Обзор архитектуры KAN: Kolmogorov-Arnold Networks

Александров Кирилл

26 апреля 2025 г.

Аннотация

В данной работе представлен обзор новой архитектуры нейронных сетей KAN (Kolmogorov-Arnold Networks), основанной на теореме Колмогорова-Арнольда о представлении многомерных функций. Ссылку на оригинальную статью можно найти [здесь].Обзор будет полезен для первичного знакомства с архитектурой и ее возможностями, потому что в нем рассмотрены основные принципы работы, математическое обоснование и сравнение с классическими MLP-сетями.Этот обзор включает в себя много примеров, реализацию которых можно найти [7] и [8].

1 Мотивация

В основе традиционных полносвязных нейронных сетей (MLP) лежит универсальная теорема аппроксимации (UAT). Авторы статьи предложили альтернативный подход, основанный на теореме Колмогорова-Арнольда.

Теорема 1.1 (Колмогорова-Арнольда)**.** *Каждая непрерывная функция многих переменных может быть представлена в виде суперпозиции непрерывных функций одной переменной.*

На основе этой теоремы была разработана новая архитектура нейронных сетей KAN (Kolmogorov-Arnold Networks).

2 Основные идеи

KAN обладает следующими ключевыми особенностями:

1. Обучаемые функции активации вместо фиксированных
2. Использование [B-сплайнов] для представления функций активации
3. Замена матрицы весов на матрицу функций активации

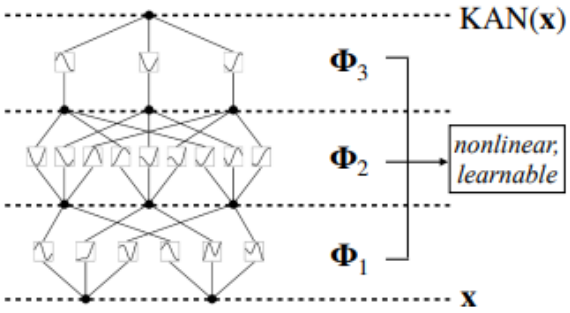

Рис. 1: Схема архитектуры KAN

4. Каждый нейрон получает сумму входных сигналов, предназначенных только для него
5. Обучаемые нелинейные преобразования между слоями

3 Реализация

3.1 Функции активации

Функции активации представляются в виде:

$$\phi(x) = w_1 b(x) + w_2 \text{spline}(x) \tag{1}$$

где:

- $b(x)$ - базисная функция (например, SiLU: $b(x) = x/(1 + e^{-x})$)
- $\text{spline}(x) = \sum_i c_i B_i(x)$ - сплайн-аппроксимация
- $B_i(x)$ - B-сплайны
- c_i - обучаемые параметры

3.2 Архитектура сети

Сеть характеризуется:

- L - глубиной сети
- n_i - шириной i-го слоя
- $[n_0, n_1, ..., n_L]$ - структурой сети

Обозначение: KAN$[n_0, n_1, ..., n_L]$. Например, KAN[2,1,1] - сеть с двумя входами и одним выходом и одним скрытым слоем.

3.3 Математическая формализация

Активация нейрона $(l+1, j)$:

$$x_{l+1,j} = \sum_{i=1}^{n_l} \phi_{l,j,i}(x_{l,i}), \quad j = 1, ..., n_{l+1} \tag{2}$$

В матричной форме:

$$\mathbf{x}_{l+1} = \Phi_l \mathbf{x}_l \tag{3}$$

где Φ_l - матрица функций активации.

Полное преобразование:

$$\text{KAN}(\mathbf{x}) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_0)\mathbf{x} \tag{4}$$

4 Теоретическое обоснование

Авторы статьи доказали следущюю теорему, которая обосновывает подход используемый в архитектуре.

Теорема 4.1 (Теория аппроксимации для KAN (KAT)). *Пусть* $\mathbf{x} = (x_1, ..., x_n)$. *Если функция* $f(\mathbf{x})$ *представима в виде:*

$$f = (\Phi_{L-1} \circ \cdots \circ \Phi_0)\mathbf{x} \tag{5}$$

где $\Phi_{l,i,j}$ $(k+1)$ *раз непрерывно дифференцируемы, то существуют:*

- *Константа* C*, зависящая от* f
- *B-сплайны* $\Phi_{l,i,j}^G$ *порядка* k *на сетке размера* G

такие что для любого $0 \leq m \leq k$*:*

$$\|f - (\Phi_{L-1}^G \circ \cdots \circ \Phi_0^G)\mathbf{x}\|_{C^m} \leq CG^{-k-1+m} \tag{6}$$

где C^m*-норма:*

$$\|g\|_{C^m} = \max_{|\beta| \leq m} \sup_{\mathbf{x} \in [0,1]^n} |D^\beta g(\mathbf{x})| \tag{7}$$

5 Тестирование

В данной секции приведены тесты KAN на классических задачах машинного и глубокого обучения.

5.1 Датасеты

Тестирование KAN проводилось на следующих классических задачах машинного обучения:

5.1.1 California Housing Dataset

- Источник: [4]
- Размер: 20,640 samples (8 признаков + целевая цена)
- Признаки:
  - MedInc (средний доход)
  - HouseAge (возраст дома)
  - AveRooms (комнаты)
  - AveBedrms (спальни)
  - Population (население)
  - AveOccup (жильцы)
  - Latitude, Longitude
- Цель: Предсказание медианной стоимости дома (в сотнях тысяч $)

Модель: `KAN(width=[8,10,1], grid=3, k=3)`
Результат: MAE (mean absolute error) = 0.37
Для сравнения: трехслойная MLP показывает в среднем MAE = 0.4.

5.1.2 Flight Delays Dataset

- Источник: [5]
- Содержание: Информация о задержках рейсов (авиакомпании, аэропорты, время вылета/прилёта, причины задержек и т.д.)
- Целевая переменная: Бинарная классификация (Задержка (1) / Нет задержки (0), если задержка > 15 минут)

Модель: `KAN(width=[11,13,2], grid=4, k=3, seed=2024)`
Результат: Accuracy = 0.83 (доля правильных ответов) Но в данной задаче происходит предсказание целевой переменной, у которой один класс представлен сильно больше чем другой (рейсы редко задерживают), поэтому метрика Accuracy нерелевантна. Будем использовать Presion (доля предсказанных положительных примеров действительно являются положительными) , здесь к сожалению модель работает чуть лучше случайного классификатора (presion = 0.55). В то время как Логистическая регрессия(Линейнная модель, в обучении которой используется loss-функция бинарной кросс-энтропии) показывает presion(0.99)

5.1.3 Exercise Detection Dataset

- Источник: [6]
- Содержание: Коллекция видео упражнений (отжимания, прыжки, подтягивания, приседания) с покадровым анализом углов между ориентирами
- Признаки: Углы плеч, локтей, бедер, коленных и лодыжечных суставов
- Цель: Классификация типа упражнения

Модель: `KAN(width=[10,15,5], grid=5, k=3, seed=2024)`
Результат: Accuracy = 0.8 В это же время четырех-слойная нейронная сеть показывает accuracy = 0.92.

5.2 Интерполяция функций

5.2.1 Пример 1: $f(x_1, x_2) = exp(sin(x_1) - (x_2)^2)$

Модель: `KAN(width=[2,1,1], grid=grids[i], k=k, seed=0)`

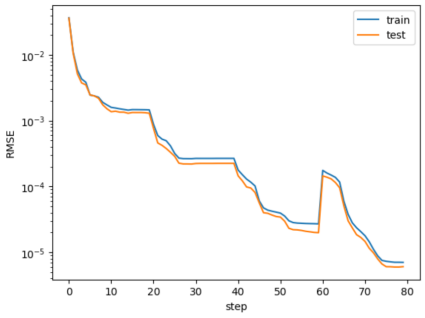

Рис. 2:

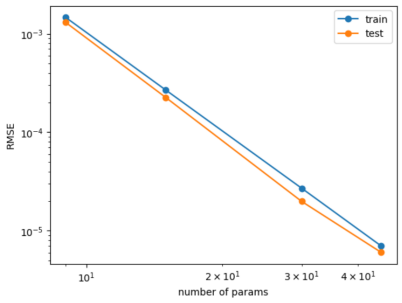

Рис. 3:

В первом эксперименте обучалась модель, у которой каждые 20 итераций изменялся параметр G (количество обучаемых параметров функции активации). На Рис.2 видно, как меняется RMSE(корень из среднеквадратичной ошибки) с увеличением параметра G. Также на Рис.3 можно наблюдать, как изменяется ошибка в зависимости от количетва праметров модели. Данный тест показывает то, как KAN реагирует на изменение сетки у B-сплайнов.

Во втором эксперементе обучается модель, у которой меняется не парметр G, а степень B-сплайнов. На Рис.4 и Рис.5 можно видеть что после

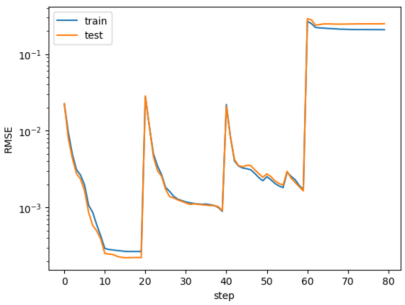

Рис. 4:

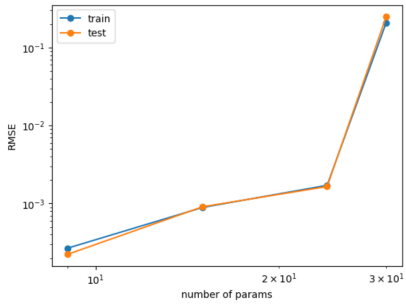

Рис. 5:

каждых 20 итераций ошибка подскакивает и после обучения результат становится хуже, чем при использовании B-сплайнов меньшей степени.
Такие резульаты связаны с тем, что при обновлении праметра G, мы дообучаем модель, а при обновлении параметра k, мы обучаем модель заново. Поэтому и происходят характерные скачки. Из этого можно сделат вывод о том, что при обучении модели лучше колибравать параметр G, а не k.

5.3 Вывод

KAN были обучены на разных датасетах, с различными особенностями представления данных и разными типами задач (регрессия, бинарная классификация, многоклассовая классификация). Архитектура показала сравнительно хорошие результаты, что демонстрирует ее способность к обучению. Однако KAN существенно уступает по скорости обучения из-за большого количества параметров при одинаковом количестве слоев, по сравнению с традиционными MLP. Остальные различия архитектур будут разобраны в седущей секции.

6 Сравнение KAN и MLP

6.1 Интерполяция функций

В секции тестирование мы убедились, что сеть KAN способна решать различные задачи классического машинного обучения. Но во всех приведенных выше прмерах она существенно уступала MLP моделям той же глубины по времени. Это происходит из-за того, что у KANов больше параметров. Так для обычной нейронной сети у которой глубина L и количество нейроннов на каждом слое равно N, количество обучаемых праметров равно. $O(N^2L)$. В тоже время KAN со схожей архитектурой имеет $O(N^2LG)$ параметров.
Вернемся к примеру 5.2.1 и обучим MLP модель на тех же данных. Жирные точки на графике разделяют модели с разным количеством параметров. По Рис.6 отчетливо видно, что классическая нейронная сеть с 1000 парамет-

ров имеет RMSE больше чем 0.1. В то время как KAN ,буквально с 10 параметрами, имеет ошибку чуть большую 0.001 (Рис.3) Данный пример показывает превосходсво KANов в предсказании функциональных зависимостей.

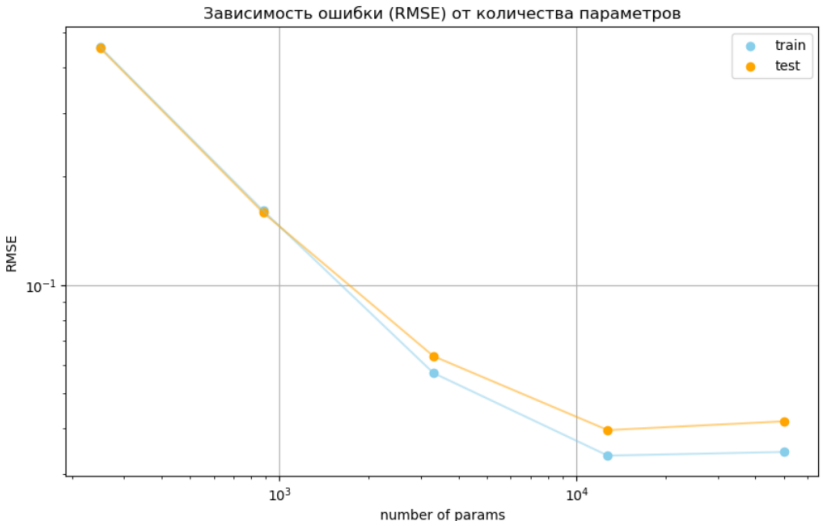

Рис. 6:

6.2 Решение дифференциальных уравнений

В прошлой секции мы поняли,что KANы довольно хорошо справляются с задачами интерполяции функций, поэтому будет не лишним протестировать их и в решении дифференциальных уравнений. На этот раз раз сравним их с PINN моделями. PINN (Physics-informed neural networks)- это MLP модель про которую можно подробнее прочитать здесь [9].
Рассмторим следующее дифференциальное уравнение:

$$\begin{cases} \dfrac{dy}{dx} = -y, \\ y(0) = 1. \end{cases}$$

Обучим две модели PINN и KAN.

$y = e^{-x}$ решение данной системы. Как мы видим по Рис.7 обе модели отлично справились с поставленной задачей. Но KANы решают ее существенно дольше.

Далее предлагается рассмотреть графики зависимости ошибки от эпохи обучения Рис.8. Как можно видеть, KANы достигают ошибки равной 10^{-3} за чуть больше чем 500 эпох обучения,а MLP модель приближается к этой

Характеристика	KAN	MLP
Глубина(L)	3	3
Количество параметров	15	18
Время обучения	5сек	50сек
Ошибка диффернциального Уравнения	0.000216	0.001318

Таблица 1: Сравнение KAN и MLP

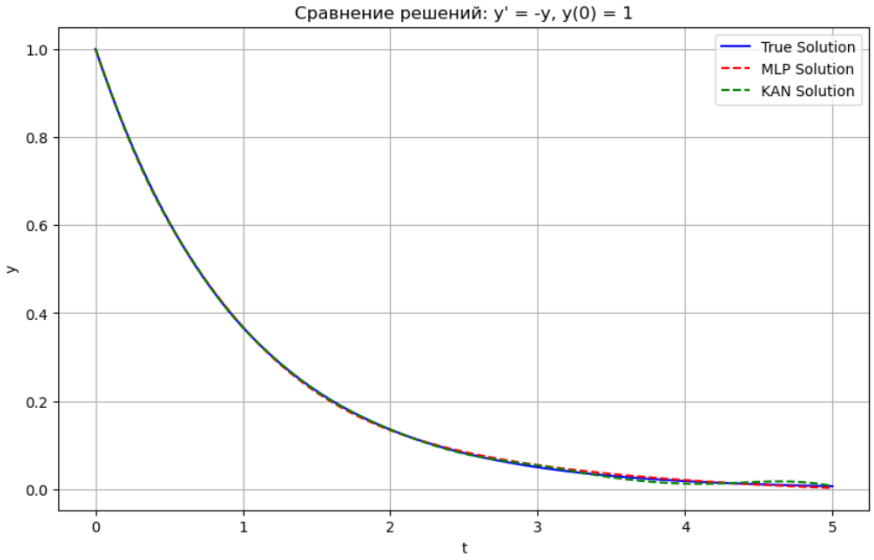

Рис. 7:

отметке только за 2500 эпох. Из этого можно сделать вывод о том, что KANы в данной задаче можно тренировать в пять раз меньше и такой подход не испортит результата, поэтому время обучения можно сократить с 50сек до 10сек.

6.3 Итог

По итогу, сравнение KAN и MLP можно описать Таблицей.2. KANы долго обучаются, но при этом могут выдавть точно такой же результат за меньшее количество эпох или же при меньшем количестве параметров. Но самым главным преимуществом новой архитектуры является интерпретируемость, про нее мы поговорим в следующей части.

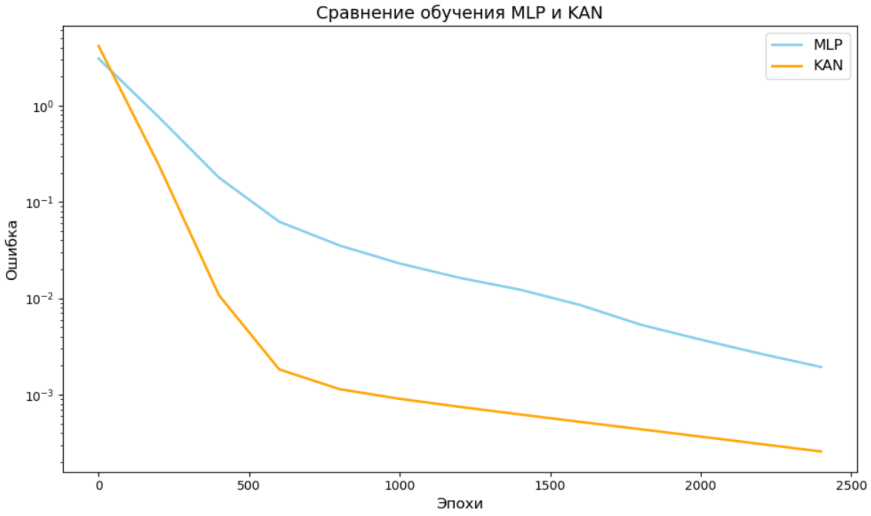

Рис. 8:

Характеристика	KAN	MLP
Функции активации	Обучаемые	Фиксированные
Весовые параметры	Матрица функций	Матрица весов
Интерпретируемость	Высокая	Низкая
Скорость обучения	Медленнее	Быстрее
Требуемая глубина	Меньше	Больше

Таблица 2: Сравнение KAN и MLP

7 Интерпретация результатов

Сейчас в машинном обучении глубокие сети играют ключевую роль, так как они довольно легко маштабируются и решают большой класс задач. Однако у подхода используемого в глубоком обучении есть свои недостатки. Мы не знаем почему модели выдают тот или иной ответ, мы можем смотреть только на результат. Но понимание вывода модели , может быть очень выжным. Например, врачам наобходимо видеть на основе чего модель приняла такое решение. Поэтому в машинном обучении появились задачи интепретации результата. Сети Колмогорова-Арнольда, благодаря своей архитектуре, могут решать задачу символьной регрессии (предсказывание символьного вида функциональной зависимости. Это может быть полезно в:

- Получении интерпретируемых формул вместо “чёрного ящика”. Данная способность нейронных сетей очень важна в медицине, где цена ошибки очень велика, и врачу важно понимать, почему машина выдала именно такой ответ.

- Поиске аналитических зависимостей в экспериментальных данных (физика, химия, биология).Например, определение функции плотности случайной величины.

Проведем еще один эксперемент для примера 6.2. Так как мы знаем исходное решение дифференциального уравнения, будем обучать не KAN[1,3,1], а KAN[1,1]. После обучения мы можем посмотреть на функции активации. Для это нужно применить команду:

$$model.plot()$$

Мы получим следующий результат:

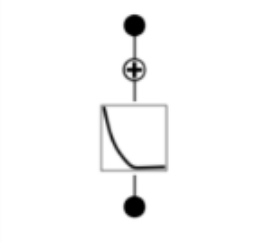

Рис. 9:

Очень похоже на e^{-x}.Также можно применить команду, которая может автоматически определять, какая символьная функция получилась:

$$model.suggest\_symbollic(l, m, k)$$

Функция, которая лучше всего описывает полученный результат, относительно R^2 критерия, это функция exp.В данной табличке выписываются названия функция с учетом коефицентов.

	function	fitting r2	r2 loss	complexity	complexity loss	total loss
0	0	0.000000	0.000014	0	0	0.000003
1	exp	0.994338	-7.461890	2	2	0.107622
2	1/x^2	0.984419	-6.003180	2	2	0.399364
3	x	0.667537	-1.588690	1	1	0.482262
4	1/x	0.976895	-5.435044	2	2	0.512991

Рис. 10:

Мы поняли, что если нам известен наперед вид функции(то как она может быть представлена по теореме Колмогорова-Арнольда), то можно предсказать ее символьный вид. Давайте рассморим еще один более сложный пример.

Пример: $f(x_1, x_2) = e^{sin(\pi x_1)+x_2^2}$
Будем обучать модель не зная истинного устройства фунцкии *model = KAN(width=[2,5,1], grid=5, k=3, seed=1)*

1. Первый этап обучения Рис.11.

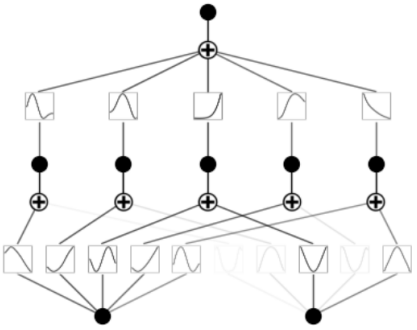

Рис. 11:

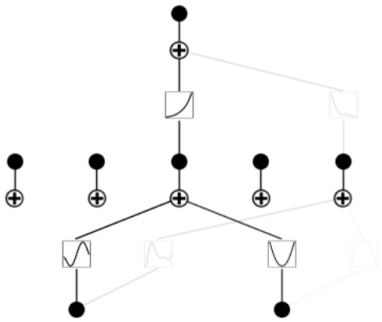

Рис. 12:

2. Второй этап обучения Рис.12. Добавляем дополнительные параметры *lamb* и *lamb_ coef*. Первый отвечает за дополнительный штраф модели, а второй штрафует если обучаемые парметры слишком большие. Такие методы регуляризации позволяют добиться хорошего результата.

3. Последний шаг. Как мы видим, некоторые нейроны более якркие чем другие. Чем тусклее нейрон, тем меньше вклада он дает в общую сумму. Таким образом, применив команду *model.prune()* мы избавимся от слабых нейронов и получим следующий результат:

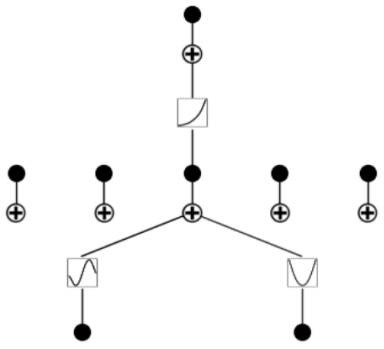

Рис. 13:

Каждая из оставшихся функций активации, соответствует своей функции в исходной формуле.

Как видно из данного примера, KANы могут отлично работать даже при условии того, что мы не знаем истинную конструкцию функции.

Итог: В этом блоке мы убедились, что KANы могут решать задачу интепретации результата, что очень важно в современном машинном обучении. Но в подходе примененном в последнем примере есть один нюанс. Заранее наверняка неизвестно, какие методы регуляризации необходимо применять, так в более сложных случаях не всегда удавалось получать успешный результат соответвующий реальности. Способность KANов быть интепретируемыми является большим плюсом.

8 Заключение

KAN представляет собой перспективную архитектуру, обладающую:

- Способностью к интерпретации
- Потенциалом для эффективного обучения
- Теоретическим обоснованием на основе теоремы Колмогорова-Арнольда

Можно смело говорить, что KANы хорошо решают задачи связанные с предсказанием функциольных зависимостей. Однако данная архитектура обладает рядом особенностей, которые могут сильно ограничивать область применения, поэтому можно выделить следующие основные направления дальнейших исследований:

- Оптимизация скорости обучения
- Применение в различных задачах
- Развитие теории аппроксимации
- Улучшение интерпретации

9 Ссылки

1. Оригинальная статья: `https://arxiv.org/abs/2404.19756`
2. Теория B-сплайнов: `https://www.brnt.eu/phd/node11`
3. Теорема Колмогорова-Арнольда: [1]
4. California Housing Dataset: `https://www.kaggle.com/datasets/camnugent/california-housing-prices`
5. Flight Delays Dataset: `https://www.kaggle.com/datasets/usdot/flight-delays`

6. Exercise Detection Dataset: https://www.kaggle.com/datasets/mrigaankjaswal/exercise-detection-dataset
7. https://github.com/212-Aleksandrov-2023/KAN-neural-network
8. https://github.com/KindXiaoming/pykan/tree/master/tutorials
9. https://habr.com/ru/articles/829090/

Список литературы

[1] A.N. Kolmogorov. *On the representation of continuous functions of several variables as superpositions of continuous functions of a smaller number of variables*. Dokl. Akad. Nauk SSSR, 108(2):179-182, 1956.