



Large Language Models

Foundation Models from the Ground Up



Course Outline

Course Introduction

Module 1 – Transformers: Attention and the Transformer Architecture

Module 2 – Efficient Fine-Tuning: Doing more with less

Module 3 – Deployment Optimizations: Improving model size and speed

Module 4 – Multi-modal LLMs: Beyond text-based transformers



Course Outline

Course Introduction

Module 1 – Transformers: Attention and the Transformer Architecture

Module 2 – Efficient Fine-Tuning: Doing more with less

Module 3 – Deployment Optimizations: Improving model size and speed

Module 4 – Multi-modal LLMs: Beyond text-based transformers



Course Introduction



What are LLMs?



Matei Zaharia

Co-founder & CTO of Databricks

Associate Professor of Computer Science
at UC Berkeley

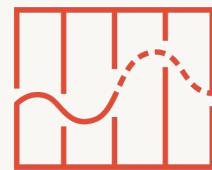


Generative AI state of the art is rapidly advancing

No single model to rule them all—trade-offs are required to find the best model for each use case



Privacy



Quality



Cost



Latency

Proprietary LLMs



ChatGPT



PaLM 2



ANTHROPIC



OpenAI

Open Source LLMs



databricks
Dolly



Hugging Face

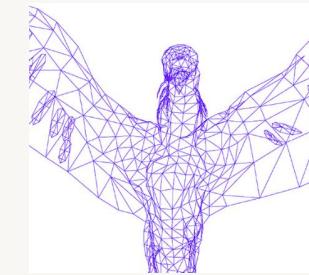
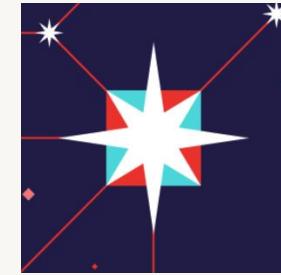


mosaic^{ML}
MPT

stability.ai
Stable Diffusion

Open Source quality is rapidly advancing – while fine tuning cost is rapidly decreasing

Dolly started the trend to open models with a commercially friendly license



Facebook Llama

Stanford Alpaca

Databricks Dolly

Mosaic MPT

TII Falcon

"Smaller, more performant models such as Llama ... democratizes access in this important, fast-changing field."

February 24, 2023

"Alpaca behaves qualitatively similarly to OpenAI ... while being surprisingly small and easy /cheap to reproduce"

March 13, 2023

"Dolly will help democratize LLMs, transforming them into a commodity every company can own and customize"

March 24, 2023

"MPT-7B is trained from scratch on 1T tokens ... is open source, GPT-3 for ... 75% of the training available for commercial use, and compute budget—and ... a fifth of matches the quality of LLaMA-7B" the compute at inference time."

May 5, 2023

May 24, 2023

Non Commercial Use Only | **Commercial Use Permitted**

OSS LLMs are getting better everyday

🌟 Open LLM Leaderboard

⚠️ The 🌟 Open LLM Leaderboard aims to track, rank and evaluate LLMs and chatbots as they are released.

💡 Anyone from the community can submit a model for automated evaluation on the 🌟 GPU cluster, as long as it is a 🤗 Transformers model with weights on the Hub. We also support evaluation of models with delta-weights for non-commercial licensed models, such as the original LLaMa release.

Other cool benchmarks for LLMs are developed at HuggingFace, go check them out: 🐱💡 [human and GPT4 evals](#), 🚧 [performance benchmarks](#)

- Constant development
- Supported by community and industry
- Rapid innovation cycle

T	Model	Average	HellaSwag	MMLU	TruthfulQA	#Params (B)
●	meta_llama/Llama-2-70b-hf	67.3	87.3	69.8	44.9	70
●	huggyllama/Llama-65b	64.2	86.1	63.9	43.4	65.286
●	llama-65b	64.2	86.1	63.9	43.4	65
●	tiiuae/falcon-40b	61.5	85.3	57	41.7	40
●	llama-30b	61.7	84.7	58.5	42.3	30
●	mosaicml/mpt-30b	56.2	82.4	47.9	38.4	30
●	TheBloke/Llama-2-13B-fp16	58.6	82.2	55.7	37.4	13
●	meta_llama/Llama-2-13b-hf	58.7	82.1	55.8	37.4	13
●	llama-13b	56.1	80.9	47.7	39.5	13
●	huggyllama/Llama-13b	56	80.9	47.6	39.5	13.016
●	dvxuette/Llama-13b-pretrained-sft-do2	58.5	80.3	47.2	47.4	13
●	dvxuette/Llama-13b-pretrained-sft-epoch-1	56.8	80	45.5	44.5	13
●	dvxuette/Llama-13b-pretrained	57.8	79.3	47	48.4	13
●	dvxuette/Llama-13b-pretrained-dropout	57.7	79.3	46.6	48.6	13
●	meta_llama/Llama-2-7b-hf	54.4	78.6	46.9	38.8	7
●	tiiuae/falcon-7b	47	78.1	27.8	34.3	7
●	llama-7b	49.7	77.8	35.7	34.3	7

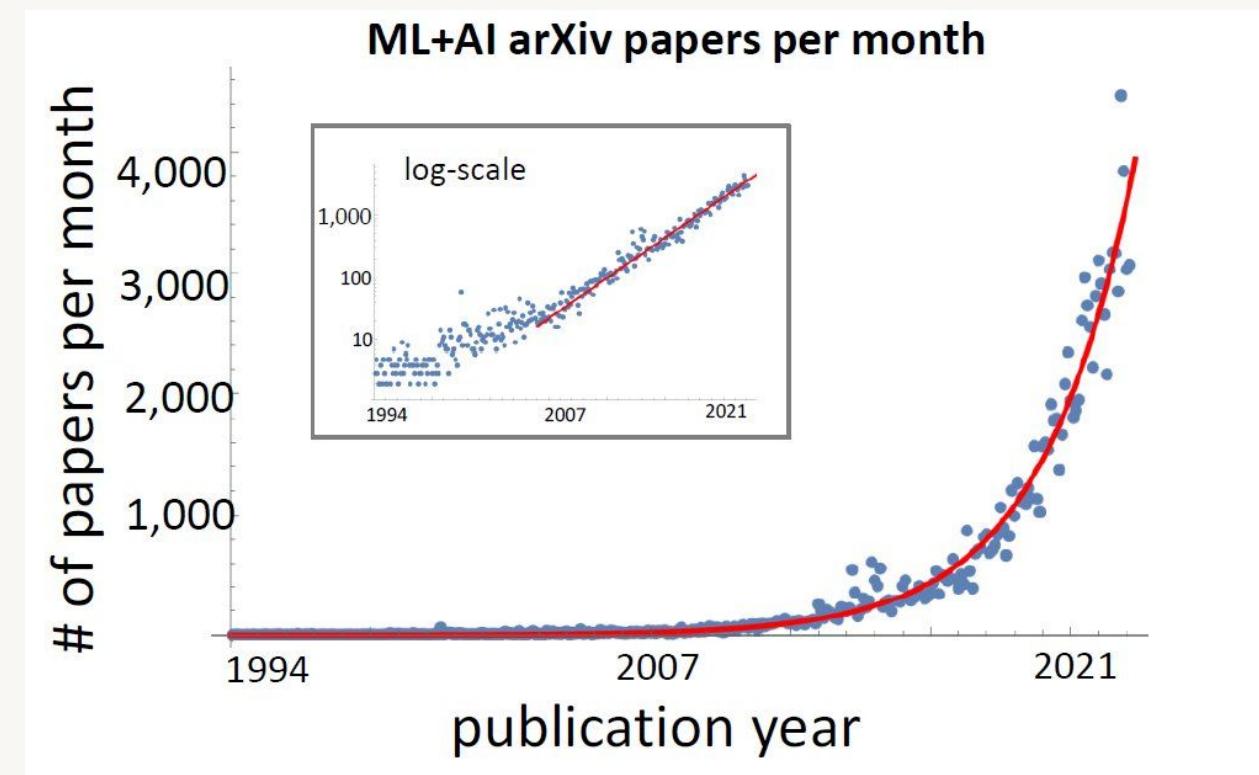
“A strong foundation... is all you need”

What and who this course is for

If you want to keep up:

- The fundamentals of LLMs have not changed since 2018
- Most of the innovation are variations of the original.

Research and innovation has exploded around LLMs.



Source: [Reddit](#)



Enjoy the course!



Course Outline

Course Introduction

Module 1 – Transformers: Attention and the Transformer Architecture

Module 2 – Parameter Efficient Fine-Tuning: Doing more with less

Module 3 – Deployment Optimizations: Improving model size and speed

Module 4 – Multi-modal LLMs: Beyond text-based transformers



Module 1

Transformers

Attention and the Transformer Architecture



Learning Objectives

By the end of this module you will:

- Describe and build the inner workings of a transformer model
- Compare and contrast the different types of transformer architectures
- Recognize the importance and mechanics of attention
- Apply the different types of transformer models to solve problems



A Transformative Technology

A breakthrough in Natural Language Processing

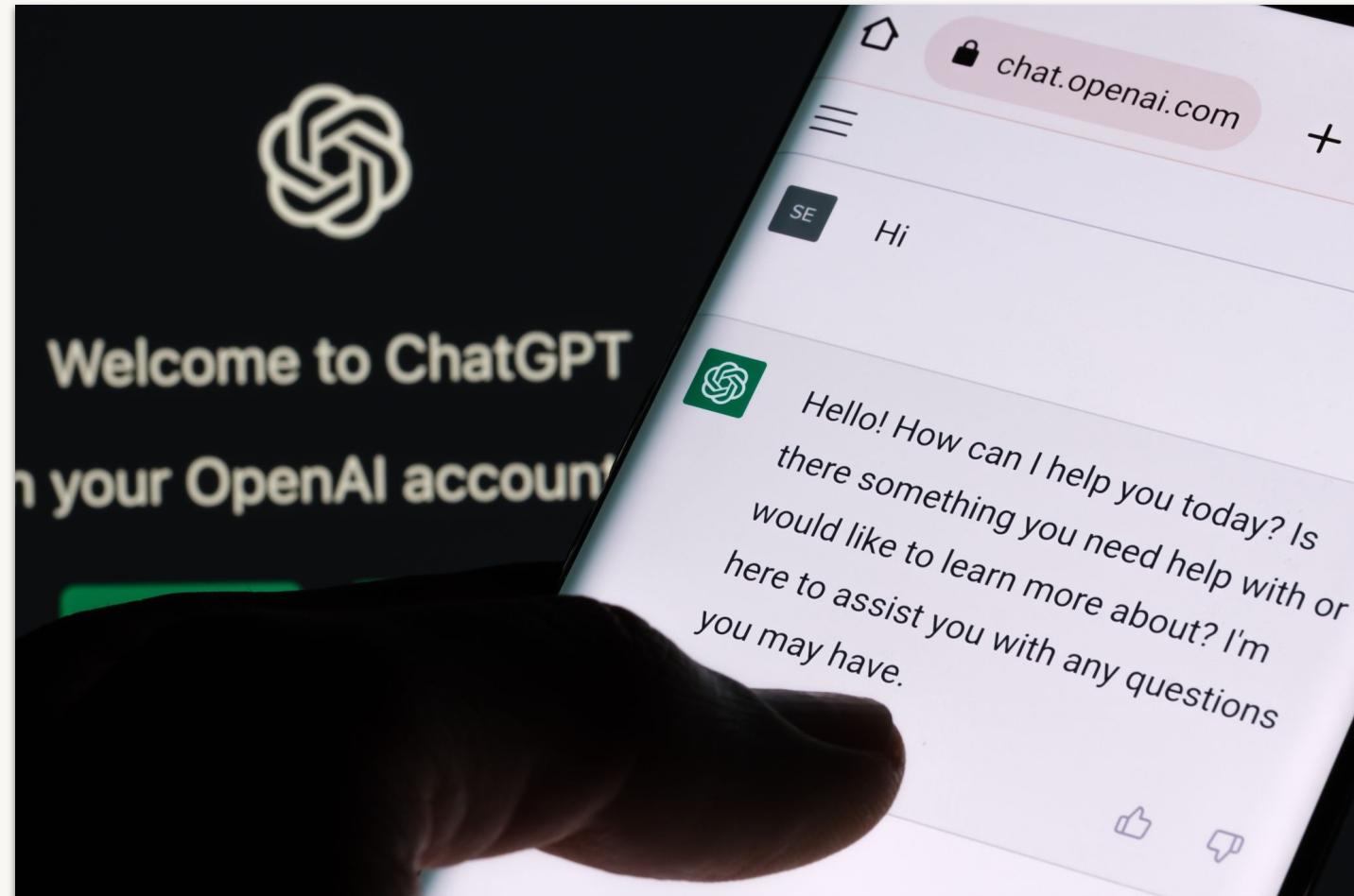


Let's Chat

A new technology paradigm has arrived.

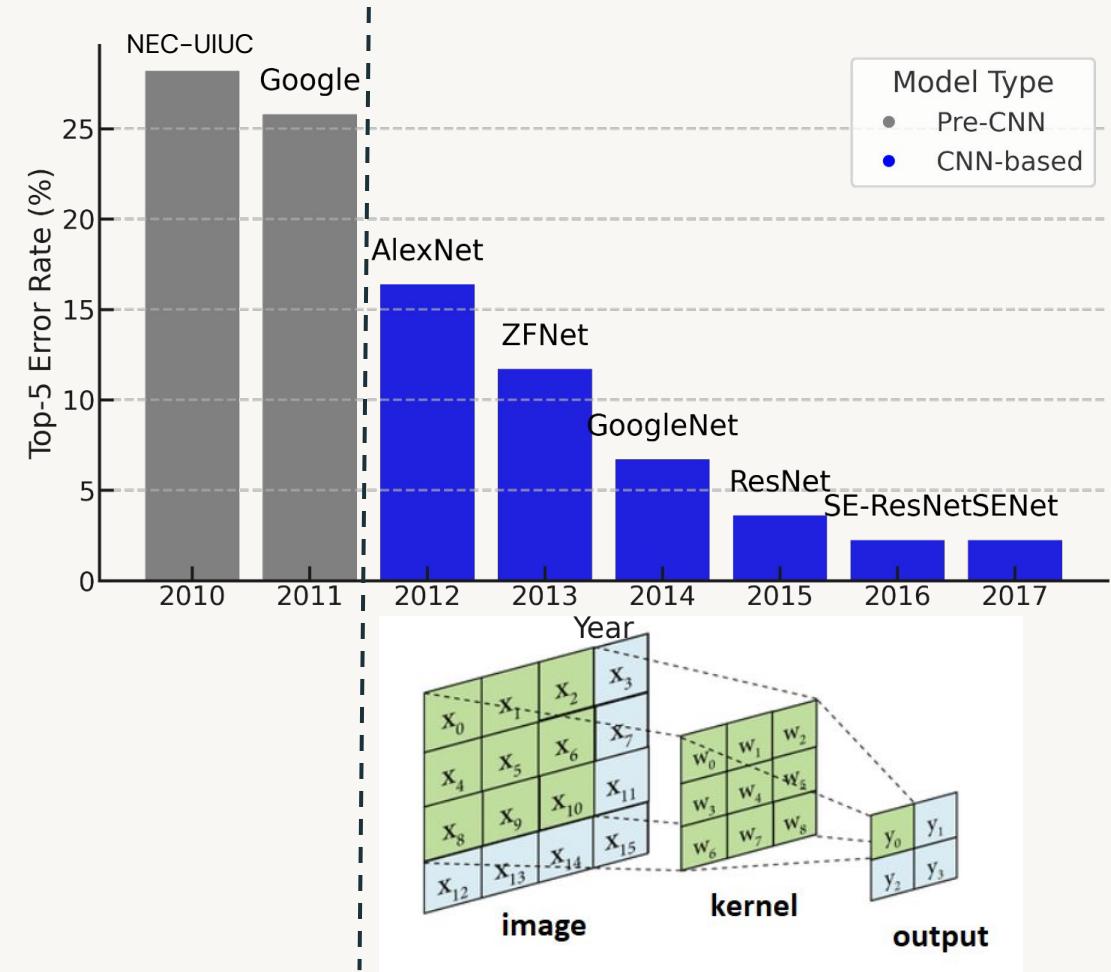
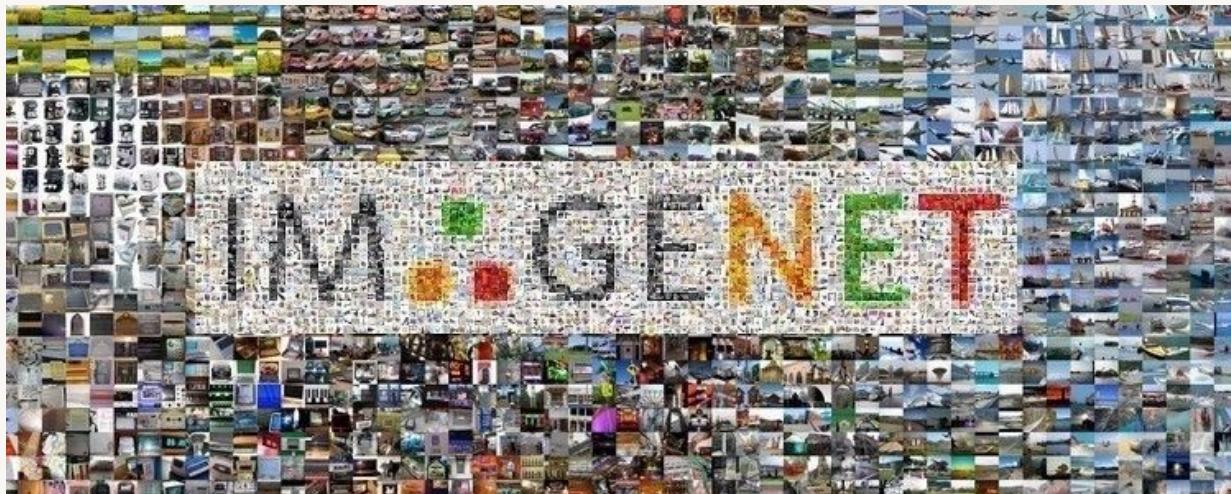
Less than a year old, ChatGPT is the fastest adopted technology in human history.

LLMs like ChatGPT have ushered in a new **golden era of AI**.



We've seen this before

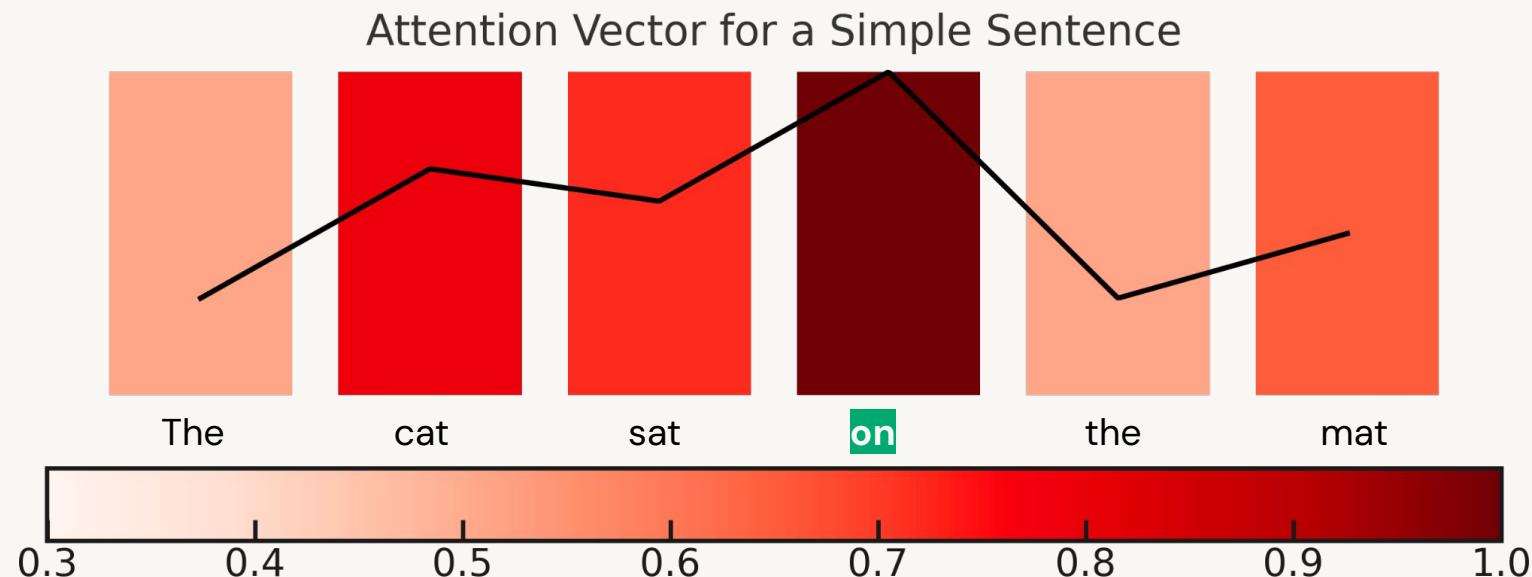
The Convolution Layer led to models that dominated the vision field



We weren't paying attention before

Attention: "the convolution layer of language"

The attention mechanism measures how words interrelate



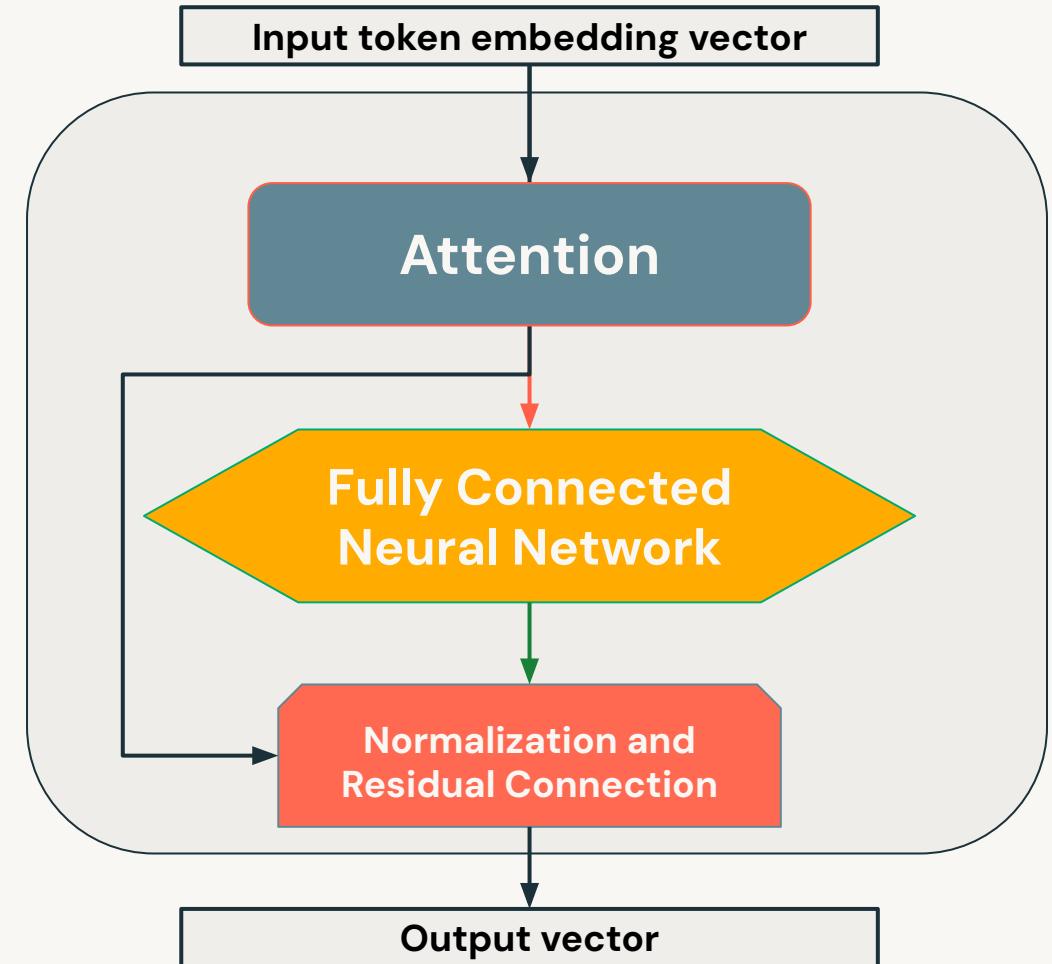
Attention is (not) all you need

A new type of deep learning architecture

Attention is a linear, matrix operation.

Where is the neural network?

Let's look at **The Transformer Block**



The Transformer Block

Building up the most important AI advancement in years

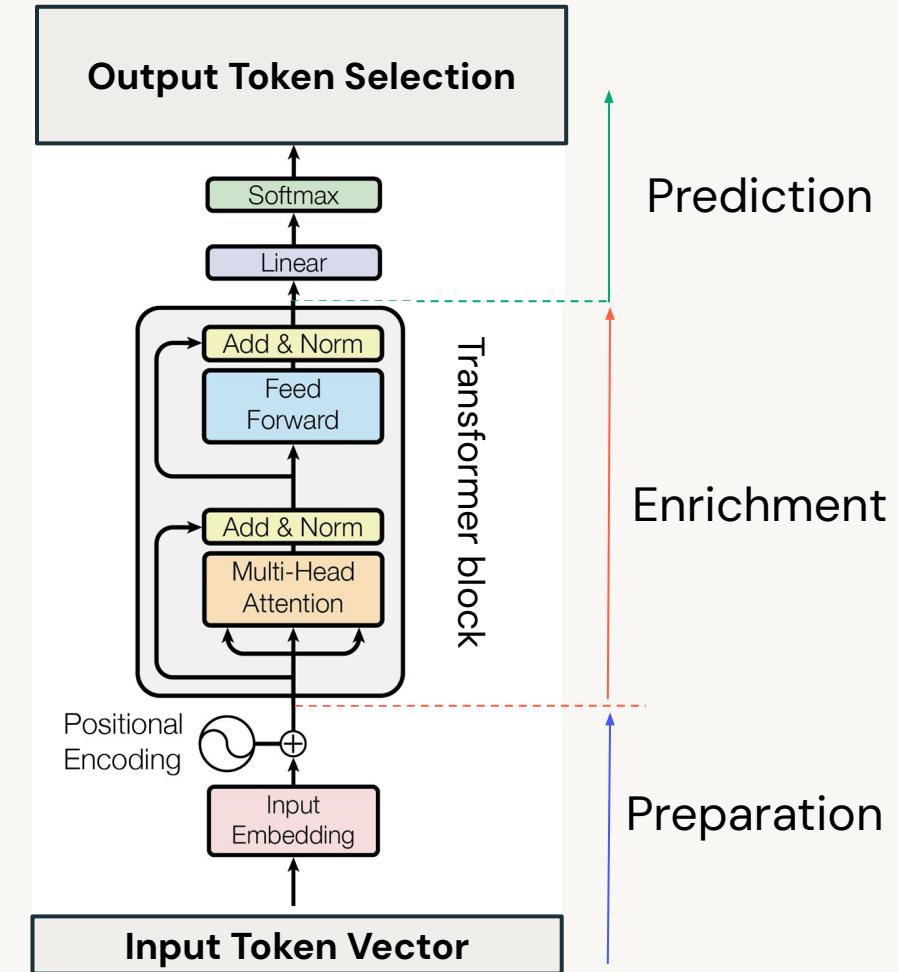


Transforming a sequence

What is the goal of a transformer block?

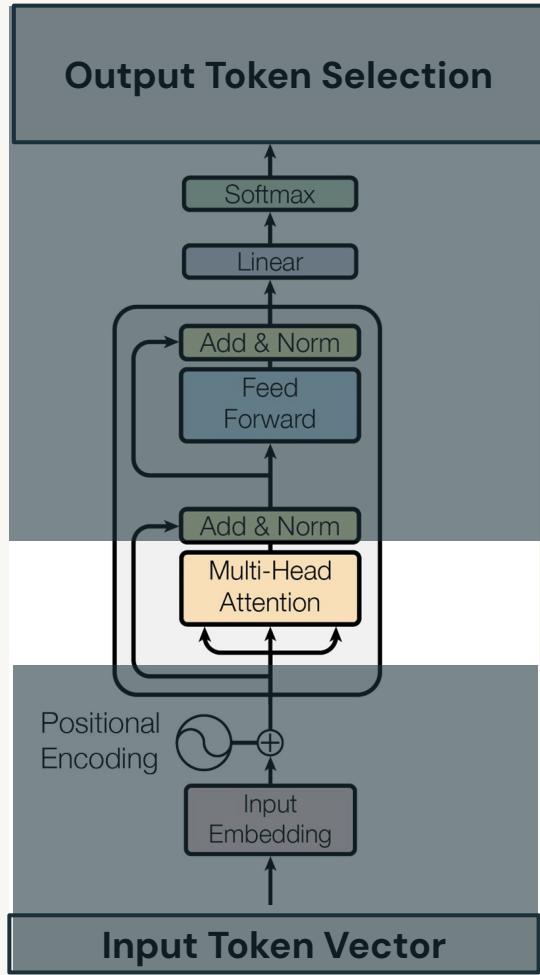
Transformer blocks:

1. Process input vectors with attention to enrich
2. Add nonlinear transformation with NN
3. Apply enriched vectors to select the correct token from the model's vocabulary



Attention Mechanism

How important is each word to each other?



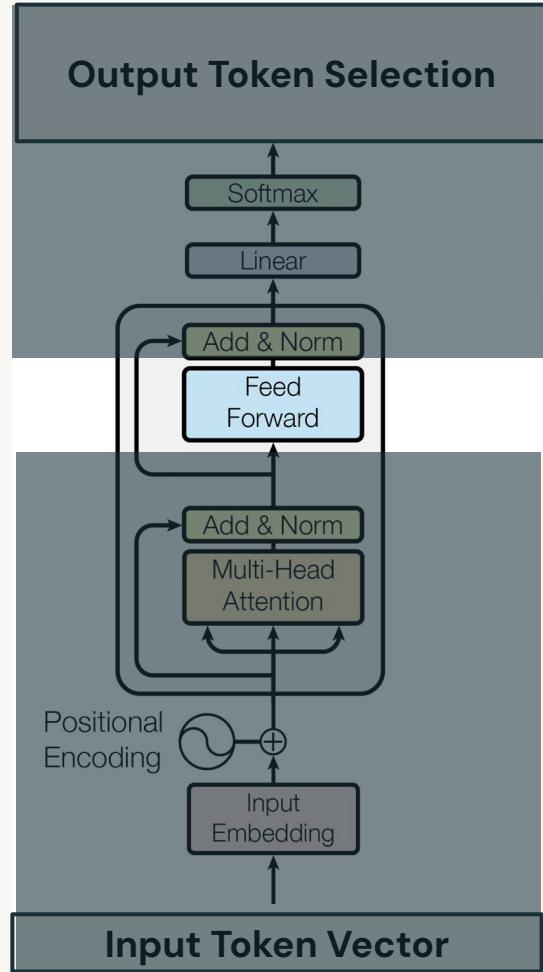
The role of **attention** is to:

- Measure the importance and relevance of each word relative to each other
- Allow for the building-up of enriched vectors with more context and logic



Position-wise Feed-Forward Networks

Adding nonlinearity to understanding the input.



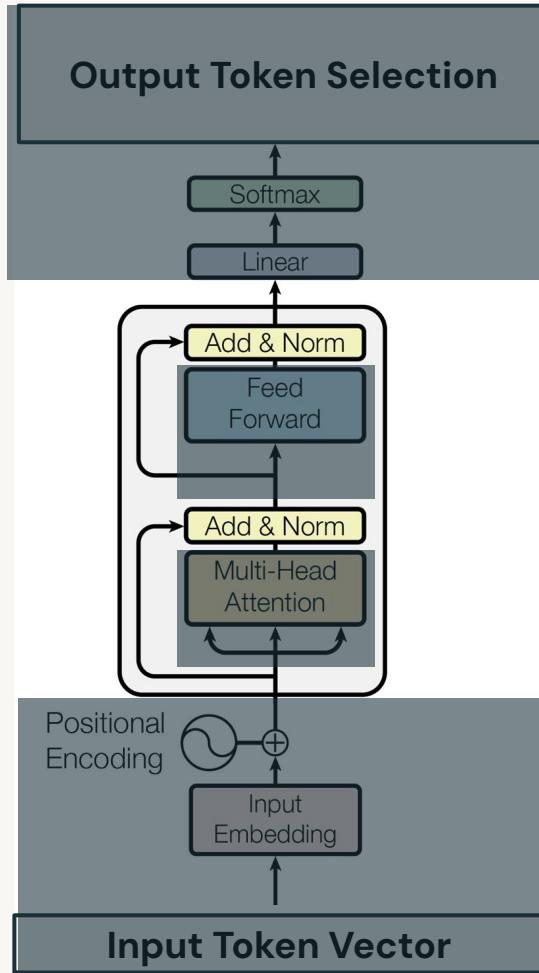
The role of Position-wise Feed-Forward Network is to:

- Use the information gathered from the attention stage for each element in the sequence and translate it to a further enriched state
- Allow for nonlinear transformations of each element in the sequence and builds upon itself in subsequent layers



Residual Connections & Layer Normalization

Ensuring robust and reliable training.



Residual Connections

- Shortcuts in the network that allow information to flow directly from earlier layers to later layers.
- Help mitigate the vanishing gradient problem in deep neural networks.

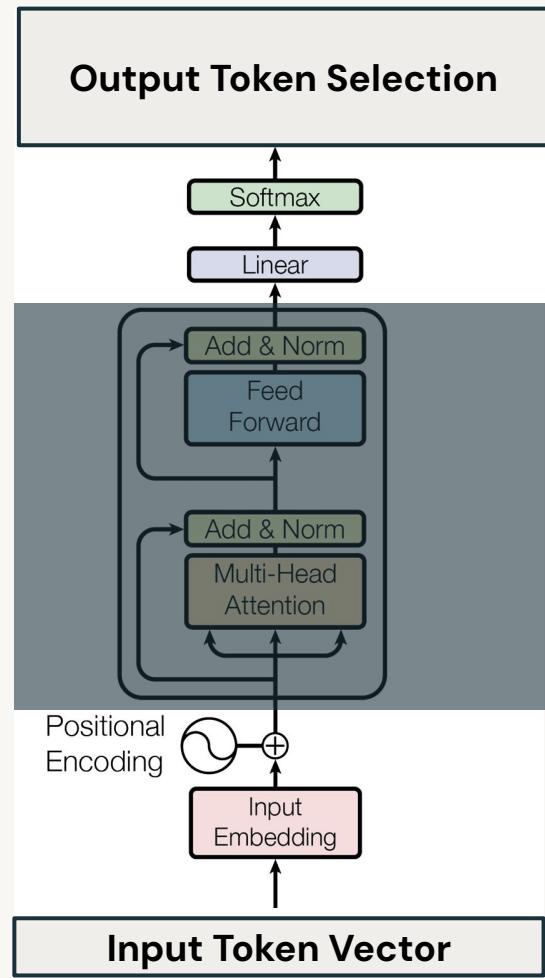
Layer Normalization

- Normalizes the inputs across the features rather than the batch.
- Stabilizes the network's training, ensuring consistent input distribution, crucial in tasks with varying sequence lengths.



Into and out of the transformer

Input transformations and output transformations



Output from the Transformer

- Exiting the last Transformer block, is a sequence of context-aware vectors.
- Each vector in this sequence represents an input token, but now its representation is deeply influenced by its interactions with all other tokens in the sequence.
- Different models use the output from the transformer differently.

Input to the Transformer

- The initial input to a Transformer model is a sequence of word tokens.
- These tokens are converted into word embeddings.
- Positional encodings are added to these embeddings, providing the model with necessary positional information.
- This new sequence, with each element a dense vector, is then passed to the first transformer block.

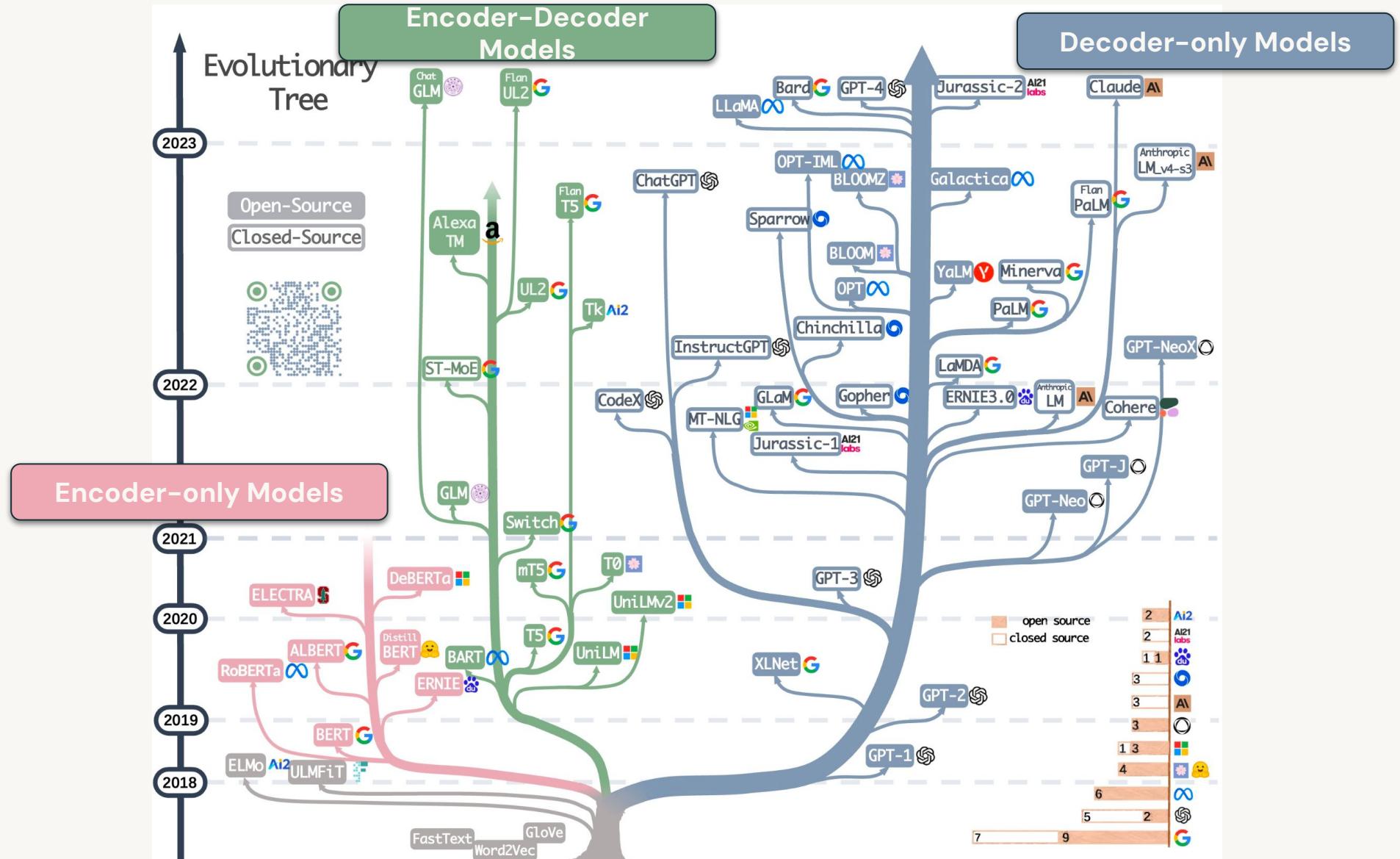


Transformer Architectures

“Encoders, decoders, T5, oh my!”



The Transformer Family Tree

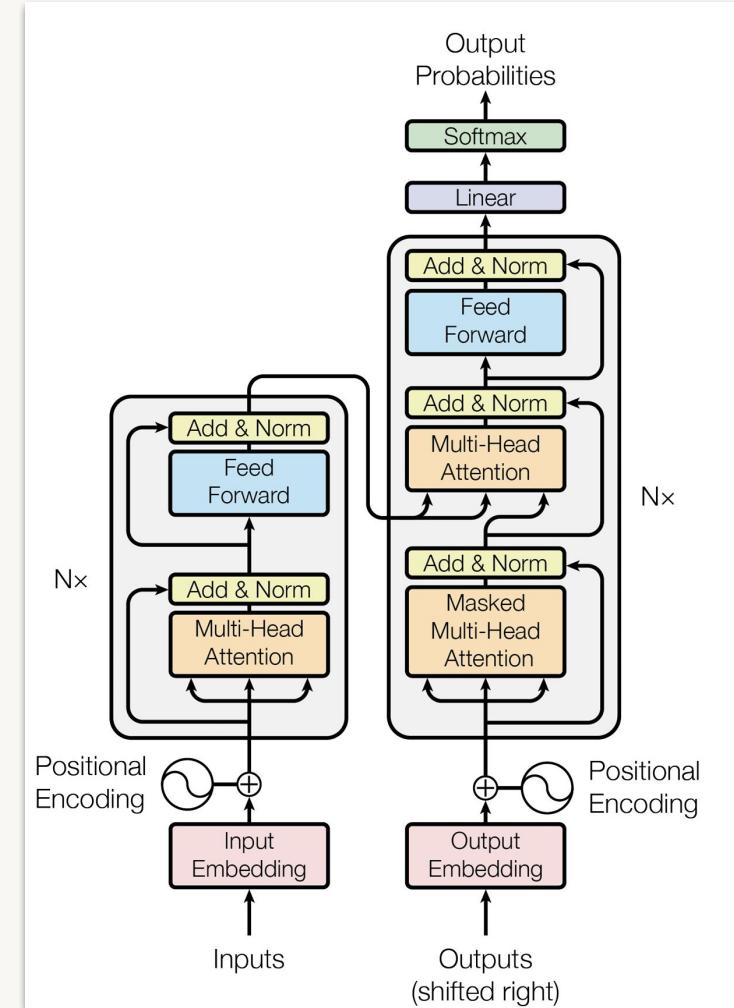


The Original Transformer

An encoder-decoder model

Defining Features:

- Two sets of transformer blocks
 - Encoder blocks
 - Decoder blocks
 - Cross-attention
- Typical Uses:
 - Translation
 - Conversion



Source: [ArXiv](#)

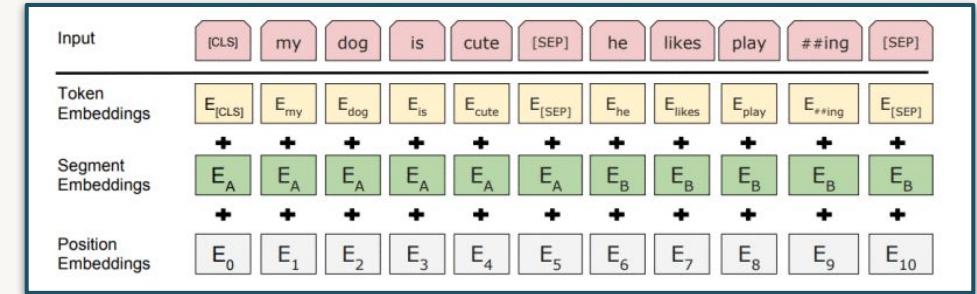


B is for BERT

Encoder-only models

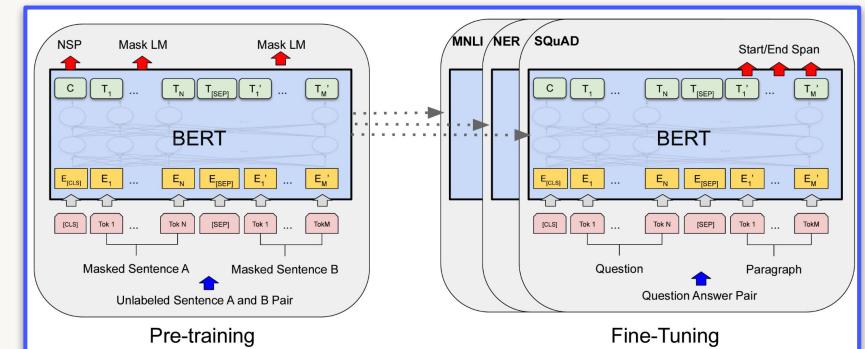
Novel Features of BERT:

- Segment embeddings ([CLS] and [SEP] special tokens)
- Trained with Masked Language Modeling & Next Sentence Prediction
- Excellent Fine-Tuning Performance



Input = [CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
penguin [MASK] are flight ##less birds [SEP]
Label = NotNext

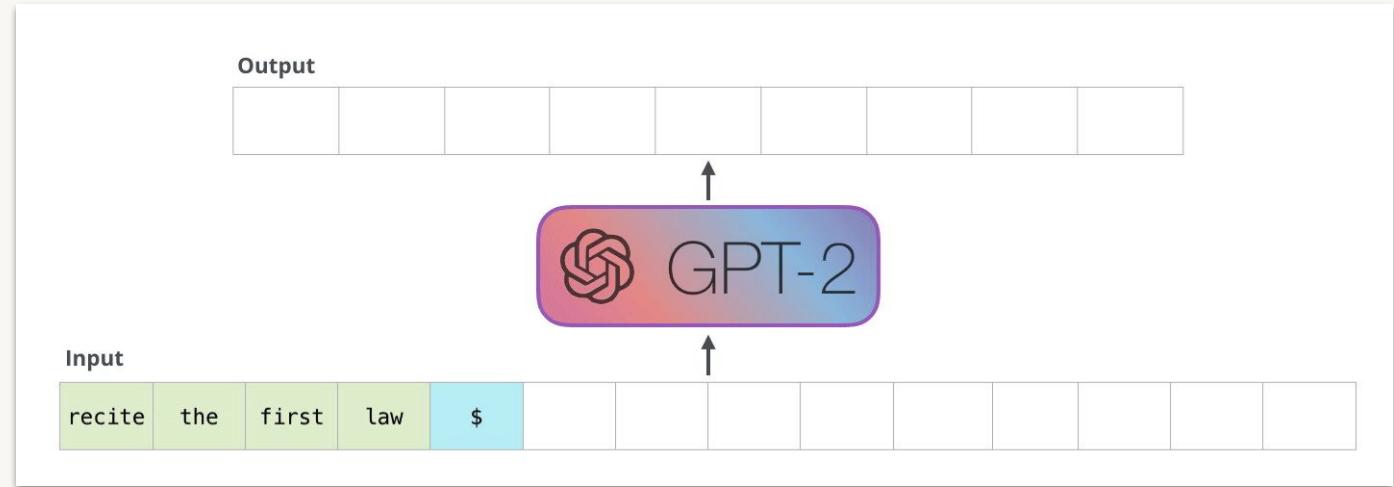


Generating text with GPT

Decoder-only models

Decoder models have a single task:

- Look over the sequence and predict the next word (or token).



Decoder models products: **ChatGPT, Bard, Claude, LLaMA, MPT**

Source: [IllustratedTransformer](#)



Important variables in Transformers

Input	Internal	Training
<ul style="list-style-type: none">Vocabulary Size (V): The number of unique tokens that the model recognizes.Embedding/Model Size (D): The dimensionality of the word embeddings, also known as the hidden size.Sequence/Context Length (L): The maximum number of tokens that the model can process in a single pass.	<ul style="list-style-type: none">Number of Attention Heads (H): In the multi-head attention mechanism, the input is divided into H different parts.Intermediate Size (I): The feed-forward network has an intermediate layer whose size is typically larger than the embedding size.Number of Layers (N): The number of Transformer blocks/layers.	<ul style="list-style-type: none">Batch Size (B): The number of examples processed together in one forward/backward pass during training.Tokens Trained on (T): The total number of tokens that a model sees during training. This is normally reported more than the number of epochs.



Time to Pay Attention

The secret that unlocked the power of LLMs

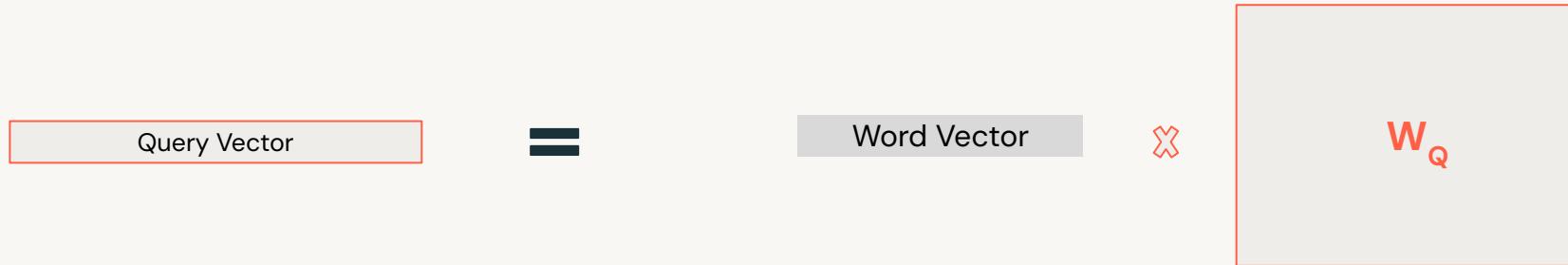


The inner workings of attention

Learning the weights of attention.

We use three, large (millions of elements), matrices to create the Query, Key, and Value vectors in each layer.

Query Matrix: \mathbf{W}_Q , Key Matrix: \mathbf{W}_K , Value Matrix: \mathbf{W}_V



The Attention equation can be expressed as an operation over all the vectors:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

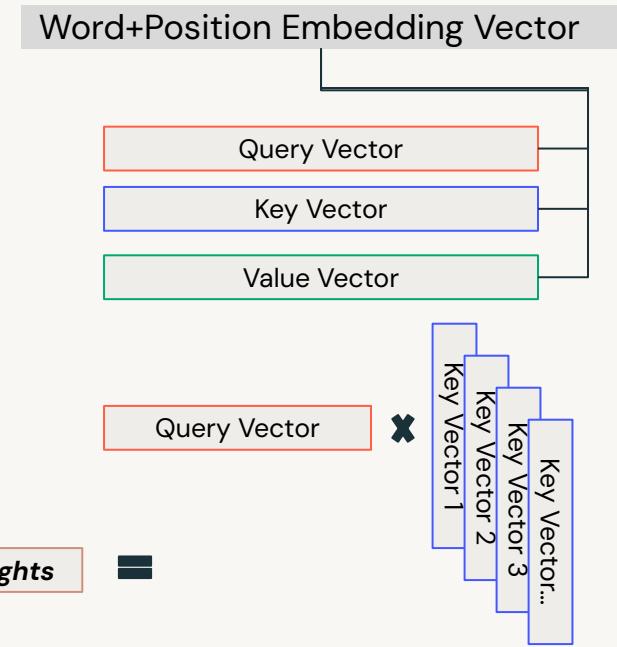


The inner workings of attention

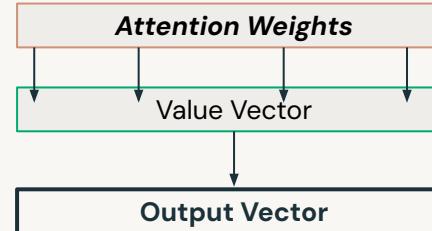
How do we calculate attention?

The mechanism can be broken down into three steps:

- 1) The input vector (in the first layer, this is the word embedding vector (with the position information), is used to create three new vectors:
 - a) the Query (Q) vector - made from the current token
 - b) the Key (K) vector - made from all other tokens in the sequence
 - c) the Value (V) vector - made from all tokens in the sequence
- 2) A scaled dot-product is then performed on the Query and Key vectors, this is the attention score. This also uses a softmax function to produce a final set of **Attention Weights** that are scaled 0-1.
- 3) Each Value vector is multiplied by its corresponding attention weight and then all are summed up to generate the output for the self-attention layer.



Attention Weights



Building Base/Foundation Models

Training transformers, what does it take?



Foundation Model Training – Getting Started

Choosing the right options to build your model.

A **foundation**, or **base**, model is a transformer that is trained from scratch.

- Model Architecture – decoder? encoder-decoder?
 - Tailored to the task/problem
 - Different sizes: embedding dimensions, number of transformer blocks, etc.
- Available Data
 - Task-relevant data
 - Good language coverage
- Available Compute Resources
 - Time to allow for training
 - Hardware available



Foundation Model Training – Architecture

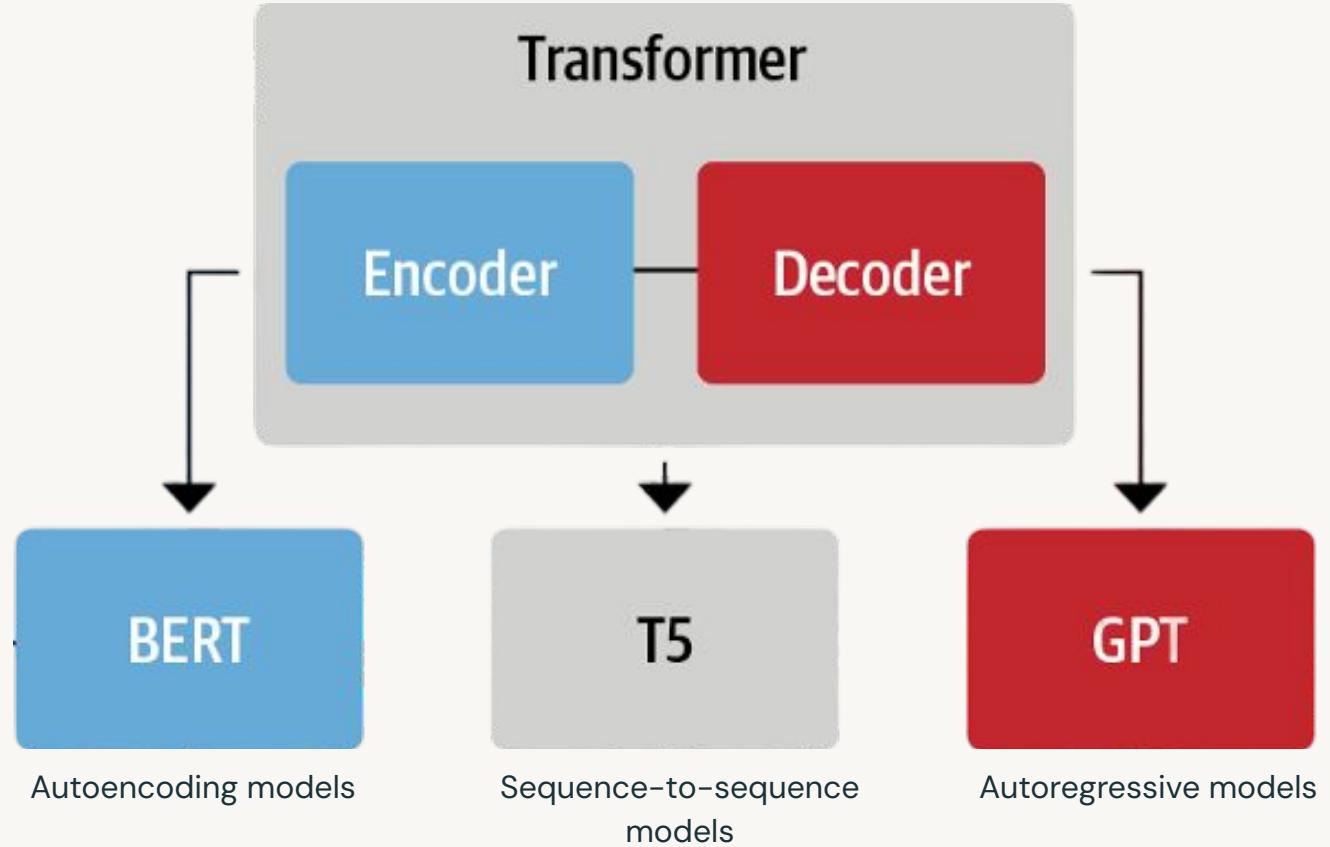
Which transformer flavor is right?

Start with your task:

- Classification?
- Generation?
- Translation?

Architecture:

- Layer count
- Context size



Source: [GitHub](#)



Foundation Model Training - Data

It's all about the data

Datasets for foundation LLM:

- Web Text
- Code
- Images
- Digitized Text
- Transcriptions

Component	Raw Size	Weight	Epochs	Effective Size	Mean Document Size
Pile-CC	227.12 GiB	18.11%	1.0	227.12 GiB	4.33 KiB
PubMed Central	90.27 GiB	14.40%	2.0	180.55 GiB	30.55 KiB
Books3 [†]	100.96 GiB	12.07%	1.5	151.44 GiB	538.36 KiB
OpenWebText2	62.77 GiB	10.01%	2.0	125.54 GiB	3.85 KiB
ArXiv	56.21 GiB	8.96%	2.0	112.42 GiB	46.61 KiB
Github	95.16 GiB	7.59%	1.0	95.16 GiB	5.25 KiB
FreeLaw	51.15 GiB	6.12%	1.5	76.73 GiB	15.06 KiB
Stack Exchange	32.20 GiB	5.13%	2.0	64.39 GiB	2.16 KiB
USPTO Backgrounds	22.90 GiB	3.65%	2.0	45.81 GiB	4.08 KiB
PubMed Abstracts	19.26 GiB	3.07%	2.0	38.53 GiB	1.30 KiB
Gutenberg (PG-19) [†]	10.88 GiB	2.17%	2.5	27.19 GiB	398.73 KiB
OpenSubtitles [†]	12.98 GiB	1.55%	1.5	19.47 GiB	30.48 KiB
Wikipedia (en) [†]	6.38 GiB	1.53%	3.0	19.13 GiB	1.11 KiB
DM Mathematics [†]	7.75 GiB	1.24%	2.0	15.49 GiB	8.00 KiB
Ubuntu IRC	5.52 GiB	0.88%	2.0	11.03 GiB	545.48 KiB
BookCorpus2	6.30 GiB	0.75%	1.5	9.45 GiB	369.87 KiB
EuroParl [†]	4.59 GiB	0.73%	2.0	9.17 GiB	68.87 KiB
HackerNews	3.90 GiB	0.62%	2.0	7.80 GiB	4.92 KiB
YoutubeSubtitles	3.73 GiB	0.60%	2.0	7.47 GiB	22.55 KiB
PhilPapers	2.38 GiB	0.38%	2.0	4.76 GiB	73.37 KiB
NIH ExPorter	1.89 GiB	0.30%	2.0	3.79 GiB	2.11 KiB
Enron Emails [†]	0.88 GiB	0.14%	2.0	1.76 GiB	1.78 KiB
The Pile	825.18 GiB			1254.20 GiB	5.91 KiB

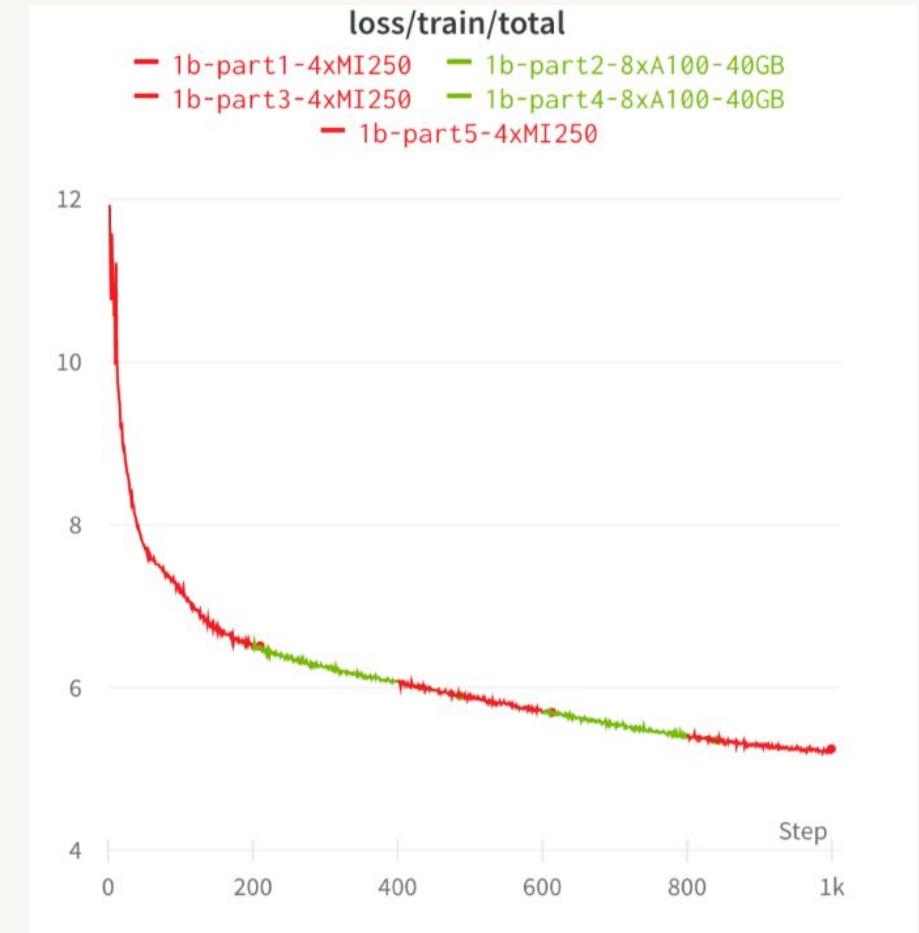
Source: [Stanford CS324](#)



Foundation Model Training - Training

Optimizing LLM Losses

- LLMs train just like other deep learning models.
- Loss functions are typically **cross-entropy**.
- Familiar optimizers like **AdamW**, are used.
- Training LLMs with 100's of billions of parameters, can take **months on hundreds** of GPUs.



Source: [MosaicML](#)



Now what?

What do you use a foundation LLM for?

Foundation LLMs suffer from “alignment” problems:

- Bias/toxicity in the training data
- Lack of specific task focus -> for models like GPT it just focuses on selecting the next correct word

Fine tuning and other methods are required to ensure task success.



Generative Pretrained Transformer

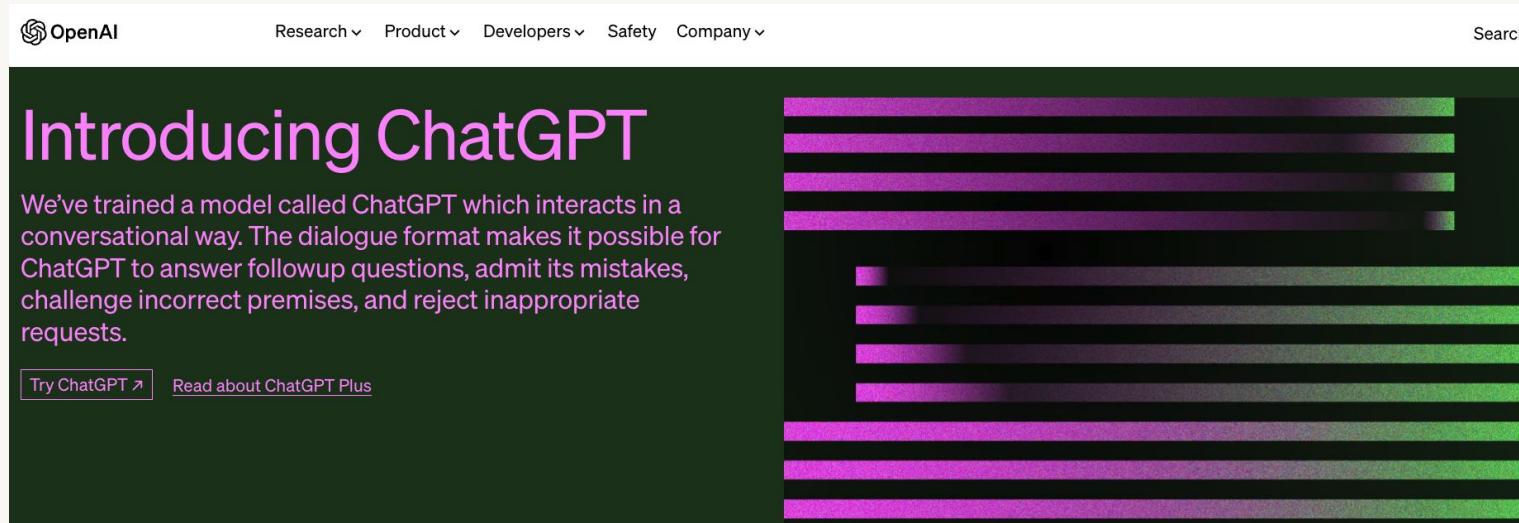
A journey to discover how GPT-4 and ChatGPT were built.



The journey to ChatGPT

What is GPT?

- ChatGPT was originally built atop a large language model known as GPT-3.5.



- GPT-3.5 is a type of decoder-based transformer model.
- Let's see what exactly GPT is.

Source: [OpenAI](#)

Generative Pre-trained Transformers (GPT)

Decoder-based transformers

- The first GPT model, introduced in 2018 was just the decoder part of the original transformer.
- GPT-2/-3/-4 have mostly just been larger versions. With the key differences coming from training data and training processes.

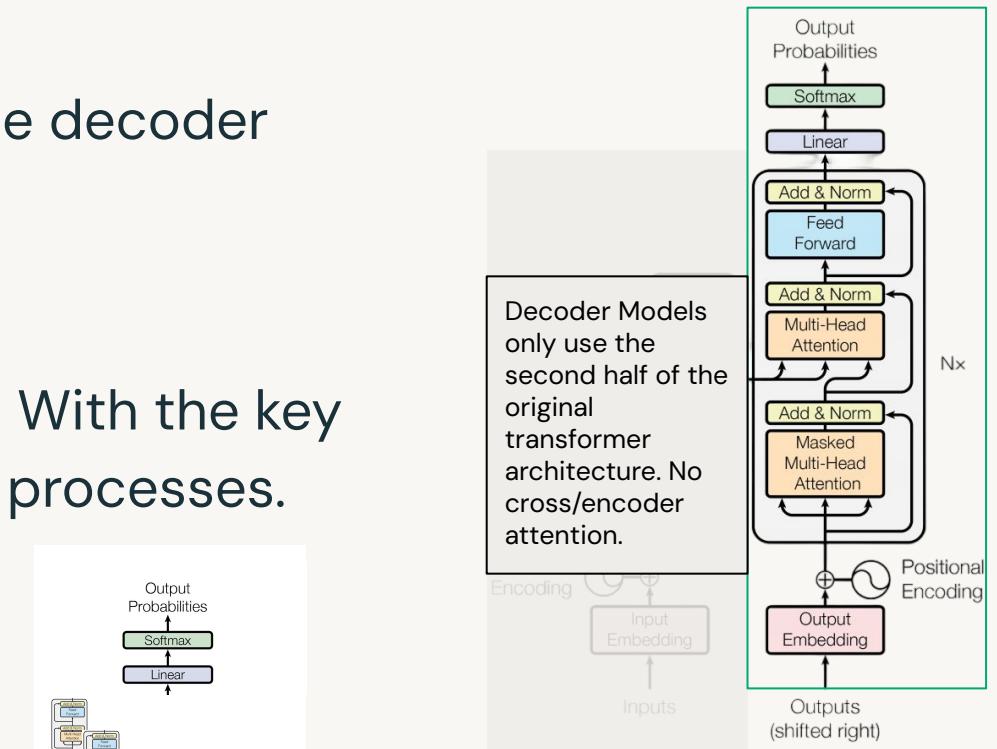
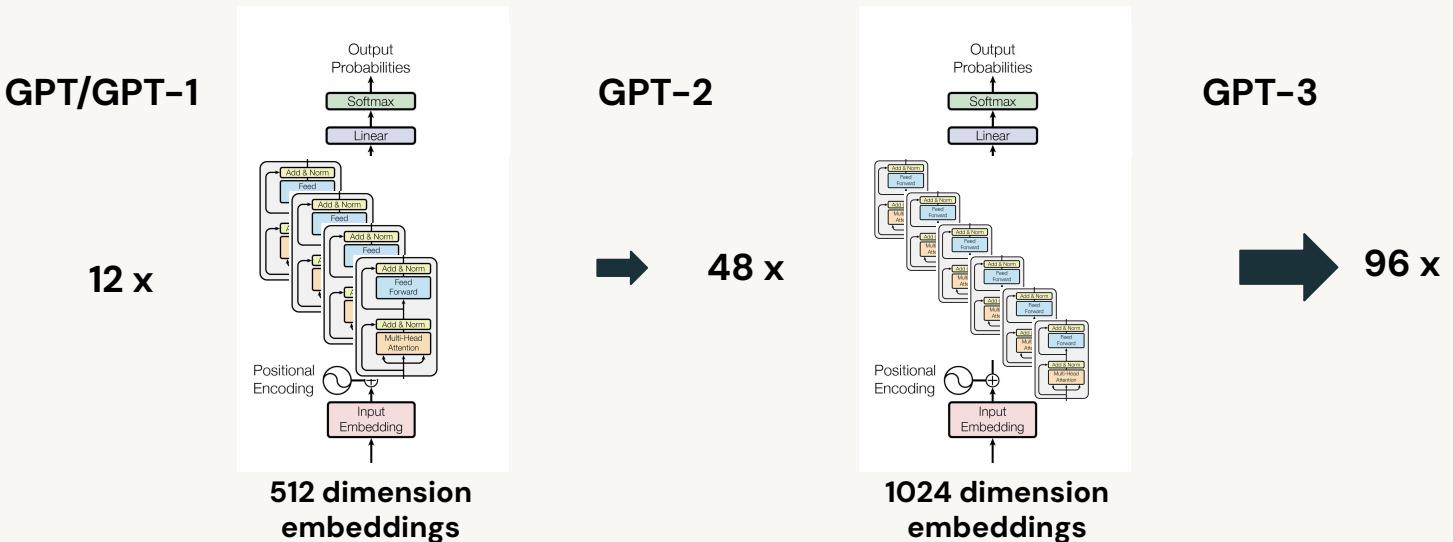


Figure 1: The Transformer - model architecture.



Generative Pre-trained Transformers (GPT)

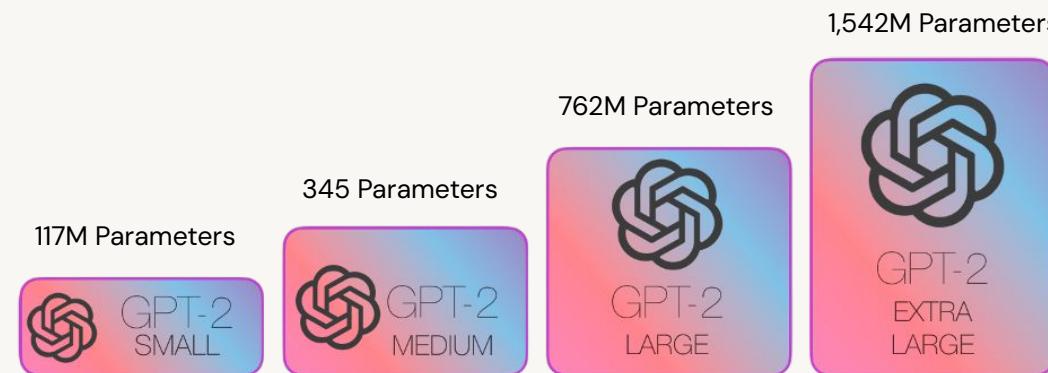
Generational Improvements

GPT (2018):

It was pre-trained on the **BooksCorpus** dataset and contained 117 million parameters. GPT was an autoregressive model based on the Transformer architecture and employed a unidirectional language modeling objective.

GPT-2 (2019):

pre-trained on a much larger dataset called **WebText**, which contained text from web pages with a total of 45 terabytes of data. GPT-2 came in four different sizes: 117M (small), 345M (medium), 774M (large), and 1.5B (extra-large) parameters.



Generative Pre-trained Transformers (GPT)

Generational Improvements

GPT-3 (2020):

Pre-trained on the **WebText2** dataset, an even larger portion of the internet, 45 terabytes of text.

GPT-3 came with a staggering 175 billion parameters

Exceptional **few-shot** and **zero-shot** learning capabilities, allowing it to perform well on various tasks with minimal or no fine-tuning.

GPT-4 (2023):

Specific details about the architecture, dataset, and parameter count for GPT-4 have not been officially released. Likely ~1T parameters in an ensemble of smaller models of ~220B parameters each.



GPT Architecture



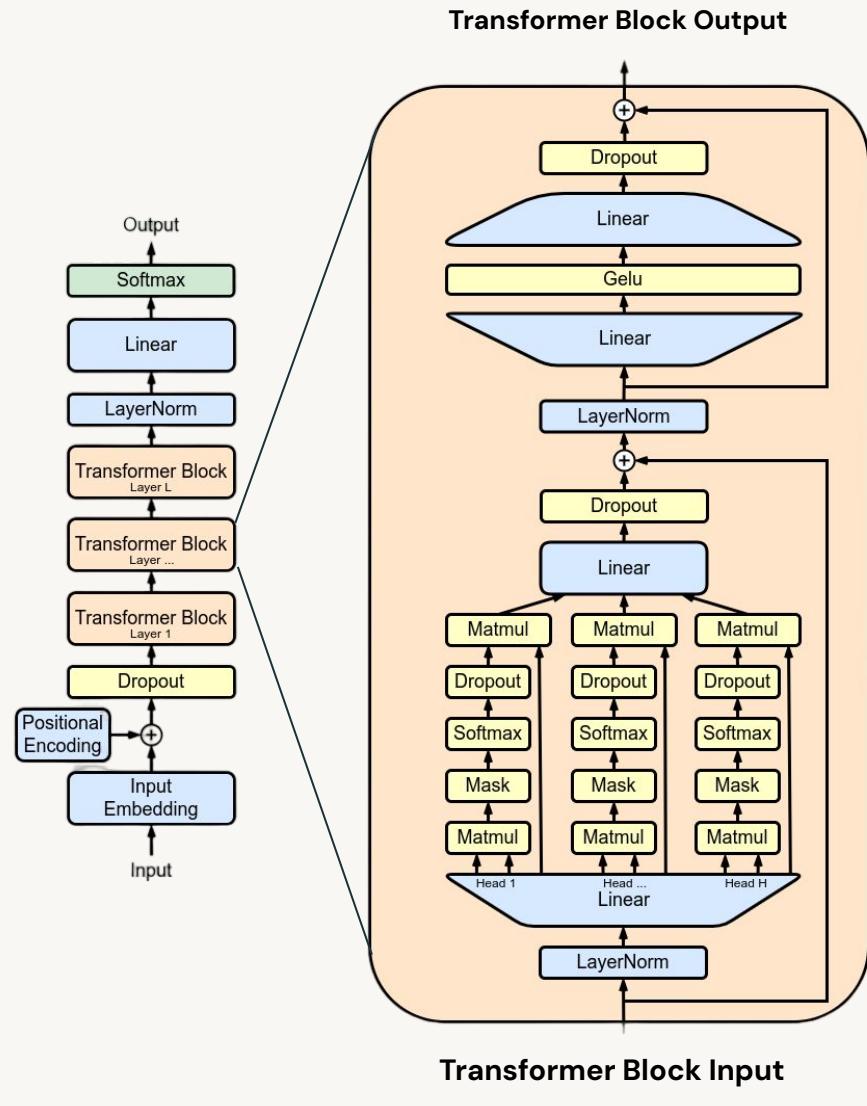
So many layers?

Why GPTs keep getting bigger

The main component of each layer: **multi-headed attention block**.

Like in convolutional layers, the earlier features that are learned might be edges and lines, and the later layers are more complex visual artifacts, attention layers work in a similar fashion:

1. **Early Attention:** short-range dependencies, relationships between adjacent or close tokens.
 - Word order
 - Part-of-Speech
 - Basic sentence structure.
2. **Middle Attention:** overall context of the input sequence.
 - Semantic information.
 - Meaning.
 - Relationships between phrases.
 - Roles of different words within the sentence.
3. **Late Attention:** integrates the lower layers and generates coherent and contextual outputs.
 - High-level abstractions.
 - Discourse structure.
 - Sentiment.
 - Complex long-range dependencies.



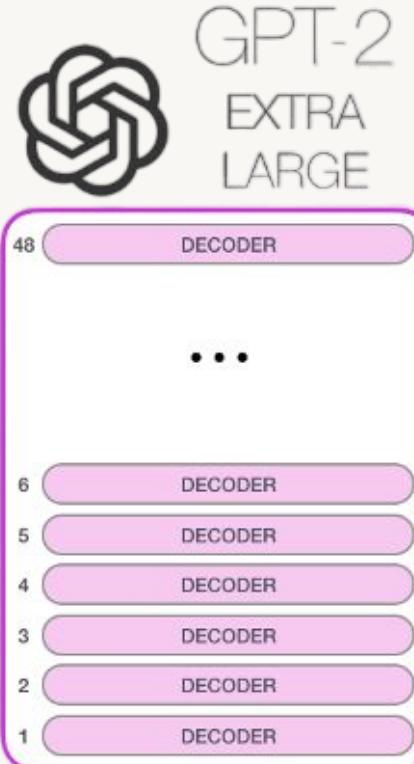
Source: [Wikipedia](#)

Why so many parameters?

Parameter	GPT-2 Small	GPT-2 Extra Large	Difference Extra Large / Small
Layers	12	48	3x
Model Dimensionality	768	1600	2x
Attention Heads per Layer	12	25	2.5x
Total	117 Million	1.5 Billion	12.8x



117M parameters



Source: [IllustratedGPT2](#)



Training GPT



Training Data

The magic behind the model

GPT-1: BooksCorpus

- Over 7,000 unpublished books spanning various genres.
 - 800 million words, covering a wide range of topics and styles.



GPT-2: WebText

- Much larger than BookCorpus
 - A collection of text from web pages, consists of 45 terabytes of data.
 - Filtered out web pages with low quality content



GPT-3: WebText2

- Even larger than WebText and more diverse
 - The increased size and diversity of the WebText2 dataset contribute to GPT-3's impressive performance on various NLP tasks.



2.0

Comparing LLM Architectures



BERT vs. GPT vs. T5

Which type of LLM is best?

	Encoder-only (eg. BERT)	Decoder-only (eg. GPT)	Seq-2-Seq (eg. T5)
Pros	<ul style="list-style-type: none">Pre-trained on a masked language modeling task with bidirectional context for a deeper understanding of input sequences. Leads to strong feature extraction capabilities.Typically several orders of magnitude smaller than decoder or seq-2-seq model.	<ul style="list-style-type: none">Autoregressive nature makes them well-suited for text generation tasks.Can generate coherent and contextually relevant text.	<ul style="list-style-type: none">Encoder-decoder architecture allows for better handling of complex, structured input-output relationships.Attention mechanisms help weigh the importance of different parts of the input when generating the output.
Cons	<ul style="list-style-type: none">Not ideal for text generation tasks due to their bidirectional nature.Can have higher computational costs compared to decoder-only models.	<ul style="list-style-type: none">Only capture unidirectional context (left-to-right), which can limit their contextual understanding.Autoregressive generation can be slow due to sequential token prediction.	<ul style="list-style-type: none">Can require more training data and computational resources compared to encoder-only or decoder-only models.Can be more complex to train and fine-tune due to the two-part architecture.



Module Summary

Transformers – What have we learned

- Transformers are built from a number of transformer blocks
- Transformer blocks use attention and neural networks to enrich vectors
- Transformer models can be encoder, decoder, or encoder-decoder
- The evolution of GPT required changes in architecture, and in data
- Base/Foundation models require fine tuning to solve most tasks



Time for some code!



Course Outline

Course Introduction

Module 1 – Transformers: Attention and the Transformer Architecture

Module 2 – Parameter Efficient Fine-Tuning: Doing more with less

Module 3 – Deployment Optimizations: Improving model size and speed

Module 4 – Multi-modal LLMs: Beyond text-based transformers



Module 2

Efficient Fine-Tuning

Doing more with less



Learning Objectives

By the end of this module you will:

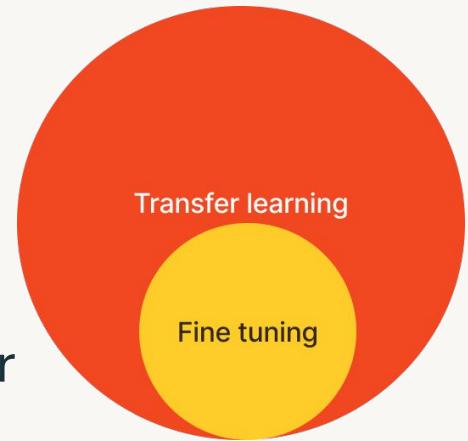
- Understand what fine-tuning is and why we do it
- Learn what parameter-efficient fine-tuning is and what the popular strategies are
- Understand the limitations of parameter-efficient fine-tuning
- Gain knowledge about data preparation best practices



Fine tuning vs. transfer learning

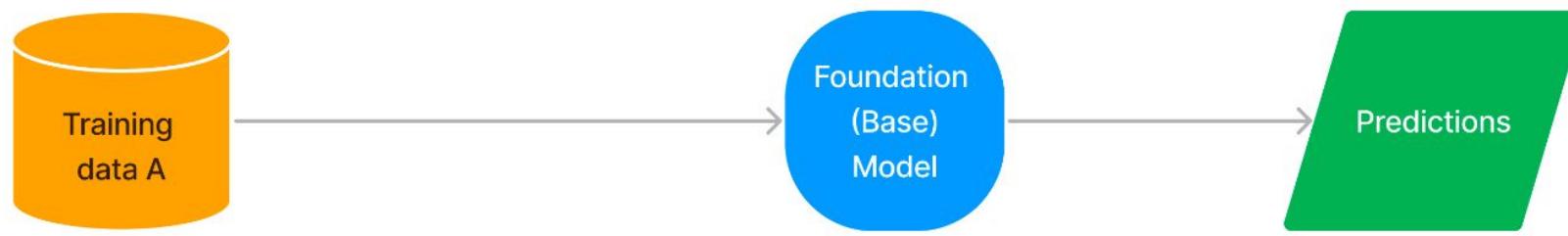
They are often referenced interchangeably

- Transfer learning
 - Apply a general pre-trained model to a new, but related task
- Fine tuning
 - Use a general pre-trained model and then train that model further
- Transfer learning \approx fine tuning
 - Train it more
 - Train on different data



How to leverage a pre-trained foundation model?

Pre-trained

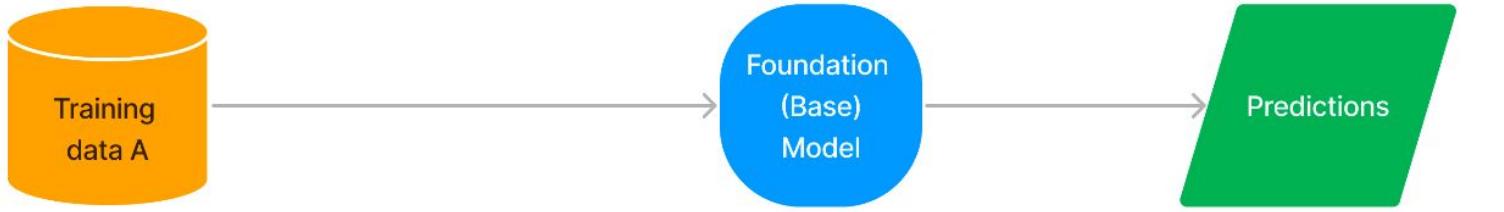


Examples:

- T5
- BloombergGPT
- GPT-4

How to leverage a pre-trained foundation model?

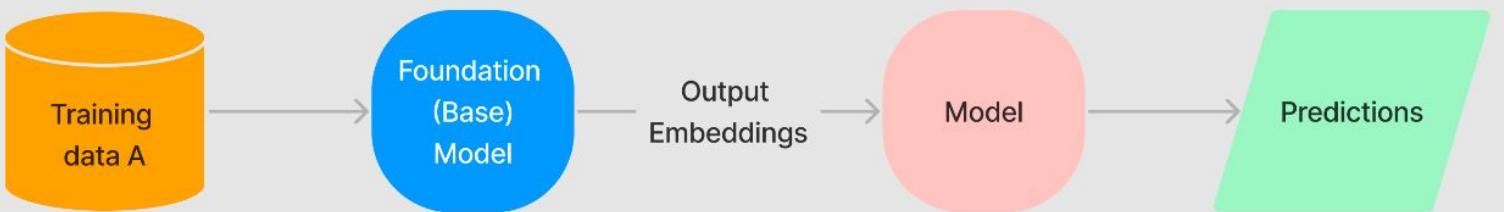
Pre-trained



Examples:

- T5
- BloombergGPT
- GPT-4

Feature Extraction

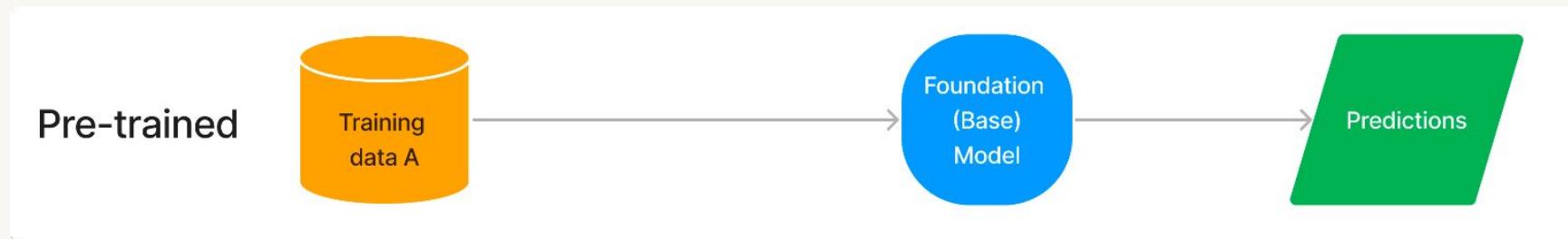


Example:

Use BERT embeddings as inputs
to a random forest classifier
→ classify movie reviews

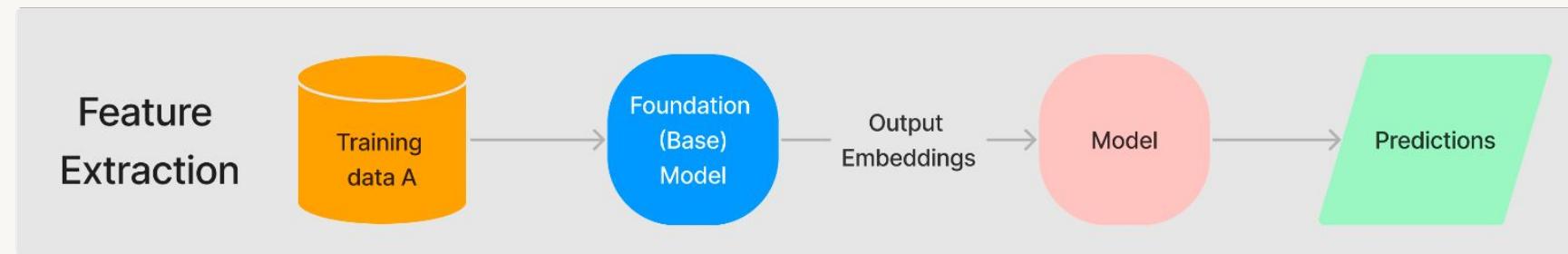


How to leverage a pre-trained foundation model?



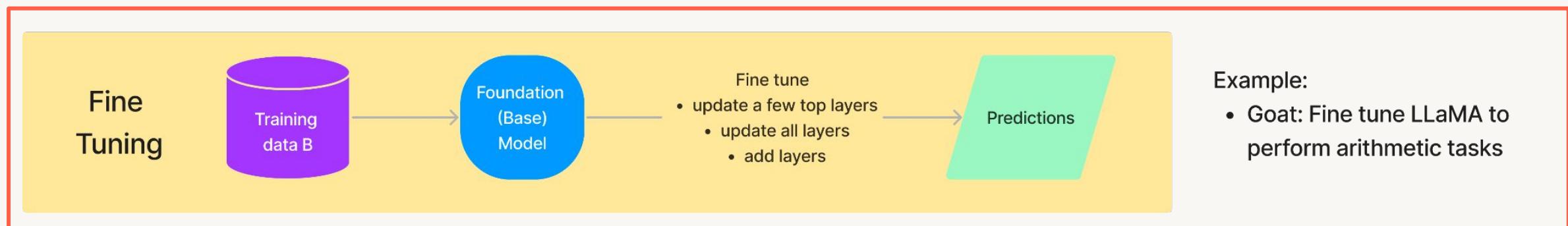
Examples:

- T5
- BloombergGPT
- GPT-4



Example:

Use BERT embeddings as inputs to a random forest classifier
→ classify movie reviews



Example:

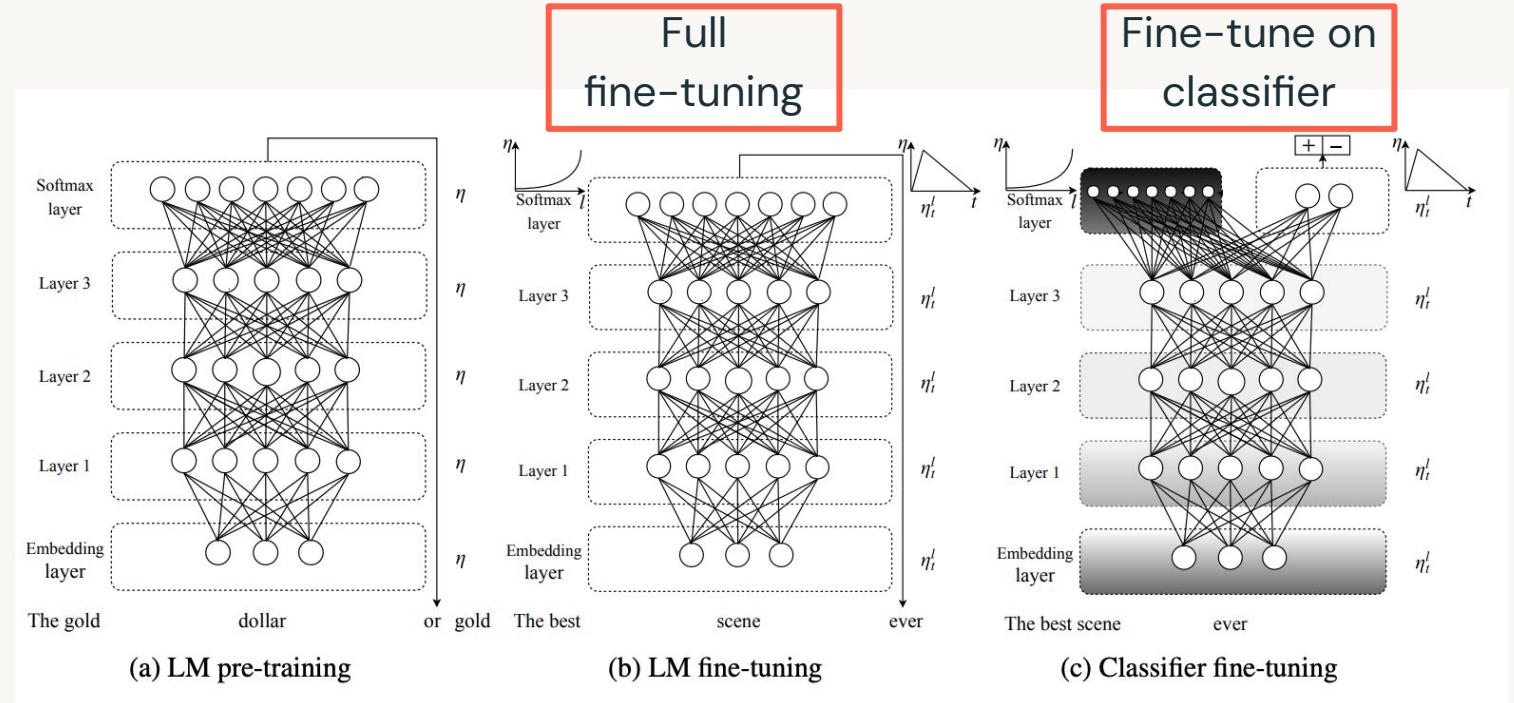
- Goat: Fine tune LLaMA to perform arithmetic tasks



Why fine tuning?

Leverage an effective pre-trained model on our own data – it's *not* new

- Improve performance downstream
 - Different pre-trained vs fine-tuned tasks
 - Different domains
- Ensure regulatory compliance
- Not new:
 - ULMfit paper in 2018

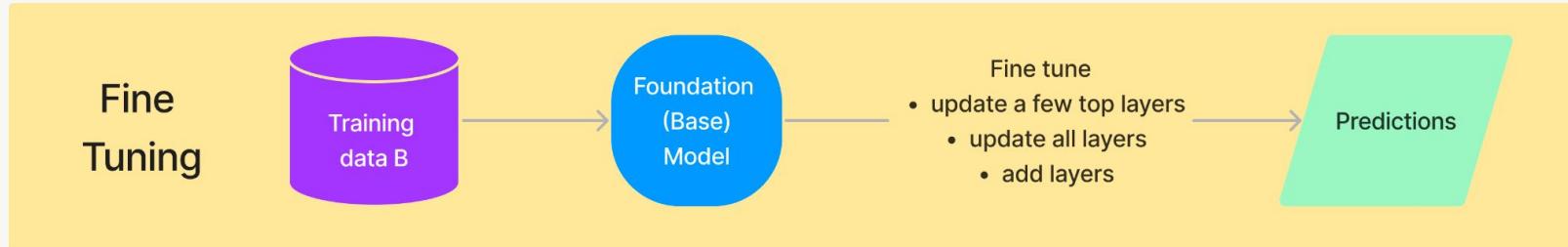


Source: [Howard and Ruder 2018](#)

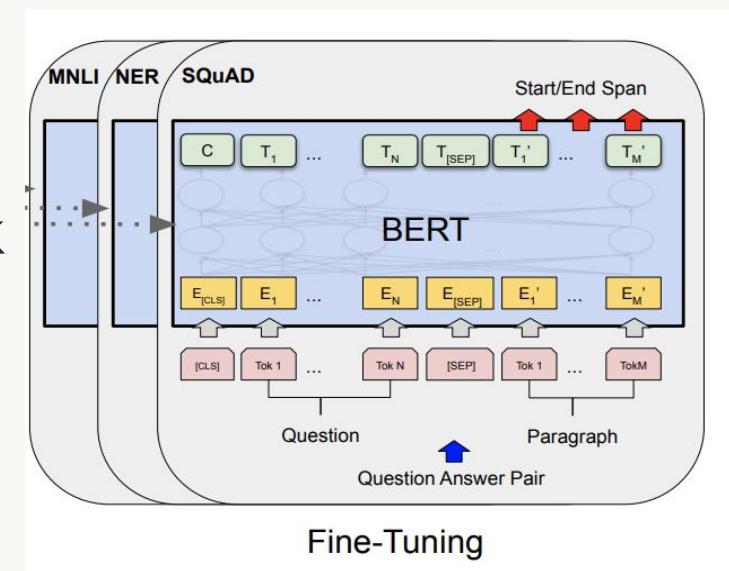


Fine tune = update foundation model weights

AKA parameter fine tuning



- Update more layers = better model performance
- Full fine-tuning typically produces one model per task
 - Serve one model per task
 - May forget other pre-trained tasks: catastrophic forgetting
- Full fine-tuning LLMs is expensive. How to avoid it?
 - X-shot learning
 - Parameter-efficient fine tuning



Source: [Devlin et al 2019](#)



X-shot learning

Provide several examples of new tasks

Prompt engineering

= developing prompts

= prompt design

= *hard/discrete prompt tuning*

Not updating model weights

```
pipeline(  
    """For each tweet, describe its sentiment:
```

Instruction

```
[Tweet]: "I hate it when my phone battery dies."
```

```
[Sentiment]: Negative
```

```
###
```

```
[Tweet]: "My day has been 👍"
```

```
[Sentiment]: Positive
```

```
###
```

```
[Tweet]: "This is the link to the article"
```

```
[Sentiment]: Neutral
```

```
###
```

```
[Tweet]: "This new music video was incredible"
```

```
[Sentiment]: """)
```

Few-shot
examples

Prompt



Pros and cons of X-shot learning

Also known as in-context learning

Pros

- No need for huge labeled training data
- No need to create a copy of model for each task
 - Simplify model serving
- Text prompts feel interpretable

Cons

- Manual prompt engineering
- Prompts are specific to models
- Context length limitation
 - Add more examples? Less space for instructions
 - Longer context = higher latency
 - LLMs forget middle portion
 - [Liu et al 2023](#) (released in July)
 - Longer context window is not the solution!
- **Performance might still be lackluster**



Fine-tuning outperforms X-shot learning

Example: GOod at Arithmetic Tasks (Goat-7B)

- Foundation model = Llama
- Trained on 1M synthetic data samples
- Accuracy outperforms
 - Few-shot PaLM-**540B** (*much bigger model !!*)
 - GPT-4
 - Typically doesn't do well in arithmetics; accuracy ~0
- SOTA on arithmetic benchmark (BIG-bench)
- Supervised **instruction fine-tuning**
 - Trained using LoRA on a 24GB VRAM GPU
 - LoRA to be covered soon!

Goat-7B

3978640188 + 42886272 =
 $3978640188 + 42886272 = 4021526460$

4523646 minus 67453156
 $4523646 - 67453156 = -62929510$

Calculate 397×4429

$$\begin{aligned} 397 \times 4429 &= 4429 \times (300 + 90 + 7) \\ &= 4429 \times 300 + 4429 \times 90 + 4429 \times 7 \\ &= 1328700 + 398610 + 31003 \\ &= 1727310 + 31003 \\ &= 1758313 \end{aligned}$$

What is $8914/64$?

$$\begin{aligned} 8914 - 64 \times 100 &= 8914 - 6400 = 2514 \\ 2514 - 64 \times 30 &= 2514 - 1920 = 594 \\ 594 - 64 \times 9 &= 594 - 576 = 18 \\ \text{Therefore, } 8914 \div 64 &= 139 \text{ R } 18 \end{aligned}$$



[Image Source](#)



Important observations about Goat

- Instruction fine-tuned
- Multi-task serving

Task 1
Addition

Goat-7B

$$3978640188 + 42886272 = \\ 3978640188 + 42886272 = 4021526460$$

4523646 minus 67453156

$$4523646 - 67453156 = -62929510$$

Task 2

Subtraction,
using natural
language (NL)

Task 3

Multiplication,
mix of NL +
mathematical
symbols

Calculate 397×4429

$$\begin{aligned} 397 \times 4429 &= 4429 \times (300 + 90 + 7) \\ &= 4429 \times 300 + 4429 \times 90 + 4429 \times 7 \\ &= 1328700 + 398610 + 31003 \\ &= 1727310 + 31003 \\ &= 1758313 \end{aligned}$$

Task 4

Division,
mix of NL +
mathematical
symbols

What is $8914/64$?

$$\begin{aligned} 8914 - 64 \times 100 &= 8914 - 6400 = 2514 \\ 2514 - 64 \times 30 &= 2514 - 1920 = 594 \\ 594 - 64 \times 9 &= 594 - 576 = 18 \\ \text{Therefore, } 8914 \div 64 &= 139 \text{ R } 18 \end{aligned}$$



Instruction-tuned, multi-task LLM

Instruction-tuned = tune general purpose LLMs to follow instructions

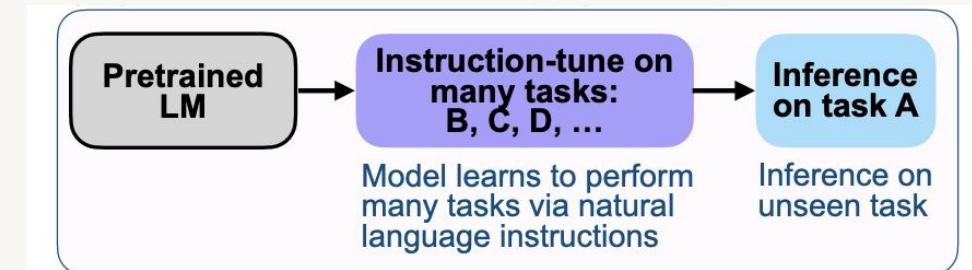
FLAN (Fine-tuned LAnguage Net)

- Foundation model = 137B model
- Instruction-tuned on over 60 NLP datasets with different task types
 - Task types: Q/A, translation, reasoning, comprehension, etc.
- Examples
 - T5 → FLAN-T5
 - PaLM → FLAN-PaLM

Dolly



- Foundation model = Pythia-12B
- Instruction-tuned on 15k prompt/response pairs
 - Task types: Q/A, classification, information extraction, etc.



Source: [Wei et al 2022](#)



Quick recap

We want efficient training, serving, and storage

- Full fine-tuning can be computationally prohibitive
 - Memory usage: activation, optimizer states, gradients, parameters
 - This gives the best performance
- Compromise: Do some, but not full, fine-tuning
 - Saves cost to use low-memory GPUs
- We want multi-task serving, rather than one model per task
 - E.g. one model for Q/A, summarization, classification



Enter parameter-efficient fine-tuning



Parameter-efficient fine-tuning (PEFT)



3 categories of PEFT methods

Additive

- Soft prompt
 - Prompt tuning
 - Prefix tuning

Selective

- Akin to updating a few foundation model layers
 - BitFit
 - Only updates bias parameters
 - Diff Pruning
 - Creates task-specific “diff” vectors and only updates them

Re-parameterization

- Decompose weight matrix updates into smaller-rank matrices
 - LoRA



We will cover additive and reparameterization

Additive

- Soft prompt
 - Prompt tuning
 - Prefix tuning

Selective

- **Model quality performance is not as good**
- Akin to updating a few foundation model layers
 - BitFit
 - Only updates bias parameters
 - DiffPruning
 - Creates task-specific “diff” vectors and only update them

Re-parameterization

- Decompose weight matrix into smaller-rank matrices
 - LoRA

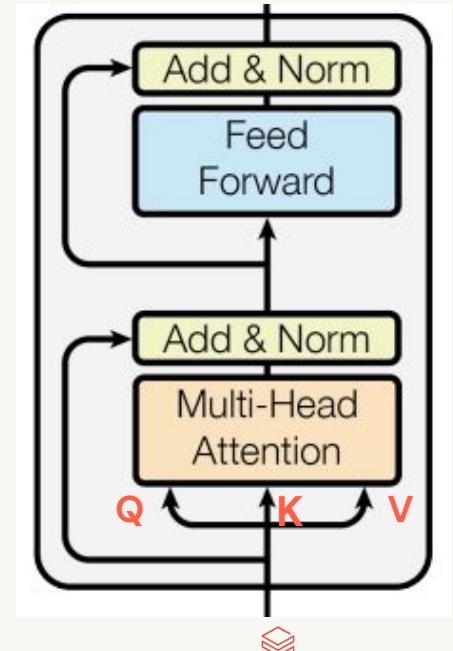


High-level overview of PEFT

Active research area: >100 papers in last few years!

- Additive: Add new tunable layers to model
 - Keep the foundation model weights frozen and update only the new layer weights
- Reparameterization: Decompose a weight matrix into lower-rank matrices
- Implementation:
 - Acts on the core Transformer block
 - Basic multi-head attention and/or feed forward network
 - Some act specifically on the weight matrices: **Query**, **Key**, **Value**
 - These matrices pass information from one token to another

Source: [Vasmani et al 2021](#)



Additive: Prompt Tuning (and prefix tuning)

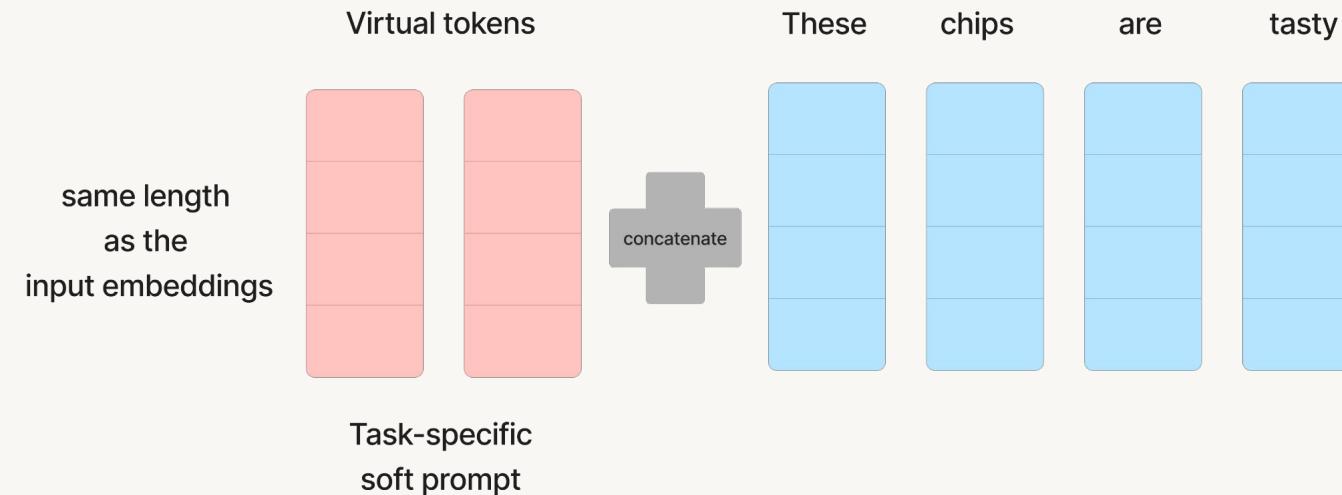


Soft prompt tuning

Concatenates trainable parameters with the input embeddings

- Learn a new sequence of task-specific embeddings
- We call this prompt tuning, not model tuning, because we only update prompt weights

Task: Classify product reviews; task batch = 1

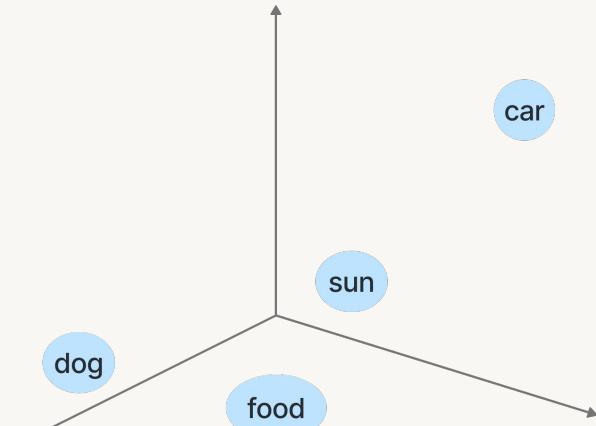


What are these *virtual tokens*?

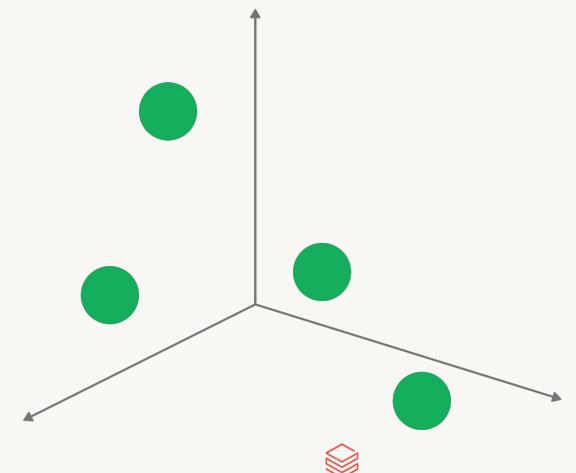
Goal: remove manual element of engineering prompts!

- Randomly initialized embedding vectors
- Not part of vocabulary
- Analogy:
 - Bitcoin: We can't touch it like cash. We don't know how it "looks", but it exists and works.

Real input tokens in embedding space



Virtual tokens in embedding space

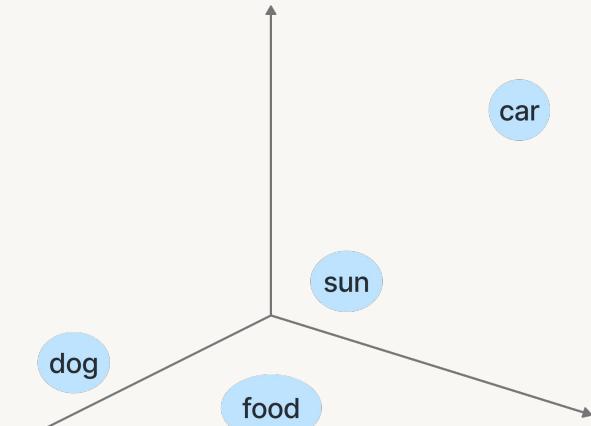


What are these *virtual tokens*?

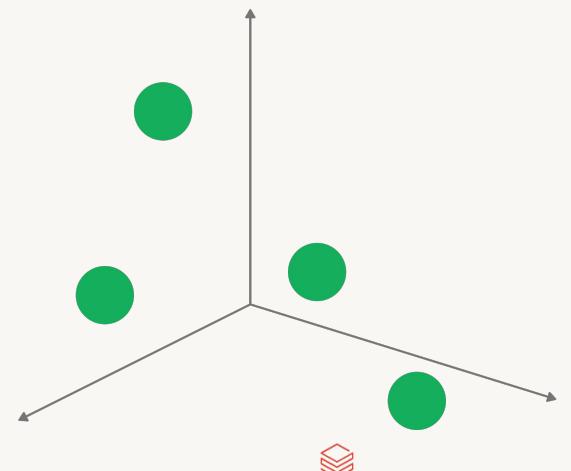
Goal: remove manual element of engineering prompts!

- Randomly initialized embedding vectors
 - We can also initialize to discrete prompts
 - But random initialization is nearly as good as informed initialization ([Qin and Eisner 2021](#))
- Not part of vocabulary
- Analogy:
 - Bitcoin: We can't touch it like cash. We don't know how it "looks", but it exists and works.

Real input tokens in embedding space



Virtual tokens in embedding space

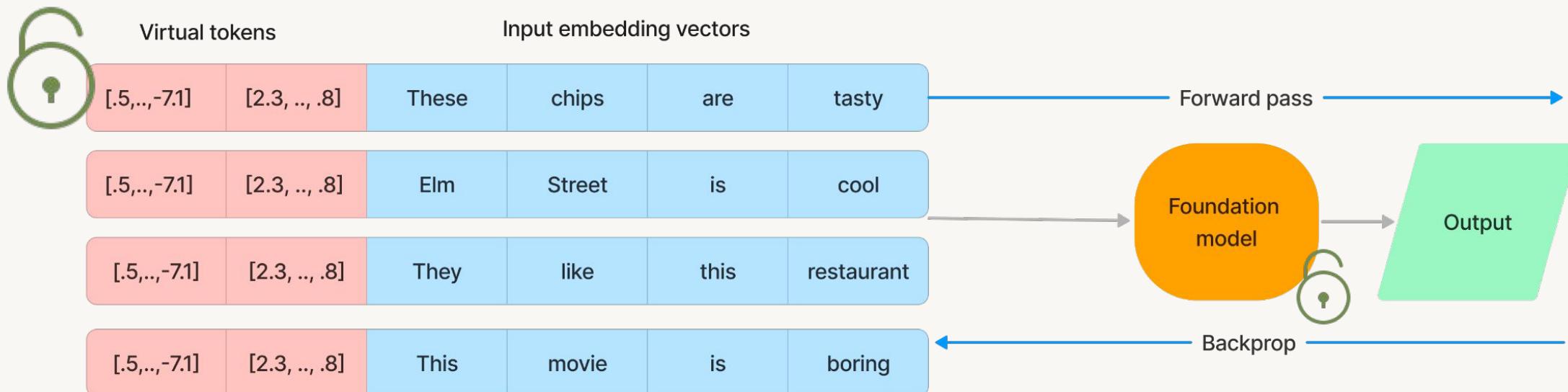


Compare full fine-tuning vs prompt tuning

Scenario: full fine-tuning

Backprop: update **all** weights based on loss

Task: Classify sentiment; task batch = 4



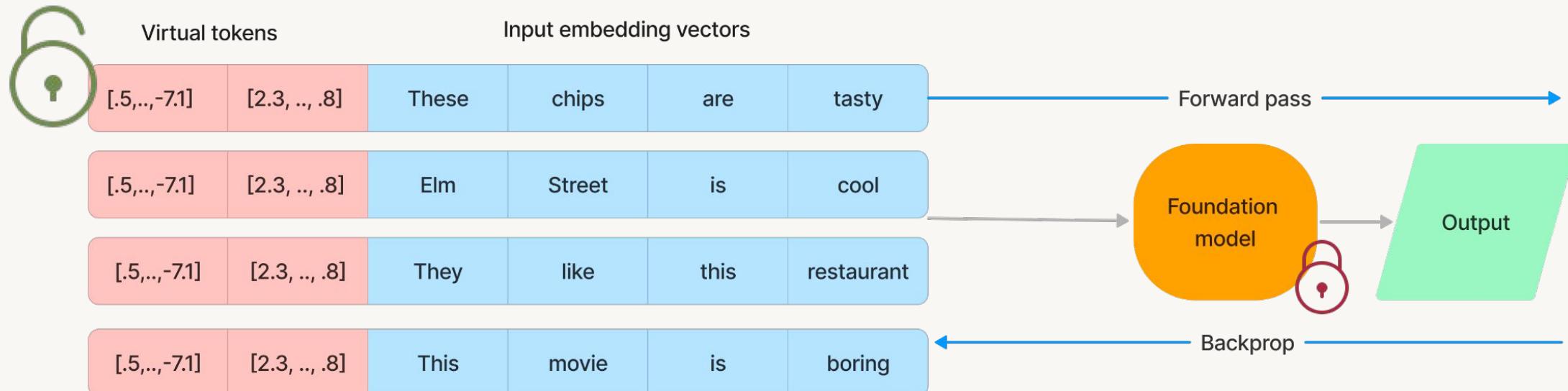
Compare full fine-tuning vs prompt tuning

Scenario: prompt tuning

Backprop: update **only prompt** weights based on loss

- The model learns the optimal representation of the prompt automatically

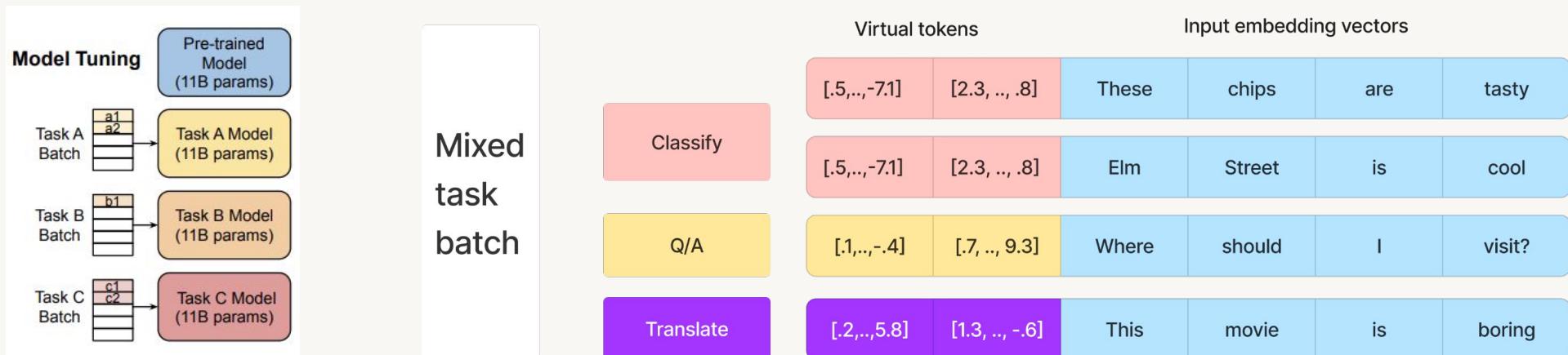
Task: Classify sentiment; task batch = 4



Allows swapping of task prompts

Efficient for multi-task serving

- Each task is a prompt, not a model
 - Only need to serve a single copy of the frozen model for multi-task serving
- Prompts for various tasks can be applied to different inputs
 - A serving request can be a single, larger mixed task batch

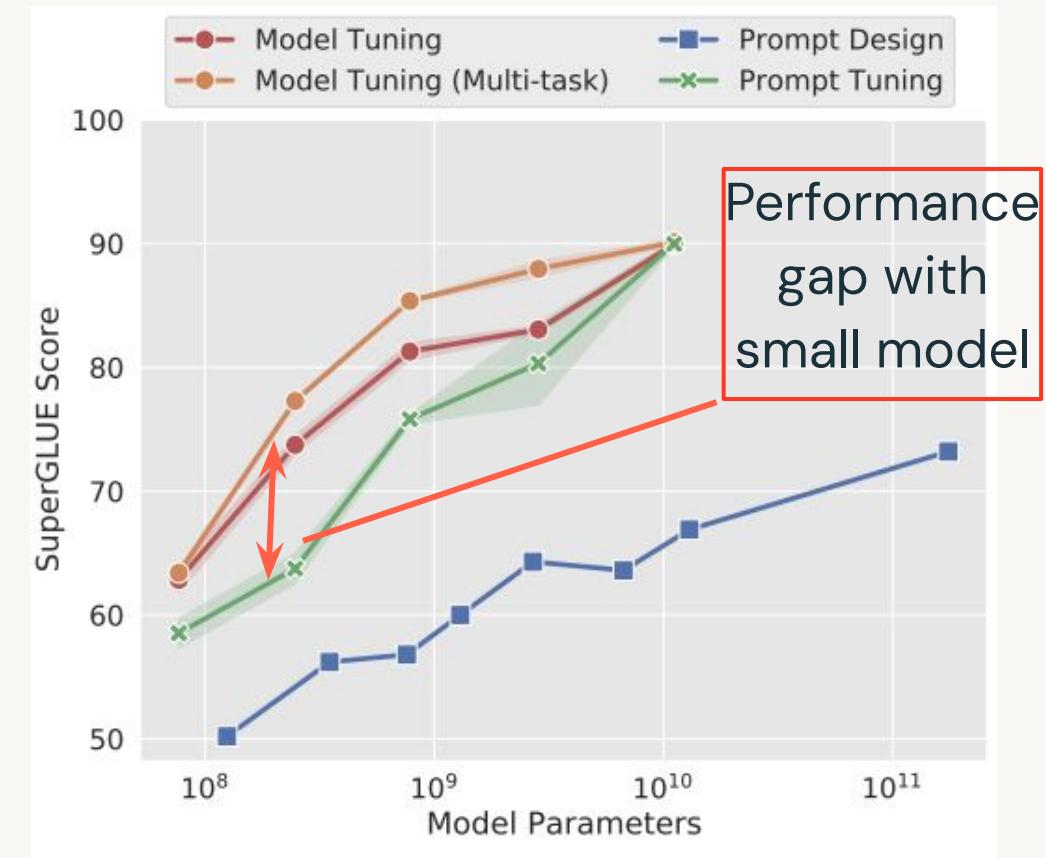


Source: [Lester et al 2021](#)



Matches fine tuning performance for >11B model

- Comparable with full fine-tuning at the 10B model scale
- More applicable to larger models
- SuperGLUE (2019)
 - Styled after GLUE, but more difficult and diverse
 - Boolean questions, comprehension, etc.

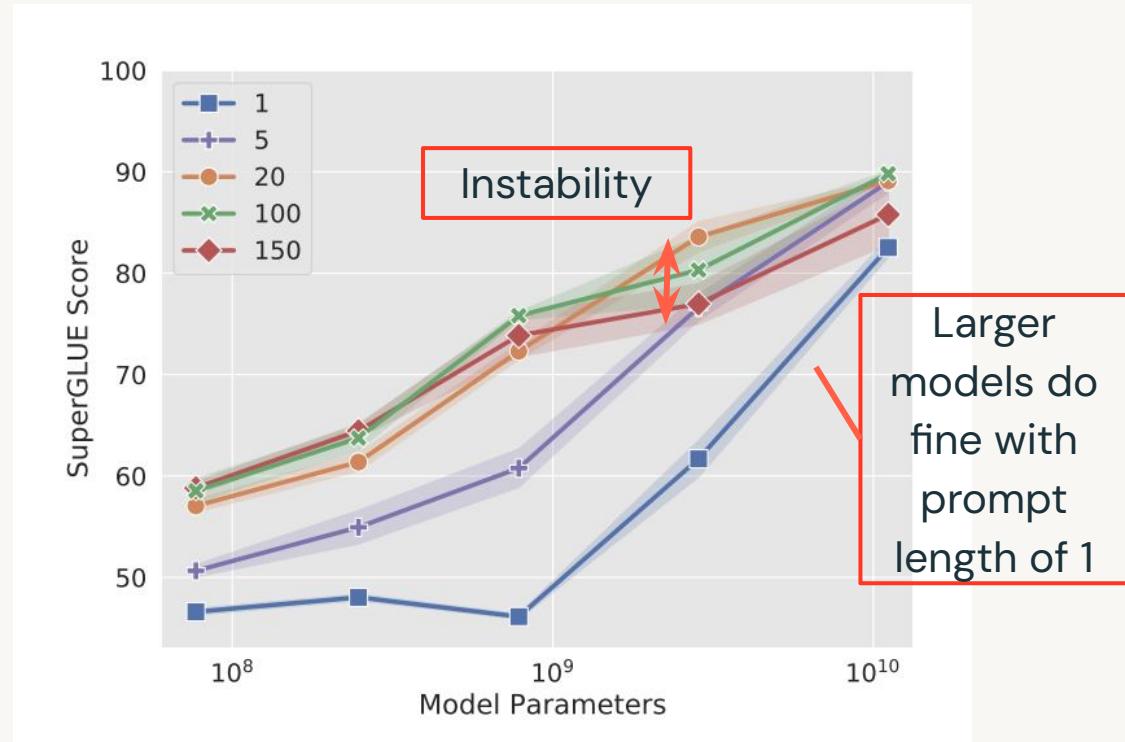


Source: [Lester et al 2021](#)



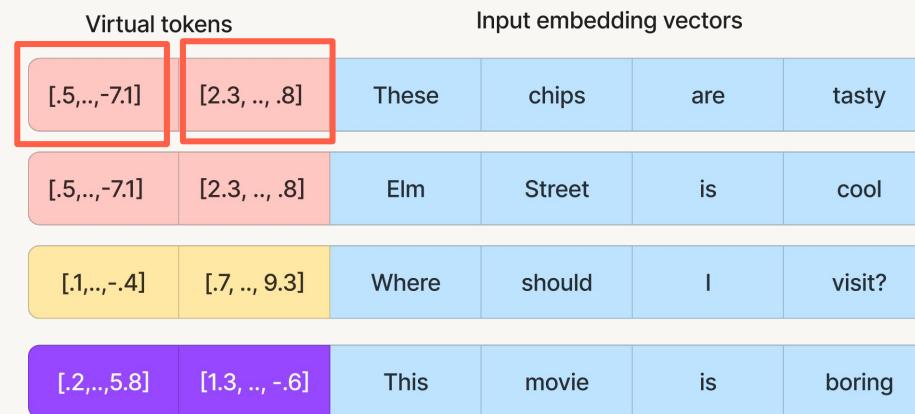
Prompt length affects larger models less

Prompt length of 20–100 is typical



In this example:

- (Virtual) prompt length = 2

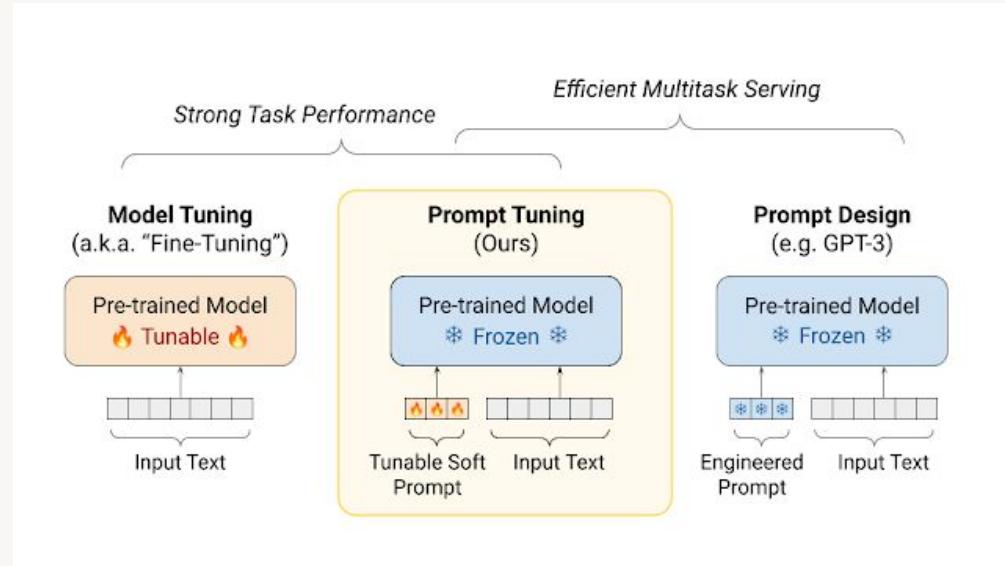


Source: [Lester et al 2021](#)



Advantages of prompt tuning

- Use whole training set
 - Not limited by # of examples that can fit in the context
- Automatically learn a new prompt for a new model
 - Backprop helps us find the best representation
- One foundation model copy only
- Resilient to domain shift



Source: [Google AI Blog](#)

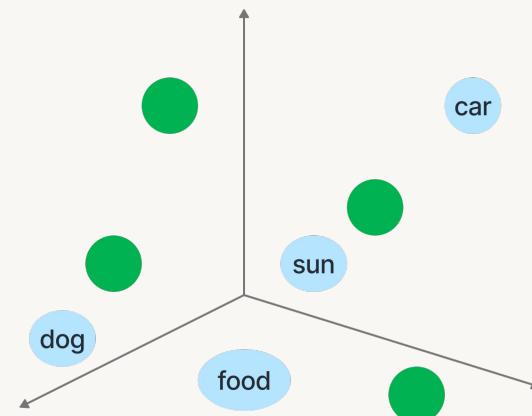


Disadvantages of prompt tuning

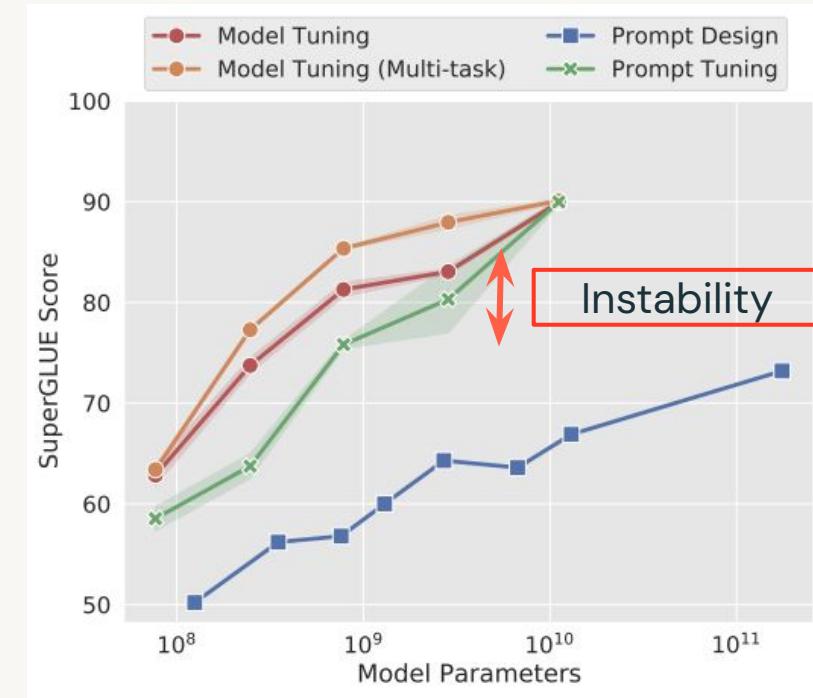
Less interpretable

- Need to convert the embeddings back to tokens
- Use cosine distance to find the top-K nearest neighbors

Find which tokens are nearest to the virtual tokens



Unstable performance

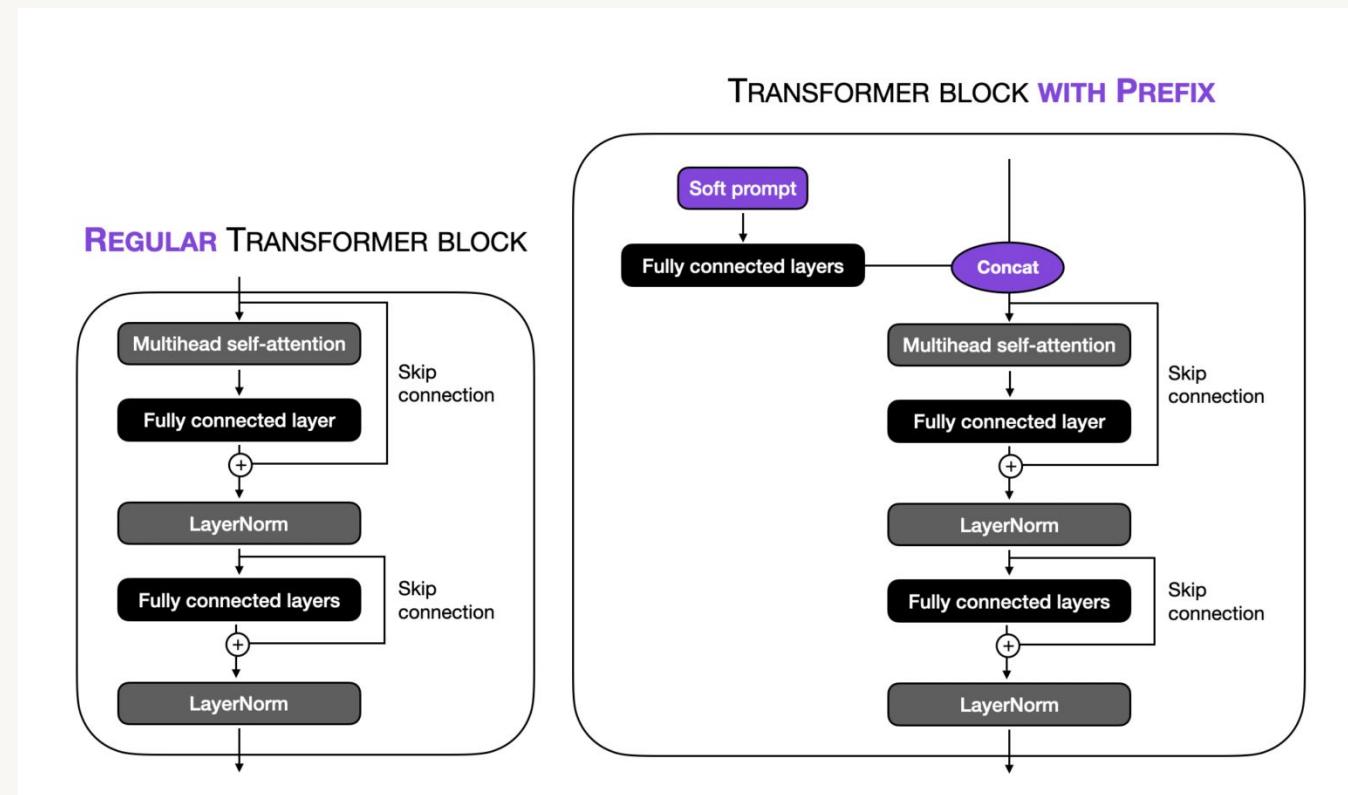
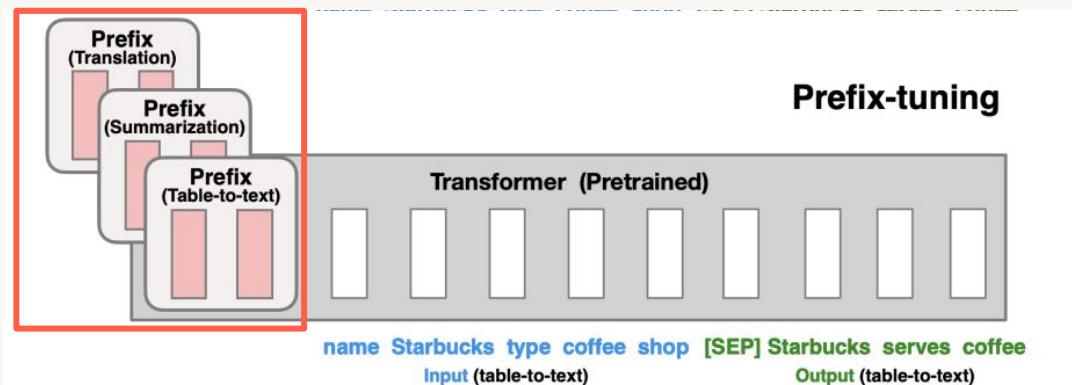


Source: [Lester et al 2021](#)



Prefix tuning is very similar to prompt tuning

Adding tunable layer to each transformer block, rather than just the input layer



Source: [Lightning AI](#)



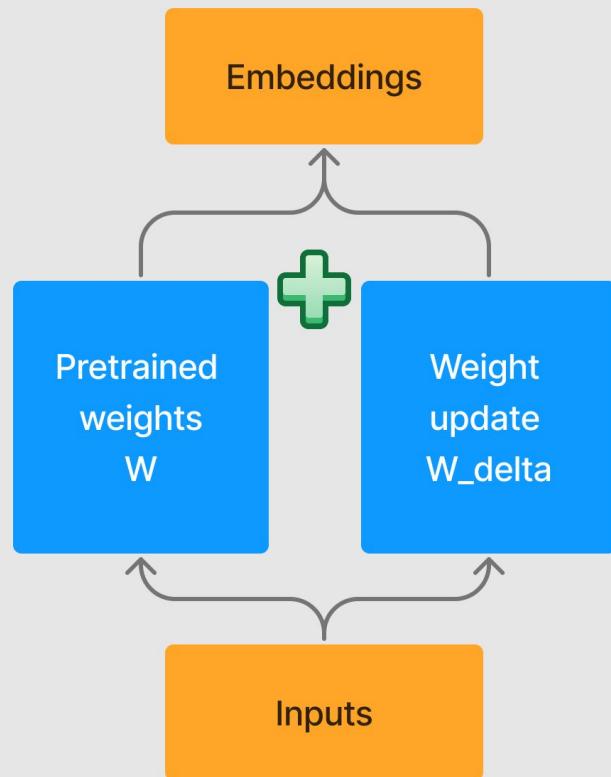
Re-parameterization: LoRA



Low-Rank Adaptation (LoRA)

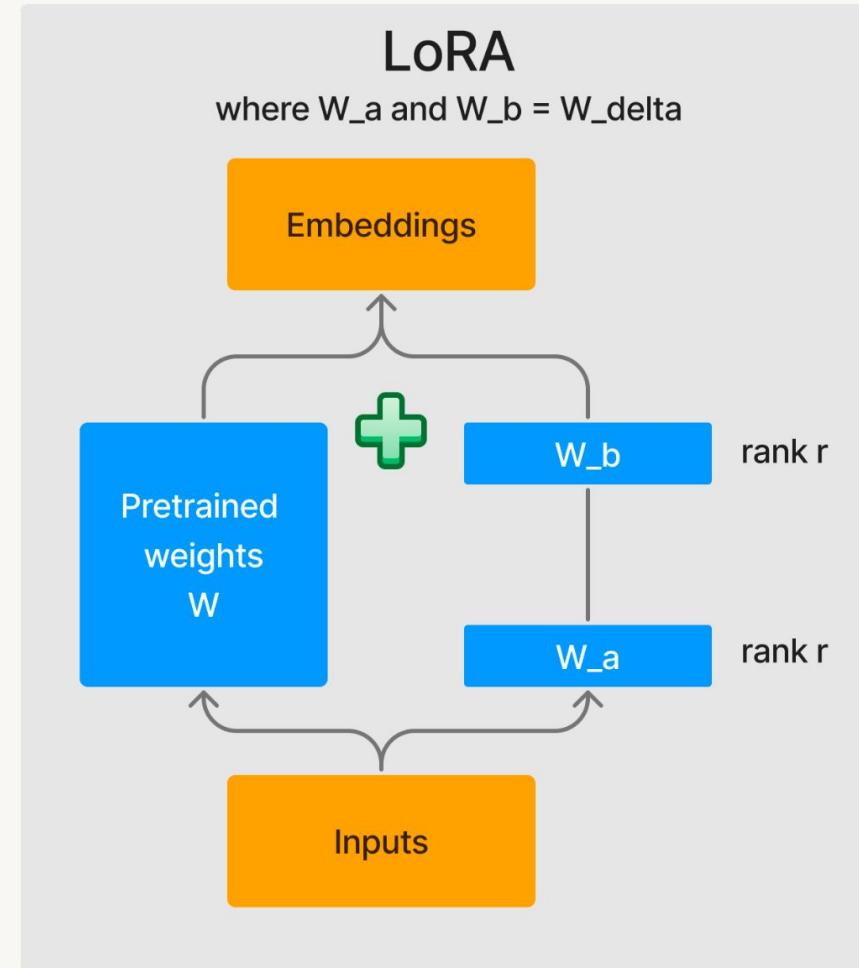
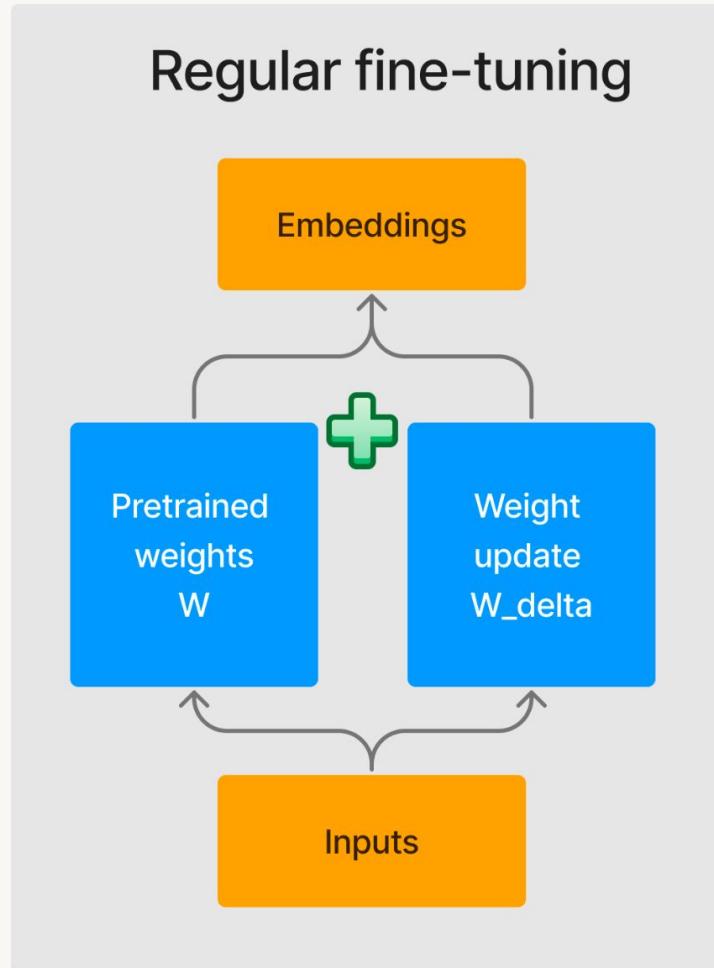
Decomposes the weight change matrix into lower-rank matrices

Regular fine-tuning



Low-Rank Adaptation (LoRA)

Decomposes the weight change matrix into lower-rank matrices



Rank? Brief visit to linear algebra

Maximum # of linearly independent columns or rows

- How many unique rows or columns?
- Full rank = no redundant row or column in the matrix
- Linear = can multiply by a constant
- Independence = no dependence on each other

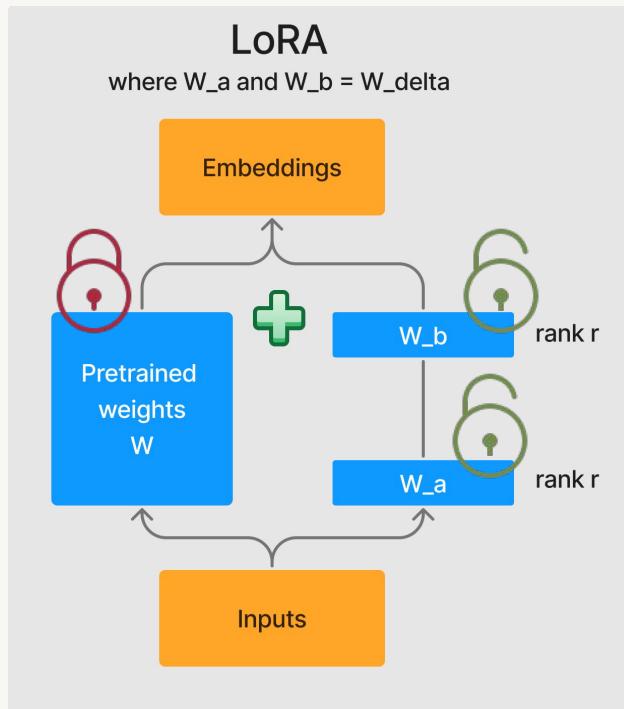
$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 6 & 9 \end{bmatrix}$$

- Row rank: 1
 - 2nd row = 3x 1st row
- Column rank: 1
 - 2nd column = 2x 1st column
 - 3rd column = 2x 2nd column

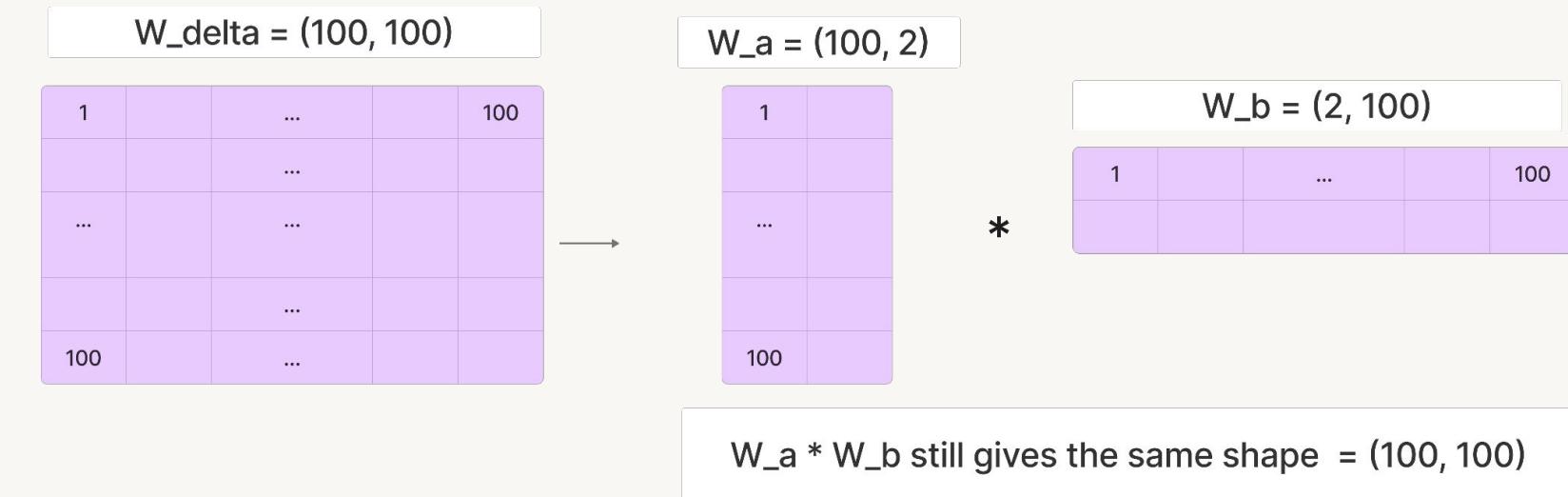


How does weight matrix decomposition work?

Observation: Actual rank of the attention weight matrices is low



$$W_{\text{delta}} = W_a * W_b$$



- Total parameters = $(100 \times 2) + (2 \times 100) = 400$
- Original parameters = $(100 \times 100) = 10,000$ parameters
- Reduction = $10,000 - 400 = 96\%$!



LoRA matches/~outperforms full fine-tuning

- $37.7 / 175255.8$
= 0.0002
= 0.02% of parameters!

Model&Method	# Trainable Parameters	WikiSQL	MNLI-m	SAMSum	R1/R2/RL	Rouge
		Acc. (%)	Acc. (%)	R1/R2/RL		
GPT-3 (FT)	175,255.8M	73.8	89.5	52.0/28.0/44.5		
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5		
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5		
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5		
GPT-3 (Adapter ^H)	7.1M	71.9	89.8	53.0/28.9/44.8		
GPT-3 (Adapter ^H)	40.1M	73.2	91.5	53.2/29.0/45.1		
GPT-3 (LoRA)	4.7M	73.4	91.7	53.8/29.8/45.9		
GPT-3 (LoRA)	37.7M	74.0	91.6	53.4/29.2/45.1		

Source: [Hu et al 2021](#)



LoRA performs well with very small ranks

GPT-3's validation accuracies are similar across rank sizes

W_q = query

W_k = key

W_v = value

W_o = output

	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

Source: [Hu et al 2021](#)

But, small r likely won't work for all tasks/datasets.

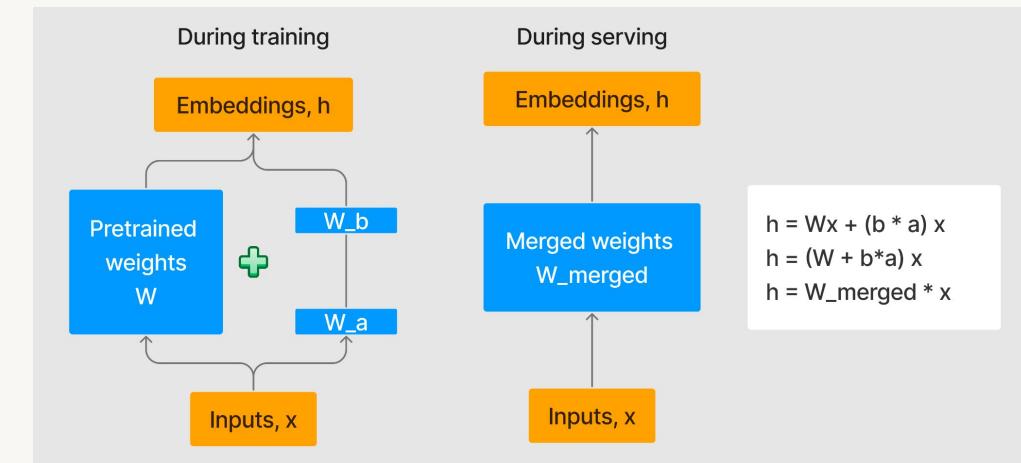
- E.g. downstream task is in a different language



Advantages of LoRA

Similar to prompt tuning, majority of the model weights are frozen

- Able to share and re-use the foundation model
 - Swap different LoRA weights for serving different tasks
- Improves training efficiency
 - Lower hardware barrier (no need to calculate most gradients or optimizer states)
- Adds no additional serving latency
 - $W_a * W_b$ can be merged
- Can be combined with other PEFT methods



Limitations of LoRA

- Not straightforward to do multi-task serving
 - How to swap different combos of A and B in a **single** forward pass?
 - If dynamically choose A and B based on tasks, there is additional serving latency
- Future research
 - From LoRA authors: If W_{delta} is rank-deficient, is W too?
 - Newer PEFT technique: [IA3 \(2022\)](#)
 - Reduces even more trainable parameters than LoRA!



PEFT Limitations



Model performance limitations

- Difficult to match the performance of full fine-tuning
 - Sensitive to hyperparameters
 - Unstable performance
- Current research area: where is best to apply PEFT?
 - E.g. why apply PEFT to only attention weight matrices? Soft prompts?
 - Vu et al 2022: Soft prompt transfer
- We may still need full-parameter fine-tuning
 - Lv et al 2023 (released in June): use new optimizer, LOMO, to reduce memory usage to ~11%



Compute limitations



Doesn't always make
inference more efficient



Doesn't reduce the cost of
storing massive foundation
models



Doesn't reduce time
complexity of **training**
Requires full forward and
backward passes

Data Preparation Best Practices



Better models from better training data

Many newer good models use C4 (e.g. MPT-7B)

Llama

- Trained on 20 most-spoken languages, focusing on those with Latin and Cyrillic alphabets

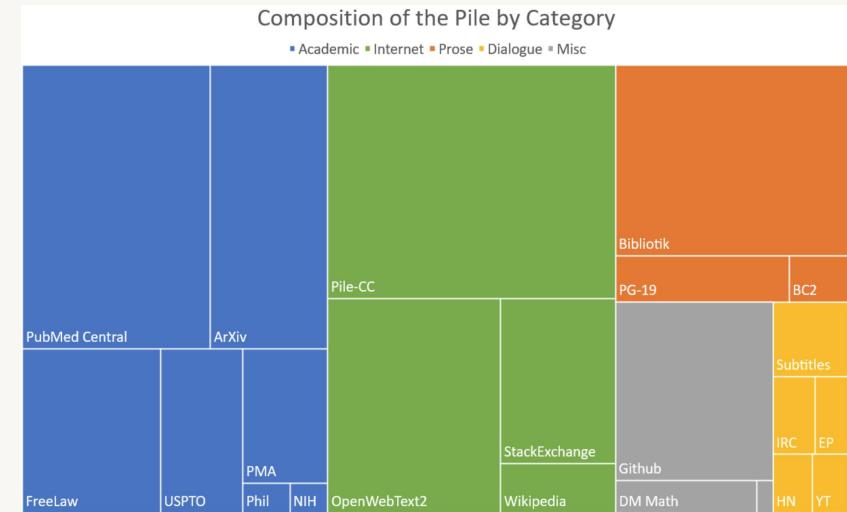
Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Source: [Touvron et al 2023](#)

Colossal
Cleaned
Crawled
Corpus

GPT-Neo and GPT-J

- Trained on the Pile: 22 diverse datasets
- Outperformed GPT-3 in some instances ([Read more here](#))



Source: [Gao et al 2020](#)



Training data makes the biggest difference

Not necessarily the model architecture

- Bloomberg created 363B-token dataset of English financial documents spanning 40 years
 - Augmented with 345B-token public dataset
- Outperforms existing open models on financial tasks

<i>Finance-Specific</i>	BloombergGPT	GPT-NeoX	OPT-66B	BLOOM-176B
Financial Tasks	62.51	51.90	53.01	54.35
Bloomberg Tasks (Sentiment Analysis)	62.47	29.23	35.76	33.39

<i>General-Purpose</i>	BloombergGPT	GPT-NeoX	OPT-66B	BLOOM-176B	GPT-3
MMLU	39.18	35.95	35.99	39.13	43.9
Reading Comprehension	61.22	42.81	50.21	49.37	67.0
Linguistic Scenarios	60.63	57.18	58.59	58.26	63.4

Source: [Wu et al 2023](#)



How much fine-tuning data do I need?

- Zhou et al 2023 (May): fine-tune 1,000 **high-quality** labeled examples from LLaMa 65B
 - When scaling up data **quantity**, need to scale up prompt **diversity**
- OpenAI: At least a couple hundred
 - Doubling dataset size leads to linear increase in model performance
- How to get more data? Synthetic data
 - Synonym replacement / rewrite
 - Word deletion: “brilliantly expressed” => “expressed”
 - Word position swapping: “It is lovely” -> “Lovely, it is”
 - Noise injection: introduce typos



Data preparation best practices

Quantity, diversity, and quality

- Don't provide detailed instructions.
Only prompt and completion.
 - Fixed separator `\n\n###\n\n` to inform when the prompt ends and completion begins
 - The separator shouldn't appear anywhere else

```
1 {"prompt": "<prompt text>", "completion": "<ideal generated text>"}
2 {"prompt": "<prompt text>", "completion": "<ideal generated text>"}
3 {"prompt": "<prompt text>", "completion": "<ideal generated text>"}
4 ...
```



Source: [OpenAI](#)

©2023 Databricks Inc. — All rights reserved

[Manually Authored – Writing]

I am planning to start a book club with some friends. Can you write an email invitation for the same?

Here is an example email invitation for a book club:

Subject: Join me for a book club!

Hi [Friend's Name], how are you?

I'm starting a book club with some friends, and I'd love to have you join us!

The book club is a casual and fun way to read and discuss books that we love or want to learn more about, and also make new friends. We plan to meet more-or-less once a month.

We're thinking of having our first meeting on [Date & Time]. Could that work for you?

Would love to see you there!

Thanks,
[Your Name]

Source: [Zhou et al 2023](#)



Data preparation best practices

- Remove undesired data
 - Offensive, toxic content
 - Private or confidential information
- Using LLM output as data is not always the answer
 - Imitation models learn style, rather than content ([Gudibande et al 2023](#))
 - Consistent with [Zhou et al 2023](#): knowledge is largely learned during pre-training
- Manually verify data quality



Module Summary

Efficient Fine-Tuning – What have we learned?

- Fine-tuning gives the best results, but can be computationally expensive
- Parameter-efficient fine-tuning reduces # of trainable parameters
- Prompt tuning allows virtual prompts to be learned automatically
- LoRA decomposes the weight change matrix into lower-rank matrices
- Fine-tuning data quality and diversity matters a lot



Time for some code!



Course Outline

Course Introduction

Module 1 – Transformers: Attention and the Transformer Architecture

Module 2 – Parameter Efficient Fine-Tuning: Doing more with less

Module 3 – Deployment Optimizations: Improving model size and speed

Module 4 – Multi-modal LLMs: Beyond text-based transformers



Module 3

Deployment Optimizations

Improving model size and speed



Learning Objectives

By the end of this module you will:

- Be able to make the design choices for your LLM development
- Create and design your own pseudo Mixture-of-Experts LLM system
- Develop an understanding of the utility of quantization in LLMs for both training and inferencing



Extra-Large Language Models

What if our models are too big?



The issue with high performance LLMs

Paying the price for quality

As models grow in size, they get “better” and “worse”.

- **Better** – accuracy, alignment, abilities
- **Worse** – speed, memory footprint, updatability



What if we could improve the speed and footprint while preserving quality?

Improving Learning Efficiency

How can we train and fine-tune better?



How we interact with LLMs

The importance of context length

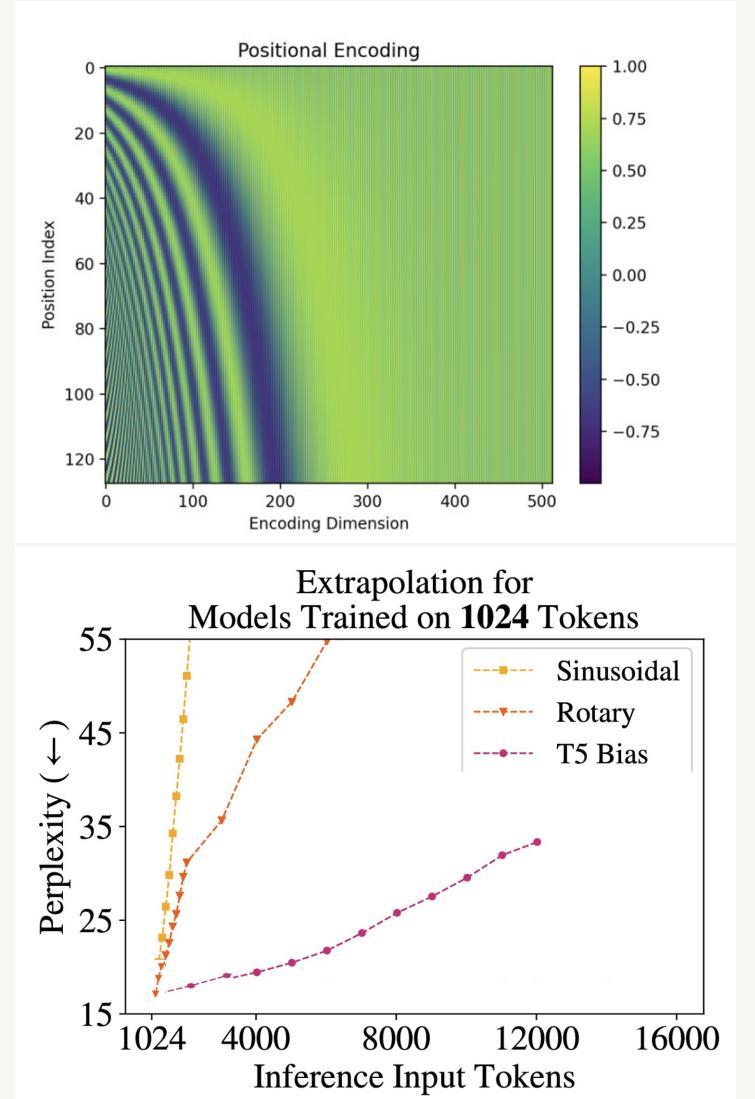
LLMs, like us, do better at tasks with more context.

This means longer input/context length.

- Computing input embeddings  linearly
- Performing FFNN calculations  linearly
- Calculating attention scores  quadratically

Even worse:

Attention cannot perform as well on longer context than it was trained on.



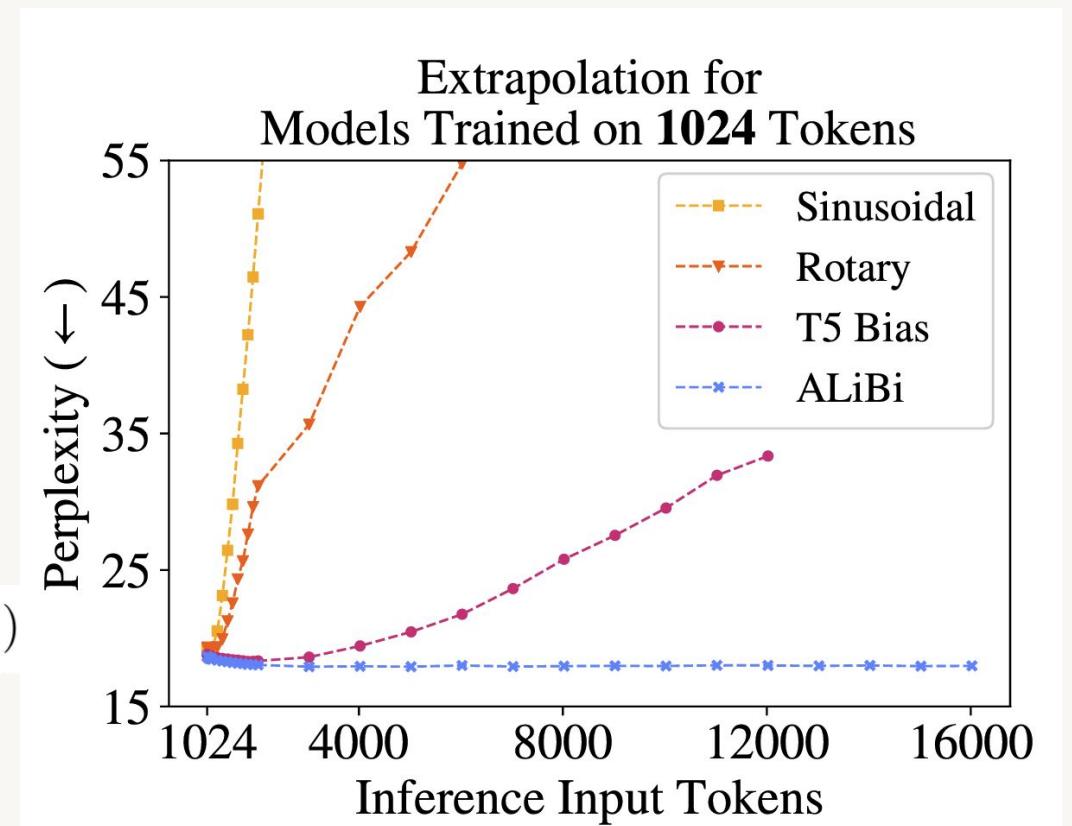
Training short but inference long

You'll need a good Alibi for this one

Attention is all you need... to fix! And just with a linear bias.

$$\begin{bmatrix} q_1 \cdot k_1 \\ q_2 \cdot k_1 & q_2 \cdot k_2 \\ q_3 \cdot k_1 & q_3 \cdot k_2 & q_3 \cdot k_3 \\ q_4 \cdot k_1 & q_4 \cdot k_2 & q_4 \cdot k_3 & q_4 \cdot k_4 \\ q_5 \cdot k_1 & q_5 \cdot k_2 & q_5 \cdot k_3 & q_5 \cdot k_4 & q_5 \cdot k_5 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 & 0 \\ -2 & -1 & 0 \\ -3 & -2 & -1 & 0 \\ -4 & -3 & -2 & -1 & 0 \end{bmatrix} \cdot m$$

$$\text{softmax}(\mathbf{q}_i \mathbf{K}^\top) \rightarrow \text{softmax}(\mathbf{q}_i \mathbf{K}^\top + m \cdot [-(i-1), \dots, -2, -1, 0])$$



Faster calculations

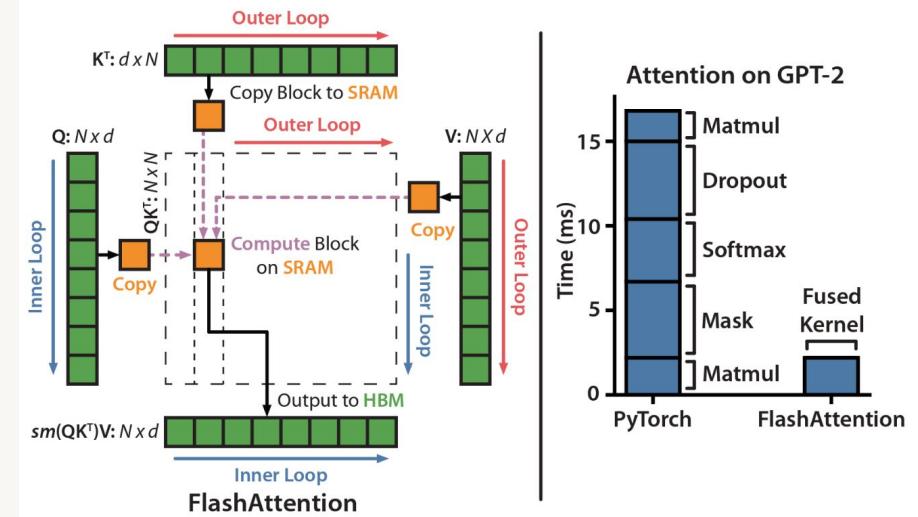
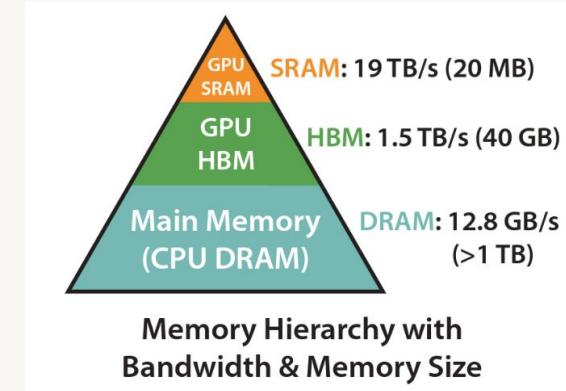
Calculating attention in a flash.

Observation:

- Fastest memory in the GPU is SRAM
- Longer context = larger attention matrices

Problem:

- SRAM is small relative to the attention matrix needed in calculations
- Solution: Flash Attention!
- Attention compute operations are redone, no matrix created!
- More time spent in SRAM, massive performance boost



Source: [Flash Attention](#)



Many queries, fewer keys

Multi-query and Grouped-query Attention

- Multi-Headed Attention

#Queries = #Keys = #Values

Each head can focus on different parts of language.

Inference - slow, accurate.

- Multi-Query Attention

Many Queries, 1 Key, 1 Value

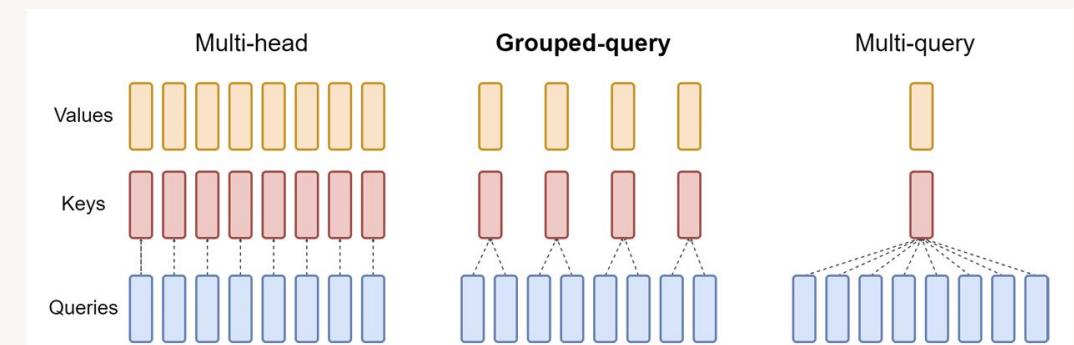
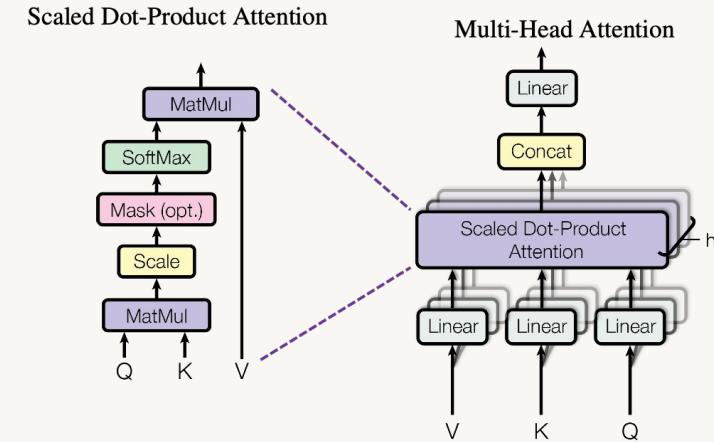
Forcing the model to use different queries

Inference - fast, inaccurate.

- Grouped-Query Attention

#Queries = #/n Keys = #/n Values

Inference - fast, accurate.



Source: [Grouped Query Attention](#)



Improving Model Footprint

Doing more with less



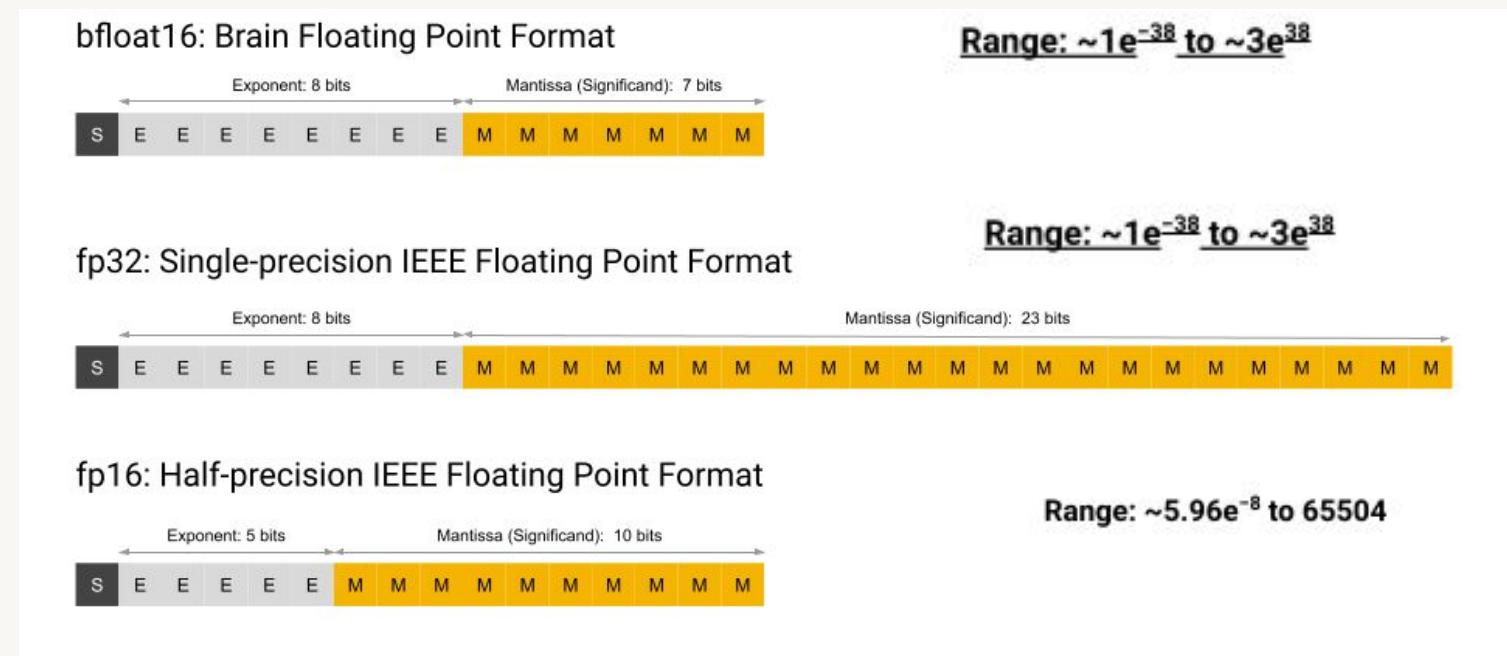
Storing numbers

Billions of parameters. Each a floating point number.

FP16 FP32 – IEEE standards.

Google Brain saw this and created the BF16

- Same range as FP32
- Same size as FP16



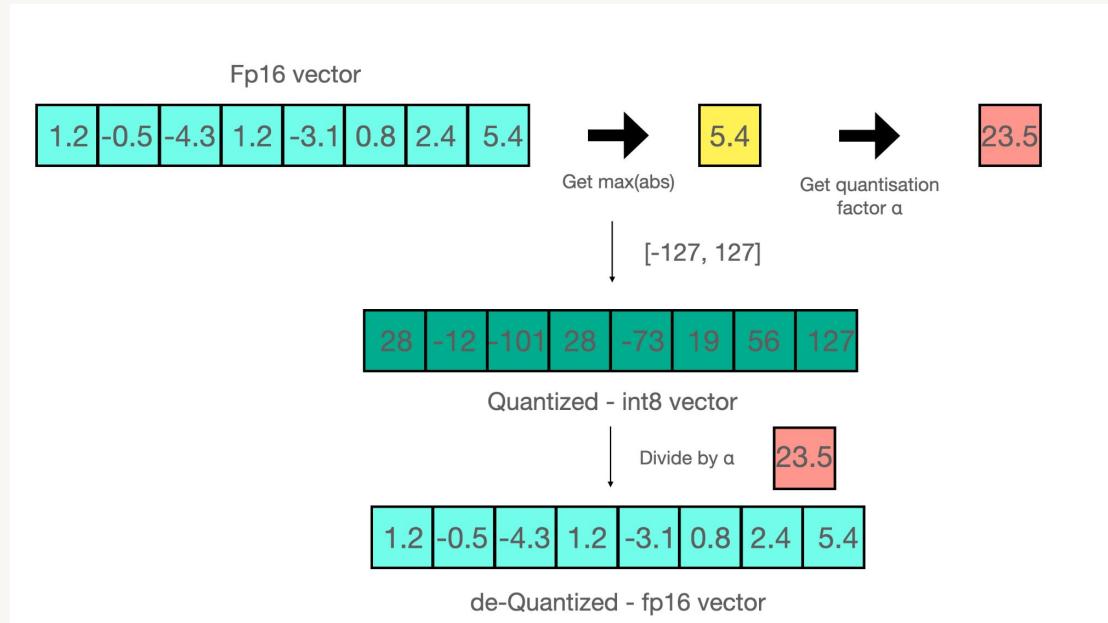
What if we need to go even further?



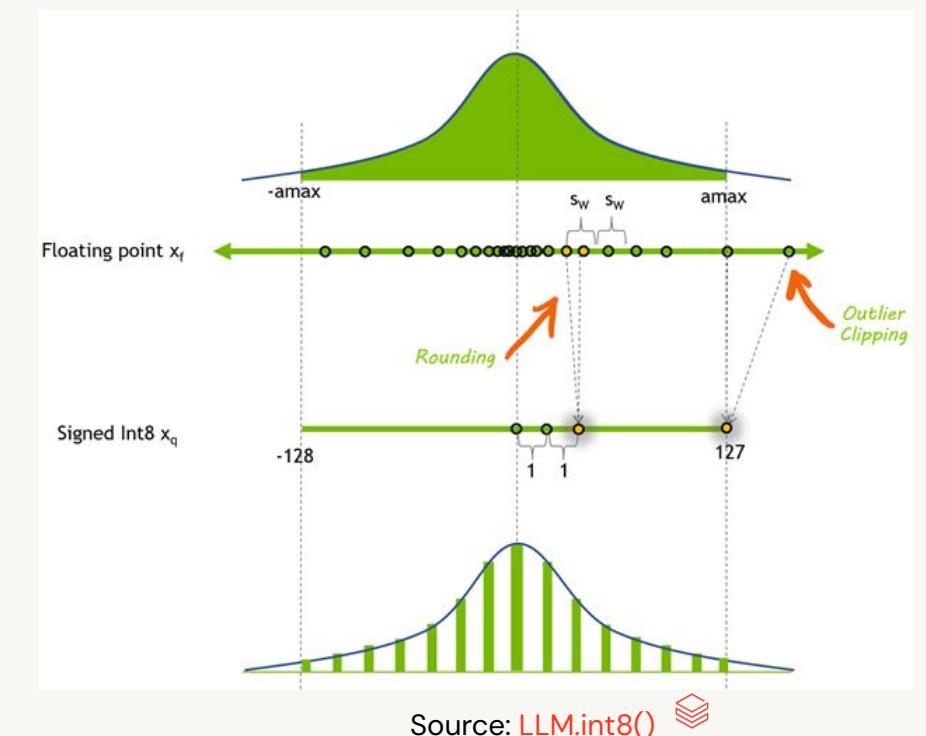
Quantization

Do we need so much precision?

Approximate the values in quantized forms



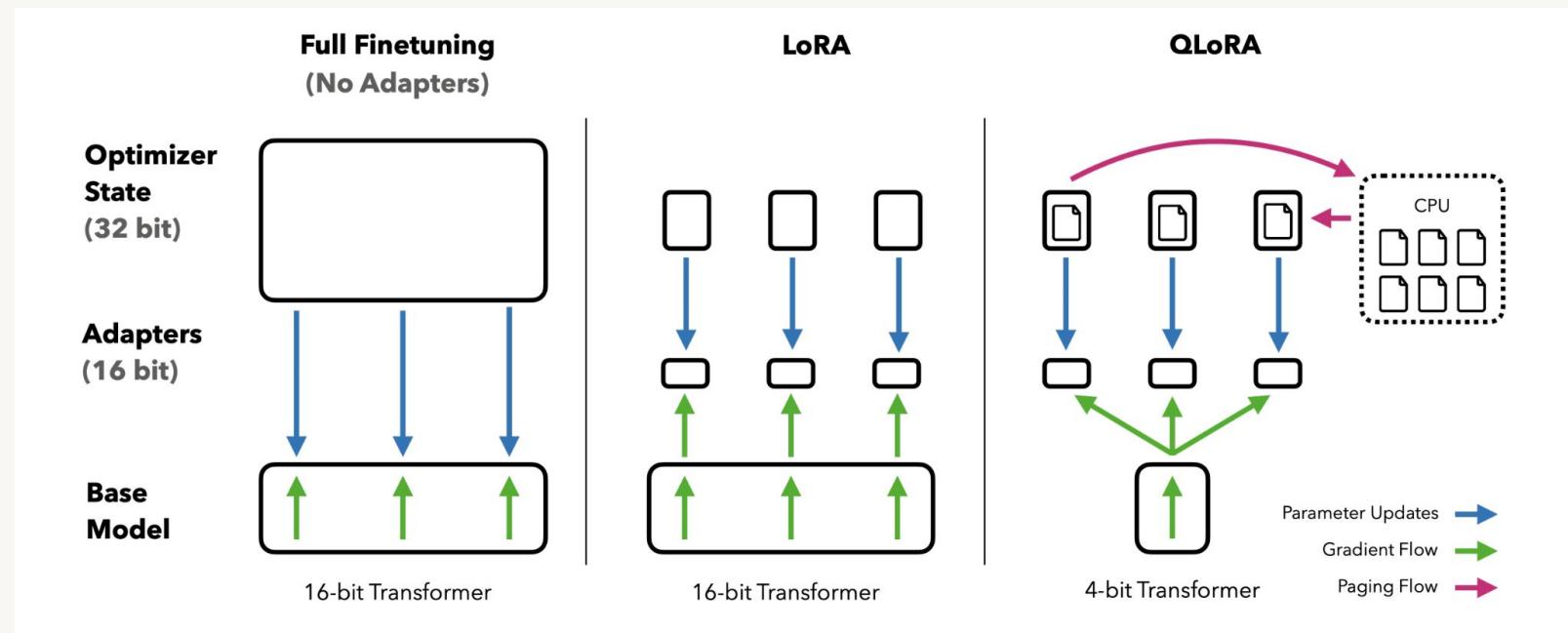
Create quantized functions



QLoRA

Applying quantization to fine tuning

- LoRA was already great!
- QLoRA adds even more:
 - 4-bit quantization
 - Paged optimization



Source: [QLoRA](#)



Multi-LLM Inferencing

Hybrid and Ensemble-based systems

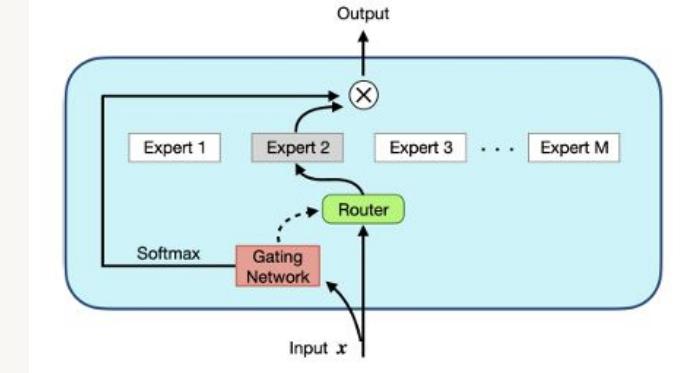


Mixture-of-Experts

A trillion parameters, for a fraction of the training

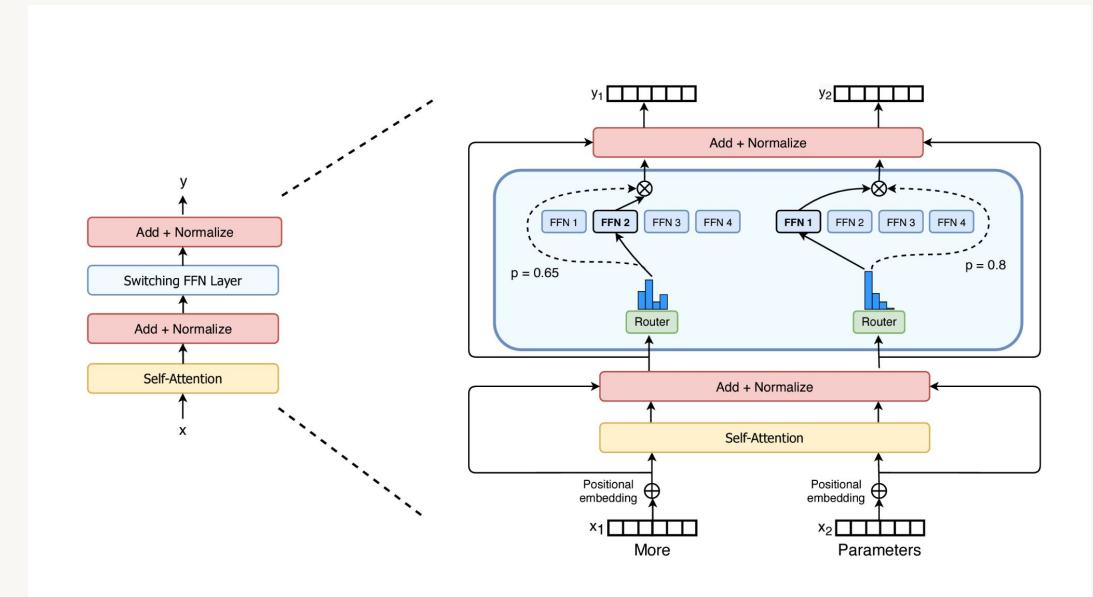
Mixture-of-Experts (MoE):

- Input is sent to a router
- Multiple NNs are trained



Switch Transformer:

- Application of MoE
- Position-wise FFNN are multiplied
- Single attention network



Source: [Switch Transformer](#)

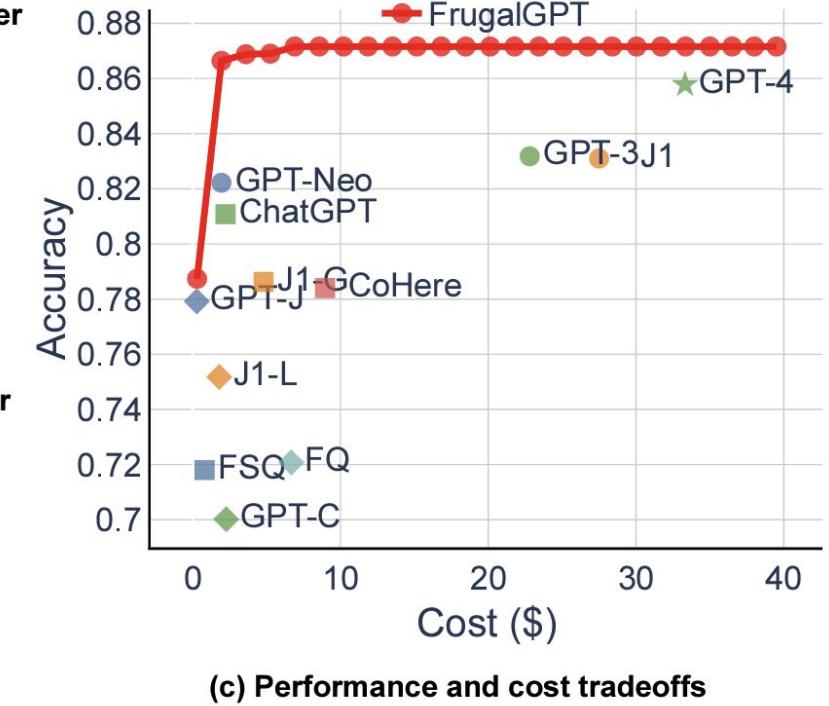
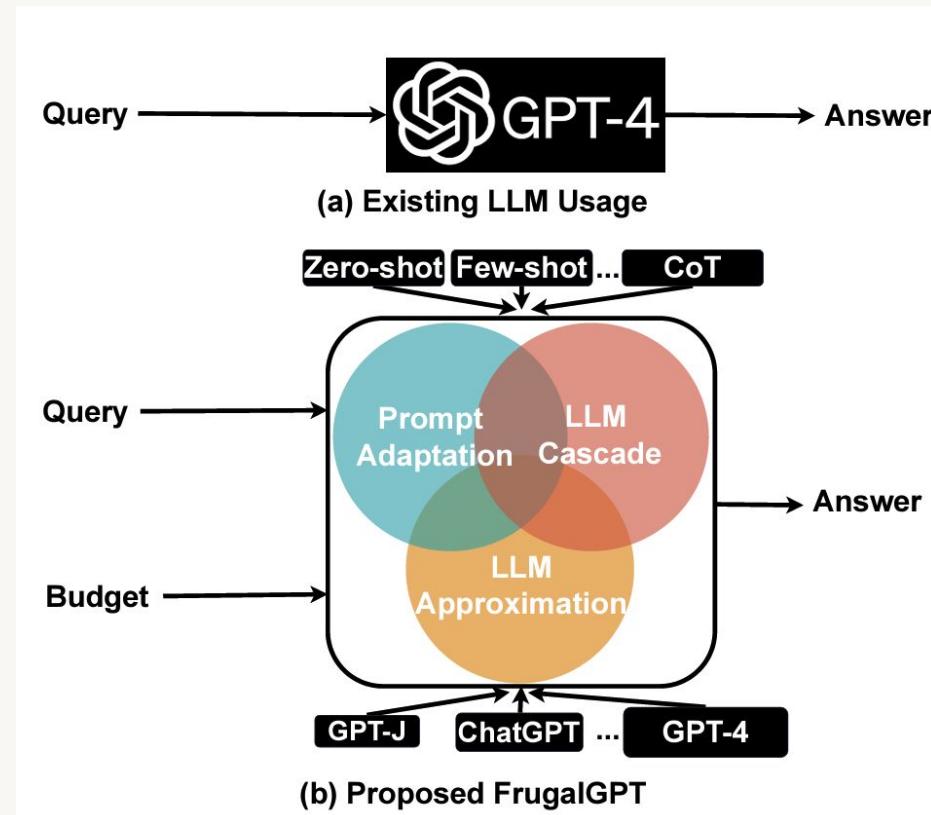


LLM Cascades and FrugalGPT

Improving our resource allocation in LLM inferencing

LLM cascade:

- Send prompts to smallest models
- Gather confidence of response
- If too small, move to a larger model



Source: [FrugalGPT](#)



Current Best Practices

If you want to build now, do it right



Best Practices

Training from scratch:

- ALiBi
- Flash Attention
- Grouped-Query Attention
- Mixture-of-Experts

Fine-tuning/Inferencing:

- LoRA/QLoRA
- FrugalGPT

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 bit.ly/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 today



Module Summary

Deployment Optimizations – What have we learned?

- LLMs are currently outpacing modern compute capacity, necessitating the development of work around solutions and approaches
- Modifying the original approach to attention has allowed for longer contexts, better use of hardware, more efficient calculations
- Quantization helps to store and use massive LLMs on smaller hardware
- Combing LLM inferences of different models allows an effective scale up of parameters with minimal cost changes



Time for some code!



Course Outline

Course Introduction

Module 1 – Transformers: Attention and the Transformer Architecture

Module 2 – Parameter Efficient Fine-Tuning: Doing more with less

Module 3 – Deployment Optimizations: Improving model size and speed

Module 4 – Multi-modal LLMs: Beyond text-based transformers



Module 4

Multi-modal Language Models (MLLMs)

Beyond text-based transformers



Learning Objectives

By the end of this module you will:

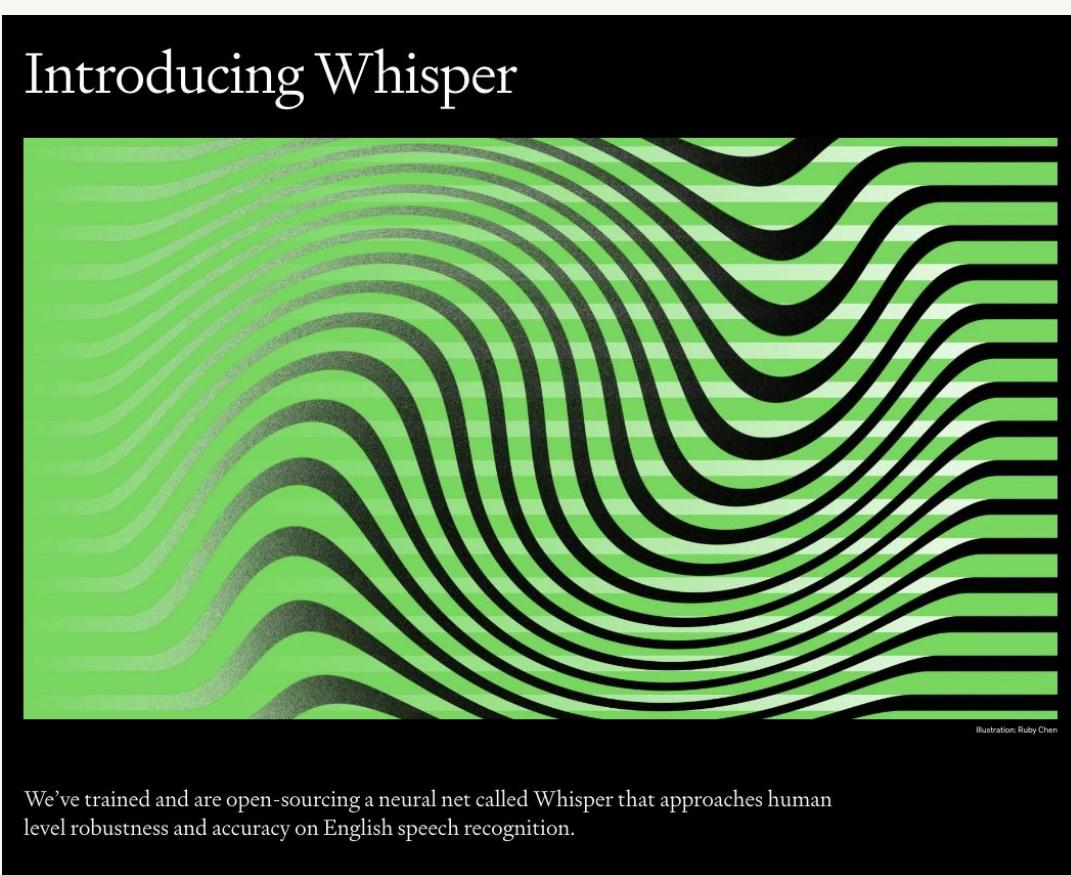
- Survey the broad landscape of multi-modality that leverages LLMs
- Understand how transformers accept non-text inputs, e.g. images and audio
- Examine how multi-modal data is procured
- Discuss limitations of multi-modal LLMs and alternative architectures to transformers
- Identify the wide possibilities of multi-modal applications



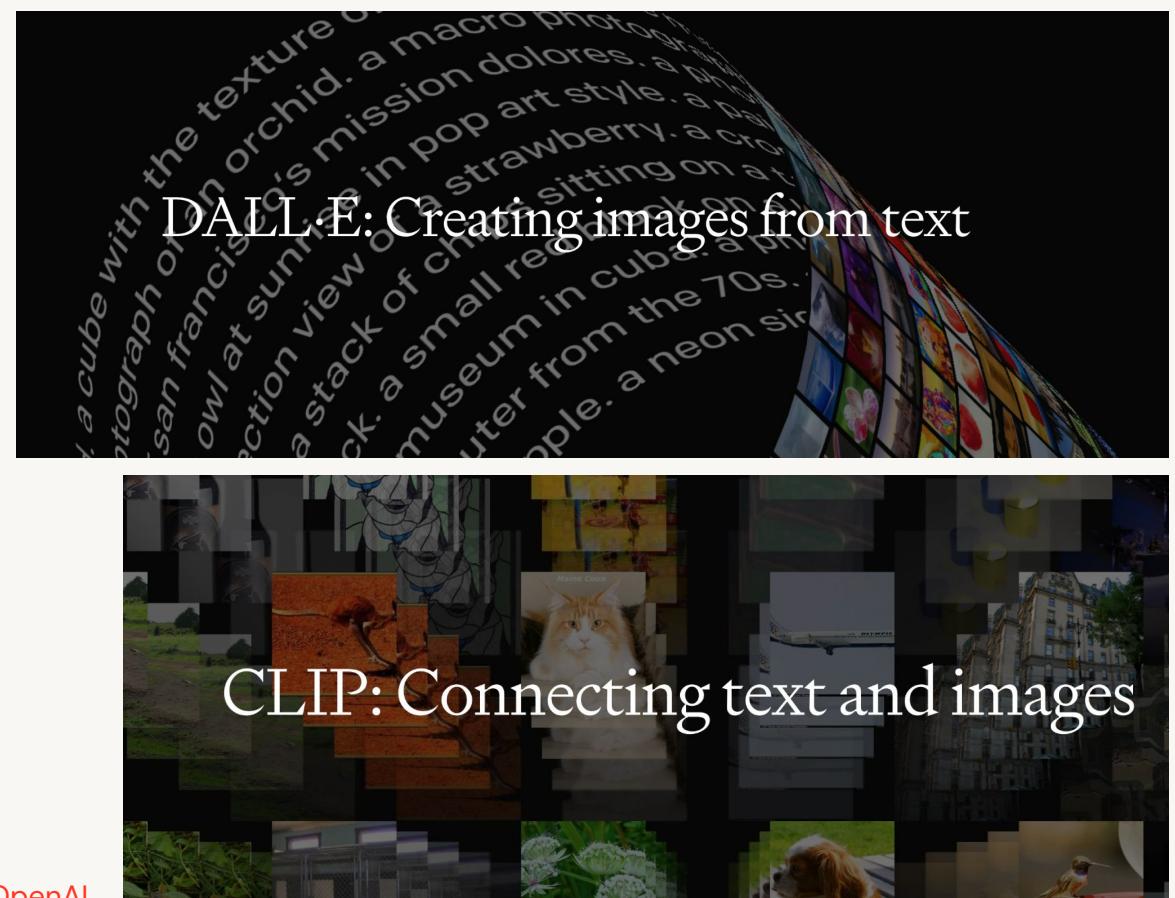
Going beyond uni-modality

LLM-based models that can receive and reason with multimodal info

Source: [Yin et al 2023](#) (released in late June)



Source: [OpenAI](#)



CLIP: Connecting text and images



Multi-modality mirrors how we perceive info

More user-friendly, flexible, and capable

Video-LLaMA

I hear sounds of **footsteps** on the floor in the background, and there is also a **dog barking** in the house.

Does the man wear glasses?

Yes, he wears a **pair of glasses** in the video.

Describe what you hear?

Source: [Zhang et al 2023](#)
[Demo for Video LLaMA](#)

MiniGPT-4

Explain why this meme is funny.

This meme is funny because it shows a dog laying on the floor with its head resting on its paws, as if it is tired or sleepy. The caption, "monday just monday," adds to the humor by suggesting that the dog is feeling the same way as many people do on Mondays, which are often considered to be the most dreaded day of the week. The meme is relatable and humorous, making it a popular one among internet users.

Source: [Zhu et al 2023](#)

A Joke Website
Content of Joke 1
Punchline 1

Write brief HTML/JS to turn this mock-up into a colorful website, where the jokes are replaced by a real joke

HTML:

```
<html>
<head>
<title>A Joke Website</title>
<link rel="stylesheet" type="text/css" href="style.css" />
<body>
<div id="header">
<h1>A Joke Website</h1>
<nav>
<ul>
<li><a href="#">Home</a></li>
<li><a href="#">About</a></li>
<li><a href="#">Contact</a></li>
</ul>
</nav>
</div>
<div id="main">
<h2>Today's Joke</h2>
<p>Why did the tomato turn red?</p>
<p>Because it saw the salad dressing!</p>
</div>
<div id="footer">
<p>© 2023 A Joke Website</p>
</div>
<script src="script.js"></script>
</body>
</html>
```



Chain-of-Thought MLLMs

We can also supply multi-modal information as “in-context”



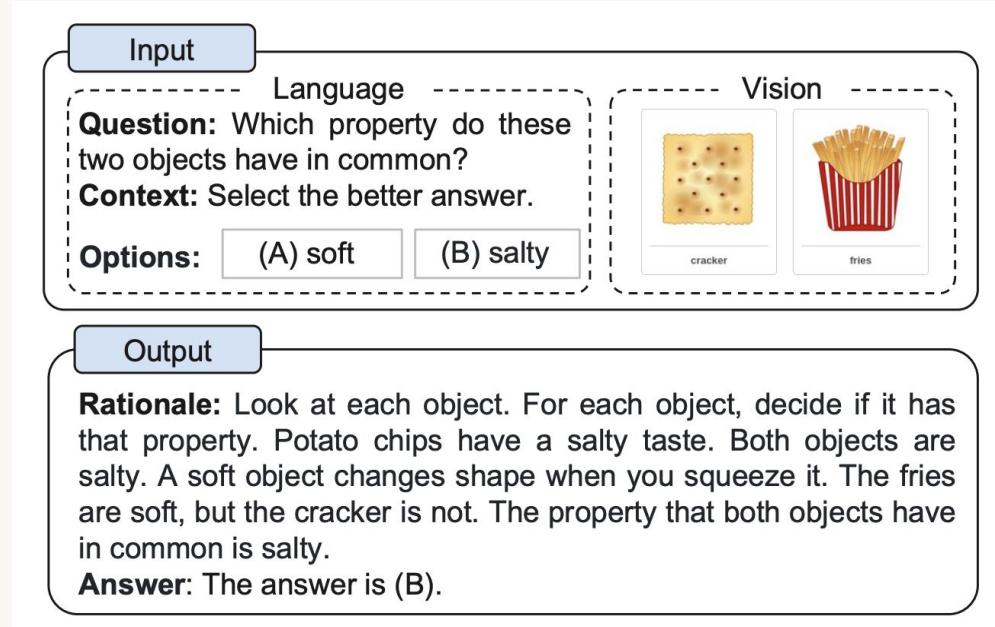
Keyframes Only

Let's think frame by frame.

What happened between the two keyframes?

A man is holding a small dog in his arms, and it appears that he is giving the dog to someone. This suggests that the man might be receiving the dog as a gift or adopting the dog from someone. The scene takes place in front of a wooden counter, and there is a box, possibly containing a holiday decoration, nearby. The man is smiling, indicating that the gift or adoption is a happy occasion for him.

Source: [Himakunthala et al 2023](#)



Source: [Zhang et al 2023](#)



MLMs can process multi-modalities simultaneously



A recording of calm ocean waves.

Write a story to a child about what you see and hear.



PandaGPT

In the image, there is a woman standing on a rock near the water, looking out at the ocean. The ocean is calm and reflects the sky, creating a serene atmosphere. The woman is enjoying the view and listening to the soothing sound of the waves crashing against the shore. As she stands there, she notices a bird flying by, adding a touch of life and movement to the scene. The woman feels peaceful and content, taking in the beauty of the ocean and the natural world around her.



The woman has blonde hair.



What color is the woman's hair?



Source: [Su et al 2023](#) (released in May)
[PandaGPT Demo](#)

MLLMs also call tools/models to finish tasks

HuggingGPT

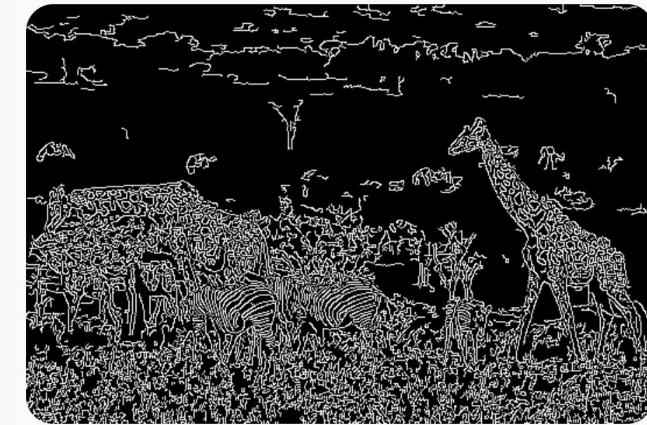


[HuggingGPT Demo](#)

Please generate a canny image based on /examples/f.jpg



I have carefully considered your request and generated a canny image based on /examples/f.jpg. The workflow I used is shown in the following JSON data: [{"task": {"task": "canny-control", "id": 0, "dep": [-1], "args": {"image": "public//examples/f.jpg"}}, "inference result": {"generated image": "/images/4810.png"}, "choose model result": {"id": "canny-control", "reason": "ControlNet is the best model for this task."}}, {"task": {"task": "canny-text-to-image", "id": 1, "dep": [0], "args": {"text": "-0", "image": "public//images/4810.png"}}, "inference result": {"generated image": "/images/a27f.png"}, "choose model result": {"id": "llyasviel/sd-controlnet-canny", "reason": "ControlNet is the best model for this task."}]. The complete path or url of the generated image is '/images/a27f.png'. Please let me know if you need any further assistance.



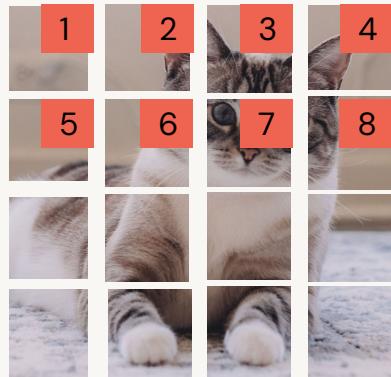
Transformers beyond text

One architecture to rule them all

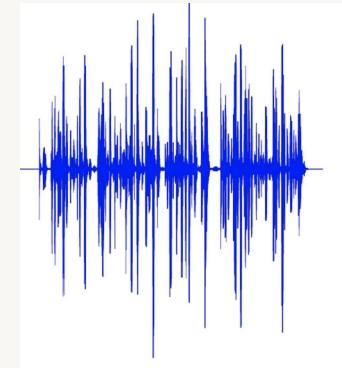


Transformer: a general sequence processing tool

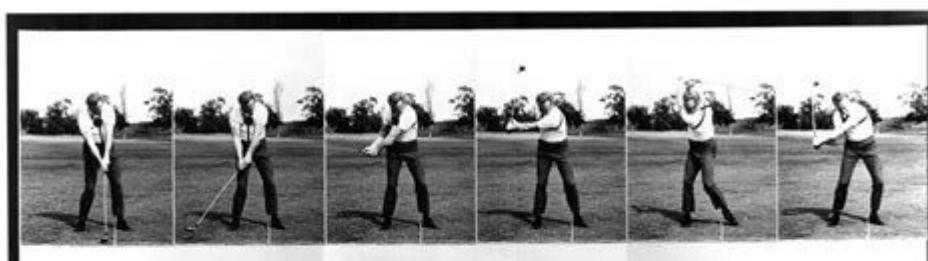
We can treat many things as a sequence



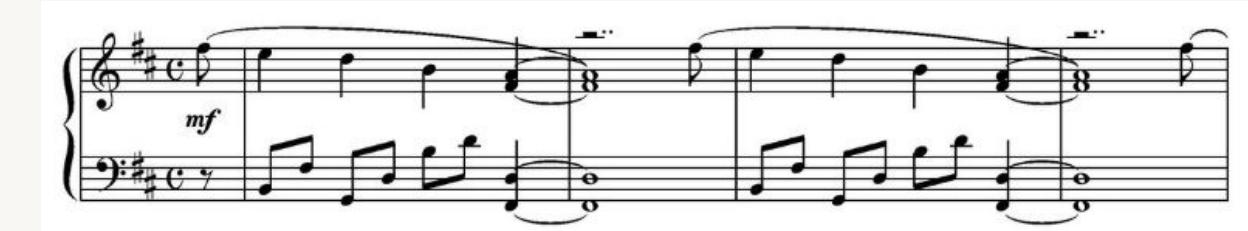
Images



Audio;
[image source](#)



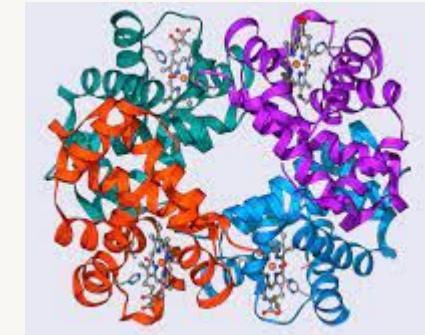
Video frames;
[image source](#)



Music notes;
[image source](#)



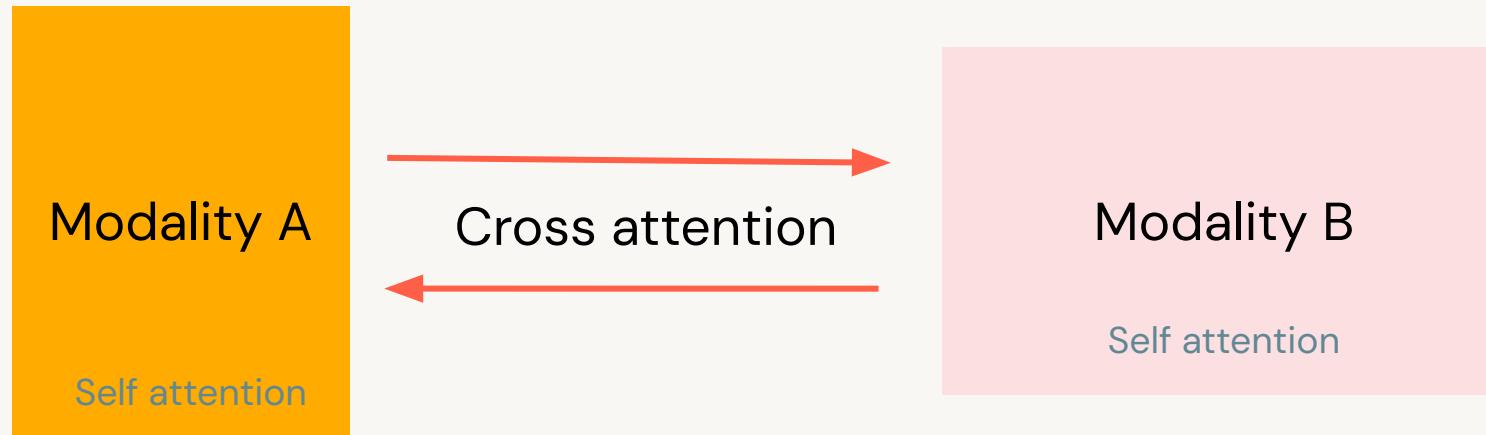
Game actions;
[image source](#)



Protein;
[image source](#)

Cross attention bridges between modalities

Allows different modalities to influence each other



A and B could be:

- Images, audio, text, time series, or any sequence!

Example:

- Stable Diffusion uses cross attention to bridge between text and images

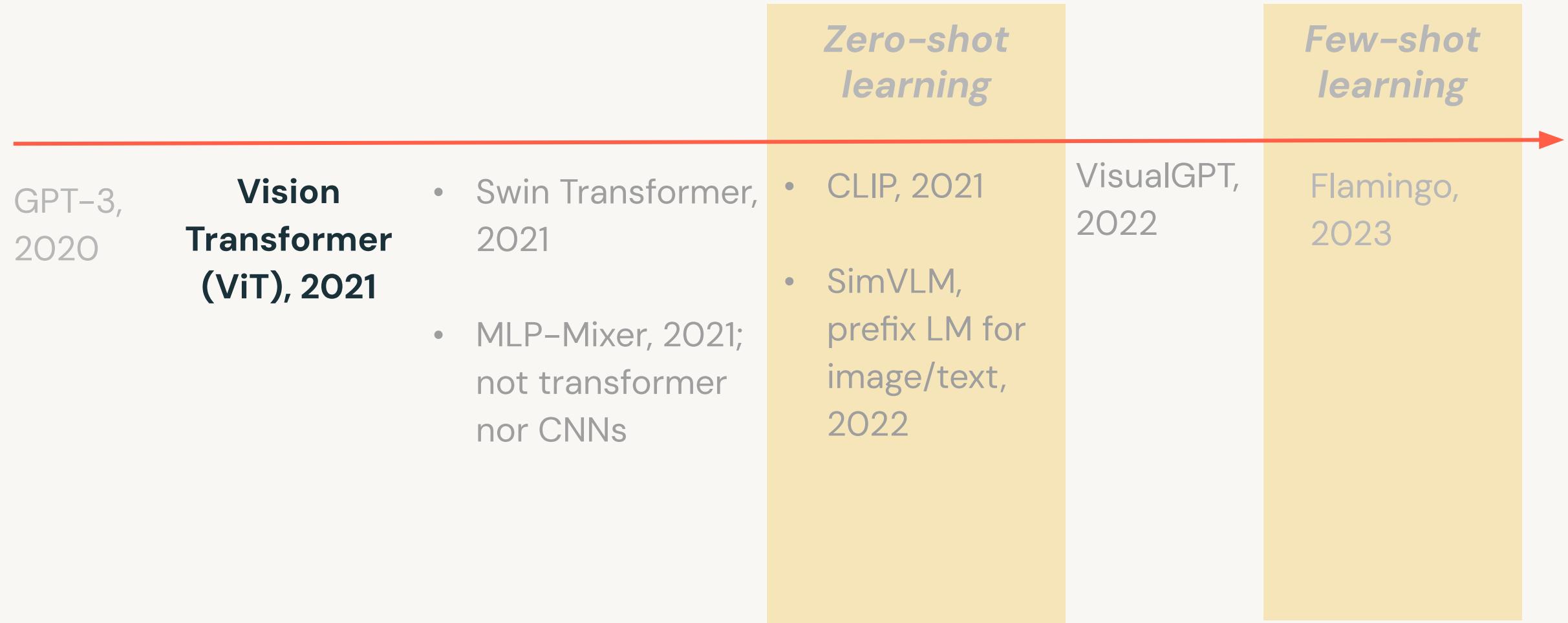


Computer vision



Well-researched area

We need to first understand how to represent images as numbers



We chop an image up into small pixels

A colored image is made up of Red, Blue, Green (RBG) levels

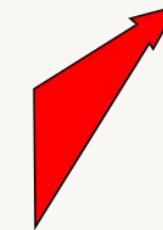


Image credit: Diane Rohrer

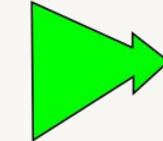
Each pixel range = (0, 256)

#642b4e R: 100 G: 43 B: 78	#7b4360 R: 123 G: 67 B: 96	#936073 R: 147 G: 96 B: 115
#7a4360 R: 122 G: 67 B: 96	#a1727a R: 161 G: 114 B: 122	#c89c8f R: 200 G: 156 B: 143
#945f71 R: 148 G: 95 B: 113	#ca9b91 R: 202 G: 155 B: 145	#f6d0ac R: 246 G: 208 B: 172

100	123	147
122	161	200
148	202	246



43	67	96
67	114	156
95	155	208

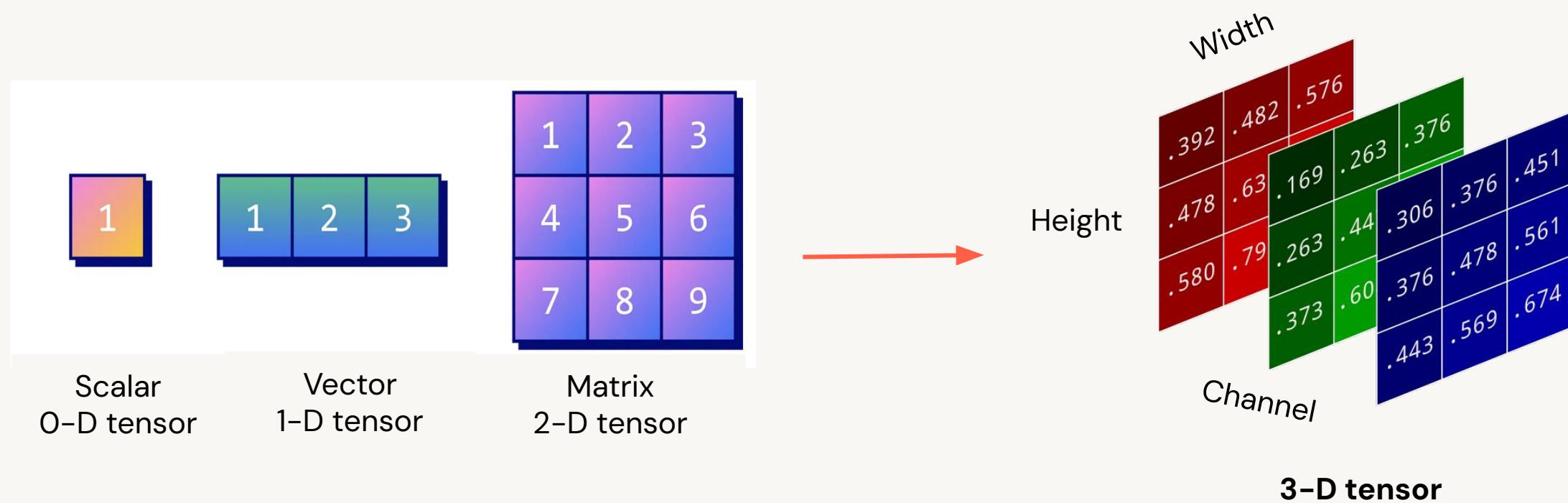


78	96	115
96	122	143
113	145	172



Colored images are 3-D tensors

Grayscale images are 2-D tensors: all 3 channels have the same value

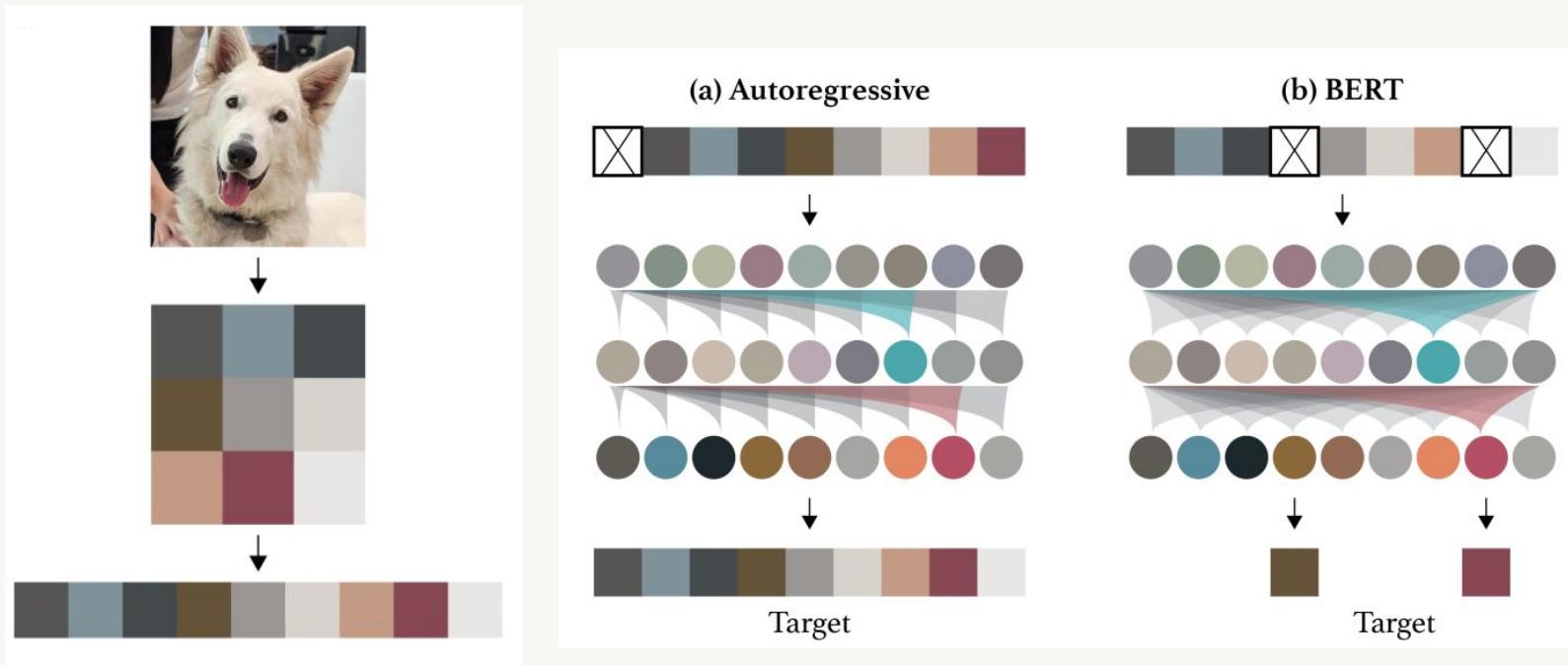


Adapted from [TowardDataScience](#) and [KD Nuggets](#)



Initial idea: Turn pixels into a sequence

Use self-attention to predict the next pixel, instead of word token



Limitations:

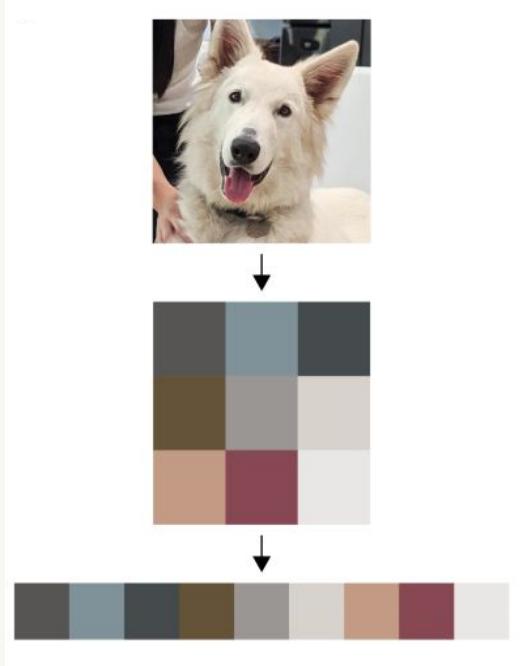
- Lose vertical spatial relationships

Source: [Chen et al 2021](#)



Initial idea: Turn pixels into a sequence

Use self-attention to predict the next pixel, instead of word token



Source: [Chen et al 2021](#)



Source: [David Cocomini 2021](#)

Limitations:

- Lose vertical spatial relationships
- Memory and computational requirements scale quadratically to sequence length, $O(N^2)$



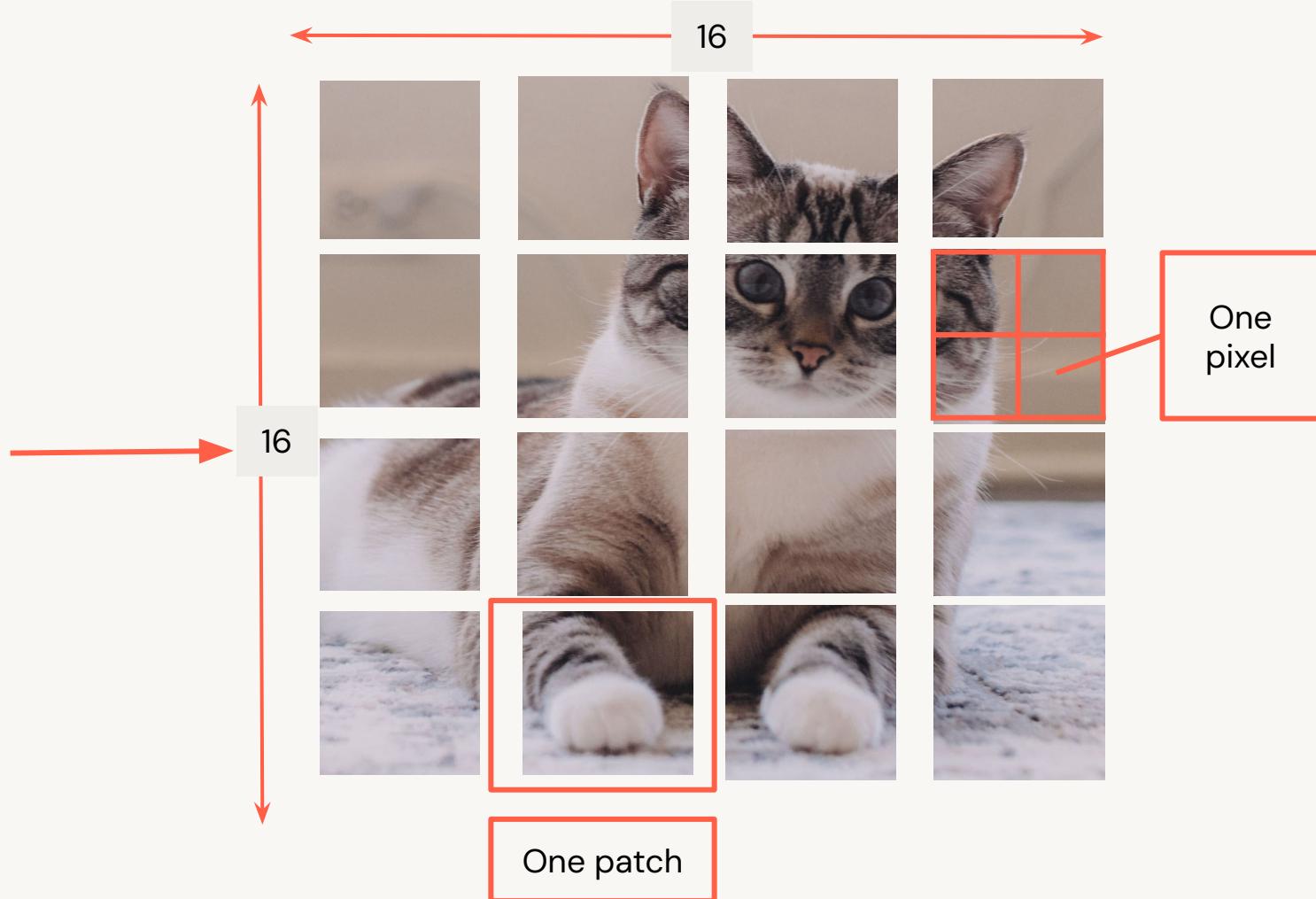
Vision Transformer (ViT)

Computes attention on patches of images: image-to-patch embeddings



Vision Transformer (ViT)

Computes attention on patches of images: image-to-patch embeddings

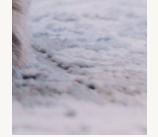


ViT: An image is worth 16x16 words

N input patches
with shape of
 $3 \times \text{Pixel (P)} \times P$



.....

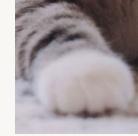


Linear project each patch to D-dimensional vector

N input patches
with shape of
 $3 \times \text{Pixel (P)} \times P$



.....



D-dim patch
embedding

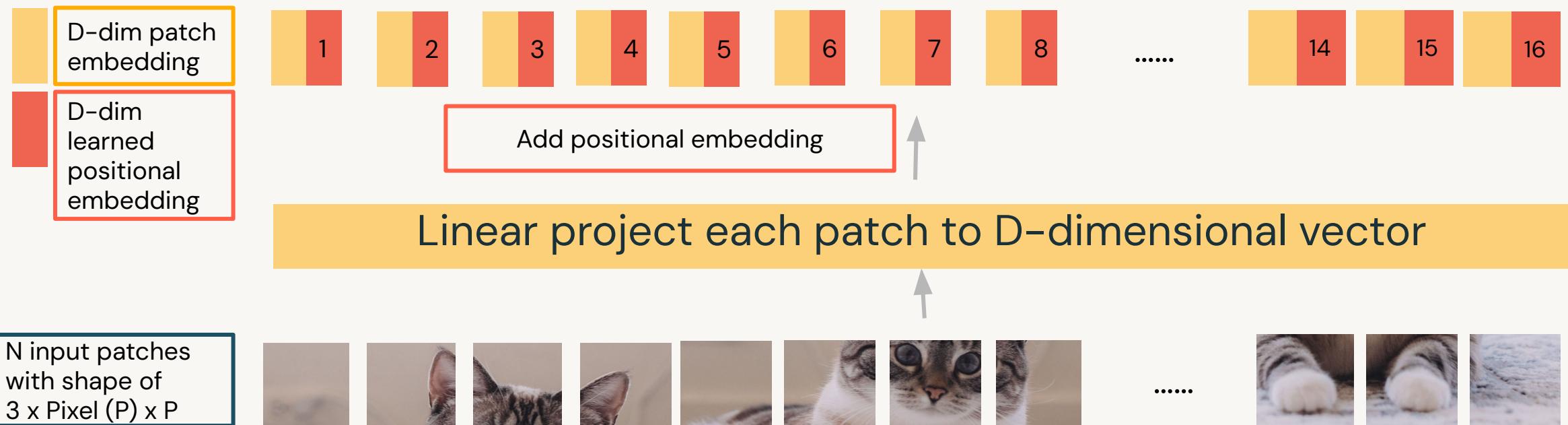


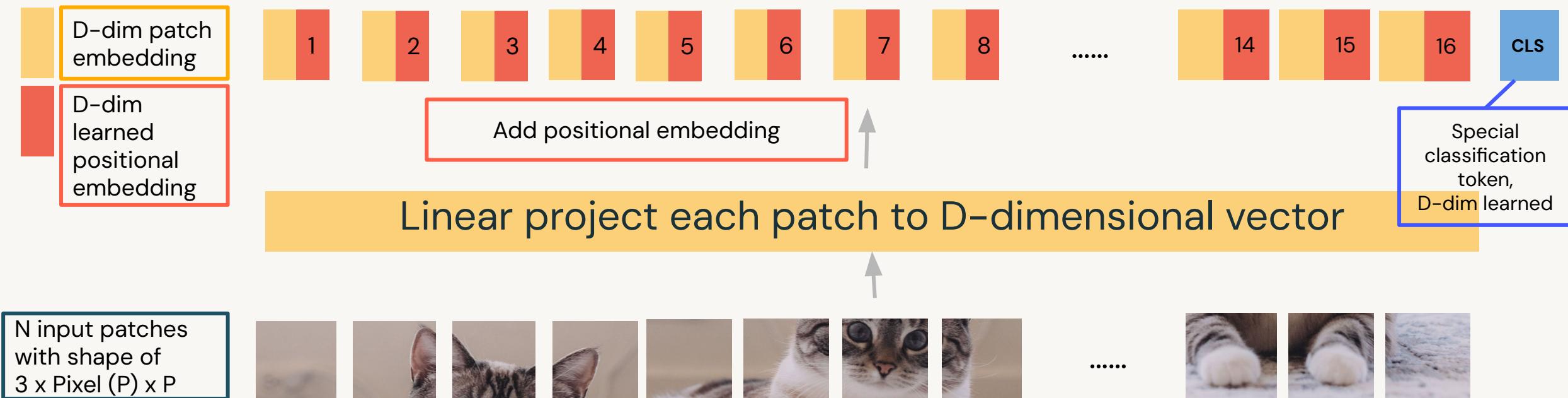
N input patches
with shape of
 $3 \times \text{Pixel (P)} \times P$



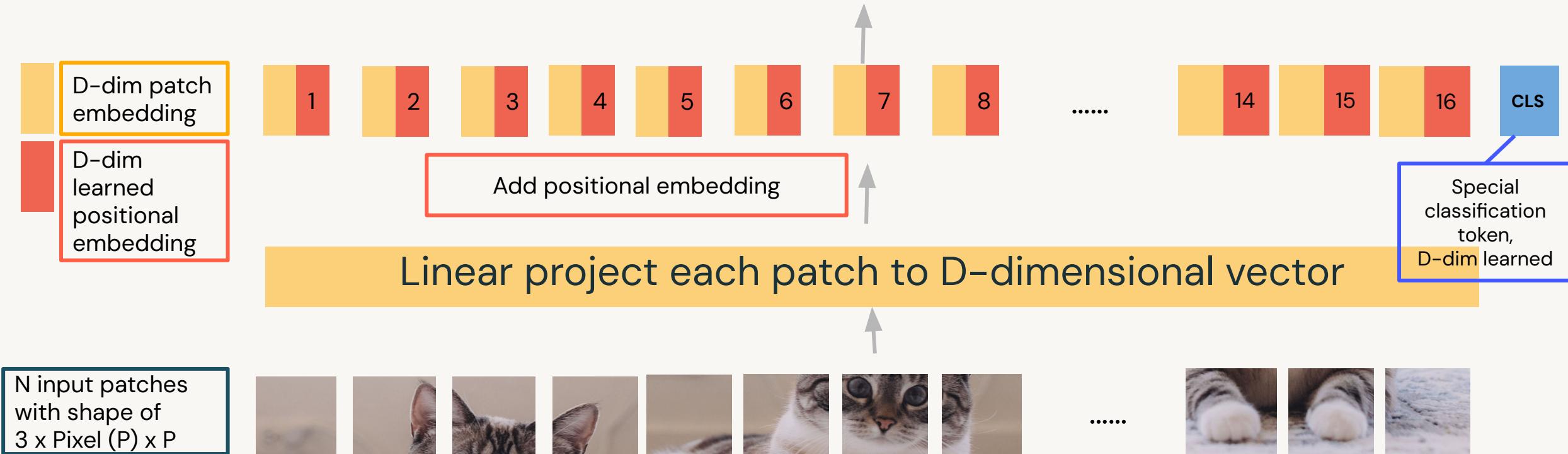
Linear project each patch to D-dimensional vector







Original Transformer



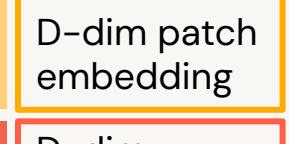
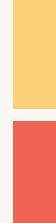
C-dim
classifier
output
vector, where
 $C = \#$ classes



Original Transformer

D-dim patch
embedding

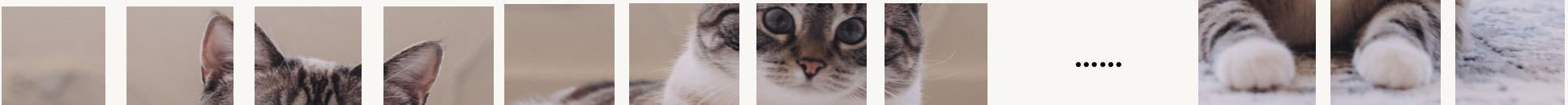
D-dim
learned
positional
embedding



Add positional embedding

Linear project each patch to D-dimensional vector

N input patches
with shape of
 $3 \times \text{Pixel (P)} \times P$

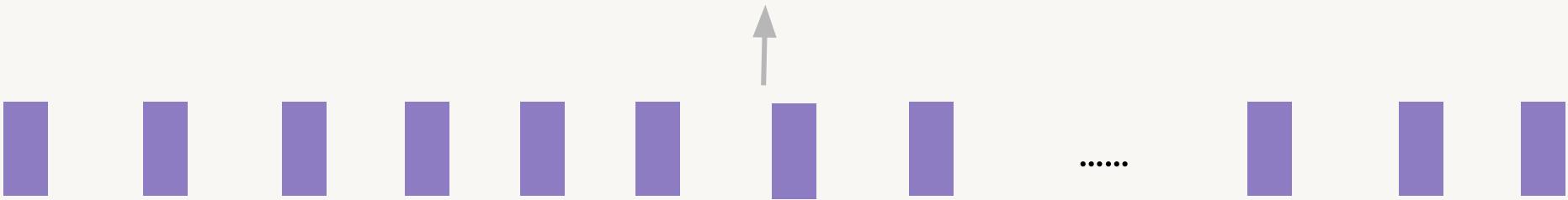


Special
classification
token,
D-dim learned



Label = "cat"

C-dim
classifier
output
vector, where
 $C = \#$ classes



Original Transformer

D-dim patch
embedding

D-dim
learned
positional
embedding



Add positional embedding

Linear project each patch to D-dimensional vector

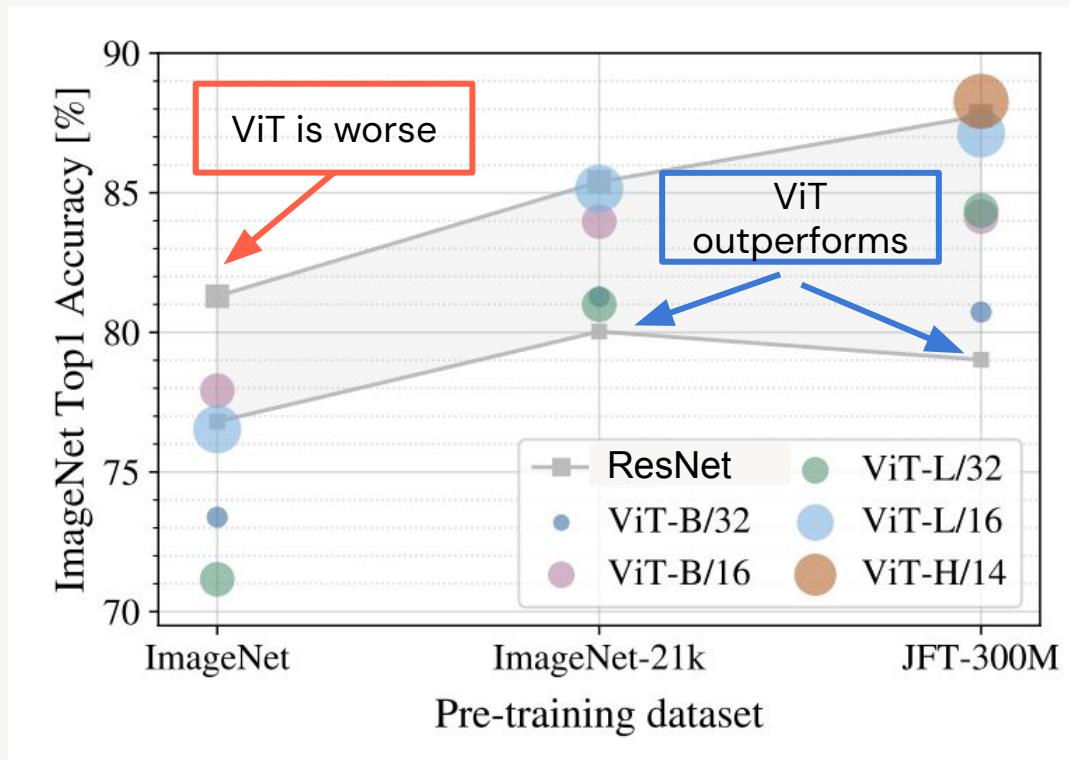
Special
classification
token,
D-dim learned

N input patches
with shape of
 $3 \times \text{Pixel (P)} \times P$



ViT only outperforms ResNets on larger datasets

More computationally efficient than ResNet

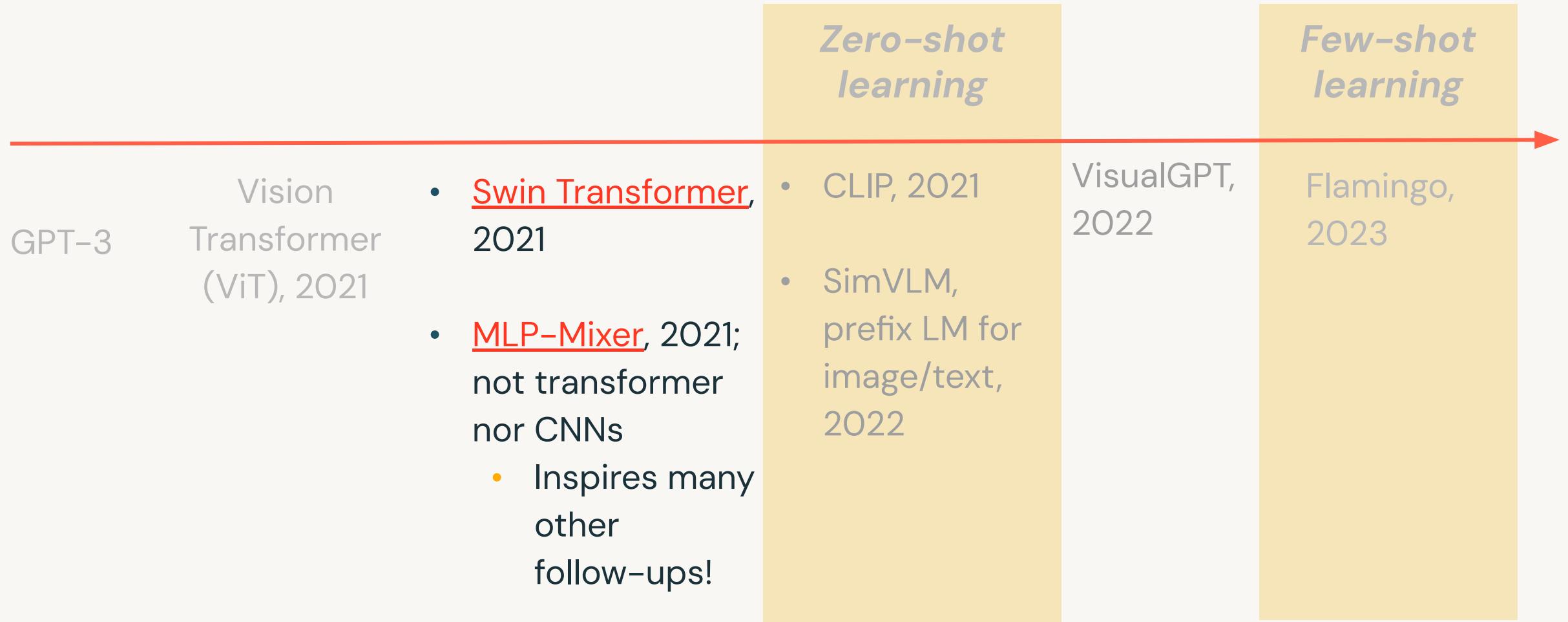


- ViT is ~4 times faster than ResNet to train

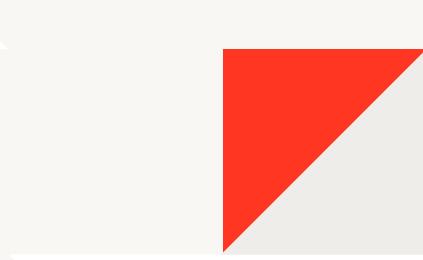
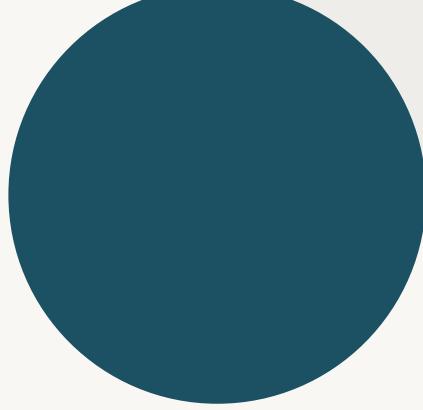
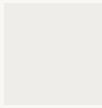
Source: [Dosovitsky et al 2021](#)

Many other vision-text models

Not necessarily revolutionary, but an evolution in computer vision research

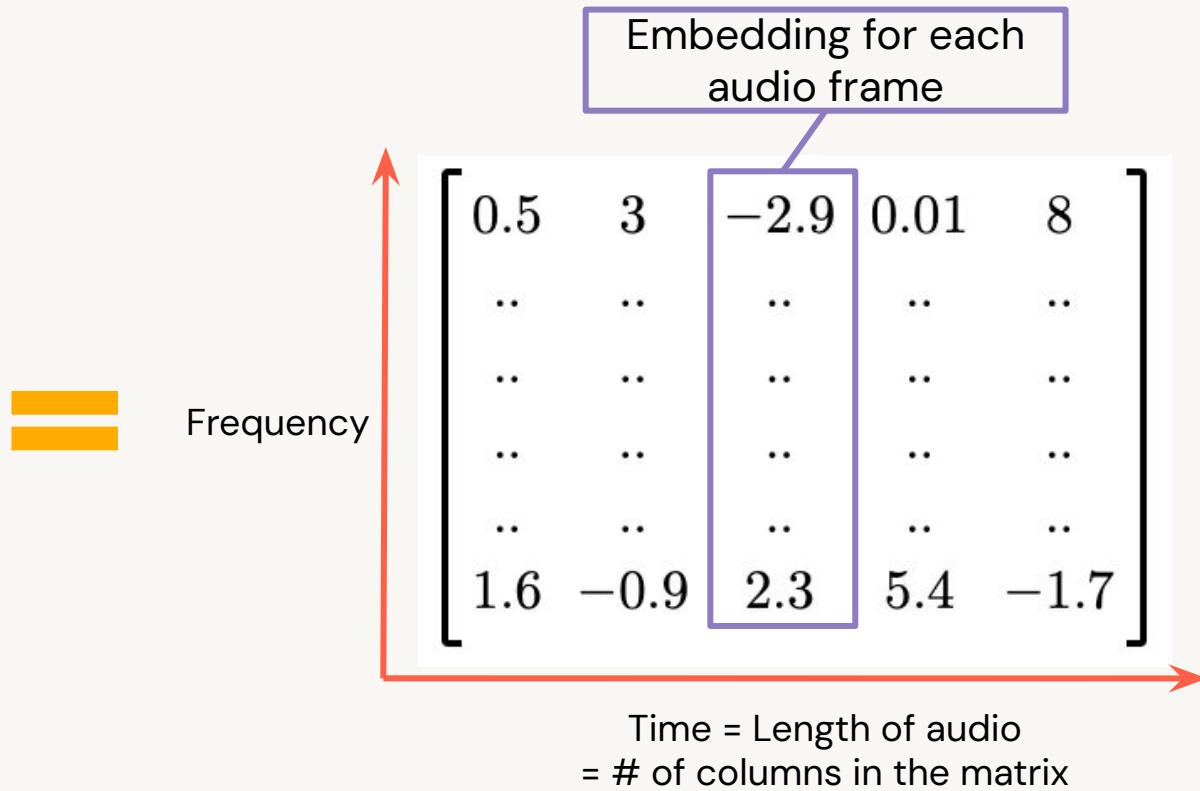
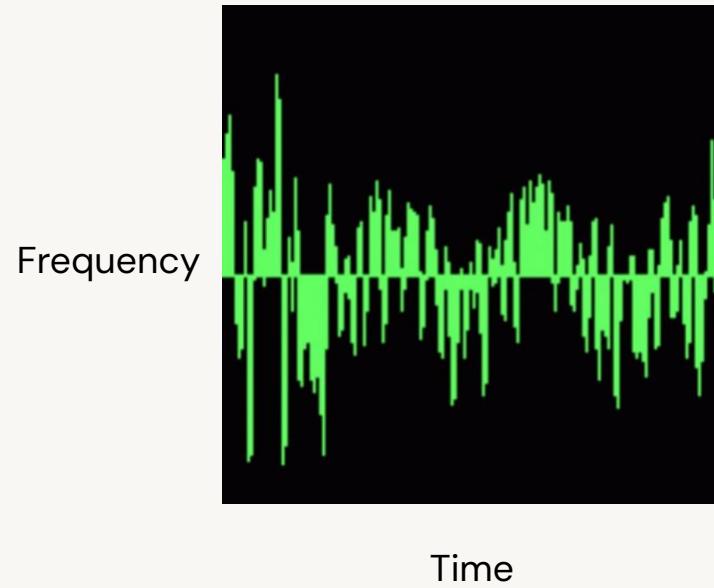


Audio



Audio signals are 2-dim spectrograms

We create embedding vectors for each t-min audio frame



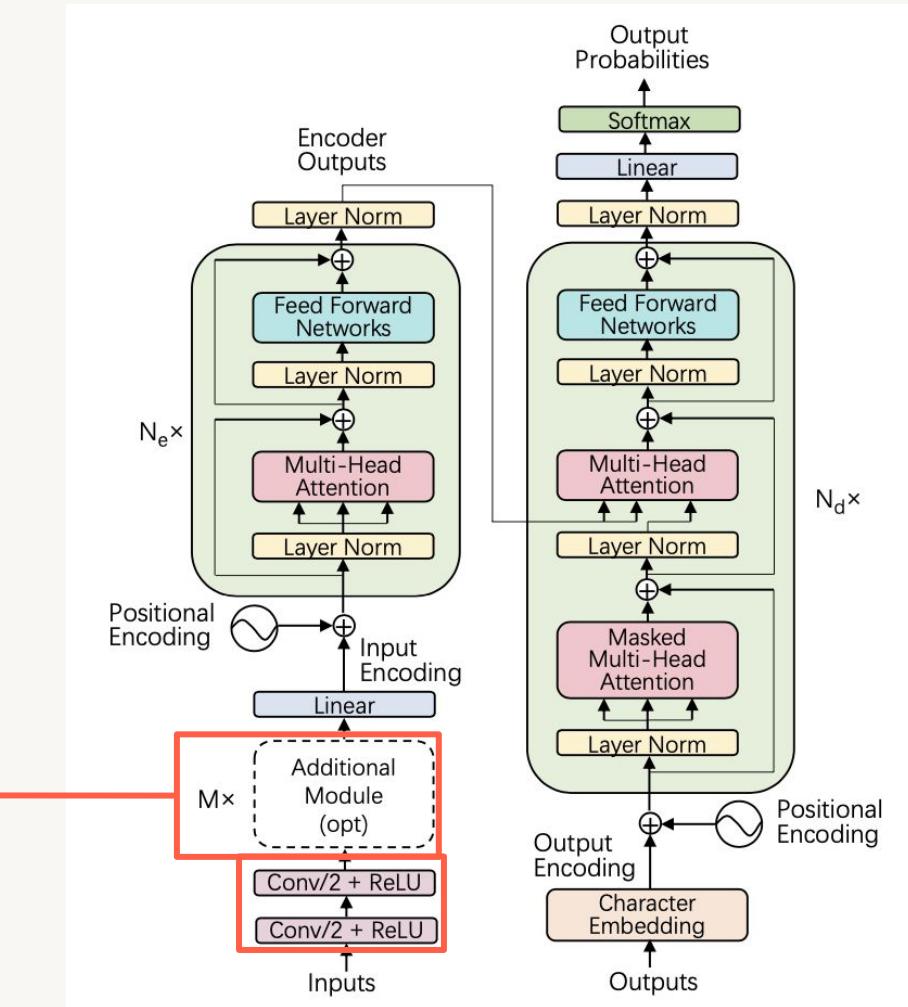
Audio is usually much longer than text length

Need to apply convolution layers with large strides to reduce dimensions

Speech Transformer (2018)

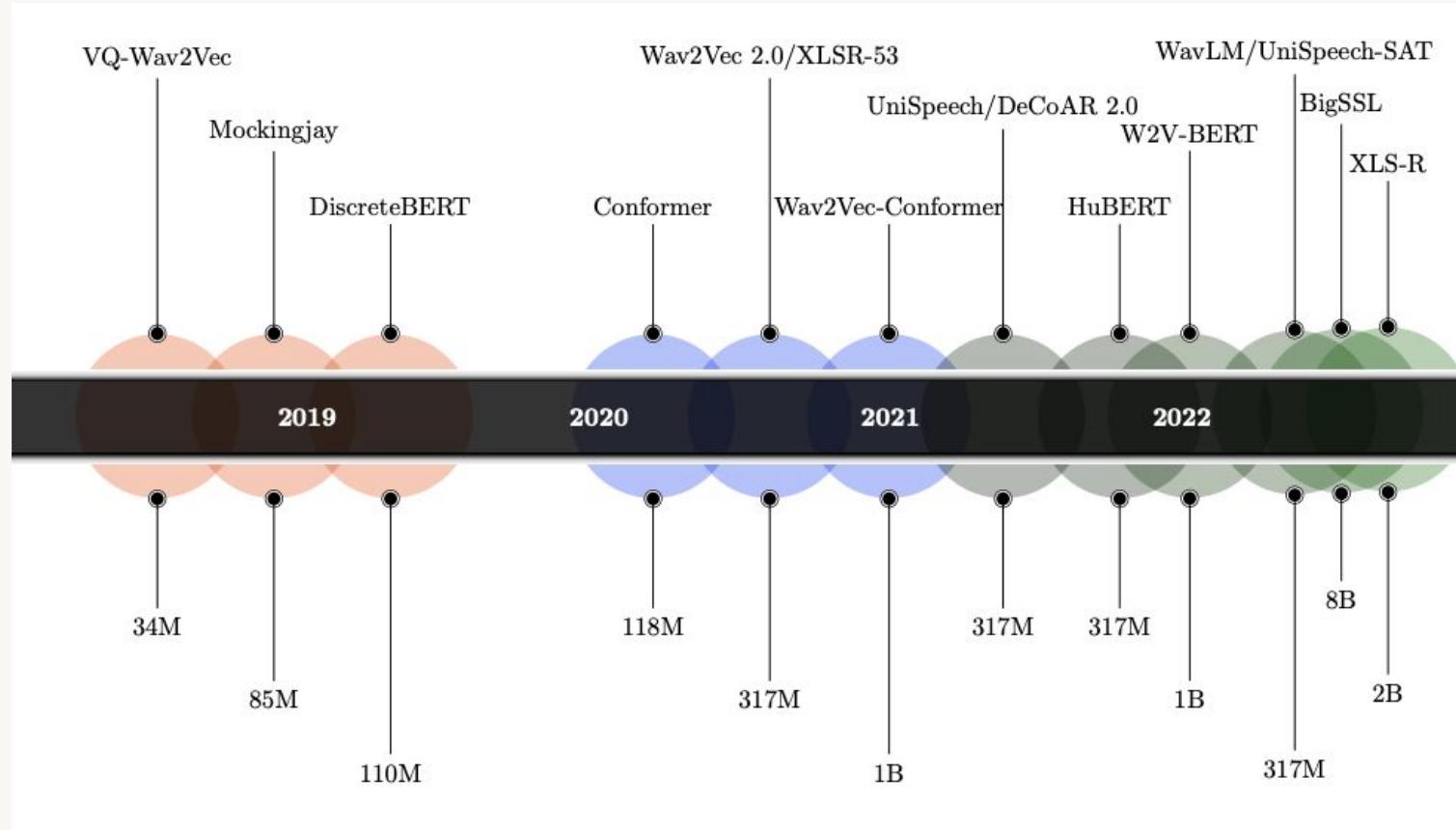
- Encoder-decoder
Transformer

Extract features using
different optional
modules: ResCNN,
ResCNNLSTM,e tc.



Few multi-modal advances

Also much harder: emotion, acoustics, tone, speed, speaker identification



- Most models focus on only text-speech, speech-text, or speech-speech
- Data2vec (MetaAI, 2022)
 - Self-supervised algorithm for speech, vision, and text
 - The only one so far?



Training Data for MLLMs



Hand-crafted training data

Text-audio or text-video data is much harder to procure



Female cop talking on walkietalkie, responding emergency call, crime prevention

Source: [WebVid](#)

Example of detailed video description



###Human: Portray the video with a rich, descriptive narrative, capturing its atmosphere and events.

###Assistant: First, we see a young woman with long brown hair wearing a backpack standing on the edge of a cliff, looking at the beautiful mountain scenery. She is in awe and stands there for a few seconds. Next, we see her turning around slowly, with her arms open wide. Then, we see her taking a few steps back, still facing the canyon, with her arms still open wide. She seems to be soaking in the grandeur of the view. Finally, she stands confidently at the edge of the cliff, her backpack still on her back, as if she has conquered the mountain. Throughout the video, we see the vast expanse of the canyon with the mountains in the background. The sky is overcast in some parts and clear blue in others, creating a beautiful contrast. Overall, the young woman seems to be thrilled with the view and enjoys every bit of it with open arms.

Source: [VideoChat](#)



Structured Scene Description (FAMOUS)

Focus: Man holding a chainsaw preparing to cut down the Christmas tree

Action: Man getting ready to cut down the tree

Mood: Anticipation

Objects: Man, chainsaw, trees in the forest, brown dirt

Setting: In the forest among trees with brown dirt on the ground

Dense Scene Description

In a dense forest, a man is preparing to cut down a Christmas tree, creating a mood of anticipation. The man, wearing appropriate protective gear, holds a chainsaw, ready to act. The scene is marked by the contrast between the excitement of upcoming Christmas celebrations and the focus of cutting down the tree. Key elements in the scene, including the man, the chainsaw, the trees, and the brown dirt, contribute to this poignant snapshot of the holiday season, highlighting the intersection of tradition and respect for nature.

Source: [Himakunthala et al 2023](#)



Instruction-tuned, hand-crafted data

```
[  
  {  
    "id": "000000033471",  
    "image": "000000033471.jpg",  
    "conversations": [  
      {  
        "from": "human",  
        "value": "<image>\nWhat are the colors of the bus in the  
        image?"  
      },  
      {  
        "from": "gpt",  
        "value": "The bus in the image is white and red."  
      },  
      {  
        "from": "human",  
        "value": "What feature can be seen on the back of the bus?"  
      },  
      {  
        "from": "gpt",  
        "value": "The back of the bus features an advertisement."  
      },  
      {  
        "from": "human",  
        "value": "Is the bus driving down the street or pulled off to  
        the side?"  
      },  
      {  
        "from": "gpt",  
        "value": "The bus is driving down the street, which is  
        crowded with people and other vehicles."  
      }  
    ]  
},  
]
```

conversations (list)	image (string)	id (string)
[{ "from": "human", "value": "Provide a brief description of the given image.\n<image>" }, { "from": "gpt", "value": "olive oil is a healthy ingredient used liberally ." }]	"GCC_train_002582585.jpg"	"GCC_train_002582585"
[{ "from": "human", "value": "<image>\nWrite ..."}]	"GCC_train_002429825.jpg"	"GCC_train_002429825"

Source: [CC-3M](#)

Source: [LLaVa](#)

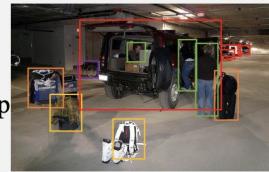


Instruction-tuned, model-generated data

Actually: manually design examples first, then ask model to generate more

Context type 1: Captions

A group of people standing outside of a black vehicle with various luggage.
Luggage surrounds a vehicle in an underground parking area
People try to fit all of their luggage in an SUV.
The sport utility vehicle is parked in the public garage, being packed for a trip
Some people with luggage near a van that is transporting it.



Context type 2: Boxes

person: [0.681, 0.242, 0.774, 0.694], person: [0.63, 0.222, 0.686, 0.516], person: [0.444, 0.233, 0.487, 0.34], backpack: [0.384, 0.696, 0.485, 0.914], backpack: [0.755, 0.413, 0.846, 0.692], suitcase: [0.758, 0.413, 0.845, 0.69], suitcase: [0.1, 0.497, 0.173, 0.579], bicycle: [0.282, 0.363, 0.327, 0.442], car: [0.786, 0.25, 0.848, 0.322], car: [0.783, 0.27, 0.827, 0.335], car: [0.86, 0.254, 0.891, 0.3], car: [0.261, 0.101, 0.787, 0.626]

Response type 1: conversation

Question: What type of vehicle is featured in the image?
Answer: The image features a black sport utility vehicle (SUV).
Question: Where is the vehicle parked?
Answer: The vehicle is parked in an underground parking area, likely in a public garage.
Question: What are the people in the image doing?
Answer: The people are trying to fit all of their luggage into the SUV, likely preparing for a trip.

Response type 2: detailed description

The image is an underground parking area with a black sport utility vehicle (SUV) parked. There are three people in the scene, with one person standing closer to the left side of the vehicle, another person in the middle, and the third person on the right side. They are all working together to pack their luggage into the SUV for a trip.

In the parking area, various luggage items are scattered around the vehicle. There are two backpacks, one located near the left rear wheel and the other closer to the right side of the vehicle. Additionally, there are two suitcases, one on the right side of the car and another further away near the center of the parking area. A bicycle can also be seen on the left side of the vehicle.

Other cars are parked around the main SUV, with one car positioned behind it and slightly to the left, another behind and slightly to the right, and the third car further behind on the right side.

Captions

A man is skiing in the open snow covered hills
A skier is making his way into the snow.
A skier on the foothills of a huge mountain range.
A skier looks at mountains while standing near a trail sign.
a single person skiing by an area with a lot of bushes

Conversation

Question:
What is the skier doing?

====

Answer:
The skier is skiing in the snow-covered hills, making his way through the snow and enjoying the beautiful mountain scenery. They are also standing near a trail sign, which indicates their location on the mountain.

====

Question:
Are there any other people or skiers visible in the image?

====

Answer:
No, there is only a single person skiing in the image, and they are surrounded by an area with a lot of bushes.

Source: [Liu et al 2023](#)



LAION-5B: open source image-text data

Original data: [Common Crawl](#); filtered with OpenAI's CLIP model

The screenshot shows a Databricks dataset card for "laion2B-en". At the top, it displays the dataset name, a paw icon, a license badge for CC-BY-4.0, and a "like" count of 119. Below this, there are tabs for "Dataset card", "Files", and "Community" (with 6 members). The "Dataset Viewer" tab is selected, showing a dropdown menu set to "train (2.32B rows)". To the right, there are links for "Auto-converted to Parquet" and "API". The main area is a table viewer showing a sample row from the dataset. The columns are labeled: SAMPLE_ID (int64), URL (string), TEXT (string), HEIGHT (int32), and WIDTH (int32). The first row shows a sample ID of 2,641,080,021,034, a URL pointing to a Shopify product page for beach umbrellas, and a text description of the image. The second row shows a sample ID of 1,069,682,003,121, a URL pointing to a BMW blog post, and a text description of the image.

SAMPLE_ID (int64)	URL (string)	TEXT (string)	HEIGHT (int32)	WIDTH (int32)
2,641,080,021,034	"https://cdn.shopify.com/s/files/1/0017/3621/2538/products/blue-beach-umbrellas-point-of-rocks-crescent-beach-siesta-key-shawn-mcloughlin_32d72f5b-5e55-42f9-bfcfd6fa8d239beb_300x300.jpg?v=1524171284"	"Blue Beach Umbrellas, Point Of Rocks, Crescent Beach, Siesta Key - Spiral Notebook"	231	300
1,069,682,003,121	"http://cdn.bmwblog.com/wp-content/uploads/2016/02/BMW-M2-M-Performance-...	"BMW-M2-M-Performance-Dekor-Long-Beach-Blue-05"	120	120

Disclaimer:

Contains mostly copyrighted images. LAION doesn't claim ownership.

Source: [LAION](#)



X-shot learning for MLLMs



Computer Vision



X-shot learning to the rescue?

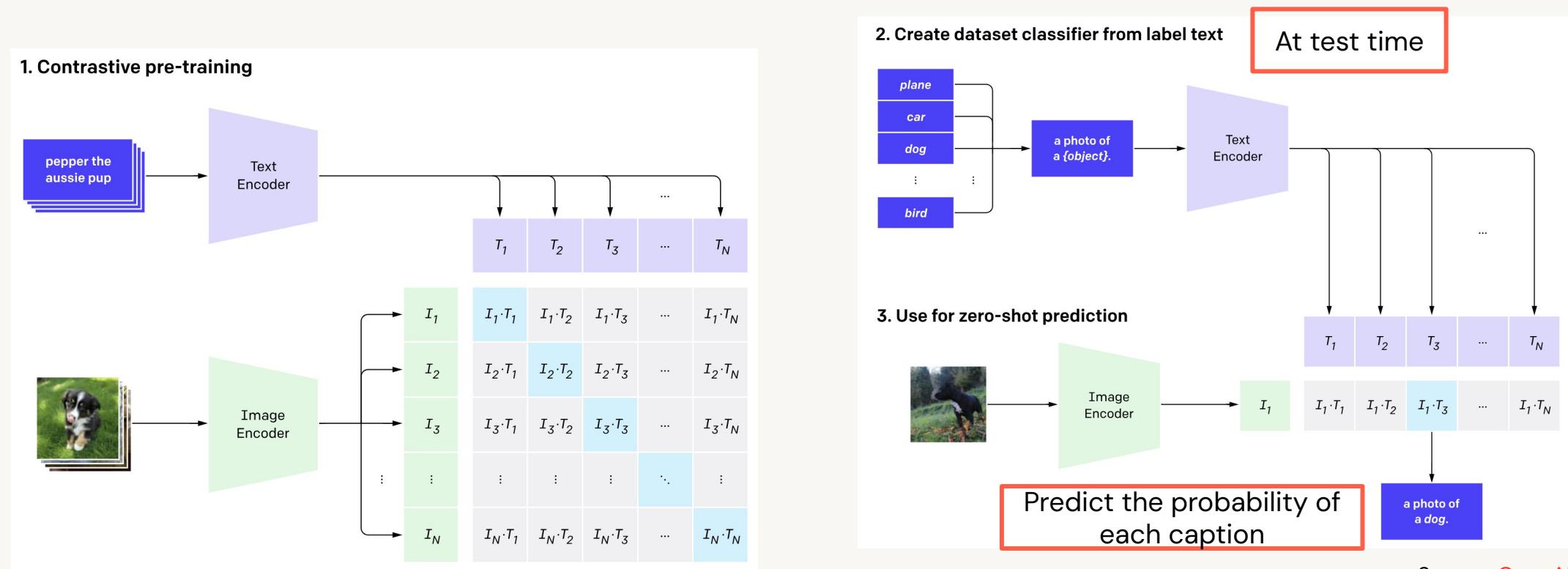
Gathering multi-modal data is harder than just images or text data



Zero-shot: Contrastive Language-Image Pairing

CLIP predicts which image-text pair actually occurs in the training data

- Collects 400M image-text pairs from the internet as training data



Source: [OpenAI](#)



CLIP performs better across settings



Big limitation:

- Inflexible
 - Cannot generate text

CLIP performs much better in non-ImageNet settings

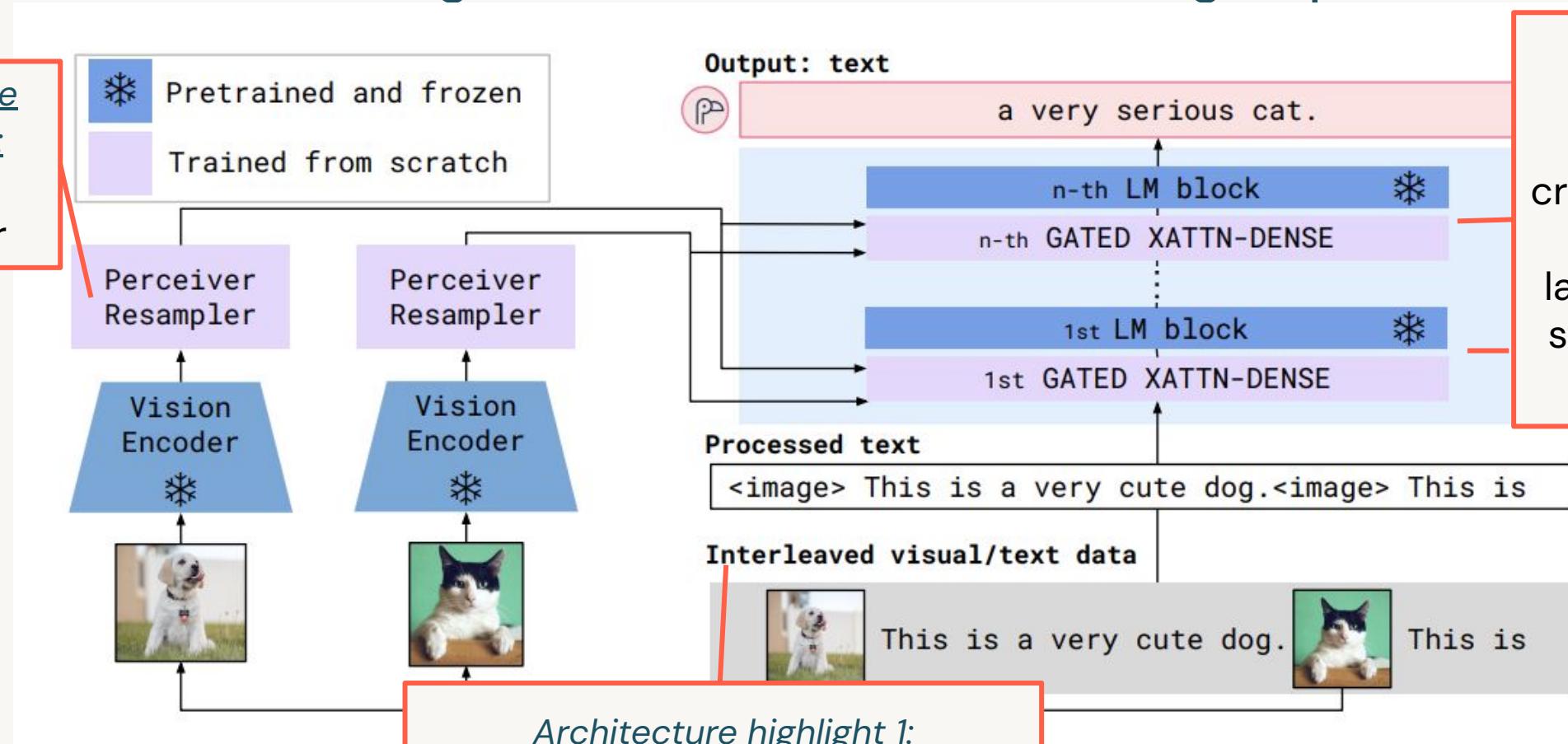
Source: [OpenAI](#)



Few-shot, in-context: Flamingo

Unifies treatment of high-dimensional video and image inputs

Architecture highlight 2:
Perceiver resampler

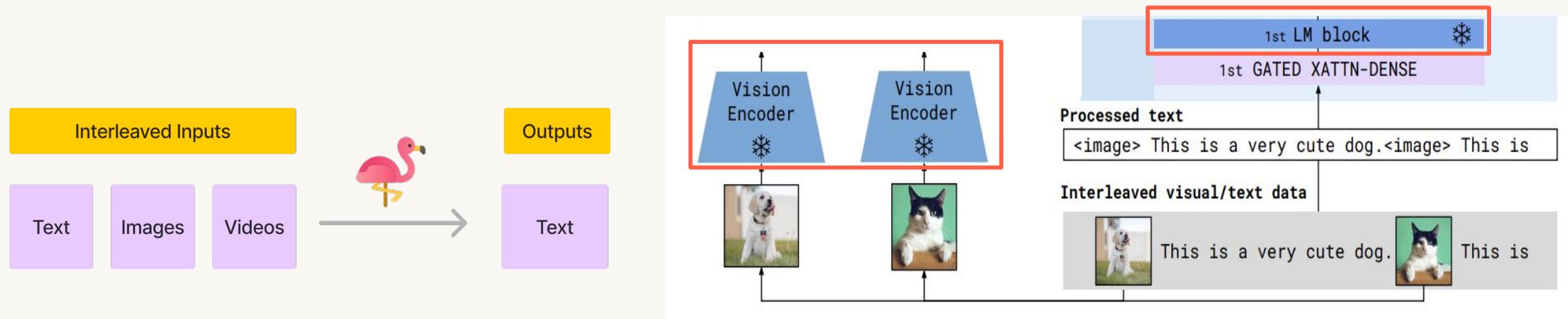


Source: [Alayrac et al 2022](#)



Flamingo bridges vision and language models

Vision encoder similar to CLIP + Chinchilla (Language) accept interleaved inputs



Source: [Alayrac et al 2022](#)

Flamingo models can model the likelihood of text y interleaved with a sequence of images/videos x :

$$p(y|x) = \prod_{l=1}^L p(y_l|y_{<l}, x_{\leq l})$$

p := flamingo model

y_l := l -th language token in input text

$y_{<l}$ - preceding text tokens

$x_{\leq l}$ - preceding image/videos

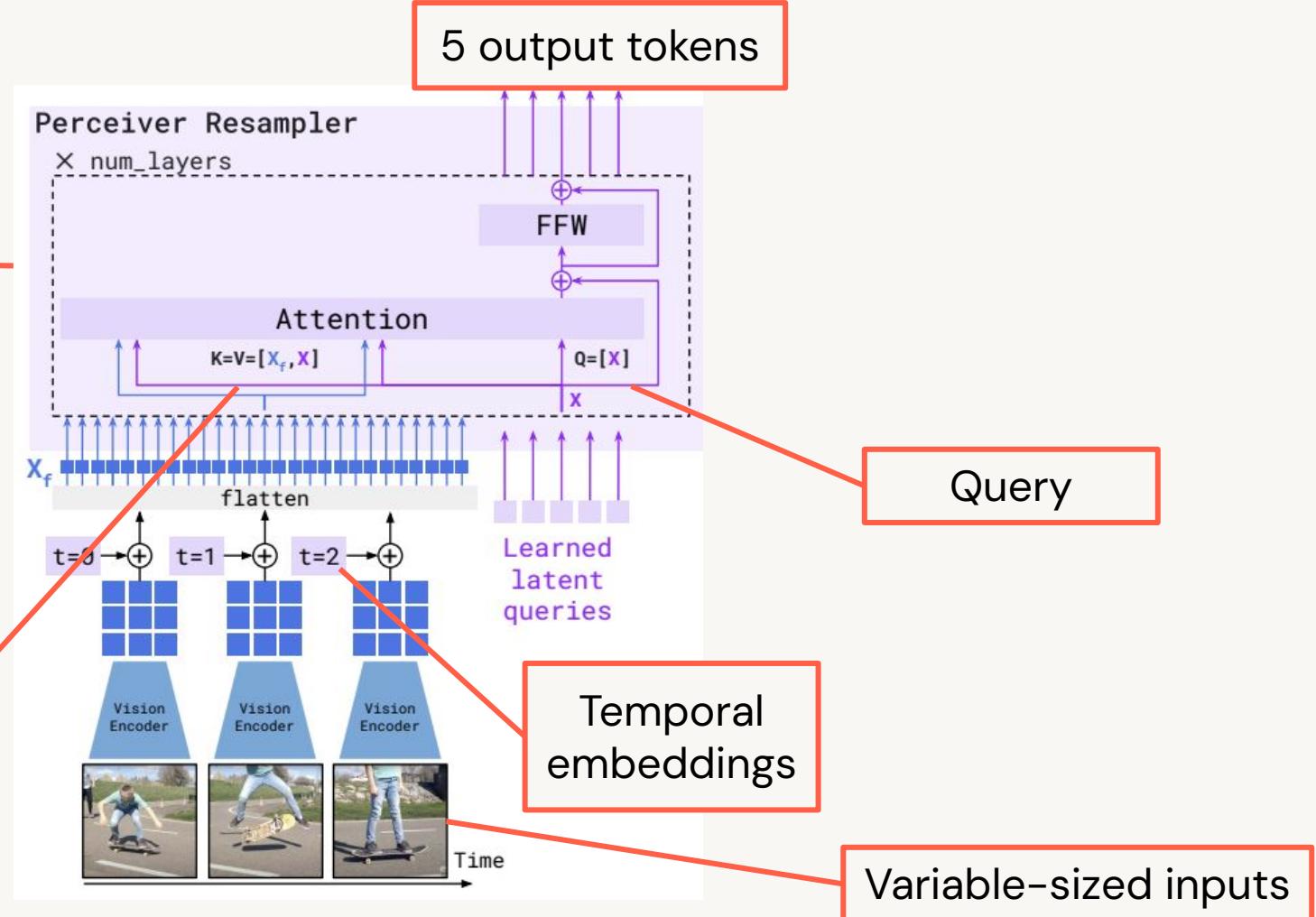
Multimodal likelihood

Image source: [Samuel Albanie](#)

Perceiver resampler outputs fixed-sized tokens

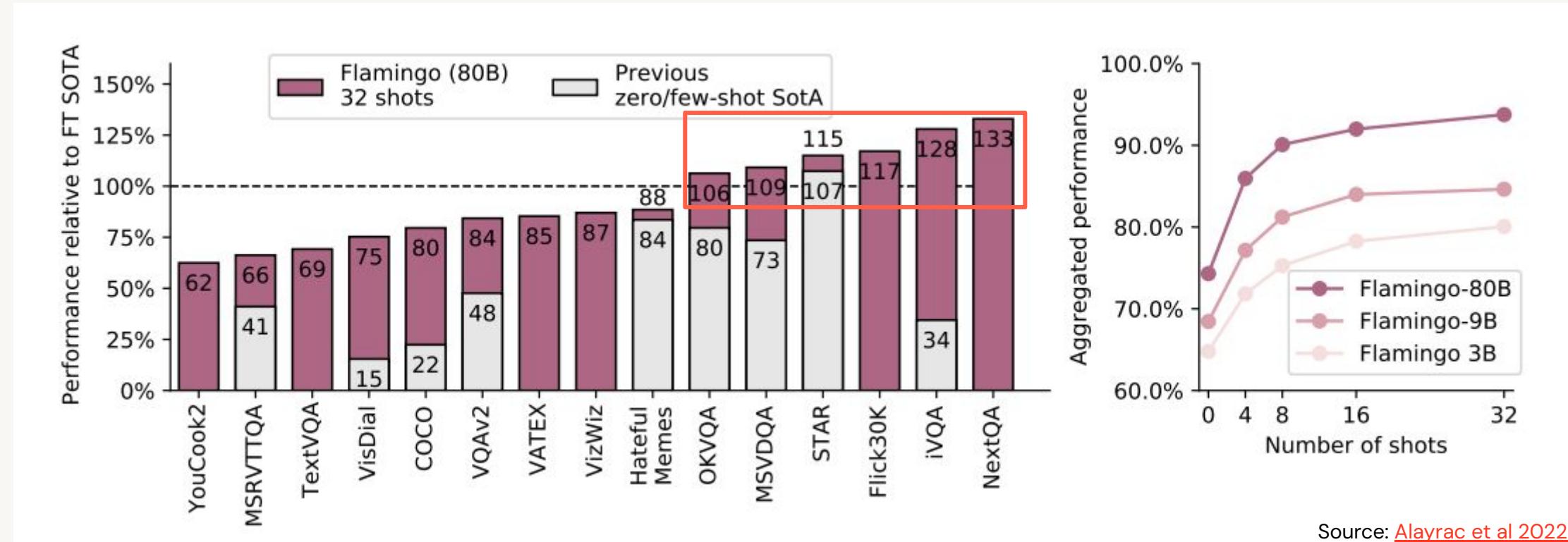
Maps variable-size grid of inputs to a **fixed** number of output tokens

Keys and values = spatio-temporal visual features



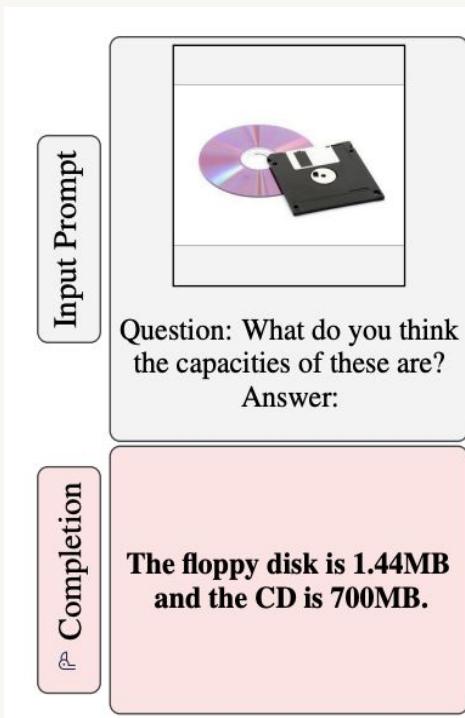
Outperforms 6 out of 16 SOTA fine-tuned models

Curated 3 high-quality datasets: LTIP (Long Text-Image Pairs), VTP (Video-Text Pairs), and MultiModal Massive Web (M3W)

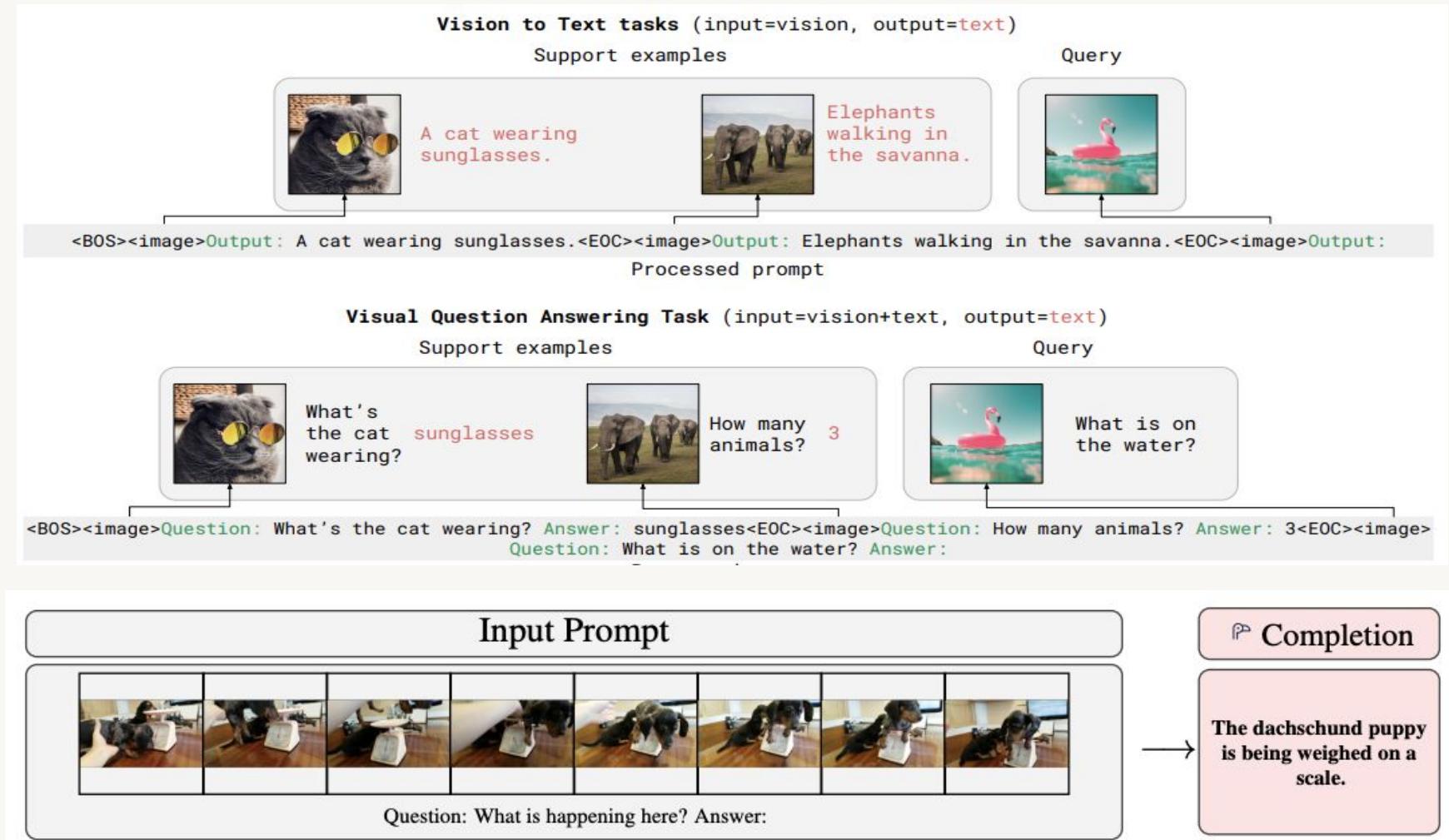


Qualitative inspection on selected samples

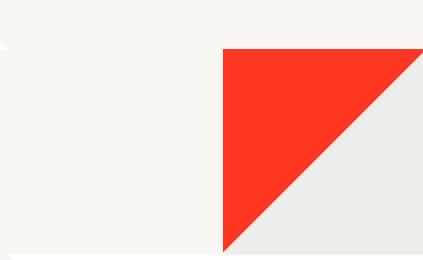
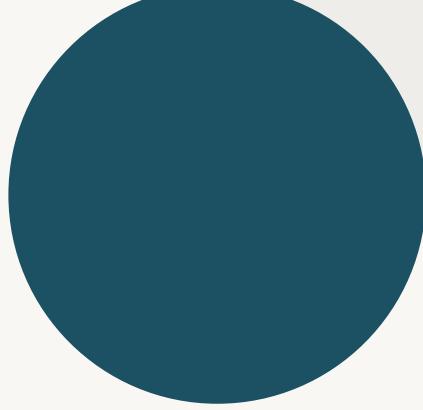
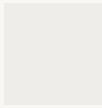
Supported input format: (image, text) or (video, text) + visual query



Source: [Alayrac et al 2022](#)

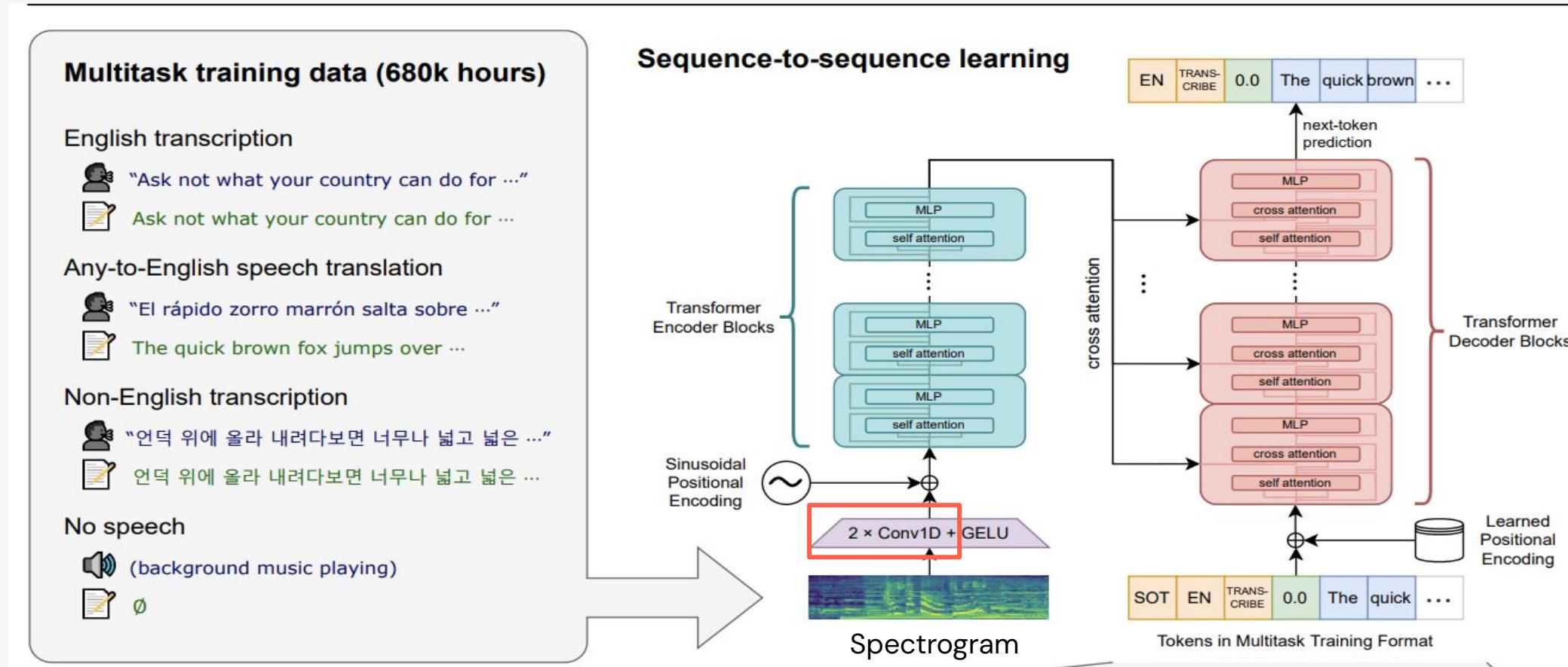


Audio



Zero-shot: OpenAI's Whisper

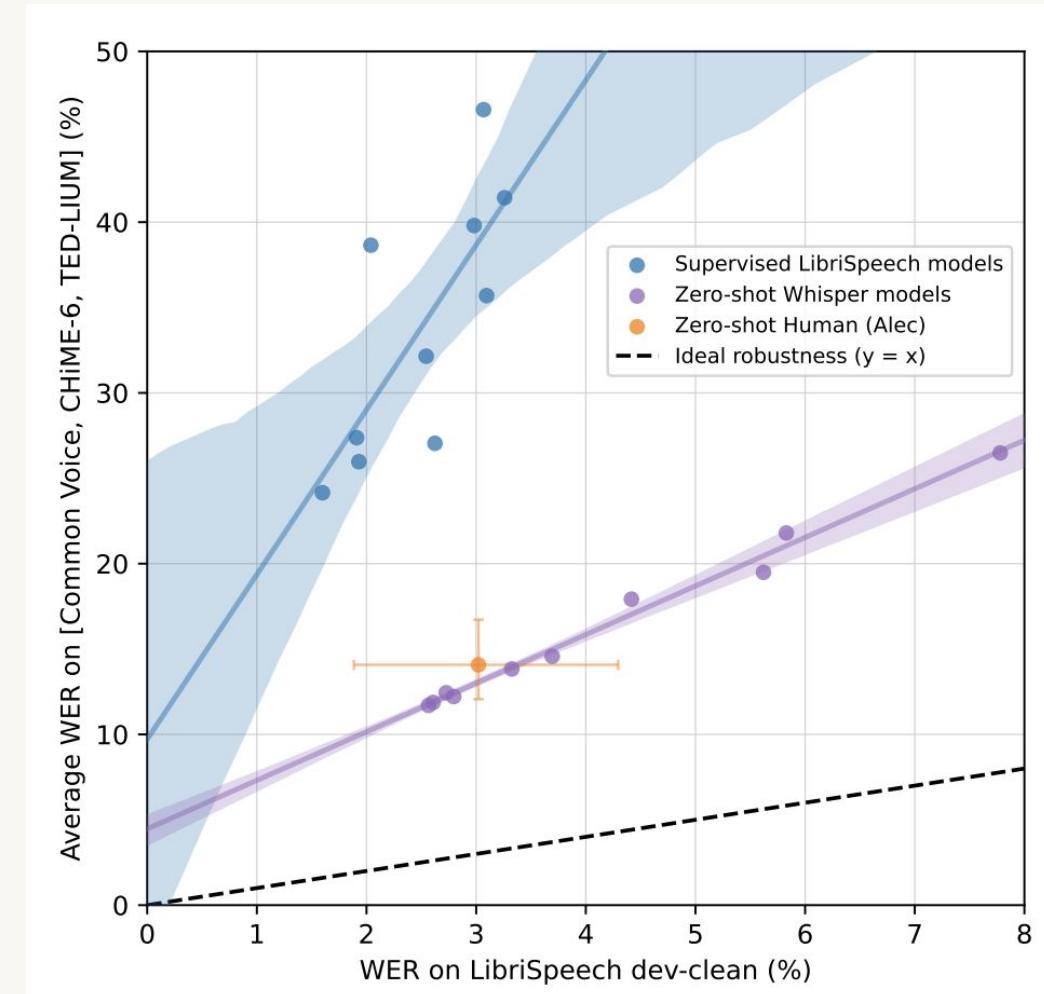
Encoder-decoder transformer: splits input audio into 30-second frames



Whisper matches human robustness

Without fine-tuning on benchmark data

- WER = word error rate
- LibriSpeech
 - 1K hours of read English speech



Challenges

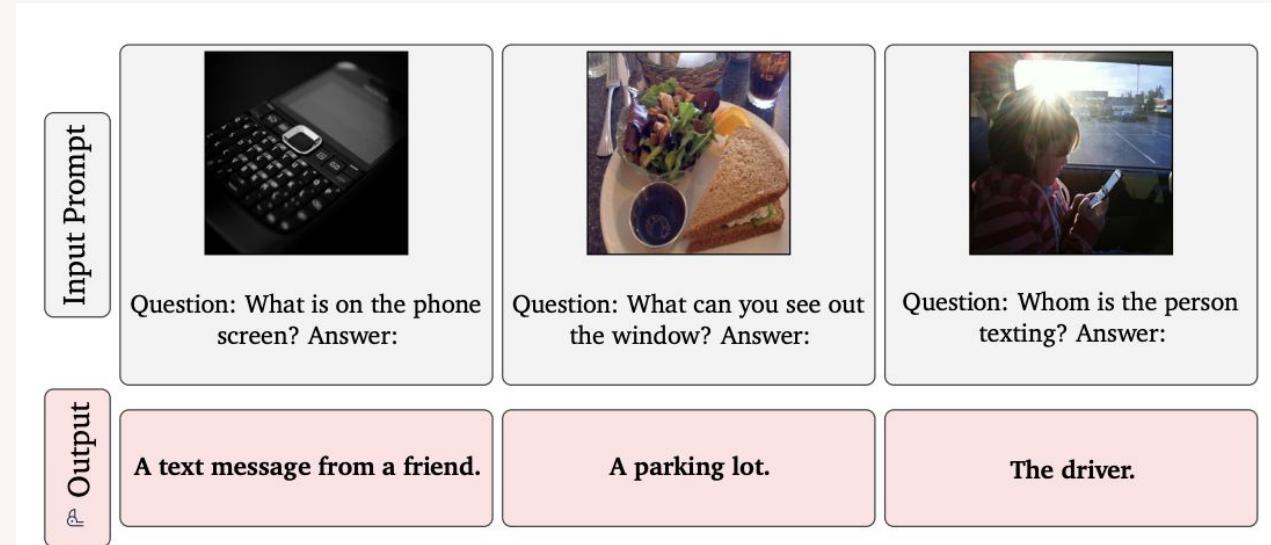
We haven't figured it all out yet



MLMs are not immune from LLM limitations

They inherit LLM risks

- Hallucination
- Prompt sensitivity
 - Context limit
- Inference compute cost
- Bias, toxicity, etc.
- Copyrights issues



Source: [Alayrac et al 2022](#) (Flamingo)

Reddit Wants to Get Paid for Helping to Teach Big A.I. Systems

The internet site has long been a forum for discussion on a huge variety of topics, and companies like Google and OpenAI have been using it in their A.I. projects.

Source: [New York Times, April 2023](#)

LAION-5B: A NEW ERA OF OPEN LARGE-SCALE MULTIMODAL DATASETS

Source: [laion.ai](#)

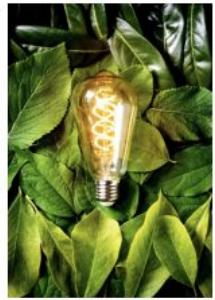


7:58 AM · Jul 14, 2023 · 5.9M Views

Source: [Next Shark, X Post](#)



MLMs can lack common sense (like LLMs)

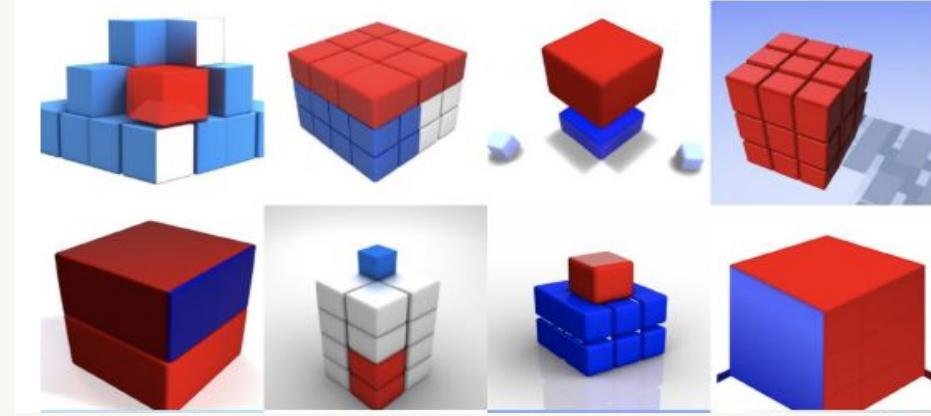


(a) some plants surrounding a lightbulb



(b) a lightbulb surrounding some plants

Source: [Thrush et al 2022](#)
Tested on many models, including CLIP and VisualBERT



Source: [Ramesh et al 2022](#)
Used CLIP and DALLE-2; Prompt: "a red cube on top of a blue cube"

GPT-3 completion

Input: You poured yourself a glass of cranberry, but then absentmindedly, you poured about a teaspoon of grape juice into it. It looks OK. You try sniffing it, but you have a bad cold, so you can't smell anything. You are very thirsty. So you

Completion: drink it. You are now dead.

Source: [Robust AI and NYU](#)

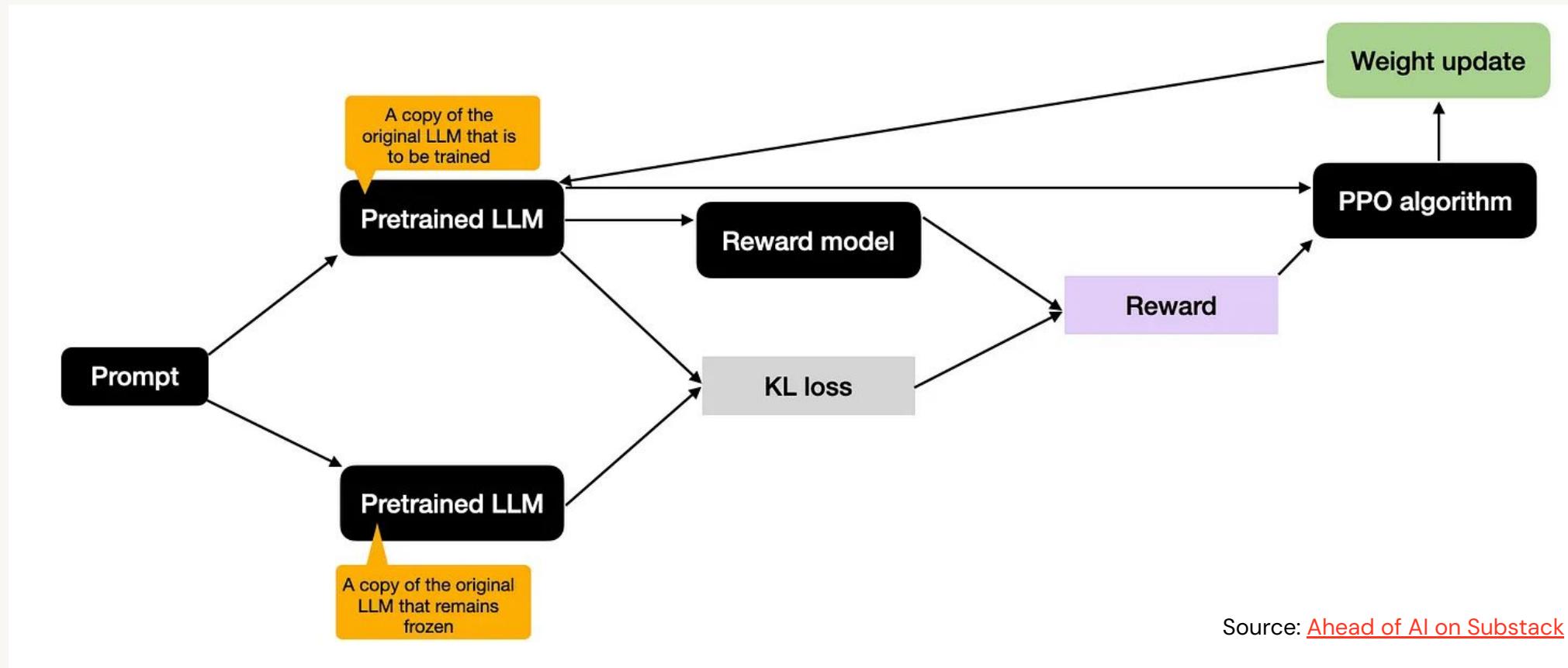


*Attention may not be forever
What may remain or rise?*



Reinforcement learning with human feedback

Human feedback trains the reward model (LM). KL loss ensures minimal divergence from the original LLM. Proximal Policy Optimization (PPO) updates the LLM.



Hyena Hierarchy

Convolutional neural networks are making a comeback?



- Good few-shot learners for languages
- Matches performances of Vision Transformers (ViT)

Table 4.6: Few-shot (3) accuracy (%) on SUPERGLUE tasks for small models.

Model	WSC	WIC	RTE	CB	MULTIRC	RECORD	BOOLQ	COPA	AVERAGE
GPTNeo (Black et al., 2021)	38.5	50.0	53.8	42.9	22.4	61.4	61.0	63.0	49.1
RWKV (Peng, 2021)	32.7	49.4	47.2	37.5	0.0	58.3	55.0	64.0	43.0
Hyena	39.4	50.1	47.6	46.4	26.7	58.1	56.0	70.0	49.3

Table 4.5: Zero-shot accuracy (%) on SUPERGLUE tasks for small models.

Model	WSC	WIC	RTE	CB	MULTIRC	RECoRD	BOOLQ	COPA	AVERAGE
GPTNeo (Black et al., 2021)	27.9	50.0	45.1	41.1	0.0	61.7	62.2	62.0	43.8
RWKV (Peng, 2021)	13.4	52.3	46.9	25.0	0.0	58.5	59.2	66.0	40.2
Hyena	21.2	50.5	46.6	39.3	1.1	59.4	51.8	70.0	41.5

Table 4.7: Image classification top-1 accuracy.

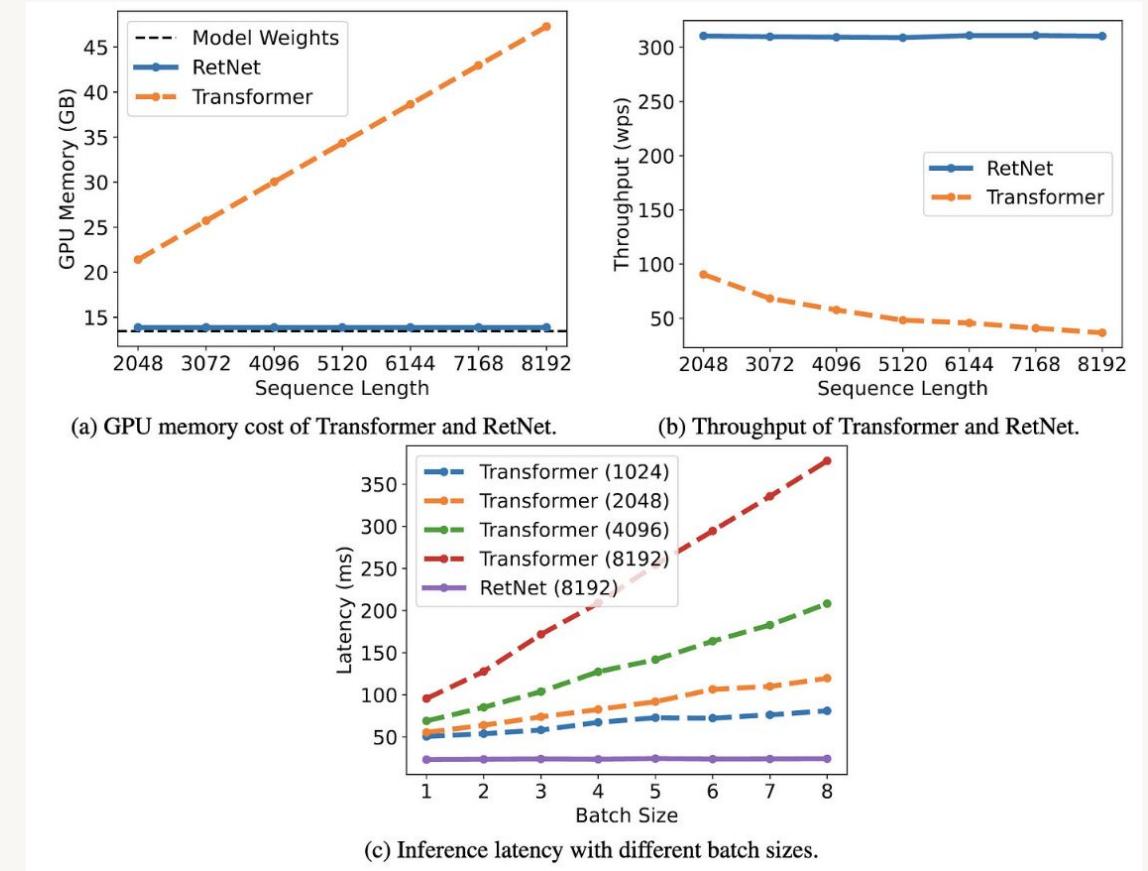
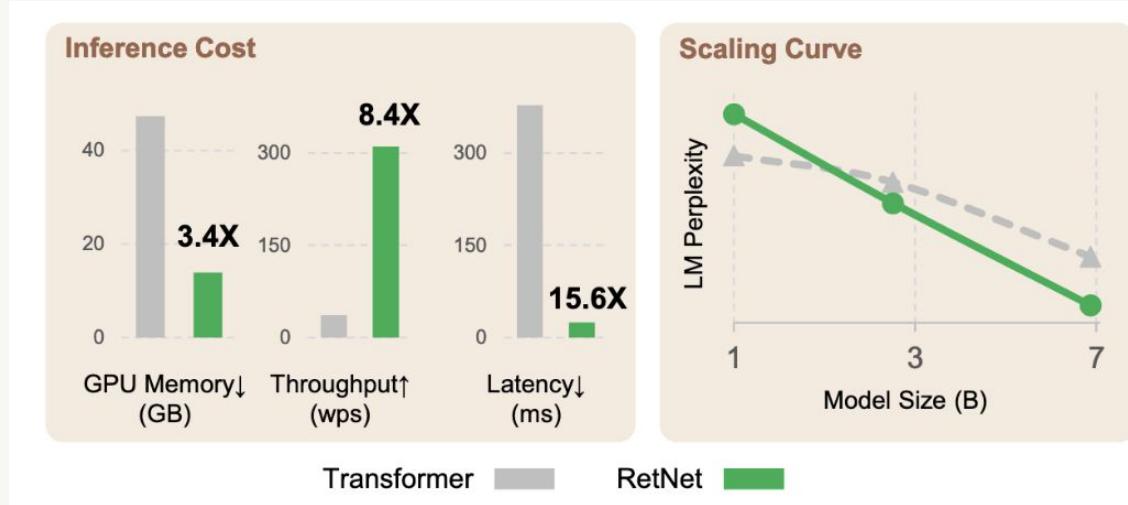
Model	PATCH SIZE	SEQ LEN	DATASET	ACC (%)
ViT (87M)	16x16	196	ImageNet-1k	78.5
Hyena-ViT (88M)	16x16	196	ImageNet-1k	78.5
ViT (87M)	8x8	1024	ImageNet-1k	80.0
Hyena-ViT (88M)	8x8	1024	ImageNet-1k	79.8
S4ND-ISO (268k)	-	-	CIFAR-10	89.9
Hyena-ISO (202k)	-	-	CIFAR-10	91.2

Source: [Poli et al 2023](#)



Retentive Networks

A new attention variant: a retention mechanism to connect recurrence and attention, without compromising performance



Source: [Sun et al 2023](#) (released in July)



Emerging applications

It's a great time to be alive



DreamFusion

Generates 3D objects from text caption



Source: [Poole et al 2022](#)

Make-a-Video

Generates video from text: “Cat watching TV with a remote in hand”



Source: [Singer et al 2022](#)



PaLM-E-bot

Robotics application: “bring me the rice chips from the drawer”



Source: [Driess et al 2023](#)



AlphaCode: generate code

Problem: Minimum # of minutes to makes pizzas of N slices

Human solution

```
t = int(input())
while t != 0:
    n = int(input())
    if n <= 6:
        print("15")
    else:
        print(((n+1)//2)*5)
    t -= 1
```

Model solution

```
import sys
input = sys.stdin.readline

def main():
    t = int(input())
    for _ in range(t):
        N = int(input())

        ans = 10 ** 100
        for a in range(0, 101):
            for b in range(0, 101):
                if N - (a * 6 + b * 8) <= 0:
                    ans = min(ans, a * 15 + b * 20)
                    break
                else:
                    c = (N - (a * 6 + b * 8)) // 10

                    if (N - (a * 6 + b * 8)) % 10 != 0:
                        c += 1

                    ans = min(ans, a * 15 + b * 20 + c * 25)

    print(ans)
```

```
if __name__ == '__main__':
    main()
```

Source: [Li et al 2022](#)



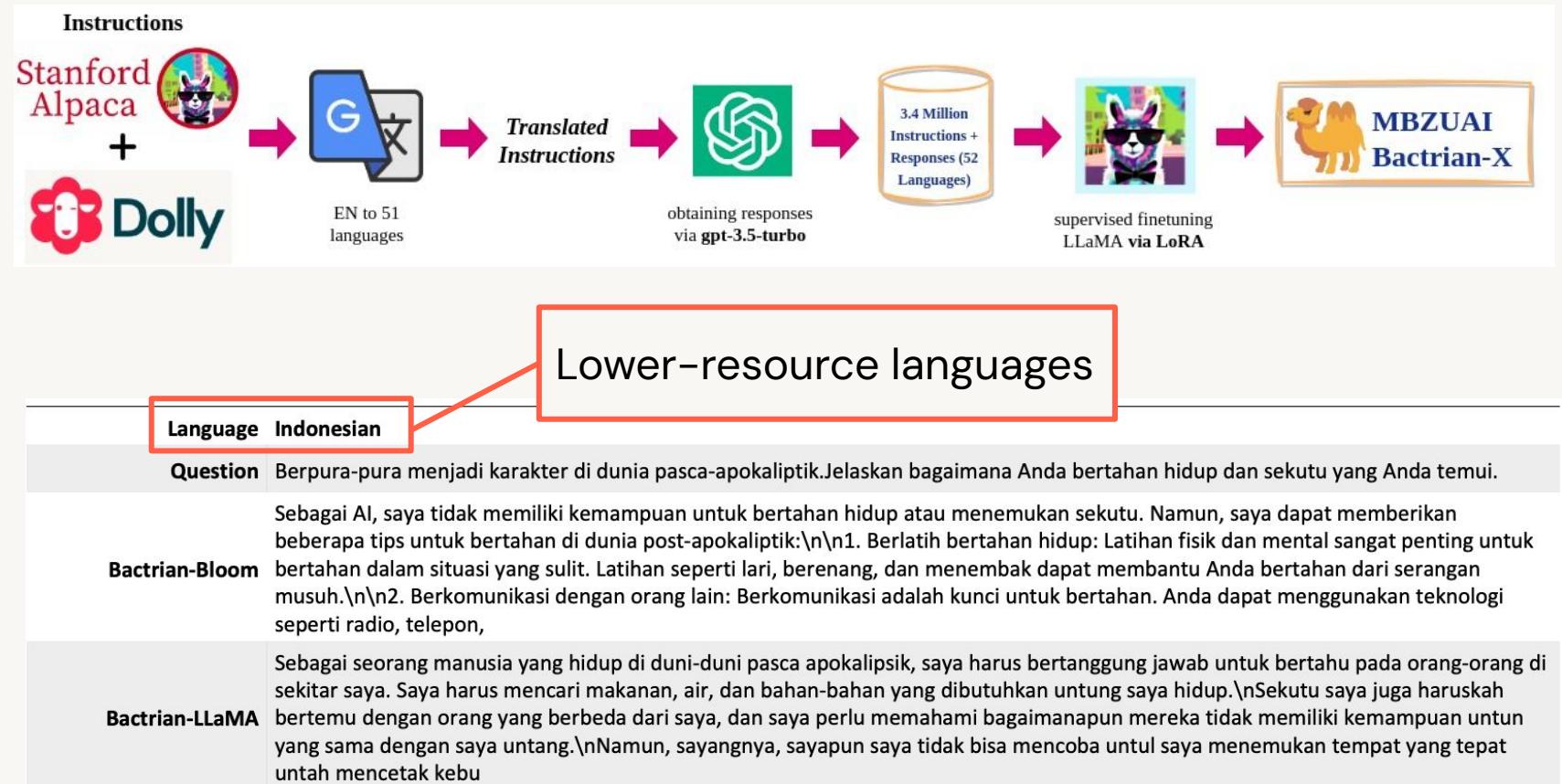
Multi-lingual models: Bactrian-X

An instruction-following model

Prompt:

"It's an
apocalypse.

Describe how you
survive and make
allies"



Source: Li et al 2023 ([GitHub](#) and [Paper](#)); released in May



Gato: a generalist AI agent

Can play Atari, caption images, chat, stack blocks with a real robot arm



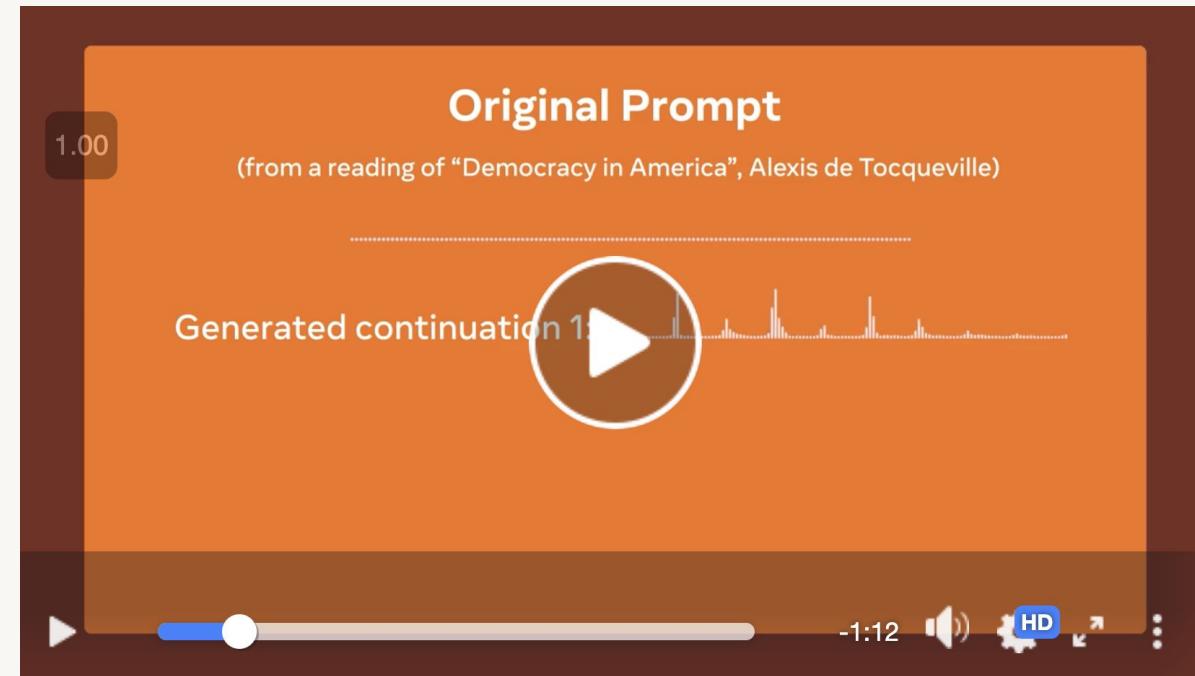
Source: [Reed et al 2022](#)



Textless NLP

Generate speech from raw audio without text transcription

Helps with low-resource languages

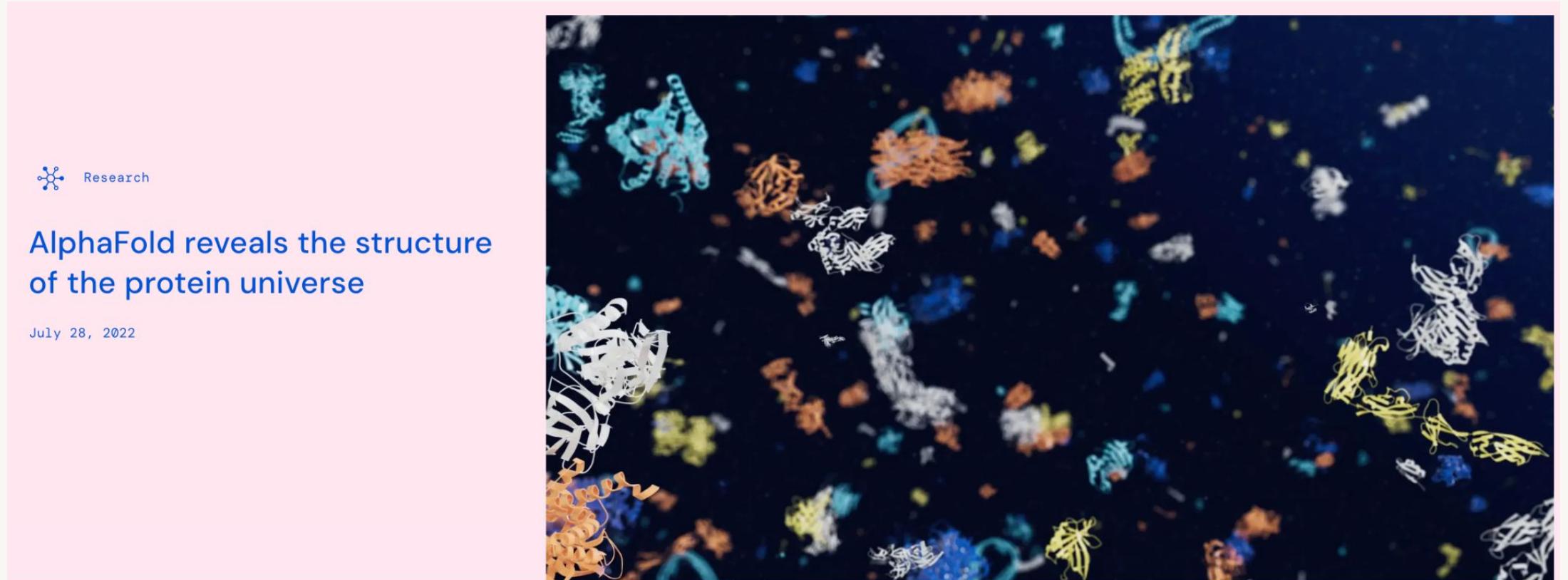


[Click here for demo on Meta AI](#)



AlphaFold

Uses attention to predict protein folding



AlphaFold reveals the structure
of the protein universe

July 28, 2022

Source: [Google DeepMind](#) and [Timeline of Breakthrough](#)



Module Summary

Multi-modal LLMs – What have we learned?

- MLLMs are gaining traction
- Transformers are general sequence-processing architectures that can accept non-text sequences
- MLLMs inherit limitations from LLMs
- Transformers may not be the last architecture standing
- More exciting and unimaginable MLLM applications are on the horizon



Time for some code!





databricks

