CA6001 Chapter 3
Deep Learning

Dr Zhang Jiehuang

College of Computing and Data Science

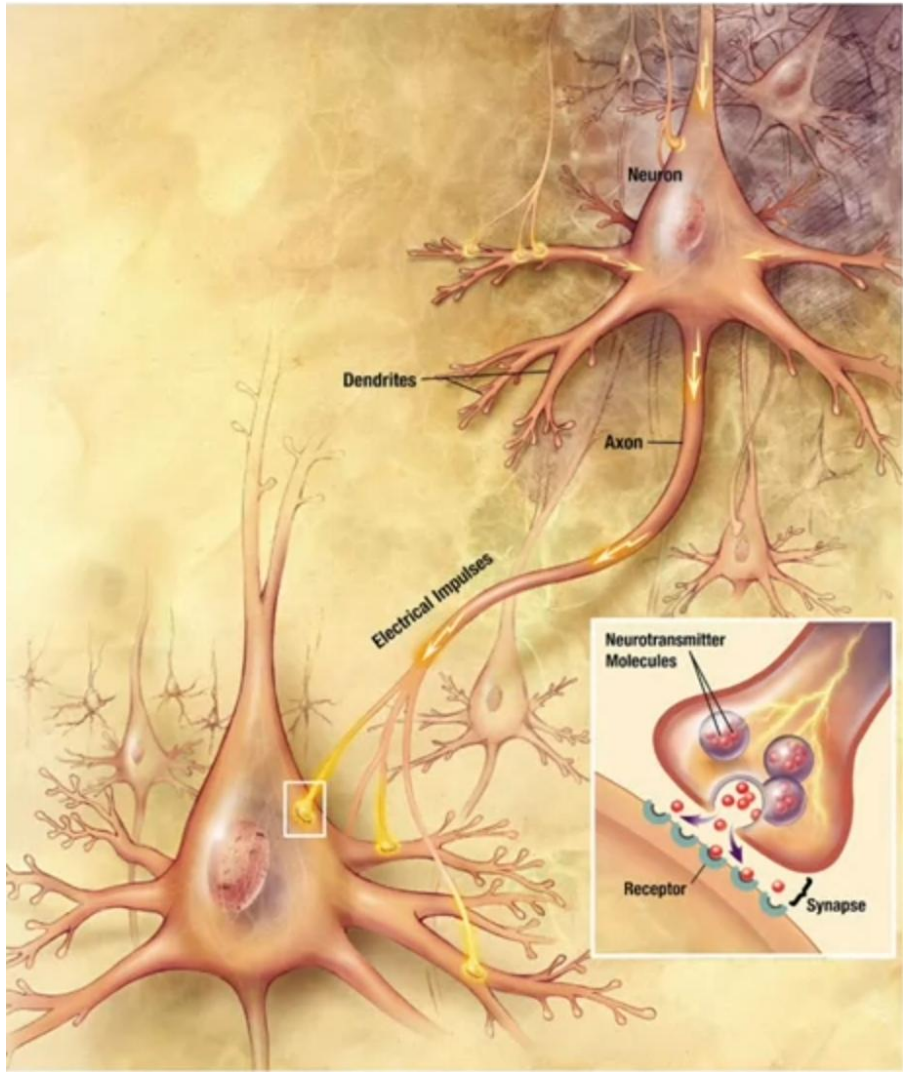
Nanyang Technological University

email: *jiehuang.zhang@ntu.edu.sg*

Deep Learning Fundamentals

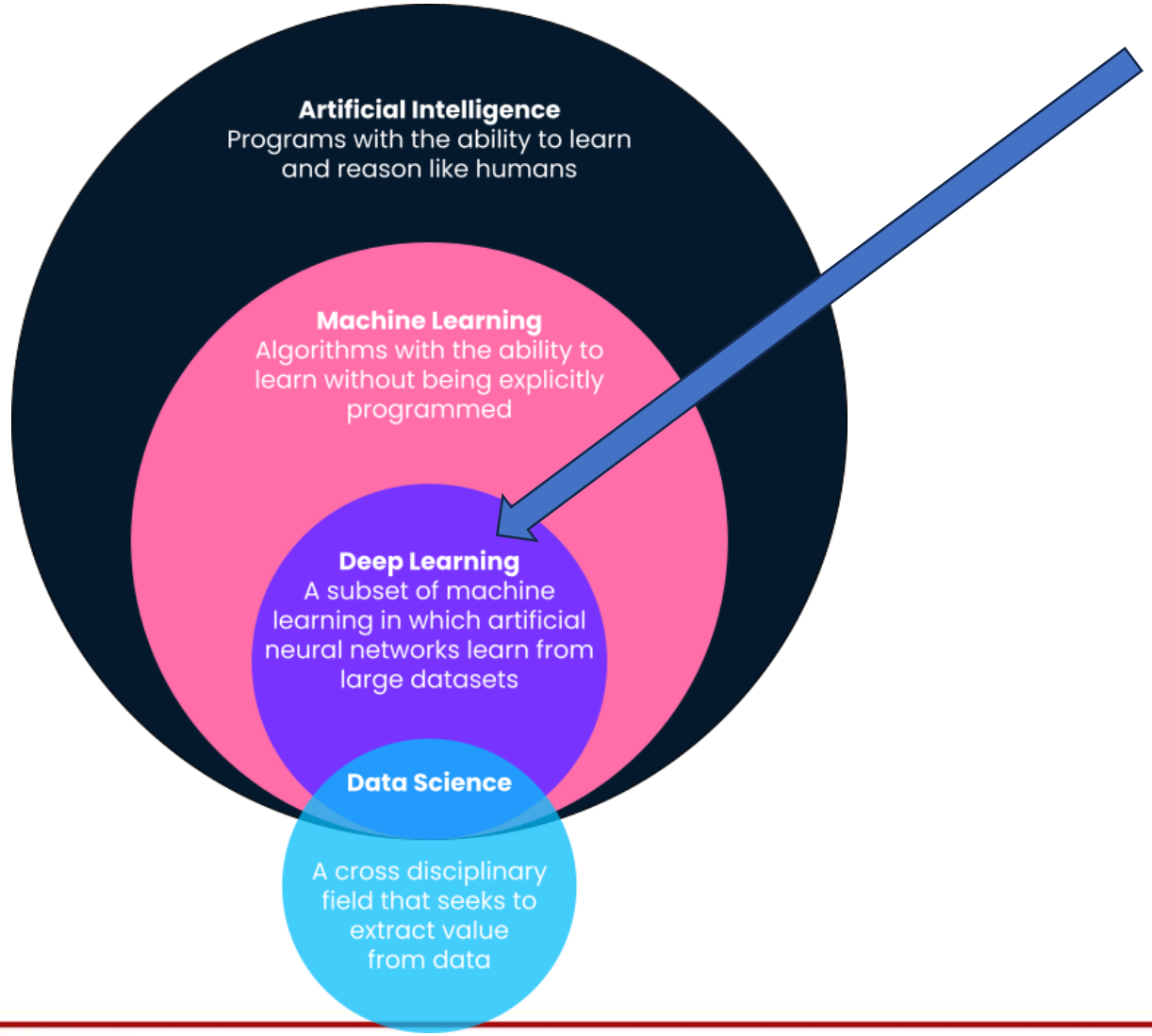
1. What Is Deep Learning – Key Concepts
2. Why the Resurgence of Deep Learning
3. Building Blocks of Deep Learning – Neural Networks
4. Forward and Backward Propagation
5. Gradient Descent & Activation Functions
6. Popular Architectures
7. Challenges and Limitations

What is Deep Learning?



1. Deep Learning has origins in algorithms that try to imitate the brain
2. Input is sent to the first neuron, then some calculations are made in it, then sent to the next neuron
3. Deep Learning works in a similar manner, input is sent to an artificial neuron, then calculations made and sent to next neuron
4. Deep Learning learns the selection of features without it being hard coded
5. Used in the 1980s onwards, gain resurgence in the 2005s

Machine Learning VS Deep Learning?

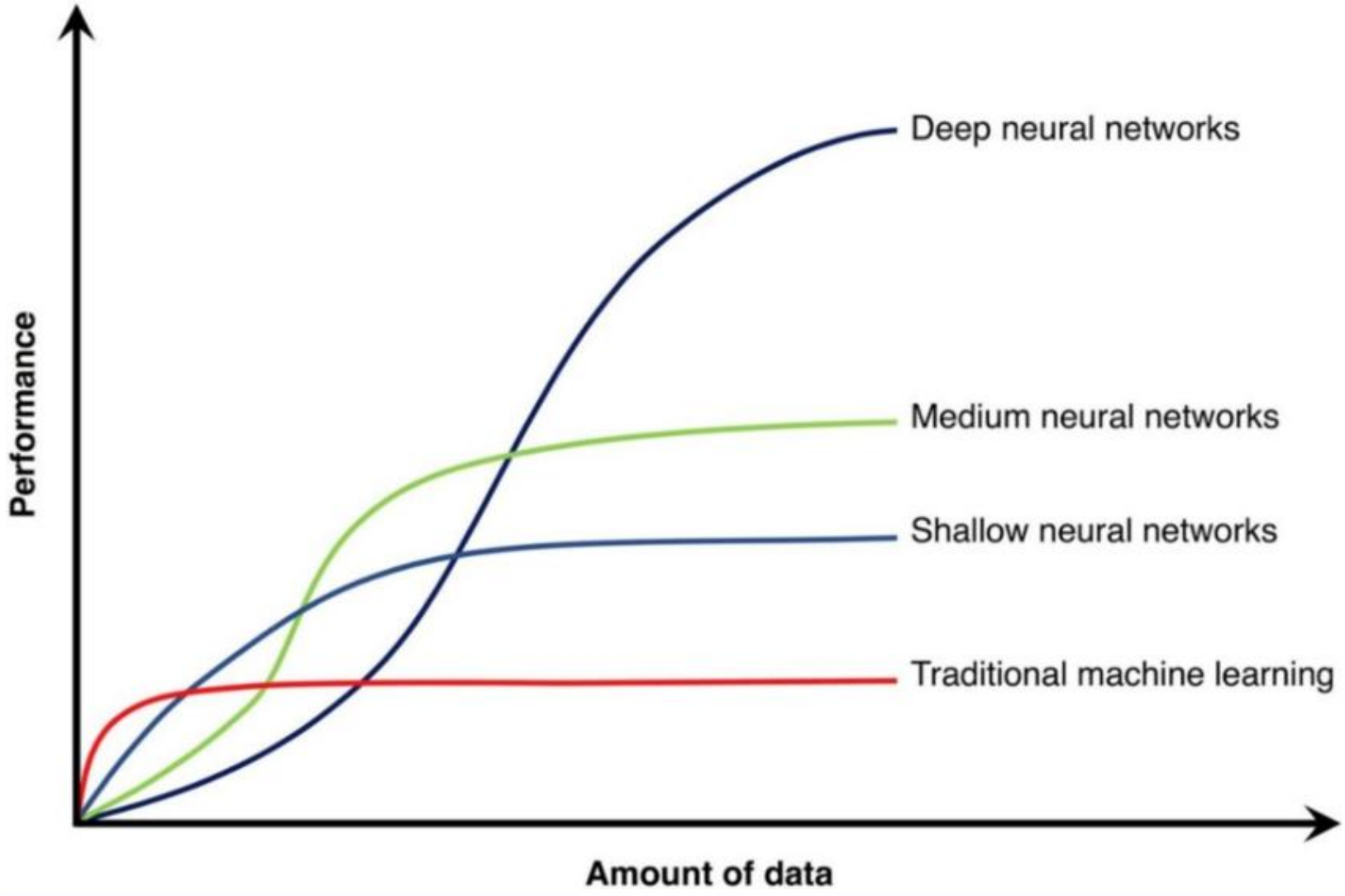


1. Deep Learning is a subset of Machine Learning
2. Deep learning learns features at multiple levels of abstraction (low level features are combined to detect high level patterns)
3. Model automatically learns the best features during training
4. Input Data flows through layers to get the final output
5. Works well with large datasets and large computational resources (GPUs/TPUs)
6. Performs well when trained on large, diverse and labelled datasets

Machine Learning VS Deep Learning?

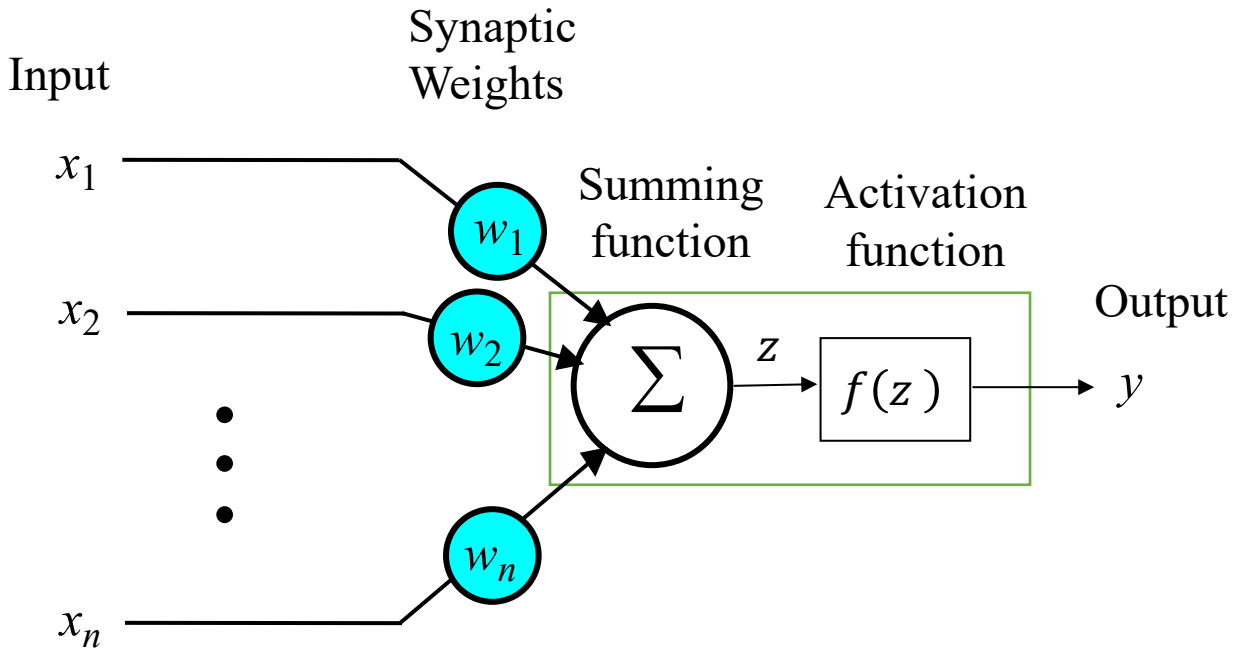
Aspect	Machine Learning	Deep Learning
Feature Extraction	Manual feature engineering required.	Features are learned automatically.
Complexity	Handles simpler problems effectively.	Excels at complex tasks (e.g., vision, NLP).
Data Requirements	Requires smaller datasets.	Needs large datasets to perform well.
Computation	Can run on CPUs.	Often requires GPUs/TPUs.

Why the Resurgence of Deep Learning?



https://www.researchgate.net/figure/DNN-Performance-vs-Amount-of-data-29_fig1_372648984

Deep Learning: Basic Building Blocks



Input $\boldsymbol{x} = (x_1 \quad x_2 \quad \cdots \quad x_n)^T$

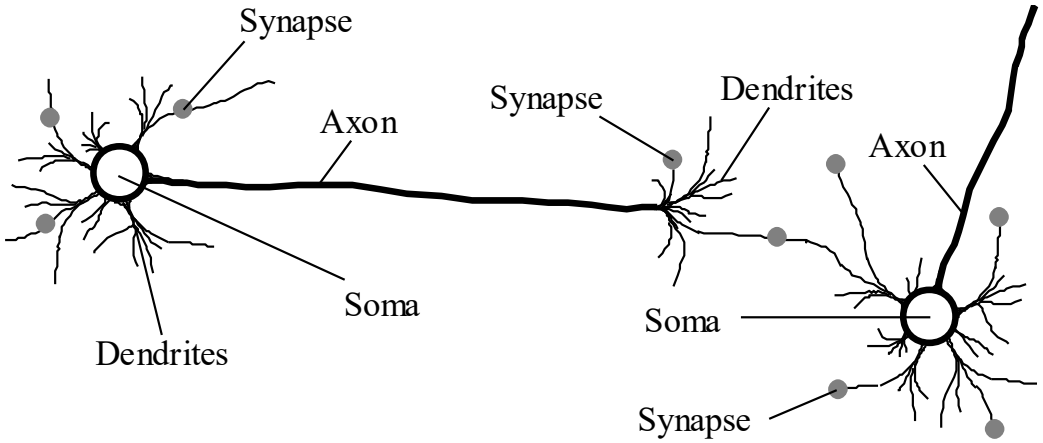
z – total synaptic input

f – activation function

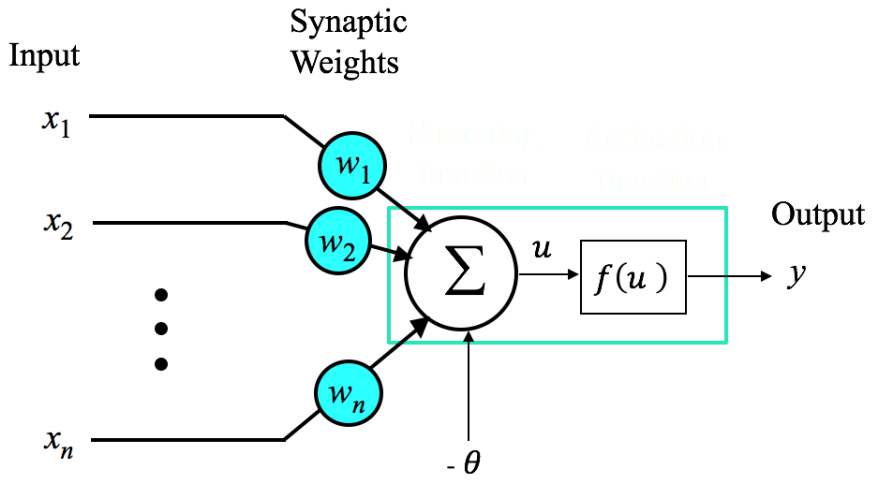
y - output

Biological VS Artificial Neurons

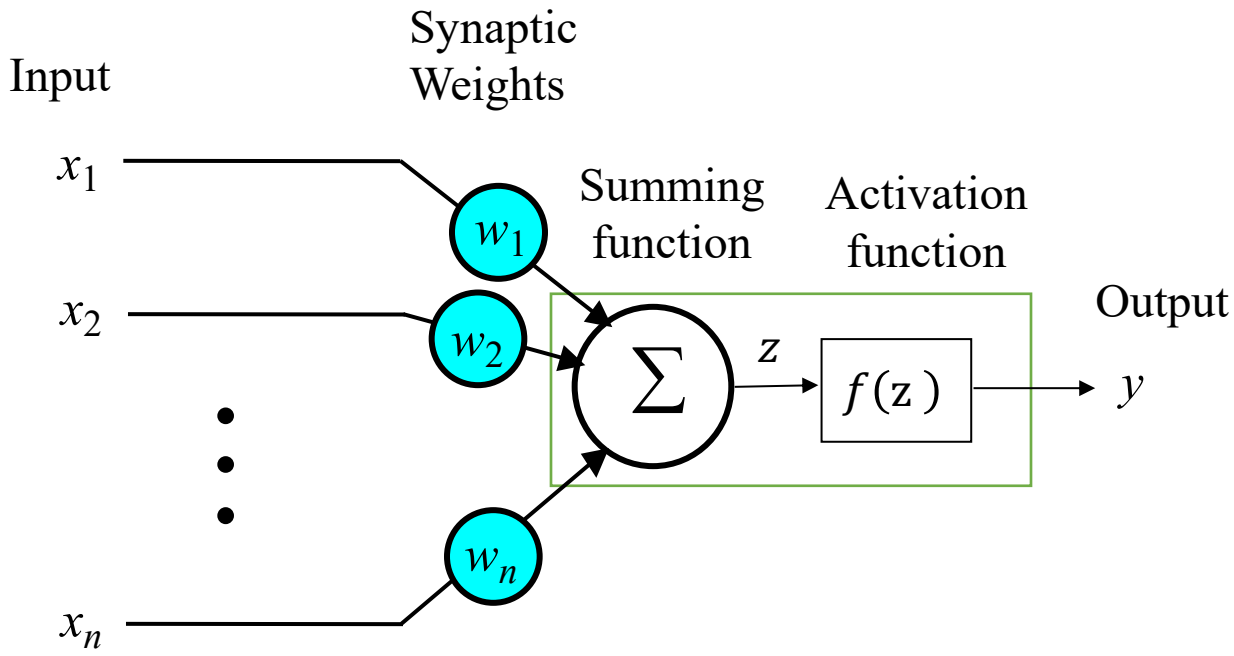
Biological neuron



Artificial neuron



Forward Propagation



x: input

w: synaptic weight(tells you how important it is)

b: bias (another parameter for importance)

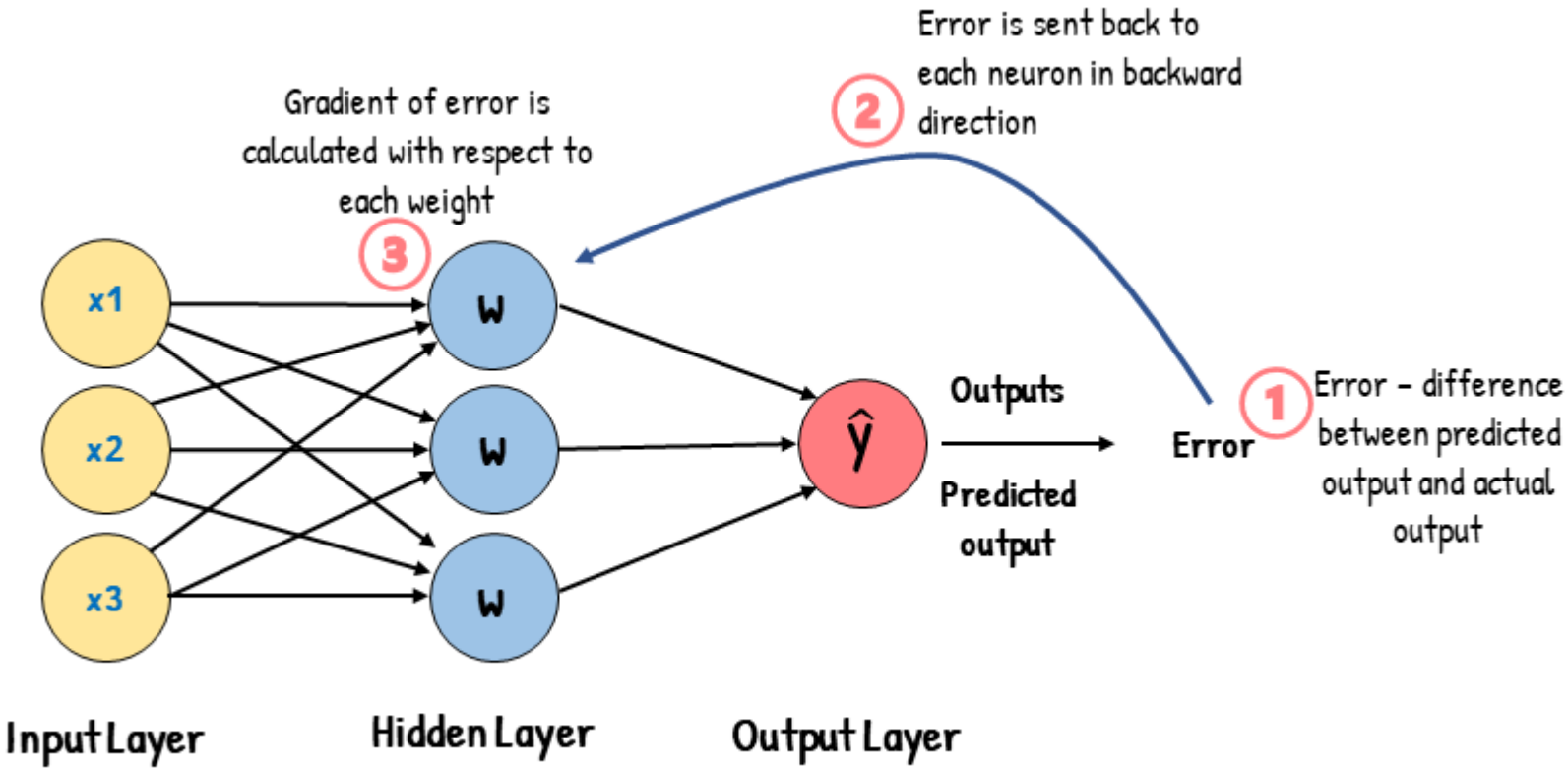
z: synaptic output

$$z \begin{cases} z_1 = w_1 x_1 + b_1 \\ z_2 = w_2 x_2 + b_2 \\ .. \\ z_n = w_n x_n + b_n \end{cases}$$

f(z): activation function

$$y = f(z)$$

Backward Propagation



Backprop allows NN to adjust weights to reduce the error

1. Error is calculated to estimate the size of the difference
2. Error value is sent back to the neurons in each layer
3. Gradient of error is calculated with respect to each weight w

After thousands of iterations, the weight and biases reach the optimal, "correct" values

https://www.analyticsvidhya.com/blog/2023/01/gradient-descent-vs-backpropagation-whats-the-difference/

When is FP and BP used

Forward Propagation

•**When it's used:** During the prediction or inference phase and during the training process.

•**What it does:** It involves passing the input data through the neural network to compute the predicted output (e.g., class probabilities, regression values).

Purpose:

- During **inference**, the predicted output is used as the final result.
- During **training**, the loss value is needed for backward propagation

Backward Propagation

•**When it's used:** During the training phase only.

•**What it does:** It calculates the gradients of the loss function with respect to the network's weights and biases using the **chain rule of calculus**. These gradients are used to update the weights in a way that minimizes the loss.

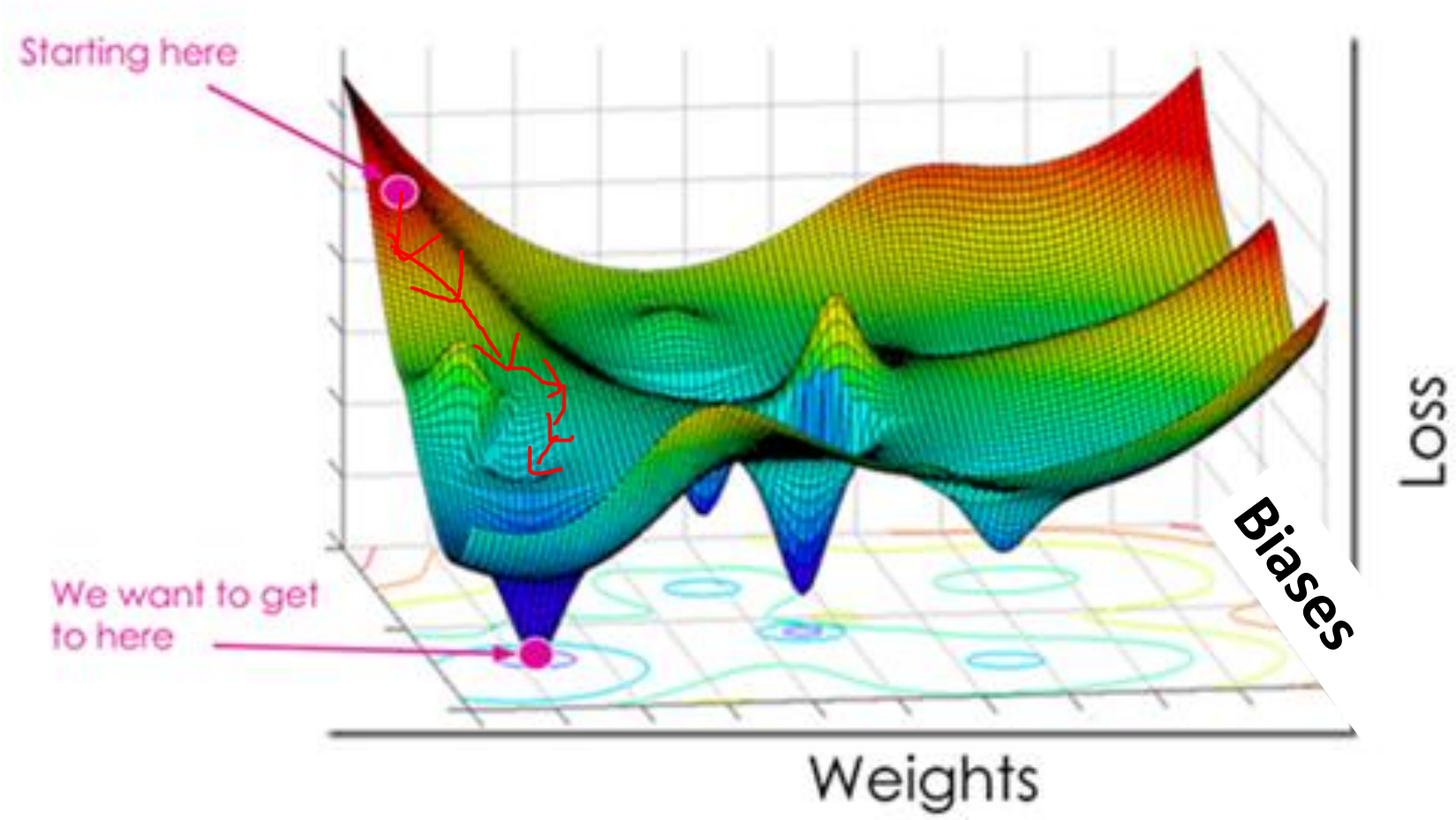
Purpose:

- Adjust the weights to reduce the error and improve the model's predictions.

Key Differences Forward Propagation vs Backpropagation

Aspect	Forward Propagation	Backward Propagation
Used for	Computing predictions	Updating weights and biases
Direction	Input → Output	Output → Input
Phase	Both training and inference	Training only
Mathematics Operations	Matrix multiplications, activations	Gradients via the chain rule

Gradient Descent



Gradient Descent is the optimization algorithm employed to find the optimal value of weights w and b that minimizes cost function (convergence)

We start from an initiation point, then the algo searches nearby and moves towards the minima step by step

Activation Functions

Z = wx + b models a linear relationship
but what if the relationship is non linear?

Activation functions are mathematical functions applied to the output of neurons in a neural network to determine whether they should be "activated" (pass their signal to the next layer). They allow the network to learn **complex patterns and relationships** in the data.

Why Are Activation Functions Important To Learn Complex Patterns And Relationships?

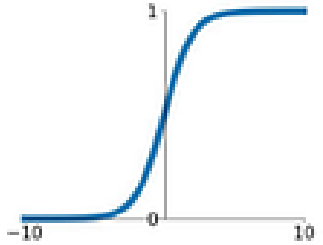
1.Non-linearity: Without activation functions, the network would behave like a simple linear model, no matter how many layers it has.
2.Feature Learning: They enable the network to learn complex patterns, such as images, text, and speech.
3.Gradient Flow: Activation functions help in effective backpropagation by controlling how gradients are propagated through layers.

Activation Functions

Activation Functions

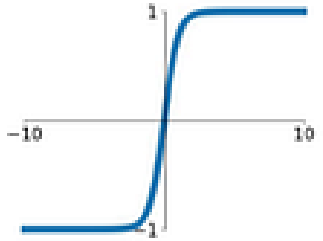
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



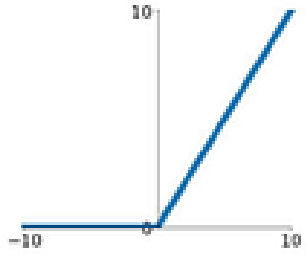
tanh

$$\tanh(x)$$



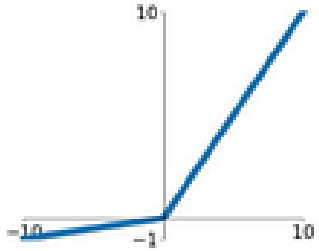
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

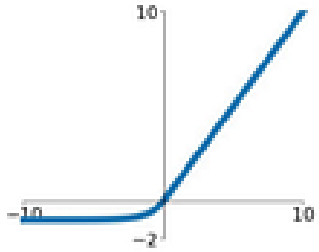


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Tensorflow Playground

https://playground.tensorflow.org/

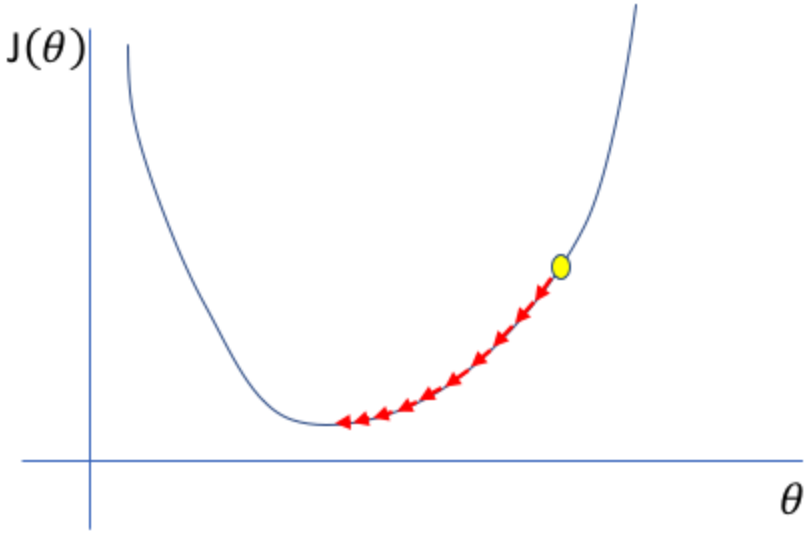
Explore and play with the different components of neural networks

https://www.youtube.com/watch?v=WQYCK1YpsjE

Neural Networks visualised

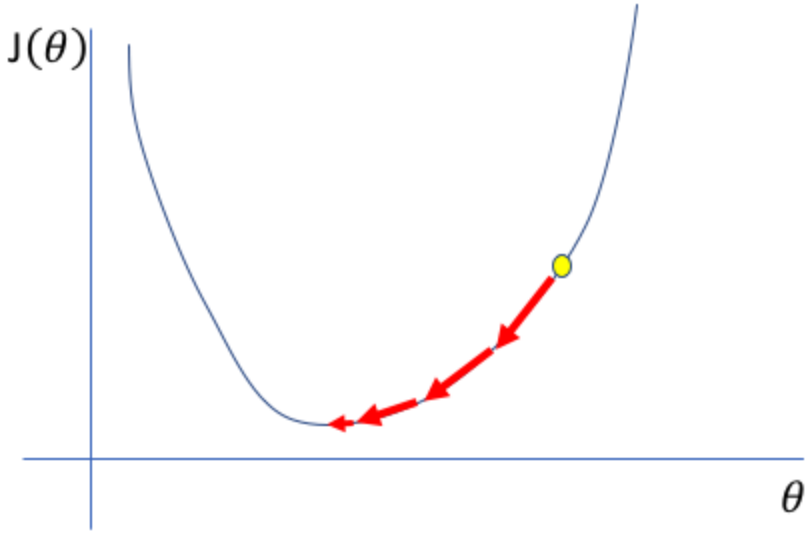
How to choose learning rate α?

Too low



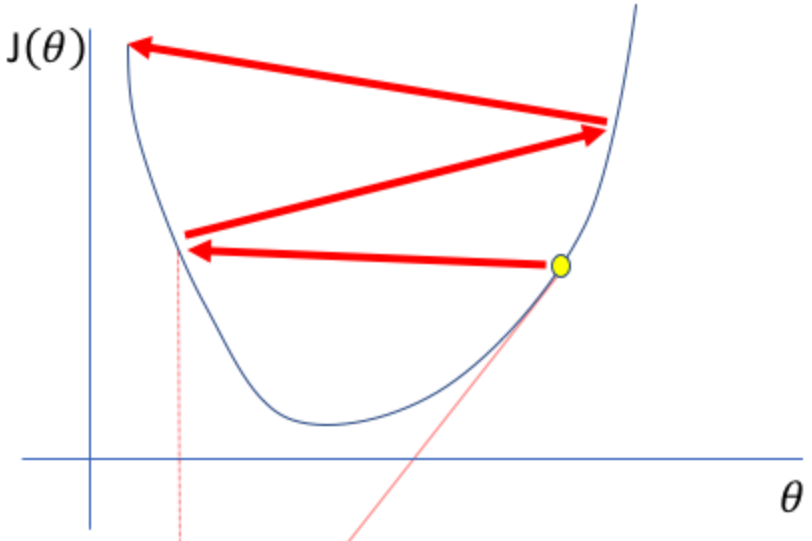
A small learning rate requires many updates before reaching the minimum point

Just right



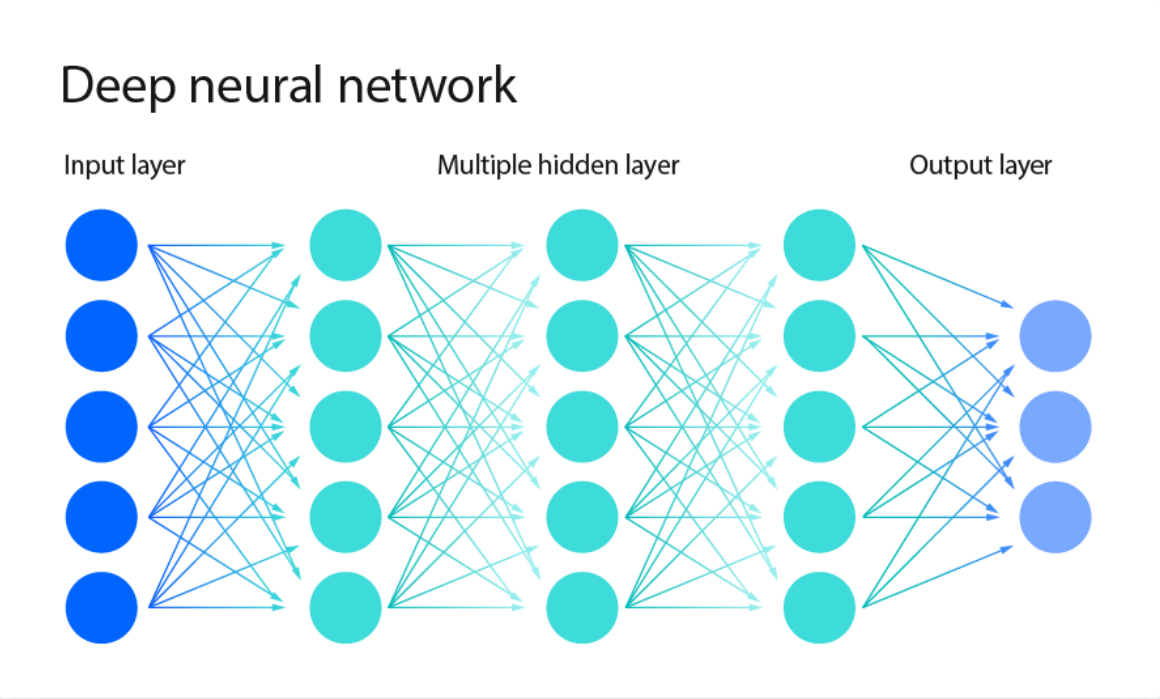
The optimal learning rate swiftly reaches the minimum point

Too high



Too large of a learning rate causes drastic updates which lead to divergent behaviors

Prediction with Neural Networks



Neural Networks are a new paradigm from traditional machine learning, as they can automatically learn which features are important

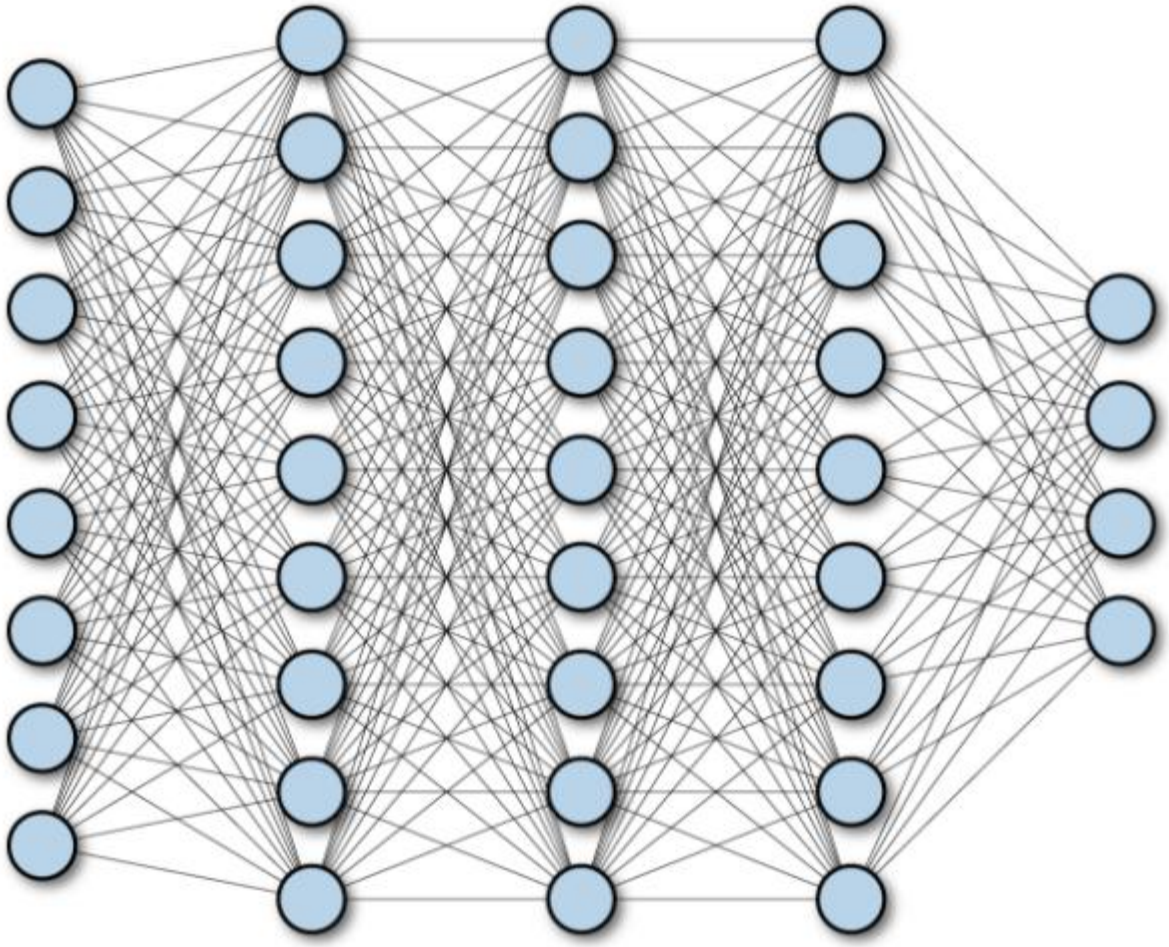
They need to be trained with large amounts of data first, then once trained they can predict labels for new data (some NNs are called Pretrained, P in ChatGPT)

Examples

1. Regression for continuous output labels (age, income)
2. Classification for discrete output labels (group label, cat or dog etc)

https://www.ibm.com/think/topics/neural-networks

Fully Connected (Dense) Layers



Feature Combination and High Level Feature Extraction: Transform the high-level, abstract features extracted earlier into forms suitable for making precise predictions

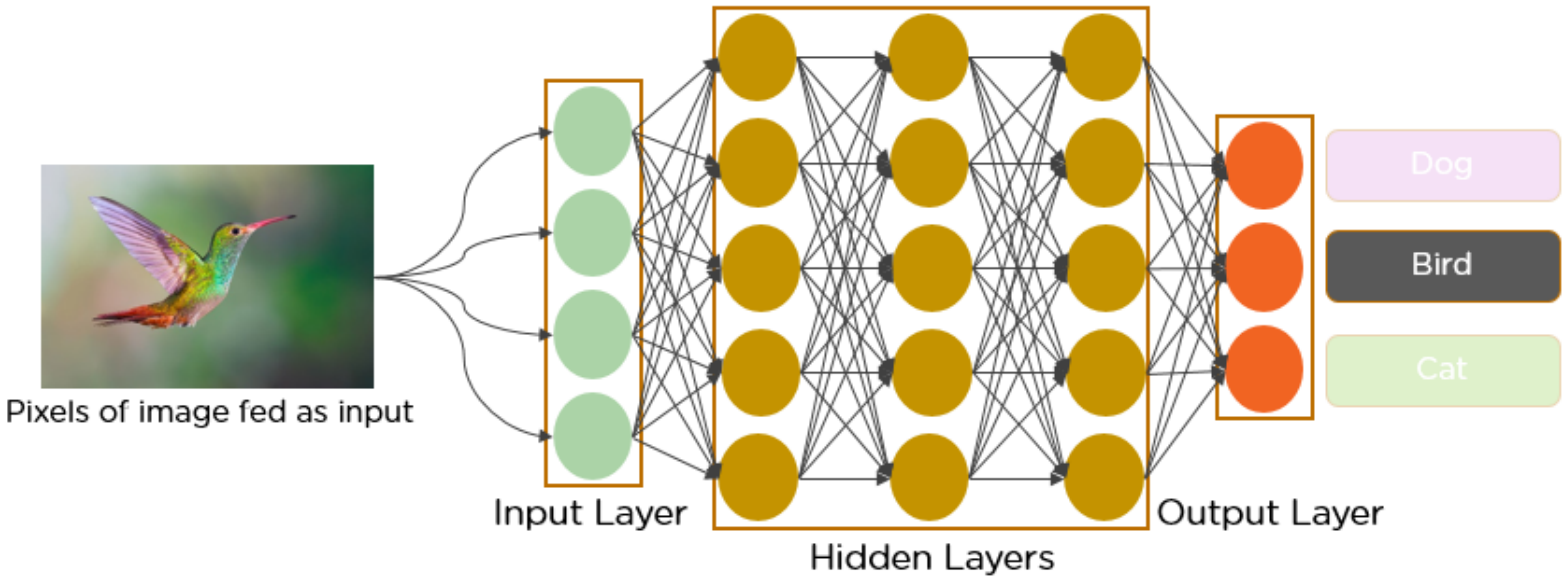
Decision Making and Output Generation: FC layers process high-level features into scores that are typically passed through a Softmax function to generate probabilistic class predictions

Non-Linearity: Activation Functions allow NNs to learn complex, non linear relationships within the data

Regularization and Overfitting Control: to mitigate overfitting, we can use techniques like Dropout (deactivate a portion of neurons during training) or L2 regularization(penalize large weights)

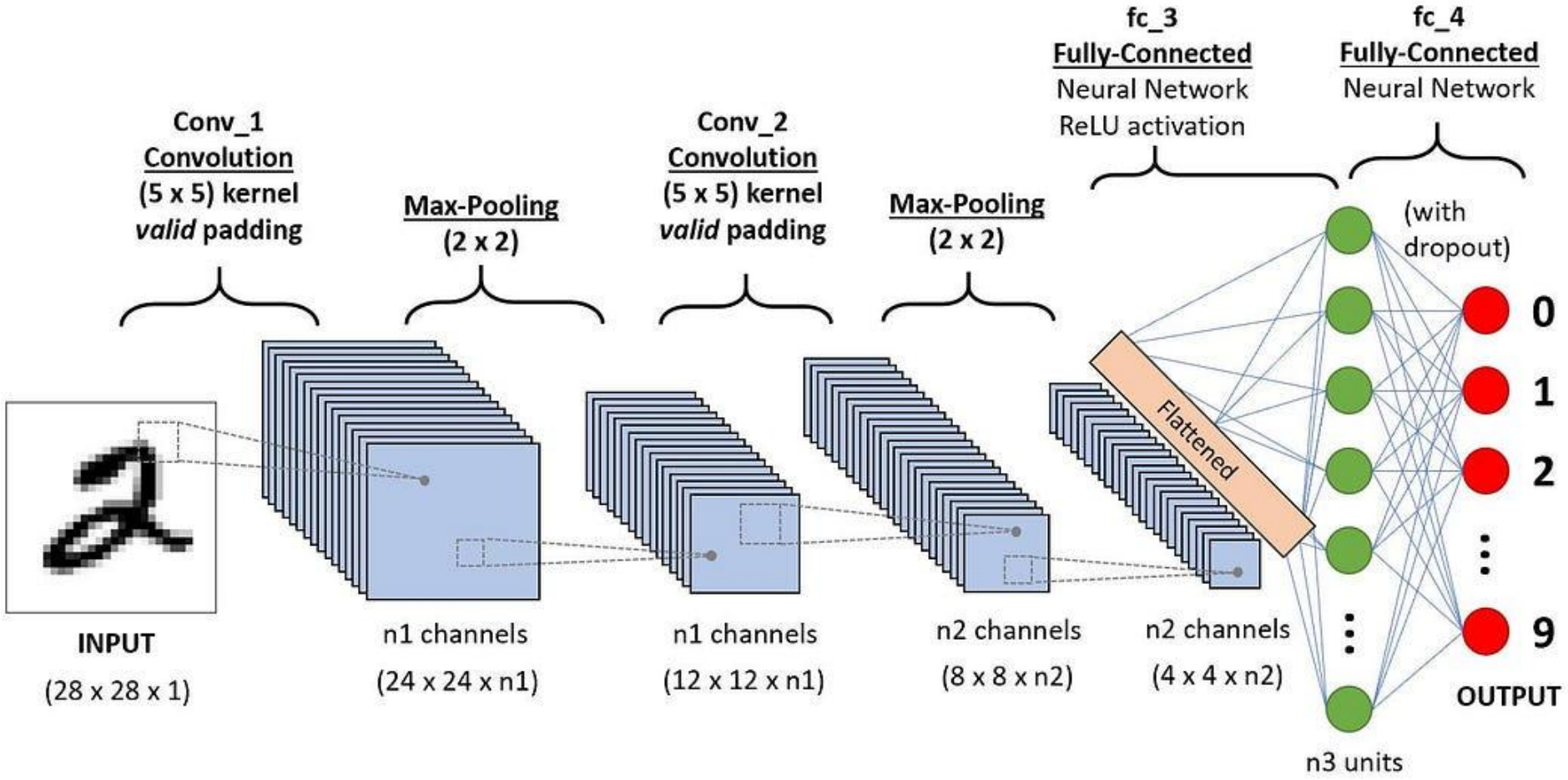
https://www.geeksforgeeks.org/what-is-fully-connected-layer-in-deep-learning/

Neural Networks in Computer Vision



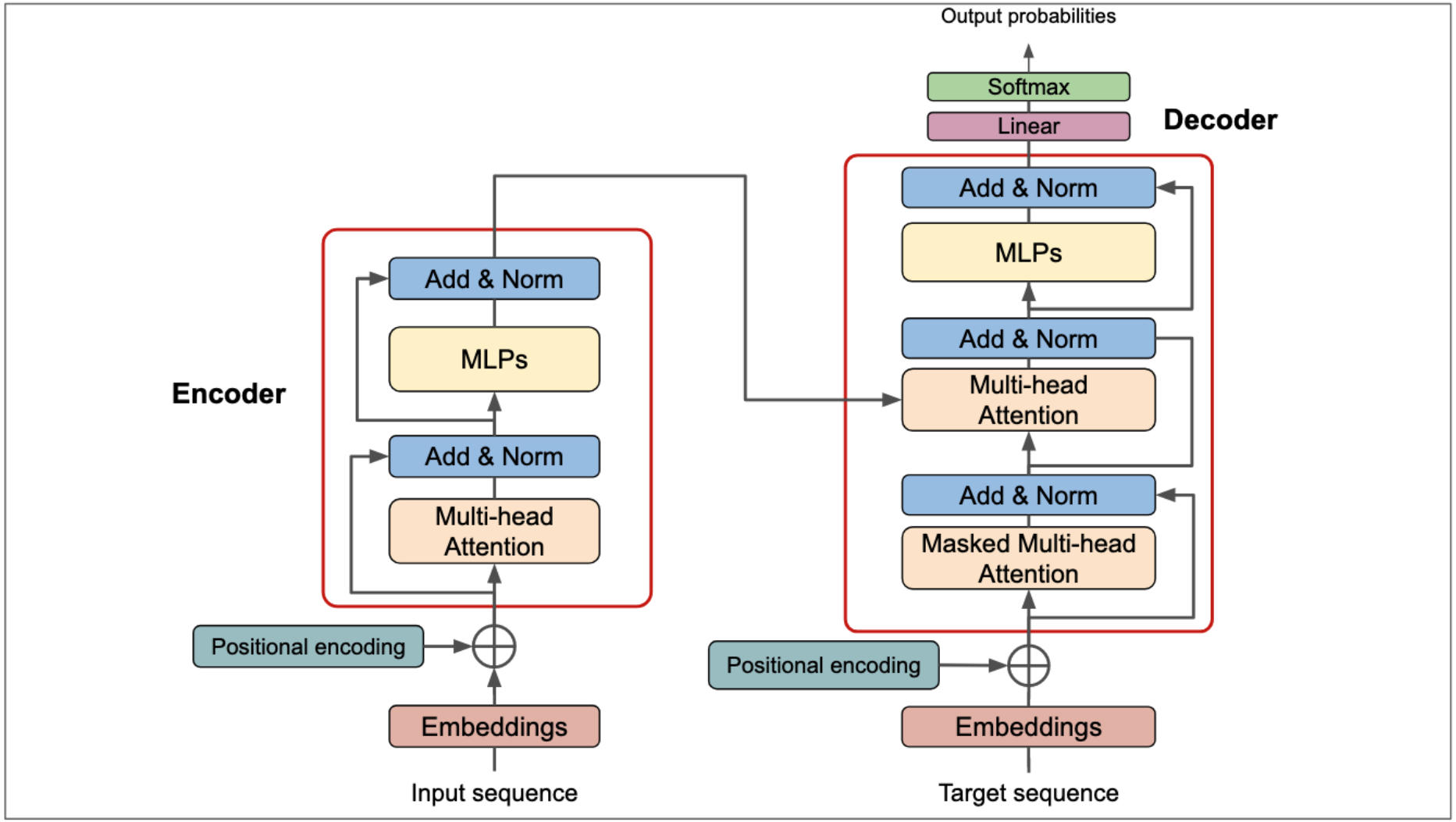
1. This neural network is a 4-layer network, with 3 hidden layers and 1 output layer
2. Input data is the pixels of the image of the bird
3. Each pixel is a number with the intensity of the RGB color palette
4. For networks with large number of hidden layers, they are referred to as deep neural networks

Popular Architectures: Convolutional Neural Networks



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Popular Architectures: Transformer

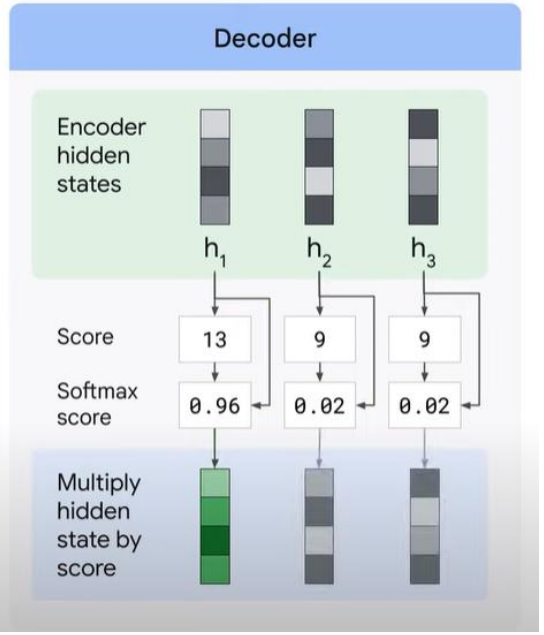


https://deeprevision.github.io/posts/001-transformer/

Attention and Transformers

To focus on the most relevant parts of the input:

1. Look at the set of encoder hidden states that it received.
2. Give each hidden state a score.
3. Multiply each hidden state by its soft-maxed score.



Attention mechanisms and transformer models consider contextual information and bidirectional relationships between words, leading to more advanced language representations.

Attention mechanisms were introduced to improve the ability of neural networks to focus on specific parts of the input sequence when making predictions. Instead of treating all parts of the input equally, attention mechanisms allow the model to selectively attend to relevant portions of the input.

Transformers use a self-attention mechanism to capture relationships between different words in a sequence. This mechanism allows each word to attend to all other words in the sequence, capturing long-range dependencies.

https://www.ibm.com/think/topics/word-embeddings#:~:text=Word%20embeddings%20capture%20semantic%20relationships,more%20nuanced%20understanding%20of%20language.

Self-Attention Mechanism

The quick brown fox jumped over the wall…. It was so agile. What does "It" mean?

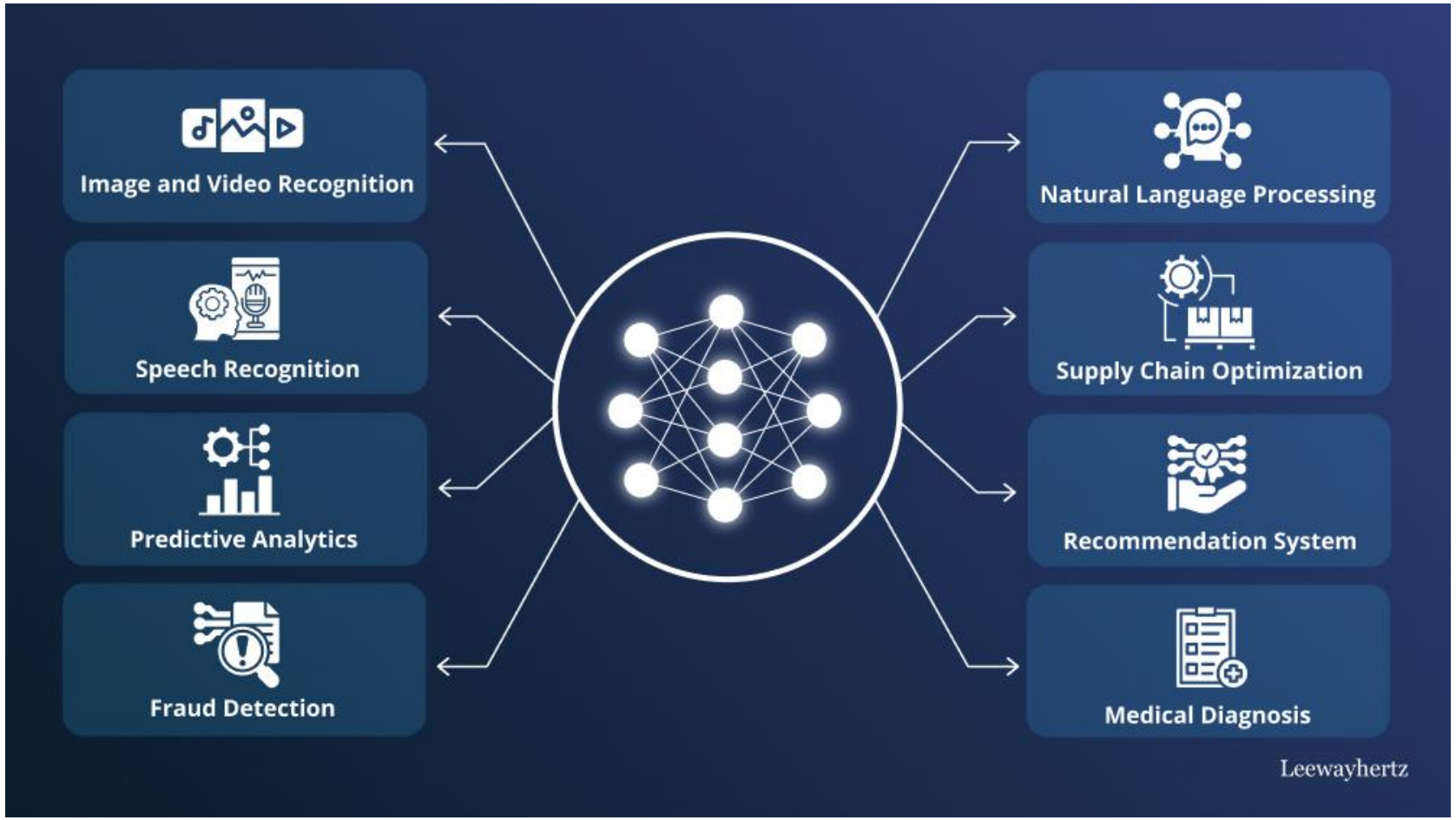
1. Input Representation:
Each word in the sequence is converted into a numerical vector (embedding), which contains its meaning in a high-dimensional space.

2. Three Key Matrices: Query, Key, and Value:
•**Query (Q):** Represents the word we're focusing on.
•**Key (K):** Represents all the other words in the sequence.
•**Value (V):** Contains the actual information of the words.
These matrices are learned during training and help the model calculate relationships between words.

3. Calculating Attention Scores:
The attention score measures how relevant each word is to the word being focused on. This is done by:
•Taking the **dot product** of the Query vector with each Key vector (indicates similarity).
•Scaling the score by dividing by the square root of the Key's dimension (helps with stable gradients).
•Applying a **softmax function** to convert the scores into probabilities.
For example, if "it" is the Query, the scores determine how much attention should be given to each word like "cat," "mat," etc.

4. Weighted Summation:
Each word's Value vector is multiplied by its attention score (from Step 3), and the results are summed up. This gives a new representation of the word "it," now enriched with context from other relevant words.

Applications of Deep Learning



- **Vision:** Object detection, facial recognition
- **Natural Language Processing (NLP):** Translation, sentiment analysis
- **Healthcare:** Diagnostics, drug discovery
- **Autonomous Systems:** Self-driving cars, robotics
- **Finance:** Fraud detection, algorithmic trading

Challenges and Limitations

1. Data requirements: High volume and quality needed
2. Computational cost: High GPU/TPU demand
3. Interpretability: Black-box nature
4. Ethical concerns: Bias, misuse, and privacy
5. Overfitting and generalization issues

References for Chapter 9 – Deep Learning

1. https://www.coursera.org/learn/advanced-learning-algorithms/lecture/couAA/neurons-and-the-brain
2. https://www.researchgate.net/figure/DNN-Performance-vs-Amount-of-data-29_fig1_372648984
3. https://www.analyticsvidhya.com/blog/2023/01/gradient-descent-vs-backpropagation-whats-the-difference/
4. https://www.ibm.com/think/topics/neural-networks
5. https://www.geeksforgeeks.org/what-is-fully-connected-layer-in-deep-learning/
6. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
7. https://deeprevision.github.io/posts/001-transformer/

Note: All online articles were accessed between Oct to Nov 2024

The End
Questions?