Introduction to Large Language Models

Definitions, Architectures, and Emerging Capabilities from a Computer Science Perspective.

Leszek J. Cierniak
leszek.cierniak@gmail.com

November 2025

Abstract

Large Language Models (LLMs) represent a transformative advancement in natural language processing (NLP), building upon foundational Language Models (LMs) to achieve human-like language understanding and generation through massive scale and sophisticated architectures. This paper provides a comprehensive overview from a computer science lens, defining LMs and LLMs, dissecting the Transformer-based architecture central to LLMs, exploring their functionalities, and contrasting them with traditional LMs. Key components like self-attention and positional encodings are detailed with mathematical formulations, while a glossary and references ensure accessibility. By highlighting scaling laws and emergent abilities, we underscore LLMs' role in enabling zero-shot learning and multimodal applications, alongside challenges like computational efficiency and ethical considerations. This analysis serves as a primer for researchers and practitioners who are looking to navigate the evolution of AI-driven language technologies while offering a systematic framework to compare LLM architectures and emerging behaviors.

Author Information

Leszek J. Cierniak holds a *Master's degree in Computer Science* from the **University of Wrocław** and has authored publications in expert systems and inference engines.

(MSc: *University of Wrocław*; former researcher: *University of Szczecin*; currently: *Independent Researcher*).

Leszek is an *IT Software* professional with over forty years of experience in software research, design, and development. His work spans enterprise and mobile systems, AI-driven solutions, and extensive international projects on the Temenos T24 banking platform. He has held roles ranging from architect and

developer to business analyst and quality leader. His technical portfolio includes Java, Flutter/Dart, Python, Prolog, and systems for symbolic computation.

Table of Contents

1. Introduction

In Computer Science, particularly in the fields of Machine Learning and Natural Language Processing (NLP), language models are key tools for modeling human language. This paper discusses the definitions, architecture, and functionality of LLMs, followed by a comparison with LMs. The analysis is based on fundamental concepts from computer science, such as Probabilistic Algorithms, Neural Networks, and Deep Learning Architectures.

The rapid evolution of LLMs has been marked by key milestones, each pushing the boundaries of scale, efficiency, and capability. Below is a chronological list of notable LLM models, highlighting their release years, developers, and approximate parameter sizes where applicable:

- **BERT (2018, Google, 340M parameters)**: Pioneered bidirectional pre-training, revolutionizing understanding tasks.
- **GPT-2 (2019, OpenAI, 1.5B parameters)**: Introduced scalable generative pre-training, sparking concerns over misuse.
- **T5 (2019, Google, 11B parameters)**: Unified text-to-text framework for diverse NLP tasks.
- **GPT-3 (2020, OpenAI, 175B parameters)**: Demonstrated few-shot learning at unprecedented scale.
- **PaLM (2022, Google, 540B parameters)**: Advanced chain-of-thought prompting for reasoning.
- **BLOOM (2022, BigScience, 176B parameters)**: First multilingual open-source LLM, emphasizing ethical training.
- **LLaMA (2023, Meta, 7B - 65B parameters)**: Efficient open models sparking the open-source LLM wave.
- **GPT-4 (2023, OpenAI, undisclosed, ~1.7T)**: Multimodal capabilities and superior reasoning.

- **Claude 2 (2023, Anthropic, undisclosed)**: Focused on safety and helpfulness in interactions.
- **Mistral 7B (2023, Mistral AI, 7B parameters)**: Compact, high-performing open model.
- **Grok-1 (2023, xAI, 314B parameters)**: Designed for real-time knowledge and humor.
- **Gemini 1.0 (2023, Google, undisclosed)**: Native multimodal architecture.
- **LLaMA 2 (2023, Meta, 7B - 70B parameters)**: Enhanced for safety and broader accessibility.
- **GPT-4o (2024, OpenAI, undisclosed)**: Optimized for speed and multimodality.
- **Claude 3 (2024, Anthropic, undisclosed)**: Improved vision and tool-use integration.
- **Grok-2 (2024, xAI, undisclosed)**: Advanced image understanding and efficiency.
- **LLaMA 3 (2024, Meta, 8B - 405B parameters)**: State-of-the-art open reasoning.
- **Gemini 1.5 (2024, Google, undisclosed)**: Expanded context windows to 1M+ tokens.
- **Claude 3.5 Sonnet (2024, Anthropic, undisclosed)**: Enhanced long-horizon planning and RAG/tool integration.
- **DeepSeek R1 (2025, DeepSeek AI, undisclosed)**: Specialized in coding and math.
- **Grok-4 (2025, xAI, undisclosed)**: Frontier model with real-time web integration.

These models illustrate the field's progression toward larger, more versatile systems, with ongoing trends in openness, multimodality, and efficiency as of November 2025. The progression of LLMs over the past decade illustrates not only an increase in parameter counts but also the evolution of underlying architectures and training paradigms. Early models such as BERT and GPT-2 demonstrated the value of bidirectional and generative pre-training, while subsequent models like GPT-3 and PaLM revealed the emergent capabilities enabled by scaling both data and network size. More recent models—including LLaMA, GPT-4, and Gemini—highlight trends toward multimodality, optimized efficiency, and open-source accessibility, reflecting an ongoing shift from task-specific language models to versatile, general-purpose systems. This chronological overview underscores the interplay between architectural innovations, such as Transformers and self-attention mechanisms, and scaling principles, which together form the foundation of modern LLMs.

The following sections delve into these architectures in detail, providing the technical framework necessary to understand the capabilities and limitations of large-scale models.

2. Definitions

2.1. Language Model

A Language Model (LM) is a probabilistic mathematical model that assigns probabilities to sequences of words (or tokens) in natural language.
Formally, an LM estimates the probability

$$P(w_1, w_2, \ldots, w_n) = \prod_{t=1}^{n} P(w_t \mid w_1, \ldots, w_{t-1})$$

for a sequence of words w_1 to w_n, which allows for predicting the next word based on the previous ones (next-word prediction).

LMs are the foundation of many NLP tasks and rely on the Markov assumption, where the probability depends primarily on a limited history of context. LMs can be statistical (e.g., n-grams) or neural (e.g., based on recurrent neural networks, RNNs). Their goal is to model the distribution of language, enabling text generation, error correction, or auto-completion. Traditional LMs, particularly statistical ones, are limited by fixed context windows and sparsity of observed sequences, which restricts their ability to capture long-range dependencies in language.

These models form the computational foundation upon which modern Large Language Models (LLMs) are built.

2.2. Large Language Model

Building upon the probabilistic formulation of traditional LMs, a Large Language Model (LLM) represents a scaled deep learning system.

A Large Language Model (LLM) is an advanced, scaled-up version of an LM, characterized by a massive number of parameters (typically from hundreds of millions to trillions) and trained on enormous datasets (e.g., billions of tokens from the internet). LLMs, such as GPT-4 or LLaMA, are designed to understand and generate language in a way that approximates human-like behavior, exhibiting emergent abilities (e.g., step-by-step reasoning) that go beyond simple word prediction.

Emergent abilities often manifest when model scale crosses thresholds in parameters, dataset size, or context window length (Wei et al., 2022). For example, few-shot learning typically appears in models with parameters ranging from 100B (LLaMA 3) upward, a property absent in smaller LMs.

From a computer science perspective, LLMs are instances of deep learning where scale—number of parameters, data, and computational power—leads to zero-shot or few-shot generalization, i.e., the ability to solve new tasks without additional training.

3. LLM Transformer Architecture

Large Language Models are overwhelmingly based on the **Transformer** architecture, introduced by Vaswani et al. in 2017. The Transformer's key innovation is its reliance on the attention mechanism, allowing for parallel processing of the input sequence and the effective capture of long-range dependencies, overcoming the sequential limitations of prior Recurrent Neural Networks (RNNs).

Key components:

- **Encoder** — Converts input tokens to dense vectors.
- **Decoder** — Converts target tokens to output vectors.

The architecture is typically composed of a stack of identical blocks, each containing a Multi-Head Self-Attention layer and a position-wise Feed-Forward Network (FFN), with residual connections and layer normalization applied after each sub-layer.

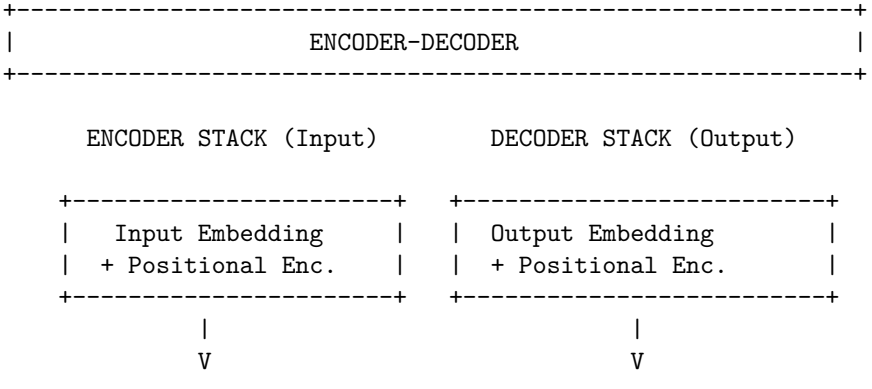
Modern LLMs are predominantly *decoder-only* Transformers, which are autoregressive models optimized for text generation. However, the architecture has three main types: Encoder-only (e.g., BERT, focused on understanding and representation), Decoder-only (e.g., GPT, LLaMA, focused on generation), and Encoder-Decoder (e.g., T5, focused on Seq2Seq tasks like translation).

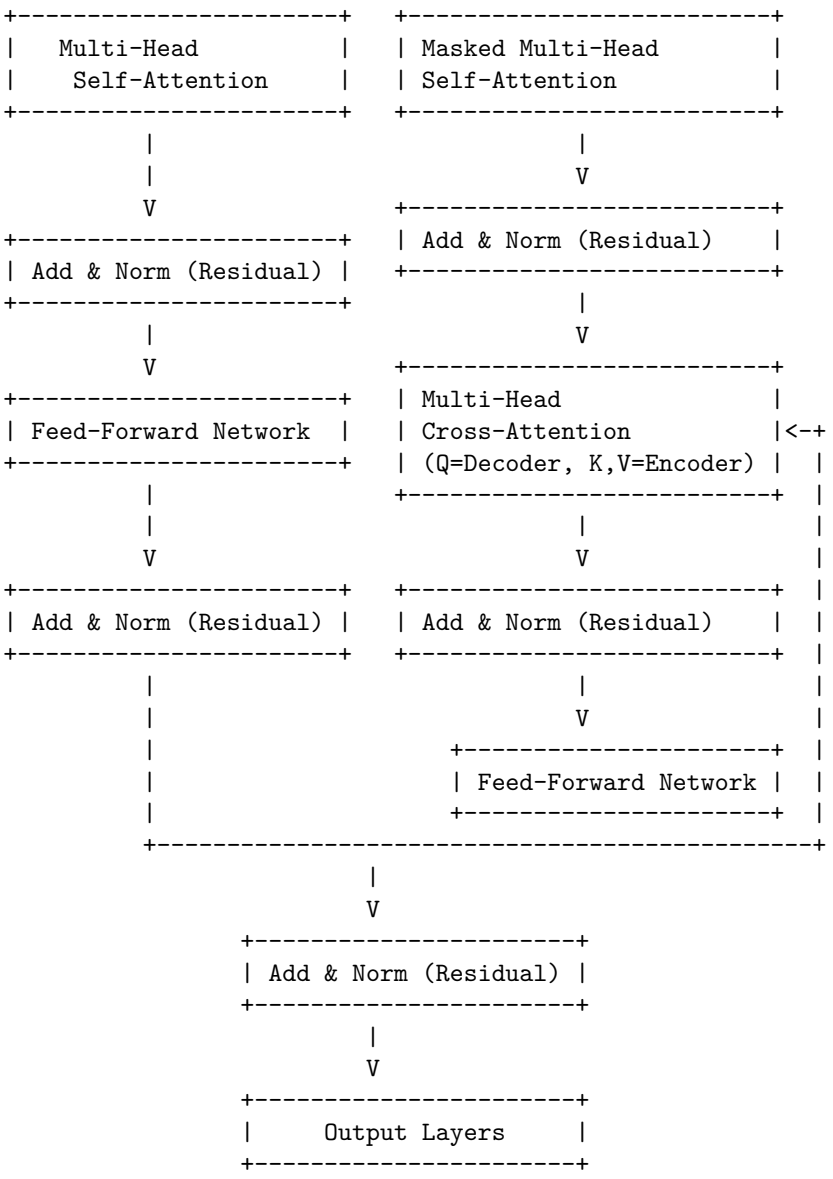
The following paragraphs give a clear, concise visual and textual reference for the three Transformer variants.

3.1. Encoder-Decoder Architecture (e.g., T5, BART, Original Transformer)

This is the original architecture proposed in the "Attention Is All You Need" paper. It's used for Sequence-to-Sequence (Seq2Seq) tasks where the input and output sequences are different (e.g., translating a sentence). (Seq2Seq tasks were introduced by Sutskever et al., 2014 and Cho et al., 2014)

Figure 3.1: Schematic of the Encoder-Decoder Transformer Architecture (inspired by Vaswani et al., 2017).

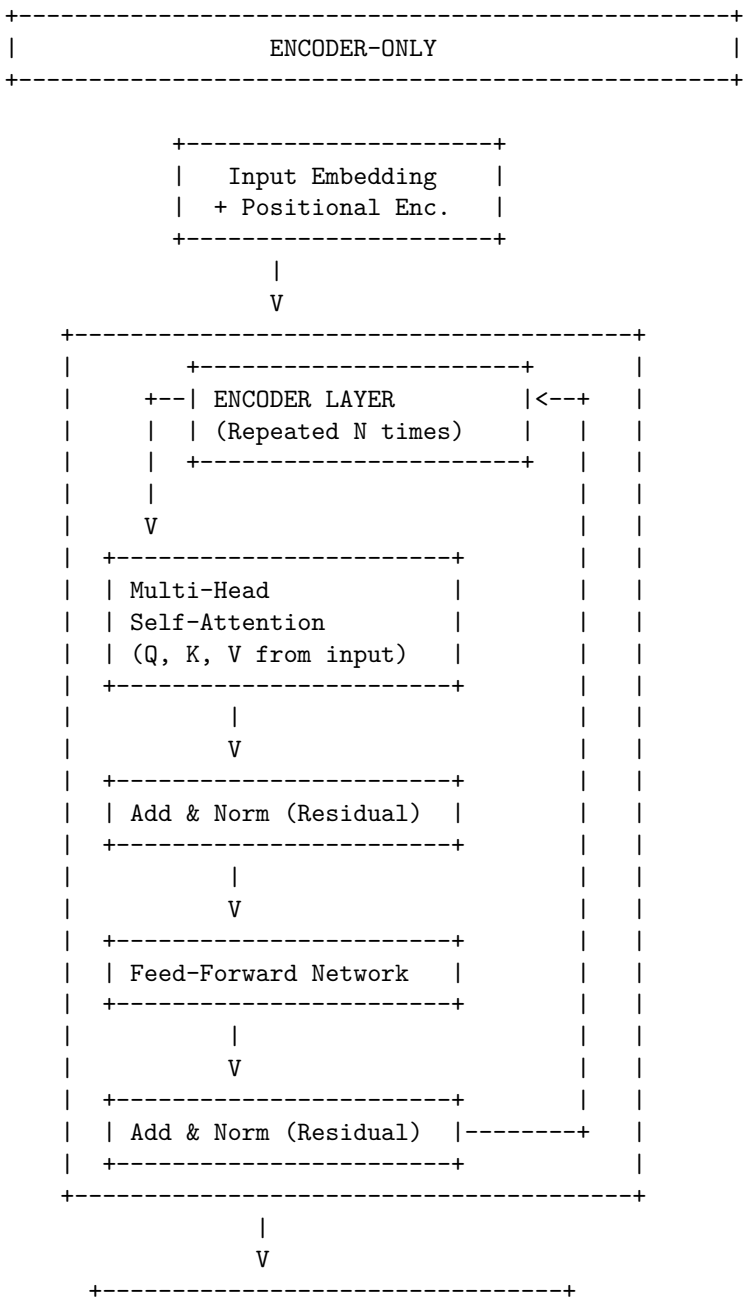


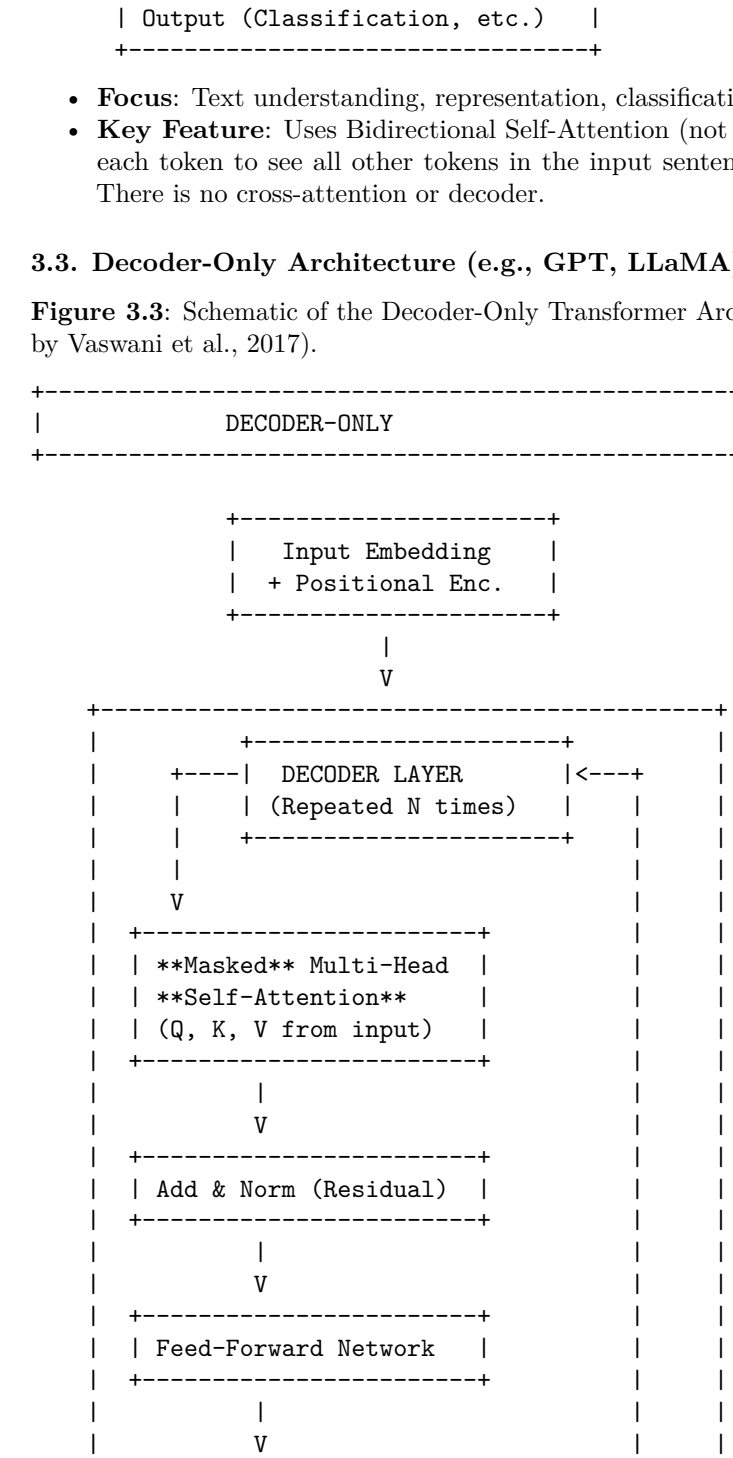


- **Focus**: Sequence-to-Sequence tasks like machine translation and summarization.
- **Key Feature**: Connects the Encoder (which reads the input) and the Decoder (which generates the output) via a Cross-Attention layer.

3.2. Encoder-Only Architecture (e.g., BERT, RoBERTa)

Figure 3.2: Schematic of the Encoder-Only Transformer Architecture (inspired by Vaswani et al., 2017).

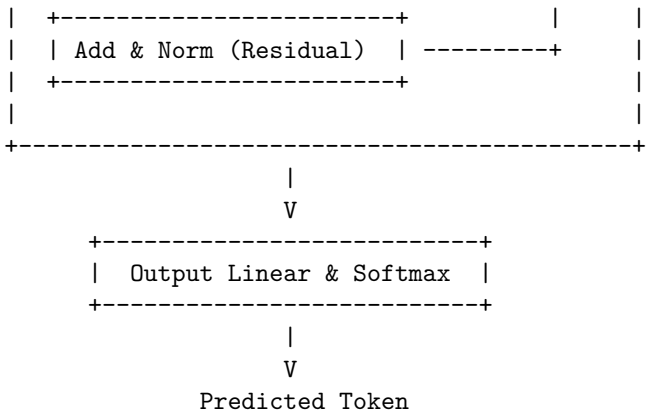




- **Focus**: Text understanding, representation, classification.
- **Key Feature**: Uses Bidirectional Self-Attention (not masked), allowing each token to see all other tokens in the input sentence simultaneously. There is no cross-attention or decoder.

3.3. Decoder-Only Architecture (e.g., GPT, LLaMA)

Figure 3.3: Schematic of the Decoder-Only Transformer Architecture (inspired by Vaswani et al., 2017).



- **Focus**: Text generation, auto-regressive tasks (predicting the next token).
- **Key Feature**: Uses Masked Self-Attention, preventing tokens from "seeing" future tokens. This makes it ideal for generating new text one word at a time. This is the most common LLM architecture.

3.4. Comparative Analysis of LLM Transformer Architectures

Table 3.4.1 Primary Task

Model Type	Primary Task
Encoder-Only	Understanding Representation
Decoder-Only	Autoregressive Generation
Encoder-Decoder	Translation/Summarization Seq2Seq

Table 3.4.2 Attention Type

Model Type	Attention Type
Encoder-Only	Bidirectional Self-Attention
Decoder-Only	Masked Self-Attention
Encoder-Decoder	Enc: Bidirectional Self-Attention. Dec: Masked Self-Attention + Cross-Attention

Table 3.4.3 Key Limitation

Model Type	Limitation
Encoder-Only	Not suitable for generation (sees future tokens)
Decoder-Only	Less effective for understanding full input context
Encoder-Decoder	Needs paired input/output sequences for training

Model Type	Limitation

Table 3.4.4 Example Models

Model Type	Example
Encoder-Only	BERT, RoBERTa, Electra
Decoder-Only	GPT, LLaMA, PaLM
Encoder-Decoder	T5, BART, Original Transformer

4. Advances in Transformer Design

4.1. The Self-Attention Mechanism

The core of the Transformer is the Scaled Dot-Product Attention. It calculates an output sequence that is a weighted sum of the input values, where the weights are determined by the compatibility of the queries and keys.

For an input sequence matrix X, the mechanism uses three learned projection matrices, W_Q, W_K, and W_V, to transform X into Query (Q), Key (K), and Value (V) matrices: $Q = XW_Q$, $K = XW_K$, and $V = XW_V$.

The mathematical formulation for the Scaled Dot-Product Attention is given by:

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

This formulation, as introduced by Vaswani et al. (2017), enables efficient dependency modeling.

where:

- Q: The Query matrix, representing what the current token is searching for.
- K: The Key matrix, representing what each token in the sequence has to offer.
- V: The Value matrix, holding the actual content to be aggregated.
- d_k: The dimension of the Key vectors, used as a scaling factor to stabilize the gradients.

While this mechanism is powerful, its quadratic computational complexity with respect to the input sequence length, $O(N^2)$ where N is the sequence length, becomes a significant bottleneck when dealing with the massive context windows of modern LLMs (e.g., Gemini 1.5's 1M+ tokens). The main high-level extensions focus on improving efficiency, incorporating external knowledge, and adapting the attention scope.

4.1.1. Efficiency and Context Length Extension The primary challenge is the $O(N^2)$ complexity. Extensions address this by either changing the attention pattern or by optimizing the calculation.

- **Sparse Attention** — Instead of calculating attention between *every* pair of tokens, sparse attention models calculate it for a predefined subset of pairs. The goal is to capture long-range dependencies efficiently while maintaining performance and reducing the complexity to $O(N\sqrt{N})$ or even $O(N)$ for some patterns.
- **Windowed or Localized Attention** — This approach restricts the self-attention calculation to a fixed-size window around the current token. It assumes that most relevant context for a token lies nearby.
- **Linear Attention** — This extension modifies the attention operation to avoid the explicit QK^T matrix calculation, achieving linear complexity ($O(N)$) with respect to the sequence length.

4.1.2. Incorporating External Knowledge The standard self-attention mechanism only operates on the information present *within* the current input sequence. Extensions aim to augment this with external, factual knowledge.

- **Retrieval-Augmented Attention (RAA)** — A Transformer architecture in which the attention mechanism integrates externally retrieved information directly into its key/value space. A retrieval component selects relevant documents or memory vectors from an external knowledge base, and the model attends jointly to the input tokens and the retrieved content. This improves *grounding, factual accuracy, and reduces hallucinations.*
- **Memory-Augmented Attention (MAA)** — A Transformer mechanism that integrates an *external memory* directly into the attention module, allowing the model to read (and sometimes write) additional key/value vectors beyond the current sequence and thereby use information from much longer or multiple contexts.

4.1.3. Adapting Attention for Modality and Scope These extensions adapt the mechanism for tasks beyond text and for hierarchical reasoning.

- **Cross-Attention (Multimodal)** — While self-attention focuses on one sequence, *cross-attention* is fundamental to *multimodal models* (e.g., GPT-4V, Gemini). It calculates attention between two *different* input sequences (modalities), such as a text query (Q) and an image representation (K and V). This allows the model to align and reason across different data types.
- **Hierarchical Attention** — Designed to process structured inputs (like long documents or code), this mechanism applies attention at multiple levels: a *local attention* for words within a sentence/paragraph, and a *global attention* for combining the information from different local units. This mirrors how humans process nested information.

4.2. Modern Positional Encoding: Rotary Position Embedding (RoPE)

Since the self-attention mechanism is permutation-invariant—meaning it doesn't inherently understand the order of tokens—positional encodings are necessary to inject sequence order information. While early Transformers used fixed sinusoidal or learned absolute encodings, contemporary LLMs rely on **Rotary Position Embedding (RoPE)** to improve context length extrapolation.

RoPE, as used in models like Llama, Gemma, and Mistral, encodes positional information by **rotating** the Query (Q) and Key (K) vectors in the self-attention layer by an angle proportional to their absolute position.

The primary advantages of RoPE are:

- **Relative Position Awareness:** The dot product $Q \cdot K$ between two tokens becomes a function of their *relative distance*, not their absolute position, which is a stronger inductive bias for language.
- **Extrapolation:** The continuous rotational formulation allows the model to generalize effectively to sequence lengths significantly longer than those encountered during training.

4.3. Attention Optimizations for Inference: Grouped-Query Attention (GQA)

The **Multi-Head Attention (MHA)** layer, where multiple attention mechanisms (heads) operate in parallel, is key to the Transformer's representational power. However, MHA suffers from a high memory bandwidth requirement during inference, especially due to the size of the **Key-Value (KV) Cache**—the memory storage for K and V vectors of previous tokens.

To mitigate this bottleneck, modern high-performance LLMs employ **Grouped-Query Attention (GQA)**, which acts as a compromise between MHA and the faster but lower-quality Multi-Query Attention (MQA).

- **Multi-Head Attention (MHA)**: Each Query head has its own Key and Value head ($H_Q = H_K = H_V$). (Highest quality, slowest inference).
- **Multi-Query Attention (MQA)**: All Query heads share a single Key and Value head ($H_K = H_V = 1$). (Fastest inference, lower quality).
- **Grouped-Query Attention (GQA)**: The Query heads are partitioned into G groups, where each group of Query heads shares a single Key and Value head ($1 < H_K = H_V = G < H_Q$).

By sharing key/value heads across multiple query heads, GQA significantly reduces the size of the KV cache and the memory bandwidth required, achieving near-MHA quality at a speed approaching MQA. This optimization is crucial for efficient deployment of very large models.

4.4. Stabilization and Training Efficiency

To allow for the stacking of hundreds of layers, *a requirement for LLMs with billions of parameters*, two crucial components are used:

- **Residual Connections** (or **Skip Connections**): These connections add the input of a sub-layer to its output, ensuring that the gradient can flow directly through the network during backpropagation. This prevents the vanishing gradient problem, which is a significant barrier to training deep neural networks.
- **Layer Normalization:** Applied after the residual connection, normalization adjusts the features across the layer for each individual token. This stabilizes the hidden layer inputs, accelerating convergence and reducing the need for extensive training-time hyperparameter tuning during the pre-training phase.

4.5. Computational Scaling Challenges and Attention Alternatives

The primary limitation of the vanilla Transformer is the $O(N^2)$ computational complexity of self-attention, where N is the sequence length. This quadratic scaling makes training and inference prohibitively expensive for very long context windows (e.g., 1M+ tokens from Gemini 1.5).

This bottleneck manifests in two ways:

- **Computational Cost**: The time required to compute the attention matrix QK^T.
- **Memory Cost**: The large memory footprint of the *Key-Value (KV) Cache* during auto-regressive inference, which stores the K and V vectors for all previously generated tokens.

Modern LLMs address this through two high-level strategies:

- **Optimization**: Techniques like *Grouped-Query Attention (GQA)* reduce the KV Cache size by sharing key/value heads, accelerating inference without compromising quality.
- **Approximation**: Methods like *Sparse Attention* or *Linear Attention* modify the attention pattern (e.g., using a fixed window or a kernel function) to achieve a near-linear $O(N)$ or $O(N\sqrt{N})$ complexity, enabling significantly longer context processing.

Emerging alternatives like State Space Models (SSM), e.g., Mamba architectures integrated in DeepSeek R1 (2025), offer truly linear O(N) complexity for ultra-long contexts, complementing Transformer approximations.

5. From Generation to General-Purpose Agents

The true power of Large Language Models lies in their ability to generalize and perform complex tasks that were not explicitly included in their supervised train-

ing data. Modern LLM functionality extends far beyond simple text generation, moving toward capabilities required for true artificial general intelligence (AGI).

5.1. In-Context Learning and Emergent Abilities

In-Context Learning (ICL) is a core, distinguishing capability of large Transformer models, enabling them to learn a new task simply by analyzing a few examples provided within the prompt's context window, *without* requiring an update to the model's millions or billions of weights.

- **Zero-Shot Learning (ZSL)**: The model performs a task (e.g., translation) based purely on its vast pre-trained knowledge, requiring only the task instruction.
- **Few-Shot Learning (FSL)**: Performance is dramatically boosted by providing a handful of input-output examples directly in the prompt, allowing the model to quickly infer the task format and constraints.
- **Emergence**: ICL is a capability that *emerges* only after the model crosses a certain threshold of scale (parameters, data, or both), as predicted by *scaling laws*.

5.2. Multimodal Integration and Cross-Domain Understanding

Modern LLMs are increasingly **multimodal**, extending their processing and generation capabilities beyond text to integrate various data types (e.g., images, video, and audio). This requires an architectural shift:

- **Specialized Encoders**: Separate neural network modules (e.g., Vision Transformers) pre-process non-textual data, converting it into a dense vector representation (embedding).
- **Modality Projector/Adapter**: A lightweight neural network layer bridges the semantic gap, translating the multimodal embeddings into the same dimensional space as the core language model's input embeddings. This allows the LLM's **Cross-Attention** mechanism to seamlessly fuse information from all modalities, enabling reasoning and response generation based on a holistic, integrated understanding.

5.3. Tool Use and Action Planning (LLMs as Agents)

Tool Use transforms the LLM from a passive prediction machine into an *active, embodied agent* capable of interacting with external systems and the real world. The LLM's role shifts to that of a *planner* and *reasoner* that decides *which tool* to use, *when*, and *how to interpret the results*.

- **Knowledge Augmentation**: For instance, Grok-4 (xAI, 2025) leverages real-time X API integration for dynamic fact-checking during agentic workflows. The model dynamically calls a search engine or external database API to retrieve real-time or domain-specific information. This directly

addresses the model's knowledge cutoff problem and drastically improves *factuality*.

- **Functionality Extension**: The model generates and executes programmatic calls to structured external tools (e.g., a Python interpreter for math/coding, a calculator for arithmetic, or a robotic API for physical control). This allows the model to reliably perform tasks it cannot handle intrinsically.

5.4. Advanced Reasoning and Self-Correction

To solve complex, multi-step problems reliably and reduce the tendency to *"hallucinate"*, LLMs employ advanced prompting and iterative refinement techniques:

- **Chain-of-Thought (CoT) Prompting**: The model is explicitly prompted to generate intermediate reasoning steps *before* providing the final answer (e.g., "Let's think step by step"). This provides a computational scratchpad, significantly increasing the success rate on multi-step reasoning tasks.
- **Self-Correction/Verification**: For tasks requiring high fidelity (e.g., coding, robotic planning), the LLM is prompted to critique its own initial output or plan using formal logic or execution feedback. This iterative process allows the model to identify failures and re-generate a refined plan, dramatically increasing success rates on **long-horizon planning** tasks.

6. LM vs LLM

In summary, LMs are the foundation of NLP, but LLMs represent a breakthrough in scalable machine learning, where increases in scale lead to qualitative leaps via scaling laws (Kaplan et al., 2020). LLMs generalize LMs, enabling applications like intelligent assistants, with future focus on efficiency (e.g., sparse models in LLaMA 4) and ethics (bias mitigation in Claude Sonnet 4).

The following tables contain the key differences from a computer science perspective, highlighting the evolution from simple probabilistic models to scalable deep networks.

Table 6.1 Definition

Model Type	Definition
Language Model	Probabilistic model of word sequences; can be small and statistical (e.g., n-grams).
Large Language Model	Scaled-up LM with over 1B parameters; deep learning-based; emergent abilities.

Table 6.2 Architecture

Model Type	Architecture
Language Model	Simple: n-grams (statistical) or RNN/LSTM (neural, sequential). Limited scale (millions of parameters).
Large Language Model	Transformer-based (self-attention, multi-head). Decoder-only or encoder-decoder. Scale: billions of parameters, parallel processing.

Table 6.3 Functionality

Model Type	Functionality
Language Model	Basic: word prediction, auto-completion, simple generation. Requires task-specific training.
Large Language Model	Advanced: zero/few-shot learning, reasoning, multimodality. Emergent skills (e.g., coding, math).

Table 6.4 Training

Model Type	Training
Language Model	Small data; supervised/unsupervised on specific datasets. Low computational power.
Large Language Model	Massive data (terabytes of text); pre-training + fine-tuning. High power: hundreds of GPUs, months of training.

7. Summary and Future Directions

Having established the fundamental distinctions between traditional LMs and modern LLMs, we can synthesize key insights and identify research directions shaping the next generation of language models.

This exploration illustrates how LLMs extend the probabilistic roots of LMs into a new era of AI capabilities, driven by Transformer innovations and scaling principles. Key insights from this analysis include:

- **Scalability and Emergent Abilities**: LLMs demonstrate that increasing model parameters, data, and context window size leads to

qualitative improvements, including few-shot and zero-shot learning, and chain-of-thought reasoning.

- **Architectural Innovations**: Transformer-based architectures remain central to enabling parallelism, long-range dependency modeling, and multimodal integration.

- **Real-World Applications**: LLMs are increasingly applied in code generation, scientific reasoning, multimodal processing, and interactive AI assistants.

- **Challenges**: Despite their power, LLMs require substantial computational resources, pose ethical and safety concerns (bias, hallucinations), and raise sustainability issues due to training energy costs.

Future directions include:

1. **Hybrid Architectures**: Integrating symbolic reasoning with neural networks to enhance reasoning and interpretability. This direction falls under the broad area of *Learning AI Models*, which aims to build systems that acquire knowledge autonomously. A key challenge is *Language-Grounded Learning*, where an AI system starts from *tabula rasa* (complete ignorance) and acquires structured knowledge solely through *pedagogical dialogue* with a human teacher (e.g., learning the rules of a domain like checkers). This investigates the challenge of grounding linguistic concepts into executable, symbolic knowledge. Recent advances, such as quantum-inspired scaling in LLaMA 4 (Meta, 2025), further bridge neural-symbolic gaps.
2. **Multimodal Expansion**: Extending LLM capabilities to audio, video, and robotics applications.
3. **Efficient Training**: Sparse models, LoRA (Low-Rank Adaptation), and quantization techniques for energy-efficient inference.
4. **Ethics and Safety**: Improved bias mitigation, model interpretability, and robustness checks for responsible deployment.
5. **Open-Source Exploration**: Experimenting with models like LLaMA, Grok, and Gemini to advance reproducibility and accessibility.

This framework provides a roadmap for researchers to extend the boundaries of LLM capabilities while addressing the technical, ethical, and societal challenges inherent in large-scale AI systems.

8. Glossary

- **AI Agent**: An advanced system that utilizes an LLM as its core reasoning and planning engine to interact with and take autonomous actions within a structured environment.
- **Attention Mechanism**: Computes the relevance of tokens within a sequence, allowing the model to focus dynamically on important parts.

- **Autoregressive Generation**: Predicting one token at a time, each conditioned on all previous tokens.
- **Causal Masking**: A technique in decoder-only Transformers to prevent attention to future tokens during generation.
- **Chain-of-Thought Prompting (CoT)**: Providing intermediate reasoning steps in prompts to improve model inference.
- **Cross-Attention**: An attention mechanism used to compute the relevance between tokens from *two different sequences* or modalities (e.g., a text query and image features).
- **Emergent Abilities**: Capabilities (e.g., reasoning, arithmetic) that arise only in sufficiently large models.
- **Feed-Forward Network (FFN)**: A position-wise neural network applied independently to each token to enhance model capacity.
- **Few-Shot / One-Shot / Zero-Shot Learning**: Solving tasks with few, one, or no examples provided during inference.
- **Grouped-Query Attention (GQA)**: An inference optimization where multiple Query heads share a single Key and Value head (grouping them) to reduce KV Cache memory and accelerate performance.
- **In-Context Learning (ICL)**: The ability of an LLM to learn a new task from examples provided in the prompt *without* updating model weights, encompassing zero-shot and few-shot learning.
- **KV Cache**: A memory storage that holds the Key (K) and Value (V) vectors of previously processed tokens during auto-regressive inference, saving re-computation but consuming significant memory bandwidth.
- **Layer Normalization**: Normalizes features in each layer to stabilize training in deep networks.
- **Linear Attention**: Attention mechanisms that modify the core dot-product operation to achieve a linear $O(N)$ computational complexity with respect to the sequence length N.
- **Long-Horizon Planning**: The capability of an LLM or AI Agent to decompose and execute a complex task that requires multiple sequential steps and potentially self-correction over an extended period.
- **Low-Rank Adaptation (LoRA)**: A fine-tuning technique updating only a small subset of model parameters, significantly reducing computation and storage cost.
- **Masked Self-Attention**: Self-attention with future tokens masked, essential for autoregressive generation.
- **Memory-Augmented Attention (MAA):** A mechanism that integrates an external memory directly into the attention module.
- **Modality Projector**: A lightweight neural network layer used in multimodal LLMs to map non-textual embeddings (e.g., from an image encoder) into the language model's embedding space.
- **Multi-Head Attention**: Parallel attention heads that capture diverse relationships and are concatenated for richer representations.
- **Multimodal Model**: A model capable of processing more than one modality, e.g., text, image, audio.

- **Next-Token Prediction**: The primary pre-training objective for LLMs, predicting the next token in a sequence.
- **Positional Encoding**: Encodes token positions to compensate for Transformers' permutation invariance.
- **Pre-Training**: Unsupervised learning on large corpora to establish general representations.
- **Prompt Engineering**: Crafting input prompts to elicit desired LLM behavior without fine-tuning.
- **Quantization**: Reducing numerical precision of model parameters to improve efficiency.
- **Residual Connection**: Adding the input of a sub-layer to its output to maintain gradient flow.
- **Retrieval-Augmented Attention (RAA)**: A mechanism in which a model retrieves external information (e.g., from a vector store or key/value memory) inside the attention layers.
- **Retrieval-Augmented Generation (RAG)**: A method combining external knowledge retrieval with generative models to improve factuality and reduce hallucination.
- **Rotary Position Embedding (RoPE)**: A positional encoding method that rotates the Query and Key vectors in the self-attention mechanism to capture relative position and enable sequence length extrapolation.
- **Scaling Laws**: Empirical relationships linking model performance to parameter count, data, and compute.
- **Self-Attention**: Attention computed within the same sequence to model dependencies.
- **Sequence-to-Sequence (Seq2Seq)**: The general framework of converting one sequence into another sequence using an encoder-decoder structure.
- **Sparse Attention**: Attention mechanisms that restrict the calculation of the attention matrix to a small, non-zero subset of token pairs (e.g., local windows or pre-defined patterns) to reduce computational complexity from $O(N^2)$ to near $O(N)$.
- **State Space Model (SSM)**: A class of sequence models that uses a latent state to summarize historical information, offering linear computational complexity with sequence length (e.g., Mamba).
- **Tokenization**: Dividing text into discrete units (tokens) for model input.
- **Tool Use**: The capability of an LLM to generate and execute an external function call (e.g., an API, code interpreter, or search engine) to gather information or perform an action outside of its internal network.
- **Transformer**: Core architecture of LLMs, using stacked attention and feed-forward layers.
- **Zero-Shot Learning**: Solving tasks without any examples provided in the prompt.

9. Acknowledgments

The author thanks the open research community for its ongoing efforts to advance transparency and reproducibility in large language models.

AI assistance was used for structural review, including support from Grok (xAI), Gemini (Google; Flash 2.5 model), ChatGPT (OpenAI; GPT-5.1), and Claude Sonnet 4.5 (Anthropic).

10. References

- Anthropic. (2025). *Introducing Claude Sonnet 4.5: Enhanced planning and RAG*. https://www.anthropic.com/news/claude-sonnet-4-5
- Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). *Layer Normalization*. arXiv preprint arXiv:1607.06450. https://arxiv.org/abs/1607.06450
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). *Language models are few-shot learners. Advances in Neural Information Processing Systems*, 33, 1877-1901. https://arxiv.org/abs/2005.14165
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171-4186. Association for Computational Linguistics. https://arxiv.org/abs/1810.04805
- Gu, A., & Dao, T. (2023). *Mamba: Linear-time sequence modeling with selective state spaces*. arXiv preprint arXiv:2312.00752. https://arxiv.org/abs/2312.00752
- Haoyan Xu, et al.; LLM-Powered Text-Attributed Graph Anomaly Detection via Retrieval-Augmented Reasoning; https://arxiv.org/abs/2511.17584
- Jérémie Chalopin, Maria Kokkou; Constant-Size Certificates for Leader Election in Chordal Graphs and Related Classes; https://arxiv.org/abs/2511.19208
- Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). *Scaling laws for neural language models*. arXiv preprint arXiv:2001.08361. https://arxiv.org/abs/2001.08361
- Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Empirical Methods in Natural Language Processing (EMNLP), 2014. arXiv preprint arXiv:1406.1078. https://arxiv.org/abs/1406.1078
- Raghu, M. & Schmidt, E. A survey of deep learning for scientific discovery. arXiv preprint arXiv:2302.04761. https://arxiv.org/abs/2003.11755
- Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer,

L., Cancedda, N., & Scialom, T. (2023). *Toolformer: Language Models Can Teach Themselves to Use Tools*. arXiv preprint arXiv:2302.04761. https://arxiv.org/abs/2302.04761

- Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In Advances in Neural Information Processing Systems (NIPS), 2014. arXiv preprint arXiv:1409.3215. https://arxiv.org/abs/1409.3215
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). *LLaMA: Open and efficient foundation language models*. arXiv preprint arXiv:2302.13971. https://arxiv.org/abs/2302.13971
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need*. *Advances in Neural Information Processing Systems*, 30, 5998–6008. https://arxiv.org/abs/1706.03762
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E. H., Le, Q., & Zhou, D. (2022). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. arXiv preprint arXiv:2201.11903. https://arxiv.org/abs/2201.11903
- Wei, J., et al. (2022). *Emergent abilities of large language models*. arXiv preprint arXiv:2206.07683. https://arxiv.org/abs/2206.07683
- Yuan An, et al.; Rate-Distortion Guided Knowledge Graph Construction from Lecture Notes Using Gromov-Wasserstein Optimal Transport; https://arxiv.org/abs/2511.14595