Exploring Speed/Quality Trade-offs in Dimensionality of Attention Mechanism

Optimization with Grouped Query Attention and Diverse Key-Query-Value Dimensionalities

Khalit Gulamov

Supervisor(s): Prof.dr. Arie van Deursen, Prof.dr. Maliheh Izadi, Aral de Moor

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Khalit Gulamov
Final project course: CSE3000 Research Project
Thesis committee: Prof.dr. Thomas Abeel, Prof.dr. Arie van Deursen, Prof.dr. Maliheh Izadi, Aral de Moor

Abstract

The advent of transformer architectures revolutionized natural language processing, particularly with the popularity of decoder-only transformers for text generation tasks like GPT models. However, the autoregressive nature of these models challenges their inference speed, crucial for real-time applications and resource-constrained environments. Memory bandwidth is a significant bottleneck, especially in autoregressive decoding, where constant loading of large key and value tensors dominates. The Multi-Query Attention (MQA) architecture was proposed to reduce memory access by shrinking the key-value cache, enhancing inference speed at the cost of generation quality. Grouped-Query Attention (GQA) was introduced to mitigate this quality decline, serving as an interpolation between Multi-Head Attention (MHA) and MQA. We explore the trade-offs between inference speed and quality in decoder-only models by experimenting with various proportions of query groups relative to attention heads during pre-training. Additionally, we investigate the impact of reducing the size of key and value vectors compared to GQA and explore a hybrid method combining GQA with shortened key-value vectors. This study aims to expand the list of possible trade-offs and help select an optimal architecture based on specific needs.

1 Introduction

The invention of the transformer architecture by Vaswani et al. [11] led to the rise of various language models. Decoder-only transformers, used for text generation, are exceptionally popular (as a notable example - GPT models [3] used in ChatGPT[1]). One challenge researchers have faced with such models is the speed of the inference (output generation), which is influenced by their auto-regressive nature. Solving this bottleneck is essential, whether aiming to provide a good user experience in real-time applications, saving on computational costs when running the model, or allowing the model to run in an environment with limited computational and memory resources.

In particular, memory bandwidth is a significant limitation in modern highly parallelizable hardware [9, 10, 17]. In auto-regressive (incremental) decoding, as each token in a sequence is generated one at a time and depends on all of the previous ones, most bandwidth is consumed by the constant loading of large key and value tensors in the attention mechanism (KV cache) [9, 10]. To address this issue, Multi-Query Attention (MQA) architecture was proposed with the goal of reducing the amount of memory access by making the KV cache smaller [10] by sharing keys and values across all the heads. It was shown to increase the inference speed significantly though the generation quality was slightly diminished [10].

However, Ainslie et al. [1] argue that MQA quality declines as the number of parameters and attention heads increases due to significant capacity reduction. To address this, Grouped-Query Attention (GQA) was introduced as an interpolation between standard Multi-Head Attention (MHA) and MQA. This architecture maintains a consistent cut proportion (the ratio of groups to attention heads) across differently-sized models. In our paper, we extend the work of Ainslie et al. [1] by examining previously unexplored properties of GQA through pre-training models with GQA to better understand its impact on performance. We assess both inference speed and quality across different numbers of groups and attention heads to explore the trade-offs, enabling more effective future applications of GQA. Unlike the original paper, we conduct our assessment on increasingly popular decoder-only models, as the encoder component of encoder-decoder models gains minimal benefit from GQA due to the absence of autoregression [10].

Furthermore, in the original transformer architecture, the dimensionality of key, query, and value vectors (further referred to as kqv vectors) was assumed to be equivalent to the embedding size ($d_k = d_q = d_v = d_{model}$) [11]. Shazeer [10] proposed an orthogonal approach to speeding up inference and reducing the KV cache by decreasing the dimensionality of kqv vectors (such models will further be referred to as KQV-models). While its performance was tested against MQA, it is unclear whether it will yield similar results compared to GQA models, as the reduction of kqv vectors will be less to match GQA. Additionally, we explore combining GQA with reduced kqv vectors to determine if this hybrid approach offers more advantages than either method alone.

This research aims to understand potential trade-offs better and help select an optimal architecture based on specific needs. Our main contributions are as follows:

- Introduced an approach for GQA that allows for more possible group configurations;
- Examined relationships between speed and quality changes in GQA;
- Compared the performance of KQV models with GQA models across various group configurations;
- Introduced a combined approach to further expand the trade-off search space;
- A replication package[2] for reproducing our findings, and our models[3] published on HuggingFace.

2 Background and Related Work

2.1 Preliminaries

Our study explores different speed/quality trade-offs for methods affecting the attention's mechanism dimensionality. This section covers the background information necessary to understand the upcoming sections.

Multi-Head Attention (MHA)
Conceptually, each attention head independently applies learnable projection matrices ($W_k^{h_i}$, $W_q^{h_i}$, $W_v^{h_i}$) to transform token embeddings into key, query, and value representations.

[1] https://chatgpt.com/

[2] https://github.com/AISE-TUDelft/tiny-transformers

[3] https://huggingface.co/collections/AISE-TUDelft/brp-tiny-transformers-666c352b3b570f44d7d2a519

These transformations are computed separately for each attention head (h_i), which then computes its own attention output using all previous keys and values. The outputs from all heads are concatenated and further transformed using the output projection matrix W_O.

In practice, matrices $W_k^{h_i}$, $W_q^{h_i}$, $W_v^{h_i}$ from each head are concatenated into single matrices W_k, W_q, W_v for computational efficiency. This consolidation allows each token to be initially transformed into larger key/query vectors of size d_k and value vectors of size d_v. These larger projected vectors are then partitioned into smaller subspaces specific to each attention head. Each partitioned vector has dimensions $d_k^h = \frac{d_k}{num\_heads}$ for keys and queries, and $d_v^h = \frac{d_v}{num\_heads}$ for values (where h denotes the vector belongs to a particular attention head's subspace).

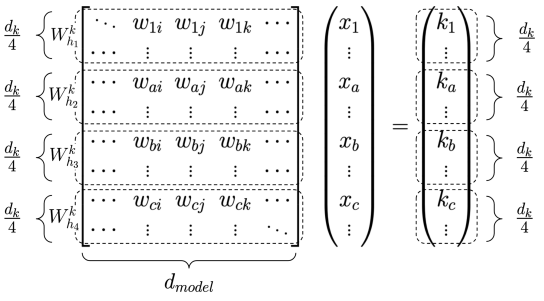

Figure 1: Obtaining a key from an embedding vector (for one sequence) in a model with 4 attention heads. d_{model} is the embedding dimension of each token (dimension of the vector x); d_k is the dimensionality of a key, which in this figure is equal to d_{model}; $\frac{d_k}{4} = d_k^h$; letters i, j, k, a, b, c are random letters used for illustrating weights.

Next, after calculating and distributing key and value vectors among attention heads, they are added to the KV cache for reuse in computing subsequent attention outputs. This allows each subsequent token to attend to all previous tokens within the context window size, ensuring efficient memory usage while maintaining context for generating coherent outputs.

Once the attention outputs are calculated for each head, the results are concatenated alongside the head dimension before being passed to be transformed by the output projection matrix, which is the final step in the mechanism. The exact formulas were omitted for conciseness but can be found in the paper by Vaswani et al. [11].

Inference Bottlenecks in Multi-Head Attention

The KV cache stores previous keys and values for all past tokens, which can consume significant memory, especially with multiple attention heads and parallel input processing. In incremental (autoregressive) decoding, the attention mechanism is called repeatedly for token generation, requiring constant fetching of the KV cache to compute attention outputs. This frequent memory access is a primary bottleneck causing inference slowdowns [9, 10, 17]. According to Shazeer [10], reducing memory usage, particularly from the KV cache, while increasing computations helps to get rid of this bottleneck [10]. Another strategy for speedup involves decreasing memory and computational demands while maintaining a balanced ratio. To improve inference speed per token, Shazeer recommends increasing parallel processing of input sequences and reducing the KV cache [10]. This reduction can be achieved by selectively omitting stored keys and values for specific attention heads using techniques like Multi-Query Attention (MQA) or Grouped-Query Attention (GQA) (described in Subsections 2.1 and **??** respectively); removing some heads entirely; or reducing the dimensionality of key and value vectors.

Group and Multi Query Attentions (GQA and MQA)

Multi-Query Attention was introduced by Shazeer [10] and was an initial step to optimizing the inference speed by dismissing keys and values for certain attention heads. Namely, in this approach, keys and values in all heads but one are omitted such that queries from all heads attend to keys and values from only one head. That means that W^k and W^v are reduced to just $W_{h_1}^k$ and $W_{h_1}^v$ respectively (*Figure 2*). At the same time, W^q remains the same as depicted in *Figure 1* except that the matrix is for query projections.

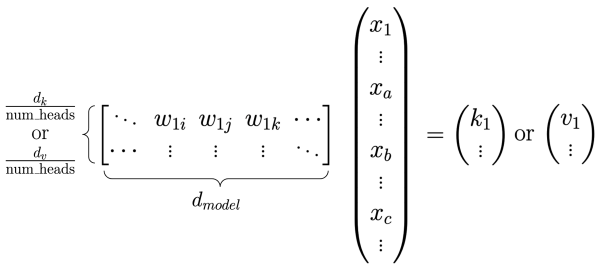

Figure 2: Obtaining a key OR value from an embedding vector (for one sequence) in MQA with an arbitrary number of attention heads. d_{model} stands for an embedding dimension (dimension of the vector x); d_k is a dimensionality of key vectors, d_v - of value vectors; i,j,k are random letters to illustrate the weights.

As a result, each query vector from each query head attends to key vectors from a single key head multiplied by vectors belonging to a single value head (*Figure 3*). This is equivalent to shrinking the KV cache size by a factor equal to the total number of heads num_heads.

MQA has shown a significant inference speedup when having a large batch size. However, it led to some quality degradation [10]. Moreover, MQA might cause a too-aggressive cut when having bigger models with a large number of attention heads as the proportion $\frac{1}{num\_heads}$ is too small [1]. To address this issue, Grouped-Query Attention was introduced by Ainslie et al. [1]. GQA is a generalization over MQA (interpolation between MHA and MQA, including both) that instead of omitting all the key-value heads one omits an arbitrary number of them. The ratio of this exact number to the total number of heads will be referred to as group proportion. This ability to select the number of key-value heads allows for flexibility in balancing the trade-off between the model's speed and quality. Figure 3 depicts differences between MHA, MQA, and GQA.

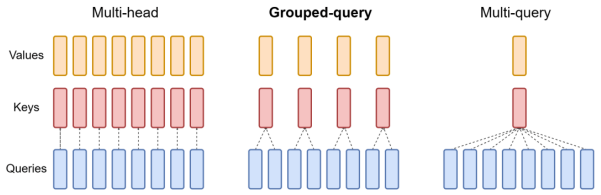

Figure 3: MHA, MQA, and GQA comparison. Taken from Ainslie et al. [1]

2.2 Small Language Models

Transformer models have scaled enormously, reaching hundreds of billions of parameters to enhance performance. However, this scaling has increased computational demands, making training and operation more costly and less sustainable. Eldan and Li [5] highlight that smaller language models (SLMs) with fewer than 10 million parameters can match or exceed the language understanding of larger models like GPT-2 (125 million parameters) when trained on well-curated datasets. Their TinyStories dataset, composed of bedtime stories for young children, illustrates this point by demonstrating models' coherent and diverse language generation and even showing some reasoning abilities [5]. The dataset's cohesiveness within a restricted domain (reduced breadth) allows small models to achieve comparable performance at a significantly reduced training cost.

Furthermore, Eldan and Li suggest that TinyStories and small models can be ideal for exploring and testing architectural decisions, as insights often carry over to larger-scale models [5]. In our experiments, we utilized GPT-NEO[4] models configured with approximately 8 million parameters and trained them on the TinyStories dataset.

2.3 Orthogonal Existing Approaches

There are a few other approaches that can be adapted to optimize the inference speed:

Attention Head Pruning: Removing attention heads is a viable approach to reduce the Key-Value (KV) cache size [10]. Furthermore, this reduction in attention heads also decreases computational consumption. Michel et al. [8] propose a method to prune attention heads selectively, targeting those less utilized by the model. This pruning strategy improves inference speed while maintaining model quality effectively.

Model Quantization: Model quantization is a method that reduces the precision of numerical values representing a neural network's parameters (e.g., using 8 bits instead of 32), which has been demonstrated to improve inference speed by reducing memory usage [4].

Model Distillation: Model distillation, as introduced by Hinton [6], transfers knowledge from large models to smaller ones, ensuring preserved quality. When applied to transformers, deploying the smaller model for inference can significantly increase the speed.

Model Partitioning: Another way to enhance inference efficiency involves optimizing hardware utilization. Pope et al. [9] propose partitioning models across multiple accelerator chips to mitigate memory limitations.

While other strategies exist for optimizing inference, this paper specifically explores methods that reduce attention dimensionality, such as Grouped Query Attention (GQA) and dimensionality reduction of kqv vectors.

3 Approach

3.1 Reducing Key and Value Dimensions

To reduce the dimensions of key and value vectors, the linear transformation matrices W_k, W_q, and W_v (Figure 1) were to have fewer rows. After downsizing, the matrices yield smaller vectors that are still split by the number of heads, resulting in smaller split vectors leading to smaller KV cache size. The multiplicative factor (lying in between (0, 1]) by which we reduce the matrices will be referred to as KQV proportion. Next, the standard Multi-Head Attention (Subsection 2.1) was utilized to compute the attention outputs. The output projection is adjusted accordingly for new smaller vectors to return them back to an embedding dimension.

3.2 GQA at the Pre-training

To better assess the impact of GQA on the models' quality, we adapt it during pre-training. Similarly to the approach described in Subsection 3.1, the key and value projection dimensions were reduced by a multiplicative factor called group proportion. The only difference at this stage is that the query projection matrix remains the same.

In GQA, a notable distinction from KQV models is that the size of split vectors among heads remains consistent with that of an MHA model, resulting in fewer vectors of size equal to those of the standard attention mechanism.

The original study assumed that the number of queries (or in other words, the number of newly obtained key-value heads) in each group should be equivalent. Then, the total number of attention heads should be divisible by the number of groups to be able to construct them. We omit this restriction, and instead, the queries were grouped so that the size difference among the groups was at most 1 query-head (Figure 6). This approach allows the experiment on a larger number of group factors while having a relatively small number of attention heads.

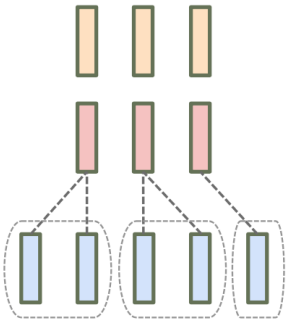

Figure 6: Odd-sized GQA to Adapt for Small Model Sizes. Inspired by Ainslie et al. [1]

[4]https://github.com/EleutherAI/gpt-neo

3.3 Widening the FFN

Both applying GQA and reducing the size of key and value vectors (as described in Subsections 3.2 and 3.1 respectively) results in fewer rows in transformation matrices (e.g., Figure 2). Effectively, this reduces the number of the model's learnable parameters. To maintain consistency in parameter comparison across models and focus solely on the impact of these methods on memory consumption of the KV cache, this parameter reduction needs to be compensated for. As proposed by Shazeer [10], we widen the FFN to achieve this. Below are the derived formulas (omitting the derivation for brevity); the first is for GQA, and the second is for KQV models:

$$width_{gqa\_new} = (width_{old} + (d_k - (d_k \cdot g\_p))$$

$$width_{kqv\_new} = (width_{old} + 2 \cdot d_{model} \cdot (1 - k\_p))$$

where g_p and k_p are Group and KQV proportions, respectively.

3.4 Ensuring Memory Bottleneck

Given the small size of the models, the KV cache size could potentially be less of a bottleneck given naturally smaller context window lengths and sizes of keys and values that the cache grows proportionally with. This could prevent the optimization techniques from increasing the inference speed as intended. To compensate for this, the memory usage was artificially inflated by processing many inputs in parallel (larger batch size) during the inference speed tests. In Section 5 we present findings for both large and small batch sizes to evaluate their impact on token generation time and the speed enhancements achieved by the optimization techniques.

4 Experimental Setup

We conduct experiments on smaller variations of GPT-NEO [2], pre-trained on the TinyStories dataset [5] as motivated in 2.2. We assess model quality in terms of natural language understanding via the BabyLM [14] pipeline. This evaluation pipeline consists of performance metrics such as GLUE [13], SuperGLUE [12], and BLiMP [16]. The speed of inference was assessed in separate experiments using the 'time' module available in Python.

This section introduces the research questions and lists all the details about the experimental setup used to answer them, including the dataset, models' configuration, and evaluation metrics. In addition, the results of the experiments are presented.

4.1 Research Questions

This section presents the research questions of our study. RQ1 addresses the gaps in the paper presented by Ainslie et al.; RQ2 examines how the approach of reducing kqv sizes compares to GQA (while Shazeer [10] only compared it to MQA); RQ3 proposes the combination of both methods to explore how it performs against using each method individually.

The independent variables are the hyperparameters affecting the dimensionality of the attention mechanism (e.g., group proportions - ratios of key-value heads to total heads in GQA; or KQV proportions - reduction factors for kqv sizes), while the dependent variables are quality (natural language understanding) and speed measured as described in Subsection 4.5.

Below we present the questions:

RQ1 How do the total number of attention heads and the ratio of key-value heads to total (query) heads affect the performance of GQA models? Specifically:

RQ1.1 **Does model quality decrease proportionally to the inference speedup gained from reducing the ratio of key-value heads to query heads (GQA), while keeping the total number of attention heads fixed?**

Even though Ainslie et al. [1] have discovered that the inference speed increments monotonically along with reducing the ratio of key-value heads to query heads (group proportion), it has not been investigated how the quality is affected. This knowledge is essential for selecting a group proportion. To address this, we conduct experiments with several such ratios for a fixed number of total heads and measure the quality and speed changes relative to the baseline.

RQ1.2 **How does the total number of attention heads affect the quality deterioration in GQA models when the ratio of key-value heads to query heads is fixed?**

Another gap that has not been addressed by Ainslie et al.[1] is whether keeping the same group proportion (ratio) among the models with a different number of attention heads results in similar quality declines. Investigating this would help better understand which proportion to choose when pretraining a model, given a certain number of heads. To explore this, we select one group proportion for GQA models with 3 configurations for attention heads and measure the quality decline relative to their corresponding baselines.

RQ2 **How do models with reduced *kqv* sizes compare to corresponding GQA models that lead to an equivalent KV-cache reduction in terms of quality and inference speed?**

Shazeer [10] demonstrated that reducing KQV by a factor equivalent to the KV cache reduction in MQA leads to worse quality than MQA. However, it's unclear if this holds for various group proportions of GQA. For this, for each GQA model from RQ1, we trained a corresponding model with reduced kqv sizes and compared their quality and speedups.

RQ3 **How does a hybrid approach combining GQA and reduced *kqv* sizes perform in terms of quality and inference speed?**

Depending on the models' usage and environment, applying one approach (GQA) or another (reduced kqv sizes) can be more desirable. To unlock more flexibility, we propose combining these approaches to widen the spectrum of possible trade-offs. To achieve this, we select one model with both GQA applied and kqv sizes

reduced and see how the speedup and quality compare to the corresponding (leading to the same KV-cache cut) models using either just GQA or smaller kqv vectors.

All the exact model configurations used to answer each of these research questions are presented in detail in Subsection 4.4, while hyperparameters that are common across all models can be found in Subsection 4.3.

4.2 Dataset[5]

The TinyStories dataset is a dataset of short, grammatically correct bedtime stories for children, with a total of 2.1 million samples and an average of 175.4 words per story. The dataset is generated by GPT-3.5 and GPT-4 [5]. As the dataset does not require models to learn much of domain knowledge, smaller models trained on it can perform as well as, or even better than, much larger counterparts (e.g., the 125-million parameter GPT-2 model) on the natural language understanding tasks described in Subsection 4.5. The tokenizer used for the dataset tokenization had a vocabulary size of 10,000. Its full configuration can be found in our replication package[3].

4.3 Models' Common Configuration

All the models trained to answer the research questions presented in Subsection 4.1 are based on GPT-NEO architecture implemented in the *Hugging Face* 'transformers' library. We change the hyperparameters and the architecture (to implement GQA and reduce kqv sizes) for each research question to match the RQ's goal. All of the models (for all the RQs) have constant $vocab\_size = 10000$ (number of tokens in vocabulary), $max\_position\_embeddings = 512$ (maximum sequence length), $window\_size = 256$ (for local attention). Moreover, all the models have 3 transformer blocks with alternating $attention\_types$ starting from global (i.e., our models have the types set to ["global," "local," "global"]). In addition, all the models were set to have a $hidden\_size = 384$ and a total number of parameters of ~ 8 million.

All of the models are pre-trained for 1 epoch with a learning rate of 0.001, training batch size of 16, and 16 gradient accumulation steps.

4.4 Research Question-Specific Configurations

To answer RQ1.1, we train 5 models with 8 attention heads such that the ratio of key-value heads to total heads (group proportion) equals 1.0 (MHA, baseline), 0.75, 0.5, 0.25, and 0.125 (MQA). That leads to having 8, 6, 4, 2, and 1 key-value heads, respectively.

For RQ1.2, we select a group proportion of 0.75 and train two additional models—with 4 heads and 16 heads—and separate baselines with 4 and 16 heads for each of these, respectively. In addition to these 4 models, from RQ1.1, we take the model with a 0.75 group proportion and 8 heads and its corresponding baseline. Thus, we obtain performance results for a total of 6 models.

[5]Our research group (partially) shares the experimental setup. It was provided by the supervisor and extended by each member to work with their custom models; as such, parts of writing are also shared among the group members.

Addressing RQ2, for each group proportion selected for RQ1.1 (except the baseline), we train 4 corresponding (leading to the same KV cache cut) models with 8 attention heads with reduced kqv sizes. This means that if the group proportion equals 0.5, the KQV proportion will also be 0.5 since, as explained in Section 3, it leads to the same KV-cache cut.

For RQ3, we train one additional model with 8 attention heads, combining GQA and reduced kqv sizes. For easier comparison to the models trained for RQ1.1 and RQ2 (that use either GQA or reduced kqv sizes but not both), we select a group proportion and kqv proportion both equal to 0.5. This way, the model's performance can be compared to the GQA model with a 0.25 group proportion and KQV model with a 0.25 KQV proportion.

The baselines (for models with any number of attention heads) are always trained with an intermediate (FFN) size of 1024. Each GQA and KQV model adjusts this number as described in Subsection 3.3 to keep the number of parameters the same (at around ~ 8 mil.) to have KV-cache size as the only contributor to the memory consumption.

4.5 Evaluation Setting Metrics[5]

Quality Assessment

In our study, the term "quality" refers to a model's proficiency in understanding natural language. We employed the BabyLM pipeline [14] to evaluate this aspect, which comprises three main components: BLiMP, GLUE, and SuperGLUE.

The BLiMP (Benchmark of Linguistic Minimal Pairs) [16, 15] evaluates models' ability to differentiate between grammatically correct and incorrect sentences across various linguistic domains. It features pairs of minimally different sentences, with one sentence containing a grammatical error. Model performance is assessed based on its accuracy in identifying the grammatically correct sentence.

GLUE (General Language Understanding Evaluation) [13] and SuperGLUE [12] are benchmarks designed to evaluate language models' performance in tasks such as single-sentence classification, similarity and paraphrase detection, and natural language inference, emphasizing text comprehension and reasoning. The BabyLM pipeline integrates selected tasks from both GLUE and SuperGLUE.

Speed Assessment

We train and evaluate quality and speed for all the models on the same hardware. Namely, an NVIDIA GeForce RTX 3080 GPU with 10 GB of VRAM is utilized. To measure speed for each model, we generate inputs for batch sizes of 16 and 512 to see how they affect the speedup induced by the applied techniques. Each input sequence in each batch has a length of 32 tokens and an output length of 256 tokens.

Before running the batches used for evaluation, we run generation on 50 warm-up batches to ensure the GPU is in the same state for all the consequent generations. We run a total of 150 batches to find an average generation time per one token which is achieved by dividing the overall time by the number and size of batches. This method ensures consistency among the runs without encountering any outliers.

The time is tracked 'time.time()' function from the Python module 'time' and does not include tokenizer operations such as decoding outputs to human-readable strings. To calculate the speedup, we use the Formula 1 that calculates by how many percent the baseline model (t_{new}) is slower than the new one (t_{old}).

$$\Delta v = (\frac{t_{old}}{t_{new}} - 1) * 100 \quad (1)$$

5 Results

RQ1.1

Figure 7 shows how GQA models' quality and speedup vary with different group proportions relative to MHA models with batch sizes of 16 and 512. The speedup consistently increases as the group proportion decreases, as depicted. Conversely, the average of BLiMP and GLUE/SuperGLUE scores decreases steadily, indicating an inverse relationship between quality and speed. For a proportional relationship, the middle gray curve should align closely with a straight line, observed more clearly with the larger batch size. However, when examining individual quality scores rather than averages, the relationship between speed and quality is neither strictly inverse nor proportional.

RQ1.2

Table 1 presents the evaluated quality of GQA models with a group factor of 0.75, using 4, 8, and 16 attention heads, alongside corresponding baseline results. The decline in quality is visualized in Figure 8. While individual quality scores show no distinct trend, the average indicates that models with more attention heads experience less deterioration in quality for the same group proportion.

RQ2

Table 2 compares GQA and KQV models, ensuring an equivalent KV-cache cut by matching group proportions in GQA to KQV proportions. Bold highlights the best results in each pair. KQV models consistently excel in speed, while GQA models generally achieve higher quality. However, Figure 9 illustrates that as proportion values increase, the quality gap narrows. For proportions of 0.75 and 0.5, KQV models perform nearly as well as GQA models and sometimes even surpass them in specific metrics while maintaining significantly faster speeds.

RQ3:

Table 3 compares the combined approach, setting both group and KQV proportions to 0.5 (resulting in a 4x KV cache reduction), against individual approaches with group and KQV proportions set to 0.25 separately, achieving the same cut. The quality of the combined approach, both in individual metrics and on average, falls between that of GQA and KQV models (highlighted in bold). Similarly, the speed for large batch sizes also falls between the GQA and KQV approaches (highlighted in bold). However, the combined approach exhibits the slowest performance for small batch sizes.

6 Discussion

6.1 Implications

First, from Figure 7, we can imply that the relationship between speed and quality in GQA models is inversely related

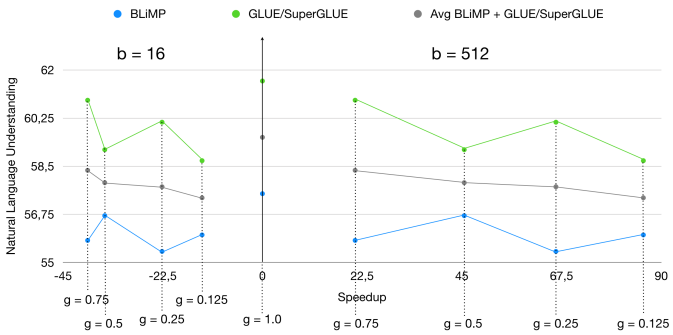

Figure 7: Speed changes and quality for various GQA group proportions (indicated with 'g' at the bottom). Negative speedup on the left is for a batch size of 16 (relative to the baseline with batch size 16), and on the right is for a batch size of 512 (relative to the baseline with batch size 512).

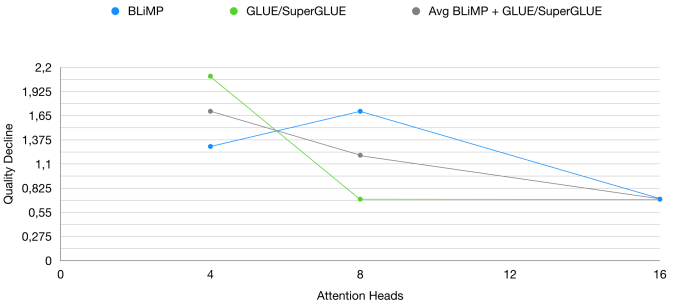

Figure 8: GQA quality deterioration for different numbers of attention heads with a fixed group proportion of 0.75.

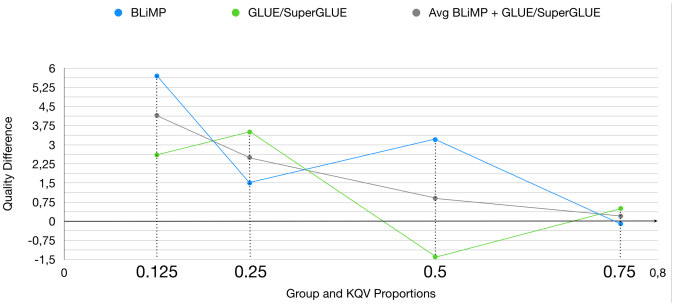

Figure 9: Quality differences between GQA models and corresponding KQV models (with KQV proportion equal to group proportions) for various proportions. The negative difference means a KQV model performs better on the given metrics.

Table 1: Comparison of quality of GQA models to MHA (vanilla GPT-Neo) baselines with different numbers of attention heads. The group proportion is kept fixed for all GQA models at 0.75.

Attention Type	MHA	GQA	MHA	GQA	MHA	GQA
Attention Heads	4	4	8	8	16	16
FFN width	1024	1120	1024	1120	1024	1120
BLiMP	58.1%	56.8%	57.5 %	55.8%	54.1%	53.4%
GLUE/ SuperGLUE	61.2%	59.1%	61.6%	60.9%	59.5%	58.8%
Average of BLiMP and GLUE/ SuperGLUE	59.65%	57.95%	59.55%	58.35%	56.8%	56.1%

(lower group proportion leads to higher speed but lower average quality) for both batch sizes. Additionally, the relationship appears proportional for larger batch sizes, where memory becomes more of a bottleneck than computation. This proportionality is not observed in smaller batch sizes, likely due to the additional computational overhead introduced by the approach described in Subsection 3.2, which affects the speedup differently based on group proportions and diminishes in larger batches. Therefore, the speedup results for smaller batch sizes are less reliable. To this end, we recommend using the modified GQA approach, which accommodates any arbitrary number of key-value heads solely for assessing GQA's quality-related properties. For real-world applications and speed measurements, using the standard version of GQA is advisable to avoid computational overhead.

Next, Figure 8 indicates that increasing the number of attention heads reduces the quality decline for the same group proportion. However, notably, the baseline quality with 16 heads is substantially lower (Table 1), indicating that more attention heads do not always improve performance and should be balanced with the overall model size. This observation suggests that the reduced quality difference might be due to GQA effectively removing unnecessary heads. Nevertheless, the quality difference between the baselines with 4 and 8 heads is insignificant, yet, the application of GQA still decreases the quality gap. This implies that even with similar baseline quality, models with more heads can afford a lower group proportion with GQA, effectively allowing for more head reduction.

Besides, Table 2 highlights that KQV models consistently achieve faster speeds than GQA models with equivalent KV-cache reductions. The main factor contributing to GQA's slower performance is its computational overhead, which results in negative speedups initially but transitions to positive in larger batches due to memory constraints outweighing computations. Additionally, reducing the size of kqv vectors reduces floating-point operations, unlike GQA, which primarily cuts memory usage but may introduce computational overhead. The variance in quality with different KQV proportions underscores the need for careful selection, as excessive reduction can significantly degrade quality, unlike GQA. However, smaller KQV proportions often provide better speedups with comparable quality, making them a potentially preferable choice over GQA in certain scenarios.

Finally, as the combined approach's quality and speed fall between the pure GQA and KQV models' results (Table 3), it can be viewed as a valid way to expand the range of options available for choosing the desired trade-offs for transformer inference.

6.2 Threats to Validity

Internal validity assesses whether the observed results are genuinely caused by the independent variables rather than other influences. External validity addresses how well the study's findings can be applied to different contexts beyond the study itself. Construct validity evaluates whether the measurement tools accurately capture the intended concepts for the study.

Internal Validity

There are three main threats to our internal validity. First, widening the FFN width to match parameter counts introduces a new variable that might affect results. Second, modifying the GQA approach to allow more group proportions adds overhead that slows inference. Lastly, due to time and resource constraints, we train each model for only one epoch, likely preventing convergence and affecting quality metrics.

External Validity

The small size of the selected models raises questions about scalability with more parameters. The models required larger batch sizes to show significant effects from GQA and reduced kqv sizes. Additionally, the synthetic TinyStories [5] dataset generated by GPT-3.5 and GPT-4 may not reflect the quality of real-world datasets. Finally, our modified GQA approach, intended only for quality evaluation with various proportions, may not yield speed results applicable to standard GQA experiments.

Construct Validity

To ensure the validity of our speed measurements, we evaluated all models on a single GPU with no background tasks, warmed up the GPU before measurement, and averaged results over many batches. This way, we address potential construct validity threats related to measurement consistency. However, using GLUE/SuperGLUE to evaluate decoder-only models remains a construct validity concern, as the models were fine-tuned for classification tasks of these benchmarks from scratch.

6.3 Future Work

Addressing the threats to validity (Subsection 6.2), future investigations should include extended epochs of model training to ensure result consistency. Additionally, evaluating speedups for unmodified GQA models (adjusting attention logic without retraining) could yield improved outcomes without added computational overhead. Experimentation with larger models would also provide valuable insights into scalability.

Variable GQA[6] presents an intriguing area where group

[6]https://deci.ai/blog/decilm-15-times-faster-than-llama2-nas-generated-llm-with-variable-gqa/

Table 2: GPT-Neo models' performance for different GQA group proportions against KQV models with equivalent KQV proportions. The models' metrics are compared pair-wise, and the better results out of the two are highlighted in bold. Each model has 8 attention heads and fixed sizes of d_k and d_v equal to 384. The FFN is adjusted to keep the number of parameters the same across all the models. The speedup results are presented for the batch sizes 16 and 512 relative to the baseline.
The baseline is a vanilla GPT-Neo with 8 heads and standard Multi-Head Attention.

Attention Type		Baseline	GQA	KQV	GQA	KQV	GQA	KQV	GQA	KQV
Group Proportion		–	0.75	–	0.5	–	0.25	–	0.125	–
KQV Proportion		–	–	0.75	–	0.5	–	0.25	–	0.125
FFN Width		1024	1120	1216	1216	1408	1312	1600	1360	1696
BLiMP		57.5%	55.8%	**55.9%**	**56.7%**	53.5%	**55.4%**	53.9%	**56.0%**	50.3%
GLUE/ SuperGLUE		61.6%	**60.9%**	60.4%	59.1%	**60.5%**	**60.1%**	56.6%	**58.7%**	56.1%
Average of BLiMP and GLUE/ SuperGLUE		59.55%	**58.35%**	58.15 %	**57.9%**	57.0%	**57.75%**	55.25%	**57.35%**	53.2%
Average Time per Token Generation (μs)	b = 16	136.67	225.66	135.19	212.16	133.51	176.69	134.44	158.22	134.15
	b = 512	17.31	14.32	11.82	11.90	7.48	10.42	6.47	9.31	5.99
Speedup Over the GPT-Neo Baseline	b = 16	–	−39.44%	**1.09%**	−35.58%	**2.36%**	−22.65%	**1.66%**	−13.62%	**1.87%**
	b = 512	–	20.92%	**46.50%**	45.54%	**131.36%**	66.23%	**167.54%**	85.96%	**188.84%**

Table 3: Comparison of speed and quality between the combined approach and the GQA and KQV models, resulting in an equivalent KV-cache reduction. Metrics that fall between those of the individual approach models are highlighted in bold.

Attention Type	GQA	COMBINED	KQV
Group Proportion	0.25	0.5	–
KQV Proportion	–	0.5	0.25
FFN width	1312	1504	1600
BLiMP	55.4%	54.5%	53.9%
GLUE/ SuperGLUE	60.1%	58.5%	56.6%
Average of BLiMP and GLUE/SuperGLUE	57.75%	**56.5%**	55.25%
Time per Token (μs) (b = 16)	176.69	207.79	134.44
Time per Token (μs) (b = 512)	10.42	9.01	6.47
Speedup (b = 16)	−22.65%	-34.23%	1.66%
Speedup (b = 512)	66.23%	**92.12%**	167.54%

proportions vary among transformer blocks, unlike standard GQA where they remain uniform. Utilizing this approach post-pre-training (via uptraining [1]) and integrating with importance scores for attention head pruning [8] could optimize layer-wise head reduction, minimizing quality loss while achieving significant speedups. Furthermore, combining GQA and KQV techniques with other compression methods offers opportunities to explore additional trade-offs.

7 Conclusion

This study aimed to investigate trade-offs in transformer models between inference speed and natural language understanding, particularly focusing on techniques that reduce the dimensionality of the attention mechanism. Specifically, we examined the properties of Group-Query Attention (GQA), explored an orthogonal approach involving dimensionality reduction of keys and values, and investigated their combined effects.

Our findings showed an inverse relationship between quality decline and speedup in Grouped-Query Attention: higher group counts improve quality but worsen speedup. This effect becomes more pronounced in scenarios with substantial memory consumption, approaching a proportional relationship. Furthermore, increasing the number of attention heads in the model suggests a reduced optimal ratio of groups in GQA to maintain comparable quality levels.

Furthermore, we observed that in certain configurations, models employing reduced key and value vectors closely approximate those using GQA, yet exhibit superior speedup. Nevertheless, reducing these vectors too much may lead to a significant decline in quality compared to GQA-based models.

Finally, combining GQA with reduced key and value vectors demonstrates intermediary results between individual approaches, expanding the range of trade-off options available for optimization.

8 Responsible Research

To address the current issue of reproducibility in machine learning research, we provide a replication package for reproducing our findings and models. Additionally, Section 4 details all model configurations and hardware used. All models were pre-trained and evaluated with fixed seeds to ensure consistency.

To prevent test set contamination, we pre-trained our models on datasets separate from those used for evaluation. This strict separation between training and evaluation datasets preserves the integrity of our findings and enhances the reliability of our research.

We adhered to principles of scientific integrity by ensuring accurate reporting and proper citations, avoiding fabrication and plagiarism. Guided by the Netherlands Code of Conduct for Research Integrity, we incorporated honesty, transparency, and responsibility into our research practices. By following the educational and normative framework outlined in chapters 2 and 3 of the Code, we emphasized good research practices that foster a responsible research environment [7].

References

[1] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints, December 2023. arXiv:2305.13245 [cs].

[2] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners, 2020. Version Number: 4.

[4] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale, November 2022. arXiv:2208.07339 [cs].

[5] Ronen Eldan and Yuanzhi Li. TinyStories: How Small Can Language Models Be and Still Speak Coherent English?, May 2023. arXiv:2305.07759 [cs].

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, March 2015. arXiv:1503.02531 [cs, stat].

[7] KNAW, NFU, NWO, TO2-Federatie, Vereniging Hogescholen, and VSNU. Nederlandse gedragscode wetenschappelijke integriteit, 2018.

[8] Paul Michel, Omer Levy, and Graham Neubig. Are Sixteen Heads Really Better than One?, November 2019. arXiv:1905.10650 [cs].

[9] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently Scaling Transformer Inference, November 2022. arXiv:2211.05102 [cs].

[10] Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need, November 2019. arXiv:1911.02150 [cs].

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, December 2017. arXiv:1706.03762 [cs].

[12] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems, February 2020. arXiv:1905.00537 [cs].

[13] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding, February 2019. arXiv:1804.07461 [cs].

[14] Alex Warstadt, Leshem Choshen, Aaron Mueller, Adina Williams, Ethan Wilcox, and Chengxu Zhuang. Call for Papers – The BabyLM Challenge: Sample-efficient pretraining on a developmentally plausible corpus, January 2023. arXiv:2301.11796 [cs].

[15] Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. Findings of the BabyLM Challenge: Sample-Efficient Pretraining on Developmentally Plausible Corpora. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–6, Singapore, 2023. Association for Computational Linguistics.

[16] Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. BLiMP: The Benchmark of Linguistic Minimal Pairs for English, February 2023. arXiv:1912.00582 [cs].

[17] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, April 2009.