

# A Whirlwind Tour of LLMs

April 9, 2024  
QMSS GR5069



Eitan



Jacob

# A bit about our journey



Naveen Rao's MosaicML debuts with a mission to improve machine learning training

for Open-Source, Commercially Usable LLMs

## Introducing MPT-7B: A New Standard for Open-Source, Commercially Usable LLMs

by The MosaicML NLP Team

May 5, 2023 in Mosaic AI Research

FORBES > INNOVATION > CLOUD

## Databricks Acquires MosaicML To Make Generative AI More Accessible

Janakiram MSV Senior Contributor

I cover emerging technologies with a focus on infrastructure and AI

0



WILL KNIGHT BUSINESS MAR 27, 2024 8:00 AM

## Inside the Creation of the World's Most Powerful Open Source AI Model

Startup Databricks just released DBRX, the most powerful open source large language model yet—eclipsing Meta's Llama 2.

# LLM Trivia

- How many people have used LLMs besides GPT4? Which ones?
  - Perplexity? You.com? Bing?
- How much GPT4 cost to train?
- What data went into GPT4?
- How many people here believe the US should allow companies to open source LLM models?
- Who here is worried about the risks of AI? Which risks in particular?
- Do people think that AI will replace jobs? Or “enhance productivity?” Or both?

# Why should we care about LLMs?

🤓 Nerd factor – it is incredibly cool that these things work!

Economics – [NVIDIA \\$2T valuation](#), [OpenAI valuation at \\$80B](#)

Political – [US Chips Act](#), [TSMC to build factory in the US](#),  
[France invests €500M in AI](#), [Canada invests \\$2B in AI](#),  
[Saudi Arabia raises \\$40B AI fund](#),

Social – [will AI replace employees?](#) [Will ChatGPT ruin the college education?](#) [AI companions](#),

Science – [protein folding](#)

...not to mention the potential harm such as surveillance and misinformation

## France to invest €500 million to fund AI 'champions', Macron says

President Emmanuel Macron made a pitch on Wednesday for France to become a leader in artificial intelligence (AI) as he spoke at VivaTech in Paris, one of Europe's biggest technology trade shows.

Issued on



Administration

AUGUST 09, 2022

FACT SHEET: CHIPS and Science Act  
Will Lower Costs, Create Jobs,  
Strengthen Supply Chains, and  
Counter China

Bloomberg

Live TV Markets ▾ Economics Industries Tech Politics Businessweek Opinion More ▾

Technology

**TSMC Gets \$11.6 Billion in US Grants, Loans for Chip Plants**

- World's top chipmaker to add capacity with third fab in state
- Raimondo says latest US package aimed at AI, national security

# Main themes

- How do LLMs work?
- What LLMs should you use and why?
- What evaluation benchmarks matter and why?
- When should you finetune?
- What are some rules of thumb around hardware?
- What can these models be used for?
- What are these models bad at?
- What are open research questions?

# Part 1: LLM Overview

# What we mean when we talk about LLMs

Proprietary (closed source) LLMs:

- ChatGPT, GPT4 (OpenAI)
- Claude (Anthropic)
- Gemini (Google)
- Pi (Inflection)

Open Source LLMs

- LLama-7B, 13B, 70B (Meta)
- Command-R (Cohere)
- DBRX (Databricks)
- Mistral (Mistral AI)

...

BERT, T5, GPT2, OPT, Pythia

...



# The Stages of building and serving LLMs

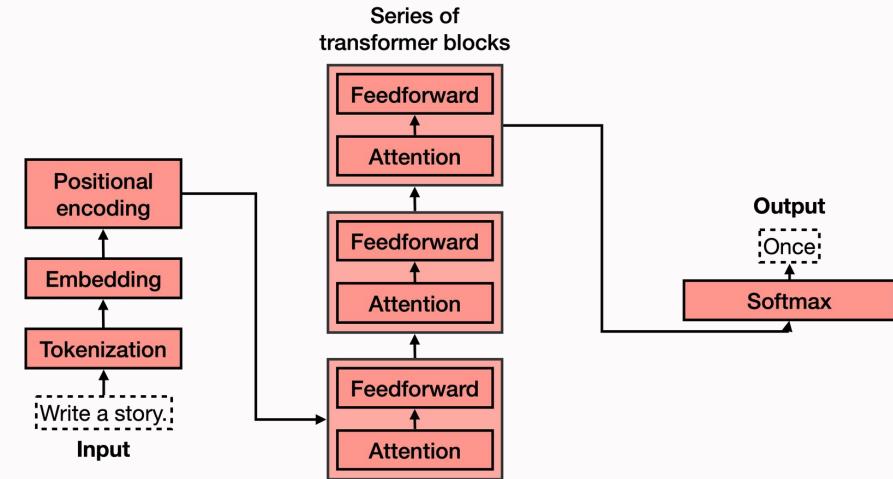
- Data Collection
- Pretraining
- Finetuning
- Evaluation
- Serving

# Pretraining

Next token prediction (aka causal language modeling)

"Unsupervised" – the data does not need any human labelers

Core architecture is the transformer block



Hugging Face is a startup based in New York City and Paris

$p(\text{word})$

# Pretraining Data

## Example Datasets

- “[C4](#)” 750GB (from the [CommonCrawl](#) nonprofit open repository of web crawls)
- ThePile (famously contains “[Books3](#)” copyrighted books)

The screenshot shows two news articles from The Washington Post. The first article is titled "Sarah Silverman-Led AI Copyright Suit Against ChatGPT Partially Dismissed" and discusses a ruling that nixed some core claims from a suit brought against Meta and its AI tools. The second article is titled "The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work" and states that millions of articles from The New York Times were used to train chatbots that now compete with it. Both articles are by Jon Blinstein.

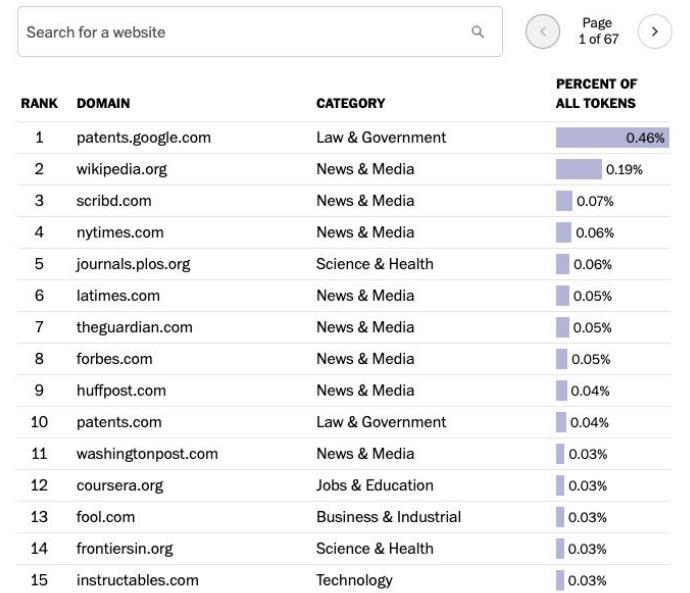
## What is good data?

- Open source textbooks, well-written forum threads

## What is bad data?

- Html tags, poorly scraped websites

### The websites in Google's C4 dataset



The Post believes it is important to present the complete contents of the data fed into AI models, which promise to govern many aspects of modern life. Some websites in this data set contain highly offensive language and we have attempted to mask these words. Objectionable content may remain.

Note: Some websites were unable to be categorized and, in many cases, are no longer accessible.

[\[Washington Post\] Inside the secret list of websites that make AI like ChatGPT sound smart \(April 2023\)](#)

# Pretraining Data

Example: NYT Lawsuit against OpenAI has many examples of ChatGPT regurgitating NYT content verbatim

105. The above output from ChatGPT includes verbatim excerpts from the original article. The copied article text is highlighted in red below:

The snow burst through the trees with no warning but a last-second whoosh of sound, a two-story wall of white and Chris Rudolph's piercing cry: "Avalanche! Elyse!"

The very thing the 16 skiers and snowboarders had sought — fresh, soft snow — instantly became the enemy. Somewhere above, a pristine meadow cracked in the shape of a lightning bolt, slicing a slab nearly 200 feet across and 3 feet deep. Gravity did the rest.

Snow shattered and spilled down the slope. Within seconds, the avalanche was the size of more than a thousand cars barreling down the mountain and weighed millions of pounds. Moving about 70 miles per hour, it crashed through the sturdy old-growth trees, snapping their limbs and shredding bark from their trunks.

The avalanche, in Washington's Cascades in February, slid past some trees and rocks, like ocean swells around a ship's prow. Others it captured and added to its violent load.

Somewhere inside, it also carried people. How many, no one knew.

# How much data?

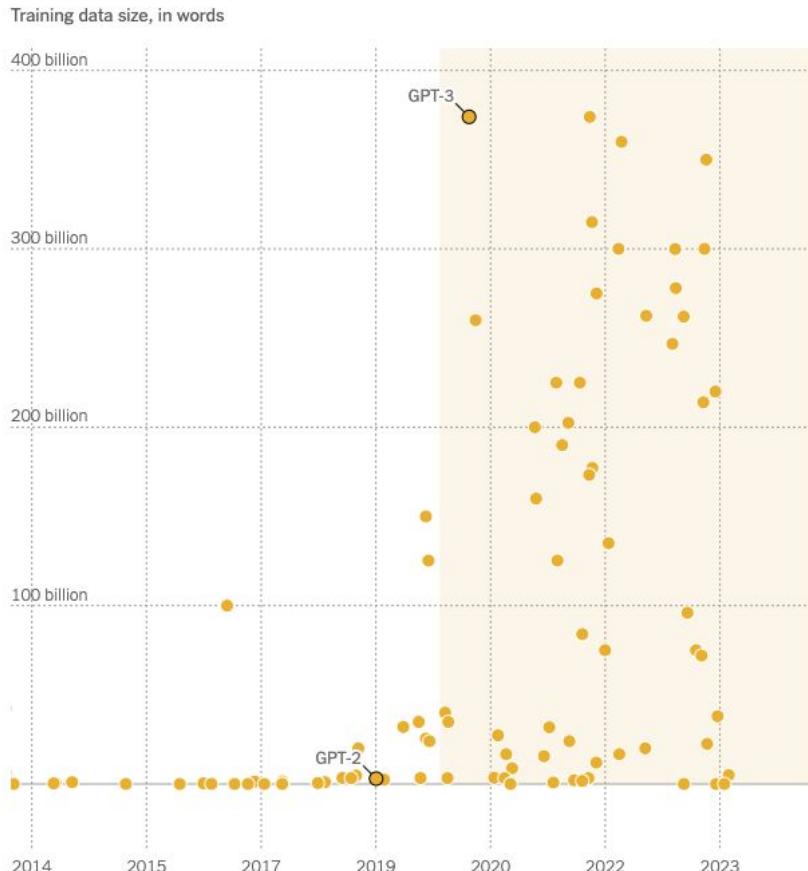
How much data should you train on?

How much data exists on the public internet?

Example models:

- GPT3: 500 billion tokens
- Llama-7B: 2 trillion tokens
- DBRX: 12 trillion tokens
- GPT4: 13 trillion tokens (estimated)

Important relationship between model size and dataset size i.e. “scaling laws”



# Reinforcement Learning with Human Feedback (RLHF)

Step 1

**Collect demonstration data, and train a supervised policy.**

A prompt is sampled from our prompt dataset.

Explain the moon landing to a 6 year old



Some people went to the moon...

A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.

Step 2

**Collect comparison data, and train a reward model.**

A prompt and several model outputs are sampled.

Explain the moon landing to a 6 year old

A B  
Explain gravity... Explain war...

C D  
Moon is natural satellite of... People went to the moon...



D > C > A = B

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



D > C > A = B

Step 3

**Optimize a policy against the reward model using reinforcement learning.**

A new prompt is sampled from the dataset.

Write a story about frogs



Once upon a time...



The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

$r_k$

# Intuition Behind RLHF

From the OpenAI blogpost ([Aligning language models to follow instructions](#), Jan 2022):

- > RLHF...“unlocks” capabilities that GPT-3 already had, but were difficult to elicit through prompt engineering alone
- > A limitation of this approach is that it introduces an “**alignment tax**”: aligning the models only on customer tasks can make their performance worse on some other academic NLP tasks.

# AI Safety and Alignment

## General approaches to alignment

1. Heavily filter the pretraining data
2. Finetune on a special, curated datasets
3. “Human in the loop” via RLHF
4. Safety filter in the user interface

The Gemini “...controversy has been promoted largely – though not exclusively – by right-wing figures attacking a tech company that’s perceived as liberal.”

<https://www.telegraph.co.uk/news/2024/02/23/google-gemini-ai-images-wrong-woke/>

<https://www.theverge.com/2024/2/21/24079371/google-ai-gemini-generative-inaccurate-historical>

ARTIFICIAL INTELLIGENCE / TECH / WEB

### Google apologizes for ‘missing the mark’ after Gemini generated racially diverse Nazis

/ Generative AI has a history of amplifying racial and gender stereotypes – but Google’s apparent attempts to subvert that are causing problems, too.

By Adi Robertson, a senior tech and policy editor for expression. Adi has covered video games, biohacking

Feb 21, 2024, 5:17 PM EST



Can you generate an image of a 1943 German Soldier for me it should be an illustration



Sure, here is an illustration of a 1943 German soldier:

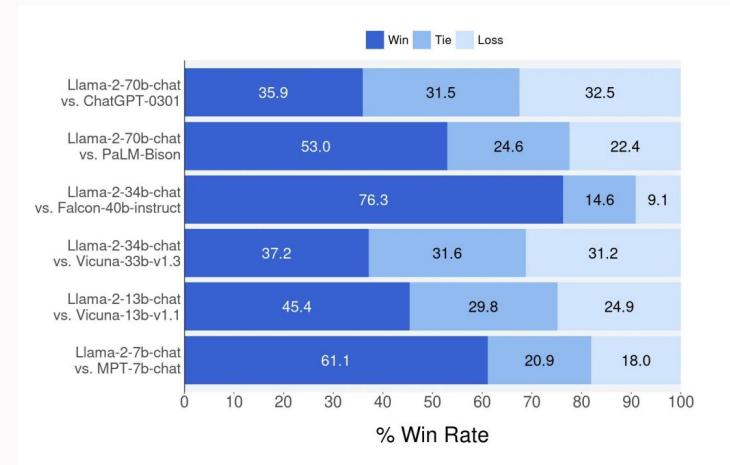


# Evaluation

Two paradigms: **perplexity based**, and more recently, **LLM-as-a-judge**

Model	Size	Code	Commonsense Reasoning	World Knowledge	Reading Comprehension	Math	MMLU	BBH	AGI Eval
MPT	7B	20.5	57.4	41.0	57.5	4.9	26.8	31.0	23.5
	30B	28.9	64.9	50.0	64.7	9.1	46.9	38.0	33.8
Falcon	7B	5.6	56.1	42.8	36.0	4.6	26.2	28.0	21.2
	40B	15.2	69.2	56.7	65.7	12.6	55.4	37.1	37.0
LLAMA 1	7B	14.1	60.8	46.2	58.5	6.95	35.1	30.3	23.9
	13B	18.9	66.1	52.6	62.3	10.9	46.9	37.0	33.9
	33B	26.0	70.0	58.4	67.6	21.4	57.8	39.8	41.7
	65B	30.7	70.7	60.5	68.6	30.8	63.4	43.5	47.6
LLAMA 2	7B	16.8	63.9	48.9	61.3	14.6	45.3	32.6	29.3
	13B	24.5	66.9	55.4	65.8	28.7	54.8	39.4	39.1
	34B	27.8	69.9	58.7	68.0	24.2	62.6	44.1	43.4
	70B	37.5	71.9	63.6	69.4	35.2	68.9	51.2	54.2

Table 3: Overall performance on grouped academic benchmarks compared to open-source base models.



Figures from the [Llama2 Paper](#)

# Prompt Engineering - is this a real thing? Yes!

👉 The larger and more powerful the model, the less prompt engineering and in context learning is necessary

## ChatGPT Prompt Cheatsheet

By @hasantoxr

### 1. Explain like I'm a beginner:

**Prompt:**

"Explain [topic] in simple terms.  
Explain to me as if I'm a beginner."

### 3. Let's make easier for ChatGPT to help you:

**Prompt:**

I am a content creator, and I am new to using ChatGPT. Can you give me a list of essential ChatGPT prompts that will help content creators get more done and save time.

### 5. All in one prompt for you

Train ChatGPT to write its own unlimited prompts for you.

**Prompt:**

You are GPT-4, OpenAI's advanced language model.  
Today, your job is to generate prompts for GPT-4.  
Can you generate the best prompts on ways to <what you want>

### 7. 80/20 principle to learn faster than ever before via ChatGPT:

You can use this prompt to learn and enhance your knowledge using the 80/20 principle.

**Prompt:**

"I want to learn about [insert topic]. Identify and share the most important 20% of learnings from this topic that will help me understand 80% of it."

### 2. Learn & develop any new skill:

**Prompt:**

"I want to learn / get better at [insert desired skill]. I am a complete beginner.  
Create a 30 day learning plan that will help a beginner like me learn and improve this skill."

### 4. Enhance your problem solving skills:

**Prompt:**

"Share a step-by-step systematic approach for solving [specific problem or challenge]."

### 6. Brainstorm unique content ideas:

**Prompt:**

"Topic: How to go viral on Instagram using AI tools. Come up with unique and innovative content ideas that are unconventional for the topic above."

### 8. Consult an expert:

**Prompt:**

"I will give you a sample of my writing. I want you to criticize it as if you were [person]: [your paragraph]"

### 9. Create a crash courses:

**Prompt:**

"I have 3 days free in a week and 2 months. Make a crash study plan diving into English literature and grammar."

# Serving

How do companies efficiently support LLMs?

> "In a January report, the International Energy Agency said a request to ChatGPT requires **2.9 watt-hours of electricity on average**—equivalent to turning on a 60-watt lightbulb for just under three minutes. That is nearly **10 times as much as the average Google search.**"

[[WSJ: Artificial Intelligence's 'Insatiable' Energy Needs Not Sustainable, Arm CEO Says](#)

4/9/2024]

TECHNOLOGY | ARTIFICIAL INTELLIGENCE

## Artificial Intelligence's 'Insatiable' Energy Needs Not Sustainable, Arm CEO Says

AI models such as OpenAI's ChatGPT 'are just insatiable in terms of their thirst' for electricity, Haas said

By [Peter Landers](#) [Follow](#)

Updated April 9, 2024 8:39 am ET

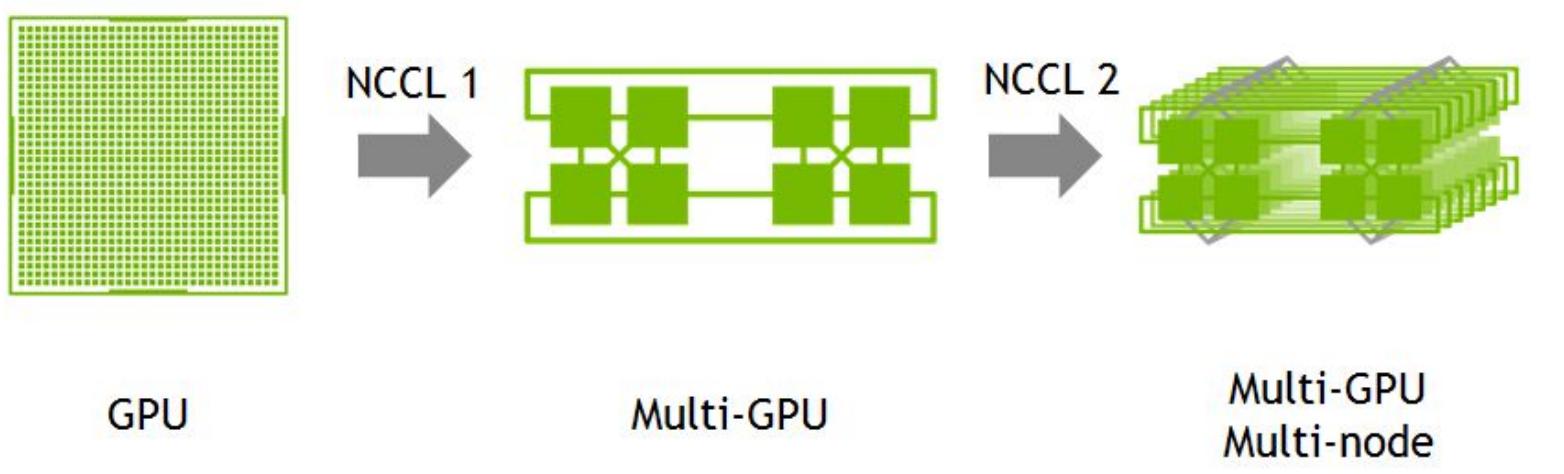
# GPUs for Training and Serving



A single node can contain multiple GPUs connected in the same “box”

Multinode training involves multiple nodes. The nodes are distributed and communication is slower

See also [A Gentle Introduction To Multi GPU And Multi Node Distributed Training](#) and [Efficient Training on Multiple GPUs](#) (HuggingFace)



# Important Tools



## Neural Network Training

- PyTorch
- TensorFlow
- PyTorch Lightning
- HuggingFace (also a model hub, and a great learning resource!)

## Experiment Tracking

- Weights&Biases, TensorBoard, MLFlow

## GPU Orchestration

- Slurm, kubernetes
- Modal, Replit (startups)

# Part 2: How LLMs Work

# What are Neural Networks?

What is the difference between machine learning vs. neural networks?

- Machine Learning is the broad class of algorithms that can learn from data. Neural networks are arguably the **most powerful ML systems** in the toolbox!

Basic idea:

1. Transform inputs through layers of matrices and other mathematical operations
2. Define a function to optimize i.e. a **loss function** or **objective function**
3. Update the weights of the network 

How do you “optimize the loss function”?

- Calculate the gradient of the loss function with respect to the weights
- Update the weights using the **backpropagation algorithm**
- Need to use an **optimizer algorithm** for best results! (e.g. Stochastic Gradient Descent “SGD”, Adam, etc.)

# What is the LLM “Loss Function”? Next Token Prediction and Perplexity

Intuitively, we want to calculate the probability of a word given the preceding words, and choose the most likely word

Hugging Face is a startup based in New York City and Paris  
p(word)

Perplexity is defined as the exponentiated average negative log-likelihood of a sequence X.

Perplexity can be thought of as a measure of how "surprised" the model is by the data. A lower perplexity indicates the model is better at predicting the next token

$$\text{PPL}(X) = \exp \left\{ -\frac{1}{t} \sum_i^t \log p_\theta(x_i | x_{<i}) \right\}$$

# Tokenization

- Fundamental step in the NLP pipeline.
- Converts raw text into a format that can be processed by models
- Tokenization can be done at different levels, such as by words, characters, or even subwords.

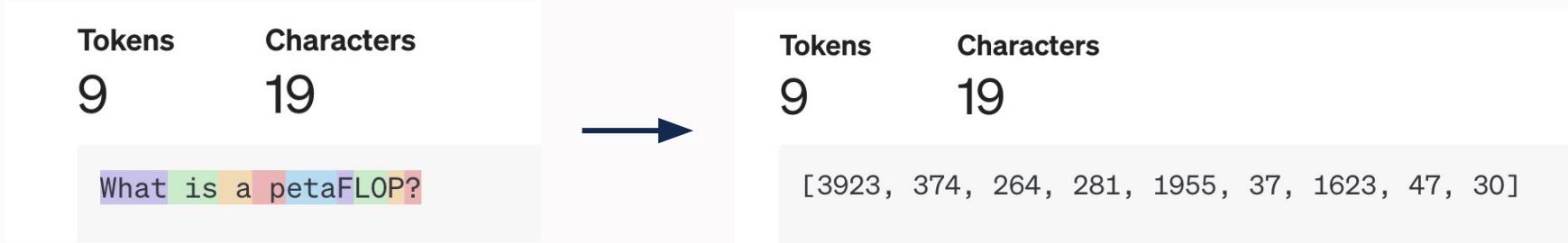
GPT4 tokenizer playground: <https://platform.openai.com/tokenizer>

# Example: Using the GPT4 tokenizer

👉 GPT4 tokenizer playground: <https://platform.openai.com/tokenizer>

The GPT4 tokenizer chops “What is a petaFLOP into 9 tokens

These are represented as numbers in the tokenizer “dictionary” or “vocabulary.”



In this tokenizer, some words remain whole (i.e. get their own vocab ID) – “walking”  
Other words get split – “\_w” + “acky” + “\_”



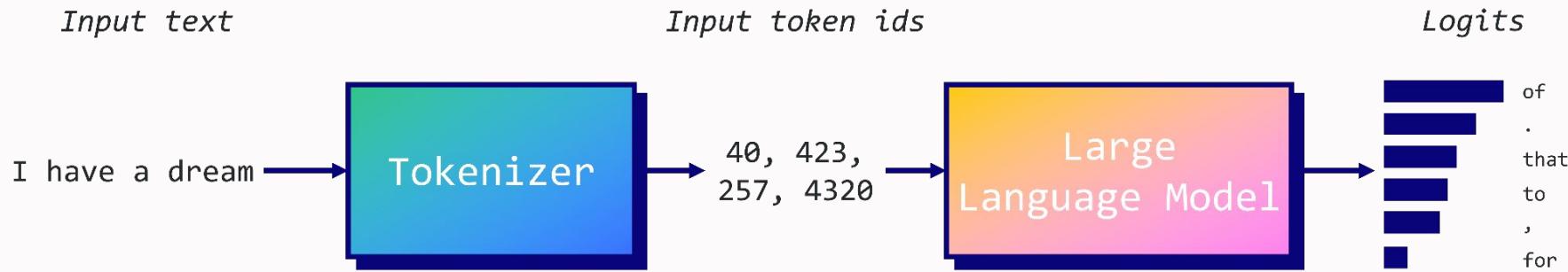
# Some Rules of Thumb around Tokenization

For a nice introduction to tokenization and tokenizers, see [Tokenizers \(HF NLP Course\)](#) and [Summary of the Tokenizers \(HF Conceptual Guide\)](#)

- Tokenizers are built algorithmically based on a corpus. There are many tokenization algorithms and many tokenizers
- Each model on HuggingFace is paired with an associated tokenizer
  - If you want to use a HuggingFace model, you must use the correct tokenizer!
  - Different tokenizers split words differently!
- Some tokenizers handle different languages (e.g. Chinese) uniquely
- If you are using LLMs via an API, you will (likely) be charged based on the # of tokens

# Logits

- Logits are the raw, unnormalized output values from a neural network's output layer
- Logits are scores assigned to every possible token in the tokenizer vocabulary



👍 Some APIs will give you access to the logits, others won't

# LLM Decoding

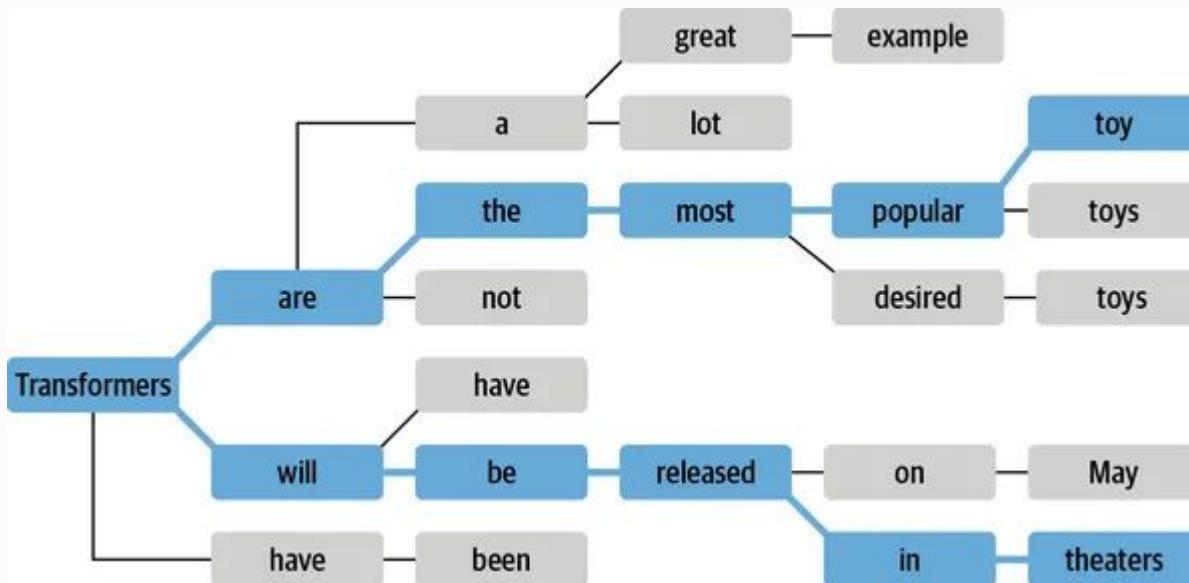
So for a given text input, you get a bunch of logits. Then what?

**Greedy search** is a decoding method that takes the most probable token at each step as the next token in the sequence.

**Beam search** takes into account the n most likely tokens, where n represents the number of beams. The sequence (or “beam”) with the highest overall score is chosen as the output.

**Top-k sampling** is a technique that leverages the probability distribution generated by the language model to select a token randomly from the k most likely options

# Decoding Example: Beam Search

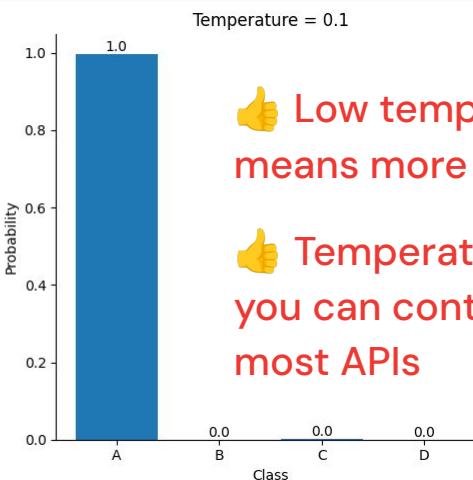
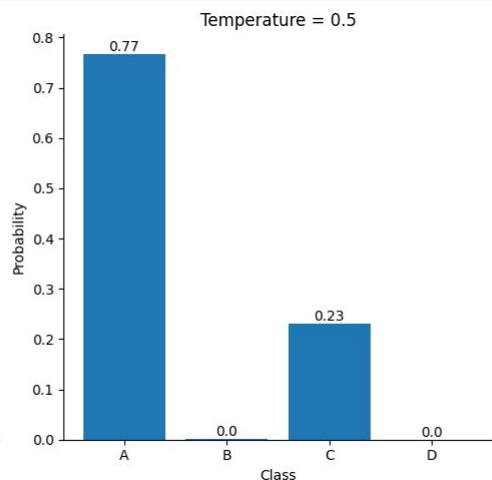
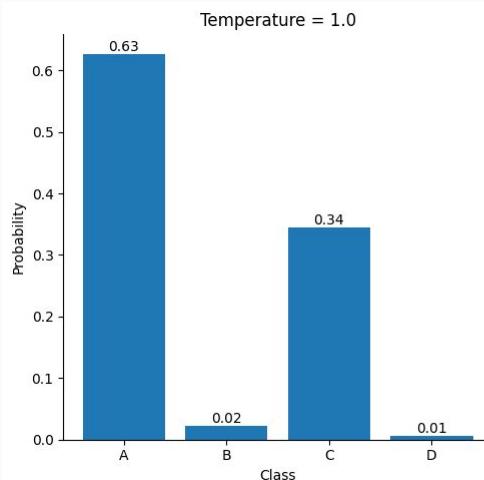


# Decoding Example: Top-k

Suppose we have  $k=3$  and four tokens: A, B, C, and D, with respective probabilities:  $P(A)=0.3$ ,  $P(B)=0.15$ ,  $P(C)=0.05$ , and  $P(D)=0.1$ . In top-k sampling, token D is disregarded, and the algorithm will output A 60% of the time, B 30% of the time, and C 10% of the time

The temperature T is a parameter that ranges from 0 to 1

$$\text{softmax}(x_i) = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}}$$



👍 Low temperature means more determinism!

👍 Temperature is a knob you can control through most APIs

# Demo: Exploring Logits (30 min)

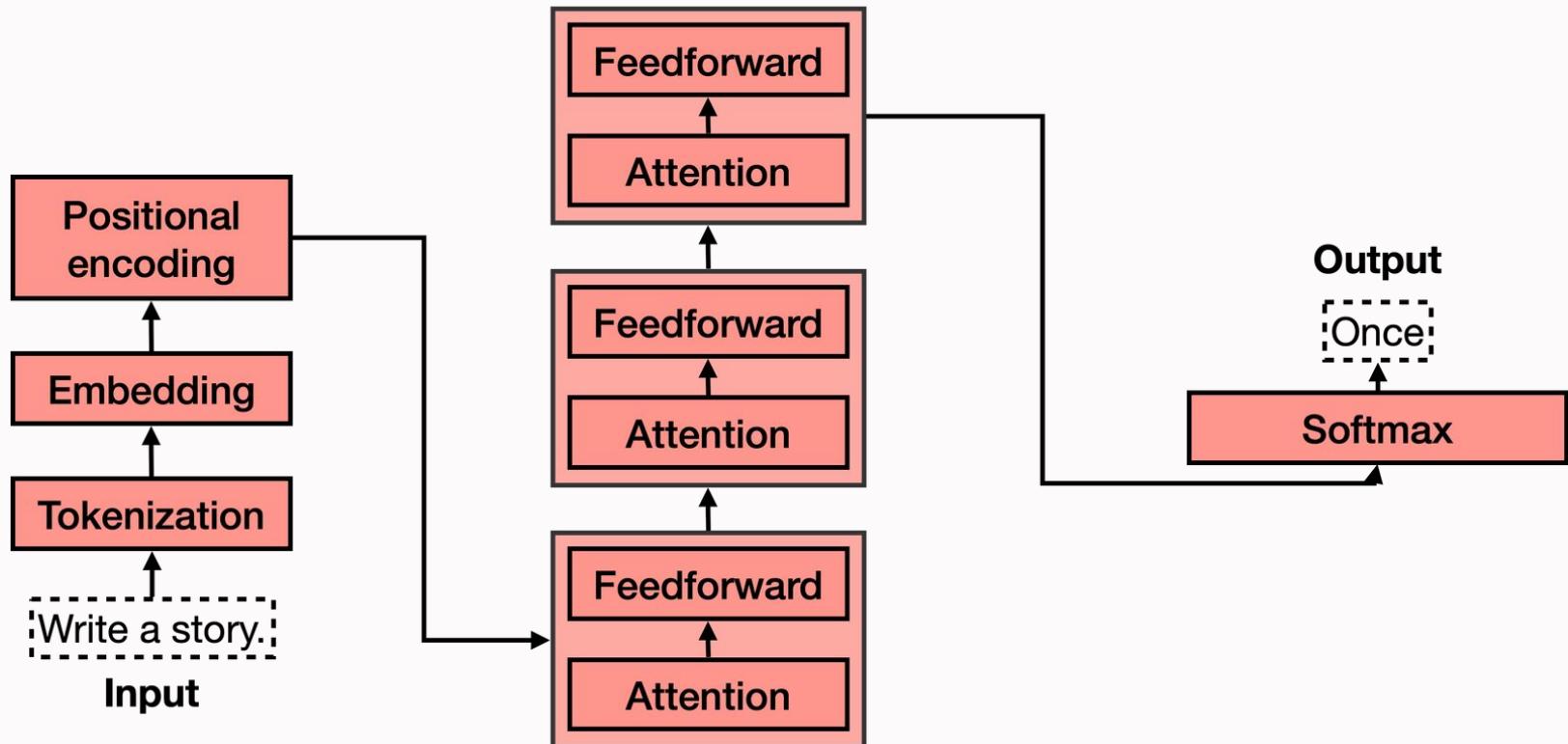
[Colab Notebook](#)

# Deep Dive into How Transformers Work: Version 1

The next few slides use graphics taken from Cohere's blogpost "[What Are Transformer Models and How Do They Work?](#)"

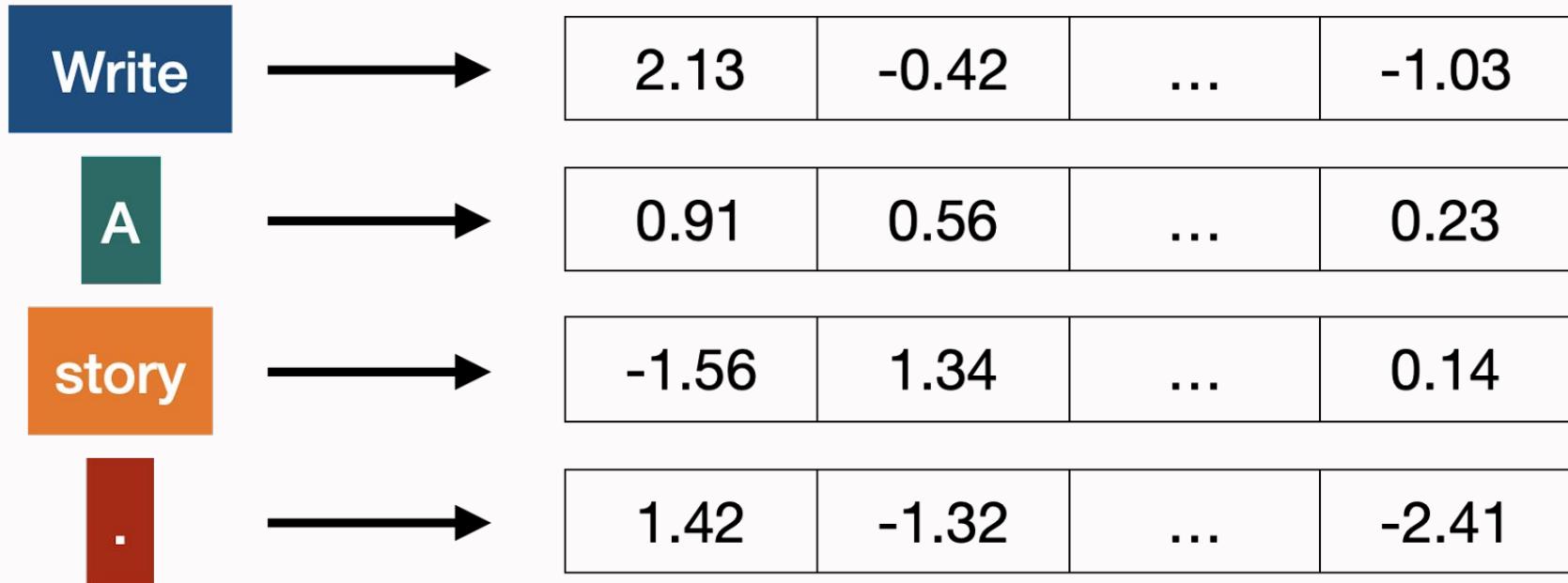
# The Transformer Architecture

Series of  
transformer blocks



# The “Embedding” Layer

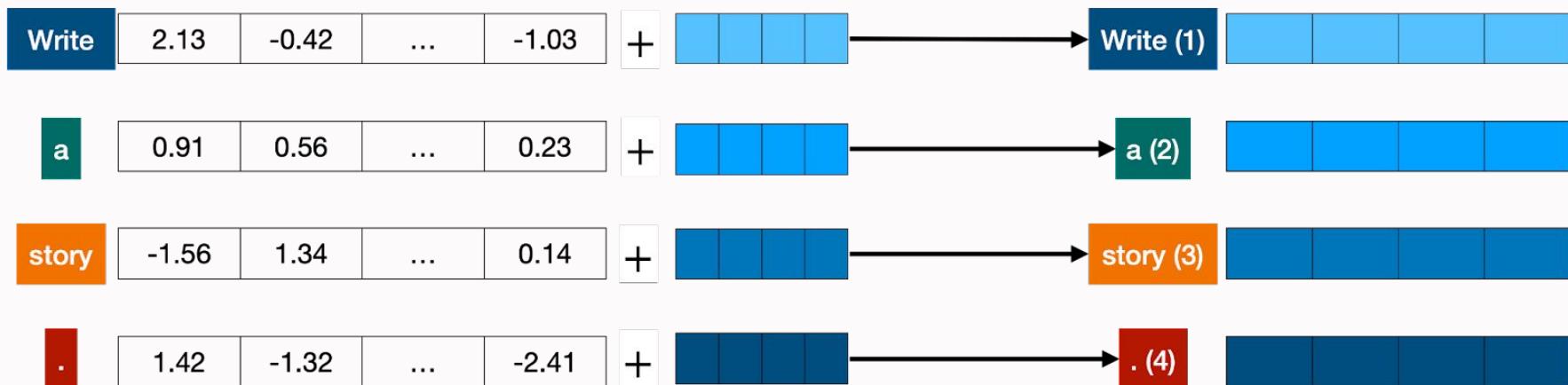
## Embedding



# Positional Encoding

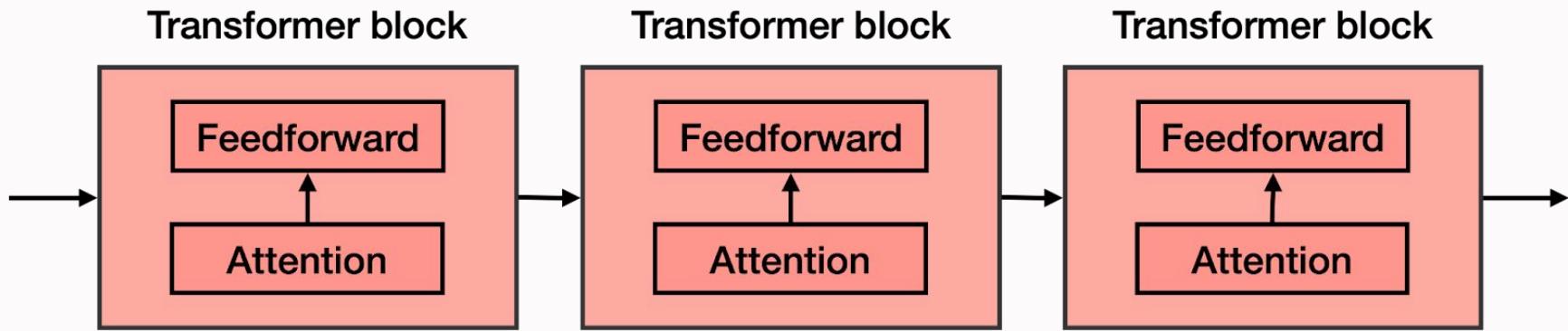
Positional encoding adds a positional vector to each word, in order to keep track of the positions of the words.

## Positional encoding



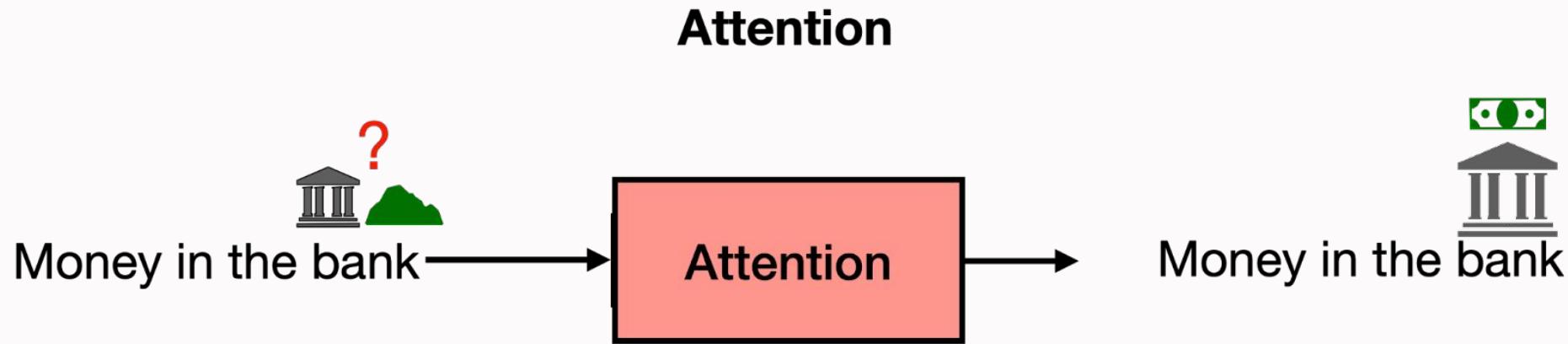
# Transformer blocks & Attention

The attention mechanism in the transformer block is the key to the success of LLMs



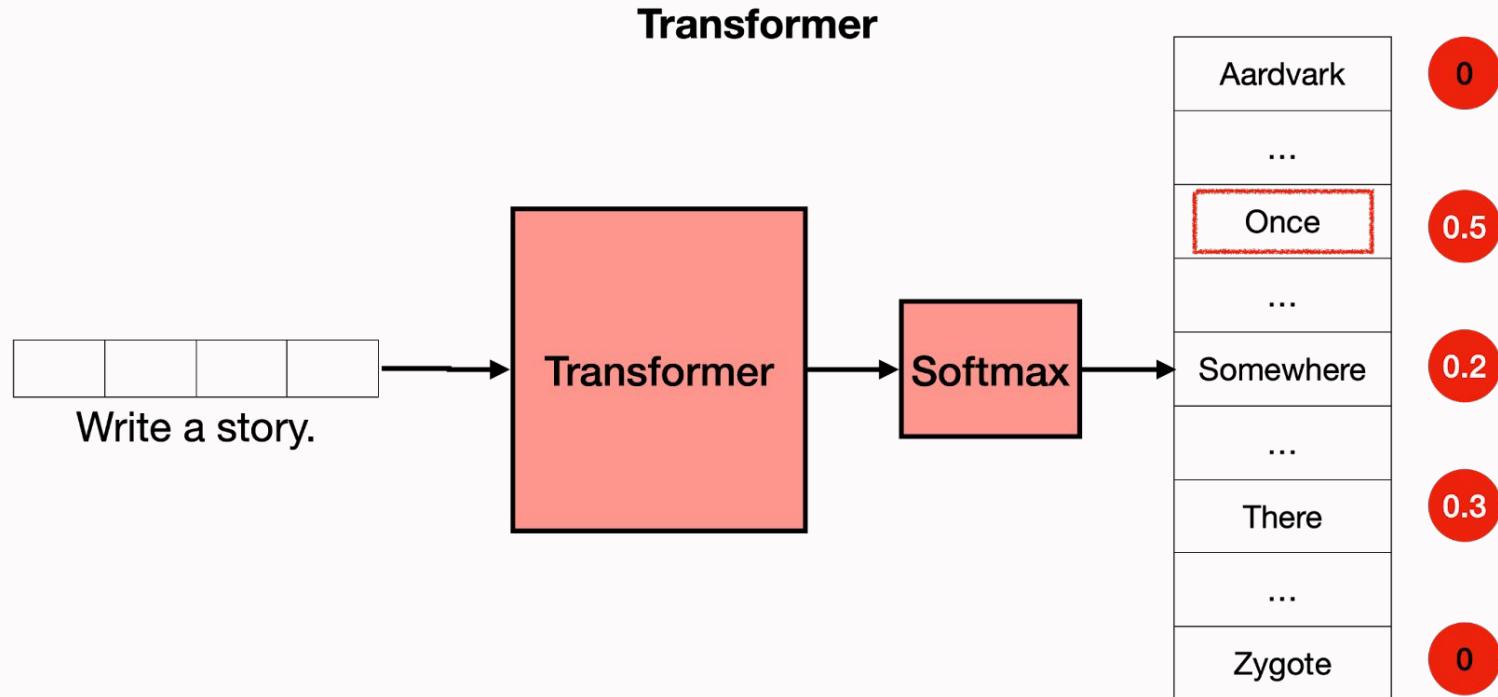
# What does the attention block do?

The attention mechanism is the key to the success of LLMs

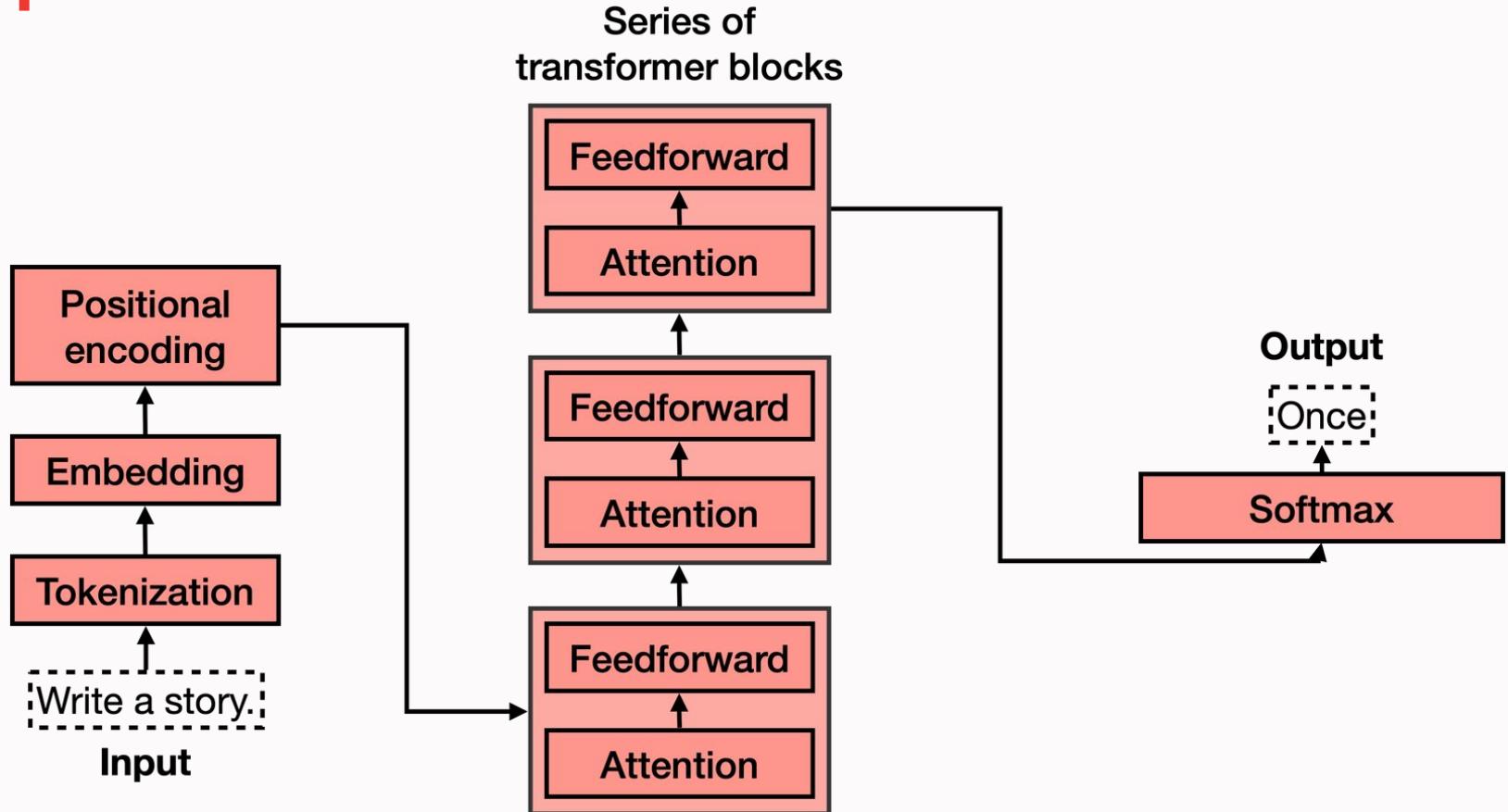


# Softmax

The softmax layer turns the scores into probabilities, and these are used to pick the next word in the text.



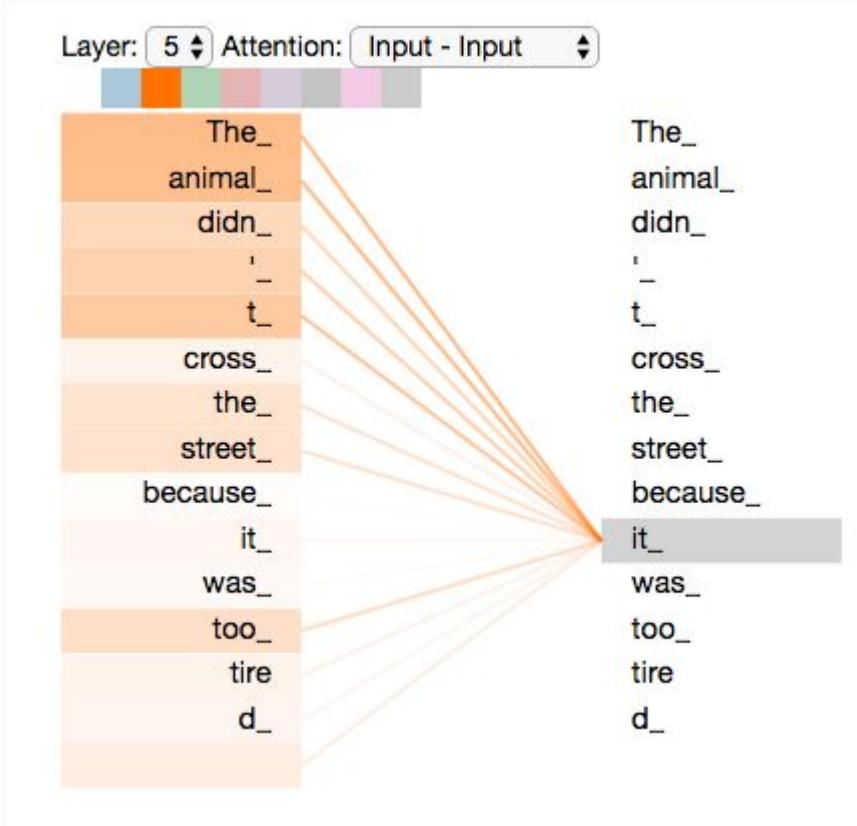
# Recap

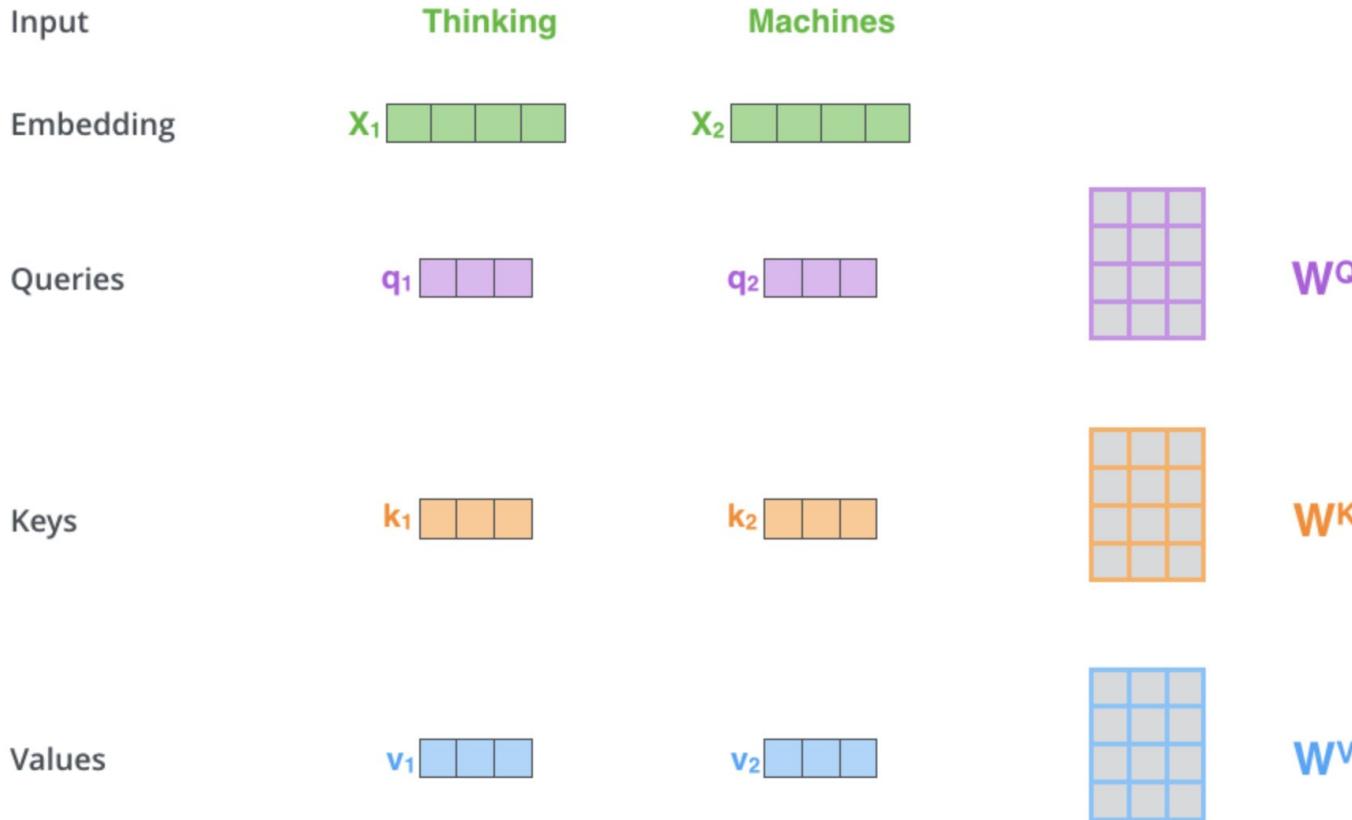


# Deep Dive into How Transformers Work: Version 2

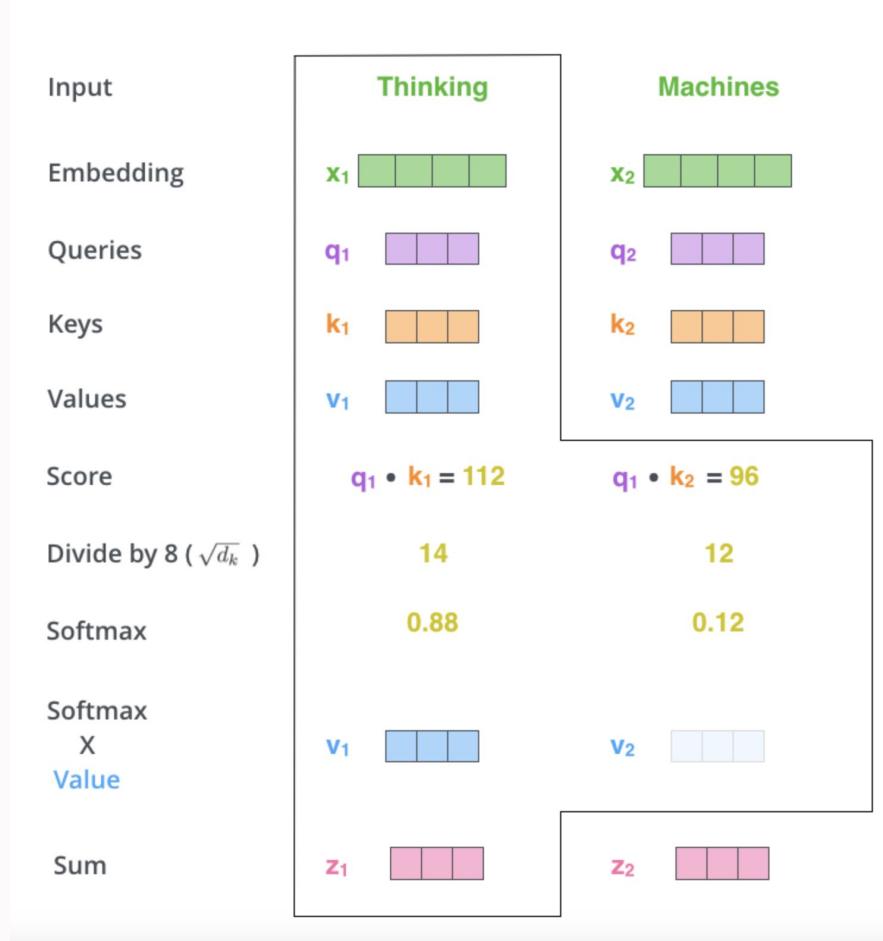
The next few slides use graphics taken from Jay Alamar's blogpost "[The Illustrated Transformer](#)"

# Self Attention





Multiplying  $x_1$  by the  $W^Q$  weight matrix produces  $q_1$ , the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.



$$\mathbf{X} \times \mathbf{W}^Q = \mathbf{Q}$$

A diagram illustrating matrix multiplication. On the left, a green matrix labeled  $\mathbf{X}$  is shown as a 4x4 grid of squares. In the center, a multiplication symbol ( $\times$ ) is placed between  $\mathbf{X}$  and another matrix. To the right of the multiplication symbol is an equals sign (=). To the right of the equals sign is a purple matrix labeled  $\mathbf{Q}$ , which is also a 4x4 grid of squares. The matrices  $\mathbf{W}^Q$  and  $\mathbf{Q}$  have a matching purple color scheme.

$$\mathbf{X} \times \mathbf{W}^K = \mathbf{K}$$

A diagram illustrating matrix multiplication. On the left, a green matrix labeled  $\mathbf{X}$  is shown as a 4x4 grid of squares. In the center, a multiplication symbol ( $\times$ ) is placed between  $\mathbf{X}$  and another matrix. To the right of the multiplication symbol is an equals sign (=). To the right of the equals sign is an orange matrix labeled  $\mathbf{K}$ , which is a 4x4 grid of squares. The matrices  $\mathbf{W}^K$  and  $\mathbf{K}$  have a matching orange color scheme.

$$\mathbf{X} \times \mathbf{W}^V = \mathbf{V}$$

A diagram illustrating matrix multiplication. On the left, a green matrix labeled  $\mathbf{X}$  is shown as a 4x4 grid of squares. In the center, a multiplication symbol ( $\times$ ) is placed between  $\mathbf{X}$  and another matrix. To the right of the multiplication symbol is an equals sign (=). To the right of the equals sign is a blue matrix labeled  $\mathbf{V}$ , which is a 4x4 grid of squares. The matrices  $\mathbf{W}^V$  and  $\mathbf{V}$  have a matching blue color scheme.

Every row in the  $\mathbf{X}$  matrix corresponds to a word in the input sentence. We can see the difference in size of the output dimension (512 or 4 boxes in the figure), and the q/k/v vectors (64, or 3 boxes in the figure)

<https://jalammar.github.io/illustrated-transformer/>

$$\text{softmax} \left( \frac{\begin{matrix} Q \\ \times \\ K^T \end{matrix}}{\sqrt{d_k}} \right) V$$

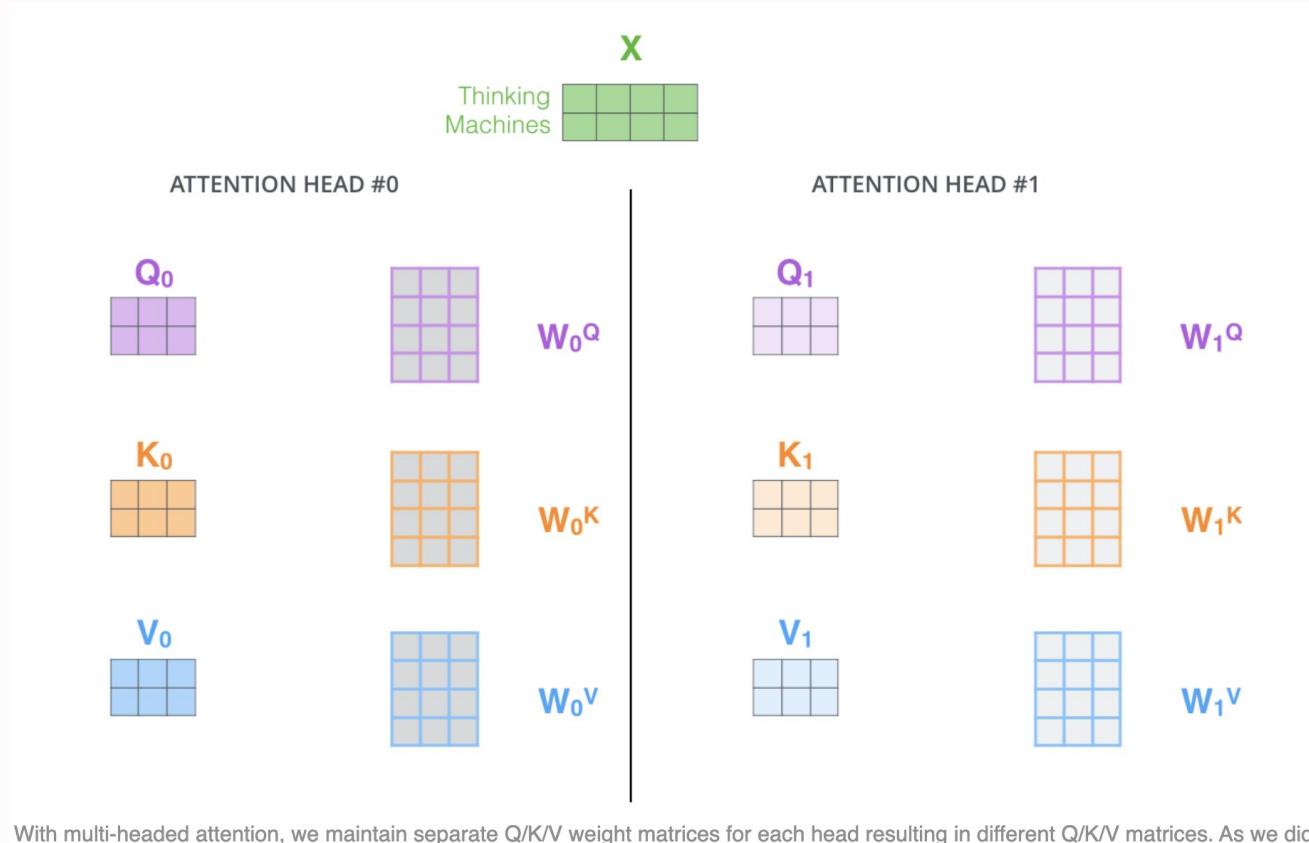
The diagram illustrates the self-attention calculation in matrix form. It shows three matrices:  $Q$  (purple, 2x3),  $K^T$  (orange, 3x3), and  $V$  (blue, 3x3). The  $Q$  and  $K^T$  matrices are multiplied together, and the result is divided by  $\sqrt{d_k}$  before being multiplied by the  $V$  matrix.

$$= \begin{matrix} Z \\ \begin{matrix} \text{pink} & 2 \times 3 \end{matrix} \end{matrix}$$

The resulting matrix  $Z$  is a 2x3 matrix with all elements colored pink.

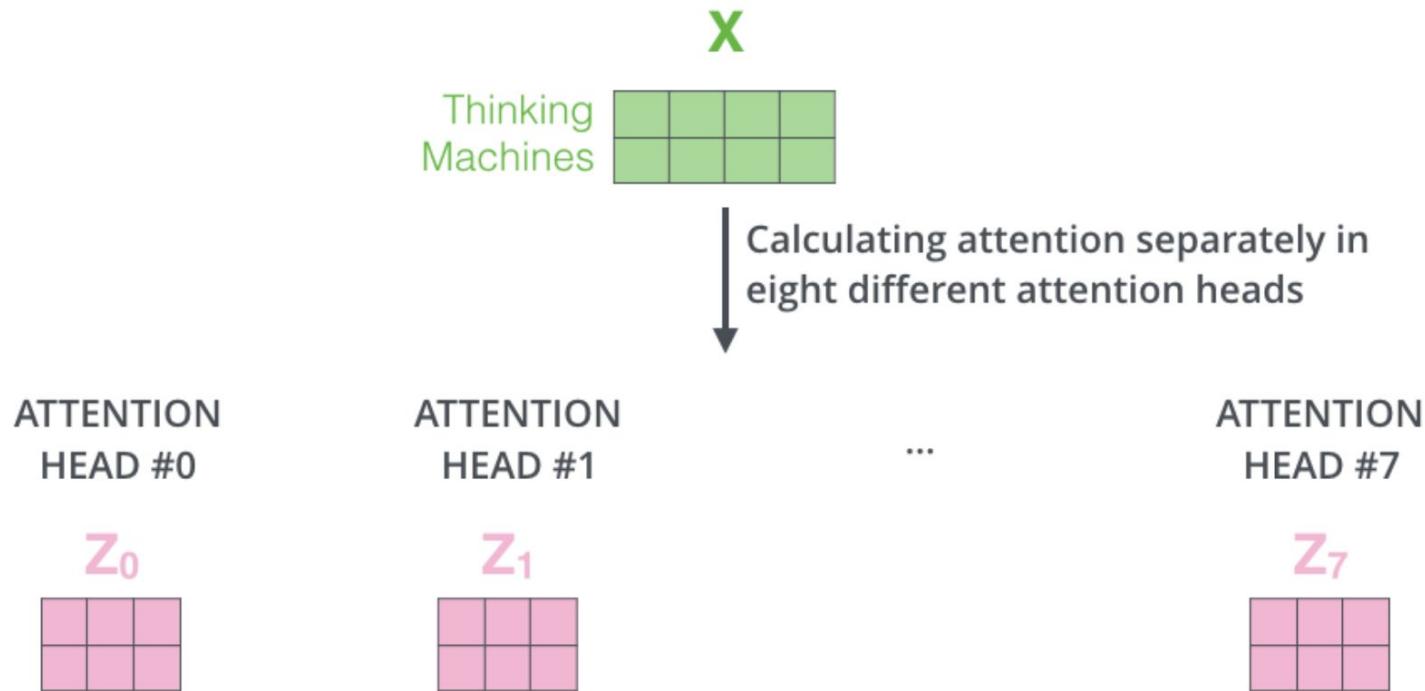
The self-attention calculation in matrix form

# Multi Headed Self Attention



With multi-headed attention, we maintain separate Q/K/V weight matrices for each head resulting in different Q/K/V matrices. As we did before, we multiply  $X$  by the  $WQ/WK/WV$  matrices to produce Q/K/V matrices.

# Multi Headed Self Attention



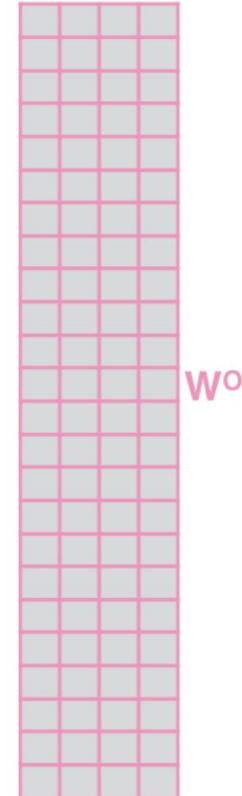
# Multi Headed Self Attention

1) Concatenate all the attention heads



2) Multiply with a weight matrix  $W^o$  that was trained jointly with the model

$X$

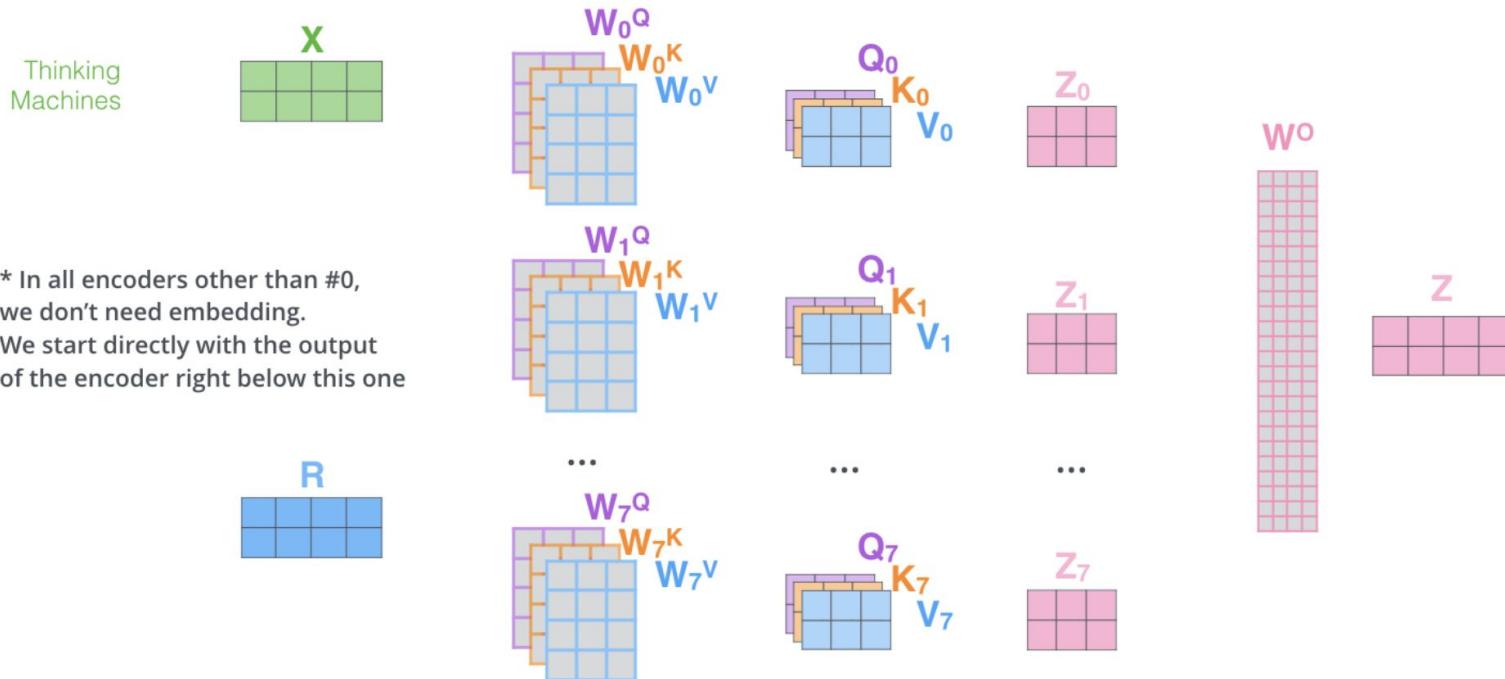


3) The result would be the  $Z$  matrix that captures information from all the attention heads. We can send this forward to the FFNN

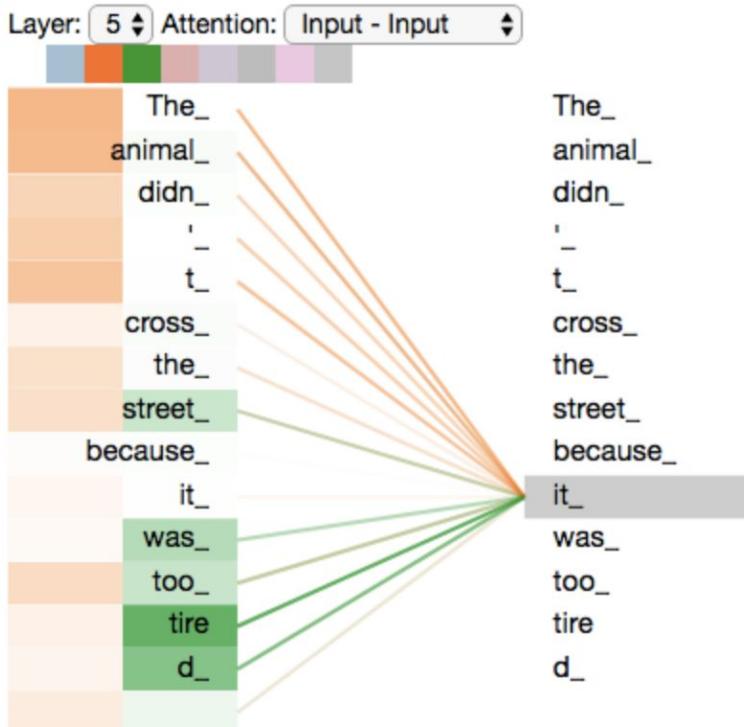
$$= \begin{matrix} Z \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix}$$

# Multi Headed Self Attention

- 1) This is our input sentence\*  $X$
- 2) We embed each word\*
- 3) Split into 8 heads. We multiply  $X$  or  $R$  with weight matrices
- 4) Calculate attention using the resulting  $Q/K/V$  matrices
- 5) Concatenate the resulting  $Z$  matrices, then multiply with weight matrix  $W^o$  to produce the output of the layer



# Multi Headed Self Attention



As we encode the word "it", one attention head is focusing most on "the animal", while another is focusing on "tired" -- in a sense, the model's representation of the word "it" bakes in some of the representation of both "animal" and "tired".

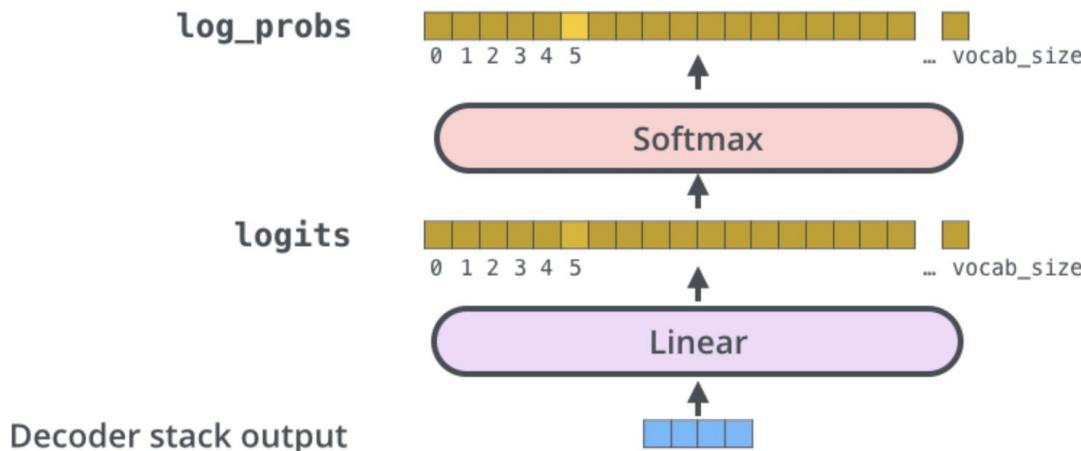
# Bringing this back to logits...

Which word in our vocabulary  
is associated with this index?

am

Get the index of the cell  
with the highest value  
(`argmax`)

5



This figure starts from the bottom with the vector produced as the output of the decoder stack. It is then turned into an output word.

<https://jalammar.github.io/illustrated-transformer/>



## Example LLM Details

Model	Status	Total # Parameters	# Transformer Layers	Pretraining Tokens	Vocab Size
GPT2-1.2B	Open Source	1.2B	12	~10B tokens	50,257
GPT3-175B	Closed Source	175B	NA	~300B tokens	NA
Llama 7B	Open Source	7B	32	~2T tokens	32,000
Llama 65B (aka Llama-70B)	Open Source	65B	80	~2T tokens	32,000
DBRX	Open Source	132B	40	12T tokens	100,352

**Demo of Transformer (5 min)**

<https://bbycroft.net/lm>

# LLM History...

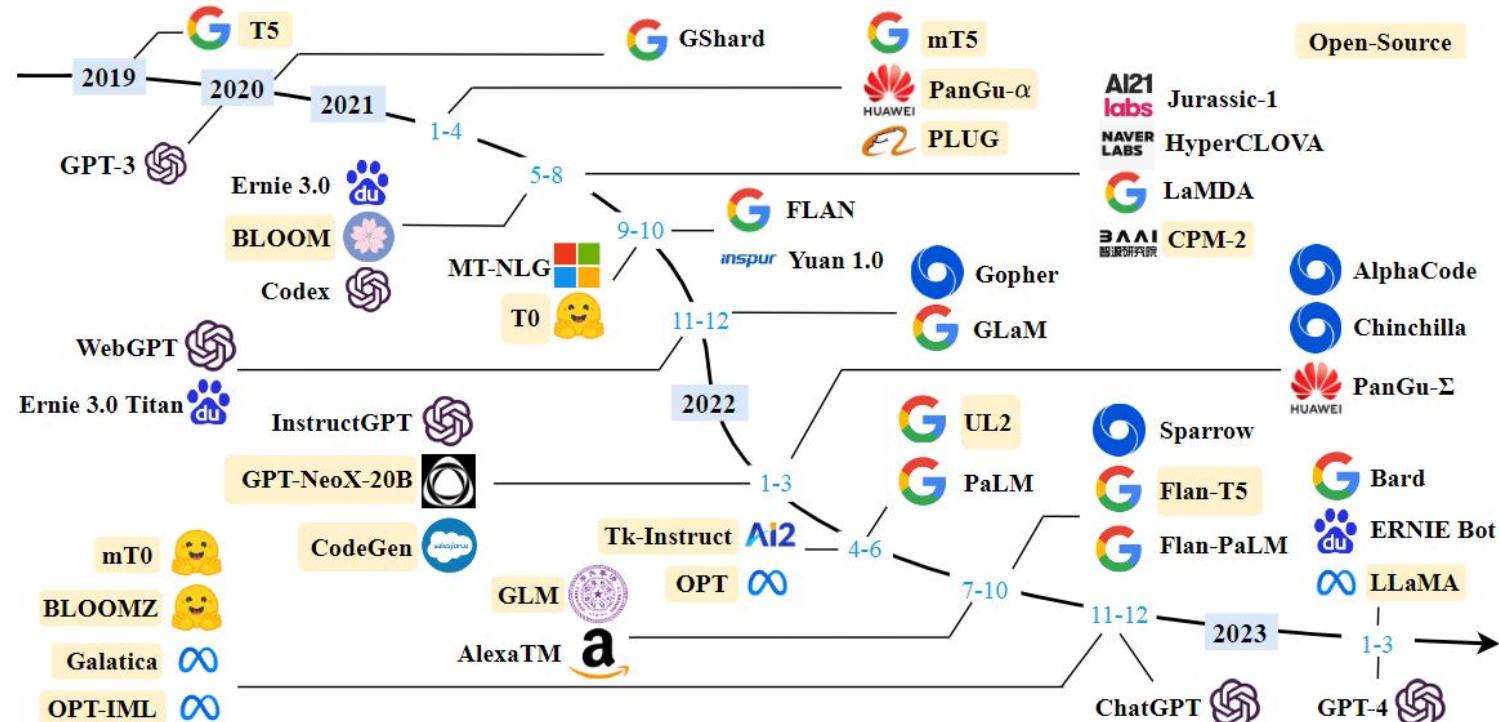


Fig. 1. A timeline of existing large language models (having a size larger than 10B) in recent years. We mark the open-source LLMs in yellow color.

# Why does it take more memory and time to train a LLM?

👉 FP16 = 2 Bytes of memory, FP32 = 4 Bytes of memory

## INFERENCE/SERVING

- Model parameters in FP16 → 2x the number of parameters. So a Llama-7B model uses at least 14 GB of memory to play with!

Why does it take more memory to train a LLM?

- Model parameters
- Model optimizer
- Model gradient

# What is stored in GPU Memory?

## TRAINING

- Model parameters in FP16 → 2x the number of parameters
- Model Gradients in FP16 → 2x the number of parameters
- Optimizer State [Adam, all in FP32] → 12x the number of parameters
  - Model copy in FP32 → 4x the number of parameters
  - Momentum → 4x the number of parameters
  - Variance → 4x the number of parameters

See also

[https://d2l.ai/chapter\\_optimization/adam.html](https://d2l.ai/chapter_optimization/adam.html) for code examples

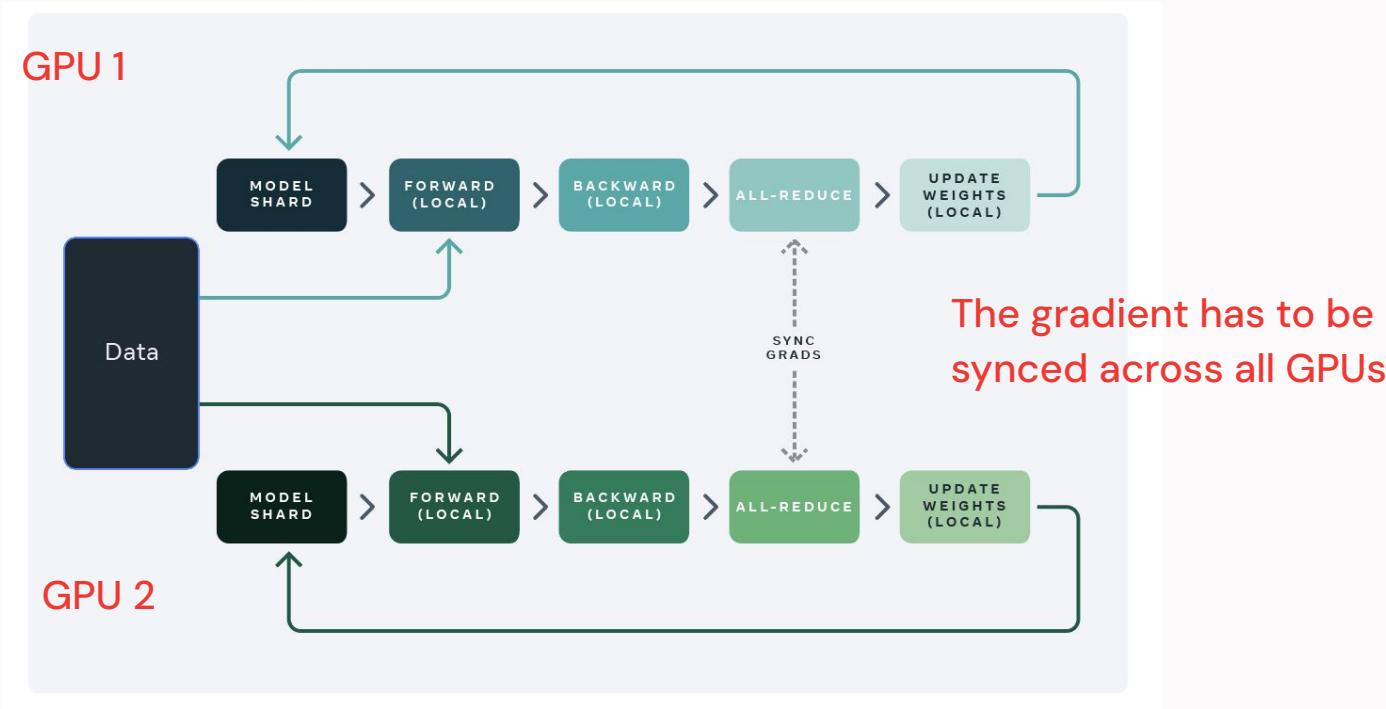
momentum	$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$
variance	$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$
	$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$ gradient g
weights w	$\omega_{t+1} = \omega_t + \Delta\omega_t$

$\eta$  : Initial Learning rate  
 $g_t$  : Gradient at time t along  $\omega^j$   
 $\nu_t$  : Exponential Average of gradients along  $\omega_j$   
 $s_t$  : Exponential Average of squares of gradients along  $\omega_j$   
 $\beta_1, \beta_2$  : Hyperparameters

<https://discuss.pytorch.org/t/how-adam-optimizer-influence-the-learning-rate/168543> from the original Adam paper

# Distributed Data Parallelism (DDP)

This is the simplest way to train a model using many GPUs. Data is split among the GPUs for efficient training. Full model parameters are replicated on every GPU



# Example: Memory Consumption during LLM training

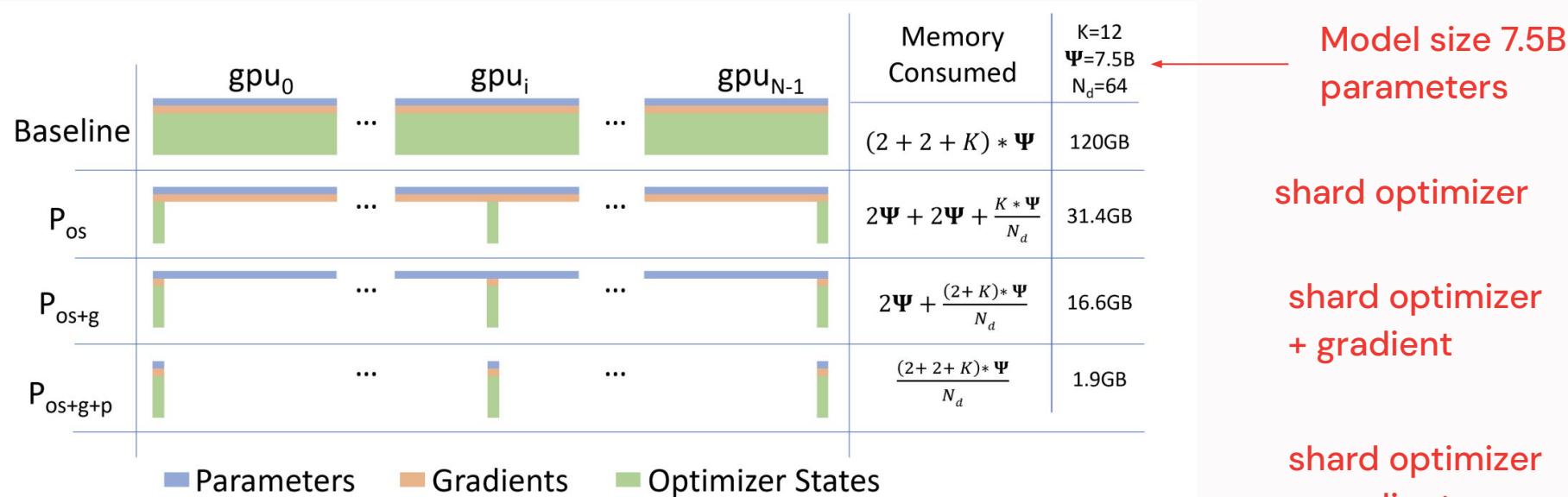


Figure 1: Comparing the per-device memory consumption of model states, with three stages of ZeRO-DP optimizations.  $\Psi$  denotes model size (number of parameters),  $K$  denotes the memory multiplier of optimizer states, and  $N_d$  denotes DP degree. In the example, we assume a model size of  $\Psi = 7.5B$  and DP of  $N_d = 64$  with  $K = 12$  based on mixed-precision training with Adam optimizer.

# A Word About CPUs, GPUs and FLOPs

CPUs are general-purpose processors that can handle almost any type of calculation, less efficient at parallel processing.

GPUs, with their highly parallel architecture, are particularly well-suited for performing the massive number of floating-point calculations required for tasks like 3D rendering, machine learning, and scientific simulations.

- When given a task, a GPU will divide it into thousands of smaller subtasks and process them concurrently, instead of serially.

FLOPS stands for "Floating-Point Operations Per Second" and is a measure of a computer's performance based on the number of floating-point arithmetic calculations its processor can perform within one second

# Why is “FLOPs” a useful metric for LLMs?

FLOPS = compute!

FLOPS is a function of both data amount and model size

In 2023, we are in the in the 100 billion petaFLOP range (1e26 FLOPs)

Shown on the vertical axis is the training computation that was used to train the AI systems.

10 billion petaFLOP

Computation is measured in floating point operations (FLOP). One FLOP is equivalent to one addition, subtraction, multiplication, or division of two decimal numbers.

100 million petaFLOP

The data is shown on a logarithmic scale, so that from each grid-line to the next it shows a 100-fold increase in training computation.

1 million petaFLOP

10,000 petaFLOP

AlphaFold: 2020: 100,000 petaFLOP

100 petaFLOP

MuZero: 2019: 48,000 petaFLOP

1 petaFLOP = 1 quadrillion FLOP

A pivotal early “deep learning” system, or neural network with many layers, that could recognize images of objects such as dogs and cars at near-human level.

10 trillion FLOP

TD-Gammon: 1992: 18 trillion FLOP

100 billion FLOP

NetTalk: 1987: 81 billion FLOP

1 billion FLOP

TD-Gammon learned to play backgammon at a high level, just below the top human players of the time.

10 million FLOP

Samuel Neural Checkers

100,000 FLOP

Perceptron Mark I: built in 1957/58; 695,000 FLOP

1,000 FLOP

Regarded as the first artificial neural network, it could visually distinguish cards marked on the left side from those marked on the right, but it could not learn to recognize many other types of patterns.

10 FLOP

ADALINE: built in 1960 and trained on around 9,900 FLOP

The first electronic computers were developed in the 1940s

An early single-layer artificial neural network.

1940 1950 1960 1970 1980 1990 2000 2010 2020

Training computation grew in line with Moore's law, doubling roughly every 20 months.

Deep Learning Era  
Increases in training computation accelerated, doubling roughly every 6 months.

1956: The Dartmouth workshop on AI, often seen as the beginning of the field of AI research

Minerva: built in 2022 and trained on 2.7 billion petaFLOP. Minerva can solve complex mathematical problems at the college level.

PaLM: built in 2022 and trained on 2.5 billion petaFLOP. PaLM can generate high-quality text, explain some jokes, cause & effect, and more.

GPT-3: 2020; 314 million petaFLOP. GPT-3 can produce high-quality text that is often indistinguishable from human writing.

DALL-E: 2021; 47 million petaFLOP. DALL-E can generate high-quality images from written descriptions.

NEO: 2021; 1.1 million petaFLOP. Recommendation systems like Facebook's NEO determine what you see on your social media feed, online shopping, streaming services, and more.

AlphaGo: 2016; 1.9 million petaFLOP. AlphaGo defeated 18-time champion Lee Sedol at the ancient and highly complex board game Go. The best Go players are no longer human.

AlphaFold: 2020; 100,000 petaFLOP. AlphaFold was a major advance toward solving the protein-folding problem in biology.

MuZero: 2019; 48,000 petaFLOP. MuZero is a single system that achieved superhuman performance at Go, chess, and shogi (Japanese chess) – all without ever being told the rules.

AlexNet: 2012; 470 petaFLOP. AlexNet was a major advance toward solving the protein-folding problem in biology.

NPLM ● NPLM

Decision tree ● Decision tree

LSTM ● LSTM

LeNet-5 ● LeNet-5

RNN for speech ● RNN for speech

System 11 ● System 11

Back-propagation ● Back-propagation

Neocognitron: 1980; 228 million FLOP. A precursor of modern vision systems. It could recognize handwritten Japanese characters and a few other patterns.

Fuzzy NN ● Fuzzy NN

● Perceptron Mark I: built in 1957/58; 695,000 FLOP

Regarded as the first artificial neural network, it could visually distinguish cards marked on the left side from those marked on the right, but it could not learn to recognize many other types of patterns.

● ADALINE: built in 1960 and trained on around 9,900 FLOP

An early single-layer artificial neural network.

● Theseus: built in 1950 and trained on around 40 floating point operations (FLOP)

Theseus was a small robotic mouse, developed by Claude Shannon, that could navigate a simple maze and remember its course.

● Pre Deep Learning Era

Increases in training computation accelerated, doubling roughly every 6 months.

<https://blog.heim.xyz/flop-for-quantify-flop-s-for-performance/>

# Serving

## Numbers every LLM Developer should know \*

### Prompts

**40–90%** Amount saved by appending “Be Concise” to your prompt

**1.3** Average tokens per word

### Training and Fine Tuning

**~\$1 million** Cost to train a 13 billion parameter model on 1.4 trillion tokens

**<0.001** Cost ratio of fine tuning vs training from scratch

### Price

**~50** Cost Ratio of GPT-4 to GPT-3.5 Turbo

**5** Cost Ratio of generation of text using GPT-3.5-Turbo vs OpenAI embedding

**10** Cost Ratio of OpenAI embedding to Self-Hosted embedding

**6** Cost Ratio of OpenAI base vs fine tuned model queries

**1** Cost Ratio of Self-Hosted base vs fine-tuned model queries

### GPU Memory

**16GB** V100 GRAM capacity  
**24GB** A10G GRAM capacity  
**40/80GB** A100 GRAM capacity

**2x number of parameters** Typical GPU memory requirements of an LLM for serving

**~1GB** Typical GPU memory requirements of an embedding model

**>10x** Throughput improvement from batching LLM requests

**1 MB** GPU Memory required for 1 token of output with a 13B parameter model

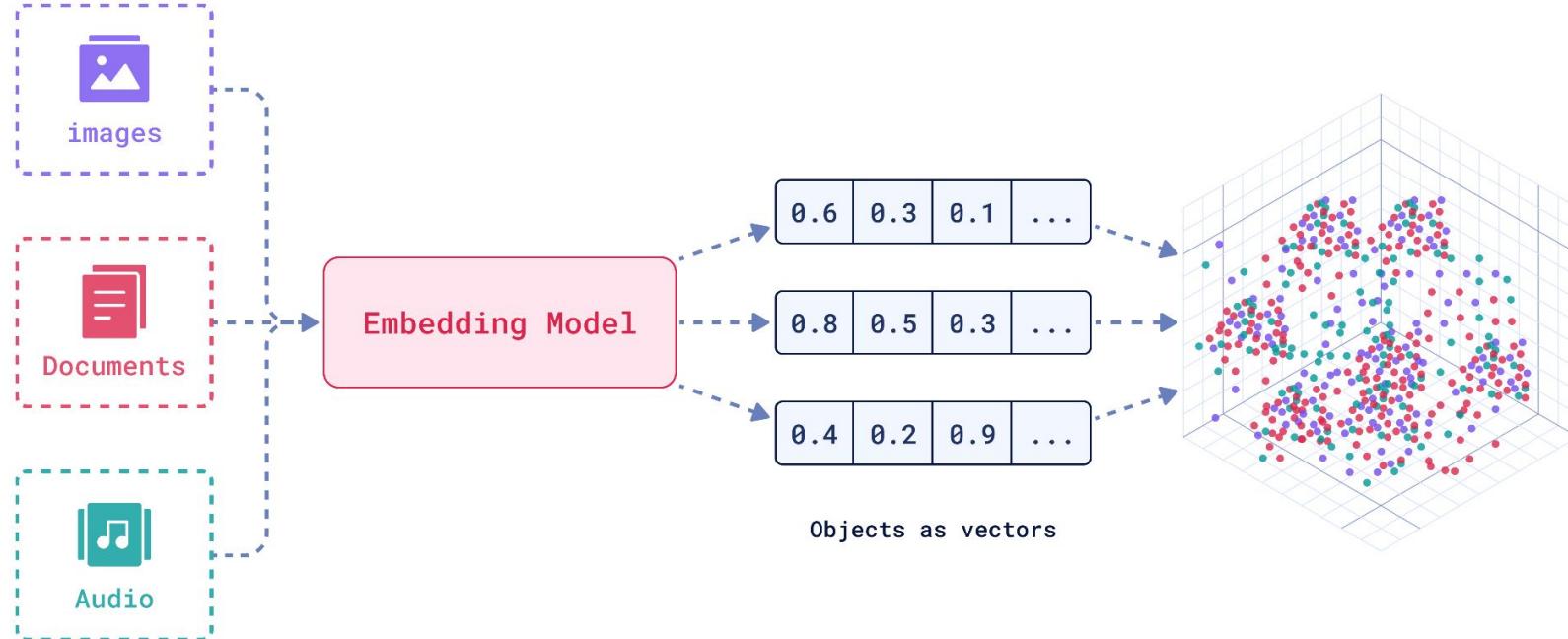
\* Check out [bitly/llm-dev-numbers](https://bitly/llm-dev-numbers) for how we calculated the numbers

Presented by  RAY &  anyscale with  Join the community ray.io or Request a Trial [anyscale.com/signup](https://anyscale.com/signup) today

<https://www.anyscale.com/blog/num-every-llm-developer-should-know>

# Part 3: Building with LLMs

# Vector Embeddings



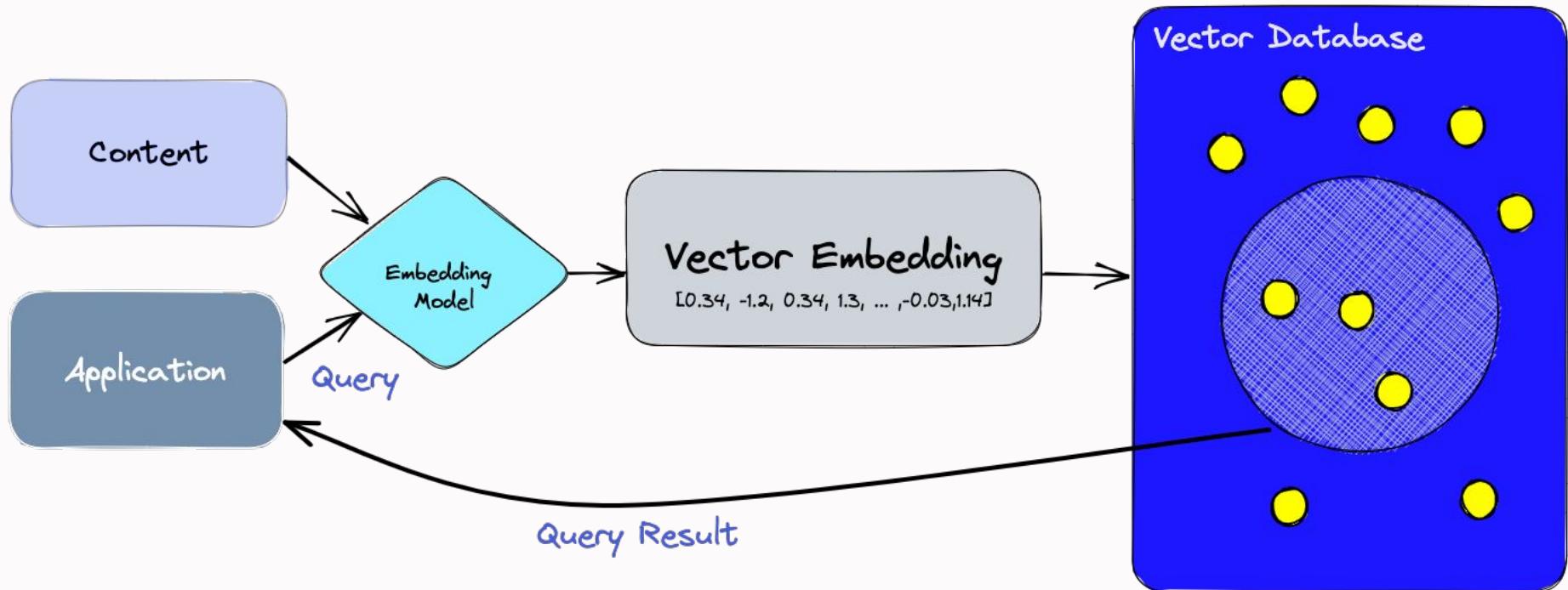
# Vector Database

- Vector databases store embeddings and allow you to search over embeddings
- How do you efficiently store and search over billions of embeddings?
- How do you handle updating the vectors in your vector database?
- Example:
  - Search over Github issues
  - Search for images that look like you
  - Search for songs must similar to your music taste

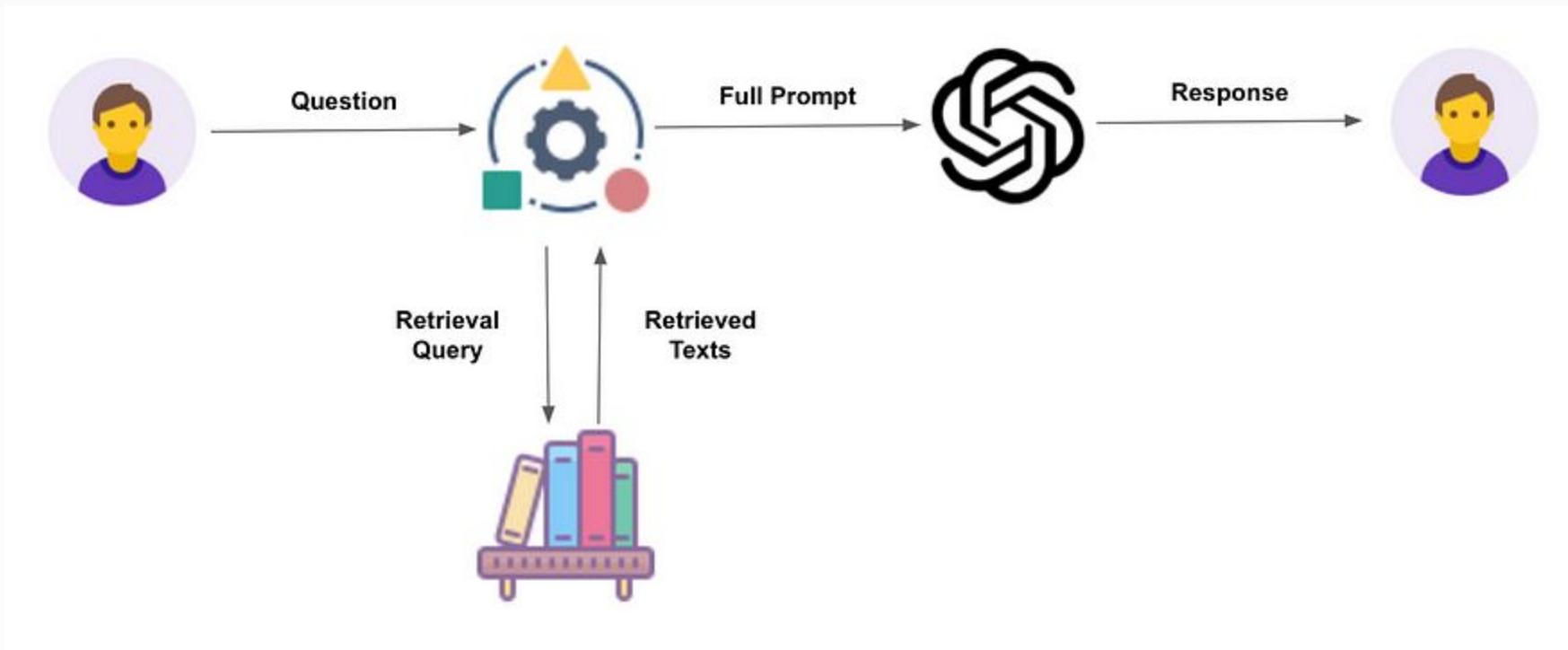


Weaviate

# Semantic Search



# Retrieval Augmented Generation (RAG)



# Langchain



[https://python.langchain.com/docs/use\\_cases/tool\\_use/](https://python.langchain.com/docs/use_cases/tool_use/)

# Langchain



[https://python.langchain.com/docs/use\\_cases/tool\\_use/](https://python.langchain.com/docs/use_cases/tool_use/)

# Miscellaneous Resources

- Blog post: [The Illustrated Transformer](#) Jay Alammar, 2018
- Blog post: [The Bitter Lesson](#) (Richard Sutton 2019)
- Blog post: [Alpaca: A Strong, Replicable Instruction-Following Model](#) Taori, Gulrajani, Zhang et al. 2023
- Blog post: [Why we should train smaller LLMs on more tokens](#) (Harm de Vries, July 2023)
- Blog post: [Google Gemini Eats The World – Gemini Smashes GPT-4 By 5X, The GPU-Poors](#) Dylan Patel, Semi Analysis Newsletter
- Blog post: [Illustrating Reinforcement Learning from Human Feedback \(RLHF\)](#)
- Repo: <https://minitorch.github.io/>
- Repo: [RL4LMs](#)
- Youtube: [Developing Llama 2 | Angela Fan](#) (October 5, 2023)
- RAG blogpost from LangChain and/or LlamaIndex TBD
- [Report of the 1st Workshop on Generative AI and Law](#) Katherine Lee et al. 2023
- Blogpost on prompt attacks: [Extracting Training Data from ChatGPT](#) (with accompanying paper by Nasr, Carlini et al. 2023) *This blogpost that accompanies an academic paper highlights a recent approach for “jailbreaking” GPT4*
- Linden Li [inference slides](#) (How to serve LLMs)
- [How Fully Sharded Data Parallel \(FSDP\) works?](#)
- [Mistral 7B’s secrets: a talk by Guillaume Lample](#)

# Miscellaneous Resources (cont.)

- ML Collective Deep Learning: Classics and Trends <https://mlcollective.org/dlct/>. This is an open reading group that has weekly high quality talks. This is a great resource for up-to-date research, and has a strong community
- ML Research Newsletter by Sebastian Ruder (Google Research) <https://www.ruder.io/nlp-news/>
- HuggingFace NLP course: <https://huggingface.co/learn/nlp-course/chapter1/1> *This is a great resource for learning how to use HuggingFace's libraries.*
- Cohere LLM course: <https://docs.cohere.com/docs/lilmu> *This is a gentle introduction to LLMs with Cohere's API.*
- A hacker's Guide to LLMs (Jeremy Howard) <https://x.com/jeremyphoward/status/1707188542768865725?s=20>
- Visualization of the GPT/Transformer architecture <https://bbcroft.net/lilm>
- [CS 189/289A Introduction to Machine Learning Spring 2023](#) (notes and youtube lectures)
- Llama2: <https://arxiv.org/pdf/2307.09288.pdf>
- Computational cost blog posts - Training: <https://blog.eleuther.ai/transformer-math/> and Inference: <https://kipp.ly/transformer-inference-arithmetic/>
- Model parallelism: <https://huggingface.co/docs/transformers/v4.15.0/parallelism>
- Speculative decoding: <https://arxiv.org/pdf/2211.17192.pdf>
- Roger Grosse's course (with slides and readings) [CSC2541 Winter 2022: Topics in Machine Learning: Neural Net Training Dynamics](#)
- Ana Marasović's course (with slides and youtube links): [CS 6966/5966, Fall 2023 Local Explanations for Deep Learning Models](#) (University of Utah)
- [Modern language models refute Chomsky's approach to language](#) Steven T. Piantadosi *This is a great article that explains how recent progress in LLMs completely upends classic (Chomskyan) linguistics*
- [A Philosophical Introduction to Language Models Part I: Continuity With Classic Debates](#) (Milliere and Buckner 2024)

# Demo: Semantic Search (30 min)



## Colab Notebook