A BRIEF SURVEY OF MODEL COMPRESSION IN LANGUAGE MODELS

BRANDON DONG (BJDONG), RUSSELL EMERINE (REMERINE), ARESH POURKAVOOS (APOURKAV), AND HYUNWOO PARK (HP2)

Abstract. Large Language Models are the state of the art for many natural language processing tasks, and their versatility makes them appealing in a variety of applications. However, such models have serious storage and computational demands, and the field of model compression works to lower those requirements. While model compression, especially for deep neural networks, has existed as a field before the advent of LLMs, the size and scope of these models present unique challenges to the pre-existing methods. Promising solutions include new techniques in knowledge distillation, quantization, pruning, and low-rank factorization that can generate significantly smaller models while remaining performant. In this report we provide a detailed review and some empirical verification of these methods, compare their relative advantages and disadvantages, and discuss the state of the field today. For brevity, we will not discuss inference acceleration, efficient prompting, or hardware methods; we focus instead on techniques that concern themselves with memory footprint.

Introduction

Large language models (LLMs) like GPT-4 are exceptional at a wide variety of natural language tasks, and beyond a certain size they can exhibit powerful and surprising emergent abilities. However, their storage and computational requirements make them challenging to train and use, and on platforms with fewer resources, such as edge devices, or with more stringent latency requirements, these constraints can be prohibitive. For instance, the GPT-175B model (Brown et al. 2020) requires at least 320GB of storage, as well as at least 5 A100 GPUs for inference. Even smaller LLMs can have hundreds of millions to billions of parameters (Touvron et al. 2023; Dey et al. 2023), while today's frontier models can exceed 500 billion.

Naturally, research has been focused on decreasing the latency, size, and computational requirements of LLMs as much as possible while maintaining performance. There are several active areas, including knowledge distillation, quantization, low rank factorization, and structured and unstructured parameter pruning (Cheng et al. 2020). While each of these subfields has made progress in the compression of other deep neural network architectures, LLMs in particular suffer from challenges unaddressed by earlier literature on model compression, due to both their sheer size and their varied functionality (Ma, Fang, and X. Wang 2023). In this review we'll discuss these challenges and examine possible solutions. Such solutions can be broadly classified as follows:

Knowledge distillation is a model compression technique whereby a single "student" model trains against one or more large "teacher" models. For classifiers, the student may learn the teacher model's class distribution for each training example instead of the ground truth label (Hinton, Vinyals, and Dean 2015). In LLMs, however, the student can train on not only the labels from the teacher model, but also explanations given by the teacher for each answer (S. Sun et al. 2019). One unique advantage of knowledge distillation is that it is *architecture agnostic*, i.e. the student and teacher architecture do not need to match (Tang et al. 2019). Compared to other model compression techniques, this provides the opportunity to train a shallow neural architecture with

CMU F23 10-701 Final Project Report

task-specific knowledge more effectively than from scratch. Often, however, student models trained via KD struggle with out-of-distribution samples; in particular, the unique versatility of LLMs is often lost in the process of knowledge distillation, in favor of maintaining performance on specific tasks. Furthermore, the training process can often be data hungry (Hsieh et al. 2023) and even unlabeled datasets in natural language processing can be hard to obtain (Tang et al. 2019). For these reasons, we may look to other methods of LLM compression.

Quantization attempts to compress a model by storing each parameter in a type with fewer bits, such as a less precise floating point number, an integer, or another custom type. This compression can maintain performance while reducing the model's storage footprint and accelerating inference (Dettmers et al. 2022). There are two major forms of quantization: **post-training quantization** (PTQ), where quantization is applied to the weights after training, and **quantization-aware training** (QAT), where the model is fine-tuned with quantization in mind before applying PTQ. For example, the model might be fine-tuned with its weights clamped to 8-bit or 16-bit representations (Gholami et al. 2021; Zhu et al. 2023). QAT often achieves better results than PTQ alone, as it makes the model more resilient to quantization before it is applied.

Low-rank factorization is a method of model compression in which weight matrices are decomposed into a product of two smaller matrices, whose shared dimension is typically the rank of the original matrix, using singular value decomposition (SVD) or similar. While there is a relative dearth of active research solely focused on improving low-rank factorization for LLMs, it remains an interesting technique due to its potential in combination with other approaches that can mitigate its drawbacks.

Structured and unstructured pruning selectively remove components of a neural network, setting the associated parameters to zero for faster inference. In **structured pruning**, entire components of the network are removed, such as neurons, channels, or layers. On the other hand, **unstructured pruning** removes individual weights, resulting in an irregular structure less amenable to manipulation or compression. Pruned models may require significant retraining to regain accuracy (Zhu et al. 2023), which can be extensive for LLMs, both in computational power and in training data. Methods that approach pruning LLMs with the intent of avoiding the retraining step are some of the most promising new works we'll examine in this review.

Methods

Each section below covers a different major area of active research in model compression, with an eye towards their applications to LLMs in particular. We will discuss knowledge distillation, quantization, pruning, and low rank factorization, tracing their development from the earliest advances in each method to the present day.

Knowledge Distillation (KD) is a compression technique that can be applied to any neural network, and in particular is an intuitive approach to language modeling. Instead of directly building a smaller model from a large model, we can train a small "student" model against a large "teacher" model. LLMs are a special subject of study in the literature, due to their unique versatility across different NLP tasks. Furthermore, some very large advanced LLMs exhibit powerful Emergent Abilities, including In-Context Learning (ICL), Chain of Thought (CoT), and Instruction Following (IF), that are highly desirable in smaller models. Thanks to ICL and IF, LLMs can perform reasonably well on specialized tasks, e.g. natural language inference, common sense question answering, or arithmetic word problems. This can be used as a noisy generator of training data for a student model specialized for one task. CoT adds a teacher model's explanation for an output (by literally

asking the LLM for an explanation). These applications will be covered in the following sections; KD has not always been applied in this context.

The first work to do KD was for general supervised classification tasks in Bucila, Caruana, and Niculescu-Mizil 2006, where the authors use an ensemble of teacher models to label a large unlabelled dataset before training the student neural network to match the output of the ensemble. The key premise of the idea is that since neural networks are universal approximators, they can use the shallow neural network to mimic the superior performance of the ensemble by simply learning its behavior instead. However, the seminal work in knowledge distillation is Hinton, Vinyals, and Dean 2015, which revived this idea and improved upon it in several key ways. They refine the technique by introducing a more general technique - and the first instance of calling it "distillation" - through raising the temperature of the final softmax for the teacher model to put more relative probability into the targets with smaller logits, generating a softer distribution, and using this same higher temperature when training the student model. This cleverly solves the problem of the probabilities for unlikely labels having very low absolute values, despite their relative values encoding valuable information about similarities over the data, and therefore not influencing the cost function during the transfer stage of learning.

The next important step in knowledge distillation was the possibility of transferring task specific knowledge from LLMs in Tang et al. 2019. They successfully distilled knowledge for a particular natural language task from BERT into a single layer BiLSTM, a first for NLP. Further work on distillation of LLMs followed, such as S. Wang et al. 2021, Smith et al. 2022, and Arora et al. 2022, which each attempt to distill the knowledge of an LLM into a smaller model for a particular task. Finally, we arrive at the current state of the field.

We consider both white-box and black-box knowledge distillation for LLMs:

In the **white-box** knowledge distillation technique, the student model has access to the internal parameters of the teacher model as well as the output. This allows the student model to see the teacher model's calculations directly. Sometimes, atypical divergences occur. For example, MiniLLM (Gu et al. 2023) optimizes reverse Kullback-Leibler divergence.

In the **black-box** knowledge distillation technique, the student model only has access to the output of the teacher model. This is friendly to the black-box API that many popular modern LLMs like GPT-4 have, which is part of why black-box KD is more popular in research. For instance, Ho, Schmid, and Yun 2023 is able to train smaller models to do Chain of Thought (CoT) reasoning after seeing examples from larger teacher models such as GPT-175B. Other work attempts to preserve the instruction following and in-context learning abilities of LLMs in smaller models in a similar way (Huang et al. 2022; M. Wu et al. 2023).

One interesting advance in black-box knowledge distillation is Zephyr (Tunstall et al. 2023). One of the most desirable properties of LLMs is the ability to answer natural prompts from users, but current approaches like distilled supervised fine-tuning to improve task accuracy result in models that do not respond well to such prompts. The authors of the paper approach the task of distilling a small aligned LLM by generating preference data via AI Feedback from an ensemble of teacher models - one teacher model scores the outputs of several other models on a set of prompts, and the prompts and their scores become the preference data - and fine-tuning the student model using distilled direct preference optimization, which maximizes the likelihood of ranking the preferred prompt response at the top in a preference model. Notably, this process preserves a complex property of the teacher LLM, alignment, in a much smaller student model.

However, KD techniques generally cannot produce a smaller model with all the capabilities of an LLM. Indeed, it has been shown that under some circumstances, student models trained with the goal of explicitly imitating LLMs can replicate the style of teacher models easily, but fail to replicate their accuracy. Meanwhile, KD techniques can be very effective to train small models on specific tasks, thanks to LLMs' Emergent Abilities, as mentioned above. Giving the explanation for the label generated by the teacher to the student model can make the student model learn more accurately

with less training data, outperforming other techniques such as fine-tuning pretrained small models. Since the student model is specialized to its task, it also has the potential to outperform the more powerful but more flexible teacher model, which was relying on ICL (Hsieh et al. 2023).

The results of KD are quite impressive. However, they do come with some drawbacks. For example, LLMs show impressive Emergent Abilities, but do not perform them perfectly. Large proportions of GPT-3's CoT explanations fail to introduce new explanatory information or have nothing to do with the prior task (P. Wang et al. 2023). This is hopefully a bit less of a problem as LLMs continue to become more powerful.

KD is possibly the most interesting type of compression in language models due to their ability to conserve Emergent Abilities, which are unique to LLMs.

Quantization is a model compression technique applicable to neural networks, particularly useful in language modeling. It involves reducing the precision of a model's numerical parameters, effectively shrinking the model's size without needing to reconstruct it from scratch. Initially crucial for reducing model sizes in image classification, as demonstrated in the pioneering work of Han, Mao, and Dally 2016, quantization has evolved to address the requirements of language processing.

The advent of mixed-precision training marked a key development in the quantization of language models. This method, detailed by Jacob et al. 2018, improves the balance between model size and performance by allowing for different model components to be quantized selectively, based on their impact on the overall performance.

A noteworthy contribution in this area is the "Q8BERT: Quantized 8Bit BERT" model, discussed in Zafrir et al. 2019. This research demonstrated the practicality of compressing the BERT model to 8-bit integers, reducing its size with a minimal reduction in performance. Q8BERT stands as a prime example of how quantization can be effectively applied to large-scale language models, showing that with careful, quantization-aware training, even compressed models can handle complex language understanding and generation tasks.

Additionally, one of the most striking recent advances in this area for LLMs specifically is GPTQ (Frantar, Ashkboos, et al. 2023), which is a one-shot weight quantization method for GPT models. Its primary contribution is to build on the Optimal Brain Compression algorithm previously introduced by the authors in Frantar, Singh, and Alistarh 2023 by making clever optimizations, but these result in significant compression gains without loss of accuracy, able to quantize to 4 and even 3 bits without significant issues, as well as a significant run-time speedup, and the ability to fit on a single GPU.

Lin et al. 2023 introduced Activation-aware Weight Quantization (AWQ), which matches the performance of mixed-precision training without the hardware inefficiency of storing and operating on multiple datatypes simultaneously. Since the precision of the quantized values of a given weight matrix depends on the maximum absolute value of that matrix, it is advantageous for more of the weights to be near the maximum to decrease the relative rounding error. This can be accomplished by increasing all weights leading out of a given neuron by a constant factor, and decreasing the activation of that neuron by the same factor to compensate.

What sets AWQ apart is its ability to preserve the generalization capabilities of LLMs without resorting to traditional methods such as backpropagation or reconstruction, which are typically used to maintain model integrity during optimization processes. This methodology may hasten the deployment of LLMs on edge devices, where computational resources are often limited.

Post-training quantization, which involves uniformly reducing the precision of all parameters after a model is fully trained, presents another strategy. However, it is essential to consider the trade-offs between computational efficiency and potential impacts on model performance.

Despite its advantages, quantization has its limitations. Excessive quantization can lead to a decline in performance, particularly in tasks that require nuanced language understanding and the preservation of complex inferences. The need to maintain a delicate balance between reducing

model size and preserving performance is emphasized in the research by X. Wu, Yao, and He 2023, indicating that while quantization can significantly decrease model size, it requires careful calibration to avoid undermining the model's capabilities.

Despite these challenges, quantization stands out in the realm of language model compression due to its ability to significantly reduce model sizes while retaining core linguistic capabilities, making it a valuable tool in the deployment of efficient and effective language models.

Low Rank Factorization Low-rank factorization is a model compression technique that tries to compress matrices or tensors in a model by decomposing them into products of smaller matrices or tensors. For a toy example, if we have a weight matrix A then we can set $A = UV$ where if $A \in F^{m \times n}$ then we'd want $U \in F^{m \times r}$ and $V \in F^{r \times n}$, setting $\text{rank}(A) = r << m, n$ or even just some $r << m$ not necessarily the rank.

One of the first examples of low-rank factorization for pre-trained language models is Noach and Goldberg 2020, which introduces the idea of trying to approximate each weight matrix $W \in \mathbb{R}^{n \times d}$ with some $W' = AB$ initialized not randomly, but via SVD, which is guaranteed to produce the best rank-r approximation for W, before switching to doing feature distillation on a training data set and also optimizing for task loss to regain performance. However, there is a relative dearth of low-rank factorization literature over the last 3 years, in contrast to the fast pace of progress in both quantization and pruning for LLMs.

Low-rank factorization seems to be a technique that matters primarily for the fine-tuning of language models - in this regard it's a huge space and time saver, and often the "rank" that you pick can be extremely low, down to 1 or 2 (Hu et al. 2021). However, it seems it isn't often used in the realm of wholesale model compression. Presumably there are several reasons, but it might be the fact that while there's SVD for matrices, tensors don't have an equivalent easily performed decomposition into lower rank tensors, and even SVD is an expensive operation; then to decompose a high-dimension tensor in a DNN, it has to be first decomposed into two-dimensional matrices before a matrix decomposition method can be applied. Furthermore, this approximation can often be very coarse; matrices in transformer models are often high-rank. Low-rank approximation for LLMs on its own suffers from many of the problems present in other methods; it's computationally intensive, requires additional data and compute to do retraining after compression to regain accuracy, and makes the model less robust (Zhu et al. 2023). However, recent approaches to mitigate these drawbacks include Li et al. 2023, which approximates the weight matrices in an LLM by the sum of a sparse pruned matrix and a low-rank matrix, and Hsu et al. 2022, which attempts to pick better approximations by doing SVD weighted by Fisher information indicating the relative importance of weights toward model performance instead of regular SVD.

Pruning Neural networks can also be pruned; that is, some of their weights can be effectively set to zero. The motivation behind pruning is the behavior of the mammlian brain; neurons that fire together, wire together, and those that don't have the connections between them removed in a pruning process.

Work on pruning stretches back to LeCun, Denker, and Solla 1989 and Karnin 1990, but Han, Mao, and Dally 2016 was the work that led to a sustained exploration of pruning as a technique for deep neural networks; it achieved compression rates between 35× and 49× through a combination of pruning, quantization, and Huffman coding. For the pruning step in particular, in magnitude pruning (Han, Pool, et al. 2015) one calculates an importance score for each weight, a global or per-layer threshold from the weights, and weights with scores below that threshold are zeroed out. This simple idea can be modified in several ways, as we'll see. Before we continue, it's worth mentioning the **Lottery Ticket Hypothesis** presented in Frankle and Carbin 2019, which states that dense, randomly initialized feed forward networks contain subnetworks - often a tenth of the size or less - which, when trained in isolation, reach test accuracy comparable to the original dense network.

Pruning naturally uncovers these subnetworks, and the hypothesis raises an intriguing question: why is it that neural network do seem to contain sparse subnetworks that maintain a large degree of their accuracy? This is especially true of LLMs, which often have outlying features which are orders of magnitude larger than average (M. Sun et al. 2023). Questions of interpretability and theoretical bounds on the level of sparsity achievable without harming performance remain enticing but out of reach.

More practically, we can discuss a taxonomy for pruning techniques. There are typically two kinds of pruning; **Unstructured pruning** decides whether to remove each weight individually, e.g. by comparing its absolute value against some threshold, while **structured pruning** removes groups of weights in order to more effectively reduce the computation required for inference, e.g. removing an entire neuron along with its connected weights.

While both quantization and pruning have been popular techniques for the compression of deep neural networks, for LLMs in particular the latter has been much less appealing. One reason may be that with most pruning techniques, the network must be trained for a short time after pruning to regain accuracy so that the remaining weights can make up for the pruned ones (Han, Pool, et al. 2015; Han, Mao, and Dally 2016; Liu et al. 2019). This process can often be data-hungry and time consuming, and these concerns are only more salient when considering the compute and storage intensive nature of LLMs. In particular, the simplest possible method of pruning, magnitude pruning (Han, Pool, et al. 2015), suffers from a serious drop in the performance of the model after it is pruned.

However, there are promising unstructured methods like SparseGPT (Frantar and Alistarh 2023) which do not require retraining; SparseGPT in particular treats the pruning of each weight matrix as a sparse regression problem which is approximated iteratively. At each iteration, a mask of the weights is fixed, and the masked weights are adjusted based on the Hessian, which is used to estimate (and minimize) the loss after pruning. Another novel method that requires no retraining is Wanda (M. Sun et al. 2023), which prunes a network in one shot by augmenting the standard technique of magnitude pruning with information about the input activations to each weight, estimated using a small set of calibration data, in a forward pass. While both approaches are more computationally complex than magnitude pruning, the pruned models remain performant relative to their dense counterparts, especially for models with a large number of parameters.

One other interesting approach that M. Sun et al. 2023 investigates is that magnitude pruning can get a significant boost by being more specific about the way that the threshold is calculated for removing weights; in particular, when doing magnitude pruning each weight is typically assigned to a *comparison group* before performing the pruning procedure. For example, in the simplest form of magnitude pruning the only comparison group is the entire set of weights; a slightly less naive approach might be setting the comparison groups by layer. However, even this second approach makes the implicit assumption that the weights connected to different outputs nevertheless should be approximately homogeneous with respect to magnitude and therefore comparable. Wanda divides weights into comparison groups based on which output node they're connected to, a more fine grained approach that yields a performance boost relative to magnitude pruning even standalone.

Unstructured pruning remains an area of active research, but for LLMs the technical challenges of applying pruning techniques that require additional retraining or a computationally intensive selection process to a large model have somewhat limited its application to date. Furthermore, depending on implementation there is no computational speedup for models that have undergone unstructured pruning; most software and hardware frameworks cannot accelerate the computation of sparse matrices. Even when acceleration is possible, the unstructured methods mentioned above often have the drawback of requiring high levels of sparsity to achieve a significant reduction in inference time, which can significantly hurt model performance (Mishra et al. 2021).

As an alternative approach, structured pruning techniques attempt to significantly regulate the pruning of weights in various ways; a more principled method can maintain performance and still

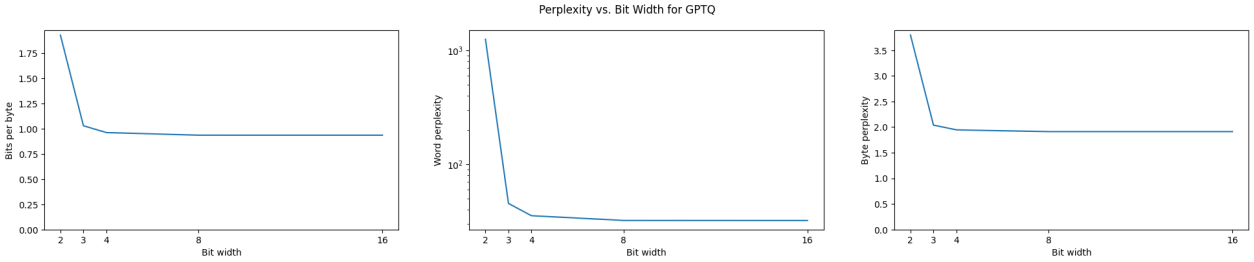
provide storage and inference latency gains. For instance, LLM-Pruner (Ma, Fang, and X. Wang 2023) attempts to identify coupled structures in the model, i.e. to partition the neurons between layers such that the weights that cross partitions have only a small effect on loss. This effect is again estimated with the Hessian. After a suitable partition has been found, the matrix of weights in each layer essentially becomes a few blocks of nonzero values, each connecting a pair of partitions between layers, and zero values elsewhere. Since block matrices are faster to multiply than dense matrices, this pruning method also speeds up inference.

Often, structured pruning results result in a wall-clock speedup but sacrifice performance by removing whole neurons or layers, which often encode key information in the network, and unstructured pruning can stay performant more easily while struggling to realize any latency gains. A structured pruning approach detailed in Mishra et al. 2021 is at the intersection of these issues; an NVIDIA project, its approach is motivated by hardware concerns and seeks to make high levels of sparsity in neural network weight matrices practical by introducing a form of structured 2:4 pruning, where two out of every four consecutive weights in a matrix are pruned. This allows for fine-grained pruning of the weights, and in fact is used in M. Sun et al. 2023, while accelerating execution.

Experimental Results

We used Hugging Face to determine the effect of quantization on model performance. Hugging Face has an implementation of AWQ which was merged from Casper Hansen's AutoAWQ package in November 2023. However, due to the recent release date of the merge, its full functionality has not yet been integrated with the rest of Hugging Face's Transformers package; in particular, it only supports quantization to 4 bits. These limitations are poorly documented, and we were not aware of them until attempting run the full experiment after a successful trial run.

Fortunately, Hugging Face also implements GPTQ, which accepts not only a variable bit width but also a dataset against which to optimize. We chose to quantize Facebook's smallest Open Pretrained Transformer (OPT) with 125M parameters, and evaluate its perplexity on the WikiText dataset, which contains a subset of high-quality Wikipedia articles. The original model's parameters are 16-bit floats, and GPTQ supports 2-, 3-, 4-, and 8-bit quantization, all of which were used in the experiment. To ensure that the quantized model generalized correctly, we gave GPTQ data outside of WikiText, namely a subset of the C4 (Colossal Cleaned Common Crawl) dataset. GPTQ also separates parameters into groups of a fixed size and chooses a separate scaling factor for each group, so we used the recommended group size of 128. Finally, we evaluated the model's word perplexity, byte perplexity, and "bits per byte," whose name suggests that it would be a simple rescaling of byte perplexity by a factor of ln(2). However, this is not the case, and as before, the documentation is lacking. Nonetheless, all three metrics suggest the same conclusion:



From the original 16 bits, performance is largely unaffected by 8- and 4-bit quantization, slightly degraded for 3-bit, and significantly so for 2-bit. In particular, the word perplexity is extremely high in the 2-bit case, to the point that it suggests that the model is no longer numerically stable at that resolution.

Discussion and Analysis

As we've seen, the field of model compression has become the subject of intense interest from the machine learning community. It has the potential to democratize access to large language models and other storage and compute intensive deep neural networks by allowing them to be compressed, fine-tuned, and deployed with relative ease at scale; in particular, clever combinations of the methods we discuss, e.g. M. Sun et al. 2023 and Li et al. 2023, allow LLMs to be made orders of magnitude smaller than their fully-sized counterparts while preserving much of their functionality.

However, many obstacles remain. While model compression for many kinds of deep neural network architectures has matured since works like Han, Mao, and Dally 2016, many LLM compression techniques continue to sacrifice performance significantly (Frantar and Alistarh 2023; Ho, Schmid, and Yun 2023). Additionally, most of these methods have little rigorous justification. Almost every paper attempts to justify why their particular methods ought to work, but it remains to be understood which of them are correct. Model compression shares another challenge with other subfields of machine learning, namely a lack of standardized benchmarks to fairly evaluate different techniques. Even if one fixes a measure of a particular compresed model's performance, e.g. on a certain dataset, one technique may outperform another at one memory footprint and fall behind at a different footprint, and it is unclear which is more efficient. The development of standard benchmarks for model compression techniques is an important future work in this field.

Each of the subfields we've covered here has promising new results and directions for future work; knowledge distillation has the potential capacity to transfer powerful, complicated abilities from closed-source models like GPT-4 to small, open source ones, while pruning methods are finding workarounds for retraining, as researchers become more precise about the way weights should be pruned. Quantization is seeing extremely rapid development as a technique and increasingly appears to be one of the methods of choice for language model compression; it does well on benchmarks, is relatively simple to implement even for mixed-precision quantization, and doesn't suffer as acute a performance drop in many cases to achieve similar reductions in model size. In conclusion, model compression is a promising field of study as frontier language models become increasingly common; the possibility of fitting a powerful language model onto an edge device like a laptop or mobile phone promises to make LLMs widely accessible.

References

LeCun, Yann, John Denker, and Sara Solla (1989). "Optimal Brain Damage". In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann. URL: https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

Karnin, E.D. (1990). "A simple procedure for pruning back-propagation trained neural networks". In: *IEEE Transactions on Neural Networks* 1.2, pp. 239–242. DOI: 10.1109/72.80236.

Bucila, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil (2006). "Model Compression". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '06. Philadelphia, PA, USA: Association for Computing Machinery, pp. 535–541. ISBN: 1595933395. DOI: 10.1145/1150402.1150464. URL: https://doi.org/10.1145/1150402.1150464.

Han, Song, Jeff Pool, et al. (2015). *Learning both Weights and Connections for Efficient Neural Networks*. arXiv: 1506.02626 [cs.NE].

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean (2015). *Distilling the Knowledge in a Neural Network*. arXiv: 1503.02531 [stat.ML].

Han, Song, Huizi Mao, and William J. Dally (2016). *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. arXiv: 1510.00149 [cs.CV].

Jacob, Benoit et al. (2018). *Quantization and training of neural networks for efficient integer-arithmetic-only inference*. arXiv: 1712.05877 [cs.CV].

Frankle, Jonathan and Michael Carbin (2019). *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*. arXiv: 1803.03635 [cs.LG].

Liu, Zhuang et al. (2019). *Rethinking the Value of Network Pruning*. arXiv: 1810.05270 [cs.LG].

Sun, Siqi et al. (2019). *Patient Knowledge Distillation for BERT Model Compression*. arXiv: 1908.09355 [cs.CL].

Tang, Raphael et al. (2019). *Distilling Task-Specific Knowledge from BERT into Simple Neural Networks*. arXiv: 1903.12136 [cs.CL].

Zafrir, Ofir et al. (2019). "Q8BERT: Quantized 8Bit BERT". In: *Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS 2019*. DOI: 10.48550/arXiv.1910.06188. arXiv: 1910.06188 [cs.CL].

Brown, Tom B. et al. (2020). *Language Models are Few-Shot Learners*. arXiv: 2005.14165 [cs.CL].

Cheng, Yu et al. (2020). *A Survey of Model Compression and Acceleration for Deep Neural Networks*. arXiv: 1710.09282 [cs.LG].

Noach, Matan Ben and Yoav Goldberg (2020). "Compressing Pre-trained Language Models by Matrix Decomposition". In: *AACL*. URL: https://api.semanticscholar.org/CorpusID:227905681.

Gholami, Amir et al. (2021). *A Survey of Quantization Methods for Efficient Neural Network Inference*. arXiv: 2103.13630 [cs.CV].

Hu, Edward J. et al. (2021). *LoRA: Low-Rank Adaptation of Large Language Models*. arXiv: 2106.09685 [cs.CL].

Mishra, Asit et al. (2021). *Accelerating Sparse Deep Neural Networks*. arXiv: 2104.08378 [cs.LG].

Wang, Shuohang et al. (2021). *Want To Reduce Labeling Cost? GPT-3 Can Help*. arXiv: 2108.13487 [cs.CL].

Arora, Simran et al. (2022). *Ask Me Anything: A simple strategy for prompting language models*. arXiv: 2210.02441 [cs.CL].

Dettmers, Tim et al. (2022). *LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale*. arXiv: 2208.07339 [cs.LG].

Hsu, Yen-Chang et al. (2022). *Language model compression with weighted low-rank factorization*. arXiv: 2207.00112 [cs.LG].

Huang, Yukun et al. (2022). *In-context Learning Distillation: Transferring Few-shot Learning Ability of Pre-trained Language Models*. arXiv: 2212.10670 [cs.CL].

Smith, Ryan et al. (2022). *Language Models in the Loop: Incorporating Prompting into Weak Supervision*. arXiv: 2205.02318 [cs.LG].

Dey, Nolan et al. (2023). *Cerebras-GPT: Open Compute-Optimal Language Models Trained on the Cerebras Wafer-Scale Cluster*. arXiv: 2304.03208 [cs.LG].

Frantar, Elias and Dan Alistarh (2023). *SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot*. arXiv: 2301.00774 [cs.LG].

Frantar, Elias, Saleh Ashkboos, et al. (2023). *GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers*. arXiv: 2210.17323 [cs.LG].

Frantar, Elias, Sidak Pal Singh, and Dan Alistarh (2023). *Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning*. arXiv: 2208.11580 [cs.LG].

Gu, Yuxian et al. (2023). *Knowledge Distillation of Large Language Models*. arXiv: 2306.08543 [cs.CL].

Ho, Namgyu, Laura Schmid, and Se-Young Yun (2023). *Large Language Models Are Reasoning Teachers*. arXiv: 2212.10071 [cs.CL].

Hsieh, Cheng-Yu et al. (2023). *Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes*. arXiv: 2305.02301 [cs.CL].

Li, Yixiao et al. (2023). *LoSparse: Structured Compression of Large Language Models based on Low-Rank and Sparse Approximation*. arXiv: 2306.11222 [cs.LG].

Lin, Ji et al. (2023). *AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration*. DOI: 10.48550/arXiv.2306.00978. arXiv: 2306.00978 [cs.CL].

Ma, Xinyin, Gongfan Fang, and Xinchao Wang (2023). *LLM-Pruner: On the Structural Pruning of Large Language Models*. arXiv: 2305.11627 [cs.CL].

Sun, Mingjie et al. (2023). *A Simple and Effective Pruning Approach for Large Language Models*. arXiv: 2306.11695 [cs.CL].

Touvron, Hugo et al. (2023). *LLaMA: Open and Efficient Foundation Language Models*. arXiv: 2302.13971 [cs.CL].

Tunstall, Lewis et al. (2023). *Zephyr: Direct Distillation of LM Alignment*. arXiv: 2310.16944 [cs.LG].

Wang, Peifeng et al. (2023). *SCOTT: Self-Consistent Chain-of-Thought Distillation*. arXiv: 2305.01879 [cs.CL].

Wu, Minghao et al. (2023). *LaMini-LM: A Diverse Herd of Distilled Models from Large-Scale Instructions*. arXiv: 2304.14402 [cs.CL].

Wu, Xiaoxia, Zhewei Yao, and Yuxiong He (2023). *ZeroQuant-FP: A Leap Forward in LLMs Post-Training W4A8 Quantization Using Floating-Point Formats*. arXiv: 2307.09782 [cs.CL].

Zhu, Xunyu et al. (2023). *A Survey on Model Compression for Large Language Models*. arXiv: 2308.07633 [cs.CL].