Пример: предсказание цены недвижимости

Классификация

- Выходное пространство: $\mathcal{Y} = \{1, \dots, C\}$ (дискретное)
- Функция потерь: кросс-энтропия
- Пример: распознавание цифр (MNIST)

Единая формулировка:

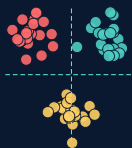
$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^N L(y_i, f(\mathbf{x}_i)) + \lambda \Omega(f)$$

где $\Omega(f)$ — регуляризатор, предотвращающий переобучение.

Классификация vs Кластеризация

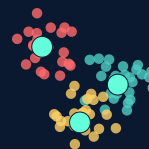
Классификация

(с учителем)



Кластеризация

(без учителя)



- Четкие метки классов
- Группировка по сходству
- Границы решений
- Центроиды кластеров

Градиентный спуск (14)

Градиентный спуск — итеративный алгоритм минимизации функции потерь $J(\theta)$:

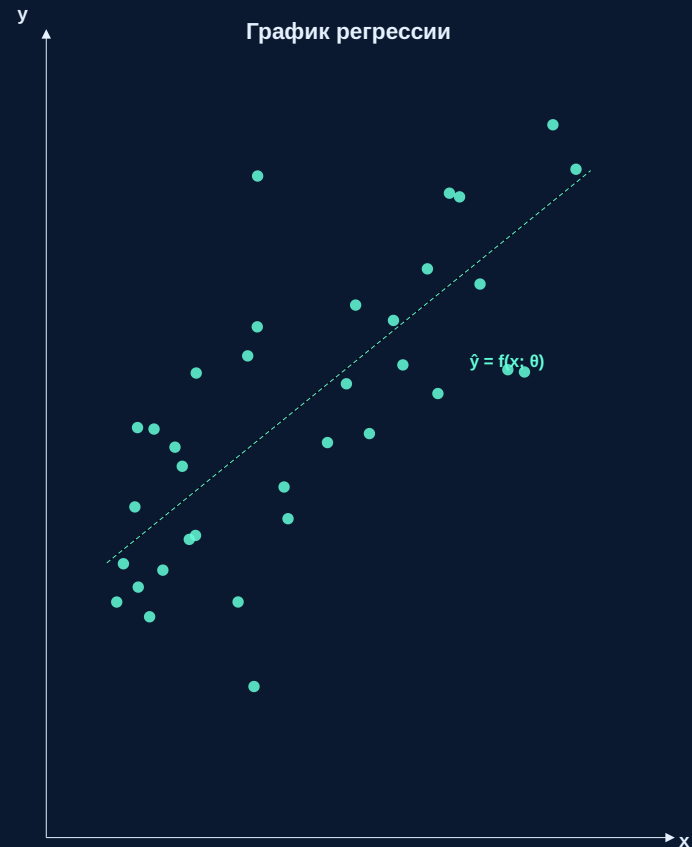
$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} J(\theta_t)$$

Где:

- θ — параметры модели
- $\eta > 0$ — скорость обучения
- $\nabla_{\theta} J$ — градиент функции потерь

Интуиция: движение в направлении наискорейшего спуска по поверхности ошибки.

Варианты: стохастический (SGD), с моментом (Momentum), Adam.



Обучение без учителя: кластеризация (15)

Кластеризация — группировка объектов так, чтобы объекты внутри кластера были более похожи друг на друга, чем на объекты из других кластеров.

Метрики качества:

- Силуэт: $s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$
- Внутрикластерная дисперсия

Алгоритмы:

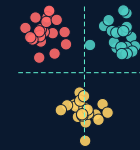
- K-means: минимизация внутрикластерной дисперсии
- Иерархическая кластеризация: построение дендрограммы
- DBSCAN: обнаружение кластеров произвольной формы

Применения: сегментация рынка, обнаружение аномалий.

Классификация vs Кластеризация

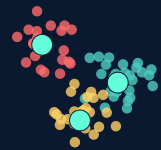
Классификация

(с учителем)



Кластеризация

(без учителя)



- Четкие метки классов
- Группировка по сходству
- Границы решений
- Центроиды кластеров

Нейронные сети: основа индуктивного подхода (16)

Нейронная сеть — вычислительная модель, реализующая индуктивный вывод на основе обучающих примеров.

Математическая модель нейрона:

$$a = \sigma(\mathbf{w}^T \mathbf{x} + b)$$

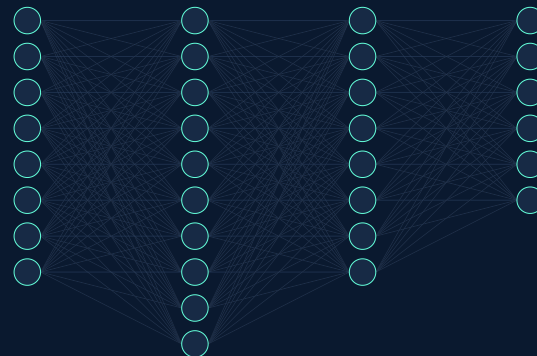
где $\sigma(\cdot)$ — нелинейная функция активации (ReLU, sigmoid, tanh).

Архитектурные компоненты:

- Входной слой: кодирование исходных данных
- Скрытые слои: иерархическое извлечение признаков
- Выходной слой: формирование результата

Теорема универсальной аппроксимации: полносвязная сеть с одним скрытым слоем может аппроксимировать любую непрерывную функцию с произвольной точностью.

Нейронная сеть



Автоматическое кодирование (автоэнкодеры) (17)

Автоэнкодер — архитектура для нелинейного снижения размерности через «бутылочное горлышко»:

Энкодер: преобразует вход $\mathbf{x} \in \mathbb{R}^D$ в компактное скрытое представление $\mathbf{z} = f_\phi(\mathbf{x}) \in \mathbb{R}^d$, $d \ll D$

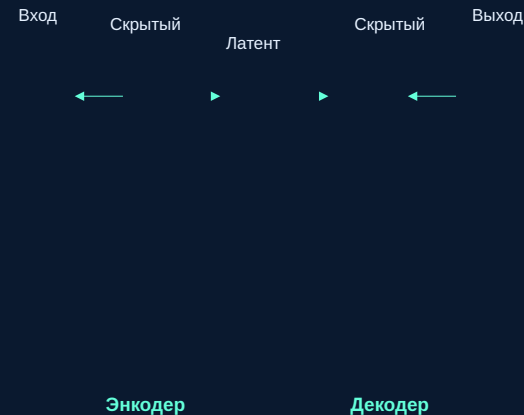
Бутылочное горлышко: узкое скрытое пространство, фиксирующее существенные признаки

Декодер: восстанавливает данные из кода $\hat{\mathbf{x}} = g_\theta(\mathbf{z})$

Применения: сжатие данных, денойзинг, генерация.

Архитектура автоэнкодера

Сжатие → Латентное пространство → Восстановление



Сравнение автоэнкодера и вариационного автоэнкодера (18)

Характеристика	Автоэнкодер (AE)	Вариационный автоэнкодер (VAE)
Скрытое пространство	Детерминированное	Вероятностное
Генерация	Ограниченная	Высококачественная
Обучение	Минимизация ошибки реконструкции	Максимизация ELBO
Интерполяция	Нелинейная	Плавная и семантически осмысленная

Ключевое уравнение VAE (ELBO):

$$\log p_{\theta}(\mathbf{x}) \geq \mathbb{E}_{q_{\phi}}[\log p_{\theta}(\mathbf{x}|\mathbf{z})] - \text{KL}(q_{\phi}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

Сравнение: Автоэнкодер vs Вариационный автоэнкодер

Автоэнкодер (AE)

Вариационный автоэнкодер (VAE)

Вход

Энкодер

z

(точка)

Декодер

Выход

Вход

Энкодер

(распределение)

Декодер

Выход

AE: фиксированный код | VAE: распределение → плавная интерполяция между объектами

Генеративно-состязательные сети (GAN) (19)

GAN реализует минимаксную игру между двумя сетями:

Генератор G_θ : отображает шум $\mathbf{z} \sim p_z(\mathbf{z})$ в пространство данных $\mathbf{x}' = G_\theta(\mathbf{z})$

Дискриминатор D_ϕ : оценивает вероятность принадлежности образца реальным данным

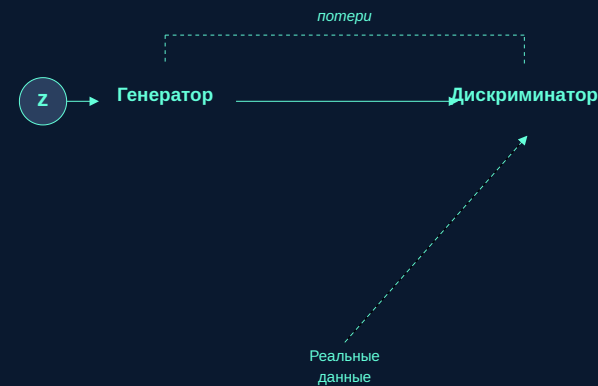
Целевая функция:

$$\min_{\theta} \max_{\phi} V(D_\phi, G_\theta) = \mathbb{E}_{\mathbf{x} \sim p_{data}} [\log D_\phi(\mathbf{x})] +$$

$$+ \mathbb{E}_{\mathbf{z} \sim p_z} [\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

Равновесие: генератор производит неотличимые от реальных данные ($D_\phi(\mathbf{x}) = 0.5$).

Архитектура GAN



Диффузионные модели (20)

Диффузионные модели основаны на постепенном разрушении и восстановлении структуры данных:

Прямой процесс: последовательное добавление шума к данным

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

Обратный процесс (обучаемый): постепенное удаление шума

$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_{\theta}(\mathbf{x}_t, t), \Sigma_{\theta}(\mathbf{x}_t, t))$$

Преимущества:

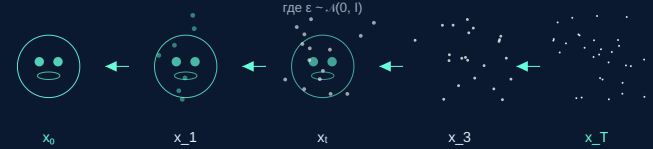
- Теоретически обоснованная сходимость
- Отсутствие проблемы коллапса мод (в отличие от GAN)
- Высокое качество генерации

Диффузионные модели: процесс шумоподавления

Прямой процесс: добавление шума



Обратный процесс: удаление шума (генерация)



$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_{\theta}(\mathbf{x}_t, t), \Sigma_{\theta}(\mathbf{x}_t, t))$$



Чистое изображение



Полный шум

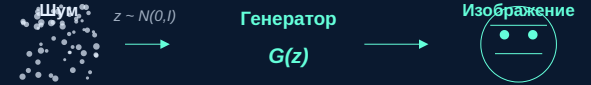
Поток генерации изображений (21)

Процесс генерации изображения в диффузионных моделях:

1. **Старт** ($t = T$): случайный шум $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$
2. **Итерация** t : модель $\epsilon_\theta(\mathbf{x}_t, t)$ предсказывает шум
3. **Денойзинг**: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
4. **Результат** ($t = 0$): фотореалистичное изображение \mathbf{x}_0

Каждая итерация уточняет изображение, постепенно удаляя шум и добавляя детали.

Генерация контента с помощью ИИ



Модели, обучающиеся распределению данных для создания новых, правдоподобных примеров

Детектирование объектов (22)

Детектирование объектов решает две задачи в едином конвейере:

1. **Локализация:** определение положения объекта через ограничивающий бокс $\mathbf{b} = (x_{min}, y_{min}, x_{max}, y_{max})$
2. **Классификация:** определение категории объекта внутри бокса $c \in \{1, \dots, C\}$

Метрика качества: Intersection over Union (IoU)

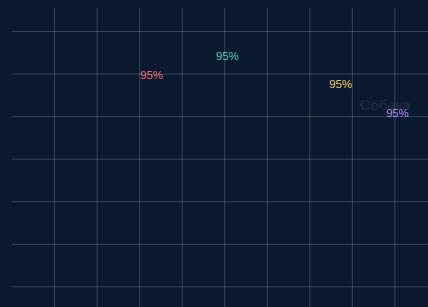
$$\text{IoU}(\mathbf{b}_{pred}, \mathbf{b}_{gt}) = \frac{|\mathbf{b}_{pred} \cap \mathbf{b}_{gt}|}{|\mathbf{b}_{pred} \cup \mathbf{b}_{gt}|}$$

Архитектурные подходы:

- Двухступенчатые (Faster R-CNN)
- Одноступенчатые (YOLO, SSD)
- Трансформерные (DETR)

Детектирование объектов

Локализация + классификация на изображении



Одноэтапные детекторы (YOLO) — высокая скорость

Двухэтапные детекторы (Faster R-CNN) — высокая точность

YOLO: единый проход для детекции (23)

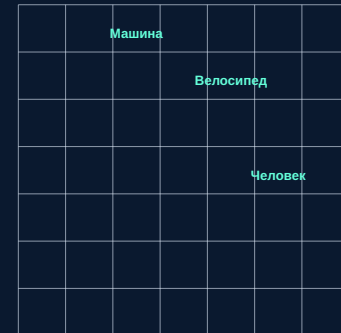
Принцип работы: разделение изображения на регулярную сетку $S \times S$, где каждая ячейка предсказывает:

- B ограничивающих боксов с координатами (x, y, w, h) и уверенностью

Преимущества: скорость (реальное время), глобальный контекст, простота архитектуры.

Архитектура YOLO (You Only Look Once)

Разделение изображения на сетку 7×7



Бокс 1

Бокс 2 (альтернатива)

Активная ячейка

$$\text{Loss} = \lambda_{\text{coord}} \sum (x_i - \hat{x}_i)^2 + \sum (\text{conf}_i - \hat{\text{conf}}_i)^2 + \lambda_{\text{noobj}} \sum \dots$$

Сегментация изображений (24)

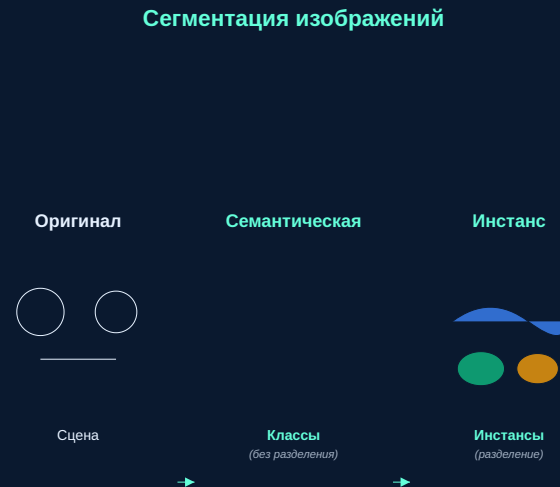
Семантическая сегментация:

- Каждому пикселю присваивается метка класса
- Все объекты одного класса неразличимы
- Архитектура: FCN, U-Net
- Метрика: mean IoU

Сегментация по инстансам:

- Каждый объект получает уникальный идентификатор
- Разделение объектов одного класса на отдельные инстансы
- Подходы: Mask R-CNN, кластеризация по признакам
- Метрика: Average Precision по маскам

Применения: автономные автомобили, медицинская визуализация, робототехника.

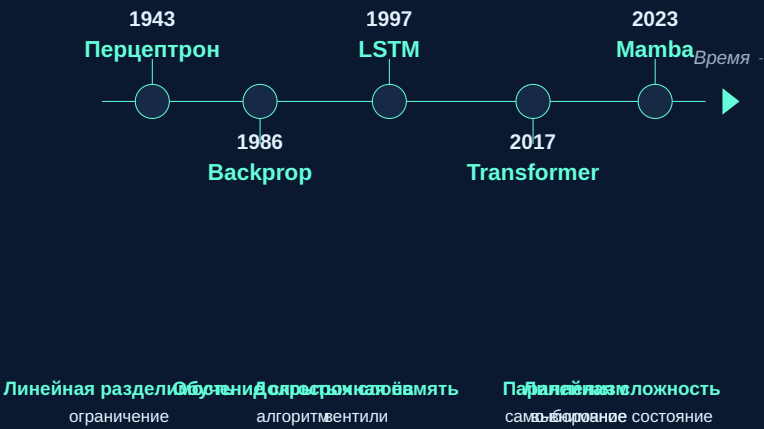


Семантическая: каждый пиксель → класс | Инстанс: каждый объект → уникальная маска

Хронология развития архитектур глубокого обучения (25)

Год	Архитектура	Ключевой вклад	Ограничение
1943	Перцептрон	Биологическая модель нейрона	Линейная разделимость
1986	Обратное распространение	Обучение многослойных сетей	Затухающие градиенты
1997	LSTM	Долгосрочная память через вентили	Вычислительная сложность
2012	AlexNet	Глубокие свёрточные сети	Требовательность к данным
2017	Трансформер	Параллельная обработка через внимание	Квадратичная сложность $O(N^2)$
2023	Mamba	Выборочное состояние (SSM)	Специализированная реализация

Эволюция архитектур нейросетей



Архитектура трансформера (26)

Трансформер состоит из двух основных компонентов:

Энкодер:

- Стек из N идентичных слоёв
- Каждый слой: много-головое само-внимание + полносвязная сеть с пропусками
- Обработка входной последовательности $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$

Декодер:

- Стек из N идентичных слоёв
- Каждый слой: маскированное внимание + внимание к выходам энкодера + полносвязная сеть
- Авторегрессивная генерация выхода $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_m]$

Механизм внимания:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V}$$

Архитектура Трансформера



Mamba: линейная сложность для длинных последовательностей (27)

Проблема трансформеров: квадратичная вычислительная сложность $O(N^2)$.

Решение Mamba: выборочный механизм состояния (Selective State Space Model):

$$\mathbf{h}_t = \bar{A}(\Delta_t)\mathbf{h}_{t-1} + \bar{B}(\Delta_t)\mathbf{x}_t$$

$$\mathbf{y}_t = \bar{C}(\Delta_t)\mathbf{h}_t + \bar{D}(\Delta_t)\mathbf{x}_t$$

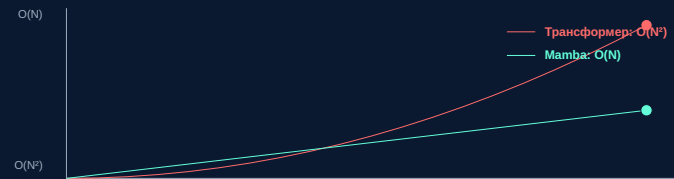
Ключевые свойства:

- **Адаптивная память:** динамическое решение о сохранении/забывании информации
- **Игнорирование нерелевантного:** подавление шума без дополнительных вычислений
- **Линейная сложность $O(N)$:** обработка последовательностей любой длины

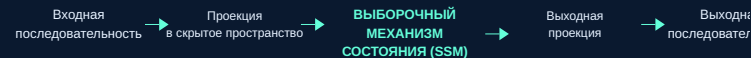
Результат: эффективная обработка контекстов длиной 100 000+ токенов.

Mamba: линейная сложность для длинных последовательностей

Сравнение вычислительной сложности



Архитектура Mamba: выборочный механизм состояния



Ключевое преимущество:

Выборочный механизм состояния адаптирует память к контексту, игнорируя нерелевантную информацию → линейная сложность $O(N)$

Сложность вычислений:
Transformer: $O(N^2)$
Mamba: $O(N)$

KAN: Kolmogorov-Arnold Networks (28)

Альтернатива многослойным перцептронам на основе теоремы Колмогорова-Арнольда:

Теорема: любая многомерная непрерывная функция может быть представлена как суперпозиция функций одной переменной:

$$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \phi_{q,p}(x_p) \right)$$

Архитектурное отличие от MLP:

- MLP: обучаемые веса + фиксированные нелинейности
- KAN: фиксированные веса + обучаемые нелинейности на рёбрах

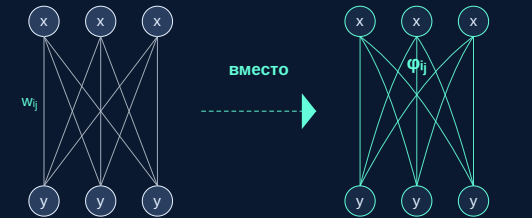
Преимущества:

- Интерпретируемость: каждая функция ϕ_{ij} соответствует влиянию признака
- Эффективность: меньше параметров при той же выразительности

Kolmogorov-Arnold Networks (KAN)

Альтернатива полносвязным сетям (MLP)

MLP (Multi-Layer Perceptron) KAN (Kolmogorov-Arnold Network)



Линейные веса между узлами

Обучаемые функции на рёбрах

$f(x) = \sum_i \Phi_i \left(\sum_j \phi_{ij}(x_j) \right)$ — Теорема Колмогорова-Арнольда

Будущее архитектур ИИ: четыре ключевых направления (29)

1. Вычислительная эффективность

- Снижение сложности с $O(N^2)$ до $O(N)$ (Mamba, SSM)
- Оптимизация для периферийных устройств (edge AI)

2. Интерпретируемость и объяснимость

- Отказ от «чёрных ящиков» (KAN, attention visualization)
- Формальная верификация поведения моделей

3. Мультиmodalность

- Единые архитектуры для текста, изображений, аудио, видео
- Кросс-модальное рассуждение и перенос знаний

4. Этическая ответственность

- Снижение предвзятости в данных и алгоритмах
- Прозрачность и подотчётность систем ИИ

Ключевые тренды развития ИИ



Ключевые выводы (30)

1. **Иерархия понятий:** ИИ \supset интеллектуальные системы \supset (символьные системы \cup системы машинного обучения)
2. **Два фундаментальных подхода:**
 - Дедуктивный (экспертные системы): знания \rightarrow вывод
 - Индуктивный (нейросети): данные \rightarrow обобщение
3. **Эволюция архитектур:** от рекуррентности к вниманию к выборочным состояниям
4. **Будущее ИИ:** инструмент расширения человеческих возможностей при условии этического проектирования и прозрачности.

