ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В последние годы в области глубокого обучения активно развиваются новые архитектуры нейронных сетей, способные преодолеть ограничения традиционных подходов. Одним из таких перспективных направлений являются сети Колмогорова-Арнольда (KAN), основанные на фундаментальной математической теореме о представлении непрерывных функций. В отличие от классических многослойных перцептронов с фиксированными функциями активации, KAN предлагают принципиально иной подход, используя адаптивные и обучаемые нелинейные преобразования на каждом ребре сети. Это обеспечивает им ряд существенных преимуществ, включая повышенную гибкость модели, лучшую способность к аппроксимации сложных зависимостей, а также более высокую интерпретируемость результатов.

KAN демонстрируют особую эффективность при работе с многомерными данными и сложными нелинейными задачами, такими как решение дифференциальных уравнений или непрерывное обучение. Их архитектура позволяет наглядно анализировать вклады отдельных компонентов, что открывает новые возможности для понимания внутренней структуры данных. Однако эти преимущества сопровождаются определенными компромиссами - в частности, процесс обучения KAN требует значительно больше вычислительных ресурсов и времени по сравнению с традиционными MLP-сетями аналогичного размера.

Целью данной работы является комплексное исследование архитектуры KAN, анализ их ключевых особенностей и сравнительная оценка эффективности. В работе рассматриваются как теоретические основы данного подхода, так и его практические реализации, позволяющие оценить перспективы применения KAN в современных задачах машинного обучения. Особое внимание уделяется сравнению производительности KAN с классическими нейросетевыми архитектурами.

ГЛАВА 1

Сети Колмогорова-Арнольда (KAN)

1.1 Теорема Колмогорова-Арнольда

Архитектура сети KAN опирается на теорему представления, сформулированную Колмогоровым и Арнольдом в 1957 году.

Теорема Колмогорова-Арнольда, на которой построена архитектура сети, утверждает, что любая многомерная непрерывная функция может быть представлена в виде суперпозиции непрерывных функций одной переменной. [3]

$$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right).$$

Рисунок 1.1 Представление многомерной функции

1.1.1 B-сплайны

Этими более простыми одномерными функциями являются сплайны - функции, позволяющие создавать гладкие кривые, имея набор точек. Сплайны позволяют гибко изменять форму кривой, обеспечивая непрерывность и плавность между соседними сегментами.

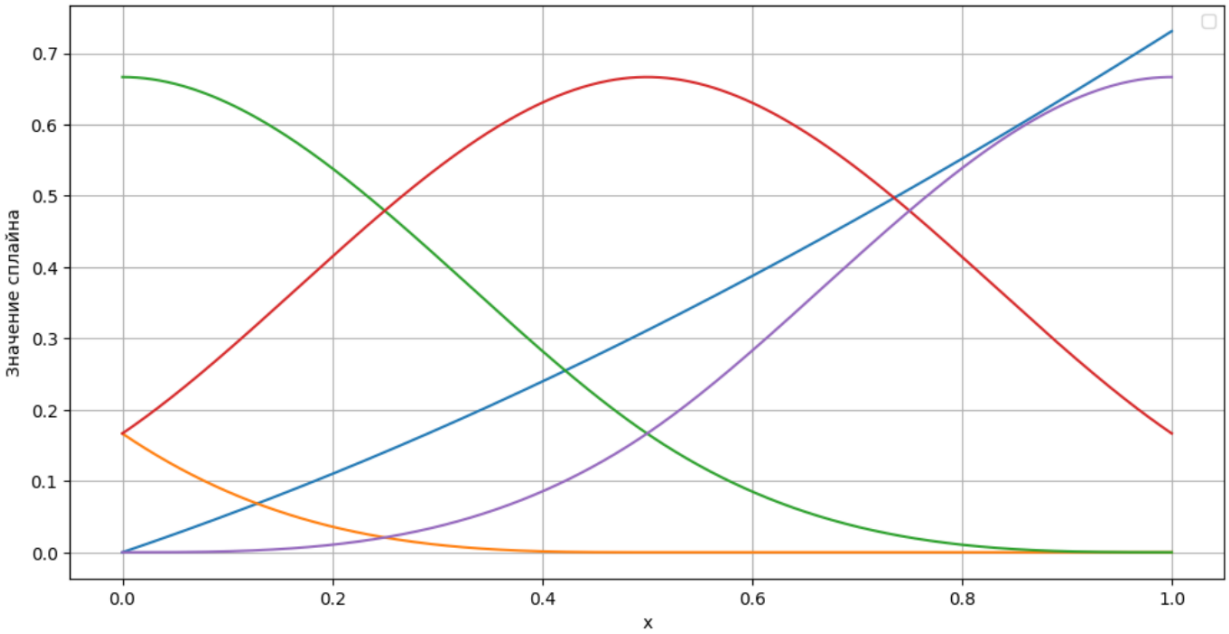

Рисунок 1.2 B-сплайны

Кривая строится интерполяцией или аппроксимацией между точками с помощью B-сплайнов.

Их главная особенность заключается в том, что они позволяют с высокой точностью приближать самые разные математические функции, от простых полиномов до сложных тригонометрических и экспоненциальных зависимостей. Это возможно благодаря тому, что любая гладкая функция может быть представлена в виде линейной комбинации B-сплайнов, причем точность такого представления зависит от степени сплайнов и распределения узлов.

Степень B-сплайна (p) определяет его основные свойства. Например, сплайны первой степени — это обычные ломаные линии, кубические сплайны (p=3) формируют гладкие кривые с непрерывной второй производной. На практике чаще всего используют кубические B-сплайны, так как они обеспечивают хороший баланс между точностью аппроксимации и вычислительной сложностью.

1.1.2 Проклятие размерности MLP

Проклятие размерности (curse of dimensionality) — это термин, введенный Ричардом Беллманом в 1957 году, описывающий различные феномены, которые возникают при анализе и организации данных в пространствах высокой размерности. То есть в 1-мерном или 2-мерном пространстве мы можем огра-

ничиться несколькими сотнями точек, в тоже время в k-мерном пространстве, чтобы сохранить плотность данных нужно M^k точек, где M - количество точек на одно измерение. Очевидно, насколько сильно в таком случае должно увеличиться количество тренировочных данных. [4, с. 250]

Решением этой проблемы выступают разные методы такие, как: метод главных компонент, другие архитектуры, но также и KAN.

Согласно теореме Колмогорова-Арнольда, если мы можем найти все "одномерные функции" – нам больше не нужно переходить на высокие размерности – проблема проклятия размерности решена.

1.2 Архитектура KAN

Ученые из Массачусетского технологического института, Калифорнийского технологического института и Института ИИ и фундаментальных взаимодействий представили в 2024 году новую архитектуру нейронной сети Kolmogorov-Arnold Networks (KAN).

В отличие от MLP, где преобразование данных определяется фиксированной структурой (линейные веса + нелинейные активации), KAN использует принципиально иной подход. В этой архитектуре каждое соединение между нейронами представляет собой параметризованную сплайн-функцию, а не просто весовой коэффициент. Это позволяет всей сети адаптивно подстраивать характер преобразования данных на каждом этапе обработки. [2].

1.2.1 Механизм работы

Особенность нейрона Колмогорова-Арнольда (KAN) заключается в краевой функции ϕ, определяемой как линейная комбинация функций b_k:

$$x_i^{\text{mid}} = \phi\left(x_i^{\text{in}}\right) := \sum_{k=1}^{K} w_k^i b_k\left(x_i^{\text{in}}\right) \tag{1.1}$$

Следовательно, производная записывается так:

$$\frac{\mathrm{d}x_i^{\mathrm{mid}}}{\mathrm{d}w_k^i} = b_k\left(x_i^{\mathrm{in}}\right), \tag{1.2}$$

$$\frac{\mathrm{d}x_i^{\mathrm{mid}}}{\mathrm{d}x_i^{\mathrm{in}}} = \sum_{k=1}^{K} w_k^i b_k'\left(x_i^{\mathrm{in}}\right) \tag{1.3}$$

В (1.1) предполагается, что функции b_k должны быть B-сплайновыми, за исключением $b_1(x) := x/(1+e^{-x})$.

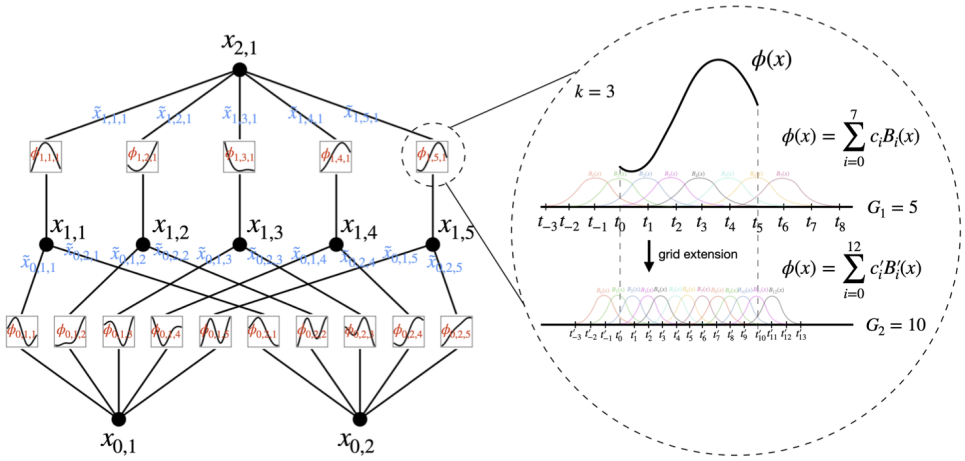

Рисунок 1.3 Схема прохождения по сети KAN

1.2.2 Обратное распространение ошибки в KAN

Функция «потерь» $\mathcal{L}$, описанная ниже, оценивает качество прогнозирования сети по отношению к истине. Наша цель — узнать значения весов и смещений $\{w, b\}$, минимизирующие потерю. Для этой цели мы оцениваем производную потери по параметрам и обновляем их с помощью градиентного спуска. Затем мы можем разделить эти производные на более мелкие части с помощью правила цепочки следующим образом:

Функция «потерь» $\mathcal{L}$ количественно оценивает расхождение между предсказаниями модели и эталонными значениями. Оптимизационная задача состоит в

habr.com: KAN: Kolmogorov–Arnold Networks

нахождении таких параметров $\{w, b\}$, при которых достигается минимум функции $\mathcal{L}$. Для решения этой задачи применяется метод градиентного спуска, требующий вычисления градиентов функции потерь по параметрам (в KAN bias=0):

$$\nabla \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial w}, \frac{\partial \mathcal{L}}{\partial b} \right) \tag{1.4}$$

Вычисление этих градиентов осуществляется благодаря цепному правилу (chain rule):

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w_j^i} = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}x^{\text{out}}} \frac{\mathrm{d}x^{\text{out}}}{\mathrm{d}w_j^i}, \qquad \forall\, i, j \tag{1.5}$$

Выход по весам и входу определяется аналогично:

$$\frac{\mathrm{d}x^{\text{out}}}{\mathrm{d}w_j^i} = \frac{\mathrm{d}x^{\text{out}}}{\mathrm{d}x_i^{\text{mid}}} \frac{\mathrm{d}x_i^{\text{mid}}}{\mathrm{d}w_j^i}, \qquad \forall\, i, j \tag{1.6}$$

$$\frac{\mathrm{d}x^{\text{out}}}{\mathrm{d}x_i^{\text{in}}} = \frac{\mathrm{d}x^{\text{out}}}{\mathrm{d}x_i^{\text{mid}}} \frac{\mathrm{d}x_i^{\text{mid}}}{\mathrm{d}x_i^{\text{in}}}, \qquad \forall\, i \tag{1.7}$$

$$x_i^{\text{mid}} = \phi\left(x_i^{\text{in}}\right) := \sum_{k=1}^{K} w_k^i f_k\left(x_i^{\text{in}}\right) \tag{1.8}$$

Значит производная будет:

$$\frac{\mathrm{d}x_i^{\text{mid}}}{\mathrm{d}w_k^i} = f_k\left(x_i^{\text{in}}\right), \quad \frac{\mathrm{d}x_i^{\text{mid}}}{x_i^{\text{in}}} = \sum_{k=1}^{K} w_k^i f_k'\left(x_i^{\text{in}}\right) \tag{1.9}$$

По цепному правилу $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}x_i^{\text{in}}}$ определяется:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}x_i^{\text{in}}} = \sum_{n=1}^{N} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}x_n^{\text{out}}} \frac{\mathrm{d}x_n^{\text{out}}}{\mathrm{d}x_i^{\text{in}}}, \quad \forall\, i \tag{1.10}$$

Прежде чем приступить к наложению слоев и построению полной сети прямого распространения, необходимо определить функцию потерь. Она позволяет оценить, насколько точно предсказания модели совпадают с реальными значениями, представленными в данных.

Обозначим y как выход последнего слоя, то есть предсказание модели. В

задачах регрессии y^{train} представляет собой действительный вектор, а функция потерь определяется как квадрат разницы между прогнозируемым и истинным значением(MSE).

$$\mathcal{L}^{\text{S}} := \mathcal{L}^{\text{S}}(\mathbf{y}, \mathbf{y}^{\text{train}}) = \sum_i \left(y_i - y_i^{\text{train}}\right)^2 \tag{1.11}$$

с соответствующей производной:

$$\frac{\mathrm{d}\mathcal{L}^{\text{S}}}{\mathrm{d}y_i} = 2\left(y_i - y_i^{\text{train}}\right), \qquad \forall\, i \tag{1.12}$$

Во время обучения сеть получает список пар вида $\{(\mathbf{x}^{\text{train}}, \mathbf{y}^{\text{train}})\}$, содержащих входные данные и их истинные значения.

При прямом проходе $\mathbf{x}^{\text{train}}$ поступает в первый слой, где каждый нейрон вычисляет промежуточные переменные $\{\mathbf{x}^{\text{mid}}, \mathbf{x}^{\text{out}}\}$, передает их дальше и определяет соответствующие производные. В результате формируется выходное значение y, после чего вычисляется функция потерь $\mathcal{L}$.

На этапе обратного распространения градиент функции потерь по входу слоя сначала инициализируется следующим образом:

$$\delta_i := \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y_i}, \quad \forall\, i \tag{1.13}$$

а затем обновляется на каждом слое в обратном порядке, от последнего к первому, через (1.10):

$$\delta_i \leftarrow \sum_{n=1}^{N} \delta_n \frac{\mathrm{d}x_n^{\text{out}}}{\mathrm{d}x_i^{\text{in}}}, \quad \forall\, i \tag{1.14}$$

Каждый нейрон слоя вычисляет градиент функции потерь по своим параметрам, где выражение $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}x^{\text{out}}}$ соответствует обратно распространяемому δ.

В завершение веса и смещения нейронов корректируются с использованием градиентного спуска:

$$w_j^i \leftarrow w_i^j - \epsilon \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w_j^i}, \quad \forall\, i, j \tag{1.15}$$

1.3 Сравнение эффективности KAN и MLP на синтетических данных

Для примера проведем сначала сравнительный анализ двух архитектур нейронных сетей - KAN и классического многослойного перцептрона (MLP) на синтетически сгенерированной функции: $f(x) = \sin(x^2) + \cos(x + 3)$.

Выбор именно такой тестовой функции обусловлен необходимостью проверить способность моделей:

- Аппроксимировать относительно сложные нелинейные зависимости
- Улавливать композицию различных математических операций

Особый интерес представляет сравнение скорости обучения и точности аппроксимации между двумя подходами.

В эксперименте были следующие параметры:

- Диапазон входных значений: $x \in [-5, 5]$
- Объем обучающей выборки: 1000 случайно распределенных точек
- структура MLP и KAN: 1 скрытый слой с 5 нейронами
- batch size = 50
- Критерий остановки: loss < 0.3

1.3.1 Результаты обучения

В ходе эксперимента были получены следующие показатели:

Таблица 1.1 Сравнительные характеристики моделей

Модель	KAN	MLP
Время обучения	3 с	37 с
Итерации до остановки	2	99
Финальная ошибка (MSE)	0.028	0.029

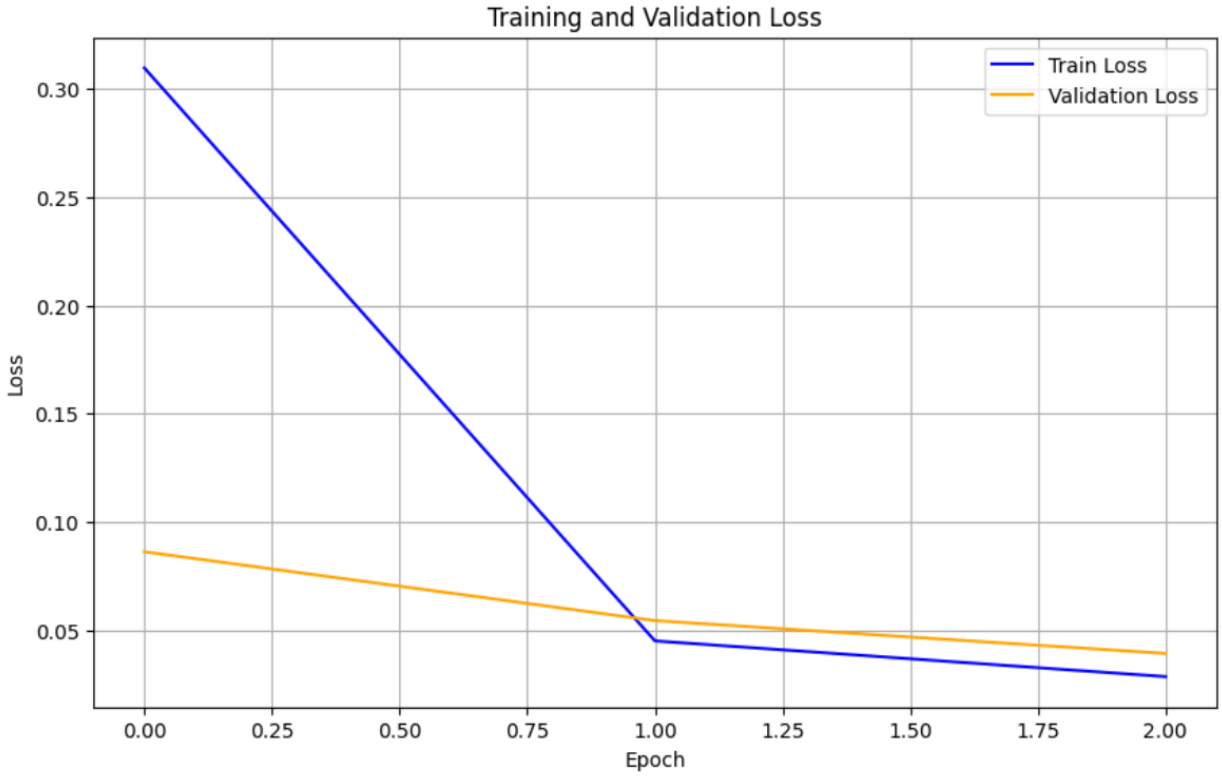

Рисунок 1.4 Функция потерь при обучении KAN

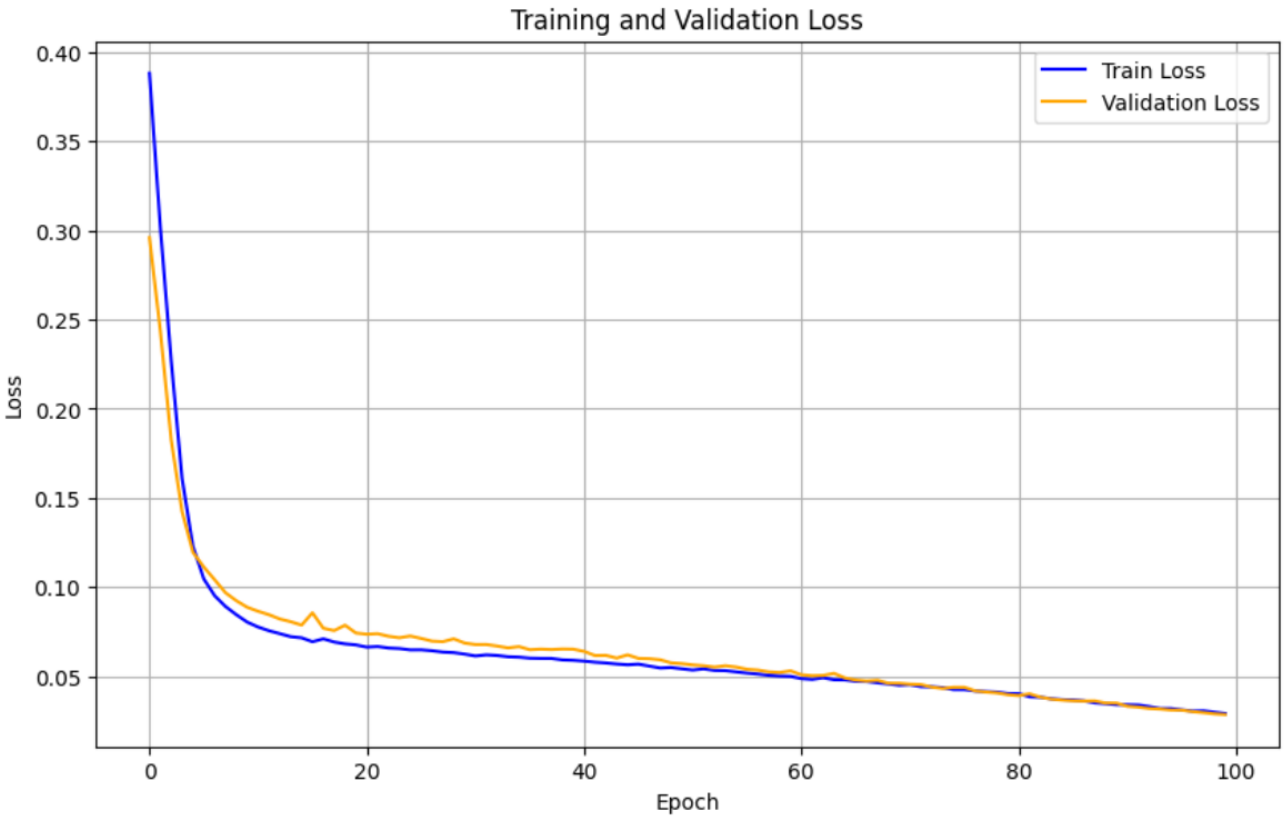

Рисунок 1.5 Функция потерь при обучении MLP

```
model_kan.train(x_train.reshape(-1,1), y_train.reshape(-1,1),
            x_val=x_val.reshape(-1,1), y_val=y_val.reshape(-1,1),
            eps=0.13, n_iter_max=500, loss_tol=0.030,
            decay_rate=0.995, min_lr=0.0001,
            patience=20, plateau_factor=0.5, batch_size=50)
```

```
  0%|                                        | 2/500 [00:03<13:54,  1.68s/it, train_loss: 0.028. Convergence attained!]
```

Рисунок 1.6 Процесс обучения KAN

```
model_perc.train(x_train.reshape(-1,1), y_train.reshape(-1,1),
            x_val=x_val.reshape(-1,1), y_val=y_val.reshape(-1,1),
            eps=0.13, n_iter_max=500, loss_tol=0.030,
            decay_rate=0.995, min_lr=0.0001,
            patience=20, plateau_factor=0.5, batch_size=50)
```

```
 20%|██████                                 | 99/500 [00:37<02:32,  2.63it/s, train_loss: 0.029. Convergence attained!]
```

Рисунок 1.7 Процесс обучения MLP

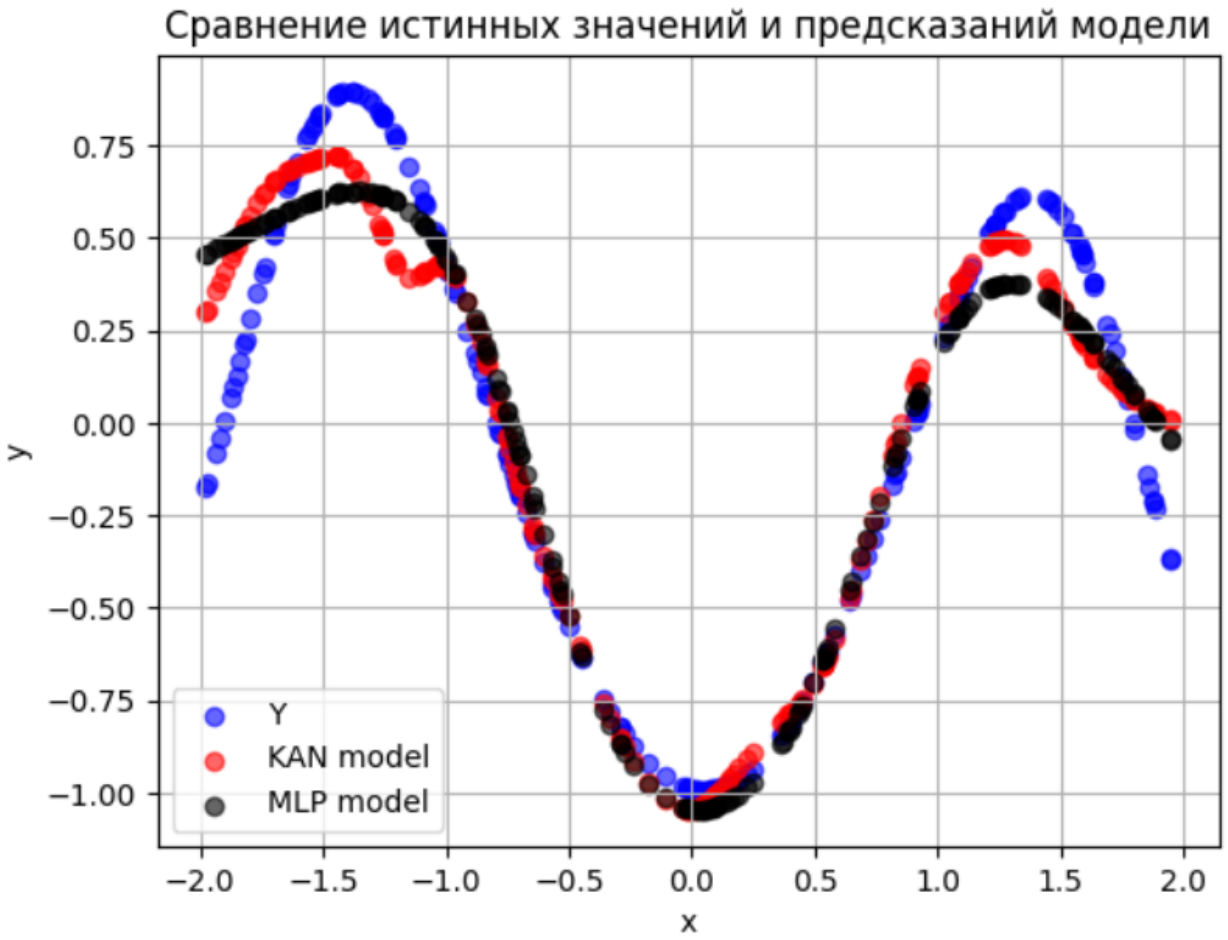

Рисунок 1.8 Сравнение истинных значений и предсказаний моделей

Согласно результатам:

- KAN обеспечивает практически идеальную аппроксимацию уже на 2-й итерации за счет B-сплайнов, которые более гибко аппроксимируют такие функции
- MLP требует значительного времени для выхода на сопоставимую точность

Проведенный эксперимент наглядно демонстрирует преимущества KAN-архитектуры при работе с гладкими аналитическими функциями. KAN существенно эффективнее MLP в задачах аппроксимации сложных функциональных зависимостей и лучше подходит для задач с ярко выраженными осциллирующими компонентами, в чем мы и убедились, за счет своей структуры состоящей из B-сплайнов, которыми можно приблизить такие и более сложные функции

за, буквально, несколько итераций. В то время как MLP будет вынужден комбинировать множество "ломаных"линий, для того, чтобы приблизиться к такому же результату

ГЛАВА 2

Прогнозирование плотности наножидкостей. Описание и предварительная обработка данных

2.1 Описание данных

Используемый в исследовании датасет является результатом работы Nanofluid Density Prediction, авторы которой собирали его по различным статьям, а затем опубликовали на Kaggle.

В данном исследовании рассматривается задача прогнозирования плотности гибридных наножидкостей (Hybrid Nanofluid Density Prediction) на основе комплекса параметров. Наножидкости представляют собой перспективные материалы, состоящие из базовой жидкости с добавлением наночастиц различных типов (оксиды металлов, углеродные наноструктуры и др.). Точное предсказание их плотности имеет критическое значение для:

- Оптимизации теплообменных систем
- Разработки новых охлаждающих жидкостей для электроники
- Создания эффективных термомагистралей в энергетике
- Медицинских применений (например, в системах направленной доставки лекарств)

Датасет содержит следующие ключевые параметры:
Категориальные признаки:

- Тип наночастиц (Nanoparticle type) - определяет физико-химические характеристики системы
- Базовая жидкость (Base fluid) - основной носитель наночастиц

Непрерывные признаки:

- Температура (°C)
- Объемная концентрация (ϕ)
- Плотность первичных наночастиц (ρ_1)
- Плотность вторичных наночастиц (ρ_2)
- Объем смеси первой частицы
- Объем смеси второй частицы

Целевой переменной является плотность наножидкости ($\mathrm{kg/m^3}$).

2.2 Предварительная обработка

Перед построением моделей был выполнен комплекс предобработки данных:

- удаление дубликатов
- преобразование категориальных признаков
- удаление линейно зависимых столбцов (Объемные соотношения смесей)
- данные были нормализованы с использованием метода Min-Max с масштабированием в диапазон $[-1, 1]$ для ускорения процесса сходимости

Для работы с нейросетевыми моделями категориальные переменные (тип наночастиц и базовая жидкость) были преобразованы методом one-hot encoding.

Это создало N новых бинарных признаков вместо исходных категориальных.

При обучении данные были поделены в соотношении 80/20:

- обучающая выборка (80%)
- валидационная (20%)

В связи с ограниченным объемом данных, валидационная выборка одновременно выполняла функцию тестовой. Такой подход, хотя и не является идеальным, был оправдан недостаточным количеством наблюдений для выделения отдельных валидационного и тестового наборов.

2.3 Анализ особенностей

Исходный датасет продемонстрировал несколько характерных особенностей:

при помощи матрицы корреляции была выявлена линейная зависимость между параметрами (Volume Mixture of Particle 1, Volume Mixture of Particle 2).

	Volume Concentration (ϕ)	Density of Nano Particle 1 (ρnp)	Density of Nano Particle 2 (ρnp)	Density of Base Fluid (ρbf)	Volume Mixture of Particle 1	Volume Mixture of Particle 2	Density (ρ)
Volume Concentration (ϕ)	1.000000	-0.042983	-0.270959	-0.273470	0.582202	-0.582202	0.283905
Density of Nano Particle 1 (ρnp)	-0.042983	1.000000	-0.398005	-0.248904	0.004921	-0.004921	-0.298498
Density of Nano Particle 2 (ρnp)	-0.270959	-0.398005	1.000000	0.498946	-0.521510	0.521510	0.170874
Density of Base Fluid (ρbf)	-0.273470	-0.248904	0.498946	1.000000	-0.401379	0.401379	0.773982
Volume Mixture of Particle 1	0.582202	0.004921	-0.521510	-0.401379	1.000000	-1.000000	-0.029027
Volume Mixture of Particle 2	-0.582202	-0.004921	0.521510	0.401379	-1.000000	1.000000	0.029027
Density (ρ)	0.283905	-0.298498	0.170874	0.773982	-0.029027	0.029027	1.000000

Рисунок 2.1 Матрица корреляции данных

Плотность базовой жидкости (Density of Base Fluid) и общая плотность (Density): корреляция 0.774 показывает сильную зависимость параметров.

ГЛАВА 3

Прогнозирование плотности наножидкостей.

3.1 Гибридная модель с KAN-нейронами в первом скрытом слое

В ходе исследования была выдвинута гипотеза о том, что использование чистых KAN-сетей для обработки бинарных категориальных данных, полученных после one-hot кодирования, является вычислительно неэффективным. Основное предположение заключается в том, что для бинарных входов $x_i \in \{0, 1\}$ сплайн-аппроксимация KAN избыточна, поскольку линейное преобразование способно точно выразить зависимость между входом и выходом без потери информации.

Теоретическое обоснование

Рассмотрим KAN-ребро с функцией активации $\phi\colon \mathbb{R} \to \mathbb{R}$. Для бинарного входа x_i преобразование определяется двумя значениями:

$$\phi(0) = a, \quad \phi(1) = b.$$

В этом случае любая непрерывная функция ϕ, удовлетворяющая данным условиям, может быть заменена линейной зависимостью:

$$\phi(x_i) = (b - a)x_i + a,$$

поскольку через две точки $(0, a)$ и $(1, b)$ можно провести единственную прямую. Следовательно, KAN-преобразование для бинарных данных вырождается в линейное.

в KAN, чаще всего,используется кубические сплайны для аппроксимации ϕ, что требует вычисления базисных функций.

Для бинарных входов достаточно одного линейного преобразования $w_i x_i + \beta_i$, что снижает вычислительную сложность с $O(C_{\text{spline}})$ до $O(1)$ на один вход.

Агрегация N бинарных признаков в KAN-слое:

$$\sum_{i=1}^{N} \phi(x_i) = \sum_{i=1}^{N} ((b_i - a_i)x_i + a_i) = \underbrace{\sum_{i=1}^{N}(b_i - a_i)x_i}_{\text{линейная комбинация}} + \underbrace{\sum_{i=1}^{N} a_i}_{\text{смещение}}$$

Данное выражение в точности соответствует выходу полносвязного слоя с весами $w_i = b_i - a_i$ и смещением $\beta = \sum a_i$.

Такая архитектура позволяет достичь значительного улучшения вычислительной эффективности нейронной сети при сохранении ключевых преимуществ подхода KAN для обработки непрерывных признаков. Замена сплайн-аппроксимации на линейные преобразования для бинарных входных данных обеспечивает существенное сокращение вычислительных затрат - сложность обработки одного бинарного признака снижается с $O(C_{\text{spline}})$ до $O(1)$, что особенно важно при работе с высокоразмерными категориальными данными. При этом для непрерывных переменных сохраняется возможность аппроксимации с помощью KAN-нейронов, поддерживая высокую выразительную способность модели.

Полученное решение демонстрирует баланс между производительностью и функциональностью: линейные преобразования эффективно обрабатывают бинарные признаки, тогда как KAN-слои обеспечивают необходимую гибкость для работы с непрерывными данными.

3.2 Реализация процесса обучения

3.2.1 Метод обучения

Обучение нейронной сети осуществляется с помощью **алгоритма градиентного спуска**, который минимизирует функцию потерь $\mathcal{L}$ путём итеративного обновления весовых коэффициентов. Основное правило обновления параметров:

$$w_{ij} \leftarrow w_{ij} - \epsilon \frac{\partial \mathcal{L}}{\partial w_{ij}} \tag{3.1}$$

где ϵ — скорость обучения.

Пакетная обработка

Реализованы три режима обучения:

- Полный градиентный спуск (batch mode) при $batch\_size = None$
- Стохастический градиентный спуск при $batch\_size = 1$
- Мини-пакетный режим при $1 < batch\_size < n\_samples$

Преимущества пакетной обработки:

- Эффективное использование вычислительных ресурсов
- Уменьшение дисперсии оценок градиента
- Ускорение процесса обучения

3.2.2 Адаптация скорости обучения

Экспоненциальное затухание

После каждой эпохи происходит обновление:

$$\epsilon_{new} = \max(\epsilon \cdot \gamma, \epsilon_{min}) \tag{3.2}$$

где $\gamma = 0.995$ — коэффициент затухания, $\epsilon_{min} = 10^{-4}$ — минимальное значение.

Ранняя остановка

При отсутствии улучшения функции потерь в течение $k = 30$ эпох:

$$\epsilon_{new} = \epsilon \cdot \alpha \tag{3.3}$$

где $\alpha = 0.5$ — коэффициент уменьшения.

3.2.3 Критерии остановки

Обучение прекращается при:

1. Достижении максимального числа итераций T_{max}

2. Выполнении условия $\mathcal{L} < \mathcal{L}_{tol}$

3. Отсутствии улучшений в течение заданного числа эпох

3.3 Результаты обучения

Сравнение производительности гибридной архитектуры с KAN-нейронами с классическим MLP показало следующие результаты:

```
model_2.train(features_train, labels_train.reshape(-1,1),
            x_val=features_test, y_val=labels_test.reshape(-1,1),
            eps=0.1, n_iter_max=500,loss_tol=0.015,
            decay_rate=0.995,
            min_lr=0.00001,
            patience=20,
            plateau_factor=0.5, batch_size=38)
```

```
 30%|███████▌                  | 151/500 [00:55<02:08,  2.72it/s, train_loss: 0.015. Convergence attained!]
```

Рисунок 3.1 MLP

```
model_hybrid.train(np.hstack((features_train[:,18:], features_train[:,:18])), labels_train.reshape(-1,1),
            x_val=np.hstack((features_test[:,18:], features_test[:,:18])), y_val=labels_test.reshape(-1,1),
            eps=0.1, n_iter_max=500,loss_tol=0.015,
            decay_rate=0.997,
            min_lr=0.00001,
            patience=20,
            plateau_factor=0.5, batch_size=38)
```

```
 10%|██▌                       | 50/500 [01:33<14:04,  1.88s/it, train_loss: 0.015. Convergence attained!]
```

Рисунок 3.2 Гибридная модель с KAN-нейронами

Традиционный перцептрон продемонстрировал более быстрое время обучения, однако ему потребовалось большее количество эпох.

Гибридная модель с KAN-нейронами показала:

- Меньшее количество итераций
- Более длительное время сходимости

Обе модели использовали идентичные гиперпараметры:

- Размер батча = 38
- Минимальная ошибка до остановки = 0.007

Основной вывод

Хотя гибридный подход сохраняет структурные преимущества KAN-нейронов для обработки непрерывных признаков, классический MLP в текущей реализации оказался более эффективным с точки зрения скорости обучения. [1]

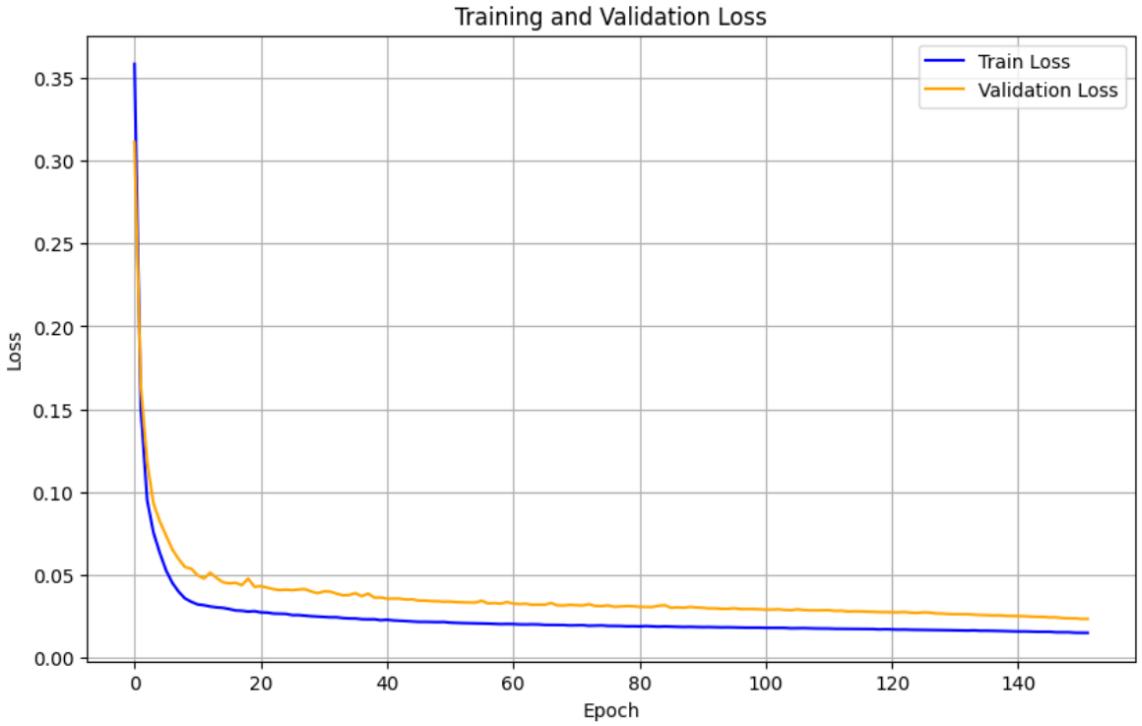

Рисунок 3.3 Функция потерь при обучении MLP

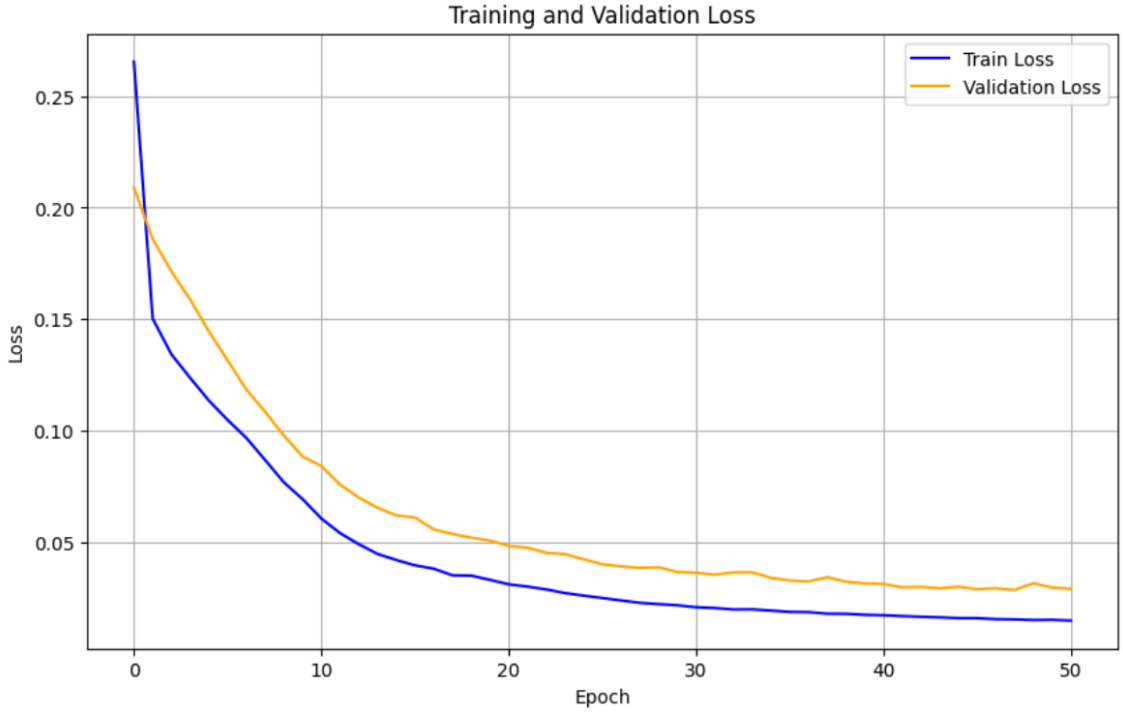

Рисунок 3.4 Функция потерь при обучении гибридной модели с KAN-нейронами

ЗАКЛЮЧЕНИЕ

В данной работе проведено комплексное исследование нейросетевых архитектур для прогнозирования плотности гибридных наножидкостей. Основные результаты:

Ключевые выводы

- Классический MLP показал лучшую эффективность в текущей задаче (время обучения 55 сек при 151 итерации)
- Гибридная KAN-архитектура требовала меньше итераций - 50, но имела большее время обучения 93 сек)

Перспективы KAN-сетей

Несмотря на проигрыш в данной задаче, KAN-архитектуры перспективны для:

- Работа с гладкими математическими функциями
- Точная аппроксимация сложных кривых и осциллирующих зависимостей
- Научные вычисления и физическое моделирование
- Случаи с ограниченным объемом обучающих данных
- Задачи, требующие интерпретируемости модели

Рекомендации по выбору

Основным ограничением KAN-архитектур в текущей реализации является скорость обучения - они работают примерно в 10 раз медленнее, чем традиционные MLP с аналогичным количеством параметров. Однако важно подчеркнуть, что это скорее временная инженерная проблема, а не фундаментальное ограничение метода.

MLP остаются лучшим выбором, когда критически важна скорость обучения

KAN следует рассматривать в случаях, где приоритетом являются интерпретируемость модели и высокая точность аппроксимации, а скорость обучения не является определяющим фактором

Перспективы развития KAN связаны в первую очередь с оптимизацией вычислительной эффективности. Учитывая их преимущества в точности и интерпретируемости, такие архитектуры представляют особый интерес для научных исследований и прикладных задач, где эти качества имеют первостепенное значение.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] Van Duy Tran, Tran Xuan Hieu Le, Thi Diem Tran, Hoai Luan Pham, Vu Trung Duong Le, Tuan Hai Vu, Van Tinh Nguyen, Yasuhiko Nakashima. Exploring the limitations of kolmogorov-arnold networks in classification: Insights to software training and hardware implementation. 07, 2024.

[2] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Thomas Y. Hou, Marin Soljačić, Max Tegmark. Kan: Kolmogorov-arnold networks. 04 2024.

[3] Колмогоров А.Н, Фомин С.В. *Элементы теории функций и функционального анализа.* Физматлит, 2004.

[4] Хайкин С. *Нейронные сети. Полный курс.* Вильямс, 2008.

ПРИЛОЖЕНИЕ

QR-код репозитория проекта на GitHub