

Center of Professional Development and Community Outreach

Generative AI Training Program - Professional Development (Reskilling) Training Program

Generative AI - The Transformer Architecture

Module 2-B: The Transformer Architecture

(The technical heart of LLMs)

Dr. Abeer Al-Hyari

Abeer.alhyariups@htu.edu.jo

Module-2-B

Module	Module Name and Focus	Key Topics to Cover	PLOs Addressed
Module 2-B	The Transformer Architecture (The technical heart of LLMs)	<ol style="list-style-type: none">1. Deep Dive into the Transformer Block (Encoder/Decoder).2. The Mechanics of Self-Attention.3. Positional Encoding and Tokenization.4. Evolution of Transformers: GPT (Decoder-only) vs. BERT (Encoder-only) vs. T5 (Encoder-Decoder).5. Introduction to Foundational Models (LLMs) and their capabilities.	Foundations (Primary), Model Development

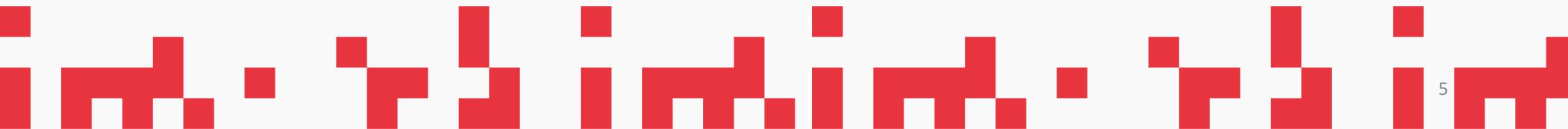
Objectives

1. Demonstrate an understanding of the Transformer's input preparation pipeline.
2. Explain the core mechanics of the **Self-Attention** mechanism (Q, K, V).
3. Differentiate the roles and internal components of the **Encoder Block** (for comprehension) and the **Decoder Block** (for generation) within the full Transformer model.
4. Differentiate between the three main Transformer architectural variants (**Encoder-Only**, **Decoder-Only**, and **Encoder-Decoder**).
5. Explain the relationship between the **Transformer Architecture** and the emergence of modern **Foundation Models (LLMs)**.
6. Analyze the concept of **Emergent Capabilities**.



Topics Covered

- Deep Dive into the **Transformer Block** (Encoder/Decoder).
- The Mechanics of **Self-Attention**.
- Positional Encoding and Tokenization.
- Evolution of Transformers: GPT (Decoder-only) vs. BERT (Encoder-only) vs. T5 (Encoder-Decoder).
- Introduction to Foundational Models (LLMs) and their capabilities.

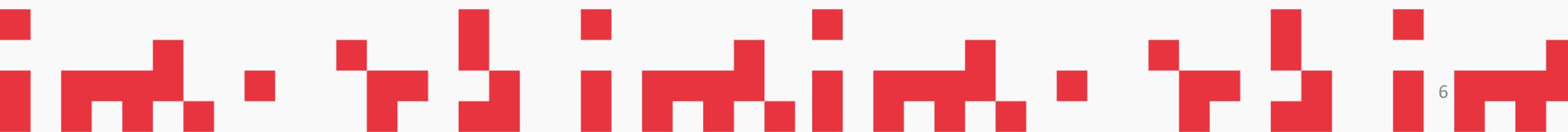


The Transformer Architecture: Deep Dive

- The Transformer is built on two main components—the **Encoder** and the **Decoder**—and its entire process relies on converting human language into numerical vectors through three stages: **Tokenization, Embeddings, and Positional Encoding.**

I. Input Preparation: Converting Text to Vectors

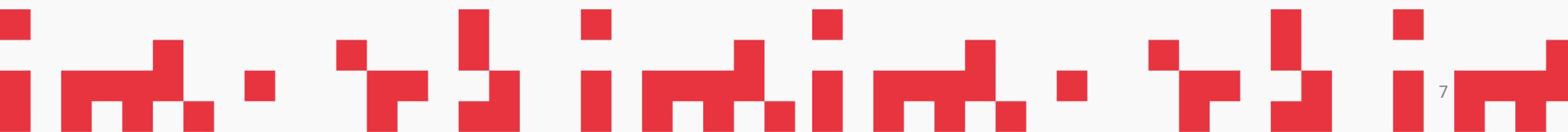
The first task is to transform raw text into a mathematical format that the model can process.



Step 1: Tokenization

- This is the process of breaking down raw text (like a sentence) into manageable units called tokens.
 - Process: The model uses a pre-defined Vocabulary to map sub-words, words, or characters to a unique numerical ID.
 - Example: "unsupervised learning" might become three tokens with IDs:

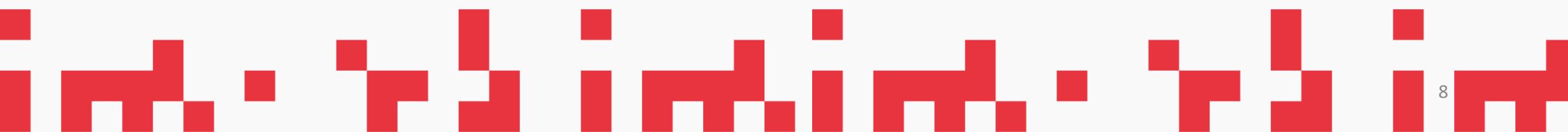
21, 550,1928



Step 2: Input Embeddings

This layer converts the numerical token IDs into dense **vectors** (lists of numbers).

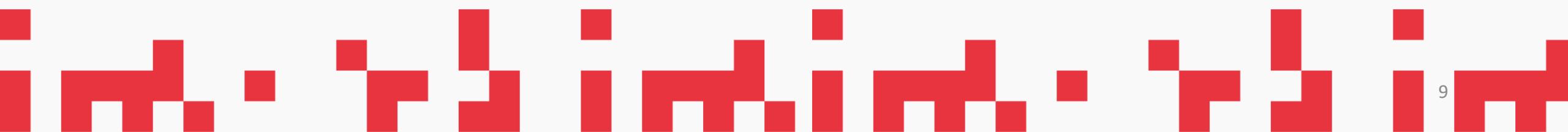
- **Function:** Each vector captures the **semantic meaning** of the token. Words with similar meanings (e.g., "king" and "queen") are positioned closer together in this high-dimensional vector space.



Positional Encoding (Order Matters)

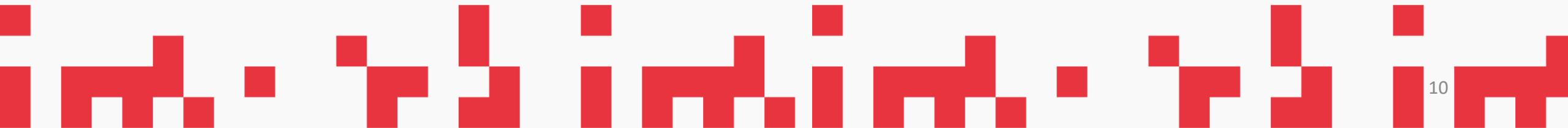
Because the Transformer processes all words simultaneously (in parallel), it inherently loses information about word order.

- **Function:** **Positional Encoding** is a second vector added element-wise to the word embedding. This vector contains a unique pattern (often based on sine and cosine functions) that tells the model the absolute or relative **position** of the token in the sequence.
- **Outcome:** The final input vector for a word contains both its meaning (*what*) and its position (*where*).



The Attention Core: Mechanics of Self-Attention

- The **Self-Attention** mechanism is the revolutionary component that allows the Transformer to calculate the contextual relevance between all tokens in a sequence simultaneously.

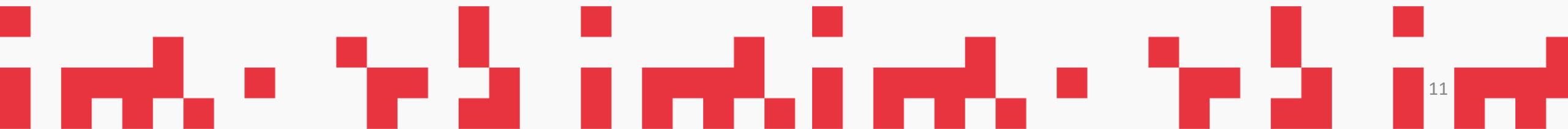


Q-K-V: The Math Behind Attention

The Mechanics (Q, K, V)

For each token in the sequence, the model generates three dedicated vectors by multiplying its embedding by learned weight matrices:

1. **Query (Q)**: What I am **looking for** (the current token's request for context).
2. **Key (K)**: What I **offer** as context (the identity/tag of all other tokens).
3. **Value (V)**: The **content** or actual information of all other tokens.



Multi-Head Attention

Multi-Head Attention enhances the basic Self-Attention mechanism by providing the model with multiple perspectives on the data.

- **Process:** The Query, Key, and Value matrices are linearly projected and split into several smaller, parallel sets (**Attention Heads**).
- **Benefit:** Each head learns to look for a **different type of relationship** (e.g., one head may track subject-verb dependencies, another tracks adjective-noun relationships).
- **Output:** The results from all heads are calculated independently, then **concatenated** and passed through a final linear layer. This combines the diverse contextual information into a single, comprehensive vector representation.

The Transformer Block Components

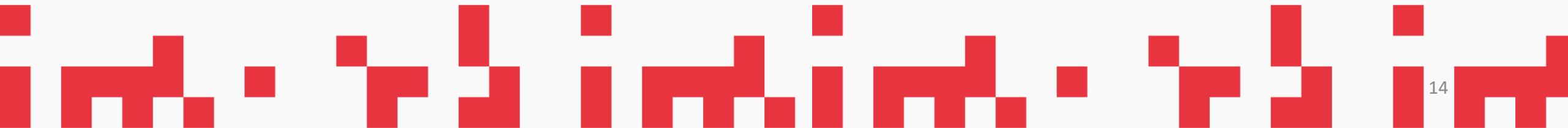
Summarize a single block

1. Multi-
2. Feed-Forward Network (Position-wise FFFN).
3. Residual Connections & Layer Normalization (for stability and gradient flow).



The Original Seq2Seq (Encoder-Decoder)

- The **Original Seq2Seq (Encoder-Decoder)** architecture is the foundational concept for all sequence transduction tasks in deep learning, such as machine translation. While originally built with Recurrent Neural Networks (RNNs), its modern and most powerful form is the **full Transformer architecture** introduced in 2017.
- It is defined by its two distinct, communicating components:



The Original Seq2Seq (Encoder-Decoder)

1. The Encoder Stack (Understanding the Input)

The Encoder's job is to read the entire input sequence and convert it into a rich, contextual representation.

- **Function:** Processes the input sentence (e.g., "The quick brown fox.") to generate a sequence of deep, bidirectional representations.
- **Attention:** Uses **Multi-Head Self-Attention** which allows every word to look at all other words in the input simultaneously, building a complete understanding of the context.
- **Output:** The final layer of the Encoder produces a set of contextualized vectors (the encoded "understanding") which will be passed to the Decoder.

The Original Seq2Seq (Encoder-Decoder)

2. The Decoder Stack (Generating the Output)

The Decoder's job is to generate the new output sequence one token at a time, using the understanding provided by the Encoder.

- **Function:** Generates the target sequence (e.g., the French translation) in an **autoregressive** (left-to-right) manner.
- **Causal Self-Attention:** The Decoder uses a **Masked Multi-Head Self-Attention** mechanism to ensure that when generating the current word, it can only see the words it has **already generated** (the past), not the words it is about to generate (the future).
- **Cross-Attention (The Link):** This is the key feature connecting the two stacks. The Decoder uses a **Cross-Attention** layer to query the Encoder's final output. This allows the Decoder to dynamically focus its attention on the most relevant parts of the *input* sequence for generating the *next* word in the *output* sequence.

The Original Seq2Seq (Encoder-Decoder)

- <https://youtu.be/L8HKweZIOmg>



The Original Seq2Seq (Encoder-Decoder)

Primary Use Cases

- The Encoder-Decoder architecture is ideal for tasks that require mapping a complete input sequence to a transformed output sequence, where the output length or structure is different from the input.
- **Machine Translation:** Translating a sentence from English to Arabic. The Encoder reads the full English sentence first, then the Decoder generates the Arabic one word at a time, cross-attending to the source text as needed.
- **Abstractive Summarization:** Reading a long document (Encoder) and generating a concise, novel summary (Decoder).
- **Image Captioning:** An image encoder extracts features, which are then cross-attended by a language decoder to generate a descriptive sentence.

Encoder-Only Architectures (BERT)

Key Characteristics of Encoder-Only Models

Encoder-only models are optimized for Natural Language Understanding (NLU) tasks that require a deep, contextual interpretation of the entire input text.

1. Architecture: Single-Stack Encoder

An encoder-only model consists only of the **stack of Encoder blocks** from the original Transformer architecture. It completely omits the Decoder stack.

2. Attention: Bidirectional Context

The Self-Attention mechanism within the encoder layers is **bidirectional** (non-causal).

- **Function:** Each word (token) in the input sequence can simultaneously attend to **all other words** in the sentence, regardless of whether they come before or after it.
- **Benefit:** This provides a rich, holistic understanding of the full context, which is crucial for determining the meaning of ambiguous words or resolving references.

Encoder-Only Architectures (BERT)

Key Characteristics of Encoder-Only Models

3. Pre-training Task: Masked Language Modeling (MLM)

BERT, the key encoder-only model, is pre-trained using a technique called **Masked Language Modeling (MLM)**.

- The model randomly **masks** about 15% of the words in a sentence (e.g., "The dog ate the [MASK] and ran away.").
- The model is trained to predict the original masked words based on the context provided by **all the surrounding, unmasked words** (both left and right).
- This forces the model to build a deep, contextual representation of every token.

Encoder-Only Architectures (BERT)

Key Characteristics of Encoder-Only Models

4. Output and Use Cases (Discriminative)

Encoder-only models do not generate free-form text. Their output is a sequence of **contextualized vector embeddings** for the input tokens. These embeddings are then fed into a smaller task-specific layer (**task head**) for classification or prediction.

Encoder-Only Architectures (BERT)

Use Case	Description
Text Classification	Predicting a label for an entire document (e.g., sentiment analysis: "Positive" or "Negative").
Named Entity Recognition (NER)	Identifying and categorizing key entities in a sentence (e.g., identifying "Google" as an Organization).
Extractive Question Answering	Reading a paragraph and extracting the exact span of text that answers a given question.
Retrieval-Augmented Generation (RAG)	Generating highly accurate and contextual embeddings for searching and retrieving documents, which are then used by a separate generative model.

Encoder-Only Architectures (BERT)

- https://www.youtube.com/watch?v=GDN649X_acE



Decoder-Only Architectures (GPT, Llama)

- The **Decoder-Only Architecture** is the fundamental design for all modern, publicly available Large Language Models (LLMs) used for generation and conversation, such as **GPT** (Generative Pre-trained Transformer), **Llama**, **Mistral**, and **Claude**.
- It is a simplification of the original Transformer that is specialized for one task: **autoregressive text generation**.

Decoder-Only Architectures (GPT, LLaMA)

Architecture and Mechanism

A Decoder-Only model consists solely of the **Decoder stack** from the original Encoder-Decoder architecture, with two critical modifications:

1. Masked Self-Attention (The Core of Generation)

The key component is the **Masked Multi-Head Self-Attention** mechanism. This layer ensures the model only uses the context of **previous tokens** when making a prediction for the current token.

- **Causality:** This implements *causal*/attention (or *unidirectional*/attention), preventing the model from "cheating" by looking at future words in the sequence.

Decoder-Only Architectures (GPT, Llama)

- **Autoregression:** This causal restriction forces the model to generate text **autoregressively**, predicting one token at a time (x_i) based only on the tokens that have already been generated (x_1, x_2, \dots, x_{i-1}). This process repeats until the model generates an end-of-sequence token.

2. Removal of Cross-Attention

Because the Encoder stack is removed, the Decoder block's **cross-attention layer** (which typically links the Decoder to the Encoder's output) is also eliminated.

The entire model is a single, deep stack of these modified decoder blocks.

Decoder-Only Architectures (GPT, Llama)

Task	Description	Advantage
Generative & Creative	Chatbots, poetry, fiction writing, and long-form content generation.	High fluency and contextual coherence across long text spans.
Instruction Following	Answering questions, summarizing documents, or writing code based on a prompt.	The pre-training process makes them excellent zero-shot learners —they can adapt to new tasks just from instructions.
Efficiency in Chat	Handling multi-turn conversations efficiently.	During conversation, the model can cache the attention keys and values of all previous turns. When a new user message arrives, it only has to compute the attention for the new tokens, saving significant computation time.

Decoder-Only Architectures (GPT, LLaMA)

- <https://www.youtube.com/watch?v=bQ5BoolX9Ag>



Summary: Choosing the Right Transformer

- Choosing the right Transformer architecture depends entirely on your machine learning task's goal: **Are you trying to generate, understand, or transform text?**
- You must match the **required direction of attention** (unidirectional or bidirectional) and the **input-output mapping** to the most suitable architecture.

Summary: Choosing the Right Transformer

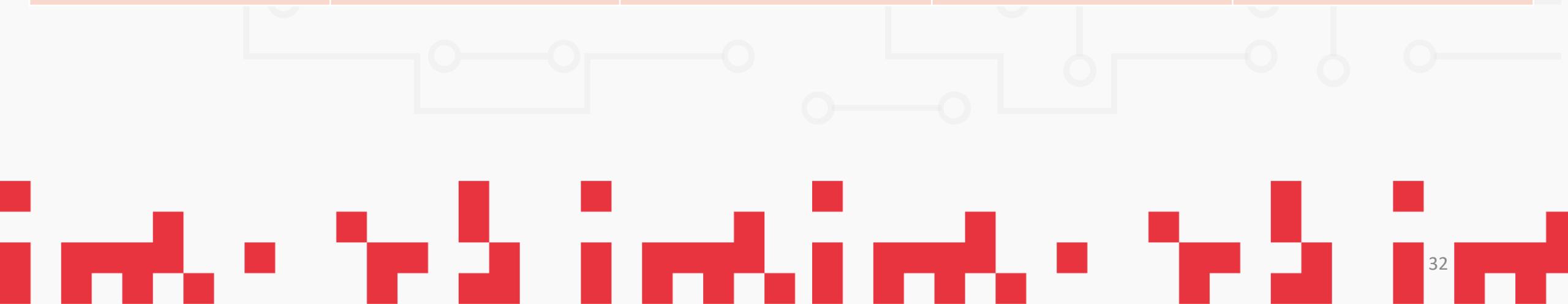
Architecture	Primary Goal	Attention Mechanism	Example Models	Best for...
1. Decoder-Only	Generation (Create new content)	Causal/Masked Self-Attention (looks only backwards).	GPT series, Llama, Mistral, Claude	Autoregressive Tasks: Chatbots, creative writing, content generation, and code completion. <i>Generates text fluently.</i>

Summary: Choosing the Right Transformer

Architecture	Primary Goal	Attention Mechanism	Example Models	Best for...
2. Encoder-Only	Understanding (Analyze existing content)	Bi-directional Self-Attention (looks both forward and backward).	BERT, RoBERTa, ELECTRA	Classification Tasks: Sentiment analysis, Named Entity Recognition (NER), extractive Q&A, and generating embeddings for Retrieval-Augmented Generation (RAG).

Summary: Choosing the Right Transformer

Architecture	Primary Goal	Attention Mechanism	Example Models	Best for...
3. Encoder-Decoder	Transformation (Map one sequence to a different one)	Encoder (Bi-directional) and Decoder (Causal + Cross-Attention).	T5, BART, UniLM	Sequence-to-Sequence Tasks: Machine Translation, Abstractive Summarization, and conditional generation.



Summary: Choosing the Right Transformer

Task Example	Best Choice	Rationale
Translate English to Arabic		Requires full reading of the source (Encoder) before creating the target (Decoder).
Summarize a 10-page document		Encoder-Decoder is strong for <i>abstractive</i> summarization; Decoder-Only is good for <i>extractive</i> or <i>prompt-based</i> summarization.
Identify the author's tone in a review		Requires comprehensive understanding of the whole text; no new text is generated.
Chatbot to answer user questions		Designed for seamless, multi-turn, and human-like generation .
Generate code from a comment		Treats the comment as a prompt and generates the code sequence autoregressively.

Summary: Choosing the Right Transformer

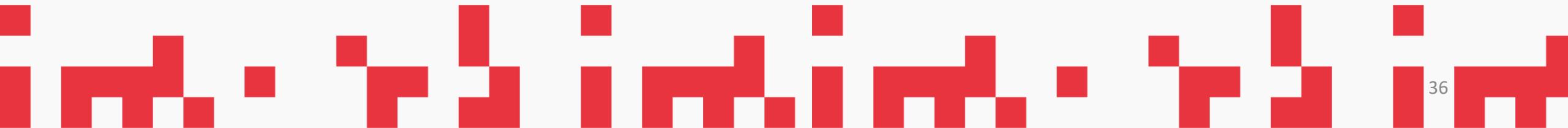
Task Example	Best Choice	Rationale
Translate English to Arabic	Encoder-Decoder	Requires full reading of the source (Encoder) before creating the target (Decoder).
Summarize a 10-page document	Encoder-Decoder (or Decoder-Only)	Encoder-Decoder is strong for <i>abstractive</i> summarization; Decoder-Only is good for <i>extractive</i> or <i>prompt-based</i> summarization.
Identify the author's tone in a review	Encoder-Only	Requires comprehensive understanding of the whole text; no new text is generated.
Chatbot to answer user questions	Decoder-Only	Designed for seamless, multi-turn, and human-like generation .
Generate code from a comment	Decoder-Only	Treats the comment as a prompt and generates the code sequence autoregressively.

Introduction to Foundation Models (LLMs)

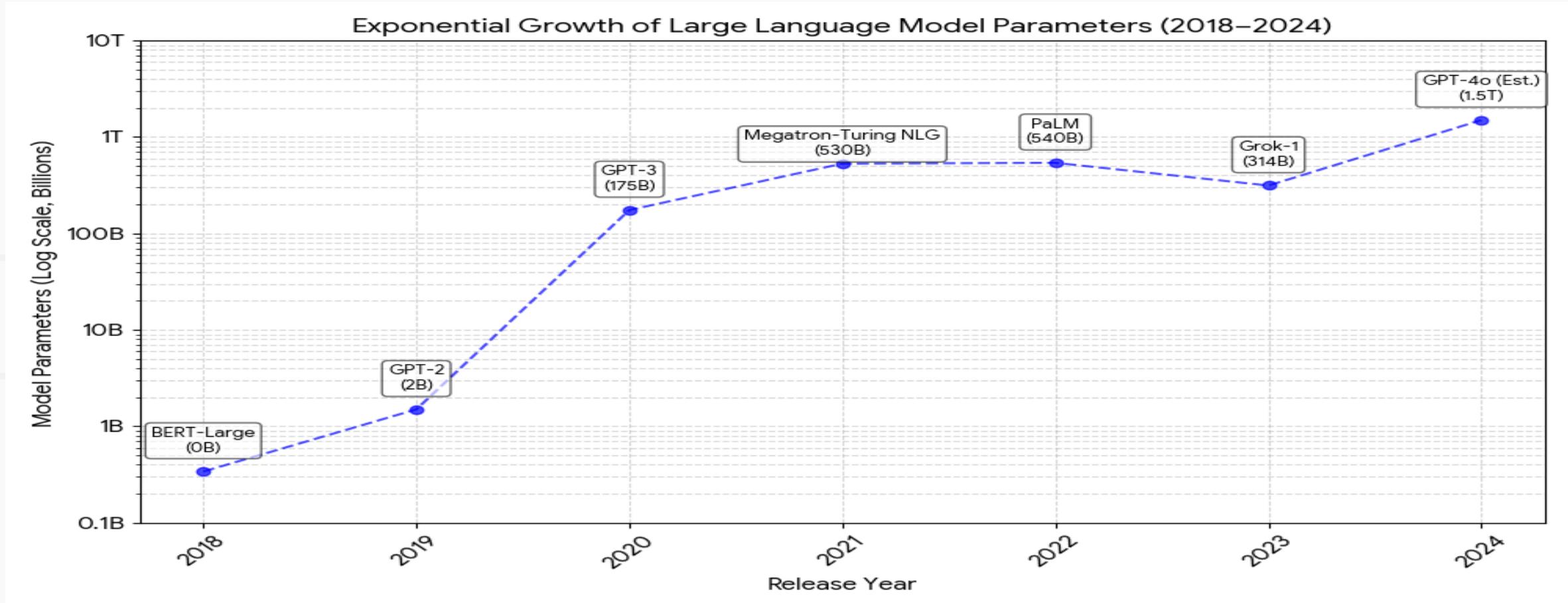


LLMs as Scaled Transformers

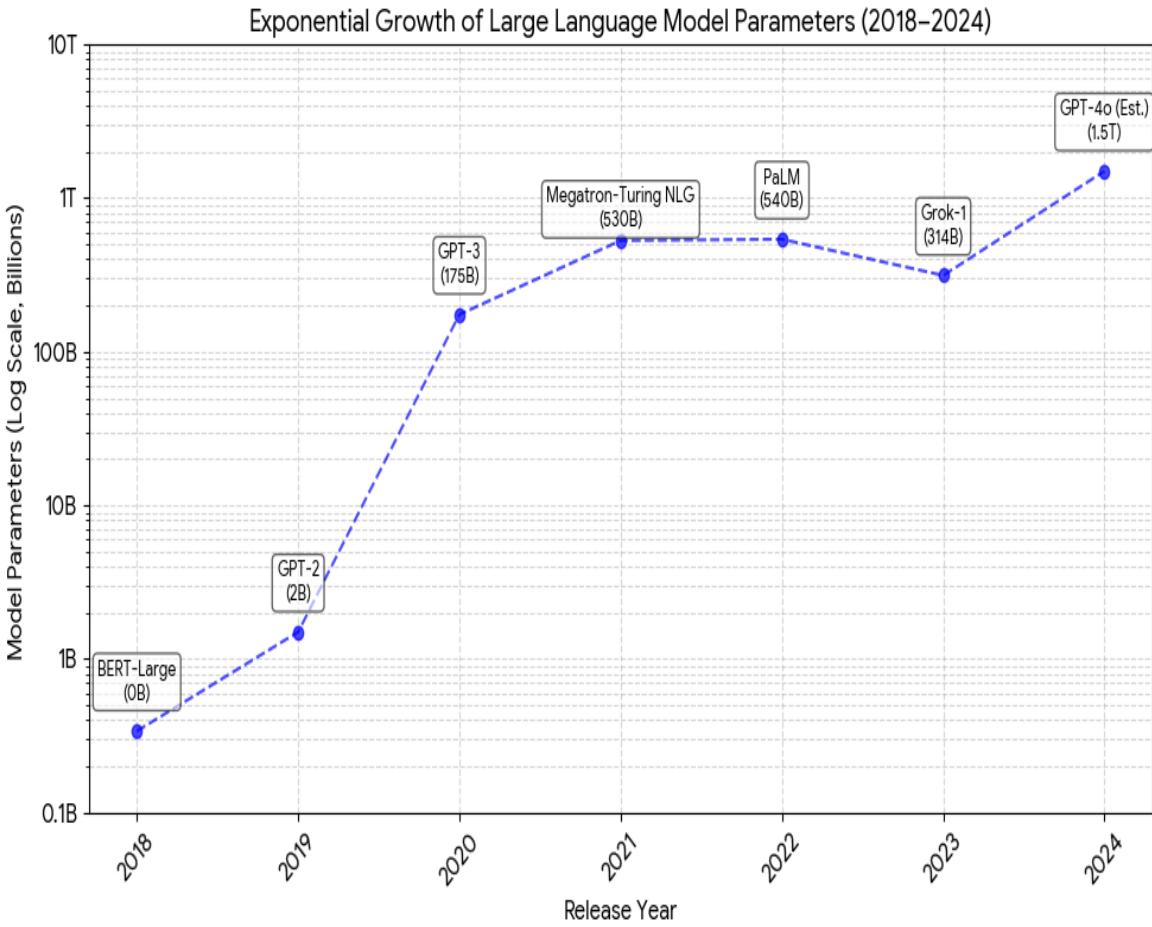
- **Definition:** A **Large Language Model (LLM)** is a **Foundation Model (FM)** (a model pre-trained on massive data) that exclusively uses the **Transformer** architecture (usually the **Decoder-Only** stack).



LLMs as Scaled Transformers



LLMs as Scaled Transformers



- **BERT-Large (2018):** 0.34 B parameters, representing the start of the massive Transformer era.
- **GPT-3 (2020):** A jump to 175 B parameters, which proved the non-linear benefits of extreme scale (Emergent Abilities).
- **PaLM (2022):** Continued scaling to 540 B parameters.
- **GPT-4o (Est. 2024):** Represents the current frontier, with an estimated size around 1.5T parameters (1.5 Trillion).

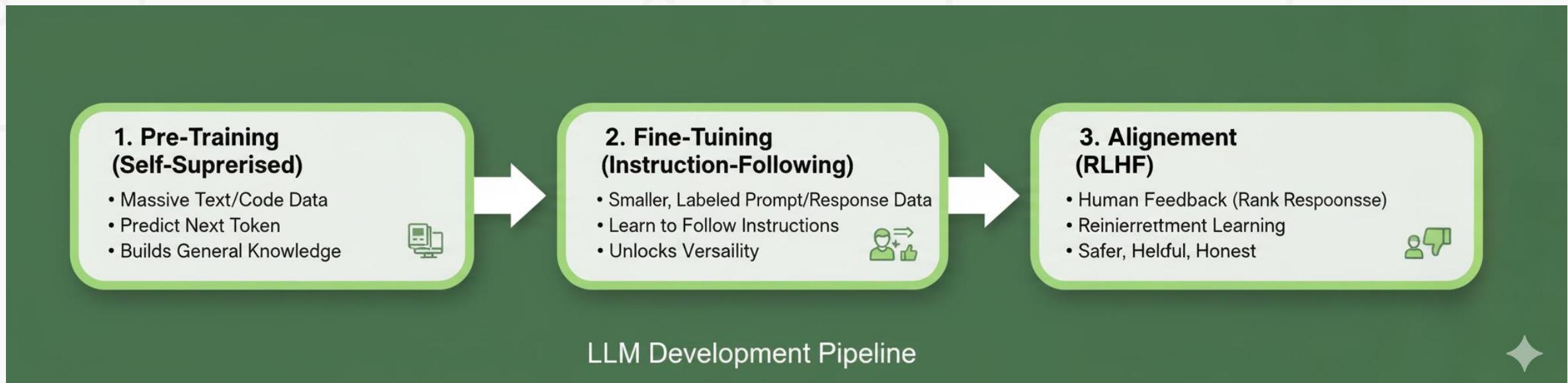
Capabilities Unlocked by Scale

Key Emergent Capabilities Highlighted:

- **In-Context Learning (ICL):** The model's ability to learn a new task from just a few examples or instructions provided directly in the prompt, without needing any explicit training updates.
- **Chain-of-Thought (CoT) Reasoning:** The ability to break down complex problems into intermediate steps, showing its "thinking process," which significantly improves performance on multi-step reasoning tasks (e.g., math word problems, logic puzzles).
- **Complex Code Generation:** Generating intricate, functional code snippets or entire programs based on natural language descriptions.

The LLM Development Pipeline

- 1. Pre-Training (Unsupervised):** Transformer is trained on raw internet data (predicting next token).
- 2. Alignment (Supervised/RLHF):** Fine-tuned to follow human instructions, making it a chatbot/assistant.



Module 2 Summary & Transition

- The Transformer solved the parallelization problem using attention, leading directly to the scaled, versatile LLMs we use today.

