Advancements in Architectures, Optimization Techniques, and Tokenization/Embeddings for Large Language Models

Samrat Kar | BL.SC.R4CSE24007

Introduction

Large Language Models (LLMs) represent a significant leap in artificial intelligence, demonstrating an unprecedented ability to process, generate, and understand human language.1 These sophisticated models, including prominent examples such as GPT-3, BERT, and T5, are built upon deep learning techniques and are trained on massive datasets of textual information.1 Their success spans a wide range of Natural Language Processing (NLP) tasks, including text generation, machine translation, question answering, summarization, and sentiment analysis, showcasing their capacity to capture intricate patterns and contextual understanding within language.1

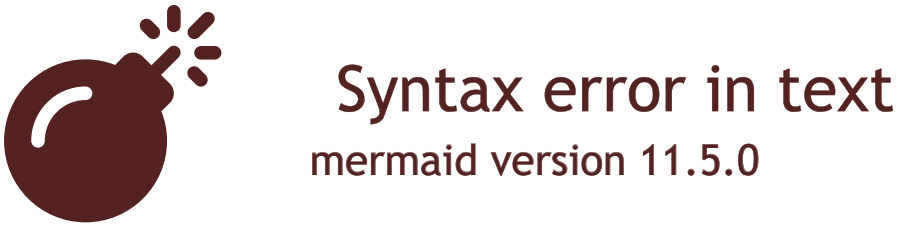
The foundation for this remarkable progress lies in the Transformer architecture, which has emerged as the cornerstone of modern NLP.1 Unlike earlier sequential processing models like Recurrent Neural Networks (RNNs), Transformers leverage a self-attention mechanism that enables parallel processing of input data.1 This parallelization significantly improves computational efficiency, allowing Transformers to handle longer sequences of text, capture complex dependencies within language, and scale more effectively with larger datasets.1 The original Transformer architecture, with its introduction of the attention mechanism, established a highly scalable and parallelizable model capable of handling long-range dependencies in data.1

The field of LLMs is in a state of continuous evolution, with ongoing research focused on enhancing their architecture, optimizing training methodologies, and refining the way they process and represent language through tokenization and embeddings.1 These efforts aim to overcome existing limitations and further improve the capabilities and efficiency of LLMs, addressing challenges such as computational complexity, memory constraints, and ethical considerations surrounding model biases.1 This report provides a comprehensive survey of recent advancements in these critical areas, exploring the innovations that are shaping the future of Large Language Models.

Key Concepts Central to the Research

1. **Transformer Architecture Enhancements**: Innovations aimed at improving the efficiency, scalability, and performance of transformer models.
2. **Optimization Techniques in Deep Learning**: Methods such as quantization, pruning, and knowledge distillation that enhance model efficiency and reduce computational resources.
3. **Tokenization Strategies**: Approaches to segmenting text into tokens, impacting model understanding and performance.
4. **Embedding Methods**: Techniques for representing tokens in continuous vector spaces, capturing semantic and syntactic information.
5. **Training and Fine-tuning Strategies**: Procedures for pretraining and adapting models to specific tasks, including methods like Reinforcement Learning from Human Feedback (RLHF).

Concept Map



Relevant Keywords and Search Terms

- Transformer architecture improvements
- Efficient transformers
- Transformer optimization techniques
- Quantization in deep learning
- Knowledge distillation in NLP
- Tokenization methods in LLMs
- Byte Pair Encoding vs. Unigram LM
- Embedding techniques in transformers
- Positional embeddings
- Word importance embedding
- Training strategies for LLMs
- Fine-tuning large language models
- RLHF in NLP
- Sparse attention mechanisms
- Memory-efficient transformers

Search Strategy Across Academic Databases

To ensure a comprehensive literature survey, the following databases were utilized:

1. **arXiv**: For the latest preprints on transformer architectures and optimization techniques.
2. **IEEE Xplore**: To access peer-reviewed articles on deep learning and transformer models.
3. **ACM Digital Library**: For conference proceedings related to NLP and machine learning.
4. **ScienceDirect**: To find journal articles on embedding methods and tokenization strategies.
5. **SpringerLink**: For comprehensive reviews and surveys on transformer models.
6. **Google Scholar**: To identify highly cited papers and track recent advancements.

Search Strategy:

- Utilized Boolean operators to combine keywords (e.g., "transformer AND optimization").
- Applied filters for publication years (2017-2025) to focus on recent developments.
- Reviewed abstracts to assess relevance before full-text analysis.

Advancements in Transformer Architecture for LLMs

Core Components of the Original Transformer Architecture

The Transformer architecture, which underpins most modern LLMs, is built around several key components that enable its powerful language processing capabilities.1

The architecture comprises two primary modules: the **encoder** and the **decoder**.1 The encoder is responsible for processing the input text and transforming it into a rich, contextualized representation that captures the essence of the input.4 The decoder then takes this encoded representation and uses it to generate the output sequence, such as a translation or a continuation of the input text.4 These two components work in concert, leveraging the power of attention mechanisms to understand and generate language.1

At the heart of the Transformer's encoder and decoder lies the **attention mechanism**, particularly **self-attention**.1 The attention mechanism allows the model to weigh the importance of different parts of the input sequence when processing a particular token.1 Self-attention extends this concept by enabling each token within the input sequence to attend to every other token in the sequence.1 This process allows the model to understand the relationships and dependencies between words, even if they are far apart in the sentence, which is crucial for capturing long-range context.1

The Transformer architecture further employs **multi-head attention**, which enhances the model's ability to capture diverse relationships within the data.1 Instead of using a single attention mechanism, multi-head attention utilizes multiple attention heads that operate in parallel.4 Each head can focus on different aspects of the input, allowing the model to simultaneously learn about various linguistic patterns, such as grammar and semantics, leading to a richer understanding of the text.4 The original Transformer model utilized eight such attention heads.7

Finally, since Transformers process all tokens in parallel, they lack an inherent understanding of the order of words in a sequence.4 To address this, the Transformer architecture incorporates **positional encoding**.4 This technique adds a unique vector to the embedding of each token, representing its position in the sequence.4 By providing the model with information about the order of tokens, positional encoding ensures that the Transformer can understand the flow and structure of language, which is vital for interpreting meaning.4

Key Innovations and Modifications in Transformer Architectures Tailored for LLMs

While the original Transformer architecture laid a robust foundation, the development of increasingly powerful LLMs has necessitated several key innovations and modifications to address the unique challenges of processing and generating natural language at scale.

Handling Long Context

One of the primary limitations of the standard Transformer architecture when applied to LLMs is the computational cost associated with the self-attention mechanism, which scales quadratically with the length of the input sequence.9 This quadratic complexity poses significant challenges when dealing with long documents or conversations, as the computational resources required become prohibitive.9 Many early LLMs were thus constrained to processing relatively short text snippets.8

To overcome this limitation, researchers have developed various **sparse attention mechanisms**.1 These techniques aim to reduce the number of attention calculations by selectively computing attention between only a subset of token pairs, rather than all pairs.16 This approach leverages the observation that not all token interactions are equally important for understanding the context.16 By focusing on the most relevant query-key pairs, sparse attention mechanisms can significantly reduce the computational burden, allowing LLMs to process much longer sequences while maintaining performance.16 Examples include blockwise sparse attention, which organizes tokens into blocks and focuses attention calculations within or between these blocks.16

Furthermore, there has been extensive research into **context window extension techniques** and **memory mechanisms** that enable LLMs to handle and generate extremely long sequences more effectively.6 These advancements explore various strategies to optimize the efficiency of the attention mechanism and to augment the Transformer architecture with external memory components that can store and retrieve information over extended contexts.9 Comprehensive surveys in this area provide a detailed overview of the progress in enhancing the long-context capabilities of Transformer-based LLMs across the entire model lifecycle, from pre-training to inference.6

Efficiency Improvements

Beyond handling long context, enhancing the overall **computational efficiency** of Transformer architectures for LLMs is a critical area of research.1 The quadratic complexity of the attention mechanism remains a bottleneck, and various techniques have been explored to mitigate this.1 Subquadratic-time architectures, such as those employing linear attention or gated convolution models, have been developed to address the computational inefficiencies of Transformers on long sequences.14

Advancements in the attention mechanism itself have also contributed to efficiency gains. **Multi-query attention (MQA)** and **grouped-query attention (GQA)** are two such innovations that can lead to increased processing speed and reduced computational load, particularly during the inference phase.3 These techniques reduce redundancy in the attention computation, making the model faster and more memory-efficient.26

Multi-Modal Capabilities

The evolution of Transformer models extends beyond processing just natural language to handling **content across different modalities**.1 This allows LLMs to be applied to a broader range of tasks, including those involving computer vision, speech processing, and other data types.7 For instance, the introduction of the Vision Transformer (ViT) demonstrated the adaptability of the Transformer architecture to image-based applications.2

Specific Architectural Variants

While the core principles of the Transformer remain influential, several specific architectural variants have emerged as prominent choices for building LLMs. **Decoder-only transformer architectures**, exemplified by models like GPT-3 and GPT-4, have become particularly prevalent for language generation tasks.29 Additionally, the foundational contributions of models like **BERT** (Bidirectional Encoder Representations from Transformers), which utilizes a bidirectional encoder for understanding tasks, and **GPT** (Generative Pre-trained Transformer), which employs an autoregressive decoder for language generation, have significantly shaped the field.1

Table 1: Key Architectural Advancements in Transformers for LLMs

Advancement	Description	Relevant Snippet IDs	Key Benefits for LLMs
Sparse Attention	Reduces the number of attention calculations by selectively focusing on important token interactions.	1	Enables processing of longer sequences, reduces computational cost.
Context Window Extension	Techniques and memory mechanisms to handle and generate extremely long sequences.	6	Allows for reasoning over extended information, improves performance on tasks requiring long-range context.
Multi-Query Attention (MQA) / Grouped-Query Attention (GQA)	Attention mechanisms that share key and value projections across multiple query heads.	3	Increases processing speed, reduces computational load, especially during inference.
Multi-Modal Capabilities	Adaptations of the Transformer architecture to process and understand content across different data types beyond text.	1	Extends the applicability of LLMs to tasks involving images, audio, and other modalities.

Optimization Techniques for Training LLMs

Training Large Language Models, which often contain billions or even trillions of parameters and are trained on massive datasets, demands significant computational resources and time.27 To make this process more feasible and efficient, a variety of optimization techniques have been developed and are continuously being refined.

Efficient Fine-tuning (Parameter-Efficient Fine-Tuning - PEFT)

Parameter-Efficient Fine-Tuning (PEFT) represents a suite of techniques designed to adapt pre-trained LLMs to specific downstream tasks in a cost-effective manner.35 The core idea behind PEFT is to minimize the number of trainable parameters and the computational overhead during the fine-tuning process, while still achieving performance comparable to or even better than full fine-tuning.36 This is particularly crucial when working with LLMs on hardware with limited computational capabilities.38

One of the most popular PEFT methods is **Low-Rank Adaptation (LoRA)**.35 LoRA works by freezing the weights of the pre-trained model and injecting trainable low-rank matrices into each layer of the Transformer architecture.35 This significantly reduces the number of parameters that need to be updated during fine-tuning, leading to substantial savings in GPU memory and training time.45 For example, LoRA can reduce the number of trainable parameters by up to 10,000 times compared to fine-tuning the entire model.45

Other notable PEFT techniques include **adapter tuning**, which involves inserting small, task-specific neural networks (adapters) between the layers of the pre-trained model and only training these adapters.36 **Prompt tuning** is another approach where the input prompts are optimized to guide the pre-trained model to perform the desired task effectively,

often by learning soft prompts which are sequences of embedding vectors.36 **Prefix tuning** focuses on optimizing continuous prompts specifically for generation tasks.27

Model Compression

Model compression techniques aim to reduce the size and computational complexity of LLMs, making them more suitable for deployment and inference, especially in resource-constrained environments.27 These methods enhance the speed, efficiency, and overall performance of LLMs while trying to maintain the quality of the original, uncompressed model.51

Quantization is a widely used model compression technique that involves converting the weights and activations of an LLM from high-precision data types (e.g., 32-bit floating-point numbers) to lower-precision data types (e.g., 8-bit or 4-bit integers).27 This reduction in precision leads to smaller model sizes, lower memory consumption, and faster inference speeds.49 Techniques like **QLoRA** further optimize this by quantizing the original weights of the base LLM to 4 bits, significantly reducing memory requirements and making it feasible to run very large models on a single GPU.49

Pruning is another effective compression method that focuses on identifying and removing redundant weights from the model while trying to preserve its performance.27 Pruning can be either unstructured, where individual weights are removed resulting in a sparse model, or structured, where entire components like rows, columns, or even attention heads are removed to maintain a more regular network structure.51 Pruning often requires retraining to maintain the model's quality, but recent research explores zero-shot pruning methods that aim to achieve this without significant retraining.51

Knowledge distillation is a technique used for both fine-tuning and compressing models.27 The goal is to transfer the knowledge from a larger, more complex "teacher" model to a smaller, more efficient "student" model.51 This is typically done by training the student model to mimic the outputs of the teacher model on a given dataset.51

Distributed Training

Given the massive scale of LLMs, **distributed training** across multiple computing devices, such as GPUs or specialized AI accelerators, has become essential to make the training process feasible and accelerate it.34 Distributed training involves dividing the training workload across multiple machines or processors that work in parallel and coordinate their efforts to train a single model.34

One common approach is **data parallelism**, where the training dataset is split into smaller batches, and each device in the distributed system processes a different batch of data to compute updates to the model's parameters.34 These updates are then aggregated to adjust the overall model.58

Another strategy is **model parallelism**, which is employed when the model itself is too large to fit into the memory of a single device.26 In this case, different parts of the model are placed on different devices, and the computation is distributed across them.26 **Pipeline parallelism** is a form of model parallelism where the model is broken down into a sequence of stages, and each stage is processed on a different device, allowing for more efficient utilization of computational resources.26

Memory Optimization

Optimizing memory usage during the training of LLMs is crucial due to the limited memory capacity of individual devices. Techniques like **gradient accumulation** allow for training with a larger effective batch size than would otherwise fit in

memory.26 This is achieved by accumulating gradients over several forward and backward passes before performing a weight update.26 Additionally, **key-value caching** in LLMs, which stores the key and value tensors from previous layers to speed up the generation process, can consume a significant amount of memory that grows with the sequence length.14 Research is ongoing to develop more memory-efficient methods for managing this cache.14

Optimized Optimizers

The choice of optimizer plays a significant role in the efficiency and effectiveness of training LLMs. Traditional optimizers like Stochastic Gradient Descent (SGD) are commonly used, but **adaptive optimizers** such as AdamW and LAMB are often preferred for training LLMs as they automatically adjust the learning rate for each parameter, helping the model learn faster and more effectively.58 Furthermore, the development of **8-bit optimizers** aims to reduce the memory footprint and accelerate data sharing during distributed training by lowering the precision of model parameters and calculations.26

Table 2: Summary of Optimization Techniques for LLM Training

Optimization Technique	Category	Description	Relevant Snippet IDs	Key Benefits for LLM Training
Low-Rank Adaptation (LoRA)	Efficient Fine-tuning	Freezes pre-trained weights and trains low-rank matrices.	35	Reduces memory usage, speeds up fine-tuning.
Quantization	Model Compression	Converts model weights and activations to lower precision.	27	Reduces model size, lowers memory consumption, accelerates inference.
Pruning	Model Compression	Identifies and removes redundant weights from the model.	27	Reduces model size and computational complexity.
Data Parallelism	Distributed Training	Divides training data across multiple devices.	34	Accelerates training by processing data in parallel.
Model Parallelism	Distributed Training	Splits the model itself across multiple devices.	26	Enables training of models too large for a single device.
Gradient Accumulation	Memory Optimization	Accumulates gradients over multiple steps to simulate larger batch sizes.	26	Allows training with larger effective batch sizes within memory limits.
Adaptive Optimizers (AdamW, LAMB)	Optimized Optimizers	Automatically adjust learning rates for each parameter.	58	Improves learning speed and effectiveness.

Optimization Technique	Category	Description	Relevant Snippet IDs	Key Benefits for LLM Training
8-bit Optimizers	Optimized Optimizers	Lowers precision of optimizer states to reduce memory.	26	Reduces memory footprint during training.

Tokenization and Embedding Methods in LLMs

The way in which text data is processed and represented is fundamental to the performance of Large Language Models. **Tokenization** is the initial crucial step that involves breaking down raw text into smaller units called tokens.35 These tokens serve as the basic building blocks that the model can understand and process.70 The consistency of the tokenization process is vital for pre-trained models to function correctly.66 Following tokenization, **embeddings** are used to convert these tokens into dense vector representations that capture their semantic meaning and relationships.62

Different Tokenization Methods

Various tokenization methods exist, each with its own advantages and trade-offs.64

Word-level tokenization is a straightforward approach that splits text into individual words based on spaces and punctuation.35 While simple, this method can lead to a very large vocabulary size, especially for languages with rich morphology, and it struggles to handle rare or out-of-vocabulary words.63

Character-level tokenization treats each individual character as a token.63 This results in a much smaller vocabulary but can lead to very long sequences, making it harder for the model to learn meaningful representations.63

Subword tokenization techniques aim to strike a balance between word-level and character-level tokenization by breaking words into smaller, more frequent units called subwords.35 This approach helps in handling rare words by decomposing them into known subwords and also reduces the vocabulary size compared to word-level tokenization.64 Several popular subword tokenization algorithms are used in LLMs:

- **Byte-Pair Encoding (BPE)** is a widely adopted algorithm that starts with a character-level vocabulary and iteratively merges the most frequent pairs of adjacent tokens to create new tokens until a desired vocabulary size is reached.27 It is used in models like GPT-2, GPT-3, RoBERTa, and LLaMA.35
- **WordPiece** is another subword tokenization method, employed in models such as BERT, DistilBERT, and Electra.35 Similar to BPE, it builds a vocabulary of subword units but uses a different scoring mechanism to decide which pairs of tokens to merge.74 WordPiece tends to prefer longer subword units.74
- **SentencePiece** is a language-independent subword tokenizer that can train subword models from raw sentences without any pre-tokenization based on whitespace.35 It supports both BPE and unigram language models and is used in models like T5, ALBERT, and GPT-J.35

Hybrid tokenization involves combining different tokenization methods to leverage their respective strengths and optimize for efficiency and flexibility.63 For example, the GPT-3 tokenizer uses a hybrid approach of BPE and character-level tokenization.63

Table 3: Comparison of Tokenization and Embedding Methods in LLMs

Method	Type	Description	Key Impacts on LLMs
Byte-Pair Encoding (BPE)	Tokenization	Iteratively merges frequent pairs of bytes.	Balances vocabulary size and handling of rare words.
WordPiece	Tokenization	Similar to BPE but uses a scoring mechanism for merging.	Tends to prefer longer subword units.
SentencePiece	Tokenization	Language-independent, trains from raw sentences, supports BPE and unigram.	Handles whitespace as a symbol, allows lossless detokenization.
Contextual Embeddings	Embedding	Vector representation of a word varies based on its context.	Captures nuanced meaning of words in different contexts.
T-FREE	Embedding	Tokenizer-free, embeds words using sparse activation patterns over character triplets.	Memory-efficient, potentially better for multilingual data.

Evolution of Embedding Techniques

The way in which tokens are converted into numerical vectors (embeddings) has also evolved significantly, impacting the ability of LLMs to understand language.70

Traditional word embeddings, such as those generated by Word2Vec and GloVe, assign a single, fixed vector to each word in the vocabulary.62 These embeddings capture semantic relationships between words but fail to account for the fact that a word can have different meanings depending on the context in which it is used.63

Contextual embeddings, on the other hand, are generated by modern LLMs like BERT and GPT, where the vector representation of a word varies based on the surrounding words in a sentence.63 This allows the model to understand the nuanced meaning of words in different contexts, significantly improving performance on various NLP tasks.81

Recent research has also explored **tokenizer-free embedding approaches**.72 These methods aim to bypass the traditional tokenization step by directly embedding words or sub-character units using techniques like sparse representations based on character triplets.72 Such approaches can offer potential advantages in terms of efficiency, handling of multilingual data, and capturing morphological similarities.72

Impact of Tokenization and Embedding Choices

The choice of tokenization and embedding methods has a profound impact on the performance and efficiency of LLMs.64 The tokenization strategy determines the model's vocabulary size and its ability to handle rare words and different languages.64 Embeddings, by providing numerical representations of tokens, enable the model to understand the semantic meaning and relationships within the text.62 The evolution from static word embeddings to contextual embeddings and the exploration of tokenizer-free methods highlight the ongoing efforts to improve the way LLMs process and understand the complexities of human language.72

Table 4: Comparison of Tokenization and Embedding Methods in LLMs

Method	Type	Description	Key Impacts on LLMs
Byte-Pair Encoding (BPE)	Tokenization	Iteratively merges frequent pairs of bytes.	Balances vocabulary size and handling of rare words.
WordPiece	Tokenization	Similar to BPE but uses a scoring mechanism for merging.	Tends to prefer longer subword units.
SentencePiece	Tokenization	Language-independent, trains from raw sentences, supports BPE and unigram.	Handles whitespace as a symbol, allows lossless detokenization.
Contextual Embeddings	Embedding	Vector representation of a word varies based on its context.	Captures nuanced meaning of words in different contexts.
T-FREE	Embedding	Tokenizer-free, embeds words using sparse activation patterns over character triplets.	Memory-efficient, potentially better for multilingual data.

Conclusion and Future Directions

This report has surveyed the significant advancements in the core components that drive the capabilities of Large Language Models. The evolution of the Transformer architecture continues to be a central focus, with innovations aimed at enhancing the ability of LLMs to process and generate long sequences efficiently. Techniques such as sparse attention and context window extension are crucial steps towards enabling LLMs to tackle more complex, real-world tasks that require understanding and reasoning over extended information.

The optimization of LLM training remains a critical area, driven by the immense computational resources required. Parameter-Efficient Fine-Tuning methods like LoRA have made it more feasible to adapt these massive models for specific applications. Model compression techniques, including quantization and pruning, are essential for deploying LLMs in resource-constrained environments. Furthermore, distributed training strategies are fundamental for handling the sheer scale of modern LLMs.

The methods used to tokenize and embed text have also seen considerable progress. Subword tokenization algorithms like BPE, WordPiece, and SentencePiece provide effective ways to balance vocabulary size and the handling of rare words. The shift towards contextual embeddings has significantly improved the ability of LLMs to understand the nuances of language. Emerging tokenizer-free embedding approaches offer promising avenues for future research, potentially leading to more efficient and adaptable language models, especially for multilingual applications.

Despite these remarkable advancements, several challenges persist. The computational cost of training and deploying LLMs remains high, and further research is needed to develop more efficient architectures and optimization techniques. Handling extremely long contexts effectively without sacrificing performance is another ongoing challenge. Additionally, ensuring the fairness, reducing biases, and addressing ethical concerns associated with LLMs are crucial areas of future work.

Looking ahead, the field is likely to see continued innovation in all these areas. We can anticipate the development of novel Transformer architectures that are inherently more efficient and capable of handling even longer contexts. Further advancements in PEFT and model compression will likely lead to more accessible and deployable LLMs. The exploration of

new tokenization and embedding strategies, including those that move beyond traditional subword approaches, could lead to more robust and versatile language models. The continuous pursuit of these improvements promises to unlock even greater potential for LLMs across a wide range of applications, further revolutionizing how we interact with and leverage artificial intelligence.

Works cited

1. (PDF) Advancements in Transformer Architectures for Large Language Models AUTHOR, accessed on April 20, 2025, https://www.researchgate.net/publication/387295170_Advancements_in_Transformer_Architectures_for_Large_Language_Models_AUTHOR
2. A Historical Survey of Advances in Transformer Architectures - MDPI, accessed on April 20, 2025, https://www.mdpi.com/2076-3417/14/10/4316
3. Transformer (deep learning architecture) - Wikipedia, accessed on April 20, 2025, https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture)
4. Demystifying Transformer Architecture in Large Language Models - TrueFoundry, accessed on April 20, 2025, https://www.truefoundry.com/blog/transformer-architecture
5. How Transformers Work: A Detailed Exploration of Transformer Architecture - DataCamp, accessed on April 20, 2025, https://www.datacamp.com/tutorial/how-transformers-work
6. Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey arXiv, accessed on April 20, 2025, https://arxiv.org/html/2311.12351v2
7. Transformers and large language models in healthcare: A review - PMC - PubMed Central, accessed on April 20, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC11638972/
8. Paper page Advancing Transformer Architecture in Long-Context ..., accessed on April 20, 2025, https://huggingface.co/papers/2311.12351
9. arxiv.org, accessed on April 20, 2025, http://arxiv.org/pdf/2311.12351
10. Multi-Head Attention and Transformer Architecture Pathway, accessed on April 20, 2025, https://pathway.com/bootcamps/rag-and-llms/coursework/module-2-word-vectors-simplified/bonus-overview-of-the-transformer-architecture/multi-head-attention-and-transformer-architecture/
11. What is Multi-head Attention and how does it improve model performance over single Attention head? AIML.com, accessed on April 20, 2025, https://aiml.com/what-is-multi-head-attention-and-how-does-it-improve-model-performance-over-single-attention-head/
12. 1909.00188 Improving Multi-Head Attention with Capsule Networks - arXiv, accessed on April 20, 2025, https://arxiv.org/abs/1909.00188
13. Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey | PDF - Scribd, accessed on April 20, 2025, https://www.scribd.com/document/782625300/3
14. Advanced Transformer Architectures - About deep2Read - GitHub Pages, accessed on April 20, 2025, https://qdata.github.io/deep2Read/fmefficient/L26/
15. MoA: Mixture of Sparse Attention for Automatic Large Language ..., accessed on April 20, 2025, https://openreview.net/forum?id=konDsSUSqg
16. Natively Sparse Attention (NSA) for Efficient Long-Context LLMs - Ajith's AI Pulse, accessed on April 20, 2025, https://ajithp.com/2025/02/21/natively-sparse-attention-nsa-the-future-of-efficient-long-context-modeling-in-large-language-models/
17. arxiv.org, accessed on April 20, 2025, https://arxiv.org/abs/2410.13276
18. 2311.12351 Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey - arXiv, accessed on April 20, 2025, https://arxiv.org/abs/2311.12351

19. Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey - Scholars.io, accessed on April 20, 2025, https://app.scholars.io/research/10381/advancing-transformer-architecture-in-long-context-large-language-models-a-comprehensive-survey
20. Strivin0311/long-llms-learning: A repository sharing the literatures about long-context large language models, including the methodologies and the evaluation benchmarks - GitHub, accessed on April 20, 2025, https://github.com/Strivin0311/long-llms-learning
21. Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey - Semantic Scholar, accessed on April 20, 2025, https://www.semanticscholar.org/paper/Advancing-Transformer-Architecture-in-Long-Context-Huang-Xu/4ea5ca620122e6a9a2b000444d36491cebf49c7c
22. 2501.06098 ELFATT: Efficient Linear Fast Attention for Vision Transformers - arXiv, accessed on April 20, 2025, https://arxiv.org/abs/2501.06098
23. arxiv.org, accessed on April 20, 2025, https://arxiv.org/abs/2412.02919
24. arxiv.org, accessed on April 20, 2025, https://arxiv.org/abs/2412.02344
25. arxiv.org, accessed on April 20, 2025, https://arxiv.org/abs/2405.05219
26. Efficient Deep Learning: A Comprehensive Overview of Optimization Techniques, accessed on April 20, 2025, https://huggingface.co/blog/Isayoften/optimization-rush
27. AIoT-MLSys-Lab/Efficient-LLMs-Survey: [TMLR 2024 ... - GitHub, accessed on April 20, 2025, https://github.com/AIoT-MLSys-Lab/Efficient-LLMs-Survey
28. Advances in Transformers for Robotic Applications: A Review - arXiv, accessed on April 20, 2025, https://arxiv.org/html/2412.10599v1
29. Towards Smaller, Faster Decoder-Only Transformers: Architectural Variants and Their Implications - arXiv, accessed on April 20, 2025, https://arxiv.org/html/2404.14462v2?ref=cohere-ai.ghost.io
30. Large Language Models: A Comprehensive Survey on Architectures, Applications, and Challenges - ResearchGate, accessed on April 20, 2025, https://www.researchgate.net/publication/387305663_Large_Language_Models_A_Comprehensive_Survey_on_Architectures_Applications_and_Challenges
31. (PDF) ADVANCEMENTS IN TRANSFORMER ARCHITECTURES FOR LARGE LANGUAGE MODEL: FROM BERT TO GPT-3 AND BEYOND - ResearchGate, accessed on April 20, 2025, https://www.researchgate.net/publication/380530250_ADVANCEMENTS_IN_TRANSFORMER_ARCHITECTURES_FOR_LARGE_LANGUAGE_MODEL_FROM_BERT_TO_GPT-3_AND_BEYOND
32. Efficient Training of Large Language Models on Distributed Infrastructures: A Survey - arXiv, accessed on April 20, 2025, https://arxiv.org/pdf/2407.20018
33. Efficient Large Language Models: A Survey - arXiv, accessed on April 20, 2025, https://arxiv.org/html/2312.03863v2
34. Distributed Training of Large Language Models - IEEE Computer Society, accessed on April 20, 2025, https://www.computer.org/csdl/proceedings-article/icpads/2023/307100a840/1VECXZMDcc0
35. Training Large Language Models (LLMs): Techniques and Best Practices - Nitor Infotech, accessed on April 20, 2025, https://www.nitorinfotech.com/blog/training-large-language-models-llms-techniques-and-best-practices/
36. Parameter-Efficient Fine-Tuning for Foundation Models - arXiv, accessed on April 20, 2025, https://arxiv.org/html/2501.13787v1
37. 2501.13787 Parameter-Efficient Fine-Tuning for Foundation Models - arXiv, accessed on April 20, 2025, https://arxiv.org/abs/2501.13787
38. arxiv.org, accessed on April 20, 2025, https://arxiv.org/abs/2403.14608
39. Train Small, Infer Large: Memory-Efficient LoRA Training for Large Language Models - arXiv, accessed on April 20, 2025, https://arxiv.org/abs/2502.13533
40. Train Small, Infer Large: Memory-Efficient LoRA Training for Large ..., accessed on April 20, 2025, https://openreview.net/forum?id=s7DkcgpRxL

41. scalerel Train Small, Infer Large: Memory-Efficient LoRA Training for Large Language Models - arXiv, accessed on April 20, 2025, https://arxiv.org/html/2502.13533v1
42. Literature Review Train Small, Infer Large: Memory-Efficient LoRA Training for Large Language Models - Moonlight, accessed on April 20, 2025, https://www.themoonlight.io/review/train-small-infer-large-memory-efficient-lora-training-for-large-language-models
43. Paper page - Train Small, Infer Large: Memory-Efficient LoRA Training for Large Language Models - Hugging Face, accessed on April 20, 2025, https://huggingface.co/papers/2502.13533
44. Memory-efficient Training of LLMs with Larger Mini-batches - arXiv, accessed on April 20, 2025, https://arxiv.org/html/2407.19580v1
45. 2106.09685 LoRA: Low-Rank Adaptation of Large Language Models - arXiv, accessed on April 20, 2025, https://arxiv.org/abs/2106.09685
46. GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection : r/LocalLLaMA - Reddit, accessed on April 20, 2025, https://www.reddit.com/r/LocalLLaMA/comments/1b80wei/galore_memoryefficient_llm_training_by_gradient/
47. TRAIN SMALL, INFER LARGE: MEMORY-EFFICIENT LORA, accessed on April 20, 2025, https://openreview.net/pdf/f3645a8d3db7ea3d0264384dea8c23dd492ef70a.pdf
48. What is the motivation for parameter-efficient fine tuning if there's no significant reduction in runtime or GPU memory usage? : r/MachineLearning - Reddit, accessed on April 20, 2025, https://www.reddit.com/r/MachineLearning/comments/186ck5k/d_what_is_the_motivation_for_parameterefficient/
49. A Guide to Quantization in LLMs | Symbl.ai, accessed on April 20, 2025, https://symbl.ai/developers/blog/a-guide-to-quantization-in-llms/
50. 5 Tips for Optimizing Language Models - KDnuggets, accessed on April 20, 2025, https://www.kdnuggets.com/5-tips-for-optimizing-language-models
51. Inference Optimizations for Large Language Models: Effects, Challenges, and Practical Considerations - arXiv, accessed on April 20, 2025, https://arxiv.org/html/2408.03130v1
52. A Survey on Model Compression for Large Language Models ..., accessed on April 20, 2025, https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00704/125482/A-Survey-on-Model-Compression-for-Large-Language
53. LLM Quantization: Techniques, Advantages, and Models - TensorOps, accessed on April 20, 2025, https://www.tensorops.ai/post/what-are-quantized-llms
54. Understanding Model Quantization in Large Language Models | DigitalOcean, accessed on April 20, 2025, https://www.digitalocean.com/community/tutorials/model-quantization-large-language-models
55. Paper page - Shortened LLaMA: A Simple Depth Pruning for Large ..., accessed on April 20, 2025, https://huggingface.co/papers/2402.02834
56. arxiv.org, accessed on April 20, 2025, https://arxiv.org/abs/2501.02086
57. FASP: Fast and Accurate Structured Pruning of Large Language ..., accessed on April 20, 2025, https://openreview.net/forum?id=f4b0YVwKUO
58. Innovations in Training Techniques for Large Language Models - IEEE Computer Society, accessed on April 20, 2025, https://www.computer.org/publications/tech-news/trends/training-techniques-large-language-models/
59. DeepMind looks at distributed training of large AI models - The Register, accessed on April 20, 2025, https://www.theregister.com/2025/02/11/deepmind_distributed_model_training_research/
60. Distributed training of large language models on AWS Trainium ..., accessed on April 20, 2025, https://www.amazon.science/publications/distributed-training-of-large-language-models-on-aws-trainium
61. LLM Distributed Training R: r/MachineLearning - Reddit, accessed on April 20, 2025, https://www.reddit.com/r/MachineLearning/comments/1i0vrg3/llm_distributed_training_r/

62. Tokenization vs Embedding - How are they Different? - Airbyte, accessed on April 20, 2025, https://airbyte.com/data-engineering-resources/tokenization-vs-embeddings
63. Exploring Foundations of Large Language Models (LLMs): Tokenization and Embeddings, accessed on April 20, 2025, https://dzone.com/articles/llms-tokenization-and-embeddings
64. Demystifying Tokens and Embeddings in Large Language Models, accessed on April 20, 2025, https://arbs.io/2024-01-14-demystifying-tokens-and-embeddings-in-llm
65. Tokenization | Mistral AI Large Language Models, accessed on April 20, 2025, https://docs.mistral.ai/guides/tokenization/
66. Tokenization in Large Language Models (LLMs) - ingoampt - Artificial Intelligence integration into iOS apps and SaaS + Education, accessed on April 20, 2025, https://ingoampt.com/tokenization-in-large-language-models-llms/
67. Understanding tokens - .NET - Learn Microsoft, accessed on April 20, 2025, https://learn.microsoft.com/en-us/dotnet/ai/conceptual/understanding-tokens
68. confused about embeddings and tokenization in LLMs : r/learnmachinelearning - Reddit, accessed on April 20, 2025, https://www.reddit.com/r/learnmachinelearning/comments/1cs29kn/confused_about_embeddings_and_tokenization_in_llms/
69. The Technical User's Introduction to LLM Tokenization - Christopher Samiullah, accessed on April 20, 2025, https://christophergs.com/blog/understanding-llm-tokenization
70. Machine-Learning/Tokens and Tokenization in Large Language ..., accessed on April 20, 2025, https://github.com/xbeat/Machine-Learning/blob/main/Tokens%20and%20Tokenization%20in%20Large%20Language%20Models%20in%20Python.md
71. Introduction to LLM Tokenization - Airbyte, accessed on April 20, 2025, https://airbyte.com/data-engineering-resources/llm-tokenization
72. T-FREE: Tokenizer-Free Generative LLMs via Sparse Representations for Memory-Efficient Embeddings - arXiv, accessed on April 20, 2025, https://arxiv.org/html/2406.19223v1
73. What is WordPiece? - H2O.ai, accessed on April 20, 2025, https://h2o.ai/wiki/wordpiece/
74. WordPiece tokenization - Hugging Face LLM Course, accessed on April 20, 2025, https://huggingface.co/learn/llm-course/chapter6/6
75. Tokenization - SentencePiece | Continuum Labs, accessed on April 20, 2025, https://training.continuumlabs.ai/training/the-fine-tuning-process/tokenization/tokenization-sentencepiece
76. Summary of the tokenizers - Hugging Face, accessed on April 20, 2025, https://huggingface.co/docs/transformers/tokenizer_summary
77. From Smør-re-brød to Subwords: Training LLMs on Danish, One Morpheme at a Time - arXiv, accessed on April 20, 2025, https://arxiv.org/html/2504.01540v1
78. T-FREE: Subword Tokenizer-Free Generative LLMs via Sparse Representations for Memory-Efficient Embeddings - arXiv, accessed on April 20, 2025, https://arxiv.org/html/2406.19223v2
79. Assessing the Importance of Frequency versus Compositionality for Subword-based Tokenization in NMT - arXiv, accessed on April 20, 2025, https://arxiv.org/html/2306.01393v3
80. google/sentencepiece: Unsupervised text tokenizer for ... - GitHub, accessed on April 20, 2025, https://github.com/google/sentencepiece
81. Explanation of Contextual Embeddings | Sapien's AI Glossary, accessed on April 20, 2025, https://www.sapien.io/glossary/definition/contextual-embeddings
82. Contextual Embeddings NLP Insights | Restackio, accessed on April 20, 2025, https://www.restack.io/p/embeddings-answer-contextual-embeddings-nlp-cat-ai
83. Introducing Contextual Retrieval \ Anthropic, accessed on April 20, 2025, https://www.anthropic.com/news/contextual-retrieval

84. Latest Developments in Vector Embeddings for AI Applications - CelerData, accessed on April 20, 2025, https://celerdata.com/glossary/vector-embeddings-for-ai-applications
85. 2406.19223 T-FREE: Subword Tokenizer-Free Generative LLMs via Sparse Representations for Memory-Efficient Embeddings - arXiv, accessed on April 20, 2025, https://arxiv.org/abs/2406.19223
86. T-FREE: Tokenizer-Free Generative LLMs via Sparse Representations for Memory-Efficient Embeddings | AI Research Paper Details - AIModels.fyi, accessed on April 20, 2025, https://www.aimodels.fyi/papers/arxiv/t-free-subword-tokenizer-free-generative-llms
87. T-FREE: Subword Tokenizer-Free Generative LLMs via Sparse Representations for Memory-Efficient Embeddings - Synthical, accessed on April 20, 2025, https://synthical.com/abs/2406.19223?is_dark=true&utm_source=dark_medium
88. Tokenizer-Free Generative LLMs via Sparse Representations for Memory-Efficient Embeddings | PromptLayer, accessed on April 20, 2025, https://www.promptlayer.com/research-papers/t-free-tokenizer-free-generative-llms-via-sparse-representations-for-memory-efficient-embeddings
89. T-FREE: Subword Tokenizer-Free Generative LLMs via Sparse Representations for Memory-Efficient Embeddings | Request PDF - ResearchGate, accessed on April 20, 2025, https://www.researchgate.net/publication/386202375_T-FREE_Subword_Tokenizer-Free_Generative_LLMs_via_Sparse_Representations_for_Memory-Efficient_Embeddings
90. T-FREE: Tokenizer-Free Generative LLMs via Sparse Representations for Memory-Efficient Embeddings - ChatPaper, accessed on April 20, 2025, https://chatpaper.com/chatpaper/paper/33267
91. T-FREE: Tokenizer-Free Generative LLMs via Sparse Representations for Memory-Efficient Embeddings - ResearchGate, accessed on April 20, 2025, https://www.researchgate.net/publication/381770835_T-FREE_Tokenizer-Free_Generative_LLMs_via_Sparse_Representations_for_Memory-Efficient_Embeddings
92. Paper page - T-FREE: Tokenizer-Free Generative LLMs via Sparse Representations for Memory-Efficient Embeddings - Hugging Face, accessed on April 20, 2025, https://huggingface.co/papers/2406.19223
93. Egalitarian Language Representation in Language Models: It All Begins with Tokenizers - arXiv, accessed on April 20, 2025, https://arxiv.org/pdf/2409.11501
94. 2205.11490 Local Byte Fusion for Neural Machine Translation - arXiv, accessed on April 20, 2025, https://arxiv.org/abs/2205.11490