LLM Bootcamp 2023

LLM Foundations

Sergey Karayev

April 21, 2023

Agenda

00

Foundations of ML

Speedrun
key ideas in ML

01

Transformer Architecture

Core ideas and
notable examples

02

Notable LLMs

T5, GPT, Chinchilla, et al

03

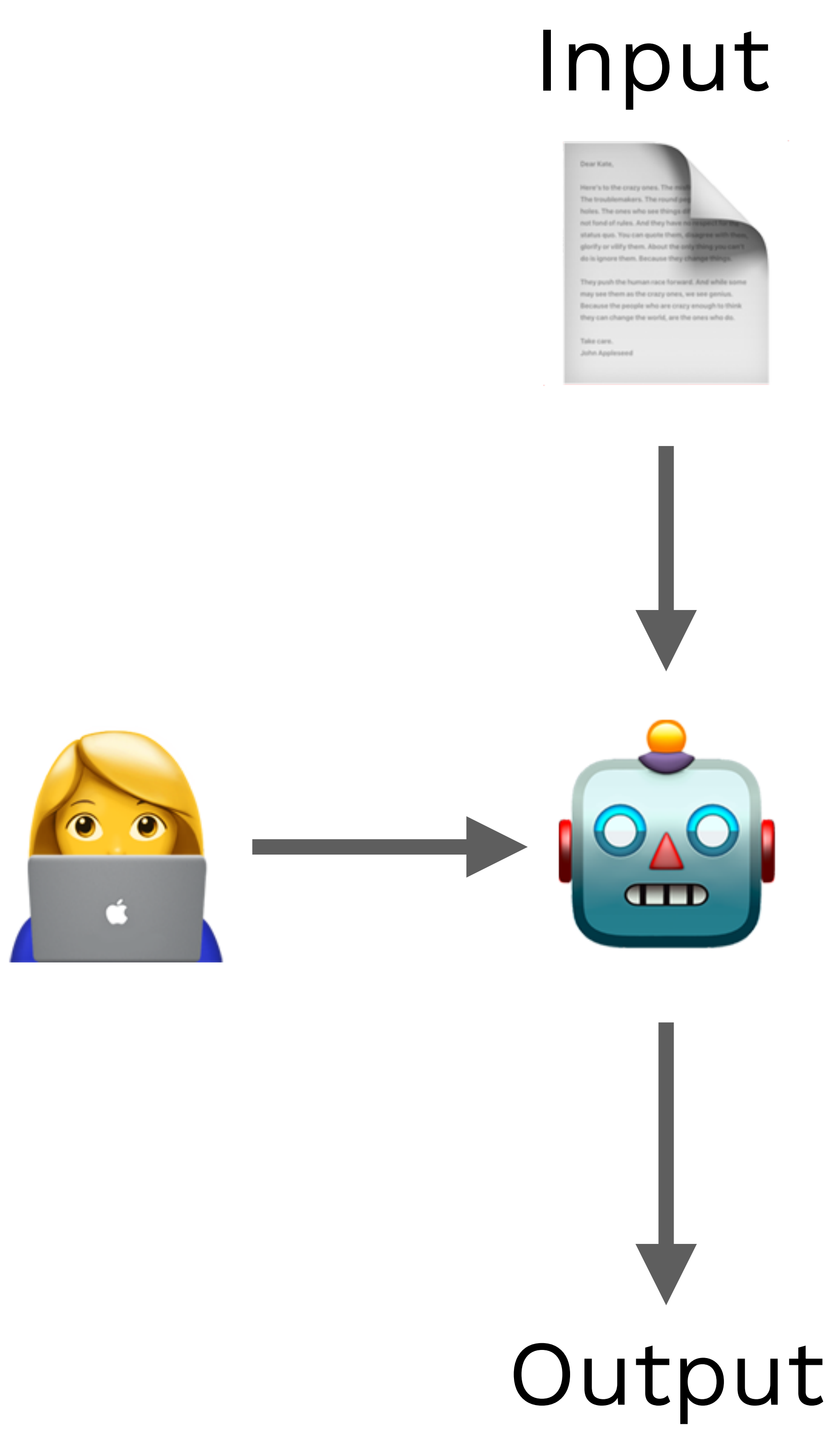
Training & Inference

Running a Transformer

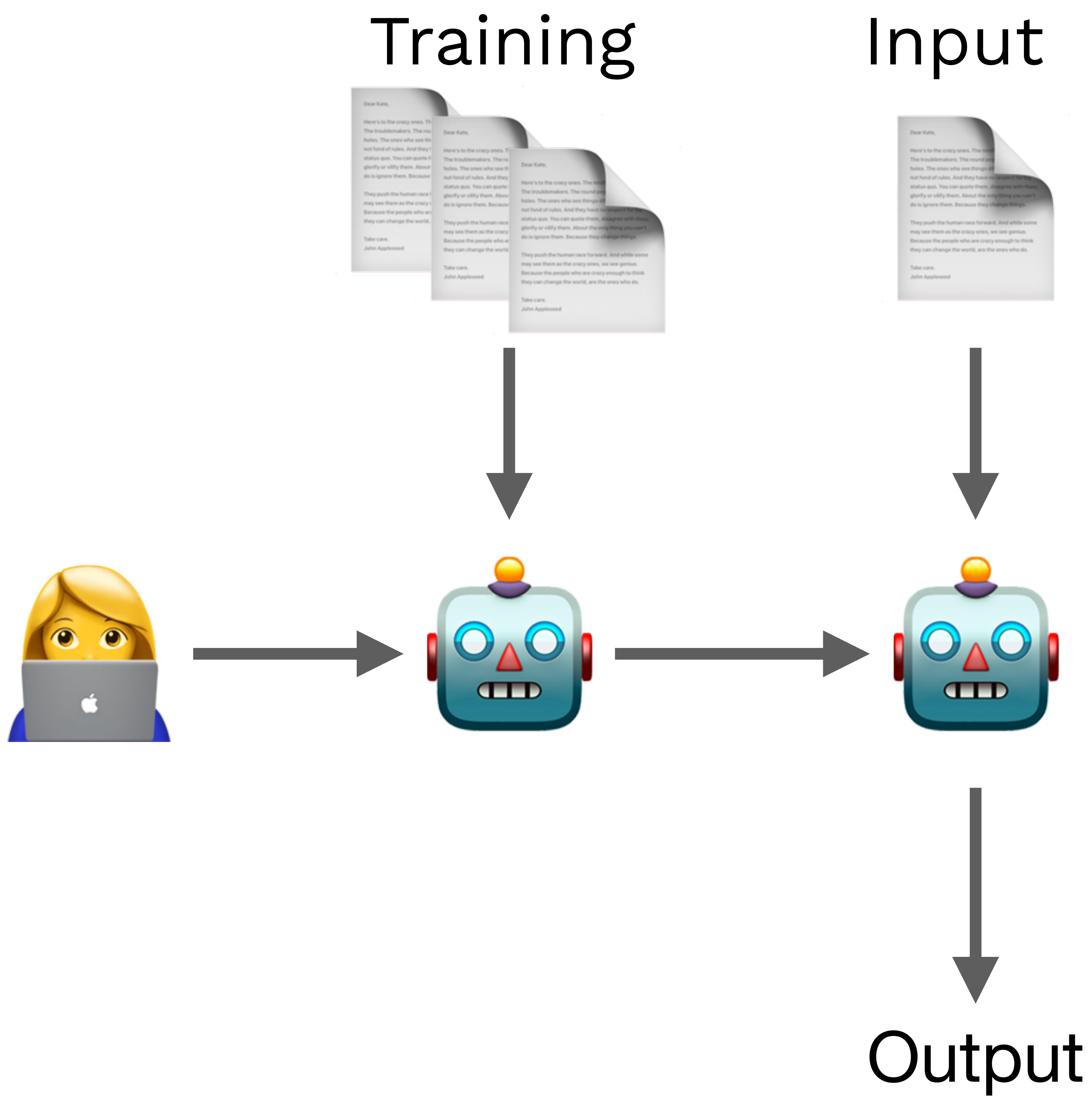
00

Foundations of Machine Learning

Traditional Programming vs Machine Learning



Software 1.0



Software 2.0

Types of Machine Learning

Unsupervised Learning	Supervised Learning	Reinforcement Learning
Learn structure of data to generate more data	Learn how data maps to labels to recognize or predict	Learn how to act in an environment to obtain reward

"This product does what it is supposed __"

→ *cat*

→ *"Hey Siri"*

Converging on just...

Supervised or Self-supervised Learning

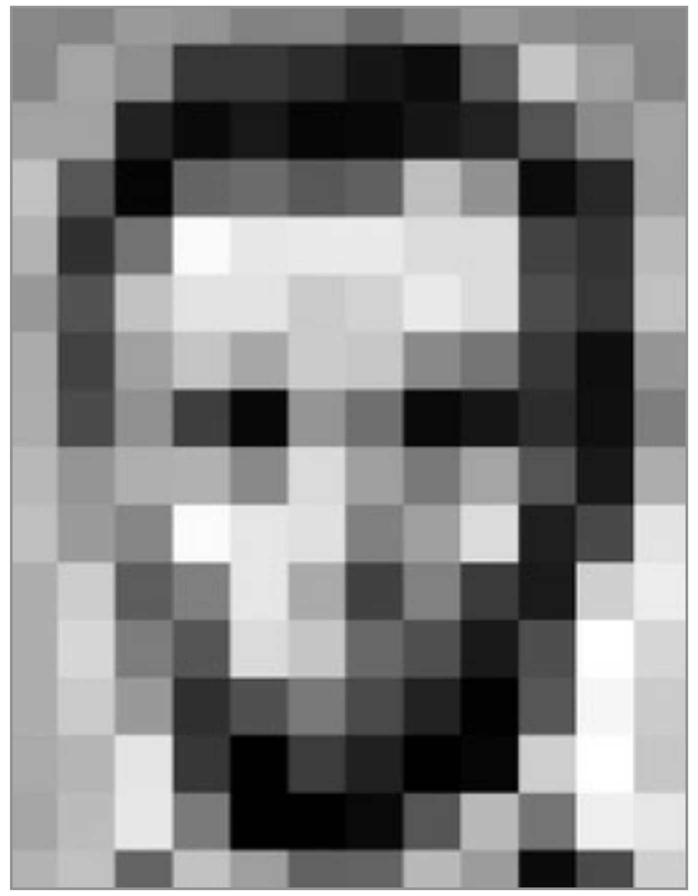
"This product does what it is supposed __" → *"to."*

→ *cat*

→ *"Hey Siri"*

→ *next move*

Inputs and outputs are always just numbers

	Input	Output
What we see		"Lincoln"
What the machine "sees"	(table below)	[76, 105, 110, 99, 111, 108, 110]

157	153	174	168	150	152	129	151	172	161	155
155	182	163	74	75	62	33	17	110	210	180
180	180	50	14	34	6	10	33	48	106	159
206	109	5	124	131	111	120	204	166	15	56
194	68	137	251	237	239	239	228	227	87	71
172	105	207	233	233	214	220	239	228	98	74
188	88	179	209	185	215	211	158	139	75	20
189	97	165	84	10	168	134	11	31	62	22
199	168	191	193	158	227	178	143	182	106	36
205	174	155	252	236	231	149	178	228	43	95
190	216	116	149	236	187	86	150	79	38	218
190	224	147	108	227	210	127	102	36	101	255

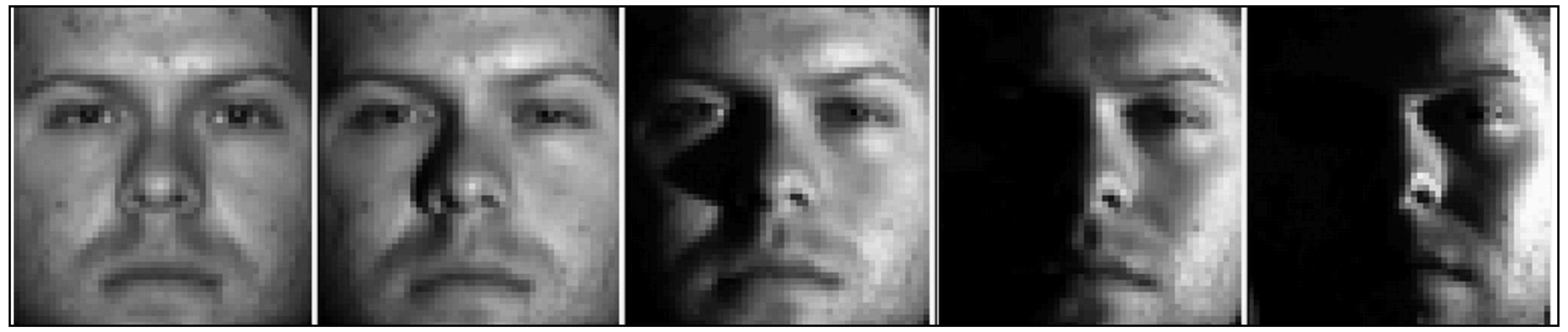
Why is this hard?

- Infinite variety of inputs can all mean the same thing
- Meaningful differences can be tiny
- Structure of the world is complex

"I loved this movie"

"As good as The Godfather"

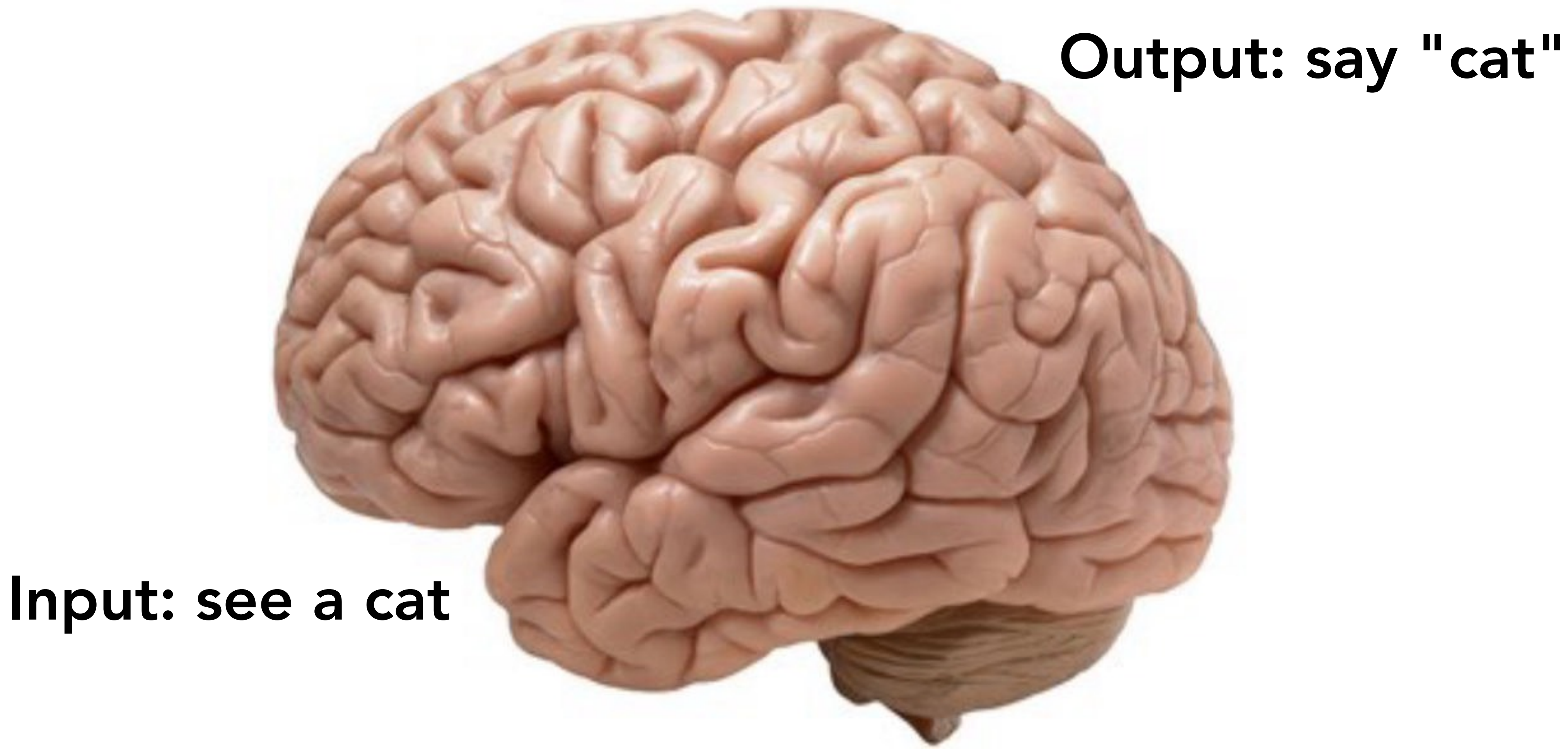
"🔥 no cap"



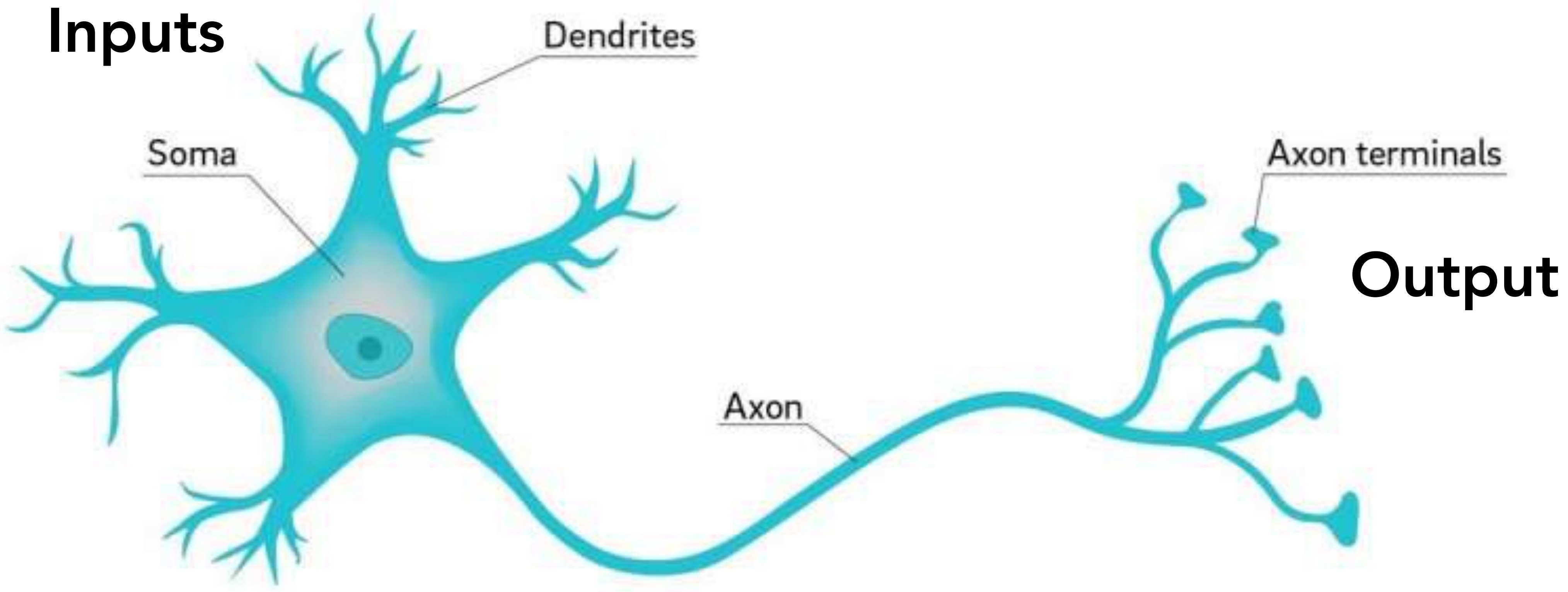
How is it done?

- Many methods for Machine Learning
  - Logistic Regression
  - Support Vector Machines
  - Decision Trees (xgboost)
- But one is dominant
  - Neural Networks (also called Deep Learning)

Inspiration



https://www.the-scientist.com/the-nutshell/what-made-human-brains-so-big-36663

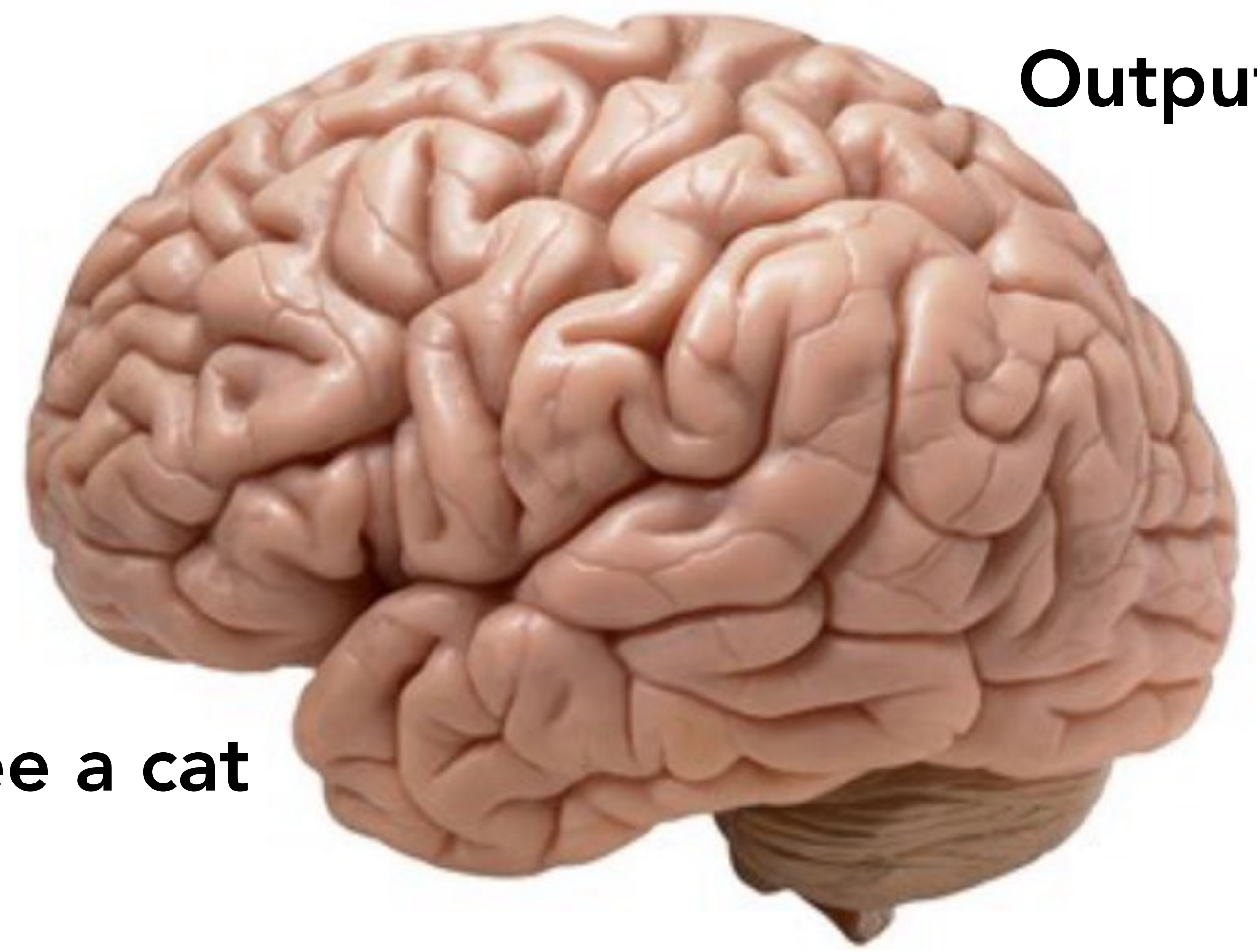


https://medicalxpress.com/news/2018-07-neuron-axons-spindly-theyre-optimizing.html

- Inspired by what we know to be intelligent: the **brain**
- The brain is composed of billions of **neurons**
- Each neuron receives electrical **inputs** and sends an electrical **output**
- The brain itself has high-level inputs and outputs

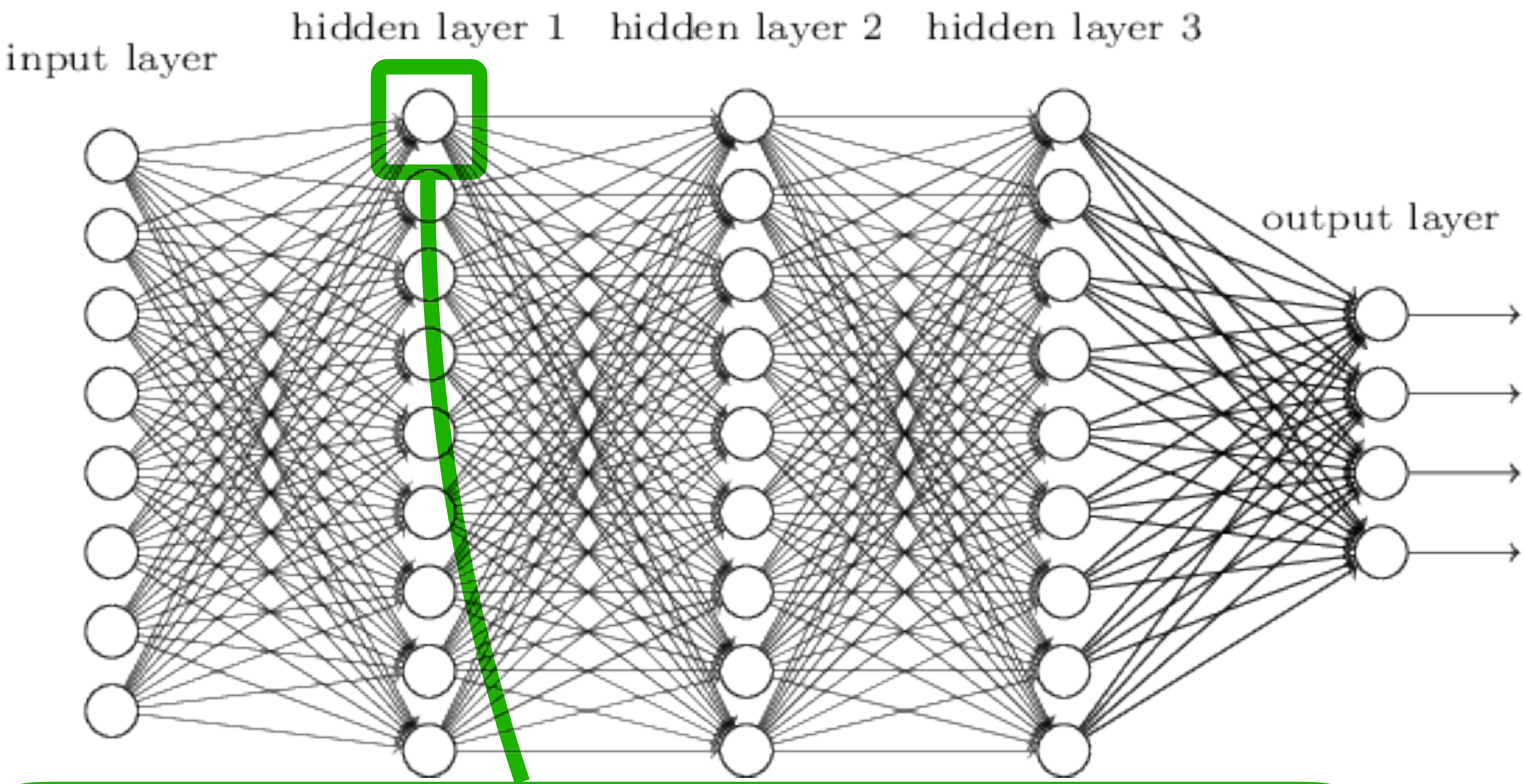
Formalization

Output: say "cat"

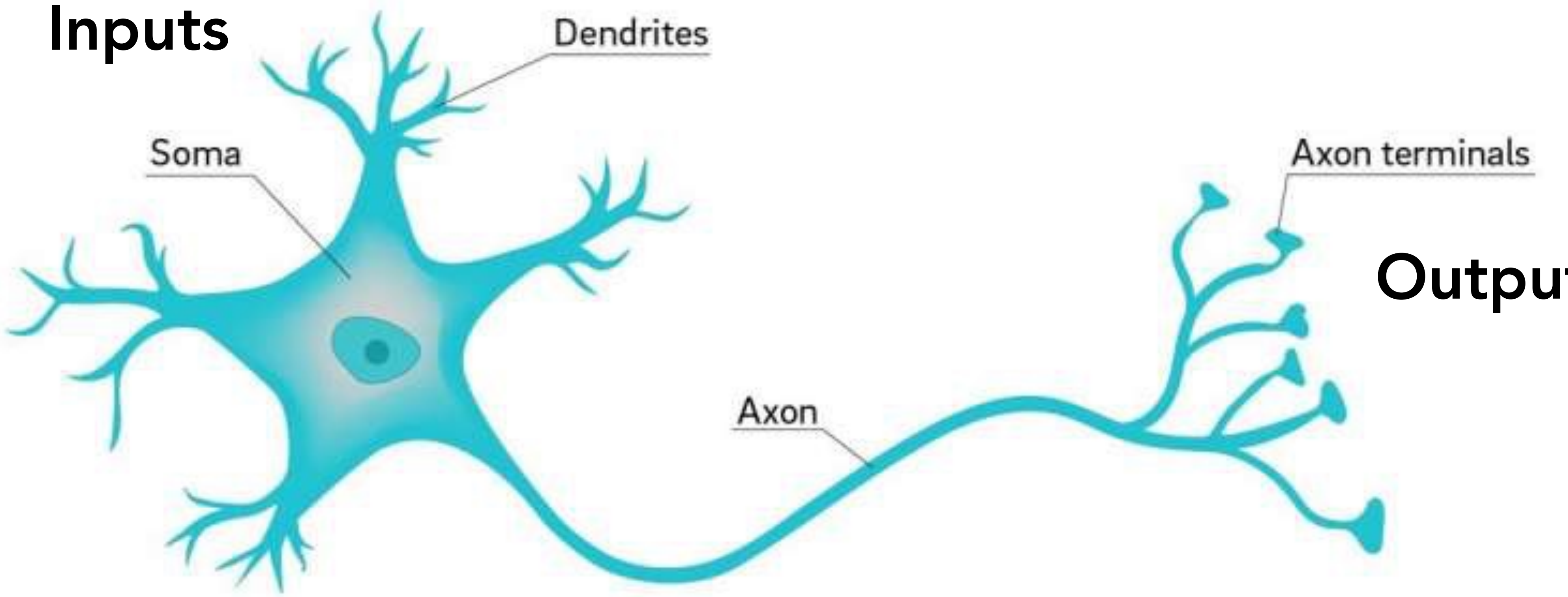


Input: see a cat

https://www.the-scientist.com/the-nutshell/what-made-human-brains-so-big-36663



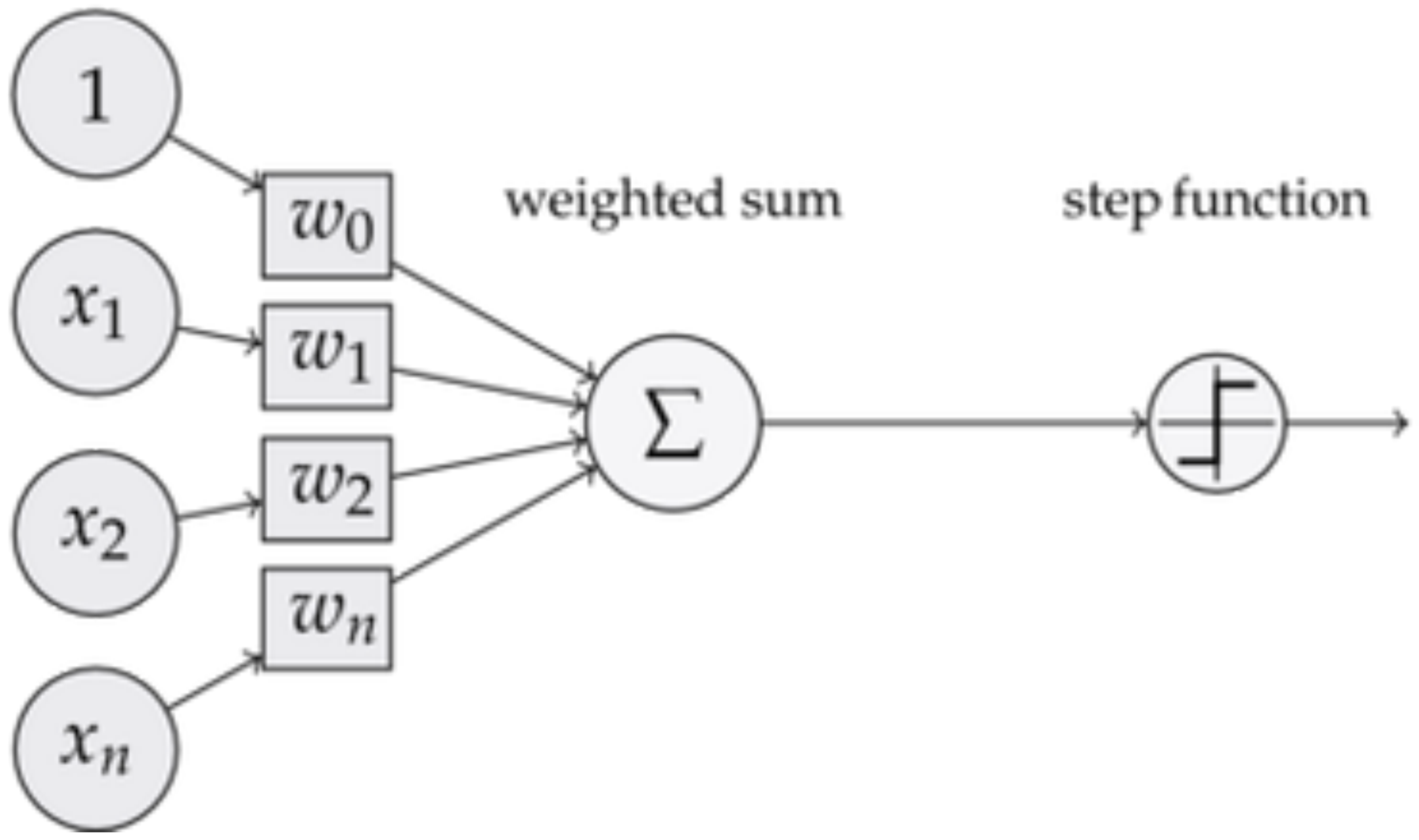
Inputs



Output

https://medicalxpress.com/news/2018-07-neuron-axons-spindly-theyre-optimizing.html

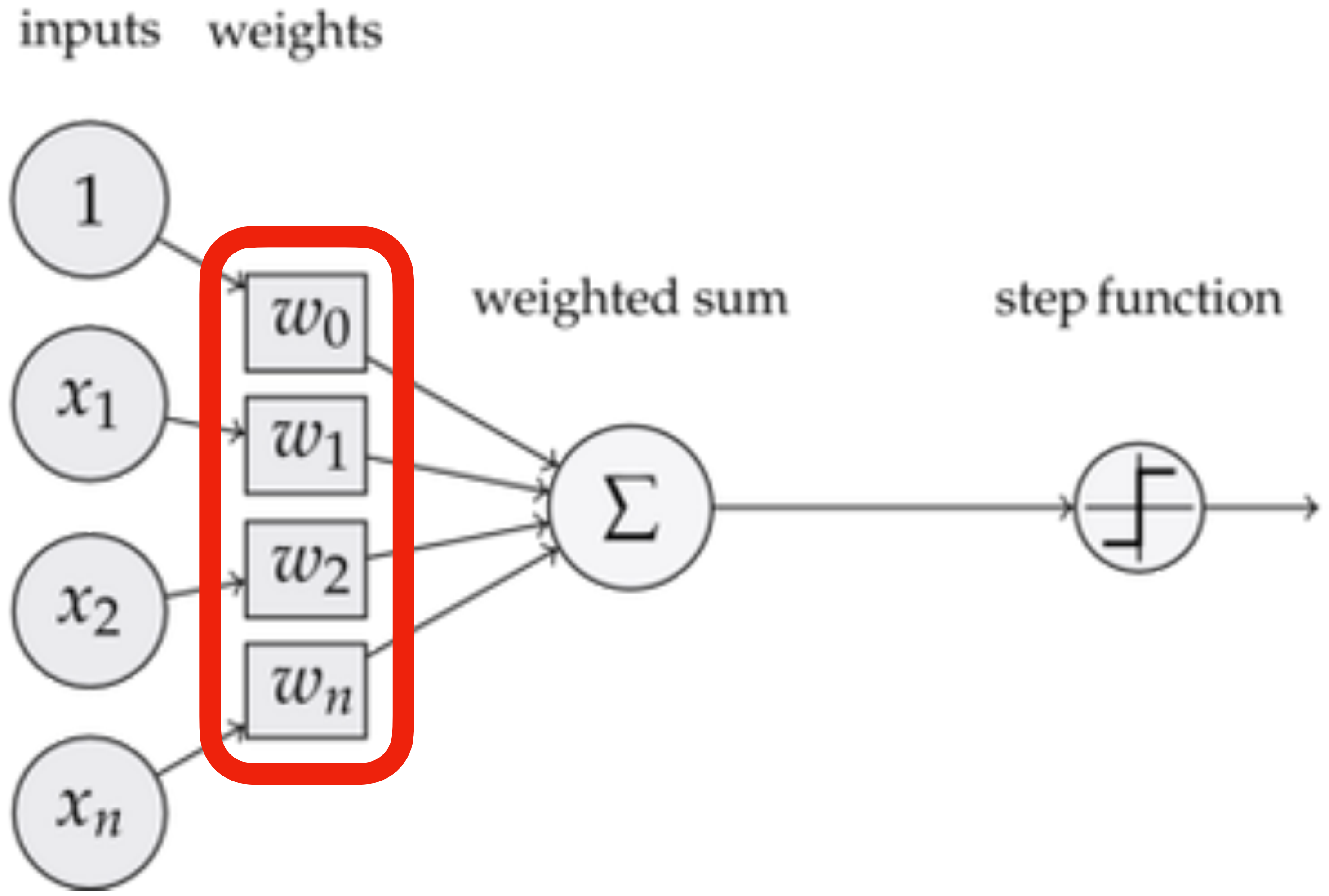
Inputs



Output

https://www.jessicayung.com/explaining-tensorflow-code-for-a-multilayer-perceptron/

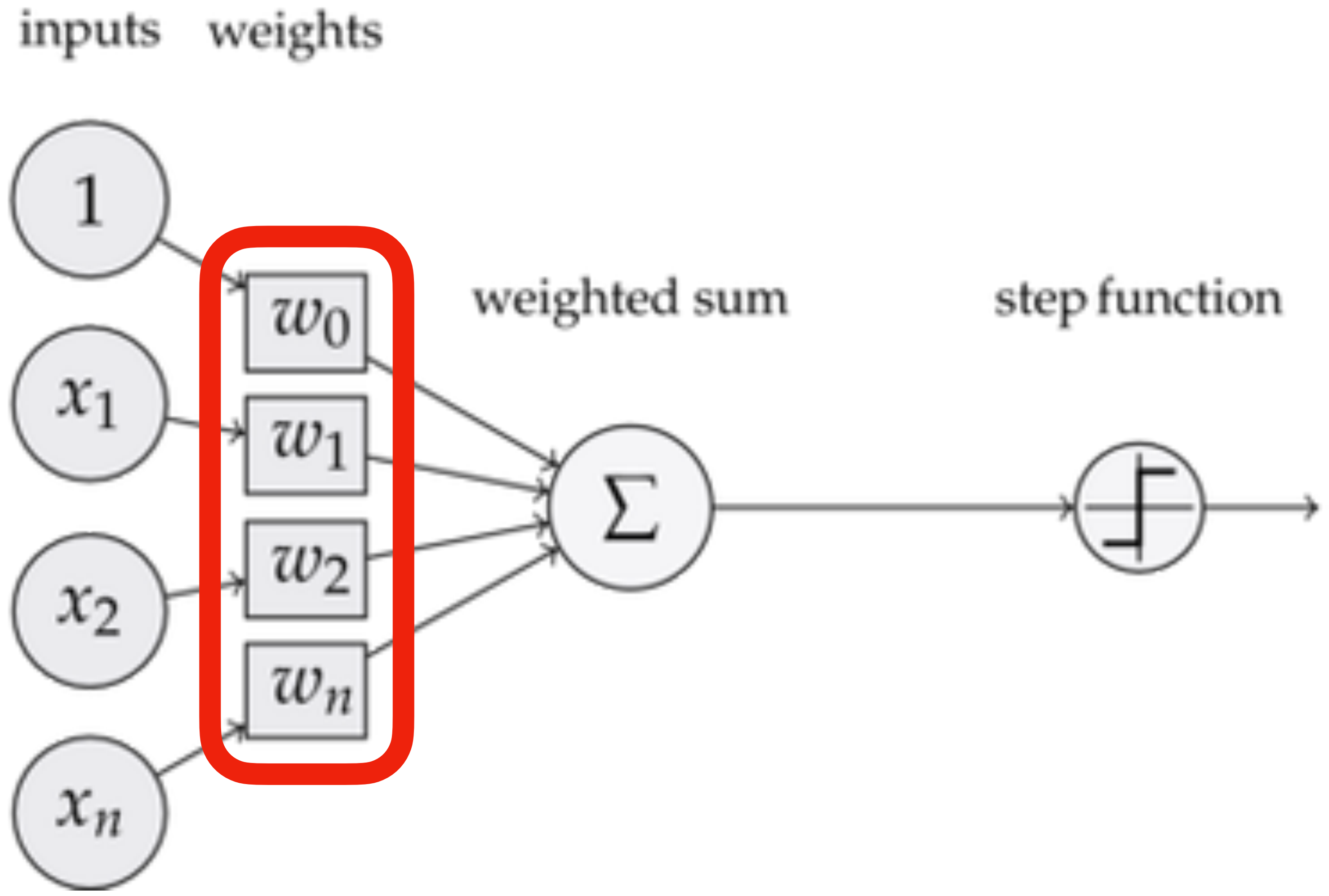
A "perceptron" is a vector of numbers



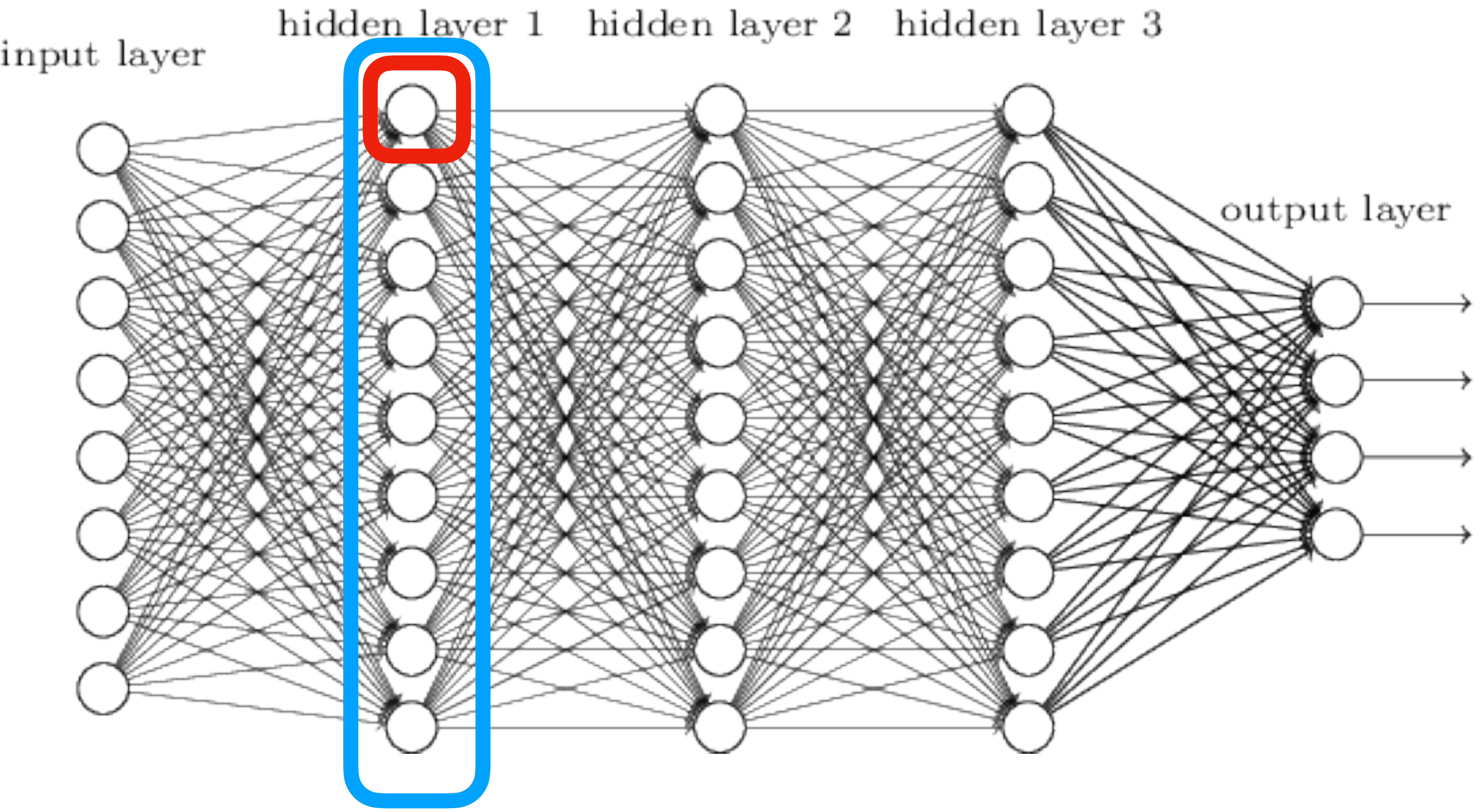
https://www.jessicayung.com/explaining-tensorflow-code-for-a-multilayer-perceptron/

```
[[0.58  0.841 0.835 0.18 ]
```

A "layer" is a matrix of numbers



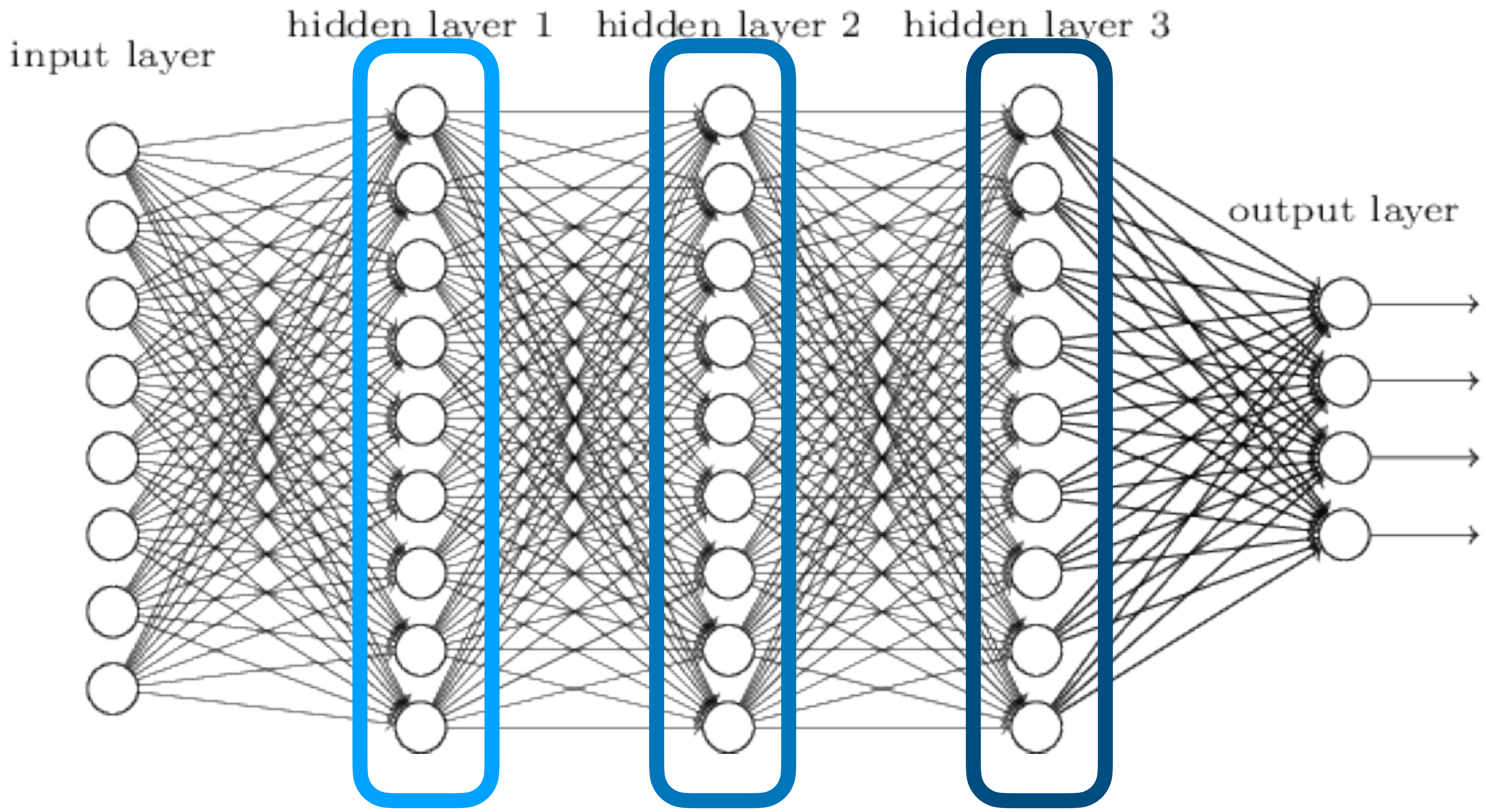
https://www.jessicayung.com/explaining-tensorflow-code-for-a-multilayer-perceptron/

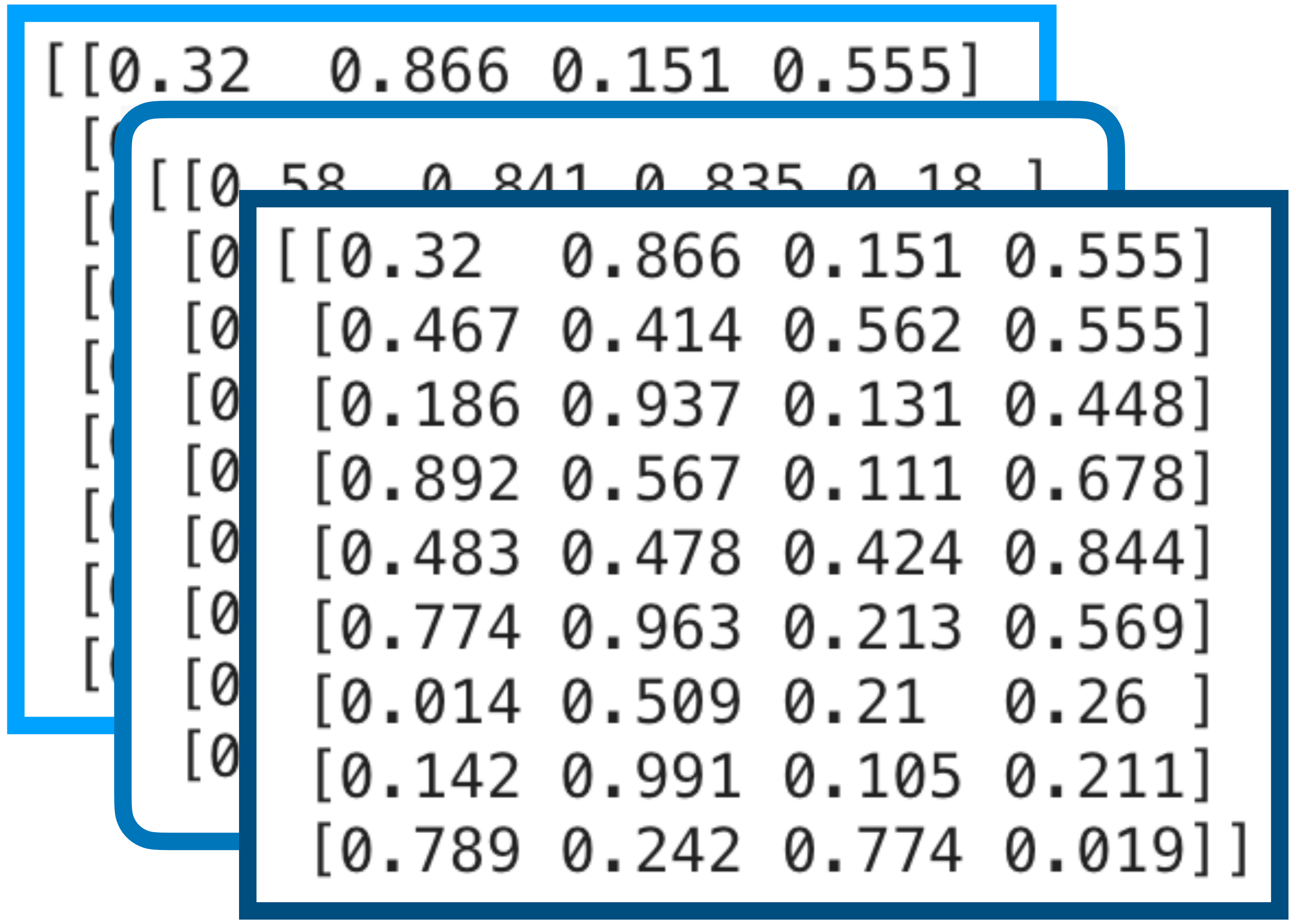



```
[[0.58  0.841 0.835 0.18 ]
 [0.405 0.813 0.309 0.562]
 [0.422 0.229 0.46  0.152]
 [0.673 0.429 0.441 0.243]
 [0.9   0.744 0.234 0.856]
 [0.971 0.486 0.175 0.248]
 [0.258 0.588 0.478 0.266]
 [0.236 0.496 0.077 0.557]
 [0.413 0.322 0.372 0.741]]
```

The neural network is a set of matrices

Called "parameters" or "weights"

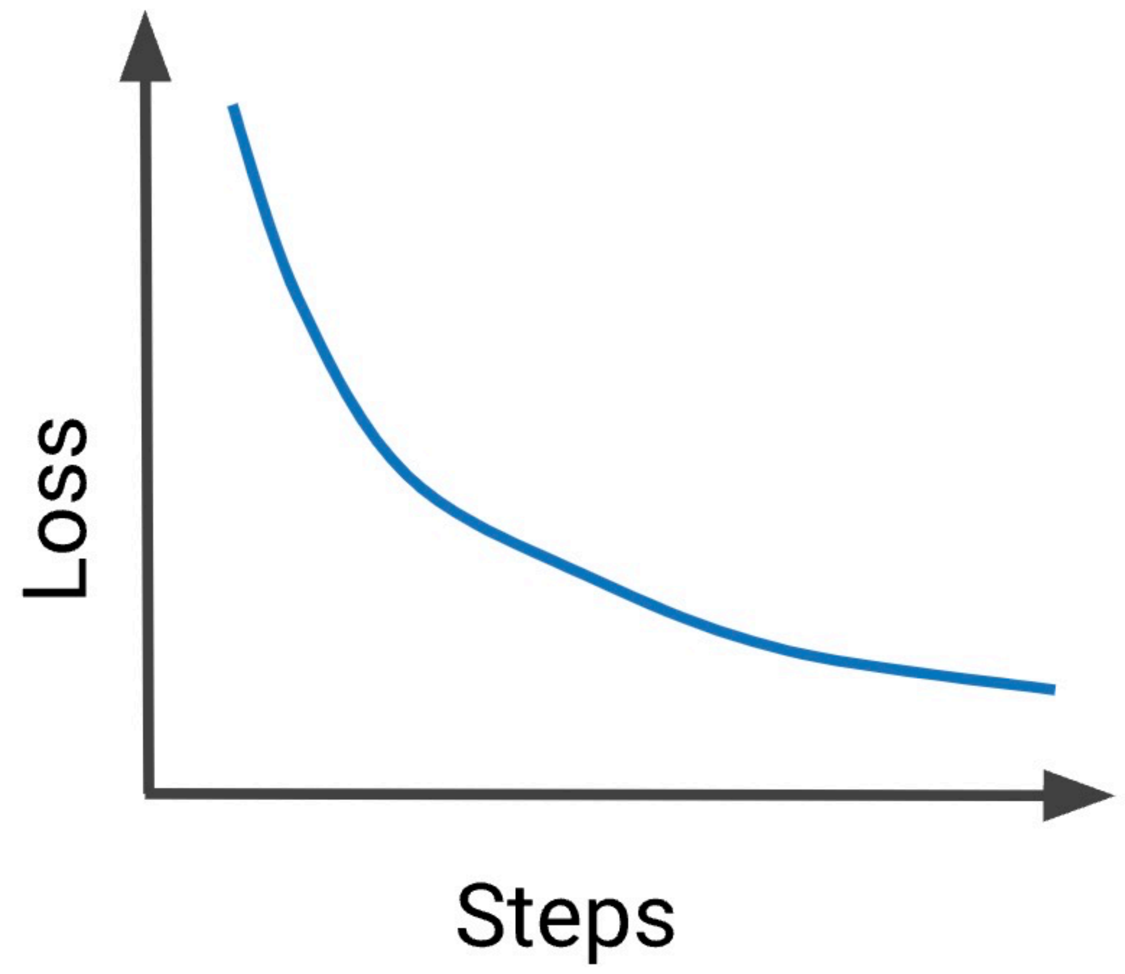




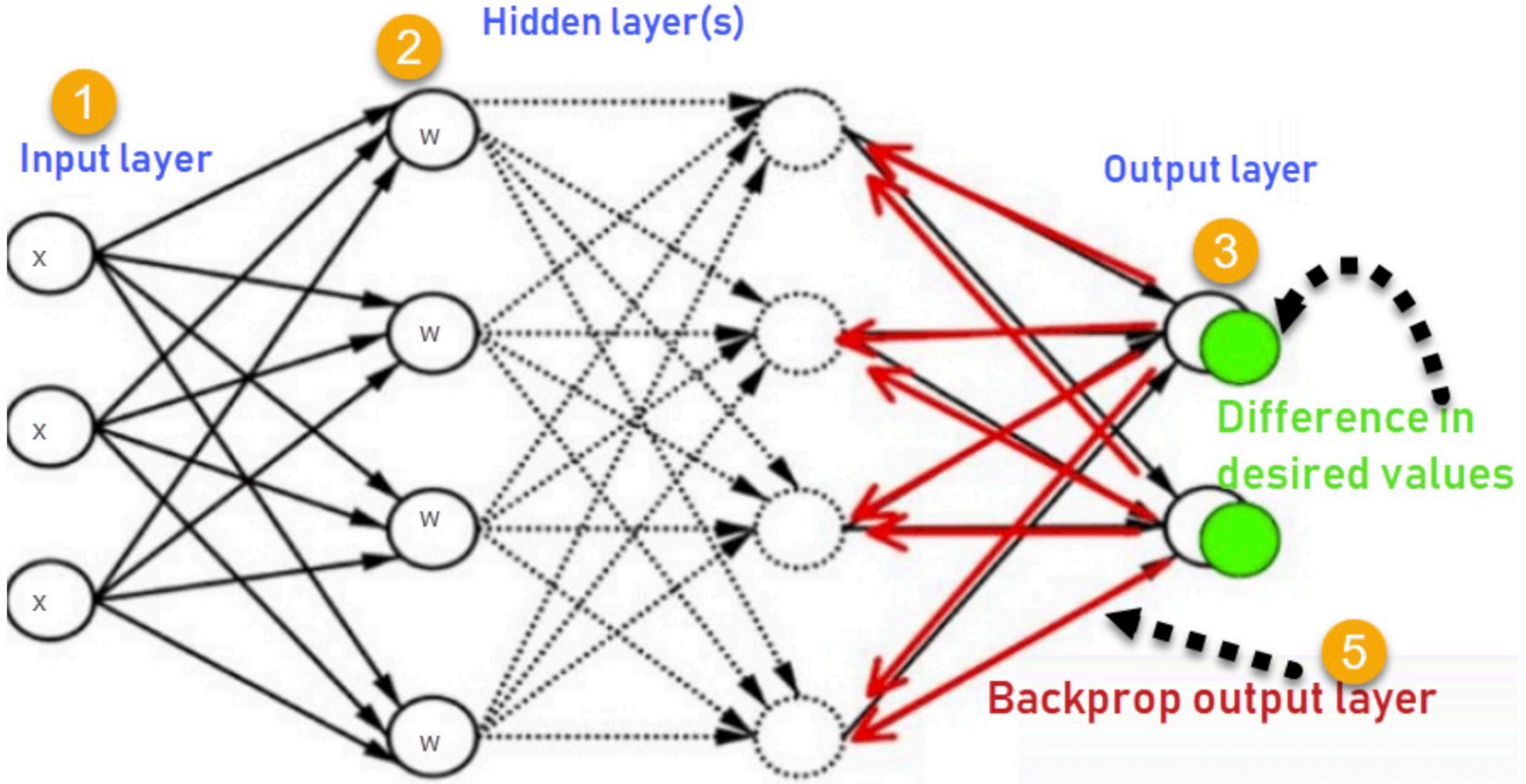
NN operations are just matrix multiplications.
GPUs are really fast at matrix multiplications.

Training

$$loss_{CE} = -\sum(y_i * \log(y'_i))$$



- Data X (e.g. images), labels y (e.g. labels)
- Take a little batch of data x:
  - Use the current model to make a prediction x -> y'
  - Compute loss(y, y')
  - Back-propagate the loss through all the layers of the model
- Repeat until loss stops decreasing

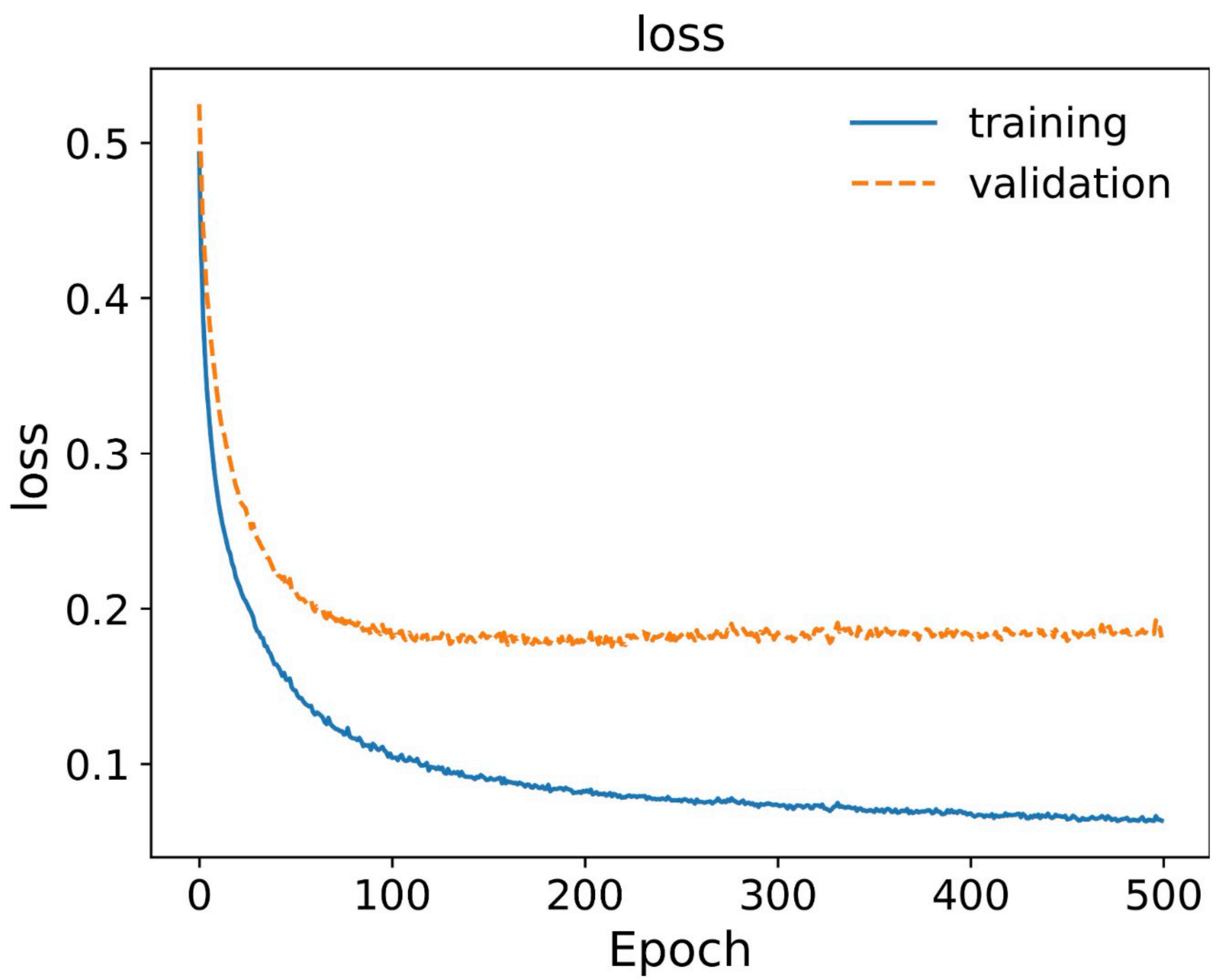


https://www.guru99.com/backpropogation-neural-network.htm

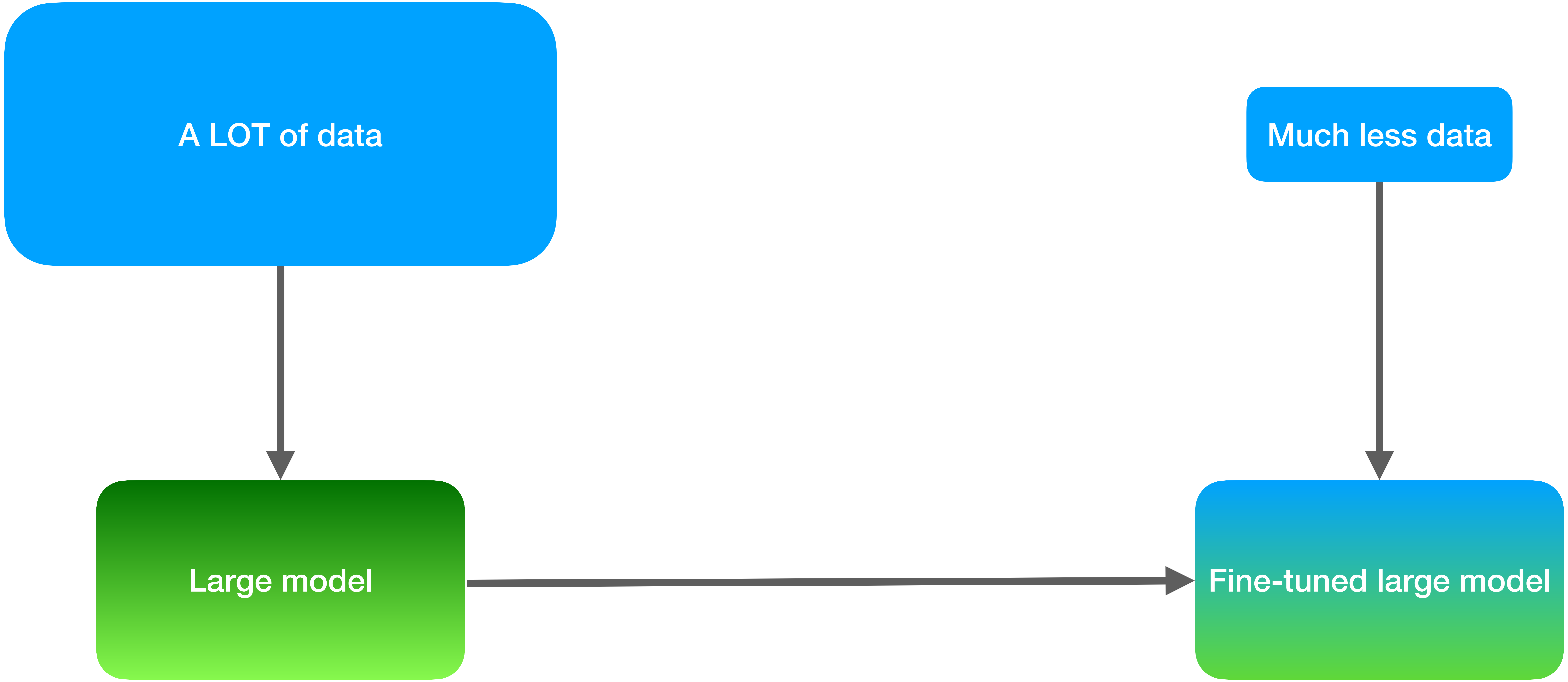
https://developers.google.com/machine-learning/testing-debugging/metrics/interpretic

Dataset Splitting

- Split (X, y) into training (~80%), validation (~10%), and test (~10%) sets
- Validation set is for
  - ensuring that training is not "overfitting"
  - setting hyper-parameters of the model (e.g. number of parameters)
- Test set is for measuring validity of predictions on new data



THIS APPLIES TO YOUR EXPERIMENTATION WITH PROMPTS!

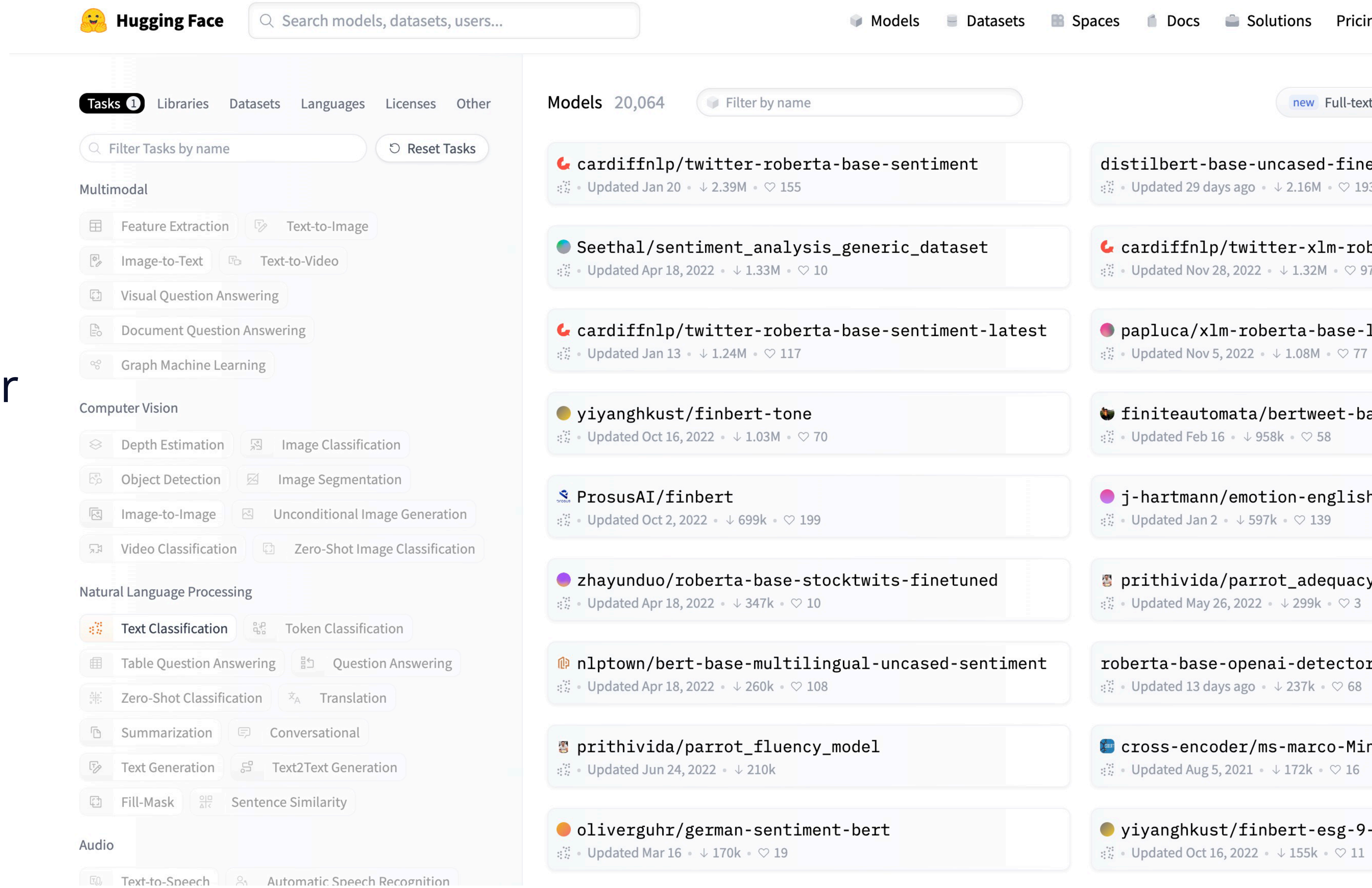


Pre-training:
slow training on a lot of data

Fine-tuning:
fast training on a little data

Model Hubs

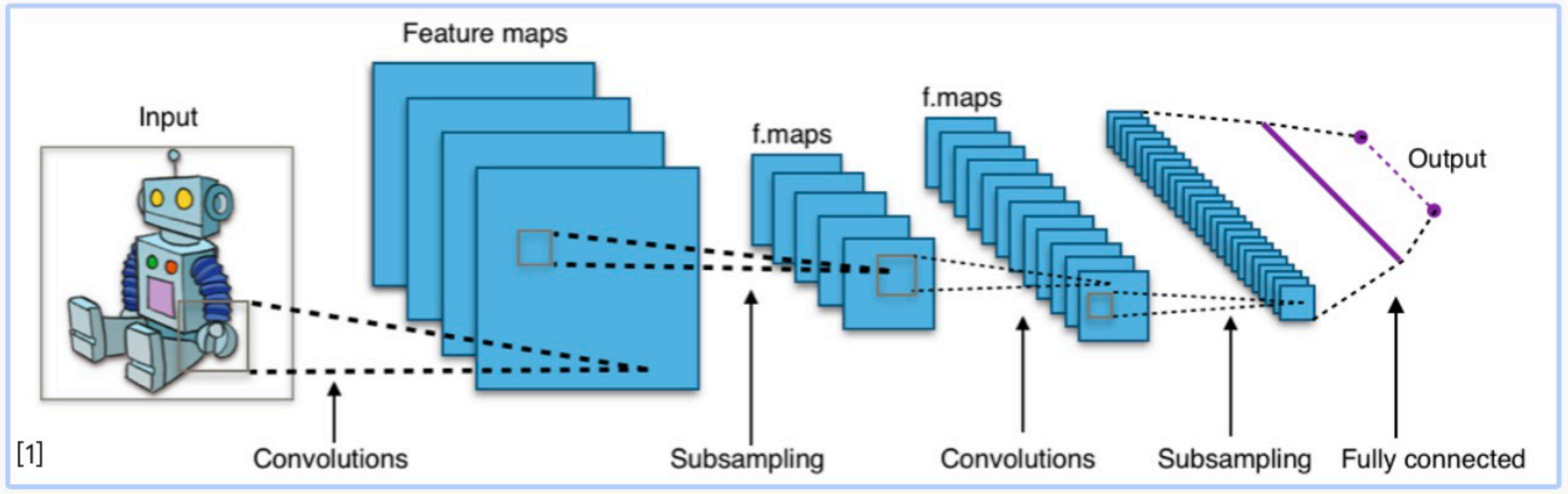
- People share pre-trained models!
- 🤗: the most popular Model Hub
  - 180K models
  - 30K datasets



Before ~2020: each task had its own NN architecture

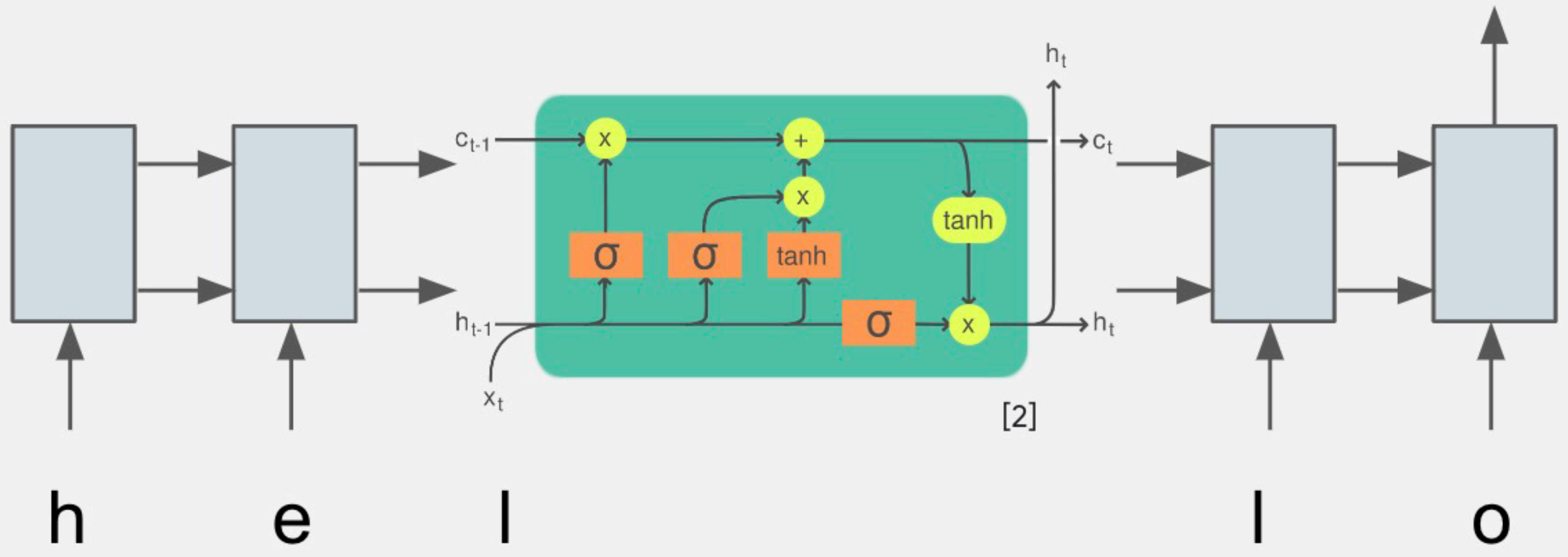
Computer Vision

Convolutional NNs (+ResNets)



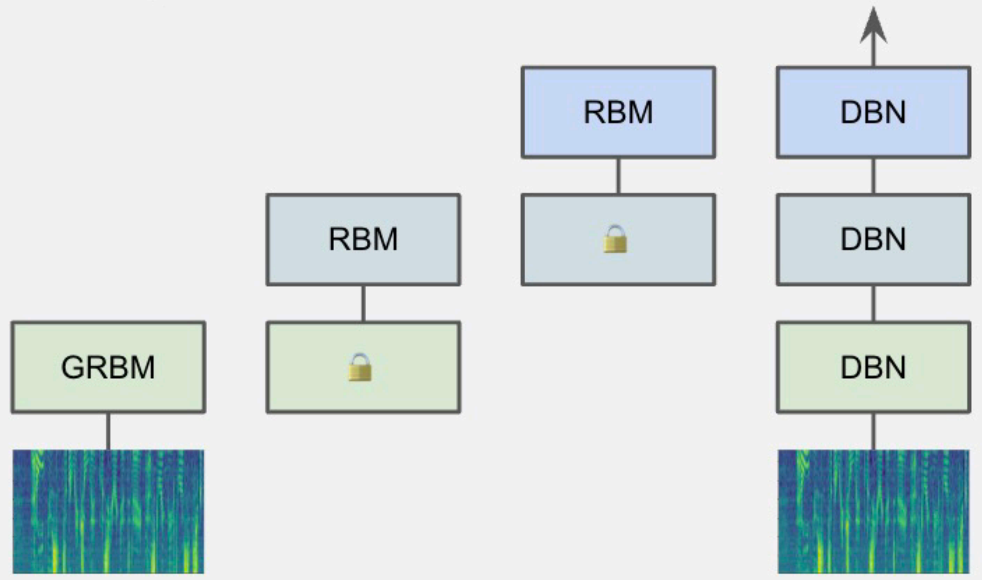
Natural Lang. Proc.

Recurrent NNs (+LSTMs)



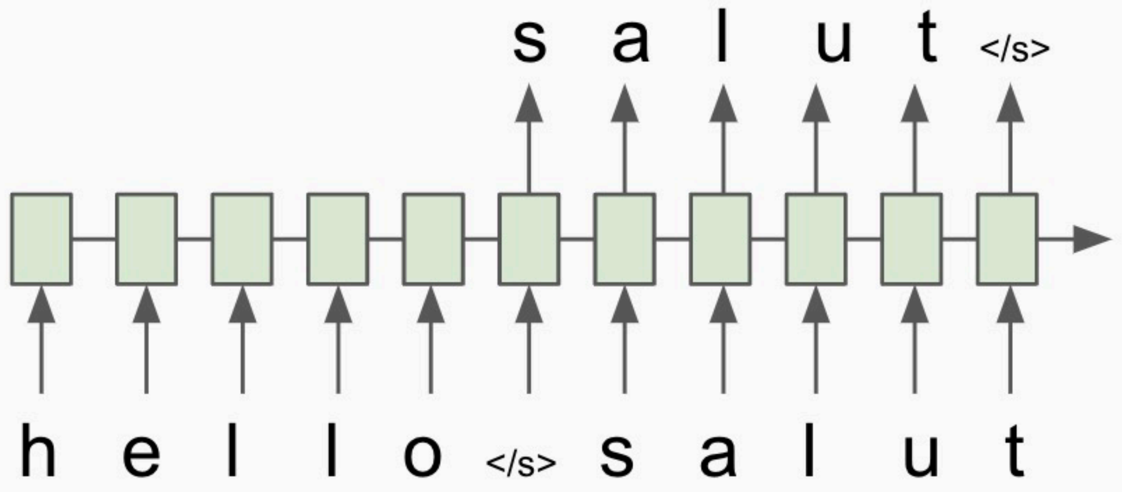
Speech

Deep Belief Nets (+non-DL)



Translation

Seq2Seq



RL

BC/GAIL

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \quad (17)$$

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

$$\begin{aligned} &\hat{\mathbb{E}}_{\tau_i}[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)] - \lambda \nabla_\theta H(\pi_\theta), \\ &\text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \end{aligned} \quad (18)$$

6: **end for**

http://lucasb.eyer.be/transformer

Now: all is Transformers

Transformer cartoon (DALL-E)

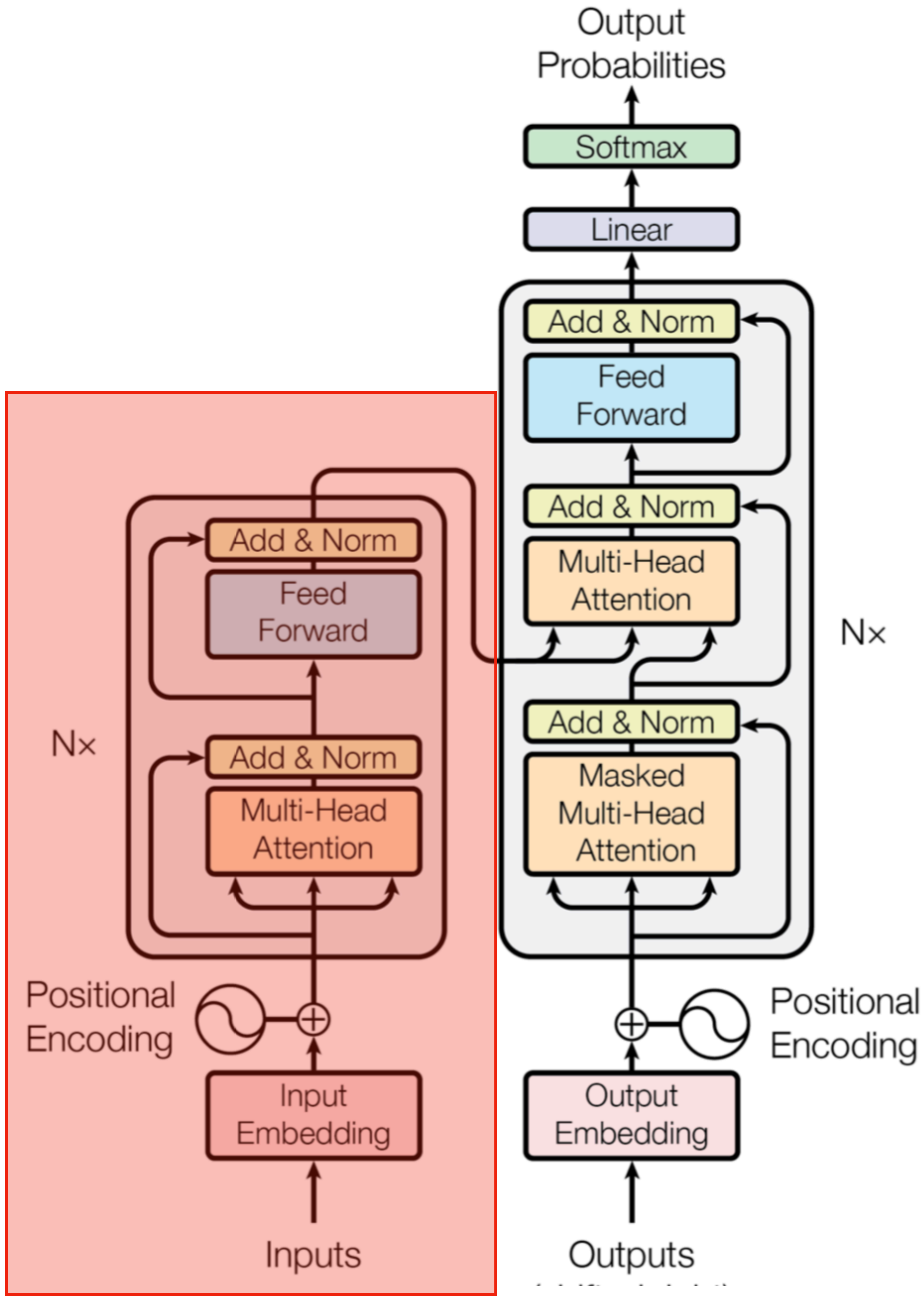
01

The Transformer Architecture

Attention is all you need (2017)

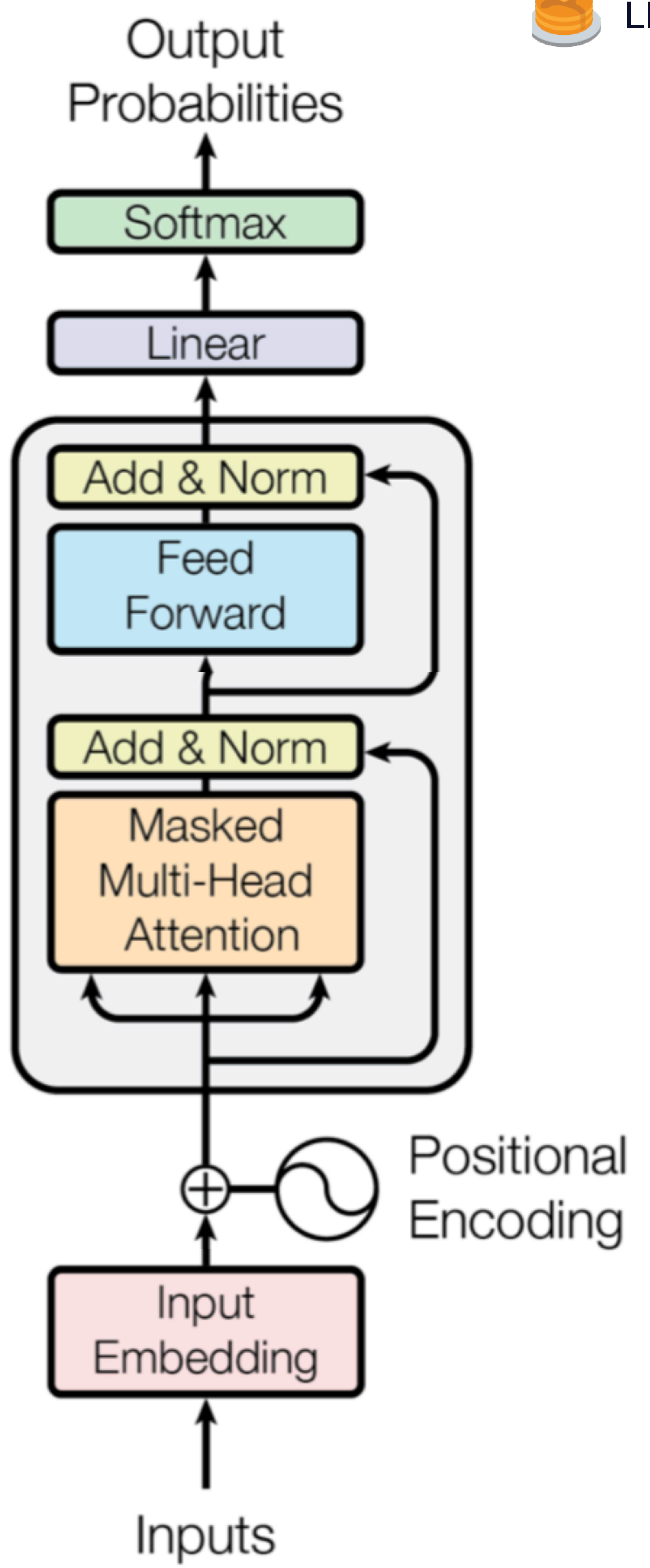
- Ground-breaking architecture that set SOTA on first translation and later all other NLP tasks

- For simplicity, can just look at one half of it



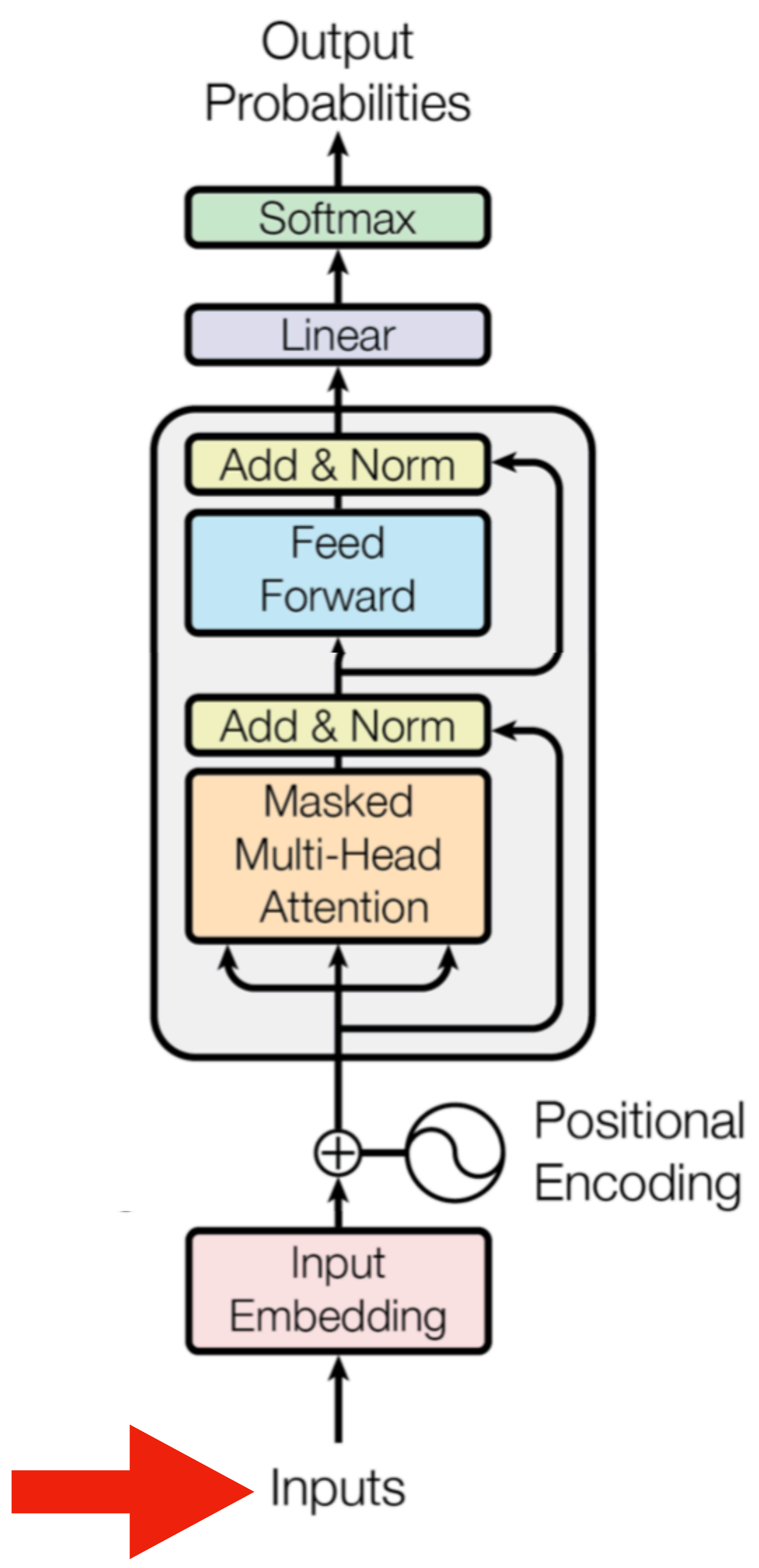
Transformer Decoder Overview

- Task is to complete text
  - "It's a blue" -> "sundress"
- Inputs: a sequence of N tokens
  - [It's, a, blue]
- Output:
  - Probability distribution over the next token
- Inference:
  - Sample the next token from the distribution, append it to inputs, run through the model again, sample, append, etc.



Inputs

- Inputs need to be vectors of numbers
- Start with original text:
  - "It's a blue sundress."
- Turn into a sequence of tokens:
  - [<SOS>, It, 's, a, blue, sund, ress, ., <EOS>]
- Turn into vocabulary IDs:
  - [0, 1026, 338, 257, 4171, 37437, 601, 13, 1]
- Each ID can be represented by a one-hot vector
  - e.g. 3 -> [0, 0, 0, 1, 0, 0, 0, 0, ...]



Input Embedding

- One-hot vectors are poor representations of words or tokens
  - e.g. distance between "cat" and "kitten" is the same as between "cat" and "tractor"
- Solution: learn an embedding matrix!
  - (The simplest NN layer type)

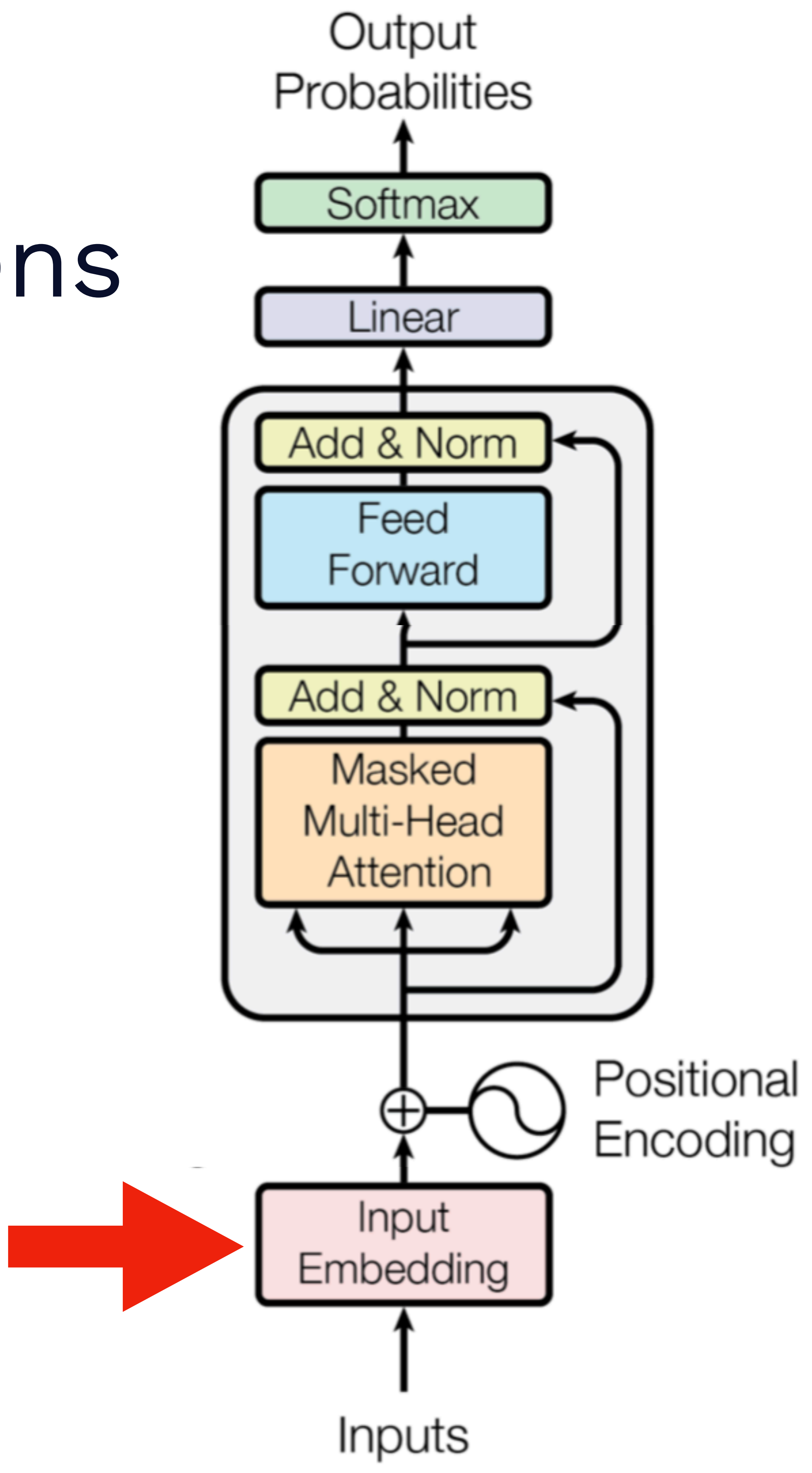
aardvark	1	0	0	...	0	0	0
black	0	0	...	1	...	0	0
cat	0	0	...	1	...	0	0
duvet	0	0	...	1	...	0	0
zombie	0	0	0	...	0	0	1

VxV

VxE
embedding matrix

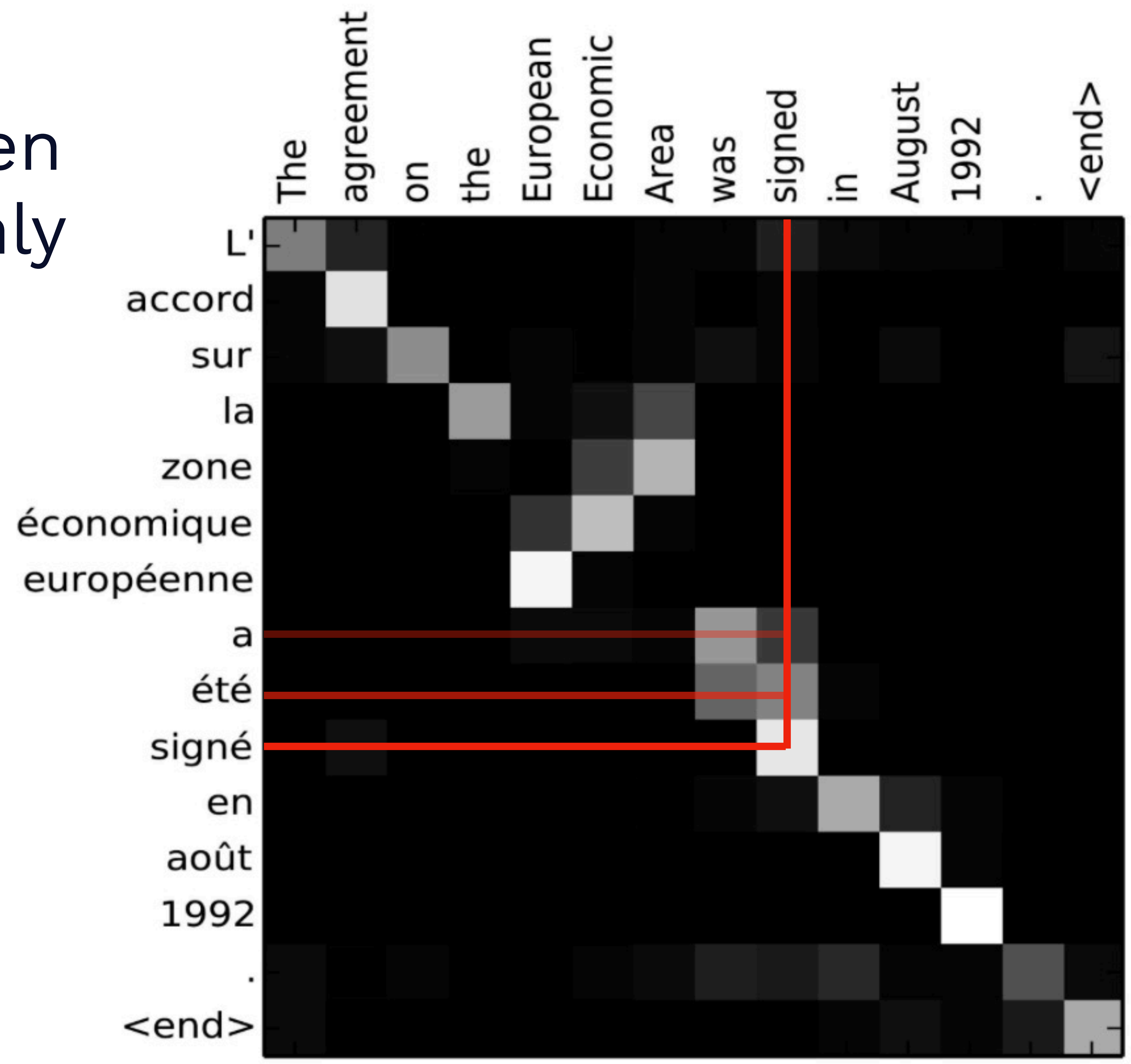
aardvark	0.97	0.03	0.15	0.04
black	0.07	0.01	0.20	0.95
cat	0.98	0.98	0.45	0.35
duvet	0.01	0.84	0.12	0.02
zombie	0.74	0.05	0.98	0.93

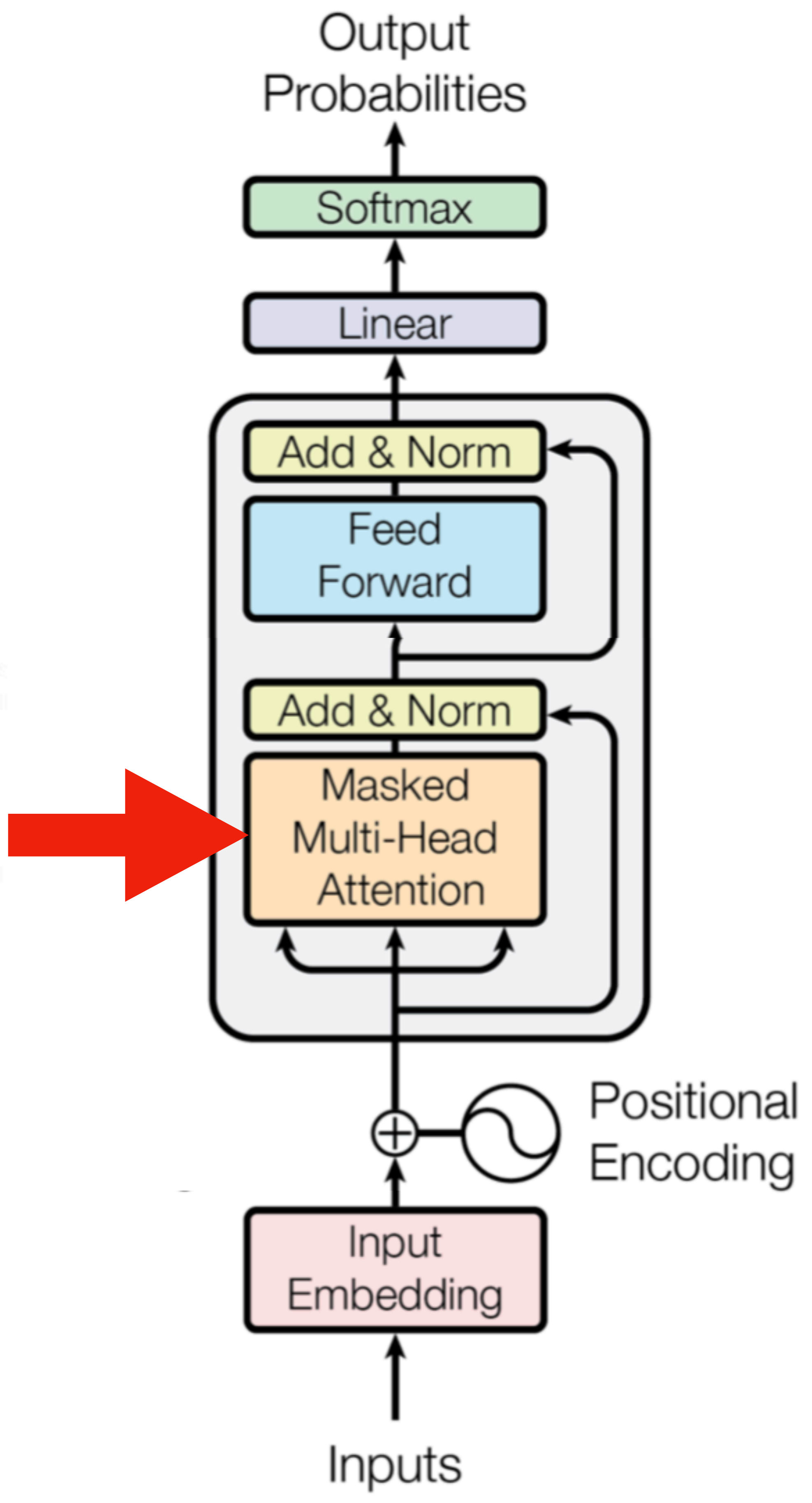
VxE



Attention

- (Ignore "Masked Multi-Head" for now)
- Key insight: for a given token in the output sequence, only one or a few tokens in the input sequence are most important
- Introduced in 2015 for translation tasks





https://lilianweng.github.io/posts/2018-06-24-attention/

Basic self-attention

- Input: sequence of vectors

$$\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_t$$

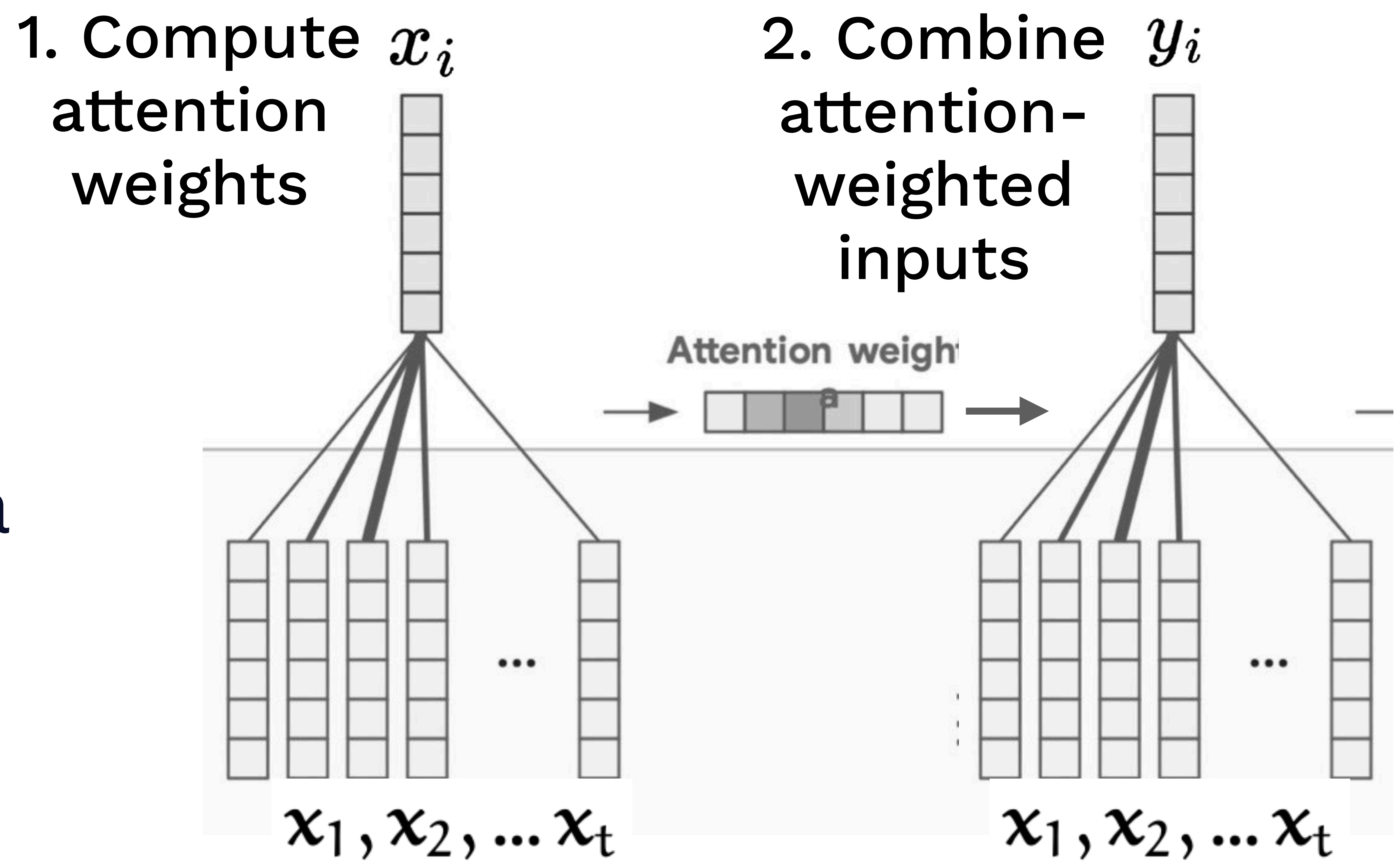
- Output: sequence of vectors, each one a weighted sum of the input sequence

$$\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_t$$

$$\mathbf{y}_i = \sum_j w_{ij} \mathbf{x}_j$$

- weight is just dot product between input vectors $\quad w'_{ij} = \mathbf{x}_i{}^\mathsf{T} \mathbf{x}_j$

- (made to sum to 1) $\quad w_{ij} = \dfrac{\exp w'_{ij}}{\sum_j \exp w'_{ij}}$

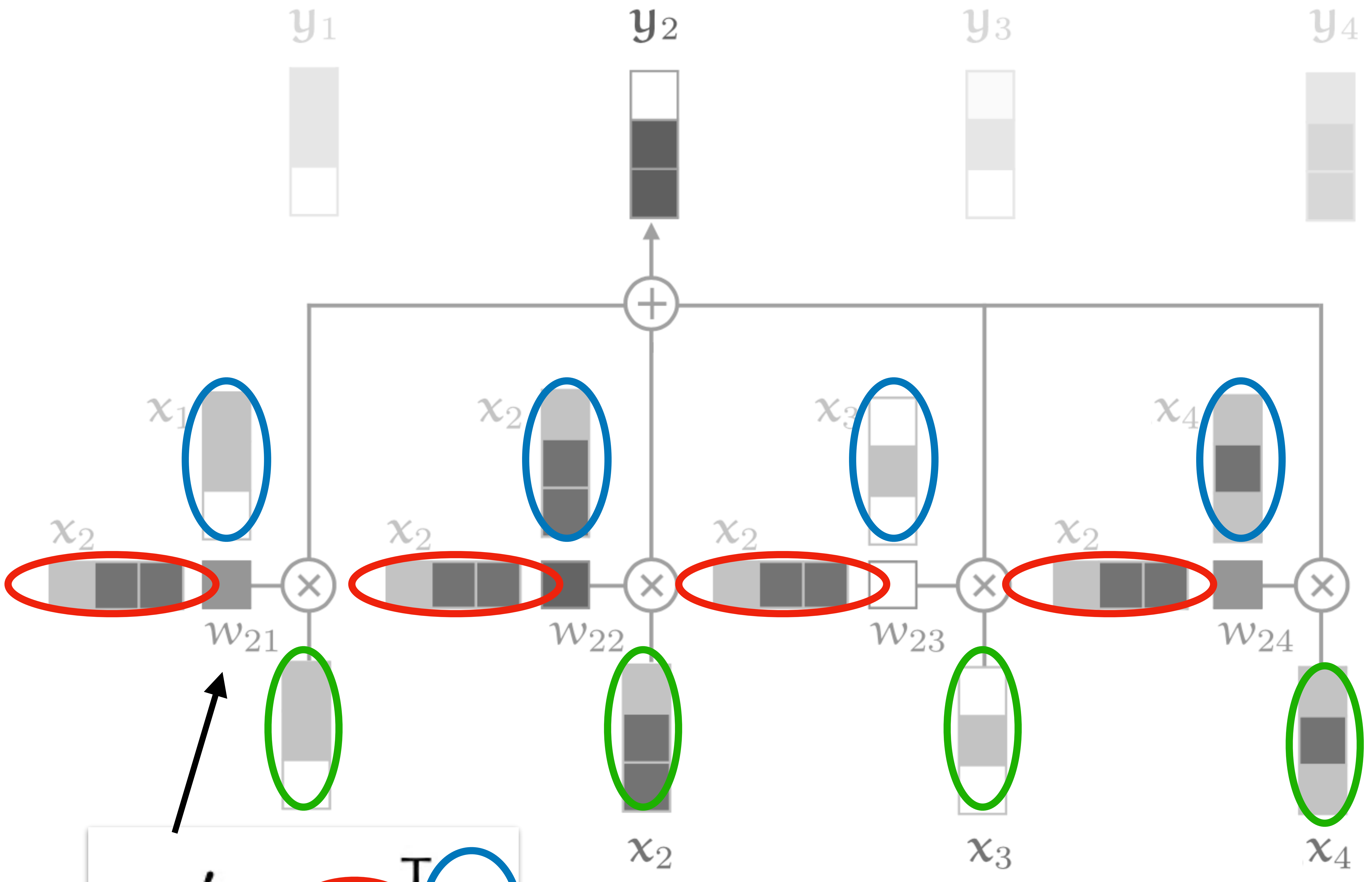


http://www.peterbloem.nl/blog/transformers

Basic self-attention

$$\mathbf{y}_i = \sum_j w_{ij} \mathbf{x}_j$$

- Note that every input vector x_i is used in 3 ways:
  - Query x_i
  - Key $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_t$
  - Value $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_t$

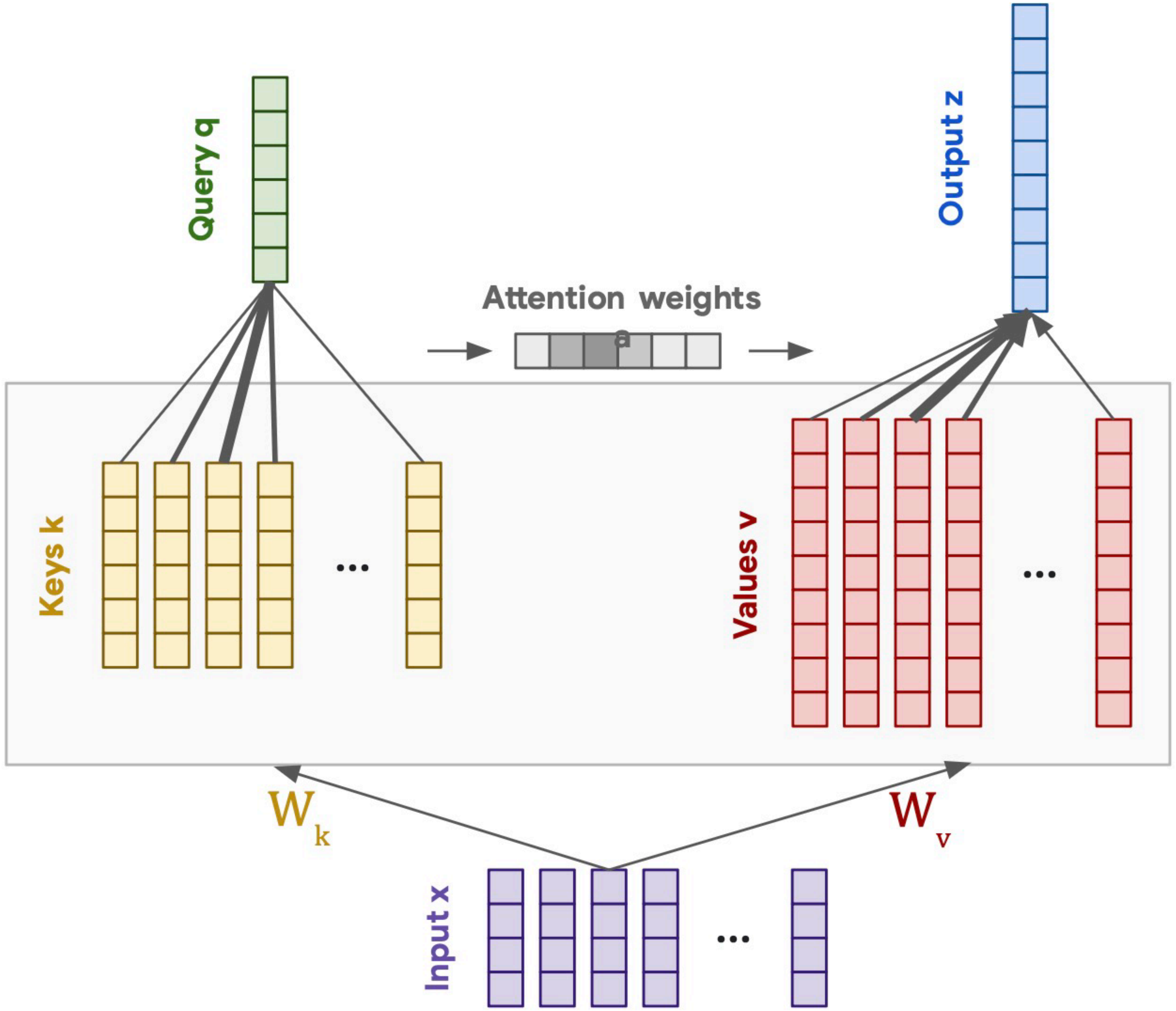


$$w'_{ij} = \mathbf{x}_i^{\mathsf{T}} \mathbf{x}_j$$

http://www.peterbloem.nl/blog/transformers

Basic self-attention

- Problem: there's no learning involved!
- Solution: project inputs into query, key, value roles
- Learning these matrices = learning attention



http://lucasb.eyer.be/transformer

$$\mathbf{q}_i = \mathbf{W}_q \mathbf{x}_i \qquad \mathbf{k}_i = \mathbf{W}_k \mathbf{x}_i \qquad \mathbf{v}_i = \mathbf{W}_v \mathbf{x}_i$$

$$w'_{ij} = {\mathbf{q}_i}^\mathsf{T} \mathbf{k}_j$$

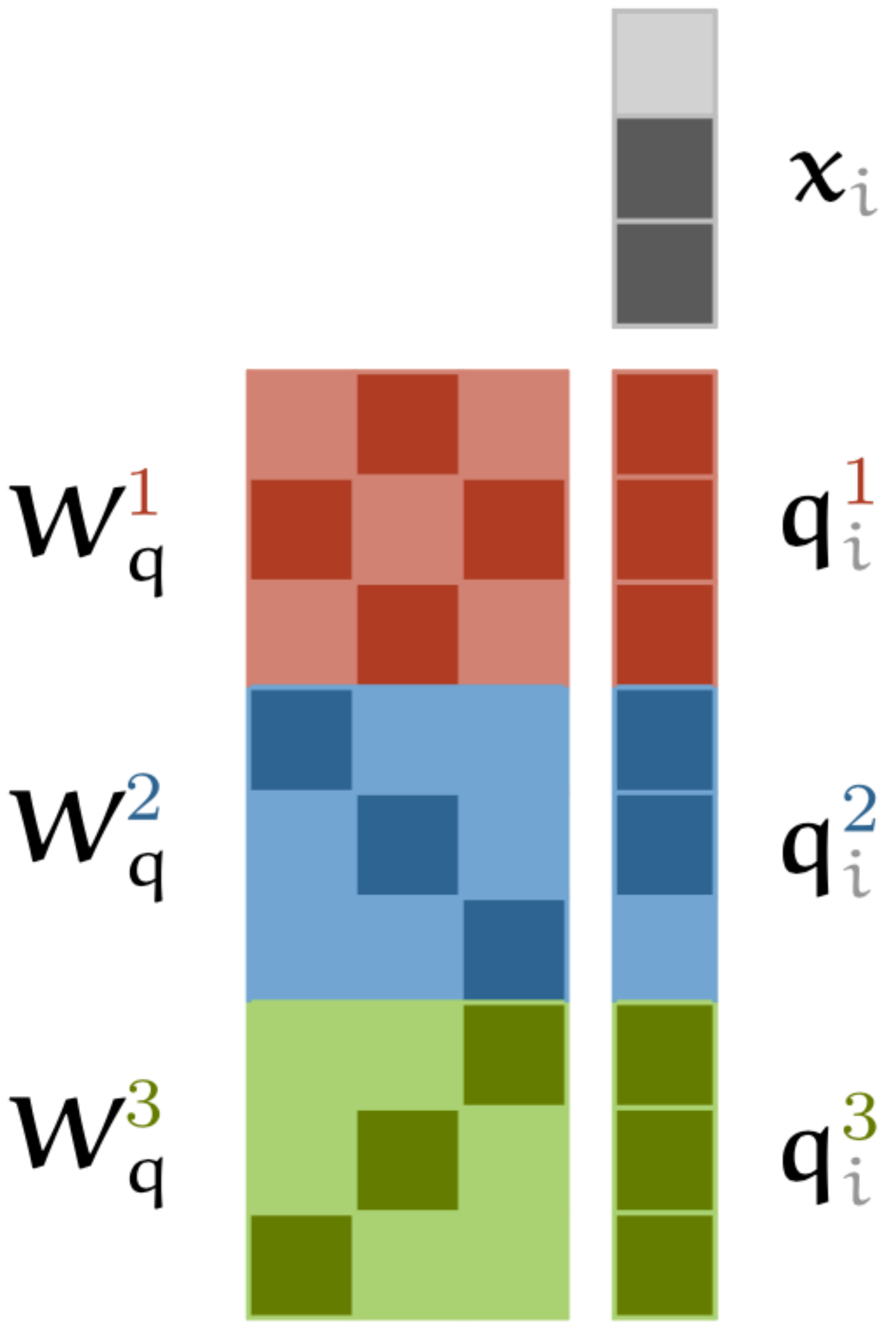
$$w_{ij} = \operatorname{softmax}(w'_{ij})$$

$$\mathbf{y}_i = \sum_j w_{ij} \mathbf{v}_j \,.$$

http://www.peterbloem.nl/blog/transformers

Multi-head attention

- We can allow different ways of transforming into queries, keys, and values to be learned
- Simply means learning different sets of W_q, W_k, and W_v matrices simultaneously.
  - (Actually implemented as a single matrix, anyway.)



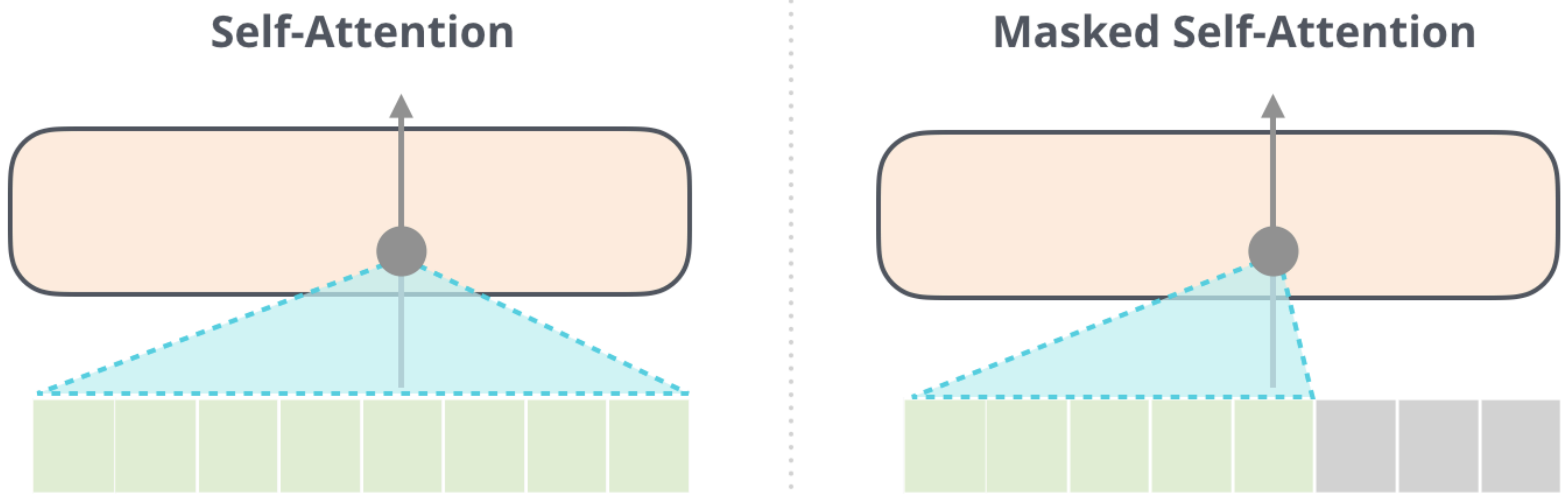
3-headed attention

http://www.peterbloem.nl/blog/transformers

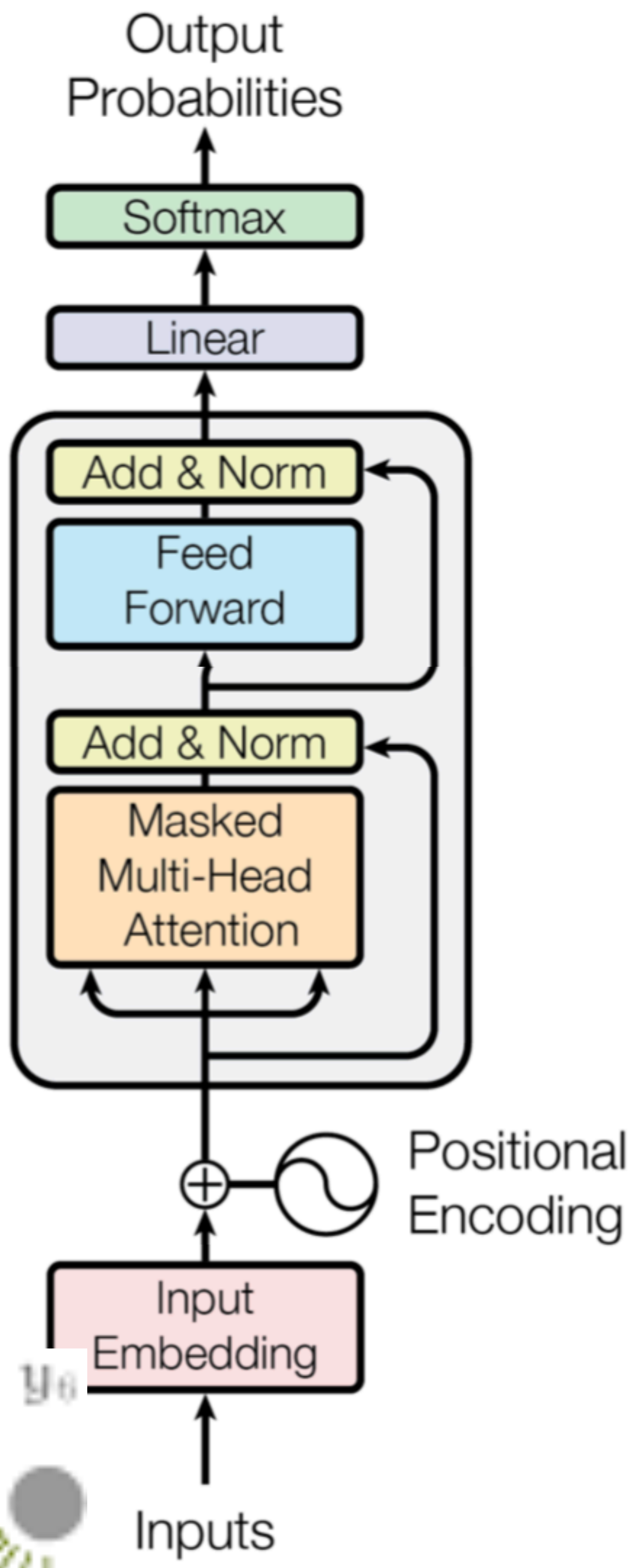
Masking attention

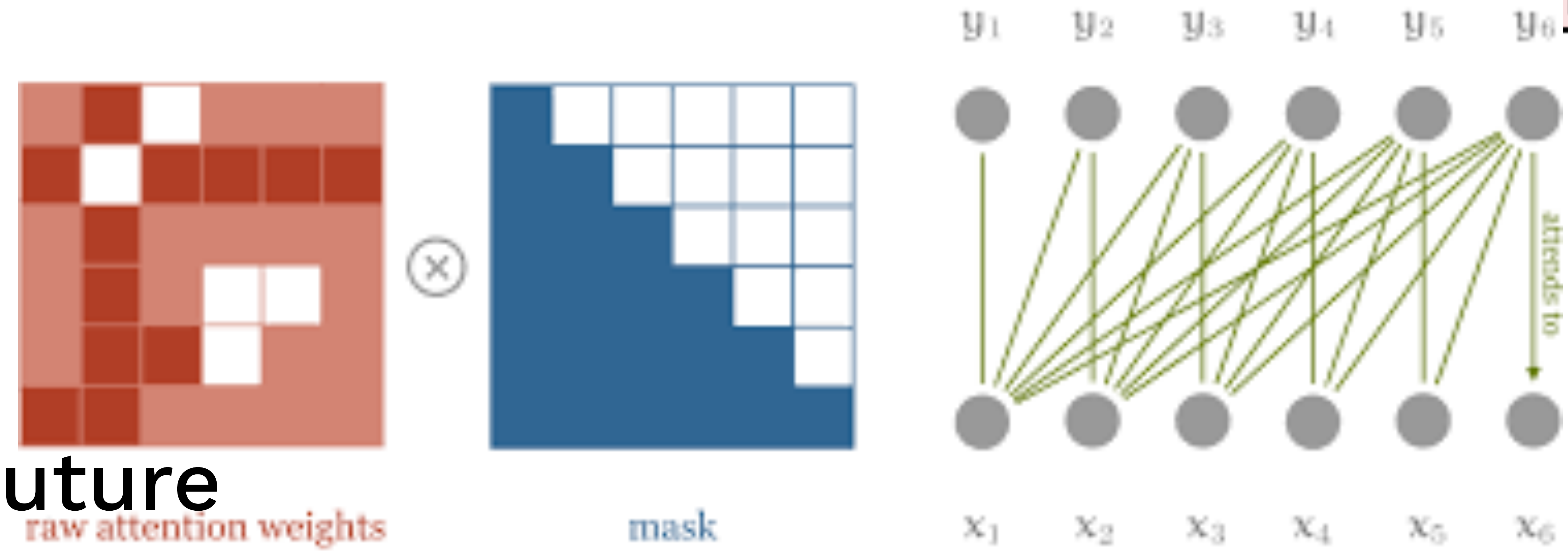
In training:

- Inputs: a sequence of N tokens
  - [It's, a, blue, <BLANK>, <BLANK>, .
- Ground-truth Outputs: a sequence of N tokens
  - [a, blue, sundress, <BLANK>, <BLANK>, ...]
- Actual Outputs: Instead of words, vectors of probabilities over the vocabulary
  - **Crucially: all output probabilities are computed at the same time!**



https://jalammar.github.io/illustrated-



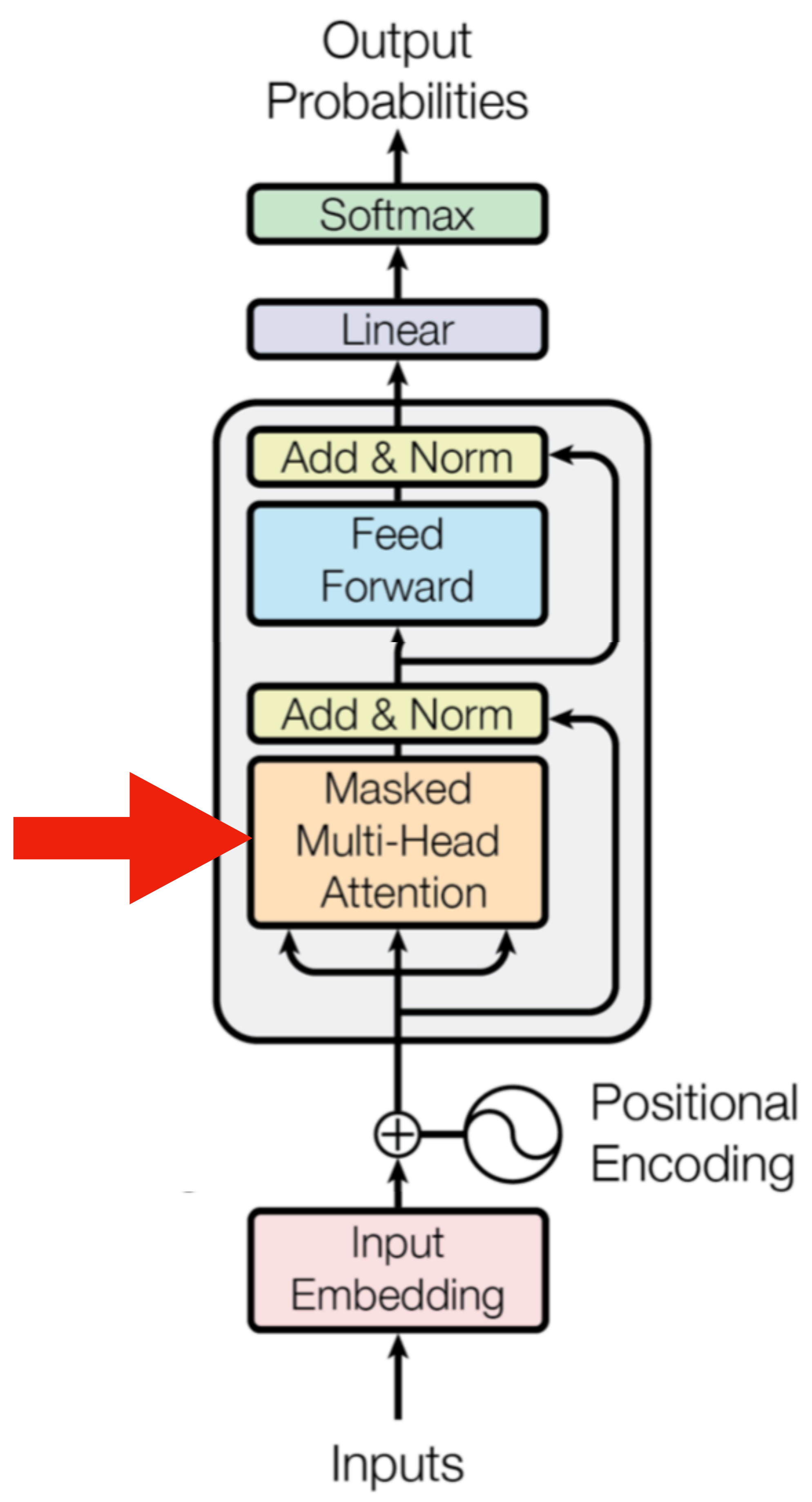


Note how you shouldn't see future tokens when predicting

http://www.peterbloem.nl/blog/transformers

Masked Multi-Head Attention

- Conceptual view:
  - token comes in
  - gets "augmented" with previously-seen tokens that seem relevant (masked self-attention)
  - this happens in several different ways simultaneously (multiple heads)
- NOTE: there's no notion of "position" so far!



Positional Encoding

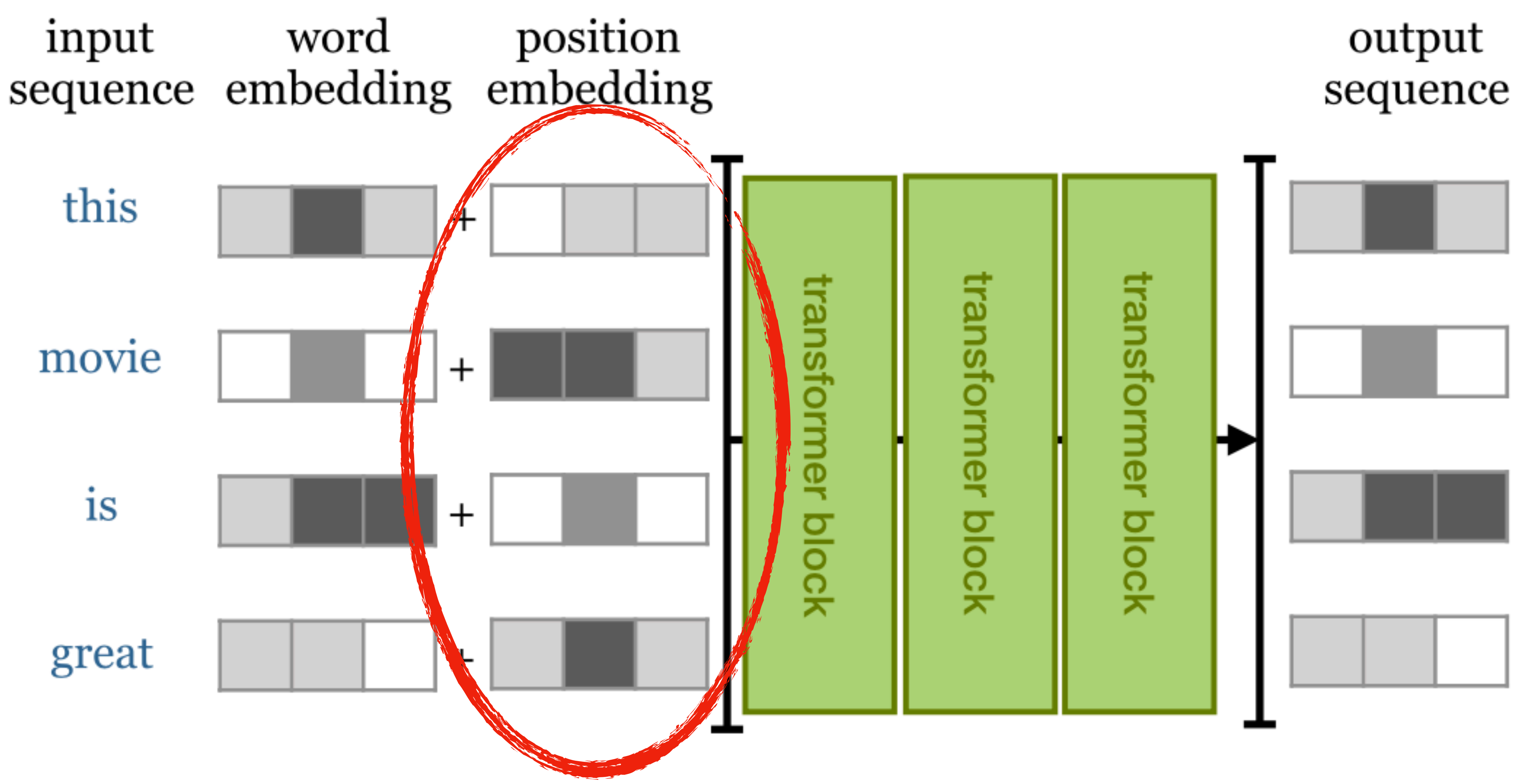
$$q_i = W_q x_i \qquad k_i = W_k x_i \qquad v_i = W_v x_i$$

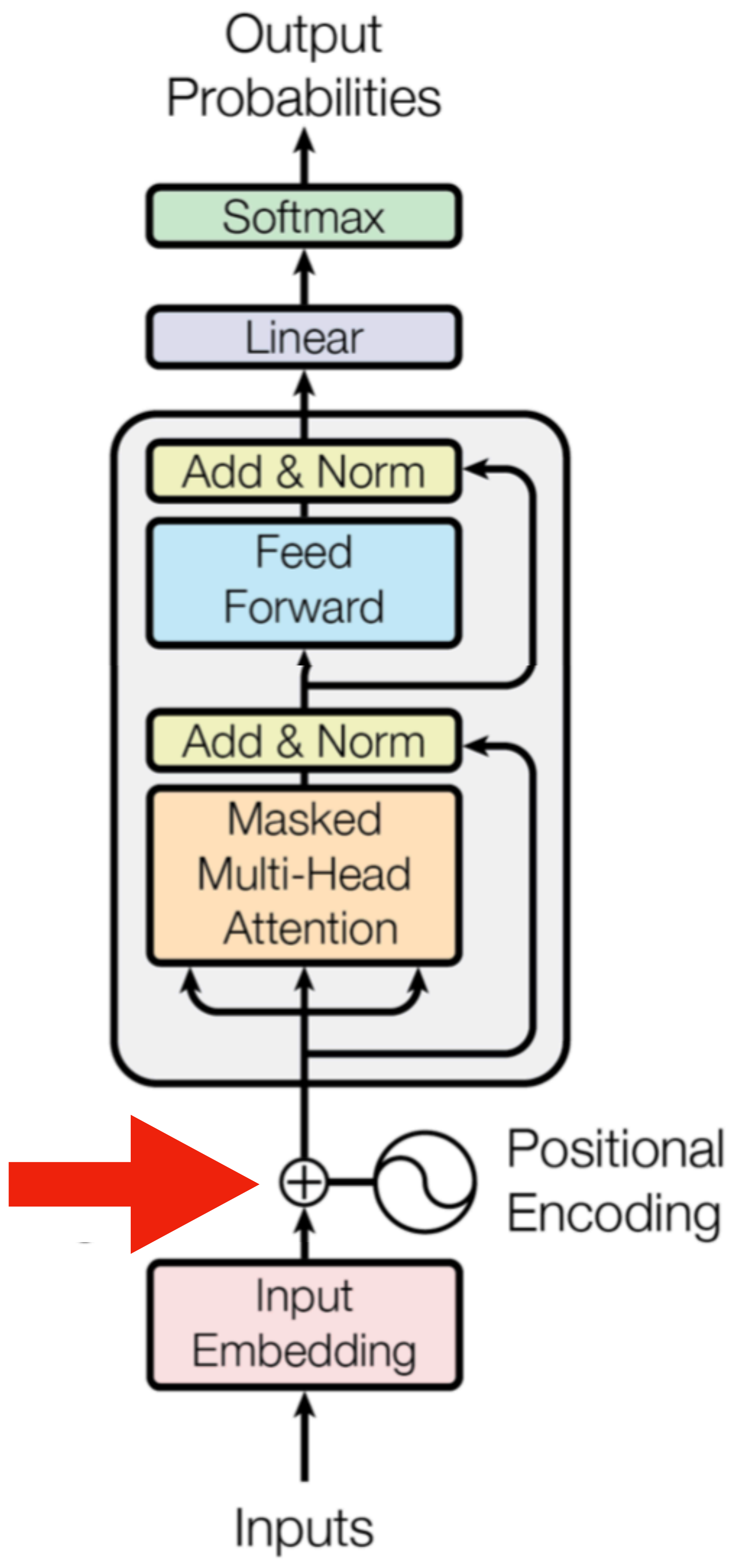
$$w'_{ij} = {q_i}^\mathsf{T} k_j$$

$$w_{ij} = \text{softmax}(w'_{ij})$$

$$y_i = \sum_j w_{ij} v_j \,.$$

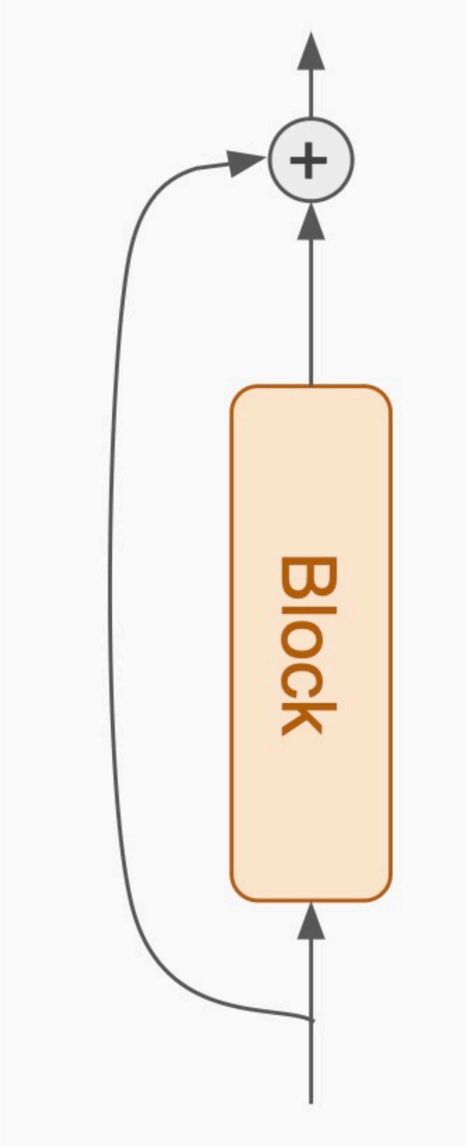
- Attention is totally position-invariant!
  - e.g. [this, movie, is, great] is the same as [movie, this, great, is]
- So, let's add position-encoding vectors to embedding vectors
  - It really is that simple

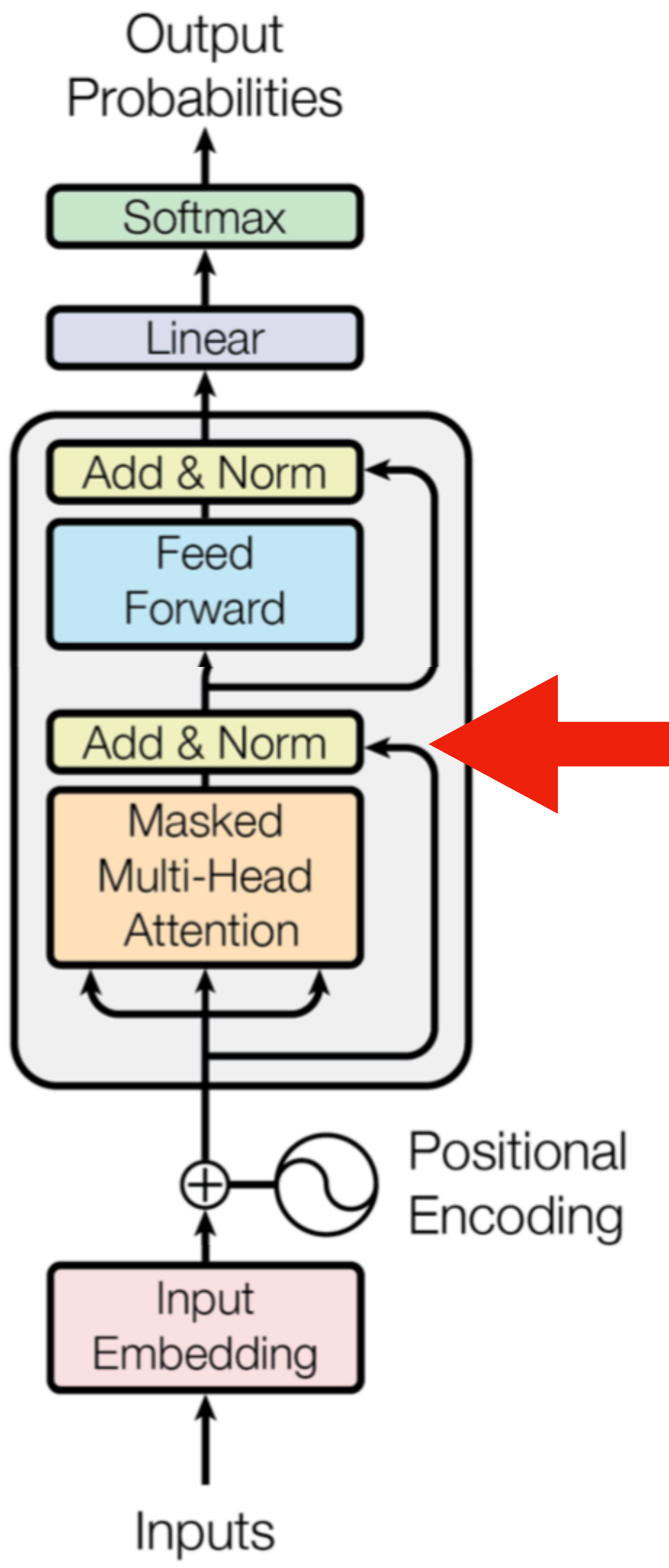




Add

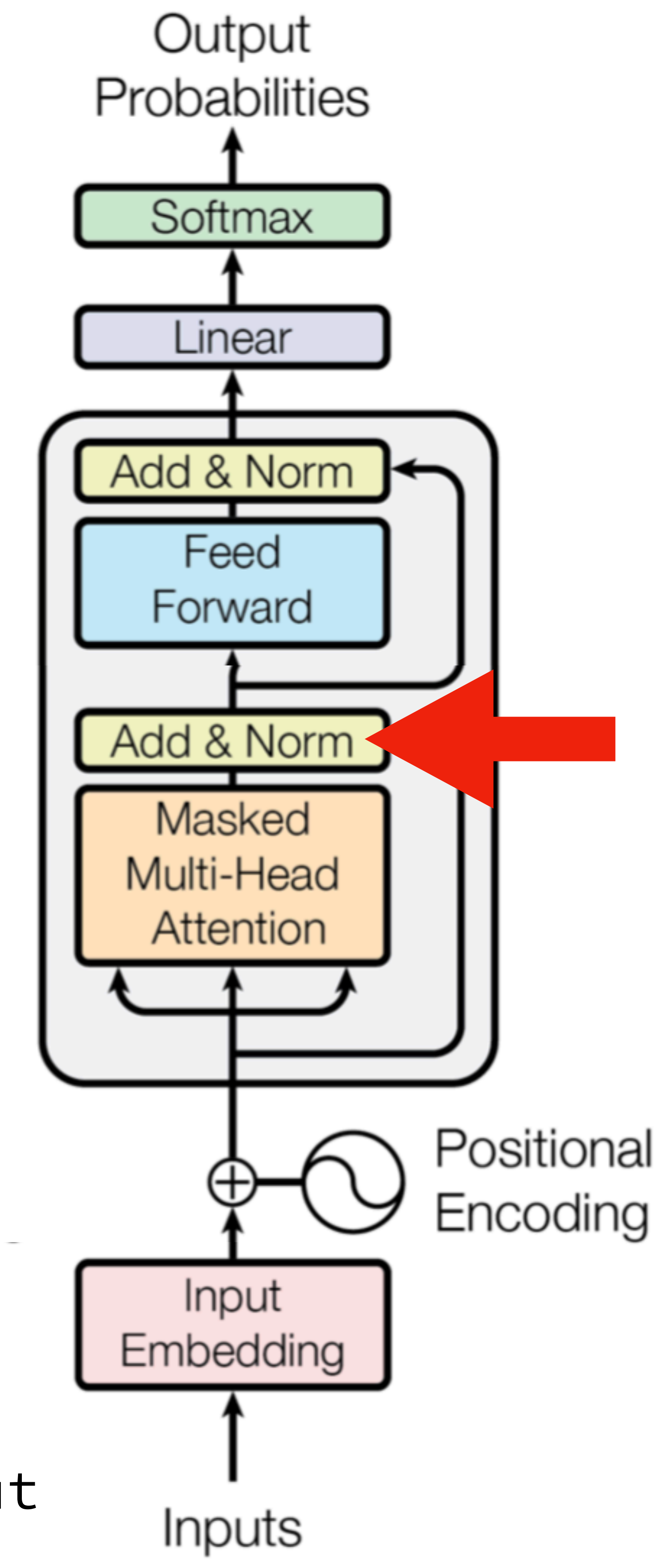
- "Skip connections" aka "residual blocks"
  - output = module(input) + input
  - Allows gradient to flow from the loss function all the way to the first layer
  - (Possible because each module's output is the same shape as its input)

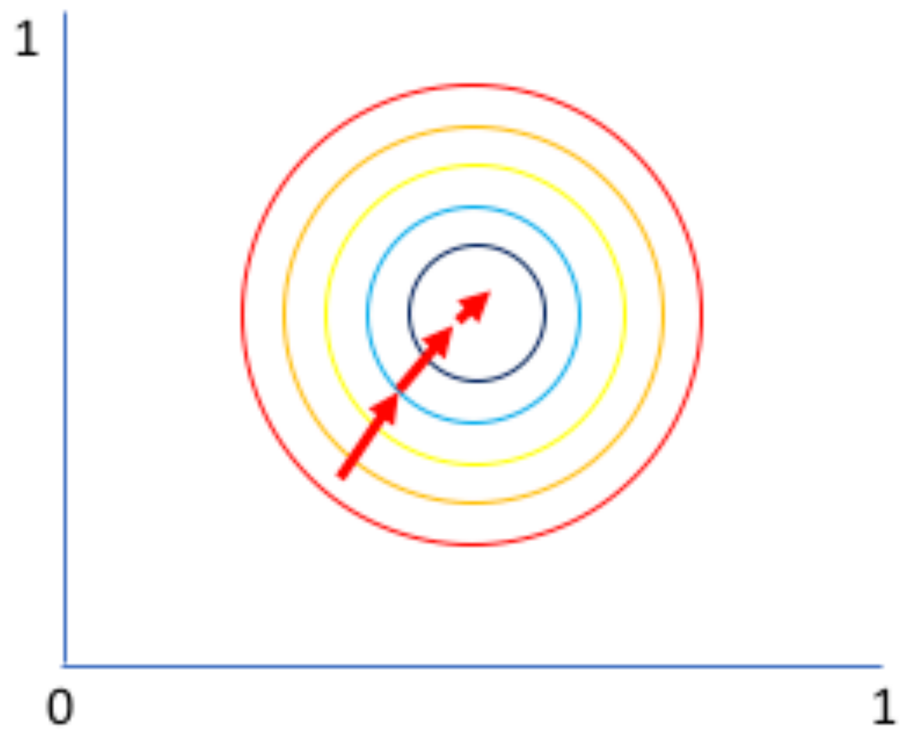




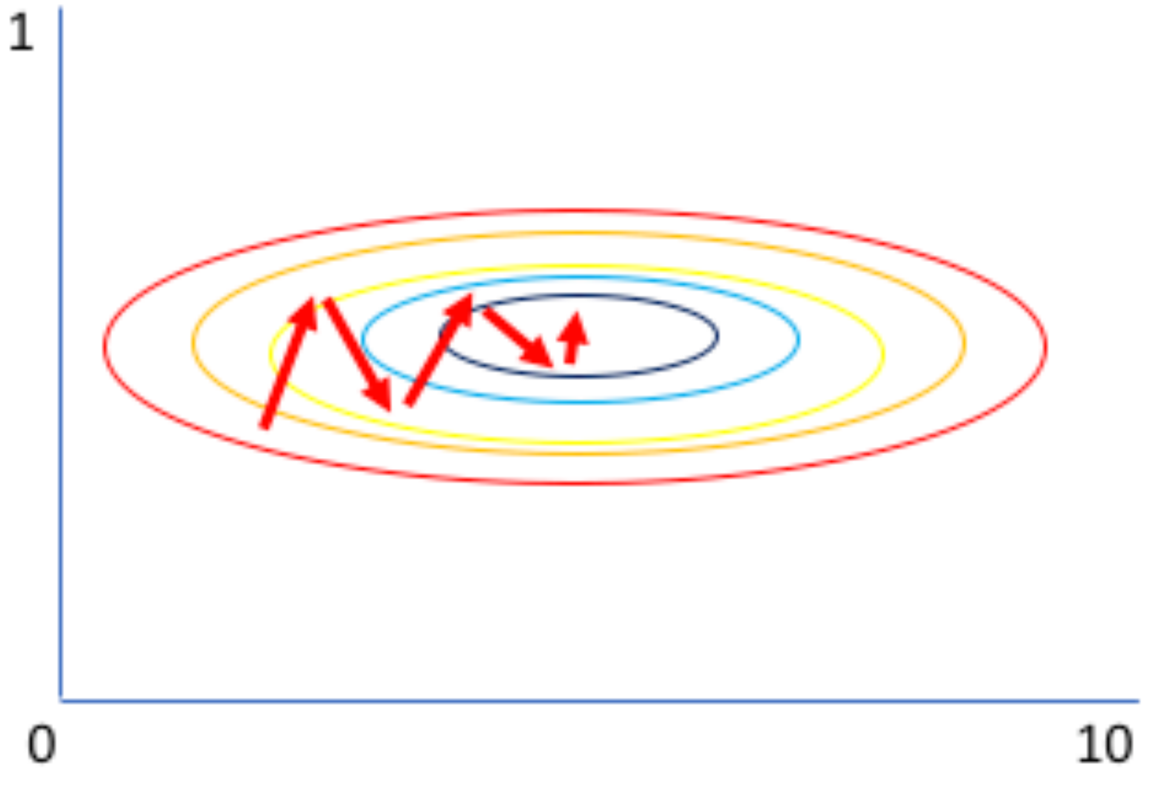
Layer Normalization

- Neural net modules perform best when input vectors have uniform mean and std in each dimension.
- As inputs flow through the network, means and std's get blown out.
- Layer Normalization is a hack to reset things to where we want them in between layers.





Both parameters can be updated in equal proportions



Gradient of larger parameter dominates the update

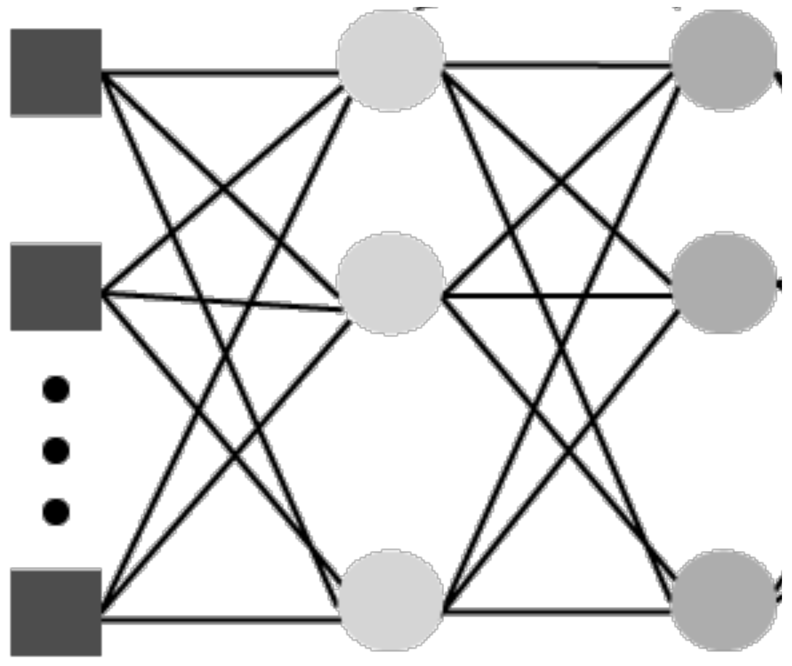
```
output = module(layer_norm(input)) + input
```

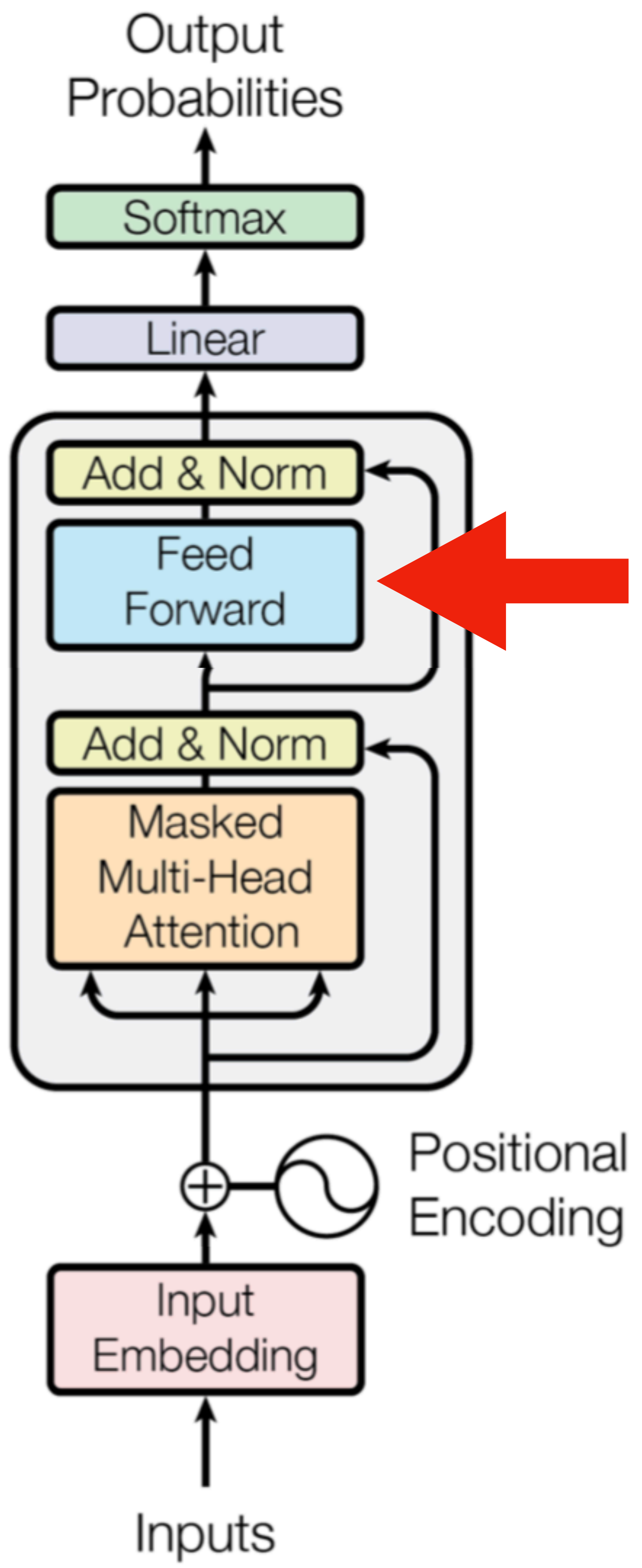
$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}}$$

https://arxiv.org/pdf/1803.08494.pdf

Feed Forward Layer

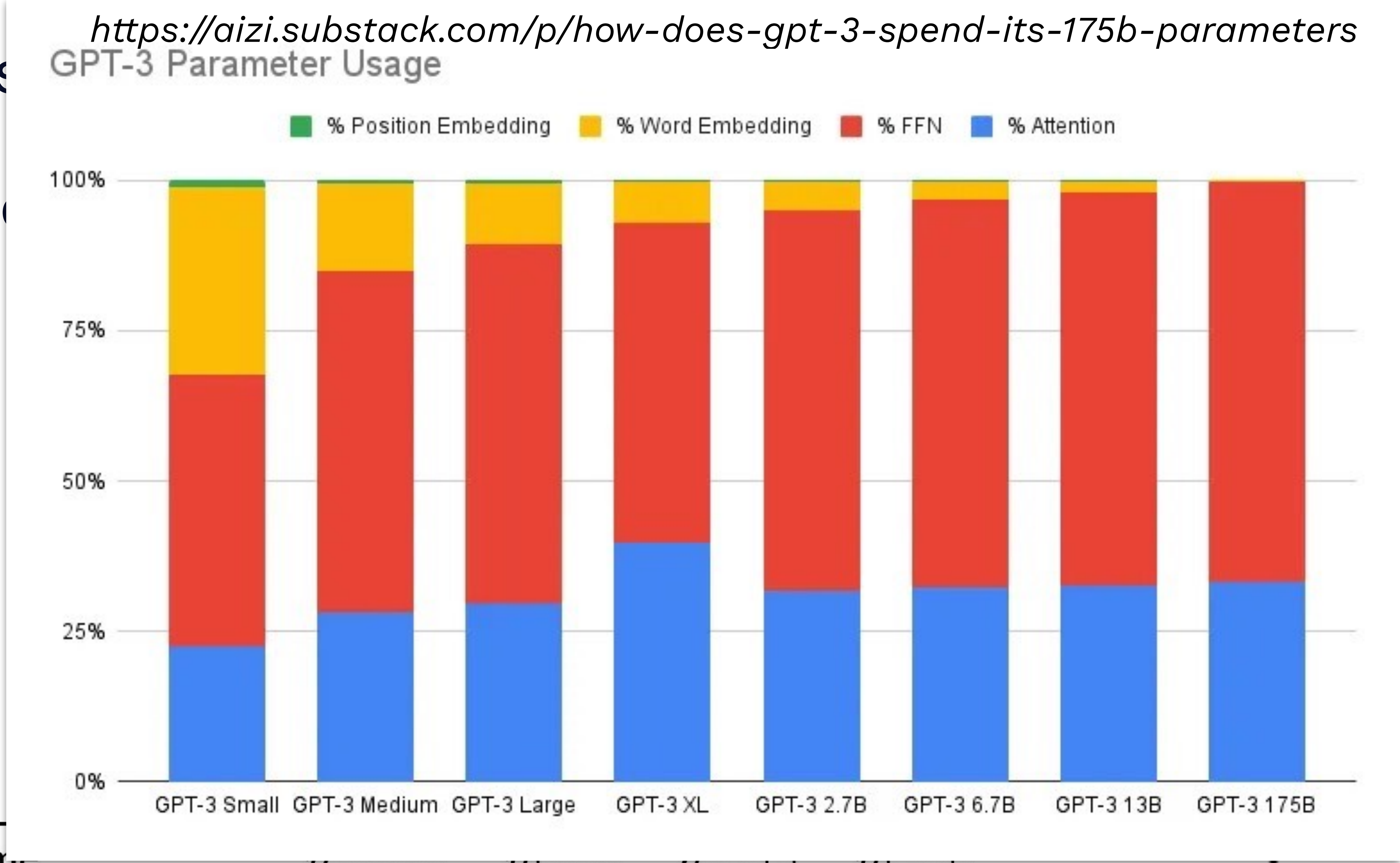
- Standard Multi-Layer Perceptron with one hidden layer
- Defined $y = W_2(GeLU(W_1 x + b_1)) + b_2$
- Conceptual view:
  - token (augmented with other relevant tokens that it has seen) comes in...
  - ...and "upgrades" its representation



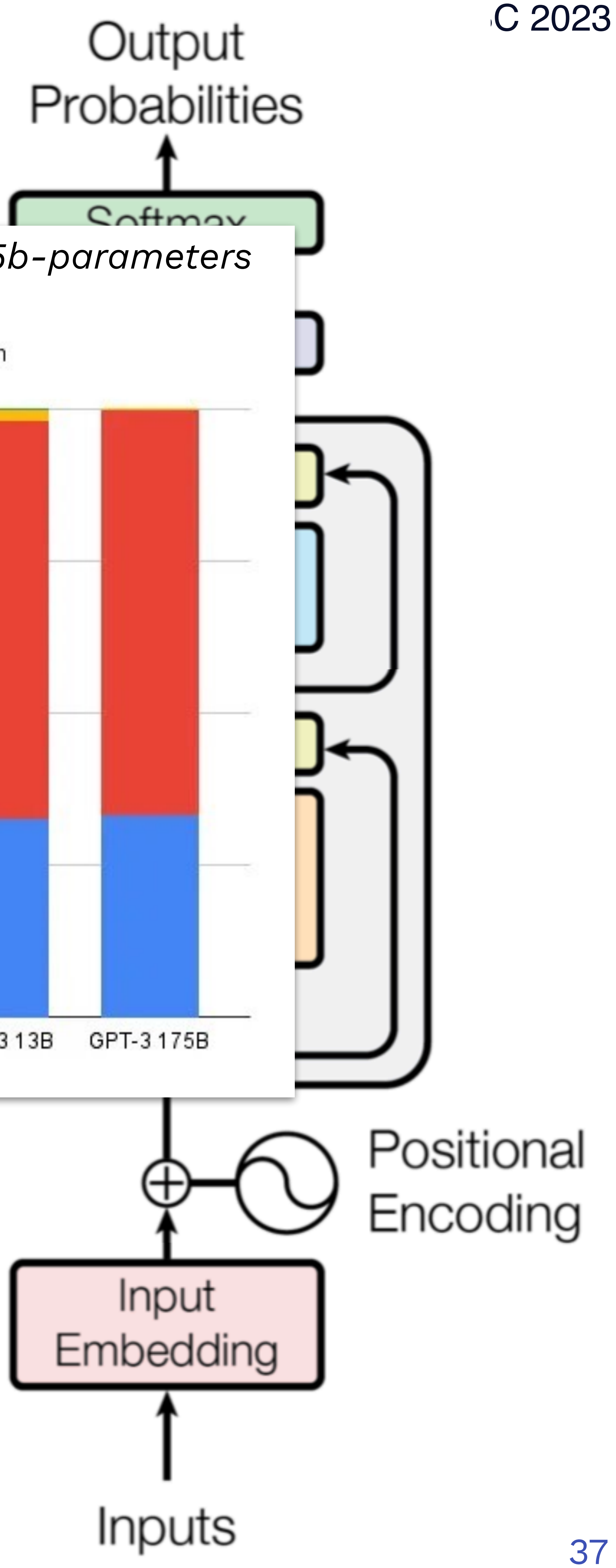


Transformer Architecture

- The main Transformer Layer is s
- The overall hyperparameters ar
  - Number of layers
  - Embedding dimension
  - Number of attention heads
- The largest models are ~70% feed-forward weights



Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}
GPT-3 Small	125M	12	768	12
GPT-3 Medium	350M	24	1024	16
GPT-3 Large	760M	24	1536	16
GPT-3 XL	1.3B	24	2048	24
GPT-3 2.7B	2.7B	32	2560	32
GPT-3 6.7B	6.7B	32	4096	32
GPT-3 13B	13.0B	40	5140	40
GPT-3 175B or "GPT-3"	175.0B	96	12288	96



Why does this work so well?

Andrej Karpathy
@karpathy

The Transformer is a magnificient neural network architecture because it is a general-purpose differentiable computer. It is simultaneously:
1) expressive (in the forward pass)
2) optimizable (via backpropagation+gradient descent)
3) efficient (high parallelism compute graph)

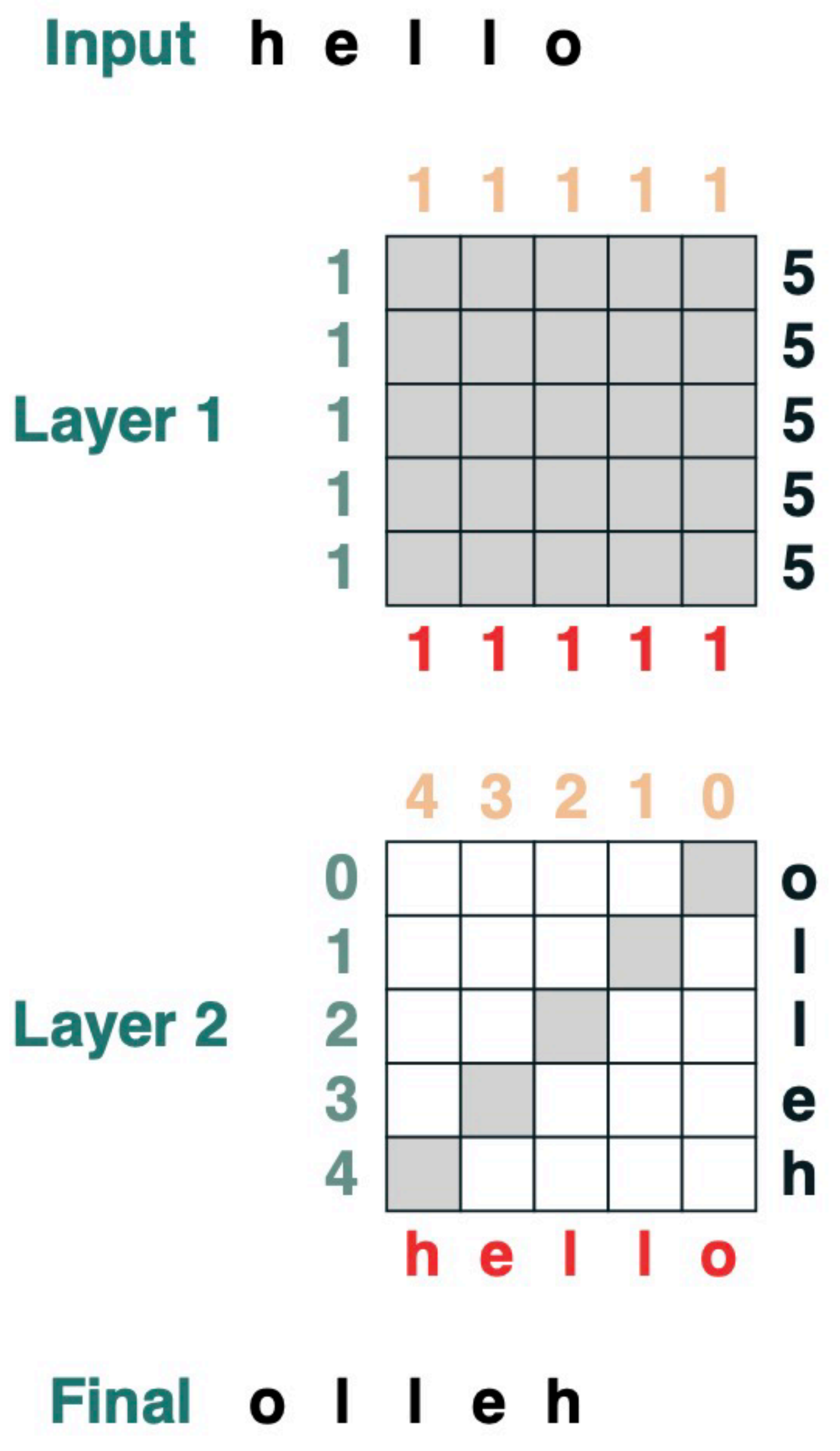
11:54 AM · Oct 19, 2022

490 Retweets **39** Quotes **3,670** Likes **1,023** Bookmarks

Thinking like Transformers

```
def flip():
    length = (key(1) == query(1)).value(1)
    flip = (key(length - indices - 1) == query(indices)).value(tokens)
    return flip
flip()
```

- Restricted Access Sequence Processing (RASP, 2021): programming language of Transformer-implementable operations

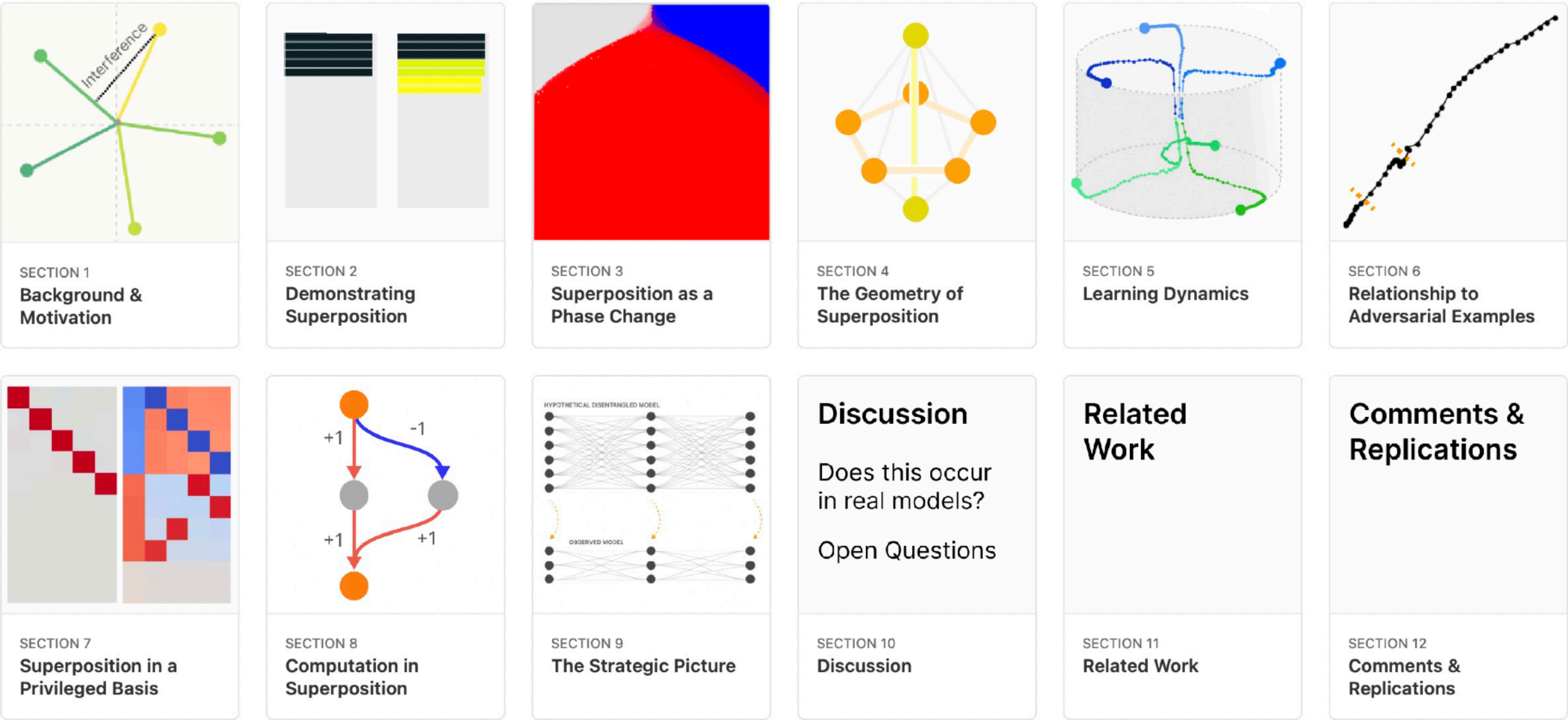


https://srush.github.io/raspy/

We mostly don't understand it, though

- Much great work from Anthropic if this has captured your curiosity!

Toy Models of Superposition



AUTHORS

Nelson Elhage*, Tristan Hume*, Catherine Olsson*, Nicholas Schiefer*, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg*, Christopher Olah‡

AFFILIATIONS

Anthropic, Harvard

PUBLISHED

Sept 14, 2022

* Core Research Contributor; ‡ Correspondence to colah@anthropic.com; Author contributions statement below.

In-context Learning and Induction Heads

AUTHORS

Catherine Olsson*, Nelson Elhage*, Neel Nanda*, Nicholas Joseph†, Nova DasSarma†, Tom Henighan†, Ben Mann†, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, Chris Olah‡

AFFILIATION

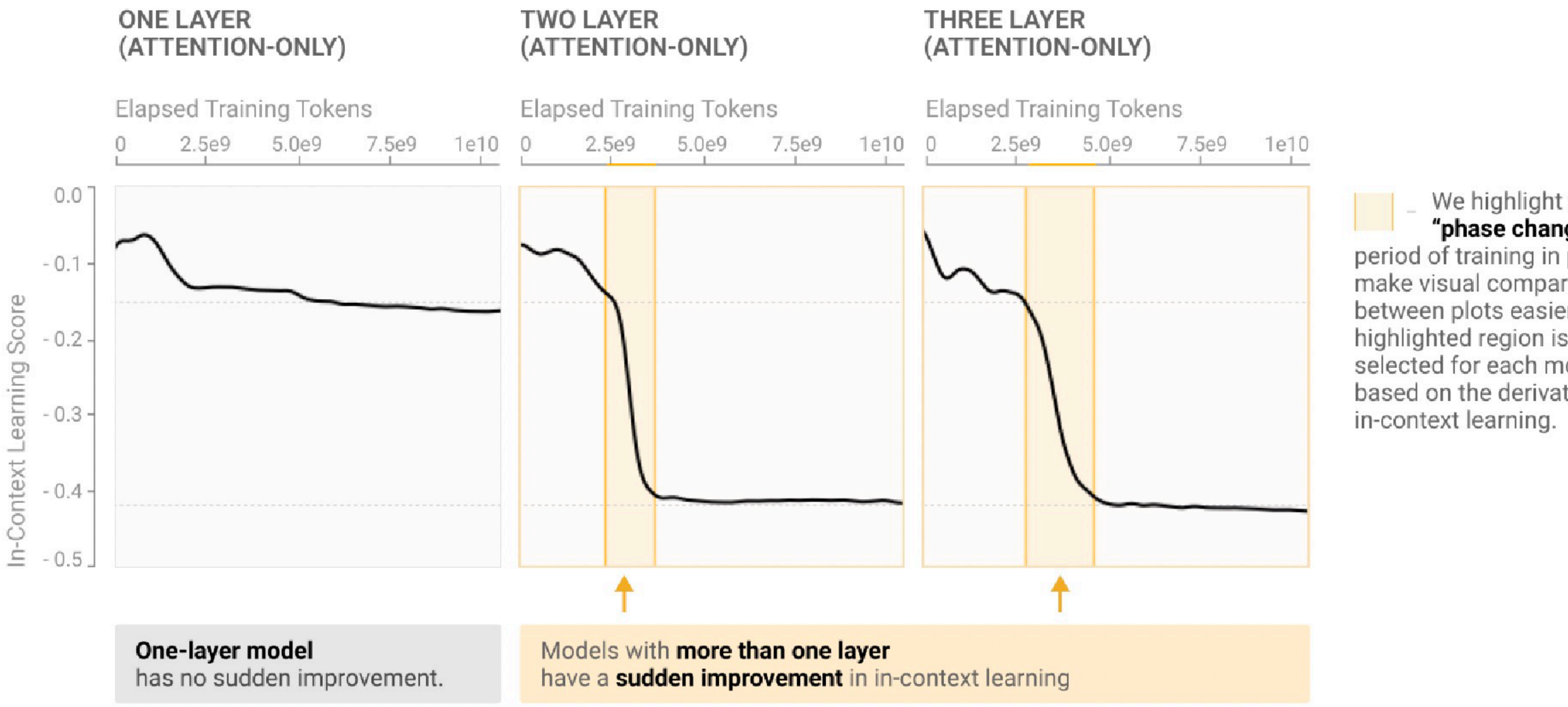
Anthropic

PUBLISHED

Mar 8, 2022

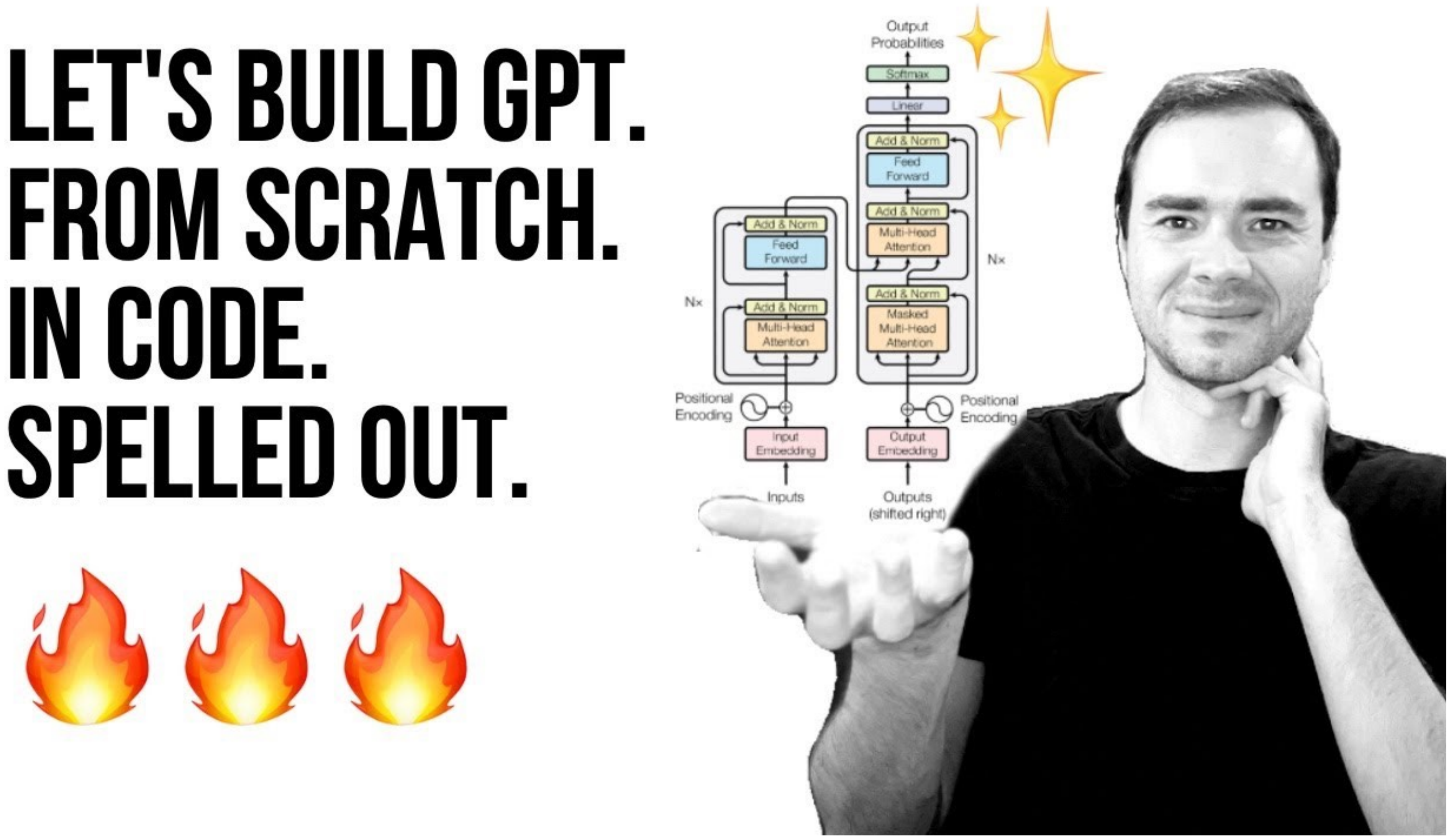
* Core Research Contributor; † Core Infrastructure Contributor; ‡ Correspondence to colah@anthropic.com; Author contributions statement below.

MODELS WITH MORE THAN ONE LAYER HAVE AN ABRUPT IMPROVEMENT IN IN-CONTEXT LEARNING



Should you be able to code a Transformer?

- Definitely not necessary!
- BUT: it's not difficult, it is fun, and is probably worth doing
- Andrej Karpathy's GPT-2 implementation is <400 lines of code, including Attention and MLP blocks



https://www.youtube.com/watch?v=kCc8FmEb1nY

Resources

- Lucas Beyer's Lecture on Transformers
- Peter Bloem's "Transformers from Scratch"
- Nelson Elhage's "Transformers for Software Engineers" for a different view
- Andrej Karpathy's entire Neural Networks: Zero to Hero video series
- Lillian Weng's "The Transformer Family v2" megapost

Questions?

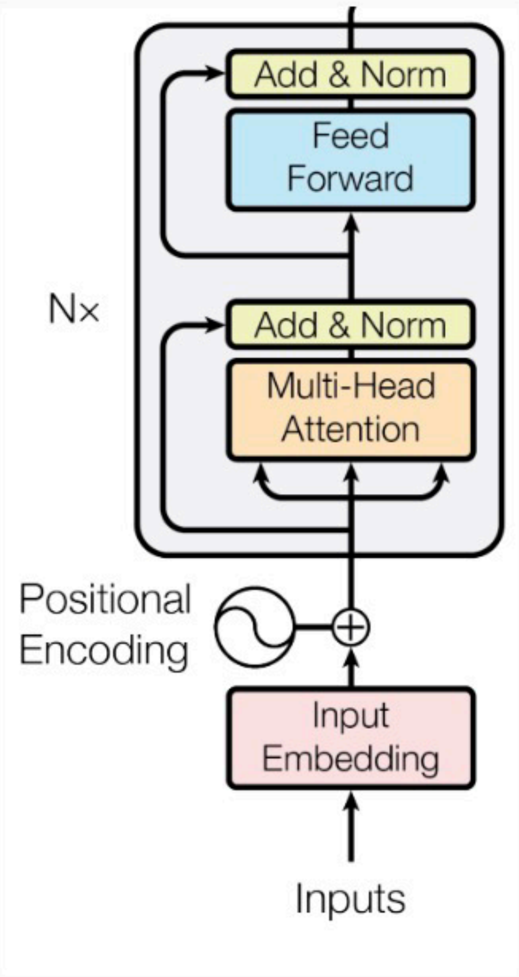
03

Notable LLMs

Three Easy Pieces

BERT

[*] [*] **[sat_]** [*] **[the_]** [*]



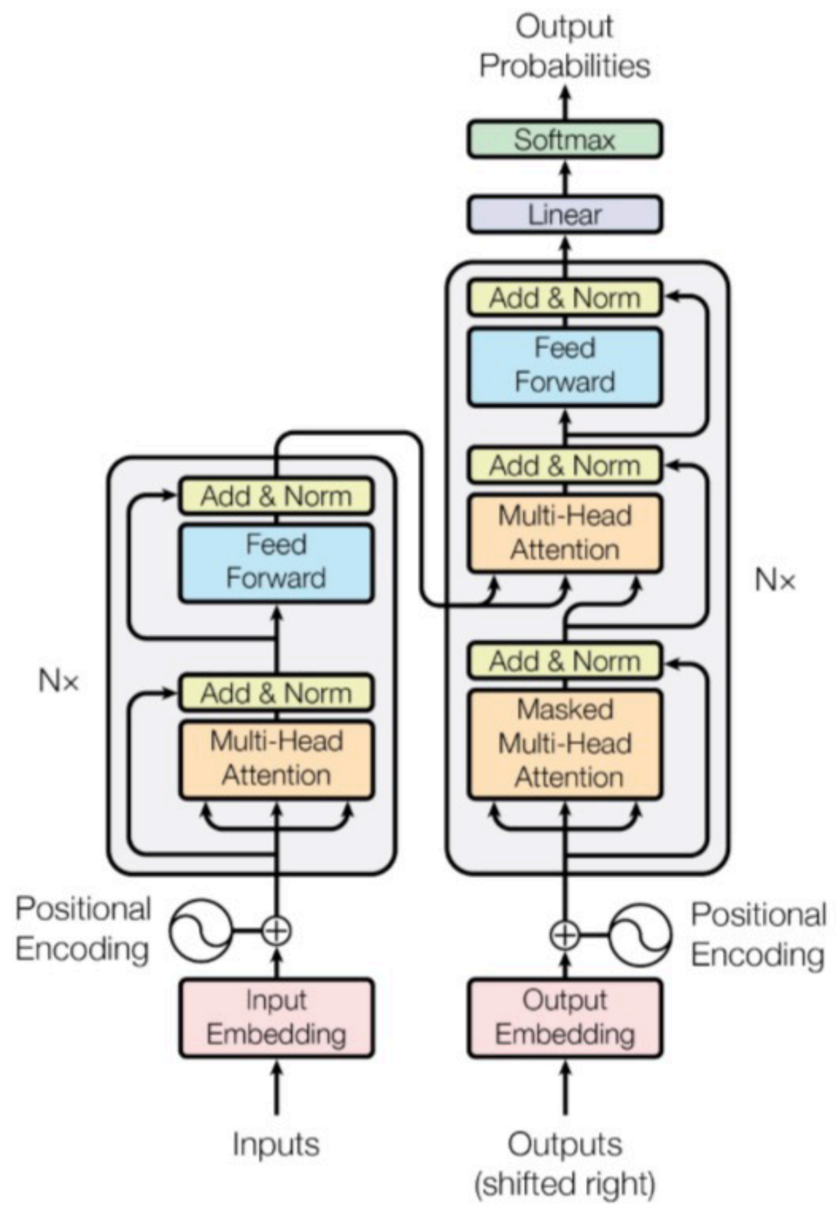
[The_] [cat_] **[MASK]** [on_] **[MASK]** [mat_]

T5

Das ist gut.

A storm in Attala caused 6 victims.

This is not toxic.



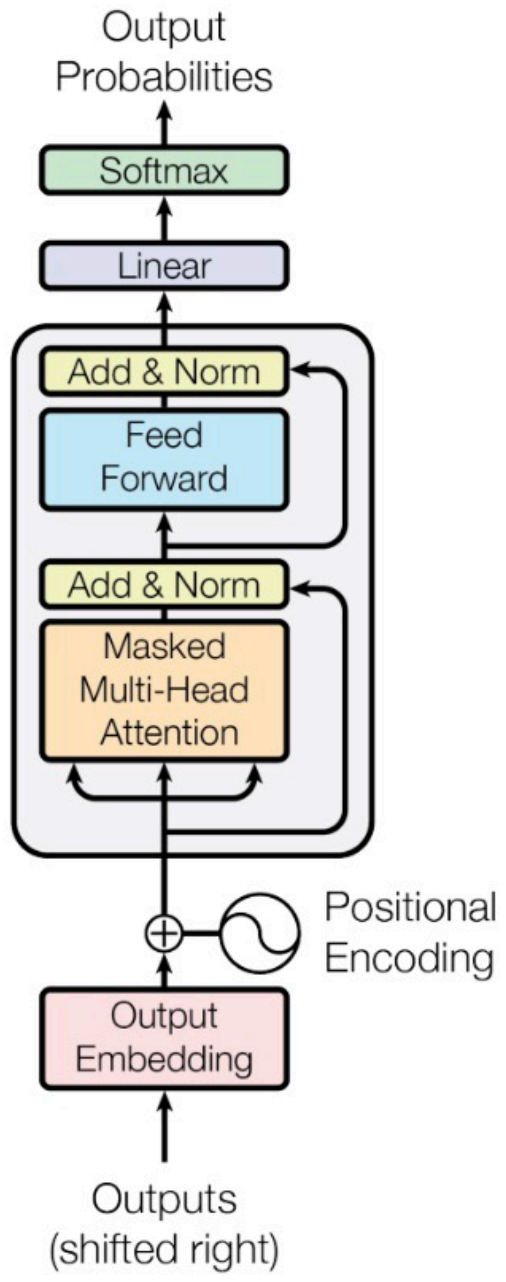
Translate EN-DE: This is good.

Summarize: state authorities dispatched…

Is this toxic: You look beautiful today!

GPT

[sat_]



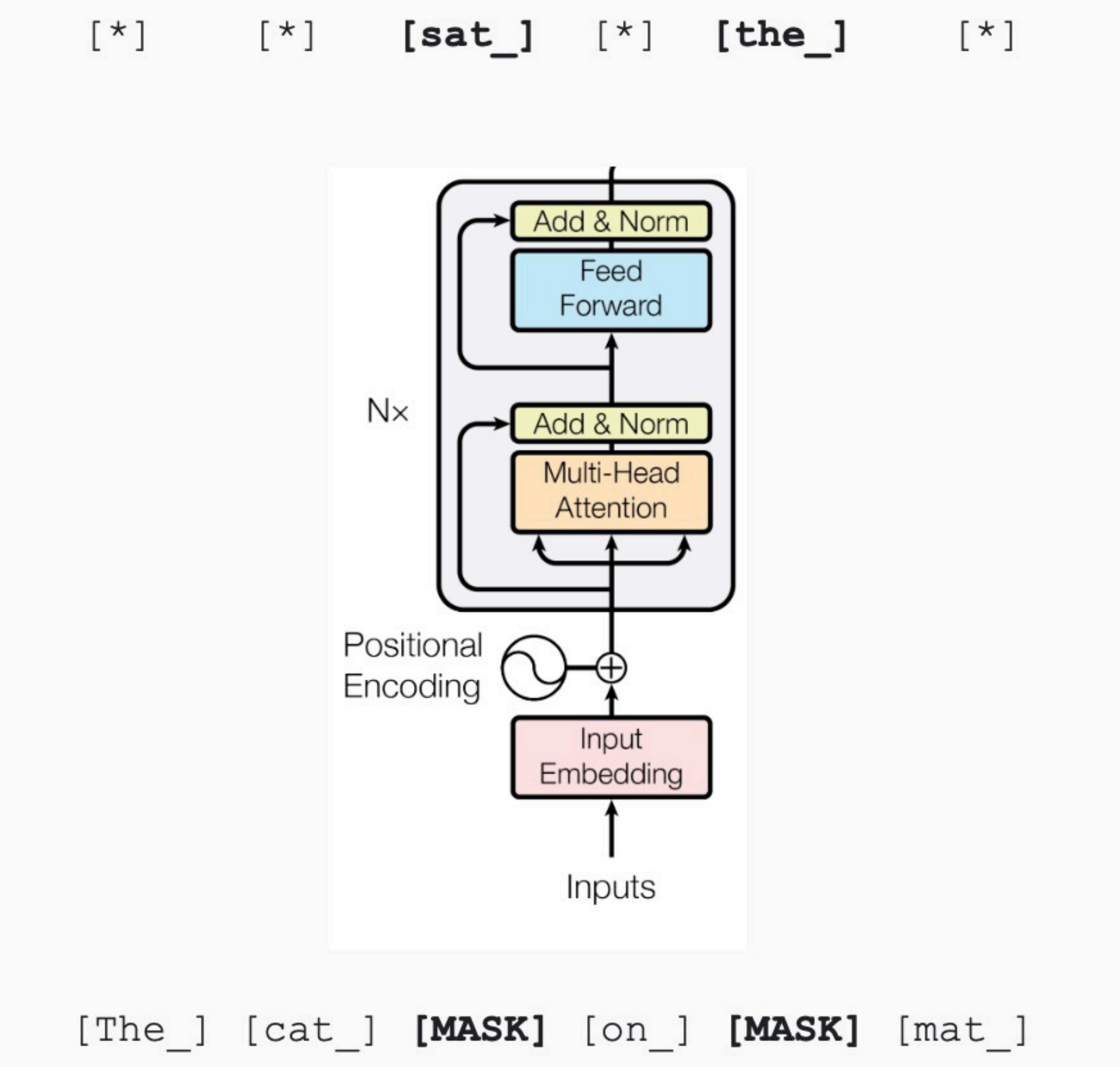
[START] [The_] [cat_]

Transformer image source: "Attention Is All You Need" paper

From Lukas Beyer's lecture Intro to Transformers.

BERT (2019)

- *Bidirectional* Encoder Representations from Transformers
- Encoder-only (no attention masking)
- 110M params
- 15% of all words masked out
- Was great, now dated



From Lukas Beyer's lecture Intro to Transformers.

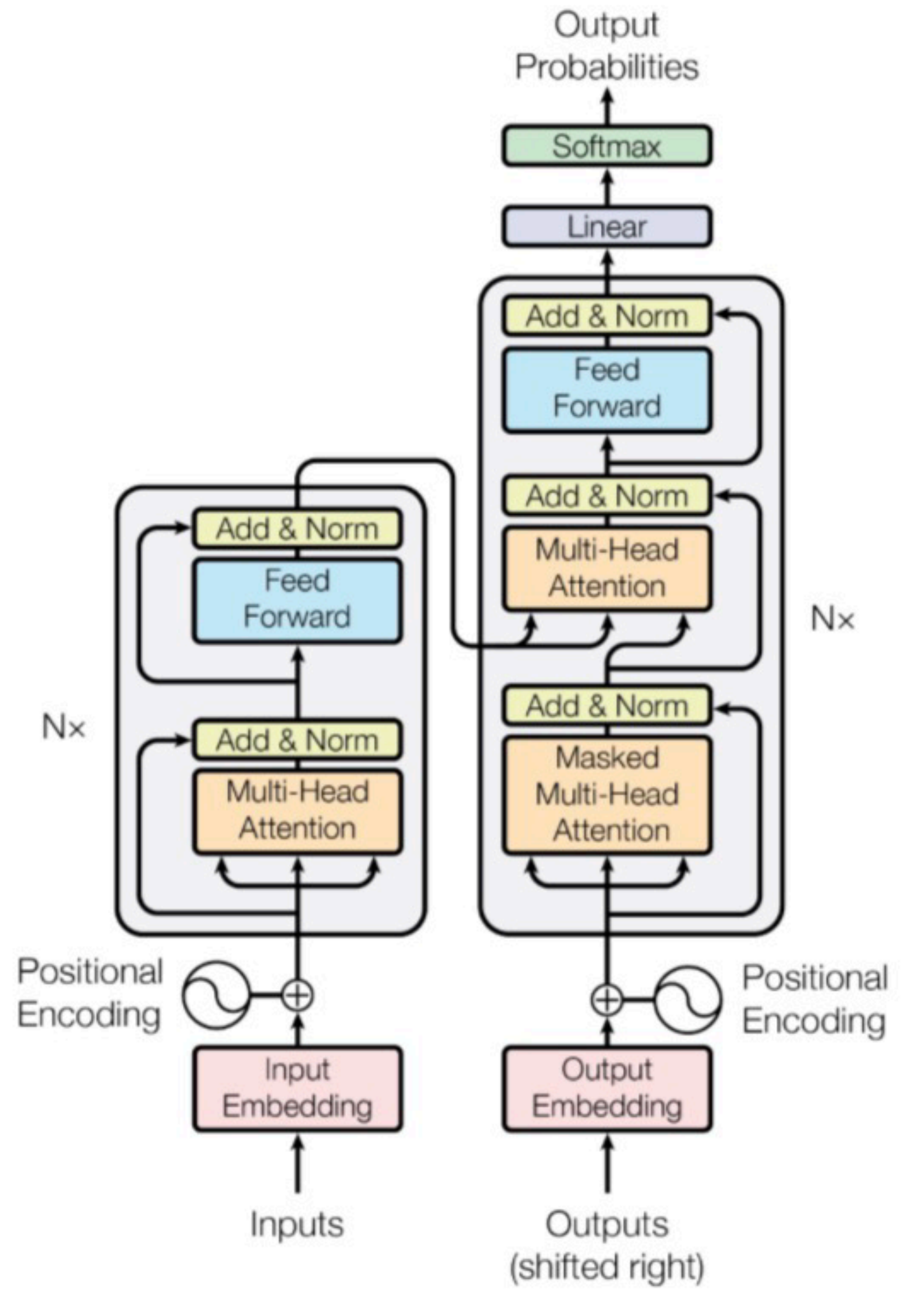
T5: Text-to-Text Transfer Transformer (2020)

- Input and output are both text strings
- Encoder-Decoder architecture
- 11B parameters
- Still could be a good choice for fine-tuning!

`Das ist gut.`

`A storm in Attala caused 6 victims.`

`This is not toxic.`



`Translate EN-DE: This is good.`

`Summarize: state authorities dispatched…`

`Is this toxic: You look beautiful today!`

From Lukas Beyer's lecture Intro to Transformers.

T5 Training Data

- Unsupervised pre-training on Colossal Clean Crawled Corpus (C4)
  - Start with Common Crawl (over 50TB of compressed data, 10B+ web pages)
  - Filtered down to ~800GB, or ~160B tokens
- Also trained on academic supervised tasks

- We discarded any page with fewer than 5 sentences and only retained lines that contained at least 3 words.
- We removed any page that contained any word on the "List of Dirty, Naughty, Obscene or Otherwise Bad Words".[6]
- Some pages inadvertently contained code. Since the curly bracket "{" appears in many programming languages (such as Javascript, widely used on the web) but not in natural text, we removed any pages that contained a curly bracket.
- To deduplicate the data set, we discarded all but one of any three-sentence span occurring more than once in the data set.

https://paperswithcode.com/dataset/c4

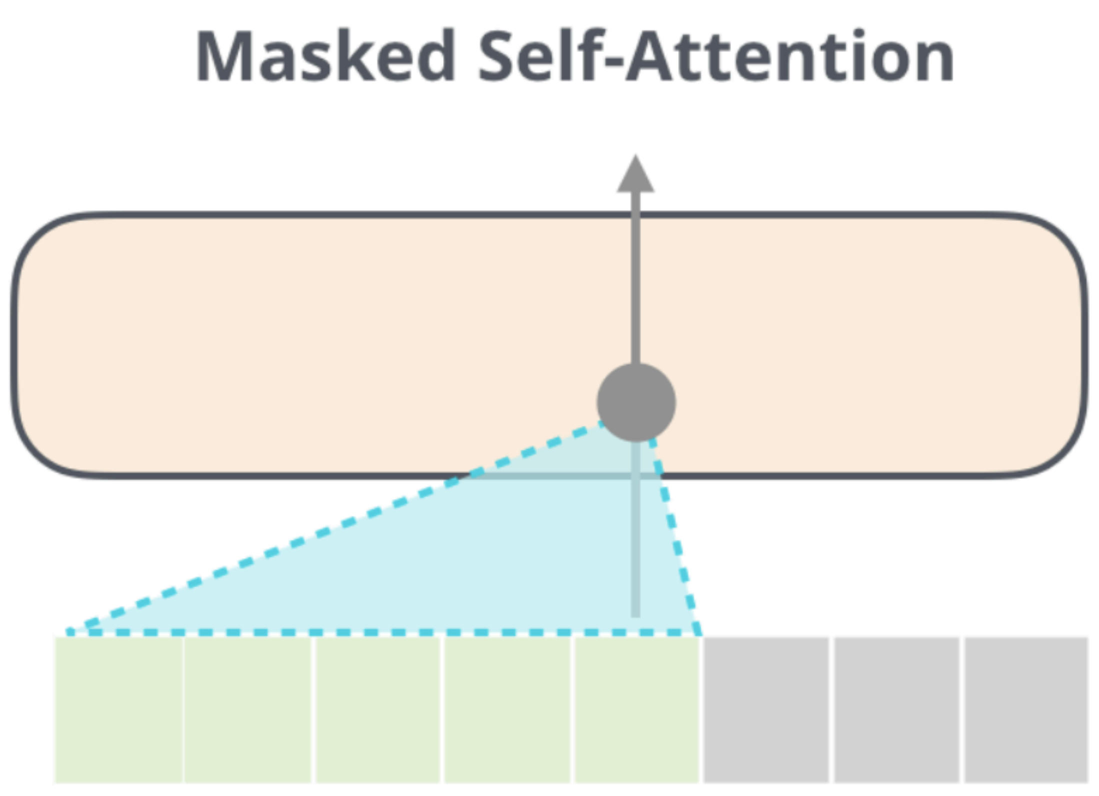
2. **Datasets used for Supervised text-to-text language modeling objective**

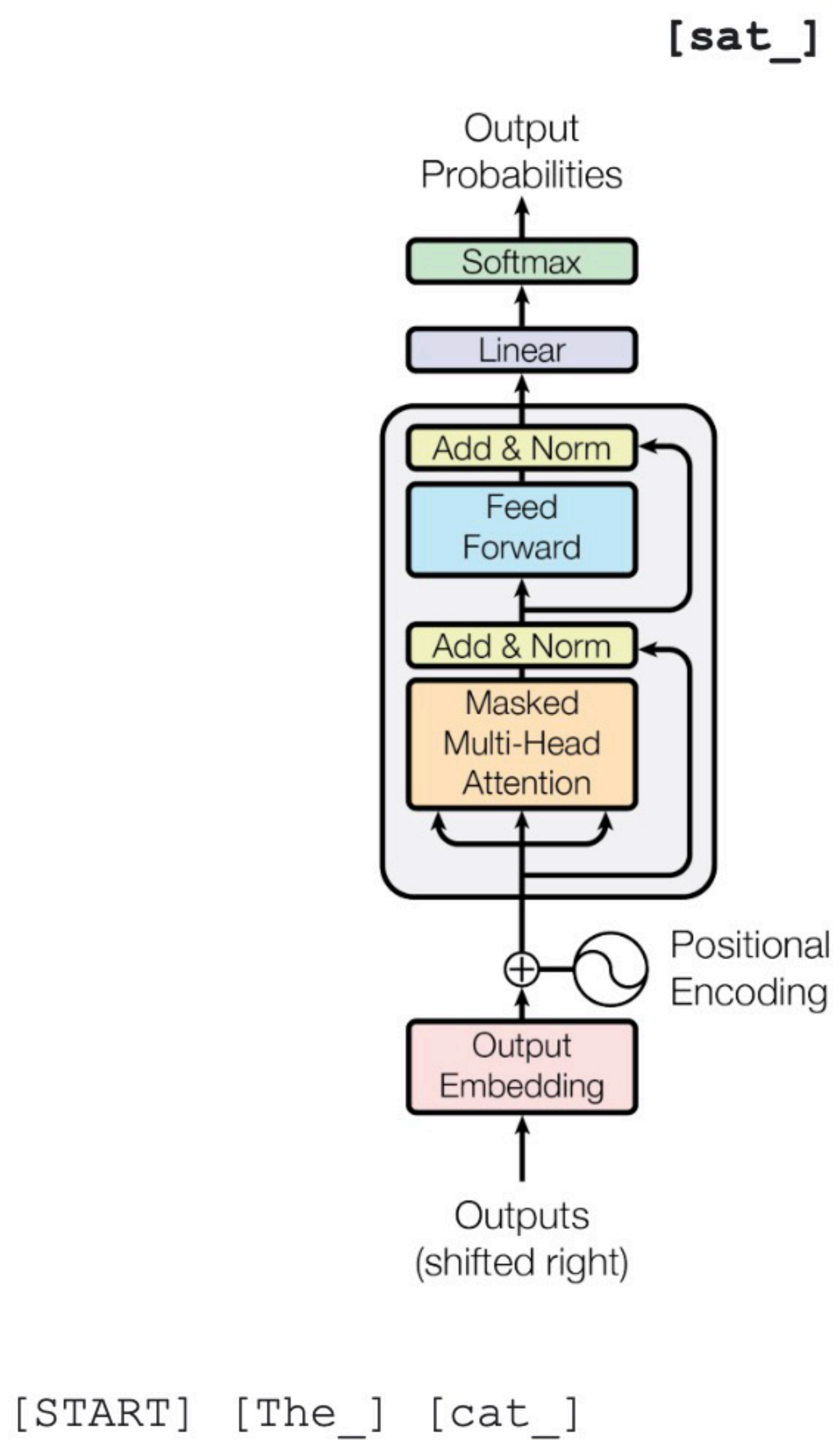
- Sentence acceptability judgment
  - CoLA Warstadt et al., 2018
- Sentiment analysis
  - SST-2 Socher et al., 2013
- Paraphrasing/sentence similarity
  - MRPC Dolan and Brockett, 2005
  - STS-B Ceret al., 2017
  - QQP Iyer et al., 2017
- Natural language inference
  - MNLI Williams et al., 2017
  - QNLI Rajpurkar et al.,2016
  - RTE Dagan et al., 2005
  - CB De Marneff et al., 2019
- Sentence completion
  - COPA Roemmele et al., 2011
- Word sense disambiguation
  - WIC Pilehvar and Camacho-Collados, 2018
- Question answering
  - MultiRC Khashabi et al., 2018
  - ReCoRD Zhang et al., 2018
  - BoolQ Clark et al., 2019

https://stanford-cs324.github.io/winter2022/lectures/data/

GPT / GPT-2 (2019)

- Generative Pre-trained Transformer
- Decoder-only (uses masked self-attention)
- Largest model is 1.5B





From Lukas Beyer's lecture Intro to Transformers.

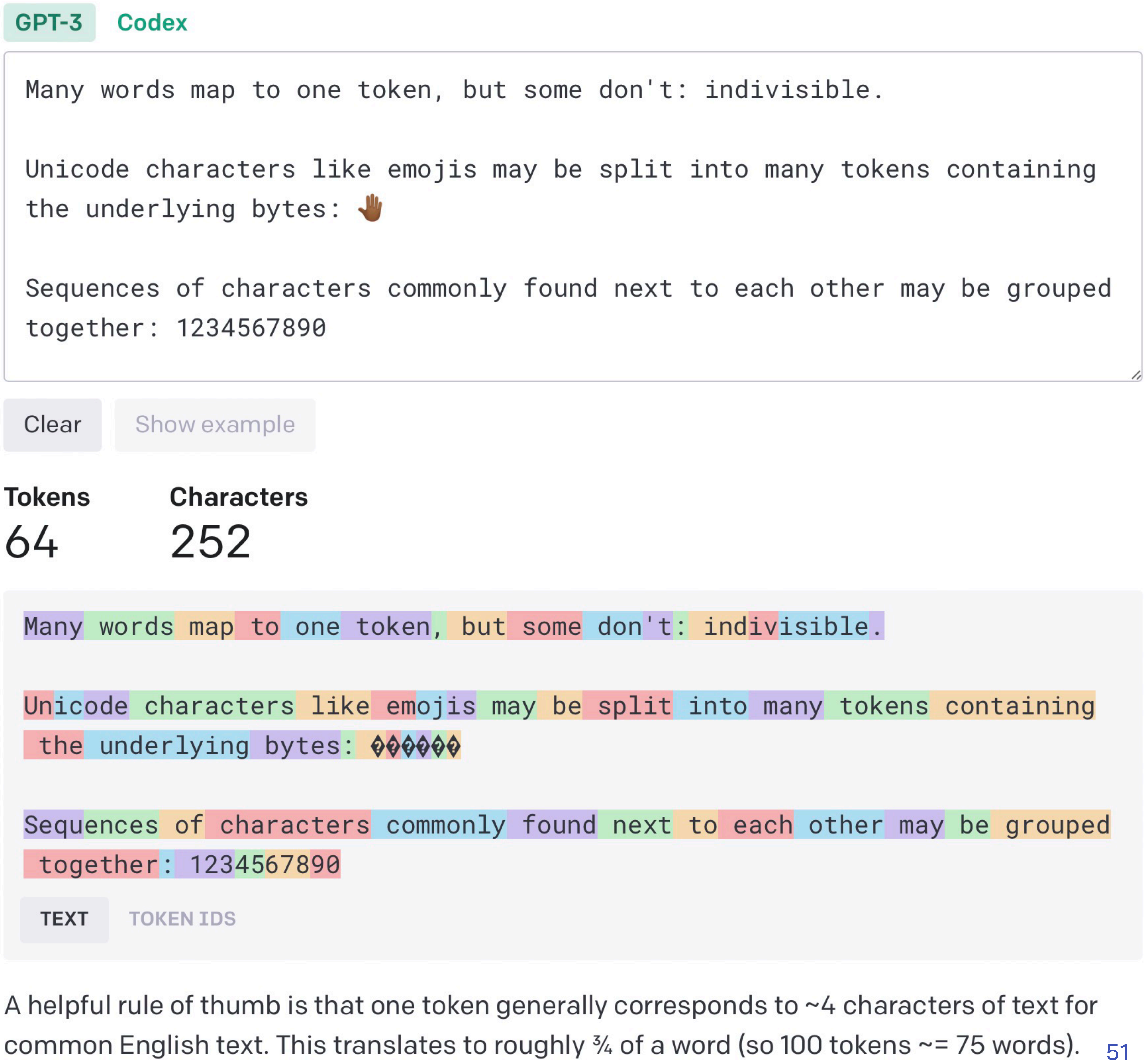
GPT-2 Training Data

- Found that Common Crawl has major data quality issues
- Formed the WebText dataset
  - scraped all outbound links (45M) from Reddit which received at least 3 karma
- After de-duplication and some heuristic filtering, left with 8M documents for a total of 40GB of text

https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

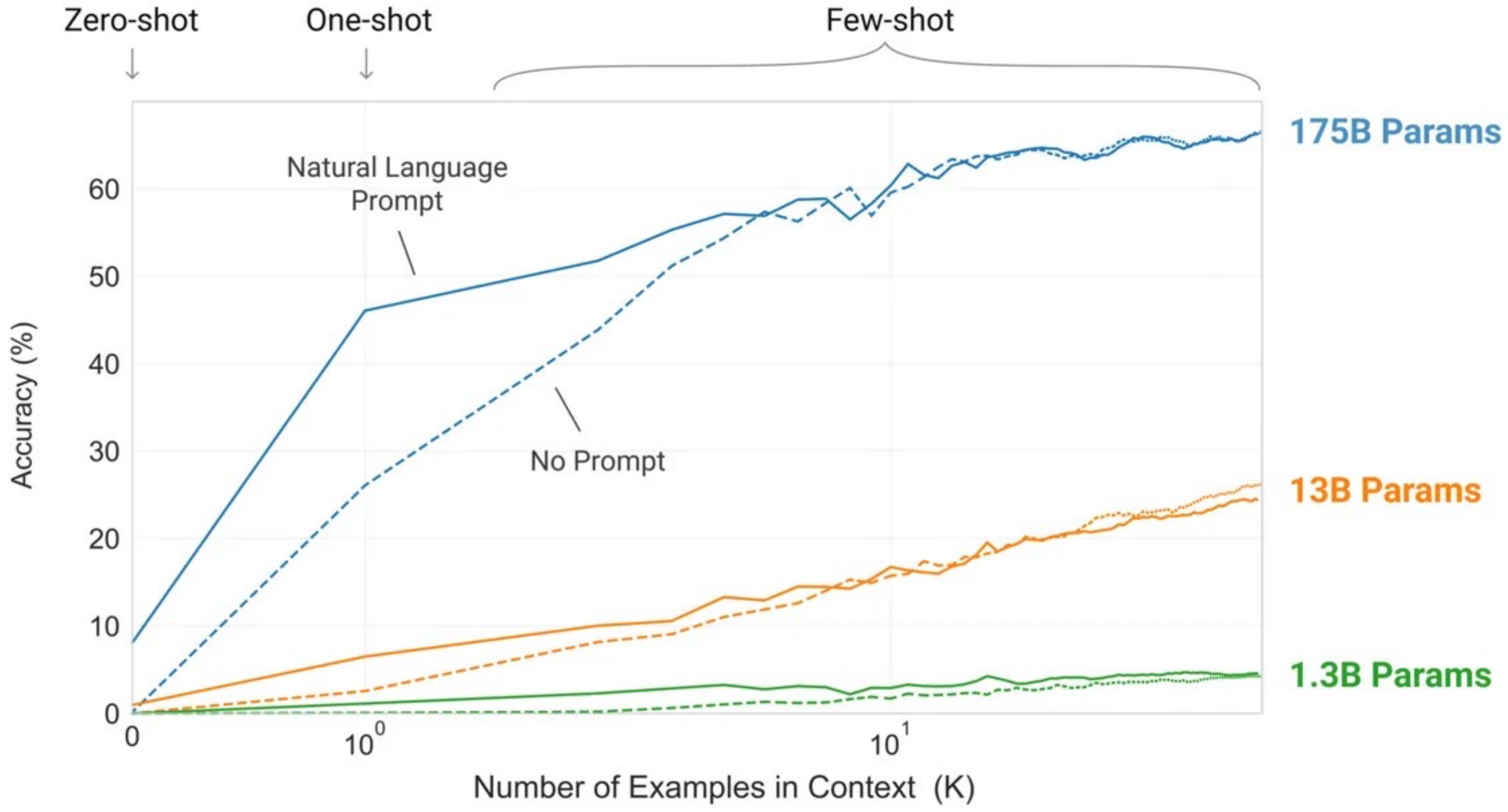
Byte Pair Encoding

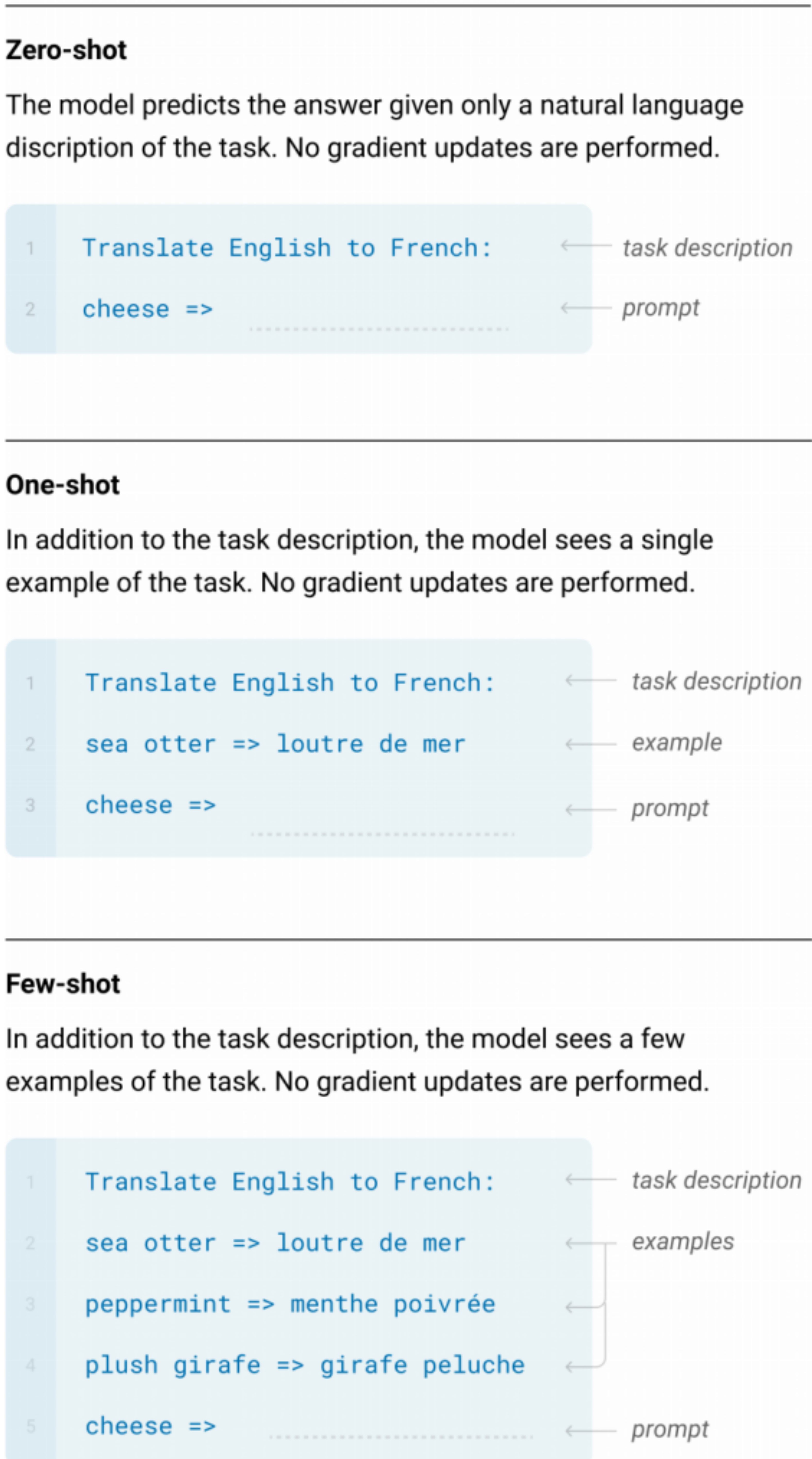
- How does GPT tokenize?
- Middle ground between
  - old-school NLP tokenization, where out-of-vocab words would be replaced by a special token
  - UTF-8 bytes



GPT-3 (2020)

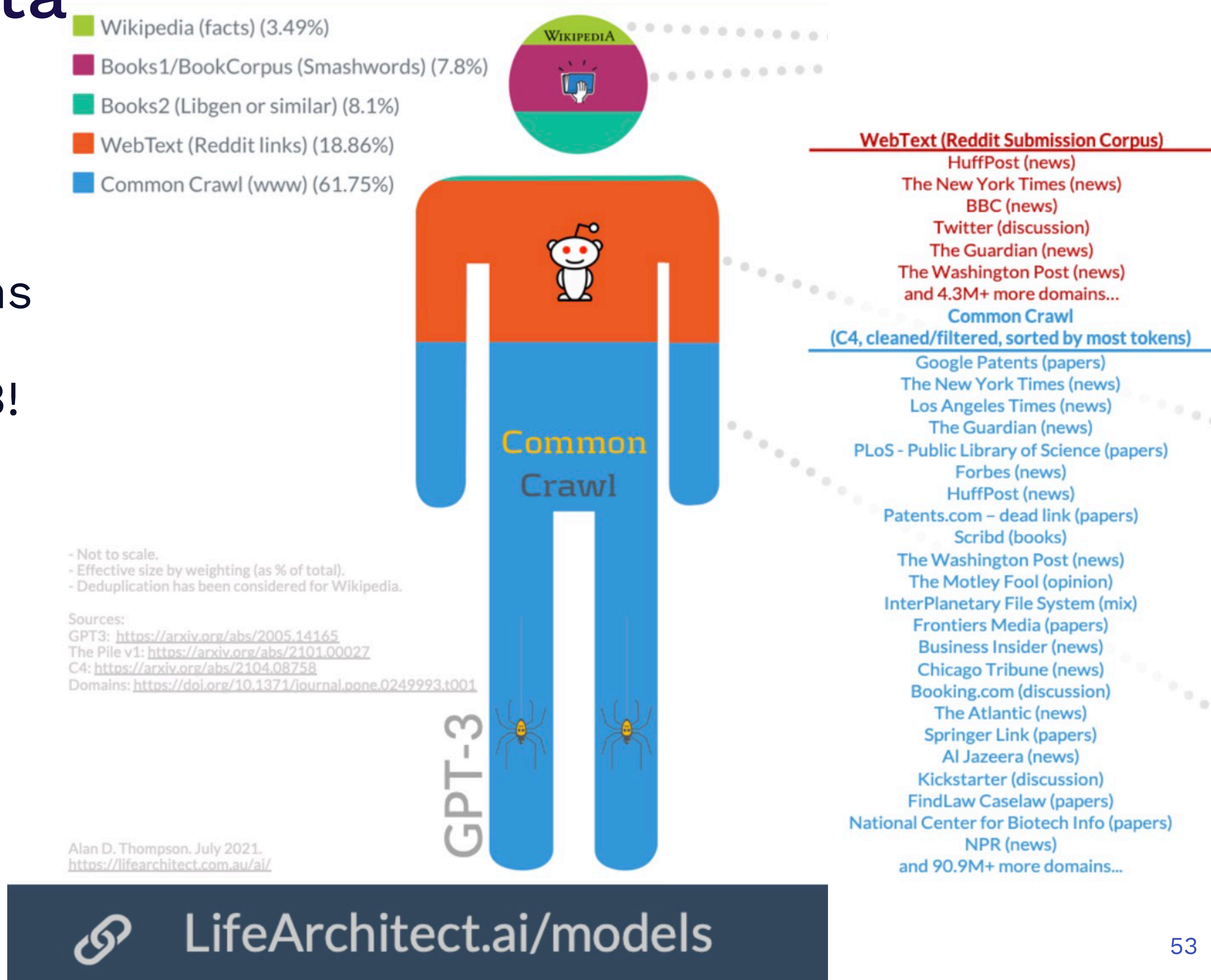
- Just like GPT-2, but 100x larger (175B params)
- Exhibited unprecedented few-shot and zero-shot learning





GPT-3 Training Data

- For a total of 500B tokens
- But trained on only 300B!



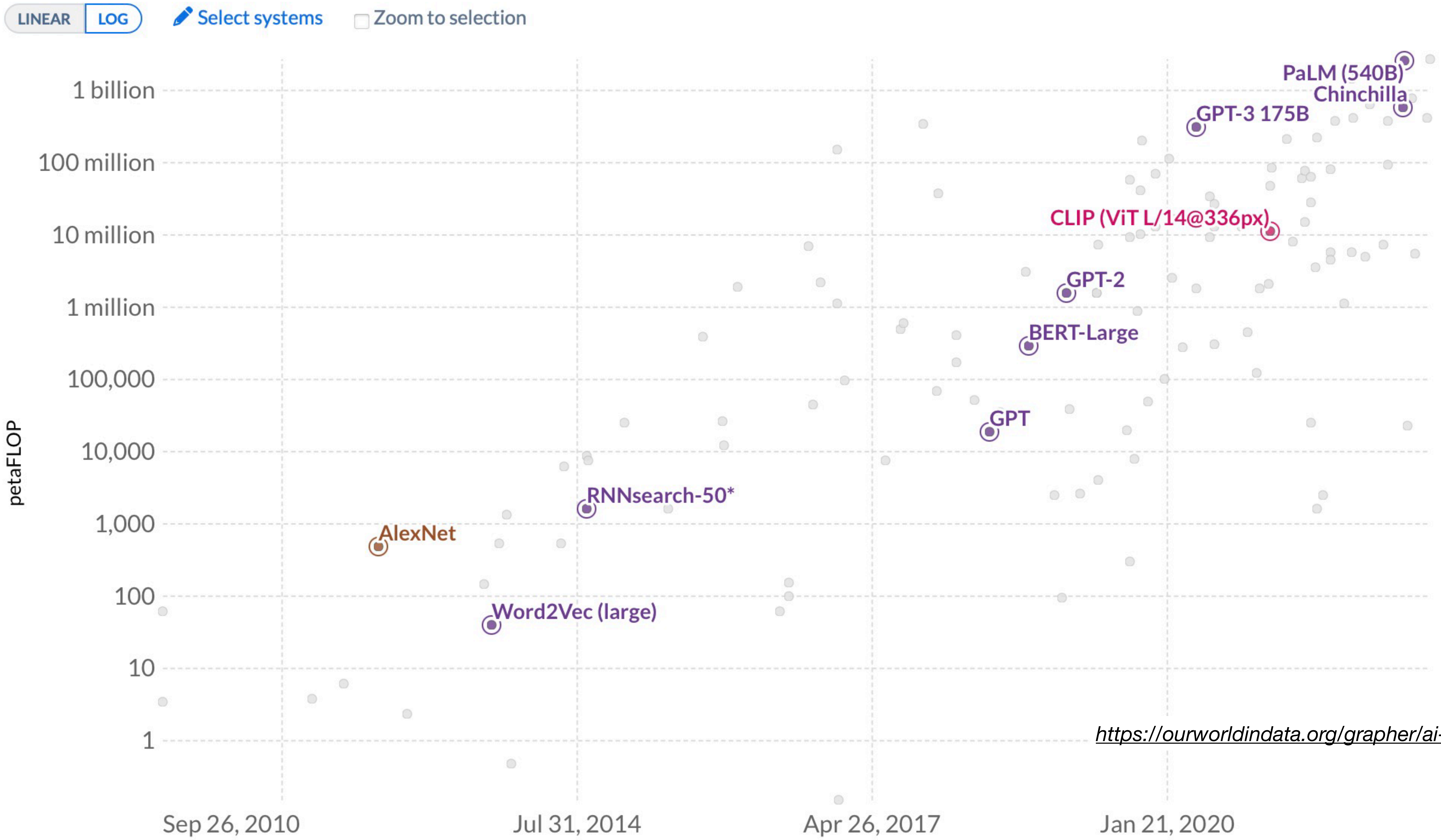
https://arxiv.org/pdf/2005.14165.pdf

GPT-4 (2023)

This report focuses on the capabilities, limitations, and safety properties of GPT-4. GPT-4 is a Transformer-style model [39] pre-trained to predict the next token in a document, using both publicly available data (such as internet data) and data licensed from third-party providers. The model was then fine-tuned using Reinforcement Learning from Human Feedback (RLHF) [40]. Given both the competitive landscape and the safety implications of large-scale models like GPT-4, this report contains no further details about the architecture (including model size), hardware, training compute, dataset construction, training method, or similar.

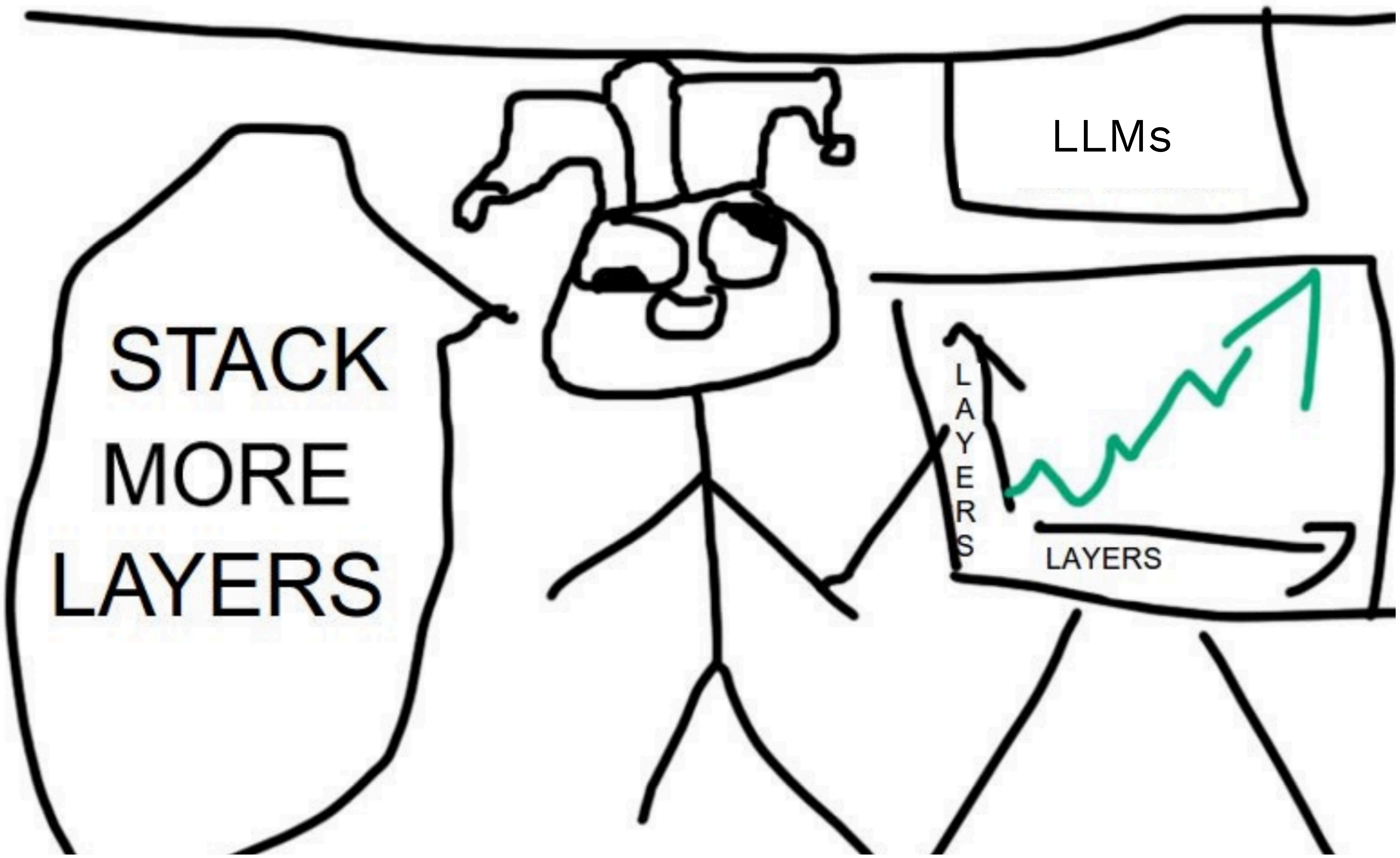
Computation used to train notable AI systems

Computation is measured in petaFLOP, which is 10^{15} floating-point operations.



https://ourworldindata.org/grapher/ai-training-computation

The Bitter Lesson



But what exactly is the relationship between model size and dataset size?

Chinchilla (2022)

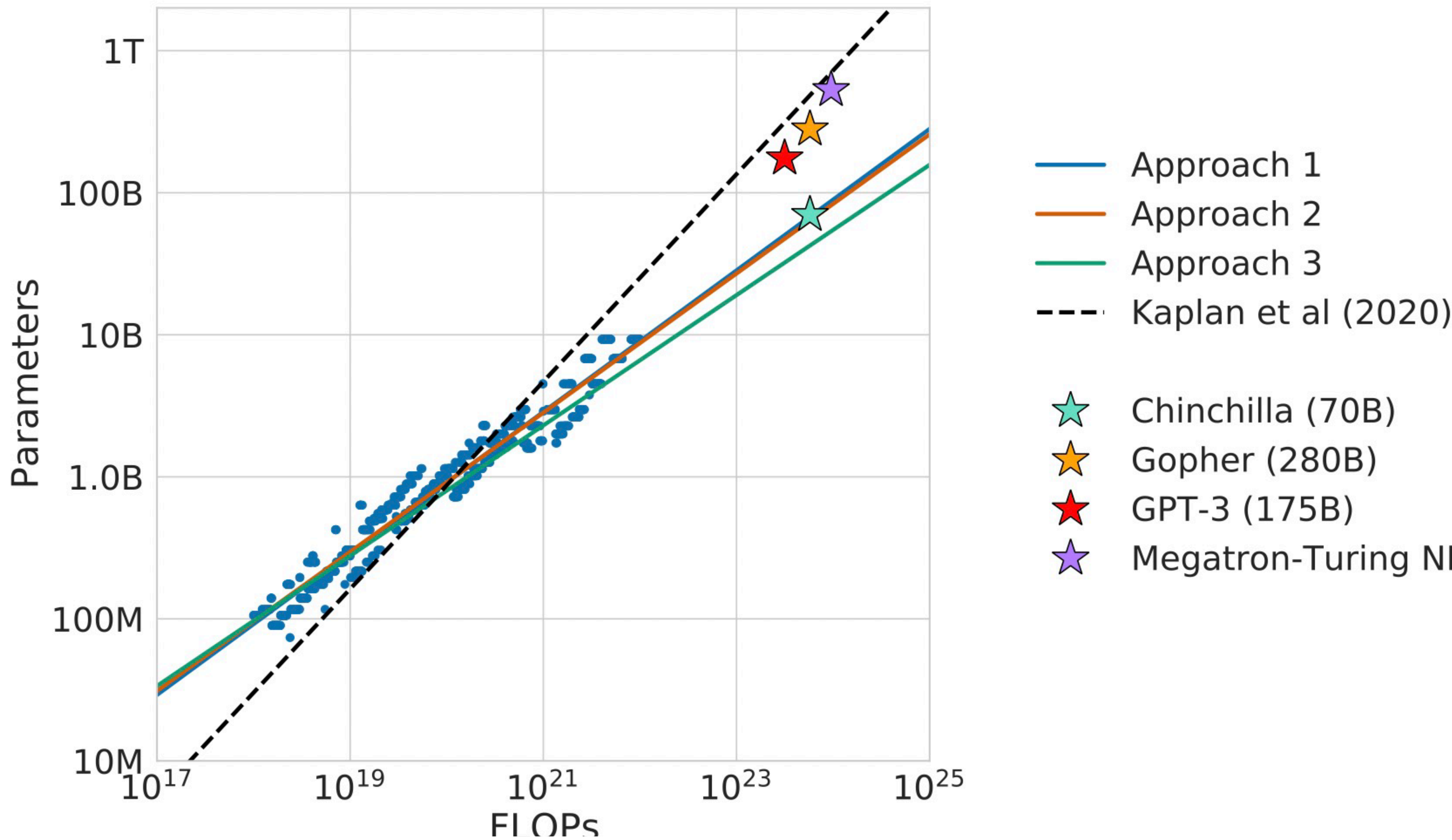
- Empirically derived formulas for optimal model and training set size given a fixed compute budget
- Found that most LLMs are "undertrained"
- Trained Chinchilla (70B) vs Gopher (280B) at the same compute budget, by using 4x fewer params and 4x more data
- (Note that this is for one epoch)

https://arxiv.org/pdf/2203.15556.pdf

DeepMind

Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*
*Equal contributions



Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

LLaMA (2023)

- "Chinchilla-optimal" open-source LLMs from Meta
- Several sizes from 7B to 65B, trained on at least 1T tokens
- Benchmarks competitively against GPT-3 and other LLMs
- Open-source, but non-commercial

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

		Humanities	STEM	Social Sciences	Other	Average
GPT-NeoX	20B	29.8	34.9	33.7	37.7	33.6
GPT-3	175B	40.8	36.7	50.4	48.8	43.9
Gopher	280B	56.2	47.4	71.9	66.1	60.0
Chinchilla	70B	63.6	54.9	79.3	**73.9**	67.5
PaLM	8B	25.6	23.8	24.1	27.8	25.4
	62B	59.5	41.9	62.7	55.8	53.7
	540B	**77.0**	**55.6**	**81.0**	69.6	**69.3**
LLaMA	7B	34.0	30.5	38.3	38.1	35.1
	13B	45.0	35.8	53.8	53.3	46.9
	33B	55.8	46.0	66.7	63.4	57.8
	65B	61.8	51.7	72.9	67.4	63.4

Table 9: **Massive Multitask Language Understanding (MMLU).** Five-shot accuracy.

LLaMA Training Data

- Custom quality-filtering of CommonCrawl + some C4 + Github + Wikipedia + Books + ArXiV + Stack Exchange

- RedPajama: open-source recreation

	RedPajama	LLaMA*
CommonCrawl	878 billion	852 billion
C4	175 billion	190 billion
Github	59 billion	100 billion
Books	26 billion	25 billion
ArXiv	28 billion	33 billion
Wikipedia	24 billion	25 billion
StackExchange	20 billion	27 billion
Total	1.2 trillion	1.25 trillion

https://www.together.xyz/blog/redpajama

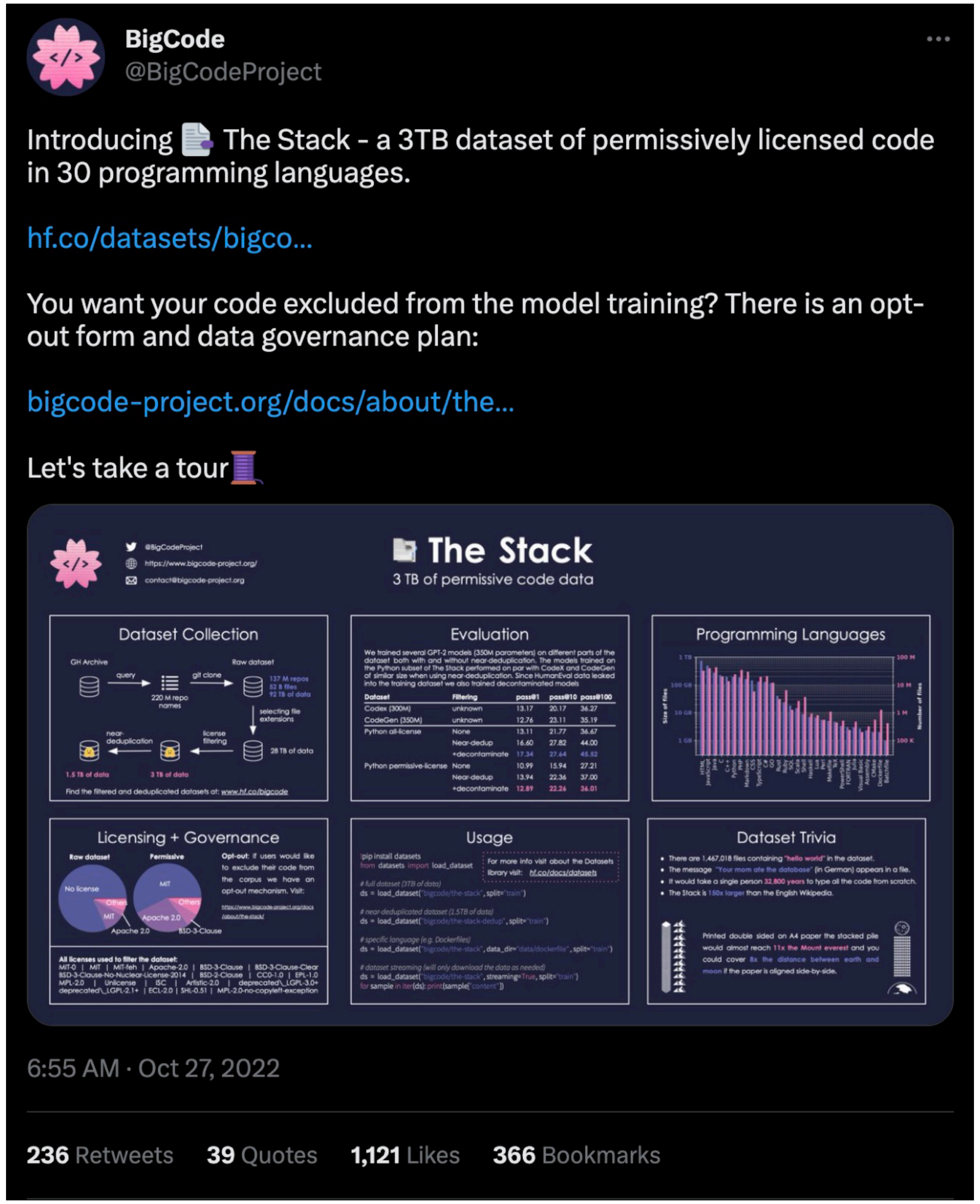
Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

https://arxiv.org/pdf/2302.13971.pdf

Including code in training data

- T5 and GPT-3 (2020) specifically removed code. But most recent models are trained on ~5% code. Why?
- Code-specific models such as OpenAI Codex (2021) was GPT-3 further trained on public GitHub code.
- Empirically, this improved performance on non-code tasks!
- Open-source dataset: The Stack (3TB of permissively licensed source code)

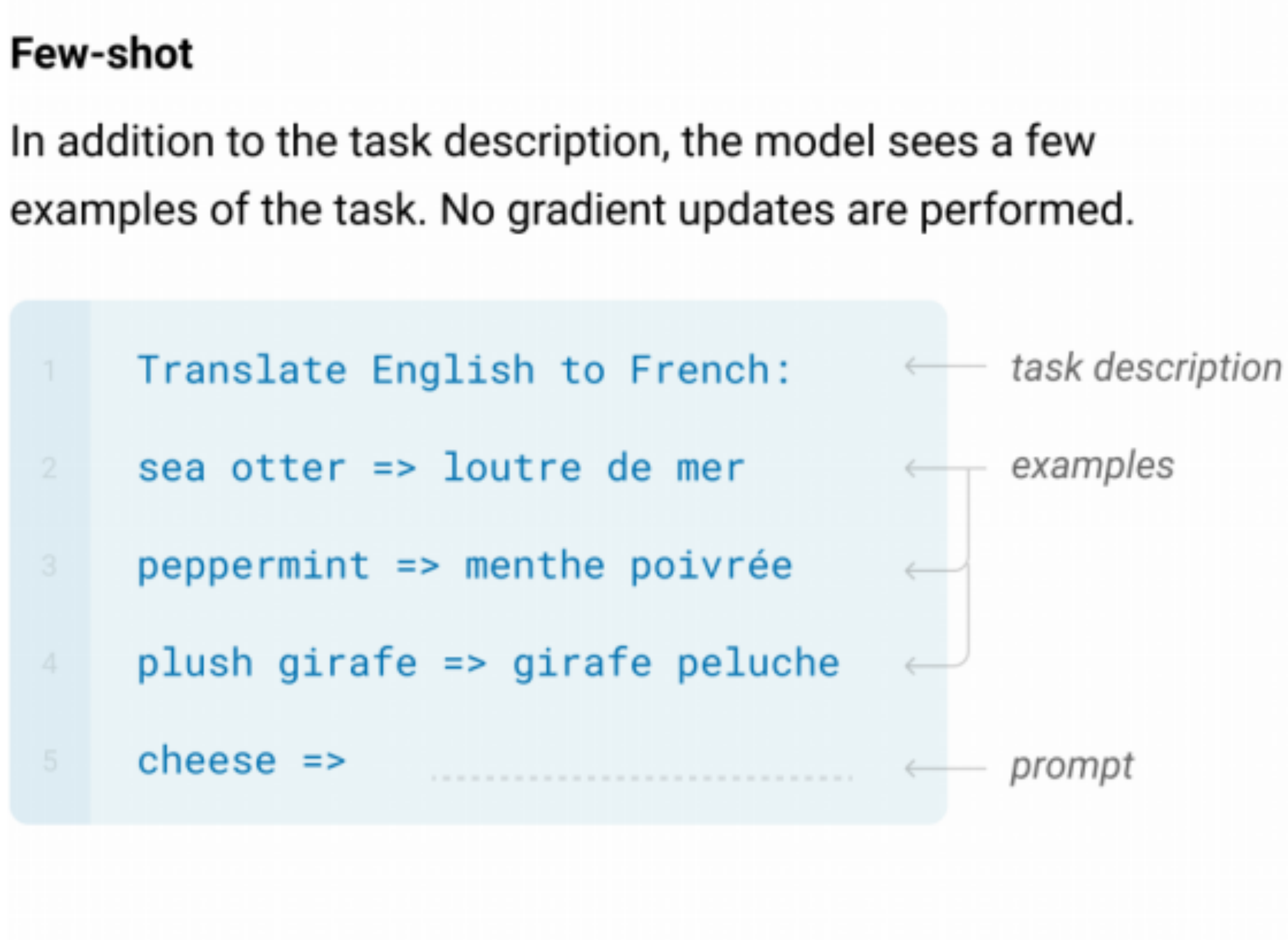


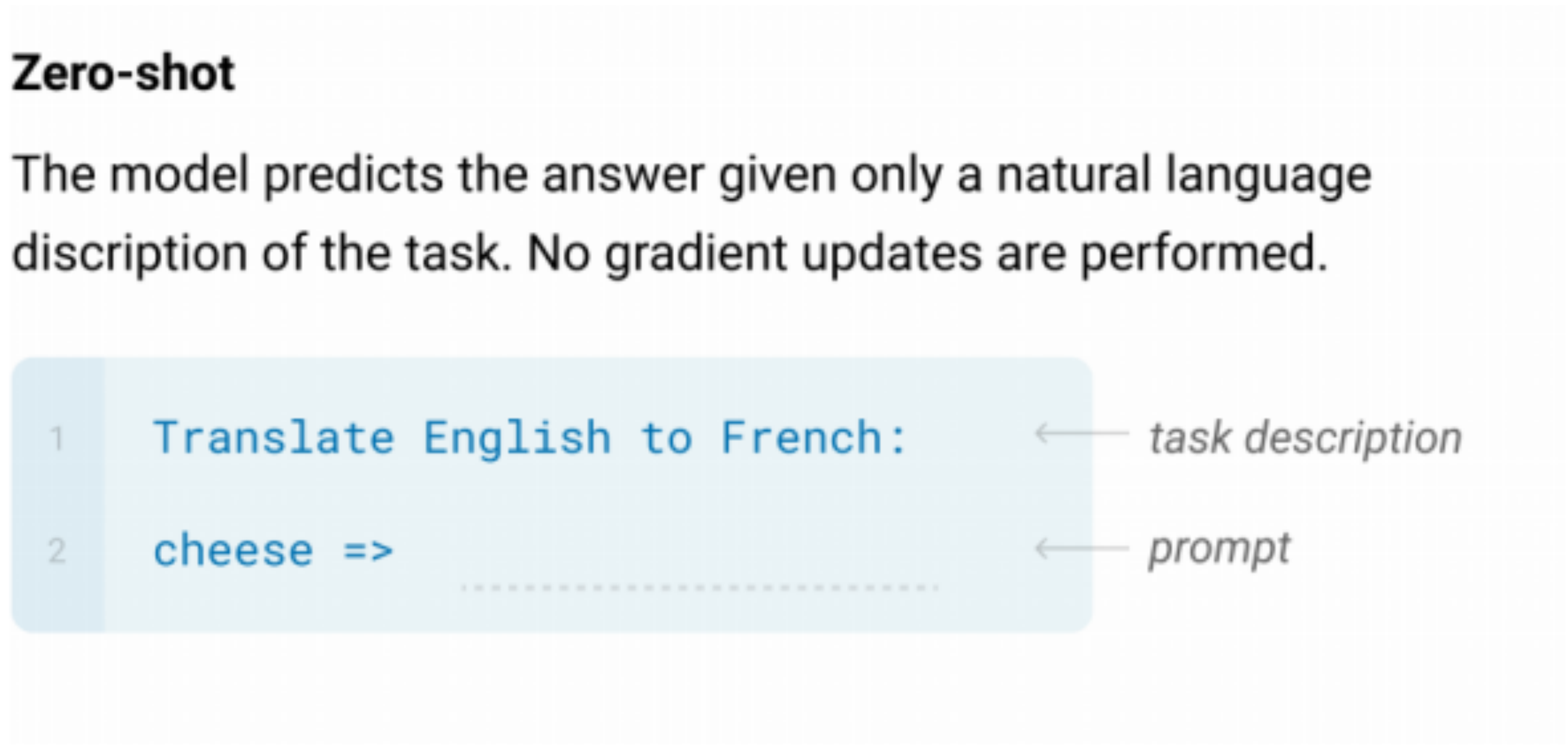
Yao Fu et al. How does GPT Obtain its Ability?

And there's another important part of the story: Instruction Tuning

Few-shot vs Zero-shot

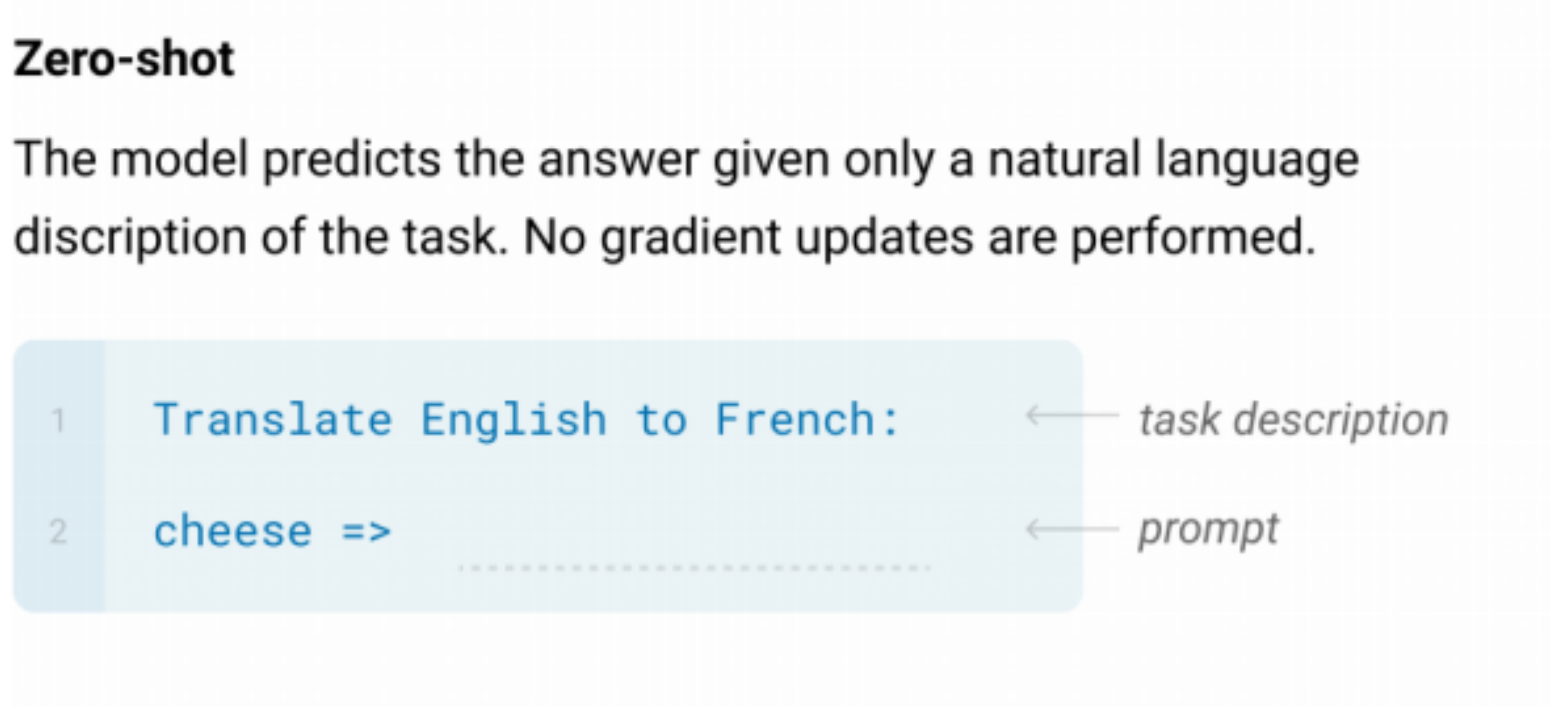
- At the time of GPT-3 (2020), the mindset was mostly few-shot
  - e.g. text completion
- By the time of ChatGPT (2022), the mindset was all zero-shot
  - e.g. instruction-following

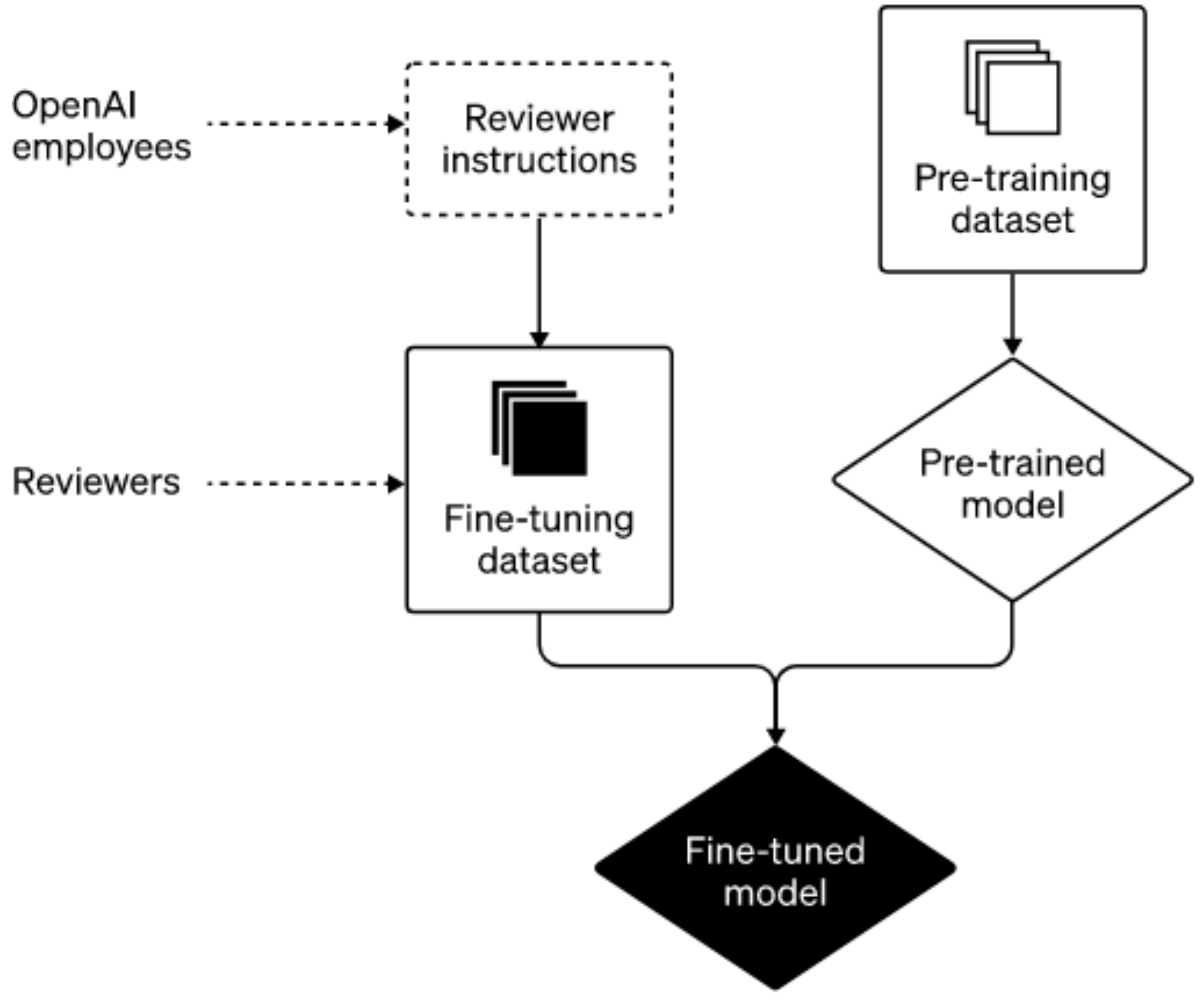




Supervised Fine-tuning

- Very little text in the original GPT-3 dataset is of the zero-shot form.
- To improve performance on zero-shot inputs, fine-tuned on a smaller high-quality dataset of instructions-completions
- (Sourced from thousands of contractors)





https://openai.com/blog/how-should-ai-systems-behave

InstructGPT/GPT-3.5

- Had humans rank different GPT-3 outputs, and used RL to further fine-tune the model
- **Much** better at following instructions
- Released as text-davinci-002 in OpenAI API

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

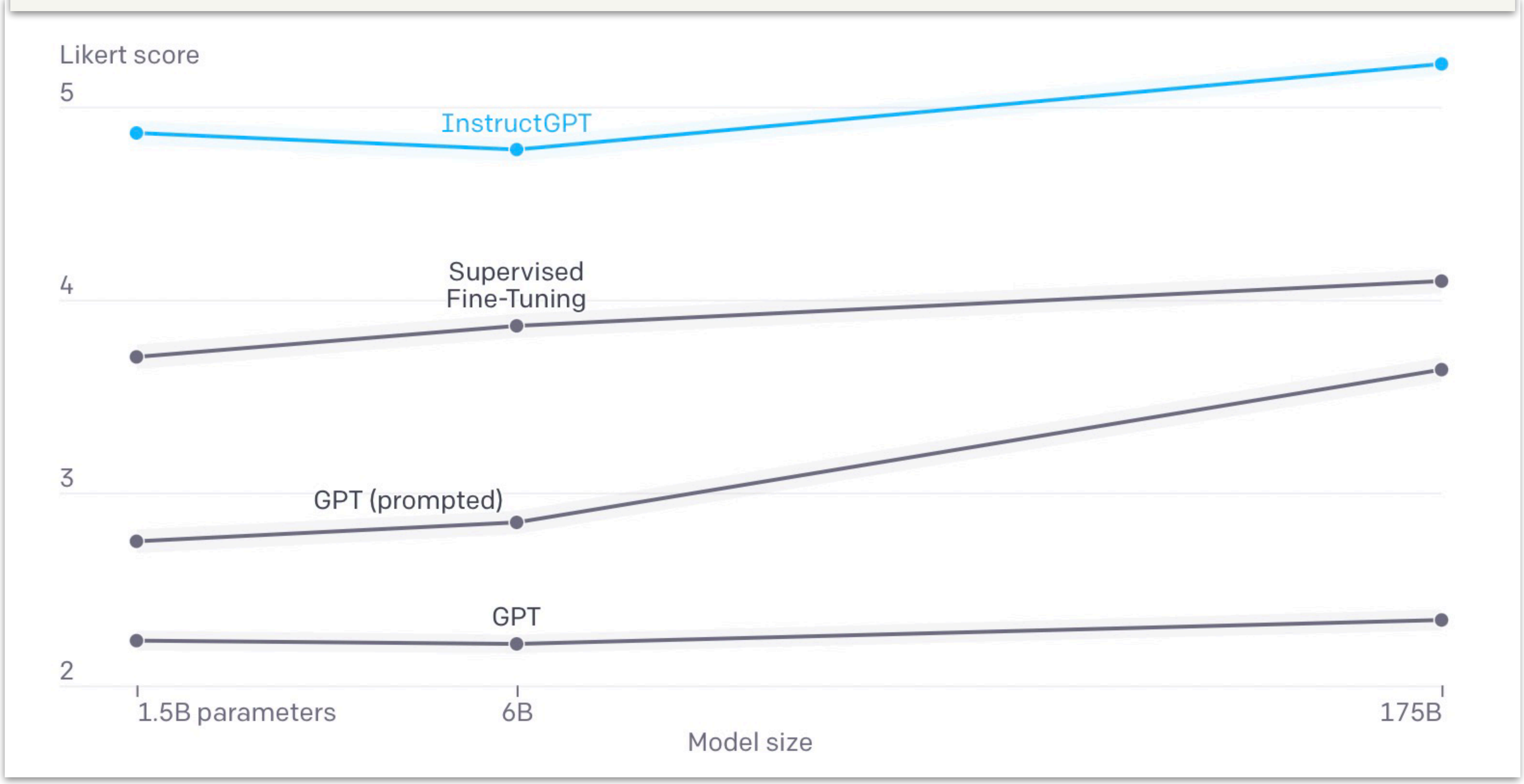
COMPLETION

GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

InstructGPT

```
People went to the moon, and they took pictures of what they saw, and sent them
back to the earth so we could all see them.
```



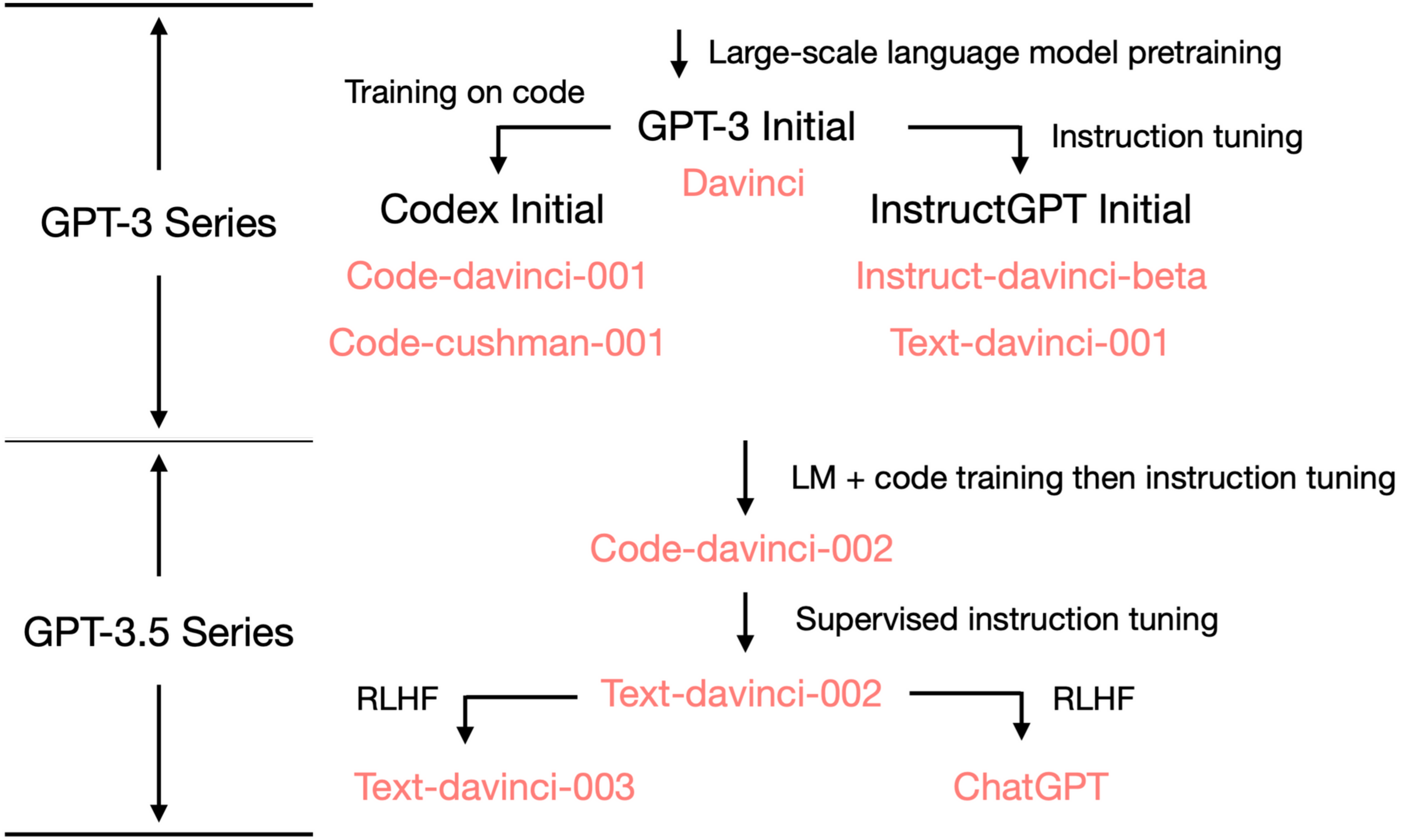
https://openai.com/blog/instruction-following/

ChatGPT

- Further RLHF on **conversations**
- ChatML format (messages from system, assistant, user roles)

```
openai.ChatCompletion.create(
  model="gpt-3.5-turbo",
  messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Who won the world series in 2020?"},
        {"role": "assistant", "content": "The Los Angeles Dodgers won the Worl
        {"role": "user", "content": "Where was it played?"}
    ]
)
```

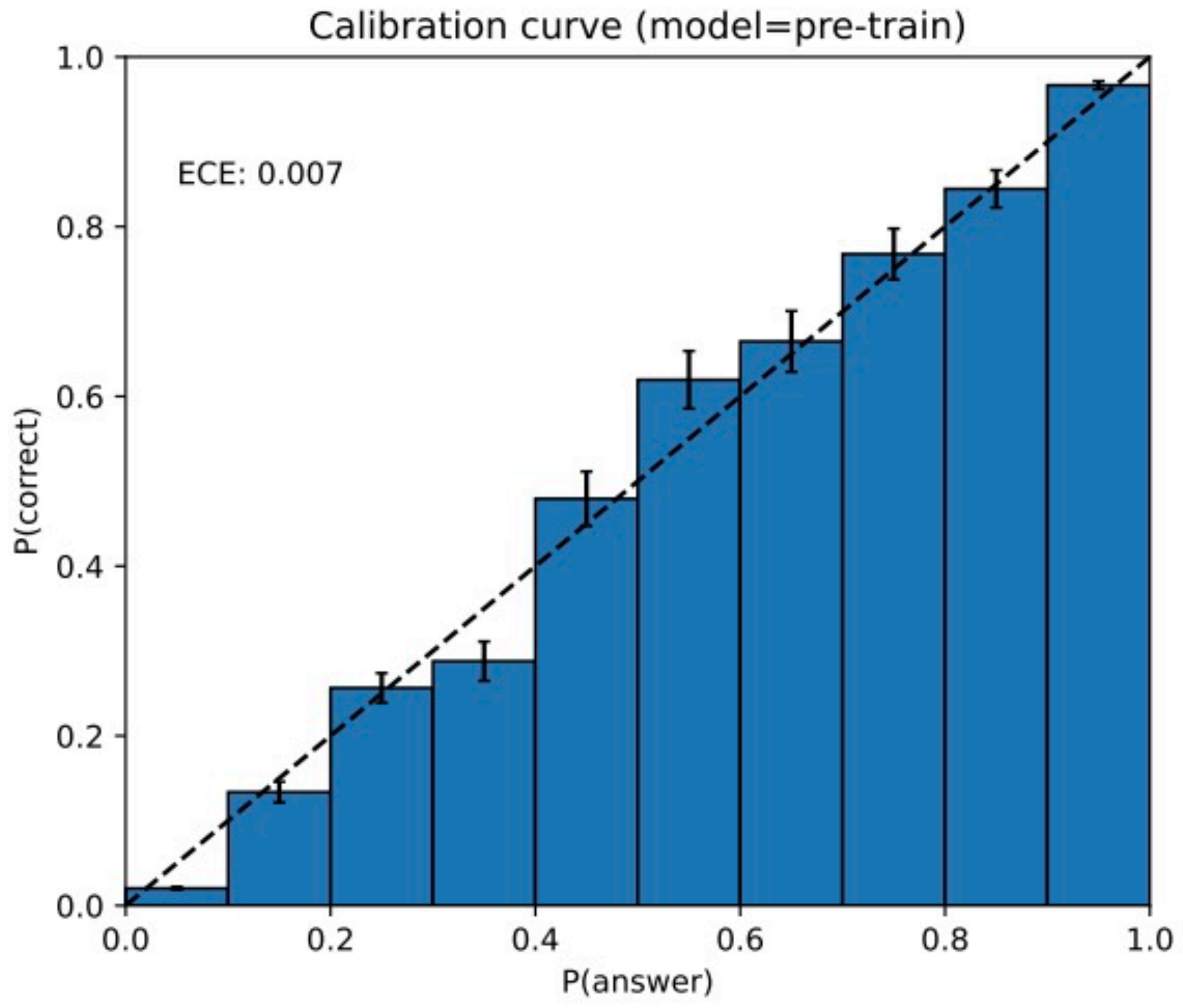
The GPT Lineage



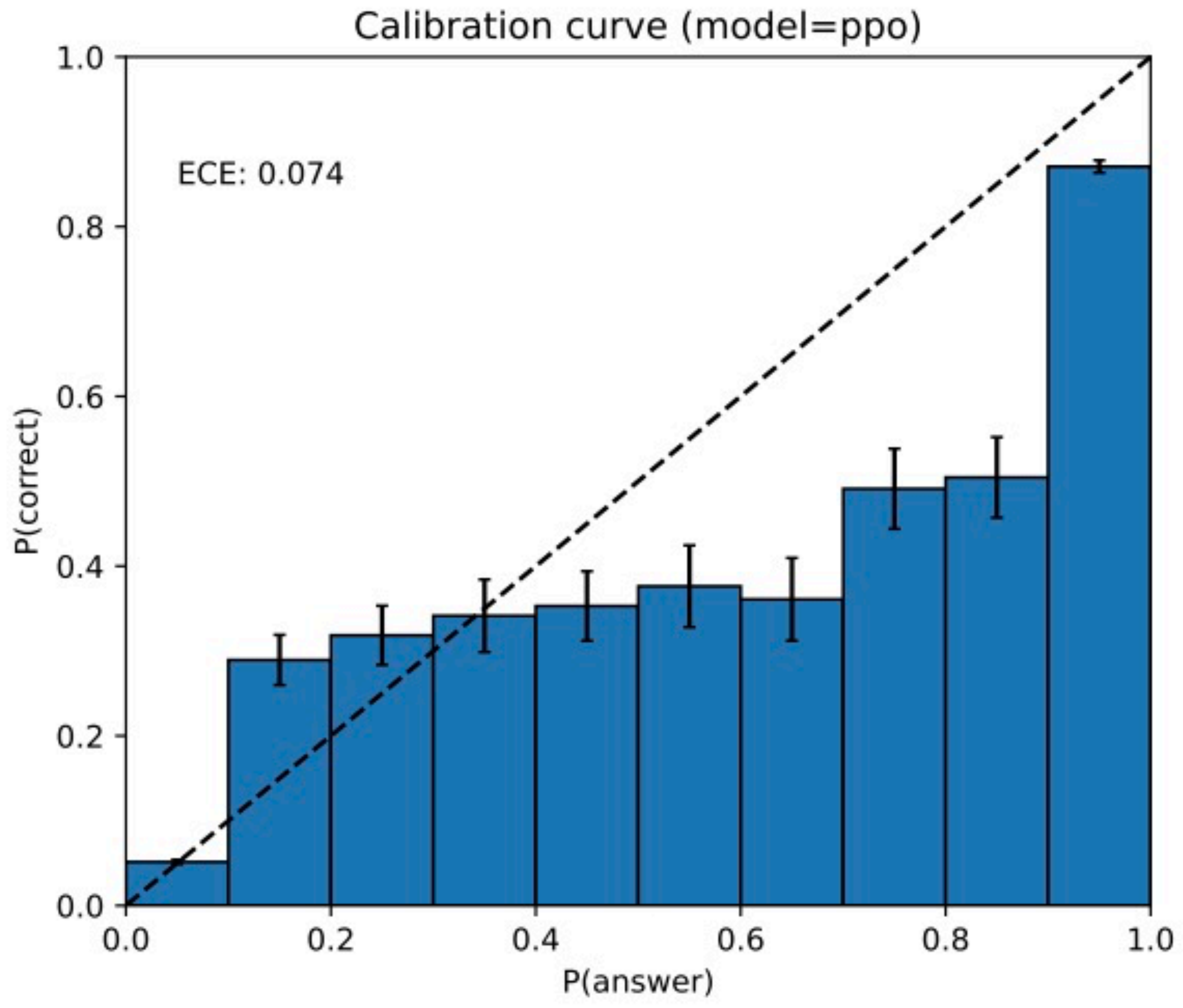
Yao Fu's <u>How does GPT Obtain its Ability?</u>

"Alignment Tax"

- Instruction-tuning increases the model's zero-shot ability, but at a cost
  - Confidence becomes less calibrated
  - Few-shot ability suffers



Base GPT-4



RLHF GPT-4

GPT-4 Technical Report - https://arxiv.org/pdf/2303.08774.pdf

It's possible to "steal" RLHF

*We introduce **Alpaca 7B**, a model fine-tuned from the LLaMA 7B model on 52K instruction-following demonstrations. On our preliminary evaluation of single-turn instruction following, Alpaca behaves qualitatively similarly to OpenAI's text-davinci-003, while being surprisingly small and easy/cheap to reproduce (<600$).*

Web Demo GitHub

Stanford
Alpaca

- Got 52K instruction-following demonstrations from text-davinci-003, then fine-tuned LLaMA on them.

OpenAssistant

- April 2023 dataset release
- 160K messages across 66K conversation trees, 35 languages, 460K quality ratings, 13.5K volunteers

OpenAssistant Conversations - Democratizing Large Language Model Alignment

Andreas Köpf*
andreas.koepf@provisio.com

Yannic Kilcher*
yannic@ykilcher.com

Dimitri von Rütte **Sotiris Anagnostidis** **Zhi-Rui Tam** **Keith Stevens**

Abdullah Barhoum **Nguyen Minh Duc** **Oliver Stanley** **Richárd Nagyfi** **Shahul ES**

Sameer Suri **David Glushkov** **Arnav Dantuluri** **Andrew Maguire**

Christoph Schuhmann **Huu Nguyen**

Alexander Mattick
alexander.mattick@googlemail.com

https://huggingface.co/datasets/OpenAssistant/oasst1

And one last idea

Retrieval-enhanced Transformer (2021)

- Instead of both learning language and memorizing facts in the model's params, why not just learn language in params, and retrieve facts from a large database?
- BERT-encode sentences, store them in large DB (>1T tokens)
- Then, fetch matching sentences and attend to them.
- Doesn't work as well as large LLMs. Yet.

DeepMind

Improving language models by retrieving from trillions of tokens

Sebastian Borgeaud[†], Arthur Mensch[†], Jordan Hoffmann[†], Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae[‡], Erich Elsen[‡] and Laurent Sifre[†,‡]
All authors from DeepMind, [†]Equal contributions, [‡]Equal senior authorship

https://arxiv.org/pdf/2112.04426.pdf Dec 2021

Resources

- Lillian Weng's "The Transformer Family v2" megapost
- Xavier Amatriain's Transformer Models Catalog
- Yao Fu's How does GPT Obtain its Ability?

Questions?

04

Training & Inference

Problems with training LLMs

- Massive amounts of data
- Massive models don't fit on a single GPU or even a single multi-GPU machine
- Long training runs are painful

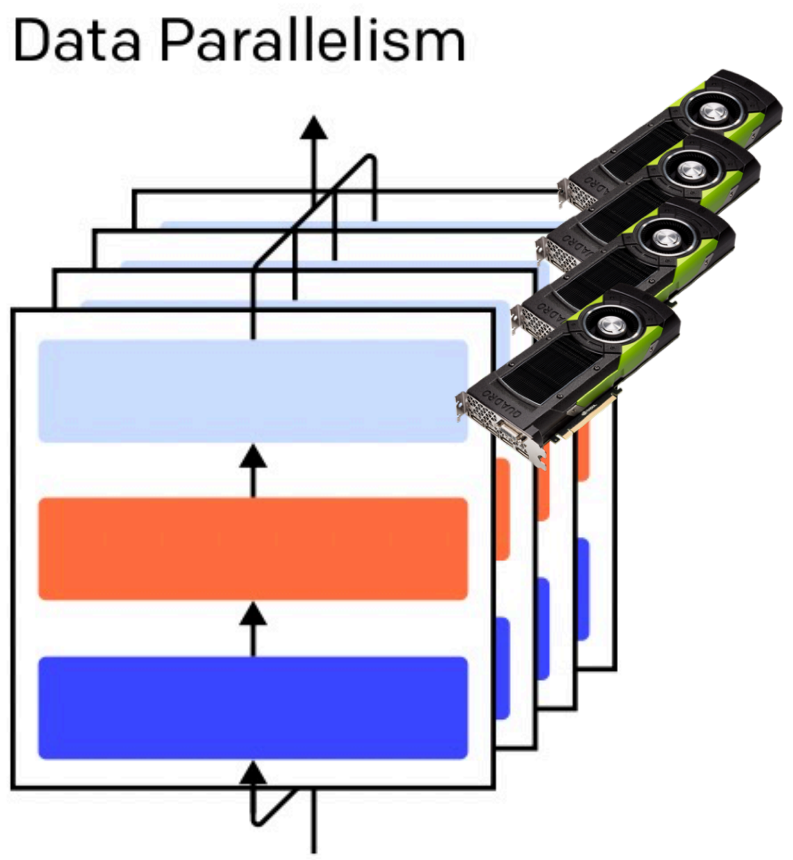
Problems with training LLMs

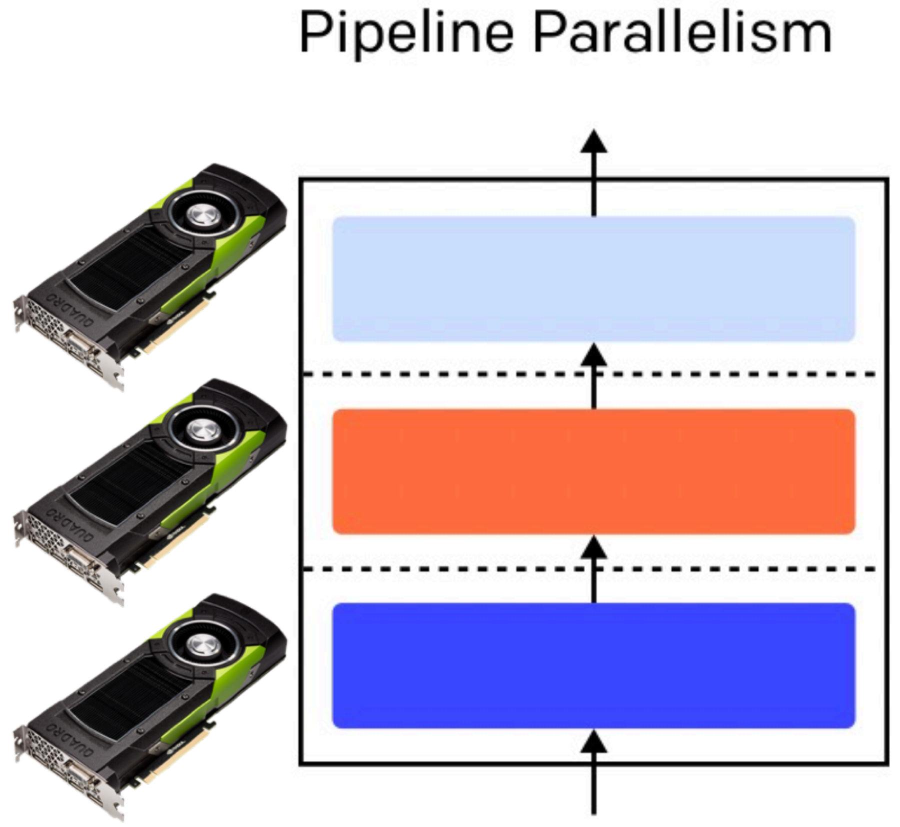
- **Massive amounts of data**
- Massive models don't fit on a single GPU or even a single multi-GPU machine
- Long training runs are painful

Problems with training LLMs

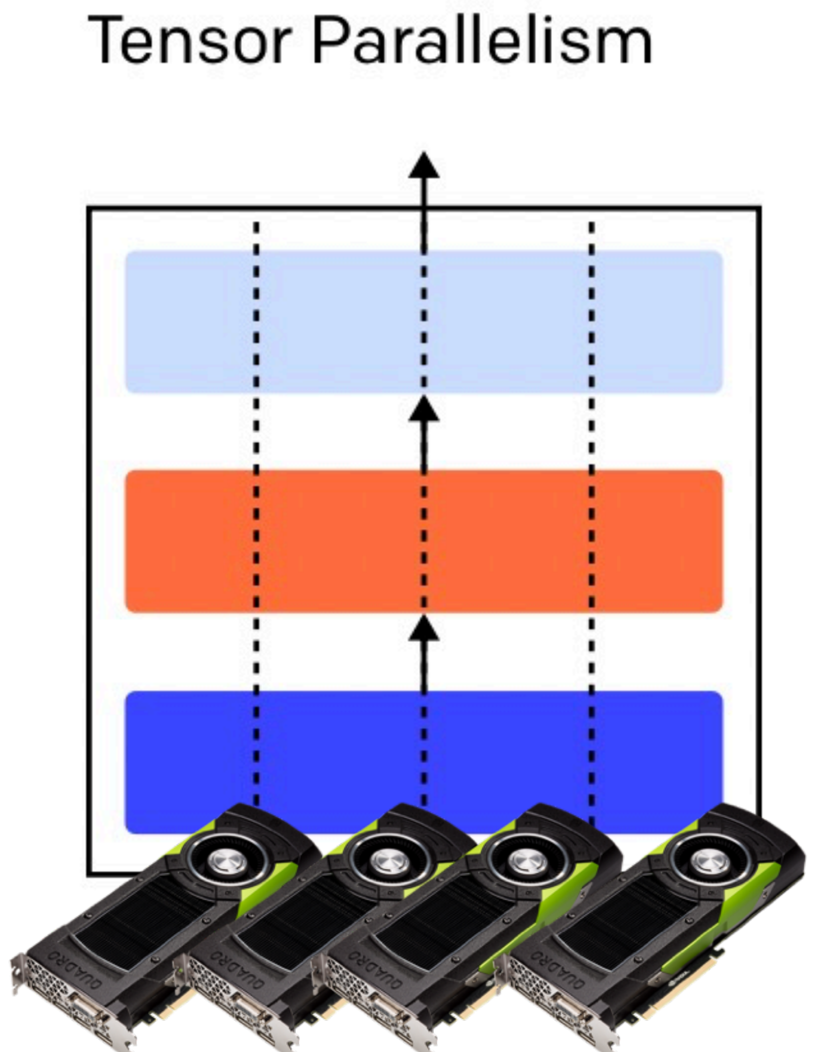
- Massive amounts of data
- **Massive models don't fit on a single multi-GPU machine**
- Long training runs are painful

Parallelism





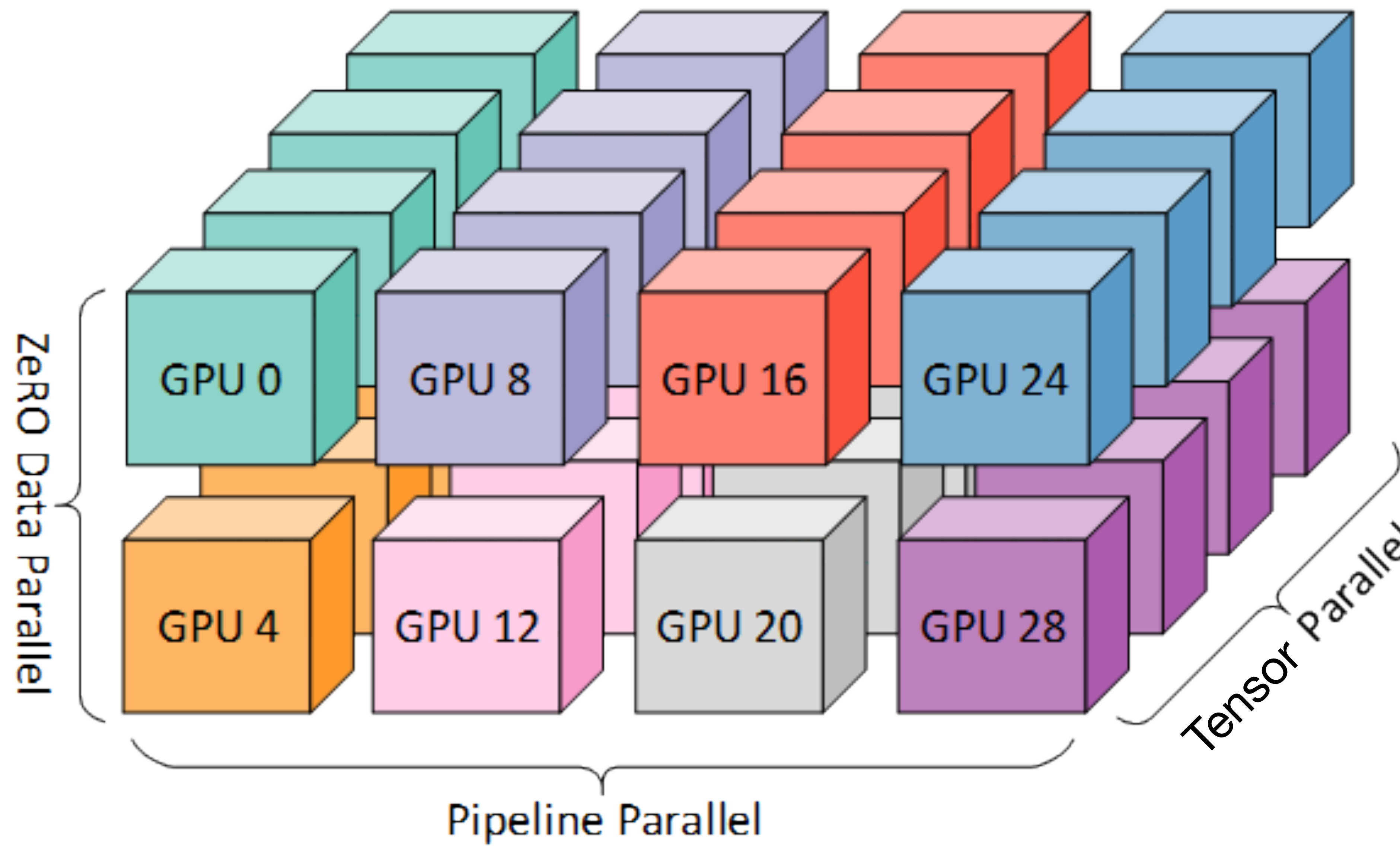
- Data parallelism: spread a single batch of data across GPUs
- Model parallelism: spread the model's layers across GPUs
- Tensor parallelism: spread a single matrix op across GPUs



https://openai.com/blog/techniques-for-training-large-neural-networks/

BLOOM (GPT-3 sized LM)

Component	DeepSpeed	Megatron-LM
ZeRO Data Parallelism	V	
Tensor Parallelism		V
Pipeline Parallelism	V	
BF16Optimizer	V	
Fused CUDA Kernels		V
DataLoader		V

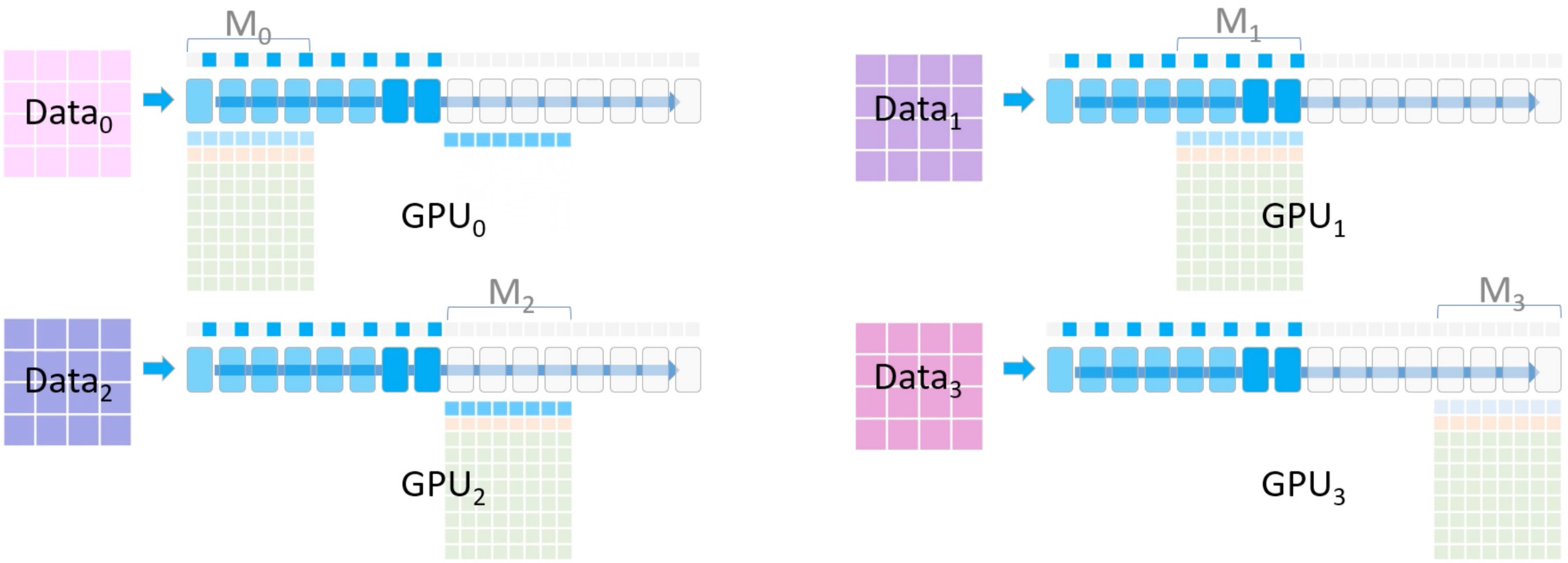


Had to use multiple tricks ("3D parallelism")
from two great libraries: DeepSpeed and Megatron-LM

https://huggingface.co/blog/bloom-megatron-deepspeed

https://www.deepspeed.ai/training/

Sharded Data-Parallelism



GPU_2 broadcasts the parameters for M_2

Literally pass around model params between GPUs as computation is proceeding!

Helpful video:
https://www.microsoft.com/en-us/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

Problems with training LLMs

- Massive amounts of data
- Massive models don't fit on a single GPU or even a single multi-GPU machine
- **Long training runs are painful**

A glimpse into training hell

OPT: Open Pre-trained Transformer Language Models

Susan Zhang*, Stephen Roller*, Naman Goyal*,
Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li,
Xi Victoria Lin, Todor Mihaylov, Myle Ott†, Sam Shleifer†, Kurt Shuster, Daniel Simig,
Punit Singh Koura, Anjali Sridhar, Tianlu Wang, Luke Zettlemoyer
Meta AI
{susanz,roller,naman}@fb.com

- Dozens of manual restarts, 70+ automatic restarts due to HW failures
- Manual restarting from checkpoints when loss would diverge
- Switching optimizers and software versions in the middle of training

2021-11-28 10:09am ET [Stephen]: 12.30

- 12.29 failed with the same `filename storages not found`
  - Since the exception said pg0-55, i ssh'd into it and tried manually loading its checkpoints. All 6 parts got the same storages exception!
  - I could replicate this with the storages I had manually downloaded
  - Conclusion: 33250 checkpoints are corrupt. Maybe from R12.26 and R12.25 aggressively overwriting the checkpoints.

https://arxiv.org/pdf/2205.01068.pdf

"Training run babysitting"

GPT-4 contributions

Pretraining

Core contributors
Christopher Berner *Supercomputing lead*
Greg Brockman *Infrastructure lead*
Trevor Cai *Throughput lead*
David Farhi *Manager of optimization team*
Chris Hesse *Infrastructure usability co-lead*
Shantanu Jain *Infrastructure usability co-lead*
Kyle Kosic *Uptime and stability lead*
Jakub Pachocki *Overall lead, optimization lead*
Alex Paino *Architecture & data vice lead*
Mikhail Pavlov *Software correctness lead*
Michael Petrov *Hardware correctness lead*
Nick Ryder *Architecture & data lead*
Szymon Sidor *Optimization vice lead*
Nikolas Tezak *Execution lead*
Phil Tillet *Triton lead*
Amin Tootoonchian *Model distribution, systems & networking lead*
Qiming Yuan *Dataset sourcing and processing lead*
Wojciech Zaremba *Manager of dataset team*

Compute cluster scaling
Christopher Berner, Oleg Boiko, Andrew Cann, Ben Chess, Christian Gil
Emy Parparita, Henri Roussez, Eric Sigler, Akila Welihinda

Data
Sandhini Agarwal, Suchir Balaji, Mo Bavarian, Che Chang, Sheila Dunni
Gordon, Peter Hoeschele, Shawn Jain, Shantanu Jain, Roger Jiang, Hee
Kaiser, Nitish Shirish Keskar, Jong Wook Kim, Aris Konstantinidis, Chak
Bianca Martin, David Mély, Oleg Murk, Hyeonwoo Noh, Long Ouyang, A
Pong, Alec Radford, Nick Ryder, John Schulman, Daniel Selsam, Ian Sol
Weng, Clemens Winter, Tao Xu, Qiming Yuan, Wojciech Zaremba

Distributed training infrastructure
Greg Brockman, Trevor Cai, Chris Hesse, Shantanu Jain, Yongjik Kim, K
Litwin, Jakub Pachocki, Mikhail Pavlov, Szymon Sidor, Nikolas Tezak, M
Amin Tootoonchian, Qiming Yuan

Hardware correctness
Greg Brockman, Shantanu Jain, Kyle Kosic, Michael Petrov, Nikolas Tezak, Amin Tootoonchian, Chelsea Voss, Qiming Yuan

Optimization & architecture
Igor Babuschkin, Mo Bavarian, Adrien Ecoffet, David Farhi, Jesse Han, Ingmar Kanitscheider, Daniel Levy, Jakub Pachocki, Alex Paino, Mikhail Pavlov, Nick Ryder, Szymon Sidor, Jie Tang, Jerry Tworek, Tao Xu

Training run babysitting
Suchir Balaji, Mo Bavarian, Greg Brockman, Trevor Cai, Chris Hesse, Shantanu Jain, Roger Jiang, Yongjik Kim, Kyle Kosic, Mateusz Litwin, Jakub Pachocki, Alex Paino, Mikhail Pavlov, Michael Petrov, Nick Ryder, Szymon Sidor, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Chelsea Voss, Ben Wang, Tao Xu, Qiming Yuan

Vision

Core contributors
Trevor Cai *Execution lead*
Mark Chen *Vision team co-lead, Deployment lead*
Casey Chu *Initial prototype lead*
Chris Hesse *Data load balancing & developer tooling lead*
Shengli Hu *Vision Safety Evaluations lead*
Yongjik Kim *GPU performance lead*
Jamie Kiros *Overall vision co-lead, deployment research & evaluation lead*
Daniel Levy *Overall vision co-lead, optimization lead*
Christine McLeavey *Vision team lead*
David Mély *Data lead*
Hyeonwoo Noh *Overall vision co-lead, research lead*
Mikhail Pavlov *Scaling engineering lead*
Raul Puri *Overall vision co-lead, engineering lead*
Amin Tootoonchian *Model distribution, systems & networking lead*

Architecture research
Casey Chu, Jamie Kiros, Christine McLeavey, Hyeonwoo Noh, Raul Puri, Alec Radford, Aditya Ramesh

Compute cluster scaling
Andrew Cann, Rory Carmichael, Christian Gibson, Henri Roussez, Akila Welihinda

Distributed training infrastructure
Trevor Cai, Yunxing Dai, Chris Hesse, Brandon Houghton, Yongjik Kim, Łukasz Kondraciuk, Hyeonwoo Noh, Mikhail Pavlov, Raul Puri, Nikolas Tezak, Amin Tootoonchian, Tianhao Zheng

Hardware correctness
Oleg Boiko, Trevor Cai, Michael Petrov, Alethea Power

Data
Jong Wook Kim, David Mély, Reiichiro Nakano, Hyeonwoo Noh, Long Ouyang, Raul Puri, Pranav Shyam, Tao Xu

Alignment Data
Long Ouyang

Training run babysitting
Trevor Cai, Kyle Kosic, Daniel Levy, David Mély, Reiichiro Nakano, Hyeonwoo Noh, Mikhail Pavlov, Raul Puri, Amin Tootoonchian

Deployment & post-training
Ilge Akkaya, Mark Chen, Jamie Kiros, Rachel Lim, Reiichiro Nakano, Raul Puri, Jiayi Weng

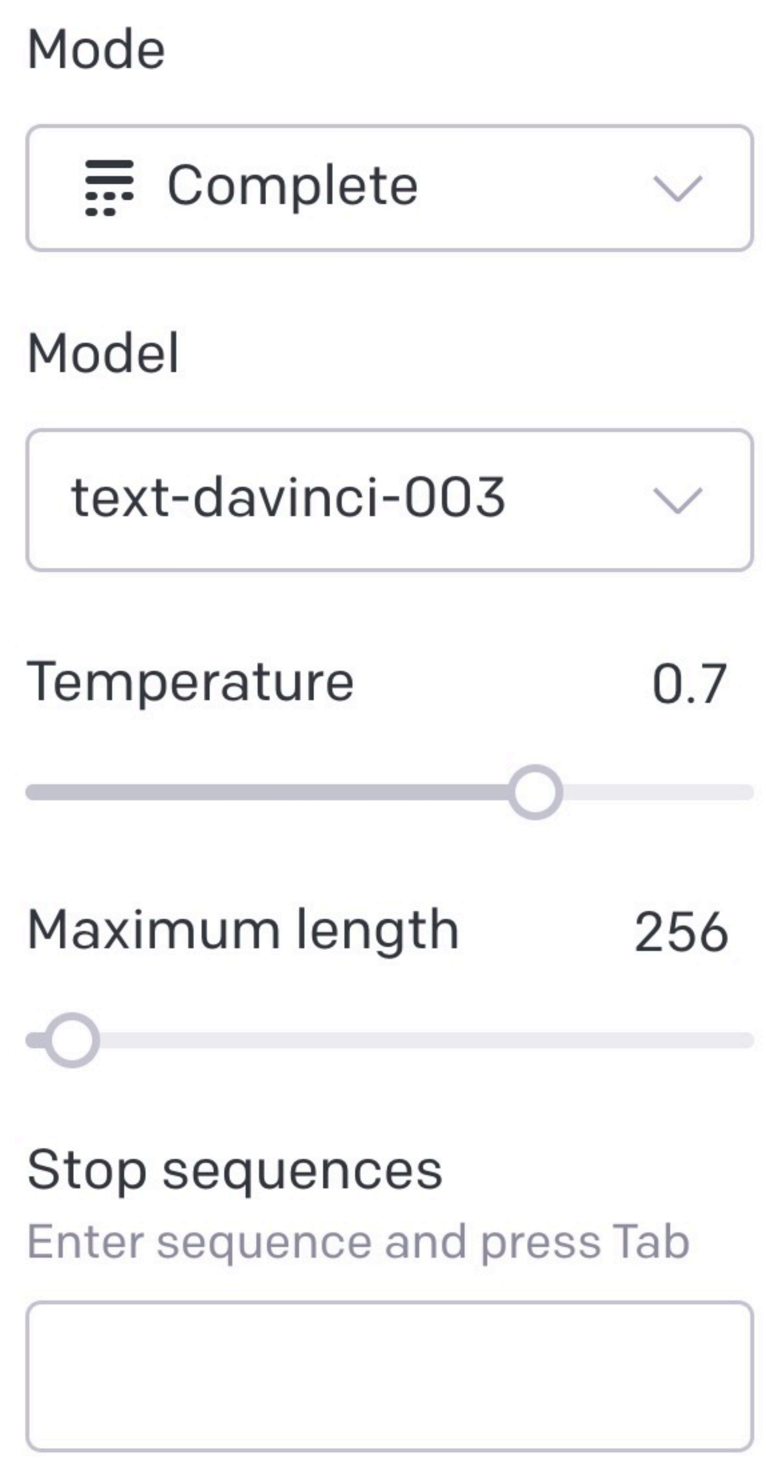
https://openai.com/contributions/gpt-4

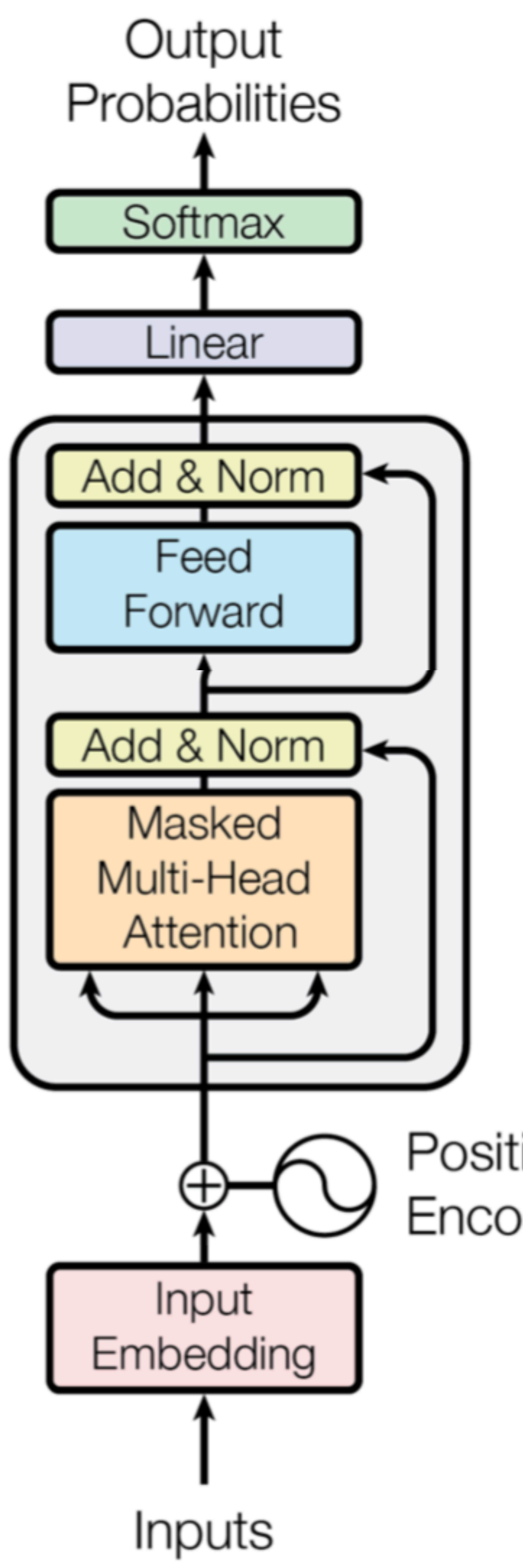
Considerations for LLM inference

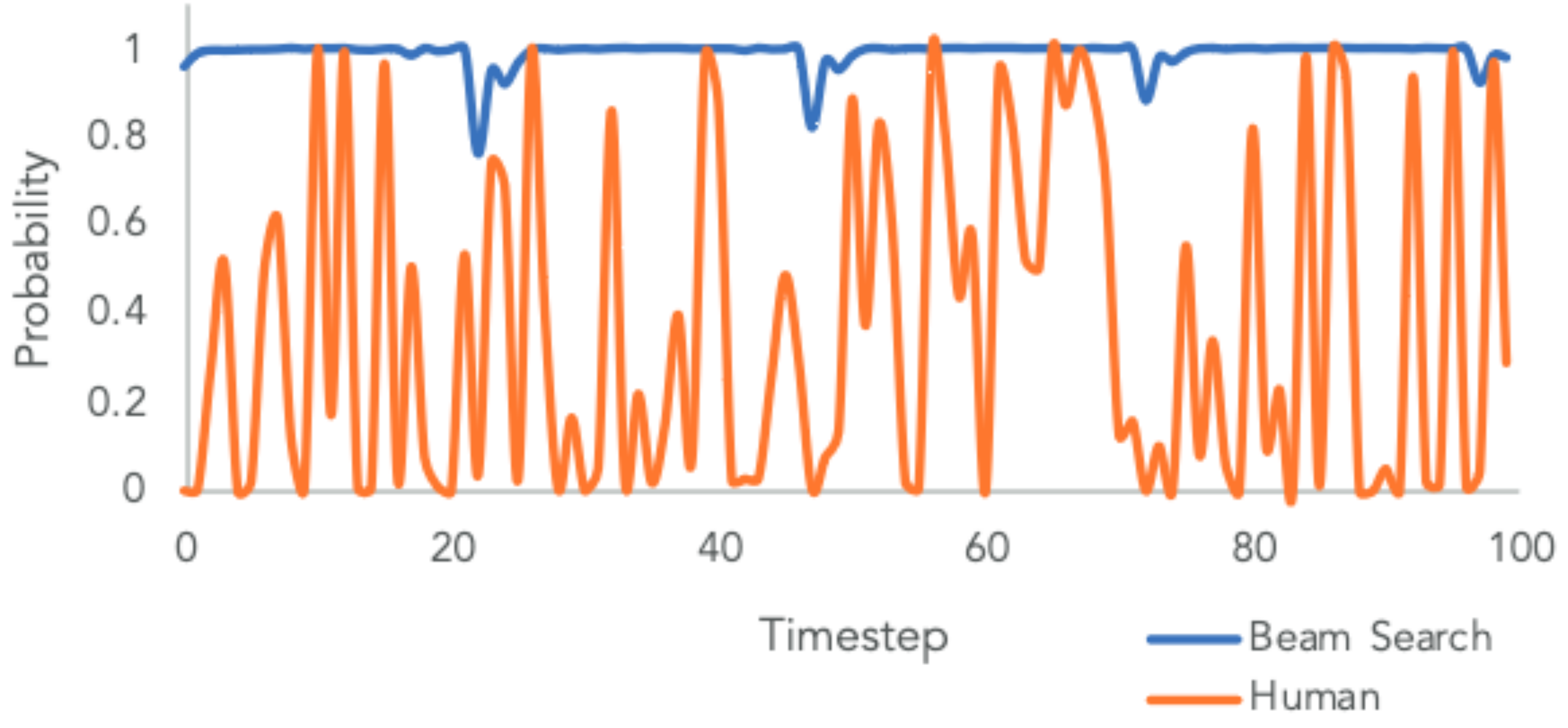
- Understanding auto-regressive sampling
- Improving (or not) runtime complexity
- Dealing with large model size

Auto-regressive Sampling

- Remember that we sample tokens one at a time
  - [It's, a, blue, ...
- The softmax outputs a peaky probability distribution over possible next tokens
- Temperature parameter makes it less peaky
  - t=0 will always sample the most likely next token;
  - t=1 will often sample less-likely ones
- Human text is not all high-probability next words!



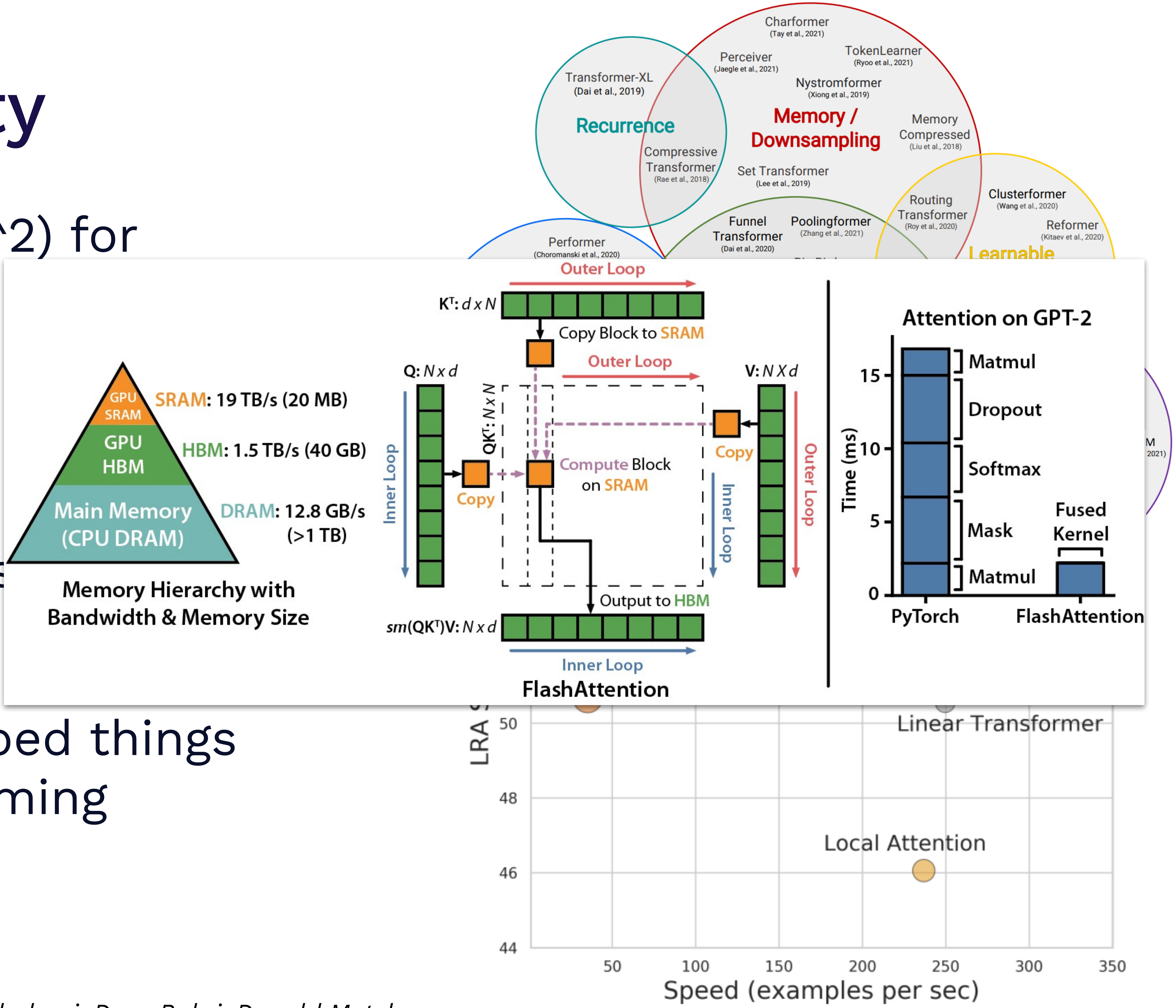




https://arxiv.org/abs/1904.09751

Runtime Complexity

- Self-attention runs in O(N^2) for sequence length N
- Many O(N) approximations developed
- But none have provided a s
improvement
- Recently, FlashAttention sped things up via smart GPU programming



Based on "Efficient Transformers: A Survey" by Yi Tay, Mostafa Dehghani, Dara Bahri, Donald Metzler
and "Long Range Arena: A Benchmark for Efficient Transformers" by Y Tay, M Dehghani, S Abnar, Y Shen, D Bahri, P Pham, J Rao, L Yang, S Ruder, D Metzler

http://lucasb.eyer.be/transformer *https://github.com/HazyResearch/flash-attention*

Dealing with Large Model Sizes

- Large subject! Lilian Weng from OpenAI has a thorough post
- Quantization is most relevant to us
  - LLM weights are usually in float32 or 16
  - Recent work (LLM.int8) has shown that 8-bit post-quantization is basically fine
  - Even 4-bit seems fine!

The case for 4-bit precision:
k-bit Inference Scaling Laws

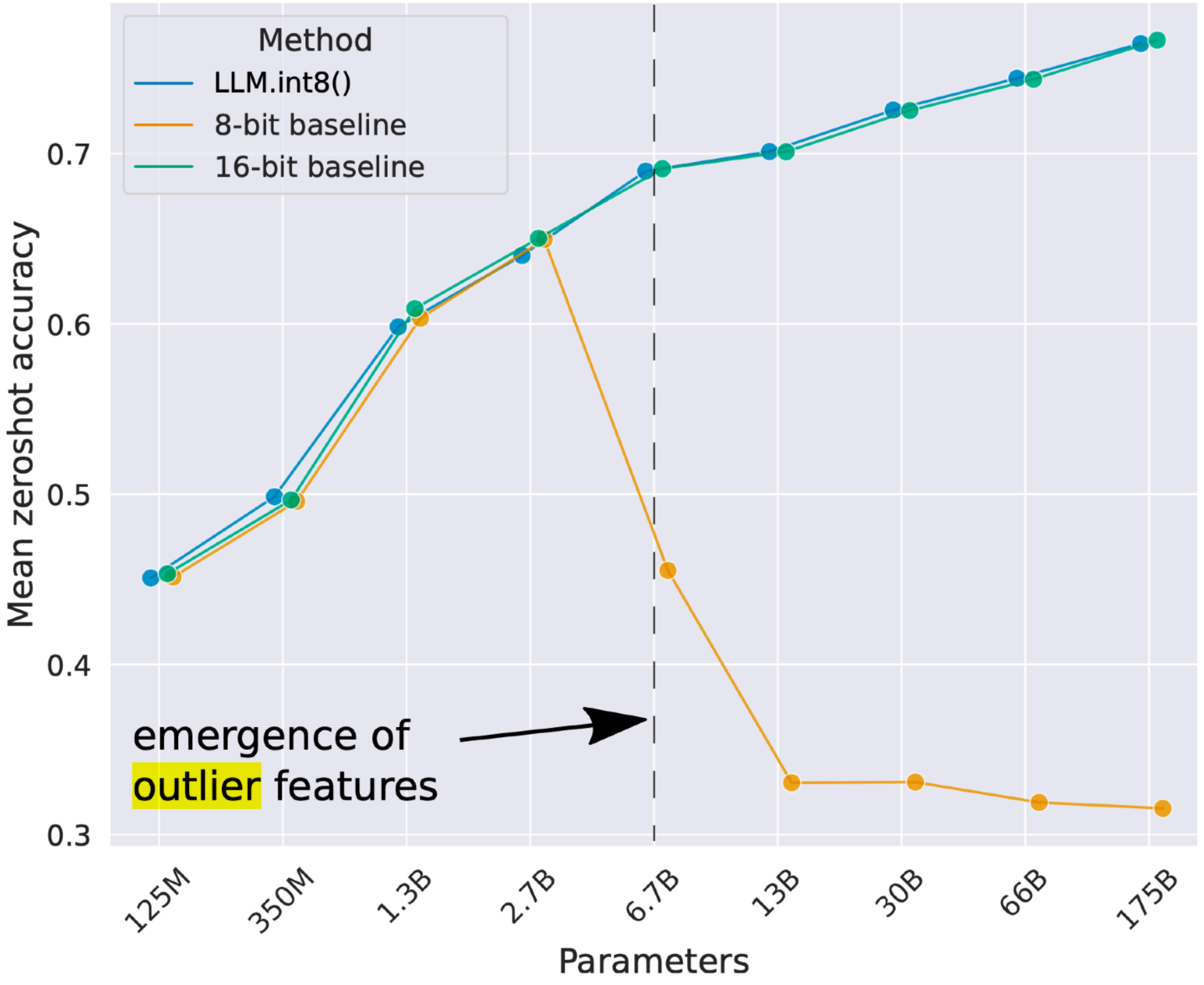
Tim Dettmers [1] **Luke Zettlemoyer** [1]

https://arxiv.org/pdf/2212.09720.pdf

LLaMA Quantization

model	original size	quantized size (4-bit)
7B	13 GB	3.9 GB
13B	24 GB	7.8 GB
30B	60 GB	19.5 GB
65B	120 GB	38.5 GB

https://github.com/ggerganov/llama.cpp



https://arxiv.org/pdf/2208.07339.pdf

Resources

- Megatron-LM (GitHub): probably still the best insights into training LLMs at scale.
- OpenAI post Techniques for Training Large Neural Networks
- Lillian Weng's "Large Transformer Model Inference Optimization"

Questions?

Questions?

Thanks!

@sergeykarayev

@full_stack_dl

`/imagine green tropical parrot eating stack of pancakes, flapjack breakfast, hyper-realistic portrait, DSLR Canon R5, chromatic aberration, accent lighting, super resolution, hyper-detailed, cinematic, OpenGL - Shaders`