Data 188: Introduction to Deep Learning

NLP Large-scale pretraining

Speaker: Eric Kim
Lecture 22 (Week 14)
2026-04-21, Spring 2026. UC Berkeley.

Announcements

- HW4 released: "Transformers for NLP (machine-translation)"
  - Groups of 4!
    - Ed post: "(HW4) Group finder thread"
  - Start early!

- Final exam scheduling google form sent out
  - See Ed post: "(Action Needed) Final Exam Scheduling"
  - DSP students with exam accommodations: we sent you an email with a separate form.
  - **Due date: must submit form by Wed April 22nd 11:59 PM PST!**

- University Course Evaluations **released! Due: May 10th, 2026.**
  - Extra Credit if enough students fill it out (details TBD on Ed)

Today's lecture

Linear probing

NLP Large-scale pretraining

Encoder Only: BERT

Decoder only: GPT, GPT-2, GPT-3, GPT-4

Linear probing

Linear probing: a technique to quickly and easily add a lightweight classifier on top of an existing pretrained model.

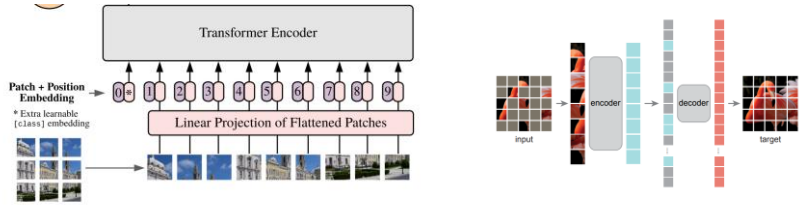
Ex: suppose I pretrained a ViT model via MAE.

To build an ImageNet-1k classifier: add a single Linear layer at the "end" of the model (produces the logits).

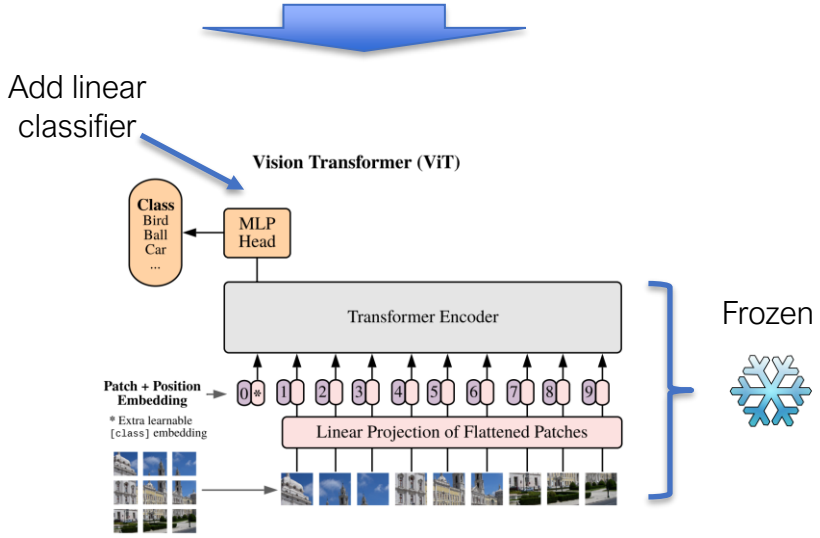
Then, finetune only this Linear layer (all other model weights are frozen!).

Linear probing will underperform "full finetuning", but is a decent way of quickly estimating the embedding representation quality.

Stage 1: pretrain



Stage 2: finetune classifier head



Training methodology

Recall that transformer models, while powerful, are extremely data and compute hungry.

Traditional training methodology:

1. Randomly initialize model weights
2. Train on target dataset (ie ImageNet-1k).

(New) Large-scale pretraining methodology:

1. Randomly initialize model weights
2. Pretrain on a large pretraining dataset, typically with a self-supervised task (ex: "fill in the blank"). "Foundation model".
3. Finetune on your target dataset/task (ie ImageNet-1k).

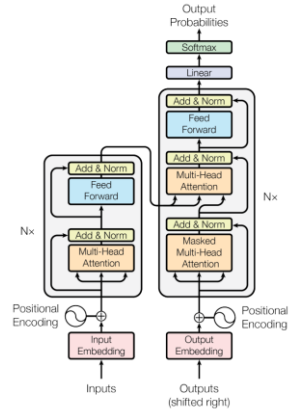


Figure 1: The Transformer - model architecture.

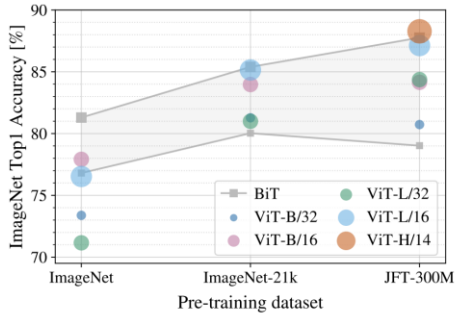


Figure 3: Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows.

BERT

BERT (2018): a transformer encoder for various NLP tasks, including:

- General Language Understanding Evaluation "GLUE"
- "Stanford Question Answering Dataset (SQuAD v1.1, v2.0)

Two phases:

Pre-training: MLM, NSP

Fine tuning: NLP tasks (ex: GLUE, SQuAD, etc)

BERT Pre-training: MLM

"Masked Language Modeling" (MLM). Aka "fill in the blank(s)".

Task: masked token classification.

Input: "A dog is [MASK] a red [MASK]"
GT: "A dog is chasing a red car".

BERT Pre-training: NSP

Next Sentence Prediction (NSP).

Task: Given two sentences A, B, predict whether B is after A.

Binary classification: IsNext, IsNotNext.

Text source (wiki): Deep learning (also called deep structured learning or hierarchical learning) is a kind of machine learning. As with other kinds of machine learning, learning sessions can be unsupervised, semi-supervised, or supervised.

Sentence A: "Deep learning (also called deep structured learning or hierarchical learning) is a kind of machine learning"
Sentence B: " As with other kinds of machine learning, learning sessions can be unsupervised, semi-supervised, or supervised. "
GT: IsNext

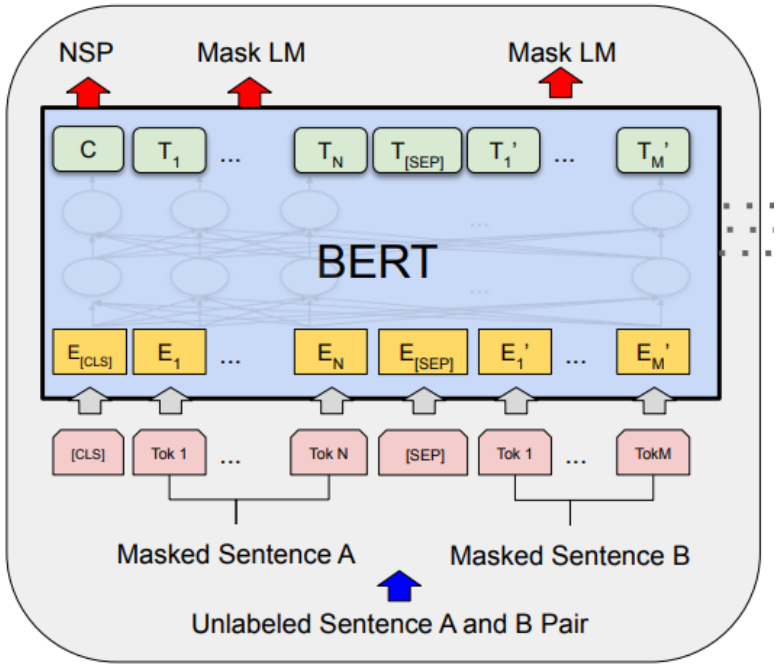
Sentence A: "Deep learning (also called deep structured learning or hierarchical learning) is a kind of machine learning "
Sentence B: "It is defined by its fast-paced, energetic tempos, and emphasis on classic pop songcraft, as well as adolescent and anti-suburbia themes."
GT: IsNotNext

BERT: Pre-training model arch

Input special tokens: [CLS], and [SEP] (separates SentenceA and SentenceB).

MLM: predict masked token T_i

NSP: Use [CLS] embedding to predict "Does SentenceB go after SentenceA?"



Pre-training

BERT fine-tuning: [SEP]

The NLP tasks that BERT handles are diverse:

- Text classification
- Question answering
- Paired sentence semantics ("Is A mean the same as B?")

BERT uniformly handles these by organizing its inputs to accept one or two input sentences, separated by a [SEP] token.

All NLP tasks are expressed via either:

- Single sentence classification task(s)
- Paired sentence task(s)

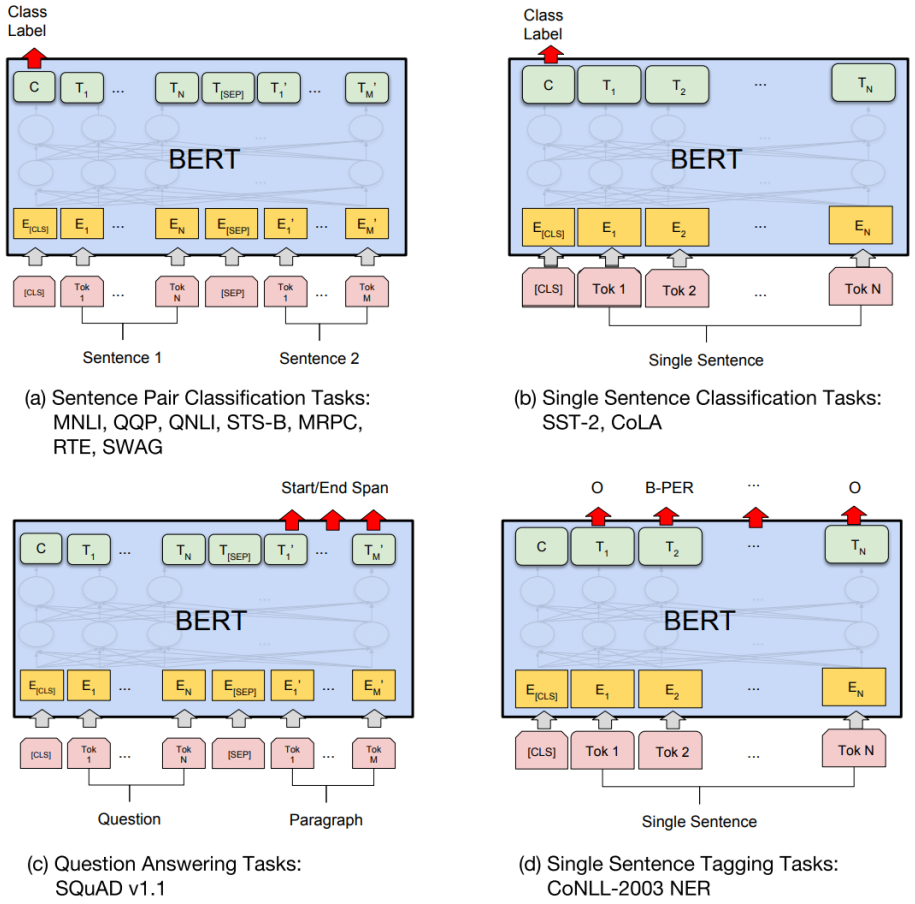


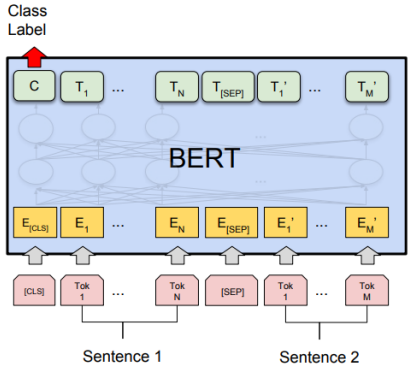
Figure 4: Illustrations of Fine-tuning BERT on Different Tasks.

BERT fine-tuning: single/pair classification

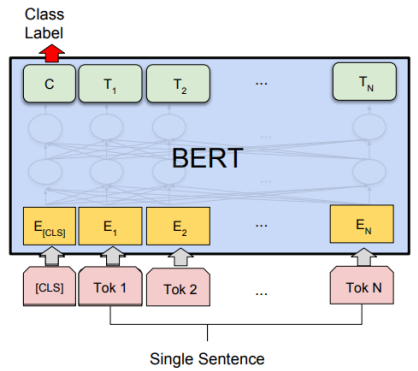
For classification tasks that involve predicting something about the entire input sentence (or pair of sentences): use the [CLS] token embedding for classification.

Examples:

- **Quora Question Pairs:** binary classification task. Determine if two questions asked on Quora are semantically equivalent.
- **The Stanford Sentiment Treebank** (SST-2): single-sentence sentiment classification task on movie reviews.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA

(optional) SQuAD v1.1 dataset: question answer

SQuAD setup: given a question (SentenceA) and a text passage (SentenceB), find which part of the passage answers the question.

Passage: A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself. A natural number greater than 1 that is not a prime number is called a composite number.

Question: What are numbers greater than 1 that can be divided by 3 or more numbers called?

Answer: composite number

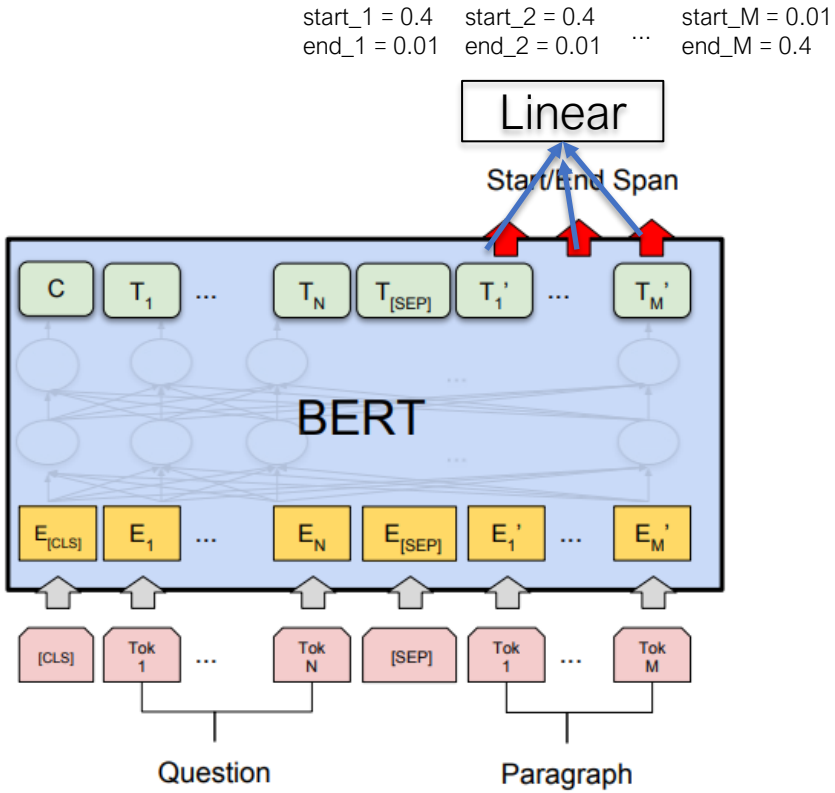
https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/Prime_number.html?model=r-net+%20(ensemble)%20(Microsoft%20Research%20Asia)&version=1.1

(optional) BERT fine-tuning: question/answer

BERT frames this as predicting which token i is [START], and which token j is [END], where i < j.

Learn a Linear layer that, at each token position i, predicts two scores (start_i, end_i).

Prediction: choose i, j that maximizes scores start_i, end_j, where i < j.



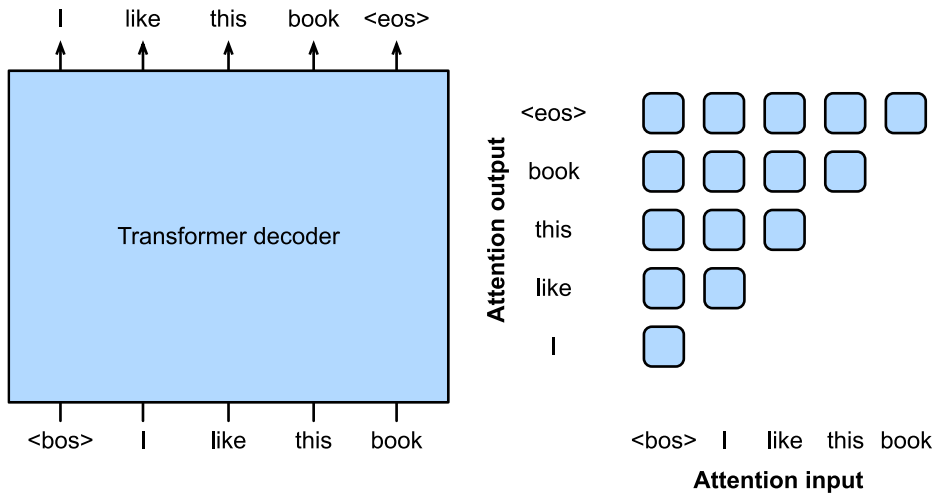
(c) Question Answering Tasks: SQuAD v1.1

Decoder-only pretraining: GPT

GPT: "Generative Pretrained Transformer"

Decoder-only transformer model.

Task: next-token prediction ("language modeling").



https://d2l.ai/chapter_attention-mechanisms-and-transformers/large-pretraining-transformers.html#decoder-only

GPT-1 (2018)

In June 2018, OpenAI released GPT-1. "Improving Language Understanding by Generative Pre-Training"

117M parameters.

Main idea: large-scale pretraining on "next token prediction" task ("language modeling"), followed by fine-tuning, works well!

Pretraining dataset: BookCorpus. "It contains over 7,000 unique unpublished books from a variety of genres including Adventure, Fantasy, and Romance. Crucially, it contains long stretches of contiguous text, which allows the generative model to learn to condition on long-range information.".

Fun fact: BookCorpus was scraped in violation of terms of service, so no longer "officially" exists (oops!)

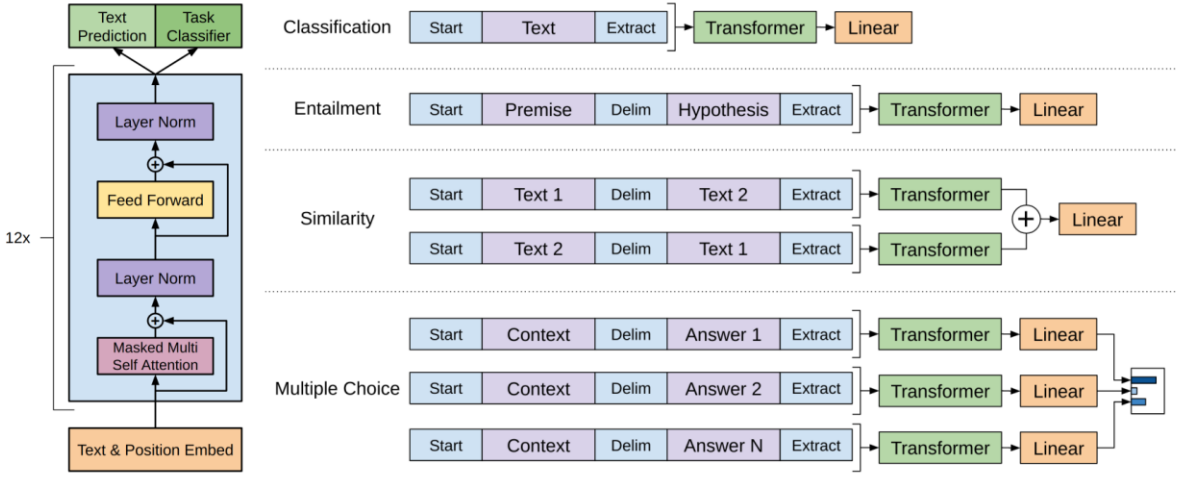


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Similar to how BERT (and T5) carefully organize their inputs to handle a variety of NLP tasks, GPT-1 encodes each task by structuring the inputs appropriately (see paper for details).

GPT-2 (2019)

Release: 2019. "Language Models are Unsupervised Multitask Learners"

Pretraining dataset: **WebText**. 8 million web pages. Scrape pages linked to by Reddit posts with at least 3 karma.

Model size: 1.5B parameters

Pretraining task: next-token prediction.

Estimated train cost: $43k (32 TPU v3 chips for 168 hours, at $8 per TPU per hour)

Main contribution: NO task-specific fine-tuning!

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

Table 2. Architecture hyperparameters for the 4 model sizes.

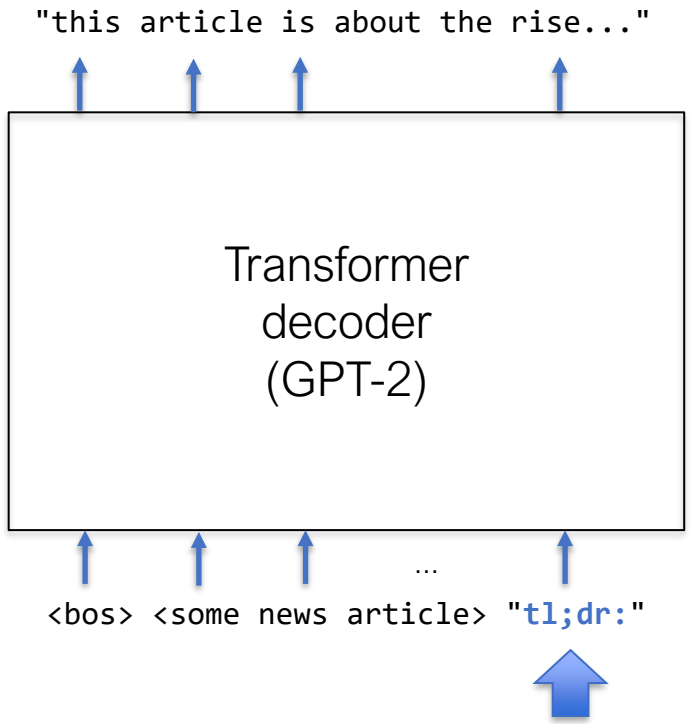
GPT-2: emergent behavior

By pretraining GPT-2 on tons of WebText data (next token prediction), the model could perform many NLP tasks **without explicitly being trained for the task.**

Article summarization: "To induce summarization behavior we add the text TL;DR: after the article and generate 100 tokens... **We use the first 3 generated sentences in these 100 tokens as the summary**".

Rather than train GPT to do summarization, you simply "ask" the decoder to do summarization. Wow!

Similar "task-injection" tricks for: translation, question answering, etc.



GPT-3 (2020)

"Language Models are Few-Shot Learners" (May 2020).

Essentially the same approach as GPT-2, but "bigger and better".

Model size: 175B parameters. Transformer decoder.

Pretraining dataset: Larger (and noisier) scrape of the internet (Common Crawl) + others.

No task-specific fine-tuning!

Impact: significantly better quality of generated text. More "emergent" behavior.

Shows that scaling up both the model size and pretraining dataset continues to improve GPT quality.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table 2.2: Datasets used to train GPT-3. "Weight in training mix" refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

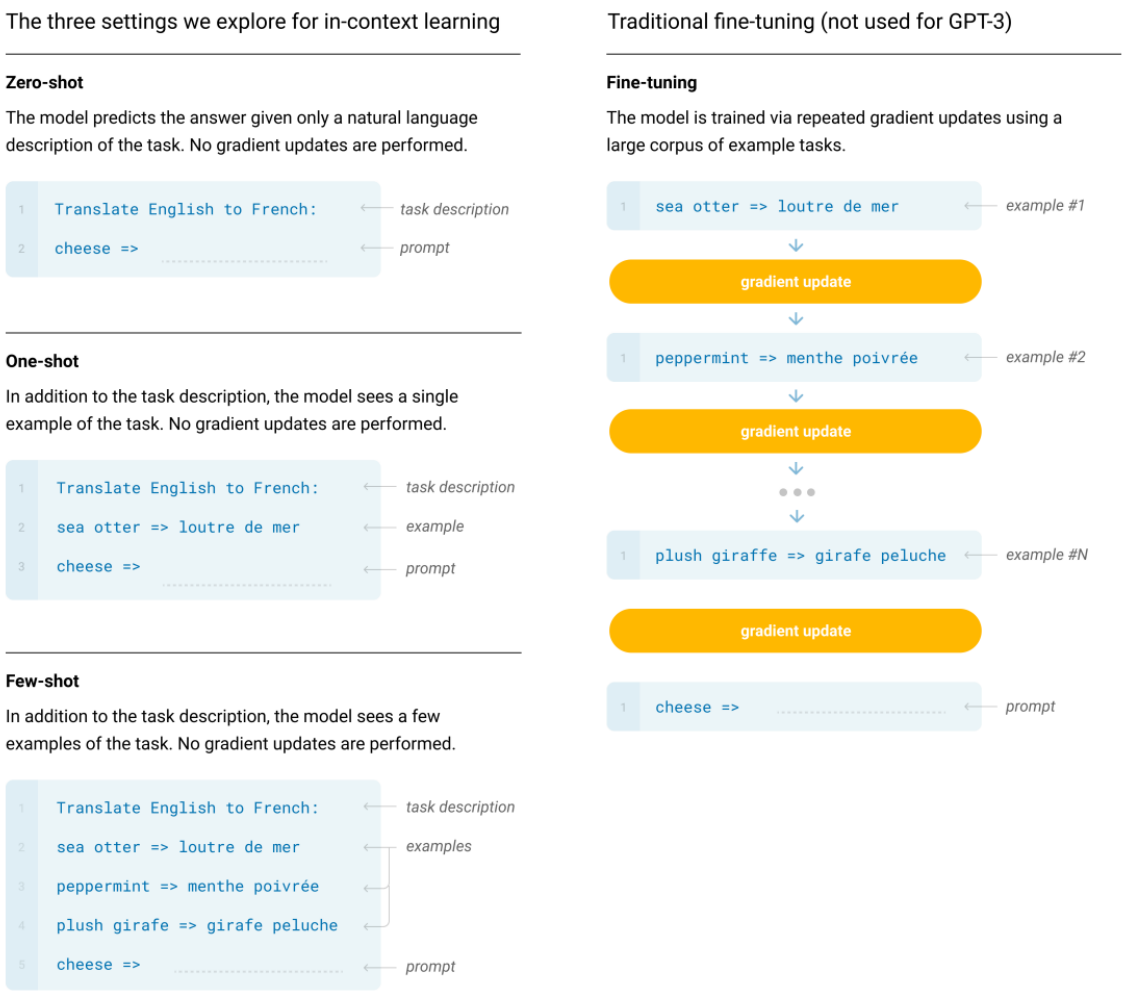
GPT-3: in-context learning

GPT-3 doesn't have a separate fine-tuning stage.

Instead: you ask the model to do a task with natural language text.

Further: to get better responses, you can "teach" GPT-3 what you want by providing examples within the decoder input itself ("in-context learning")

Zero shot vs one/few shot learning.



GPT-3: model scaling

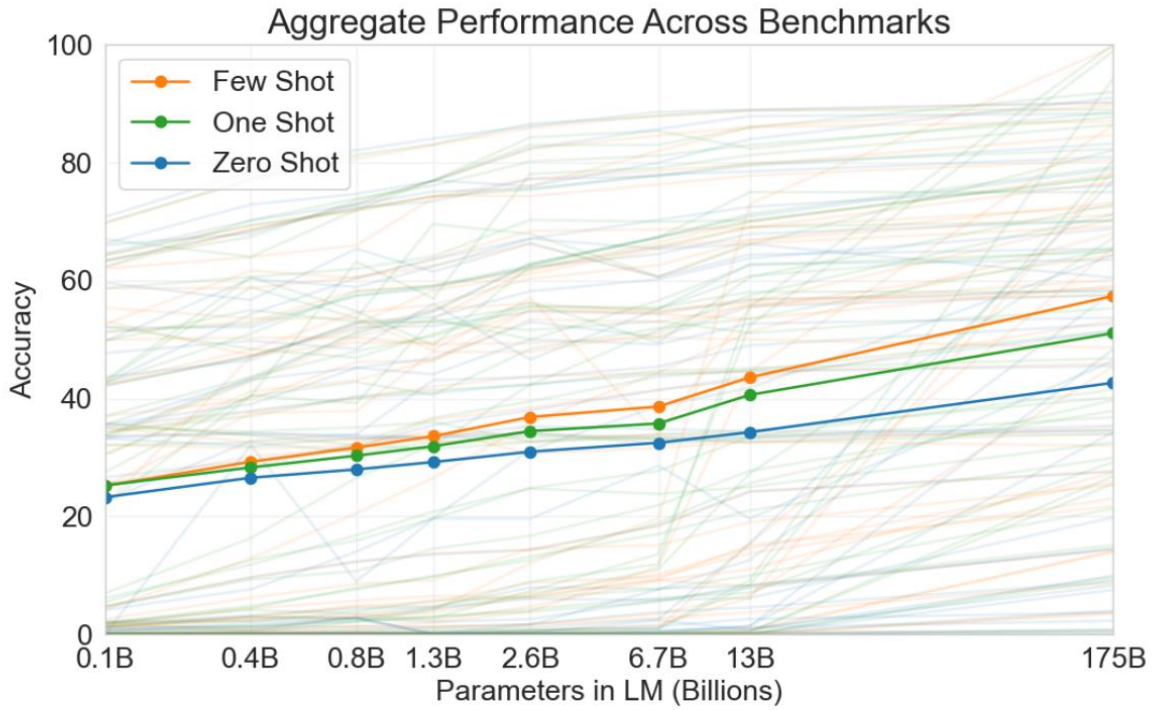


Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis on SuperGLUE, a standard NLP benchmark suite.

LLM: Scaling Laws

"Scaling Laws for Neural Language Models" (Jan 2020)

Main findings:

- Test loss scales as a power-law with: model size, dataset size, and training compute
- Larger models are more sample efficient
- "Optimally compute-efficient training": train the largest model you can on "relatively modest" amount of training data, and stop significantly before convergence.

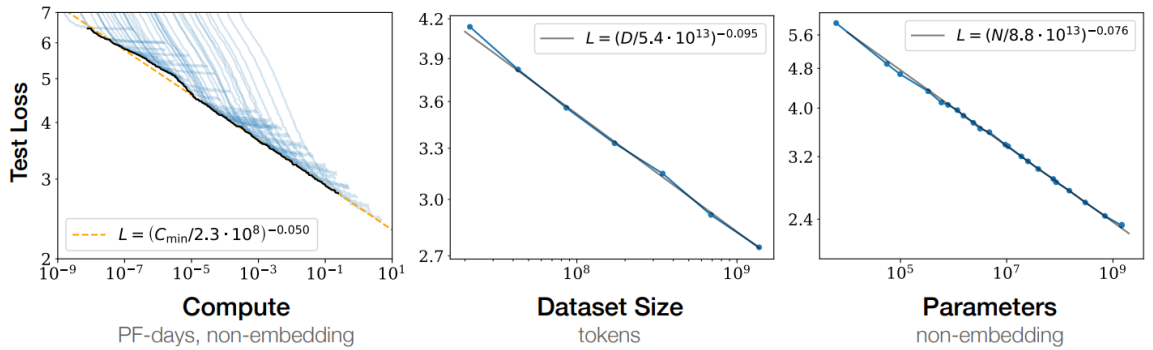


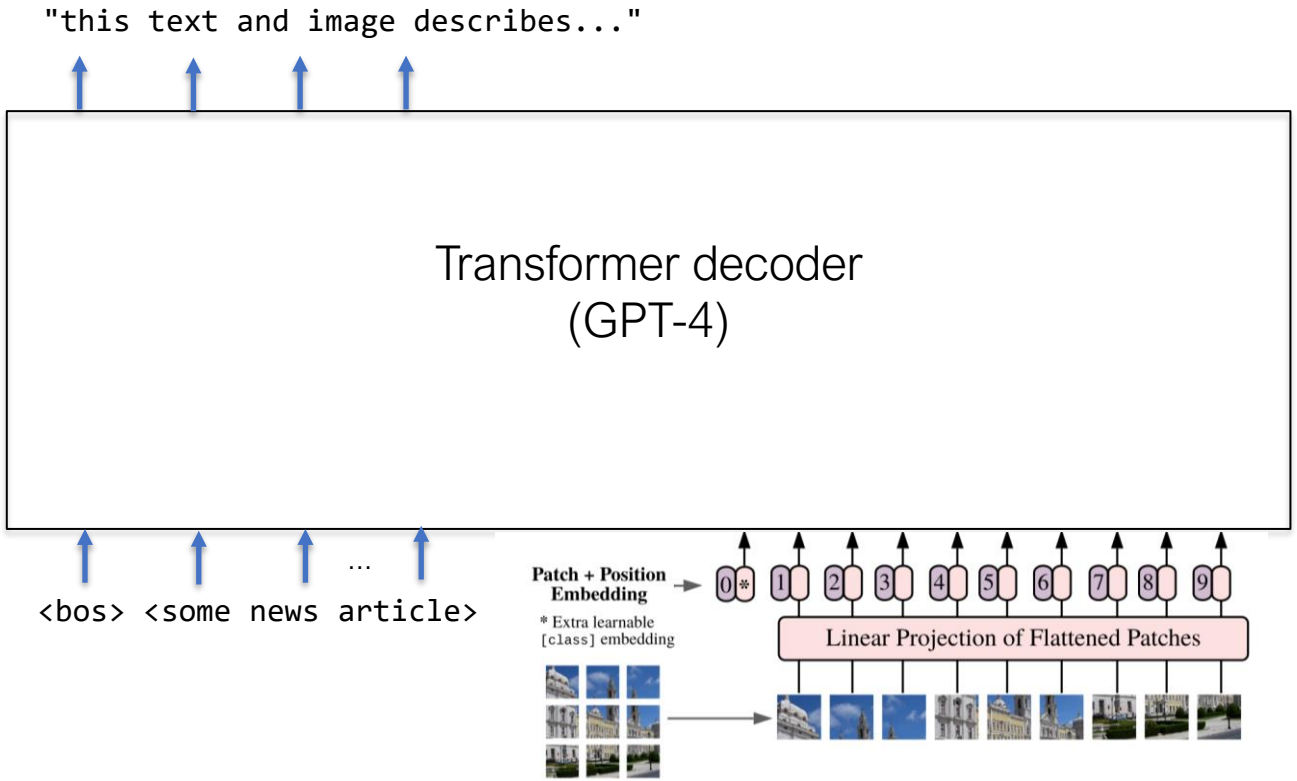
Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when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

GPT-4 (2023)

"GPT-4 Technical Report" (2023).

Main idea: by further scaling up the model, and adding multimodal (text+image) capabilities, GPT-4 can achieve human-level performance on "various professional and academic benchmarks, including passing a simulated bar exam with a score around the top 10% of test takers."

Model size: 1.7T parameters (est)



Multimodal model

ChatGPT: GPT + RLHF

OpenAI released ChatGPT in November 30th, 2022 (GPT-3.5).

There is a finetuning round to encourage the GPT model to generate outputs aligned with the ChatGPT product's goals, such as: truthfulness, helpfulness, etc.

This finetuning is known as **"Reinforcement Learning from Human Feedback" (RLHF)**.

"Aligning language models to follow instructions" (Jan 2022).

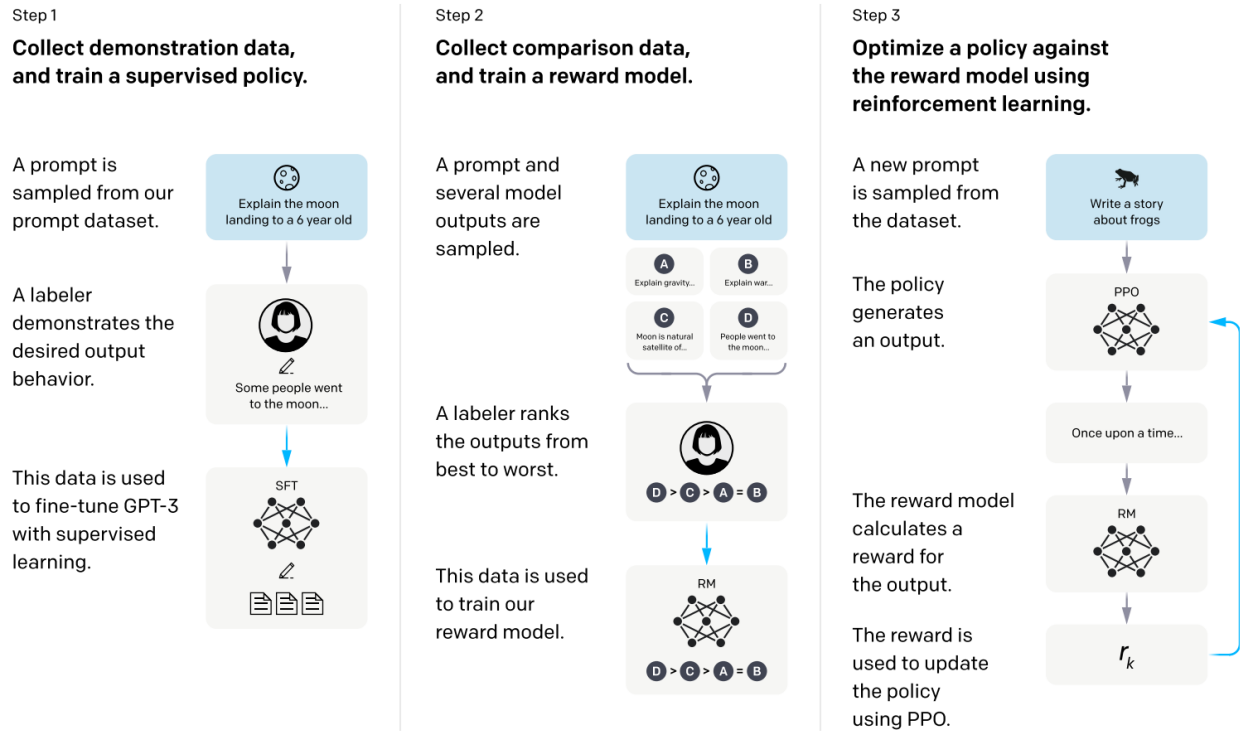


Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers. See Section 3 for more details on our method.

"Human in the loop"

GPT: progression

Name	Release date	Model size	Context window (tokens)	Pretraining dataset	In-context learning?	Multimodal?
GPT-1	June 11, 2018	117M	512	BookCorpus	No	No
GPT-2	February 14, 2019 - November 5, 2019 (full)	1.5B	1024	WebText	Yes	No
GPT-3	May 29, 2020	175B	2048 - 16k	CommonCrawl, WebText2, and more...	Yes	No
GPT-4	March 14, 2023	1.8T	8192 - 128k	?	Yes	Yes: text, image. 4o: audio
GPT-5	August 7, 2025	Rumor: 860B - 50T (MoE)	400k	?	Yes	Yes: text, image.

LLM/VLM adoption

As of 2026, core Large Language Models and Vision Language Models (aka LLM, VLM) continue to improve due to modeling, dataset, and compute improvements.

On the application side: **rapid growth** in experimentation and adoption.

LLMs/VLMs are used for many real-world use cases, such as: coding, writing e-mails, planning trips, creating app prototypes, etc.

"Prompt engineering": the art of effectively instructing an LLM/VLM to do the thing you want. Sort of like writing a project doc for an intern/junior dev to follow.

"Agentic AI": give the LLM access to systems to perform tasks (ex: write+execute code, orchestrate calls to other LLMs, order pizza, etc). Model Context Protocol (MCP), aka tool calling".

Impact of Generative AI on Society?

Opinions are extremely mixed on the impact of GenAI on humanity and society.

Examples: AI overuse leading to worse learning outcomes in school.

"Offshoring intelligence/thinking to GenAI companies" (!)

GenAI threatening the jobs of artists/actors of your favorite shows/movies.

Or: will the promise of dramatically increased productivity and capabilities lead to a net benefit to all?

What do you think?