Efficient Self-Attention Mechanisms Via Vector Quantization

George Ankeney
ICME, Stanford University
ankeneyg@stanford.edu
SUNet ID: ankeneyg

Juan Muneton Gallego
ICME, Stanford University
jmuneton@stanford.edu
SUNet ID: jmuneton

1 Introduction

Transformers have revolutionized fields like natural language processing, audio processing, and computer vision, but their quadratic time and space complexity with sequence length poses challenges for long-sequence tasks. Existing solutions, such as sparse factorizations, locality-sensitive hashing, and linear attention mechanisms, reduce complexity but often face issues like gradient instability and degraded inference performance. We propose a novel self-attention mechanism using vector quantized-variational autoencoders (VQ-VAEs) to achieve sub-quadratic run-time complexity. By compressing input representations of keys and queries in the self-attention mechanism via vector quantization, our approach enables efficient attention over long contexts while maintaining performance stability. Our benchmarks demonstrate that the VQ-based attention mechanism outperforms vanilla attention and baseline models in throughput, memory efficiency, and computational throughput across sequence lengths from 10^3 to 10^5, offering a scalable and effective solution for long-sequence processing at inference time. The codebase for our experiments and algorithm design can be found here `https://github.com/jmunetong/vq_attn/tree/gpu`. *Note: we built this algorithm from scratch.*

2 Related Work

Transformers have demonstrated remarkable performance capabilities across various domains such as natural language processing [3], audio [11], and vision [10]. However, these algorithms suffer from a quadratic space-time complexity concerning the input length. Thereby, requiring large computing and memory resources in tasks that are naturally dependent on long sequences.

To address this limitation, previous work has focused on identifying methodologies that increase the context length while preserving computing and memory capabilities. For example, researchers have proposed using sparse matrix factorizations of the attention matrix to reduce the run-time complexity to $O(N\sqrt{N})$ [1]. Other work has focused on replacing the dot-product component of the attention mechanism with locality-sensitive hashing, which reduces the algorithm to a run-time complexity of $O(N\log(N))$ [6]. While these models have been shown to have better run-time performances in long-sequence training, their run-time at inference pales in comparison to autoregressive modeling inference [5].

Another line of research focuses on expressing the self-attention mechanism as a linear dot-product of kernel feature maps, allowing the use of the associative property of matrix multiplication and reduces the complexity to $O(N)$ [5]. These so-called linear attention mechanisms, however, suffer from two important bottlenecks, which affect performance compared to the vanilla attention mechanism. One of these drawbacks is that most current linear transformers adopt the attention mechanism from the original vanilla transformer, which scales attention scores to keep them within the range [0,1] [9]. However, researchers have demonstrated that this scaling approach causes gradients to become unbounded in linear transformer models, resulting in instability during convergence [9]. Furthermore, linear transformers suffer from attention dilution, a phenomenon by which the model distributes attention scores over the entire sequence including in early layers [9]. These bottlenecks have resulted in a performance drop in linear transformers when compared to vanilla attention.

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

3 Methodology

For our project, we built a self-attention mechanism that quantizes both keys and queries via two codebook representations for each of the matrices Q and K. To achieve this, we constructed two learnable functions focused on creating appropriate mappings for each row of the queries and keys. We will call these functions vector quantizers. Our work follows the initial proposal by [7] which only introduces vector quantization for keys. We will be taking a step further by quantizing queries and keys and introduce a new formulation of the self-attention mechanism.

Definition 1. A *vector quantizer* [7] is a function $\text{VQ}(\cdot; \mathbf{C})$: $\mathbb{R}^D \to \mathbb{R}^D$ such that, for an input $\mathbf{x}$, its quantized output $\hat{\mathbf{x}}$ is given by

$$z \triangleq \arg\min_s \|\mathbf{x} - \mathbf{C}_s\|^2 \to \hat{\mathbf{x}} \triangleq \mathbf{C}_z \tag{1}$$

where $\mathbf{C} \in \mathbb{R}^{S \times D}$ is known as the *codebook*. The row indices $\{0, \ldots, S-1\}$ are known as shortcodes, and the rows of $\mathbf{C}$ the are called *codewords* or *codevectors*. In our project, these codebook matrices will be represented as learnable embeddings. The learning component of these codebooks will follow from the work proposed in [7, 8]. Assuming that we have learned codebook representations for *Q* and K, we will be able to take advantage of the associative property of matrix multiplication to formulate a new self-attention mechanism (assuming noncausal mask), explained below.

Let $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{T \times D}$. Let $\mathbf{B} \in \mathbb{R}^{T \times T}$ such that $B_{i,j} = 0$ for all i, j, $\mathbf{C_Q}, \mathbf{C_K} \in \mathbb{R}^{S \times D}$ be the codebook representations for $\mathbf{Q}$ and $\mathbf{K}$ and ϕ_w is the row-wise softmax nonlinearity. Then the attention weights can be factored:

$$\begin{aligned}
\mathbf{W} &\triangleq \phi_w(\mathbf{QK} + \mathbf{B}) = \phi_w(\mathbf{QK}) && (2)\\
&\approx \phi_w(\text{VQ}(\mathbf{Q}; \mathbf{C_Q})\text{VQ}(\mathbf{K}; \mathbf{C_K})) && (3)\\
&= \phi_w(\hat{\mathbf{Q}}\hat{\mathbf{K}}^\top) = \phi_w(\Delta_Q \mathbf{C_Q} \mathbf{C_K}^T \Delta_K) && (4)\\
&= \text{Diag}\left(\exp(\Delta_Q \mathbf{C_Q} \mathbf{C_K}^T \Delta_K)\mathbf{1}\right)^{-1} \exp(\Delta_Q \mathbf{C_Q} \mathbf{C_K}^T \Delta_K) && (5)\\
&= \text{Diag}\left(\Delta_Q \exp(\mathbf{C_Q} \mathbf{C_K}^T)\Delta_K \mathbf{1}\right)^{-1} \Delta_Q \exp(\mathbf{C_Q} \mathbf{C_K}^T)\Delta_K && (6)\\
&= \text{Diag}\left(\Delta_Q \mathbf{M} \Delta_K \mathbf{1}\right)^{-1} \Delta_Q \mathbf{M} \Delta_K && (3)
\end{aligned}$$

where $\mathbf{1} \in \mathbb{R}^T$, $\text{Diag}\left(\Delta_Q \mathbf{M} \Delta_K \mathbf{1}\right)^{-1} \Delta_Q \mathbf{M} \Delta_K \in \mathbb{R}^{T \times S}$, $\Delta \in \mathbb{R}^{S \times T}$, and $\Delta_{s,t} \triangleq \delta_{s,z_t}$. Here, $\delta_{.,.}$ denotes the Kronecker delta function and z_t is the VQ shortcode for timestep t. By using these properties, we are able to achieve sub-quadratic time complexity.

Having explained the derivation of our proposed self-attention mechanism, it is important to highlight that if the matrix M is learned during training, the softmax operation on the quantized queries and keys matrices can be precomputed just *once*. This implies that the main computational challenge lies in efficiently constructing the Kronecker matrices Δ_K and Δ_Q, which are responsible for selecting the appropriate code vectors for these low-rank representations in the precomputed softmax matrix M.

Remark 1: *Proposed Training Loss*

Although our work did not involve training the quantized self-attention mechanism on a specific dataset, we introduced a tailored loss function designed to train the self-attention mechanism while simultaneously learning optimal codebook representations for the quantization process. For some dataset$\{(x_i, y_i)\}_{i=1}^N$ with $x_i \in \mathbb{R}^D$, $y_i \in \mathbb{R}^C$, a neural network function $f : \mathbb{R}^D \to \mathbb{R}^C$ whose output probabilities are given for C classes is defined as follows:

$$\mathcal{L}_{\text{pred}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij} \log\left(f_j(x_i)\right) \tag{7}$$

Now, for the codebook loss representation for one layer of self-attention, suppose we are attempting to quantize some input $\bar{x} \in \mathbb{R}^D$ using a quantizer encoder and decoder structure z_e and z_q, respectively

The input $\bar{x}$, that is passed through an encoder producing output $z_e(\bar{x})$. The discrete latent variables z are then calculated by a nearest neighbor look-up using the shared embedding space C as shown in equation The posterior categorical distribution $q(z|\bar{x})$ probabilities are defined as one-hot as follows:

$$q(z = k|\bar{x}) = \begin{cases} 1 & \text{for } k = \arg\min_j \|z_e(\bar{x}) - C_j\|_2, \\ 0 & \text{otherwise} \end{cases}$$

similarly, the decoder can be written in this form:

$$z_q(x) = C_k, \quad \text{where } k = \arg\min_j \|z_e(x) - C_j\|_2; \quad s.t \quad C \in \mathbb{R}^{S \times D}$$

We view this model as a VAE in which we can bound $\log p(\bar{x})$ with the ELBO. Hence, we we are able to represent our loss function for one codebook representation with the given formula:

$$L_{ct} = \log p(\bar{x} \mid z_q(\bar{x})) + \|\text{sg}[z_e(\bar{x})] - C\|_2^2 + \beta\|z_e(\bar{x}) - \text{sg}[C]\|_2^2, \tag{8}$$

where sg is the stop gradient function and β is a weighted parameter, which previous work has recommended to be approximately 0.25 [8]. Now, the second loss term of L_{ct} is the embedding error that moves each codevector representation closer to $z_e(\bar{x})$. Finally the third term is responsible for ensuring that the encoder commits to an embedding in its output does not grow large, this is called the commitment loss [8]. Hence, for training two codebooks for queries and keys, the full loss for attention mechanism with t attention layers can be described as follows:

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \sum_{i=1}^{t} \left(L_{ct}^{Q(i)} + L_{ct}^{K(i)} \right) \tag{9}$$

4 Experiments & Results

4.1 Varying Sequence Length Experiment

For this set of experiments, we were interested in quantifying the performance of hyperattention, vanilla attention and VQ attention as the sequence input length increases. We configured the batch size and head size to 1, with each token encoded using an embedding dimension of 512. This setup allowed us to scale efficiently for longer sequences. The vector quantization model used a standardized codebook size of 128 to ensure consistency across experimental conditions. To fairly evaluate the self-attention mechanism, we pre-computed the keys, queries, and values matrices, as well as the Δ_Q and Δ_K matrices for the vector quantized algorithms. This approach isolated the computational cost of the attention step, eliminating preprocessing overhead. Our evaluation metrics included throughput (sequences processed per second), memory efficiency (GB/s), computational throughput (TFLOPs), and the impact of varying codebook sizes.

From Table 1 and Figure 1, we observed that as sequence length increased, the computational

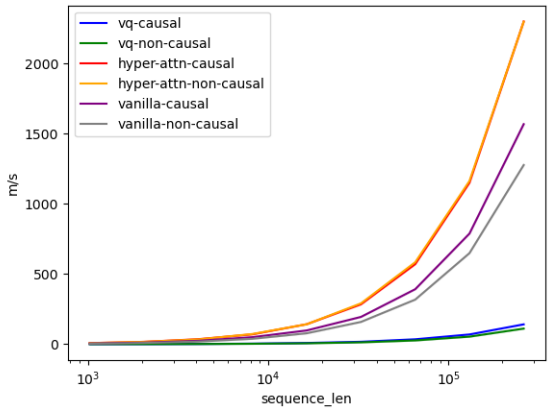

Figure 1: Runtime comparison of our models against benchmark models over increasing sequence lengths

demand rose significantly, leading to longer runtimes across all algorithms. Notably, the VQ attention algorithm consistently outperformed hyper-attention and vanilla attention in both causal and non-causal settings. While causal attention introduced additional overhead due to elementwise Hadamard products, VQ attention remained approximately 30x faster than other algorithms. Although hyper-attention was expected to outperform vanilla attention, the lack of a compatible virtual instance to enable CUDA-based optimizations, such as flash attention, limited its performance.

Table 1: Performance comparison for different sequence lengths and models. All values are reported in Milliseconds (M/s).

Sequence Length	VQ-Causal	VQ-Non-Causal	Hyper-Attn-Causal	Hyper-Attn-Non-Causal	Vanilla-Causal	Vanilla-Non-Causal
1024	**0.579**	**0.604**	12.577	12.512	9.691	7.290
4096	**1.484**	**1.176**	49.886	49.651	38.450	28.989
8192	**2.733**	**2.055**	99.350	99.296	78.095	60.070
16384	**5.235**	**3.945**	197.378	198.046	151.777	115.937
65536	**19.763**	**14.986**	794.919	775.080	620.142	480.433
131072	**39.908**	**29.708**	1590.774	1588.871	1223.857	954.854
262144	**79.184**	**59.296**	3203.771	3195.184	2494.375	1935.017

Computational observations for these experiments are described below:
Memory Efficiency: Figure 2a highlights the memory usage of each model. The VQ-based approaches maintained high memory efficiency, with the VQ-non-causal mechanism achieving up to 3.5 GB/s at longer sequence lengths. This efficiency surpasses vanilla attention models and aligns with the scalability required for long-sequence tasks.
Computational Throughput: Figure 2b illustrates the computational throughput in TFLOPs. The VQ-based models achieved competitive performance, with VQ-non-causal models slightly exceeding VQ-causal models. Both configurations demonstrated significant gains over vanilla and hyper-attention baselines, reflecting the computational efficiency of the proposed method.

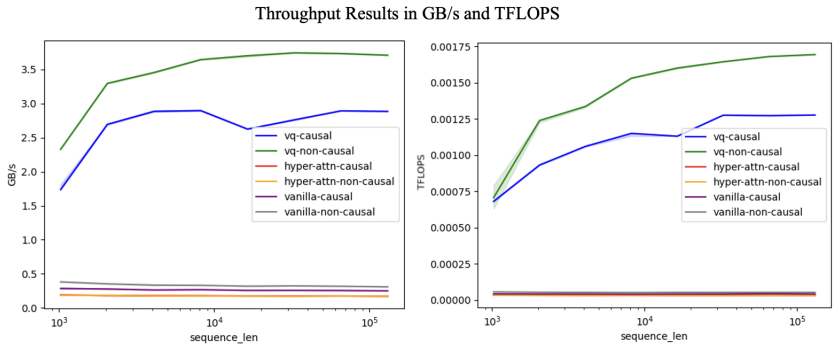

Figure 2: Left figure shows the comparison of the GB/s and right figure provides TFLOPs comparison of our models against benchmark models over increasing sequence lengths in the self-attention mechanism

4.2 Varying Codebook Size Experiment

In this experiment, we quantified performance degradation as a function of codebook size, keeping the sequence length fixed at approximately 16k with the same data hyperparameters and conditions described in the previous experiments. Figure 3 and Table 2 highlight how varying codebook size affects model performance. While larger codebooks enhance the model's ability to capture nuanced attention patterns, improving throughput and computational efficiency, excessively large codebooks introduced diminishing returns due to increased computational overhead without notable performance gains. In fact, with a codevector size 8192, our algorithm underperformed even compared to hyper-attention. This shows that careful selection of the number of codevectors may be required to leverage the commutative properties of matrix computation that we derived.

Table 2: Performance comparison for different n_code values and models. All values are rounded to three decimals as a function of milliseconds m/s.

Codevector Size	VQ-Causal	VQ-Non-Causal	Hyper-Attn-Causal	Hyper-Attn-Non-Causal	Vanilla-Causal	Vanilla-Non-Causal
256	**2.626**	**1.873**	201.971	196.738	157.307	123.022
512	**5.184**	**3.922**	206.907	204.299	160.389	118.089
1024	**12.111**	**9.457**	206.515	196.847	159.178	122.274
2048	**31.23**	**26.278**	207.211	205.622	162.565	120.775
4096	**93.393**	**83.776**	200.044	197.264	163.883	121.786
8192	315.182	295.593	208.255	209.254	**162.582**	**121.458**

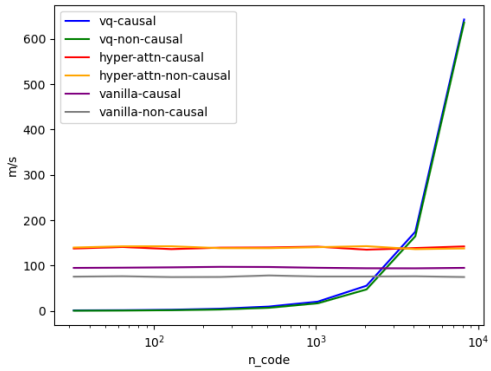

Figure 3: Runtime comparison of our models against benchmark models over varying codebook sizes

5 Conclusions, Limitations & Future Work

In this work, we introduced a novel VQ-based self-attention mechanism designed to address the inefficiencies of traditional transformer architectures when processing long sequences. By leveraging vector quantization and matrix properties, our approach significantly reduces the computational and memory complexity of self-attention while maintaining competitive performance. Experimental results demonstrated substantial improvements in memory efficiency and computational throughput across various sequence lengths and codebook sizes, underscoring the scalability and effectiveness of the proposed method. Notably, careful selection of the codebook size emerged as a key factor in optimizing the performance of the self-attention mechanism.

However, this work has certain limitations that merit further exploration. Due to hardware constraints and incompatibilities, we were unable to evaluate the performance of flash attention and hyper-attention using CUDA, limiting the scope of our performance analysis. Additionally, this study did not investigate tools like Triton for optimizing matrix computations in VQ or Faiss for more efficient codevector search. Furthermore, the lack of formal scaling laws limits our ability to predict the feasibility of this approach for larger models.

Despite these limitations, this work opens promising avenues for future research. Applying this method to diverse domains could help evaluate the robustness of constraining attention weights to finite codevector projections. Moreover, examining the effects of varying quantized query and key sizes could provide deeper insights into the model's limitations and expand its potential applications, enabling further refinements.

Overall, this study demonstrates that vector quantization of queries and keys offers a viable alternative to traditional approaches for optimizing attention mechanisms. By reducing the runtime dependence on sequence length, this method paves the way for more efficient and scalable implementations of self-attention in large-scale models.

6 Contributions

Both George and Juan focused a lot of efforts to develop this algorithm from scratch. During the setup of our preliminary experiments, we encountered challenges due to limited computational resources, which ultimately led to the failure of one of the initial tests. To resolve this issue, we are transitioning the experimental framework to a virtual machine hosted on Google Cloud, a platform equipped with the requisite processing power to handle the computational demands and ensure successful execution. We managed to deliver experiments on time but this time did not allow us to explore using faiss to do efficient search of codevectors and evaluate the efficiency of this component of our model. Juan was responsible for organizing, compiling, and setting up the experiments. George focused on getting baseline experiments results for HyperAttention and FlashAttention in different hardware on GCP, but none of the available options allowed us to run our experiments. Both worked on using a Virtual Machine to run tests. We both contributed equally to this paper.

7 Literature Overview

References

[1] Rewon Child et al. *Generating Long Sequences with Sparse Transformers*. Apr. 23, 2019. arXiv: 1904.10509[cs,stat]. URL: http://arxiv.org/abs/1904.10509 (visited on 10/05/2024).

[2] Tri Dao et al. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. June 23, 2022. arXiv: 2205.14135[cs]. URL: http://arxiv.org/abs/2205.14135 (visited on 10/05/2024).

[3] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. May 24, 2019. arXiv: 1810.04805[cs]. URL: http://arxiv.org/abs/1810.04805 (visited on 10/05/2024).

[4] Insu Han et al. *HyperAttention: Long-context Attention in Near-Linear Time*. Dec. 1, 2023. arXiv: 2310.05869[cs]. URL: http://arxiv.org/abs/2310.05869 (visited on 10/05/2024).

[5] Angelos Katharopoulos et al. *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention*. Aug. 31, 2020. arXiv: 2006.16236[cs,stat]. URL: http://arxiv.org/abs/2006.16236 (visited on 10/05/2024).

[6] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. *Reformer: The Efficient Transformer*. Feb. 18, 2020. arXiv: 2001.04451[cs,stat]. URL: http://arxiv.org/abs/2001.04451 (visited on 10/05/2024).

[7] Lucas D. Lingle. *Transformer-VQ: Linear-Time Transformers via Vector Quantization*. Feb. 25, 2024. arXiv: 2309.16354[cs]. URL: http://arxiv.org/abs/2309.16354 (visited on 10/05/2024).

[8] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. *Neural Discrete Representation Learning*. May 30, 2018. arXiv: 1711.00937[cs]. URL: http://arxiv.org/abs/1711.00937 (visited on 10/05/2024).

[9] Zhen Qin et al. *The Devil in Linear Transformer*. Oct. 19, 2022. arXiv: 2210.10340[cs]. URL: http://arxiv.org/abs/2210.10340 (visited on 10/05/2024).

[10] Prajit Ramachandran et al. *Stand-Alone Self-Attention in Vision Models*. June 13, 2019. arXiv: 1906.05909[cs]. URL: http://arxiv.org/abs/1906.05909 (visited on 10/05/2024).

[11] Matthias Sperber et al. *Self-Attentional Acoustic Models*. June 18, 2018. arXiv: 1803.09519[cs]. URL: http://arxiv.org/abs/1803.09519 (visited on 10/05/2024).

[12] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. Dec. 14, 2014. arXiv: 1409.3215[cs]. URL: http://arxiv.org/abs/1409.3215 (visited on 10/05/2024).