Investigating the Impact of Compression on Parametric Knowledge in Language Models

Satya Sai Srinath Namburi, Makesh Sreedhar, Srinath Srinivasan, Frederic Sala
University of Wisconsin - Madison
{sgnamburi, msreedhar, srinivasan32, fsala}@wisc.edu

Abstract

Compressing large language models (LLMs), often consisting of billions of parameters, provides faster inference, smaller memory footprints, and enables local deployment. Two fundamental compression techniques are pruning and quantization, with the former eliminating redundant connections in model layers and the latter representing model parameters with as few as 4 bits. The key tradeoff is between the degree of compression and the impact on the quality of the compressed model. Existing research on LLM compression primarily focuses on performance in terms of general metrics like perplexity or downstream task accuracy. More fine-grained metrics, such as those measuring parametric knowledge, remain significantly underexplored. To help bridge this gap, we present a comprehensive analysis across multiple model families (ENCODER, ENCODER-DECODER, and DECODER) using the LAMA and LM-HARNESS benchmarks in order to systematically quantify the effect of commonly employed compression techniques on model performance. A particular focus is on tradeoffs involving parametric knowledge, with the goal of providing practitioners with practical insights to make informed decisions on compression. We release our codebase[1] to enable further research.

1 Introduction

Large language models (LLMs) have demonstrated exceptional performance across diverse tasks. However, their deployment in real-world applications is hindered by their substantial size and the associated costs, even for inference [39, 41]. For instance, the LLama-65B model [44] uses approximately 130GB of RAM for 16-bit inference. To address this challenge, recent research has focused on developing novel compression techniques that enable efficient local deployment and inference. Notable examples of such techniques include SparseGPT [12] and LLM.int8() [7].

The tradeoff between model compression and quality is typically studied either through general metrics like perplexity [40, 31] or accuracy on standardized benchmarks [27, 10] like GLUE [47]. Furthermore, much of the literature studies such tradeoffs for one model or a particular class of models. Unfortunately, as a result, practitioners *do not have access to reliable insights or rules-of-thumb* to ensure they can make an informed decision for compression in their own models. This is because

- Metrics like perplexity are too general, while benchmark prediction metrics can be insensitive. For instance, recent findings suggest that distilled versions of foundational LLMs, known as imitation models, may exhibit stylistic similarities but potentially lack knowledge capabilities when compared to the models they seek to imitate [19].

[1] https://github.com/NamburiSrinath/LLMCompression

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

- Most of the recent research on compression techniques has primarily focused on DECODER models. The applicability and effectiveness of such techniques for large ENCODER and ENCODER-DECODER models [5] has yet to be extensively studied.

These difficulties suggest that there is a need for a more fine-grained understanding of the effects of compression schemes, comparing a variety of model families, compression techniques, and specialized measurements.

We address these challenges, specifically focusing on the preservation of *parametric knowledge*, i.e., knowledge acquired during pretraining and stored in model weights. This is particularly crucial for tasks involving reasoning and for specialized applications. Concretely, we examine the impact of different compression schemes on parametric knowledge across multiple model families (ENCODER, ENCODER-DECODER and DECODER) where we apply pruning and quantization approaches and analyze the performance of such techniques on downstream reasoning tasks. To the best of our knowledge, this work represents one of the first large-scale investigations in this direction. Our insights include:

- We observe that pruning all model modules together has the most significant impact on parametric knowledge, compared to pruning specific modules.
- We observe that at pruning levels of >50%, the parametric knowledge of all the models declines rapidly,
- Quantizing attention modules has less impact on performance compared to quantizing feed-forward networks for all the models,
- Across all models, structured pruning at the final layer has detrimental effects compared to unstructured pruning.

2 Methodology and Setup

In this section, we present a comprehensive overview of our experimental setup, including the rationale behind our design choices, along with the selection of models and datasets.

2.1 Settings Under Consideration

The general transformer block consists of an attention module followed by a feed-forward network. We consider three choices for compression: compress the attention module alone §2.2, compress the feed-forward network alone §2.3, or compress both together §2.4. We hypothesize that compressing each of these modules may have a different impact for specialized measurements like parametric knowledge, necessitating studying the above possibilities. Fig. §1 depicts the modules visually.

Our chosen compression techniques include pruning, quantization, and a combination of the two. Following the methodology proposed in [20], we adhere to the sequential order of pruning the selected group of modules first and then applying quantization. In addition, we also investigate the impact on distilled models and explore the effects of employing various combined compression techniques.

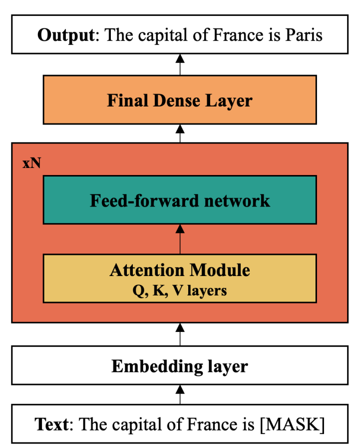

Figure 1: Block diagram of a simplified Transformer describing modules we compressed in our experiments.

2.2 Attention-only Global Compression

We include all the linear layers within all the attention modules of the model. For encoder-decoder models, we also consider the cross-attention blocks.

Attention-only Global Pruning, (Att_{GP}): We apply pruning to all the linear layers within the attention modules.

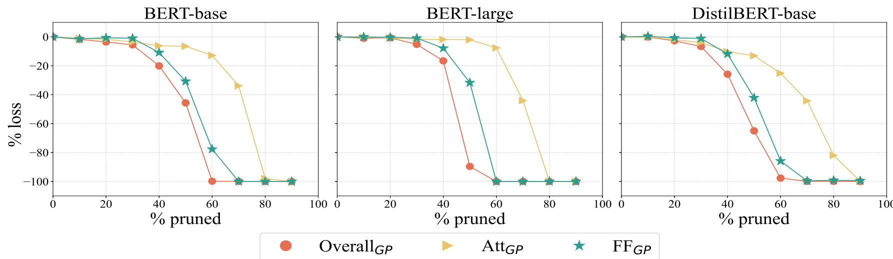

Figure 2: Averaged drop in Top-1 accuracy for encoder-only models for global pruning.

Attention-only Global Quantization, (Att_{GQ}): We quantize all the linear layers within the attention modules.

Attention-only Global Pruning + Quantization, (Att_{GPQ}): We prune the linear layers in the attention modules and subsequently quantize them.

2.3 Feed-forward-only Global Compression

We include all the linear layers within all the feed-forward networks of the model.

Feed-forward-only Global Pruning, (FF_{GP}): We employ pruning to all the linear layers within the feed-forward networks.

Feed-forward-only Global Quantization, (FF_{GQ}): We quantize all the linear layers within the feed-forward networks.

Feed-forward-only Global Pruning + Quantization, (FF_{GPQ}): We prune all the linear layers from feed-forward networks and subsequently quantize them.

2.4 Overall Global Compression

We specifically target the linear layers within the attention and feed-forward network. For experiments involving pruning as the compression method, experiments involving the final dense layer are discussed in §2.5. Under this compression, the different setups are:

Overall Global Pruning, ($Overall_{GP}$): We employ pruning to all the linear layers (except the final dense layer).

Overall Global Quantization, ($Overall_{GQ}$) - +: We apply quantization to all the linear layers.

Overall Global Pruning + Quantization ($Overall_{GPQ}$) +: We first apply pruning to all the linear layers (except the final dense layer), and subsequently, we quantize all the linear layers.

2.5 Final Dense Layer Pruning, (FL_P):

Recent studies [32, 33, 30] provide evidence suggesting that the final layers of a language model play a significant role in storing information. Accordingly, we focus on understanding how knowledge is encoded in the final layer. We treat the final layer as an individual module in our experimental setup and prune it. We consider L1-structured and L1-unstructured pruning as in §6.1. Additional details are provided in Appendix 7.

3 Experimental Results and Insights

To facilitate our discussion, we categorize pruning levels as follows:

- p_{low}: Sparsity levels of 10-30%
- p_{medium}: Sparsity levels of 30-50%
- p_{high}: Sparsity levels of >50%

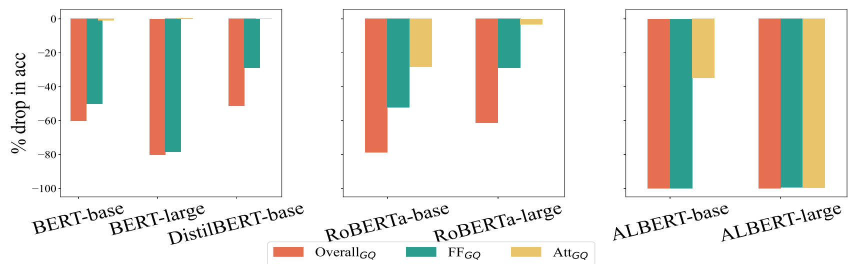

Figure 3: Averaged drop in Top-1 accuracy for encoder-only models for global quantization.

For encoder-only(§7.2.1) models, we report the % drop in top-1 accuracy, averaged across all the probes in LAMA (§7.3). For the decoder-only(§7.2.3) and encoder-decoder(§7.2.2) models, we report the % drop in accuracy, averaged across BoolQ, PIQA and Winogrande(§7.3). In the decoder-only and encoder-decoder plots, the majority-baseline indicates the accuracy when all the predictions are assigned to the majority class. For each scenario, additional details and plots are provided in Appendix §8.

3.1 Global Pruning

We observe that for encoder-only models (Fig. 2, 19), there is a minimal decline in performance at p_{low}. At p_{medium}, the drop in performance is more significant for **pruning feed-forward networks** (FF_{GP}) compared to **attention modules** (Att_{GP}). But for *encoder-decoder* (Fig. 6, 12) and *decoder-only* models (Fig. 5), pruning the *attention module* (Att_{GP}) has **more** impact on performance compared to **pruning feed-forward** networks (FF_{GP}).

3.2 Global Quantization

We observe that across all the models (Fig. 3, 14, 15), the performance drop is **less significant** when quantizing **attention modules** (Att_{GQ}) compared to quantizing **feed-forward** networks alone (FF_{GQ}). This contrasts with the results from global pruning (§3.1), where pruning attention-only modules had a more detrimental effect on encoder-decoder and decoder-only models.

3.3 Global Pruning + Quantization

For all the models (Fig. 7, 16, 17), at 20% sparsity, **compressing attention modules** (Att_{GPQ}) results in a **smaller** performance drop compared to **compressing feed-forward networks** (FF_{GPQ}). At 40% sparsity, the same trend is observed for encoder-only and decoder-only models.

3.4 Final Dense layer Pruning

For encoder-only models (Fig. 9), **L1-unstructured pruning** has a **smaller** impact compared to **L1-structured** pruning. We hypothesize that the final layer of the encoder-only models might encode knowledge in a structured or modular manner, and *any form of structured compression would disrupt this encoding*, resulting in a larger performance drop. Such a result would be consistent with existing approaches that enable editing knowledge in language models and rely on structure [32].

4 Conclusion

Compression is crucial when using and deploying LLMs. Despite its importance, existing empirical studies predominantly rely on generic measurements such as perplexity or standardized benchmark metrics when investigating the effects of compression.

In contrast, we provide a large-scale study that focuses on fine-grained effects on quantities like parametric knowledge. We study a variety of compression choices across multiple model families, providing usable insights into what types of compression schemes have the least and most significant impact on models. We hope this work serves as a useful step towards developing users' intuition for rules-of-thumb when selecting appropriate compression techniques in large language models. For future work, we hope to add additional, more specialized techniques for large language model compression.

5 Limitations

Our research has tackled a diverse combination of models, compression schemes, and compression targets within the vast large language model research area. We note that sophisticated and specialized compression techniques tailored to specific objectives for a particular class of models may exhibit distinct behavior compared to the findings presented in this study. Hence, our work does not aim to present an exhaustive set of findings that universally characterize the impact on parametric knowledge across all conceivable models and compression approaches. We believe that our study serves as a valuable starting point, offering a nuanced examination of prevalent methodologies.

We note, additionally, that we do not directly address the tradeoff between wall-clock inference time versus compression. While this is also an important tradeoff, the impact of compression on inference time contains many intricacies that are best treated with a separate large-scale study.

References

[1] H. Bai, W. Zhang, L. Hou, L. Shang, J. Jin, X. Jiang, Q. Liu, M. Lyu, and I. King. Binarybert: Pushing the limit of bert quantization. *arXiv preprint arXiv:2012.15701*, 2020.

[2] Y. Bisk, R. Zellers, J. Gao, Y. Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[3] A. Buluc and J. R. Gilbert. Challenges and advances in parallel sparse matrix-matrix multiplication. In *2008 37th International Conference on Parallel Processing*, pages 503–510. IEEE, 2008.

[4] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[5] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[6] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[7] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

[8] T. Dettmers, R. Svirschevski, V. Egiazarian, D. Kuznedelev, E. Frantar, S. Ashkboos, A. Borzunov, T. Hoefler, and D. Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[10] M. Du, S. Mukherjee, Y. Cheng, M. Shokouhi, X. Hu, and A. Hassan Awadallah. What do compressed large language models forget? robustness challenges in model compression. *arXiv e-prints*, pages arXiv–2110, 2021.

[11] A. Ettinger, A. Elgohary, C. Phillips, and P. Resnik. Assessing composition in sentence vector representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1790–1801, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics.

[12] E. Frantar and D. Alistarh. Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023.

[13] 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.

[14] T. Gale, E. Elsen, and S. Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[15] L. Gao, J. Tow, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, K. McDonell, N. Muennighoff, J. Phang, L. Reynolds, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. A framework for few-shot language model evaluation, Sept. 2021.

[16] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer. A survey of quantization methods for efficient neural network inference, 2021.

[17] Y. Goldberg. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*, 2019.

[18] M. A. Gordon, K. Duh, and N. Andrews. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*, 2020.

[19] A. Gudibande, E. Wallace, C. Snell, X. Geng, H. Liu, P. Abbeel, S. Levine, and D. Song. The false promise of imitating proprietary llms, 2023.

[20] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[21] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[22] G. Jawahar, B. Sagot, and D. Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics.

[23] N. Kassner and H. Schütze. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818, Online, July 2020. Association for Computational Linguistics.

[24] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[25] B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[26] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, Aug. 2021. Association for Computational Linguistics.

[27] C. Liang, S. Zuo, M. Chen, H. Jiang, X. Liu, P. He, T. Zhao, and W. Chen. Super tickets in pre-trained language models: From model compression to improving generalization. *arXiv preprint arXiv:2105.12002*, 2021.

[28] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021.

[29] 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, 2019.

[30] K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual knowledge in gpt. *arXiv preprint arXiv:2202.05262*, 2022.

[31] P. Michel, O. Levy, and G. Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.

[32] E. Mitchell, C. Lin, A. Bosselut, C. Finn, and C. D. Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

[33] E. Mitchell, C. Lin, A. Bosselut, C. D. Manning, and C. Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR, 2022.

[34] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort. A white paper on neural network quantization, 2021.

[35] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel. Language models as knowledge bases?, 2019.

[36] G. Prato, E. Charlaix, and M. Rezagholizadeh. Fully quantized transformer for machine translation. *arXiv preprint arXiv:1910.10485*, 2019.

[37] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[38] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[39] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

[40] A. See, M.-T. Luong, and C. D. Manning. Compression of neural machine translation models via pruning. *arXiv preprint arXiv:1606.09274*, 2016.

[41] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics.

[42] A. Talmor, Y. Elazar, Y. Goldberg, and J. Berant. olmpics – on what language model pre-training captures, 2020.

[43] C. Tao, L. Hou, W. Zhang, L. Shang, X. Jiang, Q. Liu, P. Luo, and N. Wong. Compression of generative pre-trained language models via quantization. *arXiv preprint arXiv:2203.10705*, 2022.

[44] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[45] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics.

[46] J. Wallat, J. Singh, and A. Anand. Bertnesia: Investigating the capture and forgetting of knowledge in bert, 2021.

[47] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[48] N. Weir, A. Poliak, and B. Van Durme. Probing neural language models for human tacit assumptions. *arXiv preprint arXiv:2004.04877*, 2020.

[49] M. Wu, A. Waheed, C. Zhang, M. Abdul-Mageed, and A. F. Aji. Lamini-lm: A diverse herd of distilled models from large-scale instructions. *arXiv preprint arXiv:2304.14402*, 2023.

[50] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, and D. Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

[51] Z. Yao, S. Cao, W. Xiao, C. Zhang, and L. Nie. Balanced sparsity for efficient dnn inference on gpu. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5676–5683, 2019.

[52] A. H. Zadeh, I. Edo, O. M. Awad, and A. Moshovos. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 811–824. IEEE, 2020.

[53] 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)*, pages 36–39. IEEE, 2019.

6 Background

In this section, we briefly discuss the various compression techniques we use in our study.

6.1 Pruning

Pruning involves reducing the model size by eliminating unnecessary or redundant connections between neurons or entire neurons altogether. Broadly speaking, pruning approaches can be classified into two types:

Unstructured Pruning: Each connection is treated as an individual entity, and sparsity is attained by eliminating connections with lower saliency. Although this approach enables the removal of less important connections without compromising performance, it leads to sparse matrix operations, which may not be optimal for certain hardware accelerators[2] [3, 14]

Structured Pruning: This involves removing a group of connections, such as channels or entire neurons, instead of individual connections. Unlike unstructured pruning, this approach avoids introducing sparse matrix operations. However, aggressive structured pruning may disproportionately impact the model's performance [51].

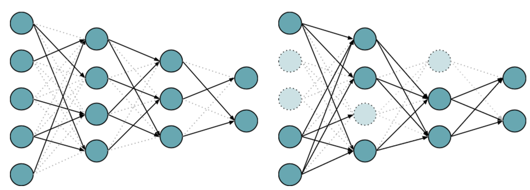

Figure 4: An illustration of unstructured (left) vs structured (right) pruning.

Choosing Saliency of Weights: When choosing the criterion to determine saliency, various factors can be taken into account, such as weight magnitude, importance to the overall network functionality, or contribution to specific tasks. Typically, the saliency of weights is determined based on their magnitudes when selecting which ones to remove during pruning. A sparsity of k% means that the least salient k% connections are removed.

The most commonly used pruning types are:

1. **L1-Unstructured:** Connections between neurons are eliminated individually, and their saliency is determined by their L_1-norm, i.e., the smallest weights are removed.
2. **Lp-Structured:** Connections are eliminated in a structured way, i.e., an entire layer/channel is removed and saliency is determined by their L_p-norm where p is a hyperparameter.

6.2 Quantization

Model parameters can be categorized into weights and activations, which are typically represented using 32 bits. Quantization aims to reduce the number of bits used for representing these parameters. A popular choice for this mapping is[3]:

$$Q(r) = \text{Int}(r/S) - Z,$$

where Q is the quantization operator, r is a real-valued input (weight or activation), S is a real-valued scaling factor, and Z is an integer zero-point. An important factor in mapping r to an integer is the scaling factor S. This is usually given by

$$S = \frac{\beta - \alpha}{2^b - 1}. \tag{1}$$

Here $[\alpha, \beta]$ denotes the clipping range and b is the quantization bandwidth. The process of determining the clipping range is known as calibration. Extensive research has been conducted to determine

[2]The current landscape is evolving as advanced accelerators are emerging that provide support for sparse multiplications.

[3]Uniform quantization maps real-values to equally spaced integers

the optimal range to reduce the bit representation while balancing accuracy, computational efficiency, and inference speed [16]. In most cases, statistics for weights are precomputed as they remain constant during inference. Often, it may be necessary to fine-tune the quantized model parameters to enhance performance on task-specific datasets. Taking these factors into account, various methods have been proposed [34]:

Post Training Static Quantization (PTSQ): The clipping range for activations is pre-calculated using a representative dataset, which is a small subset derived from the task-specific dataset. Using this clipping range, the activations are quantized in advance and thus remain static during inference.

Post Training Dynamic Quantization (PTDQ): The clipping range is dynamically calculated for each activation during inference. Although this introduces additional computational overhead during inference, it yields improved performance compared to Post Training Static Quantization (PTSQ) as the signal range is exactly calculated for each input.

Quantization Aware Training (QAT): The model undergoes a process known as fake-quantization, i.e., during training all the calculations involving forward and backward passes are performed in full-precision. Subsequently, after updating the weight parameters through gradient descent, the weights are quantized to a lower bit. While this approach achieves the highest performance, it requires finetuning the model.

We note that while a huge diversity of often sophisticated and specialized compression methods have been proposed, we focus on the standard approaches. This enables us to obtain the most general insights on compression tradeoffs.

7 Additional Experimental Details

This section contains some of the critical design choices, the models and datasets chosen.

7.1 Design Choices

- In our global pruning experiments ($Overall_{GP}$, Att_{GP}, FF_{GP}), we utilize L1-Unstructured and apply pruning percentages ranging from 10% to 90% with increments of 10%.
- For quantization experiments, as we seek to investigate the zero-shot capabilities of LLMs, we select post-training dynamic quantization §6.2, eliminating the need for finetuning (unlike quantization-aware training; QAT §6.2) or calibration of the model to a representative dataset (unlike post-training static quantization; PTSQ §6.2) and quantize to 8 bits (int8).
- Since the quantization of activations occurs during inference, which is dynamic in nature, the order of inputs within a batch has a minor impact on the final accuracy ($< 1\%$). Therefore, we seed the experiments to ensure consistent and reproducible results.
- Previous studies [18, 31] suggest that pruning levels of 30%-40% do not affect the model on downstream tasks. Such rules-of-thumb may or may not hold for parametric knowledge. In our experimental settings (GPQ, FL_P), we select 20% and 40% as the levels to understand whether and when a similar result holds.
- We note that the number of parameters compressed differs for different settings. We record all of the values required for normalizing measurements. However, our focus is predominantly aimed at *understanding the effects of compressing modules and their combinations* rather than presenting normalized results, and our insights reflect this framing. We provide full parameter counts that permit normalized quantities that can be used by practitioners who seek to directly apply our work.

7.2 Model Zoo

We consider the following models for our study. Where available, we choose both the base and large versions of the model to understand if larger models exhibit different behavior.

7.2.1 Encoder-only:

- **BERT** [9]: Pretrained on masked language modeling (MLM) and next sentence prediction (NSP) objective.

- **RoBERTa** [29]: Similar to BERT with different training choices (larger training dataset and removed NSP).
- **DistilBERT** [38]: Distilled version of BERT whose training objective includes MLM, a distillation loss, and a cosine embedding loss.
- **ALBERT** [24]: Parameter-reduced version of BERT using cross-layer parameter sharing and factorized embedding parameterization.

7.2.2 Encoder-Decoder:

- **Flan-T5** [5]: Instruction-finetuned encoder-decoder model with masked span corruption objective.
- **Lamini-Flan-T5** [49]: Flan-T5 model finetuned on LaMini instruction dataset[4] which is generated and distilled using ChatGPT output.

7.2.3 Decoder only:

- **Vicuna-7B** [4]: An instruction-based LLama derived model finetuned on user-shared conversations collected from ShareGPT.
- **WizardLM-7B** [50]: An instruction-based LLama derived model with instructions generated by LLMs (rather than humans) using the Evol-Instruct mechanism.

7.3 Datasets

We use the following datasets for our empirical analysis:

LAMA: To examine the effects of compression on encoder-only models, we use the LAMA (LAnguage Model Analysis) benchmark [35]. LAMA assesses the factual and commonsense knowledge of language models. Each example in LAMA is formulated as a cloze-style question, where either the subject or object is masked. By predicting the masked word, we can evaluate the model's ability to recover real-world facts. Specifically, we probe the encoder-only models with LAMA to investigate the impact of compression on various knowledge tasks. This benchmark consists of four datasets, namely TRex, Google-RE, ConceptNet, and SQUAD, each designed to assess specific types of relational knowledge. These datasets provide valuable insights into the model's performance and its understanding of different types of information.

Language model evaluation harness: To examine the effects of compression on encoder-decoder and decoder-only models, we utilize a subset of evaluation harness tasks [15] e.g., the BoolQ dataset [6], the PIQA dataset [2], and the Winogrande dataset [37]. These datasets provide a range of challenging prompts for each model type. We refer the reader to Table 1 for examples of samples from each dataset.

8 Insights from results

8.1 Global Pruning

We observe that for encoder-only models (Fig. 2, 19), there is a minimal decline in performance at p_{low}. At p_{medium}, the drop in performance is more significant for pruning feed-forward networks (FF_{GP}) compared to attention modules (Att_{GP}).

Finding: At p_{medium}, for encoder-only models, pruning attention modules (Att_{GP}) has a smaller impact compared to pruning feed-forward networks (FF_{GP}).

We observe that for encoder-decoder (Fig. 6, 12) and decoder-only models (Fig. 5), there is a minimal decline in performance at p_{low}. However, at p_{medium}, the drop in performance is more significant for pruning attention modules (Att_{GP}) compared to feed-forward networks (FF_{GP}).

[4]https://huggingface.co/datasets/MBZUAI/LaMini-instruction

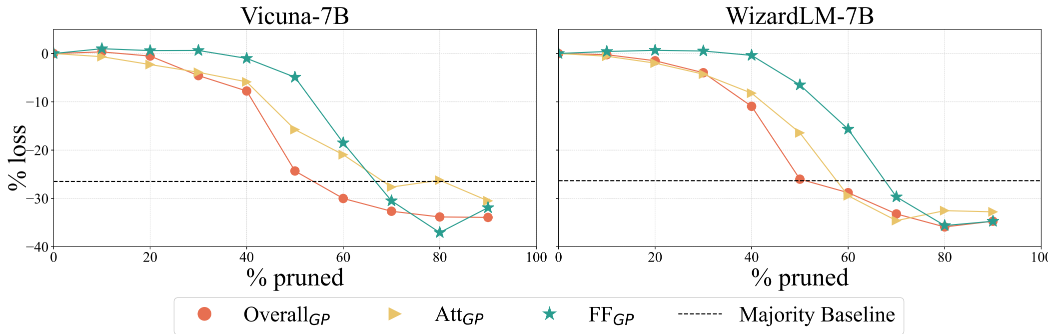

Figure 5: Averaged drop in accuracy for decoder-only models for global pruning.

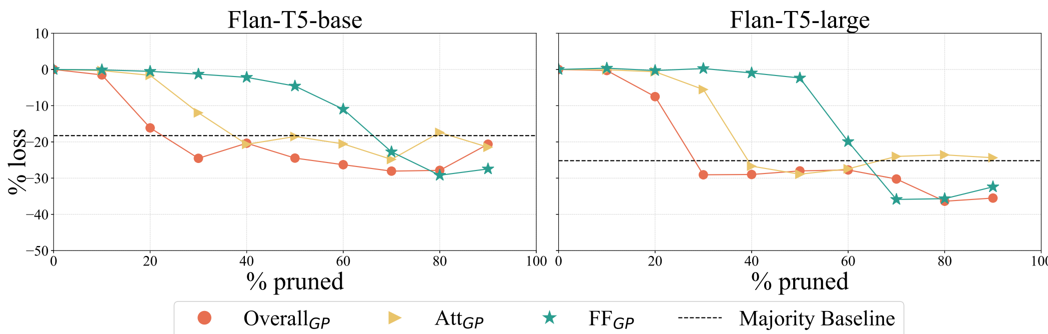

Figure 6: Averaged drop in accuracy for encoder-decoder models for global pruning.

Finding: At p_{medium}, for encoder-decoder and decoder-only models, pruning the attention module (Att_{GP}) has more impact on performance compared to pruning feed-forward networks (FF_{GP}).

We note that the number of parameters in the feed-forward networks is significantly higher than the number of parameters in the attention modules for all these models (Table 2). This observation provides a likely explanation for the pattern observed in encoder-only models, where pruning more parameters results in a higher loss of parametric knowledge. *However, it is counterintuitive for encoder-decoder and decoder-only models*, as we would expect that pruning the larger feed-forward networks would have a more significant impact on the parametric knowledge. We suspect that the feed-forward networks are over-parameterized and thus they can be pruned without a significant drop in performance.

Finding: For all the models, pruning all the modules together ($Overall_{GP}$) hurts the most.

Among all the models analyzed, pruning all modules together ($Overall_{GP}$) has the most significant negative impact on performance. This finding suggests that when compressing models, pruning all modules simultaneously leads to a greater loss of parametric knowledge compared to pruning specific modules or components individually. Therefore, it is crucial to carefully consider the implications of employing global pruning techniques. We note, additionally, that at p_{high}, the performance goes to zero as expected.

Additional results for global pruning on individual datasets for encoder-only models are shown in Fig 20, 21, 22; for decoder-only models at Fig 23; for encoder-decoder models at Fig 12, 24.

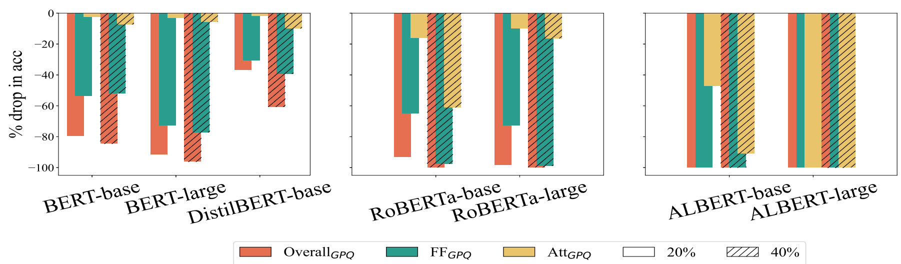

Figure 7: Averaged drop in Top-1 accuracy for encoder-only models for global pruning+quantization.

8.2 Global Quantization

We observe that across all the models (Fig. 3, 14, 15), the performance drop is less significant when quantizing attention modules (Att_{GQ}) compared to quantizing feed-forward networks alone (FF_{GQ}). This contrasts with the results from global pruning (§3.1), where pruning attention-only modules had a more detrimental effect on encoder-decoder and decoder-only models.

> **Finding:** For all the models, quantizing attention modules (Att_{GQ}) has lesser impact compared to quantizing feed-forward networks (FF_{GQ}).

We hypothesize that in the case of quantization, where all connections are preserved, the parametric knowledge in cross-attention modules may remain *relatively intact*. However, in pruning, as connections are eliminated, there may have a greater impact on the parametric knowledge in cross-attention modules, thereby affecting the overall capabilities of the model. It is also interesting to observe that the performance drop during quantization is almost similar to that of p_{medium}.

> **Finding:** For all the models, quantizing all the modules together ($Overall_{GQ}$) hurts the most.

It is intuitive that quantizing all the modules together ($Overall_{GQ}$) has the most significant negative impact. Additional results are shown in Table 3, 4, 5

8.3 Global Pruning + Quantization

For all the models (Fig. 7, 16, 17), at 20% sparsity, compressing attention modules (Att_{GPQ}) results in a smaller performance drop compared to compressing feed-forward networks (FF_{GPQ}). At 40% sparsity, the same trend is observed for encoder-only and decoder-only models. However, we notice the reverse for ENCODER-DECODER models i.e., that compressing feed-forward networks affects performance *less* than compressing the attention modules at 40% sparsity.

> **Finding:** For all the models, at 20% sparsity level, Att_{GPQ} hurts less compared to FF_{GPQ}.

We hypothesize that the sequential effects of pruning and quantization on the cross-attention modules could be responsible for this change in the order of impact. To test our hypothesis, we selectively prune and quantize the self-attention and cross-attention modules separately and found out that it is indeed the case (Fig. 8) and aligns with the claim made in [31]. Additional results for compressing attention-only modules are shown in Fig 11, 18. For fine-grained analysis on individual datasets, we refer the reader to Table 3, 4, 5.

8.4 Final Dense layer Pruning

For encoder-only models (Fig. 9), L1-unstructured pruning has a smaller impact compared to L1-structured pruning. We hypothesize that the final layer of the encoder-only models might encode knowledge in a structured or modular manner, and *any form of structured compression would disrupt*

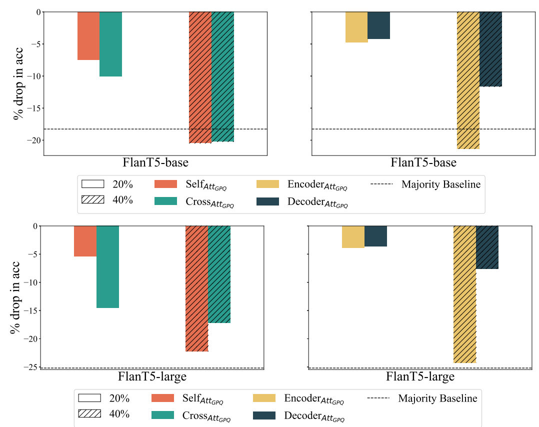

Figure 8: Averaged drop in accuracy for encoder-decoder models for different attention modules compression. $Self_{Att_{GPQ}}$: Compressing only self-attention modules, $Cross_{Att_{GPQ}}$: Compressing only cross-attention modules, $Encoder_{Att_{GPQ}}$: Compressing attention modules in encoder only, $Decoder_{Att_{GPQ}}$: Compressing attention modules in decoder only.

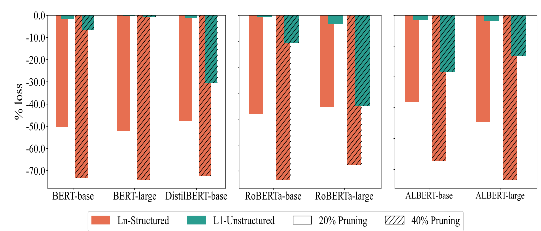

Figure 9: Averaged drop in Top-1 accuracy for encoder-only models for final layer pruning.

this encoding, resulting in a larger performance drop. Such a result would be consistent with existing approaches that enable editing knowledge in language models and rely on structure [32].

Finding: For encoder-only models, L1-unstructured leads to a smaller decrease in performance than L1-structured.

For decoder-only (Fig. 10) and encoder-decoder (Fig. 13) models, even at a sparsity level of 20%, the predicted accuracy is very close to the majority baseline. This finding aligns with the claims made in [33] that final layers encode significant amount of information. The drastic performance drop observed suggests that the final layers play a crucial role in encoding knowledge. Additional results for pruning the final layer are shown in Fig. 25, 26, 27.

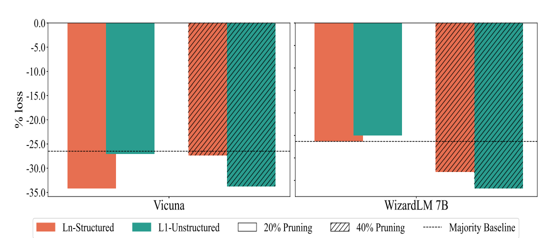

Figure 10: Averaged drop in Top-1 accuracy for decoder-only models for final layer pruning.

9 Related Work

Early works seeking to understand large language model behavior focused on contextual representations and how such models gain linguistic capabilities [17, 11, 22]. More recently, some lines of work have steered towards understanding how these models acquire factual and commonsense knowledge. Probing evolved as a way to understand the knowledge capabilities of these models [35, 23, 42, 48, 46].

Previous works such as [18, 31] pruned BERT and showed that it is resilient to a medium level of pruning. For example, [31] showed that after finetuning for a particular downstream task, it is possible to prune about 40% of the attention weights without any loss in performance. A particular focus has been to understand the importance of the attention mechanism [45, 31] by pruning the heads. In a similar fashion, works such as [53, 1, 52, 43, 36, 13, 8] pushed the limits of quantization on language models. Most of these works either has focused on one model class or one particular metric.

In another line of work, recent methods [26, 21, 28, 25] focus on alternatives to traditional finetuning of the model due to its scale. In contrast, our work primarily focuses on the in-built parametric knowledge present in the model. This means we do not finetune and instead seek to understand whether some of the previously described phenomenona are applicable to other models as well.

Also connected to this work are techniques that edit factual knowledge in models. The goal for such works is to avoid retraining or even finetuning models, instead seeking to directly change parameters connected to certain facts [32, 33, 30]. However, given our focus on compression, the main theme of our work differs. Nevertheless, it would be interesting to understand the impact of relying on compressed models when using such editing techniques.

10 Additional Results

This section contains all of the results we could not include in the body.

We first show individual plots for a set of experiments that track decrease in accuracy for several types of compression and models.

Next, we provide a table that contains information on the datasets used in our experiments. Afterwards, we provide tables with model details, including parameter counts, and explicit results for compression results across model families.

Afterwards, we show a large-scale comparison across datasets for encoder-decoder models under various attention module compression approaches. We provide LAMA probe results and finally, change-in-accuracy plots for a variety of datasets for different model classes.

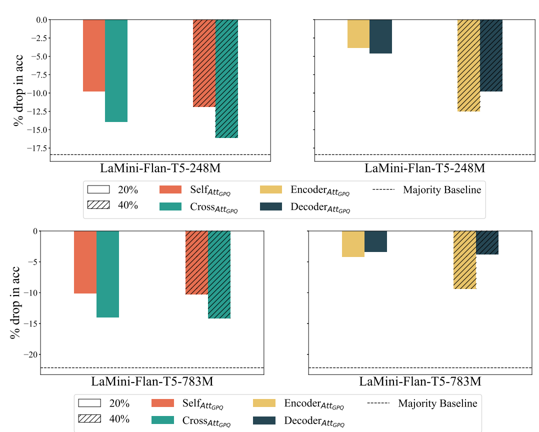

Figure 11: Averaged drop in accuracy for Lamini models under various attention modules compression

Table 1: Datasets in our experiments (we use dev sets for BoolQ, PIQA, and Winogrande)

Probe	**Type**	**#Egs**	**Question**	**Answer**
TRex	Factual	34k	Francesco Bartolomeo Conti was born in [MASK].	Florence
Google-RE	Factual	5.5k	Mareva is a [MASK] actress & former beauty Queen	French Polynesia
Squad	Factual	305	Newton played a [MASK] during Super Bowl 50.	Quarterback
ConceptNet	Commonsense	11k	Joke would make you want to [MASK].	laugh
BoolQ	Mix	3.2k	Is there any dollar bill higher than a 100?	No
PIQA	Commonsense	1.8k	**Goal:** "ice box"	Soln1
			Soln1: will turn into a cooler if you add water to it	
			Soln2: will turn into a cooler if you add soda to it	
Winogrande	Commonsense	1.2k	The trophy doesn't fit into the brown suitcase because *it's* too small.	suitcase
			The trophy doesn't fit into the brown suitcase because *it's* too large.	trophy

Table 2: Number of parameters (in million) across all the models

Model Name	$Overall_{GP}$	$Att_{GP,GQ,GPQ}$	$FF_{GP,GQ,GPQ}$	FL_P	**Total trainable parameters (for** $Overall_{GQ,GPQ}$**)**
Bert-base	86	21	64	23	109
Bert-large	303	75	226	31	334
Roberta-base	86	21	64	39	124
Roberta-large	303	75	226	51	355
Distilbert-base	43	14	28	23	66
Albert-base-v2	85	28	57	4	89
Albert-large-v2	302	101	201	4	306
FlanT5-base	198	85	113	25	223
Distil-FlanT5-base	198	85	113	25	223
FlanT5-large	717	302	311	33	750
Distil-FlanT5-large	717	302	311	33	750
Vicuna	6476	2147	4329	131	6607
Wizard-LM	6476	2147	4329	131	6607

Table 3: Results from compressing different modules for encoder-only models

Model	**Dataset**	**Baseline**	$Overall_{GQ}$	Att_{GQ}	FF_{GQ}	$Overall_{GPQ}$		Att_{GPQ}		FF_{GPQ}	
						20%	**40%**	**20%**	**40%**	**20%**	**40%**
BERT-base	TREx	30.27	11.536	29.903	16.614	5.552	4.233	29.675	29.217	14.691	14.628
	GoogleRe	10.29	4.374	10.109	5.662	2.377	1.96	9.873	9.673	4.628	5.753
	Squad	12.987	5.844	13.312	3.896	2.597	0.974	12.338	10.39	4.87	4.87
	Conceptnet	16.33	6.02	15.897	8.677	3.919	3.69	16.224	15.553	8.262	8.324
BERT-large	TREx	30.485	4.376	30.539	4.592	1.277	0.848	29.624	29.195	8.145	6.673
	GoogleRe	10.472	3.829	10.309	3.612	1.434	0.436	10.018	9.964	4.247	3.376
	Squad	15.909	2.273	16.883	2.597	1.948	0.325	15.584	14.935	3.247	2.597
	Conceptnet	19.534	4.652	19.048	5.649	1.818	1.342	18.801	18.086	5.164	4.864
RoBERTa-base	TREx	11.9	3.566	8.014	8.421	1.424	0.006	11.529	4.663	6.489	0.048
	GoogleRe	4.102	1.143	2.668	1.234	0.617	0	3.249	1.779	1.053	0.091
	Squad	8.442	0.325	4.545	2.922	0	0	5.195	1.623	1.299	0
	Conceptnet	17.036	3.769	14.467	7.247	0.83	0.026	14.865	8.147	5.746	0.856
RoBERTa-large	TREx	16.862	6.264	16.159	11.885	0.175	0.019	15.845	15.714	5.355	0.057
	GoogleRe	3.811	1.488	3.829	2.868	0.036	0	2.196	1.869	0.926	0.054
	Squad	13.636	4.87	13.312	7.468	0	0	12.013	10.714	1.299	0
	Conceptnet	19.861	8.324	19.119	15.244	1.006	0.079	18.775	17.036	7.15	0.494
DistilBERT-base	TREx	28.082	14.186	28.184	20.383	18.285	12.673	27.021	25.229	20.367	18.593
	GoogleRe	10.181	4.791	10.073	8.766	5.681	3.92	9.111	8.403	8.113	7.241
	Squad	10.39	5.195	10.714	6.494	6.494	2.273	11.688	9.74	5.519	5.195
	Conceptnet	14.308	6.391	14.132	9.101	9.339	5.976	14.105	13.346	9.727	7.238
ALBERT-base	TREx	13.016	0	9.213	0.003	0	0.003	7.595	0.845	0	0.003
	GoogleRe	1.307	0	0.762	0	0	0	0.436	0.036	0	0
	Squad	3.896	0	1.623	0	0	0	1.299	0.649	0	0
	Conceptnet	9.86	0.018	6.682	0.009	0	0	5.517	1.077	0.009	0.018
ALBERT-large	TREx	22.057	0	0	0.133	0	0.013	0.003	0.006	0	0.003
	GoogleRe	2.686	0	0	0	0	0	0	0	0.018	0
	Squad	9.74	0	0	0	0	0	0	0	0	0
	Conceptnet	14.794	0.009	0.071	0.132	0	0.009	0.026	0.044	0	0.009

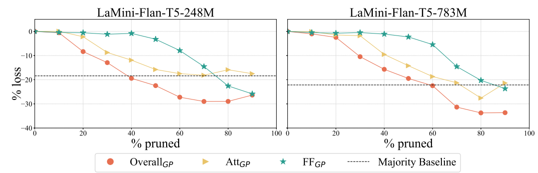

Figure 12: Averaged drop in accuracy for global pruning

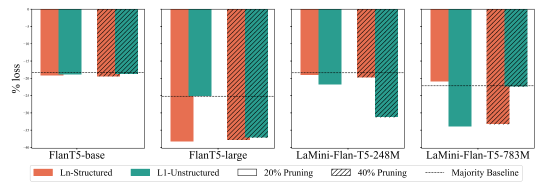

Figure 13: Drop in Top-1 accuracy (averaged across selected lm-harness) for encoder-decoder models for various local pruning

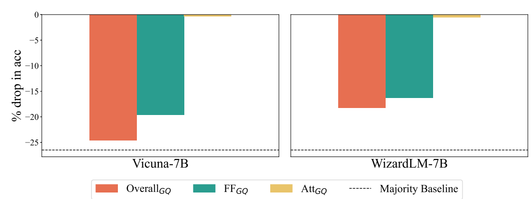

Figure 14: Averaged drop in accuracy for global quantization for decoder-only models.

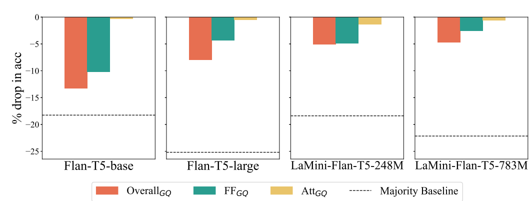

Figure 15: Averaged drop in accuracy for global quantization for encoder-decoder models.

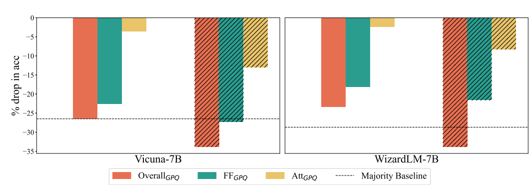

Figure 16: Averaged drop in accuracy for global pruning+quantization for decoder-only models.

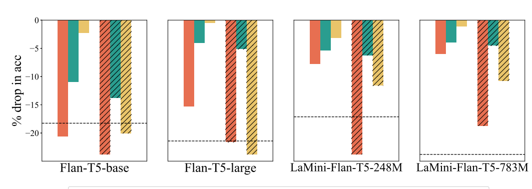

Figure 17: Averaged drop in accuracy for global pruning+quantization for encoder-decoder models.

Table 4: Results from compressing different modules of decoder-only models.
Majority baselines are - **BoolQ**: 0.621, **PIQA**: 0.504, **Winogrande**: 0.504

Model	Dataset	Baseline	$Overall_{GQ}$	Att_{GQ}	FF_{GQ}	$Overall_{GPQ}$		Att_{GPQ}		FF_{GPQ}	
						20%	**40%**	**20%**	**40%**	**20%**	**40%**
Vicuna-7B	Boolq	0.7657	0.5211	0.7645	0.5437	0.5272	0.4398	0.7125	0.556	0.5346	0.4495
	Piqa	0.778	0.611	0.7671	0.6556	0.5979	0.5332	0.7617	0.7421	0.6202	0.6257
	Winogrande	0.6725	0.5391	0.678	0.5825	0.5043	0.4925	0.663	0.6298	0.5612	0.5367
WizardLM-7B	Boolq	0.7844	0.6073	0.7841	0.6003	0.5817	0.4453	0.7514	0.6801	0.6141	0.5547
	Piqa	0.7622	0.6518	0.7508	0.6654	0.623	0.5593	0.7481	0.728	0.6556	0.6371
	Winogrande	0.6646	0.5517	0.6638	0.588	0.5312	0.5193	0.6622	0.6346	0.5738	0.5817

Table 5: Results from compressing different modules of encoder-decoder models.
Majority baselines are - **BoolQ**: 0.621, **PIQA**: 0.504, **Winogrande**: 0.504

Model	Dataset	Baseline	$Overall_{GQ}$	Att_{GQ}	FF_{GQ}	$Overall_{GPQ}$		Att_{GPQ}		FF_{GPQ}	
						20%	**40%**	**20%**	**40%**	**20%**	**40%**
FlanT5-Base	Boolq	0.7887	0.618	0.7841	0.6352	0.5049	0.482	0.7609	0.5058	0.6275	0.6125
	Piqa	0.6621	0.6251	0.6665	0.6415	0.5724	0.5419	0.6605	0.5631	0.6393	0.6077
	Winogrande	0.5422	0.4862	0.5359	0.5138	0.5051	0.4949	0.5272	0.5241	0.5075	0.498
FlanT5-Large	Boolq	0.8645	0.8034	0.8615	0.8165	0.6498	0.5618	0.856	0.4107	0.819	0.7969
	Piqa	0.7138	0.6638	0.716	0.6942	0.6181	0.5555	0.7214	0.6616	0.6953	0.6921
	Winogrande	0.5991	0.5375	0.5896	0.573	0.5185	0.5067	0.5864	0.4957	0.5596	0.5572
Lamini-Flan-T5-248M	Boolq	0.7982	0.7297	0.8015	0.7346	0.667	0.4349	0.7569	0.6266	0.7315	0.7263
	Piqa	0.6676	0.6393	0.6594	0.6507	0.6208	0.5462	0.6627	0.6208	0.6534	0.6338
	Winogrande	0.5304	0.5257	0.5083	0.513	0.543	0.5051	0.5099	0.5004	0.4964	0.5028
Lamini-Flan-T5-783M	Boolq	0.8306	0.7982	0.8294	0.7979	0.7716	0.6211	0.8226	0.6783	0.7994	0.7982
	Piqa	0.7073	0.673	0.7051	0.6899	0.6855	0.6192	0.7008	0.6937	0.6882	0.6866
	Winogrande	0.5549	0.5241	0.5454	0.5517	0.5193	0.4878	0.5478	0.5114	0.5288	0.5201

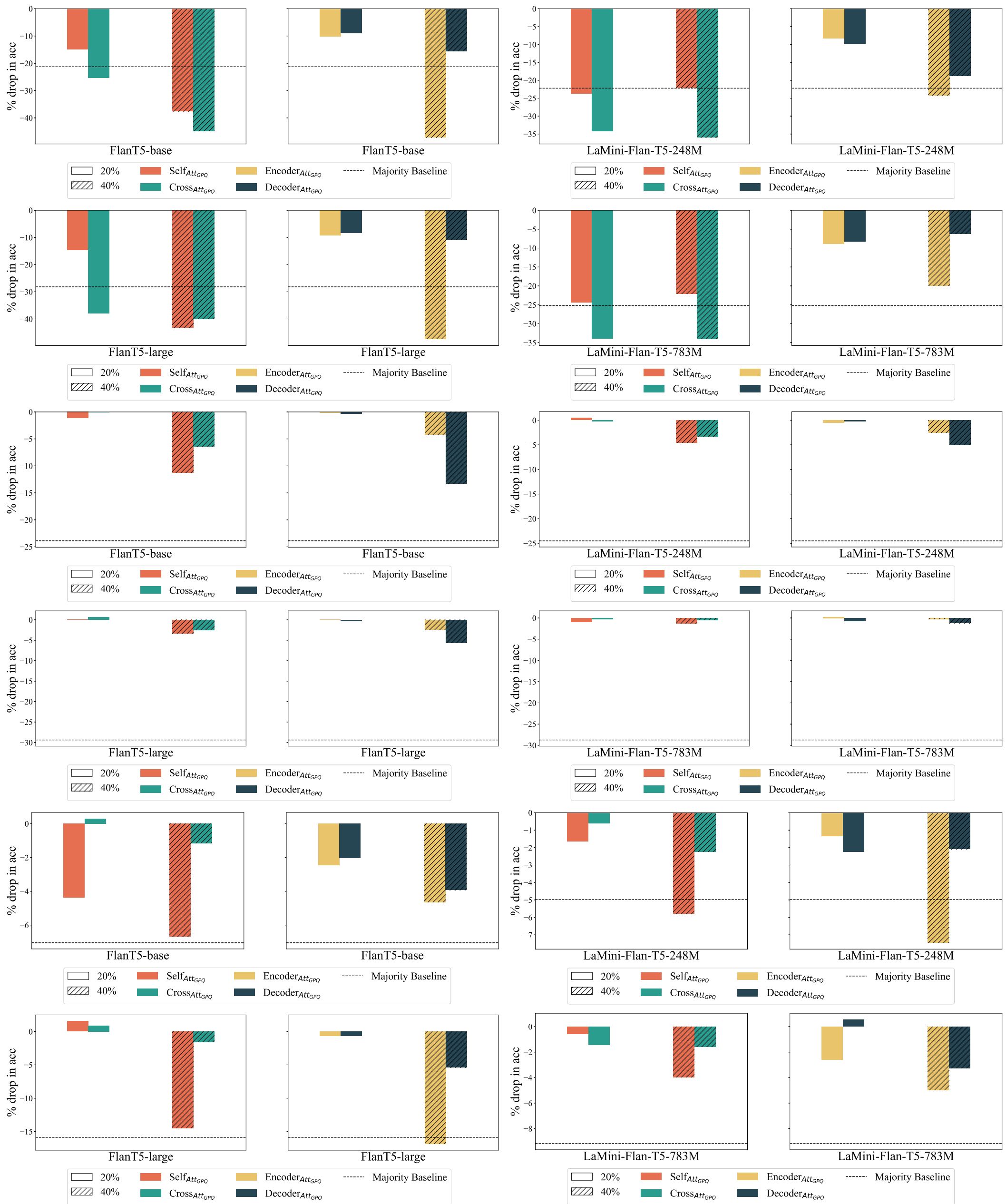

Figure 18: Drop in accuracy across various datasets for encoder-decoder models under various attention modules compression. Top-to-Bottom: BoolQ, PIQA, Winogrande

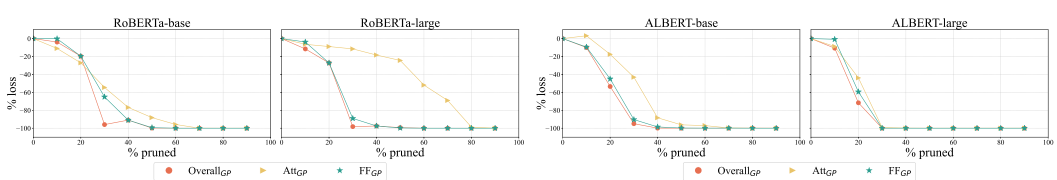

Figure 19: Averaged drop in Top-1 accuracy for encoder-only models.

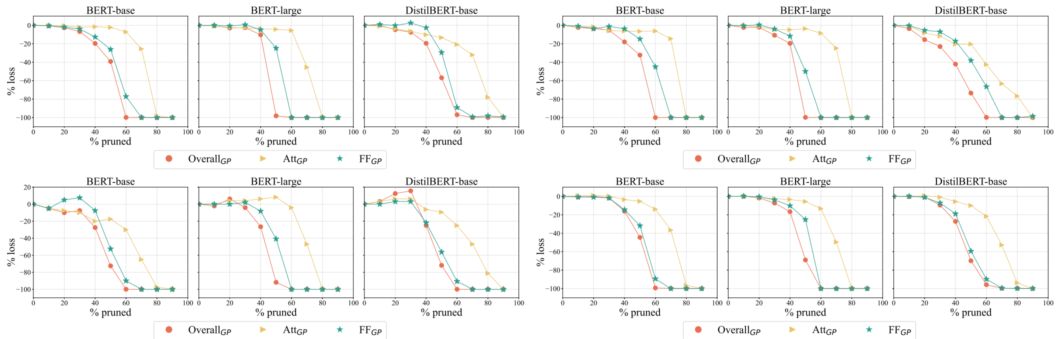

Figure 20: Drop in Top-1 accuracy for respective LAMA probes. Left-to-Right, Top-to-bottom: TREx, Google-RE, SQUAD, Conceptnet.

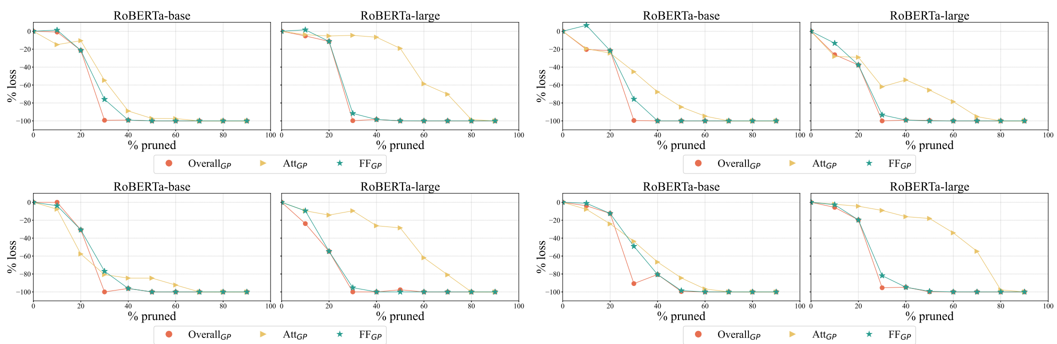

Figure 21: Drop in Top-1 accuracy for respective LAMA probes. Left-to-Right, Top-to-bottom: TREx, Google-RE, SQUAD, Conceptnet.

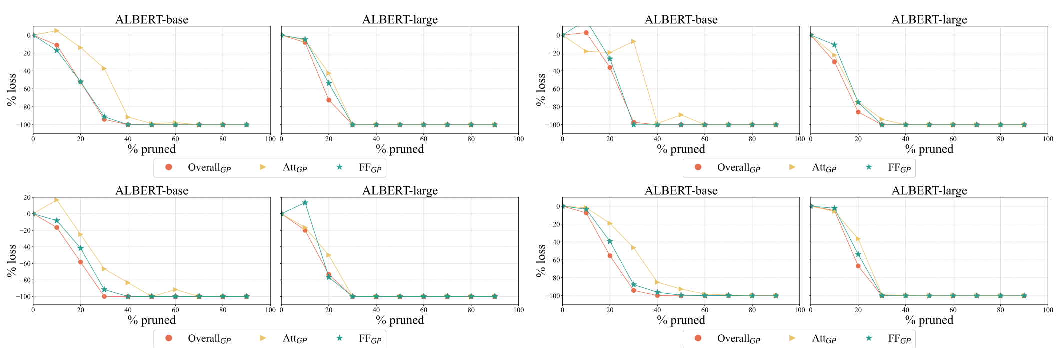

Figure 22: Drop in Top-1 accuracy for respective LAMA probes. Left-to-Right, Top-to-bottom: TREx, Google-RE, SQUAD, Conceptnet.

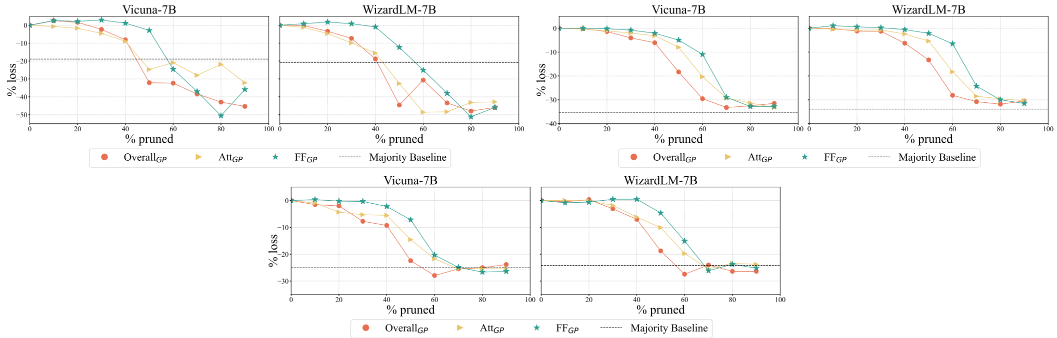

Figure 23: Drop in accuracy for decoder-only models. Left-to-Right, Top-to-Bottom: BoolQ, PIQA, and Winogrande

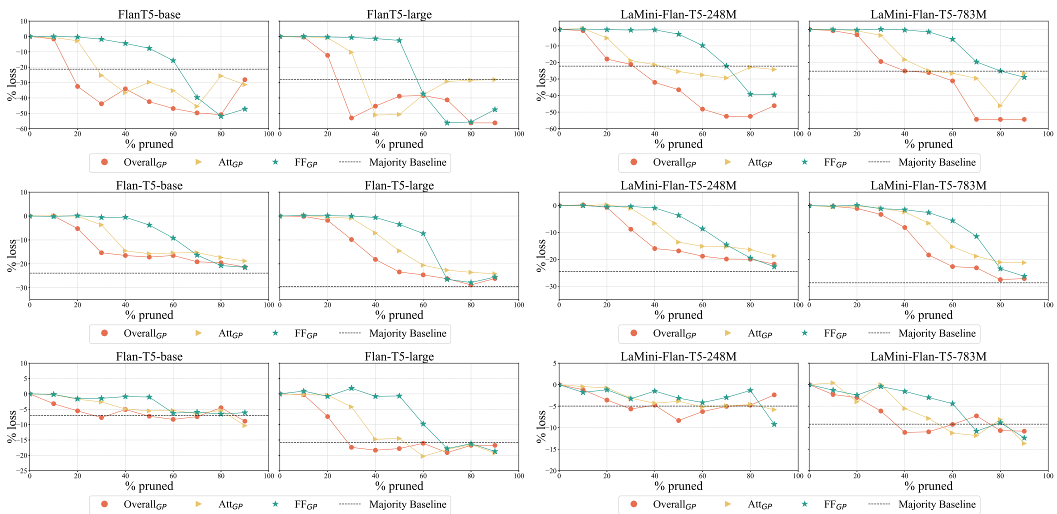

Figure 24: Drop in accuracy across various datasets for encoder-decoder models. Top-to-Bottom: BoolQ, PIQA, Winogrande

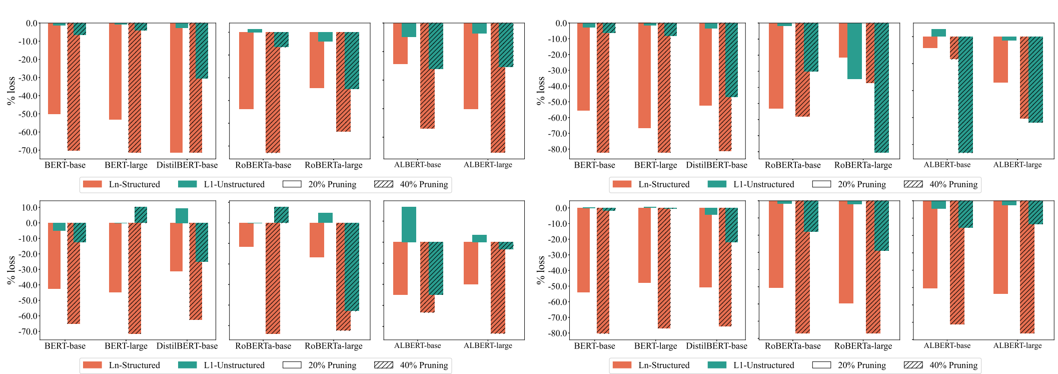

Figure 25: Drop in Top-1 accuracy across various datasets for encoder-only models for FL_P. Left-to-Right, Top-to-Bottom: TRex, Google-RE, SQUAD, ConceptNet

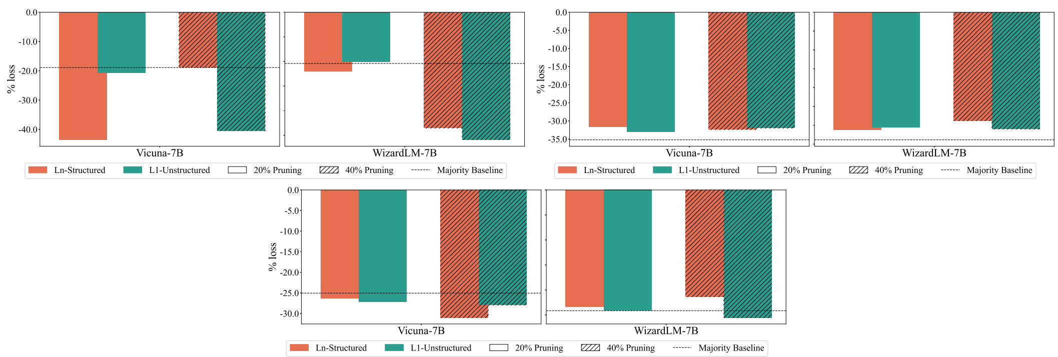

Figure 26: Drop in accuracy across various datasets for decoder-only models for FL_P. Left-to-Right, Top-to-Bottom: BoolQ, PIQA, and Winogrande

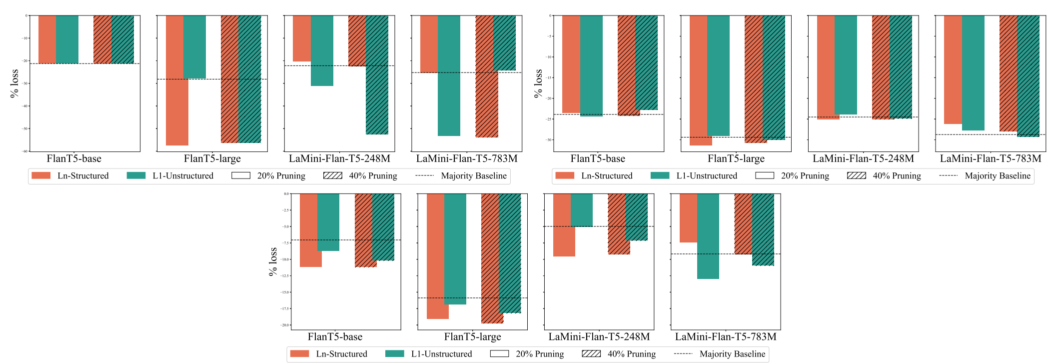

Figure 27: Drop in accuracy across various datasets for encoder-decoder models for FL_P. Left-to-Right: BoolQ, PIQA, and Winogrande