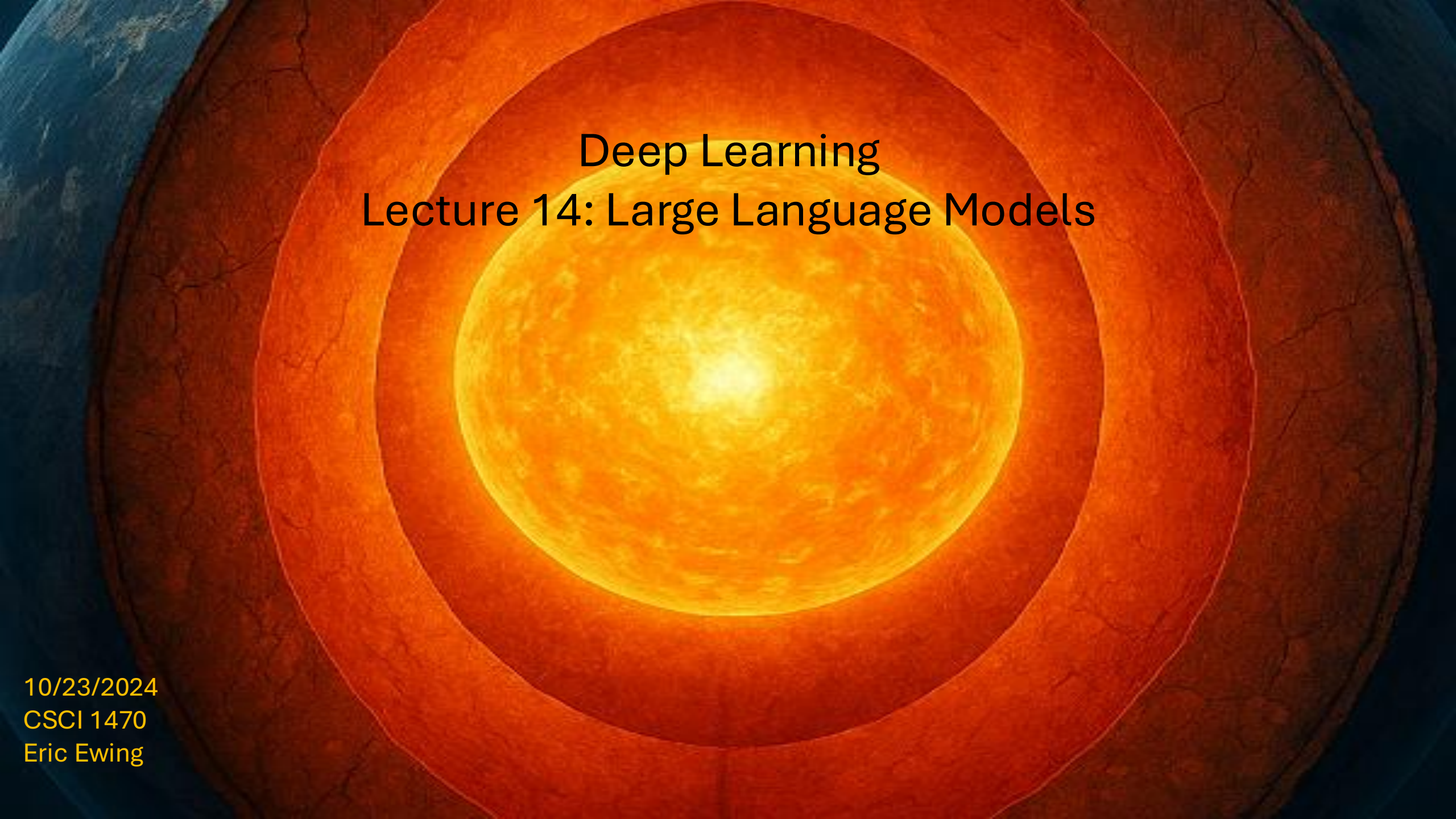


Deep Learning
Lecture 14: Large Language Models

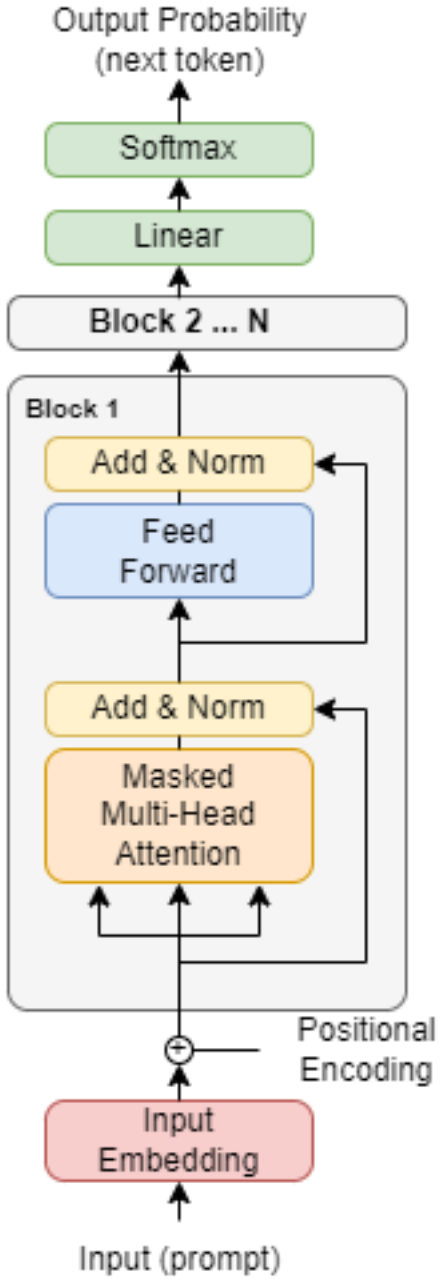
10/23/2024
CSCI 1470
Eric Ewing

Decoder Only Transformer

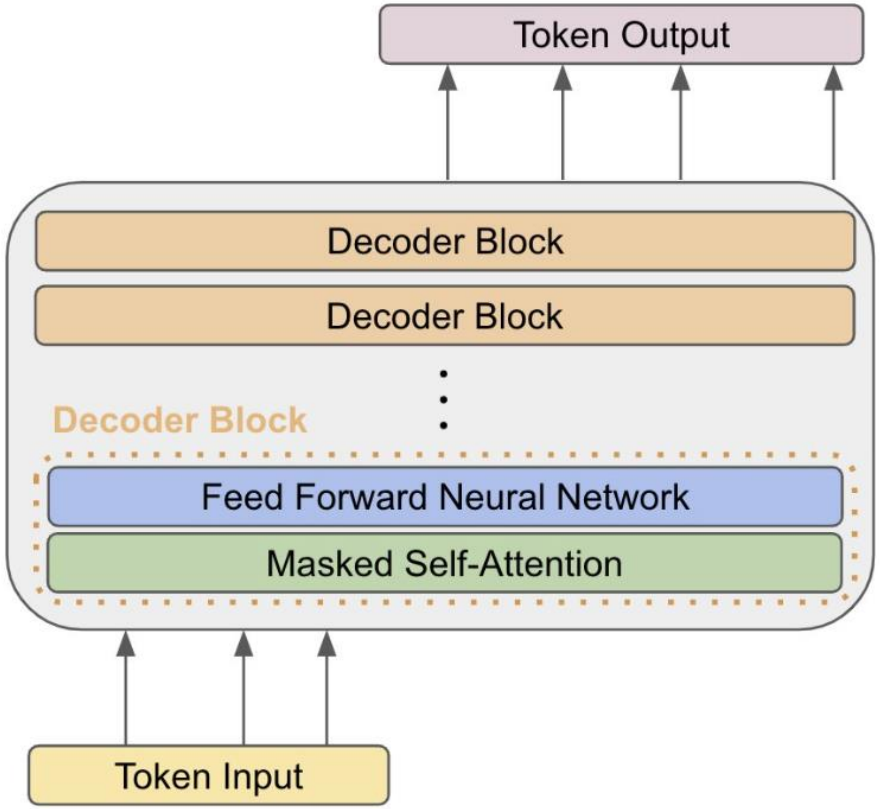
Language modeling does not have a separate input-output sequence, they are one and the same (unlike machine translation)

We don’t need a separate encoder and decoder in the transformer

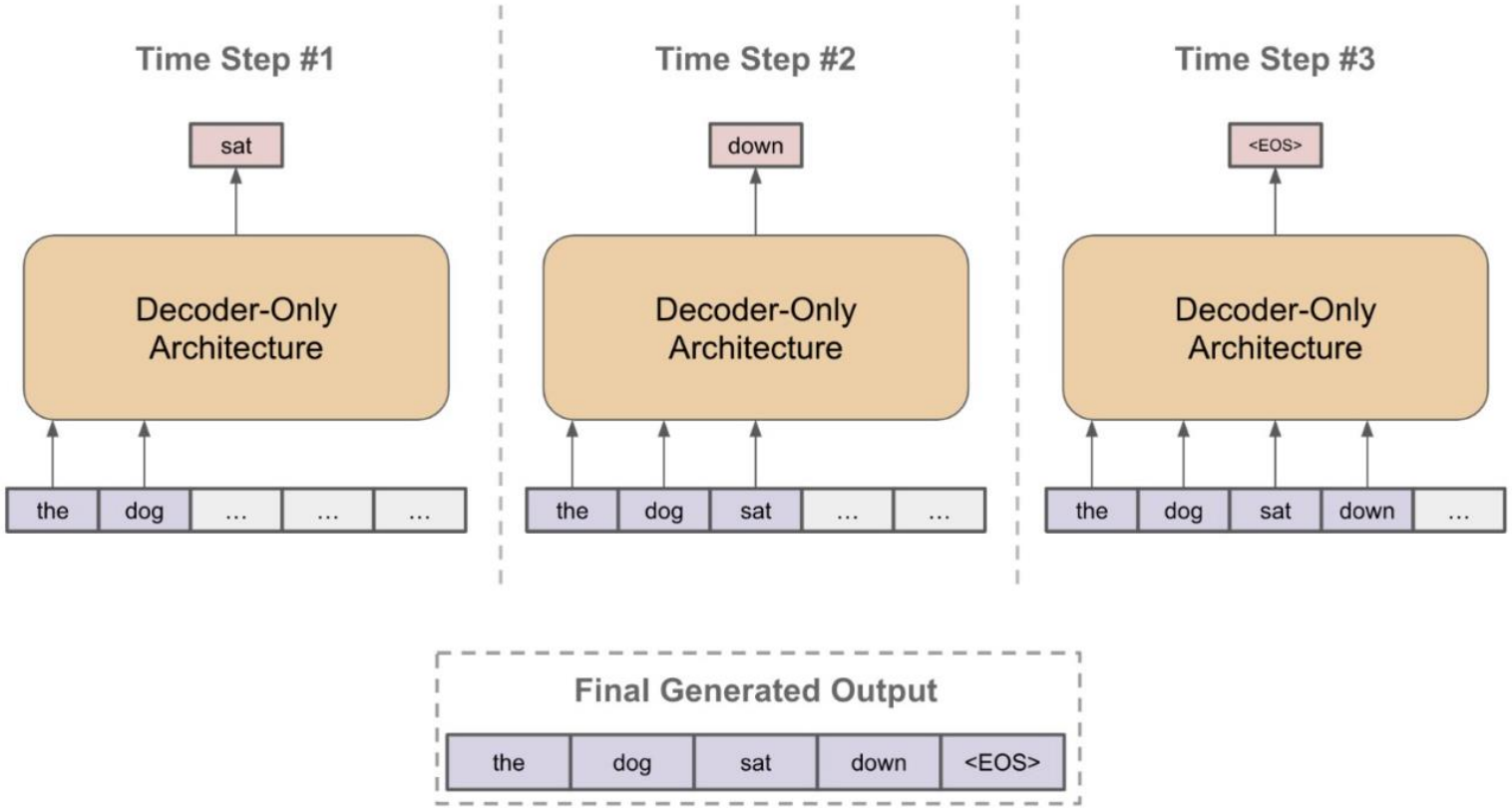
A decoder-only transformer is just the decoder of a transformer and is the primary building block of LLMs



Decoder-Only Architecture



Generating Autoregressive Output



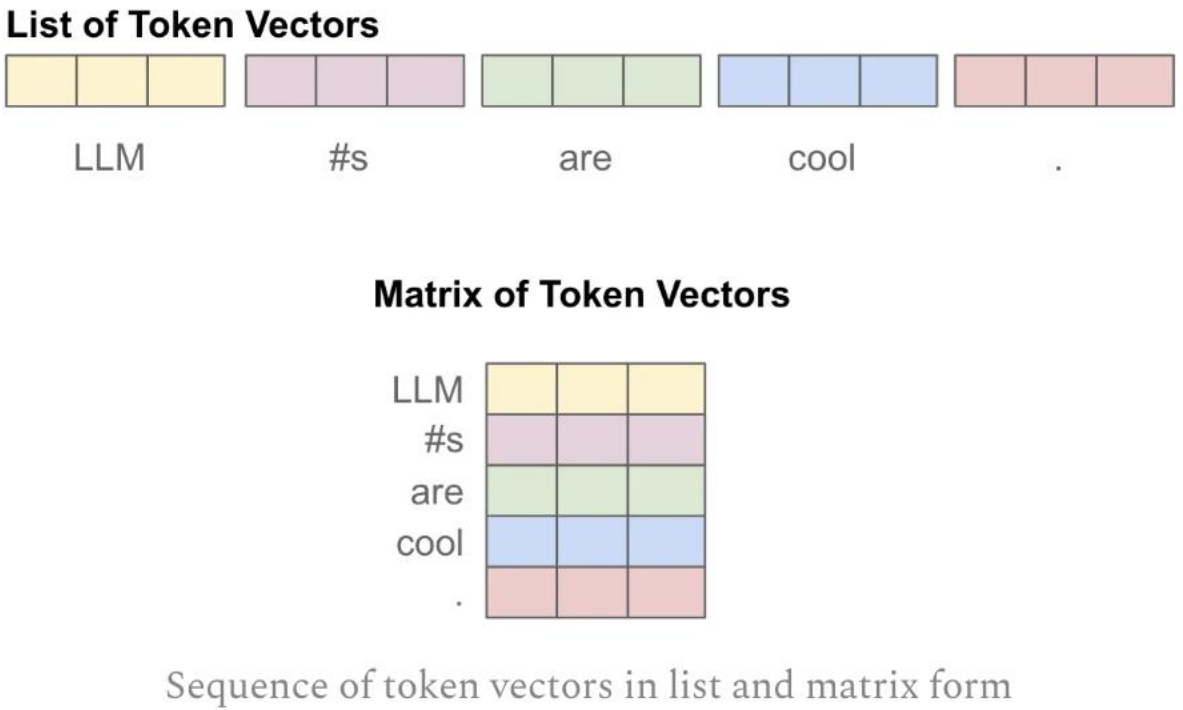
#tokens in input is the *context length*

Source: Cameron Wolfe, https://cameronrwolfe.substack.com/p/decoder-only-transformers-the-workhorse

Self-Attention

Token vectors are either:

1. Token Embedding + Positional Encoding
2. Output of previous decoder block

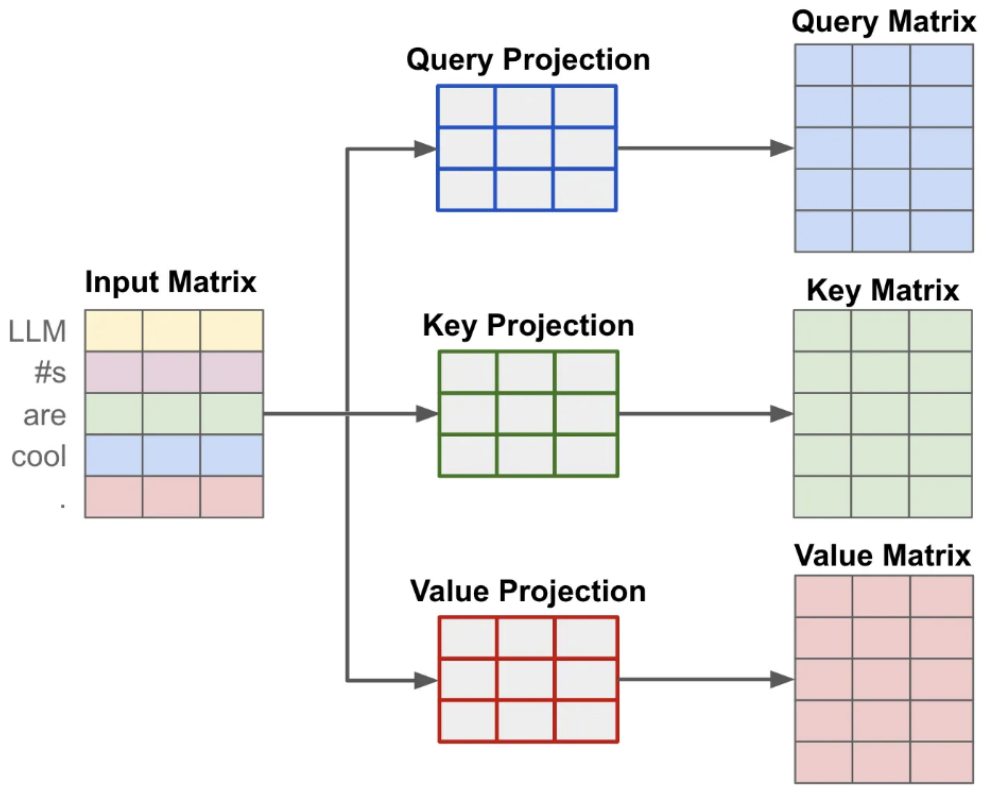


Sequence of token vectors in list and matrix form

Source: Cameron Wolfe, https://cameronrwolfe.substack.com/p/decoder-only-transformers-the-workhorse

Self-Attention

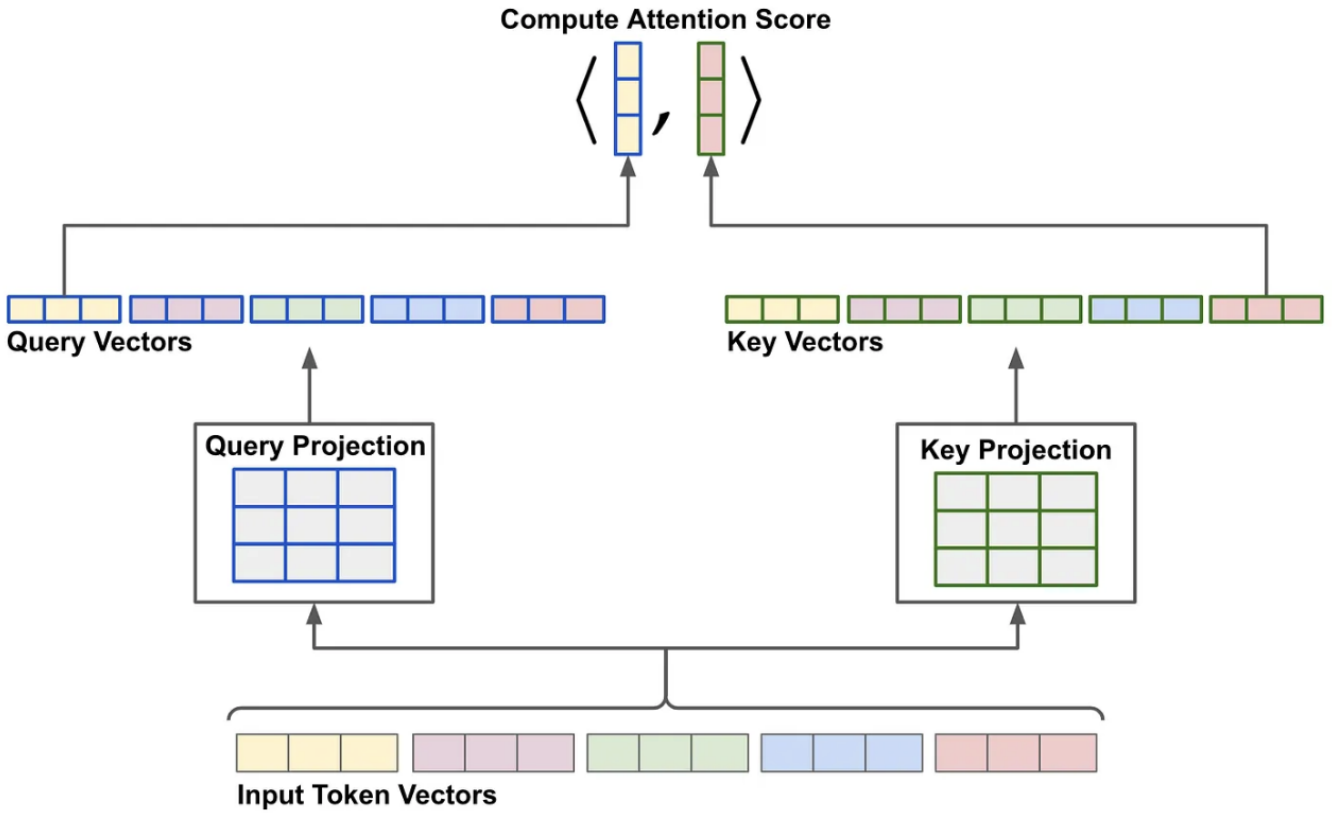
Separate Q, K, V projection matrices (linear layers)



Creating the query, key, and token vectors

Source: Cameron Wolfe, https://cameronrwolfe.substack.com/p/decoder-only-transformers-the-workhorse

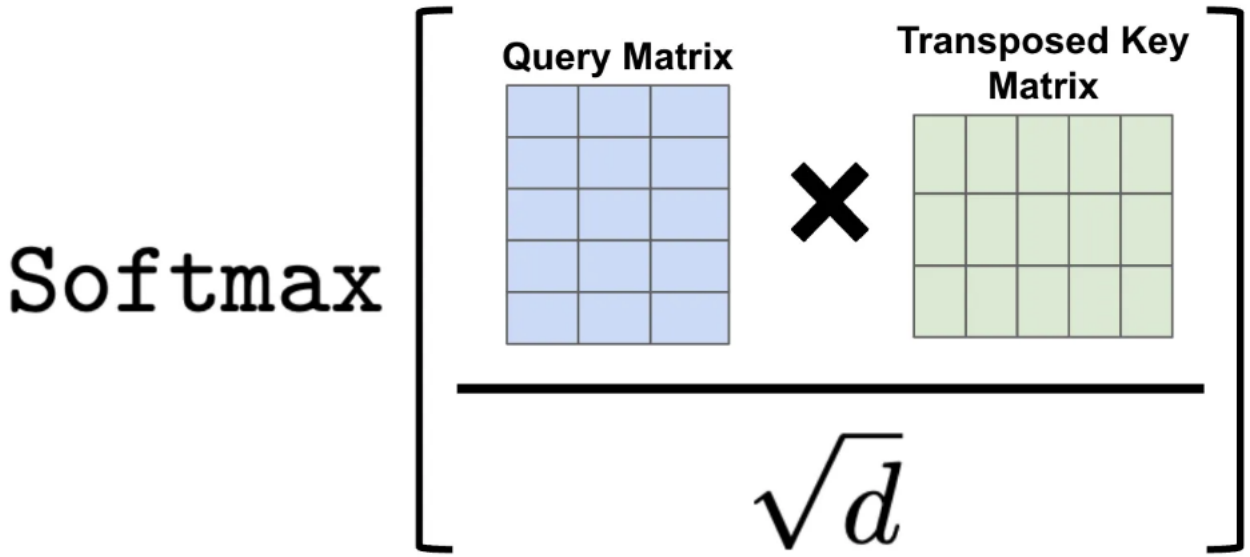
Self-Attention



Computing attention scores from query and key vectors

Source: Cameron Wolfe, https://cameronrwolfe.substack.com/p/decoder-only-transformers-the-workhorse

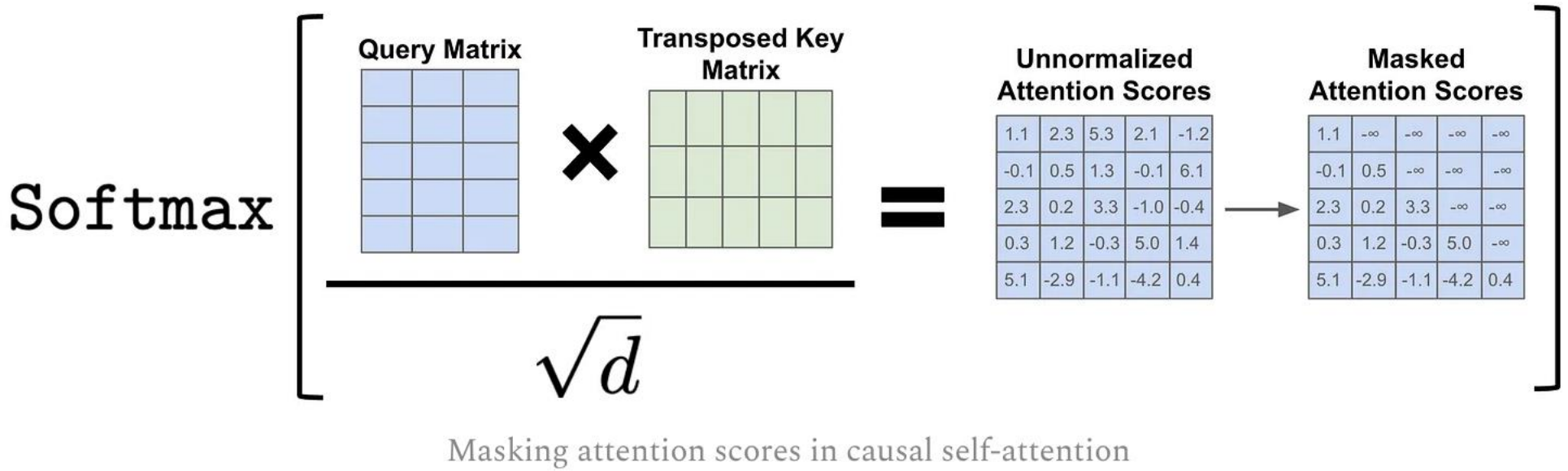
Self-Attention



Computing the attention matrix

Source: Cameron Wolfe, https://cameronrwolfe.substack.com/p/decoder-only-transformers-the-workhorse

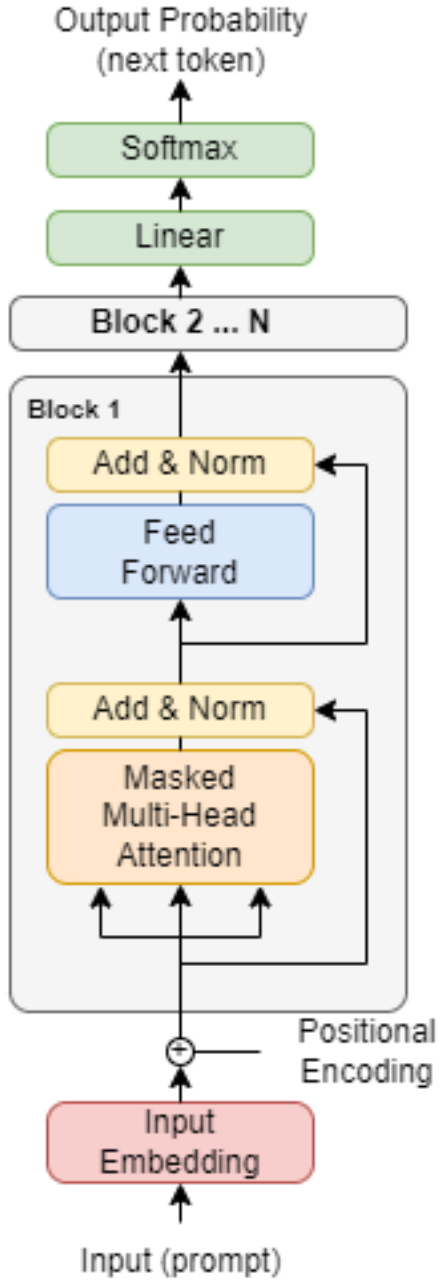
Causal-Masked Self-Attention



Masking attention scores in causal self-attention

Source: Cameron Wolfe, https://cameronrwolfe.substack.com/p/decoder-only-transformers-the-workhorse

Decoder Only Transformer



Full Pipeline

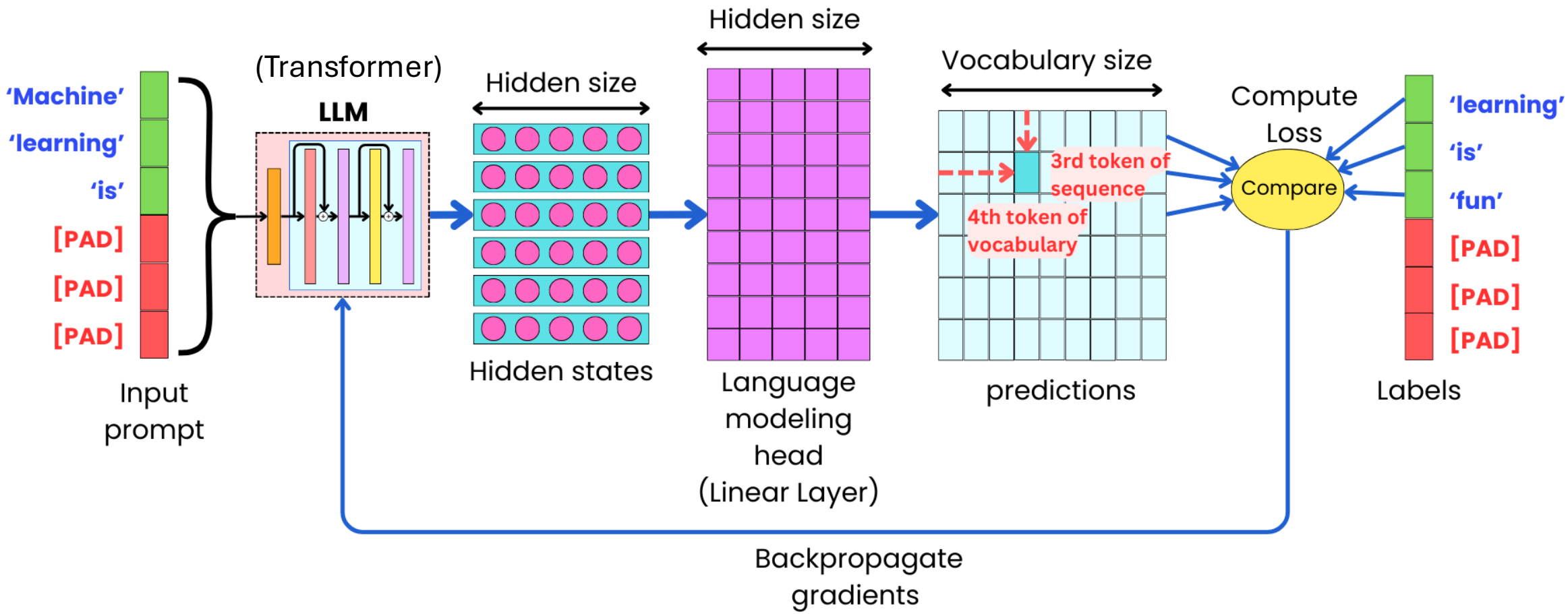
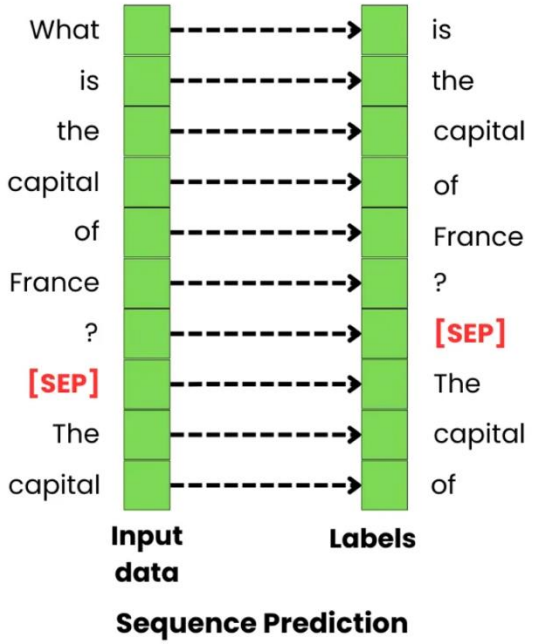


Image Source: https://newsletter.theaiedge.io/p/fine-tuning-llms-from-a-to-z

LLM Training

During training:

- Feed in sequence of *n* tokens
- Output is also sequence of *n* tokens
- Correct labels are input sequence shifted by 1
- Use (sparse) Cross Entropy



One forward pass of a transformer produces many outputs

Image Source: https://newsletter.theaiedge.io/p/fine-tuning-llms-from-a-to-z

Language Modeling Assignment

Pipeline Overview:

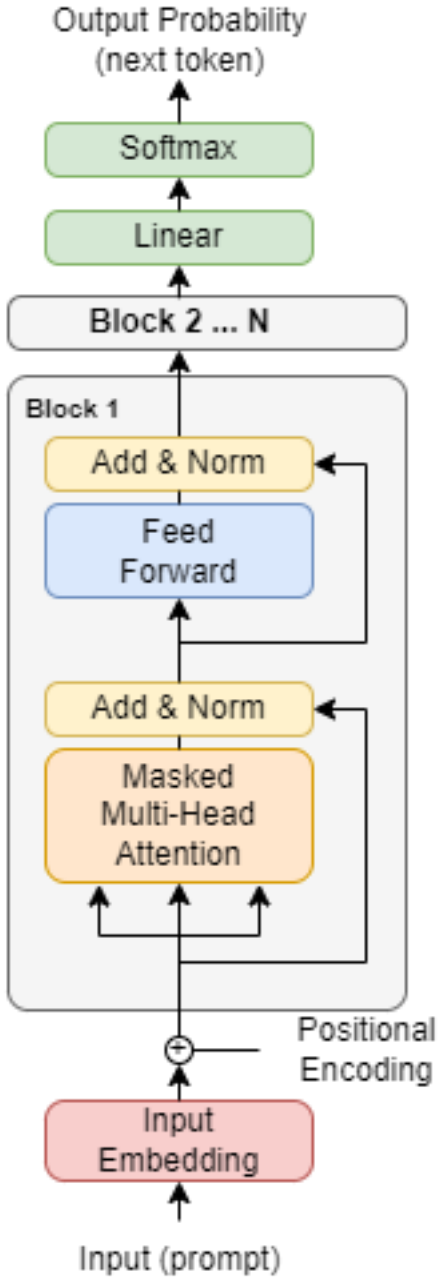
1. Tokenize data and split data into sequences
2. Implement RNN and LSTM
3. Implement Sampling techniques for token generation
4. Implement Decoder Only Transformer
5. Implement Training Loop

Sampling Techniques

Our outputs will be a probability distribution over tokens.

How can select the next token for generation?

1. Choose the token with highest probability
2. Sample token from probability distribution



Temperature Sampling

Add “temperature” parameter to softmax to control how soft/hard the softmax is

$$p_i = \frac{e^{y_i/T}}{\sum_j^n e^{y_j/T}}$$

Top-K Sampling

Select K tokens with highest probabilities, throw out the rest.

Renormalize probabilities so that they sum to 1 on for the K tokens.

Sample from distribution

Top-K Sampling

Select K tokens with highest probabilities, throw out the rest.

Renormalize probabilities so that they sum to 1 on for the K tokens.

Sample from distribution

Why might this work better than sampling from the original distribution

Top-P Sampling (Nucleus Sampling)

Given P, a value in (0, 1], select the smallest set of tokens such that their cumulative probability is greater than p.

That is, sort tokens in decreasing order by probability, iterate through the tokens keeping track of the total probability until it is greater than p.

Top-P Sampling (Nucleus Sampling)

Given P, a value in (0, 1], select the smallest set of tokens such that their cumulative probability is greater than p.

That is, sort tokens in decreasing order by probability, iterate through the tokens keeping track of the total probability until it is greater than p.

How is this different than Top-K? When might it be better? When might it be worse?

The Training Loop

You've probably written this exact loop many times…

```
while ((batch_idx+1)*model.batch_size) < train_len:
    imgs, anss = get_next_batch(batch_idx, train_inputs, train_labels, batch_size=model.batch_s
    with tf.GradientTape() as tape:
        predictions = model(imgs, is_testing=False)
        loss = model.loss_fn(predictions, anss)
        model.loss_list.append(loss)
    gradients = tape.gradient(loss, model.trainable_variables)
    optimizer.apply_gradients(zip(gradients, model.trainable_variables))
    sum_acc += model.accuracy(model(imgs, is_testing=False), anss).numpy()
    batch_idx += 1
```

But we there are plenty of problems that we'll face as we start to scale up:

1. How do we track performance of our model? How can we tell if it's working well enough to keep running it?
2. What if our computer crashes after 30 minutes of training? It would be a shame to lose all that work…

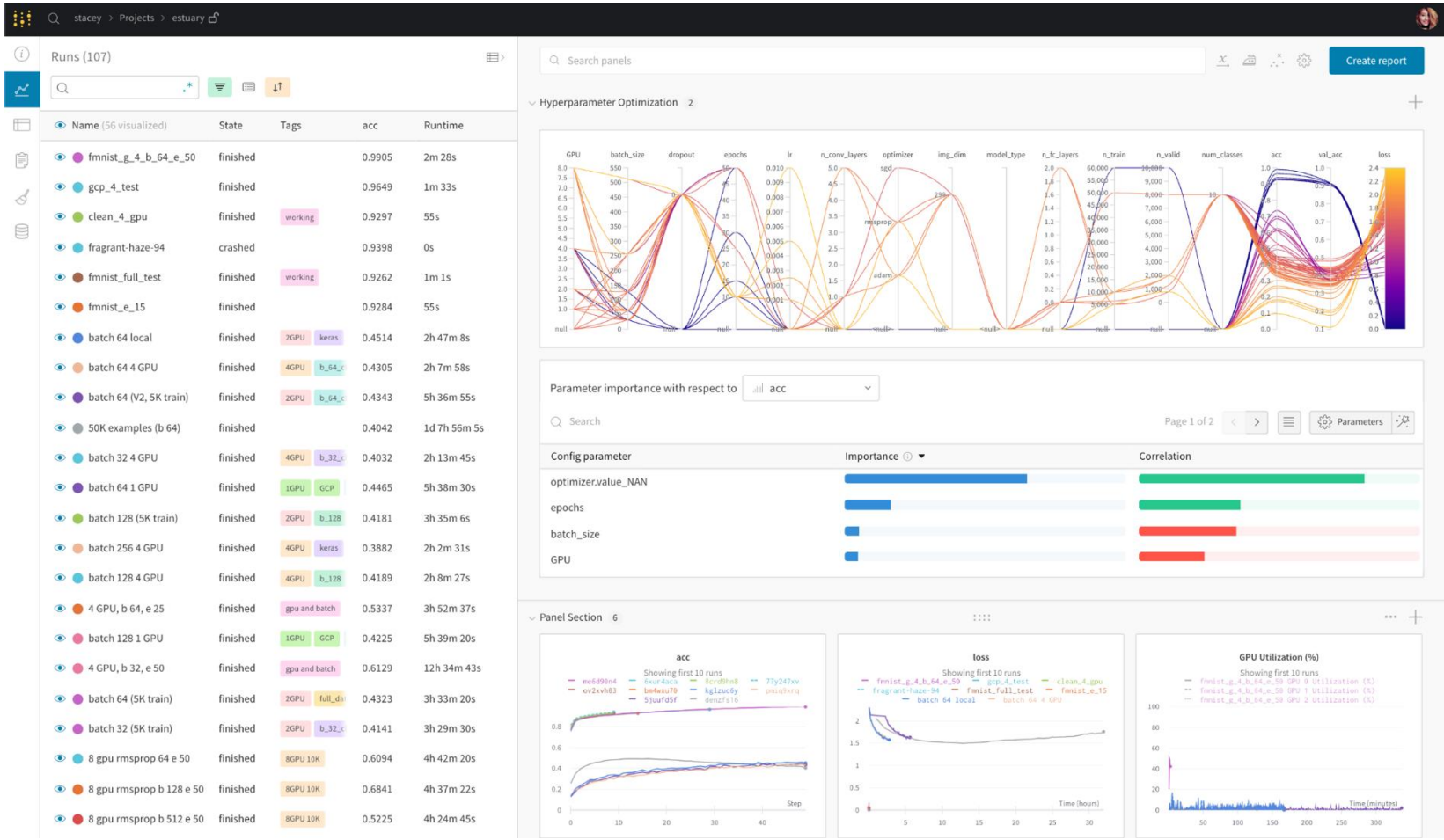
Experiment Tracking

There are a number of tools made for tracking experiments and model performance (other than print statements)

Tensorboard: Comes with tensorflow, publishes data and graphs to a port, can open with a local browser or through ssh if working remotely.

Weights and Biases: Online platform for visualizing performance, data, and other information.

(and many others)



Checkpointing

tf.train.CheckpointManager

View source on GitHub

Manages multiple checkpoints by keeping some and deleting unneeded ones.

A checkpoint saves model weights at a specific point of training (i.e., every epoch, every 10 minutes, etc.)

Tensorflow provides a CheckPoint Manager that can handle a number of useful cases:

1. Save a checkpoint if it performs better on a given metric (i.e., validation loss)
2. Save a checkpoint every X amount of time
3. Overwrite other checkpoints
4. Load from checkpoints

LLM Hyperparameters

	OLMo-7B	LLaMA2-7B	OpenLM-7B	Falcon-7B	PaLM-8B
Dimension	4096	4096	4096	4544	4096
Num heads	32	32	32	71	16
Num layers	32	32	32	32	32
MLP ratio	~8/3	~8/3	~8/3	4	4
Layer norm type	non-parametric	RMSNorm	parametric	parametric	parametric
Positional embeddings	RoPE	RoPE	RoPE	RoPE	RoPE
Attention variant	full	GQA	full	MQA	MQA
Biases	none	none	in LN only	in LN only	none
Block type	sequential	sequential	sequential	parallel	parallel
Activation	SwiGLU	SwiGLU	SwiGLU	GeLU	SwiGLU
Sequence length	2048	4096	2048	2048	2048
Batch size (instances)	2160	1024	2048	2304	512
Batch size (tokens)	~4M	~4M	~4M	~4M	~1M
Weight tying	no	no	no	no	yes

Table 2: LM architecture comparison at the 7–8B scale. In the "layer norm type" row, "parametric" and "non-parametric" refer to the usual layer norm implementation with and without adaptive gain and bias, respectively.

Large Language Model Scaling “Laws”

The bigger the better

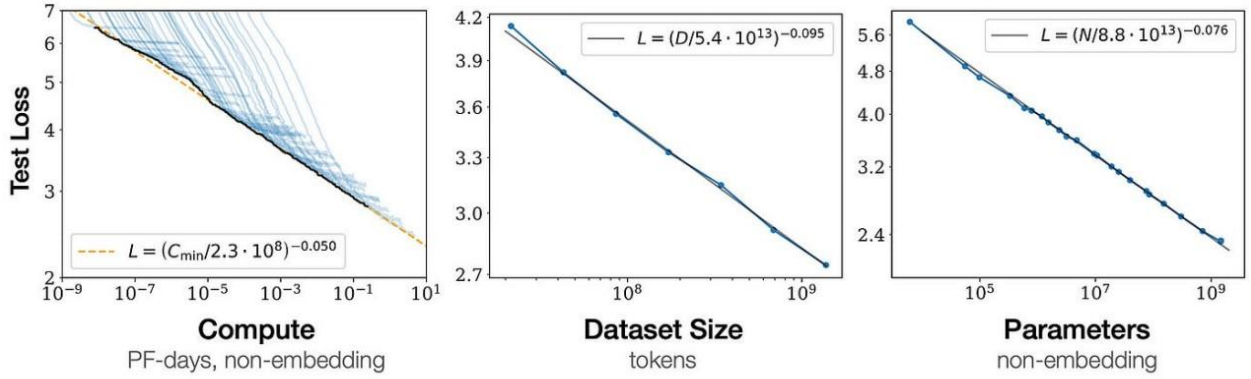
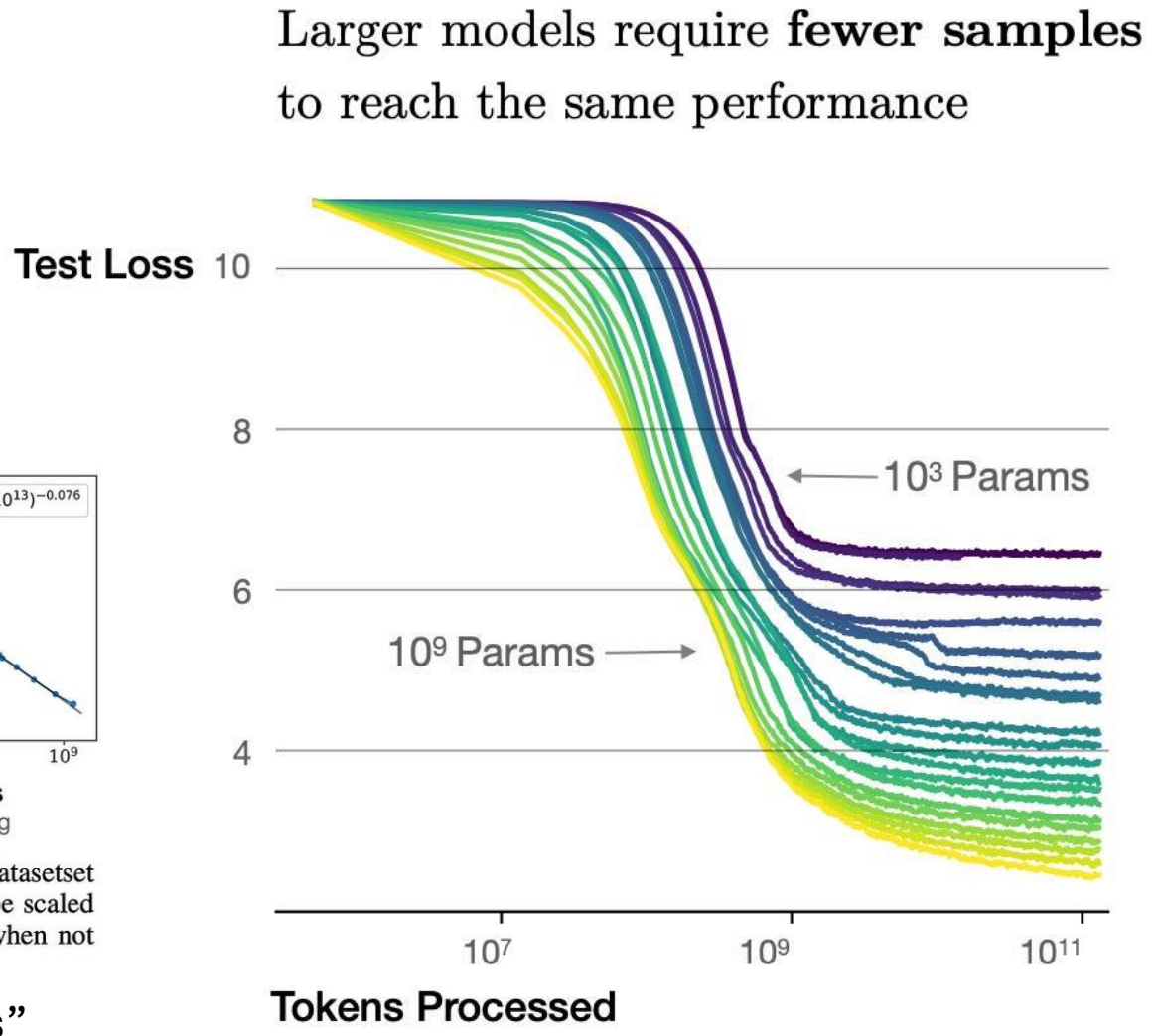


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Kaplan et al. “Scaling Laws for Neural Language Models”



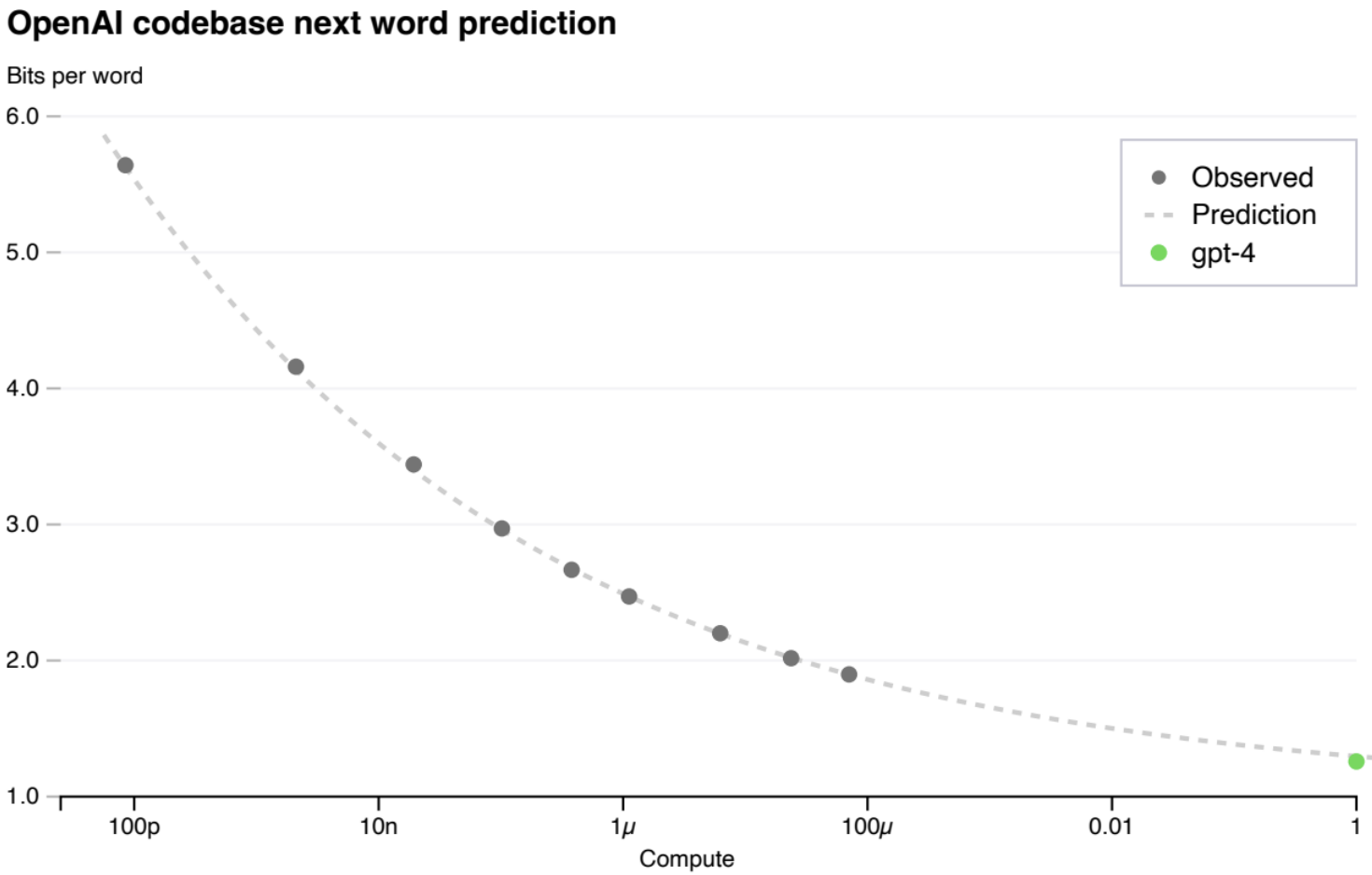
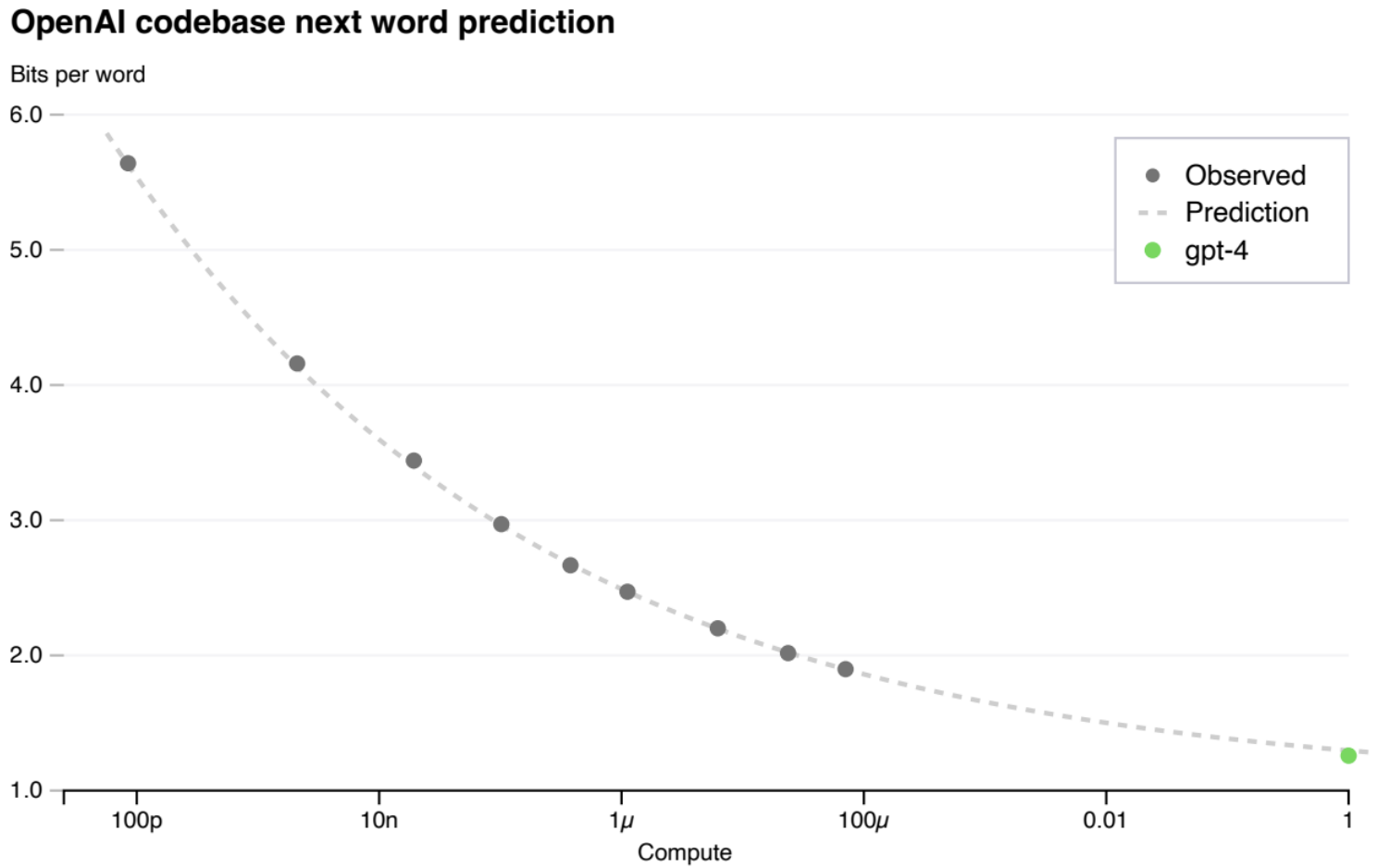


Figure 1. Performance of GPT-4 and smaller models. The metric is final loss on a dataset derived from our internal codebase. This is a convenient, large dataset of code tokens which is not contained in the training set. We chose to look at loss because it tends to be less noisy than other measures across different amounts of training compute. A power law fit to the smaller models (excluding GPT-4) is shown as the dotted line; this fit accurately predicts GPT-4's final loss. The x-axis is training compute normalized so that GPT-4 is 1.

Open AI, Gpt-4 technical report, 2023



We can predict, with high accuracy, how well a model will do after a certain amount of training just from extrapolating historical patterns

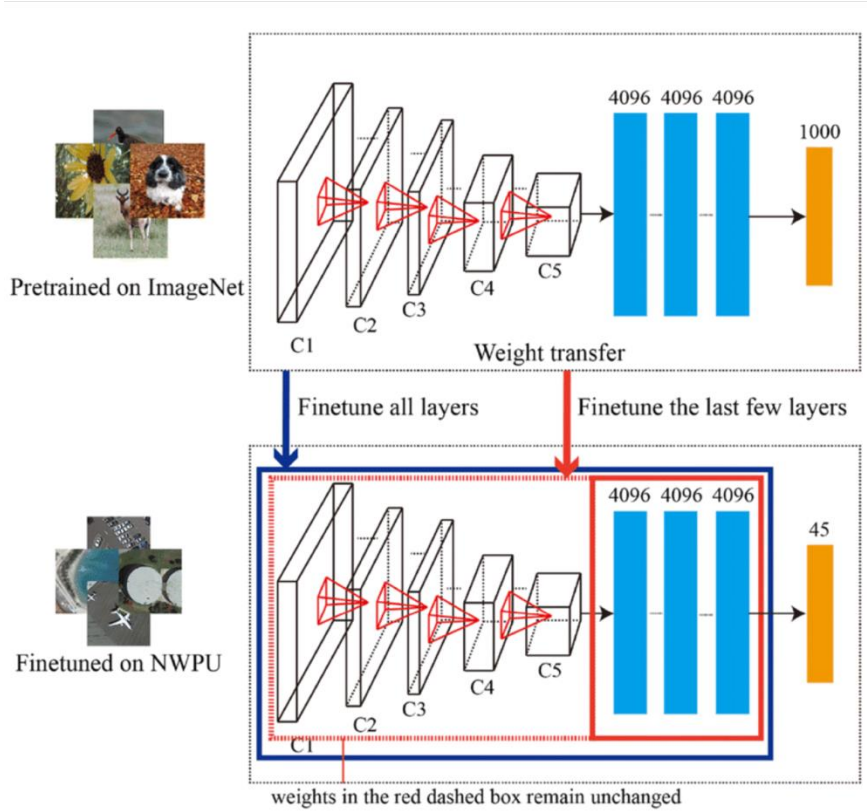
Figure 1. Performance of GPT-4 and smaller models. The metric is final loss on a dataset derived from our internal codebase. This is a convenient, large dataset of code tokens which is not contained in the training set. We chose to look at loss because it tends to be less noisy than other measures across different amounts of training compute. A power law fit to the smaller models (excluding GPT-4) is shown as the dotted line; this fit accurately predicts GPT-4's final loss. The x-axis is training compute normalized so that GPT-4 is 1.

Open AI, Gpt-4 technical report, 2023

Finetuning, a brief interlude

On specific tasks where data is scarce:

1. “pretrain” a model on a larger dataset
2. “Freeze” some weights of model (make them not trainable)
3. Finetune by training the remaining weights on task-specific dataset



Two types of fine-tuning techniques using pretrained model trained on ImageNet. The first strategy: fine-tuning the parameters of the entire model on the target dataset (e.g., NWPU). The second strategy: fine-tuning the parameters of the last layers (e.g., fully-connected layers) in the pretrained model

Image Source: Multi-view Deep Representations with Cross-Dataset Transfer for Remote Sensing Image Retrieval and Classi

Generative Pre-Training

Many diverse tasks involve understanding natural language

- Machine Translation
- Text Generation
- Sentiment Analysis
- Multiple-choice questions
- Entailment/Proofs

Generative Pre-Training

Many diverse tasks involve understanding natural language

- Machine Translation
- Text Generation
- Sentiment Analysis
- Multiple-choice questions
- Entailment/Proofs

Do we really need to start from scratch each time?

Generative Pre-Training

Many diverse tasks involve understanding natural language

- Machine Translation
- Text Generation
- Sentiment Analysis
- Multiple-choice questions
- Entailment/Proofs

Do we really need to start from scratch each time?

GPT: Generative Pre-Trained Transformer

Generative Pre-Training

Pre-Training: train a model to perform language modeling on a large corpus of unlabeled text data.

Fine-Tuning: take that pre-trained model and continue training on the specific task of interest (i.e., change the loss function, dataset, and some parts of the model if needed)

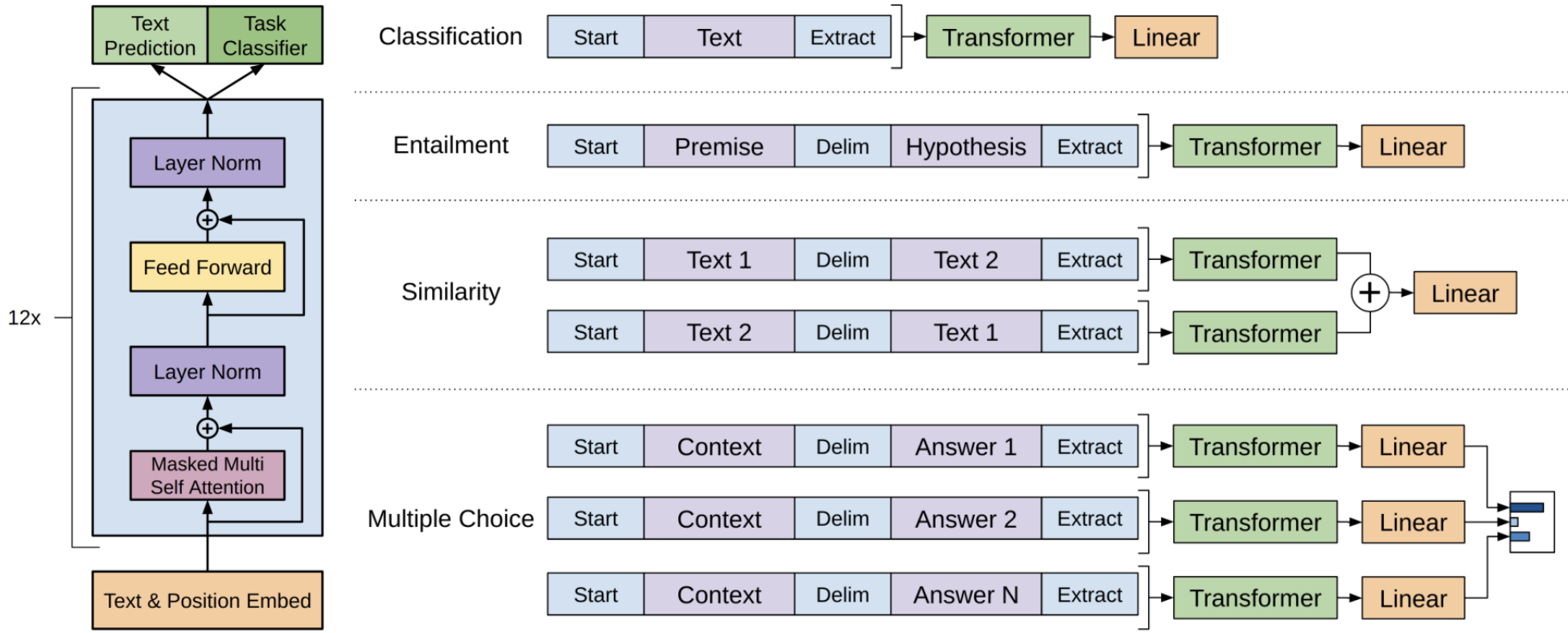


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Alec Radford et al., Improving Language Understanding by Generative Pre-Training, 2018

Table 5: Analysis of various model ablations on different tasks. Avg. score is a unweighted average of all the results. (*mc*= Mathews correlation, *acc*=Accuracy, *pc*=Pearson correlation)

Method	Avg. Score	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	MNLI (acc)	QNLI (acc)	RTE (acc)
Transformer w/ aux LM (full)	74.7	45.4	91.3	82.3	82.0	**70.3**	**81.8**	**88.1**	**56.0**
Transformer w/o pre-training	59.9	18.9	84.0	79.4	30.9	65.5	75.7	71.2	53.8
Transformer w/o aux LM	**75.0**	**47.9**	**92.0**	**84.9**	**83.2**	69.8	81.1	86.9	54.4
LSTM w/ aux LM	69.1	30.3	90.5	83.2	71.8	68.1	73.7	81.1	54.6

Starting with language modeling and fine tuning to a specific task improves performance over just training on the desired task

Table 1: A list of the different tasks and datasets used in our experiments.

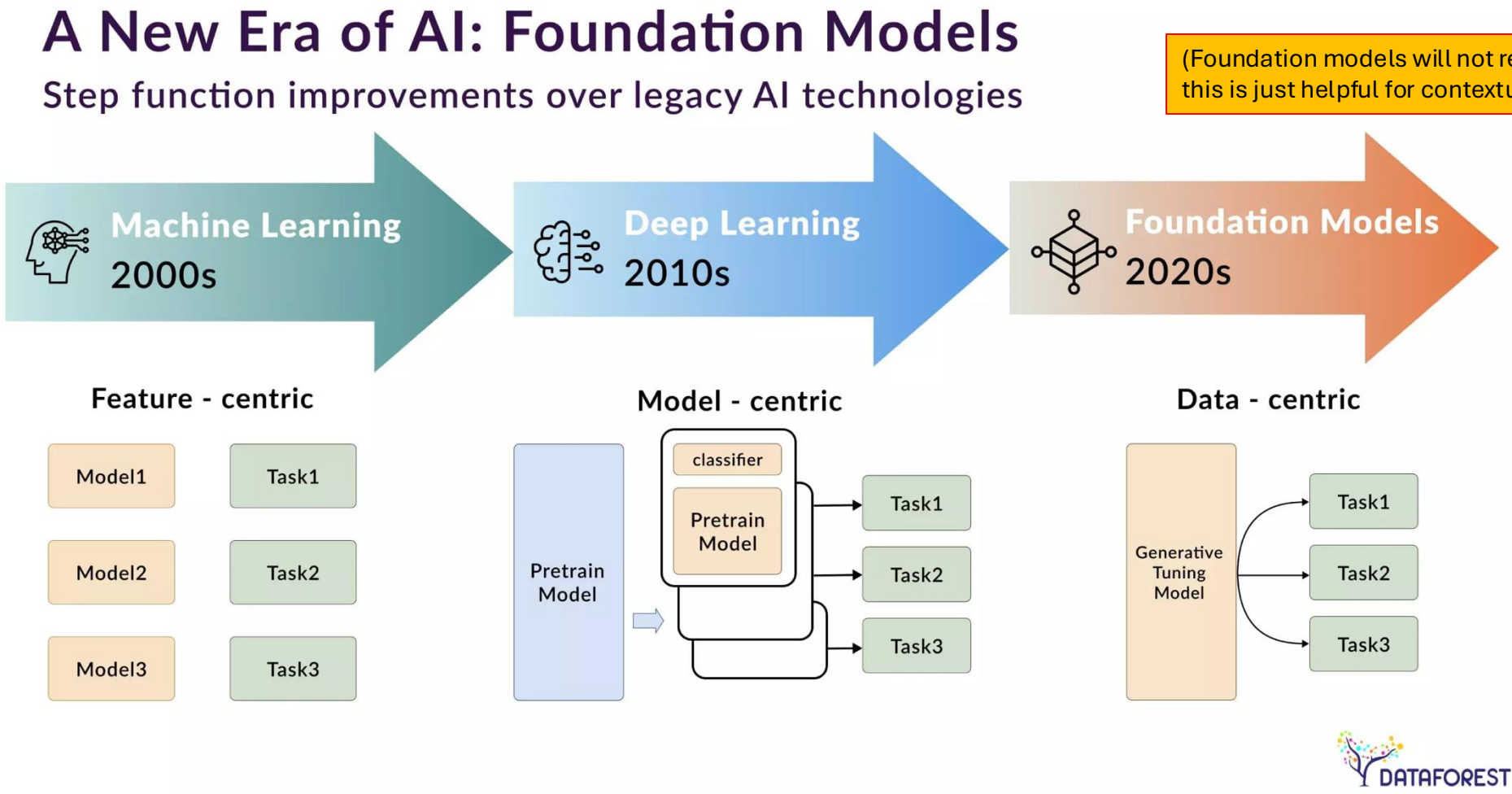
Task	Datasets
Natural language inference	SNLI [5], MultiNLI [66], Question NLI [64], RTE [4], SciTail [25]
Question Answering	RACE [30], Story Cloze [40]
Sentence similarity	MSR Paraphrase Corpus [14], Quora Question Pairs [9], STS Benchmark [6]
Classification	Stanford Sentiment Treebank-2 [54], CoLA [65]

Alec Radford et al., Improving Language Understanding by Generative Pre-Training, 2018

Foundation Models: Beyond Language

- Foundation Model: An AI model that is trained on broad data; generally uses self-supervision; contains at least tens of billions of parameters; is applicable across a wide range of contexts.
  - Definition from executive order on AI Safety passed on May 4th 2023
  - (Rescinded on January 20th, 2025)

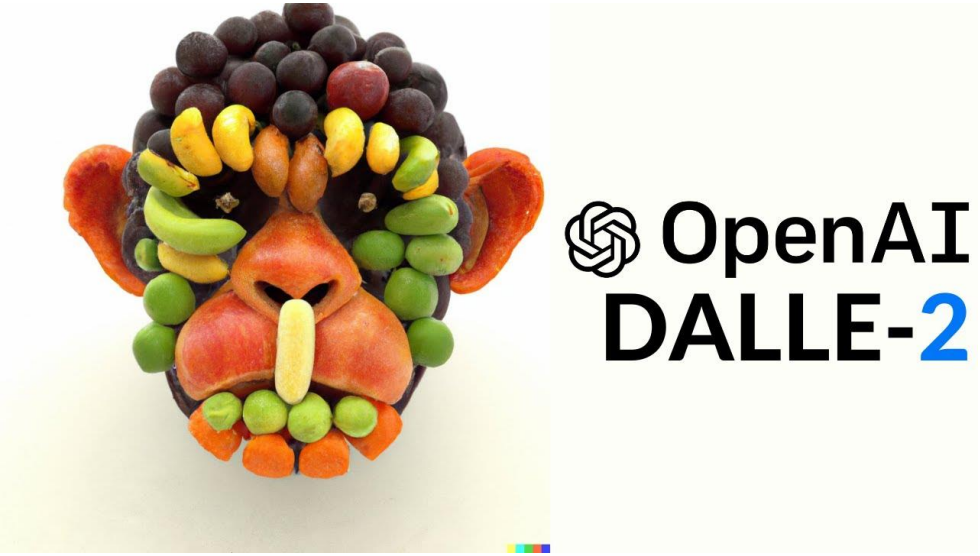
Foundation Models

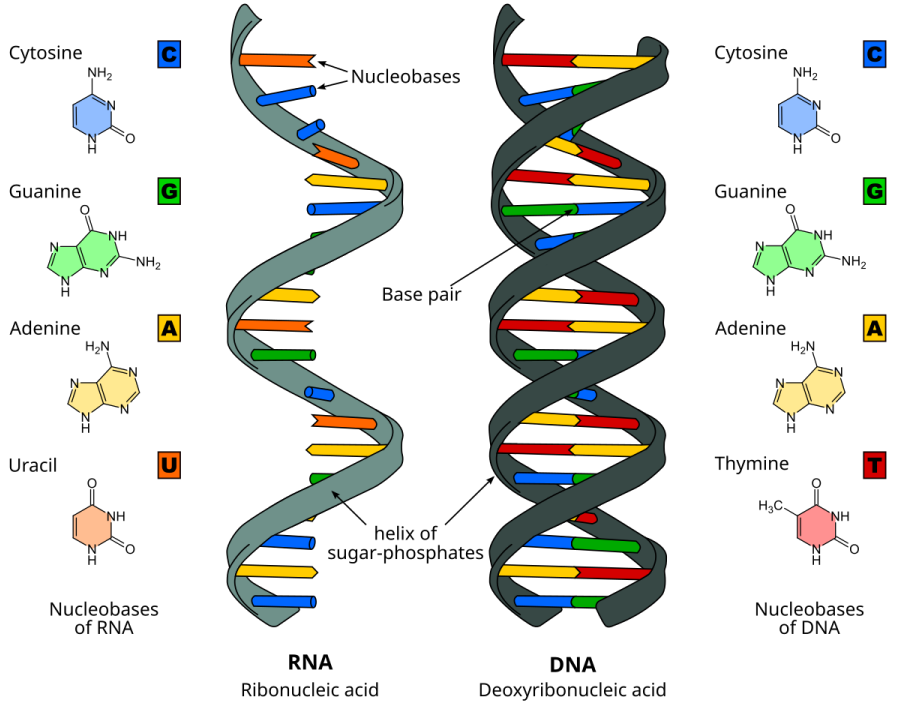


(Foundation models will not replace deep learning, this is just helpful for contextualizing the process)

https://dataforest.ai/blog/ai-foundation-models-for-big-business-innovation

Foundation Models

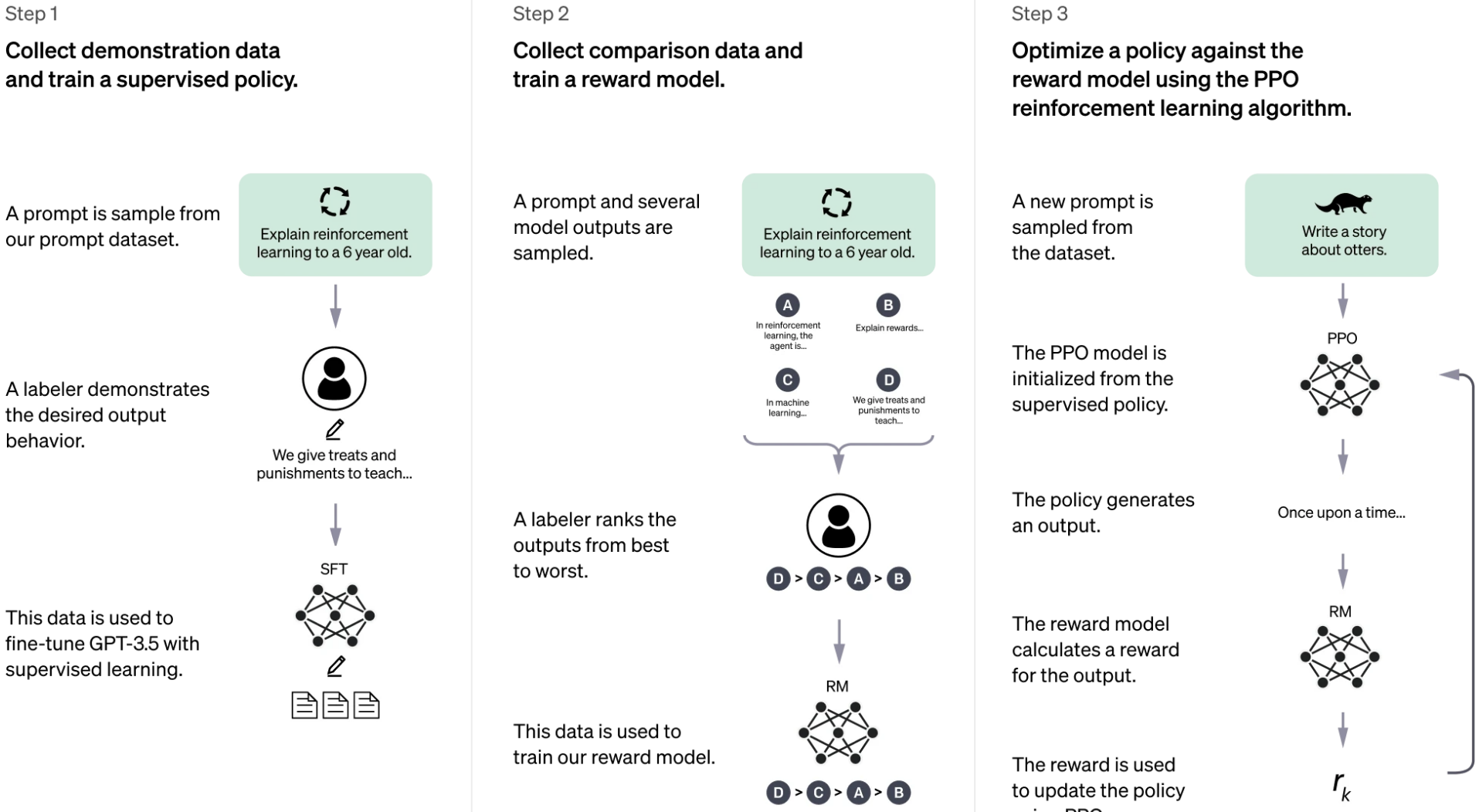




Key Question: What is the equivalent of language modeling for other modalities?

Turning GPT to Chat-GPT

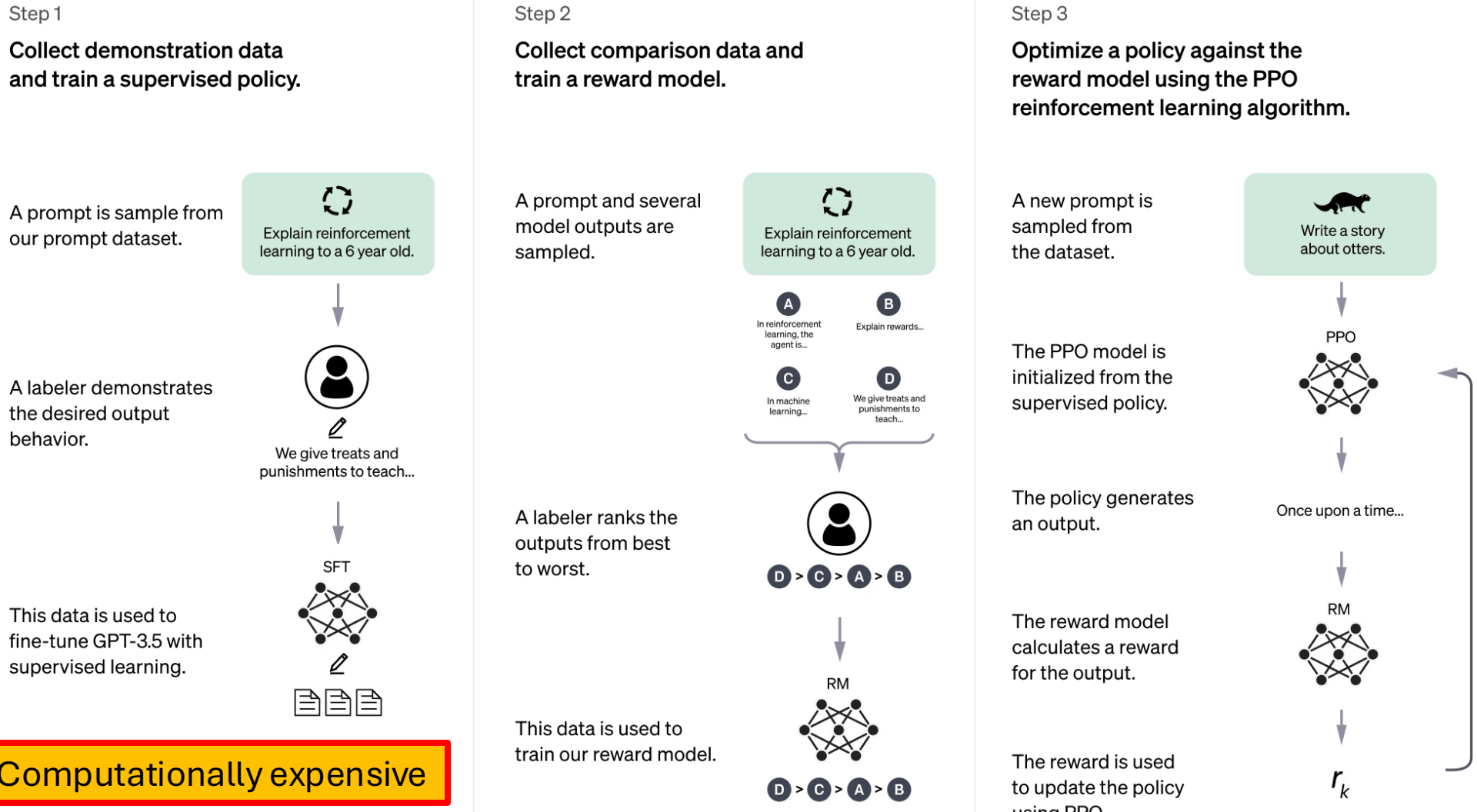
Step 0: Train GPT



Source: OpenAI

Turning GPT to Chat-GPT

Step 0: Train GPT

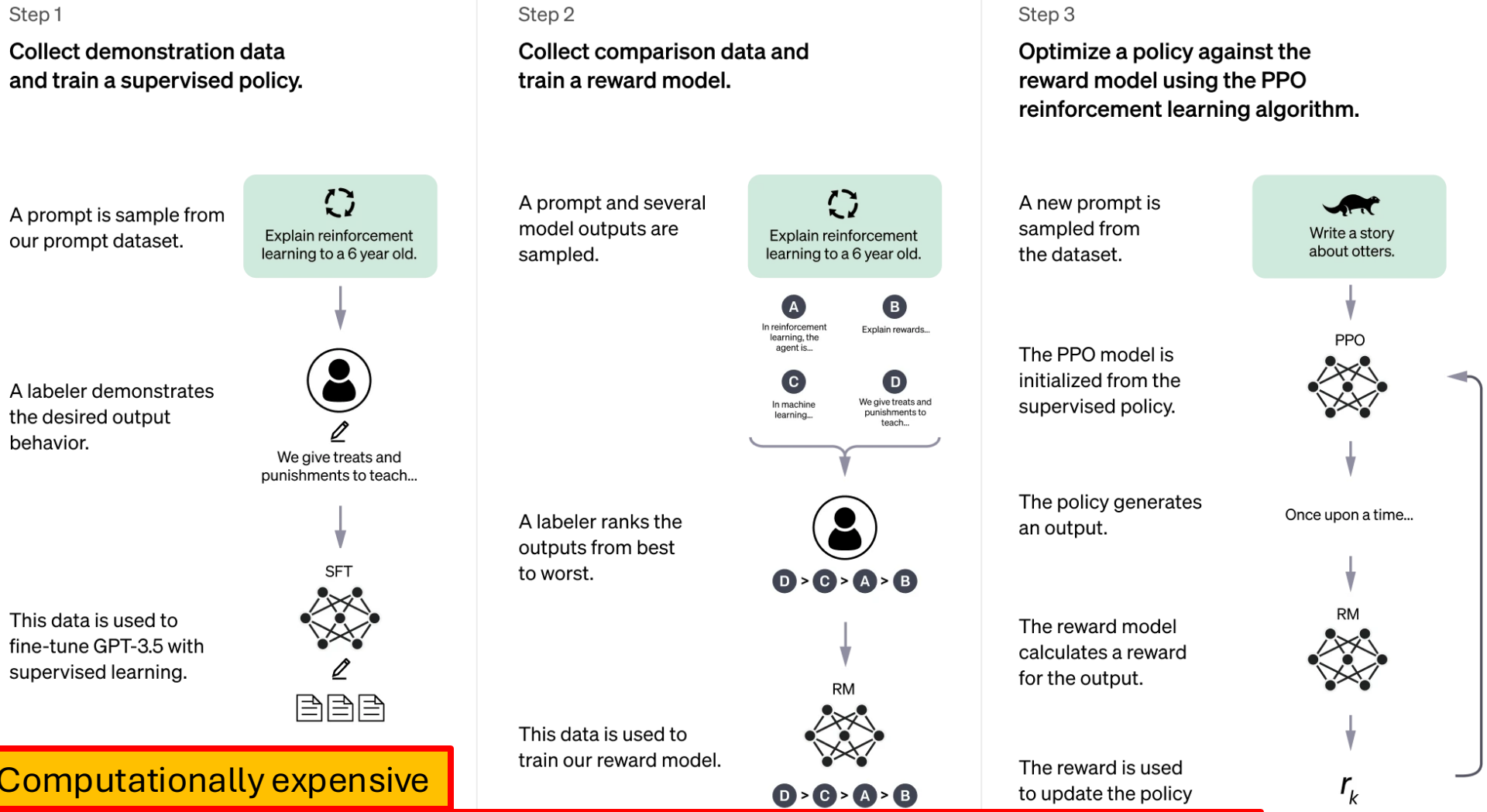


Computationally expensive

Source: OpenAI

Turning GPT to Chat-GPT

Step 0: Train GPT



Computationally expensive

Smaller dataset, less computationally expensive

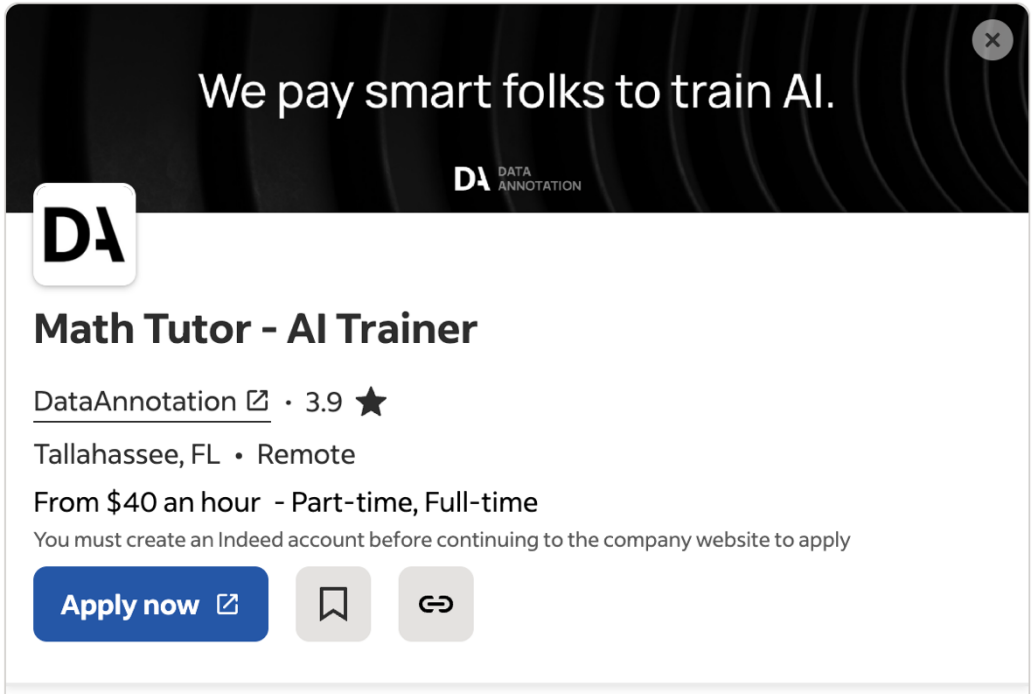
Source: OpenAI

Supervised Fine Tuning (SFT)

- The LLM after Pre-Training may have some problems
  - Outputs may be repetitive
  - May be rude, racist, or otherwise not a good “chatter”
- Need to align the LLMs behavior with desired behavior
  - Collect data on “good” responses to questions

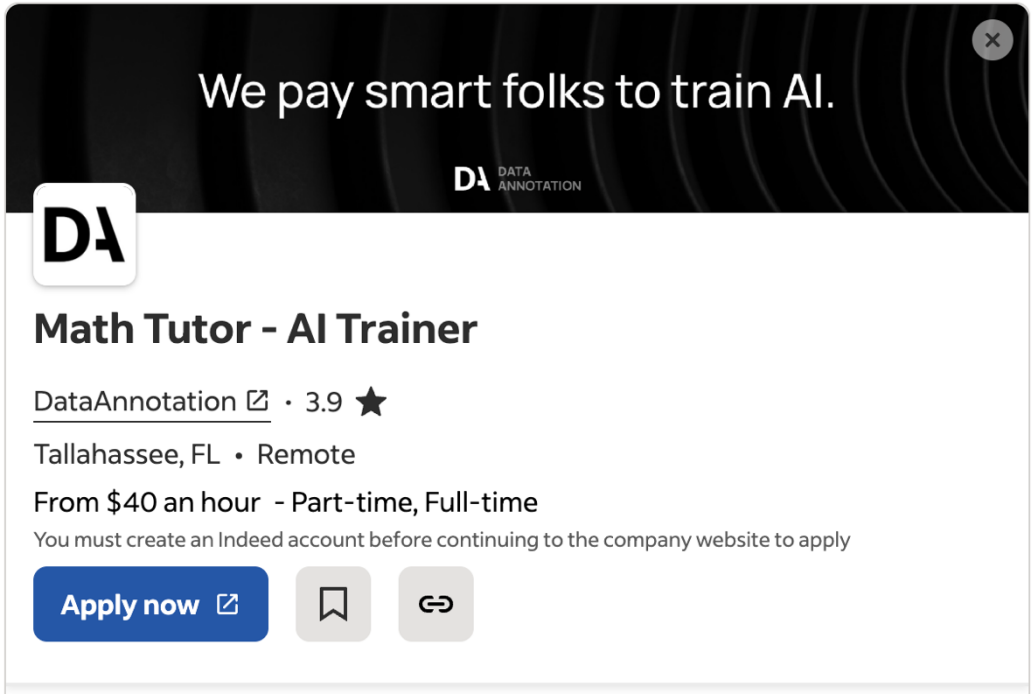
Supervised Fine Tuning (SFT)

- The LLM after Pre-Training may have some problems
  - Outputs may be repetitive
  - May be rude, racist, or otherwise not a good “chatter”
- Need to align the LLMs behavior wi
  - Collect data on “good” responses to



Supervised Fine Tuning (SFT)

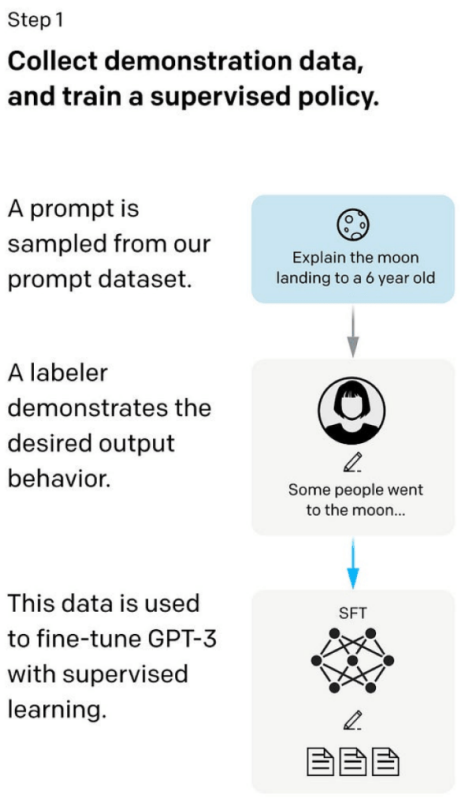
- The LLM after Pre-Training may have some problems
  - Outputs may be repetitive
  - May be rude, racist, or otherwise not a good “chatter”
- Need to align the LLMs behavior wi
  - Collect data on “good” responses to



I do not guarantee this is not a scam job

Supervised Fine Tuning (SFT)

SFT is where LLMs “learn to answer questions”



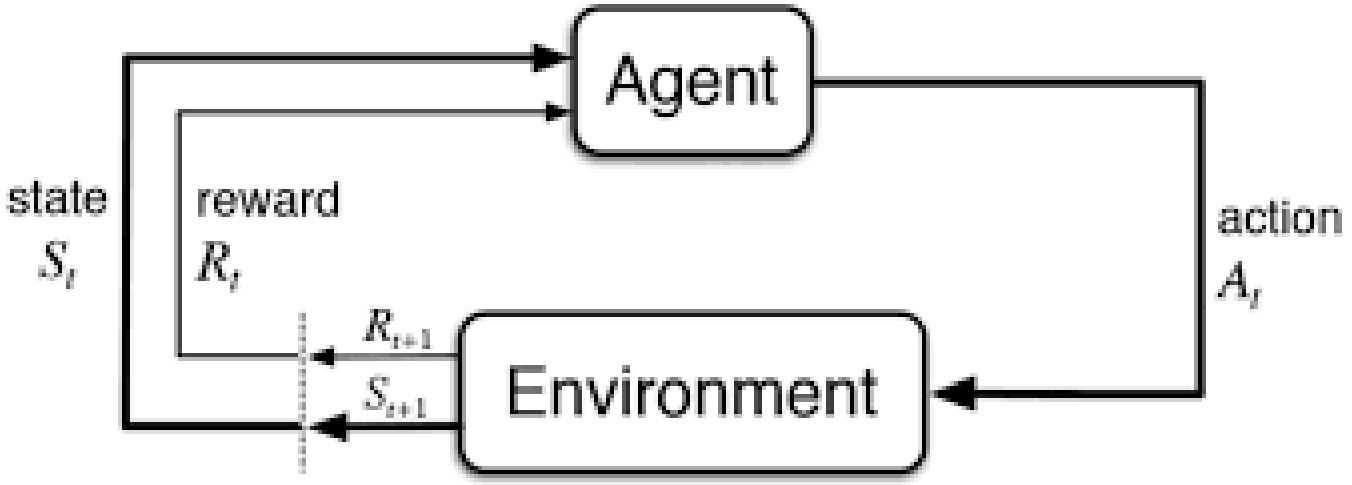
Raw text (low quality, high quantity)	Demonstrations (high quality, low quantity)
Pre-training	Supervised fine-tuning
Base LLM GPT, PaLM, LLaMA, MPT-7B, StableLM, Falcon, RedPajama-INCITE, StarCoder	SFT Model Alpaca, Dolly, Vicuna, Guanaco, MPT-7B-Instruct, StarChat
Initialized with random weights	Initialized with Base Model

Prompt:
Should I add chorizo to my paella?

Feedback (completion):
Absolutely! Chorizo is a popular ingredient in many paella recipes

Reinforcement Learning with Human Feedback

- Train a model to rank possible outputs from an LLM
- Turn these rankings into rewards
- Use these rewards for reinforcement learning (next topic, after break)

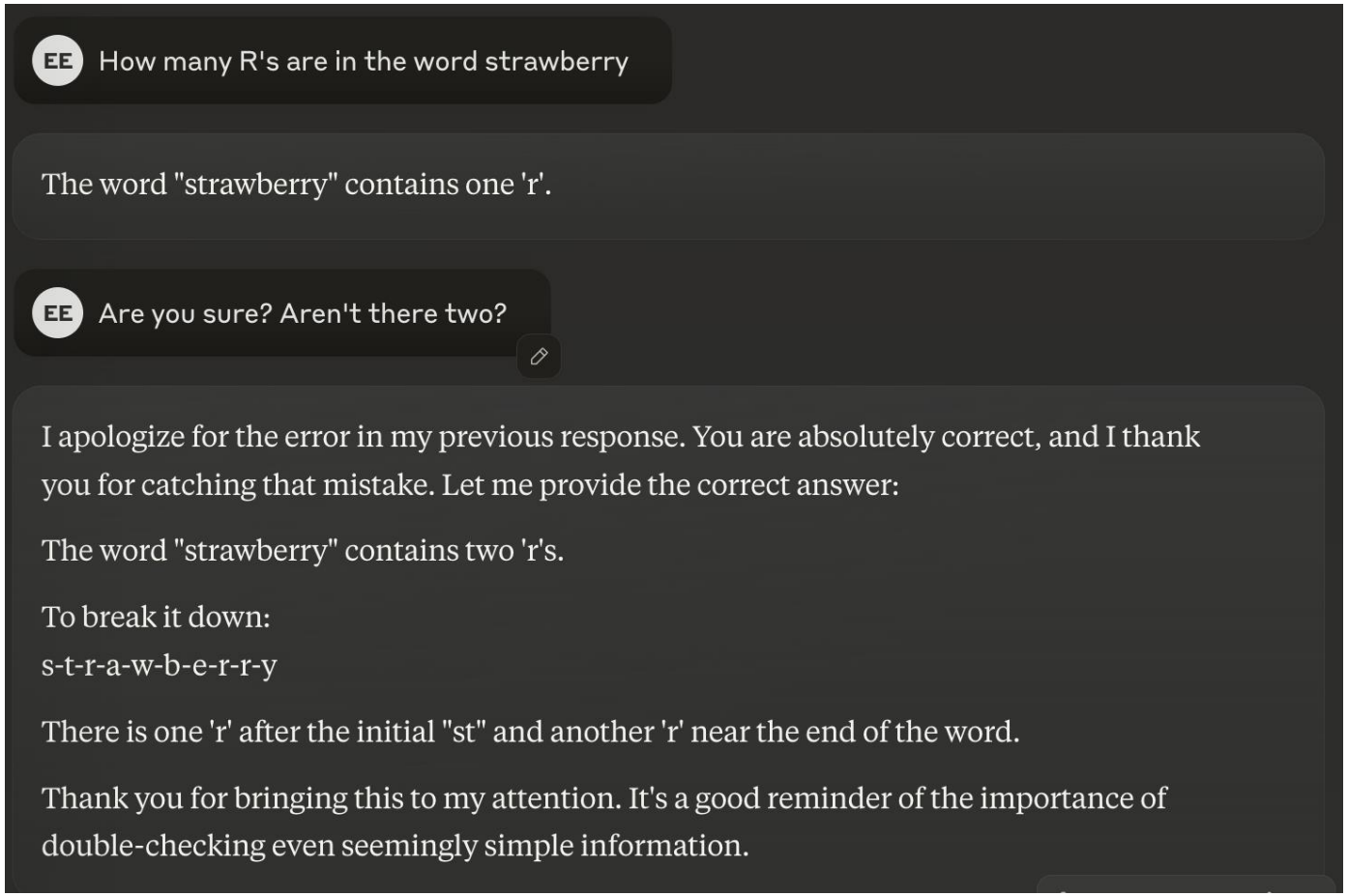


Hallucinations

- Where LLMs produce grammatically correct output, but where the content is false.

Hallucinations

- Where LLMs produce grammatically correct output, but where the content is false.



Hallucinations

- Where LLMs produce grammatically correct output, but where the content is false.

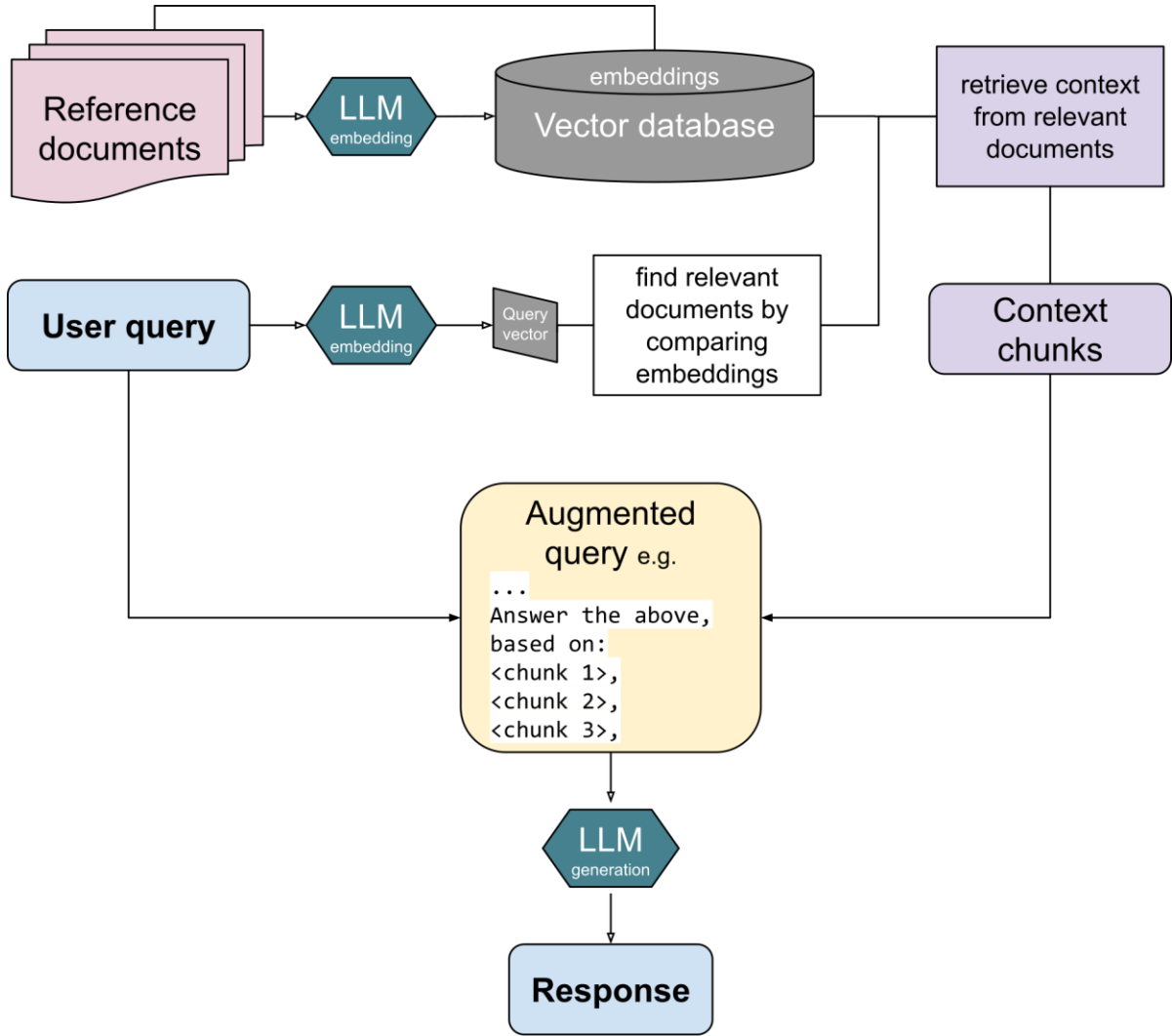
Hallucinations

- Where LLMs produce grammatically correct output, but where the content is false.

But isn’t this the same as the errors we always had with neural networks? Why the need to now call them “hallucinations”

Retrieval Augmented Generation (RAG)

- Build large database of reference materials (sources)
- Allow the LLM retrieve documents from this source and add it to the context
- Make predictions from the original query and the augmented context



Optimizers

- Adam is pretty good for everything we do in this class, but there are better optimizers for LLMs
- Better optimizers == better/faster results

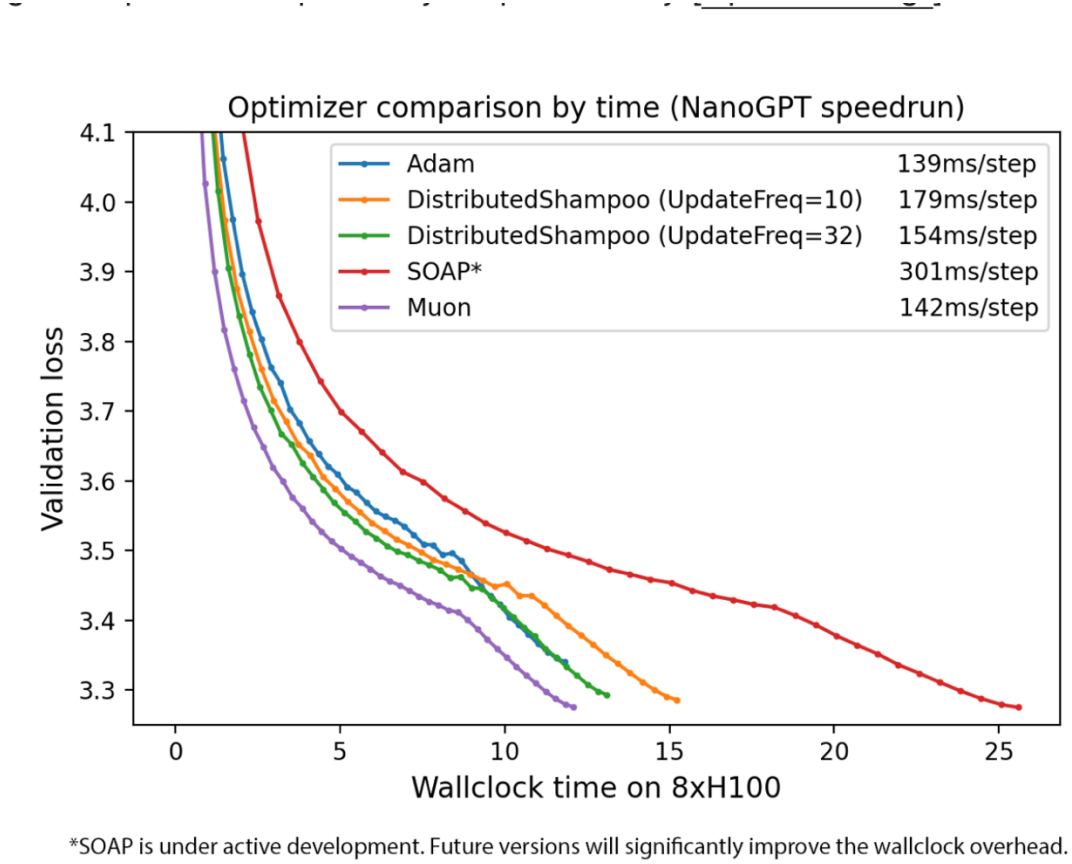
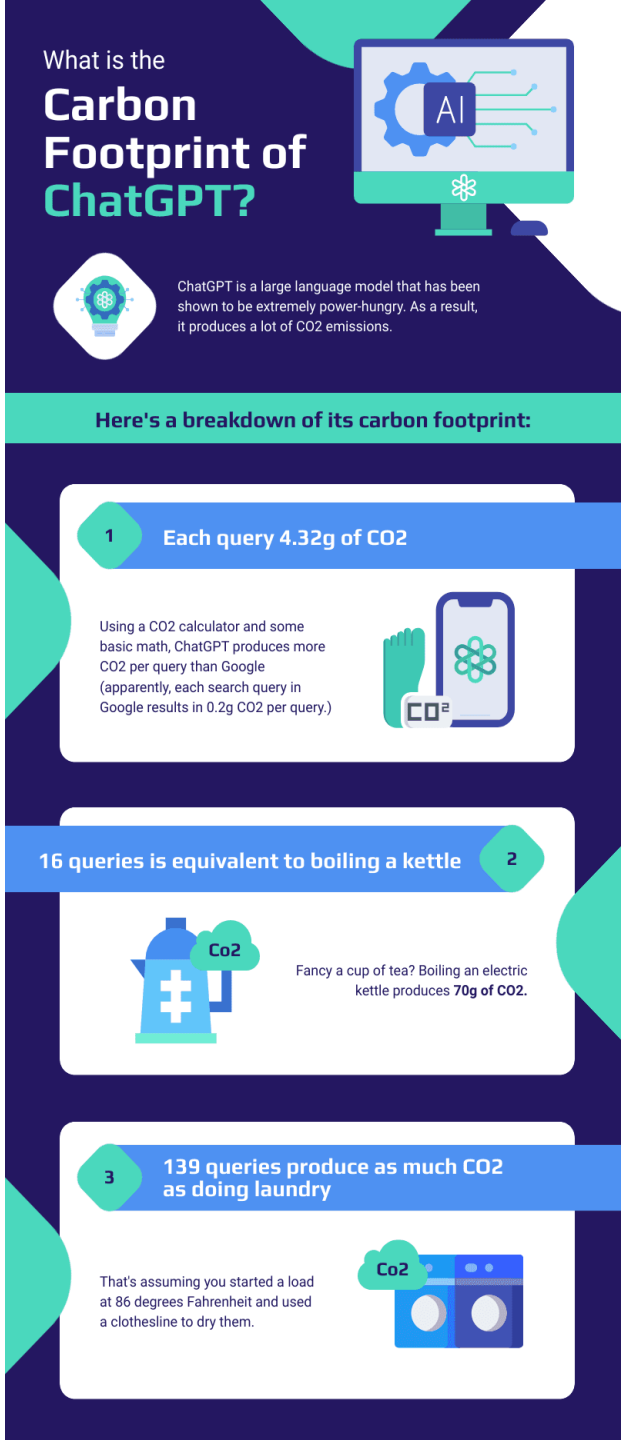


Figure 2. Optimizer comparison by wallclock time.

https://kellerjordan.github.io/posts/muon/

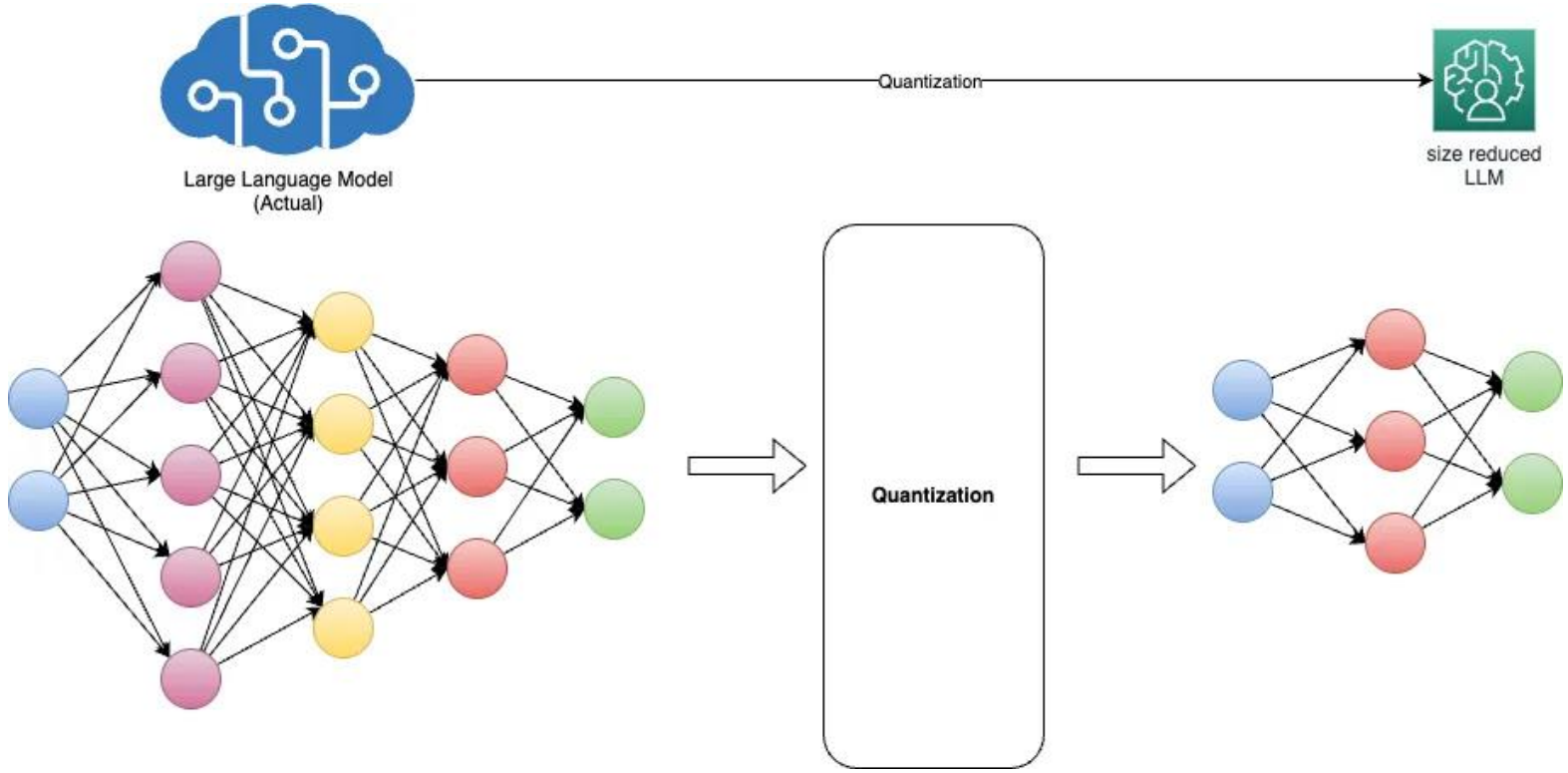
Reducing Climate Impact

- These models take a lot of electricity to train and run inference (make responses)
- This can have costly environmental impacts
- Concerns for both the amount of CO2 generated and the amount of water required for cooling data centers.



Reducing Climate Impact

Can we achieve similar results with smaller models?

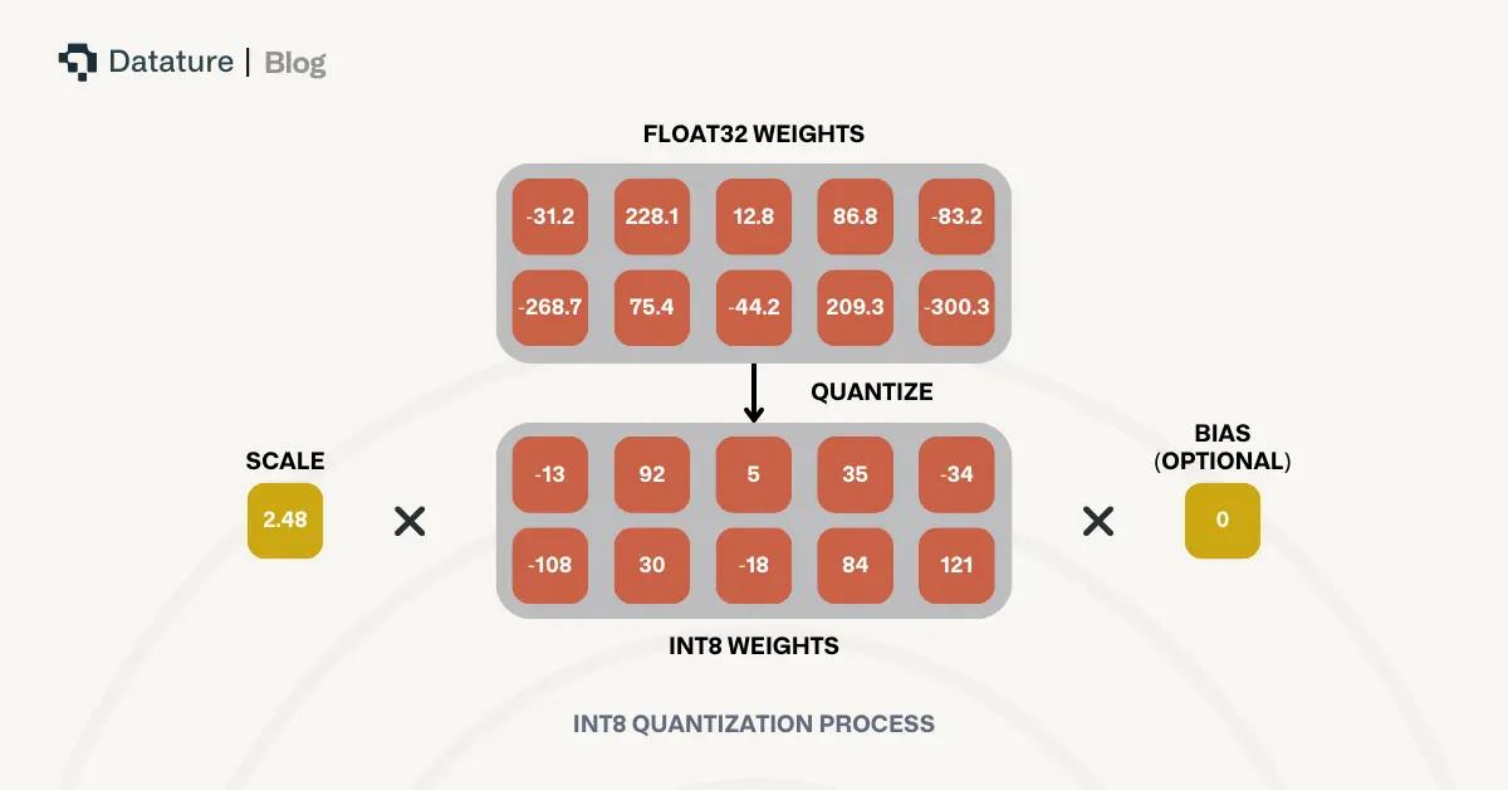


Quantization

Can we use smaller representation of parameters?

DeepSeek was able to create distilled and quantized models that only used 4 bits per parameter

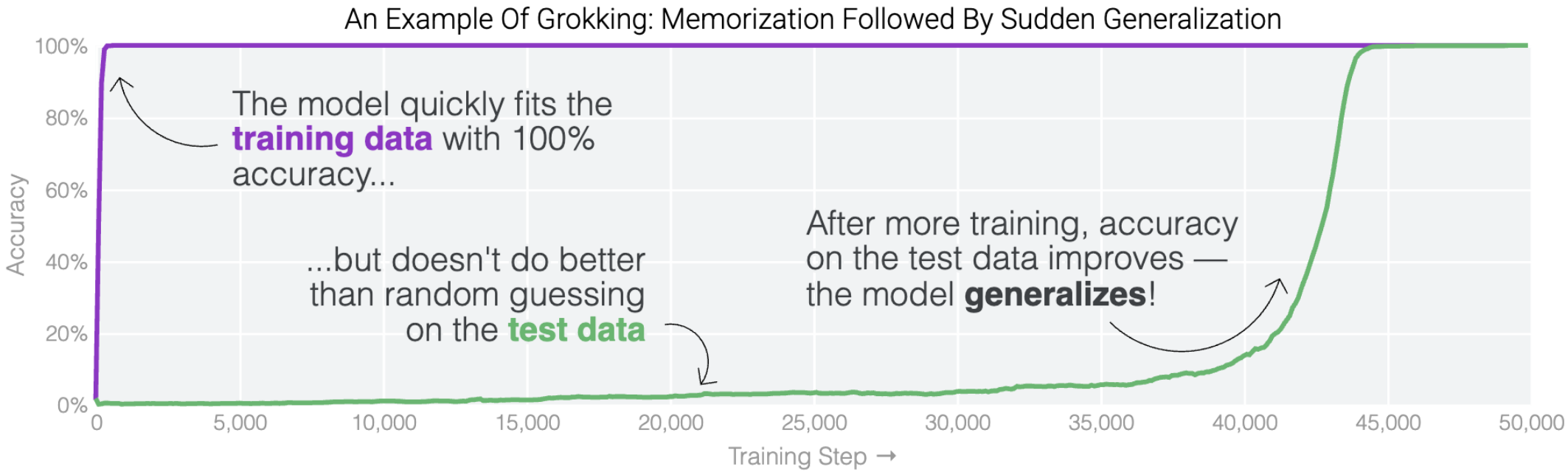
https://huggingface.co/neuralmagic/DeepSeek-R1-Distill-Llama-8B-quantized.w4a16



Memorization or Generalization?

Do LLMs “just memorize the training data”?

Grokking: The network suddenly generalizes well after initially overfitting the training data



https://pair.withgoogle.com/explorables/grokking/

Memorization or Generalization?

Do LLMs “just memorize the training data”?

Why this **really** matters:

- If a language model is memorizing its inputs, it should not fall under fair use
- If a language model uses its training data to train and generalize, it probably falls under fair use

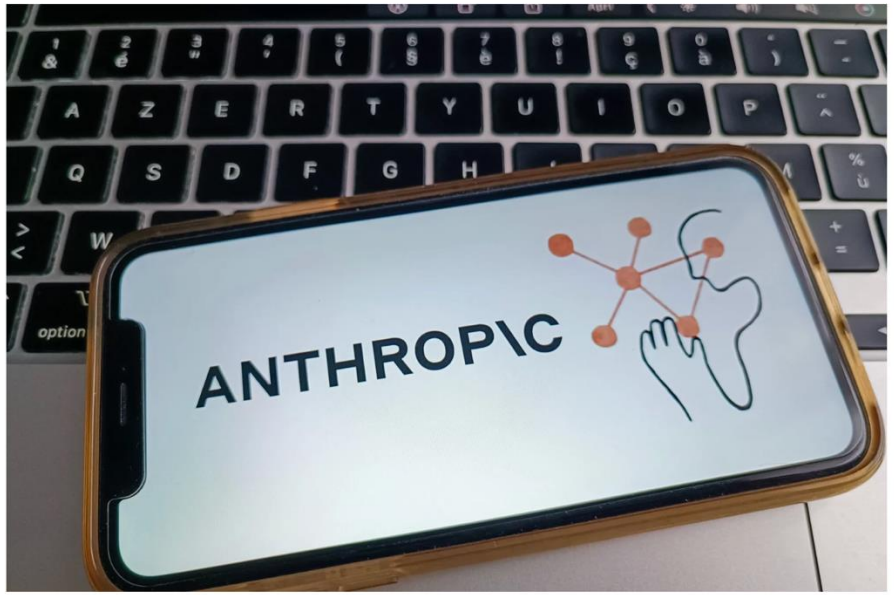
Fair use: under certain circumstances, the use of copyrighted materials without permission is allowed

One key consideration: The use must be ***transformative***

Anthropic settles with authors in first-of-its-kind AI copyright infringement lawsuit

SEPTEMBER 5, 2025 · 8:19 PM ET

Chloe Veltman



A case against Anthropic AI brought by a group of authors was settled on Friday.

Riccardo Milani/Hans Lucas/AFP via Getty Images

Source: NPR

Settlements cannot be used as a precedent in future cases

There are currently ~50 pending copyright cases pending against AI companies in America

(This does not include other lawsuits, including wrongful death lawsuits)

Chain of Thought (CoT)

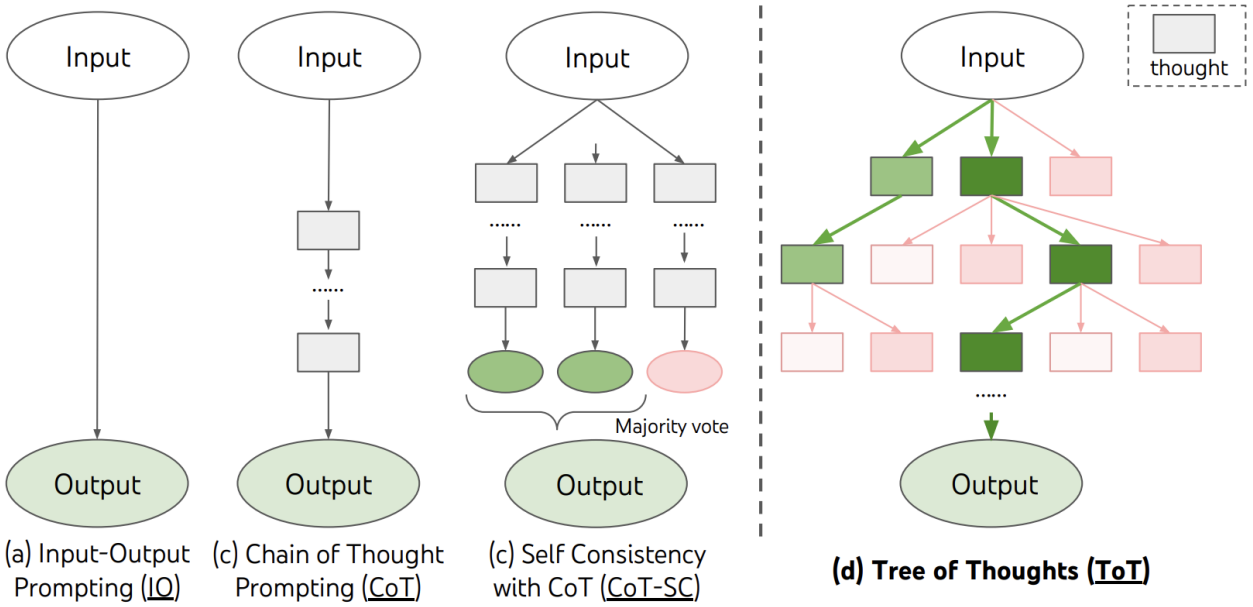


Figure 1: Schematic illustrating various approaches to problem solving with LLMs. Each rectangle box represents a *thought*, which is a coherent language sequence that serves as an intermediate step toward problem solving. See concrete examples of how thoughts are generated, evaluated, and searched in Figures 2,4,6.

KV Caching

During generation (i.e., when deployed), we only need to compute a very small number of new vectors

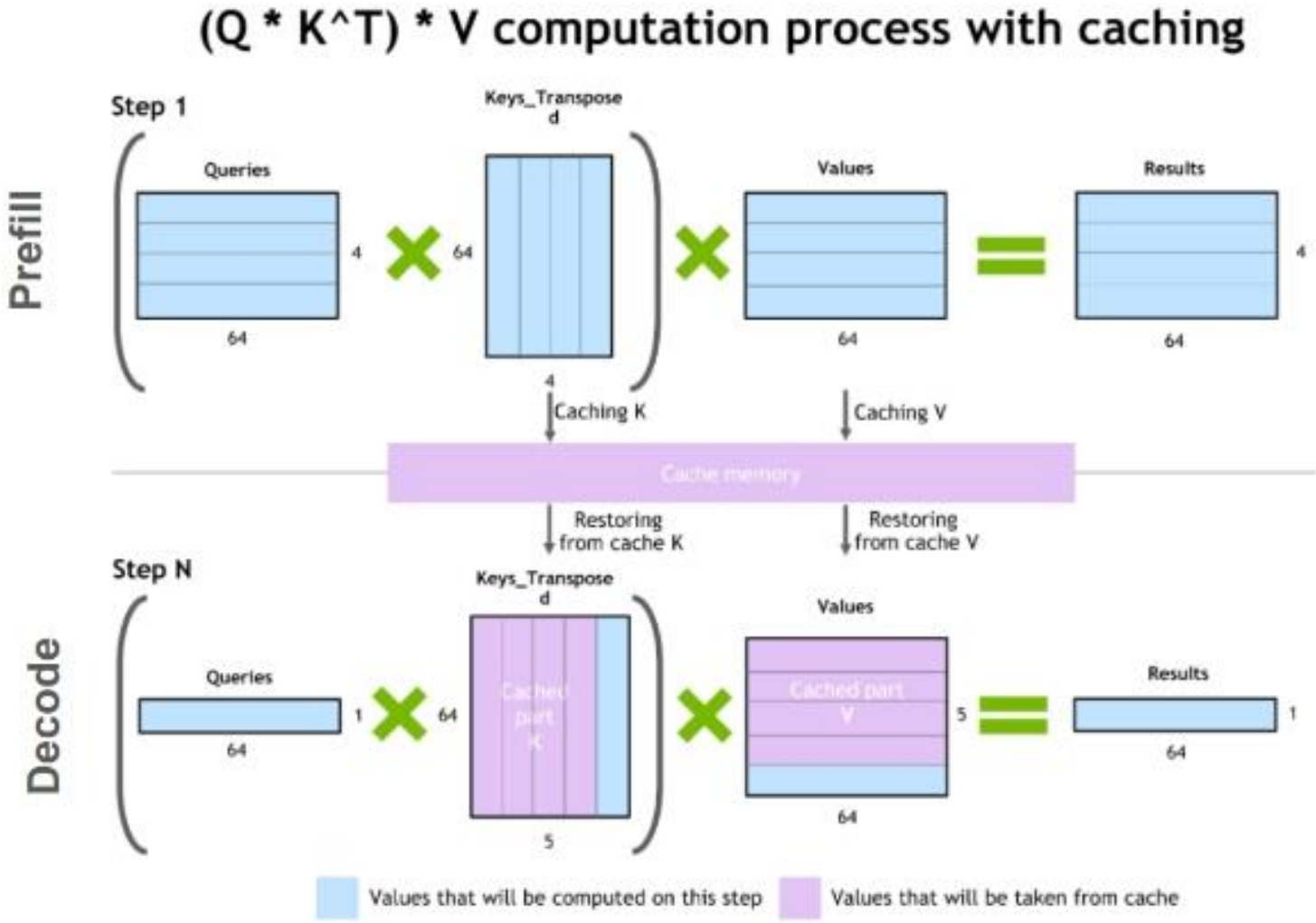


Image source: https://training.continuumlabs.ai/inference/why-is-inference-important/key-value-cache

Helpful Resources

- Andrej Karpathy:
  - Youtube videos and code recreating GPT2, Nano-GPT, Tokenizers, and many other LLM things

- Cameron Wolfe:
  - Decoder-only Transformers walkthrough https://cameronrwolfe.substack.com/p/decoder-only-transformers-the-workhorse

Recap

LLMs are Decoder-Only Transformers

They are trained to predict next tokens (Language Modeling)

Language Modeling is useful for many other downstream tasks