EFFICIENT TRANSFORMERS : POST TRAINING PRUNING AND QUANTIZATION FOR OPTIMAL LARGE MODEL INFERENCE

AYUSH GOEL, GOKUL NAIR, PRANAV GUNREDDY

ABSTRACT. This paper presents an approach to enhance transformer inference by employing post-training pruning (PTP) and quantization (PTQ). We explore how PT inference optimization methods can reduce computational complexity, memory footprint & inference latency and make things run faster while using less memory. We have presented results that demonstrate which layers to prune and quantize based on greedy search and information matrices. We have achieved substantial floating point operation reductions, latency and GPU memory footprint while maintaining competitive accuracy of BERT transformer model. This ensures effective and efficient deployment of transformer models for real life applications.

1. INTRODUCTION

Transformer models have emerged as a cornerstone in modern machine learning, permeating diverse fields such as Natural Language Processing, Computer Vision, and Speech Recognition. Their unique ability to capture intricate dependencies in sequential data has propelled them to the forefront of model architectures. Models like BERT, Vision Transformer (ViT), Segformer have set new benchmarks for performance in their fields. The versatility and effectiveness of transformers have made them indispensable across a spectrum of applications, underscoring their omnipresence in contemporary machine learning landscapes.

However, the widespread adoption of Transformer models comes with a set of formidable challenges, particularly during deployment and inference. The computational demands of transformers, especially in resource-constrained environments such as edge devices, pose significant hurdles. Heightened memory consumption and constraints on parallelization further compound the deployment issues. This becomes especially pronounced when deploying models like BERT or ViT on devices with limited computational resources. Navigating these challenges and optimizing the inference process for transformers is crucial to unlock their full potential in real-world applications. The forthcoming exploration of pruning and quantization techniques in this paper addresses these challenges, aiming to strike a delicate balance between computational efficiency and the preservation of model performance.

2. RELATED WORK

The field of transformer inference efficiency has garnered substantial attention, driven by the burgeoning size and computational demands of contemporary models. A plethora of research endeavors has been dedicated to addressing these challenges, with a particular emphasis on optimizing transformer inference. comprehensive review papers have examined entire inference pipeline, encompassing knowledge distillation, pruning, quantization, neural architecture search and related optimization techniques [1] [2].

In the realm of pruning, techniques are organized along several categories, such as weight, node, neuron, filter, head, and token pruning [3]. But primarily they can be divided into zero, first and second-order based on how they quantify the saliency of network parameters. AxFormer [4], OBERT [5] are examples. We also reviewed post-training pruning techniques and hardware-aware pruning techniques [6] [7] [8]

Similarly, Multiple approaches have been explored for Quantization of transformers. Quantization aware training (QAT) [9] [10] has been a popular choice among researchers. Although performance remains suboptimal, but due to ease in implementation and deployment, Post training quantization [11] [12] [13]is also being explored rigorously and has great scope for improvement.

Hugging face [15] and Pytorch [16] open source communities have had enormous contributions to research in this area and have APIs supporting most of the work cited above. As the community strives to strike a balance between model size and computational demands, these studies collectively contribute to the evolving understanding of transformer optimization for practical deployment.

3. BACKGROUND

3.1. **Structure of Transformer Models.** In our analysis, we concentrate on encoder-based Transformer models, specifically variants of the widely used BERT architecture. A typical Transformer encoder is composed of several layers, each layer consisting of a Multi-Head Self-Attention (MHSA) mechanism and a Position-wise Feed-Forward Network (PFFN). Formally, the MHSA operation in a layer is defined as:

$$\text{MHSA}(X) = \bigoplus_{k=1}^{K} \text{Attention}_k(X), \quad X_{\text{MHSA}} = \text{LayerNorm}(X + \text{MHSA}(X)), \tag{1}$$

where $\bigoplus$ denotes concatenation, K is the number of attention heads, Attention_k denotes the k^{th} attention mechanism, and X is the input.

The output of MHSA is passed to PFFN, consisting of a sequence of linear transformations:

$$\text{PFFN}(X) = \sum_{j=1}^{J} V_{:,j}\alpha(U_{j,:}X_{\text{MHSA}} + c_j) + d, \quad X_{\text{final}} = \text{LayerNorm}(X_{\text{MHSA}} + \text{PFFN}(X_{\text{MHSA}})), \tag{2}$$

where V, U are weight matrices, c, d are bias vectors, J is the number of filters, and α represents a non-linear activation function, typically GELU.

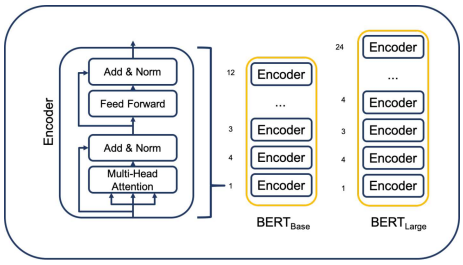

FIGURE 1. Bert Model framework

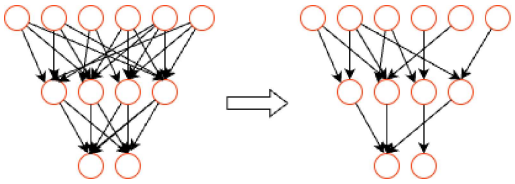

FIGURE 2. Pruning

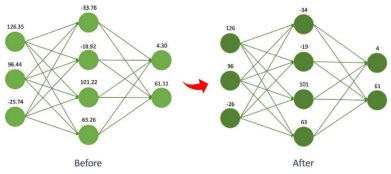

FIGURE 3. Quantization

3.2. **Pruning.** Pruning involves selectively removing unimportant weights or connections without retraining. It can be unstructured or structured, with structured pruning being more relevant for high sparsity in Transformer models. Our framework is designed to selectively prune the Transformer architecture while preserving its core functionalities. The aim is to reduce the number of parameters and operations without compromising the model's performance. We do this by strategically pruning certain elements within the MHSA and PFFN layers, ensuring the remaining architecture remains dense and computationally efficient.

We introduce an innovative approach to pruning that involves the application of adaptive masks to the Transformer's structure. These masks are designed to selectively deactivate certain components within the model's layers, effectively reducing the overall complexity while maintaining essential functionalities.

3.3. **Quantization.** Quantization refers to reducing the precision of the model parameters (weights and activations). This method reduces the precision of these numbers to lower bit widths, such as 16-bit or 8-bit integers. There are multiple ways to quantize a network while preserving the model performance. The general quantization methods can be classified into: static/ dynamic, uniform/ mixed precision, Post Training Quantization (PTQ) and Quantization-aware Training (QAT). Post-Training Quantization (PTQ) is often preferred for its simplicity inspite QAT performing better. In PTQ, a model initially trained with high-precision (e.g., floating-point) weights is converted to lower precision (e.g., integer or lower-bit floating point) after training. The transformation can be represented as:

$$W_{\text{quant}} = \text{Quantize}(W_{\text{float}}; Q_{\text{params}}), \tag{3}$$

where W_{quant} denotes the quantized weights, W_{float} are the original floating-point weights, and Q_{params} represents the quantization parameters such as scale and zero-point. This process aims to maintain the performance of the model while significantly reducing its size and computational requirements.

In our work, we implemented a greedy search algorithm to find out the most optimal combination of layers to quantize with accuracy and F1 score as the metric.

3.4. **Our work.** We have done a comprehensive literature review of various approaches to quantization and pruning of transformer models, especially post training methods due to their ease in usage and deployment. We have experimented with different ideas in relevant papers and inspired by them, came up with a competitive solution. This paper proposes a transformer inference optimization framework including post-training pruning and quantization methods tailored for Transformer models. We have demonstrated the results on BERT large base model finetuned on MRPC dataset and compared the performance against Intel's neural compressor on metrics such as accuracy drop in model, time taken and latency/inference speed up.

4. Approach

In this project, we explore two key methods for enhancing transformer models after training: Post-Training Pruning and Post-Training Quantization. Post-Training Pruning evaluates and fine-tunes specific model components, while Post-Training Quantization adjusts the precision of computations dynamically. Both techniques are vital for improving the model's efficiency and effectiveness

Post-Training Pruning.
Significance Assessment for Pruning: In this initial phase, the transformer model's components are evaluated for their relative importance using a low-rank approximation method. The process involves:

- Performing Singular Value Decomposition (SVD) on a weight matrix A of a layer: $A \approx U\Sigma V^T$.
- Retaining only the top k singular values for approximation: $A_k = U_k \Sigma_k V_k^T$.
- Calculating the reconstruction error for each component: $\text{Error}_i = \|A_i - A_{k,i}\|$.

This quantifies the impact of each component on the model's overall performance.

Optimization of Pruning Patterns: Upon identifying components of lower significance, the methodology optimizes the pruning patterns, maintaining efficiency and effectiveness of the pruned model while considering interdependencies within the model's layers.

Fine-tuning and Adjustment Post-Pruning: The final stage focuses on fine-tuning the pruned model to mitigate performance degradation, adjusting the remaining components to recover losses in accuracy or performance.

Post-Training Quantization.
Layer Selection for Quantization: The initial phase involved systematically exploring to identify the optimal combination of layers for quantization. This included testing all permutations of layer-wise quantization on a pre-trained BERT model to balance accuracy retention with a reduction in parameter size. This approach ensured the selection of the most impactful layers for quantization.

Application of Dynamic Quantization: After pinpointing the layers, dynamic quantization was applied to the selected layers of the pre-trained BERT model. The layers have been chosen from greedy search among available layers. This process involved dynamically converting the floating-point precision of weights to lower-bit representations during inference, reducing the model's size and enhancing inference speed.

$$W_{\text{quant}} = \text{round}\left(\frac{W_{\text{float}}}{S}\right) + Z \tag{4}$$

where W_{quant} represents the quantized weights, W_{float} are the original floating-point weights, S is the scale factor, and Z is the zero-point.

To evaluate the effectiveness of our optimization strategies, we conducted a comparative analysis involving three model variants: the pruned model, the quantized model, and a combination of both pruning and quantization. This analysis was geared towards understanding the individual and combined impacts of these techniques on the model's performance. We rigorously assessed each model variant in terms of accuracy, inference speed, and memory efficiency, providing a comprehensive view of the benefits and trade-offs associated with each approach.

5. Experimental Results

In this section, we present our findings from experiments conducted on BERT model finetuned on the MRPC dataset and tested on Question-Answering Pair (QQP) for pruning results. Our framework is implemented on top of PyTorch and the HuggingFace Transformers. Data preprocessing involved standard procedures like tokenization and normalization to ensure readiness for our analysis. Our focus was on crucial performance metrics: accuracy, model size, and parameter count. These metrics are essential in assessing the efficiency and efficacy of our models post-pruning and quantization, providing a comprehensive view of their performance in real-world scenarios.

These metrics were chosen to provide insights into the efficiency and accuracy of the models. Accuracy measures the model's performance in predictions, model size indicates the computational resource requirements, and the number of parameters reflects the model's complexity. The results are interpreted to understand the impact of our optimization techniques, including pruning and quantization, on the model's overall effectiveness and operational efficiency.

We compared our results with Intel's Neural compressor. (B) is baseline whose results are obtained by using the Neural Compressor and (O) is our approach.
* Pruning percentage by definition is different in Intel's package (sparsity ratio of modules) and our approach (latency or no of computations constraint) but we have chosen numbers in such a way that similar model size
* Time taken is when run on AWS g5.4xLarge instance (NVIDIA A100 Tensor Core GPUs, 24 GiB GPU memory, 16 vCPUs, 600 GiB CPU memory)

TABLE 1. Accuracy vs Pruning

Pruning Level	Accuracy (B)	Accuracy (O)(%)
0%	89.25%	89.25%
14%	88.5%	87.7%
30%	82.1%	79.8%

TABLE 2. Accuracy vs Quantization

Quantization Level	Accuracy(B)	Accuracy(O) (%)
No Quantization	89.25%	89.25%
8-bit	82%	80.1%
16-bit	88.9%	86.5%

TABLE 3. Accuracy vs Pruning + Quantization

Pruning and Quantization Level	Accuracy(B)	Accuracy(O)(%)
No Pruning + No Quantization	89.25%	89.25%
14% Pruning + 16-bit	86.8%	84.6%

Accuracy Analysis.

Inference time and speed up analysis.

TABLE 4. Model Inference time vs (Pruning, Quantization, Both)

Method	Time taken(B)	Time taken(O)
None	130s	130s
Pruning (22%) Only	93s	80s
Quantization Only(16 Bit)	91s	93s
Pruning + Quantization	105s	98s

TABLE 5. Size vs Pruning

Method	Model Size(B)	Model Size(O)
None	438 Mb	438 Mb
Pruning (22%) Only	290 Mb	278 Mb
Quantization Only(16 Bit)	230 Mb	242 Mb
Pruning + Quantization	155 Mb	171 Mb

Model Size analysis.

6. DISCUSSION

6.1. **Comparison with Intel's Neural Compressor.** Intel's Neural Compressor, known for its efficiency in neural network optimization, serves as a benchmark in our comparative analysis. We evaluated our model's performance in three aspects: pruning, quantization, and their combination.

Accuracy Analysis: Our model demonstrated a marginally higher accuracy drop on the MRPC dataset compared to Intel's Neural Compressor. The accuracy reduction was more pronounced in the combined pruning and quantization scenario, highlighting a trade-off between model size and accuracy.

Model Size Comparison: Regarding model size, our approach showed a slight increase compared to Intel's Neural Compressor, especially in the combined pruning and quantization scenarios. The size difference, although noticeable, remained within an acceptable margin. This suggests that our method maintains a competitive edge in terms of model size efficiency.

Inference Time Analysis: Our method demonstrated superior performance in terms of inference time compared to Intel's solution, particularly in scenarios involving both pruning and quantization. This marked improvement in operational efficiency highlights the effectiveness of our optimization strategy. Our approach, optimized for quicker processing, significantly reduced inference time, demonstrating its potential as a robust alternative to existing solutions.

Performance Across Different Scenarios: Our results indicate that while each method—pruning, quantization, and their combination—has its strengths, the combined approach presents a significant opportunity for balancing model efficiency and effectiveness, albeit with a marginal compromise in accuracy and inference time.

6.2. **Performance Interpretations & Challenges.** The observed variations in performance can be attributed to our model's specific focus on using a mask based approach while pruning. The possible explanation we could think of was that our mask based approach will only prune when interdependency of parameters in the mask is low while the Intel Model's sparsity based algorithm pruned taking into account the entire model. Also when we quantize, we have chosen only linear layers to quantize based on greedy search approach. Intel's library could have opted for a more optimal search to arrive at the best layers to quantize (mixed precision quantization, while we opted layer wise quantization). The enhanced latency and inference speed in our model could also be explained by the same.

One of the main challenges we faced in our approach was the high sensitivity of our model's performance to minor experimental adjustments. This led to a significant impact on results even with small changes in our experimental setup. Consequently, we had to restrict the scope of our experiments to manage this sensitivity and maintain the integrity of our performance evaluation. This limitation was a key consideration in our methodology and influenced our results and findings.

Overall, our approach offers a balanced solution in terms of accuracy and operational efficiency. While there is room for improvement in optimization time, the advantages in latency and inference speed make it a viable option in scenarios where quick data processing is essential. Future enhancements and resource allocation could further elevate its performance, aligning it more closely with industry leaders like Intel's Neural Compressor.

REFERENCES

[1] Krishna Teja Chitty-Venkata, Sparsh Mittal, Murali Emani, Venkatram Vishwanath, Arun K. Somani, A survey of techniques for optimizing transformer inference, Journal of Systems Architecture, Volume 144, 2023, 102990, ISSN 1383-7621, https://doi.org/10.1016/j.sysarc.2023.102990.

[2] Feng Tian, Hanwen Chang, Haihao Shen, and Suyue Chen, Intel® Neural Compressor, `https://github.com/intel/neural-compressor`, 2022

[3] F. Lagunas, E. Charlaix, V. Sanh, A.M. Rush, Block Pruning For Faster Transformers, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 10619–10629.

[4] A. Nagarajan, S. Sen, J.R. Stevens, A. Raghunathan, AxFormer: Accuracy-driven Approximation of Transformers for Faster, Smaller and more Accurate NLP Models, in: International Joint Conference on Neural Networks, IJCNN, 2022, pp. 1–8.

[5] E. Kurtic, D. Campos, T. Nguyen, E. Frantar, M. Kurtz, B. Fineran, M. Goin, D. Alistarh, The Optimal BERT Surgeon: Scalable and Accurate Second-Order Pruning for Large Language Models, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2022.

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

[7] Z. Liu, F. Li, G. Li, J. Cheng, EBERT: Efficient BERT Inference with Dynamic Structured Pruning, in: Findings of the Association for Computational Linguistics: ACL-IJCNLP, 2021, pp. 4814–4823.

[8] Fan H., Chau T., Venieris S.I., Lee R., Kouris A., Luk W., Lane N.D., Abdelfattah M.S. Adaptable butterfly accelerator for attention-based NNs via hardware and algorithm co-design 2022 55th IEEE/ACM International Symposium on Microarchitecture, MICRO, IEEE (2022), pp. 599-615

[9] O. Zafrir, G. Boudoukh, P. Izsak, & M. Wasserblat *Q8BERT: Quantized 8bit BERT* 2019 (https://arxiv.org/pdf/1910.06188.pdf).

[10] Mishra A., Latorre J.A., Pool J., Stosic D., Stosic D., Venkatesh G., Yu C., Micikevicius P. Accelerating sparse deep neural networks (2021) arXiv preprint arXiv:2104.08378

[11] Yuan Z., Xue C., Chen Y., Wu Q., Sun G. PTQ4ViT: Post-training quantization for vision transformers with twin uniform quantization European Conference on Computer Vision, Springer (2022), pp. 191-207

[12] Liu Z., Wang Y., Han K., Zhang W., Ma S., Gao W. Post-training quantization for vision transformer Adv. Neural Inf. Process. Syst., 34 (2021), pp. 28092-28103

[13] Elias Frantar. *GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers*. ICLR, 2023.

[14] J.Devlin, M. Chang, K. Lee and K. Toutanova *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* 2018

[15] HuggingFace Transformers (https://github.com/huggingface/transformers)

[16] Pytorch Quantization https://pytorch.org/blog/introduction-to-quantization-on-pytorch/