**Введение (рассчитано примерно на 10 минут речи)**

Добрый день, уважаемые слушатели.

Сегодня я расскажу вам об альтернативной архитектуре нейронных сетей под названием **Kolmogorov-Arnold Networks**, или кратко **KAN**, и о том, почему она была предложена как потенциальная замена традиционным **многослойным перцептронам** — MLP.

Чтобы понять, зачем вообще нужно придумывать новые архитектуры, начнём с самой идеи нейросетей.

**Почему возникла потребность заменять MLP**

**MLP** — это, по сути, основа всего, с чего начинался глубокий learning. Внутри каждого слоя нейронной сети происходит линейное преобразование входов с помощью весов, после чего результат пропускается через фиксированную нелинейную функцию активации, вроде ReLU или tanh.

Эта простая идея действительно хорошо работает. Но по мере развития машинного обучения начали проявляться ограничения:

1. **Низкая интерпретируемость**:  
   Веса в MLP мало что говорят человеку. Мы не можем легко объяснить, *почему* сеть приняла то или иное решение.
2. **Сложности с обучением**:  
   По мере увеличения числа слоёв и параметров, обучение становится всё менее стабильным, возрастают риски переобучения, требуется тонкая настройка архитектуры и регуляризация.
3. **Неоптимальное представление функций**:  
   MLP аппроксимируют функции как суперпозицию линейных преобразований и фиксированных активаций. Такая схема может быть неэффективна, особенно когда функция, которую нужно аппроксимировать, имеет сложную структуру — например, резкие переходы, или высоко локализованные особенности.

**Идея, стоящая за KAN: теория Колмогорова**

Теперь давайте сделаем шаг в сторону теории.

В середине XX века Андрей Николаевич Колмогоров, а позже его ученик Владимир Арнольд, доказали важную теорему:

**Любую непрерывную функцию многих переменных можно представить как суперпозицию непрерывных функций от одной переменной и операций сложения.**

Это фундаментальное утверждение говорит нам: вместо того чтобы приближать сложную функцию через композиции линейных преобразований, можно **приближать её с помощью суммы унарных функций** — то есть функций, каждая из которых зависит только от одного переменного.

**KAN — применение теоремы Колмогорова в нейросетях**

KAN делают именно это. Вместо линейных весов между слоями, они помещают **не линейные коэффициенты, а унарные функции**, которые **обучаются** во время тренировки модели.

Для параметризации этих функций используются **B-сплайны** — кусочно-гладкие функции, которые можно эффективно обучать и которые хорошо аппроксимируют широкие классы функций.

В отличие от MLP, где каждая связь между нейронами — это просто число (вес), в KAN **связь между двумя узлами — это целая функция**, которую сеть может адаптировать под себя во время обучения.

**Что это даёт?**

1. **Гибкость и адаптивность**:  
   Унарные функции могут быть гораздо богаче, чем просто линейные веса. Они позволяют сети выразить сложную зависимость с меньшим числом параметров.
2. **Интерпретируемость**:  
   Так как каждая функция видна и доступна после обучения, мы можем визуализировать, *как именно* сеть преобразует входы на каждом шаге. Это шаг в сторону *объяснимого ИИ*.
3. **Связь с теорией**:  
   Использование суперпозиции унарных функций не случайно — это теоретически обоснованный и математически надёжный способ представления многомерных зависимостей.

**Почему этого не было раньше?**

Возникает естественный вопрос — почему раньше никто не реализовывал сети по Колмогорову?

Дело в том, что:

* Обучение функций на ребрах было слишком ресурсоёмким и нестабильным.
* Не существовало эффективных параметризаций унарных функций.
* Не было удобных библиотек и фреймворков.

Лишь с развитием современных фреймворков, автоматического дифференцирования и GPU, обучение таких моделей стало практически осуществимым. И вот — появились **KAN**, как попытка реализовать теоретическую идею Колмогорова на практике.

**Заключение введения**

Таким образом, архитектура **Kolmogorov-Arnold Networks** — это не просто инженерный трюк, а результат глубокой теоретической идеи. Она предлагает заменить линейные веса на обучаемые функции, чтобы добиться большей гибкости, выразительности и интерпретируемости.

В дальнейших разделах я расскажу об устройстве архитектуры KAN, её отличиях от MLP, преимуществах и недостатках, а также покажу примеры, где KAN превосходят классические сети.

Хочешь — могу дополнительно подготовить слайды, диаграммы (например, отличие MLP и KAN визуально), или помочь с текстом по следующим разделам презентации.