A Comprehensive Analysis of Modern Large Language Model Architectures: From Transformers to Emerging Paradigms

Section 1: Introduction: The Architectural Blueprint of Modern AI

1.1 The Pre-Transformer Epoch: The Challenge of Sequential Data

Before the advent of the Transformer architecture, the field of natural language processing (NLP) was dominated by models designed to process data sequentially, most notably Recurrent Neural Networks (RNNs) and their more sophisticated variants, such as Long Short-Term Memory (LSTM) networks.[1] These models were conceptually intuitive, processing a sequence of words or tokens one at a time, updating an internal "hidden state" at each step that theoretically encapsulated the information from all preceding tokens.[2] This step-by-step processing, however, concealed a fundamental architectural limitation: the "fundamental constraint of sequential computation".[4]

This sequential nature posed two primary, intertwined challenges that ultimately stalled progress. The first was a practical hardware limitation. As machine learning began to leverage the massive parallel processing capabilities of Graphics Processing Units (GPUs), the inherently serialized nature of RNNs became a significant bottleneck.[5] Because the computation for a given token depended on the hidden state produced by the previous token, the entire sequence could not be processed simultaneously. This serialization posed a formidable challenge to parallelization, drastically limiting training speed and the ability to scale models to the vast datasets and parameter counts that define the modern era of large

language models (LLMs).[5]

The second challenge was a theoretical and practical limitation in capturing long-range dependencies within the data.[8] While LSTMs and their variants like Gated Recurrent Units (GRUs) were specifically designed to mitigate the "vanishing gradient" problem that plagued simple RNNs, they still struggled to maintain a coherent representation of information across very long sequences.[10] Information from the beginning of a long paragraph or document could become diluted or entirely "lost" by the time the model reached the end, creating a persistent information bottleneck.[6] In the context of sequence-to-sequence (seq2seq) tasks like machine translation, early encoder-decoder models based on RNNs would attempt to compress the entire meaning of an input sentence into a single, fixed-length context vector.[3] This approach proved to be a critical flaw; a single vector was often insufficient to capture the nuance and complexity of a long input sentence, leading to a degradation in performance as sequence length increased.[8] These models had access to the state vector only after the last word of the source text was processed, making it difficult to preserve all relevant information.[8] The inability to effectively leverage parallel hardware and the inherent struggle with long-term memory were not minor issues but fundamental architectural roadblocks that necessitated a new approach.

1.2 The Paradigm Shift: Attention Is All You Need

The solution to the constraints of recurrent models emerged in 2017 with the publication of the seminal paper "Attention Is All You Need" by a team of eight researchers at Google: Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin.[11] This paper is widely considered a foundational text in modern artificial intelligence and a primary catalyst for the subsequent AI boom.[11] It introduced a "novel, simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely".[4]

The core breakthrough of the Transformer was not the invention of the attention mechanism itself, which had been proposed earlier and was already being used to connect encoders and decoders in the best-performing recurrent models.[4] Rather, the revolutionary act was the realization that recurrence was not a prerequisite for high-performance sequence transduction. The Transformer architecture was designed to "draw global dependencies between input and output" without relying on sequential processing.[4] By eliminating the recurrent components, the model could apply its attention mechanism to all tokens in a sequence simultaneously.

This architectural choice directly addressed the primary bottleneck of its predecessors:

parallelizability. The Transformer's design allows it to be trained significantly faster and on much larger datasets, as it can fully leverage the parallel processing power of modern hardware.[8] This efficiency unlocked the ability to scale models to unprecedented sizes, directly enabling the "large" in Large Language Models and setting a new standard for performance on tasks like machine translation. On English-to-German and English-to-French translation tasks, the single Transformer model presented in the paper outperformed the previous state-of-the-art results, which were based on complex ensembles of recurrent or convolutional models.[12] The architectural liberation from sequential constraints is what allowed for the massive scaling that defines the current landscape of AI.

1.3 Thesis and Report Structure

This report provides a deep technical analysis of the Transformer architecture and its primary derivatives, demonstrating how specific design choices in each family lead to specialized capabilities and distinct performance profiles. It will deconstruct the core mechanisms of the Transformer, trace the divergence into three principal architectural families—Encoder-Only, Decoder-Only, and Encoder-Decoder—and evaluate emerging paradigms like Mixture-of-Experts and State-Space Models that address the Transformer's own scaling limitations. A comparative framework, grounded in empirical performance on standardized benchmarks, computational efficiency, and task suitability, will be presented to map the current and future landscape of LLM architectures.

The report is structured as follows: Section 2 will provide a granular deconstruction of the foundational Transformer architecture, detailing its core components, including the self-attention mechanism, multi-head attention, and the complete Transformer block. Section 3 will analyze the three main architectural families that evolved from this foundation, exploring their unique pre-training objectives and task specializations. Section 4 will investigate recent innovations designed to overcome the efficiency bottlenecks of the standard Transformer, namely Mixture-of-Experts and State-Space Models. Section 5 will present a data-driven comparative analysis of these architectures, examining their performance on key benchmarks, their computational and data requirements, and the scaling laws that govern their behavior. Finally, Section 6 will synthesize the findings and offer a concluding perspective on the evolutionary trajectory and future directions of LLM architecture.

Section 2: The Foundational Paradigm: Deconstructing the Transformer

The Transformer architecture is built upon a set of interconnected components that, together, enable it to process entire sequences of data in parallel while capturing complex, long-range dependencies. This section dissects these fundamental building blocks, from the core innovation of self-attention to the complete, stacked structure of a Transformer block.

2.1 The Core Innovation: Self-Attention

At the heart of the Transformer is the self-attention mechanism, a process that allows the model to dynamically weigh the importance of all other words in a sequence when producing a representation for a single word.[5] This mechanism is the key to creating rich, contextualized embeddings. An attention function is formally described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.[4] In the context of self-attention, these vectors are all derived from the same source: the input sequence itself.[11]

The process begins with the input sequence, where each token is represented by an embedding vector. For each of these input vectors, three new vectors are generated: a **Query** (Q), a **Key** (K), and a **Value** (V).[14] These vectors are created by multiplying the input embedding by three distinct, learned weight matrices (

WQ, WK, and WV), which are optimized during the model's training process.[5] These three vectors serve different roles in the attention calculation:

- The **Query** vector can be thought of as a representation of the current token, seeking information.
- The **Key** vectors of all tokens in the sequence act as labels or identifiers that the query can be compared against.
- The **Value** vectors contain the actual information or content of each token that will be passed on.

The calculation of attention scores then proceeds through a specific formula known as **Scaled Dot-Product Attention**, which is the foundational operation of the Transformer.[4] The formula is expressed as:

Attention(Q,K,V)=softmax(dkQKT)V
This computation can be broken down into several distinct steps:

1. **Compute Similarity Scores:** The first step is to calculate a score determining how much attention a token at a given position should pay to every other token in the sequence.

This is achieved by taking the dot product of the token's Query (Q) vector with the Key (K) vectors of all tokens in the sequence, represented by the matrix multiplication QK^T.[14] A higher dot product value between a query and a key indicates a higher degree of compatibility or relevance.[16]

2. **Scale the Scores:** The resulting scores are then scaled by dividing them by the square root of the dimension of the key vectors, denoted as d_k.[11] This scaling factor is a crucial but simple innovation that prevents the dot product scores from becoming excessively large, particularly with high-dimensional vectors. Large scores can push the softmax function into regions where its gradients are extremely small, which destabilizes and slows down the training process. The scaling ensures that the gradients remain manageable.[9]
3. **Normalize with Softmax:** The scaled scores are passed through a softmax function, which normalizes them into a probability distribution.[9] This converts the scores into positive values that sum to 1, which can be interpreted as the "attention weights".[18] These weights dictate how much of each token's "value" should be incorporated into the output representation of the current token.[16]
4. **Compute the Weighted Sum of Values:** Finally, the attention weights produced by the softmax function are multiplied by the Value (V) matrix. This step computes a weighted sum of the value vectors, where the weight assigned to each value is determined by its attention score.[4] The effect is to amplify the values of tokens that are highly relevant to the current token (those with high attention weights) while diminishing the influence of irrelevant tokens (those with low weights).[14]

The result of this process is an output vector for each token that is no longer just a representation of the token itself, but a contextualized representation that has incorporated information from the entire input sequence, weighted by relevance. This mechanism allows the model to resolve ambiguities and understand complex relationships, such as associating the pronoun "it" with "animal" in the sentence "The animal didn't cross the street because it was too tired".[14]

2.2 Enhancing Representational Power: Multi-Head Attention

While single self-attention is powerful, the Transformer architecture enhances this capability through a mechanism called **Multi-Head Attention**.[4] Instead of performing a single attention calculation with the full-dimensional Q, K, and V vectors, the model employs multiple parallel attention mechanisms, or "heads".[16]

The process works by first linearly projecting the original Q, K, and V matrices into lower-dimensional representations h times, using different, learned projection matrices for

each head.[4] This creates

h distinct sets of Q, K, and V matrices. The scaled dot-product attention function is then applied in parallel to each of these h sets, yielding h separate output vectors for each token.[4]

This parallel structure provides two key benefits:

1. **It expands the model's ability to focus on different positions and relationships simultaneously.** A single attention mechanism might be dominated by the token's relationship to itself or one other strong signal. With multiple heads, the model can concurrently attend to different parts of the input.[14]
2. **It gives the attention layer multiple "representation subspaces."** Each set of projection matrices is initialized randomly, and through training, each head learns to focus on different aspects of the input sequence.[11] For example, one head might learn to capture syntactic dependencies, another might track semantic relationships, and a third might focus on positional information.[14]

After the attention outputs from all h heads are calculated, they are concatenated together and passed through a final linear transformation (a learned weight matrix) to produce the final output of the multi-head attention layer.[4] This final projection combines the diverse information captured by each head into a single, enriched representation. By allowing the model to jointly attend to information from different representation subspaces at different positions, multi-head attention provides a more nuanced and powerful mechanism for understanding the intricate relationships within a sequence.[11] In the original Transformer paper, the authors used

h = 8 parallel attention heads.[14]

2.3 The Complete Picture: The Transformer Block

The self-attention and multi-head attention mechanisms are the core of the Transformer, but they are situated within a larger, repeating structure known as a **Transformer block** (or layer). Most modern LLMs are constructed by stacking many of these blocks sequentially.[2] Each block is designed not only to process relational information but also to ensure that the network can be trained effectively at great depths. A standard Transformer block contains two primary sub-layers [4]:

1. **A Multi-Head Self-Attention Mechanism:** This is the first sub-layer, which performs the contextual mixing and information routing between tokens as described above.
2. **A Position-wise Fully Connected Feed-Forward Network (FFN):** This is the second sub-layer. It consists of two linear transformations with a ReLU (Rectified Linear Unit)

activation function in between.[8] Unlike the attention mechanism, which processes the entire sequence at once, the FFN is applied independently and identically to each token's representation.[8] While the attention layer's role is to route and integrate information across the sequence, the FFN's role is to perform a non-linear transformation on each token's representation individually, further refining it and adding model capacity.[22]

These two sub-layers are connected by two additional crucial components: **residual connections** and **layer normalization**.[4]

- **Residual Connections:** A residual (or "skip") connection is implemented around each of the two sub-layers.[4] This means the input to a sub-layer is added directly to its output before being passed to the next stage. The operation is formally expressed as $x + \text{Sublayer}(x)$.[4] This technique, which was a key innovation in models like ResNet, is critical for training very deep neural networks. It helps mitigate the vanishing gradient problem by creating a "shortcut" for the gradient to flow through during backpropagation, ensuring that the model can learn effectively even with dozens or hundreds of layers.[23] This architectural choice embodies a philosophy where the default behavior is to preserve the original signal (x), and each sub-layer learns an additive refinement ($\text{Sublayer}(x)$) rather than a complete, potentially lossy, transformation.
- **Layer Normalization:** Following each residual connection, layer normalization is applied.[4] Layer normalization stabilizes the training process by normalizing the inputs to each layer across the features for a single data point, ensuring they have a consistent distribution.[24] The complete operation for each sub-layer is thus $\text{LayerNorm}(x + \text{Sublayer}(x))$.[4]

Finally, because the self-attention mechanism itself is permutation-invariant—meaning it would produce the same results if the words in a sentence were shuffled—the model requires an explicit way to understand word order. This is achieved through **Positional Encoding**.[9] Before the input sequence is fed into the first Transformer block, a vector representing each token's position is added to its corresponding token embedding.[2] The original paper proposed using a combination of sine and cosine functions of different frequencies to generate these unique positional vectors.[24] This provides the model with information about both the absolute and relative positions of tokens, which is essential for understanding the structure of language.[22]

Together, these components—multi-head attention for relational routing, FFNs for positional processing, residual connections and layer normalization for deep training, and positional encodings for sequence order—form the robust and highly effective building block that is replicated to create the deep architectures of modern LLMs.[2]

Section 3: The Great Divergence: Three Families of Transformer Architecture

The original Transformer model, as presented in "Attention Is All You Need," comprised both an encoder and a decoder, designed specifically for sequence-to-sequence tasks like machine translation.[26] However, in the years following its introduction, the research community discovered that by isolating and adapting these components, they could create specialized architectures optimized for different classes of NLP problems. This led to a divergence into three primary architectural families, each with its own design philosophy, pre-training objectives, and ideal applications.

3.1 Encoder-Only Architectures: Masters of Understanding (BERTology)

The first major family of models to emerge was the encoder-only architecture, famously exemplified by Google's BERT (Bidirectional Encoder Representations from Transformers).[27] These models discard the decoder stack of the original Transformer entirely, leveraging only the power of the encoder blocks.[28]

Architectural Deep Dive

The defining characteristic of encoder-only models is their use of **bidirectional self-attention**.[10] In the encoder stack, the self-attention mechanism is unmasked, meaning every token in the input sequence can attend to every other token, regardless of its position—both to the left and to the right.[32] This allows the model to build a deep, holistic understanding of the context of each word by considering the entire sentence at once. This deep bidirectionality is what makes these models exceptionally powerful for tasks that require a nuanced comprehension of language, but it also makes them inherently unsuitable for generating text in a sequential, left-to-right manner.[30]

Pre-training Objectives

To enable the model to learn from this bidirectional context in an unsupervised manner, BERT was pre-trained using two novel objectives [29]:

1. **Masked Language Modeling (MLM):** This is the core pre-training task for encoder-only models. Instead of predicting the next word in a sequence, a certain percentage of the input tokens (15% in the case of BERT) are randomly replaced with a special `` token. The model's objective is then to predict the original identity of these masked tokens based on the surrounding unmasked context.[29] This "cloze test" forces the model to fuse both the left and right context to make a prediction, thereby learning a rich, bidirectional representation of the language.[33]
2. **Next Sentence Prediction (NSP):** In this binary classification task, the model is presented with two sentences, A and B, and must determine if sentence B is the actual sentence that follows sentence A in the original text corpus. The training data consists of 50% true consecutive pairs and 50% random pairs.[29] This objective was designed to teach the model to understand relationships between sentences, which is crucial for tasks like question answering and natural language inference.[30]

Case Study: BERT

BERT was released in two primary sizes: BERT-Base, which has 12 encoder layers and 110 million parameters, and BERT-Large, with 24 layers and 340 million parameters.[29] The models were pre-trained on a large corpus comprising the Toronto BookCorpus (800 million words) and the English Wikipedia (2.5 billion words).[33]

Task Suitability

Due to their deep contextual understanding, encoder-only models are the preferred architecture for a wide range of Natural Language Understanding (NLU) tasks.[10] They are typically used as foundational models. After pre-training, the final encoder layer's output (a rich vector representation for each input token) is fed into a simple, task-specific classification head. The entire model is then fine-tuned on a smaller, labeled dataset for a specific downstream task, such as:

- **Sentiment Analysis and Text Classification:** Using the representation of the special `` token.[27]

- **Question Answering:** Taking a question and a passage as input and predicting the start and end tokens of the answer span within the passage.[30]
- **Named Entity Recognition (NER):** Classifying each token in a sequence (e.g., as a person, organization, or location).[30]

3.2 Decoder-Only Architectures: Engines of Generation (The GPT Lineage)

The second major architectural family is the decoder-only model, a lineage pioneered and popularized by OpenAI's Generative Pre-trained Transformer (GPT) series.[37] These models are constructed by using only the decoder stack from the original Transformer architecture, omitting the encoder entirely.[28]

Architectural Deep Dive

The defining feature of decoder-only models is their use of **causal self-attention**, also known as masked or unidirectional self-attention.[16] Within each decoder block, the self-attention mechanism is modified with a mask that prevents any token from attending to subsequent tokens in the sequence. A token at position

i can only attend to tokens at positions 1 to i.[16] This constraint ensures that the model's prediction for the next token can only depend on the tokens that have come before it. This

autoregressive property is fundamental to text generation, as the model must generate the output sequence one token at a time, without having access to "future" tokens that it has not yet produced.[20]

Pre-training Objective

The pre-training objective for decoder-only models is a classic language modeling task: **Next Token Prediction**.[22] The model is trained on vast quantities of unlabeled text data with the simple goal of predicting the next word in a sequence given all the preceding words.[39] Because every token in the training corpus provides a learning signal (as a label for the token

before it), this pre-training approach is highly sample-efficient compared to MLM.[42]

Case Study: The GPT Series

The GPT family began with GPT-1 (117 million parameters) and has seen a dramatic increase in scale with models like GPT-2 (1.5 billion parameters) and GPT-3 (175 billion parameters).[40] These models are trained on massive and diverse datasets scraped from the internet, including large portions of Common Crawl, WebText, books, and Wikipedia.[37] More recent iterations, such as GPT-4 and GPT-4o, are multimodal, meaning they can process and generate not only text but also other data types like images and audio.[37]

Task Suitability

The autoregressive nature of decoder-only models makes them exceptionally well-suited for **generative tasks**. They are the dominant architecture for applications that require producing fluent, coherent, and contextually relevant text, such as:

- **Conversational AI and Chatbots:** Powering systems like ChatGPT.[44]
- **Content Creation:** Generating articles, stories, marketing copy, and emails.[40]
- **Text Summarization and Translation:** Generating a summary or translation based on a provided text prompt.[46]
- **Code Generation:** Writing computer code from natural language descriptions.[40]

3.3 Encoder-Decoder Architectures: The Synthesis (T5 and BART)

The third family of models retains the full architecture of the original Transformer, utilizing both an encoder and a decoder stack.[2] These models, often referred to as sequence-to-sequence (seq2seq) models, aim to combine the strengths of both encoders and decoders to create a versatile architecture for a wide range of transformation tasks.[32]

Architectural Deep Dive

In an encoder-decoder model, the encoder first processes the entire input sequence using bidirectional self-attention, just like in an encoder-only model. This creates a rich, contextualized representation of the input.[26] The decoder then generates the output sequence autoregressively, token by token. The crucial link between the two components is the

cross-attention mechanism.[26] In addition to its own causal self-attention layer, each decoder layer contains a cross-attention layer where the Query vectors come from the decoder's own state, but the Key and Value vectors come from the final output of the encoder.[23] This allows the decoder, at each step of generation, to "look back" at the entire input sequence and focus its attention on the most relevant parts of the source text to inform its next prediction.[14]

Pre-training Objectives

The pre-training objectives for encoder-decoder models are designed to leverage this dual structure. Two prominent examples are:

1. **Text-to-Text Transfer Transformer (T5):** Developed by Google, T5 frames every NLP task as a text-to-text problem.[47] Its pre-training objective is a **span corruption** task. Random spans of tokens are removed from the input text and replaced with a single, unique sentinel token (e.g., <X>). The model is then trained to output the sequence of sentinel tokens followed by the text that was removed.[47] For fine-tuning, a task-specific prefix is added to the input (e.g., "translate English to German:..."), and the model generates the corresponding text output.[47] T5 was pre-trained on the Colossal Clean Crawled Corpus (C4), a 750 GB cleaned version of Common Crawl.[43]
2. **BART (Bidirectional and Auto-Regressive Transformer):** Developed by Meta, BART is framed as a **denoising autoencoder**.[49] It is pre-trained by taking a clean text document, corrupting it with an arbitrary noising function, and then training the model to reconstruct the original, uncorrupted document.[52] The noising functions can be diverse, including token masking (like BERT), token deletion, sentence permutation, and text infilling.[52] This approach effectively trains a BERT-like bidirectional encoder to understand the corrupted input and a GPT-like autoregressive decoder to generate the clean output, making it a powerful generalization of both architectures.[49]

Task Suitability

The flexible encoder-decoder structure makes these models highly effective for any task that requires mapping an input sequence to a new output sequence, especially when the input and output have different lengths or structures. They excel at:

- **Machine Translation:** The canonical seq2seq task.[10]
- **Text Summarization:** Condensing a long document into a shorter summary.[26]
- **Question Answering:** Generating an answer based on a given context document.[10]

The divergence of the original Transformer into these three distinct families was a necessary specialization driven by the fundamental tension between context representation and sequence generation. Encoder-only models maximize context understanding by sacrificing generative ability. Decoder-only models optimize for generation by sacrificing fully bidirectional context. Encoder-decoder models attempt to achieve a balance but introduce greater architectural complexity. The choice of architecture is therefore a direct consequence of which capability is prioritized for the intended application.

3.4 A Comparative Overview

The distinct design philosophies of the three main Transformer families result in a clear specialization of capabilities. The following table provides a concise, side-by-side comparison of their core architectural features, pre-training objectives, and primary applications.

Feature	Encoder-Only (e.g., BERT)	Decoder-Only (e.g., GPT)	Encoder-Decoder (e.g., T5, BART)
Core Components	Stack of Transformer Encoders	Stack of Transformer Decoders	Stacks of both Encoders and Decoders
Attention Mechanism	Bidirectional Self-Attention	Causal (Masked) Self-Attention	Bidirectional Self-Attention (Encoder), Causal Self-Attention & Cross-Attention (Decoder)

Information Flow	Parallel processing of entire input	Autoregressive (left-to-right)	Bidirectional encoding, then autoregressive decoding
Primary Pre-training Objective	Masked Language Modeling (MLM)	Next Token Prediction	Denoising / Span Corruption
Strengths	Deep contextual understanding, NLU tasks	Fluent text generation, generative tasks	Sequence-to-sequence transformation, flexibility
Weaknesses	Not suitable for text generation	Limited bidirectional context in input processing	Architectural complexity, potential bottlenecks
Flagship Models	BERT, RoBERTa, ELECTRA	GPT series, LLaMA, PaLM	T5, BART, Pegasus

While these architectural distinctions were stark in the early years of LLMs, the lines have begun to blur. The massive scale of modern decoder-only models, combined with advanced techniques like instruction tuning and sophisticated prompting, has enabled them to achieve surprisingly strong performance on traditional NLU tasks, challenging the established dominance of encoder-based models in that domain.[34] Similarly, instruction-tuned encoder-decoder models like Flan-T5 have demonstrated remarkable zero-shot and few-shot reasoning capabilities, rivaling much larger decoder-only models on complex benchmarks.[59] This suggests that at sufficient scale and with the right training methodologies, the constraints of the underlying architecture may become less pronounced than previously believed.

Section 4: Beyond the Standard Model: Innovations in LLM Architecture

As the standard Transformer architectures have scaled to immense sizes, their inherent computational and memory costs have become significant bottlenecks. The quadratic complexity of self-attention with respect to sequence length and the linear growth of computation with respect to parameter count have spurred research into more efficient architectural paradigms. This section explores two major innovations—Mixture-of-Experts and State-Space Models—that are redefining the trade-offs between model scale, performance, and efficiency.

4.1 Scaling with Sparsity: The Mixture-of-Experts (MoE) Approach

The Mixture-of-Experts (MoE) architecture is a powerful technique for dramatically increasing a model's parameter count without a proportional increase in computational cost.[61] It achieves this by introducing sparsity into the model's computations, activating only a fraction of the total parameters for any given input.

Technical Breakdown

In a standard Transformer, the feed-forward network (FFN) sub-layer within each block is a dense operation; every token is processed by the same set of weights. In an MoE model, this dense FFN layer is replaced with a set of n parallel FFNs, known as "experts".[61] Accompanying these experts is a small, trainable gating network, or "router".[62]

When a token enters the MoE layer, the router network calculates a score for each of the n experts and selects a small subset—typically the top k experts (where k is much smaller than n)—to process that specific token.[61] The outputs of these selected experts are then combined, often through a weighted sum where the weights are also determined by the router's scores, to produce the final output for that token.[65] This approach is known as a

Sparse MoE (SMoE) because the vast majority of experts remain inactive for any single token.[61]

The primary advantage of this design is the decoupling of model capacity from computational cost. The total number of parameters in the model can be scaled massively by increasing the number of experts, which enhances the model's ability to store knowledge. However, the computational cost (measured in FLOPs) for training and inference depends only on the number of *active* parameters—those of the k experts selected for each token.[61] This allows for the creation of models with hundreds of billions or even trillions of parameters that can be

trained and run with the computational budget of a much smaller dense model.[62] A key challenge in training MoE models is ensuring effective load balancing, where the router distributes tokens evenly across the experts to prevent some from being over-utilized while others are neglected.[61]

Case Study: Mixtral 8x7B

Mixtral 8x7B, developed by Mistral AI, is a prominent example of a high-performance open-weight SMoE model.[66] It is a decoder-only architecture where the FFN layer in each Transformer block is replaced by an MoE layer containing 8 distinct experts.[64] For every token at each layer, the router network selects the top 2 experts to process it.[67]

This design results in a model with a total of 46.7 billion parameters. However, during inference, only the parameters of the two selected experts, plus the shared self-attention parameters, are used for each token. This amounts to approximately 12.9 billion active parameters per token.[66] Consequently, Mixtral 8x7B delivers the inference speed and cost of a 13B dense model while leveraging the knowledge capacity of a 46.7B parameter model, allowing it to outperform much larger dense models like Llama 2 70B on many benchmarks.[66]

4.2 Breaking the Quadratic Barrier: State-Space Models (SSM)

While MoE addresses the challenge of scaling model parameters efficiently, another line of research has focused on tackling the other major bottleneck of the Transformer: the quadratic computational complexity of self-attention with respect to sequence length (O(N2)). State-Space Models (SSMs) have emerged as a promising alternative that offers linear scaling while maintaining strong performance.[69]

Technical Breakdown

SSMs are a class of models inspired by control theory that are designed for modeling sequences.[69] At their core, they function similarly to RNNs by maintaining a compressed hidden state that is updated at each timestep. This state is intended to capture a summary of the sequence's history, and the model's computation at each step involves updating this state

and producing an output.[69] A continuous SSM can be defined by the linear ordinary differential equations:

$$h'(t)=Ah(t)+Bx(t)$$
$$y(t)=Ch(t)+Dx(t)$$

where x(t) is the input, h(t) is the hidden state, y(t) is the output, and A,B,C,D are learned matrices that govern the system's dynamics.69
Unlike traditional RNNs, which suffer from vanishing gradients and are difficult to parallelize during training, modern SSMs have been developed with properties that overcome these issues. The **Mamba** architecture represents a significant breakthrough in this area.[71] The key innovation in Mamba is the introduction of a

Selective SSM. In this model, the state-update matrices (A,B,C) are not fixed but are instead made functions of the input data.[69] This selection mechanism allows the model to dynamically change how it updates its state based on the current token, enabling it to selectively remember relevant information and forget irrelevant context. This addresses a major weakness of prior SSMs and RNNs, which used time-invariant dynamics.[69]

Case Study: Mamba

The Mamba architecture replaces the complex attention and MLP blocks of a standard Transformer with a single, unified Mamba block built around the Selective SSM.[71] This design has several profound implications for efficiency:

- **Linear Time Complexity:** Mamba's computations scale linearly with sequence length, O(N), a dramatic improvement over the O(N2) complexity of standard attention.[69] This makes it exceptionally well-suited for processing extremely long sequences, on the order of one million tokens or more, which would be computationally prohibitive for a standard Transformer.[69]
- **Fast Inference:** During inference, Mamba operates in a recurrent mode. It only needs to maintain a fixed-size hidden state to generate the next token, rather than the ever-growing key-value (KV) cache required by Transformers. This leads to constant-time (O(1)) state updates per token and significantly higher inference throughput, especially for long sequences.[69]
- **Hardware-Aware Parallelism:** The Mamba algorithm is designed with a hardware-aware parallel scan that allows it to be trained efficiently on modern GPUs, overcoming the traditional parallelization challenges of recurrent models.[71]

Mamba and other SSMs represent a potential paradigm shift for applications requiring

long-context understanding, such as genomics, long-form document analysis, and high-resolution video processing, where the quadratic scaling of attention is a major limiting factor.[69]

4.3 The Next Frontier: A Look at Future Architectures

The innovations in MoE and SSMs are part of a broader exploration of architectures that move beyond the standard Transformer. Research is actively pursuing several other promising directions that target the fundamental limitations of current models.

One such direction is **diffusion-based LLMs**. Unlike autoregressive models that generate text token-by-token, diffusion models operate in a non-autoregressive, parallel fashion.[75] Models like LLaDA start with a sequence of masked tokens and iteratively refine the entire sequence at once, using a mask predictor to recover the original text.[75] This approach has the potential to dramatically reduce generation latency and offers more fine-grained control over the output, such as enforcing specific styles or topics.[75]

Another critical area of research is **memory-augmented architectures**. A key limitation of current Transformers is their static knowledge; they cannot learn new information at inference time without being retrained. Architectures like Titans aim to address this by integrating external memory mechanisms, allowing the model to persistently store and retrieve new information during inference.[75] This could be a crucial step towards models capable of continuous learning and deeper, long-term reasoning.

These research frontiers signal a move away from simply scaling up existing architectures and toward developing models that are fundamentally more efficient, controllable, and adaptable. While the Transformer remains dominant, these emerging paradigms are likely to play a critical role in the next evolution of artificial intelligence.

Section 5: A Comparative Analysis: Performance, Cost, and Scaling

The choice of an LLM architecture is not merely a theoretical exercise; it involves a complex series of trade-offs between model performance, computational cost, and scalability. This section provides a data-driven comparative analysis of the architectures discussed, using standardized benchmarks to quantify performance, examining the economic realities of

training and deployment, and exploring the scaling laws that guide the development of next-generation models.

5.1 Quantitative Benchmarking

Standardized benchmarks provide a crucial, albeit imperfect, lens through which to compare the capabilities of different LLM architectures. Performance on these benchmarks often reveals the inherent strengths and weaknesses tied to a model's design.

NLU Benchmarks (GLUE & SuperGLUE)

The General Language Understanding Evaluation (GLUE) and its more challenging successor, SuperGLUE, are suites of tasks designed to test a model's core language understanding capabilities, such as textual entailment, sentiment analysis, and coreference resolution.[77]

Historically, encoder-only models have excelled on these benchmarks. Their bidirectional attention mechanism allows them to form a deep contextual understanding of the input text, which is a significant advantage for these discriminative NLU tasks.[57] For instance, BERT-Large achieved an average score of 80.5 on the GLUE benchmark, setting a high standard at the time of its release.[81] Encoder-decoder models, which also possess a bidirectional encoder, have demonstrated equally strong, if not superior, performance. Google's T5-11B model achieved a score of 88.9 on SuperGLUE, approaching the human baseline of 89.8.[82] Similarly, Meta's BART was shown to match the performance of RoBERTa, a highly optimized encoder-only model, on both GLUE and the Stanford Question Answering Dataset (SQuAD).[50] More recent, efficient architectures like the Gated Linear Attention (GLA) Transformer show competitive, though slightly lower, accuracy on GLUE while exhibiting superior adversarial robustness.[84] Mamba-based models have also demonstrated competitive performance on GLUE, indicating their viability for NLU tasks.[85]

Knowledge & Reasoning Benchmark (MMLU)

The Massive Multitask Language Understanding (MMLU) benchmark is designed to evaluate a model's broad knowledge and reasoning abilities across 57 diverse subjects, including STEM,

humanities, and social sciences.[86] The questions are multiple-choice and range in difficulty from elementary to professional levels.[88]

On this benchmark, the advantage shifts dramatically towards large-scale, generative models. The sheer volume of knowledge encoded in the parameters of massive decoder-only and MoE models, especially when combined with instruction fine-tuning, allows them to excel. While the 175B parameter GPT-3 achieved a 5-shot accuracy of 43.9% [89], the much smaller but instruction-tuned Flan-T5-XXL (11B parameters) achieved a remarkable 75.2%, showcasing the power of fine-tuning on a diverse set of tasks.[60] The trend continues with the latest models: Mixtral 8x7B, an MoE model, matches or outperforms GPT-3.5.[66] State-of-the-art decoder-only models like GPT-4o and Claude 3.5 Sonnet have pushed performance even further, achieving scores of 88.7% and 88.3% respectively, which are very close to the estimated human expert baseline of around 89.8%.[88]

The benchmark results paint a clear picture of architectural specialization. For tasks that depend on a deep, nuanced understanding of a provided text (GLUE/SuperGLUE), the bidirectional context available to encoders is a distinct advantage. For tasks that require broad world knowledge and multi-step reasoning (MMLU), the massive parameter counts and generative pre-training of the largest decoder-only and MoE models are the dominant factors.

The following table consolidates performance data for several key models across these benchmarks, providing a direct comparison of their capabilities.

Model	Architecture	Parameters	GLUE (Avg)	SuperGLUE (Avg)	MMLU (5-shot %)
BERT-Large	Encoder-Only	340M	80.5	N/A	N/A
T5-11B	Encoder-Decoder	11B	N/A	88.9	48.9 (as UnifiedQA)
BART-Large	Encoder-Decoder	400M	Matches RoBERTa	N/A	N/A
Flan-T5-XXL	Encoder-Decoder	11B	N/A	N/A	75.2
GPT-3	Decoder-O	175B	72.8	71.8	43.9

	nly				
Llama 2 70B	Decoder-Only	70B	N/A	N/A	69.5
Mixtral 8x7B	MoE Decoder	46.7B (12.9B active)	N/A	N/A	70.6
GPT-4	Decoder-Only	N/A	N/A	N/A	86.4
GPT-4o	Decoder-Only (Multimodal)	N/A	N/A	N/A	88.7
Claude 3.5 Sonnet	Decoder-Only	N/A	N/A	N/A	88.3
Human Baseline	N/A	N/A	87.1	89.8	~89.8 (Expert)

5.2 The Economics of Scale: Computational Cost and Training Dynamics

The performance of modern LLMs is inextricably linked to the immense scale of the resources used to create them. Understanding the economics of this scale—the data, the compute, and the cost—is essential for appreciating the current landscape and the motivations behind architectural innovation.

Training Data

LLMs are pre-trained on vast corpora of text and code, often comprising trillions of tokens.[93]

The quality, diversity, and size of this data are critical determinants of a model's capabilities. Several key datasets have become foundational in the field:

- **Common Crawl:** A publicly available, petabyte-scale archive of raw web data crawled from billions of web pages. It is a primary source for many large models, including GPT-3 and T5.[43]
- **C4 (Colossal Clean Crawled Corpus):** A 750 GB English-language dataset created by Google by extensively cleaning and filtering a snapshot of the Common Crawl data. It was used to train the T5 model.[43]
- **BookCorpus and Wikipedia:** These are high-quality, well-structured corpora that were instrumental in training early influential models like BERT and the first GPT.[33] BookCorpus contains text from over 11,000 unpublished books, providing long-form narrative text.[43]

The process of curating these massive datasets is a significant undertaking, involving extensive cleaning to remove low-quality content, deduplication to reduce redundancy, and filtering to mitigate biases and remove harmful material.[43]

Computational Cost

The computational cost of training a state-of-the-art LLM is staggering and is often measured in **petaflop/s-days**—equivalent to performing 1015 floating-point operations per second for a full day.[97]

The training of GPT-3, with its 175 billion parameters, is estimated to have required approximately 3,640 petaflop-days of compute.[99] In monetary terms, this translates to costs ranging from an estimated $4.6 million to over $12 million for a single training run, depending on the hardware and cloud provider.[99] More recent and larger models have pushed these costs even higher; the training of GPT-4 reportedly cost over $100 million, and Google's Gemini Ultra is estimated to have cost $191 million in compute alone.[101] This immense financial barrier means that training foundational models from scratch is an endeavor feasible only for a small number of large technology corporations and well-funded research labs.[101]

The following table provides a summary of the scaling and computational costs for several landmark LLMs, illustrating the exponential growth in resource requirements over time.

Model	Architecture Family	Release Year	Parameters	Training Data Size (Tokens)	Training Data Composition	Estimated Training Compute (petaflop

					(Primary Sources)	-days)
BERT-Large	Encoder-Only	2018	340 Million	~3.3 Billion words	BookCorpus, English Wikipedia	N/A (Relatively low)
GPT-2	Decoder-Only	2019	1.5 Billion	~40 GB (WebText)	Web Scrape	N/A ($50k cost)
T5-11B	Encoder-Decoder	2020	11 Billion	~750 GB (C4)	Common Crawl (Cleaned)	~100
GPT-3	Decoder-Only	2020	175 Billion	300 Billion	Common Crawl, WebText 2, Books, Wikipedia	~3,640
PaLM	Decoder-Only	2022	540 Billion	780 Billion	Web docs, Books, Wikipedia, News, Code	N/A ($8M cost)
Llama 2	Decoder-Only	2023	70 Billion	2 Trillion	Publicly available online data	N/A
Mixtral 8x7B	MoE Decoder	2023	46.7 Billion	32k context	N/A	N/A (More efficient)

5.3 Architectural Trade-offs and Scaling Laws

The immense cost of training has driven research into understanding the fundamental principles that govern the relationship between architecture, scale, and performance. This has revealed critical trade-offs and "scaling laws" that now guide the design of new models.

Training Efficiency Trade-offs

A key trade-off exists between the pre-training efficiency of different architectures. Decoder-only models, trained on next-token prediction, are highly sample-efficient. For a sequence of length N, the model receives N-1 learning signals, as each token serves as a label for the prediction at the previous position.[42] In contrast, encoder-only models trained with MLM are far less efficient. If 15% of tokens are masked, the model only receives a learning signal from that small fraction of the input in each pass. This means a BERT-style model must process significantly more data or be trained for many more epochs to receive the same amount of learning signal as a GPT-style model, making it prohibitively expensive to scale MLM-based pre-training to hundreds of billions of parameters.[42] This fundamental difference in training efficiency is a primary reason for the market's convergence on large-scale decoder-only and MoE-decoder models for generative AI.

Scaling Laws

Empirical research has established that the performance of LLMs (as measured by their loss on a held-out test set) follows predictable power-law relationships with model size (number of parameters), dataset size (number of tokens), and the amount of compute used for training.[102]

A pivotal finding in this area is the "Chinchilla" scaling laws from DeepMind.[102] Previous research had suggested that for a given compute budget, model size should be scaled more aggressively than dataset size. The Chinchilla paper revised this, demonstrating through extensive experiments that for optimal performance, model size and training data size should be scaled in roughly equal proportions. They showed that their 70B parameter Chinchilla model, trained on 1.4 trillion tokens, outperformed the much larger 280B parameter Gopher

model, which was trained on only 300 billion tokens.[102]

These scaling laws have had a profound impact on the field, shifting the focus from simply building the largest possible models to a more balanced approach of training moderately-sized models on vast, high-quality datasets. Architectures like MoE and Mamba can be seen as attempts to alter these scaling laws by changing the fundamental relationship between parameters, computation, and performance, offering new, more efficient paths to achieving state-of-the-art results.

Section 6: Synthesis and Concluding Remarks

6.1 The Evolutionary Trajectory of LLM Architectures

The evolution of large language model architectures represents a rapid and remarkable journey of scientific and engineering innovation. The trajectory began with the inherent limitations of sequential models like RNNs, whose inability to parallelize and capture long-range dependencies created a clear ceiling for progress. The introduction of the Transformer in 2017 was not merely an incremental improvement but a paradigm shift, demonstrating that by eliminating recurrence and relying solely on a parallelizable self-attention mechanism, models could be scaled to sizes previously unimaginable.

This foundational breakthrough immediately led to a period of functional specialization. The original encoder-decoder blueprint was deconstructed into its constituent parts, giving rise to three distinct architectural families. Encoder-only models like BERT perfected the art of language understanding through bidirectional context, becoming the backbone of NLU tasks. Decoder-only models like GPT mastered language generation through an autoregressive, causal framework, powering the generative AI revolution. Encoder-decoder models like T5 and BART synthesized these two approaches, offering a flexible framework for sequence-to-sequence transformation tasks.

As these architectures were scaled to hundreds of billions of parameters, their own limitations became apparent. The quadratic complexity of attention and the immense computational cost of dense models spurred the latest wave of innovation. Architectures like Mixture-of-Experts and State-Space Models are not replacements for the Transformer but are targeted solutions to its specific scaling bottlenecks, offering new pathways to build more capable and efficient models by introducing principles of sparsity and linear-time processing.

6.2 The Current State: A Toolbox of Specialized Architectures

The current landscape of LLM architectures is not a monolithic one dominated by a single "best" design. Instead, the field has matured to a point where it offers a sophisticated toolbox of specialized architectures, each with a distinct profile of strengths, weaknesses, and resource requirements. The optimal choice is contingent upon the specific goals and constraints of the application.

- For tasks requiring deep, nuanced **understanding** of a provided text, such as fine-grained sentiment analysis or complex textual entailment, the bidirectional context of **encoder-only** models remains a powerful and efficient choice, particularly when fine-tuned on task-specific data.
- For open-ended **generation**, creativity, and conversational interaction, the autoregressive nature and massive knowledge capacity of large-scale **decoder-only** models are unparalleled. This family is the clear leader for chatbot and content creation applications.
- For structured **transformation** tasks that map a well-defined input sequence to a new output sequence, such as machine translation or summarization, the **encoder-decoder** architecture provides a robust and highly effective framework that explicitly models the relationship between source and target.
- For pushing the boundaries of **scale and knowledge capacity** while managing computational costs, the **Mixture-of-Experts** approach offers a proven path to building models with vast parameter counts that remain efficient in inference.
- For applications demanding the processing of extremely **long contexts**, such as analyzing entire books, codebases, or genomic sequences, the linear-time complexity of **State-Space Models** like Mamba presents a compelling and necessary alternative to the quadratic bottleneck of attention.

6.3 A Forward-Looking Perspective: Convergence or Divergence?

Looking ahead, the central question is whether the field will converge on a new, unified architecture that combines the strengths of these specialized designs, or if the trend of divergence and specialization will continue. While it is difficult to predict the exact path of innovation, current trends suggest a future characterized by hybridization and increased diversity.

We are already seeing the emergence of hybrid models that explicitly combine architectural

principles, such as Jamba, which interleaves Transformer and Mamba blocks to leverage the strengths of both attention and state-space mechanisms. The development of entirely novel paradigms, such as diffusion-based models for parallel text generation and memory-augmented networks for continuous learning, suggests that the exploration of post-Transformer architectures is still in its early stages.

Ultimately, the future of LLM architecture may not be about finding a single successor to the Transformer, but rather about developing a more diverse and interoperable ecosystem of models. The most sophisticated AI systems of the future may employ a "Mixture of Architectures" approach at a macro level—dynamically routing complex queries to the most suitable model or combination of models based on the task at hand. A request for a creative poem might be sent to a large decoder-only model, a legal document summarization to an encoder-decoder model with a long-context window, and a complex data analysis query to a system that leverages a reasoning-focused model. This vision suggests that the ongoing divergence of architectures is not a sign of fragmentation, but of a maturing field building a richer, more capable, and more efficient toolbox to tackle the ever-expanding frontier of artificial intelligence.

Cytowane prace

1. (PDF) Survey of Different Large Language Model Architectures: Trends, Benchmarks, and Challenges - ResearchGate, otwierano: sierpnia 26, 2025, https://www.researchgate.net/publication/383976933_Survey_of_different_Large_Language_Model_Architectures_Trends_Benchmarks_and_Challenges
2. How Transformers Work: A Detailed Exploration of Transformer ..., otwierano: sierpnia 26, 2025, https://www.datacamp.com/tutorial/how-transformers-work
3. Transformer-based Encoder-Decoder Models - Hugging Face, otwierano: sierpnia 26, 2025, https://huggingface.co/blog/encoder-decoder
4. Attention is All you Need - NIPS, otwierano: sierpnia 26, 2025, https://papers.neurips.cc/paper/7181-attention-is-all-you-need.pdf
5. What is self-attention? | IBM, otwierano: sierpnia 26, 2025, https://www.ibm.com/think/topics/self-attention
6. Survey and Evaluation of Converging Architecture in LLMs based on Footsteps of Operations - arXiv, otwierano: sierpnia 26, 2025, https://arxiv.org/html/2410.11381v1
7. What is an attention mechanism? | IBM, otwierano: sierpnia 26, 2025, https://www.ibm.com/think/topics/attention-mechanism
8. Transformer (deep learning architecture) - Wikipedia, otwierano: sierpnia 26, 2025, https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture)
9. Self - Attention in NLP - GeeksforGeeks, otwierano: sierpnia 26, 2025, https://www.geeksforgeeks.org/nlp/self-attention-in-nlp/
10. What is an encoder-decoder model? - IBM, otwierano: sierpnia 26, 2025, https://www.ibm.com/think/topics/encoder-decoder-model
11. Attention Is All You Need - Wikipedia, otwierano: sierpnia 26, 2025, https://en.wikipedia.org/wiki/Attention_Is_All_You_Need

12. Attention is All you Need - NIPS, otwierano: sierpnia 26, 2025, https://papers.nips.cc/paper/7181-attention-is-all-you-need
13. (PDF) Attention Is All You Need | Alicja - SciSpace, otwierano: sierpnia 26, 2025, https://scispace.com/papers/attention-is-all-you-need-2qmt11p7sij4
14. The Illustrated Transformer – Jay Alammar – Visualizing machine ..., otwierano: sierpnia 26, 2025, https://jalammar.github.io/illustrated-transformer/
15. Attention is all you need: Discovering the Transformer paper | Towards Data Science, otwierano: sierpnia 26, 2025, https://towardsdatascience.com/attention-is-all-you-need-discovering-the-transformer-paper-73e5ff5e0634/
16. A Deep Dive into the Self-Attention Mechanism of Transformers | by Shreya Srivastava | Analytics Vidhya | Medium, otwierano: sierpnia 26, 2025, https://medium.com/analytics-vidhya/a-deep-dive-into-the-self-attention-mechanism-of-transformers-fe943c77e654
17. The Illustrated Transformer From Scratch - Innovative Digital Transformation Jordan, otwierano: sierpnia 26, 2025, https://idtjo.hosting.acm.org/wordpress/the-illustrated-transformer/
18. What is a Transformer Model? - IBM, otwierano: sierpnia 26, 2025, https://www.ibm.com/think/topics/transformer-model
19. Understanding and Coding the Self-Attention Mechanism of Large Language Models From Scratch - Sebastian Raschka, otwierano: sierpnia 26, 2025, https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html
20. “The Illustrated Transformer” (Jay Alammar) in a NUT SHELL | by Learn With Vignesh | Aug, 2025 | Medium, otwierano: sierpnia 26, 2025, https://medium.com/@jersy718/the-illustrated-transformer-jay-alammar-in-a-nut-shell-3b5d27225270
21. (PDF) Attention is All you Need (2017) | Chat PDF - Nanonets, otwierano: sierpnia 26, 2025, https://nanonets.com/chat-pdf/attention-is-all-you-need
22. LLM Transformer Model Visually Explained - Polo Club of Data Science, otwierano: sierpnia 26, 2025, https://poloclub.github.io/transformer-explainer/
23. Transformer Architecture: Redefining Machine Learning Across NLP and Beyond - Toloka, otwierano: sierpnia 26, 2025, https://toloka.ai/blog/transformer-architecture/
24. Attention Is All You Need | by Xupeng Wang - Medium, otwierano: sierpnia 26, 2025, https://medium.com/@marvelous_catawba_otter_200/attention-is-all-you-need-f9fe38d6e2fc
25. The Illustrated Transformer: A Practical Guide | by Anote - Medium, otwierano: sierpnia 26, 2025, https://anote-ai.medium.com/the-illustrated-transformer-a-practical-guide-8fdc93963b89
26. Encoders and Decoders in Transformer Models - MachineLearningMastery.com, otwierano: sierpnia 26, 2025, https://machinelearningmastery.com/encoders-and-decoders-in-transformer-models/

27. BERT (language model) - Wikipedia, otwierano: sierpnia 26, 2025, https://en.wikipedia.org/wiki/BERT_(language_model)
28. Original transformer paper: Implementation of Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017. - GitHub, otwierano: sierpnia 26, 2025, https://github.com/brandokoch/attention-is-all-you-need-paper
29. BERT Model - NLP - GeeksforGeeks, otwierano: sierpnia 26, 2025, https://www.geeksforgeeks.org/nlp/explanation-of-bert-model-nlp/
30. A Complete Guide to BERT with Code | Towards Data Science, otwierano: sierpnia 26, 2025, https://towardsdatascience.com/a-complete-guide-to-bert-with-code-9f87602e4a11/
31. Chapter 3: Understanding Encoder and Decoder Models | by Radhika Ramsen | Medium, otwierano: sierpnia 26, 2025, https://medium.com/@radhikaramsen3131/chapter-3-understanding-encoder-and-decoder-models-152cf28db903
32. Transformer Architectures - Hugging Face LLM Course, otwierano: sierpnia 26, 2025, https://huggingface.co/learn/llm-course/chapter1/6
33. What is BERT? An Intro to BERT Models - DataCamp, otwierano: sierpnia 26, 2025, https://www.datacamp.com/blog/what-is-bert-an-intro-to-bert-models
34. Why are most LLMs decoder-only?. Dive into the rabbit hole of recent... | by Yumo Bai, otwierano: sierpnia 26, 2025, https://medium.com/@yumo-bai/why-are-most-llms-decoder-only-590c903e4789
35. Spring 2023 - Xinya Du, otwierano: sierpnia 26, 2025, https://xinyadu.github.io/cs6301/files/slides_pdf/06_BERT%20and%20Pretraining.pdf
36. BERT 101 - State Of The Art NLP Model Explained - Hugging Face, otwierano: sierpnia 26, 2025, https://huggingface.co/blog/bert-101
37. Generative pre-trained transformer - Wikipedia, otwierano: sierpnia 26, 2025, https://en.wikipedia.org/wiki/Generative_pre-trained_transformer
38. “Decoder-only” Transformer models still have an encoder...right? Otherwise how do they “understand” a prompt? - Reddit, otwierano: sierpnia 26, 2025, https://www.reddit.com/r/LanguageTechnology/comments/16nl811/decoderonly_transformer_models_still_have_an/
39. What Is GPT? GPT-4, GPT-5, and More Explained - Coursera, otwierano: sierpnia 26, 2025, https://www.coursera.org/articles/what-is-gpt
40. Introduction to Generative Pre-trained Transformer (GPT) - GeeksforGeeks, otwierano: sierpnia 26, 2025, https://www.geeksforgeeks.org/artificial-intelligence/introduction-to-generative-pre-trained-transformer-gpt/
41. Demystifying the Architecture of GPT Models: A Comprehensive Guide - 30 Days Coding, otwierano: sierpnia 26, 2025, https://30dayscoding.com/blog/understanding-the-architecture-of-gpt-models
42. [D] Why hasn't BERT been scaled up/trained on a massive dataset like GPT3? -

Reddit, otwierano: sierpnia 26, 2025, https://www.reddit.com/r/MachineLearning/comments/qklvfp/d_why_hasnt_bert_been_scaled_uptrained_on_a/
43. Open-Sourced Training Datasets for Large Language Models (LLMs) - Kili Technology, otwierano: sierpnia 26, 2025, https://kili-technology.com/large-language-models-llms/9-open-sourced-datasets-for-training-large-language-models
44. What is GPT AI? - Generative Pre-Trained Transformers Explained ..., otwierano: sierpnia 26, 2025, https://aws.amazon.com/what-is/gpt/
45. BERT vs GPT: Comparing the Two Most Popular Language Models - InvGate ITSM blog, otwierano: sierpnia 26, 2025, https://blog.invgate.com/gpt-3-vs-bert
46. GPT vs BERT. As natural language processing (NLP)... | by Abhishek Jain - Medium, otwierano: sierpnia 26, 2025, https://medium.com/@abhishekjainindore24/gpt-vs-bert-ba8f00ddda08
47. T5 (language model) - Wikipedia, otwierano: sierpnia 26, 2025, https://en.wikipedia.org/wiki/T5_(language_model)
48. A Unified Text-to-Text Framework for NLP Tasks: An Overview of T5 Model, otwierano: sierpnia 26, 2025, https://blog.paperspace.com/flan-t5-architecture/
49. BART Model for Text Auto Completion in NLP - GeeksforGeeks, otwierano: sierpnia 26, 2025, https://www.geeksforgeeks.org/artificial-intelligence/bart-model-for-text-auto-completion-in-nlp/
50. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension - Meta Research, otwierano: sierpnia 26, 2025, https://research.facebook.com/publications/bart-denoising-sequence-to-sequence-pre-training-for-natural-language-generation-translation-and-comprehension/
51. BART: Denoising Sequence-to-Sequence Pre-training for Natural ..., otwierano: sierpnia 26, 2025, https://arxiv.org/pdf/1910.13461
52. Understanding the BART Model for Accurate Text Summarization - DigitalOcean, otwierano: sierpnia 26, 2025, https://www.digitalocean.com/community/tutorials/bart-model-for-text-summarization-part1
53. Papers Explained 09: BART. BART is a denoising autoencoder built... | by Ritvik Rastogi | DAIR.AI | Medium, otwierano: sierpnia 26, 2025, https://medium.com/dair-ai/papers-explained-09-bart-7f56138175bd
54. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension - ACL Anthology, otwierano: sierpnia 26, 2025, https://aclanthology.org/2020.acl-main.703/
55. Guide to BART (Bidirectional & Autoregressive Transformer) - Analytics Vidhya, otwierano: sierpnia 26, 2025, https://www.analyticsvidhya.com/blog/2024/11/bart-model/
56. BART Explained - Papers With Code, otwierano: sierpnia 26, 2025, https://paperswithcode.com/method/bart

57. Improving the Language Understanding Capabilities of Large Language Models Using Reinforcement Learning - arXiv, otwierano: sierpnia 26, 2025, https://arxiv.org/html/2410.11020v3
58. The Comparison between the Encoder and the Decoder - Towards AI, otwierano: sierpnia 26, 2025, https://towardsai.net/p/machine-learning/the-comparison-between-the-encoder-and-the-decoder
59. Run T5 model 🦾 on 100k tokens 📚 20x faster ⚡ | by Knowledgator Engineering | Medium, otwierano: sierpnia 26, 2025, https://blog.knowledgator.com/run-t5-model-on-100k-tokens-20x-faster-5ac3aa405d3e
60. [2210.11416] Scaling Instruction-Finetuned Language Models - arXiv, otwierano: sierpnia 26, 2025, https://arxiv.org/abs/2210.11416
61. Applying Mixture of Experts in LLM Architectures | NVIDIA Technical ..., otwierano: sierpnia 26, 2025, https://developer.nvidia.com/blog/applying-mixture-of-experts-in-llm-architectures/
62. LLM Mixture of Experts Explained - TensorOps, otwierano: sierpnia 26, 2025, https://www.tensorops.ai/post/what-is-mixture-of-experts-llm
63. Mixture of Experts LLMs: Key Concepts Explained - neptune.ai, otwierano: sierpnia 26, 2025, https://neptune.ai/blog/mixture-of-experts-llms
64. Explaining the Mixture-of-Experts (MoE) Architecture in Simple Terms | by Gregory Zem, otwierano: sierpnia 26, 2025, https://medium.com/@mne/explaining-the-mixture-of-experts-moe-architecture-in-simple-terms-85de9d19ea73
65. What Is Mixture of Experts (MoE)? How It Works, Use Cases & More | DataCamp, otwierano: sierpnia 26, 2025, https://www.datacamp.com/blog/mixture-of-experts-moe
66. Mixtral of experts - Mistral AI, otwierano: sierpnia 26, 2025, https://mistral.ai/news/mixtral-of-experts
67. Mixtral 8x7B: A game-changing AI model by Mistral AI | SuperAnnotate, otwierano: sierpnia 26, 2025, https://www.superannotate.com/blog/mistral-ai-mixtral-of-experts
68. [2401.04088] Mixtral of Experts - arXiv, otwierano: sierpnia 26, 2025, https://arxiv.org/abs/2401.04088
69. Mamba Explained - The Gradient, otwierano: sierpnia 26, 2025, https://thegradient.pub/mamba-explained/
70. MAMBA and State Space Models Explained | by Astarag Mohapatra - Medium, otwierano: sierpnia 26, 2025, https://athekunal.medium.com/mamba-and-state-space-models-explained-b1bf3cb3bb77
71. Mamba (deep learning architecture) - Wikipedia, otwierano: sierpnia 26, 2025, https://en.wikipedia.org/wiki/Mamba_(deep_learning_architecture)
72. An Introduction to the Mamba LLM Architecture: A New Paradigm in Machine Learning, otwierano: sierpnia 26, 2025,

https://www.datacamp.com/tutorial/introduction-to-the-mamba-llm-architecture
73. Mamba LLM Architecture: A Breakthrough in Efficient AI Modeling | SaM Solutions, otwierano: sierpnia 26, 2025, https://sam-solutions.com/blog/mamba-llm-architecture/
74. [D] - Why MAMBA did not catch on? : r/MachineLearning - Reddit, otwierano: sierpnia 26, 2025, https://www.reddit.com/r/MachineLearning/comments/1hpg91o/d_why_mamba_did_not_catch_on/
75. Beyond Transformers: Promising Ideas for Future LLMs - Apolo.us, otwierano: sierpnia 26, 2025, https://www.apolo.us/blog-posts/beyond-transformers-promising-ideas-for-future-llms
76. Beyond Transformers: Promising Ideas for Future LLMs - theMind, otwierano: sierpnia 26, 2025, https://www.themind.io/blog-posts/beyond-transformers-promising-ideas-for-future-llms
77. GLUE Benchmark, otwierano: sierpnia 26, 2025, https://gluebenchmark.com/
78. GLUE Explained: Understanding BERT Through Benchmarks - Chris McCormick, otwierano: sierpnia 26, 2025, https://mccormickml.com/2019/11/05/GLUE/
79. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems - Alex Wang, otwierano: sierpnia 26, 2025, https://w4ngatang.github.io/static/papers/superglue.pdf
80. SuperGLUE Benchmark, otwierano: sierpnia 26, 2025, https://super.gluebenchmark.com/
81. A Conservative Estimate of Human Performance on the GLUE Benchmark - Nikita Nangia, otwierano: sierpnia 26, 2025, https://woollysocks.github.io/assets/GLUE_Human_Baseline.pdf
82. Google T5 algorithm scores 88.9 on SuperGLUE languge benchmark, compared to 89.8 human baseline. : r/linguistics - Reddit, otwierano: sierpnia 26, 2025, https://www.reddit.com/r/linguistics/comments/dmtr38/google_t5_algorithm_scores_889_on_superglue/
83. arXiv:2109.09193v1 [cs.CL] 19 Sep 2021, otwierano: sierpnia 26, 2025, http://arxiv.org/pdf/2109.09193
84. Towards Resilient and Efficient LLMs: A Comparative Study of Efficiency, Performance, and Adversarial Robustness - arXiv, otwierano: sierpnia 26, 2025, https://arxiv.org/html/2408.04585v3
85. Mamba-360: Survey of State Space Models as Transformer Alternative for Long Sequence Modelling: Methods, Applications, and Challenges - arXiv, otwierano: sierpnia 26, 2025, https://arxiv.org/html/2404.16112v1
86. What is MMLU? LLM Benchmark Explained and Why It Matters - DataCamp, otwierano: sierpnia 26, 2025, https://www.datacamp.com/blog/what-is-mmlu
87. Top LLM Benchmarks Explained: MMLU, HellaSwag, BBH, and Beyond - Confident AI, otwierano: sierpnia 26, 2025, https://www.confident-ai.com/blog/llm-benchmarks-mmlu-hellaswag-and-beyond

88. Exploring MMLU Benchmark for AI Models - Galileo AI, otwierano: sierpnia 26, 2025, https://galileo.ai/blog/mmlu-benchmark
89. Paper Breakdown #1: MMLU — LLMs Have Exams Too! A Post on Benchmarking - Medium, otwierano: sierpnia 26, 2025, https://medium.com/@alakarthika01/paper-breakdown-1-mmlu-llms-have-exams-too-a-post-on-benchmarking-a66630dfd2a6
90. Flan-T5: sweet results with the smaller, more efficient LLM - Graphcore, otwierano: sierpnia 26, 2025, https://www.graphcore.ai/posts/flan-t5-sweet-results-with-the-smaller-more-efficient-llm
91. GPT-4.1 vs GPT-4o MMLU Benchmark Comparison | Promptfoo, otwierano: sierpnia 26, 2025, https://www.promptfoo.dev/docs/guides/gpt-4.1-vs-gpt-4o-mmlu/
92. LLM Benchmarks: Overview, Limits and Model Comparison - Vellum AI, otwierano: sierpnia 26, 2025, https://www.vellum.ai/blog/llm-benchmarks-overview-limits-and-model-comparison
93. Common Corpus: The Largest Collection of Ethical Data for LLM Pre-Training - arXiv, otwierano: sierpnia 26, 2025, https://arxiv.org/html/2506.01732v1
94. Large Language Model Training in 2025 - Research AIMultiple, otwierano: sierpnia 26, 2025, https://research.aimultiple.com/large-language-model-training/
95. What are the best datasets for training NLP models? - Milvus, otwierano: sierpnia 26, 2025, https://milvus.io/ai-quick-reference/what-are-the-best-datasets-for-training-nlp-models
96. Large language model - Wikipedia, otwierano: sierpnia 26, 2025, https://en.wikipedia.org/wiki/Large_language_model
97. List of large language models - Wikipedia, otwierano: sierpnia 26, 2025, https://en.wikipedia.org/wiki/List_of_large_language_models
98. AI and compute | OpenAI, otwierano: sierpnia 26, 2025, https://openai.com/index/ai-and-compute/
99. [D] The cost of training GPT-3 : r/MachineLearning - Reddit, otwierano: sierpnia 26, 2025, https://www.reddit.com/r/MachineLearning/comments/hwfjej/d_the_cost_of_training_gpt3/
100. Navigating the High Cost of AI Compute | Andreessen Horowitz, otwierano: sierpnia 26, 2025, https://a16z.com/navigating-the-high-cost-of-ai-compute/
101. What is the cost of training large language models? - CUDO Compute, otwierano: sierpnia 26, 2025, https://www.cudocompute.com/blog/what-is-the-cost-of-training-large-language-models
102. Training Compute-Optimal Large Language Models, otwierano: sierpnia 26, 2025, https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf

103. Mistral AI (Mixtral-8x7B): Performance, Benchmarks, otwierano: sierpnia 26, 2025, https://arize.com/blog/mistral-ai