Transformer Architecture Variants: A Comprehensive Analysis of Efficiency and Performance Trade-offs

Abhishek Kumar, Lead Researcher, Valentis

Abstract

The introduction of the Transformer architecture in 2017 marked a paradigm shift in sequence modeling, primarily due to its parallelizable self-attention mechanism. However, the vanilla Transformer's self-attention operation exhibits time and memory complexity that scales quadratically with the input sequence length, posing a significant bottleneck for applications involving long contexts, such as document summarization, high-resolution image processing, and genomic analysis.[1] This scalability challenge has catalyzed a vibrant area of research focused on developing more efficient Transformer variants. This whitepaper provides a comprehensive analysis of these variants, categorizing them into distinct families based on their core efficiency strategies: sparse attention, linearized attention, recurrence mechanisms, and conditional computation. We conduct a detailed architectural comparison of seminal models within each category, including Longformer, BigBird, Linformer, Performer, Reformer, Transformer-XL, and the Switch Transformer. The analysis extends to a rigorous evaluation of their computational complexity, memory usage, and empirical performance on standardized benchmarks like the Long Range Arena (LRA). By synthesizing these theoretical and empirical trade-offs, this paper offers a structured guide for researchers and practitioners to navigate the complex landscape of efficient Transformers and select the most suitable architecture for their specific application constraints and performance requirements.[4]

1. Introduction

1.1 The Transformer Revolution and the Dawn of Self-Attention

The field of sequence transduction was fundamentally reshaped by the introduction of the Transformer architecture in the seminal paper, "Attention Is All You Need" (Vaswani et al., 2017).[6] Prior to this, the dominant models for sequence-based tasks were Recurrent Neural Networks (RNNs), including variants like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs).[8] These models process sequences token-by-token, maintaining a hidden state that carries information forward. While naturally suited for sequential data, this inherent recurrence created a fundamental constraint: the computation at each step depends on the completion of the previous step, severely limiting parallelization during training and making it difficult to capture very long-range dependencies due to issues like vanishing gradients.[8]

The Transformer proposed a radical departure from this paradigm. It dispensed entirely with recurrence and convolutions, relying solely on a mechanism known as self-attention.[11] Self-attention allows the model to weigh the influence of all other tokens in the input sequence when computing the representation for a single token.[13] This is achieved by mapping each input token to three vectors—a Query (Q), a Key (K), and a Value (V)—and calculating attention scores as the dot product of queries and keys.[14] The result is a weighted sum of the values, where the weights signify the relevance of each token to the current one. This all-to-all connectivity provides a constant path length between any two tokens in the sequence, enabling the model to capture long-range dependencies with unprecedented effectiveness.[11] By removing the sequential dependency, the Transformer architecture unlocked massive parallelization, allowing for training on much larger datasets and leading to state-of-the-art performance in machine translation and a host of other natural language processing (NLP) tasks.[6] This innovation laid the groundwork for the subsequent boom in large language models (LLMs) like BERT, GPT, and T5.[16]

1.2 The Scalability Challenge: Confronting Quadratic Complexity

The very architectural design that made the Transformer revolutionary—its parallelizable, non-recurrent self-attention mechanism—is also the source of its most significant limitation. The innovation of removing the sequential dependency of RNNs was achieved by allowing all tokens to interact simultaneously. This all-to-all interaction requires the computation of a pairwise attention score matrix of size $n \times n$, where n is the length of the input sequence.[1] The calculation

and storage of this matrix result in a computational and memory complexity of

O(n2).[1] While manageable for short sequences (e.g., the 512-token limit common in early models like BERT), this quadratic scaling becomes a prohibitive bottleneck as sequence length increases.[19]

For tasks involving long documents, high-resolution images (treated as sequences of pixels), or genomic data, where sequence lengths can extend into the thousands or millions, the O(n2) complexity makes the vanilla Transformer computationally infeasible.[2] The memory required to store the attention matrix exceeds the capacity of modern hardware accelerators, and the number of floating-point operations (FLOPs) leads to intractable training times. This fundamental scalability challenge has effectively barred the standard Transformer from a wide range of important problem domains and has become the central motivating factor for the development of a new class of "efficient" Transformer architectures. These variants seek to preserve the powerful context-modeling capabilities of self-attention while mitigating its quadratic cost, navigating the foundational trade-off between parallelizability and computational complexity.

1.3 An Overview of Efficiency-Focused Architectural Variants

In response to the scalability challenge, a diverse ecosystem of Transformer variants has emerged, each proposing a unique strategy to approximate or modify the self-attention mechanism to achieve sub-quadratic complexity. These approaches can be broadly classified into several distinct families, providing a conceptual map of the solution space.[5]

One major family of models introduces **Sparse Attention**, which constrains each token to attend to only a subset of other tokens. This effectively creates a sparse attention matrix, reducing the number of computations. Some methods impose fixed, position-based sparsity patterns, such as local windows or global "summary" tokens, while others use content-based techniques to dynamically determine which tokens should attend to each other.[23]

A second family employs **Linearized Attention**, which seeks to approximate the full attention matrix using mathematical techniques that yield linear, or near-linear, complexity. These methods often leverage kernelization to decompose the softmax function or use low-rank factorization to project the Key and Value matrices into a lower-dimensional space, thereby avoiding the explicit computation of the full n×n matrix.[23]

A third approach reintroduces a form of **Recurrence and Memory** at a higher level. Instead of

processing a long sequence in one pass, these models divide it into segments and use a caching mechanism to pass information from one segment to the next, effectively extending the context length beyond the model's fixed window size.[23]

Finally, some models leverage **Conditional Computation** via Mixture-of-Experts (MoE) layers. This strategy decouples the number of model parameters from the computational cost per input by activating only a small, specialized subset of the model's weights for each token.[5] This allows for the creation of models with trillions of parameters that maintain a constant computational cost for inference.

This whitepaper will systematically analyze representative models from each of these families, deconstructing their architectures and evaluating the trade-offs they introduce between efficiency, performance, and modeling capabilities.

2. The Vanilla Transformer and Its Computational Constraints

To fully appreciate the innovations of efficient Transformer variants, it is essential to first establish a detailed understanding of the original architecture and the precise origins of its computational bottleneck. The model proposed by Vaswani et al. (2017) is an elegant composition of several key components that work in concert to process sequential data without recurrence.[6]

2.1 Core Architectural Components

The vanilla Transformer is an encoder-decoder model, a structure common in sequence transduction tasks like machine translation.[10] The encoder maps an input sequence of symbol representations,

(x1,...,xn), to a sequence of continuous representations, z=(z1,...,zn). The decoder then uses z to generate an output sequence, (y1,...,ym), one token at a time in an auto-regressive manner.[25]

Encoder-Decoder Stacks: Both the encoder and decoder are composed of a stack of N=6

identical layers.[12] Each encoder layer contains two primary sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network (FFN). The decoder inserts a third sub-layer, which performs multi-head attention over the encoder's output, allowing the decoder to focus on relevant parts of the input sequence. Crucially, each sub-layer is wrapped in a residual connection followed by layer normalization, formulated as

LayerNorm(x+Sublayer(x)).[12] This stabilizes training in deep networks.

Scaled Dot-Product Attention: The core of the Transformer is its attention mechanism. An attention function can be described as mapping a query and a set of key-value pairs to an output.[12] The Transformer uses a specific implementation called scaled dot-product attention. The input consists of queries

Q, keys K of dimension dk, and values V of dimension dv. The output is computed as:

Attention(Q,K,V)=softmax(dkQKT)V
The dot products of the query with all keys determine the attention weights. The scaling factor, 1/dk, is a critical component that counteracts the tendency for dot products to grow large in magnitude, which could push the softmax function into regions with vanishingly small gradients, thereby impeding learning.[14] In self-attention layers,

Q, K, and V are all derived from the same input sequence (i.e., the output of the previous layer), allowing each position to attend to all other positions in the sequence.[10]

Multi-Head Attention: Rather than performing a single attention function, the Transformer employs multi-head attention. This mechanism linearly projects the queries, keys, and values h times with different, learned linear projections.[6] Attention is then applied to each of these projected versions in parallel, creating

h different "heads." The outputs of these heads are concatenated and projected again to produce the final output. This allows the model to jointly attend to information from different representation subspaces at different positions. For instance, one head might focus on syntactic relationships while another focuses on semantic ones.[12] In the base model,

h=8 heads are used, with the dimensionality of each head reduced such that the total computational cost is similar to that of single-head attention with the full dimension.[25]

Positional Encoding: Because the self-attention mechanism is permutation-invariant—it has no inherent sense of sequence order—the model must be explicitly provided with information about the relative or absolute position of tokens.[15] The original Transformer injects this information

using positional encodings, which are vectors added to the input embeddings at the bottom of the encoder and decoder stacks. The paper proposed using sine and cosine functions of different frequencies for this purpose, as this formulation could theoretically allow the model to extrapolate to sequence lengths longer than those seen during training.[8]

2.2 A Formal Analysis of $O(n^2)$ Complexity in Self-Attention

The computational and memory constraints of the vanilla Transformer originate directly from the scaled dot-product attention calculation. The central operation, as established, is the matrix multiplication QKT. Let us consider a single attention head where the input sequence has length n and the hidden dimension is d. The query matrix Q and key matrix K both have dimensions n×dk.

Computational Complexity: The matrix multiplication QKT involves multiplying an n×dk matrix by a dk×n matrix. This operation results in an n×n attention score matrix and requires O(n2dk) floating-point operations. The subsequent multiplication of this n×n attention matrix by the n×dv value matrix V requires an additional O(n2dv) operations. Since all other operations (scaling, softmax) are element-wise or have lower complexity, the dominant factor is the matrix multiplication, leading to an overall time complexity of O(n2d) per layer, where d is the model dimension.[1]

Memory Complexity: The primary memory bottleneck is the need to explicitly store the n×n attention score matrix to compute the gradients during backpropagation. This matrix contains n2 floating-point numbers, resulting in a memory requirement of O(n2).[1] For a sequence of length 64,000 tokens, this matrix alone would require over 16 GB of memory in 32-bit precision, exceeding the capacity of most contemporary hardware accelerators for even a single training example.[28]

The explicit materialization of this n×n attention matrix is the singular source of the quadratic bottleneck. Consequently, all efficient Transformer variants can be understood as strategies to circumvent the need to compute or store this full matrix. They achieve this either by computing only a sparse subset of its entries or by refactoring the mathematical operations to arrive at the final output without ever forming the intermediate n×n product. This unified perspective reveals that the diverse landscape of efficient Transformers is, at its core, a collection of sophisticated solutions to this one central problem. Furthermore, for very long sequences, the bottleneck can shift from raw computational power (FLOPs) to memory bandwidth—the rate at which data can be moved to and from the processing units—making data movement a critical factor in overall

efficiency.[29]

3. A Taxonomy of Efficient Attention Mechanisms

The proliferation of research aimed at overcoming the Transformer's quadratic bottleneck has produced a wide array of architectural variants, often referred to as X-formers.[22] To navigate this complex landscape, it is useful to organize these models into a taxonomy based on their fundamental approach to improving efficiency. This classification provides a structured framework for understanding the core principles, assumptions, and trade-offs inherent in each family of models.[5]

3.1 Sparse Attention: Imposing Structural Priors

The most direct approach to reducing the O(n2) complexity is to limit the number of token pairs for which attention scores are computed. Sparse attention methods achieve this by assuming that most of the information required to compute a token's representation comes from a small, localized subset of other tokens. This effectively replaces the dense, fully-connected attention graph with a sparse one, reducing the number of edges from O(n2) to a more manageable O(nlogn) or O(n).[18] These methods can be further divided based on how the sparsity pattern is determined.

Position-based (Fixed Patterns): In this sub-category, the attention pattern is pre-defined and fixed based on the relative positions of the tokens. Common patterns include a sliding window (or local) attention, where each token attends only to its immediate neighbors; a dilated (or strided) window, which skips tokens to increase the receptive field; and global attention, where a few pre-selected tokens (e.g., the `` token) are allowed to attend to the entire sequence.[23] Models like

Longformer and **BigBird** are prominent examples that combine these fixed patterns to balance local context modeling with global information integration.[23]

Content-based (Learned Patterns): Instead of using fixed patterns, these methods determine the sparsity pattern dynamically based on the input sequence itself. The goal is to allow tokens to

attend to other tokens that are semantically relevant, regardless of their position. A popular technique is Locality-Sensitive Hashing (LSH), used in the **Reformer** model, which groups similar query and key vectors into buckets and restricts attention computation to within these buckets.[18] Other approaches use techniques like clustering to identify relevant token groups.[23]

3.2 Linearized Attention: Approximating the Attention Matrix

A second major family of models tackles the efficiency problem from a different angle. Instead of sparsifying the attention matrix, they aim to approximate the full, dense attention computation with an operation that has linear, or near-linear, complexity. These methods are often grounded in mathematical principles from linear algebra and kernel methods.[2]

Low-Rank Factorization: This approach is based on the hypothesis that the $n \times n$ attention matrix is often low-rank, meaning it can be well-approximated by the product of two smaller matrices. The **Linformer** model operationalizes this by projecting the Key (K) and Value (V) matrices, both of size $n \times d$, down to a smaller, fixed dimension $k \times d$, where $k \ll n$. The attention is then computed with these projected matrices, reducing the complexity to $O(nk)$.[4]

Kernelization: These methods view the scaled dot-product attention as a kernel function. The standard softmax attention, $\text{softmax}(QK^T)$, can be seen as an exponential kernel. By replacing or approximating this kernel with one that can be decomposed, the order of matrix multiplication can be changed. Instead of computing $(QK^T)V$, which involves the problematic $n \times n$ intermediate matrix, these methods compute $Q(K^TV)$. This avoids the quadratic bottleneck. The **Performer** model uses random feature maps to provably approximate the softmax kernel, while the **Linear Transformer** replaces the softmax with a simpler feature map, both achieving linear complexity.[23]

3.3 Recurrence and Memory: Extending Context Beyond Fixed Lengths

This category of models reintroduces a form of statefulness, reminiscent of RNNs, but at a higher architectural level. The motivation is to enable the model to handle contexts that are much longer than the maximum sequence length it can process in a single forward pass. Instead of trying to fit the entire sequence into memory at once, these models process it in segments.[23]

The seminal model in this category is **Transformer-XL**. During the processing of the current segment, Transformer-XL caches the hidden states from the previous segment and reuses them as an extended context. This creates a segment-level recurrence mechanism that allows information to propagate over arbitrarily long distances, resolving the context fragmentation problem of the vanilla Transformer and enabling much longer-term dependency modeling.[31]

3.4 Conditional Computation: Scaling Parameters with Mixture-of-Experts (MoE)

While the previous categories focus on reducing the computational cost for a fixed model size, Mixture-of-Experts (MoE) models address a related but distinct problem: how to dramatically increase the number of model parameters (i.e., its capacity to store knowledge) without a corresponding increase in the computational cost per input.[24]

In an MoE Transformer, the dense feed-forward network (FFN) in each layer is replaced with a collection of smaller FFNs, called "experts." For each input token, a lightweight, trainable "gating network" or "router" dynamically selects a small subset of these experts (often just one or two) to process the token.[24] The outputs of the selected experts are then combined. This means that while the model may have trillions of total parameters, only a fraction of them are activated for any given input, resulting in a constant computational cost. The

Switch Transformer is a key example that simplifies this paradigm by using an efficient top-1 routing strategy, demonstrating that massive models can be trained with significantly less compute than their dense counterparts.[35]

These distinct families of models exist on a spectrum of design philosophies. Sparse attention methods, for instance, introduce strong, often handcrafted, inductive biases about the nature of dependencies in data (e.g., the importance of locality). In contrast, linearized methods make mathematical assumptions about the attention matrix or the softmax kernel but attempt to create a more general-purpose approximation of the original, highly expressive mechanism. This trade-off between imposing structural biases for efficiency versus creating a more universal and accurate approximation is a recurring theme in the design of these advanced architectures.

4. Architectural Analysis of Key Transformer Variants

This section provides a detailed examination of the mechanisms within the most influential and representative models from each category of the taxonomy. By deconstructing their core innovations, we can better understand their respective strengths, weaknesses, and underlying assumptions.

4.1 Pattern-Based Sparsity: Longformer and BigBird

Longformer and BigBird represent the pinnacle of fixed-pattern sparse attention, offering practical and effective solutions for processing long documents. Though developed concurrently, they arrived at remarkably similar architectural conclusions, highlighting the robustness of combining local and global attention patterns.

Longformer: The Longformer model directly addresses the quadratic complexity by replacing the full self-attention with a sparse attention mechanism that scales linearly with sequence length, O(n).[21] Its attention pattern is a combination of two components that reflect common intuitions about information in long texts.[37]

1. **Sliding Window (Local) Attention:** For most tokens, attention is restricted to a fixed-size window of neighboring tokens, w. Each token attends to w/2 tokens on each side. This captures local context, which is crucial for tasks like syntactic analysis and local semantic disambiguation. By stacking multiple layers, the receptive field of the top layers grows, allowing for the integration of information over a wider, though still constrained, range.[38] To further increase this receptive field without adding computational cost, Longformer can use
   dilated sliding windows, where the window has gaps, allowing it to attend to tokens at varying distances.[37]
2. **Global Attention:** To handle tasks that require a summary of the entire sequence, such as text classification or question answering, Longformer designates a few tokens to have global attention. A token with global attention attends to all other tokens in the sequence, and all other tokens attend back to it. These global tokens are typically task-specific; for example, the `` token in classification tasks or all question tokens in QA tasks are given global attention.[19] This provides a mechanism for aggregating information from the entire document into specific locations.

BigBird: The BigBird model also proposes a sparse attention mechanism that reduces

complexity to linear time while preserving the theoretical properties of the full Transformer.[3] Its architecture is composed of three key attention patterns that, when combined, provide a robust approximation of full attention.[40]

1. **Window Attention:** Similar to Longformer, BigBird includes a local sliding window attention where each token attends to its immediate neighbors. This captures the principle of locality of reference.[41]
2. **Random Attention:** To ensure that information can propagate between distant parts of the sequence, each token also attends to a small, fixed number of *randomly* selected tokens. This component is inspired by graph theory, which shows that random connections in a sparse graph can drastically reduce the average path length between any two nodes, thereby facilitating efficient information flow across the entire sequence.[40]
3. **Global Attention:** Like Longformer, BigBird incorporates global tokens that can attend to and be attended by all other tokens. This can be implemented by making some existing tokens global (Internal Transformer Construction) or by adding special tokens like `` for this purpose (Extended Transformer Construction).[41]

A key differentiator for BigBird is its strong theoretical foundation. The authors prove that their sparse attention mechanism is a universal approximator of sequence functions and is Turing complete, meaning it retains the full expressive power of the original Transformer architecture, a guarantee not provided by most other sparse methods.[3] The convergence of Longformer's empirical design and BigBird's theoretically motivated one on a "local + global" pattern suggests this combination is a highly effective heuristic for long-sequence modeling. BigBird's addition of random attention can be viewed as a method to formalize and guarantee the long-range connectivity that was more implicitly handled in Longformer.

4.2 Low-Rank and Kernelized Approximations: Linformer and Performer

This family of models achieves linear complexity not by sparsifying attention but by fundamentally reformulating the underlying mathematics. Linformer and Performer exemplify two distinct but powerful approaches: one based on assumptions about the data's structure and the other on the properties of the attention operation itself.

Linformer: The core hypothesis of Linformer is that the n×n context mapping matrix, $P=\text{softmax}(QK^T/d_k)$, is low-rank.[4] This implies that the information contained within this large matrix is compressible and can be captured in a much smaller space. Linformer exploits this by

introducing two linear projection matrices,

Ei and Fi, that project the Key and Value matrices from their original sequence length n to a much smaller, fixed dimension k.[44] The attention is then computed using these

k×d projected matrices instead of the original n×d ones. The resulting complexity is O(nk), which is linear in sequence length n because k is a hyperparameter independent of n.[4] This approach achieves linearity by projecting the problem into a smaller dimensional space before the expensive dot-product operation. The success of Linformer suggests that for many NLP tasks, the full-rank attention matrix contains significant redundancy, and its essential information can be effectively captured by a low-rank approximation.[45]

Performer: The Performer model achieves linear complexity without making any assumptions about the sparsity or low-rankness of the attention matrix.[30] Instead, it focuses on approximating the softmax function itself using a technique from kernel methods. The attention mechanism can be viewed as a kernel,

K(qi,kj)=exp(qiTkj/dk). The Performer introduces the FAVOR+ (Fast Attention Via Positive Orthogonal Random Features) mechanism, which approximates this kernel using random feature maps, $\phi(\cdot)$.[47] This allows the attention computation to be decomposed and reordered. Instead of calculating the

n×n matrix (QKT), Performer computes the product in a different order: Q′(K′TV), where Q′ and K′ are the query and key matrices transformed by the random feature map. This reordering avoids the creation of the intermediate n×n matrix, resulting in linear space and time complexity.[30] Performer is notable for its strong theoretical guarantees, providing an unbiased or nearly-unbiased estimation of the full attention matrix with low variance.[46] The contrasting approaches of Linformer and Performer reveal two distinct pathways to linearity: one exploits the perceived structure of the

data (low-rank attention scores), while the other exploits the mathematical structure of the *operation* (an approximable kernel).

4.3 Hashing-Based Attention and Memory Optimization: Reformer

The Reformer model introduces two orthogonal techniques to improve Transformer efficiency, one targeting computational complexity and the other memory usage, making it particularly

well-suited for training very deep models on extremely long sequences.[50]

Locality-Sensitive Hashing (LSH) Attention: To reduce the O(n2) computation of the dot-product attention, Reformer uses LSH to group similar tokens together.[18] The core idea is that in the softmax computation, the output is dominated by the largest dot-product scores. Therefore, for a given query, one only needs to consider the keys that are "closest" to it in the high-dimensional space. LSH is an efficient algorithm for finding these approximate nearest neighbors. In LSH attention, queries and keys are assigned to hash buckets, and attention is only computed between tokens that fall into the same bucket.[51] To ensure that queries and keys that are close to each other are hashed together, Reformer uses a shared-QK formulation, where the query and key projections are identical.[51] This approach reduces the complexity from

O(n2) to O(nlogn).[52]

Reversible Residual Layers: To tackle the immense memory cost of storing activations for backpropagation in deep networks (an O(N) factor, where N is the number of layers), Reformer employs reversible residual networks.[51] In a standard residual block, the output is

y=x+F(x), and x must be stored to compute gradients for F. A reversible layer, however, uses two sets of activations, (x1,x2), and updates them according to y1=x1+F(x2) and y2=x2+G(y1). With this formulation, the inputs (x1,x2) can be recovered from the outputs (y1,y2) during the backward pass by simply running the operations in reverse: x2=y2−G(y1) and x1=y1−F(x2).[51] This eliminates the need to store intermediate activations for every layer, dramatically reducing memory usage and making the memory cost independent of the number of layers.[28]

4.4 Integrating Recurrence: Transformer-XL

Transformer-XL addresses the limitations of the vanilla Transformer's fixed-length context, particularly in auto-regressive language modeling. The standard approach of processing a long text is to break it into fixed-size segments, but this leads to *context fragmentation*, where information cannot flow across segment boundaries, and the model must re-learn context at the beginning of each new segment.[31]

Segment-Level Recurrence: Transformer-XL introduces a recurrence mechanism at the segment level.[54] When processing the current segment, say

sτ, the model computes attention over the tokens within that segment as well as the tokens from

the *previous* segment, sτ−1. The hidden states computed for segment sτ−1 are cached and reused as an extended context for segment sτ. This creates a recurrent connection between segments, allowing information to propagate over much longer distances than the fixed segment length. This effectively creates a much larger context for the model to draw upon, enabling it to learn longer-term dependencies and generate more coherent text.[31]

Relative Positional Encoding: The standard absolute positional encoding scheme of the vanilla Transformer is not viable in this recurrent setting. If absolute positions were used, the model would receive conflicting positional information for the same token index across different segments. To resolve this, Transformer-XL introduces a novel and more sophisticated **relative positional encoding scheme**. Instead of encoding the absolute position of a token, this scheme encodes the relative distance between pairs of tokens (i.e., the offset between a key and a query). This makes the attention score calculation sensitive to relative position while remaining consistent across different segments, which is a critical innovation for enabling the segment-level recurrence mechanism.[32]

4.5 Efficient Scaling with Sparsely-Gated Experts: Switch Transformer

The Switch Transformer tackles the scaling problem from the perspective of model capacity. It is a type of Mixture-of-Experts (MoE) model designed for simplicity and efficiency, enabling the training of models with over a trillion parameters.[35]

Switch Layer: Traditional MoE models often route each token to the top-k experts, combining their outputs. This adds complexity to the routing logic and communication patterns in distributed settings. The Switch Transformer simplifies this dramatically by using a **top-1 gating** strategy.[35] The model replaces the dense FFN layer in a Transformer block with a "Switch" layer containing multiple FFN "experts." A lightweight, learnable router network takes a token's representation and outputs a probability distribution over the available experts. The token is then sent to only the single expert with the highest probability.[58]

Benefits and Challenges: This simplified routing reduces communication overhead and computational complexity compared to top-k routing.[59] The key advantage of the MoE approach is that it decouples the total number of model parameters from the FLOPs required per token. A model can have a vast number of parameters distributed across its experts, but the computational cost for a single forward pass remains constant, as only one expert is activated per token.[60] However, this approach introduces new challenges, most notably

load balancing. If the router consistently sends most tokens to a few "popular" experts, other experts will be under-trained and computational resources will be used inefficiently. To mitigate this, the Switch Transformer incorporates an auxiliary loss function that encourages the router to distribute tokens uniformly across all experts.[24] It also defines an "expert capacity"—a maximum number of tokens an expert can process in a batch—to manage buffer sizes in hardware. If an expert's capacity is exceeded, tokens are dropped and passed to the next layer via the residual connection.[35]

5. Comparative Analysis of Efficiency and Performance

While architectural descriptions provide insight into the mechanisms of efficient Transformers, a comprehensive understanding requires a direct comparison of their empirical performance and resource consumption. Standardized benchmarks, particularly those designed for long-context scenarios, are crucial for this evaluation.

5.1 Performance on Long-Context Benchmarks: The Long Range Arena (LRA)

The Long Range Arena (LRA) was introduced as a systematic and unified benchmark to evaluate the quality of efficient Transformer models specifically under long-context scenarios, with sequence lengths ranging from 1,000 to 16,000 tokens.[45] The benchmark is designed to be challenging, simple, and general, probing diverse capabilities such as hierarchical reasoning, spatial awareness, and retrieval across multiple data modalities without relying on pre-training.[45] The LRA suite includes several key tasks:

- **Long ListOps:** Evaluates modeling of hierarchical structures.
- **Byte-level Text Classification:** Tests performance on long documents (IMDb reviews).
- **Byte-level Document Retrieval:** Assesses the ability to match document similarity.
- **Image Classification:** Treats images as long sequences of pixels.
- **Pathfinder:** A synthetic task requiring the model to find connections between points in an image, testing long-range spatial dependency modeling.
- **Pathfinder-X:** An extreme version of Pathfinder with 16K sequence length.

The evaluation of ten prominent efficient Transformer variants on the LRA benchmark yielded

several critical findings.[45] Most notably, no single model emerged as a universal winner across all tasks, underscoring that different architectures possess distinct inductive biases that make them better suited for certain types of problems.

BigBird achieved the highest average accuracy across the tasks, demonstrating consistently strong, if not top-tier, performance, suggesting it is a robust generalist.[45] Kernel-based models like

Performer excelled on tasks like text and image classification but struggled significantly on the hierarchical ListOps task. Conversely, sparse models with explicit global tokens, like BigBird, performed well on retrieval tasks, where compressing a document's information into an aggregated representation is key. A stark finding was the universal failure of all models on the Pathfinder-X task; performance dropped to random chance, indicating that even these efficient architectures have a breaking point at extreme sequence lengths.[45]

5.2 A Comparative Study of Computational and Memory Efficiency

A direct comparison of the theoretical complexities and measured resource consumption of these models provides a clear picture of the efficiency gains they offer over the vanilla Transformer. The LRA benchmark provided a side-by-side analysis of speed and memory usage, which, combined with the theoretical complexities, allows for a comprehensive evaluation.[45]

Table 1 provides a summary of the theoretical complexities and core assumptions of the key Transformer variants. It illustrates the different pathways to sub-quadratic performance, from the O(nlogn) of Reformer's LSH-based approach to the true linear O(n) complexity of models like Longformer, BigBird, and the linearized attention variants.

Table 1

Complexity Comparison of Transformer Variants

Model	Core Technique	Time Complexity	Memory Complexity	Key Limitations/Assumptions
Vanilla	Full	O(n2)	O(n2)	Prohibitive for

Transformer	Self-Attention			long sequences.
Longformer	Sliding Window + Global Attention	O(n)	O(n)	Assumes locality is sufficient; global tokens must be pre-specified.
BigBird	Window + Random + Global Attention	O(n)	O(n)	Relies on random patterns to approximate global connectivity.
Reformer	Locality-Sensitive Hashing (LSH)	O(nlogn)	O(nlogn)	LSH is an approximation; performance depends on hash quality.
Linformer	Low-Rank Projection	O(n)	O(n)	Assumes the attention matrix is inherently low-rank.
Performer	Kernelization (Random Features)	O(n)	O(n)	Approximates the softmax kernel; accuracy depends on # of features.
Transformer-	Segment-Level	$O(n \cdot L2)$	$O(n \cdot L)$	Computation is

XL	Recurrence			quadratic within each segment of length L.
Switch Transformer	Mixture-of-Experts (MoE)	O(n)	O(n)	Constant FLOPs per token, but high total parameter memory.

Note. n = sequence length, L = segment length. Complexity for Transformer-XL is for total sequence length n processed in segments of length L.

Empirically, the LRA benchmark found that kernel-based models were the fastest in practice. **Performer** was measured to be 5.7 times faster than a vanilla Transformer on a 4K sequence, closely followed by **Linformer** and Linear Transformers.[45] In terms of memory,

Linformer was the most efficient, using nearly 10 times less memory than the vanilla Transformer. These results confirm that the theoretical efficiency gains translate into substantial practical improvements in resource consumption.[63]

5.3 Synthesizing the Trade-offs: A Multi-faceted Evaluation

The LRA results, when viewed in conjunction with the architectural designs, reveal a clear pattern of specialization. The benchmark did not simply rank models; it exposed their inherent strengths and weaknesses, which are direct consequences of their underlying assumptions.

- **Sparse Models (BigBird, Longformer):** The success of these models on tasks like document retrieval suggests that their explicit "aggregator" global tokens are highly effective for tasks requiring the compression of an entire sequence into a fixed representation for comparison. Their strong, generalist performance across the board indicates that the "local + global" attention pattern is a very robust heuristic for a wide range of long-sequence tasks.[45]
- **Kernel-based Models (Performer, Linformer):** Their strong performance on text and image classification but poor performance on the hierarchical ListOps task suggests that

their approximation methods are excellent for capturing semantic similarity and spatial patterns but may lose some of the precise, fine-grained relational information needed for structured, algorithmic tasks. This highlights a trade-off between general approximation quality and the ability to model rigid, symbolic structures.[45]

- **Reformer:** The LSH-based approach of Reformer was found to be slower in practice on the LRA benchmarks than other linear methods, suggesting that the theoretical O(nlogn) complexity may come with larger constant factors that make it less practical at moderate sequence lengths (e.g., 4K) compared to true O(n) methods.[45]

Ultimately, the analysis shows there is no universally superior efficient Transformer. The choice of model involves a nuanced trade-off between raw performance on a given task, computational speed, memory footprint, and the validity of the model's underlying assumptions for the problem domain at hand.

6. Application Suitability and Practitioner Recommendations

The theoretical and empirical analysis of Transformer variants provides a foundation for making informed decisions in practical applications. This section translates the preceding analysis into actionable guidance for researchers and engineers, mapping architectural strengths to specific problem domains and offering a framework for model selection based on common constraints.

6.1 Mapping Architectures to Problem Domains

Different efficient Transformer architectures have demonstrated particular aptitude for certain types of tasks, often as a direct result of their design principles.

- **Long Document Processing (NLP):** For tasks such as summarization, question answering, and classification on lengthy texts like legal documents, scientific articles, or books, models with explicit local and global attention patterns are highly suitable. **Longformer** and **BigBird** were specifically designed for this domain and have consistently shown strong performance. Their ability to process thousands of tokens in a single pass, while maintaining designated global tokens to aggregate document-level information, makes them a primary choice.[38]

- **Genomics and Biological Sequence Analysis:** Processing DNA or protein sequences often involves identifying patterns within extremely long inputs. Both **BigBird** and **Performer** have been successfully applied in this domain. BigBird's combination of local, random, and global attention can capture the varied dependencies in genomic data, while Performer's kernel-based approach provides a robust, theoretically grounded method for handling such lengths without specific structural assumptions.[30]
- **Generative Modeling and Language Modeling:** For tasks requiring the generation of long, coherent sequences of text, maintaining temporal consistency is paramount. **Transformer-XL**, with its segment-level recurrence mechanism, excels in this area. By caching past hidden states, it avoids context fragmentation and can generate text that is coherent over thousands of tokens, far surpassing the capabilities of vanilla Transformers.[31] **Reformer** has also been demonstrated on generative tasks with very long sequences, leveraging its memory efficiency to handle contexts of up to one million tokens.[51]
- **Massive-Scale Models:** When the goal is to build state-of-the-art models with hundreds of billions or trillions of parameters, the primary challenge is managing the sheer scale. **Switch Transformer** and other Mixture-of-Experts (MoE) models are the definitive solution in this space. By decoupling the number of parameters from the computational cost per token, they allow for the training of immensely large models with a fraction of the computational budget required for a dense model of equivalent performance, representing the frontier of large-scale AI.[35]

The following table synthesizes these recommendations into a suitability matrix, providing a quick-reference guide for practitioners.

Table 2

Application Suitability Matrix for Transformer Variants

Model	Long-Doc Summarization	Long-Doc QA	Genomics	Generative LM	Extreme Scale Training	Low-Memory Environments
Longformer	Excellent	Excellent	Good	Fair	Poor	Good
BigBird	Excellent	Excellent	Excellent	Fair	Poor	Good

Linformer	Good	Good	Fair	Fair	Poor	Excellent
Performer	Good	Good	Excellent	Good	Fair	Excellent
Reformer	Good	Good	Good	Excellent	Fair	Excellent
Transformer-XL	Fair	Fair	Fair	Excellent	Poor	Fair
Switch Transformer	Good	Good	Good	Good	Excellent	Poor

6.2 A Guideline for Model Selection

Choosing the right architecture involves balancing the needs of the application with available resources. The following framework provides a set of questions to guide the selection process:

1. **What is the target sequence length?**
    - **Short (< 512 tokens):** A vanilla Transformer (e.g., BERT, RoBERTa) is often sufficient and highly optimized. The overhead of efficient attention mechanisms may not be beneficial.
    - **Medium (512 - 4,096 tokens):** This is the sweet spot for models like Longformer and BigBird, which are designed to handle this range effectively for document-level tasks. Linformer and Performer also offer significant speed and memory advantages here.
    - **Very Long (> 4,096 tokens):** For extremely long sequences, models like Reformer, Performer, and Transformer-XL become necessary. Reformer's memory-saving techniques are particularly valuable for scaling to very deep models at these lengths.
2. **What are the primary hardware and memory constraints?**
    - **High Memory Constraints:** If memory is the primary bottleneck, **Linformer** and **Performer** are excellent choices due to their linear memory complexity and small practical footprint. **Reformer**, with its reversible layers, is unparalleled for training

very deep models where activation storage is the main issue.[45]
    - **High Parameter Count (VRAM):** MoE models like **Switch Transformer** require significant memory to hold all expert parameters, even though only a fraction are used during inference. This makes them unsuitable for environments with limited VRAM.[24]
3. **What are the specific requirements of the task?**
    - **Need for Full-Sequence Summary:** If the task (e.g., classification) requires aggregating information from across the entire sequence into a single representation, models with explicit global tokens like **Longformer** and **BigBird** have a built-in architectural advantage.
    - **Approximation of Full Attention:** If the goal is to approximate the behavior of a full Transformer on a long sequence without imposing strong structural biases, **Performer** is a strong candidate due to its provable approximation guarantees.[48]
    - **Auto-regressive Generation:** For generating long, coherent sequences, the stateful, recurrent nature of **Transformer-XL** is specifically designed to prevent context fragmentation and is often the best choice.[54]
4. **Is the priority training speed or inference speed?**
    - **Training Speed:** MoE models like **Switch Transformer** offer dramatic speed-ups in pre-training time-to-quality compared to dense models.[60]
    - **Inference Speed:** Kernel-based models like **Performer** and **Linformer** have shown exceptional inference speed on long sequences. MoE models also have fast inference in terms of FLOPs, but their large memory footprint can be a practical challenge for deployment.[24]

By systematically considering these factors, practitioners can move beyond a one-size-fits-all approach and select the Transformer variant that best aligns with the unique demands of their project.

7. Conclusion and Future Directions

The evolution of the Transformer architecture from its groundbreaking inception to the diverse ecosystem of efficient variants represents a remarkable trajectory of innovation in deep learning. The initial model, while revolutionary, carried within its design the seeds of its own limitation: an O(n2) complexity that constrained its application to relatively short sequences. The subsequent explosion of research has effectively addressed this bottleneck from multiple angles, yielding a suite of specialized tools rather than a single, universal successor.

Our analysis has shown that these variants are not merely incremental improvements but represent distinct design philosophies. Sparse attention models like Longformer and BigBird impose strong, intuitive structural priors, betting on the importance of local and global context. Linearized attention models like Linformer and Performer attack the problem through mathematical approximation, leveraging principles of low-rank factorization and kernel theory. Recurrent models like Transformer-XL reintroduce statefulness to elegantly manage context beyond fixed windows, while conditional computation models like the Switch Transformer redefine scalability by decoupling parameter count from computational cost.

The key takeaway for practitioners and researchers is that the selection of a Transformer architecture is now a nuanced, multi-faceted decision. There is no longer a single "best" model, but rather a "best-fit" model for a given set of constraints and objectives. The trade-offs are clear: the handcrafted efficiency of sparse patterns versus the theoretical robustness of kernel approximations; the memory savings of reversible layers versus the generative coherence of recurrence; the massive capacity of MoE models versus their high memory footprint. Standardized benchmarks like the Long Range Arena have been instrumental in illuminating these trade-offs, revealing that architectural strengths are often task-dependent.

Looking forward, several trends are poised to shape the next generation of sequence models.

- **Hybrid Models:** The architectural dichotomy between Transformers and other paradigms is beginning to blur. Emerging models like Mamba, which is based on State Space Models (SSMs), are demonstrating that combining Transformer-like components with alternative mechanisms like structured state-space representations can yield highly competitive performance with linear complexity and efficient inference.[70] The future likely lies in these hybrid designs that synthesize the best properties of different approaches.
- **Benchmark Evolution:** While the LRA was a critical step forward, recent analyses suggest that its tasks may not fully capture the nuances of long-range dependency, as performance can sometimes be dominated by short-range cues.[72] The development of new, more challenging benchmarks that force models to engage in complex, multi-step reasoning over long contexts will be essential for driving further progress.
- **Hardware-Software Co-design:** As models become more complex, the interplay between architecture and hardware becomes increasingly critical. True efficiency is not just a matter of reducing theoretical FLOPs but of designing models that are aware of hardware constraints like memory bandwidth and on-chip cache sizes. Future architectures will likely be co-designed with hardware in mind to minimize data movement and maximize the utilization of parallel processing units, a trend already visible in techniques like FlashAttention.[29]

In conclusion, the journey from the vanilla Transformer to its many efficient descendants has

transformed a powerful but constrained architecture into a versatile and scalable toolkit. The focus has shifted from a singular pursuit of performance to a sophisticated balancing act between accuracy, efficiency, and applicability. The continued exploration of hybrid architectures, more rigorous benchmarks, and hardware-aware design will undoubtedly push the boundaries of what is possible in modeling long-range dependencies, opening up new frontiers in science and technology.

References

Beltagy, I., Peters, M. E., & Cohan, A. (2020). *Longformer: The long-document transformer*. arXiv preprint arXiv:2004.05150. [74]

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., & Weller, A. (2021). Rethinking attention with performers. In *International Conference on Learning Representations*. [30]

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). *Transformer-XL: Attentive language models beyond a fixed-length context*. arXiv preprint arXiv:1901.02860. [31]

Fedus, W., Zoph, B., & Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research, 23*(120), 1-39. [35]

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). *Reformer: The efficient transformer*. arXiv preprint arXiv:2001.04451. [28]

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., & Metzler, D. (2021). Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*. [45]

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems 30* (pp. 5998-6008). [84]

Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020). *Linformer: Self-attention with*

linear complexity. arXiv preprint arXiv:2006.04768. [87]

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2020). Big bird: Transformers for longer sequences. In *Advances in neural information processing systems 33* (pp. 17283-17297). [39]

Works cited

1. On The Computational Complexity of Self-Attention, accessed on September 8, 2025, https://arxiv.org/abs/2209.04881
2. Efficient Attention Mechanisms for Large Language Models: A Survey - arXiv, accessed on September 8, 2025, https://arxiv.org/pdf/2507.19595
3. Big Bird: Transformers for Longer Sequences - arXiv, accessed on September 8, 2025, https://arxiv.org/pdf/2007.14062
4. Linformer: Self-Attention with Linear Complexity - arXiv, accessed on September 8, 2025, https://arxiv.org/pdf/2006.04768
5. Speed Always Wins: A Survey on Efficient Architectures for Large Language Models - arXiv, accessed on September 8, 2025, https://www.arxiv.org/abs/2508.09834
6. "Attention is All You Need" Summary - Medium, accessed on September 8, 2025, https://medium.com/@dminhk/attention-is-all-you-need-summary-6f0437e63a91
7. Attention is All you Need - NIPS, accessed on September 8, 2025, https://papers.nips.cc/paper/7181-attention-is-all-you-need
8. Attention Is All You Need (Vaswani et al., ArXiv 2017) | Jonathan K. Kummerfeld, accessed on September 8, 2025, https://jkk.name/reading-notes/old-blog/2017-10-20_onlyattention/
9. Transformer (deep learning architecture) - Wikipedia, accessed on September 8, 2025, https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture)
10. Attention Is All You Need - arXiv, accessed on September 8, 2025, https://arxiv.org/html/1706.03762v7
11. Attention Is All You Need | Request PDF - ResearchGate, accessed on September 8, 2025, https://www.researchgate.net/publication/317558625_Attention_Is_All_You_Need
12. Attention is All you Need - NIPS, accessed on September 8, 2025, https://papers.neurips.cc/paper/7181-attention-is-all-you-need.pdf
13. The Illustrated Transformer: A Practical Guide | by Anote - Medium, accessed on September 8, 2025, https://anote-ai.medium.com/the-illustrated-transformer-a-practical-guide-8fdc93963b89
14. Attention Is All You Need - Wikipedia, accessed on September 8, 2025, https://en.wikipedia.org/wiki/Attention_Is_All_You_Need
15. Transformer Survey - Hannes Stärk, accessed on September 8, 2025, https://hannes-stark.com/assets/transformer_survey.pdf
16. The Illustrated Retrieval Transformer - Jay Alammar, accessed on September 8, 2025, https://jalammar.github.io/illustrated-retrieval-transformer/
17. Jay Alammar – Visualizing machine learning one concept at a time., accessed on September 8, 2025, https://jalammar.github.io/
18. The Problem with Quadratic Attention in Transformer Architectures | tips - Wandb,

accessed on September 8, 2025, https://wandb.ai/wandb_fc/tips/reports/The-Problem-with-Quadratic-Attention-in-Transformer-Architectures--Vmlldzo3MDE0Mzcz
19. [2004.05150] Longformer: The Long-Document Transformer - ar5iv - arXiv, accessed on September 8, 2025, https://ar5iv.labs.arxiv.org/html/2004.05150
20. arXiv:2202.07856v2 [cs.CL] 11 Feb 2023, accessed on September 8, 2025, https://arxiv.org/pdf/2202.07856
21. Paper page - Longformer: The Long-Document Transformer - Hugging Face, accessed on September 8, 2025, https://huggingface.co/papers/2004.05150
22. [2106.04554] A Survey of Transformers - arXiv, accessed on September 8, 2025, https://arxiv.org/abs/2106.04554
23. A Survey of Transformers - arXiv, accessed on September 8, 2025, https://arxiv.org/pdf/2106.04554
24. Mixture of Experts Explained - Hugging Face, accessed on September 8, 2025, https://huggingface.co/blog/moe
25. Attention is All you Need - NIPS, accessed on September 8, 2025, https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
26. The Illustrated Transformer – Jay Alammar – Visualizing machine ..., accessed on September 8, 2025, https://jalammar.github.io/illustrated-transformer/
27. LLM Transformer Model Visually Explained - Polo Club of Data Science, accessed on September 8, 2025, https://poloclub.github.io/transformer-explainer/
28. [2001.04451] Reformer: The Efficient Transformer - ar5iv, accessed on September 8, 2025, https://ar5iv.labs.arxiv.org/html/2001.04451
29. The Evolution of Transformer Architecture: From 2017 to 2024 | by arghya mukherjee, accessed on September 8, 2025, https://medium.com/@arghya05/the-evolution-of-transformer-architecture-from-2017-to-2024-5a967488e63b
30. (PDF) Rethinking Attention with Performers (2021) | Krzysztof Choromanski | 102 Citations, accessed on September 8, 2025, https://scispace.com/papers/rethinking-attention-with-performers-1bxectc8vh
31. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context - ar5iv - arXiv, accessed on September 8, 2025, https://ar5iv.labs.arxiv.org/html/1901.02860
32. transformer-xl: attentive language models beyond a fixed-length context - AMiner, accessed on September 8, 2025, https://static.aminer.cn/misc/pdf/weixin/TRANSFORMER-XL.pdf
33. A Survey on Mixture of Experts - arXiv, accessed on September 8, 2025, https://arxiv.org/html/2407.06204v2
34. Mixture of experts - Wikipedia, accessed on September 8, 2025, https://en.wikipedia.org/wiki/Mixture_of_experts
35. Switch Transformers: Scaling to Trillion Parameter Models with ..., accessed on September 8, 2025, https://arxiv.org/abs/2101.03961
36. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity – Related Work – Interesting papers - Alastair Reid, accessed on September 8, 2025, https://alastairreid.github.io/RelatedWork/papers/fedus:arxiv:2021/
37. Longformer: The Long-Document Transformer, accessed on September 8, 2025,

https://arxiv.org/abs/2004.05150
38. Longformer: Long Sequence Transformer - Ultralytics, accessed on September 8, 2025, https://www.ultralytics.com/glossary/longformer
39. Big Bird: Transformers for Longer Sequences | Request PDF - ResearchGate, accessed on September 8, 2025, https://www.researchgate.net/publication/343279169_Big_Bird_Transformers_for_Longer_Sequences
40. Big Bird: Transformers for Longer Sequences, accessed on September 8, 2025, https://arxiv.org/abs/2007.14062
41. Review — Big Bird: Transformers for Longer Sequences | by Sik-Ho Tsang - Medium, accessed on September 8, 2025, https://sh-tsang.medium.com/brief-review-big-bird-transformers-for-longer-sequences-12ccd3430e3b
42. Big Bird: Transformers for Longer Sequences, accessed on September 8, 2025, https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf
43. [R] Linformer: Self-Attention with Linear Complexity : r/MachineLearning - Reddit, accessed on September 8, 2025, https://www.reddit.com/r/MachineLearning/comments/h0eup6/r_linformer_selfattention_with_linear_complexity/
44. Brief Review — Linformer: Self-Attention with Linear Complexity | by Sik-Ho Tsang | Medium, accessed on September 8, 2025, https://sh-tsang.medium.com/brief-review-linformer-self-attention-with-linear-complexity-d87fce25fe8f
45. [2011.04006] Long Range Arena: A Benchmark for Efficient ..., accessed on September 8, 2025, https://ar5iv.labs.arxiv.org/html/2011.04006
46. (PDF) Rethinking Attention with Performers - ResearchGate, accessed on September 8, 2025, https://www.researchgate.net/publication/344436871_Rethinking_Attention_with_Performers
47. accessed on January 1, 1970, https://arxiv.org/pdf/2009.14794
48. Rethinking Attention with Performers - OpenReview, accessed on September 8, 2025, https://openreview.net/forum?id=Ua6zuk0WRH
49. [2009.14794] Rethinking Attention with Performers - ar5iv - arXiv, accessed on September 8, 2025, https://ar5iv.labs.arxiv.org/html/2009.14794
50. Reformer: The Efficient Transformer - ResearchGate, accessed on September 8, 2025, https://www.researchgate.net/publication/338569863_Reformer_The_Efficient_Transformer
51. Reformer: The Efficient Transformer, accessed on September 8, 2025, https://arxiv.org/pdf/2001.04451
52. Reformer: The Efficient Transformer - BibSonomy, accessed on September 8, 2025, https://www.bibsonomy.org/bibtex/2afcb7d1d8971f1dc55e2816c8b3235e6/stdiff
53. Reformer: The Efficient Transformer | OpenReview, accessed on September 8, 2025, https://openreview.net/forum?id=rkgNKkHtvB
54. accessed on January 1, 1970, https://arxiv.org/abs/1901.02860

55. arXiv:2505.00929v1 [cs.LG] 2 May 2025, accessed on September 8, 2025, https://arxiv.org/pdf/2505.00929
56. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context | Request PDF, accessed on September 8, 2025, https://www.researchgate.net/publication/335781169_Transformer-XL_Attentive_Language_Models_beyond_a_Fixed-Length_Context
57. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, accessed on September 8, 2025, https://www.alphaxiv.org/overview/2101.03961v3
58. Scaling to Trillion Parameter Models With Switch Transformers | by Zia Babar - Medium, accessed on September 8, 2025, https://medium.com/@zbabar/scaling-to-trillion-parameter-models-with-switch-transformers-88ca5fb95e5c
59. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, accessed on September 8, 2025, https://www.semanticscholar.org/paper/Switch-Transformers%3A-Scaling-to-Trillion-Parameter-Fedus-Zoph/fdacf2a732f55befdc410ea927091cad3b791f13
60. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, accessed on September 8, 2025, https://www.researchgate.net/publication/348403003_Switch_Transformers_Scaling_to_Trillion_Parameter_Models_with_Simple_and_Efficient_Sparsity
61. Long Range Arena for Benchmarking Efficient Transformers - GitHub, accessed on September 8, 2025, https://github.com/google-research/long-range-arena
62. [2011.04006] Long Range Arena: A Benchmark for Efficient Transformers - arXiv, accessed on September 8, 2025, https://arxiv.org/abs/2011.04006
63. A Benchmark for Comparing Different AI Transformers - DeepLearning.AI, accessed on September 8, 2025, https://www.deeplearning.ai/the-batch/transformer-variants-head-to-head/
64. The big picture: Transformers for long sequences | by Lukas Nöbauer - Medium, accessed on September 8, 2025, https://medium.com/@lukas.noebauer/the-big-picture-transformers-for-long-sequences-890cc0e7613b
65. Longformer model in NLP - Medium, accessed on September 8, 2025, https://medium.com/@gremwang/longformer-model-in-nlp-8721f33a7f11
66. Bigbird Pegasus Large Arxiv · Models - Dataloop, accessed on September 8, 2025, https://dataloop.ai/library/model/google_bigbird-pegasus-large-arxiv/
67. Reformer: Efficient Transformer for Long Sequences | Ultralytics, accessed on September 8, 2025, https://www.ultralytics.com/glossary/reformer
68. Reformer: The Efficient Transformer - Google Research, accessed on September 8, 2025, https://research.google/blog/reformer-the-efficient-transformer/
69. Demystifying Sparse Attention: Longformer, BigBird, Reformer, and Linformer Explained | by Boopathi Raj | Medium, accessed on September 8, 2025, https://medium.com/@rajboopathiking/demystifying-sparse-attention-longformer-bigbird-reformer-and-linformer-explained-029b97588144
70. I've devised a potential transformer-like architecture with O(n) time complexity, reducible

to O(log n) when parallelized. - Reddit, accessed on September 8, 2025, https://www.reddit.com/r/compsci/comments/1flkwb2/ive_devised_a_potential_transformerlike/
71. Transformers Are Getting Old: Variants and Alternatives Exist! - Hugging Face, accessed on September 8, 2025, https://huggingface.co/blog/ProCreations/transformers-are-getting-old
72. [2501.14850] On the locality bias and results in the Long Range Arena - arXiv, accessed on September 8, 2025, https://arxiv.org/abs/2501.14850
73. [2302.01107] A Survey on Efficient Training of Transformers - arXiv, accessed on September 8, 2025, https://arxiv.org/abs/2302.01107
74. Beltagy, I., Peters, M.E. and Cohan, A. (2020) Longformer The Long-Document Transformer. - References - Scientific Research Publishing, accessed on September 8, 2025, https://www.scirp.org/reference/referencespapers?referenceid=3753133
75. [PDF] Longformer: The Long-Document Transformer - Semantic Scholar, accessed on September 8, 2025, https://www.semanticscholar.org/paper/Longformer%3A-The-Long-Document-Transformer-Beltagy-Peters/925ad2897d1b5decbea320d07e99afa9110e09b2
76. Longformer: The Long-Document Transformer - BibSonomy, accessed on September 8, 2025, https://www.bibsonomy.org/bibtex/22435246630c361e01f31009339c70f41/ghagerer
77. ICLR Poster Rethinking Attention with Performers, accessed on September 8, 2025, https://iclr.cc/virtual/2021/poster/2726
78. Oral Session 7 - ICLR 2026, accessed on September 8, 2025, https://iclr.cc/virtual/2021/session/4343
79. Rethinking Attention with Performers - Personal website - Valerii Likhosherstov, accessed on September 8, 2025, https://valerytyumen.github.io/publication/masked
80. [PDF] Transformer-XL: Attentive Language Models beyond a Fixed-Length Context, accessed on September 8, 2025, https://www.semanticscholar.org/paper/Transformer-XL%3A-Attentive-Language-Models-beyond-a-Dai-Yang/c4744a7c2bb298e4a52289a1e085c71cc3d37bc6
81. kimiyoung/transformer-xl - GitHub, accessed on September 8, 2025, https://github.com/kimiyoung/transformer-xl
82. lucidrains/reformer-pytorch: Reformer, the efficient Transformer, in Pytorch - GitHub, accessed on September 8, 2025, https://github.com/lucidrains/reformer-pytorch
83. [PDF] Long Range Arena: A Benchmark for Efficient Transformers | Semantic Scholar, accessed on September 8, 2025, https://www.semanticscholar.org/paper/Long-Range-Arena%3A-A-Benchmark-for-Efficient-Tay-Dehghani/7e9ff94476f41041c75e253e84f487db00e9c861
84. Vaswani, A., et al. (2017) Attention Is All You Need. Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, 4-9 December 2017, 6000-6010. - References - Scientific Research Publishing, accessed on September 8, 2025, https://www.scirp.org/reference/referencespapers?referenceid=3964316
85. Vaswani, A., Shazeer, N., Parmar, N., et al. (2017) Attention Is All You Need. arXiv 1706.03762. - References, accessed on September 8, 2025, https://www.scirp.org/reference/referencespapers?referenceid=3700044
86. Attention is all you need - BibSonomy, accessed on September 8, 2025,

https://www.bibsonomy.org/bibtex/c9bf08cbcb15680c807e12a01dd8c929
87. Linformer: Self-Attention with Linear Complexity | - of Madian Khabsa, accessed on September 8, 2025, https://www.madiankhabsa.com/publication/linformer-2020/
88. [PDF] Linformer: Self-Attention with Linear Complexity - Semantic Scholar, accessed on September 8, 2025, https://www.semanticscholar.org/paper/Linformer%3A-Self-Attention-with-Linear-Complexity-Wang-Li/c0b79e6a5fd88ef13aa4780df5aae0aaa6b2be87
89. [2006.04768] Linformer: Self-Attention with Linear Complexity - arXiv, accessed on September 8, 2025, https://arxiv.org/abs/2006.04768
90. Linformer: Self-Attention with Linear Complexity - BibSonomy, accessed on September 8, 2025, https://www.bibsonomy.org/bibtex/14413c1dbf74e4224b819965aca6a77b9
91. Sinong Wang - Google Scholar, accessed on September 8, 2025, https://scholar.google.com/citations?user=CYMAfxsAAAAJ&hl=en
92. Big Bird: Transformers for Longer Sequences, accessed on September 8, 2025, https://ysu1989.github.io/courses/au20/cse5539/BigBird.pdf
93. Big Bird: Transformers for Longer Sequences. - DBLP, accessed on September 8, 2025, https://dblp.org/rec/conf/nips/ZaheerGDAAOPRWY20
94. [PDF] Big Bird: Transformers for Longer Sequences – Appendix | Semantic Scholar, accessed on September 8, 2025, https://www.semanticscholar.org/paper/Big-Bird%3A-Transformers-for-Longer-Sequences-%E2%80%93/b59d64ba042afbc198a806954517007074213b86
95. Big Bird: Transformers for Longer Sequences, accessed on September 8, 2025, https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html
96. Big Bird: Transformers for Longer Sequences - alphaXiv, accessed on September 8, 2025, https://www.alphaxiv.org/overview/2007.14062