

BASED ON ARTIFICIAL INTELLIGENCE (AI)

A Brief Introduction

Amin Aslami

University of Jan Evangelista (UJEP)

aminaslami@f@gmail.com

Presentation

18 March 2025



ARTIFICIAL INTELLIGENCE (AI)

Artificial intelligence (AI) is a set of technologies that enable computers to perform a variety of advanced functions, including the ability to see, understand and translate spoken and written language, analyze data, make recommendations, and more.

AI is the basic of innovation in modern computing, unlocking value for individuals and businesses. For example, optical character recognition (OCR) uses AI to extract text and data from images and documents, turns unstructured content into business-ready structured data, and unlocks valuable insights.[1].

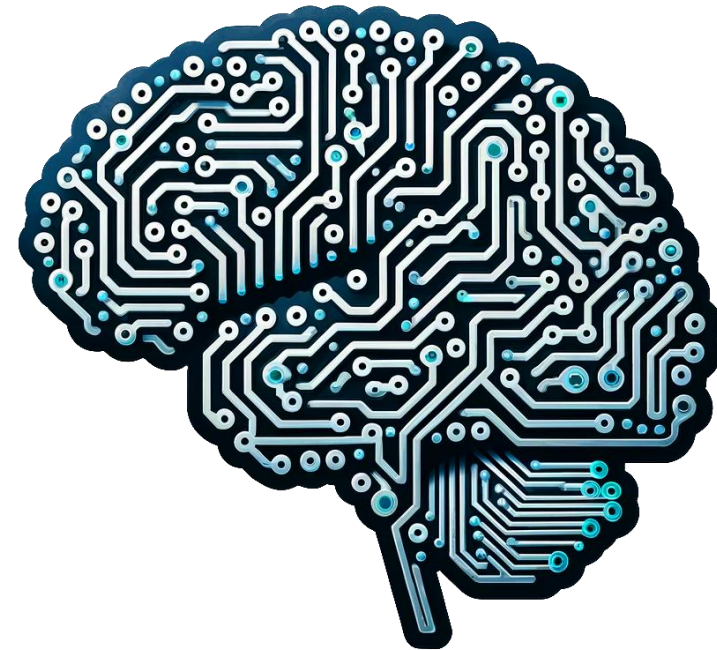


Figure 1.1. Artificial Intelligence [2]

CORE IDEA

AI aims to **mimic human cognitive functions**, such as:

- LEARNING
- REASONING, MANTIK
- PROBLEM-SOLVING
- PERCEPTION, ALGI
- LANGUAGE UNDERSTANDING

Key Fields in AI

1. Machine Learning (ML):

Algorithms that learn from data.

Example: Spam filters, recommendation systems.

2. Deep Learning (DL):

A subfield of ML using neural networks with many layers.

Example: ChatGPT, image recognition.

3. Natural Language Processing (NLP):

Understanding and generating human language.

Example: Language translation, chatbots.

4. Computer Vision:

Interpreting and understanding images/videos.

Example: Face recognition, self-driving cars.

5. Robotics:

Combining AI with physical machines.

Example: Autonomous drones, robot arms.

WHAT IS MACHINE LEARNING?

Machine learning (ML) is a branch of artificial intelligence (AI) focused on enabling computers and machines to imitate the way that humans learn, to perform tasks autonomously, and to improve their performance and accuracy through experience and exposure to more data [3].

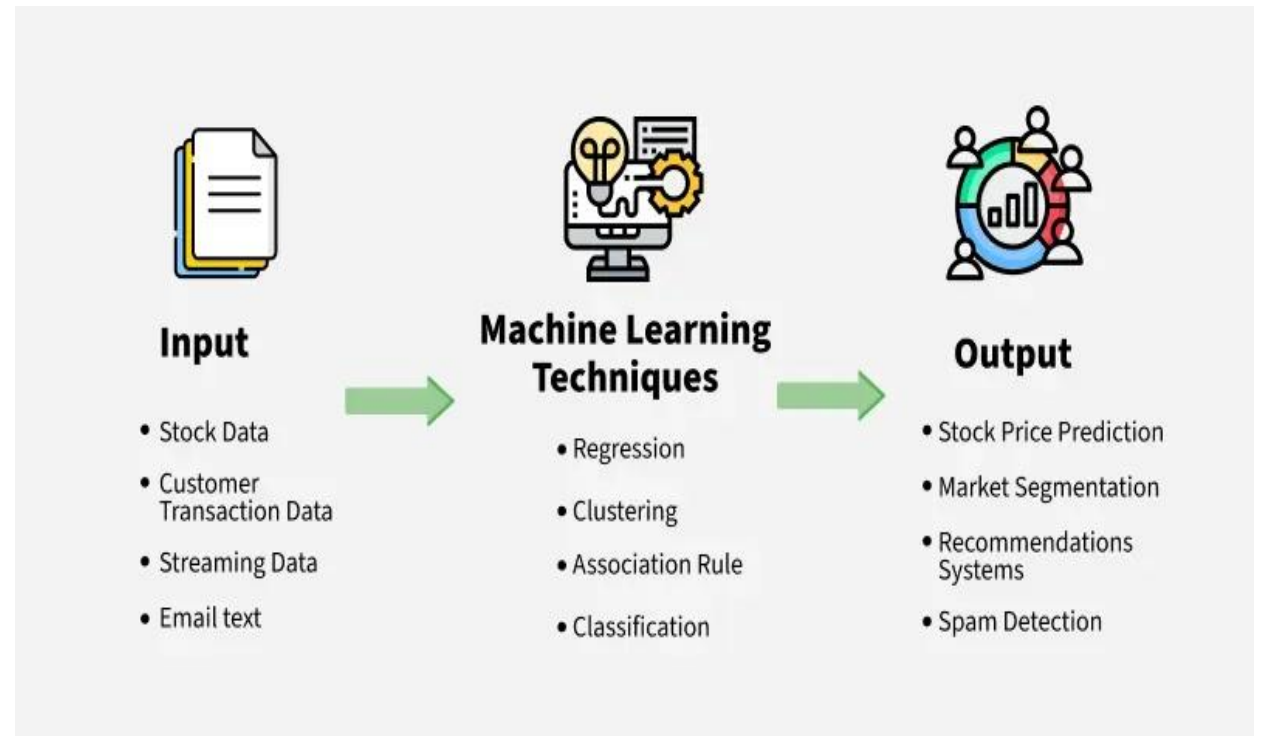


Figure 2.1. Machine Learning Techniques[4]

WHAT IS MACHINE LIFE CYCLE

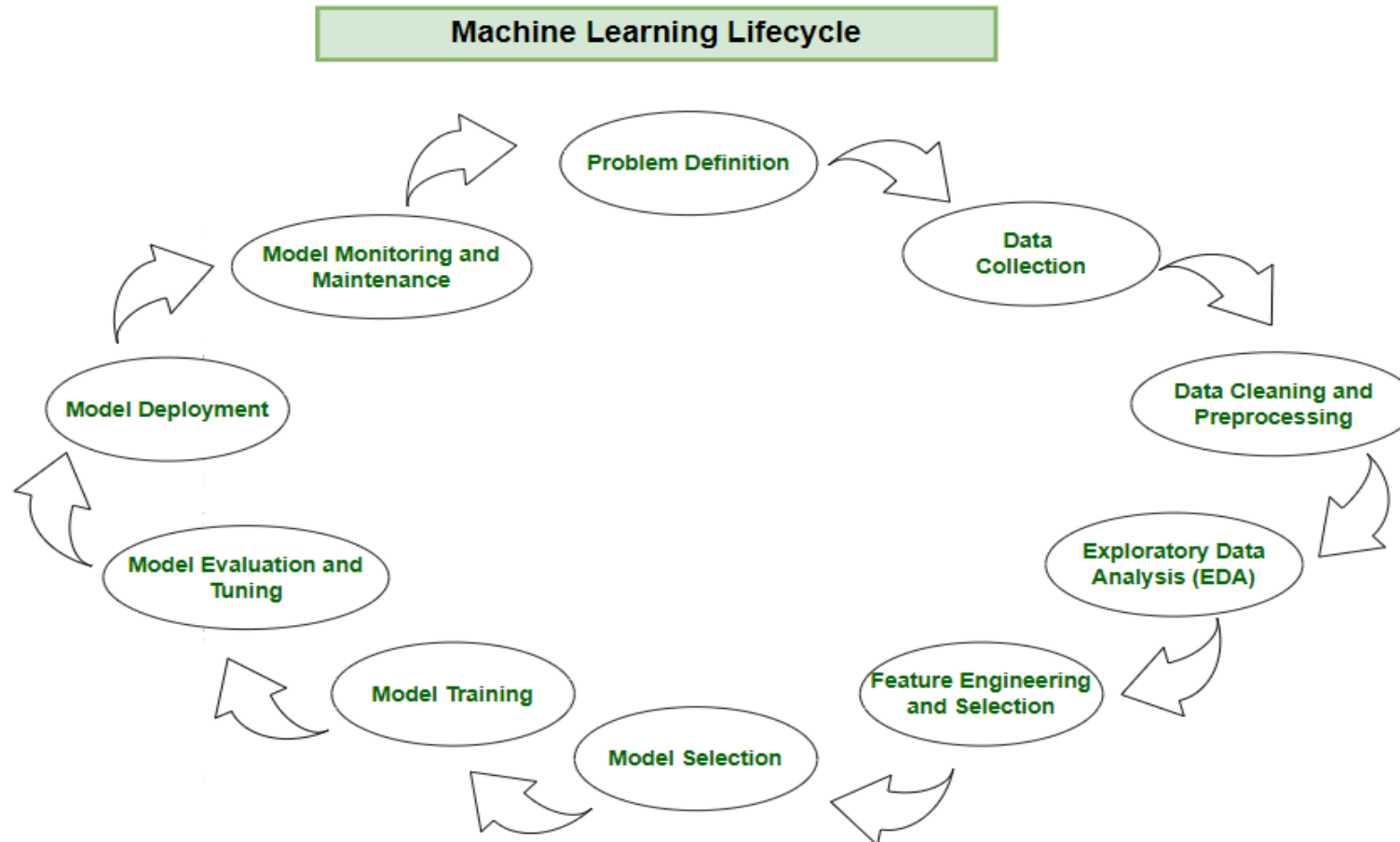



Figure 2.2. Machine Learning Life Cycle[4]



Convolutional Neural Networks (CNNs / ConvNets)

Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

So what changes? ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network [5].



CNN STRUCTURE

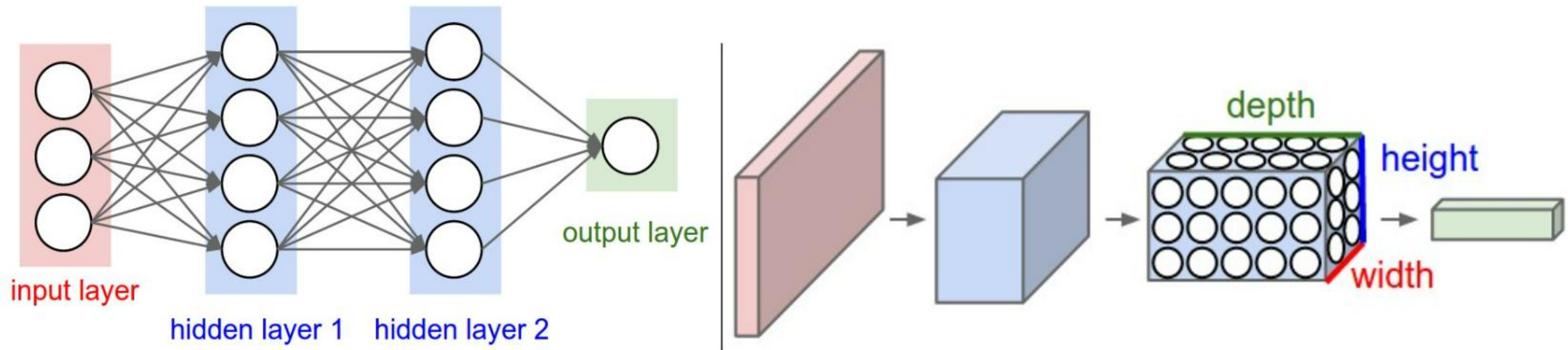


Figure 3.1. Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels) [5].

CNN OPERATION

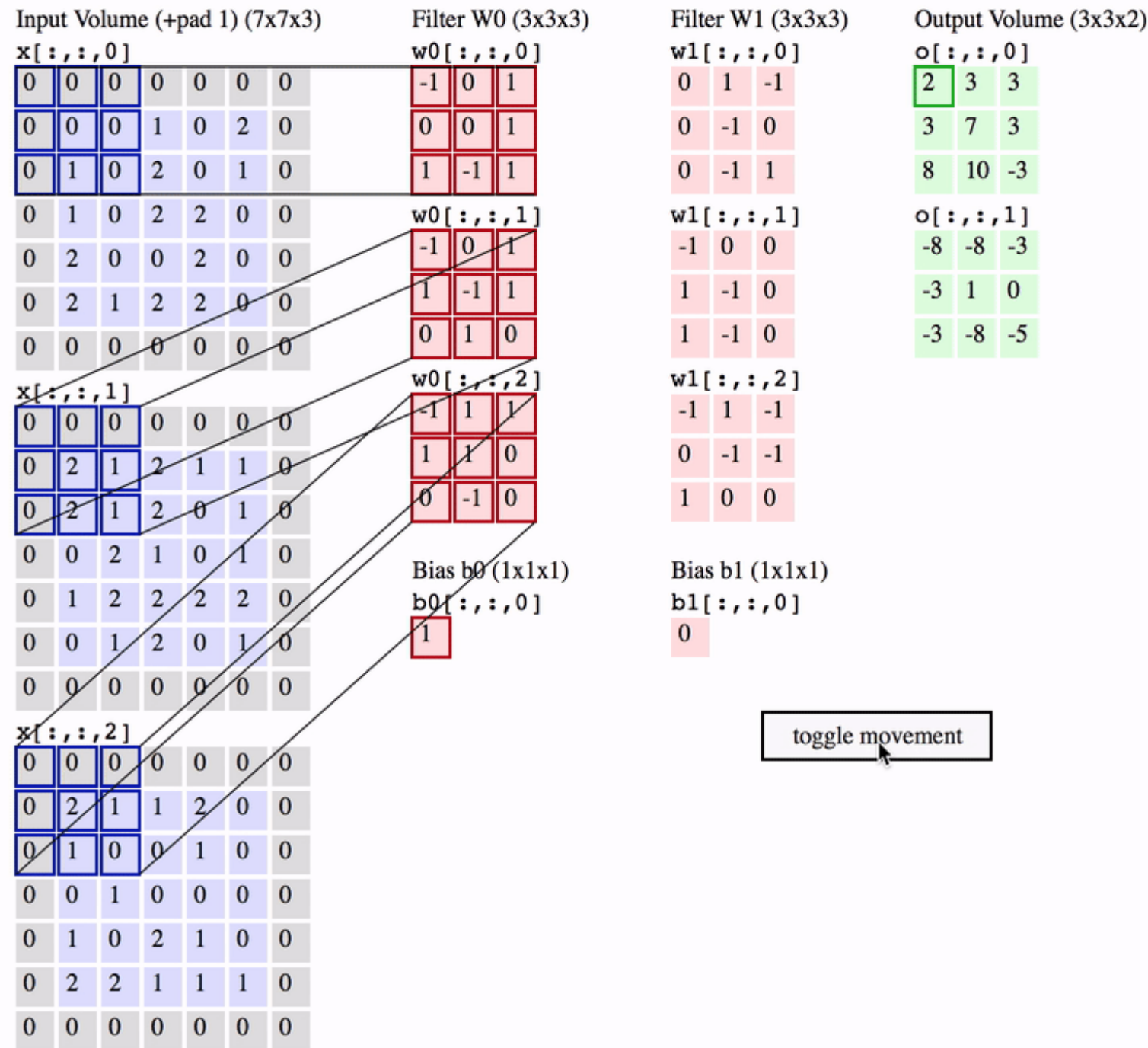


Figure 3.2. Convolution Operation [5]

POOLING LAYER

It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 down samples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged [5].

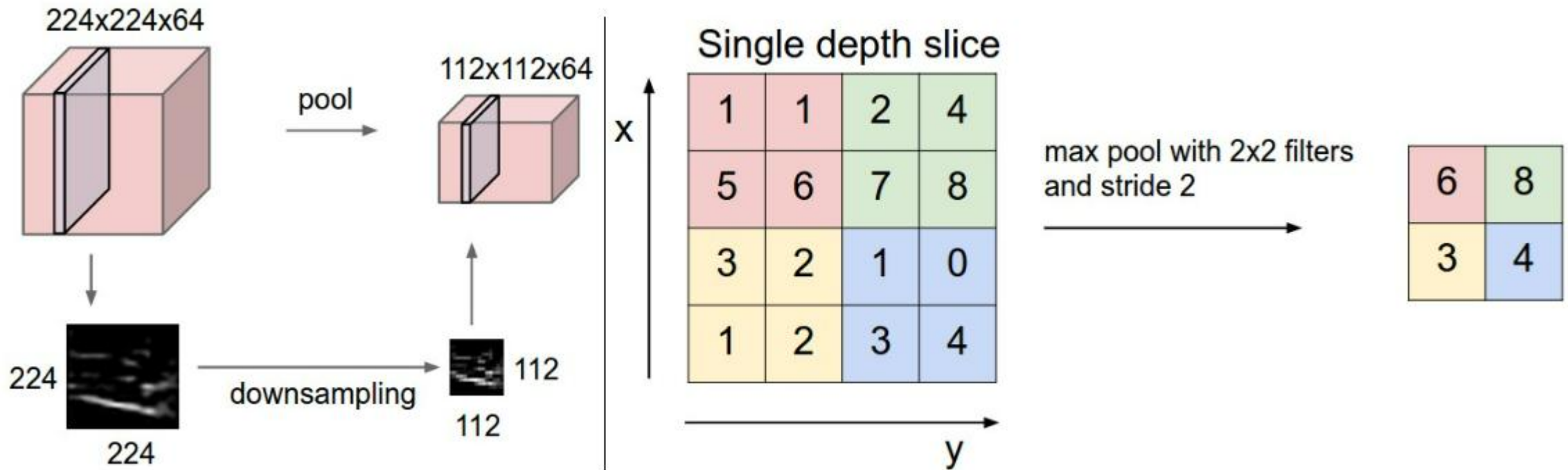


Figure 3.3. Max Pooling Layer [5]



NATURAL LANGUAGE PROCESSING (NLP)

NLP, is the discipline of building machines that can manipulate human language, or data that resembles human language, in the way that it is written, spoken, and organized.

NLP can be divided into two overlapping subfields: natural language understanding (NLU), which focuses on semantic analysis or determining the intended meaning of text, and natural language generation (NLG), which focuses on text generation by a machine. NLP is separate from, but often used in conjunction with, speech recognition, which seeks to parse spoken language into words, turning sound into text and vice versa [6].

SENTIMENT ANALYSIS

Sentiment analysis is the process of classifying the emotional intent of text. Generally, the input to a sentiment classification model is a piece of text, and the output is the probability that the sentiment expressed is positive, negative, or neutral. Typically, this probability is based on either hand-generated features, word n-grams, TF-IDF features, or using deep learning models to capture sequential long- and short-term dependencies. Sentiment analysis is used to classify customer reviews on various online platforms as well as for niche applications like identifying signs of mental illness in online comments.

Give text, sentiment analysis classifiers it's emotional quality [6].

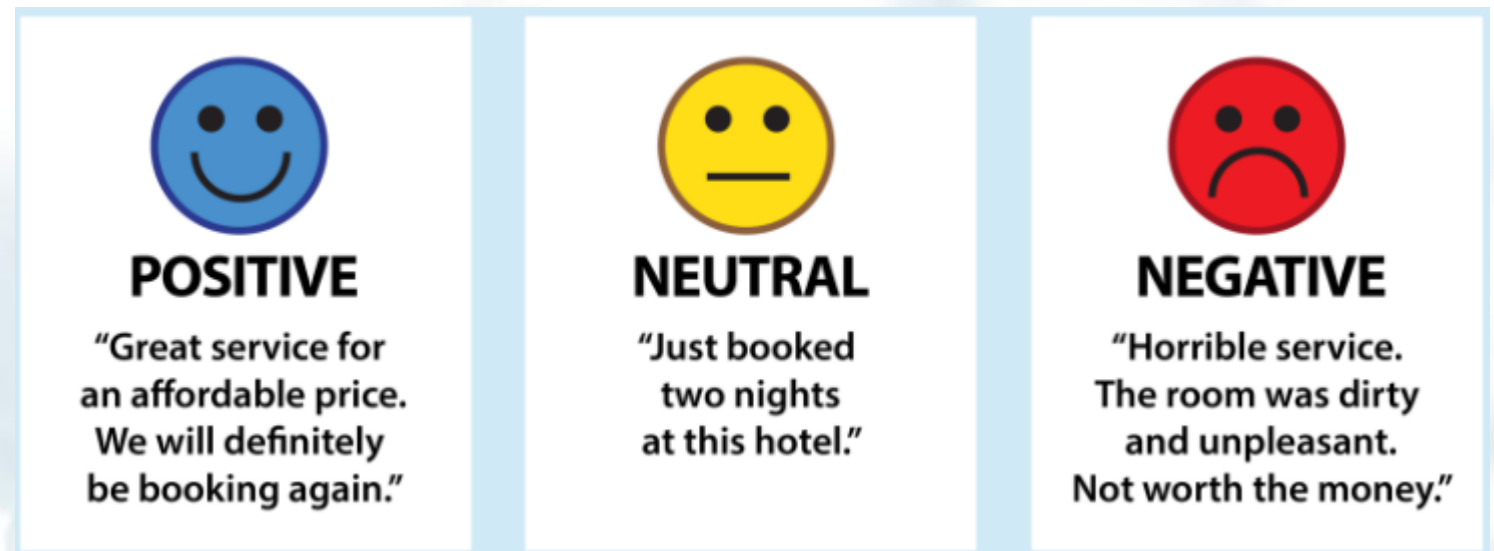


Figure 4.1. Sentiment Analysis [6]

TOKENIZATION

Tokenization splits text into individual words and word fragments. The result generally consists of a word index and tokenized text in which words may be represented as numerical tokens for use in various deep learning methods. A method that instructs language models to ignore unimportant tokens can improve efficiency. **Given a corpus of documents, a tokenizer maps every word to an index. Then it can translate any document into a sequence of numbers [6].**

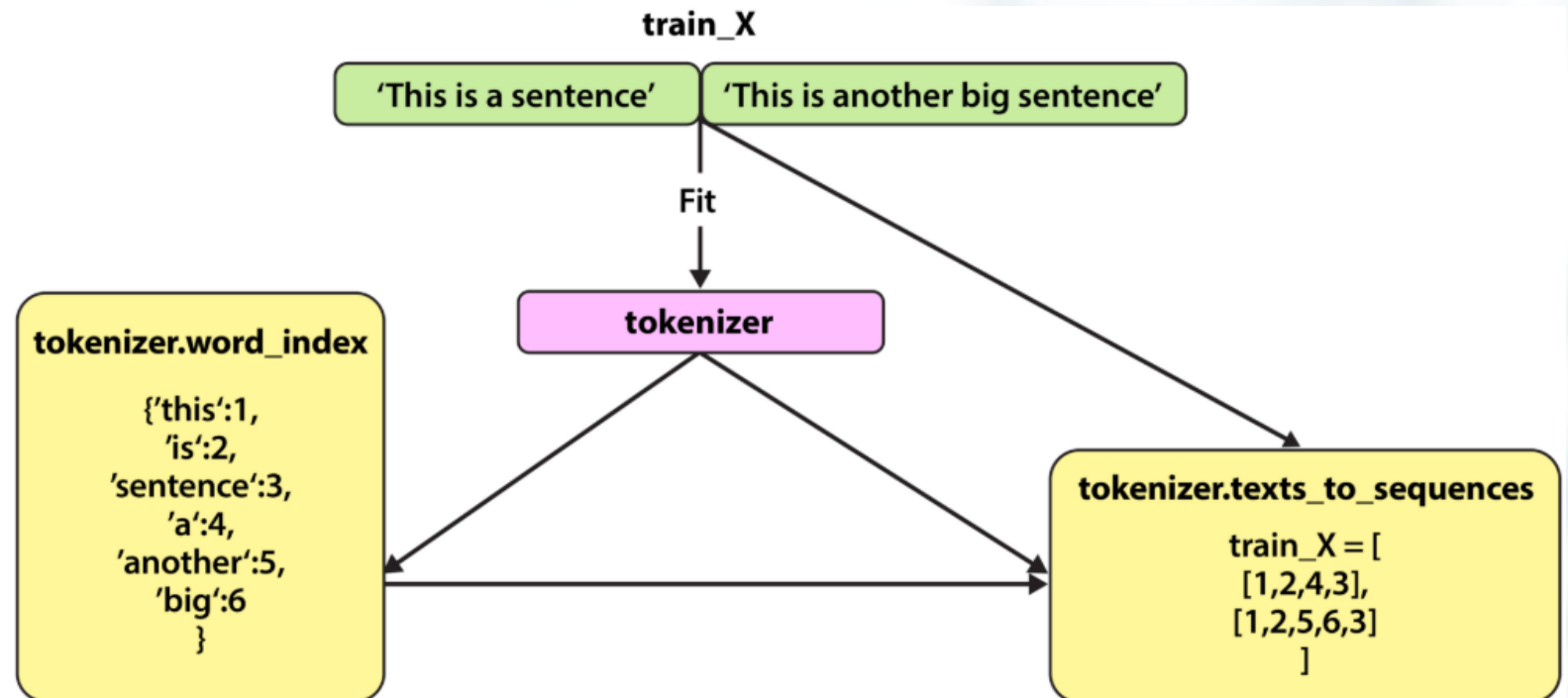


Figure 4.2. Tokenizer [6]

WORD EMBEDDING OR WORD VECTOR

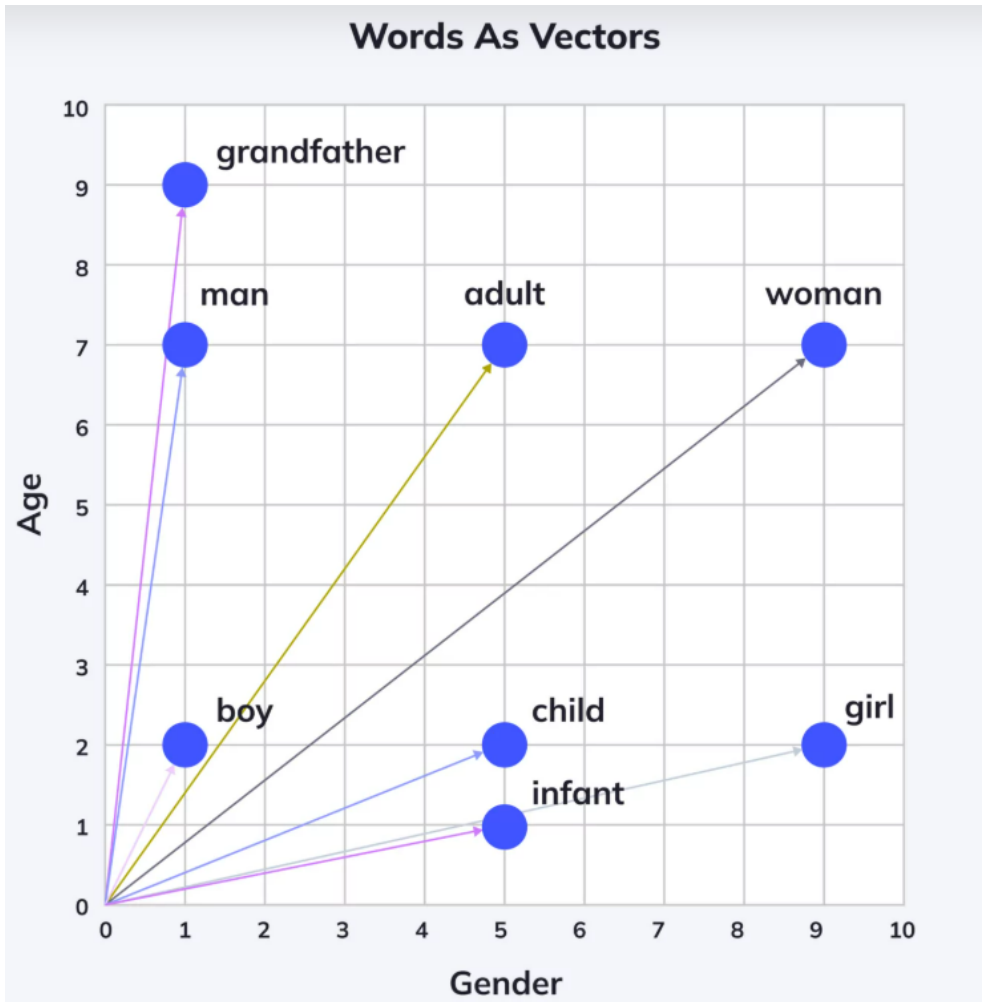


Figure 4.3. Words shown as Vectors [7]

Word embeddings are a key concept in natural language processing (NLP), a field within machine learning. Word embeddings transform textual data, which machine learning algorithms can't understand, into a numerical form they can comprehend. In addition, they can be used to capture the contextual essence of words, their semantic and syntactic similarity, and their relation with other words.

The concept of word embeddings is rooted in the idea that the semantics of a word can be represented in terms of its context. This idea is a departure from traditional bag-of-words models that represent each word as a unique entity, disregarding context and semantics [7].

Need for Word Embedding?

- To reduce dimensionality
- To use a word to predict the words around it.
- Inter-word semantics must be captured [8].

LARGE LANGUAGE MODEL (LLM)

A large language model (LLM) is a type of AI that can process and produce natural language text. It learns from a massive amount of text data such as books, articles, and web pages to discover patterns and rules of language from them.

An LLM is built using a neural network architecture. It takes an input, has a number of hidden layers that break down different aspects of language, and then an output layer. People often report how the next foundational model is bigger than the last - what does this mean? The more parameters a model has, the more data it can process, learn from, and generate [9].

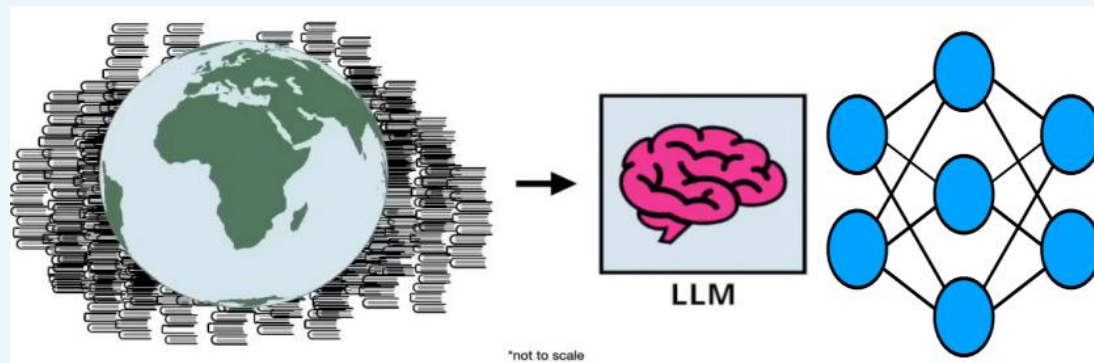
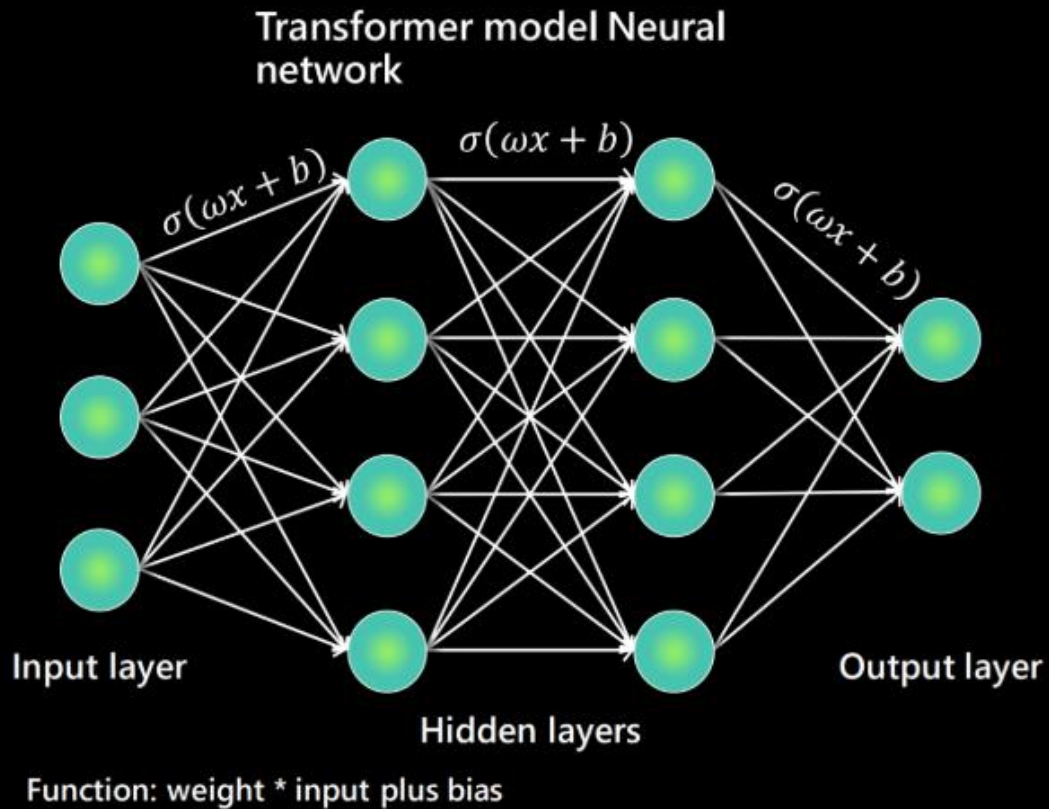


Figure 4.4. LLMs Compress World Knowledge [11]

HOW LARGE IS AN LLM?

How large are they?



BERT Large - 2018

345M

GPT2 - 2019

1.5B

GPT3 - 2020

175B

Turing Megatron NLG
2021

530B

GPT4 - 2023

1.4T (estimated)

Figure 5.1. LLM & CNN, How Large Are LLM [9]

HOW DOES LLM DIFFER FROM NLP?

Traditional NLP

Needs one model per capability

Model trained on a finite set of labelled data

Highly optimized for specific use case

[9]

Large Language Models

Reuse single model for many NLP use cases

Foundation model trained on many TBs of unlabelled data

Open-end usage – use natural language to “prompt” the model to do something

[9]

WHAT DOESN'T A LLM DO?

While large language models drive rich and powerful generative AI experiences, it's important to remember that the LLM:

- **Does NOT Understand language.** It's just a predictive engine. Based on text patterns it has previously seen, it can *predict completions* for the given text input. It does not understand the context or meaning of that content - e.g, it does not understand math.
- **Does NOT Understand facts.** There are no separate 'modes' for *information retrieval* vs. *creative writing*. The model just predicts the next most probable token in the ongoing sequence.
- **Does Not Understand manners, emotion or ethics.** Don't anthropomorphize LLMs by attributing human characteristics to them, or claiming they "understand" something. The output is simply the outcome of training data guided by the given prompts [9].

RETRIEVAL AUGMENTED GENERATION (RAG)

Retrieval augmented generation, or RAG, is an architectural approach that can improve the efficacy of large language model (LLM) applications by leveraging (yukselme) custom data. This is done by retrieving data/documents relevant to a question or task and providing them as context for the LLM. RAG has shown success in support **chatbots** and **Q&A** systems that need to **maintain up-to-date information** or access domain-specific knowledge [10].

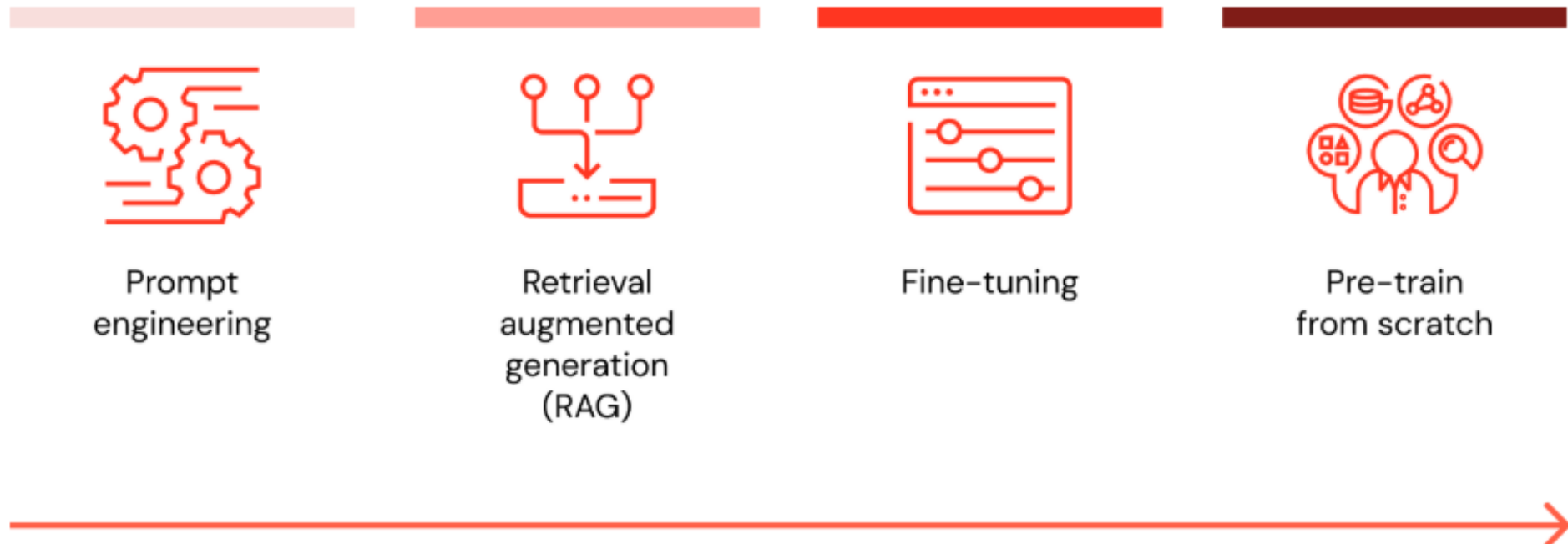



Figure 6.1. Complexity / Compute-intensiveness [10]




When I want to customize my LLM with data, what are all the options and which method is the best (prompt engineering vs. RAG vs. fine-tune vs. pretrain)?

There are four architectural patterns on the previous slide to consider when customizing an LLM application with your organization's data. These techniques are outlined below and are **not mutually exclusive**. Rather, they can (and should) be combined to take advantage of the strengths of each [10].

PROMPT ENGINEERING, RAG, FINE-TUNING, PRETRAIN

Method	Definition	Primary Use Case	Data Requirements	Advantages	Considerations
Prompt Engineering	Crafting specialized prompts to guide LLM behavior	Quick, on-the-fly model guidance	None	Fast, cost-effective, no training required	Less control than fine-tuning
Retrieval Augmented Generation	Combining an LLM with external knowledge retrieval	Dynamic datasets and external knowledge	External knowledge base or database (e.g., vector database)	Dynamically updated context, enhanced accuracy	Increases prompt length and inference computation
Fine-tuning	Adapting a pretrained LLM to specific datasets or domains	Domain or task specialization	Thousands of domain-specific or instruction examples	Granular control, high specialization	Requires labeled data, computational cost
Pretraining	Training an LLM from scratch	Unique tasks or domain-specific corpora	Large datasets (billions to trillions of tokens)	Maximum control, tailored for specific needs	Extremely resource-intensive

Table 1.1. Which Method is The Best (Prompt Engineering, RAG, Fine-tuning, Pretrain) [10]



When I want to customize my LLM with data, what are all the options and which method is the best (prompt engineering vs. RAG vs. fine-tune vs. pretrain)?

There are four architectural patterns on the previous slide to consider when customizing an LLM application with your organization's data. These techniques are outlined below and are not mutually exclusive. Rather, they can (and should) be combined to take advantage of the strengths of each [10].

WHAT IS A REFERENCE ARCHITECTURE FOR RAG APPLICATION?

There are many ways to implement a retrieval augmented generation system, depending on specific needs and data nuances. Below is one commonly adopted workflow to provide a foundational understanding of the process [10].

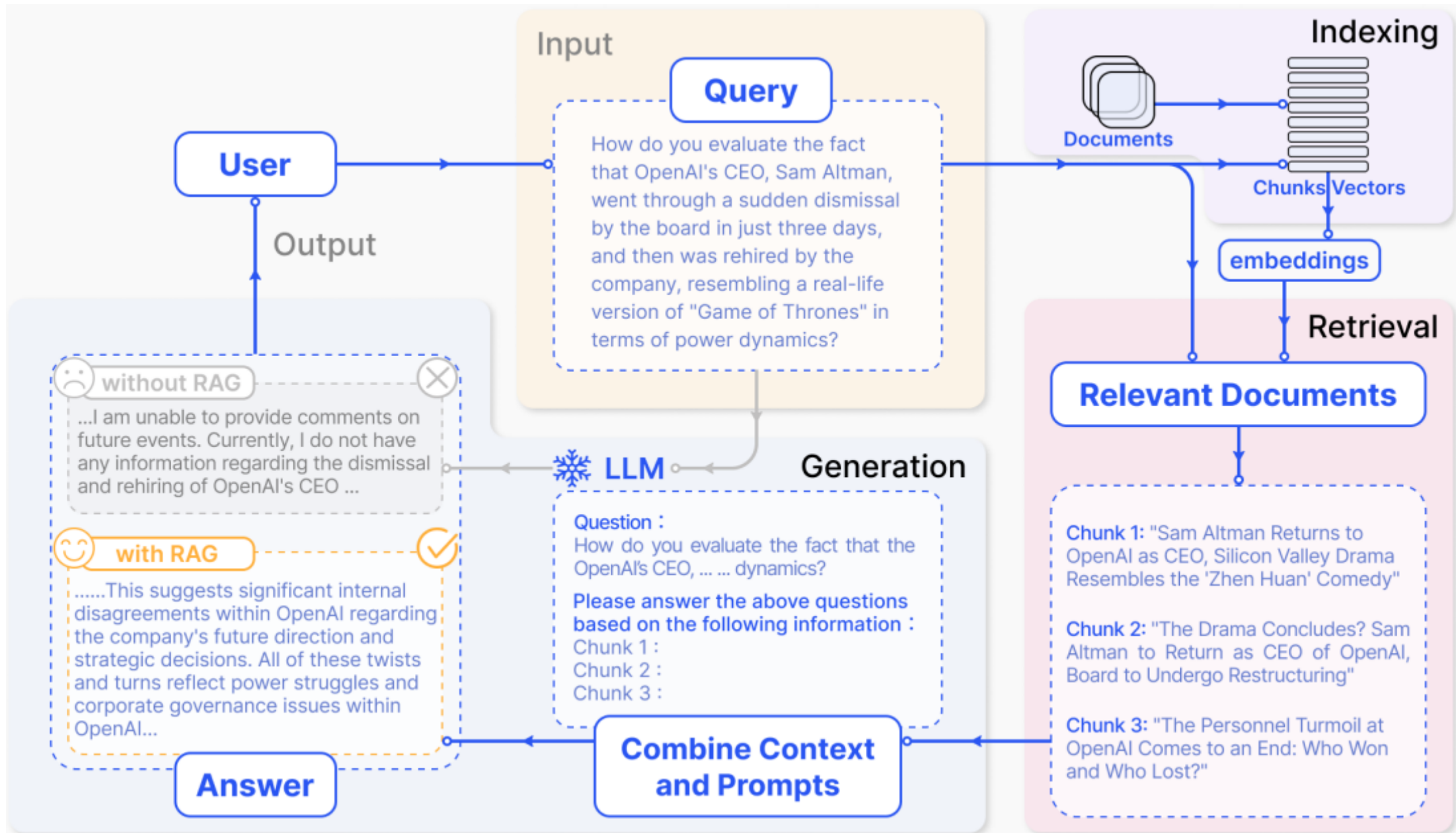


Figure 6.2. Overview of RAG [12]

SOURCES



- [1]: <https://cloud.google.com/learn/what-is-artificial-intelligence>
- [2]: [https://en.wikipedia.org/wiki/Artificial_intelligence#/media/File:Dall-e_3_\(jan_'24\)_artificial_intelligence_icon.png](https://en.wikipedia.org/wiki/Artificial_intelligence#/media/File:Dall-e_3_(jan_'24)_artificial_intelligence_icon.png)
- [3]: <https://www.ibm.com/think/topics/machine-learning>
- [4]: <https://www.geeksforgeeks.org/ml-machine-learning/>
- [5]: <https://cs231n.github.io/convolutional-networks/>
- [6]: <https://www.deeplearning.ai/resources/natural-language-processing/>
- [7]: <https://swimm.io/learn/large-language-models/5-types-of-word-embeddings-and-example-nlp-applications>
- [8]: <https://www.geeksforgeeks.org/word-embeddings-in-nlp/>
- [9]: <https://microsoft.github.io/Workshop-Interact-with-OpenAI-models/llms/>
- [10]: <https://www.databricks.com/glossary/retrieval-augmented-generation-rag>
- [11]: https://github.com/ShawhinT/YouTube-Blog/blob/main/LLMs/_slides/rag.pdf
- [12]: Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, H. Wang “Retrieval-Augmented Generation for Large Language Models: A Survey” *arXiv:2312.10997v5 [cs.CL]* 27 March 2025



THANK YOU

AMIN ASLAMI | AMINASLAMIAF@GMAIL.COM | LINKEDIN.COM/IN/AMIN-ASLAMI/