Article

Not peer-reviewed version

Efficient Inference of Large Language Models through Model Compression

James Whitmore , Clara Hastings , Amir Patel , Stephany Brody *

Posted Date: 5 August 2025

doi: 10.20944/preprints202508.0192.v1

Keywords: model compression; large language models; pruning; quantization; knowledge distillation; low-rank approximation; neural architecture search; efficient inference; NLP; deep learning deployment; edge AI; resource-constrained environments; scalable AI; model optimization; energy-efficient models

Article

Efficient Inference of Large Language Models through Model Compression

James Whitmore [1], Clara Hastings [2], Amir Patel [3] and Stephany Brody [4,*]

[1] University of Oxford
[2] University of Cambridge
[3] King's College London
[4] University of Edinburgh
* Correspondence: stephany.brody@ed.ac.uk

Abstract

The increasing scale and complexity of large language models (LLMs) have revolutionized natural language processing (NLP), driving remarkable progress in a wide range of tasks such as machine translation, text summarization, sentiment analysis, and conversational AI. However, these performance gains have come at a substantial cost: modern LLMs often require billions of parameters, resulting in excessive computational demands, extensive memory footprints, and increased energy consumption. Such challenges significantly hinder the deployment of these models in resource-constrained environments, including mobile devices, edge platforms, and cost-sensitive cloud infrastructure. Consequently, model compression techniques have emerged as a vital solution for enhancing inference efficiency, reducing resource consumption, and accelerating model deployment without compromising task performance. This survey provides a comprehensive and structured overview of model compression techniques specifically designed to improve the efficiency of large language models. We categorize existing approaches into five major paradigms: *pruning*, *quantization*, *knowledge distillation*, *low-rank approximation*, and *neural architecture search (NAS)*. For each category, we examine the underlying theoretical principles, highlight state-of-the-art methods, and discuss their practical implementations. We further analyze the trade-offs between compression ratios, inference speed, and model accuracy, shedding light on the suitability of different approaches for various real-world scenarios. In addition to covering core compression techniques, this survey explores evaluation metrics that extend beyond traditional accuracy measures. We discuss latency, throughput, memory efficiency, and energy consumption as crucial performance indicators for compressed models. Furthermore, we examine deployment strategies that integrate compressed models into diverse environments, including cloud-based services, edge devices, and on-premises infrastructure. We outline best practices for efficient serving frameworks, scalable resource management, and hardware-aware optimization techniques to maximize the benefits of compressed models in production systems. Despite significant advancements, the field of model compression faces persistent challenges. Aggressive compression can lead to catastrophic performance degradation, loss of generalization capabilities, and increased vulnerability to adversarial attacks. Furthermore, compressed models may unintentionally amplify biases present in the original models, posing ethical concerns in sensitive applications such as healthcare, finance, and legal decision-making. Additionally, deploying compressed models across heterogeneous hardware platforms presents new challenges in optimizing performance while maintaining compatibility. To address these challenges, we identify promising research directions aimed at improving model compression techniques. We highlight the need for adaptive compression frameworks that dynamically adjust model complexity based on real-time conditions. We also emphasize the importance of energy-efficient compression methods that minimize the environmental impact of large-scale AI systems. Furthermore, we advocate for developing fairness-aware compression techniques that prioritize the retention of features crucial for minority groups and marginalized populations. Finally, we encourage the creation of standardized evaluation benchmarks that provide holistic assessments of compressed models' accuracy, robustness, latency, and resource efficiency. This survey aims to equip researchers and practitioners with a comprehensive understanding of model compression techniques, empowering them to design, evaluate, and deploy efficient language models that meet the growing demands of modern NLP applications. By addressing both the technical foundations and practical deployment considerations, this work provides a valuable resource for advancing the development of scalable, cost-effective, and accessible AI systems. Through continued innovation in compression methods, the NLP community can build robust, environmentally sustainable models that unlock the potential of language AI across diverse domains.

Keywords: model compression; large language models; pruning; quantization; knowledge distillation; low-rank approximation; neural architecture search; efficient inference; NLP; deep learning deployment; edge AI; resource-constrained environments; scalable AI; model optimization; energy-efficient models

---

1. Introduction

The rapid advancements in deep learning and natural language processing (NLP) have revolutionized various domains, ranging from machine translation and sentiment analysis to conversational agents and automated summarization. Central to these advancements are large language models (LLMs), which have demonstrated remarkable capabilities in understanding, generating, and interacting with natural language [1]. Models such as GPT-4, PaLM, LLaMA, and BLOOM have achieved unprecedented success in various NLP benchmarks, outperforming previous methods by significant margins. These models, often comprising billions of parameters, leverage massive datasets and extensive computational resources to deliver impressive performance across tasks. The success of LLMs can largely be attributed to three key factors: model scale, data volume, and improved training strategies. Scaling laws, as demonstrated in recent research, indicate that increasing model size consistently enhances performance across a wide range of NLP tasks. Consequently, contemporary language models are designed with hundreds of billions of parameters, pushing the boundaries of computational capacity and infrastructure. While these models excel in tasks such as text generation, code generation, and reasoning, their size poses significant challenges when it comes to real-world deployment and efficient inference.

1.1. Motivation for Model Compression

The exponential growth in model size introduces a series of practical challenges. First and foremost, the substantial memory footprint of LLMs presents major barriers to deployment, particularly in edge devices, mobile applications, and environments with limited hardware resources [2]. For instance, GPT-3's 175 billion parameters require several hundred gigabytes of memory, restricting deployment to specialized servers with high-end GPUs or TPUs [3]. This reliance on powerful hardware not only limits accessibility but also significantly increases the financial costs associated with inference. Moreover, large-scale models are inherently computationally intensive. Inference in these models involves billions of matrix multiplications and other complex operations, resulting in increased latency [4]. This is particularly problematic for real-time applications such as chatbots, translation services, and virtual assistants, where prompt responses are essential for a seamless user experience [5]. Without appropriate optimizations, deploying such models in latency-sensitive scenarios becomes impractical [6]. Energy efficiency is another critical concern. Running LLMs on cloud infrastructure requires substantial power resources, which contributes to both environmental and economic concerns. As the AI community becomes increasingly aware of the carbon footprint associated with large-scale models, achieving efficient inference has become a major research priority. Efficient models are crucial to making powerful language models sustainable and accessible for broader use. Finally, cost considerations remain paramount. Large models require not only powerful hardware for deployment but also continuous investment in infrastructure and maintenance. Reducing the resource demands of these models can lower deployment costs, making advanced language models accessible to a wider range of developers, businesses, and researchers [7].

1.2. Model Compression as a Solution

Model compression techniques have emerged as a promising solution to address these scalability challenges by reducing model size, improving inference speed, and enhancing energy efficiency — all while maintaining the model's predictive performance. Broadly, model compression techniques can be categorized into four key strategies: *pruning*, *quantization*, *knowledge distillation*, and *low-rank factorization*.

Pruning

Pruning techniques aim to identify and remove redundant parameters from the model while preserving its core functionality [8]. Inspired by the concept of sparsity in neural networks, pruning methods focus on eliminating weights with minimal impact on the model's output. Techniques such as magnitude-based pruning, structured pruning, and dynamic pruning have demonstrated substantial reductions in model size while retaining competitive performance.

Quantization

Quantization methods reduce the precision of model parameters and activations, enabling the model to operate using lower-bit representations (e.g., 8-bit or 4-bit integers instead of 32-bit floating-point values). This technique significantly reduces memory consumption and computational overhead, making quantized models well-suited for deployment on resource-constrained hardware [9]. Advances in quantization-aware training and post-training quantization have enabled highly efficient yet accurate models [10].

Knowledge Distillation

Knowledge distillation is a teacher-student training paradigm in which a smaller, more efficient model (the "student") is trained to mimic the behavior of a larger, pre-trained model (the "teacher") [11]. By transferring the knowledge embedded in the teacher model's predictions, the student model can achieve competitive performance with significantly fewer parameters. Distillation techniques have gained significant popularity for their ability to balance compression and accuracy.

Low-Rank Factorization

Low-rank factorization techniques exploit the redundancy in neural network weight matrices by approximating them with lower-dimensional representations. By factorizing weight tensors into smaller components, these methods effectively reduce parameter counts without requiring extensive retraining. Techniques such as matrix decomposition and singular value decomposition (SVD) have demonstrated promising results in reducing model complexity [12].

1.3. Challenges in Model Compression

While compression techniques have shown considerable promise, they present several challenges that require careful consideration. One of the most significant concerns is the trade-off between compression ratio and model accuracy. Aggressive compression may lead to a substantial degradation in performance, particularly for complex tasks such as commonsense reasoning, machine translation, or code generation [13]. Additionally, the selection of appropriate compression strategies often requires extensive experimentation and tuning. Factors such as compression granularity, layer-specific pruning rates, and quantization bit-width must be carefully configured to ensure optimal performance. Compression techniques may also introduce instability, particularly during training, requiring robust methodologies to maintain convergence and generalization. Compatibility with hardware accelerators also poses a practical challenge. While certain compression techniques are highly effective in reducing parameter counts, they may introduce irregular memory access patterns that diminish the expected speedup on modern GPUs and TPUs. As a result, hardware-aware compression techniques have gained increasing attention to maximize efficiency in real-world deployments.

1.4. Contributions of This Survey

In this survey, we provide a comprehensive review of model compression techniques designed to improve the efficiency of large language models. Our contributions are as follows:

- We present an in-depth analysis of key model compression methods, including pruning, quantization, knowledge distillation, and low-rank factorization, with a focus on their applicability to LLMs.
- We highlight state-of-the-art advancements in compression techniques, detailing their theoretical foundations, practical implementations, and empirical results.
- We explore emerging trends in model compression, such as lottery ticket hypothesis frameworks, sparse training techniques, and hardware-aware optimizations.
- We discuss practical challenges and trade-offs encountered in deploying compressed models for real-world NLP applications.
- We outline key open research questions and future directions, aiming to inspire continued progress in the field of efficient LLM inference [14].

By synthesizing recent research developments and presenting practical insights, this survey aims to serve as a valuable resource for researchers, practitioners, and developers seeking to optimize language models for improved scalability and efficiency [15]. Our goal is to empower the community with actionable strategies for deploying large-scale NLP systems in diverse computational environments. The remainder of this paper is organized as follows: Section 2 outlines the theoretical foundations of LLMs and the scalability challenges they face [16]. Section 3 presents a detailed analysis of key compression techniques. Section 4 explores real-world applications of compressed models, while Section 8 discusses emerging trends and future directions. Finally, Section 9 concludes the paper with a summary of key findings.

2. Background

The rapid progress in natural language processing (NLP) has been largely driven by the development of increasingly larger and more powerful language models. These models, based on the Transformer architecture, have set new benchmarks across various NLP tasks. Understanding the underlying structure of large language models and the computational challenges they present is essential for contextualizing the importance of model compression [17].

2.1. The Transformer Architecture

The Transformer architecture, introduced by Vaswani et al. in 2017 [18], has become the foundation for modern large language models. Its self-attention mechanism enables efficient processing of sequences by allowing each token to attend to every other token in the input, capturing long-range dependencies effectively. Unlike recurrent neural networks (RNNs) and convolutional neural networks (CNNs), Transformers rely on parallel computation, making them highly scalable for modern hardware accelerators. A standard Transformer model consists of the following key components:

1. Multi-Head Self-Attention

This mechanism computes attention scores between each token in the sequence, allowing the model to learn contextual dependencies. Multiple attention heads operate in parallel to capture different types of information [19].

2. Feedforward Networks

Each Transformer layer includes a position-wise feedforward network that applies linear transformations and non-linear activations independently to each token [20].

3. Positional Encoding

Since Transformers lack a built-in notion of token order, positional encodings are added to the input embeddings to introduce sequence order information [21].

4. Layer Normalization and Residual Connections

These components improve training stability and enable deeper architectures [22].

5. Transformer Blocks

Transformer models are composed of multiple stacked layers, with each layer containing the self-attention mechanism, feedforward networks, and normalization layers [23].

2.2. Scaling Laws and the Growth of LLMs

The remarkable performance improvements achieved by models such as GPT-3, PaLM, and LLaMA have been attributed to scaling laws. Research has demonstrated that increasing model size, training data, and compute resources leads to consistent improvements in performance across diverse NLP tasks. For example, GPT-3's 175 billion parameters and PaLM's 540 billion parameters illustrate the trend toward massive models with enhanced capabilities. However, scaling models introduces several practical challenges:

- **Memory Footprint:** Large models require substantial memory capacity to store parameters, activations, and optimizer states, often demanding specialized hardware such as GPUs with extensive VRAM or TPUs.
- **Inference Latency:** Larger models require more computations per forward pass, resulting in slower inference speeds that hinder real-time applications.
- **Energy Consumption:** The computational demands of LLMs result in increased power consumption, raising environmental and economic concerns [24].
- **Deployment Constraints:** Deploying billion-parameter models on edge devices, mobile platforms, or environments with limited resources becomes highly impractical without substantial compression and optimization [25].

2.3. The Need for Model Compression

Given the substantial computational and memory requirements of LLMs, model compression has emerged as a crucial research direction. Compression techniques aim to reduce model size and computational overhead while preserving performance [26]. The motivations for model compression can be categorized as follows:

1. Efficient Deployment

Compressed models enable deployment in resource-constrained environments such as mobile devices, IoT systems, and embedded platforms [27]. Reducing model size ensures that LLMs can operate efficiently without requiring high-end hardware.

2. Reduced Latency

Optimized models require fewer computations per forward pass, improving response times for latency-sensitive applications such as chatbots, search engines, and recommendation systems [28–30].

3. Lower Power Consumption

By minimizing redundant computations and memory access patterns, compressed models reduce energy consumption, enhancing the sustainability of large-scale AI systems [31].

4. Cost Reductiony

Deploying compressed models on smaller-scale infrastructure reduces the need for expensive cloud resources, improving accessibility for startups, research labs, and developers [32,33].

2.4. *Categories of Model Compression Techniques*

Model compression techniques can be broadly categorized into four main strategies:

- **Pruning:** Removing redundant parameters to reduce model size while preserving key information [34]. Techniques include unstructured pruning, structured pruning, and dynamic pruning [35].
- **Quantization:** Reducing the numerical precision of model parameters to minimize memory usage and accelerate computation.
- **Knowledge Distillation:** Training a smaller model (student) to replicate the behavior of a larger model (teacher), transferring knowledge to improve efficiency.
- **Low-Rank Factorization:** Decomposing large weight matrices into smaller components using techniques such as singular value decomposition (SVD) to reduce parameter redundancy.

Each technique offers unique advantages and trade-offs, and combining multiple strategies often yields the most effective results.

2.5. *Trade-Offs in Model Compression*

While model compression offers substantial benefits, it also introduces several trade-offs that must be carefully managed:

1. Accuracy Degradation

Aggressive compression may result in reduced model performance, particularly on complex language tasks requiring fine-grained understanding and reasoning.

2. Stability Challenges

Some compression methods, such as quantization or pruning, may destabilize training dynamics, necessitating careful tuning and adaptive learning strategies.

3. Hardware Compatibility

Certain compression strategies may reduce model size but introduce irregular memory access patterns that hinder efficient execution on GPUs or TPUs [36].

2.6. *Scope of This Survey*

This survey aims to provide a comprehensive review of model compression techniques for large language models. We explore both theoretical foundations and practical implementations, analyzing the strengths, limitations, and real-world applicability of various approaches. By synthesizing recent advancements and identifying emerging trends, this survey aims to provide actionable insights for researchers and practitioners seeking to optimize LLMs for efficient deployment. In the following sections, we provide an in-depth examination of individual compression methods, discuss their practical implications, and highlight recent innovations in the field.

3. Compression Techniques for Large Language Models

Model compression techniques have emerged as crucial strategies for improving the efficiency of large language models (LLMs) while preserving their predictive performance. These methods focus on reducing the number of parameters, lowering computational costs, and minimizing the memory footprint. This section presents an in-depth exploration of four primary model compression techniques: pruning, quantization, knowledge distillation, and low-rank factorization. Additionally, we discuss hybrid approaches that combine multiple techniques for improved efficiency.

3.1. *Pruning*

Pruning aims to reduce model size by identifying and eliminating redundant or non-contributory parameters. Inspired by the hypothesis that over-parameterized models contain a significant number of inactive or less significant weights, pruning effectively compresses models without substantial performance degradation.

3.1.1. Types of Pruning

Pruning methods can be broadly categorized into the following:

1. Unstructured Pruning

Unstructured pruning removes individual weights based on certain criteria, typically targeting weights with the smallest absolute magnitudes. While this method can drastically reduce the number of non-zero parameters, the resulting sparse weight matrices often require specialized hardware or libraries for efficient execution.

2. Structured Pruning

Structured pruning removes entire components, such as neurons, attention heads, or convolutional filters. Unlike unstructured pruning, structured pruning preserves the model's inherent structure, enabling improved compatibility with standard hardware accelerators. Techniques such as filter pruning and head pruning in Transformer layers have shown success in maintaining performance while achieving notable compression [37].

3. Dynamic Pruning

Dynamic pruning adaptively removes parameters during runtime based on input-dependent conditions [38,39]. This approach allows for flexible resource allocation, enabling efficient inference without permanently modifying the model's architecture.

3.1.2. Pruning Strategies

Pruning techniques typically follow two main strategies:

- **Post-training Pruning:** Pruning is applied to a fully trained model, followed by fine-tuning to recover performance.
- **During-training Pruning:** Parameters are pruned progressively throughout the training process, often improving convergence stability.

3.1.3. Notable Advancements in Pruning

Recent advancements such as the Lottery Ticket Hypothesis [40] suggest that within over-parameterized models exist sparse subnetworks that can match the original model's performance when trained from scratch. This finding has inspired innovative pruning methods that identify these subnetworks early in training, reducing computational overhead [41].

3.2. Quantization

Quantization techniques aim to reduce model size and computational complexity by representing model weights and activations with lower precision data types [42]. By replacing 32-bit floating-point values with lower-bit representations, quantization achieves substantial reductions in memory consumption and accelerates inference.

3.2.1. Types of Quantization

1. Post-Training Quantization (PTQ)

PTQ involves applying quantization to a fully trained model without requiring retraining. While efficient and fast, PTQ may result in accuracy loss if aggressive quantization (e.g., 4-bit) is applied without calibration.

2. Quantization-Aware Training (QAT)

QAT incorporates quantization effects directly into the training process. By simulating low-precision arithmetic during training, QAT ensures that the model adapts to quantization constraints, often resulting in improved accuracy compared to PTQ [43].

3. Mixed-Precision Quantization

This approach assigns different bit-widths to different layers or parameters based on their sensitivity to quantization. For instance, embedding layers may use higher precision, while less critical layers are quantized more aggressively.

3.2.2. Hardware Compatibility

Quantized models are particularly effective for deployment on specialized hardware accelerators such as Google's Tensor Processing Units (TPUs), NVIDIA GPUs, and ARM-based processors. Many modern frameworks, including TensorFlow, PyTorch, and ONNX, offer built-in quantization support.

3.3. Knowledge Distillation

Knowledge distillation (KD) is a teacher-student framework in which a smaller model (the student) is trained to emulate the behavior of a larger, more complex model (the teacher). The student model aims to achieve comparable performance by leveraging the knowledge embedded in the teacher's predictions.

3.3.1. Types of Knowledge Distillation

1. Logit-Based Distillation

The student model learns by matching the soft output probabilities produced by the teacher. The softened probability distribution provides rich information about class relationships, aiding the student in generalizing effectively.

2. Feature-Based Distillation

In addition to logits, intermediate layer representations from the teacher are used to guide the student. This method enhances the student's ability to replicate the internal learning dynamics of the teacher [44].

3. Response-Based Distillation

Instead of matching logits or features, this method encourages the student to mimic the teacher's decision boundaries and overall response behavior [45].

3.3.2. Distillation Strategies

Distillation can be performed in various settings:

- **Offline Distillation:** The teacher is pre-trained, and its knowledge is transferred to the student through standard supervised learning.
- **Online Distillation:** Both teacher and student models are trained simultaneously, allowing the teacher to adapt dynamically during training [46].
- **Self-Distillation:** A single model trains itself by distilling knowledge from its earlier epochs into its later epochs [47].

Knowledge distillation has proven highly effective in reducing model size while preserving performance, particularly in NLP applications such as text classification, summarization, and translation [48].

3.4. Low-Rank Factorization

Low-rank factorization methods exploit the inherent redundancy in weight matrices by approximating them with lower-rank matrices [49–51]. This technique is based on the observation that many neural network weight matrices are highly over-parameterized and can be effectively decomposed without losing significant information.

3.4.1. Matrix Decomposition Techniques

Popular low-rank factorization techniques include:

- **Singular Value Decomposition (SVD):** Decomposes a matrix into three components to approximate the original weight matrix with a lower-rank structure.
- **CP Decomposition:** Factorizes tensors into a sum of component rank-one tensors, ideal for compressing Transformer weights [52].
- **Tensor Train Decomposition:** Factorizes tensors into smaller core tensors connected in a chain, achieving higher compression rates for extremely large models [53].

3.4.2. Challenges in Low-Rank Factorization

While low-rank factorization achieves substantial compression, the resulting factorized layers may introduce additional overhead in model execution if not carefully optimized for the target hardware. Techniques such as rank-adaptive training and low-rank fine-tuning aim to mitigate this trade-off.

3.5. Hybrid Approaches

Combining multiple compression strategies has emerged as a powerful approach for maximizing efficiency while preserving model performance [54]. For example, combining pruning with quantization achieves both weight reduction and improved hardware compatibility [55]. Similarly, distillation combined with quantization ensures that the student model retains key knowledge while benefiting from lower precision arithmetic [56].

3.6. Summary

Table 1. Comparison of Major Compression Techniques.

Technique	Compression Ratio	Accuracy Impact	Hardware Efficiency
Pruning	High (Structured)	Moderate	Requires Sparse Support
Quantization	Moderate to High	Low (with QAT)	High
Knowledge Distillation	Moderate to High	Low (if well-trained)	High
Low-Rank Factorization	Moderate	Moderate	Requires Optimization

In the next section, we explore real-world applications of compressed language models and highlight practical considerations for deploying these models effectively.

4. Applications of Compressed Language Models

The adoption of large language models (LLMs) has transformed numerous real-world applications, enabling advanced capabilities in natural language understanding, text generation, and conversational AI [57]. However, the computational and memory demands of LLMs pose significant challenges for deployment in resource-constrained environments. Compressed language models have emerged as a practical solution, enabling efficient inference while maintaining strong performance [58]. This section explores key application domains that benefit from model compression and outlines the associated technical considerations [59].

4.1. Real-Time Conversational Agents

Conversational AI systems, such as chatbots, virtual assistants, and customer support tools, require low latency and fast response times to ensure a seamless user experience. Large models like GPT-3 or Claude often struggle to meet these requirements due to their computational overhead [60].

4.1.1. Benefits of Compression

Compressed models effectively reduce inference latency by decreasing the number of parameters and minimizing the number of floating-point operations (FLOPs) per forward pass. Techniques such as quantization and structured pruning are particularly well-suited for latency reduction, enabling faster response times with minimal accuracy loss.

4.1.2. Case Study: Chatbot Deployment

For example, Meta's deployment of a compressed version of the LLaMA model enabled real-time conversational AI features on mobile devices with limited hardware resources. By combining pruning and quantization, Meta achieved substantial latency reductions while preserving conversational quality.

4.2. Edge AI and Mobile Applications

Deploying LLMs in edge environments, such as smartphones, IoT devices, and embedded systems, demands aggressive compression strategies to fit within tight memory and compute constraints [61].

4.2.1. Benefits of Compression

Compression techniques like post-training quantization (PTQ) and knowledge distillation are particularly effective in edge deployments. These methods reduce model size while ensuring compatibility with low-power hardware.

4.2.2. Case Study: Mobile Translation Models

Google's development of MobileBERT leveraged knowledge distillation to produce a lightweight yet powerful model optimized for mobile inference. By distilling knowledge from a large Transformer model into a compact architecture, MobileBERT achieved competitive performance in on-device NLP tasks such as translation and text classification.

4.3. Search Engines and Recommendation Systems

Search engines and recommendation systems rely heavily on efficient retrieval mechanisms that process vast amounts of data in real time. Large-scale language models have significantly improved ranking, query understanding, and personalized recommendations, yet deploying these models at scale introduces latency and cost concerns [62].

4.3.1. Benefits of Compression

Pruning and low-rank factorization are particularly effective in optimizing search and recommendation models. By removing redundant parameters and exploiting matrix decomposition techniques, these strategies reduce memory demands and improve throughput.

4.3.2. Case Study: E-commerce Recommendations

Amazon's personalized recommendation system applies low-rank factorization to compress Transformer-based ranking models, improving product search latency while maintaining high recommendation quality [63]. This approach enables scalable deployment across millions of users [64].

4.4. Healthcare and Biomedical NLP

In healthcare applications, language models play a crucial role in tasks such as clinical text analysis, medical question answering, and drug discovery. However, deploying LLMs in healthcare settings requires careful attention to latency, security, and hardware constraints [65].

4.4.1. Benefits of Compression

Knowledge distillation has been successfully applied to train lightweight healthcare models that retain the interpretability and performance of larger models [66]. Additionally, quantized models are particularly beneficial in clinical environments where privacy-preserving computation on secure hardware is essential [67].

4.4.2. Case Study: Medical QA Systems

IBM's Watson Health has leveraged compressed Transformer models to deliver medical question-answering systems optimized for hospital servers and clinical settings. By applying quantization and structured pruning, these models achieved faster inference speeds while preserving clinical accuracy.

4.5. Autonomous Systems and Robotics

Autonomous vehicles, drones, and industrial robots rely on NLP models for task planning, voice control, and navigation [68]. Efficient inference is crucial in these time-sensitive environments, where real-time decision-making is required.

4.5.1. Benefits of Compression

Dynamic pruning and quantization techniques are particularly effective in these domains, where adaptive resource management ensures optimal performance under variable computational constraints [69].

4.5.2. Case Study: Autonomous Vehicle Commands

Tesla's autopilot system employs compressed NLP models for interpreting natural language commands [70]. By leveraging hybrid compression strategies, the system maintains high accuracy while meeting strict latency requirements for real-time vehicle control.

4.6. Content Moderation and Social Media Analysis

Social media platforms employ NLP models to detect harmful content, filter spam, and moderate discussions. These platforms often require rapid inference across extensive text streams, necessitating efficient model designs [71].

4.6.1. Benefits of Compression

Pruning and low-rank factorization effectively reduce inference costs in large-scale moderation pipelines, enabling efficient processing of billions of user interactions.

4.6.2. Case Study: Content Moderation on Facebook

Facebook's moderation platform leverages compressed LLMs to detect harmful content in multiple languages. By applying structured pruning and quantization, Facebook improved throughput in real-time moderation pipelines while maintaining precision [72].

4.7. Financial Services and Fraud Detection

The financial sector relies heavily on NLP models for sentiment analysis, market prediction, and fraud detection. Achieving high throughput while preserving predictive accuracy is critical in these applications.

4.7.1. Benefits of Compression

Knowledge distillation combined with quantization has been shown to produce efficient financial NLP models that deliver rapid insights without compromising security [73].

4.7.2. Case Study: Fraud Detection Models

J.P. Morgan's fraud detection system employs a distilled Transformer model trained to detect suspicious transaction patterns [74]. By distilling knowledge from larger financial models into a compact network, the system achieved fast and accurate fraud detection at scale.

4.8. Trade-Offs and Practical Considerations

While model compression offers substantial benefits, practical deployment requires careful evaluation of trade-offs:

- **Accuracy vs. Efficiency:** Aggressive compression may introduce performance degradation, requiring fine-tuning to restore accuracy.
- **Hardware Constraints:** Compressed models must be optimized for target platforms to ensure efficient execution on CPUs, GPUs, or TPUs.
- **Data Distribution Shifts:** Compressed models may be less robust to distributional shifts, necessitating periodic re-evaluation and retraining.

4.9. Summary

Compressed language models have demonstrated significant success in improving the scalability, efficiency, and accessibility of NLP systems across diverse domains. By strategically applying compression techniques such as pruning, quantization, and knowledge distillation, developers can effectively deploy language models in resource-constrained environments without sacrificing performance [75]. The next section explores emerging trends in model compression, including hardware-aware optimizations, sparse training techniques, and efficient model architectures designed for scalability [76].

5. Emerging Trends in Model Compression

As language models continue to scale in size and complexity, the demand for advanced compression techniques has intensified. Recent innovations in model compression focus not only on achieving smaller model footprints but also on enhancing hardware efficiency, training dynamics, and adaptability in real-world scenarios [77]. This section explores emerging trends that are shaping the future of model compression for large language models (LLMs).

5.1. Hardware-Aware Compression

Modern hardware accelerators such as GPUs, TPUs, and custom AI processors (e.g., NVIDIA Tensor Cores, Apple Neural Engine) are increasingly influencing compression strategies [78]. Hardware-aware compression techniques aim to tailor model architectures to the underlying hardware, maximizing throughput and minimizing latency.

5.1.1. Techniques for Hardware Optimization

Several hardware-aware techniques have gained prominence:

1. Block Sparse Pruning

By pruning entire blocks of weights rather than individual connections, block sparse pruning aligns with efficient matrix multiplication routines supported by modern hardware [79].

2. Mixed-Precision Quantization

This technique assigns different bit-widths to different layers or parameters based on their sensitivity to precision loss. For example, attention layers may be quantized to 8-bit precision while embedding layers retain 16-bit precision to preserve model expressiveness [80].

3. Operator Fusion

Combining adjacent model operations into a single optimized kernel minimizes memory access overhead and improves parallelism on hardware accelerators [81].

5.1.2. Case Study: NVIDIA TensorRT

NVIDIA's TensorRT framework leverages block-sparse operations and mixed-precision quantization to accelerate compressed Transformer models, achieving up to 3x speed improvements on inference workloads.

5.2. Sparse Training Techniques

Traditional pruning methods often require iterative fine-tuning to recover performance after parameter removal. Sparse training techniques address this by enforcing sparsity constraints directly during training, producing inherently compressed models without post-hoc pruning.

5.2.1. Popular Sparse Training Strategies

- **Sparse MoE (Mixture of Experts):** Mixture of Experts architectures activate only a subset of expert layers during inference, reducing computational overhead while maintaining model flexibility.
- **Dynamic Sparse Training (DST):** DST gradually prunes and regrows model connections during training, identifying optimal sparse subnetworks without compromising performance.
- **Rigged Lottery Ticket Training:** This technique identifies sparse subnetworks early in training, directly optimizing the reduced model without additional pruning stages.

Sparse training has demonstrated strong potential in reducing LLM complexity while maintaining competitive performance on complex NLP benchmarks.

5.3. Neural Architecture Search (NAS) for Efficient Models

Neural Architecture Search (NAS) automates the design of efficient model architectures by exploring combinations of layer types, widths, and depths. NAS is increasingly being employed to construct compact yet powerful language models optimized for specific applications.

5.3.1. Techniques in NAS for Compression

- **Weight-Sharing NAS:** This method shares parameters across multiple candidate architectures to efficiently explore large design spaces [82].
- **One-Shot NAS:** A single supernet is trained, from which sub-networks can be extracted for efficient inference [83].
- **Hardware-Aware NAS:** This technique tailors model architectures to meet specific latency, power, or memory constraints on target hardware.

5.3.2. Case Study: Efficient Transformers

Research on architectures like Linformer, Performer, and Big Bird has leveraged NAS principles to design models that approximate Transformer attention mechanisms using fewer computations and smaller memory footprints.

5.4. Low-Rank Adaptation (LoRA) and Parameter-Efficient Fine-Tuning (PEFT)

Low-Rank Adaptation (LoRA) and other PEFT strategies have emerged as efficient methods for adapting large pretrained language models to new tasks. These techniques modify only a small subset of model parameters, significantly reducing storage and training overhead [84].

5.4.1. Key Characteristics of LoRA and PEFT

- **Reduced Parameter Overhead:** LoRA introduces trainable low-rank matrices into specific model layers, minimizing the number of updated parameters [85].
- **Fast Adaptation:** PEFT methods enable efficient fine-tuning by freezing most of the original model parameters, accelerating training and reducing memory requirements.

5.4.2. Case Study: GPT Fine-Tuning with LoRA

LoRA has demonstrated success in efficiently adapting large GPT models for domain-specific tasks such as legal document analysis, scientific text generation, and multilingual translation, achieving performance comparable to full fine-tuning with significantly lower costs.

5.5. Knowledge Distillation with Reinforcement Learning

Reinforcement learning (RL) is increasingly integrated into knowledge distillation frameworks to enhance student model performance [86]. RL-based distillation techniques adaptively prioritize high-value knowledge from the teacher model, improving the student's generalization capabilities [87].

5.5.1. RL-Based Distillation Strategies

- **Reward-Guided Learning:** Distillation loss functions are augmented with RL-inspired reward signals to guide the student toward robust decision-making.
- **Adaptive Sampling:** RL techniques dynamically adjust the sampling strategy for distillation data, prioritizing harder examples that maximize learning efficiency.

5.5.2. Case Study: RLHF in Distilled Models

OpenAI's reinforcement learning from human feedback (RLHF) techniques, originally designed for aligning large models like ChatGPT, have been adapted for distilling compact models that retain desirable conversational behaviors and ethical constraints.

5.6. Data-Free Model Compression

In some cases, access to the original training data may be restricted due to privacy concerns or data ownership policies. Data-free compression techniques bypass this limitation by synthesizing data or leveraging surrogate data for compression.

5.6.1. Popular Data-Free Methods

- **Zero-Shot Pruning:** This method prunes models based on intrinsic properties such as weight magnitude or gradient statistics without requiring labeled data.
- **Synthetic Data Generation:** Using generative models to create data distributions that mimic the original training set, enabling effective compression even without the original data.

5.6.2. Case Study: Privacy-Conscious NLP Systems

Data-free compression has gained traction in privacy-sensitive applications such as healthcare NLP systems, where access to proprietary medical records is restricted.

5.7. Continual Learning and Adaptive Compression

As LLMs are increasingly deployed in dynamic environments, continual learning strategies are emerging to maintain model efficiency over time [88]. Adaptive compression methods enable models to evolve incrementally while maintaining minimal resource consumption [89].

5.7.1. Adaptive Compression Techniques

- **Elastic Parameter Growth:** Dynamically allocates or prunes model parameters based on task complexity and evolving data distributions.
- **Lifelong Distillation:** Periodically distills knowledge from evolving teacher models to maintain student model robustness [90].

5.8. Summary and Future Directions

Emerging trends in model compression are increasingly driven by the convergence of algorithmic advancements and hardware optimizations [91]. Techniques such as hardware-aware pruning, low-rank adaptation, and NAS are proving essential for deploying efficient LLMs in real-world scenarios. Furthermore, innovative strategies like RL-guided distillation and data-free compression are addressing practical challenges such as data privacy and dynamic deployment. Future research is expected to focus on:

- Enhancing compression strategies for multimodal models that combine text, vision, and audio.
- Developing robust evaluation metrics that balance model size, inference speed, and predictive accuracy.
- Integrating compression techniques into mainstream NLP pipelines to enable scalable and sustainable AI deployment.

The following section presents an evaluation framework for benchmarking compressed LLMs across key performance dimensions [92].

6. Evaluation Metrics and Benchmarking

Accurately assessing the effectiveness of compressed language models is crucial to ensuring that performance gains are achieved without compromising model quality. Effective evaluation requires a combination of traditional NLP performance metrics, efficiency indicators, and deployment-specific considerations [93]. This section outlines key metrics, benchmarking frameworks, and best practices for evaluating compressed models [94].

6.1. Performance Metrics

Evaluating the accuracy, robustness, and generalization capabilities of compressed models is essential [95]. While traditional NLP benchmarks remain relevant, model compression introduces additional trade-offs that must be assessed through specialized metrics [96].

6.1.1. Accuracy and Task Performance

Maintaining task-specific performance is a primary goal when compressing language models [97]. Common metrics include:

- **Perplexity (PPL):** Commonly used in language modeling tasks, perplexity measures how well a model predicts unseen text. Lower perplexity values indicate better performance [98].
- **F1 Score:** Widely used for classification tasks, the F1 score balances precision and recall, ensuring compressed models retain predictive accuracy.
- **Exact Match (EM):** In QA systems and text generation tasks, EM measures the percentage of predictions that exactly match the expected output.
- **BLEU, ROUGE, and METEOR:** These metrics assess text generation quality by comparing generated text with reference outputs.

6.1.2. Efficiency Metrics

Compression techniques aim to enhance model efficiency, necessitating specialized metrics that capture resource utilization improvements.

- **Model Size (MB/GB):** The compressed model's total size, which directly impacts storage and memory requirements.
- **Parameter Count:** The number of trainable model parameters, serving as a key indicator of model complexity [99].
- **FLOPs (Floating Point Operations):** FLOPs measure the computational cost of inference, providing insights into latency and energy efficiency [100].
- **Inference Speed (ms/query):** This metric directly reflects model responsiveness in real-world applications.
- **Throughput (Tokens/Sec):** Throughput measures the number of tokens processed per second, a critical factor for high-traffic NLP systems.

6.1.3. Robustness and Generalization

Compressed models may be more vulnerable to adversarial inputs or distribution shifts [101]. Evaluating robustness ensures model stability across diverse environments.

- **Out-of-Distribution (OOD) Accuracy:** Measures how well a compressed model performs on data that deviates from the original training distribution [102].
- **Adversarial Robustness:** Evaluates the model's resistance to crafted adversarial examples that exploit weaknesses in language understanding.
- **Calibration Error:** Compressed models may become overconfident or underconfident in their predictions. Calibration metrics assess how well model probabilities reflect true likelihoods [103].

6.1.4. Energy and Environmental Impact

As sustainability becomes a growing concern in AI research, assessing the environmental footprint of compressed models is increasingly important [104].

- **Energy Consumption (kWh):** Measures the energy consumed during model inference, reflecting deployment efficiency.
- **Carbon Footprint (CO_2 Emissions):** This metric estimates the environmental impact of running compressed models on large-scale infrastructures.

6.2. Benchmarking Frameworks

To ensure fair and reproducible evaluations, researchers have developed standardized benchmarks for assessing compressed language models.

6.2.1. Popular NLP Benchmarks

Several established benchmarks are widely adopted in compression research:

- **GLUE (General Language Understanding Evaluation):** A comprehensive benchmark suite for evaluating NLP models across diverse tasks such as sentiment analysis, entailment, and paraphrase detection.
- **SuperGLUE:** An advanced version of GLUE designed for challenging NLP tasks that require deep reasoning and contextual understanding.
- **Eloquence Benchmark (ELOQ):** A specialized benchmark that evaluates compressed models for text generation fluency and coherence.
- **MLPerf Inference Benchmark:** This framework evaluates compressed models in real-time inference scenarios, measuring latency, throughput, and energy consumption.

6.2.2. Custom Benchmarking Pipelines

In addition to established frameworks, researchers often design custom benchmarking pipelines to evaluate compression techniques for domain-specific applications [105].

- **Edge-AI Benchmarks:** Tailored for mobile and IoT devices, these frameworks measure compressed model performance in constrained environments.
- **Latency-Aware NLP Pipelines:** Designed to evaluate the real-time responsiveness of compressed models in production environments.

6.3. Trade-Off Analysis in Model Compression

Balancing accuracy, efficiency, and deployment requirements is critical when evaluating compressed models. A comprehensive evaluation framework should incorporate trade-off analyses that highlight these competing priorities.

6.3.1. Accuracy vs. Latency Trade-Off

Compressed models may experience slight performance degradation as a result of aggressive compression. Plotting accuracy against latency reveals optimal trade-off points where performance is maintained with minimal inference delays [106].

6.3.2. Energy vs. Throughput Trade-Off

Energy-efficient models may require fewer FLOPs but may also limit throughput [107,108]. Evaluating energy efficiency alongside processing speed helps developers choose compression strategies suited to their deployment goals.

6.3.3. Memory Footprint vs [109]. Model Depth Trade-Off

Pruning and quantization techniques may reduce model size at the cost of deeper, more complex architectures. Evaluating memory efficiency in conjunction with model complexity ensures balanced design choices [110].

6.4. Best Practices for Model Evaluation

To ensure fair and reproducible assessments, model compression research should adhere to established best practices:

- Conduct evaluations on diverse datasets that capture real-world data distributions.
- Report multiple metrics, including both task performance and resource efficiency indicators.
- Compare results against strong baselines, such as uncompressed models or popular efficient architectures.
- Provide open-source code and evaluation pipelines to encourage transparency and reproducibility.

6.5. Summary

Evaluating compressed language models requires a multi-faceted approach that combines accuracy, efficiency, and robustness metrics. By leveraging established benchmarks, custom pipelines, and trade-off analyses, researchers can provide comprehensive assessments that guide the adoption of compressed models in real-world applications [111]. Future advancements in benchmarking are expected to focus on developing unified frameworks that integrate accuracy, latency, energy consumption, and scalability metrics into a cohesive evaluation methodology. The following section discusses practical deployment strategies for compressed language models, highlighting best practices for integration in real-world systems.

7. Deployment Strategies for Compressed Language Models

Deploying compressed language models effectively requires a combination of technical optimizations, infrastructure adjustments, and workflow adaptations. While compression techniques reduce computational and memory overhead, careful deployment strategies are essential to maximize these gains in real-world scenarios. This section outlines best practices, deployment architectures, and performance optimization techniques for deploying compressed models across various environments.

7.1. Infrastructure Considerations

The optimal deployment architecture for compressed language models depends on the target environment, hardware constraints, and scalability requirements [112]. Several deployment environments require distinct strategies:

7.1.1. Cloud-Based Deployment

Cloud platforms such as AWS, Google Cloud, and Microsoft Azure provide scalable infrastructure with flexible resource allocation [113]. Cloud deployment enables efficient scaling for high-demand applications [114].

Best Practices for Cloud Deployment

- Leverage auto-scaling features to dynamically adjust resource allocation based on request volume.
- Optimize compressed models using GPU-accelerated inference frameworks like NVIDIA TensorRT or Amazon Inferentia for improved latency [115].
- Employ model sharding techniques to distribute large language models across multiple cloud instances.

7.1.2. Edge and Mobile Deployment

Resource-constrained devices, such as smartphones, IoT systems, and embedded platforms, require lightweight and memory-efficient models.

Best Practices for Edge Deployment

- Use quantization techniques to reduce model precision for efficient execution on low-power processors [116].
- Deploy models with frameworks like TensorFlow Lite, ONNX Runtime, or Core ML, which are optimized for mobile and embedded environments.
- Implement caching mechanisms to reduce redundant computation and minimize latency.

7.1.3. On-Premises and Enterprise Systems

For organizations requiring data privacy, security, or offline capabilities, deploying compressed models on internal infrastructure may be preferable.

Best Practices for On-Premises Deployment

- Optimize compressed models for CPU inference by applying operator fusion and vectorized computations.
- Utilize containerization tools such as Docker or Kubernetes to streamline deployment and ensure scalability [117].
- Integrate monitoring tools to track model performance, detect data drift, and ensure stable inference behavior.

7.2. Serving Frameworks and Model APIs

Efficient serving frameworks play a crucial role in maximizing the benefits of compressed language models by optimizing inference performance.

7.2.1. Popular Serving Frameworks

- **FastAPI:** A lightweight Python framework ideal for serving compressed models with minimal latency overhead.
- **NVIDIA Triton Inference Server:** Optimized for multi-GPU and multi-model environments, Triton is effective for deploying compressed LLMs in scalable systems.
- **TorchServe:** Developed by the PyTorch team, TorchServe simplifies model packaging, inference, and scaling for production systems.

7.2.2. Model API Design Best Practices

To ensure efficient communication between client applications and compressed models:

- Implement batch inference to maximize throughput by processing multiple requests simultaneously [118].
- Utilize asynchronous inference pipelines to manage concurrent user requests effectively [119].
- Optimize data serialization formats (e.g., Protocol Buffers, MessagePack) to minimize network overhead [120].

7.3. Latency Optimization Techniques

Even with compression, achieving low latency in real-time applications requires additional optimizations [121].

7.3.1. Techniques for Reducing Latency

- **Model Batching:** Efficiently groups multiple inference requests, improving GPU utilization and reducing overhead.
- **Pipeline Parallelism:** Splits model layers across multiple devices to maximize parallel execution in large-scale models.
- **Distillation-Aware Caching:** Stores frequently queried outputs from a distilled student model to accelerate common predictions.

7.3.2. Case Study: Real-Time Translation Systems

Google Translate leverages quantized Transformer models combined with batching techniques to deliver low-latency translation services worldwide.

7.4. Resource Management and Scaling

As compressed models are integrated into large-scale systems, effective resource management ensures optimal performance [122].

7.4.1. Strategies for Resource Optimization

- **Dynamic Scaling:** Automatically adjusts computing resources based on inference workload fluctuations [123].
- **Load Balancing:** Distributes model requests across multiple servers to ensure even resource utilization.
- **Memory-Efficient Deployment:** Offloads inactive model components to disk and loads them dynamically during inference.

7.4.2. Case Study: E-Commerce Product Search

Amazon optimizes search recommendations by combining low-rank approximation techniques with load-balancing algorithms to deliver personalized results with minimal latency.

7.5. Security and Privacy Considerations

Compressed models are often deployed in sensitive environments where data protection and model integrity are paramount.

7.5.1. Techniques for Secure Model Deployment

- **Model Encryption:** Encrypts model weights to prevent unauthorized access during deployment.
- **Secure Enclaves:** Hardware-based security solutions such as Intel SGX enable encrypted model execution in trusted environments [124].
- **Adversarial Defense Mechanisms:** Apply adversarial training or robust optimization techniques to improve resistance against malicious inputs [125].

7.5.2. Case Study: Healthcare NLP Systems

Hospitals leveraging compressed models for clinical text analysis adopt secure enclave solutions to ensure patient data confidentiality and model security.

7.6. Continuous Integration and Deployment (CI/CD) Pipelines

Automating the deployment lifecycle ensures smooth model updates and efficient monitoring.

7.6.1. Key CI/CD Strategies for Compressed Models

- **Model Versioning:** Track compressed model versions to enable seamless rollbacks in case of performance degradation [126].
- **Canary Deployment:** Gradually deploy compressed models to a small subset of users before full rollout.
- **Performance Monitoring Tools:** Tools like Prometheus and Grafana provide real-time insights into latency, throughput, and resource utilization.

7.7. Post-Deployment Evaluation and Maintenance

Continuous evaluation ensures compressed models maintain optimal performance in dynamic environments.

7.7.1. Best Practices for Post-Deployment Monitoring

- Regularly monitor accuracy and latency metrics to detect performance drift.
- Implement alerting systems to respond to unusual spikes in resource consumption or model errors [127].
- Periodically retrain compressed models to adapt to evolving data distributions.

7.8. Summary

Successful deployment of compressed language models requires a combination of optimized infrastructure, efficient serving frameworks, and adaptive resource management [128]. By leveraging cloud services, edge computing platforms, and effective latency reduction strategies, organizations can maximize the performance of compressed models in real-world systems. Security measures and CI/CD pipelines further ensure stable, scalable, and secure deployment [129]. The following section explores the key challenges and open research questions that remain in the field of model compression and deployment [130].

8. Challenges and Open Research Directions

Despite significant advancements in model compression techniques, several challenges persist that hinder their widespread adoption in real-world applications [131]. These challenges span algorithmic limitations, deployment constraints, and ethical considerations. Addressing these issues presents promising avenues for future research. This section outlines key challenges and explores open research directions for improving the efficiency, reliability, and fairness of compressed language models [132].

8.1. Preserving Model Performance Post-Compression

One of the primary challenges in model compression is retaining the original model's accuracy and generalization capabilities after applying aggressive reduction techniques.

8.1.1. Catastrophic Performance Degradation

Aggressive compression techniques, such as extreme pruning or low-bit quantization, can result in severe accuracy drops, especially in complex NLP tasks requiring nuanced language understanding.

Research Directions:

- Develop adaptive compression algorithms that automatically balance model sparsity with performance preservation.
- Investigate hybrid approaches that combine knowledge distillation with selective pruning to minimize information loss.
- Explore techniques to incorporate uncertainty estimation in compressed models to improve their robustness under ambiguous inputs.

8.1.2. Fine-Tuning Challenges

Compressed models often require specialized fine-tuning strategies to regain lost performance [133]. However, fine-tuning may inadvertently introduce bias or overfit the model to specific data distributions.

Research Directions:

- Design lightweight fine-tuning strategies tailored for compressed architectures [134].
- Develop methods for transfer learning in compressed models to improve generalization without extensive retraining.

8.2. Trade-Offs Between Efficiency and Model Robustness

Compression methods may inadvertently compromise model robustness, making models more vulnerable to adversarial attacks, data perturbations, or domain shifts.

8.2.1. Adversarial Vulnerability

Compressed models may suffer from reduced resistance to adversarial inputs due to lost redundancies in the neural architecture.

Research Directions:

- Develop compression-aware adversarial training techniques to improve robustness [135].
- Investigate the role of redundant parameters in enhancing model stability and develop methods that selectively preserve crucial redundancies [136].

8.2.2. Robustness to Data Drift

Compressed models are often trained on static datasets, making them susceptible to performance degradation when deployed in dynamic environments.

Research Directions:

- Design adaptive compression frameworks that update model parameters incrementally to respond to distribution shifts.
- Develop self-correcting mechanisms that detect and mitigate drift-induced errors during inference.

8.3. Scalability and Large-Scale Model Compression

As language models continue to grow in size, compression techniques must scale effectively to handle trillion-parameter architectures.

8.3.1. Scaling Compression Algorithms

Many existing compression methods struggle to scale efficiently for extremely large models due to computational bottlenecks.

Research Directions:

- Design scalable pruning frameworks that exploit distributed computing for compressing massive models.
- Develop low-rank approximation algorithms that adaptively scale with increasing model size [137].

8.3.2. Efficient Distributed Training for Compressed Models

Training compressed models in distributed environments poses challenges in synchronizing gradients, managing data parallelism, and minimizing communication overhead [138].

Research Directions:

- Investigate decentralized training strategies that reduce communication overhead during compression.
- Explore federated learning techniques tailored for compressed models to enhance scalability and privacy.

8.4. Hardware and Deployment Limitations

Compressed models must be optimized to leverage the diverse hardware environments used in practical deployments, including CPUs, GPUs, and specialized accelerators.

8.4.1. Hardware-Aware Compression

Many compression techniques are hardware-agnostic, resulting in suboptimal performance on specialized hardware platforms.

Research Directions:

- Develop hardware-aware compression algorithms that account for the memory hierarchy, cache structure, and computational constraints of modern hardware.
- Investigate compiler-level optimizations that improve the inference efficiency of compressed models.

8.4.2. Edge Device Constraints

Deploying compressed models on resource-constrained devices, such as mobile phones and IoT systems, presents significant challenges [139].

Research Directions:

- Design lightweight compression frameworks that prioritize memory footprint reduction without compromising latency [140].
- Develop adaptive model loading strategies that selectively load portions of the model based on available resources [141].

8.5. Fairness, Bias, and Ethical Considerations

Compressed models may inadvertently amplify biases present in the original model or develop new biases during compression.

8.5.1. Bias Amplification in Compressed Models

Aggressive pruning or quantization may disproportionately impact minority class representations, resulting in skewed outputs.

Research Directions:

- Develop fairness-preserving compression techniques that prioritize the retention of critical features related to minority groups [142].
- Establish evaluation benchmarks that assess compressed models' fairness across demographic groups and sensitive attributes [143].

8.5.2. Privacy Risks in Compressed Models

Compressed models may inadvertently encode sensitive information, posing privacy risks when deployed in real-world applications.

Research Directions:

- Explore privacy-preserving compression techniques that mitigate information leakage while reducing model size.

- Investigate differentially private compression methods that enhance data confidentiality during training and deployment.

8.6. Lack of Standardized Evaluation Frameworks

Although numerous evaluation metrics exist, there is no universally accepted framework for assessing compressed language models across diverse application domains [144].

8.6.1. Evaluation Benchmark Gaps

Current benchmarks primarily assess accuracy, while real-world performance considerations such as energy efficiency, inference latency, and memory usage are often overlooked.

Research Directions:

- Develop comprehensive benchmarking frameworks that integrate task accuracy, latency, energy efficiency, and robustness metrics.
- Design dynamic evaluation pipelines that assess compressed models in real-time production environments [145].

8.7. Summary

The field of model compression presents numerous challenges that require innovative solutions to balance performance, efficiency, and fairness. Future research should focus on developing adaptive compression algorithms, enhancing robustness, and improving deployment strategies tailored to diverse hardware environments [146]. Additionally, addressing privacy concerns and ensuring fairness in compressed models will be crucial for their ethical deployment. The next section concludes this survey, summarizing key insights and outlining promising future directions for model compression in NLP [147].

9. Conclusion and Future Directions

Model compression has emerged as a crucial research area for improving the efficiency of large language models (LLMs) while maintaining their performance. As the demand for deploying LLMs in real-world applications grows, compression techniques have become essential for reducing computational costs, lowering energy consumption, and enabling deployment in resource-constrained environments. This survey has explored various compression strategies, including pruning, quantization, knowledge distillation, low-rank approximation, and neural architecture search [148]. Additionally, we have examined evaluation metrics, deployment strategies, and the remaining challenges in this evolving field.

9.1. Key Insights

Throughout this survey, several key insights have emerged:

- **Compression Trade-Offs:** Effective compression requires balancing model size reduction with minimal performance degradation [149]. Techniques such as structured pruning and mixed-precision quantization have shown promising results in preserving model accuracy while significantly reducing computational overhead [150].
- **Distillation as a Core Strategy:** Knowledge distillation has proven highly effective in transferring the capabilities of large teacher models to smaller, student models. Combining distillation with other compression techniques has emerged as a powerful strategy for improving efficiency without sacrificing task performance [151].
- **Hardware-Aware Optimization:** Efficient deployment requires tailoring compression techniques to the capabilities of target hardware, including GPUs, TPUs, and edge devices. Hardware-aware pruning and quantization frameworks have demonstrated considerable improvements in latency and energy efficiency.

- **Evaluation Beyond Accuracy:** While task accuracy remains essential, comprehensive evaluation frameworks that include latency, throughput, and robustness metrics are necessary to assess compressed models' real-world performance [152].
- **Security and Fairness Concerns:** As compressed models are increasingly deployed in sensitive domains, ensuring fairness, privacy, and robustness against adversarial threats is critical to their responsible use.

9.2. Future Directions

Despite recent advancements, several open research avenues remain for improving the efficiency, scalability, and reliability of compressed language models:

9.2.1. Adaptive and Dynamic Compression Techniques

Future compression methods should focus on adaptively adjusting model size and complexity based on real-time conditions. Dynamic compression strategies that adjust model parameters during inference can optimize efficiency without compromising performance.

Potential Research Directions:

- Develop compression frameworks that automatically adapt model sparsity based on workload characteristics.
- Explore dynamic quantization techniques that modify precision levels in response to changing resource constraints.

9.2.2. Energy-Efficient Compression Strategies

Given the rising environmental concerns associated with large-scale AI models, future research should prioritize energy-aware compression frameworks that minimize carbon footprints during both training and inference [153].

Potential Research Directions:

- Design compression-aware training algorithms that incorporate energy consumption as an optimization criterion.
- Develop model-serving frameworks that intelligently allocate hardware resources to minimize power usage.

9.2.3. Robust and Fair Compression Algorithms

Mitigating the potential for performance degradation, bias amplification, and adversarial vulnerabilities in compressed models remains a vital research area.

Potential Research Directions:

- Develop compression techniques that explicitly preserve model fairness and mitigate bias.
- Investigate adversarially robust compression algorithms to ensure model security in high-risk environments [154].

9.2.4. Scalable Compression for Large-Scale Models

The increasing size of language models presents scalability challenges for traditional compression methods [155]. Future research must explore scalable frameworks that efficiently handle trillion-parameter models.

Potential Research Directions:

- Develop distributed compression frameworks that leverage parallel processing for scalable model reduction [156].

- Investigate low-rank approximation methods capable of compressing extremely large transformer architectures.

9.2.5. Cross-Modal Compression Techniques

As multimodal models combining text, image, and audio data become increasingly popular, novel compression strategies will be required to address their complexity [157].

Potential Research Directions:

- Design unified compression frameworks that efficiently compress multimodal representations.
- Develop task-aware compression strategies that adapt compression techniques to specific cross-modal tasks [158].

9.2.6. Benchmarking and Evaluation Frameworks

Standardized evaluation frameworks are needed to assess compressed models' accuracy, efficiency, robustness, and fairness comprehensively [159].

Potential Research Directions:

- Develop end-to-end benchmarking pipelines that assess compressed model performance under real-world conditions.
- Establish community-driven leaderboards to track advancements in compression research [160].

9.3. Final Remarks

Model compression will continue to play a pivotal role in the future of NLP and artificial intelligence, enabling powerful language models to operate efficiently across diverse platforms. By addressing existing challenges and exploring new research directions, the field can advance toward building lightweight, scalable, and robust models that democratize access to AI technologies.

As large language models become integral to everyday applications, continued innovation in model compression will be crucial to ensuring that these powerful tools are accessible, cost-effective, and environmentally sustainable.

References

1. Chen, Z.; Gao, Q.; Bosselut, A.; Sabharwal, A.; Richardson, K. Disco: distilling counterfactuals with large language models. In Proceedings of the Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2023, pp. 5514–5528.
2. Kaushal, A.; Vaidhya, T.; Rish, I. Lord: Low rank decomposition of monolingual code llms for one-shot compression. *arXiv preprint arXiv:2309.14021* **2023**.
3. Sanh, V.; Wolf, T.; Rush, A. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems* **2020**, *33*, 20378–20389.
4. Wu, X.; Yao, Z.; He, Y. ZeroQuant-FP: A Leap Forward in LLMs Post-Training W4A8 Quantization Using Floating-Point Formats. *arXiv preprint arXiv:2307.09782* **2023**.
5. Li, B.; Kong, Z.; Zhang, T.; Li, J.; Li, Z.; Liu, H.; Ding, C. Efficient transformer-based large scale language representations using hardware-friendly block structured pruning. *arXiv preprint arXiv:2009.08065* **2020**.
6. Gu, Y.; Zhang, S.; Usuyama, N.; Woldesenbet, Y.; Wong, C.; Sanapathi, P.; Wei, M.; Valluri, N.; Strandberg, E.; Naumann, T.; et al. Distilling large language models for biomedical knowledge extraction: A case study on adverse drug events. *arXiv preprint arXiv:2307.06439* **2023**.
7. Li, X.; Meng, Y.; Zhou, M.; Han, Q.; Wu, F.; Li, J. Sac: Accelerating and structuring self-attention via sparse adaptive connection. *Advances in Neural Information Processing Systems* **2020**, *33*, 16997–17008.
8. Dasgupta, S.; Cohn, T.; Baldwin, T. Cost-effective Distillation of Large Language Models. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2023.
9. Lee, J.H.; Kim, J.; Kwon, S.J.; Lee, D. FlexRound: Learnable Rounding based on Element-wise Division for Post-Training Quantization. *arXiv preprint arXiv:2306.00317* **2023**.
10. Li, X.L.; Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190* **2021**.

11. Liu, J.; Gong, R.; Wei, X.; Dong, Z.; Cai, J.; Zhuang, B. Qllm: Accurate and efficient low-bitwidth quantization for large language models. *arXiv preprint arXiv:2310.08041* **2023**.
12. Shen, S.; Dong, Z.; Ye, J.; Ma, L.; Yao, Z.; Gholami, A.; Mahoney, M.W.; Keutzer, K. Q-bert: Hessian based ultra low precision quantization of bert. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2020, Vol. 34, pp. 8815–8821.
13. Kim, Y.J.; Henry, R.; Fahim, R.; Awadalla, H.H. Finequant: Unlocking efficiency with fine-grained weight-only quantization for llms. *arXiv preprint arXiv:2308.09723* **2023**.
14. Zhao, Z.; Liu, Y.; Chen, L.; Liu, Q.; Ma, R.; Yu, K. An investigation on different underlying quantization schemes for pre-trained language models. In Proceedings of the Natural Language Processing and Chinese Computing: 9th CCF International Conference, NLPCC 2020, Zhengzhou, China, October 14–18, 2020, Proceedings, Part I 9. Springer, 2020, pp. 359–371.
15. Magister, L.C.; Mallinson, J.; Adamek, J.; Malmi, E.; Severyn, A. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410* **2022**.
16. Prasanna, S.; Rogers, A.; Rumshisky, A. When BERT plays the lottery, all tickets are winning. *arXiv preprint arXiv:2005.00561* **2020**.
17. Chee, J.; Cai, Y.; Kuleshov, V.; De Sa, C. QuIP: 2-Bit Quantization of Large Language Models With Guarantees. *arXiv preprint arXiv:2307.13304* **2023**.
18. Srinivasan, V.; Gandhi, D.; Thakker, U.; Prabhakar, R. Training Large Language Models Efficiently with Sparsity and Dataflow. *arXiv preprint arXiv:2304.05511* **2023**.
19. Clark, A.; De Las Casas, D.; Guy, A.; Mensch, A.; Paganini, M.; Hoffmann, J.; Damoc, B.; Hechtman, B.; Cai, T.; Borgeaud, S.; et al. Unified scaling laws for routed language models. In Proceedings of the International Conference on Machine Learning. PMLR, 2022, pp. 4057–4086.
20. So, D.; Le, Q.; Liang, C. The evolved transformer. In Proceedings of the International conference on machine learning. PMLR, 2019, pp. 5877–5886.
21. Zhu, Y.; Wichers, N.; Lin, C.C.; Wang, X.; Chen, T.; Shu, L.; Lu, H.; Liu, C.; Luo, L.; Chen, J.; et al. SiRA: Sparse Mixture of Low Rank Adaptation. *ArXiv* **2023**, *abs/2311.09179*.
22. Chen, T.; Frankle, J.; Chang, S.; Liu, S.; Zhang, Y.; Wang, Z.; Carbin, M. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems* **2020**, *33*, 15834–15846.
23. Cui, B.; Li, Y.; Zhang, Z. Joint structured pruning and dense knowledge distillation for efficient transformer model compression. *Neurocomputing* **2021**, *458*, 56–69.
24. Wu, S.; Chen, H.; Quan, X.; Wang, Q.; Wang, R. AD-KD: Attribution-Driven Knowledge Distillation for Language Model Compression. *arXiv preprint arXiv:2305.10010* **2023**.
25. Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* **2017**.
26. Zoph, B.; Bello, I.; Kumar, S.; Du, N.; Huang, Y.; Dean, J.; Shazeer, N.; Fedus, W. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906* **2022**.
27. Wang, L.; Yang, N.; Wei, F. Learning to retrieve in-context examples for large language models. *arXiv preprint arXiv:2307.07164* **2023**.
28. Xin, J.; Tang, R.; Lee, J.; Yu, Y.; Lin, J.J. DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2020.
29. Zniyed, Y.; Nguyen, T.P.; et al. Efficient tensor decomposition-based filter pruning. *Neural Networks* **2024**, *178*, 106393.
30. Shen, Z.; Zhang, M.; Zhao, H.; Yi, S.; Li, H. Efficient attention: Attention with linear complexities. In Proceedings of the Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2021, pp. 3531–3539.
31. Zadeh, A.H.; Edo, I.; Awad, O.M.; Moshovos, A. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. In Proceedings of the 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2020, pp. 811–824.
32. Jacobs, R.A.; Jordan, M.I.; Nowlan, S.J.; Hinton, G.E. Adaptive mixtures of local experts. *Neural computation* **1991**, *3*, 79–87.
33. Voita, E.; Talbot, D.; Moiseev, F.; Sennrich, R.; Titov, I. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418* **2019**.
34. Correia, G.M.; Niculae, V.; Martins, A.F. Adaptively sparse transformers. *arXiv preprint arXiv:1909.00015* **2019**.

35. Mishra, A.; Latorre, J.A.; Pool, J.; Stosic, D.; Stosic, D.; Venkatesh, G.; Yu, C.; Micikevicius, P. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378* **2021**.
36. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* **2016**.
37. Zhang, Y.; Zhao, L.; Cao, S.; Wang, W.; Cao, T.; Yang, F.; Yang, M.; Zhang, S.; Xu, N. Integer or Floating Point? New Outlooks for Low-Bit Quantization on Large Language Models. *arXiv preprint arXiv:2305.12356* **2023**.
38. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 2704–2713.
39. Anonymous. Outlier Weighed Layerwise Sparsity (OWL): A Missing Secret Sauce for Pruning LLMs to High Sparsity. In Proceedings of the Submitted to The Twelfth International Conference on Learning Representations, 2023. under review.
40. Frantar, E.; Alistarh, D. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems* **2022**, *35*, 4475–4488.
41. Goyal, S.; Choudhury, A.R.; Raje, S.; Chakaravarthy, V.; Sabharwal, Y.; Verma, A. PoWER-BERT: Accelerating BERT inference via progressive word-vector elimination. In Proceedings of the International Conference on Machine Learning. PMLR, 2020, pp. 3690–3699.
42. Katharopoulos, A.; Vyas, A.; Pappas, N.; Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In Proceedings of the International conference on machine learning. PMLR, 2020, pp. 5156–5165.
43. Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; Li, H. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems* **2016**, *29*.
44. Team, T.A.B. RayLLM. https://github.com/ray-project/ray-llm.
45. Li, R.; Murray, G.; Carenini, G. Mixture-of-Linguistic-Experts Adapters for Improving and Interpreting Pre-trained Language Models. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2023.
46. Kim, Y.J.; Awan, A.A.; Muzio, A.; Salinas, A.F.C.; Lu, L.; Hendy, A.; Rajbhandari, S.; He, Y.; Awadalla, H.H. Scalable and efficient moe training for multitask multilingual models. *arXiv preprint arXiv:2109.10465* **2021**.
47. Wu, B.; Dai, X.; Zhang, P.; Wang, Y.; Sun, F.; Wu, Y.; Tian, Y.; Vajda, P.; Jia, Y.; Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 10734–10742.
48. Chai, Y.; Gkountouras, J.; Ko, G.G.; Brooks, D.; Wei, G.Y. INT2. 1: Towards Fine-Tunable Quantized Large Language Models with Error Correction through Low-Rank Adaptation. *arXiv preprint arXiv:2306.08162* **2023**.
49. Zniyed, Y.; Nguyen, T.P.; et al. Enhanced network compression through tensor decompositions and pruning. *IEEE Transactions on Neural Networks and Learning Systems* **2024**.
50. Narang, S.; Undersander, E.; Diamos, G. Block-sparse recurrent neural networks. *arXiv preprint arXiv:1711.02782* **2017**.
51. Fan, A.; Grave, E.; Joulin, A. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556* **2019**.
52. Kim, S.; Gholami, A.; Yao, Z.; Mahoney, M.W.; Keutzer, K. I-bert: Integer-only bert quantization. In Proceedings of the International conference on machine learning. PMLR, 2021, pp. 5506–5518.
53. Scao, T.L.; Fan, A.; Akiki, C.; Pavlick, E.; Ilic, S.; Hesslow, D.; Castagné, R.; Luccioni, A.S.; Yvon, F.; Gallé, M.; et al. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. *CoRR* **2022**, *abs/2211.05100*.
54. Gholami, A.; Kim, S.; Dong, Z.; Yao, Z.; Mahoney, M.W.; Keutzer, K. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*; Chapman and Hall/CRC, 2022; pp. 291–326.
55. Wang, W.; Wei, F.; Dong, L.; Bao, H.; Yang, N.; Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems* **2020**, *33*, 5776–5788.
56. Li, Y.; Niu, L.; Zhang, X.; Liu, K.; Zhu, J.; Kang, Z. E-Sparse: Boosting the Large Language Model Inference through Entropy-based N: M Sparsity. *arXiv preprint arXiv:2310.15929* **2023**.
57. Zhao, C.; Hua, T.; Shen, Y.; Lou, Q.; Jin, H. Automatic mixed-precision quantization search of BERT. *arXiv preprint arXiv:2112.14938* **2021**.

58. Wu, M.; Waheed, A.; Zhang, C.; Abdul-Mageed, M.; Aji, A.F. Lamini-lm: A diverse herd of distilled models from large-scale instructions. *arXiv preprint arXiv:2304.14402* **2023**.
59. Williams, M.; Aletras, N. How Does Calibration Data Affect the Post-training Pruning and Quantization of Large Language Models? *arXiv preprint arXiv:2311.09755* **2023**.
60. Li, Y.; Yu, Y.; Liang, C.; He, P.; Karampatziakis, N.; Chen, W.; Zhao, T. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659* **2023**.
61. Jiang, Y.; Chan, C.; Chen, M.; Wang, W. Lion: Adversarial distillation of closed-source large language model. *arXiv preprint arXiv:2305.12870* **2023**.
62. Radosavovic, I.; Kosaraju, R.P.; Girshick, R.B.; He, K.; Dollár, P. Designing Network Design Spaces. In Proceedings of the Conference on Computer Vision and Pattern Recognition, CVPR, 2020, pp. 10425–10433.
63. Shao, H.; Liu, B.; Qian, Y. One-Shot Sensitivity-Aware Mixed Sparsity Pruning for Large Language Models. *arXiv preprint arXiv:2310.09499* **2023**.
64. Guo, S.; Xu, J.; Zhang, L.L.; Yang, M. Compresso: Structured Pruning with Collaborative Prompting Learns Compact Large Language Models. *arXiv preprint arXiv:2310.05015* **2023**.
65. Zuo, S.; Zhang, Q.; Liang, C.; He, P.; Zhao, T.; Chen, W. Moebert: from bert to mixture-of-experts via importance-guided adaptation. *arXiv preprint arXiv:2204.07675* **2022**.
66. Kurtic, E.; Campos, D.; Nguyen, T.; Frantar, E.; Kurtz, M.; Fineran, B.; Goin, M.; Alistarh, D. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. *arXiv preprint arXiv:2203.07259* **2022**.
67. Lin, Z.; Qu, G.; Chen, Q.; Chen, X.; Chen, Z.; Huang, K. Pushing large language models to the 6g edge: Vision, challenges, and opportunities. *arXiv preprint arXiv:2309.16739* **2023**.
68. Ren, S.; Zhu, K.Q. Low-Rank Prune-And-Factorize for Language Model Compression. *arXiv preprint arXiv:2306.14152* **2023**.
69. Sun, S.; Cheng, Y.; Gan, Z.; Liu, J. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355* **2019**.
70. Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S.R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* **2018**.
71. Jaiswal, A.K.; Liu, S.; Chen, T.; Ding, Y.; Wang, Z. Instant soup: Cheap pruning ensembles in a single pass can draw lottery tickets from large models. In Proceedings of the International Conference on Machine Learning. PMLR, 2023, pp. 14691–14701.
72. Liu, K.; Wu, T.; Liu, C.; Guo, G. Dynamic group transformer: A general vision transformer backbone with dynamic group attention. *arXiv preprint arXiv:2203.03937* **2022**.
73. Team, T.W.J. Saxml. https://github.com/google/saxml.
74. Agarwal, R.; Vieillard, N.; Zhou, Y.; Stanczyk, P.; Ramos, S.; Geist, M.; Bachem, O. Generalized knowledge distillation for auto-regressive language models. *arXiv preprint arXiv:2306.13649* **2023**.
75. Xu, Z.; Liu, Z.; Chen, B.; Tang, Y.; Wang, J.; Zhou, K.; Hu, X.; Shrivastava, A. Compress, Then Prompt: Improving Accuracy-Efficiency Trade-off of LLM Inference with Transferable Prompt. *arXiv preprint arXiv:2305.11186* **2023**.
76. Yang, Z.; Cui, Y.; Yao, X.; Wang, S. Gradient-based Intra-attention Pruning on Pre-trained Language Models. *arXiv preprint arXiv:2212.07634* **2022**.
77. Wang, Y.; Agarwal, S.; Mukherjee, S.; Liu, X.; Gao, J.; Awadallah, A.H.; Gao, J. AdaMix: Mixture-of-adaptations for parameter-efficient model tuning. *arXiv preprint arXiv:2210.17451* **2022**.
78. Anonymous. Pushing Gradient towards Zero: A Novel Pruning Method for Large Language Models, 2024.
79. Kim, M.; Lee, S.; Lee, J.; Hong, S.; Chang, D.S.; Sung, W.; Choi, J. Token-Scaled Logit Distillation for Ternary Weight Generative Language Models. *arXiv preprint arXiv:2308.06744* **2023**.
80. Dettmers, T.; Lewis, M.; Belkada, Y.; Zettlemoyer, L. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339* **2022**.
81. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* **2021**.
82. Park, G.; Park, B.; Kwon, S.J.; Kim, B.; Lee, Y.; Lee, D. nuQmm: Quantized matmul for efficient inference of large-scale generative language models. *arXiv preprint arXiv:2206.09557* **2022**.
83. Shao, W.; Chen, M.; Zhang, Z.; Xu, P.; Zhao, L.; Li, Z.; Zhang, K.; Gao, P.; Qiao, Y.; Luo, P. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137* **2023**.
84. Csord'as, R.; Irie, K.; Schmidhuber, J. Approximating Two-Layer Feedforward Networks for Efficient Transformers. *ArXiv* **2023**, *abs/2310.10837*.

85. Xue, F.; Zheng, Z.; Fu, Y.; Ni, J.; Zheng, Z.; Zhou, W.; You, Y. OpenMoE: An Early Effort on Open Mixture-of-Experts Language Models, 2024, [arXiv:cs.CL/2402.01739].
86. Sahu, G.; Vechtomova, O.; Bahdanau, D.; Laradji, I.H. Promptmix: A class boundary augmentation method for large language model distillation. *arXiv preprint arXiv:2310.14192* **2023**.
87. Lee, C.; Jin, J.; Kim, T.; Kim, H.; Park, E. OWQ: Lessons learned from activation outliers for weight quantization in large language models. *arXiv preprint arXiv:2306.02272* **2023**.
88. Yvinec, E.; Dapgony, A.; Cord, M.; Bailly, K. REx: Data-Free Residual Quantization Error Expansion. *arXiv preprint arXiv:2203.14645* **2022**.
89. Park, M.; You, J.; Nagel, M.; Chang, S. Quadapter: Adapter for GPT-2 Quantization. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2022, 2022, pp. 2510–2517.
90. Gu, Y.; Dong, L.; Wei, F.; Huang, M. Knowledge Distillation of Large Language Models. *arXiv preprint arXiv:2306.08543* **2023**.
91. Zhang, Z.; Gu, Y.; Han, X.; Chen, S.; Xiao, C.; Sun, Z.; Yao, Y.; Qi, F.; Guan, J.; Ke, P.; et al. Cpm-2: Large-scale cost-effective pre-trained language models. *AI Open* **2021**, *2*, 216–224.
92. Zhang, Q.; Zuo, S.; Liang, C.; Bukharin, A.; He, P.; Chen, W.; Zhao, T. Platon: Pruning large transformer models with upper confidence bound of weight importance. In Proceedings of the International Conference on Machine Learning. PMLR, 2022, pp. 26809–26823.
93. An, Y.; Zhao, X.; Yu, T.; Tang, M.; Wang, J. Fluctuation-based Adaptive Structured Pruning for Large Language Models. *arXiv preprint arXiv:2312.11983* **2023**.
94. Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; Hashimoto, T.B. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca, 2023.
95. Xia, M.; Gao, T.; Zeng, Z.; Chen, D. Sheared LLaMA: Accelerating Language Model Pre-training via Structured Pruning. *arXiv preprint arXiv:2310.06694* **2023**.
96. Yang, N.; Jang, Y.; Lee, H.; Jeong, S.; Jung, K. Task-specific Compression for Multi-task Language Models using Attribution-based Pruning. In Proceedings of the Findings of the Association for Computational Linguistics: EACL 2023, 2023, pp. 582–592.
97. Zhu, X.; Qi, B.; Zhang, K.; Long, X.; Zhou, B. PaD: Program-aided Distillation Specializes Large Models in Reasoning. *arXiv preprint arXiv:2305.13888* **2023**.
98. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* **2021**.
99. Ye, D.; Lin, Y.; Huang, Y.; Sun, M. TR-BERT: Dynamic Token Reduction for Accelerating BERT Inference. In Proceedings of the North American Chapter of the Association for Computational Linguistics, 2021.
100. Valipour, M.; Rezagholizadeh, M.; Kobyzev, I.; Ghodsi, A. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558* **2022**.
101. Zhang, H.; Cisse, M.; Dauphin, Y.N.; Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* **2017**.
102. Xiong, Y.; Zeng, Z.; Chakraborty, R.; Tan, M.; Fung, G.; Li, Y.; Singh, V. Nyströmformer: A nyström-based algorithm for approximating self-attention. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2021, Vol. 35, pp. 14138–14148.
103. Tolstikhin, I.O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems* **2021**, *34*, 24261–24272.
104. Khetan, A.; Karnin, Z. schuBERT: Optimizing elements of BERT. *arXiv preprint arXiv:2005.06628* **2020**.
105. Yang, A.; Lin, J.; Men, R.; Zhou, C.; Jiang, L.; Jia, X.; Wang, A.; Zhang, J.; Wang, J.; Li, Y.; et al. M6-t: Exploring sparse expert models and beyond. *arXiv preprint arXiv:2105.15082* **2021**.
106. Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C.H.; Gonzalez, J.; Zhang, H.; Stoica, I. Efficient memory management for large language model serving with pagedattention. In Proceedings of the Proceedings of the 29th Symposium on Operating Systems Principles, 2023, pp. 611–626.
107. Hou, L.; Huang, Z.; Shang, L.; Jiang, X.; Chen, X.; Liu, Q. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems* **2020**, *33*, 9782–9793.
108. Riquelme, C.; Puigcerver, J.; Mustafa, B.; Neumann, M.; Jenatton, R.; Susano Pinto, A.; Keysers, D.; Houlsby, N. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems* **2021**, *34*, 8583–8595.
109. McCarley, J.; Chakravarti, R.; Sil, A. Structured pruning of a bert-based question answering model. *arXiv preprint arXiv:1910.06360* **2019**.

110. Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T.B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* **2020**.
111. Pan, H.; Wang, C.; Qiu, M.; Zhang, Y.; Li, Y.; Huang, J. Meta-KD: A meta knowledge distillation framework for language model compression across domains. *arXiv preprint arXiv:2012.01266* **2020**.
112. Aminabadi, R.Y.; Rajbhandari, S.; Awan, A.A.; Li, C.; Li, D.; Zheng, E.; Ruwase, O.; Smith, S.; Zhang, M.; Rasley, J.; et al. DeepSpeed-inference: enabling efficient inference of transformer models at unprecedented scale. In Proceedings of the SC22: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2022, pp. 1–15.
113. Diao, S.; Xu, T.; Xu, R.; Wang, J.; Zhang, T. Mixture-of-Domain-Adapters: Decoupling and Injecting Domain Knowledge to Pre-trained Language Models' Memories. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2023.
114. Lingle, L.D. Transformer-VQ: Linear-Time Transformers via Vector Quantization. *arXiv preprint arXiv:2309.16354* **2023**.
115. Zhou, Y.; Lei, T.; Liu, H.; Du, N.; Huang, Y.; Zhao, V.; Dai, A.M.; Le, Q.V.; Laudon, J.; et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems* **2022**, *35*, 7103–7114.
116. Fu, Y.; Peng, H.; Ou, L.; Sabharwal, A.; Khot, T. Specializing Smaller Language Models towards Multi-Step Reasoning. *arXiv preprint arXiv:2301.12726* **2023**.
117. Wu, Z.; Liu, Z.; Lin, J.; Lin, Y.; Han, S. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886* **2020**.
118. Boža, V. Fast and Optimal Weight Update for Pruned Large Language Models, 2024, [arXiv:cs.CL/2401.02938].
119. Zaken, E.B.; Ravfogel, S.; Goldberg, Y. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199* **2021**.
120. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* **2020**, *21*, 5485–5551.
121. Baines, M.; Bhosale, S.; Caggiano, V.; Goyal, N.; Goyal, S.; Ott, M.; Lefaudeux, B.; Liptchinsky, V.; Rabbat, M.; Sheiffer, S.; et al. Fairscale: A general purpose modular pytorch library for high performance and large scale training, 2021. https://github.com/facebookresearch/fairscale.
122. Rasley, J.; Rajbhandari, S.; Ruwase, O.; He, Y. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In Proceedings of the Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 3505–3506.
123. Gholami, S.; Omar, M. Can pruning make Large Language Models more efficient? *arXiv preprint arXiv:2310.04573* **2023**.
124. Yao, Z.; Wu, X.; Li, C.; Youn, S.; He, Y. ZeroQuant-V2: Exploring Post-training Quantization in LLMs from Comprehensive Study to Low Rank Compensation, 2023, [arXiv:cs.LG/2303.08302].
125. Wang, Z.; Huang, S.; Liu, Y.; Wang, J.; Song, M.; Zhang, Z.; Huang, H.; Wei, F.; Deng, W.; Sun, F.; et al. Democratizing reasoning ability: Tailored learning from large language model. *arXiv preprint arXiv:2310.13332* **2023**.
126. Fedus, W.; Zoph, B.; Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research* **2022**, *23*, 5232–5270.
127. Xie, Y.; Huang, S.; Chen, T.; Wei, F. MoEC: Mixture of Expert Clusters. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2023, Vol. 37, pp. 13807–13815.
128. Wan, A.; Dai, X.; Zhang, P.; He, Z.; Tian, Y.; Xie, S.; Wu, B.; Yu, M.; Xu, T.; Chen, K.; et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 12965–12974.
129. Hassibi, B.; Stork, D.G.; Wolff, G.J. Optimal brain surgeon and general network pruning. In Proceedings of the IEEE international conference on neural networks. IEEE, 1993, pp. 293–299.
130. Tay, Y.; Bahri, D.; Yang, L.; Metzler, D.; Juan, D.C. Sparse sinkhorn attention. In Proceedings of the International Conference on Machine Learning. PMLR, 2020, pp. 9438–9447.
131. Wang, J.; Chen, K.; Chen, G.; Shou, L.; McAuley, J. SkipBERT: Efficient Inference with Shallow Layer Skipping. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2022.
132. Peng, B.; Li, C.; He, P.; Galley, M.; Gao, J. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277* **2023**.
133. Roy, A.; Saffar, M.; Vaswani, A.; Grangier, D. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics* **2021**, *9*, 53–68.

134. Pfeiffer, J.; Kamath, A.; Rücklé, A.; Cho, K.; Gurevych, I. AdapterFusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247* **2020**.
135. Yang, K.; Liu, Z.; Cheng, P. MOSEC: Model Serving made Efficient in the Cloud, 2021.
136. Zhu, M.; Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878* **2017**.
137. Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; Metzler, D.; et al. Emergent Abilities of Large Language Models. *Trans. Mach. Learn. Res.* **2022**, *2022*.
138. Zuo, S.; Liu, X.; Jiao, J.; Kim, Y.J.; Hassan, H.; Zhang, R.; Zhao, T.; Gao, J. Taming sparsely activated transformer with stochastic experts. *arXiv preprint arXiv:2110.04260* **2021**.
139. Zhou, A.; Ma, Y.; Zhu, J.; Liu, J.; Zhang, Z.; Yuan, K.; Sun, W.; Li, H. Learning n: m fine-grained structured sparse neural networks from scratch. *arXiv preprint arXiv:2102.04010* **2021**.
140. Li, S.; Liu, H.; Bian, Z.; Fang, J.; Huang, H.; Liu, Y.; Wang, B.; You, Y. Colossal-ai: A unified deep learning system for large-scale parallel training. In Proceedings of the Proceedings of the 52nd International Conference on Parallel Processing, 2023, pp. 766–775.
141. Michel, P.; Levy, O.; Neubig, G. Are sixteen heads really better than one? *Advances in neural information processing systems* **2019**, *32*.
142. Child, R.; Gray, S.; Radford, A.; Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509* **2019**.
143. Pytorch. Pytorch JIT. https://github.com/pytorch/torchdynamo.
144. Louizos, C.; Welling, M.; Kingma, D.P. Learning sparse neural networks through L_0 regularization. *arXiv preprint arXiv:1712.01312* **2017**.
145. Liang, C.; Jiang, H.; Li, Z.; Tang, X.; Yin, B.; Zhao, T. Homodistil: Homotopic task-agnostic distillation of pre-trained transformers. *arXiv preprint arXiv:2302.09632* **2023**.
146. Frantar, E.; Alistarh, D. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot **2023**.
147. Zhou, W.; Zhang, S.; Gu, Y.; Chen, M.; Poon, H. Universalner: Targeted distillation from large language models for open named entity recognition. *arXiv preprint arXiv:2308.03279* **2023**.
148. Xia, M.; Zhong, Z.; Chen, D. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408* **2022**.
149. Team, T.M.M. composer. https://github.com/mosaicml/composer/, 2021.
150. Turc, I.; Chang, M.W.; Lee, K.; Toutanova, K. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962* **2019**.
151. Rahman, M.W.U.; Abrar, M.M.; Copening, H.G.; Hariri, S.; Shao, S.; Satam, P.; Salehi, S. Quantized Transformer Language Model Implementations on Edge Devices. *arXiv preprint arXiv:2310.03971* **2023**.
152. Liu, W.; Zhou, P.; Zhao, Z.; Wang, Z.; Deng, H.; Ju, Q. FastBERT: a Self-distilling BERT with Adaptive Inference Time. *ArXiv* **2020**, *abs/2004.02178*.
153. Jiang, X.; Wang, H.; Chen, Y.; Wu, Z.; Wang, L.; Zou, B.; Yang, Y.; Cui, Z.; Cai, Y.; Yu, T.; et al. Mnn: A universal and efficient inference engine. *Proceedings of Machine Learning and Systems* **2020**, *2*, 1–13.
154. Tang, R.; Lu, Y.; Liu, L.; Mou, L.; Vechtomova, O.; Lin, J. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136* **2019**.
155. Lepikhin, D.; Lee, H.; Xu, Y.; Chen, D.; Firat, O.; Huang, Y.; Krikun, M.; Shazeer, N.; Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668* **2020**.
156. Kim, J.; Lee, J.H.; Kim, S.; Park, J.; Yoo, K.M.; Kwon, S.J.; Lee, D. Memory-Efficient Fine-Tuning of Compressed Large Language Models via sub-4-bit Integer Quantization. *arXiv preprint arXiv:2305.14152* **2023**.
157. Dettmers, T.; Pagnoni, A.; Holtzman, A.; Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314* **2023**.
158. Lin, T.; Wang, Y.; Liu, X.; Qiu, X. A Survey of Transformers. *CoRR* **2021**, *abs/2106.04554*.
159. Ma, X.; Fang, G.; Wang, X. LLM-Pruner: On the Structural Pruning of Large Language Models. *arXiv preprint arXiv:2305.11627* **2023**.
160. team, M. MLC-LLM, 2023.