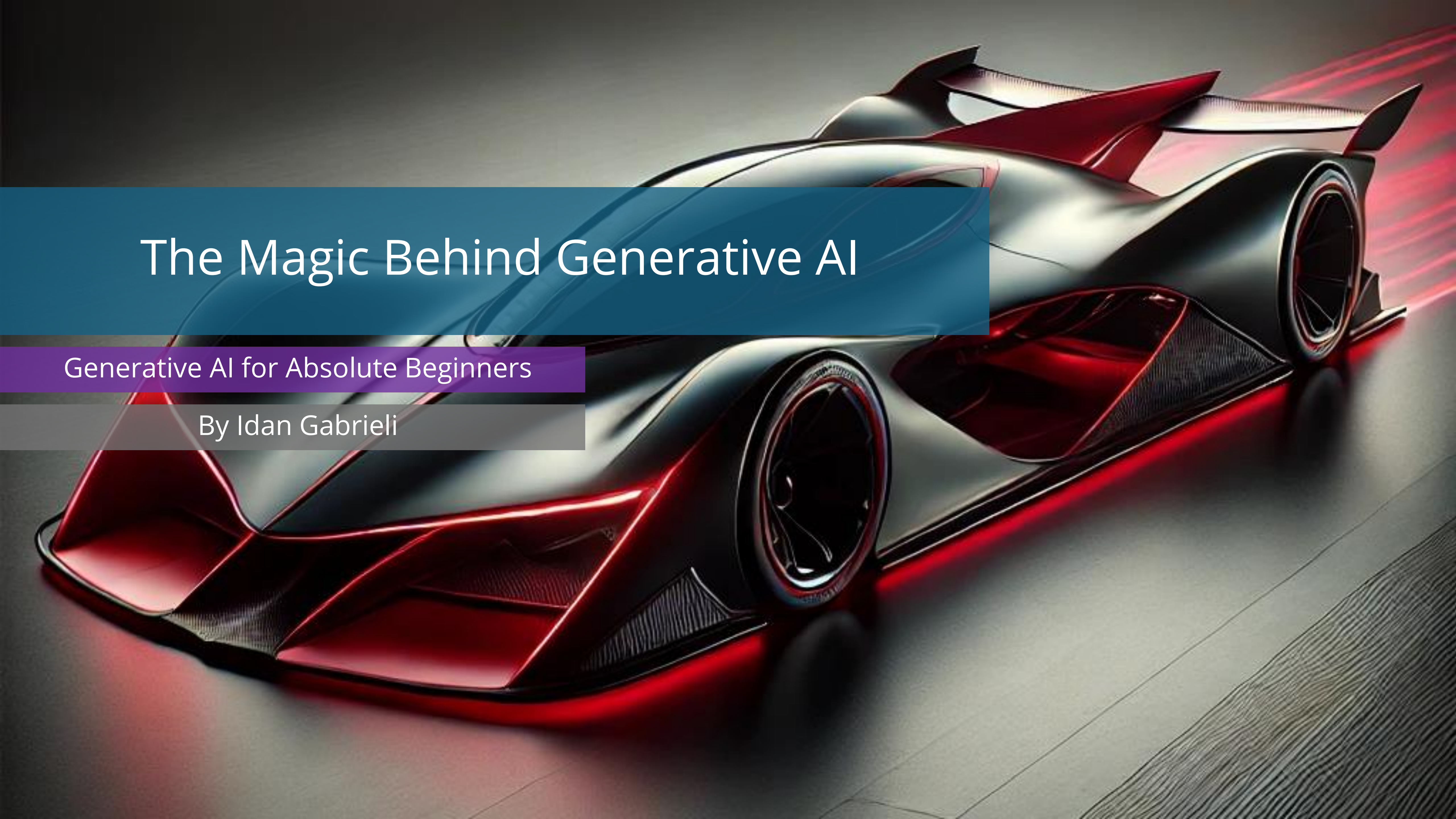
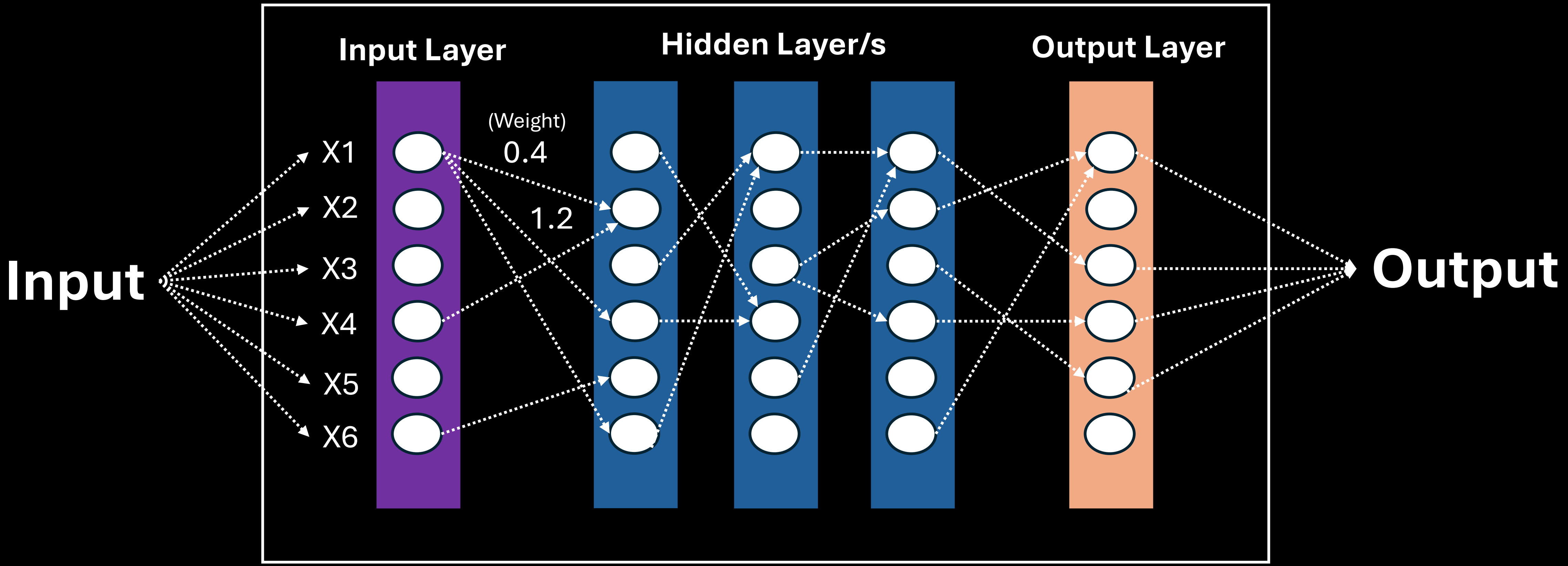


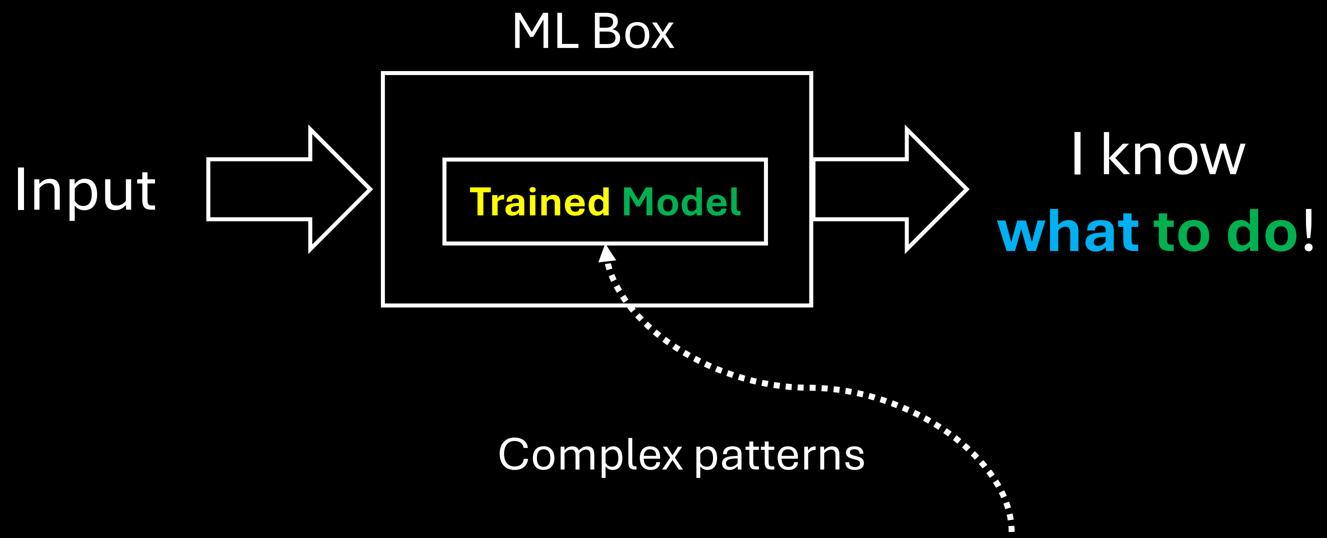
The Magic Behind Generative AI

Generative AI for Absolute Beginners

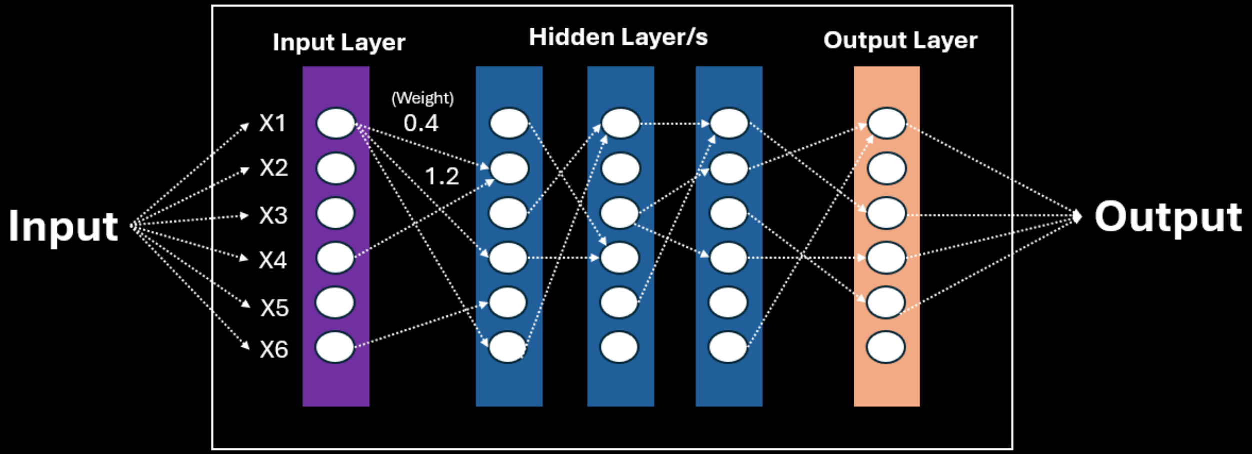
By Idan Gabrieli

Artificial Neural Networks (ANNs)





Deep Learning



Deep Learning Architectures

#1 – Recurrent Neural Networks

- Processing input data **sequentially**
- Slowly adjusting the internal model state
- Less effective for a very large amount of data

#2 – Convolutional Neural Networks

- Designed for image and video processing tasks

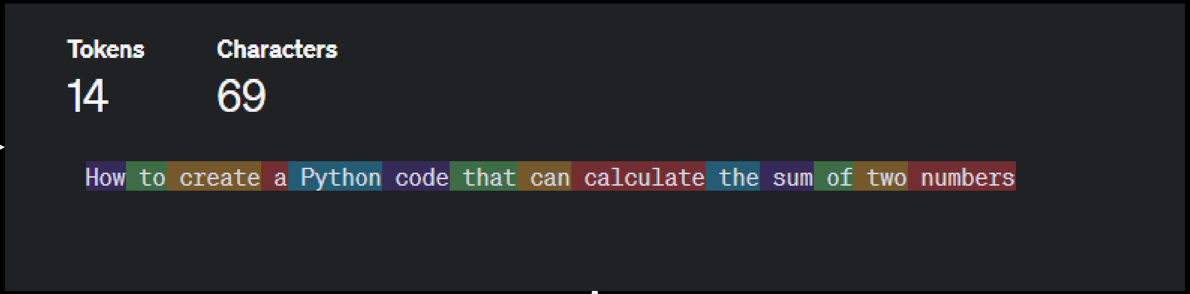
#3 – Transformers Architecture

- Processing input data **in parallel**
- Leverage **GPUs** to train models
- **Attention** mechanism → pay more attention to specific elements

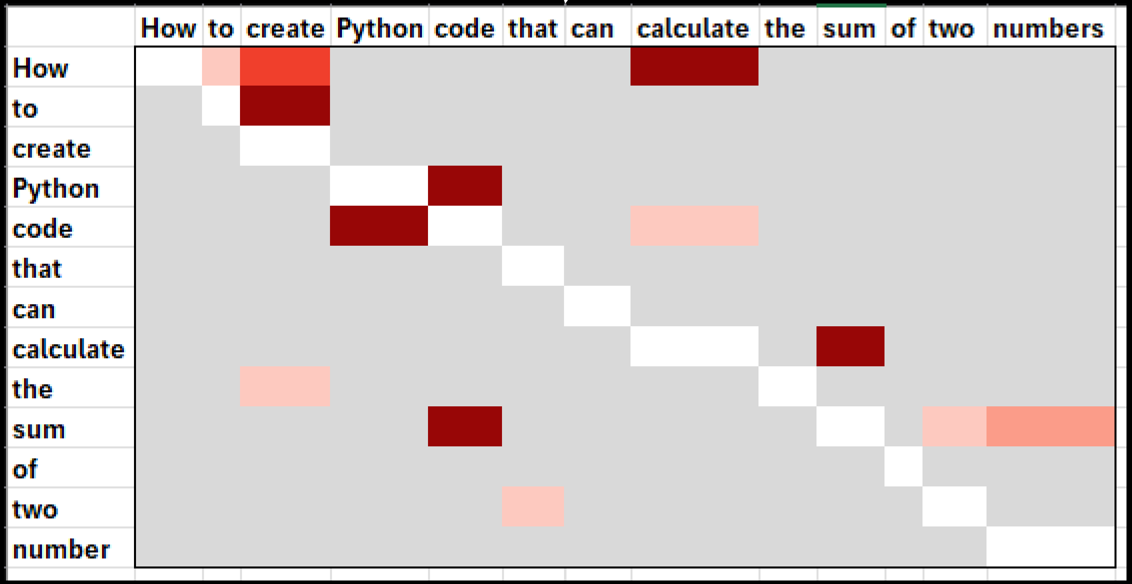
#3 – Transformers Architecture

"How to create a Python code that can calculate the sum of two numbers."

Break to tokens



Calculates attention scores



#3 – Transformers Architecture

- Processing input data **in parallel**
- Leverage **GPUs** to train models
- **Attention** mechanism → pay more attention to specific elements

Training more complex models

Input Text → **LLMs** → **Output**

Foundation models

Deep Learning
Architectures

Transformers Architecture

Train **more complex** **generic** models

Foundation Models

- **Large-scale** more generic models
- Trained on **massive** amounts of data
- Can be adapted to perform a **wide range** of tasks
- **Knowledge** on a variety of topics, millions of topics
- An expensive resource-intensive project

Job for the **BIG** players

Foundation Models

A foundation model is a **building block**

LLMs

Generative Pre-Trained Transformer (GPT)

Running ChatGPT

Generative

Generate content

Pre-Trained

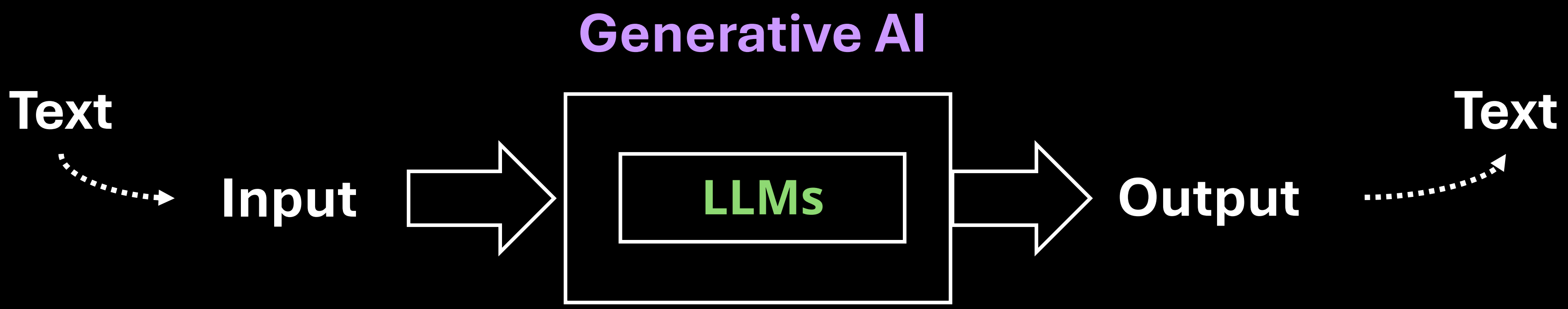
A foundation model

Transformer

Transformer Arch.

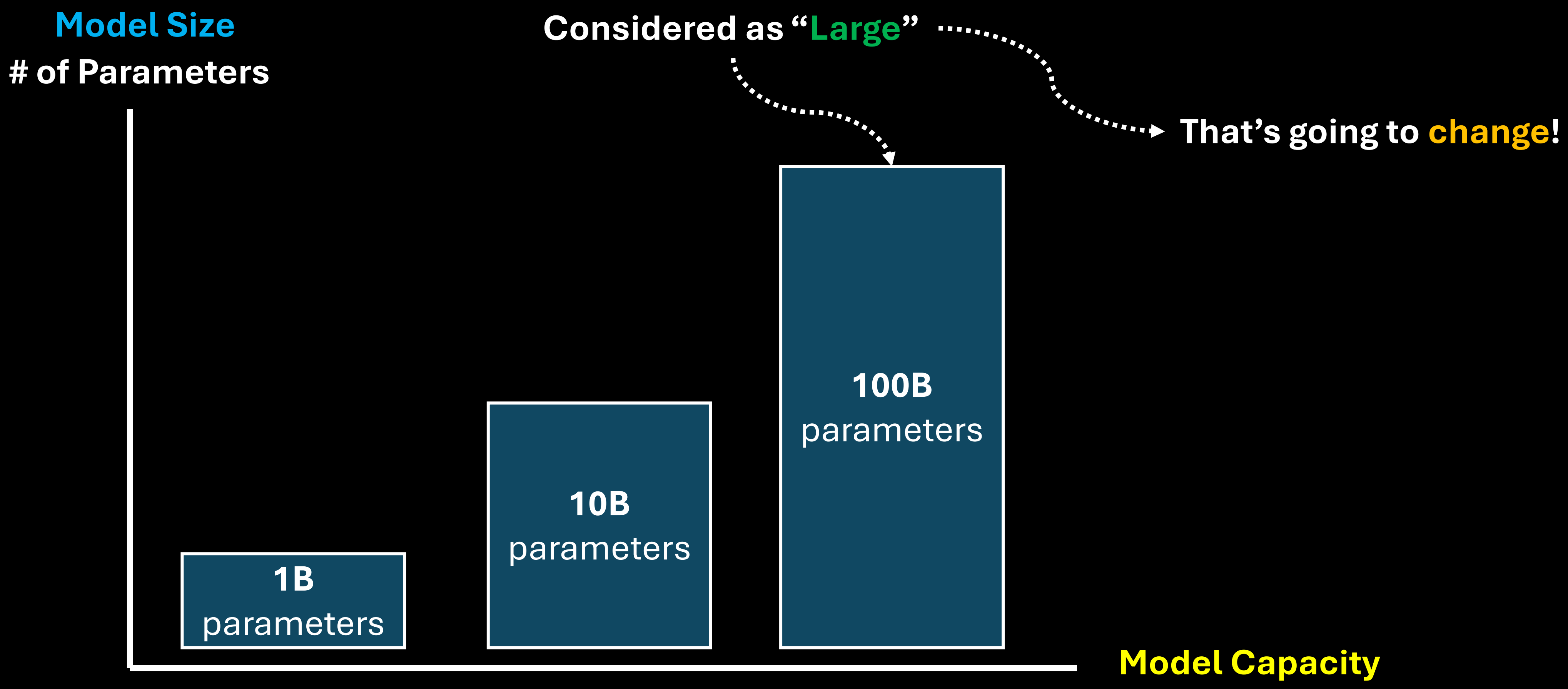
Foundation Models

Large Language Models (LLMs)



- ✓ Core capabilities of generative AI
- ✓ Handle **text** as input + output
- ✓ Analyze, understand, and generate complex responses
- ✓ ChatGPT is based on LLM

Not all LLMs are created equal → Model Size! (# of parameters)



Get me a BIGGER model?

- More computing resources to train and deploy
- Required data set and volumes to train the model
- Higher cost of cloud resources
- A larger team with more skillset

A bigger model isn't always better for **a specific use case**.

Food Type and Geo-Location → LLMs → Recommended Restaurant

Large Language Models (LLMs)

#1 – General-purpose vs. Domain-specific

General-purpose LLMs
- Handle a wide range of tasks
- trained by taking massive amounts of data
- ChatGPT/Google Gemini

Domain-specific LLMs
- also called specialized LLMs
- trained to handle tasks related to a specific domain
- Fine-tuned to handle more niche areas of a specific domain

Large Language Models (LLMs)

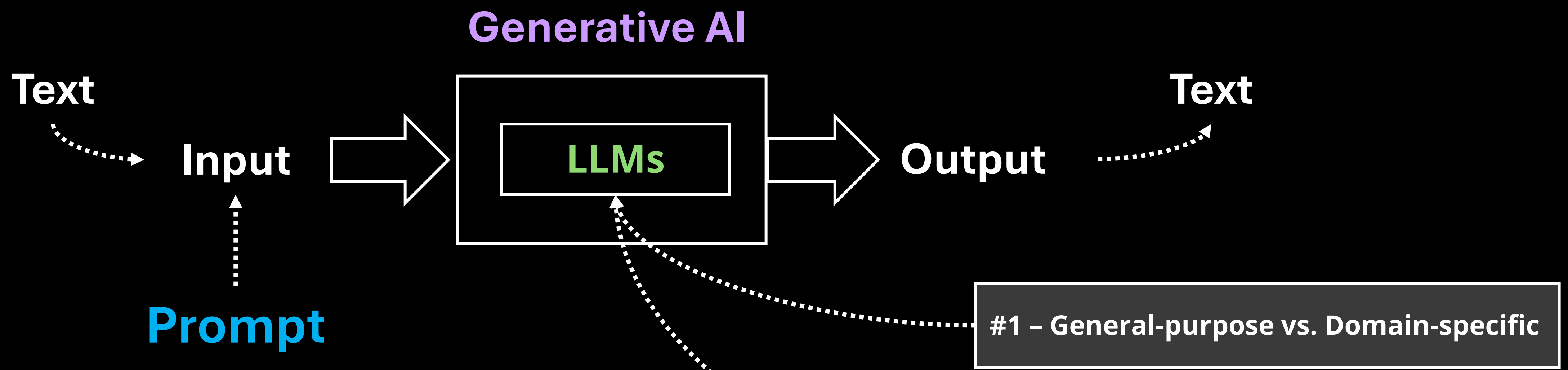
#2 – Open-source vs. Closed-source

Open-source LLMs

- Available to the public to be download
- Change and customize the model
- Full control over the model
- Lower risk of data leakage

Closed-source LLMs

- Proprietary models owned by companies
- Web-based services or via APIs
- Services that are monetizing the trained LLM
- More optimized for production
- Much faster to deploy (APIs)



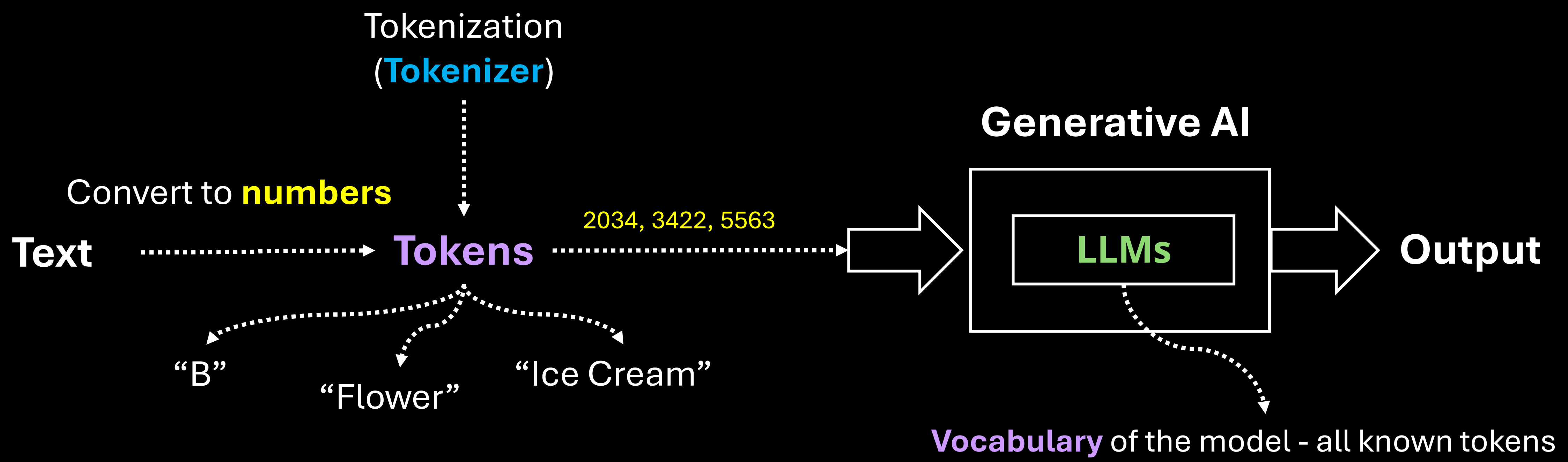
What is a prompt?

Control the output of the model

Group of words, sentences, and paragraphs

A typical natural language will have a **HUGE** number of possible words

A **TEXT** format is not the most efficient way to process data for a machine learning system



Numerical representations of characters, words or phrases

Units of text that the model processes

An **LLM** is getting a sequence of **tokens** created by a **tokenizer** breaking an input text to tokens

Total Tokens and Context Window

Tokenization
(**Tokenizer**)

Text → **Tokens** → 2034, 3422, 5563 → **LLMs** → **Output**

Tokens are a fundamental **metric** for measuring **usage**

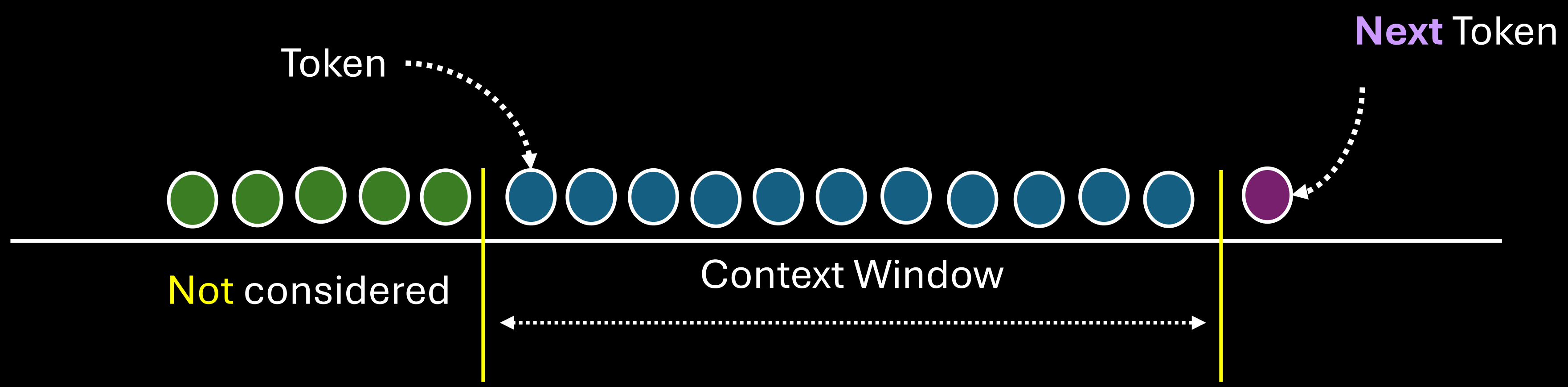
Track and **limit** the usage of Generative AI services

Model limitations – # of tokens

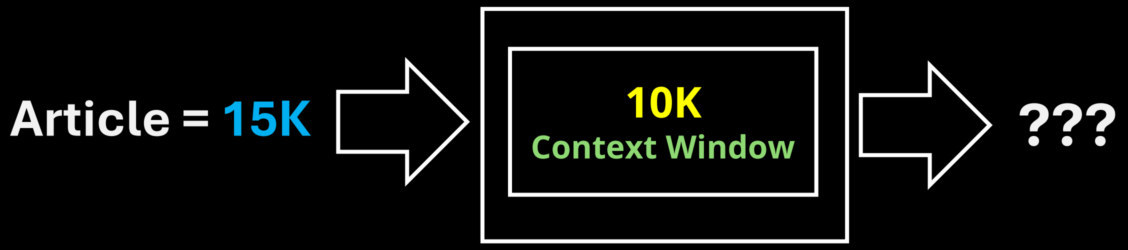
Optimize prompts
(less tokens)

My App ⇄ 3rd party **Gen AI Service**

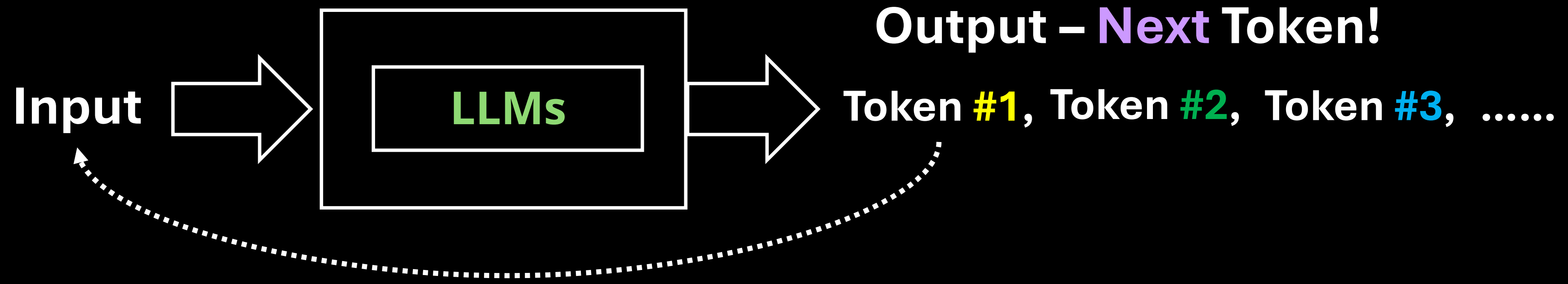
Context Window



Max # of tokens a model can process and consider at once when generating a new token



Next Token Please!



Identify a pattern and use it to predict the **next token**

An LLM model is operating in a **sequential** mode

- Predict **one single** token
- A sequential list of related tokens
- Range limit of the model context window

How does the next token is **generated?**

- **Probability distribution** of next token
- Tuned parameters of the statistical relationships between tokens

Input: "The cat sat on the [**Next Token**]"

Predict the next token based on calculating which **list of tokens** have a stronger statistical correlation

Token	Probability
Fence	12%
Sofa	**36%**
Roof	8%
Floor	**28%**
....	

Add some level of randomness

LLMs

Prompt input

Divided into tokens

Context window

After the training phase

Statistical relationships

Predict NEXT token

Trained using massive amounts of unstructured data

What is the **method** being used?

Self-Supervised Learning

A model is trained **using the data itself** in a supervised manner **without** external labels

Self-Supervised Learning

One Page

Sentence **#1** Sentence **#2**

Sentence **#3** ......

Self-Supervised Learning

Sentence **#1**

Sentence **#2**

Sentence **#3**

LLM

Masked Language Modeling

Masking → Predicting → Learning

"I love eating cheese pizza."

#1: "I love eating [MASK] pizza."

#2: pizza topping

#3: Error between Masked and Predicted value

#4: Optimize the model parameters

Token	Probability
pepperoni	10%
cheese	**70%**
margherita	20%
veggie	**5%**
....	

Self-Supervised Learning

Millions of sentences, billions, or **trillions** of words → LLM

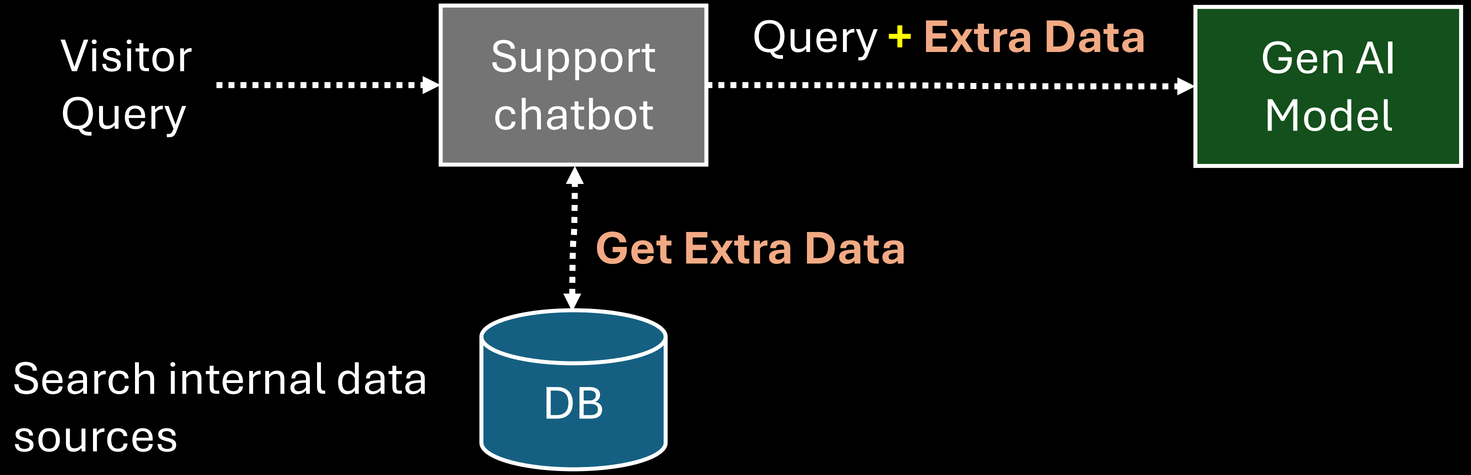
- ✓ Predict missing words with **increasing accuracy**
- ✓ Fully automated
  - → Massive scalability
  - → Leverage vast amounts of unstructured data
- ✓ Develop a deep understanding of the language
- ✓ Called: "Pre-training" step

#1 - Contextual Prompting

- Prompt engineering
- Articulate **as clearly as** possible the required task
  - **Context**
- Most cost-effective way to tune the responses

#2 - Retrieval Augmented Generation (RAG)

- Retrieve additional data from **external** knowledge sources
- Better handle private data and limited model knowledge
- **Drawbacks**
  - Context window
  - Latency
  - Cost of large # of tokens



#3 - Fine-Tuning

- **Two-step** training process
  - **Pre-Training**
    - Train foundation models
    - Massive amount of data
    - Done by the owner of the model
- Re-train a foundation model
- Output: Fine-tuned model
  - Transfer learning
  - Task-specific model
- Based on a smaller amount of data vs. pre-training
- Closed-source LLMs - Dedicated **APIs** for fine-tuning
- **Key benefits**
  - Latency
  - Optimized prompt size
- Must be done carefully!