Information-Aware Compression Schemes Using Causal Tracing

Erik Scarlatescu
Georgia Institute of Technology
escarlatescu3@gatech.edu

Ish Mehta
Georgia Institute of Technology
imehta34@gatech.edu

Abstract

With the recent advances in performance of Large Language Models (LLMs) their popularity has pushed researchers to find ways to make LLMs more accessible. This is because the higher performance comes at a cost of computational efficiency. Though the trade-off between performance and efficiency is well-known, there is still much to be learned about the relationship between model size and LLM performance. There have been several attempts to achieve this, most of which have been grounded in pruning and quantisation. In this paper, we explore a novel approach to measure redundancy among network layers, and accordingly leverage quantisation to achieve a minimal model size while retaining performance.

1 Introduction

Large Language Models (LLMs) have made remarkable strides in natural language processing (NLP). However, their massive size creates practical barriers. The computational power and memory needed to run these models restrict their use to powerful machines. Consider the GPT-175B model: its 175 billion parameters demand around 300GB of memory (FP16). Therefore, compression techniques are essential to make these models more widely accessible.

2 Background

There are a wide range of compression techniques that have been explored that seek to mitigate these computational costs, some of which include quantization, pruning, and knowledge distillation. Network pruning has been widely studied since it was first proposed. Pruning comes in several varieties, with some methods removing individual weights directly, while others remove whole rows/columns, or channels in the case of convolutional networks. These methods have seen success with smaller language models (Hoefler et al., 2021), however, for larger models, intensive retraining and pruning is needed to recover model accuracy. Additionally, most one-shot methods need significant computational resources when dealing with larger models. However, one recent method that has managed to overcome this is SparseGPT, which has achieved great results.

Quantization is another similar approach to pruning: it represents the weights and activations of neural networks with low-bit precision (Gong et al., 2024). There are two general approaches to quantization, post-training quantization (PTQ) and quantization-aware training (QAT). PTQ applies quantization after training a neural network, however this often results in performance degradation. QAT modifies the training process to allow the model to learn weights that will respond well to being quantized. While this results in a model with better performance, it is more computationally demanding and significantly more difficult to implement.

3 Motivation

The aforementioned quantization and pruning approaches generally operate uniformly on all layers of deep learning models. One can instantly see why this is not always ideal. Different layers of the model might contain more or less information, meaning that uniformly compressing every layer might compress some parts of the model too aggressively, while not compressing other parts enough. This can lead to more information loss then necessary.

Ideally, we would like to have a compression method that takes into account how much information is encoded in each layer, measure the redundancy of each layer, and use this information to determine the amount of compression to apply. In-

tuitively, this would seem to perform better than a uniform compression approach. This is the method proposed by this project. Several methods exist which can measure the redundancy of a neural network. One such category of methods stem from information theory, where it quantifies how much information is present in each part of the network. However, existing methods for this type do not scale well to larger models, meaning that another approach is needed for LLMs. One such appraoch is low-rank factorizations, which break down each weight matrix into smaller matrices. Those which can be reduced more have less information. While this seems to be what we are looking for, there a major flaw with this: each layer is considered independently, with no regard for how information flows through the model to create representations.

While the factorisation trick does not accomplish our goal, it does act as a good heuristic i.e. the magnitude of model weights does provide a rough heuristic of a layer's redundancy. Additionally, one can also look at the magnitude of the gradients of the model weights with respect to the cost function for another heuristic. Both of these approaches are somewhat simplistic, but they work well enough to be used as benchmarks for this project.

In this project, the primary heuristic we use to measure redundancy is based off of casual tracing described in "Locating and Editing Factual Associations in GPT" (Meng et al., 2023). Casual tracing is used by the authors of this paper to identify where the LLMs store facts, edit them to deliberately to change the fact stored in the model. For example, they are able to find out which part of the model stores the fact that the Space Needle is located in downtown Seattle and then edit the network to make it believe that the Space Needle is in Paris. This is impressive, and can be modified for our project. We can use their approach to identify where the model stores facts, by using a dataset of facts, and therefore where the model stores most information which can then be used to select the compression level.

Figure 1 explains how causal tracing works. The model is provided with an input fact "The Space Needle is downtown Seattle" which is referred to as the clean run. Noise is then used to corrupt the part of the input that is related to the subject, the "Space Needle", in what is referred to as the corrupted run. Once this is done, the model has no way of guessing Seattle since it does not know what the subject of the fact is. After which we iterate over the states of the MLP and attention layers. Each one is transplanted into the corrupted model one at a time, as denoted by the purple arrow in figure 2. If the model is able to guess "Seattle" for the corrupted input and a particular transplant, it means that the transplanted state from the clean run has the information that the model needs to guess "Seattle".

If this particular MLP or attention layer consistently exhibits this behavior, that means that it is generally storing a high amount of information, and therefore has low redundancy. This method for measuring information density is empirically backed, as the authors of this paper are then able to use it to precisely change the beliefs of the model. Additionally, this method runs relatively quickly, making it a good candidate to test as a heuristic for redundancy.

4 Method

The first step is to measure the redundancy in the network layers. To do so, we test the causal tracing approach and compare it to using weight and gradient magnitudes as heuristics. We consider simple weight-magnitude pruning and uniform post-training quantization as the compression scheme. This was applied to the base GPT2 model with 125 million parameters.

Redundancy scores for each layer was produced using each of the methods described above. The dataset used to determine the score for causal tracing was the same one used by the original authors in their paper. A very similar approach was used to calculate whether a particular MLP or attention layer contained the information necessary to predict the correct output. For weight magnitude and magnitude of weight gradient, the square of the Frobenius norm was used to produce the score.

After obtaining the redundancy scores for all 12 layers, they are normalized into z-scores. Layers with a z-score below -0.5 are labeled as having "low" information density, while those above +0.5 are labeled as having "high" information density. Layers with z-scores between -0.5 and +0.5 are categorized as having "medium" information density. Depending on their classification, layers with "high" information density are compressed the least while the layers with "low" information density are compressed the most.

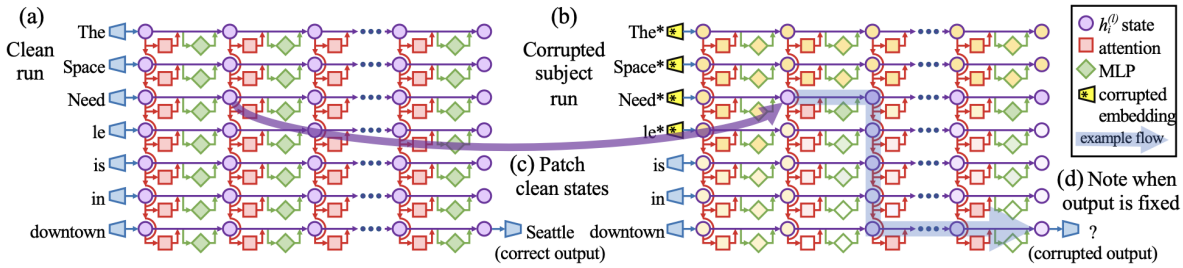

Figure 1: Explains method used by Meng et al. in (Meng et al., 2023)

5 Results

A hyper-parameter sweep was executed to assess the degree of compression necessary for layers characterized by low, medium, and high information densities. This sweep was conducted across three dimensions: pruning level, quantization level, and redundancy method, with performance evaluation conducted using the WikiText 2 dataset and reported in perplexity units. Due to the sweep encompassing three dimensions, slices are depicted concerning the chosen pruning option in Figure 2.

Each image illustrates the axis labeled quantization, representing the quantization level for low, medium, and high information densities respectively. For instance, in a row marked [8, 16, 24], quantization levels of 8 bits, 16 bits, and 24 bits are applied for layers with "low", "medium", and "high" information densities respectively. Within each image, consistent pruning levels are employed. For example, in the diagram titled "Prune: [0.3, 0.2, 0.1]", pruning ratios of 0.3, 0.2, and 0.1 are utilized for layers characterized by low, medium, and high information densities respectively. The column in each image designates the redundancy measurement method employed. The columns "constant low" and "constant high" assume the model maintains a consistent low and high information density respectively across layers, serving as a benchmark for compression without accounting for redundancy.

6 Discussion

As discussed before, all methods perform worse with larger amounts of pruning. Weight magnitude appears ineffective as a heuristic for redundancy and is comparable to simply compressing the whole network. Alternatively, the causal tracing and gradient magnitude-based heuristics perform well.

Interestingly, casual tracing appears to perform best when using a combination of aggressive quantization with mild to moderate pruning. Other methods in comparison perform either the same or worse.

This occurs for numerous reasons. It's possible that this occurs due to variance in sampling for this benchmark. This method could be run on more datasets with more samples from each one, given more computational resources. It's also possible that this may occur due to an issue within the code, despite careful debugging and testing throughout the development process.

7 Conclusion

Causal tracing emerges as a promising approach for pinpointing the sections of an LLM with the highest information density for compression. Future direction for this project would be to investigate the limits of this form of quantisation approach, and explore the conditions under which it performs well. For instance, this paper used a simple approach to pruning and quantization at varying degrees. However, the framework described here is also compatible with more complicated pruning and quantization methods, such as SparseGPT and SmoothQuant. Future works could experiment with using these in conjunction with causal tracing.

Additionally, while GPT-2 is technically an LLM, it only has 125 million parameters. In order to make more concrete conclusions, we would ideally want to test this approach (with more computational resources) on a much larger model, such as the open source Llama models released by Meta.

References

Zhuocheng Gong, Jiahao Liu, Jingang Wang, Xunliang Cai, Dongyan Zhao, and Rui Yan. 2024. What makes quantization for large language models hard? an empirical study from the lens of perturbation. *Preprint*, arXiv:2403.06408.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli

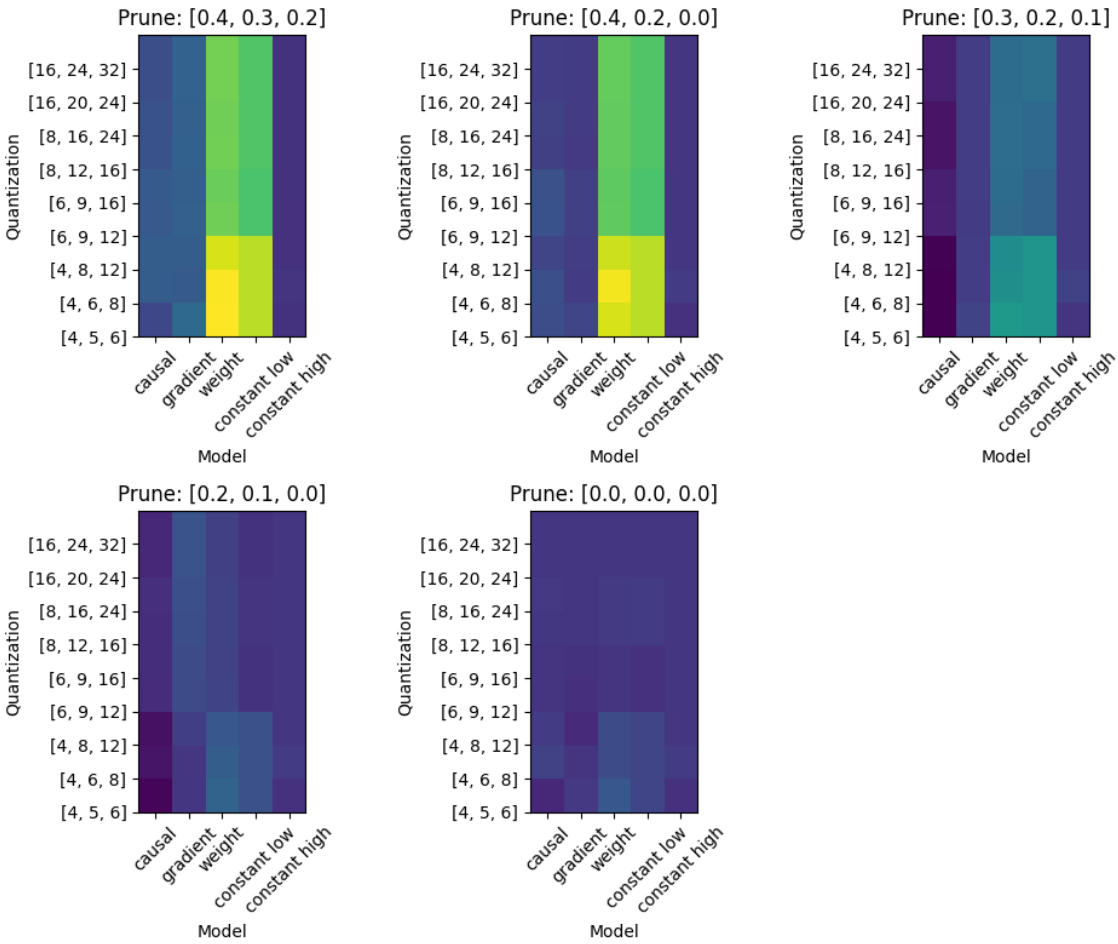

Figure 2: Heatmap for results - normalized such that the darkest blue tile corresponds to a perplexity of 5.552 and the brightest yellow tile corresponds to a perplexity of 13.275. Therefore, a darker implies a better score on the benchmark.

Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *CoRR*, abs/2102.00554.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2023. Locating and editing factual associations in gpt. *Preprint*, arXiv:2202.05262.