Universal Approximation Theorem and Kolmogorov-Arnold Networks

Niyathi Srinivasan

December 2024

Introduction

Motivation

Artificial Intelligence has become ubiquitous in the past few years. In 2022, large language models like ChatGPT were released to the public, showing millions around the world the potential of AI.

Motivation

Since then, achievements like

- predicting the 3D models of proteins (AlphaFold)
- creating art from text input (DALL-E)
- solving Go (AlphaGo)

have changed how humans think and work.

All these models have one thing in common – they are built using an Artificial Neural Network (ANN)

ANNs

But why are ANNs so powerful?

ANNs

But why are ANNs so powerful?

Almost all practical problems such as playing a game of Go or mimicking intelligent behavior can be represented by mathematical functions.

Universal Approximation Theorem

Universal Approximation Theorem

The **Universal Approximation Theorem** states that for any continuous function $f : [a, b] \longrightarrow \mathbb{R}$ and any $\epsilon > 0$, there exists a neural network N with a single hidden layer, such that

$$| f(x) - N(x) | < \epsilon$$

for all x in the domain of f.

Essentially, this theorem states that neural networks can approximate continuous functions to any desired degree of accuracy.

Universal Approximation Theorem

A single hidden layer neural network with m neurons can be expressed as:

$$N(x) = \sum (w_i * \varphi(\sum (a_{ij} * x_j + b_i)))$$

where:

- w_i, a_{ij}, b_i, and c_i are weights and biases,
- φ is the activation function (e.g., sigmoid, ReLU), and
- x_j are the components of the input vector x.

The activation function $\varphi : \mathbb{R} \to \mathbb{R}$ must be non-constant, bounded, and continuous. A common choice is the sigmoid function:

$$\frac{1}{1 + \exp(-x)}$$

Artificial Neural Networks

Structure of an ANN

ANNs consist of interconnected nodes (called **neurons**) that process and transmit data through a series of connections.

They use **Activation Functions** (represented by σ), which are functions applied to the output of a neuron that introduces non-linearity into the network

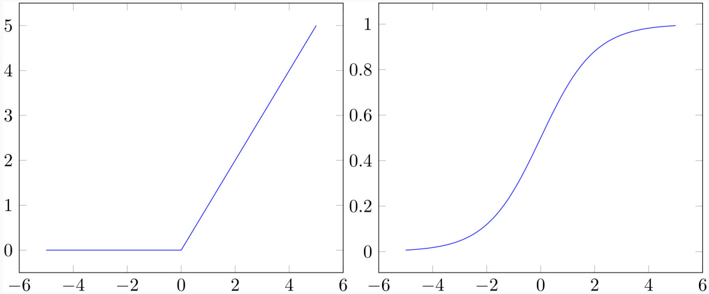
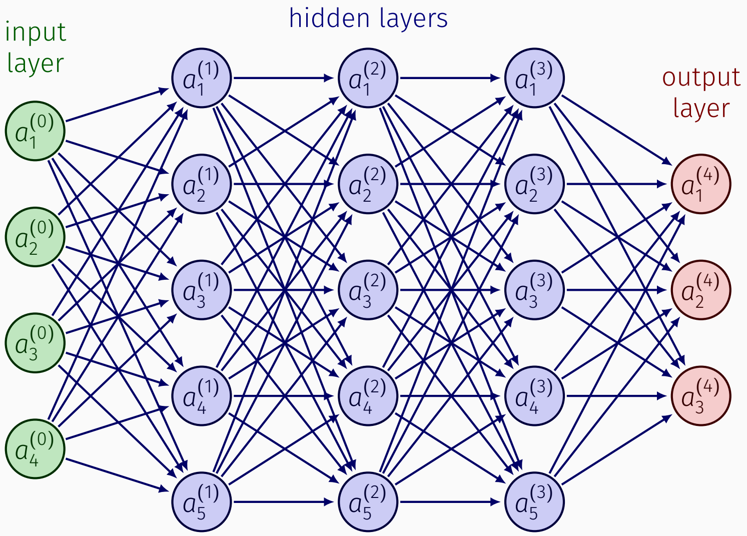


Figure 1: ReLU (left) and Sigmoid (right) activation functions

Structure of an ANN

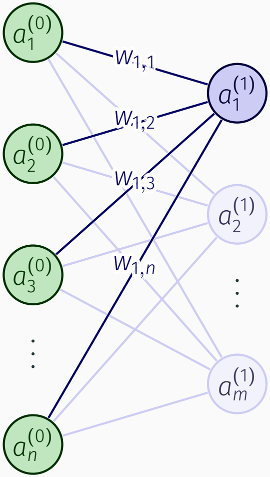


Structure of an ANN

1. **Input layer**: Receives input data and passes it on to the hidden layers without much computation
2. **Hidden layer**: The layer(s) that usually perform most of the computation; the number of neurons in each hidden layer can vary widely depending on the complexity of the net
3. **Output layer**: Receives the output from the last hidden layer, conducts the final computation, and produces the ultimate output of the entire network

Structure of an ANN

Each neuron has **inputs** $x_1, x_2, \ldots, x_n \in \mathbb{R}$ with corresponding **weights** $w_1, w_2, \ldots, w_n \in \mathbb{R}$.



$$= \sigma\left(w_{1,1}a_1^{(0)} + w_{1,2}a_2^{(0)} + \ldots + w_{1,n}a_n^{(0)}\right)$$
$$= \sigma\left(\sum_{i=1}^{n} w_{1,i}a_i^{(0)}\right)$$

Kolmorogov Arnold Networks

What are Kolmorogov-Arnold Networks?

A neural network architecture based on Kolmorogov's proof in 1957 that any continuous multivariable function can be represented as a superposition of a finite number of univariate functions.

KANs replace the fixed linear activation functions in traditional neural networks with learnable univariate functions.

KANs

Kolmorogov-Arnold Representation Theorem:

For any continuous function $f : [0,1]^n \to \mathbb{R}$, there exist continuous functions ϕ_i and ψ_i such that:

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{2n+1} \phi_i \left(\sum_{j=1}^{n} \psi_{ij}(x_j) \right)$$

KAN Algorithm

1. **Input Transformation**:

$$y_i = \sum_{j=1}^{n} \psi_{ij}(x_j)$$

   where x_j are the inputs and ψ_{ij} are continuous functions.

2. **Output Aggregation**:

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{2n+1} \phi_i(y_i)$$

   where ϕ_i are continuous functions applied to the transformed inputs y_i.

KANs vs. ANNs

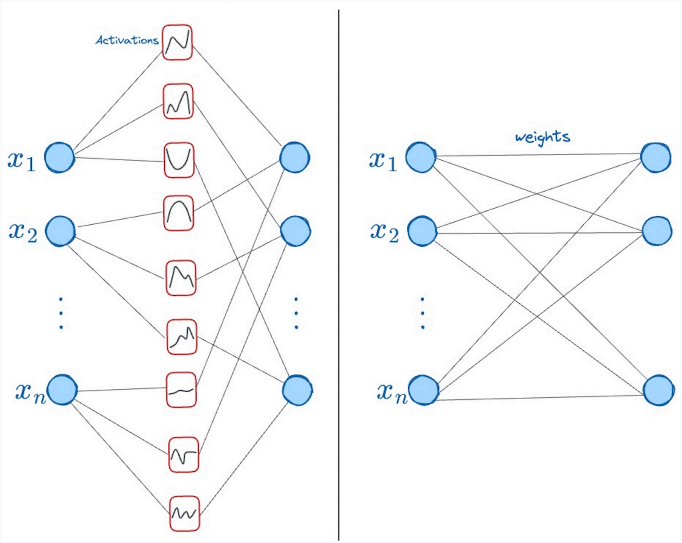


Figure 2: KAN architecture (left) vs. ANN architecture (right)

Unique Features of KANs

KAN activation functions are typically parameterized by spline functions (piecewise polynomial functions).

They are not limited to predefined forms (such as ReLU, sigmoid, etc.) but can be adjusted during the training process.

This allows KANs not only to learn features, like regular neural networks, but also to optimize these learned features to great accuracy.

Benefits of KANs

- KANs can achieve high-precision approximation with lower complexity (amount of parameters)
- Studies have shown that KAN achieves higher accuracy and efficiency in applications such as hyperspectral image classification and time series analysis through a more flexible model architecture
- Liu et al. demonstrated that KAN outperforms traditional methods in hyperspectral image classification in 2024

What’s the Catch?

Currently, the biggest bottleneck of KANs lies in its **slow training**. KANs are usually **10x** slower than regular neural nets, given the same number of parameters.

Acknowledgements

I would like to thank

- Lali, Yael, and Justin for being amazing mentors
- My parents for being extremely supportive
- The PRIMES coordinators for this wonderful opportunity