Lightweighting Large Models: A Review of Model Compression Techniques

Haoran Guan[1], Junhao Lv[1], Xi Chen[2], Lei Qi[1†]

[1]*Southeast University, Nanjing, China*

[2]*South China University of Technology, Guangzhou, China*

Abstract

With the rapid development of deep learning technology, neural network models have achieved remarkable results, but their large number of references and high computational resource requirements limit the wide range of applications.Therefore, model compression techniques emerge, aiming to reduce computational complexity, memory occupation, and energy consumption overhead of the models to meet the practical deployment requirements without sacrificing performance. In this paper, we systematically sort out the research results in the field of model compression in recent years. Firstly, we introduce the definition and goals of model compression, and discuss its challenge and evolution of the techniques. Then a variety of mainstream compression methods are elaborated in detail, including pruning, quantization, knowledge distillation, low-rank factorization, and parameter sharing, and analyze their principles, strengths, weaknesses, and applicability scenarios. We also provide a comparative analysis of these methods, highlighting their trade-offs in terms of compression rate, performance, and computational efficiency. In addition, we introduce commonly used datasets, evaluation metrics, and future research directions for model compression in image classification and natural language processing tasks. By comprehensively summarizing and analyzing the existing work, we expect to provide model compression researchers with a clear technology mapping and research roadmap to promote their researches.

Keywords: model compression; deep learning; large language models; literature review

1. Introduction

With the continuous evolution of deep learning technology, deep neural networks (DNN) have dominated the AI landscape, revolutionizing fields such as computer vision (CV), natural language processing (NLP), speech recognition, and recommender systems [1, 2, 3]. Since the breakthrough achieved by AlexNet [4] in 2012, deep learning models have consistently surpassed traditional methods in tasks such as machine translation [5, 6, 7], image generation [8, 9, 10], image classification [11, 12, 13], object detection [14, 15, 16], and speech-to-text [17, 18, 19]. Key innovations such as VGGNet [20], ResNet [11],and DenseNet [21] have notably improved the ability of models to generalize across various datasets and tasks.

†Corresponding author (Email: qilei@seu.edu.cn; ORCID:0000-0001-7091-0702)

In recent years, pre-trained models, particularly in NLP, have become the forefront of AI research. The introduction of BERT [22], for example, with its bidirectional transformer architecture, revolutionized tasks such as text classification, question and answering, and language translation. Following this, the GPT series, especially GPT-3 [23] and GPT-4 [24], demonstrated remarkable capabilities in generative tasks [25]. These models, leveraging massive amounts of data and powerful architectures, have achieved state-of-the-art performance in a variety of scenarios, including chatbots, automated writing, and even legal and medical question-answering.

As AI continues to evolve, models like the Vision Transformer [12] are beginning to replace traditional Convolutional Neural Networks (CNNs) in CV tasks. Pre-trained visual models are not only better at image classification, but also demonstrate stronger capabilities in a variety of tasks such as image generation and semantic segmentation [26]. Multimodal models such as CLIP [27], DALL-E [28] and LLaVA [29] are expanding the application of AI, offering powerful solutions in creative fields, including graphic generation and interactive experiences.

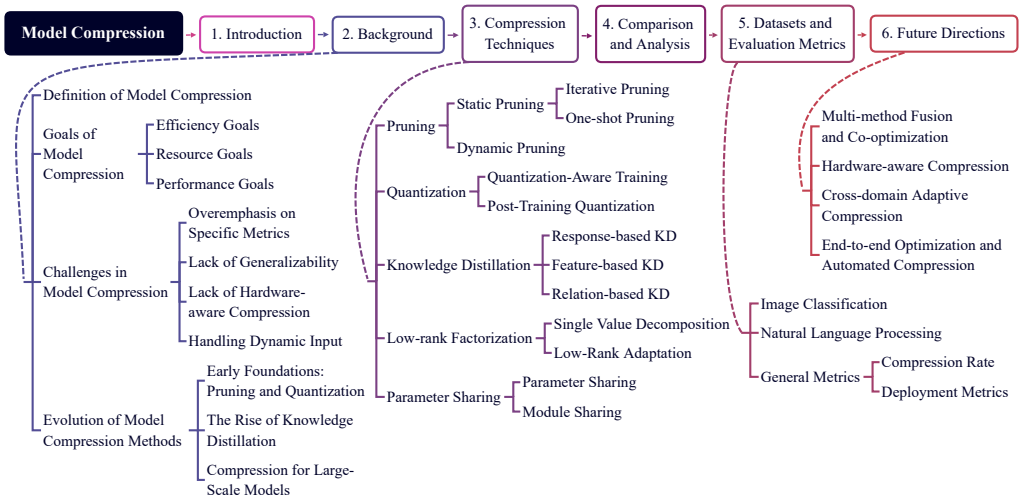

Figure 1: Overview of this survey.

Despite the success and capabilities of large pre-trained models, they are accompanied by significant computational and memory requirements. These models often require substantial computing resources for both training and inference, posing challenges for deployment on devices with limited resources, such as mobile phones, edge devices, and drones [30, 31]. Furthermore, the high energy consumption involved in training models like GPT-3, which requires thousands of GPUs and enormous amounts of electricity, makes them inaccessible to smaller research organizations or companies without substantial computational infrastructure [32, 33, 34].

For these reasons, as the demand for efficient AI applications grows, researchers are increasingly focused on reducing the computational complexity and memory footprint of deep learning models, without sacrificing the model performance. Methods such as model pruning, knowledge distillation, and quantization are being explored to enhance the deployment of large models while maintaining high accuracy. This shift towards more energy-efficient and resource-conscious models is essential to ensure the broader adoption of AI across various industries.

In this survey, we aim to provide a comprehensive overview of model compression tech-

niques, particularly focusing on the advances made in recent years. As shown in Fig. 1, we begin by defining the concept of model compression and its primary goals in §2, and discuss the challenges and trace the evolution of model compression techniques. In §3, we delve into the methods used for compressing models, including pruning, quantization, knowledge distillation, low-rank decomposition, and parameter sharing. These techniques are examined in detail, along with a comparative analysis of their advantages and challenges in §4, highlighting how each method addresses the core goals of compression. We further examines the datasets and evaluation metrics commonly used to assess compression methods in §5. Lastly, we conclude with a discussion on potential future research directions and the ongoing evolution of model compression techniques in §6.

2. Background

Model compression has emerged as a critical technique for deploying models on resource-constrained devices and enabling efficient inference in real-time applications. In this section, we first introduce the core concepts of model compression and the challenges that govern the compression process. We then discuss the specific tasks and goals of model compression, followed by an exploration of the historical development and related research areas of model compression.

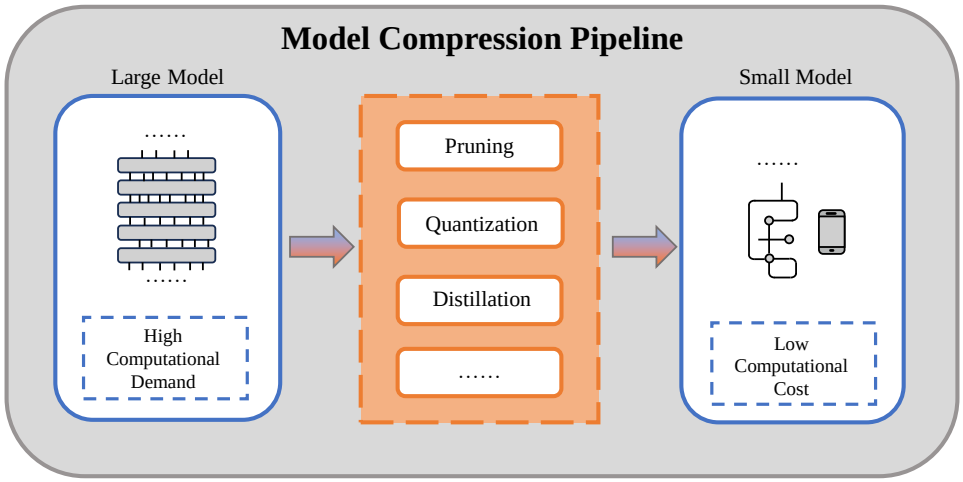

Figure 2: Illustration of the model compression pipeline. A large model with high computational demand is transformed into a smaller, more efficient model through compression techniques such as pruning, quantization, and distillation. The resulting lightweight model incurs lower computational cost and can be deployed on edge devices such as smartphones.

2.1. Definition of Model Compression

As shown in Fig. 2, model compression aims to learn a compressed mapping $f_{\theta'} : \mathcal{X} \mapsto \mathcal{Y}$ from an original, typically over-parameterized model f_θ. The goal is to significantly reduce the model's resource consumption, including the number of parameters, computational cost (measured in FLOPs), and memory footprint, while minimizing the degradation of its performance. This can be formally expressed as finding a compressed model with parameters θ' that satisfies the following conditions:

$$\begin{aligned} &\text{Parameter reduction:} && |\theta'| \leq \alpha|\theta|, \alpha \in (0,1), \\ &\text{Computation reduction:} && \text{FLOPs}(f_{\theta'}) \leq \beta\text{FLOPs}(f_\theta), \beta \in (0,1), \\ &\text{Memory reduction:} && \text{Mem}(f_{\theta'}) \leq \gamma\text{Mem}(f_\theta), \gamma \in (0,1), \end{aligned} \tag{1}$$

where α, β, and γ are compression ratios for parameters, computations, and memory, respectively. The primary and critical objective of model compression is to minimize the performance degradation $\Delta\mathcal{A} = \mathcal{A}(f_\theta) - \mathcal{A}(f_{\theta'})$, where $\mathcal{A}$ represents the accuracy of the model or other relevant performance metrics. This leads to the following optimization problem:

$$\min_{\theta'} \mathcal{L}(f_{\theta'}(x), y) \quad \text{s.t.} \quad \text{Resource}(f_{\theta'}) \leq \text{Budget}, \tag{2}$$

where, $\mathcal{L}$ is the task-specific loss function (e.g., cross-entropy for classification), and Resource($f_{\theta'}$) represents the resource constraints (e.g., FLOPs) that must be within a predefined Budget.

2.2. Goals of Model Compression

Model compression aims to reduce the size, computational overhead, and memory occupation of the model through a series of technical means, while maintaining the performance of the model in specific tasks as much as possible. With the increasing size of deep learning models, driven by the need to process complex data and achieve higher accuracy, model compression has become an important research direction. This is particularly crucial for deploying sophisticated models on resource-constrained devices and for enabling efficient inference in real-time applications. The core tasks of model compression can be summarized into the following categories.

• **Efficiency Goals:** Model compression aims to optimize the speed at which a model performs real-time inference, reducing the time required to make predictions. This can be achieved through techniques like low-rank factorization, quantization and structured pruning, all of which reduce computational complexity and enable faster processing on resource-constrained devices. This is critical in real-time scenarios such as autonomous driving, video analytics, and online recommendation systems, where low-latency responses are essential[35, 36, 30].

• **Resource Goals:** One of the core objectives of model compression is to *decrease the model's size*, making it more suitable for deployment on devices with limited storage, such as mobile phones, embedded systems, and IoT devices. *Efficient memory usage* is another key goal, aiming to reduce the model's memory footprint during inference. This helps ensure smooth deployment on devices with limited RAM and enables better performance in memory-constrained environments. Moreover, model compression seeks to *reduce the computational cost*, measured by operations such as FLOPs, allowing models to run more efficiently on devices with lower processing power [37, 38].

• **Performance Goals:** Besides improving efficiency and reducing resource consumption, it must be ensured that the compressed model retains as much of the performance of the original, uncompressed model. This implies that it is necessary to minimize model performance loss by carefully designing algorithms. This is especially crucial in high-stakes applications where maintaining high accuracy is paramount, such as medical diagnosis (e.g., detecting diseases from medical images), financial forecasting (e.g., critical investment decisions), and scientific research, where even small drops in accuracy could have severe consequences [39].

2.3. Challenges in Model Compression

Despite significant advances in model compression techniques, several challenges remain that limit their effectiveness and real-world applicability. The following are four major challenges that need to be addressed to improve the applicability of compression methods.

• **Trade-off between compression ratio and performance:** Although model compression techniques can effectively reduce the storage size and reasoning overhead of models, there is always a difficult trade-off between compression ratio and accuracy. While techniques such as pruning, quantization, and knowledge distillation can significantly reduce model size and computational requirements, they must be carefully designed to minimize performance degradation. For example, in scenarios with high compression ratios, the performance of the model tends to drop drastically, resulting in its limited usability in real-world applications. Therefore, *a more balanced method to model compression is necessary*, one that considers both compression ratio and accuracy, which enables the widespread deployment of efficient deep learning models in various applications. Achieving this balance requires careful tuning and often involves trade-offs that must be tailored to the specific application or hardware requirements [40, 41].

• **Lack of Generalizability:** While model compression has seen impressive advancements, most of these methods are tailored to specific types of models, architectures, or tasks, limiting their broader applicability across different domains [42]. For example, a certain pruning strategies or quantization techniques might work exceptionally well on CNNs used for image classification but may not be as effective on transformer-based models used in NLP. This lack of generalization is problematic because it hinders the ability to create universal compression techniques that can be applied to a wide range of deep learning models, regardless of their architecture or the task they are intended for. To address this issue, there is a growing need for *generalizable compression methods* that can work across a wide range of models and tasks. Ideally, these methods would be robust enough to handle the complexities of different deep learning architectures, as well as flexible enough to adapt to new tasks and diverse real-world data [43, 44].

• **Lack of Hardware-aware Compression:** A major challenge in model compression is the lack of optimization for specific hardware environments. While many compression techniques are designed to reduce model size, computational cost, and memory footprint, they often fail to take into account the specific characteristics and capabilities of the target hardware. This lack of hardware awareness can significantly limit the real-world effectiveness of compressed models, as the model may not leverage the full potential of the underlying hardware, resulting in suboptimal performance [45, 46]. To address this issue, *hardware-aware compression* techniques are required, as these methods tailor the compression process to the target hardware's specific strengths and weaknesses. Hardware-aware optimizations should take into account hardware factors, including the number of cores, memory hierarchy, and the parallelism of the hardware.

• **Handling Dynamic Input:** One of the lesser-discussed yet critical challenges in model compression is how well these techniques handle dynamic input variability. In numerous real-world scenarios, the input data may come from a variety of sources and can exhibit significant variability in its characteristics. But the compression techniques, such as pruning or quantization, often assume a fixed, structured input type, where the characteristics of the data remain relatively stable during inference. When the data changes dynamically (such as different image resolutions, sequence lengths, or feature distributions), these fixed assumptions may break down. As a result, the compressed model might struggle to generalize across these different input types, leading to decreased performance or accuracy. Therefore, it is necessary to develop *adaptive compression strategies* such as dynamic pruning. These methods are designed to modify the model based on the input's characteristics, allowing it to handle dynamic data effectively [47, 48].

2.4. *Evolution of Model Compression Methods*

The field of model compression has evolved significantly over the years, driven by the need to optimize large deep learning models for practical deployment. From early techniques such as pruning and quantization to more recent innovations like low-rank factorization and parameter sharing, the landscape has continuously adapted to meet the growing demands of model efficiency and performance. This section provides an overview of the key stages in the development of model compression methods, highlighting the major milestones that have shaped the field.

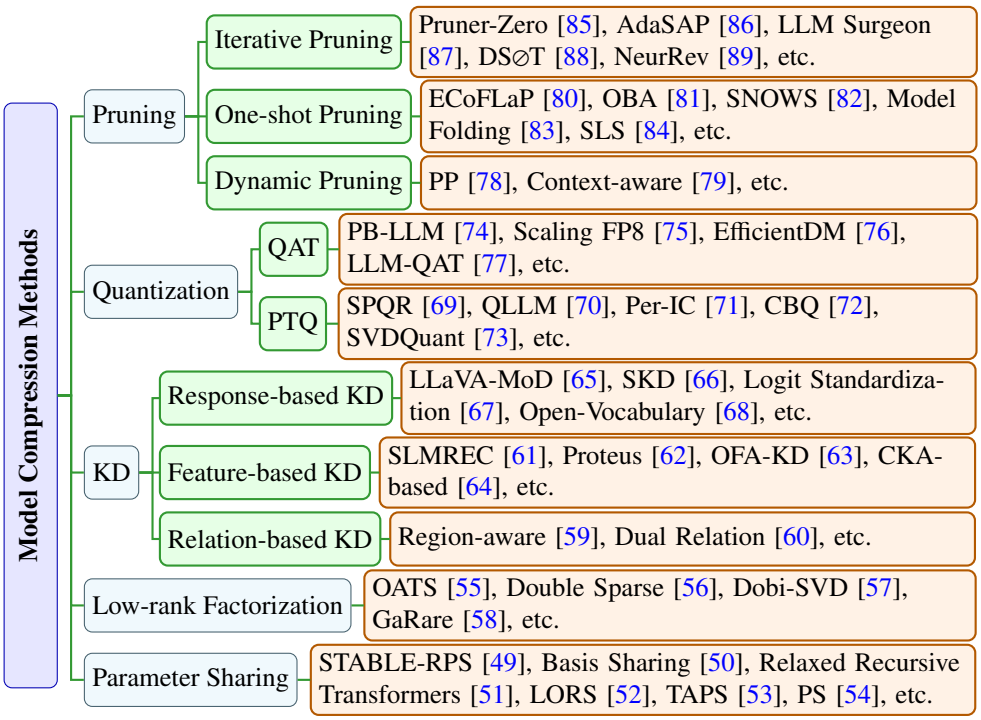

Figure 3: Taxonomy of Model Compression methods.

• **Early Foundations: Pruning and Quantization**. The initial drive for model compression stemmed from the practical limitations of deploying early deep learning models, which were often too large for real-world applications. *Pruning* emerged as a fundamental technique, focusing on removing less important connections and neurons [90]. This process identifies and eliminates redundant parameters, streamlining the network architecture and improving inference speed. *Quantization* was introduced to decrease the precision of model parameters, (e.g., from floating-point to lower-bit integers). This significantly reduces memory requirements and can accelerate computation, as integer operations are faster than floating-point operations. These early techniques established a crucial foundation for subsequent model compression research, tackling the challenges of deploying deep learning models on resource-constrained devices.

• **The Rise of Knowledge Distillation**. As deep learning models evolved, achieving greater accuracy and depth with architectures such as AlexNet [4], VGGNet [20], ResNet [11] and DenseNet [21], the discrepancy between model size and deployment capabilities grew even wider. *Knowledge distillation* became a vital technique to address this, enabling the transfer of knowledge from a large, high-capacity teacher model to a smaller, more efficient student model

[90]. The student model learns to mimic the teacher's behavior, often surpassing its own capacity by learning from the teacher's soft outputs or intermediate representations. This method facilitates the deployment of accurate models in resource-limited environments, preserving much of the teacher's performance while significantly reducing computational demands.

• **Compression for Large-Scale Models**. The advent of large-scale models, such as BERT and the GPT series, marked a new era in deep learning and introduced unprecedented challenges for model compression. These models, with their massive number of parameters, require innovative compression strategies to make them practical for deployment. *Low-rank factorization* techniques gained prominence, decomposing weight matrices into lower-rank approximations to reduce parameter redundancy and computational complexity. This method exploits the inherent redundancy in weight matrices, representing them with fewer parameters. Additionally, *parameter sharing* strategies were explored to minimize the number of parameters by sharing them across different parts of the model, further improving efficiency.

3. Compression Techniques

In recent years, researchers have proposed a variety of model compression methods, including pruning, quantization, low-rank factorization, parameter sharing, and knowledge distillation, as illustrated in Fig. 3, and Fig. 4 illustrates the fundamentals of these methods. Each of these methods has its own advantages and disadvantages, and is applicable to different scenarios. This part will systematically introduce several typical model compression methods and comparatively analyze their performance, efficiency, and applicable scenarios.

3.1. Pruning

Pruning is an important technique in model compression and its core idea is to remove unimportant weights or neurons in the neural network, thus reducing the number of parameters and computational complexity of the model. According to the strategy and implementation of pruning, pruning methods can be categorized into *static pruning* and *dynamic pruning* as illustrated in Fig. 5. Static pruning can be further categorized into *iterative pruning* and *one-shot pruning*. We discuss each method in detail and lists the results of some studies in Tab. 1.

3.1.1. Iterative Pruning

Iterative pruning gradually prunes through multiple iterations. In each iteration, a portion of the weights are first cut off according to certain criteria, and then the remaining weights are fine-tuned. Specifically, iterative pruning can be divided into the following steps:

1. Start with a pre-trained large model as the initial baseline.
2. Prune a portion of the weights according to their importance.
3. Fine-tune the pruned model to recover performance degradation.
4. Repeat the above steps until the desired sparsity is achieved.

A significant advantage of iterative pruning is its low impact on model performance. Through progressive pruning and fine-tuning, the method is able to reduce a large number of parameters while maintaining high performance, as the model is not drastically changed at each iteration. In addition, iterative pruning allows flexibility in adjusting the pruning ratio and fine-tuning strategy for each iteration to accommodate different model and task requirements.

Some studies focus on *minimizing performance loss* when removing weights [86, 91]. Their goal is to smoothly decay weights to zero prior to deletion, so that there is less impact on the

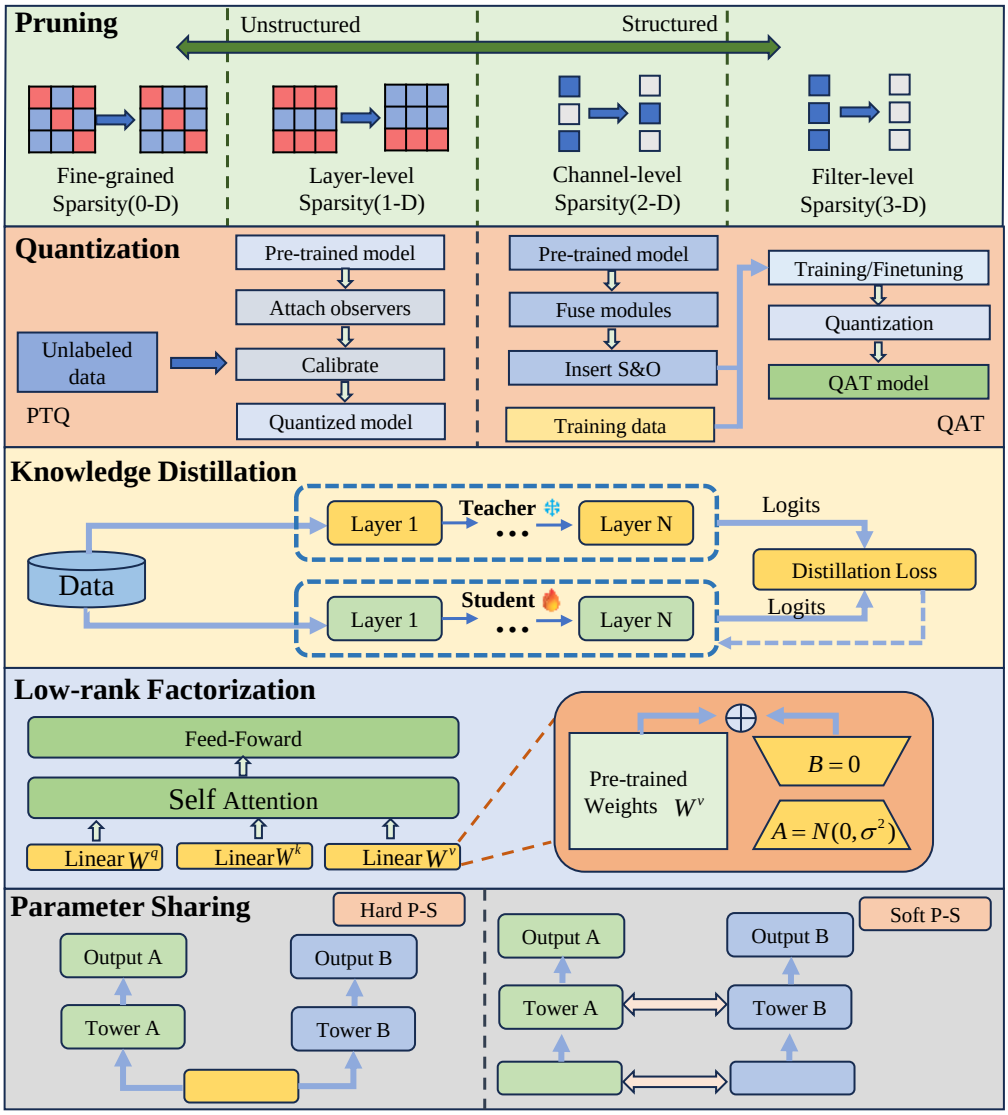

Figure 4: Overview of representative model compression techniques. The diagram illustrates five commonly used methods. Pruning removes redundant weights or neurons; quantization reduces numerical precision of weights and activations; knowledge distillation transfers knowledge from a large teacher model to a smaller student model; low-rank decomposition factorizes large weight matrices into smaller ones; and parameter sharing enables reuse of weights across different parts of the model. These techniques aim to reduce model size and computational cost while preserving accuracy.

model performance during pruning. For example, Anna et al. [86] propose a way to introduce adaptive weight perturbations during training. By adjusting the size of the perturbations according to the importance scores of neurons, the unimportant neurons will go to a flatter loss region, resulting in a smaller impact on the model performance during pruning.

Other studies focus on the *importance of weights*. For example, *Pruner-Zero* [85] formalizes the discovery process of pruning metrics as a symbolic regression problem and employs genetic

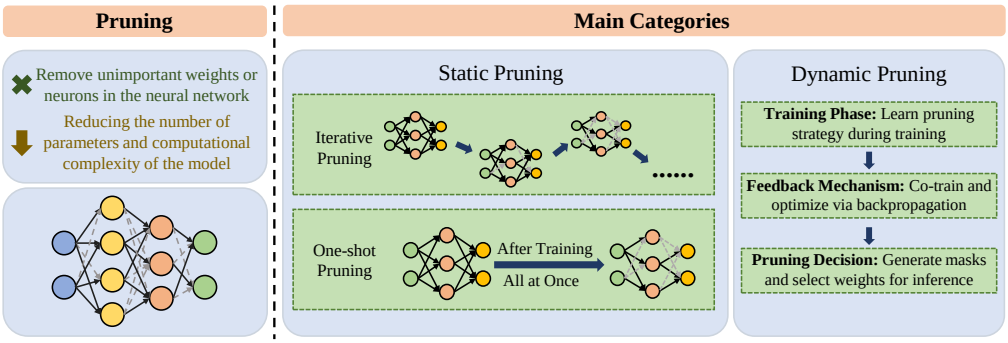

Figure 5: Illustration of pruning. Redundant weights or neurons are removed from the original model to reduce its size and computation.

programming to search for the optimal symbolic pruning metrics. Tycho et al. [87] propose an extended Kronecker decomposition curvature approximation method called *The LLM Surgeon*, which takes into account the correlation between weights, thus improving the performance of the compressed model.

Although iterative pruning performs well in reducing the number of parameters and maintaining performance, it has some limitations. In traditional iterative pruning, once each iteration is completed, the structure of the network is fixed and the pruned weights cannot be recovered [92, 93]. In addition, the model needs to be fine-tuned at each iteration, which increases the training cost [88, 83]. To overcome these limitations, researchers have proposed various improvement methods, including mask-based pruning, weight averaging, and dynamic sparse training.

• **Mask-based pruning** uses a mask to mark the weights that need to be kept or cut. The mask is a binary matrix of the same dimension as the weight matrix, with a value of 1 indicating that the weight is retained and a value of 0 indicating that the weight is clipped. The idea of this method is to transform the pruning operation of the weights into the optimization of the mask matrix. The advantage is that the weights can be manipulated without modifying the structure. In addition, mask pruning can dynamically adjust the sparsity of the network by updating the mask, which allows the network to be flexible according to the task requirements [94, 95, 96, 97, 92, 93].

• **Weight averaging** optimizes a model by calculating the average of multiple model weights. This method reduces overfitting and improves the generalizability of the model. The averaging process can naturally remove some unimportant weights and optimize the remaining weights, thus eliminating the cost of fine-tuning. Despite its advantages, weight averaging can introduce challenges in maintaining the desired sparsity levels, as the average process may cause the zero weights to become non-zero, necessitating innovative algorithmic solutions to effectively manage the trade-off between model efficiency and performance [98, 99].

• **Dynamic sparse training** aims to find the best sparse model by dynamically adjusting the connections between neurons. Dynamic sparse training begins with a sparse model (which can be achieved in any way) and then iteratively adds and subtracts weights to keep the sparsity constant during training [100, 101]. This means that previously pruned weights can be recovered in subsequent training, thus better adapting to the learning needs of the network [102, 88, 89, 103]. However, compared with traditional training, dynamic sparse training requires additional algorithms to manage the addition and deletion of weights, which is more complicated to implement.

Future research for iterative pruning could include hardware-aware optimization and efficient

pruning of large-scale models. For example, explore how to take into account the parallelism of the hardware in the pruning process to generate sparse models that are more compatible with the hardware requirements. In addition, we can also explore how to reduce the computational resources required for pruning, meeting the needs of emerging large-scale models such as LLMs.

Table 1: Pruning Results on LLMs. This table presents pruning results for various pruning methods on different LLaMA models. The compression ratios and corresponding perplexities on the Wikitext-2 dataset [104] are shown. Perplexity is used as the evaluation metric, measuring how well the model predicts the next token. Lower values indicate better performance.

Model	Method	Compression Ratio	Perplexity
LLaMA-7B	Pruner-Zero [85]	0.5	6.95
	DS⊘T [88]	0.1	5.68
		0.2	5.73
		0.4	6.28
		0.5	7.12
		0.6	10.22
	ECoFLaP [80]	0.6	9.83
	OWL [105]	0.1	5.70
		0.2	5.80
		0.4	6.39
		0.6	9.35
LLaMA-2-7B	DSF [56]	0.5	6.12
	PP [78]	0.2	8.10
		0.4	16.80
	SliceGPT [106]	0.2	6.86
		0.3	8.64
	DS⊘T	0.1	5.48
		0.2	5.49
		0.3	5.65
		0.4	5.85
LLaMA-2-13B	DSF	0.5	4.87
	PP	0.2	6.70
		0.4	11.30
	SliceGPT	0.2	6.04
		0.3	7.44
	LLM Surgeon [87]	0.1	4.70
		0.2	5.29
		0.3	6.21
		0.4	7.25
LLaMA-3-8B	OATS [55]	0.3	9.59
		0.4	9.24
		0.5	10.87
	PP	0.2	9.30
		0.4	14.90

3.1.2. One-shot Pruning

The core idea of one-shot pruning is to remove a large number of unimportant weights at one time after model training is complete, instead of iterative training. The pruning criteria are usually based on the importance score of the weights, such as the size of the weights, the gradient, or the impact on the model output. The main advantage of this method is its efficiency. Since only one training process needs to be performed, the training time and computational cost are significantly reduced. In addition, one-shot pruning methods are usually simple, easy to implement, and suitable for resource-limited environments.

However, one-shot pruning also has some obvious limitations. As a large number of weights are dropped at once, the performance of the model may be significantly degraded, especially in the case of high sparsity. Moreover, one-shot pruning usually requires fine-tuning to recover the performance, and the fine-tuning process itself requires some computational resources. To avoid intolerable performance degradation after pruning, accurately determining the importance of parameters and efficiently allocating the pruning ratio of each module becomes a key issue. For different granularity of pruning, different criteria are usually used to judge the importance.

Unstructured pruning focuses on certain weights. This method requires computing an importance score for each weight. Common criteria are magnitude and gradient. Magnitude-based pruning reduces the number of parameters in a model by removing weights with smaller absolute values. The underlying assumption is that smaller weights contribute less to the model output and can therefore be safely removed [107]. Gradient-based pruning determines the importance of weights by analyzing their gradient. Weights with larger gradients typically have a greater impact on the loss function and are therefore considered more important [80, 81, 82].

Structured pruning aims to prune certain neurons. This method can either calculate the importance score for each neuron [83, 108, 109], or calculate the sum of the importance scores of weights connected to each neuron [81]. For the neuron importance score, common criteria are the output loss and eigenvalue size. Pruning based on neuron output determines the importance of a neuron by analyzing its response error to an input value [110, 96]. Pruning based on the size of eigenvalues, on the other hand, determines the importance of a neuron by analyzing the eigenvalues of its weight matrix. Neurons with larger eigenvalues typically contribute more to the expressiveness of the model and are therefore considered more important [106, 111]. A significant advantage of structured pruning is that it enables speedups at the hardware level. Since it removes the whole set of weights, the pruned model is more efficient in both storage and computation. In contrast, unstructured pruning, although it can further reduce the number of parameters, has limited actual performance improvement due to the fact that the weight distribution after pruning is sparse and hardware acceleration is more difficult [112, 113, 114].

Layer pruning directly cuts out certain layers, such as the attention layer or the MLP layer. This method requires calculating the importance of a certain layer. Common criteria are the output characteristics of each layer or the output similarity between layers. The output characteristics of each layer can be obtained from the output of this layer [84], or obtained by comparing the loss of model performance after removing a certain layer [115]. A layer is considered unimportant if it has little effect on the output. In terms of output similarity, experiments demonstrate that layers deemed less critical often appear consecutively. This pattern gives rise to the idea of removing multiple layers at one time. If two layers have similar outputs, the layers between them can often be considered redundant [116, 117].

Future research directions for one-shot pruning could include more accurate importance assessment and pruning without fine-tuning. For example, combine explanatory techniques with

deep learning models to explore how to identify redundant weights more accurately before pruning. In addition, pruning without fine-tuning can be explored to pursue higher pruning efficiency.

3.1.3. Dynamic Pruning

The dynamic pruning decides which weights or neurons are activated during model inference, rather than pruning the model to a fixed structure. This method allows the inference process to be dynamically adapted according to the characteristics of the input data, thereby improving the inference efficiency while maintaining the model performance. Compared with static pruning, dynamic pruning can significantly improve the representation ability of the network, thus achieving better performance. Additionally, since all weights are retained and part of them are temporarily skipped, it avoids the permanent structural change [118, 119]. The core of dynamic pruning lies in input adaptation and real-time adjustment [79]. The goal is to select a mask $M(x)$ for the input such that the pruned weight matrix $W' = W \odot M(x)$ can provide best performance. The common process is divided into three phases.

1. Training Phase: The model learns the pruning strategy. For example, a lightweight selection gating network is introduced to generate a mask matrix under current input.
2. Feedback Mechanism: The pruning strategy is co-trained with the model parameters to optimize the decision-making ability of the gating through back-propagation.
3. Pruning Decision: During inference, the gating module dynamically generates sparse masks based on the input features and retains only the high response weights for computation.

However, not all methods use the gating network. Recently, Qi et al. [78] propose a dynamic structured pruning framework called *Probe Pruning (PP)*. Instead of generating masks by gating during inference, PP optimizes the model performance by selecting key data based on the residual importance in each layer of the model, running partial layers to evaluate their impact on the output, and dynamically calculating the importance scores to guide the pruning.

Future research directions for dynamic pruning could include efficiency improvements. For example, it can be investigated how to further optimize the real-time performance of dynamic pruning, especially in high-throughput scenarios, by reducing the latency of pruning decisions. In addition, more different dynamic pruning schemes can be explored.

3.2. Quantization

Quantization is a key technique in model compression, which reduces the storage requirements and computational complexity by compressing the weights and activation values into low-bit precision. Quantization methods can be classified into two categories based on the timing of the quantization: *Quantization-Aware Training (QAT)* and *Post-Training Quantization (PTQ)*. We summarize the performance of various quantization methods and list the results in Tab. 2

3.2.1. Quantization-Aware Training

QAT is an method that introduces quantization during the training phase of a model, with the aim of enabling the model to adapt to the changes brought about by quantization. Specifically, during training, weights and activation values are quantized to low-bit representations (e.g., 8-bit integers) during forward propagation and restored to floating-point numbers for gradient computation during backpropagation. This mechanism allows the model to gradually adapt to quantization errors and learn to maintain performance under quantization constraints.

A significant advantage of QAT is the ability to significantly reduce accuracy loss. Since the model has learned how to work under quantization constraints during the training process, there

Table 2: Quantization Results on LLMs. This table summarizes the performance of various quantization methods on different LLaMA models. For each method, the precision configuration (Weight-Activation) and the corresponding perplexity on the Wikitext-2 dataset are presented.

Model	Method	Precision(W-A)	Perplexity
LLaMA-2-7B	Scaling FP8 [75]	8-8	5.55
	SpinQuant [120]	4-8	5.70
		4-4	5.90
	LeanQuant [121]	4-16	5.73
		2-16	25.69
	CBQ [72]	4-16	5.52
		4-8	5.72
	OSTQuant [122]	4-16	5.64
		4-4	5.60
	ARB-LLM [123]	1.08-16	16.44
	SeedLM [124]	4-16	5.70
	STBLLM [125]	0.80-16	13.06
		0.70-16	18.74
LLaMA-2-13B	SpinQuant	4-8	5.10
		4-4	5.20
	LeanQuant	4-16	5.08
		2-16	24.43
	OSTQuant [122]	4-16	5.64
		4-4	5.14
	RRS [126]	16-4	5.22
		4-4	5.36
	ARB-LLM	1.08-16	11.85
	SeedLM	4-16	5.10
	STBLLM	0.80-16	11.67
		0.70-16	13.26
LLaMA-3-8B	SpinQuant	4-8	6.50
		4-4	7.30
	ARB-LLM	1.06-16	27.42
	OSTQuant [122]	4-16	6.53
		4-4	7.24
	RRS	16-4	7.55
		4-4	8.11
	SeedLM	4-16	7.00

is less accuracy loss in the quantized model. In addition, QAT allows flexibility in adjusting the quantization strategy during training to optimize the model performance. For example, the accuracy and computational complexity of the model can be balanced by adjusting the quantization parameters (e.g., the number of quantization bits and the quantization range) [74, 75].

Although QAT performs well in reducing the loss of accuracy, it has some limitations. First, the complexity of the training process increases significantly because quantization and inverse quantization operations need to be performed between each forward propagation and back propagation, substantially extending the training time. In addition, traditional QAT methods require a large amount of training data and computational resources, which may be difficult to achieve in some cases (e.g., data privacy or insufficient computational resources) [76, 77, 75].

Future research directions include improving outlier handling methods, designing data-free quantization methods, and exploring higher-bit quantization methods. For example, researchers could explore how to handle outliers during quantization to reduce quantization errors. In addition, designing data-free quantization methods to address data privacy issues is also an important research direction. Finally, it is possible to investigate how to quantize the model to higher bits, such as binarization, without causing a significant degradation of the model performance.

3.2.2. Post-Training Quantization

PTQ is a method that quantizes the weights and activation values of a pre-trained model to lower-bit representations directly after the model training is completed. This method saves time and computational resources by quantizing the model directly without modifying the training process. It centers on determining the quantization range, calculating quantization parameters (e.g., scales and zeros), and applying quantization operations. A significant advantage of PTQ is that it is simple to implement and consumes less computational resources [69, 127, 128, 129, 130]. However, PTQ may lead to a large loss of accuracy as the model is not adapted to the quantization operation during the training phase. In addition, in large-scale LLMs, activation outliers lead to an increase in the quantization range, causing most activations to be mapped into the same quantization bucket, resulting in poorer quantization [75, 71, 70, 73].

To overcome these limitations, researchers have proposed a series of improvements. For example, Tim et al. [69] reduce the impact of anomaly weights on the overall quantization error by identifying the anomaly weights in the model and storing them with higher precision, while other weights are compressed to 3-4 bits. Jing et al. [70] reduce the quantization error through an adaptive channel reorganization technique and an efficient gradient basis error correction strategy, which reduces the magnitude of outliers and makes the activation values easier to quantize. Jung et al. [71] group the weights in the direction of the input channels, so that the effect of outliers is limited to a single group, effectively isolating the outliers. Xin et al. [72] handle weight outliers by quantizing them across blocks and optimizing multiple blocks at once using a sliding window method that ensures connectivity between blocks. There are also methods that reduce the impact of outliers by transforming the activation values so that the outliers are dispersed among other values [120, 131].

Moreover, existing quantization methods mainly rely on reducing memory movement rather than actually reducing computation because the inverse quantization process still consumes a lot of resources. To avoid the inverse quantization process, Gunho et al. [132] use lookup-table (LUT) to calculate the multiplication of the quantized matrix, thus reducing the computational cost. In large language models, the distribution of weights may show complex patterns. Uniformly distributed quantization grids can lead to excessive quantization errors for some weights when dealing with such non-uniform distributions, thus affecting the overall quality of the model. To solve this, Tianyi et al. [121] propose a new quantization grid learning method *LEANQUANT* based on the iterative loss error quantization framework, which learns loss-error-aware grids, instead of using non-adaptive min-max affine grids.

Future research directions include improved outlier handling methods, hardware-aware quantization and multi-precision quantization. The handling of outliers is an important research direction in both PTQ and QAT. In addition, the quantized model needs to run on specific hardware, so hardware-aware quantization is an important research direction. Finally, multi-precision quantization allows different layers or parts of the model to use different quantization accuracies in order to strike a balance between accuracy and efficiency.

3.3. Knowledge Distillation

Knowledge distillation(KD) is a method of transferring knowledge learned from a "large model" to a "small model". As shown in Fig. 6, the method aims to train a student model to mimic the behavior of the teacher model, so that the student model can have a smaller parameter size while maintaining higher performance. The core idea of KD is to use the outputs or intermediate features of the teacher's model to guide the learning of the student model to reduce the complexity of the model while maintaining high performance [133]. According to the type of KD, the KD can be categorized into *response-based KD*, *feature-based KD*, and *relation-based KD*.

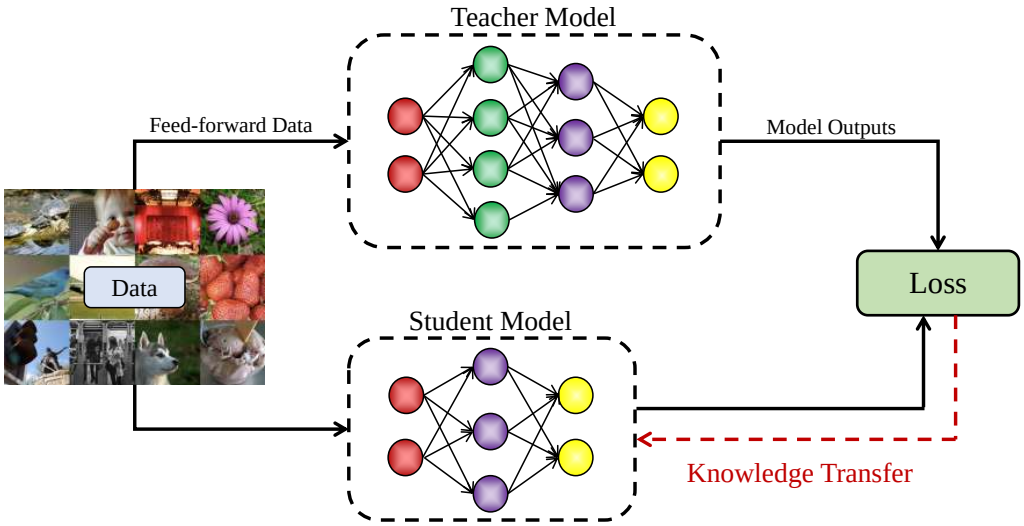

Figure 6: Illustration of KD. A smaller student model is trained to mimic the outputs or intermediate representations of a larger, pre-trained teacher model, thus achieving comparable accuracy with fewer parameters and computations.

3.3.1. Response-based KD

Response-based KD is one of the earliest proposed and widely used methods. Its core idea is to let the student model learn the output of the teacher model, which is usually the probability distribution of the categories. The teacher model generates a soft label through a softmax layer that contains information about the relative relationships between categories. The student model is trained by imitating these soft labels, thus learning the generalizability of the teacher model.

Response-based KD is an important branch in the field of KD and has made significant progress in several directions in recent years. The advantage of this method is that it is simple and easy to implement and does not require complex feature extraction and comparison mechanisms. However, since it ignores the feature representation inside the model, the student model may not be able to learn the deeper features of the teacher model. Jin et al. propose a multi-level Logit distillation method [134]. Through this framework, the prediction alignment is conducted not only at the instance level, but also at the batch and class level, through which the student model learns the instance prediction, input correlation, and category correlation simultaneously. *LLaVA-MoD* [65] balances the computational efficiency and performance by integrating the sparse Mixture of Experts (MoE) architecture. The framework significantly improves the performance of the student model by gradually transferring the knowledge from the teacher model to the student model through two phases: imitation distillation and preference distillation.

In traditional KD methods, the teacher and the student models share a global temperature parameter, which limits the performance of the student model. Sun et al. [67] propose a logit normalization method, which maps the logit to a bounded range through Z-score preprocessing, allowing the student model to have arbitrary range and variance while maintaining the logit relationship of the teacher model.

In some cases, the raw data may not be available due to copyright and privacy issues. To address this issue, researchers have proposed data-free KD methods [68]. For example, the *SKD* method [66] generates high-quality training data by interleaving sampling between teacher and student models and aligning them with the students' reasoning time distributions.

Response-based KD methods have made significant progress in both theory and application. These methods not only improve the performance of student models, but also address problems in traditional methods. In the future, response-based KD will be developed in the direction of being more efficient and more widely applied. Possible research directions include further optimization of distillation algorithms and data-free distillation techniques. Researchers can improve existing distillation methods or combine techniques such as multi-teacher distillation and adversarial distillation to enhance the performance and generalization of student models. In addition, data-free distillation will become a research hotspot to meet the needs of data privacy.

3.3.2. Feature-based KD

Feature-based KD is able to better preserve the internal structural information of the model by allowing the student model to learn the feature representation of the intermediate layer of the teacher model. This method is usually implemented through feature matching or feature alignment, where the middle layer features of the student model are trained to match the corresponding features of the teacher model [135, 136]. The advantage of this method is that it can help the student model learn richer feature representations, thus improving its generalization ability. However, its limitations are that it requires the design of complex feature extraction and comparison mechanisms and is less compatible with different model architectures.

To overcome these limitations, researchers have proposed improved methods for feature distillation. For example, Wujiang et al. [61] propose a simple and effective KD method *SLMREC* that guides the student model to learn the intermediate level feature representation of the teacher model through multi-supervised signals, thus achieving comparable performance to large language models while using only 13% of the parameters. Sayantan et al. [64] propose a hidden state matching technique based on centered kernel alignment that allows hidden states of different dimensions to be matched, thus achieving higher compression ratios. In addition, Yitian et al. [62] propose a method called Proteus. This method significantly improves knowledge transfer by removing the design that leads to dataset bias in traditional KD setups and introducing multi-level training objectives (e.g., token level, patch level, and feature level). Hao et al. [63] propose a simple but effective cross-architecture KD framework OFA-KD by projecting intermediate features to an aligned latent space and discarding architecture-specific information, thus realizing KD between different architectural models.

As an improved method to KD, multi-teacher distillation is an advanced KD method. It significantly improves the generalizability and robustness of the student models by integrating knowledge from multiple teacher models. Compared with single-teacher distillation, multi-teacher distillation can provide richer and more comprehensive knowledge to the student model from multiple perspectives, thus enabling it to show greater adaptability in the face of complex tasks and diverse data. However, this method requires training multiple teacher networks, which is computationally expensive and resource intensive. To solve this, Hossain et al. [137] propose

the TeKAP method, which generates multiple synthetic teacher knowledge by perturbing the knowledge of a single pre-trained teacher model, thus providing diverse learning perspectives for the student model without increasing computational resources.

These improved methods not only improve the performance and generalization ability of the student models, but also reduce the demand for computational resources and enhance the flexibility and adaptability of KD methods. Possible future research directions focus on feature selection and alignment optimization. Researchers can explore smarter feature selection mechanisms to dynamically identify the most valuable features for the task in the teacher's model and avoid the transfer of redundant information. By improving the feature alignment algorithm, the representation of the student model in the feature space can be enhanced, while reducing the computational complexity and realizing more efficient feature distillation.

3.3.3. Relation-based KD

Relation-based KD focuses on the relation among samples rather than on an individual sample. The assumption is that the essence of knowledge lies in the relation among samples, not just in the representation of independent features. Student models capture the global structure of the data by learning these relation and thus perform better on complex data distributions [138].

However, it increases computational complexity and requires a suitable similarity metric. To overcome these limitations, researchers have proposed improved methods. A new dual relational distillation method has been proposed in the target detection task [60]. The method solves the problems of imbalance between foreground and background features and under-representation of small target features by pixel-level relational distillation and instance-level relational distillation. The method also utilizes graph convolution to capture global pixel relations so that the student model can learn the relation between foreground and background.

In the semantic segmentation task, Region-aware Mutual Relational KD [59] decouples the foreground and background regions by introducing a region-aware module and performs feature distillation and response distillation on intermediate features and the output graph. The method utilizes region tokens to capture the interrelationships between the teacher model and the student model for more effective knowledge transfer. Other researchers have proposed Cross-Image Relational KD [139], where the global structural information in the teacher network is transferred to the student model through global pixel relational distillation.

In the vision task, token-level relationship graph-based KD [140] captures finer-grained semantic information by constructing a token relationship graph. The method divides the feature graph into fixed-size patch tokens and constructs a relational graph to achieve KD from the teacher model to the student model through local retention loss and global topology loss.

Relation-based KD is rapidly evolving and it shows great potential in several domains. However, the current research of relation-based KD are relatively limited. Future research will further expand its application scope and enhance its performance to provide a more solid foundation for the wider application of KD technology. Future research directions include designing more efficient similarity metrics. In addition, exploring how to better utilize the global structural information of teacher models in the distillation process is also an important research direction.

3.4. Low-rank Factorization

Low-rank Factorization is a method of reducing the number of model parameters by decomposing the weight matrix into the product of two or more low-rank matrices. This method utilizes the low-rank property of matrices, as the redundant information in the weight matrix can be effectively represented by a low-rank approximation. Low-rank Factorization not only significantly

reduces the storage requirements of the model, but also reduces the computational complexity, thus improving the inference efficiency of the model [58].

Given a weight matrix $W \in R^{m\times n}$, the goal of low-rank factorization is to find two low-rank matrices $U \in R^{m\times k}$ and $V \in R^{k\times n}$ such that $W \approx U \times V$. Here k is the rank, which is much smaller than m and n, thus significantly reducing the number of parameters.

For example, Wang et al. [57] propose a microscopic truncation mechanism that combines gradient-robust backpropagation to enable the model to adaptively find the optimal truncation location. In addition, the method utilizes the Eckart-Young-Mirsky theorem to derive a theoretically optimal weight updating formula, and sequentially extracts and updates the features of the weight matrix via incremental principal component analysis. Lin et al. [141] propose a holistic CNN compression framework by low rank decomposition compression of both convolution and fully connected layers, and introduces a knowledge migration training scheme that recovers cumulative accuracy loss and overcomes the gradient vanishing problem by aligning the outputs and intermediate responses of the teacher network to the student network. Stephen et al. [55] propose a new compression method that decomposes the model weights into the sum of the sparse and low-rank matrices. The second-order moment information of the input embedding is used to better capture important features in the weight matrix. Vladimir et al. propose Double Sparse Factorization [56], which further optimizes the sparsity of the neural network by decomposing the weight matrix into the product of two sparse matrices.

For large-scale pre-trained models, traditional parameter fine-tuning consumes a large amount of computational resources. Therefore, Low-Rank Adaptation (LoRA) [142] technique emerged. LoRA uses a matrix decomposition technique that introduces low-rank matrices to approximate the update of the weight matrices, thus realizing the rapid model tuning and adaptation without a significant increase in the computational and storage costs. Recently, many studies have proposed improvements to LoRA. Han et al. [143] propose a low-rank plus quantization matrix decomposition method, which decomposes the pre-training matrix into high-precision low-rank components and memory-efficient quantization components by an iterative algorithm. During the fine-tuning process, only the low-rank components are updated while the quantized components are kept fixed. NOLA [144], on the other hand, optimizes linear mixing coefficients by re-parameterizing the low-rank matrices in LoRA as linear combinations of random bases, thus decoupling the number of parameters from the rank and network architecture. Lialin et al. [145] propose a method *ReLoRA* for high-rank training via low-rank updates. ReLoRA utilizes the gradual accumulation of low-rank updates during the training process, realizing the training of a high-rank network. *MOS* [146] further unleashes the parametric efficiency of LoRA by means of a hybrid fragmentation method. This method works by decomposing the model into multiple fragments and applying LoRA to each of them. *QA-LoRA* [147] and *LOFTQ* [148] focus on further compressing the parameters by utilizing quantization. QA-LoRA further reduces the model's memory by taking quantization errors into account during the fine-tuning process. This method not only reduces time and memory usage during the fine-tuning phase, but also naturally integrates the model and auxiliary weights into a quantized model after fine-tuning, without the need for an additional quantization step. LoftQ minimizes the difference between the quantized model and the original high-precision model by alternately optimizing the quantization and the low-rank approximation, thus providing a suitable low-rank initialization for LoRA fine-tuning. The emergence of these methods has further advanced the development of LoRA techniques, making them more efficient and flexible in model compression and fine-tuning.

Low-rank factorization techniques have great potential in the field of model compression and can effectively reduce the number of model parameters while maintaining high performance. Fu-

ture research directions include the development of more efficient algorithms factorize the weight matrix. In addition, combining low-rank factorization with other model compression techniques (e.g., pruning and quantization) to achieve higher compression ratio is also an area worth exploring. Finally, the theoretical basis of low-rank factorization can be further investigated to guide the design of more efficient compression algorithms.

3.5. Parameter Sharing

Parameter sharing is a technique to reduce storage requirements and computational complexity by reducing the number of redundant parameters in a model. The core idea is to map network parameters onto a small amount of data using methods such as structured matrices or clustering.

In Convolutional Neural Networks (CNNs), one of the most widely recognized applications of parameter sharing is in the use of shared convolution kernels across the entire input image. This reduces the total number of parameters compared to a fully connected layer, where each input element is connected to each weight. As a result, convolutional layers achieve substantial parameter reduction, enabling efficient computation without a significant loss of performance [49]. For example, Jingcun et al. [50] introduce a method to decompose weight matrices across different layers into linear combinations of shared basis vectors and unique coefficients, facilitating parameter sharing across layers. This allows the model to retain the flexibility of learning diverse feature representations while reducing redundancy in parameter usage.

Additionally, Sangmin et al. [51] propose a cyclic reuse approach, where a set of unique layers are reused across different parts of the model. This cyclic approach further reduces parameter usage and ensures that the learned features remain applicable across multiple stages of the network. Similarly, Jialin et al. [52] introduce a hybrid approach where the model parameters are divided into shared and private parameters. The shared parameters are learned collectively across all modules, while the private parameters are unique to each module. This decomposition helps balance the trade-off between parameter sharing and the ability to adapt to specific tasks.

Task Adaptive Parameter Sharing (TAPS) [53] is an emerging parameter sharing method that efficiently adapts pre-trained models to new tasks by dynamically selecting a small subset of task-specific network layers for adjustment. This method excels in multitask learning scenarios, achieving performance close to full fine-tuning while introducing only a minimal number of task-specific parameters. TAPS determines which layers to adjust through a joint optimization problem and minimizes the number of task-specific parameters via sparsity constraints. Suitable for various network architectures (such as ResNet and Vision Transformers), TAPS is simple and efficient to implement, requiring only minor modifications to the training scheme. The emergence of TAPS offers new ideas for the application of parameter sharing in multitask learning, demonstrating its potential in reducing parameter redundancy and enhancing model adaptability.

In the field of Single Image Super-Resolution (SISR), significant progress has also been made in the application of parameter sharing techniques. The Partial Filter-Sharing (PS) method proposed by Park et al. [54] effectively addresses the limitations of traditional parameter sharing methods in terms of representational capability by introducing a partial filter sharing mechanism. The core of the PS method lies in dividing the filter into multiple partial filters and dynamically reconstructing the complete filter through a coefficient matrix. This method not only reduces the number of parameters but also significantly enhances the network's representational capacity by allowing each layer or task to use different filter combinations.

Currently, there are relatively few studies related to parameter sharing, suggesting more research in this method. Future research can explore more efficient algorithms to achieve better

performance. In addition, research can be conducted on how to dynamically adjust the parameter sharing based on input data and tasks. Finally, combining parameter sharing with other model compression techniques to achieve higher compression ratio is also an area worth exploring.

4. Comparison and Analysis

Pruning, quantization, knowledge distillation, low-rank factorization, and parameter sharing are the five main technical approaches introduced before. While methods like iterative pruning and QAT can significantly reduce model size and computational requirements, they tend to be more computationally intensive and may require additional training. In contrast, techniques like response-based KD and one-shot pruning, which aim to reduces the size of the model quickly, often lead to high loss of accuracy. Therefore, selecting the appropriate method often involves balancing the degree of compression, the desired performance, and the training efficiency. In the following, we will comprehensively compare the characteristics of these methods in terms of performance, efficiency, and applicability scenarios, and analyze the applicability of each method based on model architecture, task requirements, and resource constraints, as shown in Tab. 3.

Table 3: Comparison of different model compression methods across five dimensions. "Efficiency" refers to how quickly compression can be applied; "Rate" indicates the achievable reduction in model size; "Performance" reflects how well performance is preserved after compression; "Difficulty" refers to the practical complexity of implementing the method; and "Structural Change" indicates whether the method alters the original model structure.

Method	Efficiency	Rate	Performance	Difficulty	Structural Change
Iterative Pruning	Low	High	High	Hard	✓
One-shot Pruning	High	Medium	Low	Medium	✓
Dynamic Pruning	Medium	Medium	High	Hard	✗
QAT	Low	High	High	Hard	✗
PTQ	Medium	High	Medium	Medium	✗
Response-based KD	High	Medium	Medium	Easy	✗
Feature-based KD	Medium	Medium	High	Medium	✗
Relation-based KD	Medium	Medium	High	Hard	✗
Low-rank Fact	Medium	High	Medium	Medium	✓
Parameter Sharing	Medium	High	Medium	Easy	✓

Pruning reduces the number of parameters in a model by removing unimportant weights or neurons from the neural network. Traditional iterative pruning has less impact on model performance, but is more expensive to train. One-shot pruning reduces the number of parameters quickly, but can lead to significant degradation in model performance. Dynamic pruning can dynamically adjust the model structure according to the input data, but complex algorithms need to be designed in order not to introduce additional inference latency. For resource-constrained environments, one-shot pruning is an effective method because it does not require additional training. For resource-sufficient environments, iterative pruning maintains better performance.

Quantization reduces storage requirements and computational complexity by converting weights and activation to low-bit representations. QAT perceives a small loss of accuracy but high training complexity. PTQ is simple and fast to implement but has a high loss of accuracy. Quantization methods are suitable for scenarios where storage requirements and computational

complexity need to be reduced, especially on hardware platforms where low-precision operations can often significantly speed up the inference process. For scenarios that require fast inference, such as real-time image processing or speech recognition, quantization is an ideal choice because it can significantly increase inference speed while maintaining high accuracy. However, the bit-width reduction of the network parameters loses a portion of the amount of information, which can cause the inference accuracy to degrade. While it is possible to restore some of the accuracy through fine-tuning, it also brings about an increase in time cost. In addition, the quantized model may require specific hardware support to take full advantage of the low-precision operations.

Knowledge distillation allows student models to have smaller parameter sizes while maintaining high performance by transferring knowledge learned from "big models" to "small models". Response-based KD is simple and easy to implement, but is prone to overfitting. Feature-based KD can better preserve the internal structure information of the model, but is less compatible with different model architectures. KD is an effective solution for scenarios that require the deployment of high-performance models in resource-constrained environments. However, KD requires training two models (teacher model and student model). Without a pre-trained teacher model, training the student model will require a large-scale dataset and a long time.

Low-rank factorization reduces the number of parameters by decomposing the weight matrix into the product of multiple low-rank matrices. This method has a solid mathematical foundation and the factorization process is well documented. Low-rank factorization is suitable for large networks and convolutional networks, and can effectively reduce the number of parameters in the model. However, the low-rank factorization process may lose some information, resulting in an error between the reconstructed matrix and the original matrix, which affects the performance of the model and usually requires retraining to recover the performance. In addition, for large-scale models, a large computational overhead is required to factorize the matrix.

Parameter sharing reduces the number of parameters by mapping network parameters via methods such as structured matrices or clustering. Especially in the fully-connected layer, as the parameter storage occupies a large part of the whole network model, parameter sharing can play a better role in removing the redundancy. Meanwhile, due to its easy operation, parameter sharing is also suitable to be used in combination with other methods to achieve better compression and acceleration effects. However, parameter sharing may lead to optimization problems such as gradient vanishing or gradient explosion, especially in deep networks.

5. Datasets and Evaluation Metrics

The evaluation criteria for model compression not only focus on the compression rate itself, but also emphasize the performance of the compressed model in real tasks. Therefore, a reasonable selection of datasets and evaluation metrics is crucial for validating the effectiveness of compression algorithms. The evaluation benchmarks used for different task types vary, but some task-independent universal compression metrics also exist. In this section, we will start from two mainstream tasks, namely, image classification and NLP, introduce their commonly used evaluation benchmarks, and summarize the widely used general evaluation metrics in model compression at the end.

5.1. Image Classification

Image classification is one of the earliest and most widely used tasks in model compression, and many methods were initially validated in image classification scenarios. In this task, evaluation

metrics typically focus on classification accuracy (Top-1/Top-5 Accuracy), paired with model size versus computational complexity variations. Commonly used datasets include:
• **CIFAR-10** [149]:
– Number of images: 60,000 color images of 32×32 pixels divided into 10 categories.
– Data division: The dataset is divided into 50,000 training images and 10,000 test images.
– Categories: Include airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck.
• **CIFAR-100** [149]:
– Number of images: 60,000 color images of 32×32 pixels divided into 100 categories.
– Data division: Again divided into 50,000 training images and 10,000 test images.
– Categories: The categories are more granular and difficult to distinguish between categories.
• **ImageNet** [150]:
– Number of images: Over 14 million labeled images, covering over 20,000 categories.
– Data division: 12 million training images, 150,000 validation images, and 150,000 test images.
– Category Distribution: The categories are rich and diverse, which are highly complex.

In addition, there are small datasets containing images of certain specific categories that can be used to assess the generalizability of the model. Examples include MNIST [151], The Comprehensive Cars (CompCars) dataset [152], FoodX-251 [153], and so on.

5.2. Natural Language Processing

With the development of large language modeling, NLP becomes another hot direction in compression research. Different from image categorization, NLP involves multiple task types and thus has richer evaluation dimensions. In addition to task accuracy metrics (e.g., accuracy, F1 value, BLEU score, etc.), the inference cost and generalization ability of the model should also be paid attention to. Commonly used benchmark datasets and tasks include:
• **Language Modeling:** Using WikiText [104], OpenWebText [154], BooksCorpus [155], The Pile [156] and other large-scale corpora for pre-training and perplexity evaluation.
• **Text categorization and natural language inference (NLI):** e.g. GLUE [157] (including subtasks such as MNLI, QQP, QNLI, etc.), SuperGLUE [158](including BoolQ, RTE, WSC, etc.) are important benchmarks for evaluating the compression effect of language understanding.
• **Question Answering:** SQuAD v2.0 [159], TriviaQA [160], Natural Questions [161] etc. are widely used to evaluate the ability in information extraction and comprehension tasks.
• **Common Sense and Multi-Step Reasoning:** PIQA [162], SIQA [163], and OBQA [164] are used to assess the model's reasoning and real-world knowledge; MMLU [165, 166] provides categorization and quiz questions in 57 domains, and is an important benchmark for the current assessment of general cognitive ability after the compression of large models; ARC [167], GSM8K [168], HellaSwag [169], etc. are also used in complex logic and math skills.
• **Code Generation & Instruction Following:** HumanEval [170], MBPP [171], InstructEval [172], AlpacaEval [173], etc. are used to evaluate the compression model's multi-round interaction and instruction comprehension.

5.3. Generalized Metrics for Model Compression

Whether the model is applied to image classification, NLP or other tasks, the core goals of model compression is to reduce resource consumption while preserving model performance as much as possible. Therefore, despite task-related metrics, many studies have introduced task-independent generic metrics to measure the efficiency, cost, and deployability of compression effects.
• **Compression Rate Metrics**

– Number of Parameter: The total number of trainable weights and biases in the model. Parameter Compression Ratio (the ratio of the number of compressed model parameters to the number of original model parameters) is commonly used to evaluate the degree of compression.
– FLOPs: Number of floating point operations. FLOPs Ratio (the ratio of FLOPs of the compressed model to FLOPs of the original model) is also a common evaluation metric.
• **Deployment Metrics**
– Latency: The single-sample inference time on actual hardware. However, since latency varies among different hardware devices, comparing latency needs to be in the same environment.
– Throughput: The number of samples that can be processed per unit of time. As with latency, it also needs to be compared in the same environment.

6. Future Directions

As model compression techniques continue to mature, it has made significant progress in recent years and has shown great potential in practical applications. Meanwhile, along with the development of large models and multimodal systems, model compression also faces new opportunities. In this section, we will discuss the prospective research directions.

• **Automated compression.** A promising direction in model compression is the development of automated compression techniques. Automated compression aims to reduce the reliance on manual tuning and expertise by using automated methods to determine the best compression strategy for a given model. This can involve selecting the optimal compression techniques and tuning their hyperparameters to maximize performance while minimizing computational costs. Approaches such as Neural Architecture Search (NAS) and reinforcement learning are being explored to automate the compression process. As these techniques evolve, they have the potential to significantly improve the scalability and efficiency of model compression.

• **Multi-method fusion and co-optimization.** Future compression research will pay more attention to the fusion and co-optimization of different compression methods. By combining different compression techniques, the model can be co-optimized in different dimensions. For example, the combination of pruning and quantization can reduce computational complexity while ensuring model sparsity, while distillation can help maintain model performance. Research will focus on how to design compression frameworks that can be adaptive, automatically selecting and fusing different methods to cope with the requirements of specific tasks and hardware [174, 175].

• **Hardware-aware compression techniques.** With the diversification of hardware architectures, especially the wide application of dedicated gas pedals (e.g., TPUs, GPUs, edge computing devices, etc.), how to optimize the model compression in tandem with the hardware has become an important direction. Future research will focus on designing hardware-aware compression methods so that compression techniques can fully utilize the computational advantages of different hardware platforms. For example, by analyzing the parallel computing capabilities and storage limitations of hardware, compression strategies are adjusted to achieve optimal computational efficiency and storage utilization on different platforms [176].

• **Cross-domain adaptive model compression.** Different domains (e.g., image classification, NLP, etc.) have different needs for model compression, while most of the current researches focus on a single domain. Future research will explore cross-domain compression techniques that enable the same compression framework to adaptively adjust to multiple tasks, which not only helps to improve performance on different tasks, but also makes compression techniques more universal and extends their adaptability in various tasks [177].

• **End-to-end optimization and automated compression frameworks.** Future research will

further promote the development of end-to-end optimization methods, integrating all aspects of model training, compression and deployment, making the compression process more automated and efficient. With an automated model compression framework, the optimal compression strategy can be automatically selected based on the needs of a specific task. The research will explore how to design an intelligent system that automatically adjusts the strategies and parameters of model compression based on the target platform and application scenarios.

7. Conclusion

Model compression technology, as an important bridge connecting large-scale modeling capabilities and practical landing requirements, has gradually become a research hotspot. Through various methods such as pruning, quantization, distillation, etc., researchers have successfully reduced the model size and computational complexity without significant degradation of the performance. Although model compression techniques have made progress in theory and practice, optimization in specific application scenarios still faces many challenges. Future research needs to make more efforts in improving the compression rate, maintaining the model performance, and adapting to the hardware platform. Through cross-disciplinary research, the proposal of innovative algorithms, and the deep combination of compression algorithms and hardware, the model compression technology will continue to develop in the direction of being more efficient, flexible, and generalizable, and will promote the widespread implementation of deep learning technology in practical applications. With the continuous innovation and optimization of compression technology, we have reason to believe that future AI models will be more efficient and intelligent, and will be able to exert their great potential in a wider range of application scenarios.

Author Contributions

Haoran Guan: Responsible for the overall conception and writing of the paper and the organization of experimental results.

Junhao Lv: Responsible for the advantages and disadvantages of the methods and the improvement methods; participated in drawing the pictures of the article.

Xi Chen: Responsible for the innovative methods and challenges of model compression; participated in drawing the pictures of the article.

Lei Qi: As the corresponding author, guiding the research direction and the structure of the paper, reviewing the content of the paper.

Acknowledgements

The authors acknowledge all the participants and survey staffs for their participation.

References

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, 2014.

[2] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *International Conference on World Wide Web (WWW)*, p. 173–182, 2017.
[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems (NeurIPS)*, p. 1097–1105, 2012.
[5] M. R. Costa-Jussà, J. Cross, O. Çelebi, M. Elbayad, K. Heafield, K. Heffernan, E. Kalbassi, J. Lam, D. Licht, J. Maillard, *et al.*, "No language left behind: Scaling human-centered machine translation," *arXiv preprint arXiv:2207.04672*, 2022.
[6] V. Pratap, A. Tjandra, B. Shi, P. Tomasello, A. Babu, S. Kundu, A. Elkahky, Z. Ni, A. Vyas, M. Fazel-Zarandi, *et al.*, "Scaling speech technology to 1,000+ languages," *Journal of Machine Learning Research (JMLR)*, vol. 25, no. 97, pp. 1–52, 2024.
[7] L. Barrault, Y.-A. Chung, M. C. Meglioli, D. Dale, N. Dong, M. Duppenthaler, P.-A. Duquenne, B. Ellis, H. Elsahar, J. Haaheim, *et al.*, "Seamless: Multilingual expressive and streaming speech translation," *arXiv preprint arXiv:2312.05187*, 2023.
[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems (NeurIPS)*, vol. 27, 2014.
[9] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4401–4410, 2019.
[10] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems (NeurIPS)*, vol. 33, pp. 6840–6851, 2020.
[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.
[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.
[13] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.
[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.
[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems (NeurIPS)*, vol. 28, 2015.
[16] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision (ECCV)*, pp. 213–229, 2020.
[17] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.
[18] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *arXiv preprint arXiv:1508.01211*, 2015.
[19] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *International conference on machine learning (ICML)*, pp. 28492–28518, 2023.
[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations (ICLR)*, 2015.
[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.
[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *North American Chapter of the Association for Computational Linguistics (NAACL)*, vol. 1, pp. 4171–4186, 2019.
[23] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020.
[24] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.
[25] J. Liu, M. Yang, Y. Yu, H. Xu, K. Li, and X. Zhou, "Large language models in bioinformatics: applications and perspectives," *arXiv preprint arXiv:2401.04155*, 2024.
[26] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021.
[27] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning (ICML)*, vol. 139, pp. 8748–8763, 2021.
[28] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image

generation," in *International Conference on Machine Learning (ICML)*, vol. 139, pp. 8821–8831, 2021.
[29] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *Advances in neural information processing systems (NeurIPS)*, 2023.
[30] M. Xu, W. Yin, D. Cai, R. Yi, D. Xu, Q. Wang, B. Wu, Y. Zhao, C. Yang, S. Wang, *et al.*, "A survey of resource-efficient llm and multimodal foundation models," *arXiv preprint arXiv:2401.08092*, 2024.
[31] G. Menghani, "Efficient deep learning: A survey on making deep learning models smaller, faster, and better," *ACM Computing Surveys*, vol. 55, no. 12, 2023.
[32] L. Lyu, "A pathway towards responsible ai generated content," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.
[33] L. Shen, Y. Sun, Z. Yu, L. Ding, X. Tian, and D. Tao, "On efficient training of large-scale deep learning models," *ACM Computing Surveys*, vol. 57, no. 3, 2024.
[34] J. de La Torre, "Transformadores: Fundamentos teóricos y aplicaciones," *arXiv preprint arXiv:2302.09327*, 2023.
[35] Y. Li, H. Wen, W. Wang, X. Li, Y. Yuan, G. Liu, J. Liu, W. Xu, X. Wang, Y. Sun, *et al.*, "Personal llm agents: Insights and survey about the capability, efficiency and security," *arXiv preprint arXiv:2401.05459*, 2024.
[36] H. Li, H. Zhang, X. Qi, Y. Ruigang, and G. Huang, "Improved techniques for training adaptive deep networks," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1891–1900, 2019.
[37] K. T. Chitty-Venkata, S. Mittal, M. Emani, V. Vishwanath, and A. K. Somani, "A survey of techniques for optimizing transformer inference," *Journal of Systems Architecture (JSA)*, vol. 144, no. C, 2023.
[38] R. Movva, J. Lei, S. Longpre, A. Gupta, and C. DuBois, "Combining compressions for multiplicative size scaling on natural language tasks," in *International Conference on Computational Linguistics (COLING)*, pp. 2861–2872, 2022.
[39] F. Lagunas, E. Charlaix, V. Sanh, and A. Rush, "Block pruning for faster transformers," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 10619–10629, 2021.
[40] Y. Chen, Y. Yan, Q. Yang, Y. Shu, S. He, Z. Shi, and J. Chen, "Accept: An acceleration scheme for speeding up edge pipeline-parallel training," *IEEE Transactions on Mobile Computing (TMC)*, vol. 23, no. 12, pp. 10938–10951, 2024.
[41] W. Dai, J. Fan, Y. Miao, and K. Hwang, "Deep learning model compression with rank reduction in tensor decomposition," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, vol. 36, no. 1, pp. 1315–1328, 2025.
[42] X. Ma, G. Fang, and X. Wang, "Llm-pruner: On the structural pruning of large language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, pp. 21702–21720, 2023.
[43] M. Hashemizadeh, J. Ramirez, R. Sukumaran, G. Farnadi, S. Lacoste-Julien, and J. Gallego-Posada, "Balancing act: Constraining disparate impact in sparse models," in *International Conference on Learning Representations (ICLR)*, 2024.
[44] M. Du, S. Mukherjee, Y. Cheng, M. Shokouhi, X. Hu, and A. H. Awadallah, "Robustness challenges in model distillation and pruning for natural language understanding," in *European Chapter of the Association for Computational Linguistics (EACL)*, pp. 1766–1778, 2023.
[45] H.-I. Liu, M. Galindo, H. Xie, L.-K. Wong, H.-H. Shuai, Y.-H. Li, and W.-H. Cheng, "Lightweight deep learning for resource-constrained environments: A survey," *ACM Computing Surveys*, vol. 56, no. 10, 2024.
[46] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.
[47] S. Yu, P. Nguyen, A. Anwar, and A. Jannesari, "Adaptive dynamic pruning for non-iid federated learning," *arXiv preprint arXiv:2106.06921*, vol. 2, 2021.
[48] Y. Wang, X. Zhang, X. Hu, B. Zhang, and H. Su, "Dynamic network pruning with interpretable layerwise channel selection," in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 34, pp. 6299–6306, 2020.
[49] A. Desai and A. Shrivastava, "In defense of parameter sharing for model-compression," in *International Conference on Learning Representations (ICLR)*, 2024.
[50] J. Wang, Y.-G. Chen, I.-C. Lin, B. Li, and G. L. Zhang, "Basis sharing: Cross-layer parameter sharing for large language model compression," in *International Conference on Learning Representations (ICLR)*, 2025.
[51] S. Bae, A. Fisch, H. Harutyunyan, Z. Ji, S. Kim, and T. Schuster, "Relaxed recursive transformers: Effective parameter sharing with layer-wise loRA," in *International Conference on Learning Representations (ICLR)*, 2025.
[52] J. Li, Q. Nie, W. Fu, Y. Lin, G. Tao, Y. Liu, and C. Wang, "Lors: Low-rank residual structure for parameter-efficient network stacking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15866–15876, 2024.
[53] M. Wallingford, H. Li, A. Achille, A. Ravichandran, C. Fowlkes, R. Bhotika, and S. Soatto, "Task adaptive parameter sharing for multi-task learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
[54] K. Park and N. I. Cho, "Partial filter-sharing: Improved parameter-sharing method for single image super-resolution networks," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2025.

[55] S. Zhang and V. Papyan, "OATS: Outlier-aware pruning through sparse and low rank decomposition," in *International Conference on Learning Representations (ICLR)*, 2025.
[56] V. Boža and V. Macko, "Two sparse matrices are better than one: Sparsifying neural networks with double sparse factorization," in *International Conference on Learning Representations (ICLR)*, 2025.
[57] W. Qinsi, J. Ke, M. Tomizuka, K. Keutzer, and C. Xu, "Dobi-SVD: Differentiable SVD for LLM compression and some new perspectives," in *International Conference on Learning Representations (ICLR)*, 2025.
[58] X.-H. Liu, Y. Du, J. Wang, and Y. Yu, "On the optimization landscape of low rank adaptation methods for large language models," in *International Conference on Learning Representations (ICLR)*, 2025.
[59] H. Zheng, X. Lin, H. Liang, B. Zhou, and Y. Liang, "Region-aware mutual relational knowledge distillation for semantic segmentation," *Pattern Recognition (PR)*, vol. 161, p. 111319, 2025.
[60] Z.-L. Ni, F. Yang, S. Wen, and G. Zhang, "Dual relation knowledge distillation for object detection," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.
[61] W. Xu, Q. Wu, Z. Liang, J. Han, X. Ning, Y. Shi, W. Lin, and Y. Zhang, "SLMRec: Distilling large language models into small for sequential recommendation," in *International Conference on Learning Representations (ICLR)*, 2025.
[62] Y. Zhang, X. Ma, Y. Bai, H. Wang, and Y. Fu, "Accessing vision foundation models via imagenet-1k," in *International Conference on Learning Representations (ICLR)*, 2025.
[63] Z. Hao, J. Guo, K. Han, Y. Tang, H. Hu, Y. Wang, and C. Xu, "One-for-all: Bridge the gap between heterogeneous architectures in knowledge distillation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
[64] S. Dasgupta and T. Cohn, "Improving language model distillation through hidden state matching," in *International Conference on Learning Representations (ICLR)*, 2025.
[65] F. Shu, Y. Liao, L. Zhang, L. Zhuo, C. Xu, G. Zhang, H. Shi, L. Chan, TaoZhong, Z. Yu, W. He, S. Fu, H. Li, S. Liu, H. Li, and H. Jiang, "LLaVA-mod: Making LLaVA tiny via moe-knowledge distillation," in *International Conference on Learning Representations (ICLR)*, 2025.
[66] W. Xu, R. Han, Z. Wang, L. Le, D. Madeka, L. Li, W. Y. Wang, R. Agarwal, C.-Y. Lee, and T. Pfister, "Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling," in *International Conference on Learning Representations (ICLR)*, 2025.
[67] S. Sun, W. Ren, J. Li, R. Wang, and X. Cao, "Logit standardization in knowledge distillation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15731–15740, 2024.
[68] Y. Wei, Z. Hu, L. Shen, Z. Wang, C. Yuan, and D. Tao, "Open-vocabulary customization from CLIP via data-free knowledge distillationdasgupta2025improving," in *International Conference on Learning Representations (ICLR)*, 2025.
[69] T. Dettmers, R. A. Svirschevski, V. Egiazarian, D. Kuznedelev, E. Frantar, S. Ashkboos, A. Borzunov, T. Hoefler, and D. Alistarh, "SpQR: A sparse-quantized representation for near-lossless LLM weight compression," in *International Conference on Learning Representations (ICLR)*, 2024.
[70] J. Liu, R. Gong, X. Wei, Z. Dong, J. Cai, and B. Zhuang, "QLLM: Accurate and efficient low-bitwidth quantization for large language models," in *International Conference on Learning Representations (ICLR)*, 2024.
[71] J. H. Heo, J. Kim, B. Kwon, B. Kim, S. J. Kwon, and D. Lee, "Rethinking channel dimensions to isolate outliers for low-bit weight quantization of large language models," in *International Conference on Learning Representations (ICLR)*, 2024.
[72] X. Ding, X. Liu, Z. Tu, Y. Zhang, W. Li, J. Hu, H. Chen, Y. Tang, Z. Xiong, B. Yin, and Y. Wang, "CBQ: Cross-block quantization for large language models," in *International Conference on Learning Representations (ICLR)*, 2025.
[73] M. Li, Y. Lin, Z. Zhang, T. Cai, J. Guo, X. Li, E. Xie, C. Meng, J.-Y. Zhu, and S. Han, "SVDQuant: Absorbing outliers by low-rank component for 4-bit diffusion models," in *International Conference on Learning Representations (ICLR)*, 2025.
[74] Z. Yuan, Y. Shang, and Z. Dong, "PB-LLM: Partially binarized large language models," in *International Conference on Learning Representations (ICLR)*, 2024.
[75] M. Fishman, B. Chmiel, R. Banner, and D. Soudry, "Scaling FP8 training to trillion-token LLMs," in *International Conference on Learning Representations (ICLR)*, 2025.
[76] Y. He, J. Liu, W. Wu, H. Zhou, and B. Zhuang, "EfficientDM: Efficient quantization-aware fine-tuning of low-bit diffusion models," in *International Conference on Learning Representations (ICLR)*, 2024.
[77] Z. Liu, B. Oguz, C. Zhao, E. Chang, P. Stock, Y. Mehdad, Y. Shi, R. Krishnamoorthi, and V. Chandra, "LLM-QAT: Data-free quantization aware training for large language models," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 467–484, 2024.
[78] Q. Le, E. Diao, Z. Wang, X. Wang, J. Ding, L. Yang, and A. Anwar, "Probe pruning: Accelerating LLMs through dynamic pruning via model-probing," in *International Conference on Learning Representations (ICLR)*, 2025.
[79] M. Someki, Y. Peng, S. Arora, M. Müller, A. Mouchtaris, G. Strimel, J. Liu, and S. Watanabe, "Context-aware dynamic pruning for speech foundation models," in *International Conference on Learning Representations (ICLR)*,

2025.
[80] Y.-L. Sung, J. Yoon, and M. Bansal, "ECoFLap: Efficient coarse-to-fine layer-wise pruning for vision-language models," in *International Conference on Learning Representations (ICLR)*, 2024.
[81] M. Sun, Z. Fang, J. Wang, J. Jiang, D. Kong, C. Hu, Y. FANG, and R. Xu, "Optimal brain apoptosis," in *International Conference on Learning Representations (ICLR)*, 2025.
[82] R. Lucas and R. Mazumder, "Preserving deep representations in one-shot pruning: A hessian-free second-order optimization framework," in *International Conference on Learning Representations (ICLR)*, 2025.
[83] D. Wang, H. Šikić, L. Thiele, and O. Saukh, "Forget the data and fine-tuning! just fold the network to compress," in *International Conference on Learning Representations (ICLR)*, 2025.
[84] R. Han and J. Tang, "Straightforward layer-wise pruning for more efficient visual adaptation," in *European Conference on Computer Vision (ECCV)*, p. 236–252, 2024.
[85] P. Dong, L. Li, Z. Tang, X. Liu, X. Pan, Q. Wang, and X. Chu, "Pruner-zero: Evolving symbolic pruning metric from scratch for large language models," in *International Conference on Machine Learning (ICML)*, 2024.
[86] A. Bair, H. Yin, M. Shen, P. Molchanov, and J. M. Álvarez, "Adaptive sharpness-aware pruning for robust sparse networks," in *International Conference on Learning Representations (ICLR)*, 2024.
[87] T. F. A. van der Ouderaa, M. Nagel, M. V. Baalen, and T. Blankevoort, "The LLM surgeon," in *International Conference on Learning Representations (ICLR)*, 2024.
[88] Y. Zhang, L. Zhao, M. Lin, Y. Sun, Y. Yao, X. Han, J. Tanner, S. Liu, and R. Ji, "Dynamic sparse no training: Training-free fine-tuning for sparse llms," in *International Conference on Learning Representations (ICLR)*, 2024.
[89] G. Li, L. Yin, J. Ji, W. Niu, M. Qin, B. Ren, L. Guo, S. Liu, and X. Ma, "Neurrev: Train better sparse neural network practically via neuron revitalization," in *International Conference on Learning Representations (ICLR)*, 2024.
[90] H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 46, no. 12, pp. 10558–10578, 2024.
[91] D. Shi, C. Tao, Y. Jin, Z. Yang, C. Yuan, and J. Wang, "Upop: unified and progressive pruning for compressing vision-language transformers," in *International Conference on Machine Learning (ICML)*, 2023.
[92] Z. Mo, H. Shi, and S. J. Pan, "Probabilistic neural pruning via sparsity evolutionary fokker-planck-kolmogorov equation," in *International Conference on Learning Representations (ICLR)*, 2025.
[93] Y. Ding, K. Fan, Y. Wang, X. Sun, and Y. Fu, "Adaptive pruning of pretrained transformer via differential inclusions," in *International Conference on Learning Representations (ICLR)*, 2025.
[94] G. Bai, Y. Li, C. Ling, K. Kim, and L. Zhao, "SparseLLM: Towards global pruning of pre-trained language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
[95] M. Xia, T. Gao, Z. Zeng, and D. Chen, "Sheared LLaMA: Accelerating language model pre-training via structured pruning," in *International Conference on Learning Representations (ICLR)*, 2024.
[96] S. Park, H. Choi, and U. Kang, "Accurate retraining-free pruning for pretrained encoder-based language models," in *International Conference on Learning Representations (ICLR)*, 2024.
[97] Y. Liang, J. Long, Z. Shi, Z. Song, and Y. Zhou, "Beyond linear approximations: A novel pruning approach for attention matrix," in *International Conference on Learning Representations (ICLR)*, 2025.
[98] M. Zimmer, C. Spiegel, and S. Pokutta, "Sparse model soups: A recipe for improved pruning via model averaging," in *International Conference on Learning Representations (ICLR)*, 2024.
[99] M. Choi, H. Lee, G. Nam, and J. Lee, "Sparse weight averaging with multiple particles for iterative magnitude pruning," in *International Conference on Learning Representations (ICLR)*, 2024.
[100] Z. Atashgahi, X. Zhang, N. Kichler, S. Liu, L. Yin, M. Pechenizkiy, R. Veldhuis, and D. C. Mocanu, "Supervised feature selection with neuron evolution in sparse neural networks," *Transactions on Machine Learning Research (TMLR)*, 2023.
[101] G. Sokar, E. Mocanu, D. C. Mocanu, M. Pechenizkiy, and P. Stone, "Dynamic sparse training for deep reinforcement learning," in *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3437–3443, 2022.
[102] T. Chen, Y. Cheng, Z. Gan, L. Yuan, L. Zhang, and Z. Wang, "Chasing sparsity in vision transformers: An end-to-end exploration," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 19974–19988, 2021.
[103] J. Shen, Q. Xu, G. Pan, and B. Chen, "Improving the sparse structure learning of spiking neural networks from the view of compression efficiency," in *International Conference on Learning Representations (ICLR)*, 2025.
[104] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *International Conference on Learning Representations (ICLR)*, 2017.
[105] L. Yin, Y. Wu, Z. Zhang, C.-Y. Hsieh, Y. Wang, Y. Jia, G. Li, A. K. JAISWAL, M. Pechenizkiy, Y. Liang, M. Bendersky, Z. Wang, and S. Liu, "Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to high sparsity," in *International Conference on Machine Learning (ICML)*, 2024.
[106] S. Ashkboos, M. L. Croci, M. G. do Nascimento, T. Hoefler, and J. Hensman, "SliceGPT: Compress large lan-

guage models by deleting rows and columns," in *International Conference on Learning Representations (ICLR)*, 2024.
[107] J. Li, W. Gao, Q. Lei, and D. Xu, "Breaking through deterministic barriers: Randomized pruning mask generation and selection," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.
[108] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9194–9203, 2018.
[109] Z. Yao, L. Ma, S. Shen, K. Keutzer, and M. W. Mahoney, "Mlpruning: A multilevel structured pruning framework for transformer-based models," *arXiv preprint arXiv:2105.14636*, 2021.
[110] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A simple and effective pruning approach for large language models," in *International Conference on Machine Learning (ICML)*, 2023.
[111] Z. Song, R. Wang, D. Ru, Z. Peng, H. Huang, H. Zhao, X. Liang, and L. Jiang, "Approximate random dropout for dnn training acceleration in gpgpu," in *Design, Automation & Test in Europe (DATE)*, pp. 108–113, 2019.
[112] A. Sharma, N. Wolfe, and B. Raj, "The incredible shrinking neural network: New perspectives on learning representations through the lens of pruning," *arXiv preprint arXiv:1701.04465*, 2017.
[113] X. Ma, S. Lin, S. Ye, Z. He, L. Zhang, G. Yuan, S. H. Tan, Z. Li, D. Fan, X. Qian, X. Lin, K. Ma, and Y. Wang, "Non-structured dnn weight pruning—is it beneficial in any platform?," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, vol. 33, no. 9, pp. 4930–4944, 2022.
[114] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, p. 2082–2090, 2016.
[115] H. Zhang, Y. Zhou, and G.-H. Wang, "Dense vision transformer compression with few samples," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15825–15834, 2024.
[116] A. Gromov, K. Tirumala, H. Shapourian, P. Glorioso, and D. Roberts, "The unreasonable ineffectiveness of the deeper layers," in *International Conference on Learning Representations (ICLR)*, 2025.
[117] X. Chen, Y. Hu, J. Zhang, Y. Wang, C. Li, and H. Chen, "Streamlining redundant layers to compress large language models," in *International Conference on Learning Representations (ICLR)*, 2025.
[118] H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 46, no. 12, pp. 10558–10578, 2024.
[119] W. Hua, Y. Zhou, C. De Sa, Z. Zhang, and G. E. Suh, "Boosting the performance of cnn accelerators with dynamic fine-grained channel gating," in *International Symposium on Microarchitecture (MICRO)*, p. 139–150, 2019.
[120] Z. Liu, C. Zhao, I. Fedorov, B. Soran, D. Choudhary, R. Krishnamoorthi, V. Chandra, Y. Tian, and T. Blankevoort, "Spinquant: LLM quantization with learned rotations," in *International Conference on Learning Representations (ICLR)*, 2025.
[121] T. Zhang and A. Shrivastava, "Leanquant: Accurate and scalable large language model quantization with loss-error-aware grid," in *International Conference on Learning Representations (ICLR)*, 2025.
[122] X. Hu, Y. Cheng, D. Yang, Z. Chen, Z. Xu, JiangyongYu, XUCHEN, Z. Yuan, Z. jiang, and S. Zhou, "OSTQuant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting," in *International Conference on Learning Representations (ICLR)*, 2025.
[123] Z. Li, X. Yan, T. Zhang, H. Qin, D. Xie, J. Tian, zhongchao shi, L. Kong, Y. Zhang, and X. Yang, "ARB-LLM: Alternating refined binarizations for large language models," in *International Conference on Learning Representations (ICLR)*, 2025.
[124] R. Shafipour, D. Harrison, M. Horton, J. MARKER, H. Bedayat, S. Mehta, M. Rastegari, M. Najibi, and S. Naderiparizi, "SeedLM: Compressing LLM weights into seeds of pseudo-random generators," in *International Conference on Learning Representations (ICLR)*, 2025.
[125] P. Dong, L. Li, Y. Zhong, D. Du, R. FAN, Y. Chen, Z. Tang, Q. Wang, W. Xue, Y. Guo, and X. Chu, "STBLLM: Breaking the 1-bit barrier with structured binary LLMs," in *International Conference on Learning Representations (ICLR)*, 2025.
[126] K. Yi, Z. Liu, jianwei zhang, C. Li, T. Zhang, J. Lin, and J. Zhou, "Rotated runtime smooth: Training-free activation smoother for accurate INT4 inference," in *International Conference on Learning Representations (ICLR)*, 2025.
[127] S. Zhou, L. Li, X. Zhang, B. Zhang, S. Bai, M. Sun, Z. Zhao, X. Lu, and X. Chu, "LiDAR-PTQ: Post-training quantization for point cloud 3d object detection," in *International Conference on Learning Representations (ICLR)*, 2024.
[128] J. Liu, L. Niu, Z. Yuan, D. Yang, X. Wang, and W. Liu, "Pd-quant: Post-training quantization based on prediction difference metric," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24427–24437, 2023.
[129] A. Kaushal, T. Vaidhya, A. K. Mondal, T. Pandey, A. Bhagat, and I. Rish, "Surprising effectiveness of pretraining ternary language model at scale," in *International Conference on Learning Representations (ICLR)*, 2025.

[130] E. YVINEC, A. Dapogny, and K. Bailly, "Network memory footprint compression through jointly learnable codebooks and mappings," in *International Conference on Learning Representations (ICLR)*, 2024.
[131] Y. Ma, H. Li, X. Zheng, F. Ling, X. Xiao, R. Wang, S. Wen, F. Chao, and R. Ji, "Affinequant: Affine transformation quantization for large language models," in *International Conference on Learning Representations (ICLR)*, 2024.
[132] G. Park, B. park, M. Kim, S. Lee, J. Kim, B. Kwon, S. J. Kwon, B. Kim, Y. Lee, and D. Lee, "LUT-GEMM: Quantized matrix multiplication based on LUTs for efficient inference in large-scale generative language models," in *International Conference on Learning Representations (ICLR)*, 2024.
[133] J. Gou, B. Yu, S. J. Maybank, *et al.*, "Knowledge distillation: A survey," *International Journal of Computer Vision (IJCV)*, vol. 129, pp. 1789–1819, 2021.
[134] Y. Jin, J. Wang, and D. Lin, "Multi-level logit distillation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24276–24285, 2023.
[135] Y. Bai, Z. Wang, J. Xiao, C. Wei, H. Wang, A. Yuille, Y. Zhou, and C. Xie, "Masked autoencoders enable efficient knowledge distillers," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24256–24265, 2023.
[136] J. Kim, J. You, D. Lee, H. Y. Kim, and J.-H. Jung, "Do topological characteristics help in knowledge distillation?," in *International Conference on Machine Learning (ICML)*, 2024.
[137] M. I. Hossain, S. Akhter, C. S. Hong, and E.-N. Huh, "Single teacher, multiple perspectives: Teacher knowledge augmentation for enhanced knowledge distillation," in *International Conference on Learning Representations (ICLR)*, 2025.
[138] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3962–3971, 2019.
[139] C. Yang, H. Zhou, Z. An, X. Jiang, Y. Xu, and Q. Zhang, "Cross-image relational knowledge distillation for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12309–12318, 2022.
[140] Y. Liu, J. Cao, B. Li, C. Yuan, W. Hu, Y. Li, and Y. Duan, "Knowledge distillation via instance relationship graph," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7089–7097, 2019.
[141] S. Lin, R. Ji, C. Chen, D. Tao, and J. Luo, "Holistic cnn compression via low-rank decomposition with knowledge transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 41, no. 12, pp. 2889–2905, 2019.
[142] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations (ICLR)*, 2022.
[143] H. Guo, P. Greengard, E. Xing, and Y. Kim, "LQ-loRA: Low-rank plus quantized matrix decomposition for efficient language model finetuning," in *International Conference on Learning Representations (ICLR)*, 2024.
[144] S. A. Koohpayegani, N. K. L, P. Nooralinejad, S. Kolouri, and H. Pirsiavash, "NOLA: Compressing loRA using linear combination of random basis," in *International Conference on Learning Representations (ICLR)*, 2024.
[145] V. Lialin, S. Muckatira, N. Shivagunde, and A. Rumshisky, "ReloRA: High-rank training through low-rank updates," in *International Conference on Learning Representations (ICLR)*, 2024.
[146] S. Wang, L. Chen, P. CHEN, J. Dong, B. XUE, J. Jiang, L. Kong, and C. Wu, "Mos: Unleashing parameter efficiency of low-rank adaptation with mixture of shards," in *International Conference on Learning Representations (ICLR)*, 2025.
[147] Y. Xu, L. Xie, X. Gu, X. Chen, H. Chang, H. Zhang, Z. Chen, X. ZHANG, and Q. Tian, "QA-loRA: Quantization-aware low-rank adaptation of large language models," in *International Conference on Learning Representations (ICLR)*, 2024.
[148] Y. Li, Y. Yu, C. Liang, N. Karampatziakis, P. He, W. Chen, and T. Zhao, "Loftq: LoRA-fine-tuning-aware quantization for large language models," in *International Conference on Learning Representations (ICLR)*, 2024.
[149] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009.
[150] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.
[151] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
[152] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3973–3981, 2015.
[153] P. Kaur, , K. Sikka, W. Wang, s. Belongie, and A. Divakaran, "Foodx-251: A dataset for fine-grained food classification," *arXiv preprint arXiv:1907.06167*, 2019.
[154] R. Puri and B. Catanzaro, "Zero-shot text classification with generative language models," *arXiv preprint arXiv:1912.10165*, 2019.
[155] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[156] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy, "The Pile: An 800gb dataset of diverse text for language modeling," *arXiv preprint arXiv:2101.00027*, 2020.
[157] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *International Conference on Learning Representations (ICLR)*, 2019.
[158] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," *arXiv preprint arXiv:1905.00537*, 2019.
[159] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 784–789, 2018.
[160] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer, "TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1601–1611, 2017.
[161] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov, "Natural questions: A benchmark for question answering research," *Transactions of the Association for Computational Linguistics (TACL)*, vol. 7, pp. 452–466, 2019.
[162] Y. Bisk, R. Zellers, R. Le bras, J. Gao, and Y. Choi, "Piqa: Reasoning about physical commonsense in natural language," in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 34, pp. 7432–7439, 2020.
[163] M. Sap, H. Rashkin, D. Chen, R. Le Bras, and Y. Choi, "Social IQa: Commonsense reasoning about social interactions," in *Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.
[164] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, "Can a suit of armor conduct electricity? a new dataset for open book question answering," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.
[165] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," *International Conference on Learning Representations (ICLR)*, 2021.
[166] D. Hendrycks, C. Burns, S. Basart, A. Critch, J. Li, D. Song, and J. Steinhardt, "Aligning ai with shared human values," *International Conference on Learning Representations (ICLR)*, 2021.
[167] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," *arXiv preprint arXiv:1803.05457*, 2018.
[168] L. Yu, W. Jiang, H. Shi, J. Yu, Z. Liu, Y. Zhang, J. T. Kwok, Z. Li, A. Weller, and W. Liu, "Metamath: Bootstrap your own mathematical questions for large language models," *arXiv preprint arXiv:2309.12284*, 2023.
[169] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "HellaSwag: Can a machine really finish your sentence?," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4791–4800, 2019.
[170] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.
[171] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, *et al.*, "Program synthesis with large language models," *arXiv preprint arXiv:2108.07732*, 2021.
[172] A. Ajith, C. Pan, M. Xia, A. Deshpande, and K. Narasimhan, "Instructeval: Systematic evaluation of instruction selection methods," in *North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 4336–4350, 2024.
[173] Y. Dubois, P. Liang, and T. Hashimoto, "Length-controlled alpacaeval: A simple debiasing of automatic evaluators," in *Conference on Language Modeling (COLM)*, 2024.
[174] F. Liang, Z. Zhang, H. Lu, V. Leung, Y. Guo, and X. Hu, "Communication-efficient large-scale distributed deep learning: A comprehensive survey," *arXiv preprint arXiv:2404.06114*, 2024.
[175] Y. Shen, M. Sun, J. Lin, J. Zhao, and A. Zou, "Order of compression: A systematic and optimal sequence to combinationally compress cnn," *arXiv preprint arXiv:2403.17447*, 2024.
[176] Y. Zheng, Y. Chen, B. Qian, X. Shi, Y. Shu, and J. Chen, "A review on edge large language models: Design, execution, and applications," *ACM Computing Surveys*, vol. 57, no. 8, 2025.
[177] G. I. Kim, S. Hwang, and B. Jang, "Efficient compressing and tuning methods for large language models: A systematic literature review," *ACM Computing Surveys*, 2025.