CPE 490 590: Machine Learning for Engineering Applications

18 Large Language Models

Rahul Bhadani
Electrical & Computer Engineering, The University of Alabama in Huntsville

Outline

1. Introduction to LLMs

2. What Goes on in Developing LLMs?

Introduction

Recall simple n-gram language models

- Assigns probabilities to sequences of words
- Generate text by sampling possible next words
- Is trained on counts computed from lots of text

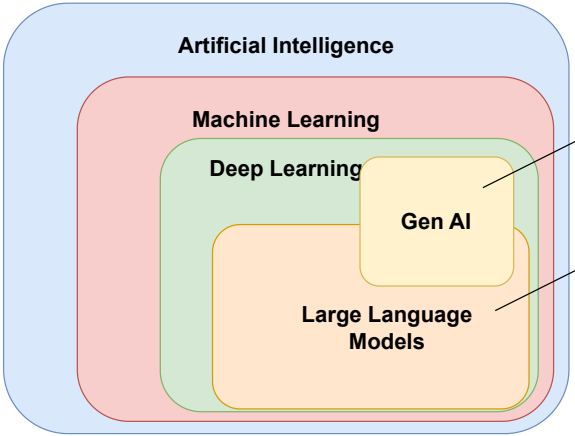
Large Language Models:

- Similar to n-gram language models
- But they are trained by learning to guess the next word

Introduction to LLMs

What is a Large Language Model?

- Based on deep neural network
- Broader capabilities compared to traditional Natural Language Processing (NLP)
- Is more than just a machine learning – software + UI
- The ”large” in large language model refers to both the model’s size in terms of parameters and the immense dataset on which it’s trained.
- Hundreds of billons of parameters
- Based on Transformer Architecture



GenAI uses some neural net architecture to create new content such as text, images or other form of media

Deep Neural Network parses and generates human-like texts

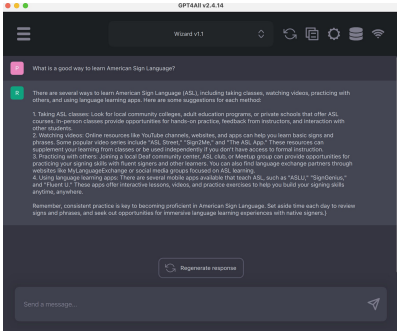
Using Large Language Models

1. Via proprietary or public services

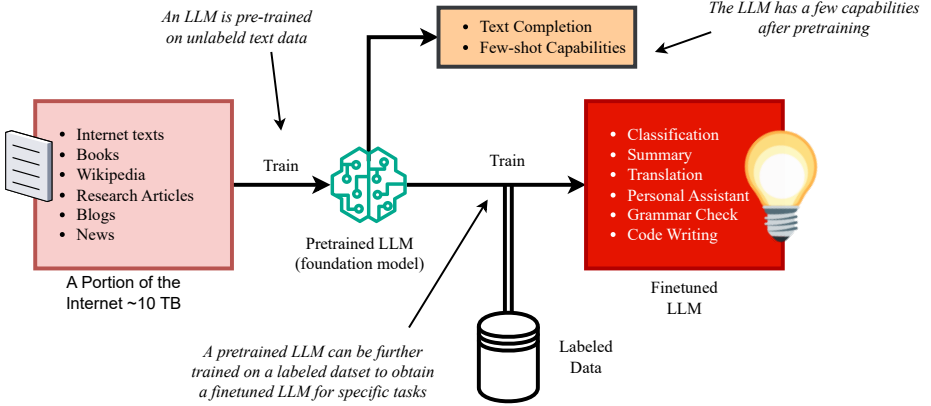
Using Large Language Models

2. Running a custom LLM locally
   https://github.com/Lightning-AI/litgpt
   https://github.com/ollama/ollama

3. Via API

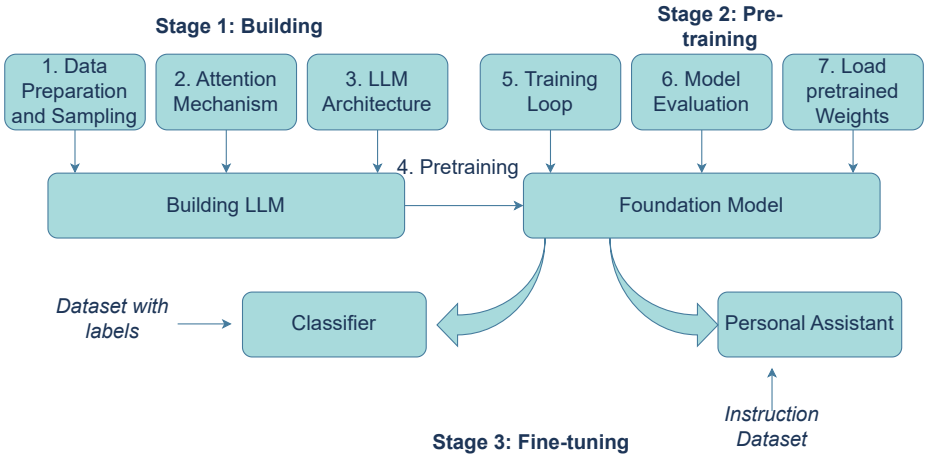


Stages of Building and Using LLMs



What Goes on in Developing LLMs?

Developing an LLM



Stage 1: Building

1. Data preparation and Sampling

The model is simply pretrained to predict the next $\rightarrow$ word.

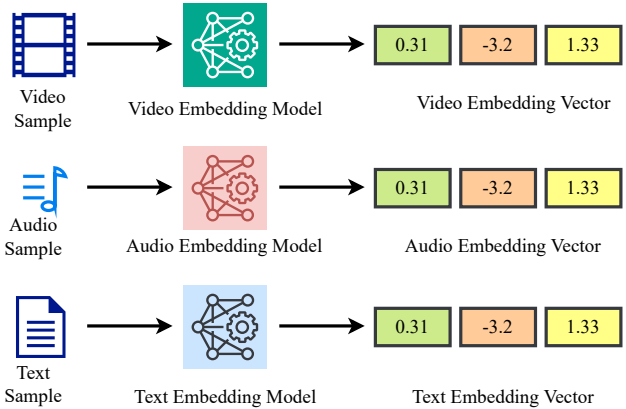
Essentially,

Next word (/token) prediction task

Data preparation and Sampling

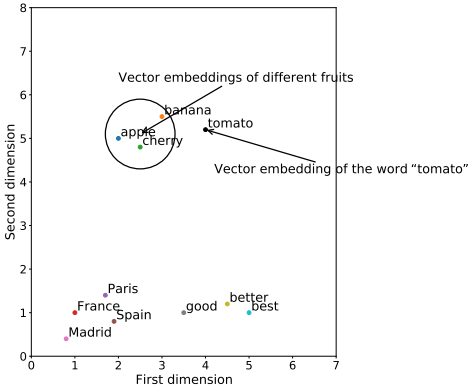
- Preparing text for large language model training
- Splitting text into word and subword tokens
- Byte pair encoding as a more advanced way of tokenizing text
- Sampling training examples with a sliding window approach
- Converting tokens into vectors that feed into a large language model

We looked at some way of tokenizing while discussing transformers earlier.



Embedding model converts raw input into vector representation.

Embedding



If embeddings have two dimensions!

Byte-pair Encoding

- The BPE was used to train LLMs such as GPT-2, GPT-3, and the original model used in ChatGPT.
- Python open-source library by OpenAI: `https://github.com/openai/tiktoken`

How does the BPE work?

Suppose we have a text corpus with the following four words: ab, bc, bcd and cde. We begin by calculating the frequencies of each byte (character). Initial vocabulary consists of all the unique characters in the corpus like {a, b, c, d, e}.

Step 1: Initialize the vocabulary

Vocabulary = a, b, c, d, e

How does the BPE work?

Step 2: Calculate the frequency of each byte or character in the text corpus

a: 1, b: 3, c:3, d:2 , e:1

How does the BPE work?

Step 3: Find the most frequent pair of two characters

Most frequent pair is bc with a frequency of 2.

How does the BPE work?

Step 4: Merge the pair to create a new subword unit. Merge bc to create a new subword unit bc.

Step 5: Update frequency counts of all the bytes or characters that contain the merged pair.
a: 1, b: 2, c:3, d:2 , e:1, bc: 1

Note: b's frequency decreases to 2 because the pair bc now represents both b and c together.
c's frequency remains 3 because it still appears in other words such as bcd and cde.

How does the BPE work?

Step 6: Add the new subword unit to the vocabulary Add bc to the vocabulary:
Vocabulary = a, b, c, d, e, bc

Repeat steps 3-6 until the desired vocabulary size is reached.

Step 7: Represent the text corpus using subword units Resulting vocabulary consists of the following subword units: {a, b, c, d, e, bc, cd, de,ab, bcd, cde}.

How does the BPE work?

Original text corpus can be represented using these subword units as follows:

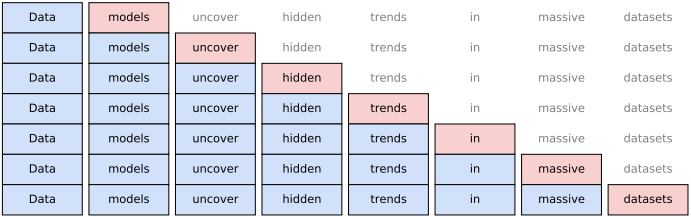
- ab $->$ a + b
- bc $->$ bc
- bcd $->$ bc + d
- cde $->$ c + de

BPE tokenizers break down unknown words into subwords and individual characters. This way, a BPE tokenizer can parse any word and doesn't need to replace unknown words with special tokens

Data Sampling with Sliding Window

- We partition the corpus into fixed-length input blocks, each serving as a subsample presented to the LLM; the model's training objective on each block is to predict the very next token that succeeds it.
- During training, any tokens beyond that target token are hidden (masked) so the model cannot see them, and note that the text is first passed through the tokenizer to convert it into tokens before being fed to the LLM.

Data Sampling with Sliding Window



In practice, we do batching – multiple training data are put into batches, we have to have same number of elements.

Token Embeddings and Word Encoding

We saw this in Transformer lecture where we choose a vocab size and a dimension and then use position encoding.

Coding Attention Mechanism

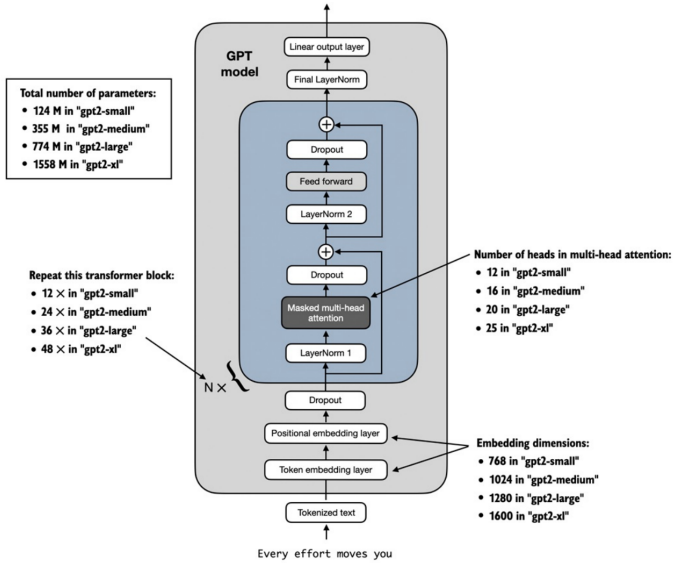
Implemented using Query-Key-Value triplet we already discussed in Transformer lecture.

- Transformer architecture is at the core of most LLMs.

LLM Architecure: Original GPT Architecture

Source: Sabastian Raschka

In Lllam 2 7B model, LayerNorm is replaced by Root-Mean Sqaure Norm



2. Pretraining on Unlabeled Data

- Llama 1: trained on 1.4 Trillion tokens.
- Dataset:
  - CommonCrawl: 3.3 TB
  - C4: 783 GB
  - Github: 328 GB
  - Wikipedia: 83 GB
  - Books: 85 GB
  - ArXiv: 92 GB
  - StackExchange: 78 GB

Pretraining on Unlabeled Data

Llama 2 was trained on 2 Trillion tokens. Llama 3 was trained 15 Trillion tokens.

- However, many companies don't share information about their training data due to copyrighted reasons, as many of the data are proprietary and yet they have been used for training LLMs.

Quantity vs Quality

For a good LLM, training on a quality data is more important than quantity for a given scale.

Pretraining Creates Foundation Model

- Standard deep learning training loop. Nothing different than what you already know.

Labels are the inputs shifted by +1

Consider the sentence: Astronomers Detect a Possible Signature of Life on a Distant Planet

Tensor contain the inputs:

```
tensor([["Astronomers", "Detect", "A", "Possible"],
    ["Signature", "of", "Life", "on"],
    ["a", "Distant", "Planet", "<PAD>"]])
```

Tensor contain the target:

```
tensor([["Detect", "A", "Possible", "Signature"],
    ["of", "Life", "on", "a"],
    ["Distant", "Planet", "<PAD>", "<PAD>"]])
```

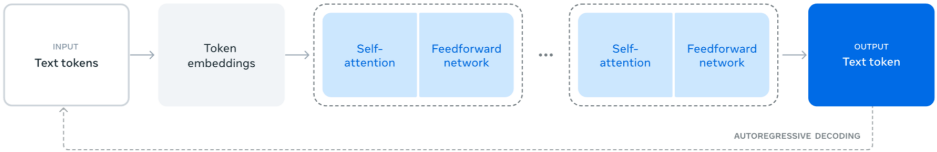
Training is Only 1-2 Epochs

Pre-training is done only for 1-2 Epochs.
Pretrained weights:

- Llama 4: 2T, 400B, 109B, 17B
- Llama 3: 8B, 70B, 405B
- Llama 2: 7B, 13B, 70B
- Deep Seek R1: 670B

Komatsuzaki, Aran. "One epoch is all you need." arXiv preprint arXiv:1906.06669 (2019).

LLama 3 Architecture and Training



LLM generates text one token at a time, each new token being conditioned on all previously generated tokens.

Grattafiori, Aaron, et al. "The Llama 3 Herd of Models." arXiv preprint arXiv:2407.21783 (2024).

Llama 3 Pretraining Data

Cut off : End of 2023.

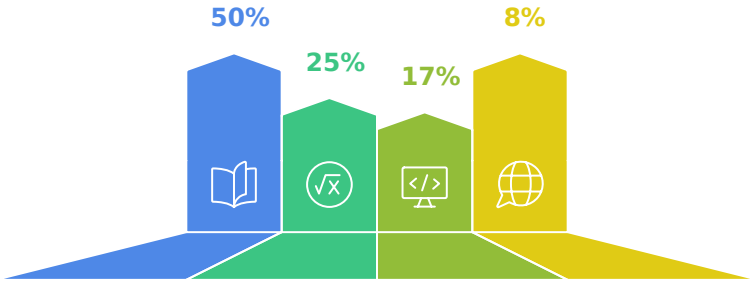
- De-duplication
- Data cleaning
- Remove Personally Identified Information and Adult Content

Various classifiers for high-quality and good data-mix

- Remove near duplicate documents (Ref: On the resemblance and containment of documents)
- Remove lines that consist of repeated content (Ref: Scaling language models: Methods, analysis & insights from training gopher)
- Remove dirty words (Ref: Exploring the limits of transfer learning with a unified text-to-text transformer)
- Fast classifiers (Ref: Bag of tricks for efficient text classification)
- Code and Math classifiers (Ref: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter)

Llama Datamix

Composition of Final Data Mix for Language Model



General Knowledge

Broad range of common information

Mathematical and Reasoning

Logical and analytical content

Code

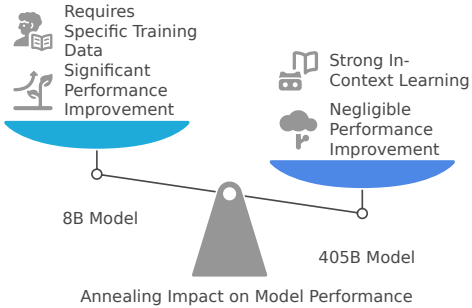
Programming and software-related data

Multilingual

Diverse linguistic content

Annealing Data for Pre-training

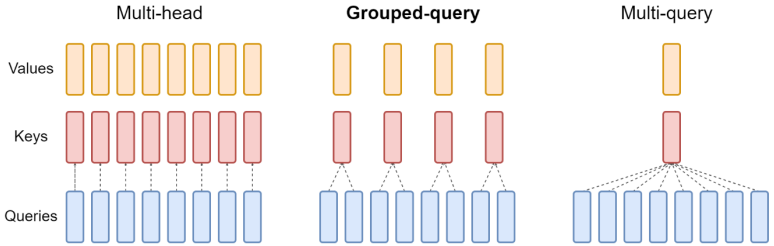
Term comes from Simulated Annealing: Choose a high-quality subset of your pre-training corpus that we introduce in a final, focused phase of pre-training and gradually "freeze in" the best configurations.



Model Architecture for LLama

- Standard transformer model with attention
- Uses grouped query attention with key-value caching
- Use attention mask
- Vocabulary size of 128K with tiktok (use BPE)
- Rotary Positional Embeddings (RoPE): base frequency hyperparameter to 50000 (Ref: RoFormer: Enhanced Transformer with Rotary Position Embedding)

Multi Query-key Value



Grouped-query attention shares single key and value heads for each group of query heads, interpolating between multi-head and multi-query attention.

GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints, Google Research

Scaling Laws

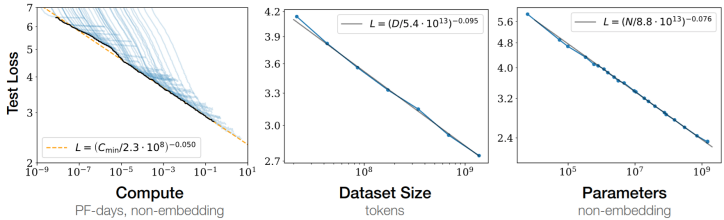


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when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

Performance improves predictably as long as we scale up number of model parameters and the dataset size in tandem, but enters a regime of diminishing returns if either N or D is held fixed while the other increases.

Kaplan, Jared, et al. "Scaling laws for neural language models." arXiv preprint arXiv:2001.08361 (2020).

LLama uses Scaling Laws for the optimal model size

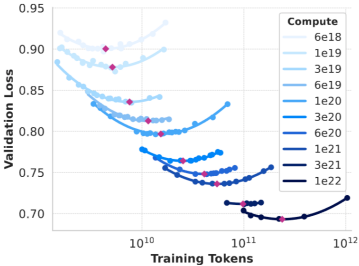


Figure 2 Scaling law IsoFLOPs curves between 6×10^{18} and 10^{22} FLOPs. The loss is the negative log-likelihood on a held-out validation set. We approximate measurements at each compute scale using a second degree polynomial.

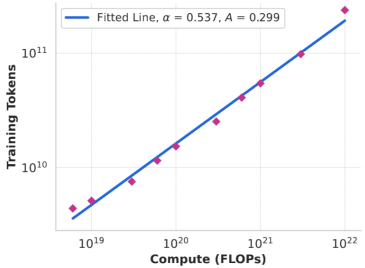


Figure 3 Number of training tokens in identified compute-optimal models as a function of pre-training compute budget. We include the fitted scaling-law prediction as well. The compute-optimal models correspond to the parabola minimums in Figure 2.

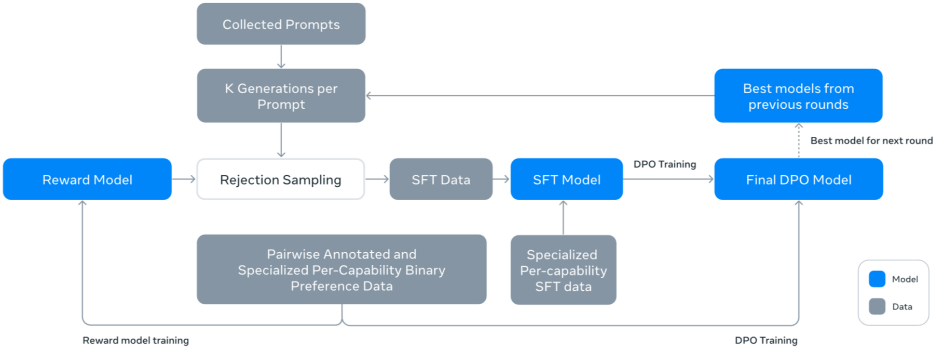
Infrastructure for LLama Training

- Meta's AI Research SuperCluster
- 16K H100 GPUs, 700W Thermal Design Power (TDP) with 80GB High-Bandwidth Memory, Gen 3 (HBM3) $\approx$ 600 millions USD.
- Grand Teton AI Server Platform
- Training Job Scheduling using MAST
- Meta's distributed file system for storage, 40 PB of storage out of 7,500 servers, throughput of 2 TB/s and a peak throughput of 7 TB/s.

Llama 3 Pretraining

- Llama 3 405B: AdamW Optimizer, peak learning rate 8×10^{-5}
- Linear warm up of 8,000 steps, cosine learning rate schedule decay, up to 8×10^{-7} over 1,200,000 steps.
- Initial batch size of 4M tokens
- Sequences of length 4,096
- Final stage: train on long sequences to support context windows of up to 128K tokens
- Do not train on long sequences earlier because the compute in self-attention layers grows quadratically in the sequence length.

Post-training/Finetuning for LLama 3



Ref: The Llama 3 Herd of Models

Fine-tuning Pre-trained Model

- Finetune pre-trained checkpoints with supervised finetuning
- Align the checkpoints with Direct Preference Optimization

Human-intervention for Supervised Finetuning

- Chat dialogue protocol for human-AI interaction
- A reward model (RM) covering different capabilities on top of the pre-trained checkpoint.

Reinforcement Learning with Human Feedback (no direct huma-labeling required but LLM finetune itself.)

Reinforcement Learning with Human Feedback

ChatGPT o4-mini-high

Thought for a few seconds >

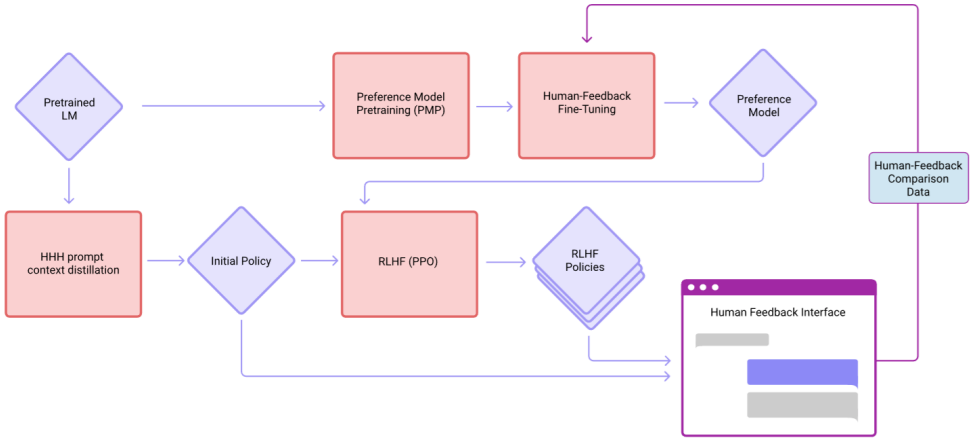
Sure! Here are two design directions for a beautiful log cabin image:

Option A:
A cozy, photorealistic log cabin nestled on the edge of a tranquil mountain lake at sunset. Smoke curls from the stone chimney, warm lights glow from the windows, and the surrounding pine trees reflect softly in the water.

Option B:
A rustic wooden cabin deep in a snowy pine forest under a star-filled night sky. The cabin's windows are aglow with warm interior light, and the Northern Lights dance overhead, casting subtle green and purple hues on the snow.

Which style would you prefer for your log cabin image?

Reinforcement Learning with Human Feedback



Ref: Bai, Yuntao, et al. "Training a helpful and harmless assistant with reinforcement learning from human feedback." arXiv preprint arXiv:2204.05862 (2022).

Direct Preference Optimization for Finetuning

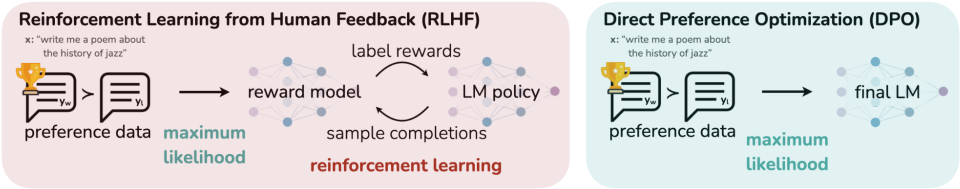
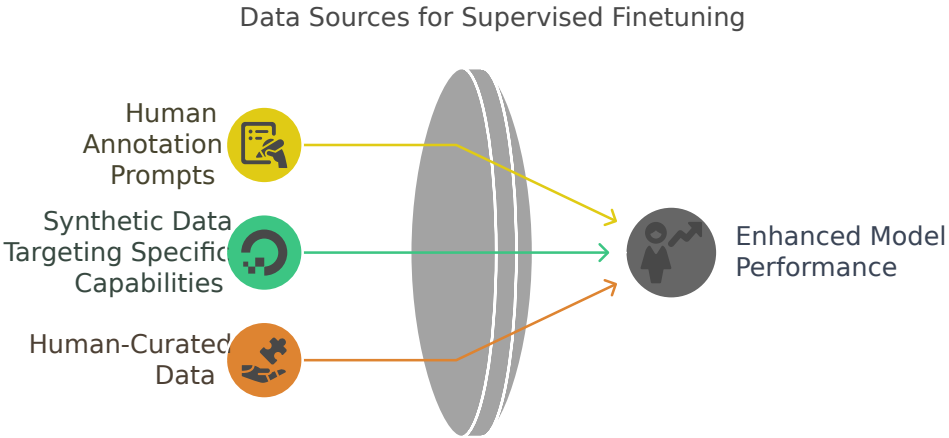


Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, without an explicit reward function or RL.

Ref: Rafailov, Rafael, et al. "Direct preference optimization: Your language model is secretly a reward model." Advances in Neural Information Processing Systems 36 (2023): 53728-53741.

For LLama 3, DPO required less compute for large-scale models and performed better, especially on instruction following benchmarks like IFEval compared to RLHF with PPO.

Supervised Finetuning Data



Benchmarking

Reading Comprehension	SQuAD V2 (Rajpurkar et al., 2018), QuaC (Choi et al., 2018), RACE (Lai et al., 2017),
Code	HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021),
Commonsense reasoning/understanding	CommonSenseQA (Talmor et al., 2019), PiQA (Bisk et al., 2020), SiQA (Sap et al., 2019), OpenBookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021)
Math, reasoning, and problem solving	GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), ARC Challenge (Clark et al., 2018), DROP (Dua et al., 2019), WorldSense (Benchekroun et al., 2023)
Adversarial	Adv SQuAD (Jia and Liang, 2017), Dynabench SQuAD (Kiela et al., 2021), GSM-Plus (Li et al., 2024c) PAWS (Zhang et al., 2019)
Long context	QuALITY (Pang et al., 2022), many-shot GSM8K (An et al., 2023a)
Aggregate	MMLU (Hendrycks et al., 2021a), MMLU-Pro (Wang et al., 2024b), AGIEval (Zhong et al., 2023), BIG-Bench Hard (Suzgun et al., 2023)

Table 8 Pre-training benchmarks by category. Overview of all benchmarks we use to evaluate pre-trained Llama 3 models, grouped by capability category.

Performance Evaluation

	Commonsense Understanding				
	CommonSenseQA	PiQA	SiQA	OpenBookQA	Winogrande
Llama 3 8B	**75.0** ±2.5	81.0 ±1.8	49.5 ±2.2	45.0 ±4.4	75.7 ±2.0
Mistral 7B	71.2 ±2.6	**83.0** ±1.7	48.2 ±2.2	47.8 ±4.4	**78.1** ±1.9
Gemma 7B	74.4 ±2.5	81.5 ±1.8	**51.8** ±2.2	**52.8** ±4.4	74.7 ±2.0
Llama 3 70B	**84.1** ±2.1	83.8 ±1.7	**52.2** ±2.2	47.6 ±4.4	83.5 ±1.7
Mixtral 8×22B	82.4 ±2.2	**85.5** ±1.6	51.6 ±2.2	**50.8** ±4.4	**84.7** ±1.7
Llama 3 405B	**85.8** ±2.0	**85.6** ±1.6	**53.7** ±2.2	**49.2** ±4.4	82.2 ±1.8
GPT-4	–	–	–	–	87.5 ±1.5
Nemotron 4 340B	–	–	–	–	**89.5** ±1.4

Table 11 Pre-trained model performance on commonsense understanding tasks. Results include 95% confidence intervals.

	Math and Reasoning				
	GSM8K	MATH	ARC-C	DROP	WorldSense
Llama 3 8B	**57.2** ±2.7	20.3 ±1.1	**79.7** ±2.3	**59.5** ±1.0	45.5 ±0.3
Mistral 7B	52.5 ±2.7	13.1 ±0.9	78.2 ±2.4	53.0 ±1.0	44.9 ±0.3
Gemma 7B	46.4 ±2.7	**24.3** ±1.2	78.6 ±2.4	56.3 ±1.0	**46.0** ±0.3
Llama 3 70B	83.7 ±2.0	41.4 ±1.4	**92.9** ±1.5	**79.6** ±0.8	**61.1** ±0.3
Mixtral 8×22B	**88.4** ±1.7	**41.8** ±1.4	91.9 ±1.6	77.5 ±0.8	51.5 ±0.3
Llama 3 405B	89.0 ±1.7	**53.8** ±1.4	96.1 ±1.1	**84.8** ±0.7	**63.7** ±0.3
GPT-4	**92.0** ±1.5	–	**96.3** ±1.1	80.9 ±0.8	–
Nemotron 4 340B	–	–	94.3 ±1.3	–	–
Gemini Ultra	88.9◇ ±1.7	53.2±1.4	–	82.4△ ±0.8	–

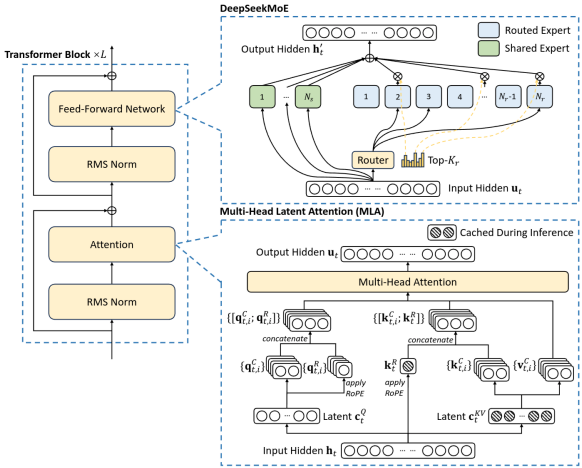
Table 12 Pre-trained model performance on math and reasoning tasks. Results include 95% confidence intervals. ◇11-shot. △Variable shot.

Deepseek V2 with Mixture of Experts

- 236B parameters
- Training Data: 8.1T Tokens
- context length of 128K tokens
- Uses Key-value Caching (https://v.douyin.com/XnneUnPxhio/)
- Innovation: Multi-head Latent Attention
- Supervised Finetuning

Deepseek V2 General Architecture

Multi-head Latent Attention (MLA) ensures efficient inference by significantly reducing the KV cache for generation, and DeepSeekMoE (Mixture of Experts) enables training strong models at an economical cost through the sparse architecture.



Multi-head Latent Attention

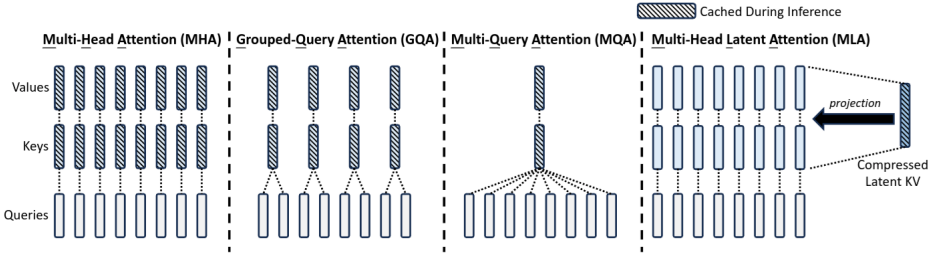


Figure 3 | Simplified illustration of Multi-Head Attention (MHA), Grouped-Query Attention (GQA), Multi-Query Attention (MQA), and Multi-head Latent Attention (MLA). Through jointly compressing the keys and values into a latent vector, MLA significantly reduces the KV cache during inference.

Deepseek Mixture of Experts

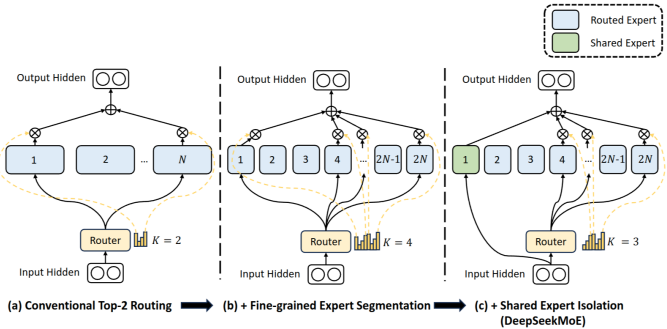


Figure 2 | Illustration of DeepSeekMoE. Subfigure (a) showcases an MoE layer with the conventional top-2 routing strategy. Subfigure (b) illustrates the fine-grained expert segmentation strategy. Subsequently, subfigure (c) demonstrates the integration of the shared expert isolation strategy, constituting the complete DeepSeekMoE architecture. It is noteworthy that across these three architectures, the number of expert parameters and computational costs remain constant.

Ref: Dai, Damai, et al. "DeepseekMoE: Towards ultimate expert specialization in mixture-of-experts language models." arXiv preprint arXiv:2401.06066 (2024).

Some Other Topics

- LLM Jailbreaking
- Security and Guardrails
- Quantization
- Mixture of Experts
- Retrieval Augmented Generation
- Model Context Protocol
- Agentic AI
- Large language diffusion models

References

- Developing an LLM: Building, Training, Finetuning: https://www.youtube.com/watch?v=kPGTx4wcm_w
- https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp/
- https://huggingface.co/learn/llm-course/en/chapter6/5
- The Llama 3 Herd of Models
- Scaling Laws for Neural Language Models
- DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model
- DeepSeek-V3 Technical Report
- Guo, Daya, et al. "Deepseek-R1: Incentivizing reasoning capability in llms via reinforcement learning," arXiv preprint arXiv:2501.12948 (2025).
- Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback
- Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks
- https://ai.meta.com/blog/llama-4-multimodal-intelligence/

Additional Resources to Explore

- YouTube Playlist: RAG from Scratch https://www.youtube.com/playlist?list=PLfaIDFEXuae2LXbO1_PKyVJiQ23ZztA0x
- YouTube Playlist: Physics-informed Machine Learning
  https://www.youtube.com/playlist?list=PLJ6U7kzSli-3h6iMQ7Ww_Madv_T1eW75E,
  https://www.youtube.com/watch?v=JoFW2uSd3Uo&list=PLMrJAkhIeNNQ0BaKuBKY43k4xMo6NSbBa&index=1
- Stanford CS336: CS336: Language Modeling from Scratch https://stanford-cs336.github.io/spring2024/
- MIT MAS S60: How to AI Almost Anything: https://mit-mi.github.io/how2ai-course/spring2025/schedule/

The End