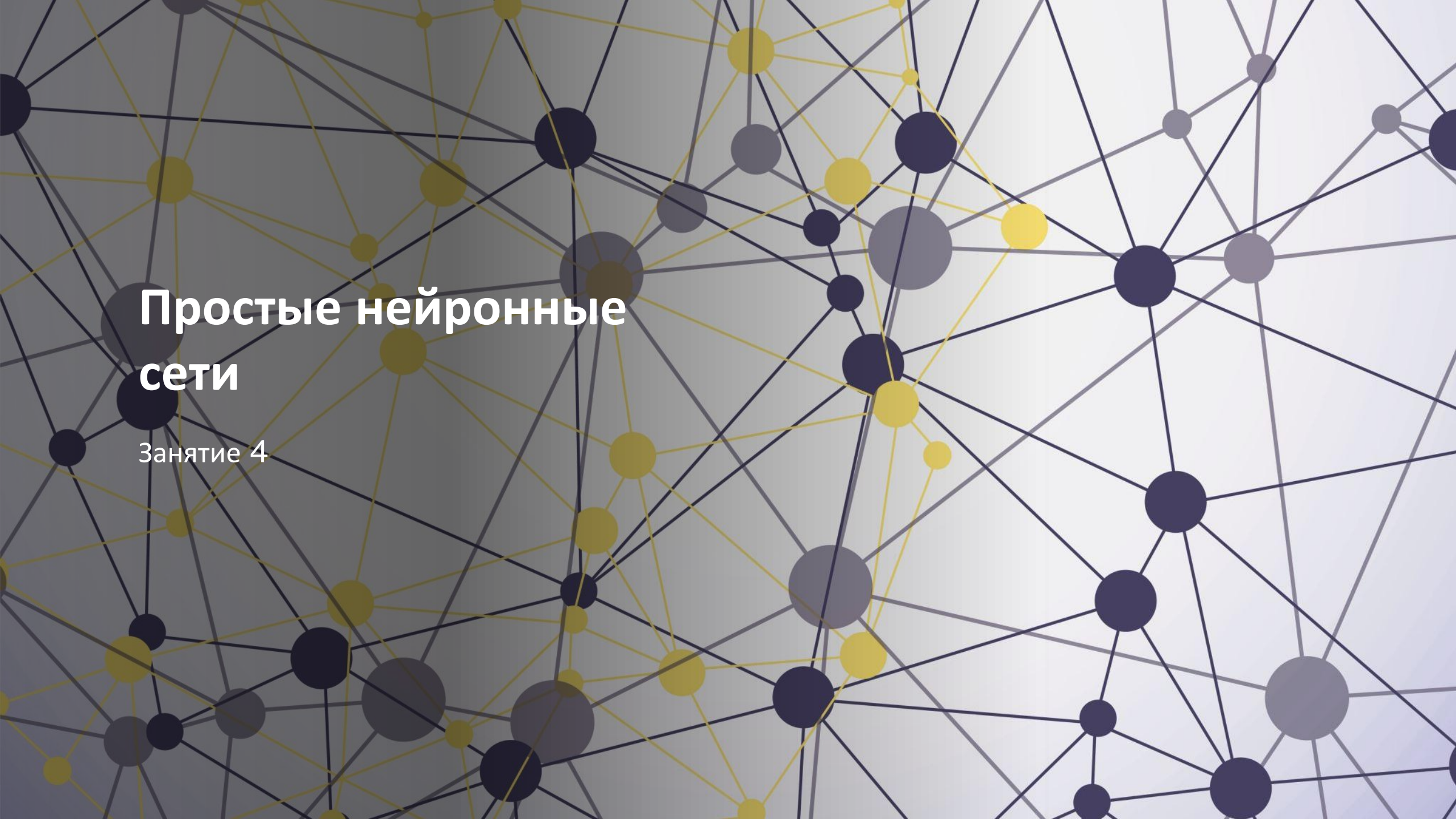


Простые нейронные сети

Занятие 4

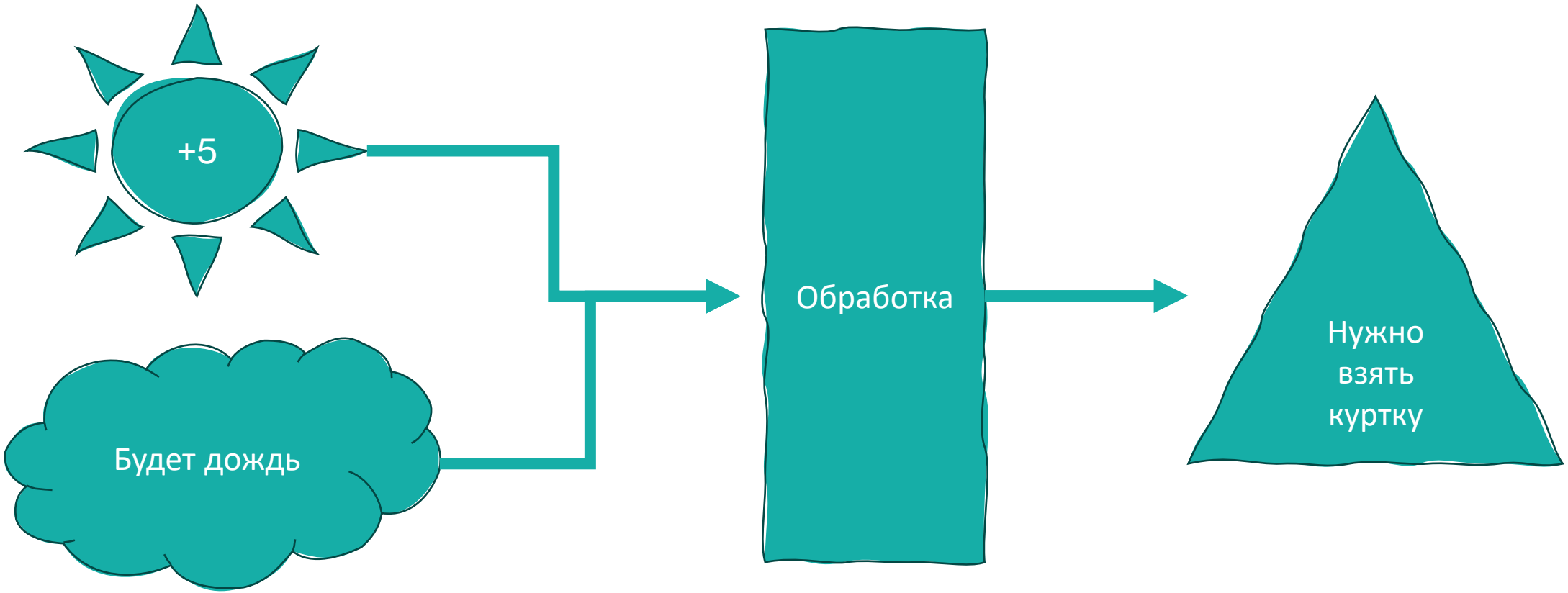
Предусловие

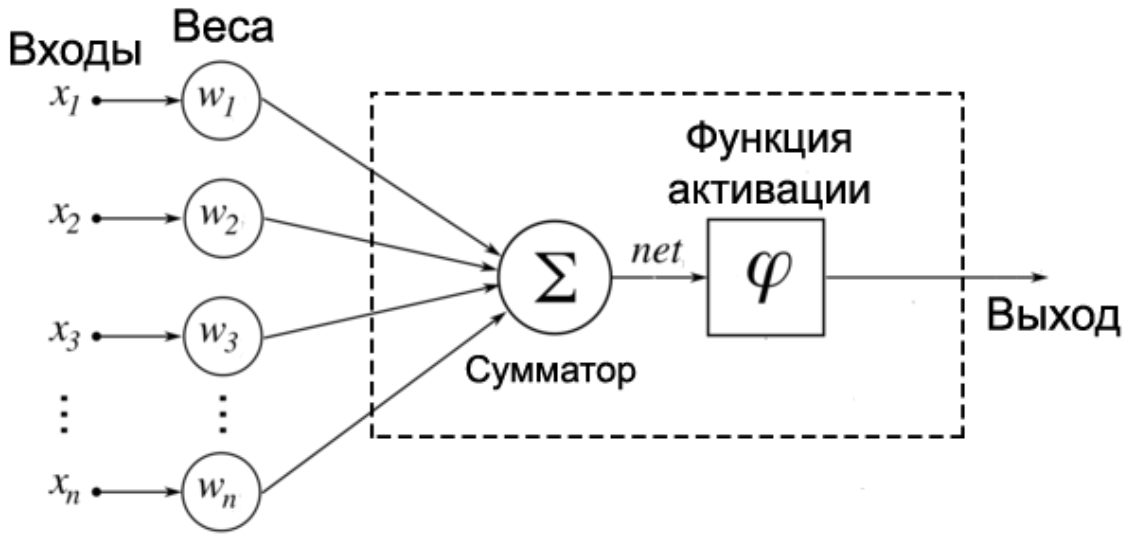
- Сохранить себе на гугл диск коллаб: https://clck.ru/3NAi6R

Что такое нейронная сеть?

- Это модель, которая имитирует работу мозга: она состоит из множества "нейронов", соединённых между собой, и может **обучаться на примерах**, чтобы выполнять задачи вроде:
- классификации (это кошка или собака?),
- прогнозирования (какая завтра будет температура?),
- распознавания (чей это голос?) и др.

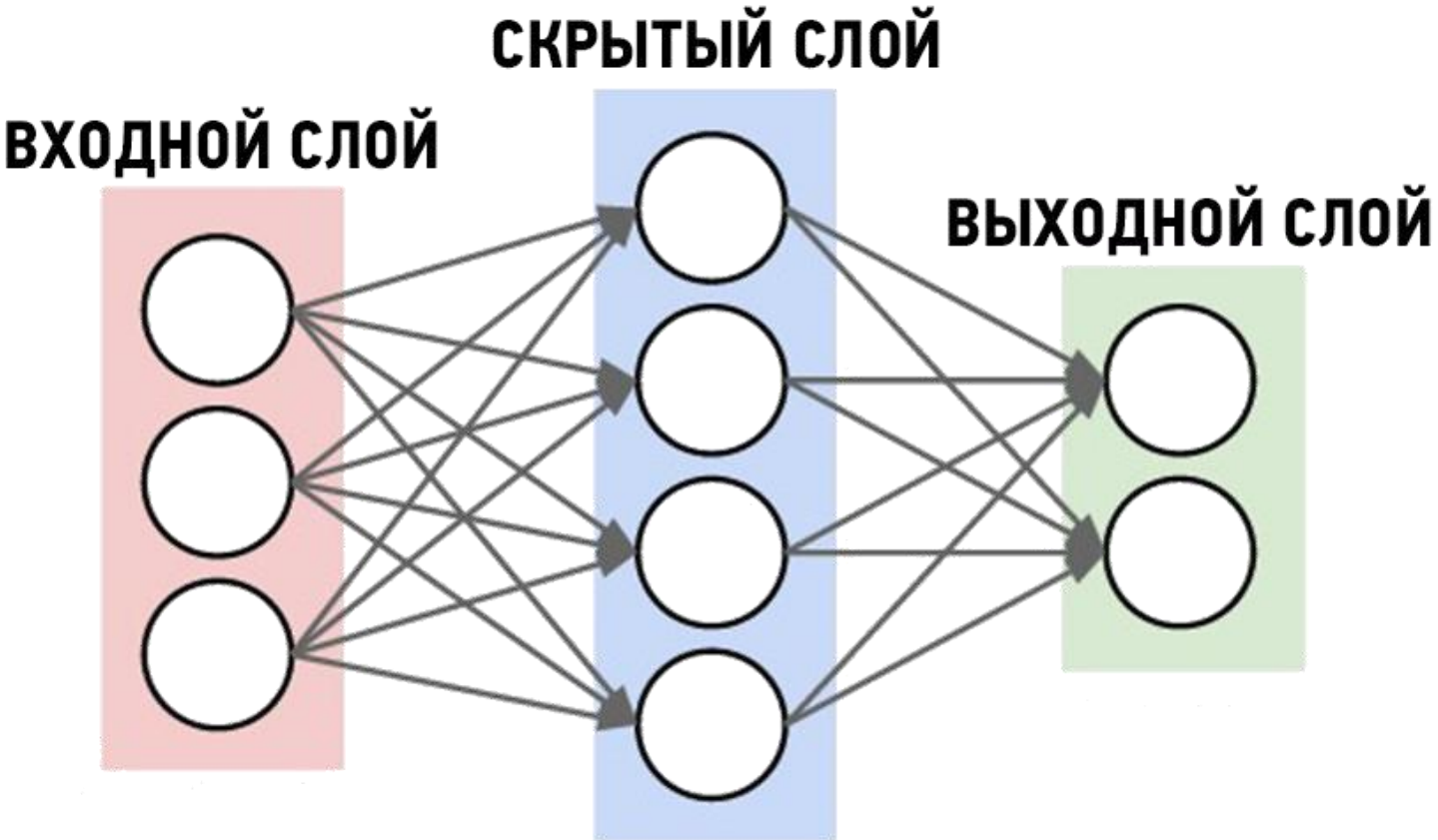
Нейросеть берёт входные данные → обрабатывает их → выдаёт результат.





Как устроен нейрон?

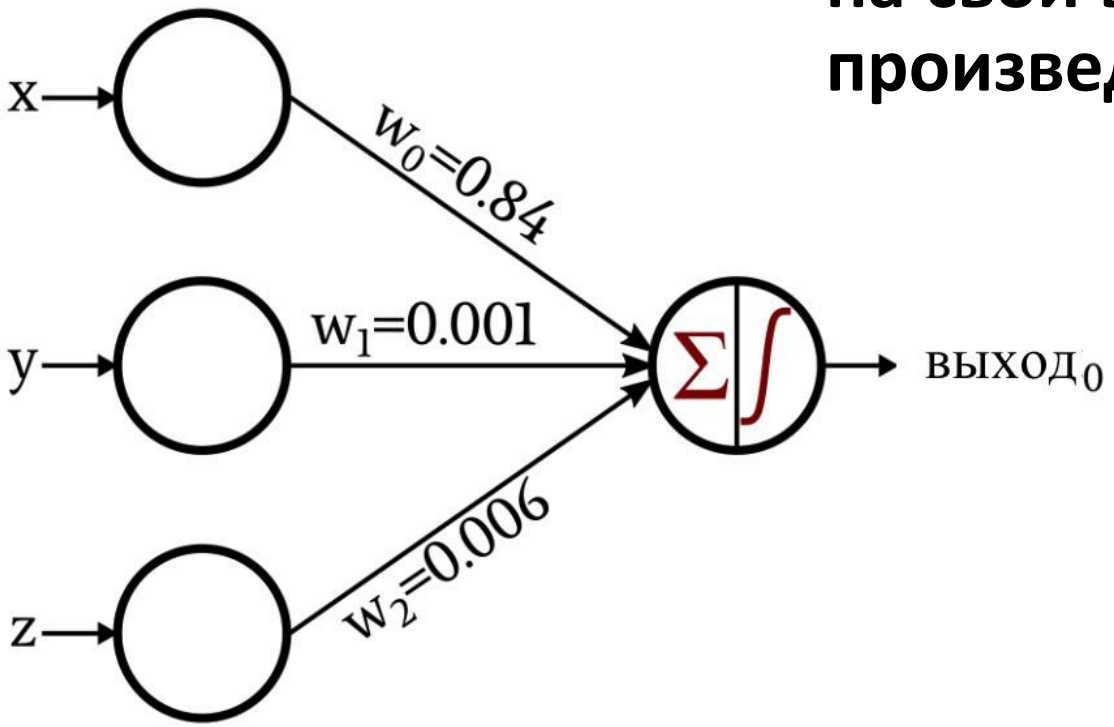
- Искусственный нейрон в нейронной сети — это математическая модель, имитирующая работу биологического нейрона. Она принимает входные данные, обрабатывает их с помощью весов и функции активации, и выдает результат, который передается дальше по сети



Как устроена нейронная связь?

- Входной слой — принимает данные
- Скрытые слои — учат связи
- Выходной слой — даёт ответ (например, 0 или 1)

Взвешенная сумма — это когда каждый элемент входа умножается на свой вес, а затем все эти произведения складываются.



- $z = x_1 \cdot w_1 + x_2 \cdot w_2 + \ldots + x_n \cdot w_n + b$

- Нейрон не просто суммирует входы. Он **учится**, какие входы важны, а какие нет, и выражает это через **веса**. Именно веса и взвешенная сумма позволяют **адаптироваться к данным**.

Как работает функция активации и вывод в нейронных связях

- Функция активации — это математическая функция, которая применяется к сумме входных сигналов нейрона (взвешенных и с добавленным смещением) и решает, насколько сильно активировать нейрон, то есть передавать сигнал дальше.
- Функция активации нужна, чтобы нейрон мог моделировать нелинейные зависимости. Если бы мы не использовали функцию активации, то сеть была бы просто линейной комбинацией, и её возможности были бы сильно ограничены.

Принцип:

- Каждый нейрон получает входные значения $x_1, x_2, ..., x_n$.
- Каждый вход умножается на свой вес $w_1, w_2, ..., w_n$.
- Суммируются взвешенные входы + добавляется смещение b:

$$z = \sum_{i=1}^{n} w_i x_i + b$$

- Далее к z применяется функция активации $\phi(z)$, и это значение становится выходом нейрона:

$$y = \phi(z)$$

Пример

- Есть обученная нейросеть, которая может определять 10 животных: 🐶 Собака, 🐱 Кошка, 🦊 Лиса, 🐰 Кролик, 🐻 Медведь, 🦁 Лев, 🐺 Волк, 🐯 Тигр, 🐷 Свинья, 🐴 Лошадь.
- Мы подаем фото лисы на вход сети.

Шаги решения

1. Подаем на вход картинку (заранее преобразованную в тензор): (224, 224, 3)
2. В слоях модель собирает абстрактные признаки.
3. Сводим к массиву чисел: [0.12, -0.03, 0.55, ..., 0.88] – числовое описание того, что находится на фото.
4. Финальный слой — **10 нейронов**, по одному на каждое животное.
   Каждый нейрон "голосует", насколько он **уверен**, что фото принадлежит к его классу.

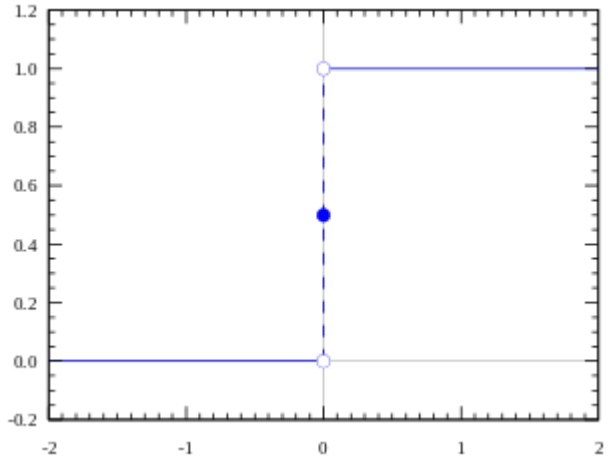
«Голосование» нейронов

Животное	Выход нейрона (после Softmax)
Собака	0.01
Кошка	0.02
Лиса	**0.93**
Кролик	0.01
Медведь	0.01
Лев	0.005
Волк	0.01
Тигр	0.01
Свинья	0.002
Лошадь	0.005

[Изображение]
↓
[Свертки → признаки]
↓
[Глубокие признаки (вектор)]
↓
[Классификатор]
↓
[Скорее всего: 🦊 Лиса – 93%]

Название	Формула	Диапазон значений	Для чего используется	Что означает значение выхода
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	(0, 1)	Используется в задачах бинарной классификации (например, "да или нет"). Преобразует любое число в вероятность.	**0.92** → "Сеть думает, что объект относится к классу 1 с вероятностью 92%"
ReLU	$f(x) = \max(0, x)$	[0, +∞)	Простая и быстрая активация. Включает нейрон только при положительном сигнале. Широко применяется в скрытых слоях.	**0** → "Этот нейрон не считает информацию важной в этом примере (молчит)"
Tanh	$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	(-1, 1)	Используется для скрытых слоёв, когда важно направление сигнала (положительное или отрицательное). Центрирована в 0.	**-0.6** → "Нейрон сигнализирует, что признак выражен в противоположном направлении"
Leaky ReLU	$f(x) = \max(0.01x, x)$	(-∞, +∞)	Почти как ReLU, но сохраняет небольшой градиент даже при отрицательных входах (чтобы нейрон не "умирал").	**-0.03** → "Нейрон почти не активен, но немного реагирует"
Softmax	$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$	(0, 1), сумма = 1	Используется в многоклассовой классификации. Преобразует выходы в вероятности по классам, сумма которых = 1.	**[0.1, 0.1, 0.7, 0.1]** → "Сеть уверена на 70%, что это объект 3-го класса"

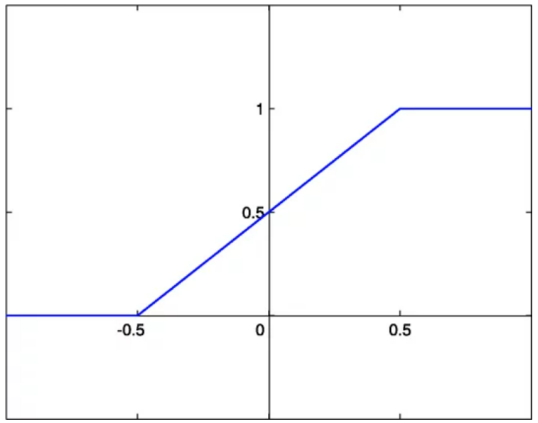
Функции активации



Ступенчатая функция

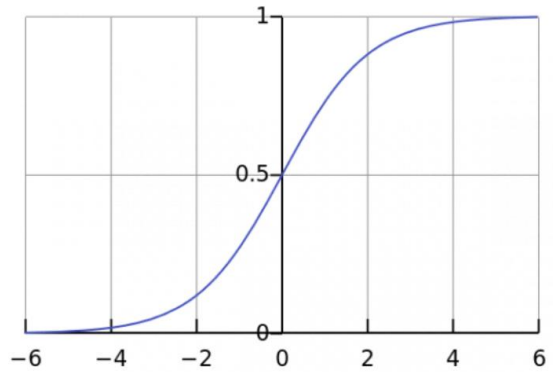
Линейная функция активации

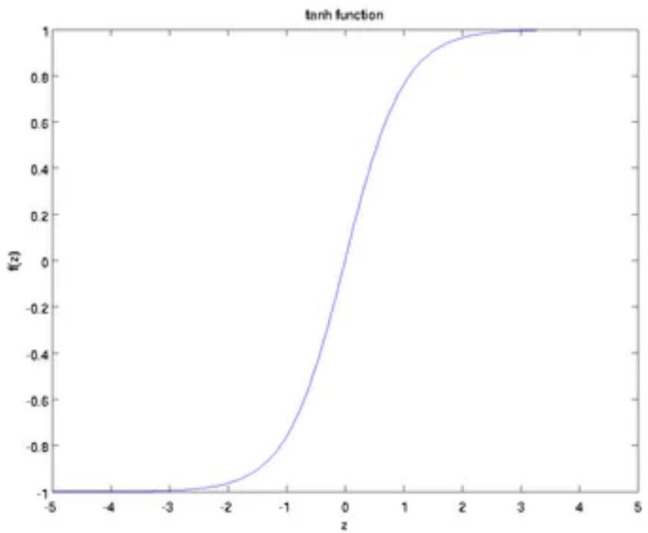
```
A = cx
```



Сигмоида

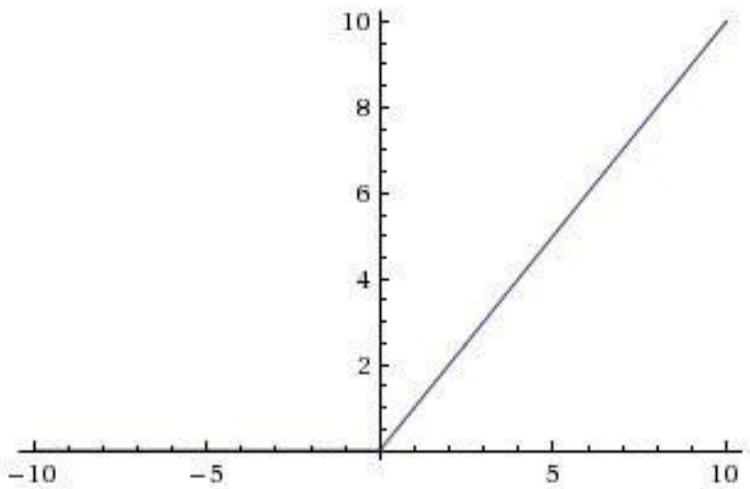
$$A = \frac{1}{1+e^{-x}}$$





Гиперболический тангенс

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$



Relu

Шаги обучения нейронной сети

- Основные шаги обучения — это:
- **Прямой проход (forward pass):** входные данные проходят через все слои сети, считаются выходы.
- **Вычисление ошибки (loss):** рассчитывается, насколько результат сети отличается от правильного ответа.
- **Обратный проход (backpropagation):** ошибка распространяется обратно через сеть, рассчитывая градиенты функции потерь по весам.
- **Обновление весов:** веса корректируются с помощью алгоритма оптимизации (например, градиентного спуска), чтобы уменьшить ошибку.

Почему так?

- Прямой проход нужен, чтобы получить предсказание.
- Ошибка нужна, чтобы понять, насколько предсказание плохое.
- Обратный проход позволяет понять, в каком направлении менять веса, чтобы улучшить результат.
- Обновление весов — это шаг к улучшению модели.
- Так повторяется множество раз (эпох), и сеть постепенно обучается.

Основные виды нейронных сетей

Вид сети	Структура / Вид связи	Главная особенность	Применение	Плюсы	Минусы
Полносвязная (Dense, FFNN)	Каждый нейрон предыдущего слоя связан с каждым нейроном следующего	Простая, полностью связная	Классификация, регрессия	Простая, универсальная	Плохо масштабируется на большие данные и изображения
Сверточная (CNN)	Связь с локальными участками входа, сверточные фильтры	Автоматически выделяет признаки, учитывает структуру изображения	Обработка изображений, видео	Эффективна для изображений	Не подходит для последовательностей
Рекуррентная (RNN)	Циклические связи: выходы зависят от предыдущих состояний	Помнит последовательность, работает с временными данными	Текст, аудио, временные ряды	Обработка последовательностей	Проблемы с долгой памятью (затухающие градиенты)