Enhancement of Grouped-Query Attention for Language Models

Luddy School of Informatics and Computing
Indiana University Bloomington
Sai Sena Chinnakonduru, Astarag Mohapatra, Nakul Sanjay Havaldar, Madhura Ashtekar
{saischin,astmohap,nahavald,mashteka}@iu.edu

Abstract— **The Transformer architecture forms the foundation of large language models. However, with increasing demands for scaling and constraints of hardware memory, the inference costs of these models remain high. To address these challenges, Multi Query Attention (MQA) Noam Shazeer et al [1] and Grouped Query Attention (GQA) [2] were introduced. While MQA uses only one key-value head, GQA uses an intermediate number of key and value heads by averaging adjacent key-value pairs, achieving Multi-Head Attention (MHA) level performance after uptraining for 5-10% of the original pretraining computation.**

In this paper, we propose a variation of Grouped-Query Attention, termed Weighted Grouped-Query Attention (WGQA). We introduce new learnable parameters for each key and value head in the decoder attention blocks, enabling the model to take a weighted average during uptraining. Our model demonstrates a performance slightly better than GQA close to traditional MHA with the same number of steps. We believe that the introduction of these parameters and subsequent finetuning inform the model about the grouping mechanism during training, thereby enhancing performance in fewer steps.

I. Introduction

At the core of the language models lies an autoregressive transformer model [3] that generates one token at a time based on the input sequence task and the previous sequence of the output's token it has generated so far. It is a sequential process and the workload is memory-bound [5]. As we scale up the model size, the inference cost will be expensive as we need to load the model to our GPU VRAM. The original transformer paper came out in 2017 [3], was trained on P100 GPUs with 5.3 TFLOPs double-precision performance and 16 GB of memory compared to the current best-performing GPU A100, which has 80GB of GPU memory and 9.8 TFLOPs. There has been a significant increase in the computation of the GPUs, with a modest increase in memory. In the ZeRO paper[4], the authors demonstrated that GPT-2 which has 1.5B parameters, required 3GB memory for its weights and it cannot be trained on a 32 GB memory due to the additional memory footprint of the activation and gradients. It will also raise challenges in full parameter fine-tuning of these models as the memory requirements increase exponentially [6].

The current state-of-the-art models have significantly higher parameters, which also increases the inference cost. According to a recent estimate, processing an LLM request can be 10× more expensive than a Google search query [7]. Due to the sequential nature of the autoregressive models, the workload needs to load the model to memory and store the KV heads based on the generated tokens so far. Also, some decoding techniques like beam search can consume additional memory space by storing the KV heads for different paths, and while discarding some paths, there will be issues with memory fragmentation due to the requirement of contiguous memory for decoding. Hence, to resolve the memory-bound workload, the authors in the paper on MQA and GQA suggested grouping the query heads and aggregating the key-value heads after pre-training, and after just fine-tuning with 5-10% of the pre-training steps, the performance converged with the Multi-Head Attention thus reducing the memory footprint. In this paper, we are proposing a parametric way of aggregating the key-value heads (WGQA) instead of the heuristic of taking the element-wise mean of the corresponding key and value heads. We show that the performance increase in WGQA is not random and the aggregated weight parameters are statistically different to 0.5, which corresponds to taking the mean of the key-value heads.

Also, we test our intuition on similarity-driven grouping of queries, where we re-arrange the query, key and value heads based on their similar twins from cosine similarity before grouping (SimGQA). The idea stems from the description that the query head transforms the input embeddings into projections that search for features to pay attention to, so grouping them would mobilize the searching mechanism and similar queries can learn from a common key and value head[8].

This paper is organized as follows: In Section 2, we provide an overview of the background. Section 3 describes our proposed WGQA and SimGQA implementation details followed by experiments in Section 4 and results in Section 5. Finally, we propose future research directions and conclusions.

II. Background

This work is focused on achieving better performance compared to GQA and MQA, which are similar to model-pruning methods, except that we aggregate the pruning layers. These kinds of work improve memory bandwidth

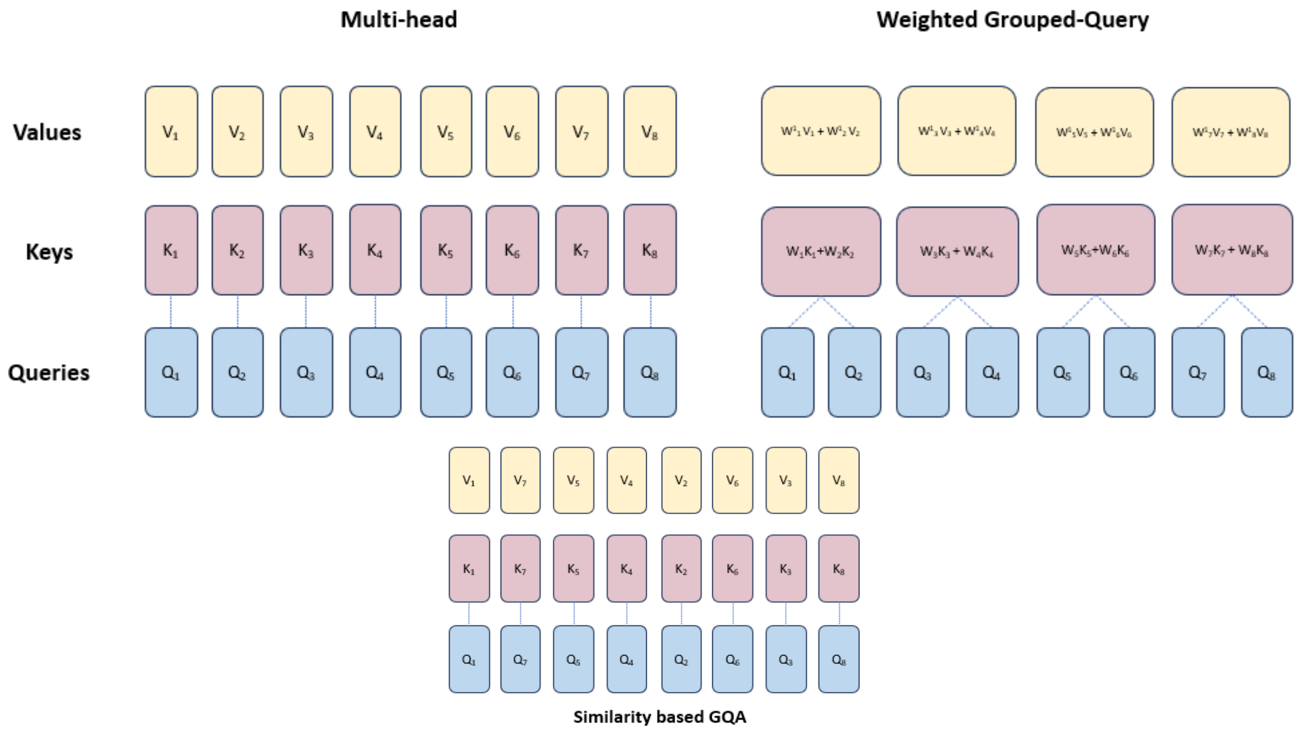

Fig. 1. Going in a clockwise direction from top to bottom (a) Vanilla Multi-Head Attention (MHA) with 8 heads, (b) Weighted Grouped -Query attention (WGQA), where for Grouped Query Attention (GQA) we have all the weights as 0.5, but for WGQA learn these weights. (c)Similarity-based GQA, where the heads are arranged according to similarity among queries and followed by the same implementation to GQA

and exploit the computational speed of GPUs. Pope et al., 2022 [9] showed that MQA is helpful for long inputs due to the reduced memory overhead.

There are other techniques for improving the memory bandwidth overhead from keys and values. Quantization (Dettmers et al.,2022 [10]; Frantar et al.,2022[11]) reduces the size of weights, activations, keys and values by shifting to INT8 or bfloat16 precision, instead of float32. The Student-teacher or model distillation techniques reduce the model size by pre-training a smaller model to replicate the performance of a bigger model (Hinton et al.,2015 [12]; Gouet al.,2021 [13]). In language models, the model distillation works by pre-training on the decoding outputs of a bigger model. There are other parameter-efficient fine-tuning (PeFT) techniques, LoRA (Edward et al., 2021 [14]) which decompose the projection heads via Singular Value Decomposition into a lower dimension and then compute the gradient steps, followed by composing the full-weight matrix again for gradient update. QLoRA (Dettmers et al.,2023 [15]) augmented LoRA by quantizing the static weight matrices, which further reduced the memory footprint. The PeFT techniques take inspiration from Li et al. (2018a) [16]; and Aghajanyan et al. (2020) [17] which show that the learned over-parametrized models reside on a low intrinsic dimension.

Finally, the uptraining procedure that the GQA paper proposed is inspired by Komatsuzakietal. (2022) [18], which uptrains standard T5 checkpoints into sparsely activated Mixture-of-Experts (MoE) models. Here the up-training procedure regains the performance of the base model.

III. Implementation

The attention module in the transformer architecture has three main components, query, key and value each with a dimension of (d, d), where d is the token embedding length. In Multi-head attention for h number of heads, the projection matrices have the dimension of $(d, \frac{d}{h})$, which transforms the input embeddings (n, d), where n is the sequence length of the input text, to h projections each of dimension $(d, \frac{d}{h})$, followed by concatenation to get the Q, K and V. Then the self-attention score is given by

$$score = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

For grouped query attention, we divide the query heads into G groups, which will reduce the number of key-value heads by a factor of $\frac{h}{G}$. Hence, the projection dimensions to obtain Q, K and V are (n, d, d), $(n, d\frac{G}{h}, d\frac{G}{h})$ and $(n, d\frac{G}{h}, d\frac{G}{h})$ respectively for a batch size of 1, which reduces the number of parameters by 33% for $G = 4$ in each attention module. The weighted GQA module adds extra scalar parameters for key-value heads for $(w_{1,k}, w_{2,k} ... w_{h,k})$ and $(w_{1,v}, w_{2,v} ... w_{h,v})$, additional $2h$ parameters for each attention module. These learnable parameters are multiplied with the key and value heads as shown in Figure 1(b), hence instead of the mean heuristic, we have a weighted average of the learnable key-value projections. This adds no additional overhead during inference, as we scale the key-value heads after the fine-tuning process.

The similarity-based implementation Figure 1(c), rearranges the query, key and value heads based on the similarity of queries, and then does Grouped-Query attention. We have three variants of implementation,(I) re-arrange once before fine-tuning, (II) re-arrange at frequent intervals of 200 steps, (III) re-arrange after longer intervals of 1000 steps. The implementation details stay the same for cross-attention in the T5 architecture, where the key-value heads come from the encoder final layer and the query from the already generated tokens. Our implementation is open-source GitHub code and we are also providing all our training runs and report on Weights and Biases.

IV. Experiments

A. Configuration

All models are based on the T5-small architecture implemented using Hugging Face transformers. All our models are initialized with T5-small pre-trained weights and uptrained using AdamW optimizer with 0.001 initial learning rate and scheduled linear decay. Key-value head grouping is only applied to decoder self-attention and cross-attention blocks, as mentioned in the original paper [2].

B. Data and Fine-tuning

We finetuned and evaluated our models using the CNN/Daily Mail dataset. We did not evaluate on other datasets like WMT 2014(English to German translation task) and TriviaQA (question-answering dataset) due to limited GPU resources. We trained all of our models for 3 epochs with a batch size of 32 using 1 NVIDIA V100 GPU which took approximately 5 hours. We used an input length of 512 and an output length of 256 for the summarization task.

C. Different Models

As we don't have benchmark performance on T5-small using GQA and MQA, we also ran those experiments. Below are the different experiments that we ran:

i **Weighted Grouped-Query Attention:** In this approach, new parameters for each key and value head in the decoder's attention blocks are initialized with a value of 0.5. A weighted sum is then taken, allowing the model to learn these parameters during fine-tuning.

ii **Grouped-Query Attention**: In GQA, key and value heads in the decoder's attention blocks are mean pooled to form G groups, which are then fine-tuned.

iii **Multi-Query Attention**: MQA involves mean pooling all key-value heads in the decoder's attention blocks to form a single key-value head that is shared across all query heads.

iv **Similarity-based Grouped-Query Attention**: SGQA rearranges all heads in the decoder's attention blocks based on cosine similarity (calculated by flattening each head). The rearranged heads are then mean pooled and fine-tuned. Also, we did experiments with interval-based similarity re-arrangement with 200 and 1000 steps.

v **Random Weighted Grouped-Query Attention**: RWGQA differs from WGQA by initializing new parameters for each key and value heads in decoder attention blocks randomly from standard normal distribution, rather than initializing with 0.5. After initialization, a mean pool is taken, followed by fine-tuning

V. Results

Model	**Rouge-1**	**Rouge2**	**RougeL**
MHA	41.12	19.56	38.35
MQA	38.79	17.46	28.04
GQA	40.31	18.35	28.68
WGQA	**40.38**	**18.36**	**28.76**
SIMGQA	38.87	17.43	28.11
RandWGQA	39.64	17.86	28.42
Long Interval SIMGQA	15.55	2.09	12.28
Short Interval SIMGQA	15.34	1.19	11.98

TABLE I
Performance comparison of different attention mechanisms using ROUGE scores

In Table 1, we present the comparative performance of various models evaluated after three epochs. The table enumerates the Rouge-1, Rouge-2, and Rouge-L scores for each model. These metrics respectively quantify the overlap of unigrams, bigrams, and the longest common subsequence between the generated summaries and the reference summaries. We observe that WGQA achieved slightly better performance than GQA. Also, around the conclusion of the epoch 2 out of 3, WGQA achieved a Rouge score similar to the score achieved by GQA at the end of the third epoch. Hence, it shows that WGQA is more efficient than GQA and can reduce the computational cost in uptraining by achieving similar performance in less number of steps. For T5-small, the number of extra parameters in WGQA and RandomWGQA is 192 (8 for each attention/cross-attention module), and its computational complexity remains the same over the training runs.

In our results, we observed that Similarity-based GQA models did not perform well. We believe this may be attributed to the rearrangement of query and key-value heads, which potentially manipulates attention scores. This alteration in attention scores could change the focus of attention from specific words, impacting the model's performance. Also, enforcing similarity at frequent intervals and infrequent intervals did not help to improve the performance, rather they performed poorly on our benchmarks. We speculate that the re-arrangement of

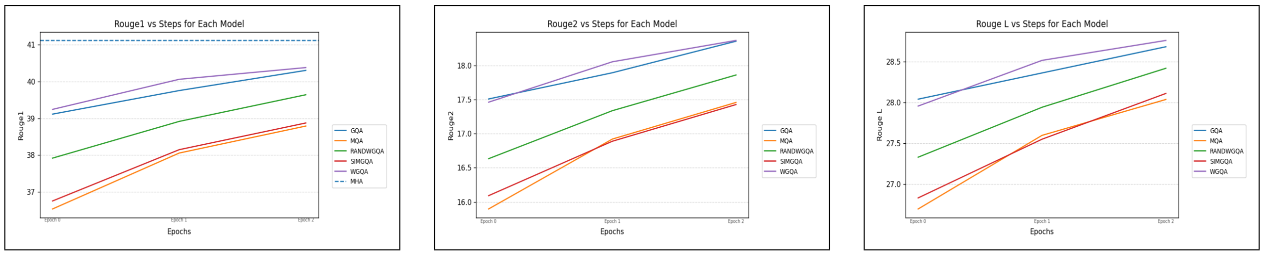

Fig. 2. Plot showing Rouge-1, Rouge-2 and Rouge-L scores over epochs for different attention mechanisms

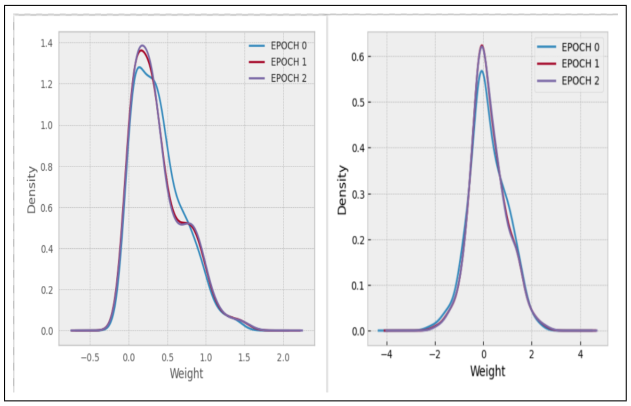

Fig. 3. The distribution of the weight scalar values for all the attention modules in the T5-small transformer over the epochs for WGQA (LEFT) and RandomWGQA (RIGHT). In the later, the weights are initialized with $\mu = 0$ and $\sigma = 1$.

projection heads in the decoder is not favourable for generation during finetuning.

A. *Significance test*

In Figure 3., the weight distribution shows that the mode of the weight vectors for all the KV-heads is around 0.2, instead of 0.5. In a two-sided test for the last epoch and its steps, for statistical significance, we have the null hypothesis as the mean of scalar weight values is equal to 0.5, and the alternative is that the mean is not equal to 0.5. The p-value ranges from $3.78e{-}6$ for the first step in the last epoch to $7.51e{-}6$ for the last epoch, hence for significance value $\alpha = 0.05$, we can reject the null hypothesis. Thus, the improved performance is not random and the results are statistically significant for the enhancement over the mean heuristic. Similarly, for random Gaussian initialization, the weights centred around 0. The WGQA performed better than RandomWGQA for all our performance metrics, which suggests that the initialization of 0.5 is a good starting point for the aggregation of KV-projection heads.

VI. Future Work

We would like to propose future research endeavours for our paper based on our extensive experimentation. The GQA paper only suggested grouping in the cross-attention and decoder layers without giving a reason why they did not do it for encoder layers. The similarity-based GQA will be interesting for encoder layers as the model compresses the information from text in these layers, hence enforcing similarity in these layers can be explored further.

In the LoRA paper, the authors proposed that we can fine-tune multi-task models, and then hot-swap the adapters based on the requested task. The learned lower dimension parameter with LoRA for the summarization task would be different to the translation task, hence you can just swap the adapters based on the task. The weights learned in the WGQA can be task-specific, hence we can load the GQA model once to our memory and we can scale the Key-value heads on demand to have a Mixture-of-Experts (MoE) model for different tasks. This will avoid loading different fine-tuned models for different tasks, and we can exploit the faster computational speed of GPUs to scale KV heads in real-time to serve different models.

Due to a limited computational budget, we ran our experiments on T5-small. We know that pruning bigger models, which are overparametrized, will lead to less loss in information as compared to a smaller model. In the GQA paper, they tested for T5-XXL, which is an 11 B parameter model compared to T5-small which is only a 60.4 M parameter model. Hence, we can run the same set of experiments for bigger models to compare the performance. Also, we can remove some redundant queries if two queries are nearly similar with similar gradient updates, which will save up more memory for the model weights

VII. Conclusion

This paper focused on improving the performance of the GQA algorithms, that reduce the memory footprint of the key and value projection heads. These optimizations will improve the performance and are important in generating longer sequences. Due to the limited computational budget, we restricted ourselves to T5-small architecture and for the summarization dataset of CNN/Daily Mail, but we have made the grouped-query attention algorithm more parametric and data-dependent, which will certainly improve the dynamic adaptation of our algorithm to other tasks. Also, we tested our intuition of similarity based on query grouping, which can certainly be explored further for explainability by probing similar queries.

VIII. Supplementary Materials

- GitHub code implementation
- Weights and Biases training runs
- Weights and Biases report

References

[1] N. Shazeer, 'Fast Transformer Decoding: One Write-Head is All You Need', arXiv [cs.NE]. 2019.

[2] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, 'GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints', arXiv [cs.CL]. 2023.

[3] A. Vaswani et al., 'Attention is all you need', Advances in neural information processing systems, vol. 30, 2017.

[4] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, 'ZeRO: Memory Optimizations Toward Training Trillion Parameter Models', arXiv [cs.LG]. 2020.

[5] W. Kwon et al., 'Efficient Memory Management for Large Language Model Serving with PagedAttention', arXiv [cs.LG]. 2023.

[6] K. Lv, Y. Yang, T. Liu, Q. Gao, Q. Guo, and X. Qiu, 'Full Parameter Fine-tuning for Large Language Models with Limited Resources', arXiv [cs.CL]. 2023.

[7] Focus: For tech giants, AI like Bing and Bard poses billion-dollar search problem Reuters.2023.

[8] Andrej Karpathy Let's build GPT: from scratch, in code, spelled out.

[9] R. Pope et al., 'Efficiently Scaling Transformer Inference', arXiv [cs.LG]. 2022.

[10] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, 'Llm.int8(): 8-bit Matrix Multiplication for Transformers at Scale', arXiv [cs.LG]. 2022.

[11] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, 'GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers', arXiv [cs.LG]. 2023.

[12]G. Hinton, O. Vinyals, and J. Dean, 'Distilling the Knowledge in a Neural Network', arXiv [stat.ML]. 2015.

[13]J. Gou, B. Yu, S. J. Maybank, and D. Tao, 'Knowledge Distillation: A Survey', International Journal of Computer Vision, vol. 129, no. 6, pp. 1789–1819, Mar. 2021.

[14] E. J. Hu et al., 'LoRA: Low-Rank Adaptation of Large Language Models', arXiv [cs.CL]. 2021.

[15] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, 'QLoRA: Efficient Finetuning of Quantized LLMs', arXiv [cs.LG]. 2023.

[16] V. Lialin, V. Deshpande, and A. Rumshisky, 'Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning', arXiv [cs.CL]. 2023.

[17] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, 'Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning', arXiv [cs.LG]. 2020.

[18] A. Komatsuzaki et al., 'Sparse Upcycling: Training Mixture-of-Experts from Dense Checkpoints', arXiv [cs.LG]. 2023.