Inference Optimizations for Large Language Models: Effects, Challenges, and Practical Considerations

Donisch, Leo
Business Faculty

Ansbach University of Applied Sciences
Ansbach, Germany
donisch19473@hs-ansbach.de

2nd Sigurd Schacht
Faculty of Business
University of Applied Science
Ansbach, Germany
sigurd.schacht@hs-ansbach.de

3rd Carsten Lanquillon

Faculty of Business

University of Applied Science

Heilbronn, Germany

carsten.lanquillon@hs-heilbronn.de

Abstract—Large language models are ubiquitous in natural language processing because they can adapt to new tasks without retraining. However, their sheer scale and complexity present unique challenges and opportunities, prompting researchers and practitioners to explore novel model training, optimization, and deployment methods. This literature review focuses on various techniques for reducing resource requirements and compressing large language models, including quantization, pruning, knowledge distillation, and architectural optimizations. The primary objective is to explore each method in-depth and highlight its unique challenges and practical applications. The discussed methods are categorized into a taxonomy that presents an overview of the optimization landscape and helps navigate it to understand the research trajectory better.

Index Terms—Neural Networks, Transformers, Inference Optimization, Quantization, Pruning, Knowledge Distillation, Attention, Attention Optimization, Decoding, Decoding Optimization

I. INTRODUCTION

In recent years, Large Language Models (LLMs) have emerged as the cornerstone of Natural Language Processing (NLP), revolutionizing various domains with unprecedented capabilities. These versatile models have demonstrated remarkable abilities in diverse applications, ranging from assisting in code generation [1] [2], to facilitating news summarization [3] [4], and even augmenting information retrieval systems for improved search accuracy and efficiency [5] [6]. Furthermore, these models' sheer scale and complexity present unique challenges and opportunities, prompting researchers and practitioners to explore novel model training, optimization, and deployment methods. Optimizing large models for speed, reducing resource consumption, and making them more accessible is a significant part of LLM research.

The primary objective of this research paper is to explore various techniques for reducing resource requirements and compressing large language models, including analyzing each method in-depth and highlighting its unique challenges and practical implications. The discussed methods include quantization, pruning, knowledge distillation, and architectural optimizations. To better understand the relationship between these

techniques, they are categorized into a taxonomy that presents an overview of the optimization landscape and helps navigate it for a better understanding of the research trajectory. Refer to figure 1 for a visual representation of the categorization and to the respective section for a more detailed look at the discussed literature in each category.

II. PRELIMINARIES

A. Transformers for Language Modeling

In recent years, the transformer has become the primary architecture for Natural Language Processing (NLP) tasks [7] [8] [9] [10]. The prominence is because of the attention mechanism, which enables the transformer to efficiently focus on different text parts and learn complex language structures [11] [12]. The mechanism aims to determine the significance of various elements in a sequence about a specific element [11] [12]. These learned structures enable the transformer to solve complex NLP problems. In addition to that, the transformer typically comprises multiple transformer blocks, each containing an attention module and a feed-forward module [11] [13]. The feed-forward model facilitates the learning of mappings between the input and output and can be used in the encoder or decoder blocks [11] [12]. The encoder processes the input sequence in parallel, while the decoder blocks are autoregressive, performing inference once per output token

B. Emergent Abilities in Large Language Models

In recent years, models have been adapted to different tasks. That means training a new model each time a different task is required, for example, summarization or question answering [14] [15]. For example a fine-tuned version of RoBERTa [16] for question answering trained on the SQuAD dataset [17] or DistillBERT [18] for text classification on the SST-2 dataset [19]. Providing a different model can be challenging if a suitable one still needs training. However, large language models have special abilities that allow them to execute new, unseen tasks simply by understanding what to do by providing a task description with a few examples. These abilities are

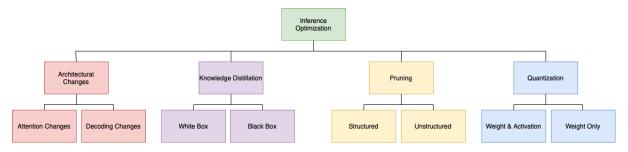


Fig. 1. Taxonomy of optimization techniques

known as *Emergent Abilities* [3] [7]. For example, an LLM can use these abilities to solve complex mathematical problems by only providing the necessary context in the prompt without retraining [20]. Therefore, addressing Emergent Abilities can be categorized by their prompting strategies. The authors of [3] divide it into Few-Shot and Prompt Augmentation strategies. Few-shot prompting provides relevant examples to infer the desired behavior. At the same time, prompt augmentation captures all strategies that help the model better understand the desired behavior without explicitly providing few-shot examples. A prominent example would be chain-of-thought [20], which provides the model with relevant intermediate steps to solve the problem and a task description.

C. Hardware Representations of Numbers

Graphic Processing Units (GPU) are used in machine learning to speed up computations [21] [22] [23]. To use the improvement, the hardware representation of numbers is in bits for the GPU to understand [24]. Typically, the representation uses 32 bits, the number of bits used to convey a number's meaning to the device. In machine learning, this is referred to as bit precision. Higher precision enables conveying more significant numbers, and lower precision means conveying fewer numbers. Reducing precision requires less memory space [25] [26] [27].

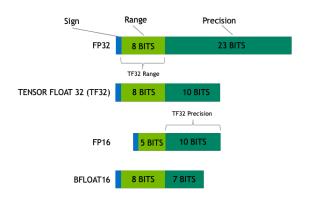


Fig. 2. Different representation formats used in machine learning [28]

Displaying a number on a hardware level involves three parts: the *Sign*, the *Range*, and *Precision*, as shown in figure 2. Whether a number is positive or negative is indicated by

the Sign [24]. The Range refers to the amount of representable numbers. The larger the Range, the more numbers (higher and lower) can be represented [24]. Precision refers to how many decimals can be accurately represented [24]. The larger the Precision, the more fine-grained the numbers can be.

For LLMs, memory usage is determined by the amount of parameters and their precision. Due to the high memory requirements of FP32, LLMs default to using FP16. For a model with approximately 175 billion parameters, the required memory space would be approximately 334 GB.

III. INFERENCE OPTIMIZATION

Inference Optimization describes the procedure for enhancing the speed, efficiency, and performance of an LLM while preserving the quality of an uncompressed baseline. For large language models, this involves processing the input more rapidly and generating output more efficiently or with greater resource efficiency.

A. Quantization

Quantization involves converting high-precision numbers into a lower-precision space while preserving their meaning and significance [29] [30] [31]. The real advantage here is that the memory requirements are significantly lowered because fewer bytes are used to represent the number in a device like a GPU. This allows the device to load data more quickly and perform computations more efficiently, leading to tangible improvements in memory consumption, computational speed, and energy efficiency [32] [12] [33] [27] [34]. These benefits enable more flexible deployment of LLMs on hardware-constrained devices, which is a practical application of this technique.

One way to achieve this is by converting the high-precision floating-point number into a lower-precision one. Another strategy reduces the precision while transforming the floating-point number into an integer [30] [35] [27]. Therefore, the following paragraphs mainly discuss integer quantization in more detail and illuminate literature focusing on float-point numbers.

Integer quantization can be divided into uniform and nonuniform strategies. These strategies describe how the transformation is performed. *Uniform quantization* transforms the floating-point numbers with evenly spaced intervals and maps each interval to an integer value. *Non-uniform quantization*

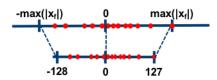


Fig. 3. Symmetric Quantization example [36]

may not have evenly spaced intervals [12] [29]. However, due to the difficulty of using non-uniform quantization with current accelerators such as GPUs, research primarily focuses on uniform quantization, as highlighted by [30] [12].

Uniform quantization is divided into symmetric and asymmetric quantization, which categorizes and describes how the transformation is performed. *Symmetric Quantization* converts the floating point range in a way that sets the zero point of the floating-point range in the same place in the integer space, as visualized in the figure 3 [30] [29] [37]. *Asymmetric Quantization* transforms the minimum and maximum values of the floating-point range to be the lowest and highest value in the integer space, visualized in the figure 4 [30] [29].

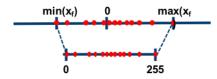


Fig. 4. Asymmetric Quantization example [36]

Appliance on the Transformer. The reduction of precision can be applied to various aspects of the transformer. The two most common strategies are the quantization of weights only or weights and activations. Therefore, the reviewed literature gets categorized into quantization papers that discuss weight-only or weight-and-activation strategies. For a deeper look into the discussed literature, refer to table I.

There are different dimensions to the appliance of quantization to an LLM. For every quantization strategy, it is important to highlight which layer, which layer elements, and what bit precision the compression affects. For example, one of the first experiments focused on compressing the encoder layers of the BERT architecture while only focusing on weights [30] [27] [35]. This evolved to experimenting with the quantization of weights and activations [37] [38]. This resulted in experimenting with different bit precision, which is currently focusing on 8bit [30] [37], 4bit precision [27] [35] or lower [39] [40] [41]. Therefore, these dimensions are highlighted in each piece of literature to provide a comprehensive image of the proposed method. First, the weight-only literature is discussed, followed by literature discussing weight-and-activation quantization.

LLM.Int8() [30] applies 8-bit weight quantization using a vector-wise transformation scheme. The study found that outlier dimensions have a significant impact when quantizing models beyond billions of parameters. They concluded that

TABLE I LITERATURE OVERVIEW FOR QUANTIZATION

Quantization Literature		
Weight only	Weight & Activation	
LLM.int8() [30] GPTQ [35] AWQ [27] OWQ [39] SpQR [41]	ZeroQuant [37] ZeroQuantV2 [38] SmoothQuant [42]	

to achieve high-quality quantization, mixed-precision decomposition is needed, which keeps these outliers in a higher precision format. GTPO [35] investigated the possibility of using 4-bit for compression. Their approach employs a layerwise scheme, where a reconstruction problem is solved in each layer. This yielded a higher quality quantization by reducing transformation inaccuracies. Dettmers and Zettlemoyer [43] researched the trade-off between quantization bit width and model size. They found that 4-bit compression is almost universally optimal for achieving performance improvements while maintaining quality. AWQ [27] focuses on linear layers and employs a per-channel quantization scheme. The activations in each channel are analyzed to identify salient weights and quantized differently to preserve the importance of these weights. However, all weights are assigned the same bit-width to avoid the performance cost of mixed-precision quantization. OWQ [39] focuses only on the linear layers and follows the same principle as AWQ by analyzing the importance of weights. They conclude with a different quantization scheme that keeps important weights in higher precision while compressing all others. SpQR [41] is based on the insights of GPTO and uses a layer-by-layer approach. The quantization process is divided into two steps. Firstly, weights are identified and isolated based on their output behavior. Secondly, a quantization step is applied to the remaining weights, reducing them to 3-4 bits.

In addition to weight-only quantization, quantization can also be applied to weights and activations. ZeroQuant [37] conducted research on the compression of weights and activations for large language models. They found that quantizing activations with a finer granularity than weights is essential to preserve quality. They conclude that weights allow 4-bit compression but keep quantized activations in higher 8-bit precision to preserve quality. They point out that for the BLOOM [44] family of models, better quantization could be achieved with a more coarse-grained granularity and higher precision rather than finer-grained granularity with lower precision. SmoothQuant [42] focuses on the linear layers and uses a per-channel smoothing technique. They conclude shifting the quantization difficulty from the activations to the weights retains more activation information. OmniQuant [40] optimizes weight and activation quantization by introducing separate learnable parameters for both weights and activations. This approach enables a wide range of quantization settings to be adjusted more effectively.

The Practical Aspect. The practical application of quantization can be divided into Quantization Aware Training (QAT) and Post Training Quantization (PTQ). QAT aims to improve the efficiency and accessibility of language models by applying quantization during training. With quantization employed during training, the model can now learn low-precision representations, and subsequent weight updates can help mitigate any potential quality degradation caused by data type transformation. The principle was first applied using small BERT models, demonstrating the quantization possibilities [29] [31]. This enables the fine-tuning of LLMs with billions of parameters with only a handful of GPUs [30] [45]. Multiple LLMs can now efficiently be trained and swapped out for inference when needed.

If the aim is to train a model for a specific task while considering high resource requirements, Quantization Aware Training can be helpful. Nevertheless, many fine-tuned models are available, like llama2-chat [46] or CodeLlama [47]. Post-training quantization can help manage storage and deployment challenges without significant weight changes or extensive training efforts. Therefore, there is a significant focus on researching PTQ techniques with minimal impact on quality.

The quantization error. Quantization can introduce inaccuracies due to data type transformation, which can significantly impact the quality of a compressed model. To make PTQ feasible, the focus is on reducing these inaccuracies as much as possible while preserving the model characteristics and knowledge representation. The challenges of using quantization effectively are discussed in the following paragraph.

LLM.Int8() [30] identified a challenge in applying quantization to transformers with more than a billion parameters: the emergence of outlier features in large models. These outliers are difficult to quantize due to their extreme magnitudes, which can cause a shift in integer representation. In addition, AWQ [27] found through an analysis of the resulting activations that not all weights are of equal importance. ZeroQuant [37], ZeroQuant-V2 [38], and SmoothQuant [42] found that quantizing activations is significantly more challenging and leads to a more significant degradation in quality. This conclusion is why the latter proposed shifting the complexity from the activations to the weights.

Practical Constraints. Practical challenges must be considered to reduce resource consumption and speed up computation. Many methods rely on hardware support for particular data types, like 8-bit integers and special operators, to use them efficiently [48]. So, there are two practical dimensions to remember: The amount of resources needed to compute the compression and the inference constraints needed to achieve a practical speed-up.

The impact of bit width, quantization schema, and granularity on quantization efficiency is significant. For instance, ZeroQuant [37] and SpQR [41] can compress models with around 175 billion parameters in a matter of hours, while other methods require more compute time to achieve the same level of compression [42] [30] [29]. Therefore, it is essential to consider this factor when selecting a quantization variant.

The resource requirements during compression and the constraints in using the compressed model are essential for consideration. Each method applies its compression differently, which could result in different inference behaviors. Methods such as LLM.Int8() [30] can effectively reduce the memory consumption of LLMs, but it does not achieve a significant computational speed-up because of the need for dequantization during inference. The cause is the proposed mixed-precision approach, which significantly impacts quantization and decomposition overhead and relies on optimized GPU kernels. Another constraint is the hardware support for the quantization target. Additional overhead will slow down the compressed model if the hardware does not support the quantization target. For example, ZeroOuant [37] provides a custom Nyidia GPU kernel named CUTLASS to improve integer multiplications and reduce data loading overhead. OWQ [39] and SpQR [41] offer custom kernels and back-end implementations to accelerate data loading overhead and matrix multiplications.

Therefore, the targeted deployment environment significantly impacts the optimization technique chosen. Furthermore, it is essential to consider that if methods rely on custom code for specific hardware, their deployment may be more complicated than expected. In AWQ [27], the authors mention that they built the algorithm upon the default PyTorch [21] API, which does not rely on custom code and makes the use of a wide range of hardware configurations possible.

In Conlusion. The choice of quantization method depends on several factors, including resource requirements, inference constraints, and the desired balance between compression efficiency and model quality.

- Quantization involves representing numerical data in a format that can be understood by hardware, such as GPUs.
- The main concept is converting precise numerical values into less precise ones while retaining their meaning, thus reducing memory and computation complexity.
- Quantization can be divided into uniform and nonuniform. Uniform quantization evenly spaces intervals of floating point numbers.
- The uniform quantization, whether symmetric or asymmetric, can affect the data distribution mapping.
- Quantization can be applied either during training, known as Quantization Aware Training (QAT), or after training, known as Post Training Quantization (PTQ).
- Inaccuracies may arise from data type transformations, which can significantly impact model quality.
- Many different methods focus on quantizing weights, activations, or both. They use different quantization schemes and bit widths.
- Resource requirements for optimization and inference constraints must be considered.
- Constraints for quantization include quantization overhead, hardware support for the quantization target, and deployment complexity.

B. Pruning

Pruning describes identifying and removing redundant weights while trying to retain the properties of the uncompressed baseline [49] [50]. There are two types of pruning: structured and unstructured pruning. Structured Pruning focuses on preserving the original structure of the network by only removing high granularity structures like rows and columns, connections, and hierarchical structures [51] [52] [53]. Unstructured Pruning can remove single weights, resulting in an irregular sparse structure [50] [54] [55]. The investigated literature can be viewed in table II.

TABLE II LITERATURE OVERVIEW FOR PRUNING

Pruning Literature		
Structured Pruning	Unstructured Pruning	
LoRaPrune [51] What Matters in Structured Pruning? [52]	SparseGPT [50] Prune and Tune [54]	
LLM-Pruner [53]	Wanda [55]	

Several studies are researching unstructured pruning. Frantar [50] proposed SparseGPT, which focuses on weight matrices and pruning individual weights. The approach aims to keep quality degradation in check by rephrasing the pruning problem as a sparse regression. This results in the pruning of current weights while adjusting not-yet-pruned weights. Prune and Tune [54] improves upon SparseGPT by introducing fine-tuning at each pruning step. The authors of [55] do not rely on retraining or iterative weight updates. This approach is similar to LLM.Int8() [30], by calculating the importance of weights based on their activations and subsequent pruning of unimportant weights.

In addition to structured pruning, several studies investigate the appliance of unstructured pruning. LoRAPrune aims to increase pruning efficiency by using LoRA mechanisms to estimate weight importance. [51]. Using structured pruning and LoRA mechanisms for importance estimation enables the pruned model to merge fine-tuned LoRa weights, which improves efficiency. The calculation of weight importance is also a major factor of [52] following a similar strategy to LLM.Int8() [30] and AWQ [27]. They propose identifying prunable weights by calculating two key figures: Sensitivity and Uniqueness. Sensitivity looks at a neuron's output and provides insights into its importance. Uniqueness provides insights into the variance, by telling how much a neuron differs from others. A low uniqueness indicates redundancy, and other neurons could reconstruct the output. Using these key figures in combination identifies prunable structures. LLM-Pruner [53] proposes a three-step process: (1) The discovery stage, where relevant structures and their dependencies get identified. (2) The estimation stage is where the identified structures get valued, and a decision is made on which structures are pruneable. (3) In the recovery stage, the error from pruning is mitigated via an efficient training step.

Challenges of pruning. Previously pruning required a retraining step to maintain the quality and characteristics of the uncompressed baseline [53] [56] [50]. However, recent research has attempted zero-shot pruning without significant retraining efforts [53] [55]. Identifying all relevant structures for pruning is challenging, resulting in either a falsely or overpruned worse model or missing potential optimizations [50] [53] [55].

Practical constraints of pruning. Unstructured and structured pruning of models with billions of parameters can be challenging to achieve efficiently due to the vast number of possible structures available for pruning. Finding suitable structures for pruning can be challenging and computationally intensive [50]. For instance, Frantar et al. [50] utilizes insights from GPTQ [35] to prune a GPT-style model with 175 billion parameters in approximately four hours. However, Sun et al. [55] improved the efficiency of the prune-metric to achieve faster compression, allowing for real-time pruning if needed. Unstructured pruning presents its own set of challenges. For example, pruned models are no longer compatible with LoRAweights due to the irregular structures that remain after compression [51]. This constraint could impede specific deployment scenarios where task-specific models are necessary and real-time pruning is applied.

It is essential to consider the constraints of using pruned models for inference. To achieve a speed-up, the hardware in the production environment has to support sparse models. LoRA-Prune [51] emphasizes that specific unique hardware configurations are needed for unstructured pruning to utilize the improvement. SpraseGPT [50] mention specific hardware support for structured pruning for their performance evaluation. Therefore, identifying if the targeted hardware supports models structured or unstructured pruned models is important.

In Conclusion.

- Pruning involves identifying and removing redundant weights while maintaining the properties of the uncompressed baseline.
- Two types of pruning: structured and unstructured pruning.
- Unstructured Pruning focuses on removing individual weights, resulting in an irregular sparse structure.
- Structured Pruning preserves the original structure of the network by removing high-granularity structures like rows, columns, and connections.
- Pruning requires retraining to maintain quality and characteristics, but recent research attempts zero-shot pruning without significant retraining.
- Identifying all relevant structures for pruning can be challenging, impacting the effectiveness of zero-shot pruning methods.
- Hardware constraints during inference, sparse model support, and pruning ratios significantly impact efficiency.

C. Knowledge Distillation

Knowledge Distillation describes a fine-tuning and compression technique that allows the transfer of knowledge of a large complex model into a smaller streamlined and efficient model [57] [58] [59] [60]. Transferring knowledge is done by training a smaller student model, with the outputs of a bigger teacher model [61]. There are two distinct approaches, *Black-Box Knowledge Distillation*, where only the outputs of the teacher model are available [62] [63] [64] [65] and *White-Box Knowledge Distillation* where also the teacher model parameters are utilized [61] [66] [67]. The detailed listing of all reviewed literature can be seen in table III

White-Box Knowledge Distillation. Utilizing the teacher model's internal workings to improve the student model's learning is called White-Box Knowledge Distillation. The core idea is to use the teacher's distribution and parameter settings to enable the student to learn more effectively. The student model's increased quality is achieved by reducing the *Kullback-Leibler Divergence* (KLD). [61] [66] [67]

The authors of MiniLLM [61] aim to minimize distribution differences between the teacher and student but point out that this can lead to drifts due to the vast output space of LLMs, which the student model may not be able to replicate. To mitigate this, they propose to optimize the inverse KLD, which encourages the student to learn probabilities that prioritize correctness. Agarwal et al. [66] investigate the distribution behavior of both models during distillation and result in two fundamental problems: (1) distribution mismatch between the teacher and the student, and (2) the student's lack of expressive power to imitate the teacher effectively. To achieve more precise imitation, they sampled the outputs of the student during training to improve learning. They also researched under-specification, where optimizing KDL was proposed. The authors of [67] criticize that the inherent principle of Knowledge Distillation poses a problem, as previous strategies focus on a specific task setting instead of transferring broad skills present in today's LLMs. To achieve task-agnostic distillation, they propose truncating the larger model and using its layers to initialize the smaller model. The smaller model is then distilled using the language modeling objective.

Black-box Knowledge Distillation. Black-Box Knowledge Distillation differs in that only a teacher's response is accessible, and utilizing parts of a teacher model is no longer possible [62] [63] [65] [73]. To effectively use Black-Box Knowledge Distillation, the learning process is augmented with recent research on emergent abilities, which enables the usage of methods like Chain-of-Thought [9] to improve quality. This results in using various prompting techniques to support the Distillation of the larger teacher model into smaller, more efficient one.

One of those prompting techniques is In-Context Learning, which uses a natural language prompt in combination with relevant descriptions and examples to enable a model to fulfill or solve a task. This technique distills smaller models to transfer in-context few-shot and language modeling abilities from the teacher model. There are two distilling strategies to transfer in-context learning: Meta In-Context Tuning (Meta-ICT) and Multitask In-context Tuning (Multitask-ICT) [62]. Meta-ICT trains the smaller model in the *Pre-Training* stage

with examples of in-context learning on diverse tasks. The result is a fine-tuned model that learns to identify different tasks and uses this ability to transfer this knowledge to unseen tasks during the *Task Adaption* stage. Multitaks-ICT, proposed by [62], instead of providing the adjustment in the Pre-Training stage, the adjustment happens during the Task-adaption stage. The results show that Multitask-ICT is more precise and has an advantage over Meta-ICT.

The following strategy for distilling uses Chain-of-Thought to help the student learn the abilities of the teacher model. Augmenting the prompt with step-by-step instructions on finding a solution is the distinction from in-context tuning, which only provides task descriptions [20]. The authors of [63] researched prompt augmenting and found that including instructions on how to infer a solution significantly helps the student at task adaption. The authors of [68] adjusted these insights for Knowledge Distillation and divided the fine-tuning phase into steps. Step one is augmenting existing instruction tuning datasets with teacher-generated Chain-of-Thought explanations. The student then learns via these augmented datasets by mimicking the teacher-generated explanations. Providing intermediate steps to find a solution and rationales for why a solution is correct further enhances the learning of the student model [69]. The authors further argue that providing more than one rationale helps the student model more effectively because complex problems can have multiple solutions. The quality of the student model is dependent on the quality of the used dataset for distilling. Not Evaluating abilities, not part of the dataset used for distilling, is recognized as a potential problem by [64]. They argue that focusing a student model on a specific set of tasks makes it a specialized model instead and incapable of generic reasoning. The authors of [70] shift the paradigm of distilling from using the question in combination with the rationale for learning to predict the answer with a fitting rationale. They leverage an unlabeled dataset and generate labels with rationales as justifications. Labels and rationales are then used to distill smaller models. Dividing the distillation problem into understanding problem structures and providing solution steps for these sub-problems, is proposed by [71]. This division results in training two models, where the first model learns to decompose the problem into smaller sub-problems and annotates the prompt with sub-questions. The second model learns to use these subquestions to generate a higher-quality answer. Using Chain-ofthought in combination with rational generation is identified as problematic by [72]. There are two fundamental problems: LLMs are prone to hallucination and could provide false rationals, which the student learns. Also, the student may take shortcuts and generate an answer independently from the provided rationale. To combat these problems, they proposed contrastive decoding to ground an answer to the rational. This decoding process means the student has to use counterfactual reasoning, which results in it being more truthful. The authors of [73] also focus on truthfulness by generating rationals in the form of small Python programs. These are then automatically evaluated, and faulty reasoning is more easily discovered.

TABLE III
LITERATURE OVERVIEW FOR KNOWLEDGE DISTILLATION

Knowledge Distillation Literature		
White-Box KD	Black-Box KD	
MiniLLM [61]	In-context Learning Distillation [62]	
On-Policy Distillation of LMs [66]	Explanations from LLMs Make Small Models Better [68]	
How To Train Your (Compressed) LLM [67]	LLMs Are Reasoning Teachers [69] Specializing Smaller Models [64] Distilling Step-by-Step! [70]	
	Distilling Reasoning Capabilities Into Smaller Models [71] SCOTT [72] PaD: Program-aided Distillation [73]	
	Can LMs Teach Weaker Agents? [74] Lion: Adversarial Distillation [75] LaMini-LM [65]	

Another approach to increase the quality is to provide feedback to the student [74]. The teacher model uses the generations of the student and decides to intervene when the student is struggling. The intervention is implemented with a Theory of Mind approach. This means the teacher creates a small mental model of the student for the intervention decision.

Another strategy for distilling would be to use only detailed descriptions of instructions without examples or descriptions of intermediate steps. To enable the smaller model to solve unseen tasks, it gets trained on a large number of different instructions [76] [77]. Lion [75] divides the distilling process into three steps. In the Imitation Stage, the student responses are aligned to the teacher's. The teacher then identifies false answers to gather complicated instructions in the subsequent Discrimination Stage. The final, Generation Stage uses gathered instructions and enriches them with more details to help the student learn more effectively. La-Mini-LM [65] investigated the current instruction following distillation strategies. They found a lack of available small-scale distilling datasets, and available datasets need to provide more diversity to produce a high-quality model. Moreover, student evaluation focuses too narrowly on one model family, and the evaluation process needs to be more precise. To improve dataset diversity, a new distilling dataset is proposed. Furthermore, trained models on this dataset are thoroughly evaluated using a comprehensive testing methodology.

Challenges. There are significant challenges that impact distillation quality. In order to achieve more students with the teacher's abilities, the reduction of the distribution mismatch is actively investigated [61] [66]. The difficulty in reduction could stem from the lack of publicly available diverse distilling datasets [65]. This results in either manually creating or

augmenting existing datasets. However, the diversity and detail of current datasets could also be a challenge in distilling capable students [65]. Also, reaching specific quality aspects for the student is challenging. The authors of [66] found that it is challenging distilling a student that has similar distribution and expressive power to the teacher. One possible reason investigated [78], which criticized that the student can only be as good as the used teacher. That means that the student learns hallucinations, bias, and toxicity behavior already present in the teacher. If only Black-Box Knowledge Distillation is available, the risk of only imitating the teacher's style without actually learning the behavior is present [78].

Practical Constraints. In many knowledge distillation strategies, a student-teacher setup is deployed. This results in two models loaded into memory during white-box and local black-box distillation, which requires a memory-potent environment [61] [66] [67]. Running two models is costly because computing and memory are expensive. These costs also apply to APIs, where memory costs are factored into API usage costs. Therefore, it has to be considered if fine-tuning is needed given the resource requirements and if it is preferred over simple optimization techniques like quantization.

In Conclusion.

- Knowledge distillation is a technique for fine-tuning and compressing models, with the goal of transferring knowledge from larger models to smaller ones.
- This entails training a smaller "student" model with the outputs of a larger "teacher" model.
- Two approaches, black-box and white-box knowledge distillation, are used.
- White-box knowledge distillation enables the student to utilize the teacher's parameters for a better understanding.

- Black-box knowledge distillation relies solely on the teacher's prediction.
- The standard student-teacher configuration in a white-box setting necessitates both models to operate in memory, which can be computationally demanding.
- Distributional imbalances, lack of diverse data sets, and imitation of undesirable teacher behaviors are among the quality challenges in knowledge distillation.
- Challenges also arise from the nature of black-box knowledge distillation, where students may mimic the teacher's style without understanding the desired behavior.

D. Architectural Optimization

The transformer performs well because of the attention mechanism. The computation of this mechanism requires much memory because of the key-value cache. During generations, predicting the next token requires the caching of all previous tokens in order to calculate the joint probabilities [12] [79]. Therefore, one big focus is making the decoding process or the attention mechanism more efficient [80]. The following discusses popular optimization strategies for optimizing the decoding phase.

Paged Attention. The authors of vLLM [79] focused on making the key-value cache more memory efficient. This optimization is achieved by transferring a memory-storing concept of operating systems and applying it to the attention mechanism. Here, the key-value cache is divided into blocks allocated efficiently in memory. The allocation is done by placing the blocks in non-contiguous parts of physical memory, which is more efficient.

Windowed Attention. Windowed attention, proposed by [80], tries to mitigate huge computational requirements with large sequence transformers. The memory requirements scale quadratically with sequence length due to the key-value cache [80]. Windowed Attention uses sliding windows to focus attention more locally, reducing complexity. This focus is beneficial for masked language modeling, where only a local focus is needed, but it is counterproductive for classification, which needs a broader focus. For mitigation, the authors propose sliding windows with an augmented global window, which helps to broaden the context.

Attention Sinks. There are challenges when using sliding window attention. The model could collapse when the sliding window exceeds the cache size, and costly recomputation has to be performed [81]. The authors found that the first tokens get significant attention scores. Therefore, they concluded that instead of discarding the first tokens after the sliding windows further progress, a couple of the first tokens are kept. This results in quality and performance improvements, as stated in [81].

Flash Attention. Not only is the complexity of the attention algorithm a potential bottleneck, but the device data management to fill it with data could also be problematic [82]. The goal of Flash-Attention [82] is to minimize the need for data movement while computing the attention matrices. They achieve this by regrouping operations from different parts

of the LLM and keeping relevant data in memory to reduce memory loading overhead.

Speculative Decoding. The decoding phase of a decoderonly transformer is costly because of the sequential properties of generation [79] [82] [83]. Instead of optimizing attention, Speculative Decoding [83] focuses on making the decoding process more efficient. It operates under the principle of performing generations in parallel while concurrently verifying their necessity. The concurrency is achieved by using two models: one primary model and one support model. The support model provides token suggestions that the primary model accepts or rejects. Speculative decoding significantly reduces inference time and resource consumption by maximizing the probability of acceptance for these speculative tasks while ensuring their outputs maintain the same distribution as those from the target model alone. The authors mention the effectiveness of speculative decoding across various tasks and model sizes, including unconditional generation, translation, summarization, and dialogue tasks, resulting in a notable 2x-3x latency improvement without impacting output quality.

```
| START| | japan | | 2 | benchmark nikkel | 22 | 15| | | |
| START| | japan | | 2 | benchmark nikkel | 22 | 15|
| START| | japan | | 2 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | | 2 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | | 2 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | | 2 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | | 2 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | | 2 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | benchmark nikkel | 225 | 100x | rose | 22 | 10|
| START| | japan | 3 | 10|
|
```

Fig. 5. Example generation out of [83], where green are accepted generations, red and blue are rejections and corrections, respectively.

In Conclusion.

- The transformers' attention mechanism requires significant computational resources.
- To optimize the key-value cache, it can be divided into blocks that are stored in non-contiguous parts of physical memory.
- As the length of a transformer sequence increases, the complexity of attention increases exponentially.
- Sliding windows and attention sinks can help reduce this complexity.
- Flash Attention minimized expensive data movement by regrouping operations and keeping relevant data in memory.
- Speculative Decoding samples generations from a more efficient model and achieves its speed up by reducing the needed runs from the larger model.

IV. CONCLUSION

This literature review explored multiple optimization techniques, their challenges, and practical applicability. It aims to provide insights into the current state of research and its practical considerations. The explored methods included quantization, pruning, knowledge distillation, and optimizations of the attention mechanism and decoding process. There are different grades of optimization development research and applicability. This grading makes categorizing these techniques into their

current state of research and applicability possible. Compared to quantization, Pruning methods have to develop more precise methods of finding all available structures for compression. In contrast, quantization literature thoroughly discusses the implications of applying it to different LLM elements. The advancement in research leads to quantization, which mainly focuses on weights and tries to push compression via bit width. Knowledge distillation provides the most flexibility for affected abilities by choosing the distilling dataset and the prompting technique. It needs more training, which makes setup and resource requirements costly. In conclusion, the chosen optimization technique is dependent on the optimization goals, available resources, and target environment.

REFERENCES

- [1] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, Q. Liu, E. Zheltonozhskii, T. Y. Zhuo, T. Wang, O. Dehaene, M. Davaadorj, J. Lamy-Poirier, J. Monteiro, O. Shliazhko, N. Gontier, N. Meade, A. Zebaze, M.-H. Yee, L. K. Umapathi, J. Zhu, B. Lipkin, M. Oblokulov, Z. Wang, R. Murthy, J. Stillerman, S. S. Patel, D. Abulkhanov, M. Zocca, M. Dey, Z. Zhang, N. Fahmy, U. Bhattacharyya, W. Yu, S. Singh, S. Luccioni, P. Villegas, M. Kunakov, F. Zhdanov, M. Romero, T. Lee, N. Timor, J. Ding, C. Schlesinger, H. Schoelkopf, J. Ebert, T. Dao, M. Mishra, A. Gu, J. Robinson, C. J. Anderson, B. Dolan-Gavitt, D. Contractor, S. Reddy, D. Fried, D. Bahdanau, Y. Jernite, C. M. Ferrandis, S. M. Hughes, T. Wolf, A. Guha, L. von Werra, and H. de Vries, "Starcoder: may the source be with you!" ArXiv, vol. abs/2305.06161, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258588247
- [2] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. Ponde, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. W. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, I. Babuschkin, S. Balaji, S. Jain, A. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating large language models trained on code," ArXiv, vol. abs/2107.03374, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235755472
- [3] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. hsin Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," *ArXiv*, vol. abs/2206.07682, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID: 249674500
- [4] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. rahman Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Annual Meeting of the Association for Computational Linguistics*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:204960716
- [5] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *ArXiv*, vol. abs/2005.11401, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:218869575
- [6] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, Q. Guo, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *CoRR*, vol. abs/2312.10997, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2312.10997
- [7] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," ArXiv, vol. abs/2005.14165, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:218971783

- [8] E. Beeching, C. Fourrier, N. Habib, S. Han, N. Lambert, N. Rajani, O. Sanseviero, L. Tunstall, and T. Wolf, "Open Ilm leaderboard," https://huggingface.co/spaces/HuggingFaceH4/open_Ilm_leaderboard, 2023.
- [9] Y. Fu, L. Ou, M. Chen, Y. Wan, H.-C. Peng, and T. Khot, "Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance," *ArXiv*, vol. abs/2305.17306, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258959433
- [10] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:160025533
- [11] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Neural Information Processing Systems*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:13756489
- [12] S. Kim, C. Hooper, T. Wattanawong, M. Kang, R. Yan, H. Genç, G. Dinh, Q. Huang, K. Keutzer, M. W. Mahoney, Y. S. Shao, and A. Gholami, "Full stack optimization of transformer inference: a survey," *ArXiv*, vol. abs/2302.14017, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257219934
- [13] 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*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID: 52967399
- [14] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," 2019.
- [15] S. Guskin, M. Wasserblat, K. Ding, and G. Kim, "Dynamic-tinybert: Boost tinybert's inference efficiency by dynamic sequence length," 2021. [Online]. Available: https://arxiv.org/abs/2111.09645
- [16] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *ArXiv*, vol. abs/1907.11692, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:198953378
- [17] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," in *Conference on Empirical Methods in Natural Language Processing*, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:11816014
- [18] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," ArXiv, vol. abs/1910.01108, 2019.
- [19] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of* the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. [Online]. Available: https: //www.aclweb.org/anthology/D13-1170
- [20] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. hsin Chi, F. Xia, Q. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," *ArXiv*, vol. abs/2201.11903, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:246411621
- [21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," ArXiv, vol. abs/1912.01703, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID: 202786778
- [22] S. Chetlur, C. Woolley, P. Vandermersch, J. M. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," ArXiv, vol. abs/1410.0759, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:12330432
- [23] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zhang, "This paper is included in the proceedings of the 12th usenix symposium on operating systems design and implementation (osdi '16). tensorflow: a system for large-scale machine learning tensorflow: a system for large-scale machine learning." [Online]. Available: https://api.semanticscholar.org/CorpusID:6287870
- [24] "Ieee standard for floating-point arithmetic," IEEE Std 754-2019 (Revision of IEEE 754-2008), pp. 1–84, 2019.

- [25] Y. Lin, Y. Li, T. Liu, T. Xiao, T. Liu, and J. Zhu, "Towards fully 8-bit integer inference for the transformer model," arXiv preprint arXiv:2009.08034, 2020.
- [26] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," Advances in neural information processing systems, vol. 31, 2018.
- [27] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, and S. Han, "Awq: Activation-aware weight quantization for llm compression and acceleration," arXiv preprint arXiv:2306.00978, 2023.
- [28] P. Kharya. NVIDIA Blogs: TensorFloat-32 Accelerates AI Training HPC upto 20x. NVIDIA Blog. [Online]. Available: https://blogs.nvidia. com/blog/tensorfloat-32-precision-format/
- [29] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Q-bert: Hessian based ultra low precision quantization of bert," in *Proceedings of the AAAI Conference on Artificial Intelli*gence, vol. 34, no. 05, 2020, pp. 8815–8821.
- [30] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm.int8(): 8-bit matrix multiplication for transformers at scale," ArXiv, vol. abs/2208.07339, 2022. [Online]. Available: https://api.semanticscholar. org/CorpusID:251564521
- [31] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8bert: Quantized 8bit bert," in 2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS). IEEE, 2019, pp. 36–39.
- [32] Z. Jia, M. Maggioni, J. Smith, and D. P. Scarpazza, "Dissecting the nvidia turing t4 gpu via microbenchmarking," arXiv preprint arXiv:1903.07486, 2019.
- [33] A. Kuzmin, M. Van Baalen, Y. Ren, M. Nagel, J. Peters, and T. Blankevoort, "Fp8 quantization: The power of the exponent," Advances in Neural Information Processing Systems, vol. 35, pp. 14651–14662, 2022.
- [34] X. Wu, Z. Yao, and Y. He, "Zeroquant-fp: A leap forward in llms post-training w4a8 quantization using floating-point formats," arXiv preprint arXiv:2307.09782, 2023.
- [35] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," arXiv preprint arXiv:2210.17323, 2022.
- [36] Quantization Neural Network Distiller. [Online]. Available: https://intellabs.github.io/distiller/algo_quantization.html
- [37] Z. Yao, R. Y. Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, "Zeroquant: Efficient and affordable post-training quantization for large-scale transformers," ArXiv, vol. abs/2206.01861, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:249395624
- [38] Z. Yao, X. Wu, C. Li, S. Youn, and Y. He, "Zeroquant-v2: Exploring post-training quantization in llms from comprehensive study to low rank compensation," 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258947018
- [39] C. Lee, J. Jin, T. Kim, H. Kim, and E. Park, "Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models," 2023. [Online]. Available: https://api.semanticscholar. org/CorpusID:267095435
- [40] W. Shao, M. Chen, Z. Zhang, P. Xu, L. Zhao, Z. Li, K. Zhang, P. Gao, Y. J. Qiao, and P. Luo, "Omniquant: Omnidirectionally calibrated quantization for large language models," ArXiv, vol. abs/2308.13137, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID: 261214575
- [41] T. Dettmers, R. Svirschevski, V. Egiazarian, D. Kuznedelev, E. Frantar, S. Ashkboos, A. Borzunov, T. Hoefler, and D. Alistarh, "Spqr: A sparsequantized representation for near-lossless llm weight compression," *ArXiv*, vol. abs/2306.03078, 2023. [Online]. Available: https://api. semanticscholar.org/CorpusID:259076379
- [42] G. Xiao, J. Lin, M. Seznec, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," *ArXiv*, vol. abs/2211.10438, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:253708271
- [43] T. Dettmers and L. Zettlemoyer, "The case for 4-bit precision: k-bit inference scaling laws," in *International Conference on Machine Learning*, 2022. [Online]. Available: https://api.semanticscholar.org/ CorpusID:254853733
- [44] T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ili'c, D. Hesslow, R. Castagn'e, A. S. Luccioni, F. Yvon, M. Gallé, J. Tow, A. M. Rush, S. Biderman, A. Webson, P. S. Ammanamanchi, T. Wang, B. Sagot, N. Muennighoff, A. V. del Moral, O. Ruwase, R. Bawden, S. Bekman, A. McMillan-Major, I. Beltagy, H. Nguyen, L. Saulnier, S. Tan, P. O.
- Suarez, V. Sanh, H. Laurenccon, Y. Jernite, J. Launay, M. Mitchell, C. Raffel, A. Gokaslan, A. Simhi, A. S. Etxabe, A. F. Aji, A. Alfassy, A. Rogers, A. K. Nitzav, C. Xu, C. Mou, C. C. Emezue, C. Klamm, C. Leong, D. A. van Strien, D. I. Adelani, D. R. Radev, E. G. Ponferrada, E. Levkovizh, E. Kim, E. Natan, F. D. Toni, G. Dupont, G. Kruszewski, G. Pistilli, H. ElSahar, H. Benyamina, H. T. Tran, I. Yu, I. Abdulmumin, I. Johnson, I. Gonzalez-Dios, J. de la Rosa, J. Chim, J. Dodge, J. Zhu, J. Chang, J. Frohberg, J. L. Tobing, J. Bhattacharjee, K. Almubarak, K. Chen, K. Lo, L. von Werra, L. Weber, L. Phan, L. B. Allal, L. Tanguy, M. Dey, M. R. Muñoz, M. Masoud, M. Grandury, M. vSavsko, M. Huang, M. Coavoux, M. Singh, M. T.-J. Jiang, M. C. Vu, M. A. Jauhar, M. Ghaleb, N. Subramani, N. Kassner, N. Khamis, O. Nguyen, O. Espejel, O. de Gibert, P. Villegas, P. Henderson, P. Colombo, P. Amuok, Q. Lhoest, R. Harliman, R. Bommasani, R. L'opez, R. Ribeiro, S. Osei, S. Pyysalo, S. Nagel, S. Bose, S. H. Muhammad, S. Sharma, S. Longpre, S. maieh Nikpoor, S. Silberberg, S. Pai, S. Zink, T. T. Torrent, T. Schick, T. Thrush, V. Danchev, V. Nikoulina, V. Laippala, V. Lepercq, V. Prabhu, Z. Alyafeai, Z. Talat, A. Raja, B. Heinzerling, C. Si, E. Salesky, S. J. Mielke, W. Y. Lee, A. Sharma, A. Santilli, A. Chaffin, A. Stiegler, D. Datta, E. Szczechla, G. Chhablani, H. Wang, H. Pandey, H. Strobelt, J. A. Fries, J. Rozen, L. Gao, L. Sutawika, M. S. Bari, M. S. Al-Shaibani, M. Manica, N. V. Nayak, R. Teehan, S. Albanie, S. Shen, S. Ben-David, S. H. Bach, T. Kim, T. Bers, T. Févry, T. Neeraj, U. Thakker, V. Raunak, X. Tang, Z.-X. Yong, Z. Sun, S. Brody, Y. Uri, H. Tojarieh, A. Roberts, H. W. Chung, J. Tae, J. Phang, O. Press, C. Li, D. Narayanan, H. Bourfoune, J. Casper, J. Rasley, M. Ryabinin, M. Mishra, M. Zhang, M. Shoeybi, M. Peyrounette, N. Patry, N. Tazi, O. Sanseviero, P. von Platen, P. Cornette, P. F. Lavall'ee, R. Lacroix, S. Rajbhandari, S. Gandhi, S. Smith, S. Requena, S. Patil, T. Dettmers, A. Baruwa, A. Singh, A. Cheveleva, A.-L. Ligozat, A. Subramonian, A. N'ev'eol, C. Lovering, D. H. Garrette, D. R. Tunuguntla, E. Reiter, E. Taktasheva, E. Voloshina, E. Bogdanov, G. I. Winata, H. Schoelkopf, J.-C. Kalo, J. Novikova, J. Z. Forde, X. Tang, J. Kasai, K. Kawamura, L. Hazan, M. Carpuat, M. Clinciu, N. Kim, N. Cheng, O. Serikov, O. Antverg, O. van der Wal, R. Zhang, R. Zhang, S. Gehrmann, S. Mirkin, S. O. Pais, T. Shavrina, T. Scialom, T. Yun, T. Limisiewicz, V. Rieser, V. Protasov, V. Mikhailov, Y. Pruksachatkun, Y. Belinkov, Z. Bamberger, Z. Kasner, Z. Kasner, A. Pestana, A. Feizpour, A. Khan, A. Faranak, A. S. R. Santos, A. Hevia, A. Unldreaj, A. Aghagol, A. Abdollahi, A. Tammour, A. HajiHosseini, B. Behroozi, B. A. Ajibade, B. K. Saxena, C. M. Ferrandis, D. Contractor, D. M. Lansky, D. David, D. Kiela, D. A. Nguyen, E. Tan, E. Baylor, E. Ozoani, F. T. Mirza, F. Ononiwu, H. Rezanejad, H. Jones, I. Bhattacharya, I. Solaiman, I. Sedenko, I. Nejadgholi, J. Passmore, J. Seltzer, J. B. Sanz, K. Fort, L. Dutra, M. Samagaio, M. Elbadri, M. Mieskes, M. Gerchick, M. Akinlolu, M. McKenna, M. Qiu, M. Ghauri, M. Burynok, N. Abrar, N. Rajani, N. Elkott, N. Fahmy, O. Samuel, R. An, R. P. Kromann, R. Hao, S. Alizadeh, S. Shubber, S. L. Wang, S. Roy, S. Viguier, T.-C. Le, T. Oyebade, T. N. H. Le, Y. Yang, Z. K. Nguyen, A. R. Kashyap, A. Palasciano, A. Callahan, A. Shukla, A. Miranda-Escalada, A. K. Singh, B. Beilharz, B. Wang, C. M. F. de Brito, C. Zhou, C. Jain, C. Xu, C. Fourrier, D. L. Perin'an, D. Molano, D. Yu, E. Manjavacas, F. Barth, F. Fuhrimann, G. Altay, G. Bayrak, G. Burns, H. U. Vrabec, I. I. Bello, I. Dash, J. S. Kang, J. Giorgi, J. Golde, J. D. Posada, K. Sivaraman, L. Bulchandani, L. Liu, L. Shinzato, M. H. de Bykhovetz, M. Takeuchi, M. Pàmies, M. A. Castillo, M. Nezhurina, M. Sanger, M. Samwald, M. Cullan, M. Weinberg, M. Wolf, M. Mihaljcic, M. Liu, M. Freidank, M. Kang, N. Seelam, N. Dahlberg, N. M. Broad, N. Muellner, P. Fung, P. Haller, R. Chandrasekhar, R. Eisenberg, R. Martin, R. L. Canalli, R. Su, R. Su, S. Cahyawijaya, S. Garda, S. S. Deshmukh, S. Mishra, S. Kiblawi, S. Ott, S. Sang-aroonsiri, S. Kumar, S. Schweter, S. P. Bharati, T. Laud, T. Gigant, T. Kainuma, W. Kusa, Y. Labrak, Y. Bajaj, Y. Venkatraman, Y. Xu, Y. Xu, Y. Xu, Z. X. Tan, Z. Xie, Z. Ye, M. Bras, Y. Belkada, and T. Wolf, "Bloom: A 176b-parameter open-access multilingual language model," ArXiv, vol. abs/2211.05100, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:253420279
- [45] J. Kim, J. H. Lee, S. Kim, J. Park, K. M. Yoo, S. J. Kwon, and D. Lee, "Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization," *ArXiv*, vol. abs/2305.14152, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID: 258841104
- [46] H. Touvron, L. Martin, K. R. Stone, P. Albert, A. Almahairi, Y. Babaei,

- N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. M. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. S. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. M. Kloumann, A. V. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," *ArXiv*, vol. abs/2307.09288, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259950998
- [47] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. P. Bhatt, C. C. Ferrer, A. Grattaflori, W. Xiong, A. D'efossez, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, and G. Synnaeve, "Code llama: Open foundation models for code," ArXiv, vol. abs/2308.12950, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:261100919
- [48] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," *ArXiv*, vol. abs/2004.09602, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:216035831
- [49] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Neural Information Processing Systems*, 1989. [Online]. Available: https://api.semanticscholar.org/CorpusID:7785881
- [50] E. Frantar and D. Alistarh, "Sparsegpt: Massive language models can be accurately pruned in one-shot," ArXiv, vol. abs/2301.00774, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID: 255372747
- [51] M. Zhang, H. Chen, C. Shen, Z. Yang, L. Ou, X. Yu, and B. Zhuang, "Loraprune: Pruning meets low-rank parameter-efficient fine-tuning," 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID: 258967906
- [52] M. Santacroce, Z. Wen, Y. Shen, and Y.-F. Li, "What matters in the structured pruning of generative language models?" ArXiv, vol. abs/2302.03773, 2023. [Online]. Available: https://api.semanticscholar. org/CorpusID:256662734
- [53] X. Ma, G. Fang, and X. Wang, "Llm-pruner: On the structural pruning of large language models," ArXiv, vol. abs/2305.11627, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258823276
- [54] A. Syed, P. H. Guo, and V. Sundarapandiyan, "Prune and tune: Improving efficient pruning techniques for massive language models," 2023. [Online]. Available: https://openreview.net/forum?id=cKlgcx7nSZ
- [55] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A simple and effective pruning approach for large language models," ArXiv, vol. abs/2306.11695, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID: 259203115
- [56] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," ArXiv, vol. abs/2305.14314, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID: 258841328
- [57] B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang, "Decoupled knowledge distillation," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11943–11952, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247476179
- [58] L. Breiman and N. Shang, "Born again trees," 1996. [Online]. Available: https://api.semanticscholar.org/CorpusID:2145744
- [59] C. Bucila, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Knowledge Discovery and Data Mining*, 2006. [Online]. Available: https://api.semanticscholar.org/CorpusID:11253972
- [60] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in Neural Information Processing Systems, 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:11536917
- [61] Y. Gu, L. Dong, F. Wei, and M. Huang, "Knowledge distillation of large language models," ArXiv, vol. abs/2306.08543, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259164722
- [62] Y. Huang, Y. Chen, Z. Yu, and K. McKeown, "In-context learning distillation: Transferring few-shot learning ability of pre-trained language models," *ArXiv*, vol. abs/2212.10670, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:254926556
- [63] S. LI, J. Chen, Y. Shen, Z. Chen, X. Zhang, Z. Li, H. Wang, J. Qian, B. Peng, Y. Mao, W. Chen, and X. Yan, "Explanations

- from large language models make small reasoners better," *ArXiv*, vol. abs/2210.06726, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252873123
- [64] Y. Fu, H.-C. Peng, L. Ou, A. Sabharwal, and T. Khot, "Specializing smaller language models towards multi-step reasoning," ArXiv, vol. abs/2301.12726, 2023. [Online]. Available: https://api.semanticscholar. org/CorpusID:256390607
- [65] M. Wu, A. Waheed, C. Zhang, M. Abdul-Mageed, and A. F. Aji, "Lamini-Im: A diverse herd of distilled models from large-scale instructions," ArXiv, vol. abs/2304.14402, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258352678
- [66] R. Agarwal, N. Vieillard, Y. Zhou, P. Stańczyk, S. Ramos, M. Geist, and O. Bachem, "On-policy distillation of language models: Learning from self-generated mistakes," 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:263610088
- [67] A. Jha, D. Groeneveld, E. Strubell, and I. Beltagy, "How to train your (compressed) large language model," 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258865382
- [68] L. C. Magister, J. Mallinson, J. Adamek, E. Malmi, and A. Severyn, "Teaching small language models to reason," ArXiv, vol. abs/2212.08410, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:254823156
- [69] N. Ho, L. Schmid, and S.-Y. Yun, "Large language models are reasoning teachers," in *Annual Meeting of the Association* for Computational Linguistics, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:254877399
- [70] C.-Y. Hsieh, C.-L. Li, C.-K. Yeh, H. Nakhost, Y. Fujii, A. J. Ratner, R. Krishna, C.-Y. Lee, and T. Pfister, "Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes," *ArXiv*, vol. abs/2305.02301, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258461606
- [71] K. Shridhar, A. Stolfo, and M. Sachan, "Distilling reasoning capabilities into smaller language models," in *Annual Meeting of the Association for Computational Linguistics*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:258762841
- [72] P. Wang, Z. Wang, Z. Li, Y. Gao, B. Yin, and X. Ren, "Scott: Self-consistent chain-of-thought distillation," in *Annual Meeting of the Association for Computational Linguistics*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258461058
- [73] X. Zhu, B. Qi, K. Zhang, X. Long, and B. Zhou, "Pad: Program-aided distillation can teach small models reasoning better than chain-of-thought fine-tuning," 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258840866
- [74] S. Saha, P. Hase, and M. Bansal, "Can language models teach weaker agents? teacher explanations improve students via personalization," 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID: 265157967
- [75] Y. Jiang, C. Chan, M. Chen, and W. Wang, "Lion: Adversarial distillation of proprietary large language models," in *Conference on Empirical Methods in Natural Language Processing*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258833333
- [76] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. E. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. J. Lowe, "Training language models to follow instructions with human feedback," ArXiv, vol. abs/2203.02155, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID: 246426909
- [77] T. Brooks, A. Holynski, and A. A. Efros, "Instructpix2pix: Learning to follow image editing instructions," 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 18392–18402, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID: 253581213
- [78] A. Gudibande, E. Wallace, C. B. Snell, X. Geng, H. Liu, P. Abbeel, S. Levine, and D. Song, "The false promise of imitating proprietary llms," *ArXiv*, vol. abs/2305.15717, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258887629
- [79] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:261697361

- [80] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," ArXiv, vol. abs/2004.05150, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:215737171
- [81] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis, "Efficient streaming language models with attention sinks," ArXiv, vol. abs/2309.17453, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID: 263310483
- [82] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. R'e, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *ArXiv*, vol. abs/2205.14135, 2022. [Online]. Available: https://api.semanticscholar. org/CorpusID:249151871
- [83] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *International Conference on Machine Learning*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:254096365