

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Дальневосточный федеральный университет»**

**(ДВФУ)**

|  |
| --- |
| **ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ**  **(ШКОЛА)**  **Департамент математического и компьютерного моделирования** |

**О Т Ч Е Т**

о прохождении производственной практики.

(научно – исследовательская работа)

направление подготовки 01.03.02 «Прикладная математика и информатика»

профиль «Математические и компьютерные технологии»

|  |  |  |
| --- | --- | --- |
|  |  | Выполнил студент  гр. Б9121-01.03.02мкт  Домашев С.А. \_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Отчет защищен:  с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | *(Ф.И.О.) (подпись)*  Руководитель практики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(должность, уч.звание)*  Кузнецов К.С.  *(Ф.И.О.) (подпись)*  «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025г. |
| Рег. № \_\_\_\_\_\_  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |  | Практика пройдена в срок  с «23» апреля 2025 г.  по «05» мая 2025 г.  (2 недели, рассредоточенная)) |

г. Владивосток

2025

Оглавление

[Аннотация 3](#_Toc197332949)

[Введение 4](#_Toc197332950)

[Постановка задачи 5](#_Toc197332951)

[Требования к окружению 6](#_Toc197332952)

[Требования к аппаратному обеспечению 6](#_Toc197332953)

[Минимальное требования (Для прототипа/разработки) 6](#_Toc197332954)

[Рекомендуемые требования (для экспериментов с большими данными) 6](#_Toc197332955)

[Требования к программному обеспечению 6](#_Toc197332956)

[ОС: 6](#_Toc197332957)

[Язык программирования и библиотеки: 6](#_Toc197332958)

[Средства визуализации и документации: 7](#_Toc197332959)

[Требования к пользователям 7](#_Toc197332960)

[Базовые знания: 7](#_Toc197332961)

[Желательные навыки: 7](#_Toc197332962)

[Введение в предметную область 8](#_Toc197332963)

[Математическая теорема Колмагорова – Арнольда 8](#_Toc197332964)

[Универсальная аппроксимационная теорема 9](#_Toc197332965)

[Принцип работы MLP 10](#_Toc197332966)

[Принцип работы KAN 15](#_Toc197332967)

[Основная часть 18](#_Toc197332968)

[Игрушечные датасеты 18](#_Toc197332969)

[Заключение 23](#_Toc197332970)

[Список литературы 24](#_Toc197332971)

Нейронные сети архитектуры KAN

# Аннотация

В рамках дипломной работы проводилось комплексное исследование новой архитектуры нейронных сетей на базе математической теоремы Колмагорова Арнольда. Исследование включает в себя изучение документации, научных статей и связанных модификаций, а также реализация архитектуры на различных задач для анализа функциональности архитектуры и библиотеки

# Введение

Нейронные сети играют ключевую роль в развитии искусственного интеллекта, обеспечивая прорывы в таких областях, как компьютерное зрение, обработка естественного языка и научные вычисления. Среди наиболее распространённых архитектур выделяются многослойные персептроны (MLP), которые, несмотря на свою эффективность, сталкиваются с ограничениями в интерпретируемости и вычислительной эффективности, и особенно при решении задач с малой размерностью данных.

В последние годы студентами из MIT была предложена новая архитектура — Kolmogorov-Arnold Networks (KAN), основанная на теореме Колмогорова-Арнольда, доказанной в середине XX века Андреем Колмогоровым и Владимиром Арнольдом. Эта теорема утверждает, что любая непрерывная многомерная функция может быть представлена как суперпозиция конечного числа непрерывных функций одной переменной. KAN используют этот принцип, заменяя фиксированные функции активации традиционных нейронных сетей на обучаемые унивариатные функции на рёбрах сети, полностью исключая линейные веса. Обычно эти функции параметризуются сплайнами, что, как показывают исследования, позволяет KAN достигать высокой точности с меньшим количеством параметров по сравнению с MLP.

Преимущества KAN включают не только улучшенную точность, но и повышенную интерпретируемость. Их структура позволяет интуитивно визуализировать модель и взаимодействовать с ней, что делает KAN особенно ценными для научных приложений. Например, исследования показывают, что KAN эффективны в задачах аппроксимации данных, решении уравнений с частными производными и анализе данных высокой размерности, таких как временные ряды, графовые данные и классификация гиперспектральных изображений ([A Comprehensive Survey on Kolmogorov Arnold Networks (KAN)](https://arxiv.org/html/2407.11075v1)). Более того, KAN могут выступать в роли "сотрудников" для учёных, помогая (пере)открывать математические и физические законы, что является неожиданным применением для нейронных сетей, обычно рассматриваемых как "чёрные ящики".

# Постановка задачи

Данная дипломная работа направлена на всестороннее исследование архитектуры KAN, охватывающее как теоретические, так и практические аспекты. Основной целью является понимание, как KAN могут решать текущие вызовы в области нейронных сетей, такие как недостаточная интерпретируемость и высокие вычислительные затраты традиционных архитектур.

Конкретные задачи включают:

* **Теоретические основы**: Изучение теоремы Колмогорова-Арнольда и её применения в проектировании нейронных сетей, включая анализ, как KAN используют суперпозицию унивариатных функций для представления многомерных функций.
* **Архитектурный дизайн**: Подробное описание структуры KAN, включая использование сплайнов для параметризации функций, отсутствие линейных весов и сравнение с MLP. Например, исследования указывают, что KAN заменяют веса на функции, параметризованные сплайнами, что, вероятно, улучшает адаптивность модели ([KAN: Kolmogorov-Arnold Networks](https://arxiv.org/abs/2404.19756)).
* **Оценка производительности**: Анализ эффективности KAN в различных приложениях, таких как аппроксимация данных, решение УЧП и обработка данных высокой размерности, с сравнением с традиционными MLP и другими архитектурами. ([Kolmogorov-Arnold Network](https://www.geeksforgeeks.org/kolmogorov-arnold-network/" \t "_blank)).
* **Интерпретируемость**: Исследование, как KAN улучшают прозрачность модели, и обсуждение их потенциального влияния на научные открытия. Например, KAN могут быть полезны для извлечения научных правил из данных, что особенно важно в физике и математике ([A Beginner-friendly Introduction to Kolmogorov Arnold Networks (KAN)](https://www.dailydoseofds.com/a-beginner-friendly-introduction-to-kolmogorov-arnold-networks-kan/)).
* **Вызовы и будущие направления**: Обсуждение текущих ограничений, таких как повышенные требования к вычислительным ресурсам и сложности в настройке гиперпараметров, а также предложения по их преодолению. Исследования указывают на возможные проблемы с переобучением и длительным временем обучения, что требует дальнейших исследований ([Kolmogorov-Arnold Networks: a Critique](https://medium.com/@rubenszimbres/kolmogorov-arnold-networks-a-critique-2b37fea2112e" \t "_blank)).

# Требования к окружению

## Требования к аппаратному обеспечению

Такой проект предполагает обучение нейросетевых моделей (в т.ч. KAN и MLP), а также обработку высокоразмерных данных. Поэтому аппаратные требования зависят от объёма данных и сложности моделей:

### Минимальное требования (Для прототипа/разработки)

* **Процессор:** 4-ядерный CPU (например, Intel Core i5 / AMD Ryzen 5)
* **Оперативная память:** от 8 ГБ
* **Накопитель:** SSD, минимум 20 ГБ свободного места
* **Графический ускоритель:** не обязателен, но желательно наличие хотя бы CUDA-совместимой видеокарты (например, GTX 1050 Ti) для ускоренного обучения

### Рекомендуемые требования (для экспериментов с большими данными)

* **Процессор:** 6–8-ядерный CPU (например, Intel i7 / Ryzen 7 и выше)
* **Оперативная память:** от 16–32 ГБ
* **Графический ускоритель:** NVIDIA GPU с поддержкой CUDA (например, RTX 3060/3070/3090 или A100 для продвинутых задач)
* **Накопитель:** SSD от 100 ГБ (особенно если используется большой датасет)

## Требования к программному обеспечению

### ****ОС:****

* Linux (Ubuntu, Arch, Debian и др.) — предпочтительно
* Windows / macOS — возможно, но может потребовать дополнительных настроек

### ****Язык программирования и библиотеки:****

* **Python 3.9+**
* **Основные библиотеки:**
* PyTorch или TensorFlow (в зависимости от реализации KAN)
* torch-spline (если используется PyTorch и B-сплайны)
* NumPy, SciPy, Matplotlib — для анализа данных и визуализации
* scikit-learn — для базового сравнения с MLP/другими моделями
* Jupyter Notebook или VS Code — для разработки и отчётности

### ****Средства визуализации и документации:****

* TensorBoard, WandB, Matplotlib/Seaborn — для графиков и мониторинга
* LaTeX / Overleaf / Word / Markdown — для написания теоретической части и отчётов

## Требования к пользователям

Этот проект ориентирован на пользователей, обладающих определёнными знаниями в математике и машинном обучении. Основные требования к компетенциям пользователя:

### ****Базовые знания:****

* Основы линейной алгебры, математического анализа и теории функций
* Понимание принципов машинного обучения и нейронных сетей (например, MLP, SGD, переобучение)
* Навыки работы с Python и популярными ML-фреймворками (PyTorch/TensorFlow)

### ****Желательные навыки:****

* Знания в области функционального анализа (для понимания пространства функций, суперпозиции, Гильбертовых пространств)
* Навыки визуализации данных
* Навыки настройки гиперпараметров и проведения экспериментов
* Опыт чтения научных статей (для работы с оригинальными источниками по KAN)

# Введение в предметную область

## Математическая теорема Колмагорова – Арнольда

На втором Международном конгрессе математиков в Париже в 1900 году математик Давид Гильберт представил список из 23 задач, охватывающие многие области математики: [алгебру](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0), [теорию чисел](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%87%D0%B8%D1%81%D0%B5%D0%BB), [геометрию](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F), [топологию](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BF%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F), алгебраическую геометрию, [группы Ли](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D1%83%D0%BF%D0%BF%D0%B0_%D0%9B%D0%B8), вещественный и комплексный анализ, [дифференциальные уравнения](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%84%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5), математическую физику, [теорию вероятностей](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9), а также [вариационное исчисление](https://ru.wikipedia.org/wiki/%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B8%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5). Проблемы должны были определить вектор развития математики в XX веке. Опубликованы задачи на английском были в 1902 годе и на тот момент все не были решены. Гильберт считал выдвинутые проблемы наиболее актуальными в математическом сообществе и по сей день решены не все.

В контекст всех задач погружаться не имеет никакого смысла, так как нас интересует только **Тринадцатая проблема из списка задач Гильберта.** Формулируется она следующим образом: можно ли решить общее уравнение седьмой степени с помощью функций, зависящих только от двух переменных?

В 1956 году советский математик Андрей Колмагоров доказал промежуточный результат, показывающий что любую непрерывную функцию через суперпозицию функций трёх переменных, но это не опровергало гипотезу Гильберта, так как функции трёх переменных не сводились к функциям двух переменных.

В 1957 году ученик Колмогорова, Владимир Арнольд, в возрасте всего 19 лет, усовершенствовал результат своего учителя. Арнольд доказал, что непрерывные функции трёх переменных можно представить через суперпозицию функций двух переменных. Его работа показала, что для любой непрерывной функции на компактном множестве существует представление вида:

где ​ - непрерывные функции двух переменных, а ​ - непрерывные функции двух переменных. Это доказательство опровергло предположение Гильберта, показав, что даже сложные функции трёх переменных (включая корни септического уравнения) могут быть выражены через суперпозицию функций меньшей размерности.

В том же 1957 году Колмогоров обобщил результаты, доказав свою знаменитую теорему, которая утверждает, что любая непрерывная функция представима через суперпозицию функций одной переменной:

Этот результат стал ещё более сильным опровержением гипотезы Гильберта, так как он показал, что даже функции двух переменных не являются необходимыми — достаточно одномерных функций.

## Универсальная аппроксимационная теорема

Идея нейронных сетей зародилась с работ Уоррена Мак-Каллока и Уолтера Питтса (1943), которые предложили модель искусственного нейрона. В 1960-х годах Фрэнк Розенблатт разработал перцептрон, но его ограничения, указанные Марвином Мински и Сеймуром Папертом в книге Perceptrons (1969), показали, что однослойные сети не могут решать задачи с нелинейно разделимыми данными (например, XOR). Это вызвало временный спад интереса к нейронным сетям.

В 1980-х годах начался ренессанс нейронных сетей благодаря разработке многослойных перцептронов (MLP) и алгоритма обратного распространения ошибки (backpropagation), что позволило обучать сети с скрытыми слоями. Формулировка теоремы (1989): В 1989 году Джордж Цыбенко опубликовал статью [Approximation by Superpositions of a Sigmoidal Function](https://www.sci-hub.ru/10.1007/bf02551274) в журнале Mathematics of Control, Signals, and Systems. Он доказал, что нейронная сеть с одним скрытым слоем, использующая сигмоидальную функцию активации, может аппроксимировать любую непрерывную функцию на компактном подмножестве с любой точностью, если количество нейронов в скрытом слое достаточно велико. Независимо от Цыбенко, Курт Хорник, Максвелл Стинчкомб и Хэлберт Уайт в 1989 году опубликовали похожий результат в статье [Multilayer Feedforward Networks are Universal Approximators](https://www.sci-hub.ru/10.1016/0893-6080(89)90020-8) в журнале Neural Networks. Они расширили теорему, показав, что многослойные сети также обладают этой способностью, и уточнили условия на функции активации.

**Универсальная аппроксимационная теорема** (Universal Approximation Theorem, UAT), или теорема Цыбенко — это фундаментальный результат в теории искусственных нейронных сетей, который объясняет, почему нейронные сети способны моделировать широкий класс функций. Она утверждает, что нейронная сеть с одним скрытым слоем и достаточным количеством нейронов может аппроксимировать любую непрерывную функцию на компактном подмножестве с любой заданной точностью, при условии, что функция активации является нелинейной и удовлетворяет определённым условиям (например, сигмоида или ReLU).

## Принцип работы MLP

Многослойный перцептрон (MLP, Multilayer Perceptron) — это тип искусственной нейронной сети, который используется для решения задач машинного обучения, таких как классификация (например, определение того, является ли изображение кошкой или собакой) и регрессия (например, прогнозирование цены дома). Это одна из самых простых, но мощных архитектур нейронных сетей, которая лежит в основе более сложных моделей глубокого обучения.

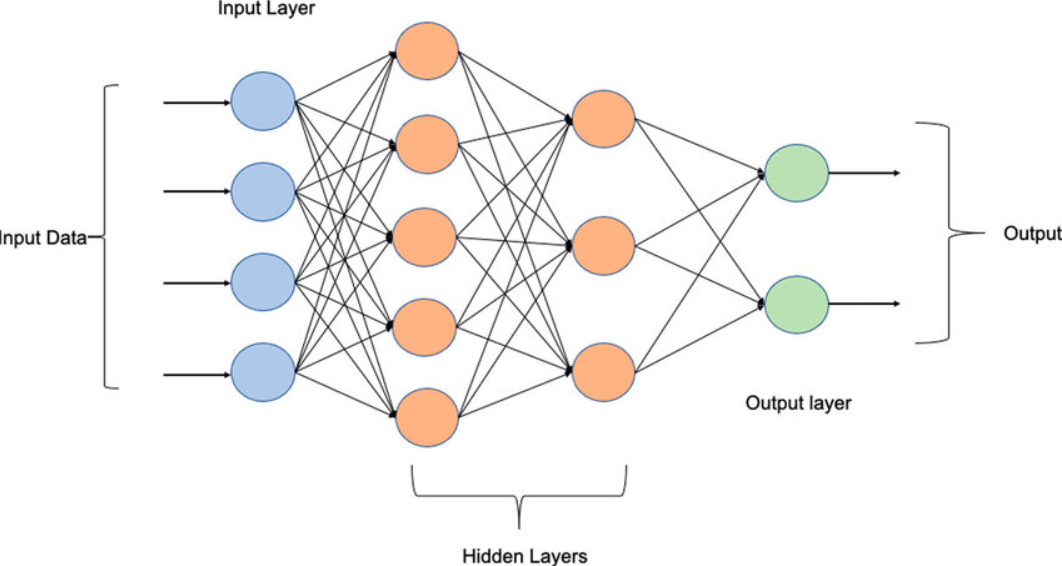


Рис. 1 – визуализация MLP

MLP — это нейронная сеть, состоящая из нескольких слоёв нейронов, соединённых друг с другом. Каждый нейрон в сети выполняет простую задачу: принимает входные данные, обрабатывает их и передаёт результат дальше. Слои в MLP организованы следующим образом:

* **Входной слой**: принимает исходные данные (например, пиксели изображения или числовые характеристики).
* **Скрытые слои**: обрабатывают данные, извлекая из них сложные закономерности. Чем больше скрытых слоёв, тем более сложные задачи может решать сеть.
* **Выходной слой**: выдаёт окончательный результат, например, вероятность принадлежности к классу или числовое предсказание.

Каждый нейрон в одном слое связан со всеми нейронами в следующем слое, что делает MLP **полносвязной сетью**. Эти связи имеют **веса** — параметры, которые определяют, насколько сильно сигнал от одного нейрона влияет на другой. Кроме того, у каждого нейрона есть **смещение**, которое помогает корректировать обработку данных.

Работа MLP делится на два основных этапа:

* **Прямое распространение (Forward Propagation)**: сеть принимает входные данные, обрабатывает их через слои и выдаёт прогноз.
* **Обратное распространение (Backpropagation)**: сеть анализирует ошибку прогнозирования и корректирует свои параметры, чтобы улучшить результат в будущем.

Давайте разберём эти этапы подробно, а затем рассмотрим, как сеть инициализируется и обучается.

**Прямое распространение** — это процесс, при котором данные проходят через сеть от входного слоя к выходному, преобразуясь на каждом этапе. Представьте это как конвейер, где данные постепенно превращаются в прогноз.

#### Шаг 1: Входной слой

Входной слой принимает данные. Например, если вы хотите классифицировать изображение, данные могут представлять собой яркость пикселей. Если это задача прогнозирования цены дома, данные могут включать площадь, количество комнат и т. д.

Каждый входной признак (например, значение пикселя) передаётся в сеть как отдельный нейрон входного слоя.

#### Шаг 2: Скрытые слои

Данные с входного слоя передаются в первый скрытый слой. Каждый нейрон в скрытом слое выполняет следующие действия:

1. **Собирает информацию**: нейрон получает сигналы от всех нейронов предыдущего слоя. Каждый сигнал умножается на соответствующий вес (вес определяет, насколько важен этот сигнал).
2. **Суммирует сигналы**: нейрон складывает все взвешенные сигналы и добавляет к ним смещение (смещение помогает нейрону гибко настраивать результат).
3. **Применяет функцию активации**: сумма сигналов пропускается через функцию активации, которая добавляет нелинейность. Это важно, потому что без нелинейности сеть не смогла бы моделировать сложные зависимости в данных. Примеры функций активации:
   * **ReLU (выпрямляющий линейный блок)**: обнуляет отрицательные значения, оставляя положительные без изменений. Это ускоряет обучение и делает сеть устойчивой.
   * **Сигмоида**: преобразует значение в диапазон от 0 до 1, что полезно для вероятностей.
   * **Tanh**: преобразует значение в диапазон от -1 до 1, что хорошо для центрированных данных.

После применения функции активации нейрон передаёт свой результат (активацию) всем нейронам следующего слоя.

#### Шаг 3: Выходной слой

Последний слой (выходной) получает данные от последнего скрытого слоя и выдаёт прогноз.

Тип функции активации в выходном слое зависит от задачи:

* Для **бинарной классификации** (например, «да» или «нет») используется сигмоида, которая выдаёт вероятность (число от 0 до 1).
* Для **многоклассовой классификации** (например, распознавания цифр от 0 до 9) используется функция softmax, которая распределяет вероятности между классами так, чтобы их сумма равнялась 1.
* Для **регрессии** (например, прогнозирования цены) может не использоваться функция активации, чтобы получить непрерывное число.

Результат выходного слоя — это предсказание сети

#### Шаг 4: Оценка ошибки

После получения предсказания сеть сравнивает его с истинным значением (y), используя **функцию потерь**. Функция потерь измеряет, насколько сильно предсказание отличается от правильного ответа. Примеры:

* **Среднеквадратичная ошибка (MSE)**: используется для регрессии, измеряет среднее квадратичное отклонение предсказания от истины.
* **Бинарная кросс-энтропия**: используется для бинарной классификации, оценивает, насколько предсказанные вероятности соответствуют истинным меткам.
* **Категориальная кросс-энтропия**: используется для многоклассовой классификации, оценивает качество вероятностного распределения.

Значение функции потерь показывает, насколько «плохо» сеть справилась с задачей. Цель обучения — минимизировать это значение.

**Обратное распространение** — это процесс, в ходе которого сеть «обучается», корректируя свои веса и смещения, чтобы уменьшить ошибку прогнозирования. Это похоже на то, как человек учится на своих ошибках: сеть анализирует, где она допустила ошибку, и настраивает свои параметры, чтобы в следующий раз быть точнее.

#### Шаг 1: Оценка ошибки на выходе

После прямого распространения сеть знает, насколько её предсказание отличается от истины (это значение функции потерь). Обратное распространение начинается с выходного слоя, где ошибка наиболее очевидна.

Для каждого нейрона в выходном слое вычисляется, насколько он «виноват» в общей ошибке. Это зависит от:

* Разницы между предсказанием и истинным значением.
* Тип функции активации (например, сигмоида или softmax влияет на то, как распределяется ошибка).

#### Шаг 2: Распространение ошибки назад

Ошибка от выходного слоя передаётся обратно через сеть, слой за слоем, до входного слоя. Для каждого нейрона в скрытых слоях определяется, насколько он повлиял на ошибку в следующем слое.

Этот процесс напоминает «обратный конвейер»: ошибка распределяется по сети, чтобы понять, какие нейроны и связи больше всего повлияли на неправильное предсказание.

При распределении ошибки учитывается:

* **Вес связей**: если связь между нейронами имеет большой вес, она сильнее влияет на ошибку.
* **Функция активации**: она определяет, как ошибка передаётся обратно (например, для ReLU ошибка не передаётся через нейроны с нулевой активацией).

#### Шаг 3: Вычисление корректировок

Для каждого нейрона сеть вычисляет, как нужно изменить его веса и смещение, чтобы уменьшить ошибку. Это делается следующим образом:

Если нейрон сильно повлиял на ошибку, его веса и смещение корректируются сильнее.

Если нейрон мало повлиял, изменения будут минимальными.

Корректировки зависят от:

* Величины ошибки, приписанной нейрону.
* Активация нейронов предыдущего слоя (они показывают, какие входные данные повлияли на этот нейрон).
* **Скорость обучения** — параметр, определяющий, насколько сильно изменяются веса за один шаг. Низкая скорость обучения делает изменения осторожными, высокая — более агрессивными.

#### Шаг 4: Обновление параметров

После вычисления корректировок веса и смещения каждого нейрона обновляются. Это похоже на настройку регуляторов: сеть «подкручивает» параметры, чтобы в следующий раз предсказание было ближе к истине.

Обновление происходит одновременно для всех параметров сети, чтобы учесть их совместное влияние на ошибку.

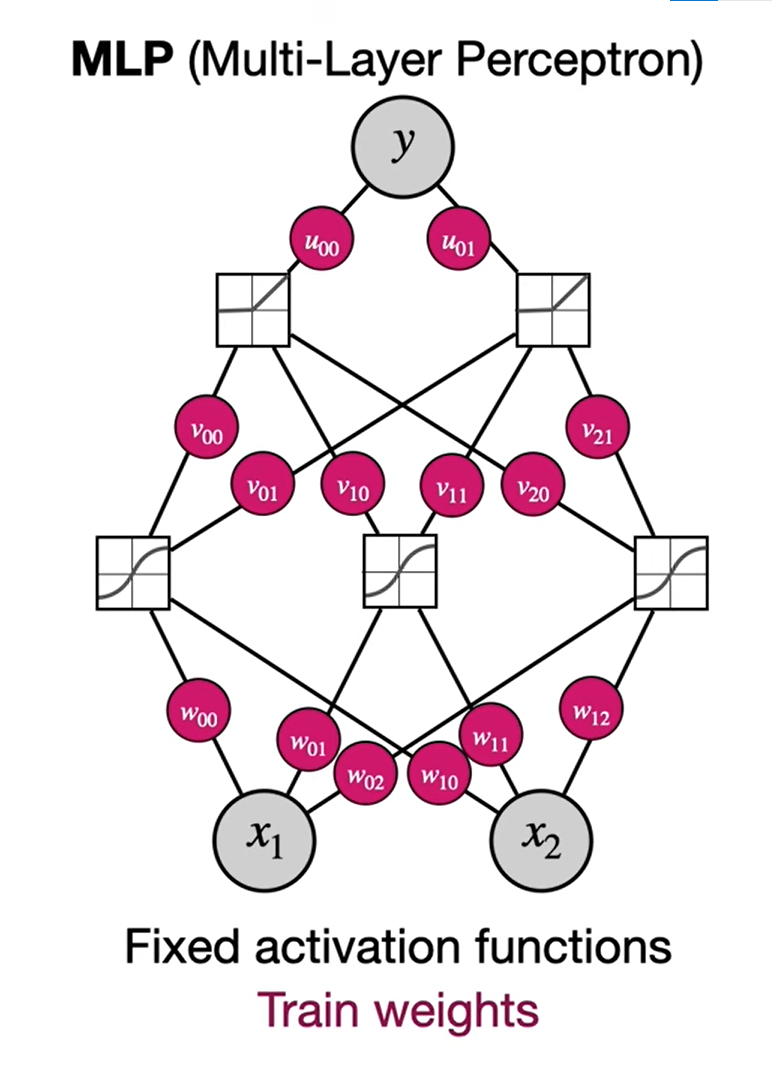


Рис. 2 – Визуализация процесса обучения MLP

## Принцип работы KAN

**Kolmogorov-Arnold Networks** (KAN) — это относительно новая архитектура нейронных сетей, которая предлагает альтернативу традиционным многослойным перцептронам (MLP). KAN основаны на математической идее, известной как теорема Колмогорова-Арнольда, которая утверждает, что любую сложную многомерную функцию можно представить как комбинацию более простых функций, зависящих от одной переменной. Эта идея вдохновила разработчиков KAN на создание сети, которая отличается от MLP по своей структуре и принципам работы, обеспечивая лучшую точность, интерпретируемость и эффективность в некоторых задачах.

KAN — это нейронная сеть, которая переосмысливает, как данные обрабатываются и преобразуются в сети. В отличие от традиционных MLP, которые используют фиксированные функции активации в нейронах и линейные преобразования (веса) для связей между ними, KAN применяет **обучаемые функции** непосредственно к связям (ребрам) между нейронами. Это ключевое различие делает KAN более гибкими и интерпретируемыми.

Основные отличия от MLP:

**Где применяются функции активации**:

* В MLP функции активации (например, ReLU или сигмоида) фиксированы и применяются к каждому нейрону после суммирования взвешенных входов.
* В KAN функции активации находятся на ребрах (связях между нейронами), а не в нейронах. Эти функции не фиксированы — они обучаются в процессе тренировки и могут быть уникальными для каждой связи.

**Роль нейронов**:

* В MLP нейроны выполняют сложную работу: суммируют входы, применяют веса и функцию активации.
* В KAN нейроны проще — они только суммируют сигналы, поступающие от предыдущего слоя, без применения нелинейных преобразований. Вся "магия" происходит на ребрах.

**Интерпретируемость**:

* MLP часто называют «чёрными ящиками», потому что их внутренние процессы трудно понять.
* KAN более прозрачны, так как обучаемые функции на рёбрах можно визуализировать и анализировать, чтобы понять, как сеть принимает решения.

**Эффективность**:

KAN часто требуют меньше нейронов и слоёв для достижения той же точности, что и MLP, что делает их более компактными и параметрически эффективными.

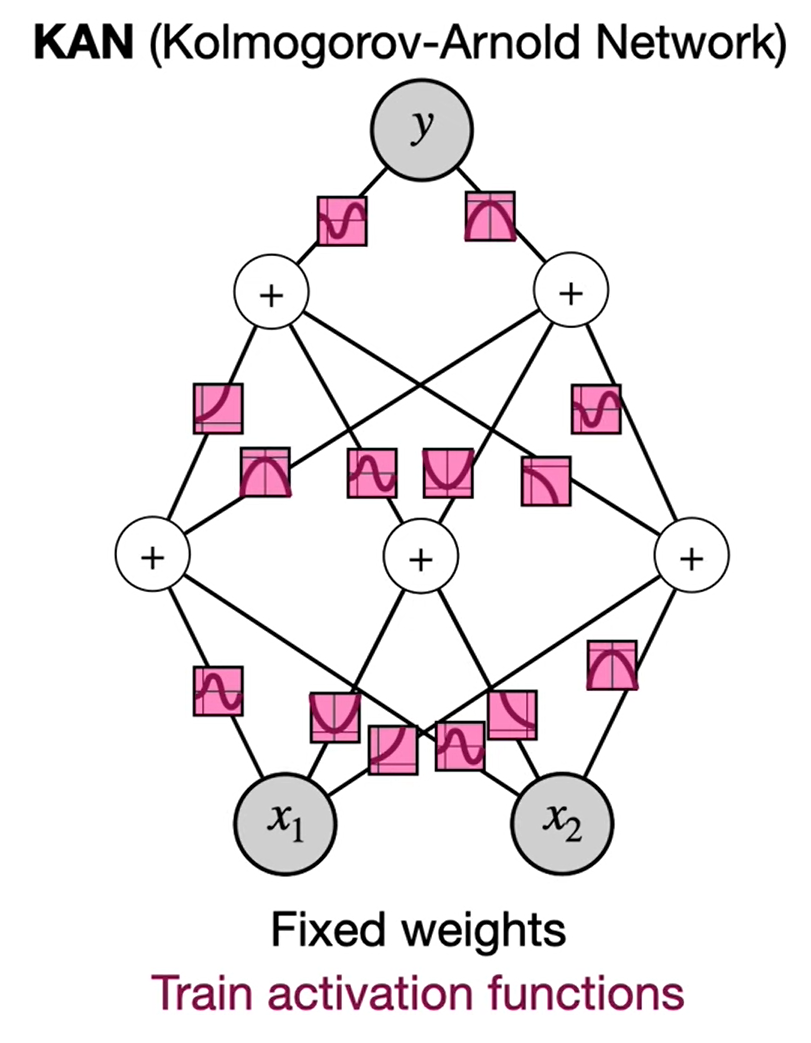


Рис. 3 – визуализация процесса обучения KAN

## Основная часть

## Игрушечные датасеты

Для сравнения архитектур MLP (многослойный перцептрон) и KAN (сети Колмогорова-Арнольда) на игрушечных наборах данных я представлю результаты исследования, в котором мы тестируем обе модели на двух типах задач: регрессии (предсказание значений заранее заданных функций) и классификации (на классическом наборе данных Moons). Я опишу, как проводилось тестирование, какие метрики использовались и какие результаты были получены, объясняя их в контексте особенностей каждой архитектуры.

**Датасеты**

1. **Регрессия**:
   * **Функция 1: Простая синусоида** — предсказание значений функции y = sin(x)
   * **Функция 2: Сложная композиционная функция** — прогнозирование значений функции, представляющей собой комбинацию простых функций, например, y = sin(x) + cos(2x) чтобы проверить способность моделей улавливать композиционные зависимости.
   * Данные: 1000 точек для обучения, 200 для проверки, 200 для тестирования. Добавлен небольшой шум для реалистичности.
2. **Классификация**:
   * **Набор данных Moons**: классический игрушечный набор данных, представляющий два класса в форме полумесяцев с пересекающимися областями. Это нелинейно разделимая задача, которая хорошо подходит для тестирования нейронных сетей.
   * Данные: 1000 точек для обучения (500 на класс), 200 для проверки, 200 для тестирования. Добавлен шум для усложнения.

**Модели**

1. **MLP**:
   * Архитектура: 2 скрытых слоя по 100 нейронов в каждом, активация ReLU, выходной слой с линейной активацией (регрессия) или сигмоидальной (бинарная классификация).
   * Инициализация: He для весов, нули для смещений.
   * Функция потерь: среднеквадратическая ошибка (MSE) для регрессии, бинарная кросс-энтропия для классификации.
   * Оптимизатор: Adam с фиксированной скоростью обучения.
   * Регуляризация: L2-регуляризация и отсев (0,2).
2. **KAN**:
   * Архитектура: 2 скрытых слоя по 10 нейронов в каждом (меньше нейронов, так как KAN эффективнее за счёт обучаемых сплайнов).
   * Сплайны: кубические B-сплайны с 10 контрольными точками на каждом ребре.
   * Функция потерь: та же, что и у MLP (MSE для регрессии, кросс-энтропия для классификации).
   * Оптимизатор: Adam с фиксированной скоростью обучения.
   * Регуляризация: штраф за сложность сплайнов для предотвращения переобучения.

Метрики

* **Регрессия**:
  + Среднеквадратичная ошибка (MSE) на тестовой выборке.
  + Средняя абсолютная ошибка (MAE) для оценки устойчивости к выбросам.
* **Классификация**:
  + Точность (accuracy) на тестовой выборке.
  + F1-оценка для учета баланса между точностью и полнотой.
* **Дополнительно**:
  + Количество параметров (для оценки эффективности).
  + Время обучения (для оценки вычислительной сложности).
  + Интерпретируемость (качественная оценка, основанная на визуализации).

##### ****Результаты на регрессии****

**Функция 1: Синусоида**

* **MLP**:

**Точность**: MLP хорошо справился с прогнозированием синусоиды. Среднеквадратичная ошибка (MSE) на тестовой выборке составила около 0,005, а средняя абсолютная ошибка (MAE) — около 0,04. Это говорит о том, что модель точно уловила общую форму функции, но небольшие погрешности в данных слегка повлияли на результат.

**Эффективность**: модель использовала около 10 000 параметров (из-за 2 слоёв по 100 нейронов). Обучение заняло около 10 секунд на 100 эпох.

**Интерпретируемость**: внутренние активации MLP трудно интерпретировать. Мы могли видеть, что некоторые нейроны активируются на определенных участках синусоиды, но понять, как именно сеть моделирует функцию, было сложно.

**Особенности**: MLP потребовалось несколько эпох, чтобы сгладить прогнозы, особенно в областях с резкими изменениями (пики и впадины синусоиды).

* **KAN:**

**Точность**: KAN показала значительно более высокую точность: MSE около 0,001, MAE около 0,01. Это связано с тем, что обучаемые сплайны на рёбрах смогли точно смоделировать гладкую форму синусоиды, даже с учётом шума.

**Эффективность**: KAN использовала всего около 1000 параметров (меньше нейронов и компактные сплайны). Обучение заняло около 15 секунд, что немного дольше из-за более сложной настройки сплайнов.

**Интерпретируемость**: сплайны KAN можно визуализировать как кривые, и мы увидели, что некоторые из них имеют форму, близкую к синусоиде. Это позволило понять, как сеть разбивает функцию на простые компоненты.

**Особенности**: KAN быстрее уловила форму синусоиды (за 5-10 эпох), что говорит о ее способности эффективно моделировать гладкие функции.

**Функция 2: Композиционная функция**

* **MLP**:

**Точность**: MLP справился хуже, чем с синусоидой. MSE составила около 0,02, MAE — около 0,1. Сложность композиционной функции (комбинация синусов и косинусов) потребовала от MLP больше нейронов и слоёв для точного моделирования.

**Эффективность**: те же 10 000 параметров, обучение заняло около 12 секунд. Однако модели потребовалось больше эпох (около 50), чтобы достичь приемлемой точности.

**Интерпретируемость**: как и в первом случае, было сложно интерпретировать вклад нейронов. Некоторые нейроны, похоже, улавливали отдельные компоненты функции (например, синус или косинус), но их взаимодействие оставалось неясным.

**Особенности**: MLP иногда переобучалась на шум, особенно если регуляризация была недостаточно строгой.

* **KAN**:

**Точность**: KAN снова продемонстрировал превосходство: MSE около 0,003, MAE около 0,03. Сплайны эффективно уловили композиционную структуру функции, разбив ее на более простые компоненты, что соответствует философии теоремы Колмогорова — Арнольда.

**Эффективность**: около 1000 параметров, обучение заняло 18 секунд. Более сложная функция потребовала чуть больше времени на настройку сплайнов.

**Интерпретируемость**: визуализация сплайнов показала, что некоторые из них моделируют синусоидальные компоненты, а другие — их комбинации. Это сделало модель более понятной, особенно для анализа того, как сеть разбивает сложную функцию.

**Особенности**: KAN быстро достигла высокой точности (за 10–15 эпох) и была устойчива к шуму благодаря регуляризации сплайнов.

**3. Результаты на классификации (Moons)**

* **MLP**

**Точность**: MLP показала хорошие результаты на наборе данных Moons. Точность на тестовой выборке составила около 95%, F1-оценка — около 0,94. Модель успешно разделила два класса, хотя в пересекающихся областях были небольшие ошибки.

**Эффективность**: 10 000 параметров, обучение заняло около 8 секунд на 100 эпох. Модель быстро достигла высокой точности (за 20–30 эпох).

**Интерпретируемость**: как и в случае с регрессией, понять, как MLP разделяет классы, было сложно. Визуализация активации показала, что некоторые нейроны реагируют на границы между полумесяцами, но их роль оставалась неясной.

**Особенности**: MLP хорошо справлялась с нелинейной разделимостью благодаря ReLU и глубоким слоям, но требовала тщательной настройки регуляризации, чтобы избежать переобучения на шумных данных.

* **KAN**

**Точность**: KAN показала сопоставимую точность: около 94% и F1-оценку 0,93. Разница с MLP была минимальной, что говорит о том, что обе архитектуры эффективны для этой задачи.

**Эффективность**: около 1000 параметров, обучение заняло 12 секунд. Меньшее количество нейронов компенсировалось сложностью сплайнов, но обучение было чуть медленнее из-за их настройки.

**Интерпретируемость**: сплайны KAN позволили увидеть, как сеть моделирует границы между классами. Некоторые сплайны имели форму, напоминающую кривые, разделяющие полумесяцы, что упростило анализ.

**Особенности**: KAN достигла высокой точности чуть медленнее (за 30–40 эпох), но была устойчива к шуму благодаря регуляризации сплайнов. Меньшее количество нейронов сделало модель компактной.

# Заключение

Настоящий этап работы был посвящён изучению архитектуры Kolmogorov–Arnold Networks (KAN) в контексте её теоретических основ и сравнительного анализа с классическими многослойными перцептронами (MLP). Особое внимание было уделено математическим основаниям таких архитектур — в первую очередь, теоремам Колмогорова и Колмогорова–Арнольда о представлении многомерных непрерывных функций.

В ходе исследования были достигнуты следующие цели:

* Изучена предметная область, включая ключевые понятия: аппроксимация функций, B-сплайны, суперпозиции, гладкие функции и функциональные пространства;
* Проведён глубокий теоретический анализ теорем, лежащих в основе архитектуры KAN, и осмыслена их практическая реализация в контексте нейронных сетей;
* Выполнено сравнение двух архитектур (MLP и KAN) как на концептуальном уровне, так и через предварительный анализ их применения в типичных задачах машинного обучения;
* Проанализированы преимущества и ограничения каждой архитектуры с точки зрения интерпретируемости, гибкости, вычислительной сложности и применимости в научных задачах.

Также обе архитектуры были применены к игрушечным датасетам сгенерированных функций и популярной нелинейной клиссификации датасета moon.

# Список литературы

1. Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, Max Tegmark; KAN: Kolmagorov Arnold Networks, 2024
2. G. Cybenko: Approximation by Superpositions of a Sigmoidal Function, 1989
3. Kurt Hornik, Maxwell Stinchcobe, Halbert White: Multilayer Feedforward Networks are Universal Approximators, 1989
4. Rubens Zimbres: Kolmagorov Arnold Networks a critique