Andrej Karpathy: Deconstructing LLMs - Part 1

Frank Richter

f.richter@em.uni-frankfurt.de

Goethe Universität Frankfurt

Tools for Natural Language Processing

Karpathy's Core Question

What are we talking to?

LLMs under the hood: Token autocomplete on steroids

An LLM is the product of a two-phase pipeline:

1. **Pretraining** → *the base model*
2. **Post-Training** → *the assistant model*

Karpathy's Core Question

What are we talking to?

LLMs under the hood: Token autocomplete on steroids

An LLM is the product of a two-phase pipeline:

1. Pretraining $\rightarrow$ *the base model*

2. Post-Training $\rightarrow$ *the assistant model*

Karpathy's Core Question

What are we talking to?

LLMs under the hood: Token autocomplete on steroids

An LLM is the product of a two-phase pipeline:

1. **Pretraining** $\rightarrow$ *the base model*
2. **Post-Training** $\rightarrow$ *the assistant model*

Phase 1: Pretraining (The Ingredients)

1. The data: The Internet

- Billions of documents (FineWeb dataset: 15 trillion tokens)
- Raw, unlabeled text (UTF-8)

2. The dictionary: Tokenization

- Turn text into numbers (tokens)
- Algorithm: Byte-Pair Encoding (BPE)
- Learns common sub-words
- Modern vocabularies have around 100k tokens (`cl100k_base`)

Phase 1: Pretraining (The Ingredients)

1. The data: The Internet

- Billions of documents (FineWeb dataset: 15 trillion tokens)
- Raw, unlabeled text (UTF-8)

2. The dictionary: Tokenization

- Turn text into numbers (tokens)
- Algorithm: Byte-Pair Encoding (BPE)
- Learns common sub-words
- Modern vocabularies have around 100k tokens (`cl100k_base`)

Phase 1: Pretraining (The Recipe)

The task: Predict the next token.

The cook: A neural network (transformer architecture)

The process (training loop):

1. Give the model a context window of tokens from the data (up to 1024 tokens for GPT-2; more recently: up to 8000 tokens).
2. Model predicts a probability distribution for the *next* token.
3. Compare prediction to the *actual* next token.
4. Calculate *loss* (measure of error).
5. Adjust billions of parameters (weights) to reduce the loss.
6. Repeat a very large number of times (can be months of compute).

Phase 1: Pretraining (The Recipe)

The task: Predict the next token.

The cook: A neural network (transformer architecture)

The process (training loop):

1. Give the model a context window of tokens from the data (up to 1024 tokens for GPT-2; more recently: up to 8000 tokens).
2. Model predicts a probability distribution for the *next* token.
3. Compare prediction to the *actual* next token.
4. Calculate *loss* (measure of error).
5. Adjust billions of parameters (weights) to reduce the loss.
6. Repeat a very large number of times (can be months of compute).

Phase 1: Result: Base Model (The Dish)

- **What it is:** A file of parameters (GPT-2: 1.5 billion numbers)
- **What it does:** Simulates Internet text, acts as a glorified autocomplete or lossy compression of the web.
- **How we use it (Inference):**
  - Give it a prompt.
  - It samples a next token from its learned distribution.
  - Append that token to the prompt, and repeat.
- It's a remix of its training documents.
- Can be prompted for tasks (few-shot / in-context learning), but it is not yet an assistant.

The Problem with the Base Model

A base model just wants to **complete text.** It doesn't want to **answer questions.**

Example Interaction

User: `What is the capital of France?`

Base Model (might respond): `What is the capital of Spain? What is the capital of Germany? What is the...`

(It's completing a quiz it saw on the internet, not answering the question.)

Phase 2: Post-Training (Refining the Dish): Assistant

Teach the model a new style of conversation.

Method: Supervised Finetuning (SFT)

- **The data:** High-quality, curated Q&A pairs
  - Created by human labelers (or now, other LLMs)
  - Example: OpenAssistant dataset
- **The format:** Special tokens for conversational structure
  - <|user|> What is the capital of France?
  - <|assistant|> The capital of France is Paris.
- **The process:** Continue training the base model on these examples.

Key Takeaways

1. **A two-step process:**
   - **Pretraining (99% compute):** Learns world knowledge and language by simulating internet text.
     - Captures statistical echoes of human patterns (knowledge, language, biases)—a *human ghost* distillation, per Karpathy's later analogy.
   - **Finetuning (1% compute):** Learns style and behavior by simulating human labelers.
2. **It's a simulator:** The model isn't thinking; it's giving a statistically likely response based on its training.
   - The base model simulates an internet document.
   - The assistant simulates a helpful human labeler.

Key Takeaways

1. **A two-step process:**
   - **Pretraining (99% compute):** Learns world knowledge and language by simulating internet text.
     - Captures statistical echoes of human patterns (knowledge, language, biases)—a *human ghost* distillation, per Karpathy's later analogy.
   - **Finetuning (1% compute):** Learns style and behavior by simulating human labelers.
2. **It's a simulator:** The model isn't thinking; it's giving a statistically likely response based on its training.
   - The base model simulates an internet document.
   - The assistant simulates a helpful human labeler.

Tool Use for the Present Slide Deck

The slides were produced in a multi-step process involving Gemini and Grok for the following purposes:

- checking the initial presentation draft against Karpathy's video and its place on the seminar syllabus (Gemini 2.5 PRO)
- typesetting in Latex, multiple rounds of fact checking and stylistic changes (Grok 4 Expert)