

The Mathematics of Large Language Models (LLMs)

Generated with ChatGPT

May 25, 2025

Abstract

This document presents a detailed mathematical explanation of how large language models (LLMs), such as GPT-4, function. It covers foundational concepts, training processes, statistical methods, and inference mechanisms with supporting equations and illustrations.

Contents

1	History and Development of LLMs	3
1.1	Early Foundations	3
1.2	Neural Language Models	3
1.3	The Transformer Era	3
1.4	The Transformer Era	3
1.5	Scaling Laws and Pretrained LMs	3
1.6	Trends in Development	3
2	Overview of Actively Maintained LLMs	3
3	Overview of LLMs	4
4	Mathematical Foundation of Transformers	4
4.1	Embedding	4
4.2	Positional Encoding	4
4.3	Self-Attention	5
4.4	Multi-Head Attention	5
4.5	Feedforward Network	5
4.6	Normalization and Residuals	5
5	Training Objectives and Statistical Methods	5
5.1	Maximum Likelihood Estimation	5
5.2	Cross-Entropy Loss	5
6	Optimization and Gradient Descent	6
6.1	SGD and Adam Optimizer	6
7	Inference and Generation Process	6
7.1	Autoregressive Sampling	6
7.2	Sampling Techniques	6
7.3	Temperature Scaling	6
7.4	Perplexity as an Evaluation Metric	6

8	Contextual Understanding and Deep Thinking	6
8.1	Semantic Layering	7
8.2	Compositional Reasoning	7
8.3	Emergent Memory	7
8.4	Capabilities	7
9	Illustrative Example	7
10	Visual Diagram	8
11	Conclusion	8

1 History and Development of LLMs

The evolution of large language models (LLMs) represents a convergence of deep learning, natural language processing, and computational scalability. Their development has been marked by increasingly sophisticated architectures, training methods, and compute resources.

1.1 Early Foundations

Language modeling began with statistical methods such as n-gram models, which approximate the probability of a word based on the previous $n - 1$ words using Markov assumptions. These models, although simple and intuitive, were limited by sparsity and the curse of dimensionality when modeling long-range dependencies.

The introduction of word embeddings, notably Word2Vec by Mikolov et al. (2013) and GloVe by Pennington et al. (2014), provided continuous vector representations of words that captured semantic relationships and improved downstream performance. These dense embeddings allowed neural networks to better generalize linguistic patterns.

1.2 Neural Language Models

- **RNNs and LSTMs** (2014–2017): Models like GRU and LSTM allowed for sequence modeling but struggled with long-term dependencies. - **Attention Mechanism**: Proposed by Bahdanau et al. (2014), attention allowed models to dynamically focus on input parts.

1.3 The Transformer Era

- **Transformer** (2017): Introduced by Vaswani et al. in "Attention is All You Need", it removed recurrence in favor of attention mechanisms, enabling better parallelization and scaling.

1.4 The Transformer Era

- **Transformer** (2017): Introduced by Vaswani et al. in "Attention is All You Need", it removed recurrence in favor of attention mechanisms, enabling better parallelization and scaling.

1.5 Scaling Laws and Pretrained LMs

- **BERT** (2018): A bidirectional model that used masked language modeling, fine-tuned for downstream tasks. - **GPT series** (2018–2023): GPT-1, GPT-2, GPT-3 and GPT-4 by OpenAI demonstrated that scaling model parameters and training data yielded strong zero-shot and few-shot capabilities. - **Chinchilla, PaLM, LLaMA, Claude, Mistral, Gemini**: Other contributors released competitive models optimized for compute, data, or specific use cases.

1.6 Trends in Development

- Scaling of parameters (from millions to trillions) - Efficient inference (e.g., quantization, sparse activation) - Instruction tuning and alignment (e.g., RLHF) - Open-source vs closed-source community dynamics

2 Overview of Actively Maintained LLMs

Many high-performance LLMs are maintained by corporate labs and the open-source community. The table below summarizes key models, their creators, and notable features.

Model	Organization	Highlights
GPT-4 / GPT-4-turbo	OpenAI	Proprietary; widely used in ChatGPT and API
Claude 3	Anthropic	Safety-aligned, large context windows
Gemini 1.5	Google DeepMind	Multimodal, integrated with Google services
Mistral / Mixtral	Mistral AI	Sparse mixture-of-experts, open weights
LLaMA 2 / 3	Meta AI	Open-access, research-friendly
Command R+	Cohere	Retrieval-augmented generation
Yi / InternLM / Qwen	Chinese labs (BAAI, Alibaba, etc.)	Multilingual, large open models
Perplexity.ai	Perplexity	LLM-powered search and answer engine
DeepSeek	DeepSeek AI	Semantic search and knowledge discovery
Grok	X (Elon Musk)	LLM integrated with X (Twitter) for real-time AI
PaLM	Google	Multimodal, conversational, powerful foundation model
Bard	Google	Conversational AI based on PaLM, search integrated
Ernie	Baidu	Leading Chinese language understanding model
OpenAssistant	Open Source Community	Community-driven, open assistant model
Alpaca	Stanford	Instruction-tuned, lightweight fine-tuned LLaMA variant
Falcon	Technology Innovation Institute	High-performance open LLMs
ChatGLM	Tsinghua University	Chinese open multilingual model
RWKV	RWKV Team	RNN-based LLM alternative architecture
Vicuna	Community	Fine-tuned LLaMA models with instruction tuning

These models are optimized for different use cases, including code generation, reasoning, search, and multilingual tasks. Some are fully open-source, while others are API-access only.

3 Overview of LLMs

Large Language Models (LLMs) are deep learning architectures trained to model natural language. They operate autoregressively, predicting the next token in a sequence given the previous context. Mathematically, they model the joint probability of a sequence $x = (x_1, x_2, \dots, x_T)$:

$$P(x) = \prod_{t=1}^T P(x_t | x_{<t})$$

This is typically optimized via maximum likelihood estimation over massive corpora.

4 Mathematical Foundation of Transformers

Transformers are the backbone of modern LLMs. Below, we break down the steps and their mathematical definitions.

4.1 Embedding

Each discrete token x_i is mapped into a continuous vector representation using an embedding matrix E :

$$e_i = E[x_i], \quad E \in \mathbb{R}^{V \times d}$$

where V is the vocabulary size and d the embedding dimension.

4.2 Positional Encoding

Because the Transformer is permutation-invariant, positional encodings inject order information:

$$PE_{i,2k} = \sin\left(\frac{i}{10000^{2k/d}}\right), \quad PE_{i,2k+1} = \cos\left(\frac{i}{10000^{2k/d}}\right)$$

$$e'_i = e_i + PE_i$$

4.3 Self-Attention

Self-attention allows each token to attend to others:

$$Q = e'W^Q, \quad K = e'W^K, \quad V = e'W^V$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

4.4 Multi-Head Attention

This process is repeated across multiple subspaces (heads):

$$\text{MHA}(Q, K, V) = [\text{head}_1; \dots; \text{head}_h]W^O$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

4.5 Feedforward Network

Each position passes through a shared MLP:

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$

4.6 Normalization and Residuals

Residual connections and layer normalization stabilize training:

$$\text{LayerNorm}(x + \text{SubLayer}(x))$$

5 Training Objectives and Statistical Methods

5.1 Maximum Likelihood Estimation

We optimize the model parameters θ by minimizing the negative log-likelihood:

$$\mathcal{L}(\theta) = -\sum_{t=1}^T \log P(x_t | x_{<t}; \theta)$$

5.2 Cross-Entropy Loss

This corresponds to the cross-entropy between the true distribution and predicted probabilities:

$$\text{Loss} = -\sum_{i=1}^T \log \hat{y}_i[x_i], \quad \hat{y}_i = \text{softmax}(z_i)$$

6 Optimization and Gradient Descent

6.1 SGD and Adam Optimizer

Gradients are computed via backpropagation. Adam updates are:

$$\begin{aligned}m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \\ \hat{m}_t &= \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \\ \theta_{t+1} &= \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}\end{aligned}$$

7 Inference and Generation Process

7.1 Autoregressive Sampling

Given the previous tokens, the next token is sampled:

$$h_t = f(h_{<t}, x_{<t}; \theta), \quad P(x_{t+1} | x_{\leq t}) = \text{softmax}(Wh_t + b)$$

7.2 Sampling Techniques

- **Greedy decoding:** $\arg \max_w P(w | x_{\leq t})$
- **Beam search:** maintains top-k hypotheses
- **Top-k/Top-p sampling:** samples from truncated distributions

7.3 Temperature Scaling

Controls output randomness:

$$P(w) = \frac{e^{z_w/T}}{\sum_{w'} e^{z_{w'}/T}}$$

7.4 Perplexity as an Evaluation Metric

Perplexity is a standard measure of how well a language model predicts a sequence. For a sequence of tokens $x_1, x_2, \dots, x_T$, the perplexity is defined as:

$$\text{Perplexity} = \exp \left(-\frac{1}{T} \sum_{t=1}^T \log P(x_t | x_{<t}) \right)$$

A lower perplexity indicates that the model assigns higher probabilities to the observed data, implying better predictive performance.

8 Contextual Understanding and Deep Thinking

LLMs show structured internal representations.

8.1 Semantic Layering

Layers build hierarchical understanding:

- Early layers focus on lexical co-occurrence
- Middle layers capture grammar
- Later layers model semantics and logic

8.2 Compositional Reasoning

Attention weights capture relations:

$$\alpha_{ij} = \frac{Q_i K_j^T}{\sqrt{d_k}}$$

8.3 Emergent Memory

The hidden states evolve to encode long-term dependencies:

$$H_t = f(H_{<t}, x_{<t})$$

8.4 Capabilities

- Zero/few-shot learning
- Tool usage and grounding
- Chain-of-thought reasoning and multi-step planning

9 Illustrative Example

Prompt: *"How do you optimize a neural network?"*

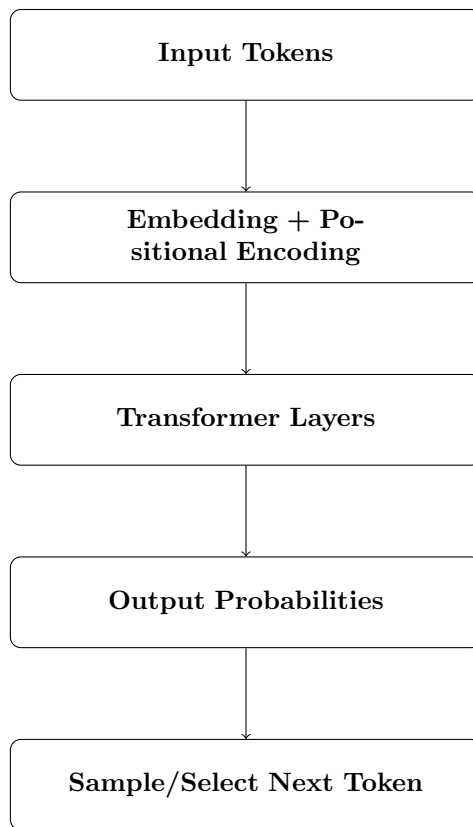
Processing Steps:

1. Tokenize the input sequence.
2. Embed tokens and add positional encodings.
3. Pass through multiple transformer layers.
4. At each step, predict the next token.
5. Sample next token using Top-p strategy.
6. Iterate until output is complete.

Sample Output:

"To optimize a neural network, define a loss function such as cross-entropy. Use an optimizer like Adam or SGD to compute gradients via backpropagation and update model weights accordingly."

10 Visual Diagram



11 Conclusion

This document explored the inner workings of large language models from a mathematical lens. By dissecting training dynamics, Transformer operations, and inference mechanisms, we revealed how these models exhibit rich behaviors like compositionality and emergent reasoning. While powerful, LLMs remain statistical approximators, and understanding their limits is key to responsible usage.