СӘТБАЕВ УНИВЕРСИТЕТІ



SATBAYEV UNIVERSITY

# *Основы искусственного интеллекта. Лекция 7. Нейронные сети и глубокое обучение. Основы нейронных сетей, введение в библиотеку TensorFlow/Keras*


Преподаватель: **Қасенхан Арай Мейрамбайқызы, доктор PhD, ассоциированный профессор кафедры «Программная инженерия»**

**a.kassenkhan@satbayev.university**

# *Содержание*

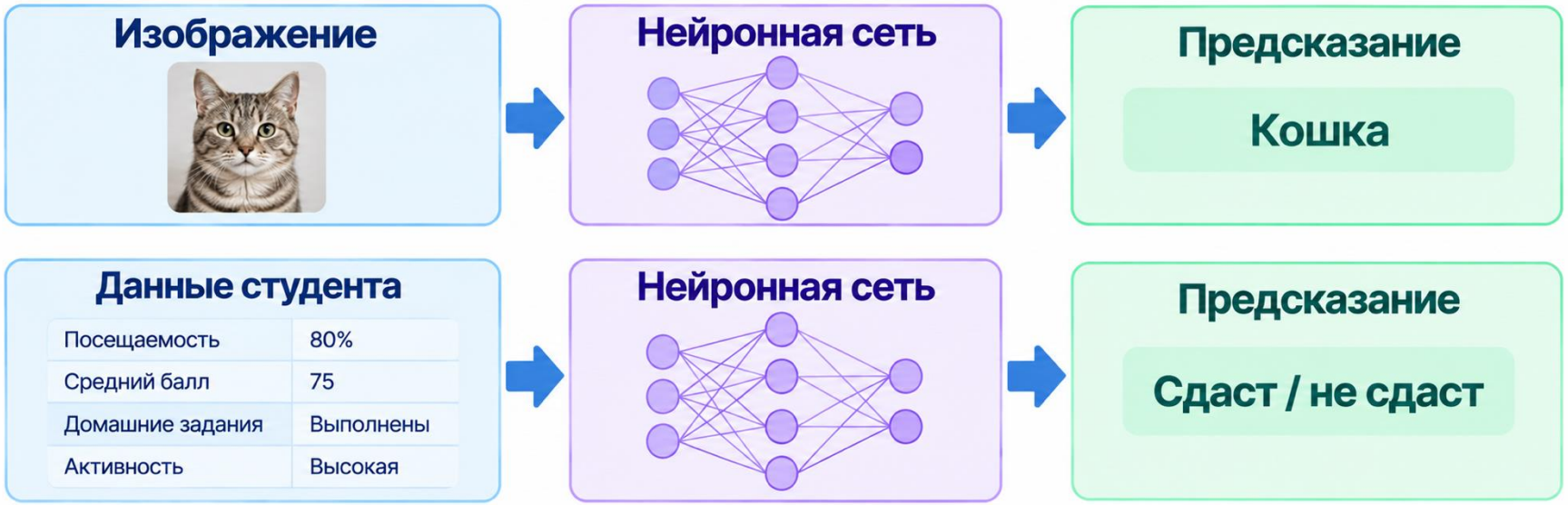

1. Что такое нейронная сеть
2. Искусственный нейрон
3. Веса, bias и функция активации
4. Слои нейронной сети
5. Как нейронная сеть обучается
6. Что такое глубокое обучение
7. MLP, CNN и RNN
8. TensorFlow и Keras
9. Создание первой нейронной сети
10. Обучение и оценка модели

# *По завершению урока Вы будете знать:*

1. **Что такое искусственная нейронная сеть**
2. **Как работает отдельный нейрон**
3. **Какую роль играют веса, bias и функции активации**
4. **Как устроены слои нейронной сети**
5. **Как сеть обучается с помощью функции потерь и backpropagation**
6. **Чем глубокое обучение отличается от классического ML**
7. **Для чего используются MLP, CNN и RNN**
8. **Как создать и обучить простую нейронную сеть в Keras**

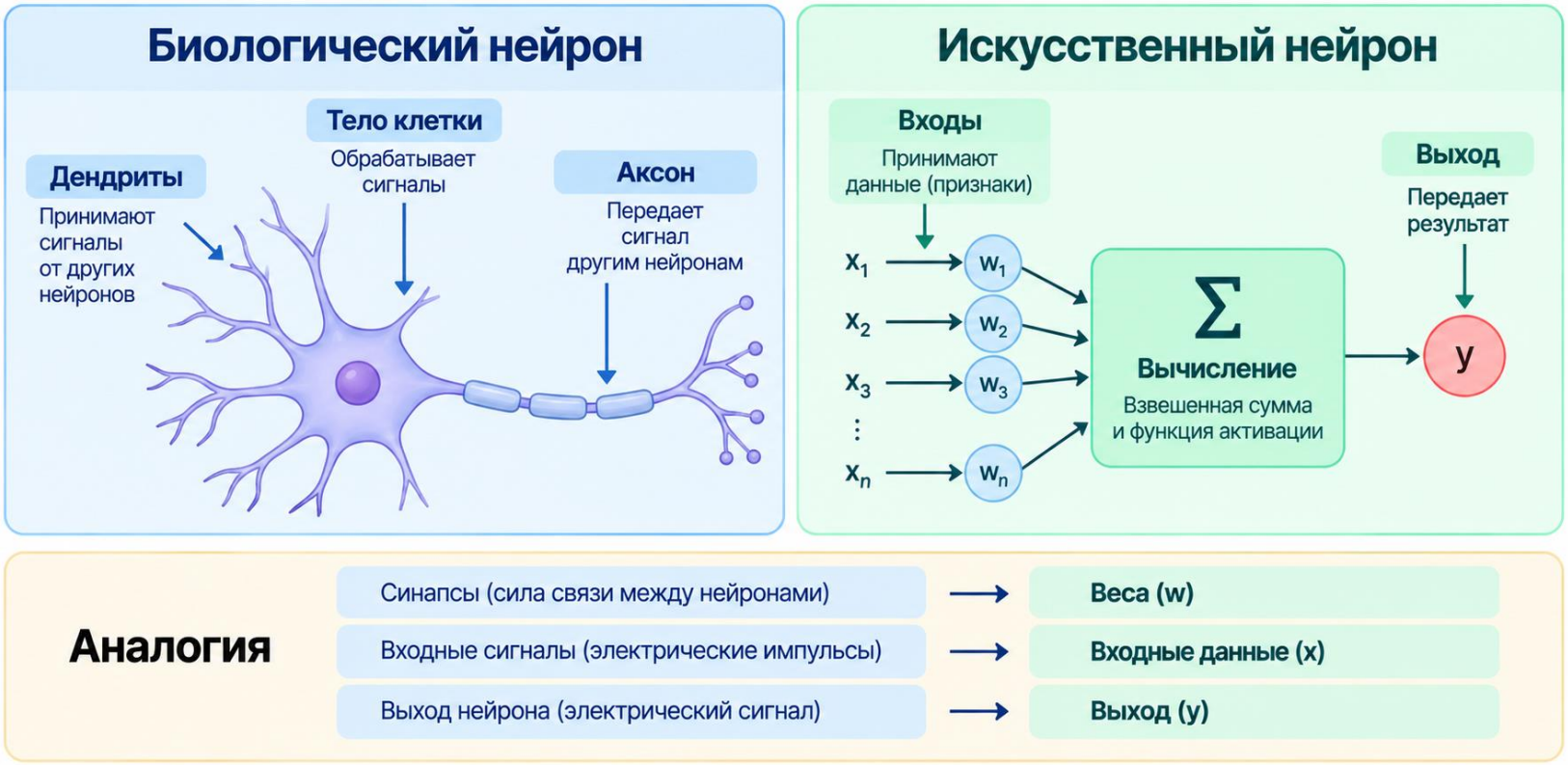
# *Что такое нейронная сеть?*

**Нейронная сеть — модель, состоящая из связанных вычислительных элементов — нейронов, которая обучается находить зависимости в данных.**




**Главная идея: сеть обучается на данных, изменяя свои параметры.**

# От биологического нейрона к искусственному




**Искусственный нейрон не копирует мозг — это математическая модель, вдохновлённая его принципом работы.**

# *Как работает искусственный нейрон?*



# *Веса и Bias: что обучает сеть?*

**Веса — weights**

**Определяют важность входных данных.**

Например:
hours × 0.8
age × 0.1

**Больший вес → большее влияние.**

**Смещение — bias**

**Позволяет дополнительно сдвигать результат:**

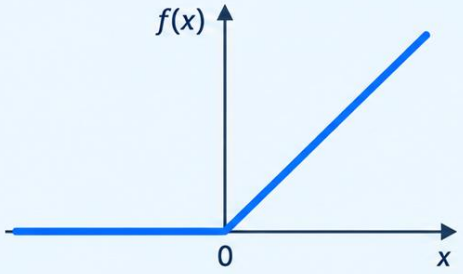
$$z = x_1w_1 + x_2w_2 + b$$

**Обучение нейронной сети — это прежде всего подбор весов и bias.**

# Зачем нужна функция активации?

## ReLU

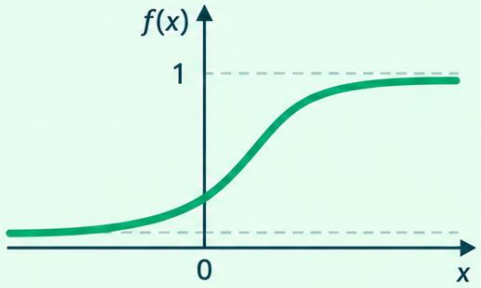
$$f(x) = \max(0, x)$$




Основной вариант для скрытых слоёв.

## Sigmoid

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




Удобна для бинарного выхода.

## Softmax

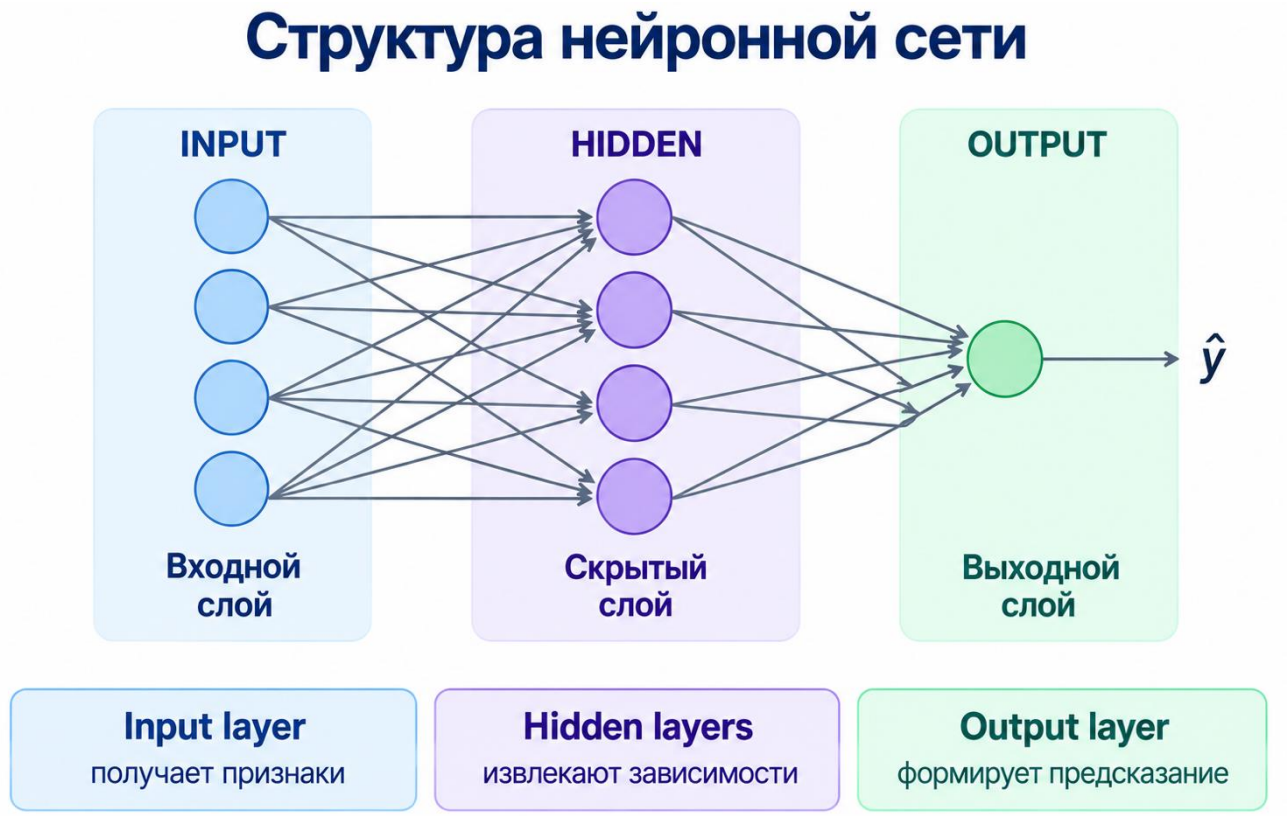
$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

| | | |
|---|---|---|
|  | Кошка | 0.80 |
|  | Собака | 0.15 |
|  | Птица | 0.05 |

Используется для многоклассовой классификации.

**Активационная функция определяет, какое значение передается от нейрона к следующему слою.**

# *Слои нейронной сети*



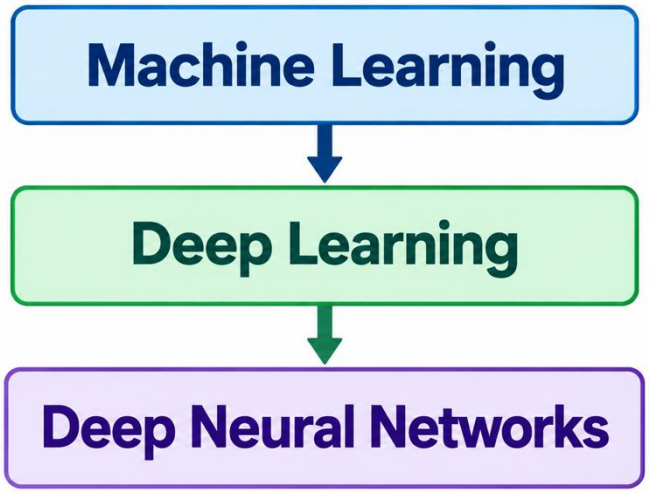

**Несколько скрытых слоёв → Deep Neural Network**

# Как нейронная сеть обучается?




**Цель обучения — уменьшить ошибку модели.**

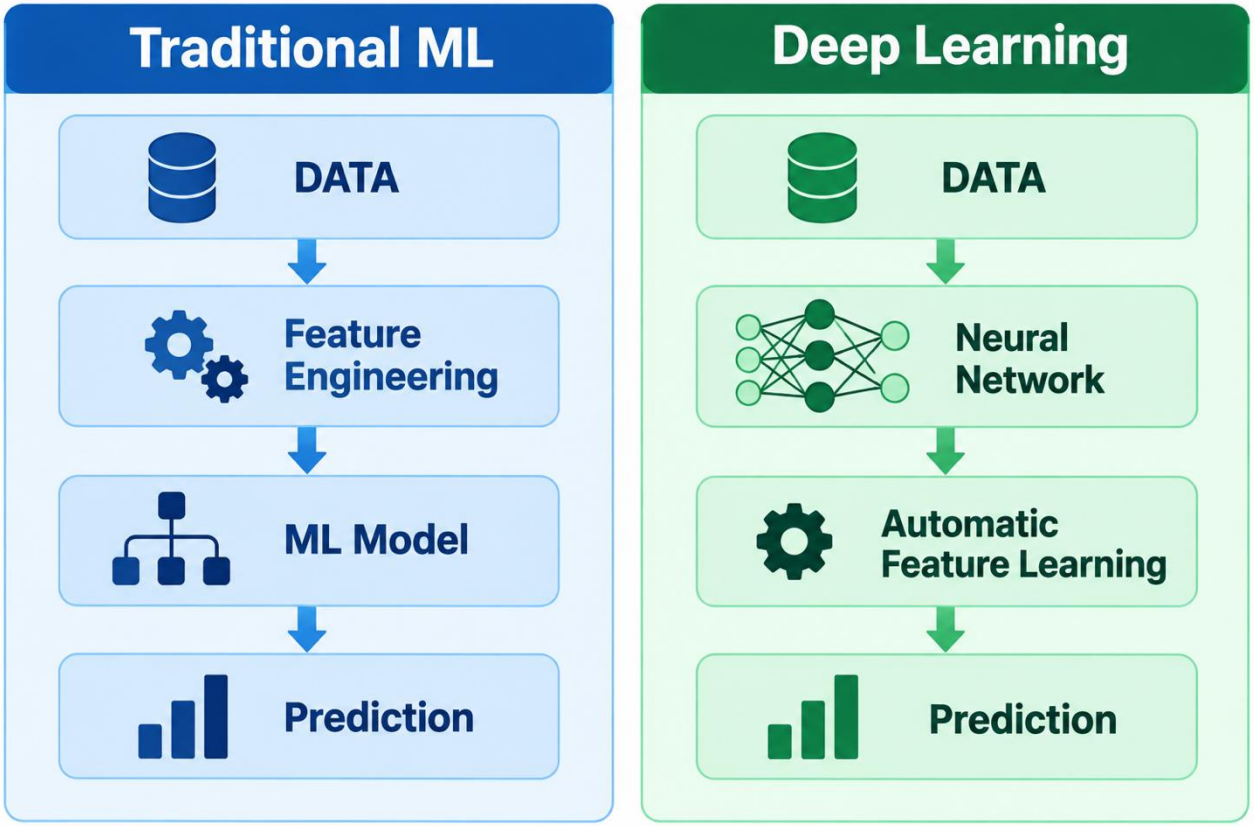
# Что такое глубокое обучение?




Глубокое обучение — направление машинного обучения, основанное на многослойных нейронных сетях.



# ML и Deep Learning

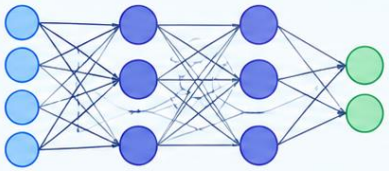



**Главное отличие: глубокая сеть может автоматически обучаться представлениям признаков.**

# Основные типы нейронных сетей

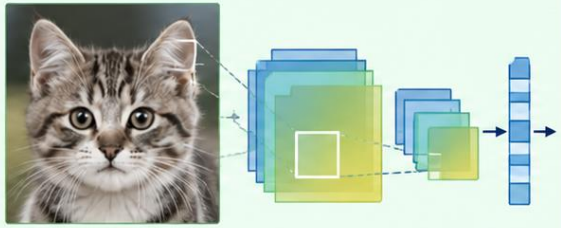
## MLP
**Табличные данные**

| Age | Income | Score | ... |
|---|---|---|---|
| 25 | 50000 | 0.8 | ··· |
| 32 | 62000 | 0.1 | ··· |
| 47 | 58000 | 0.6 | ··· |
| 28 | 52000 | 0.3 | ··· |



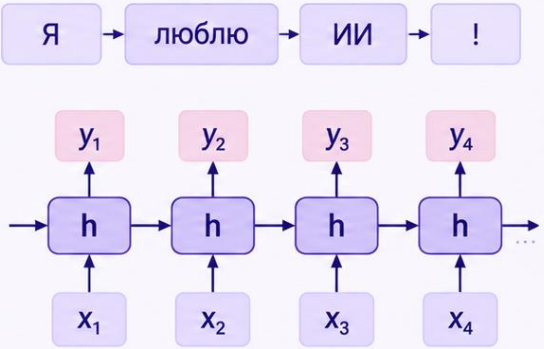
**Пример:**
прогноз / классификация

## CNN
**Изображения**



**Пример:**
распознавание объектов

## RNN
**Последовательности**




**Пример:**
текст / временные ряды

**Разные структуры данных → разные архитектуры**

# TensorFlow и Keras







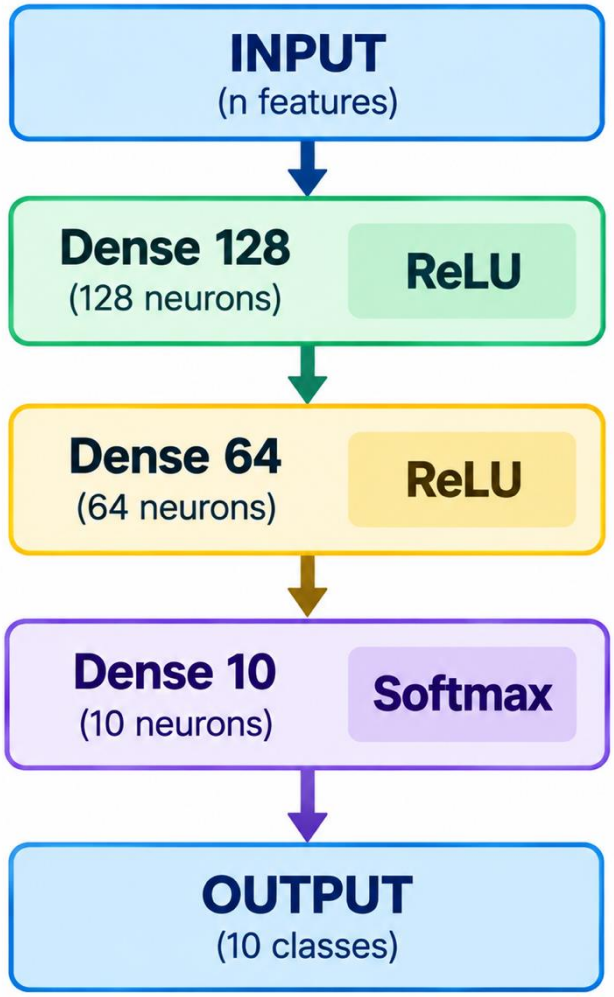
**Keras делает создание нейронных сетей проще.**

# Создаём первую нейронную сеть

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(128, activation="relu"),
    layers.Dense(64, activation="relu"),
    layers.Dense(10, activation="softmax")
])
```

128, 64, 10 → количество нейронов
ReLU → скрытые слои
Softmax → 10 классов



# Compile: как модель будет обучаться?

```
model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```









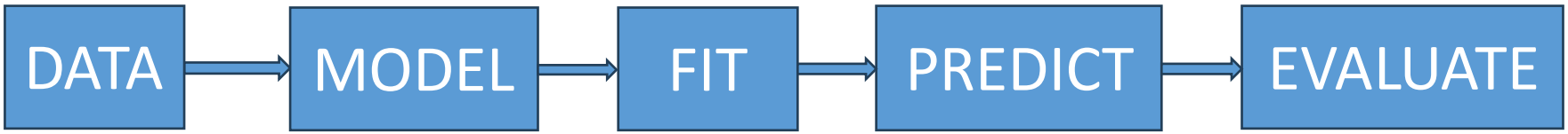
# Обучаем и оцениваем модель

```
model.fit(
    X_train,
    y_train,
    epochs=10,
    batch_size=128
)
```

```
test_loss, test_acc = model.evaluate(
    X_test,
    y_test
)

print(test_acc)
```

**epoch** → один проход по обучающим данным
**batch** → часть данных за один шаг
**accuracy** → доля правильных ответов



# Заключение




**Нейрон → базовый элемент**
**Deep Learning → многослойные нейронные сети**
**Keras → создание и обучение модели**

**Нейронная сеть обучается на данных, постепенно изменяя веса так, чтобы уменьшить ошибку и улучшить предсказания.**