Data 188 Introduction to Deep Learning
Spring 2026 Eric Kim

Discussion 12

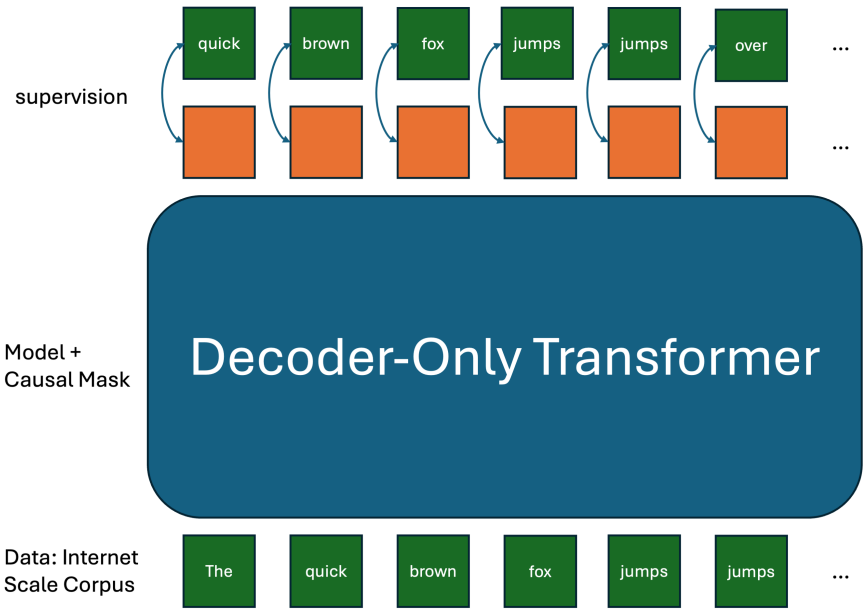
This discussion covers large-scale NLP pretraining, GPT-style language modeling, instruction/chat fine-tuning, training at scale, context-window limits, and societal impacts of LLMs.

1. GPT Pretraining

During pretraining, what is the primary task that GPT models train on? What is the training dataset like?

Solution: GPT models pretrain on the "predict next token" task, which belongs to the broad **language modeling** category. Specifically, given a sequence of previous tokens, the model is trained to predict the next token. Empirically, we split the training corpus into sequences of tokens, and for each sequence we apply causal attention mask to ensure that when predicting the i-th token, the model can only attend to the previous $i-1$ tokens. Thus, with a single forward and backward pass, the model can compute next-token predictions/losses for all positions in the sequence.

The training dataset is a large-scale human-written text corpus, typically scraped from the internet and mixed with other text sources. The exact dataset varies based on the GPT version, but generally speaking, later GPT versions use larger and noisier text datasets. Note that high quality data is perhaps one of the most important parts in the training recipe.



2. GPT Finetuning

Recall that many chatbot companies, such as OpenAI for ChatGPT, perform a second-stage fine-tuning step after pretraining (such as SFT or RLHF). What is this fine-tuning step? What could happen if they did not do this step and released the pretrained model to the public?

Solution: Internet scale pretraining gives the base model a lot of foundational knowledge through language modelling, but it is not sufficient to make the model a shippable customer-facing product. For example, it requires a lot of careful prompting to get the pretrained-only base model to produce the desired response, and the generated response might be harmful or not aligned with the product goals or safety standards.

Usually, after pretraining, companies will perform second-stage fine-tuning (aka post-training) steps such as Supervised Fine-Tuning (SFT) or Reinforcement Learning (RL) to encourage the LLM to generate outputs that align with the product goals, such as providing helpful and truthful responses while minimizing harm from hate speech, misinformation, toxicity, and other unsafe behavior. SFT requires annotated dataset of input-output pairs, for example inputs are user queries and outputs are human-written responses. Then, the model is fine-tuned on this dataset using supervised learning. Usually SFT greatly provides the model with instruction following capabilities and thus makes it much easier to prompt.

RL is a more complex training recipe (take CS 185 if you are interested in learning more!), but the high-level idea is to use feedback to guide the model to generate better outputs. One popular RL recipe is Reinforcement Learning with Human Feedback (RLHF). You might have seen ChatGPT asking you to choose the better response between two model-generated responses. This is an example of collecting human feedback for RLHF. RLHF algorithms can then use human preference data to further improve the model's alignment with human preferences and values.

Another usage of RL is for coding or math problems, where the reward (preference) signal can be automatically computed by running the generated code or checking the generated math answer, without needing human raters. Current State-of-the-Art models can achieve superhuman performances on many coding and math benchmarks.

Without the fine-tuning steps, the pretrained LLM would mostly parrot the text that is most likely according to its pretraining corpus. Since the pretraining corpus is roughly "the entire internet," or at least a sizeable chunk of it, there is a lot of content that companies generally do not want their LLM product to produce, such as hate, misinformation, or toxic content.

For a cautionary tale on the risks and challenges of releasing chatbots to the public, see Microsoft's Tay chatbot, released on March 23, 2016: `https://en.wikipedia.org/wiki/Tay_(chatbot)`

3. GPT Training at Scale

(a) Recall that GPT models are extremely large. For instance, GPT-4 is rumored to have 1.8T model parameters, which, if stored in float16, would require 3600 GB of GPU memory just to store the model weights.

Suppose we want to train GPT-4, and want to use the Adam optimizer. How much additional GPU memory would we need for Adam? Only consider additional optimizer state, ignore intermediate activations. Assume a naive, unoptimized implementation for Adam.

Solution: Let N be the number of model parameters.

Adam needs to store two optimizer-state tensors for each model parameter: the first moment estimate and the second moment estimate. This adds an additional $2N$ memory requirement in the optimizer state.

During the backward pass, we also need to keep track of the gradients themselves. This adds an additional N term for any optimizer, including Adam and SGD.

Thus, beyond the model weights themselves (N), Adam requires approximately $3N$ additional float16 values in this naive implementation:

$$\frac{3 \cdot N \cdot 2 \text{ bytes}}{10^9} = \frac{3 \cdot 1.8 \cdot 10^{12} \cdot 2}{10^9} = 10800 \text{ GB}.$$

In total, we would need approximately $3600 + 10800 = 14400$ GB of GPU memory to train GPT-4 with Adam in this naive implementation.

(b) As of 2026, it is unrealistic to have access to GPU machines that have more than tens of thousands of GB GPU memory on a single node. For example, AWS EC2 offers the p5en.48xlarge, which has 1128 GB GPU memory.

Suppose we only have access to nodes that each have 1128 GB GPU memory. How would we train a model like GPT-4?

Solution: Since the training memory footprint cannot fit on a single machine, we must use model sharding techniques such as Fully Sharded Data Parallel (FSDP), where model parameters, gradients, and optimizer state are split, or sharded, across multiple machines.

4. Article Summarization: Exceeding Context Windows

A common task for Large Language Models (LLMs) is to ask the LLM to generate a brief and accurate summary of an input document, such as a textbook chapter, conference paper, or lecture slides. This is often called article summarization.

Recall that there is a maximum context-window length. For example, GPT-4 has a limit of 128k tokens (and the newly released DeepSeek V4 supports 1M context window). Suppose we have an input text document that, after tokenization, exceeds the LLM's context-window maximum length. What would go wrong if we silently ignored this? What are some possible techniques to overcome this limitation?

Solution: If we ignored the issue, then the system would likely truncate the input sequence to the maximum context limit. This would lead to incomplete summaries, because the LLM would simply not see the entire input document.

One way to overcome this is via iterative summarization: ask the LLM to summarize subsets of the document, then perform a final summarization over those summaries. In other words, we summarize the summaries, reducing the total context needed at each LLM call.

On the other hand, expanding context window via better architectures and training recipes is an active area of research. One example is getting rid of the quadratic (on sequence length) attention bottleneck in the Transformer architecture, and some recent models have incorporated other efficient attention variants (e.g. sparse attention) or attention alternatives (e.g. State Space Models) to achieve longer context windows.

5. LLMs: Impacts on Society

As of 2026, it is undeniable that LLMs have made a significant impact on the world in nearly every industry, ranging from technical crafts such as software engineering and mathematics research to creative fields such as visual arts, design, music, and writing. At this point, the "cat is out of the bag"!

Spend some time reflecting on how this will impact our world and society, both in terms of your professional career and your own personal life. Are there dangers of over-relying on LLMs? Or do the pros outweigh the risks?

Solution: We do not have *The Answer* (sadly). Time will tell, check back in 5–10 years!

Contributors:

- Eric Kim, Zekai Wang.

