Article

Not peer-reviewed version

Bucket Attention: Fixed-Size Space for Any Length of Context

Zipeng Ye *

Posted Date: 28 August 2025

doi: 10.20944/preprints202508.1313.v2

Keywords: large language models; attention

 doi:10.20944/preprints202508.1313.v2

Article

Bucket Attention: Fixed-Size Space for Any Length of Context

Zipeng Ye

Independent Researcher; yezp13@gmail.com

Abstract

In this study, we analyze the attention mechanism and propose a novel perspective where sequential inputs within the attention mechanisms do not require strict order. We introduce an innovative approach, called bucket attention, which organizes context in large language models (LLMs) and effectively handles contexts of any length while utilizing a fixed-size space. Furthermore, we present techniques to convert pre-trained models based on traditional attention into the bucket attention framework, along with a method to train models with bucket attention from scratch. These approaches offer practical solutions to improve the efficiency and scalability of LLMs.

Keywords: large language models; attention

1. Introduction

In recent years, attention mechanisms [1] have played a key role in natural language processing, driving the rapid advancement of large language models (LLMs) such as BERT [2], GPT [3] and DeepSeek-R1 [4]. As the powerful capabilities of the attention mechanism [1] have been recognized, transformers have gradually replaced convolutional neural networks. Hypernetworks [5] is also a perspective for viewing the relationship between networks and different inputs. Studies indicate that attention mechanisms can be viewed as hypernetworks [6–8], where inputs are context. Therefore, context is crucial for LLMs, as it provides conditions and extensive information that profoundly influence the outputs.

Many tasks [9] require handling extremely long contexts, which often exceed standard context-length limits. It typically requires a large amount of space and takes a significant amount of time. To alleviate the challenges, a variety of methods have been proposed. Sparse attention [10–12] computes only a portion of attention scores, effectively reducing memory usage and improving the computation speed by limiting the interaction scope or density of each element. Essentially, it is very similar to the approach of pruning the KV cache [13,14]. A straightforward approach is to compute local attention based on the position of each element [15–17]. Furthermore, Locality-Sensitive Hashing (LSH) [18,19] is a powerful tool to compute the correlation between elements. The other approach employs hierarchical structures to capture contextual information at different levels, gradually integrating long-distance dependencies, and thus reducing spatial and temporal costs [20–22]. Another approach leverages a sleep mechanism [23] to compress the context into network parameters, but is unable to handle dynamic long-context tasks.

However, these methods have not completely resolved the issue. To better address this challenge, we propose an innovative attention mechanism that utilizes fixed-size storage, enabling the attention mechanism to effectively manage contexts of any length. Key contributions of this paper include the following:

- Through analyzing the attention mechanism, we propose a novel perspective: sequential inputs do not need to be organized in order within attention mechanisms.
- We introduce a novel approach to organizing context of LLMs that is capable of handling any length of context using a fixed size space, i.e. bucket attention.

 doi:10.20944/preprints202508.1313.v2

- We propose a method to convert a pre-trained model based on traditional attention into that of bucket attention.
- We propose a method to train a model with bucket attention from scratch.

2. Analysis

LLMs generally employ next token prediction and unidirectional attention. Therefore, throughout this paper, attention specifically refers to unidirectional attention.

2.1. Attention is Unordered

First, it is important to recognize that the attention mechanism is inherently unordered, which is why position encoding becomes essential. The organization of natural language is sequential, but fundamentally, this orderliness is linked only to position encoding, not to the attention mechanism itself. A notable fact is that we can completely rearrange the order of elements within the KV cache. Thus, we can modify the organization of the KV cache, enabling it to be structured as any efficient data structure rather than just a sequence. Numerous studies [20–22] have organized the context into hierarchical structures. However, merely rearranging the order of the context is insufficient to fit an arbitrarily long context into a fixed space. To accomplish this, it is crucial to compress or trim the context while minimizing information loss. Information theory tells us that it is impossible to compress a random sequence into a very small space. However, natural language is not a random sequence, which contains a great deal of redundancy, allowing large portions of text to be summarized effectively.

2.2. Finite-Dimensional Context

We assume that the concepts in the world are finite, and consequently, the logical relationships between them are also finite. Therefore, the state of context should reside in a finite-dimensional space. This gives us hope that context can be described using this finite-dimensional space, rather than relying on a potentially infinite sequence. It is challenging to enumerate keywords and key relationships, but models learn the language structure, extract essential contextual information, and summarize it into key concepts. We can interpret it as a state, yet unlike the abstract states within Recurrent Neural Network (RNN), this state is fundamentally a form of attention.

Based on the above assumptions and analysis, we present a conceptual attention mechanism within a finite-dimensional space. The relationship between the novel attention mechanism and the traditional attention mechanism is similar to that between the Lebesgue integration and the Riemann integration. We partition the context into a fixed number of intervals by keys, aggregating values within each interval, thereby constructing attention that uses a fixed space.

We use equations to provide a more precise description of our approach. The formula for calculating attention is as follows.

$$A_i = \mathrm{softmax}\left(\frac{Q_i K_{0\ldots i}^\top}{\sqrt{d_k}}\right) V_{0\ldots i} \triangleq \mathrm{softmax}\left(\frac{Q_i [K_0 \ldots K_i]^\top}{\sqrt{d_k}}\right) [V_0 \ldots V_i], \tag{1}$$

where $K_{0\ldots i}$ and $V_{0\ldots i}$ are keys and values of the tokens from 0-th to i-th. We partition the space of keys into N groups (buckets) and we have the following.

$$A_i = \mathrm{softmax}\left(\frac{Q_i [K_{G_0} \ldots K_{G_{N-1}}]^\top}{\sqrt{d_k}}\right) [V_{G_0} \ldots V_{G_{N-1}}], \tag{2}$$

where $G_0 \ldots G_{N-1}$ are groups of keys and each group includes multiple elements. We need to approximately convert each group containing multiple elements into a single element. Since we group

 doi:10.20944/preprints202508.1313.v2

them by keys, we first select a representative element $\vec{K}_t$ from each group G_t, where $\vec{K}_t$ is a unit vector indicating direction. Assume K_j belongs to the $t(j)$-th group (bucket) $G_{t(j)}$, $K_j \doteq \alpha_j \vec{K}_{t(j)}$.

$$A_i = \text{softmax}\left(\frac{Q_i[\alpha_0 \vec{K}_{t(0)} \dots \alpha_i \vec{K}_{t(i)}]^\top}{\sqrt{d_k}}\right)[V_0 \dots V_i] \tag{3}$$

$$= \text{softmax}\left(\frac{Q_i[\beta_0 \vec{K}_0 \dots \beta_{N-1} \vec{K}_{N-1}]^\top}{\sqrt{d_k}}\right)[\bar{V}_0 \dots \bar{V}_{N-1}], \tag{4}$$

where β_t is derived by weighting α_j from the same group and $\bar{V}_t$ is derived by weighting V_j from the same group.

$$\text{i.e.} \begin{cases} \beta_t = \log(\sum_{j \in G_t} e^{\alpha_j}) \\ \gamma_j = e^{\alpha_j}/\sqrt{d_k}/e^{\beta_t}/\sqrt{d_k} \\ \bar{V}_t = \sum_{j \in G_t} \gamma_j V_j \end{cases}. \tag{5}$$

Therefore, by employing a grouping method, we can compress an arbitrarily long context into fixed-length N buckets.

3. Bucket Attention

3.1. Training-free Approach

Building on the above analysis, we can intuitively develop an inference-time scaling algorithm that requires no training and efficiently stores arbitrarily long contexts using only a fixed space during inference. Analyzing the formula above, during inference we need to update the bucket cache of $\vec{K}_t$, $\bar{V}_t$ and β_t. Compared to typical inference with the KV cache, we will need to store an additional variable β_t. The detail is shown in Algorithm 1.

Algorithm 1 Single Bucket: Calculate A_i and Maintain State

Input: K_i, Q_i, V_i and prefilled $\vec{K}, \bar{V}, \beta$
$t \leftarrow$ the bucket index of K_i
$\alpha_i \leftarrow$ the projected length of K_i
$\beta'_t \leftarrow \log(e^{\beta_t} + e^{\alpha_i})$
$\bar{V}_t \leftarrow (e^{\beta_t}/\sqrt{d_k}\bar{V}_t + e^{\alpha_i}/\sqrt{d_k}V_i)/e^{\beta'_t}/\sqrt{d_k}$
$\beta_t \leftarrow \beta'_t$
$A_i \leftarrow \text{softmax}\left(\frac{Q_i[\beta_0 \vec{K}_0 \dots \beta_{N-1}\vec{K}_{N-1}]^\top}{\sqrt{d_k}}\right)[\bar{V}_0 \dots \bar{V}_{N-1}]$
return A_i

Furthermore, we can extend each layer of attention to span multiple buckets rather than being confined to just one. In practice, this is also essential. Note that in high-dimensional spaces, any two randomly chosen vectors are almost orthogonal. This provides a theoretical foundation for the fact that the representative elements of any two buckets are nearly orthogonal. Thus, we can avoid performing Schmidt orthogonalization on them. Consequently, a vector can be decomposed into multiple buckets by projection, which could be dense or sparse according to its principal components. The detail is shown in Algorithm 2 and Figure 1. To improve computational efficiency, we use sparse bucket attention, which limits the number of buckets involved each time, that is, $|T_i| \ll N$.

We may need to consider different application scenarios to decide whether to use bucket attention. Essentially, this approach is a variant and enhancement of the KV cache, allowing flexible switching between traditional KV cache and bucket attention.

 doi:10.20944/preprints202508.1313.v2

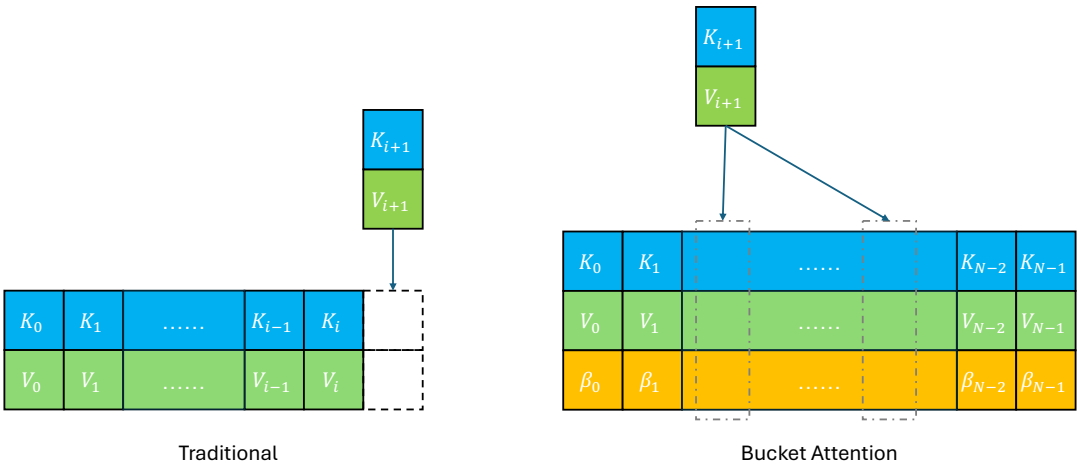

Figure 1. Illustration between the traditional KV Cache of attention and bucket attention.

Algorithm 2 Multiple Buckets: Calculate A_i and Maintain State

Input: K_i, Q_i, V_i and prefilled $\vec{K}, \bar{V}, \beta$
$T_i \leftarrow \{ \text{t} \mid \text{the bucket indices of } K_i \}$
$\{\alpha_t^i\}_t \leftarrow$ the projected length of K_i
for each t in T_i **do**
  $\beta_t' \leftarrow \log(e^{\beta_t} + e^{\alpha_t^i})$
  $\bar{V}_t \leftarrow (e^{\beta_t}/\sqrt{d_k}\bar{V}_t + e^{\alpha_t^i}/\sqrt{d_k}V_i)/e^{\beta_t'}/\sqrt{d_k}$
  $\beta_t \leftarrow \beta_t'$
end for
$A_i \leftarrow \text{softmax}\left(\frac{Q_i[\beta_0\vec{K}_0 \ldots \beta_{N-1}\vec{K}_{N-1}]^\top}{\sqrt{d_k}}\right)[\bar{V}_0 \ldots \bar{V}_{N-1}]$
return A_i

3.2. Partitioning of Buckets

Partitioning buckets is an intriguing issue to investigate. The simplest method is to divide the space uniformly, but this does not capture the distribution of the data. A more effective approach is to sample and partition based on data density, utilizing algorithms similar to K-Nearest Neighbors (KNN) [24].

An important issue to consider is the selection of the total number of buckets N and the number of principal components $|T_i|$. As the number of buckets increases, the algorithm becomes more similar to traditional attention. Therefore, the fundamental principle is to utilize as much memory as possible. Furthermore, N of each attention layer could be independently selected. Although it is common practice to keep the parameters consistent across layers, they can also be varied empirically. Another selection method involves estimating the intrinsic dimensionality of the corpus through error analysis.

3.3. Training From Scratch

If we train a model from scratch instead of converting a pre-trained model, we can simplify the process further. We can allow the model to learn the buckets implicitly, similar to learning an embedding, rather than explicitly partitioning them. We call this method implicitly bucket attention, where the neural network outputs the decomposition coefficients of keys and queries in the bucket space instead of directly outputting values. This approach has the advantage of embedding the bucket partitions within the model parameters while reducing computational effort. Moreover, we can directly employ dense bucket attention without incurring any additional computational cost.

 doi:10.20944/preprints202508.1313.v2

We describe the computation method of implicit bucket attention using equations.

$$Q_i K_{0\dots i}^\top = \boldsymbol{\theta}^i \boldsymbol{\alpha}^\top \tag{6}$$

$$\text{where} \begin{cases} K_j & = [\alpha_0^j \vec{K}_{t(0)} \dots \alpha_{N-1}^j \vec{K}_{N-1}] \triangleq \boldsymbol{\alpha}^j \\ K_{0\dots i} = \begin{bmatrix} \boldsymbol{\alpha}^0 \\ \vdots \\ \boldsymbol{\alpha}^i \end{bmatrix} & = [\alpha_0 \vec{K}_{t(0)} \dots \alpha_{N-1} \vec{K}_{N-1}] \triangleq \boldsymbol{\alpha} \\ Q_i & = [\theta_0^i \vec{K}_{t(0)} \dots \theta_{N-1}^i \vec{K}_{N-1}] \triangleq \boldsymbol{\theta}^i \end{cases}, \tag{7}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ are outputs of the networks. The form of the above equation resembles multi-head latent attention (MLA) [25], where the buckets can be viewed as a type of latent space. Note that $\boldsymbol{\alpha}$ remains a sequence and we need to accumulate $\boldsymbol{\alpha}$ to obtain $\boldsymbol{\beta}$ to represent the state.

$$Q_i K_{0\dots i}^\top = \boldsymbol{\theta}^i {\boldsymbol{\beta}^i}^\top = \boldsymbol{\theta}^i [\boldsymbol{\beta}_0^i \dots \boldsymbol{\beta}_{N-1}^i]^\top, \tag{8}$$

where

$$\boldsymbol{\beta}_t^i = \log(\sum_{j=0}^{i} e^{\alpha_t^j}). \tag{9}$$

The computation method for the value is the same as that in the training-free approach and will be omitted here.

4. Implementation

4.1. Training-Free Approach

The implementation details that need to be considered consist of two parts: one involves the partitioning of buckets, and the other pertains to the bucket attention. We utilize KNN to partition the buckets. We implement bucket attention according to Equations 5, during which we only need to maintain a fixed-size cache of $\vec{K}_t$, $\bar{V}_t$ and β_t.

4.2. Training From Scratch

Based on the formulas and analysis in Section 3.3, compared to the traditional implementation of attention mechanisms, we only need to redefine an additive operation for features. Then, we can replace individual elements with the exponential cumulative sum of these elements. The exponential cumulative sum is defined in Equations 5 and 9.

5. Conclusions

In conclusion, our analysis of attention mechanisms has led to a groundbreaking insight that sequential inputs within these mechanisms do not need to adhere to a strict order. We have successfully introduced bucket attention, a novel approach that organizes context within large language models (LLMs) and adeptly manages contexts of varying lengths using fixed-size spaces. Additionally, we have developed techniques to transform pre-trained models based on traditional attention into the bucket attention paradigm, accompanied by methodologies for training models with bucket attention from the ground up. These advances provide practical solutions to significantly enhance the efficiency and scalability of LLMs, paving the way for more robust and adaptable models in the future.

 doi:10.20944/preprints202508.1313.v2

References

1. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Advances in neural information processing systems* **2017**, *30*.
2. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), 2019, pp. 4171–4186.
3. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al. Improving language understanding by generative pre-training **2018**.
4. Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* **2025**.
5. Ha, D.; Dai, A.M.; Le, Q.V. HyperNetworks. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017, pp. 1–18.
6. Ye, Z.; Xia, M.; Yi, R.; Zhang, J.; Lai, Y.K.; Huang, X.; Zhang, G.; Liu, Y.j. Audio-driven talking face video generation with dynamic convolution kernels. *IEEE Transactions on Multimedia* **2022**, *25*, 2033–2046.
7. Ye, Z.; Sun, Z.; Wen, Y.H.; Sun, Y.; Lv, T.; Yi, R.; Liu, Y.J. Dynamic neural textures: Generating talking-face videos with continuously controllable expressions. *arXiv preprint arXiv:2204.06180* **2022**.
8. Schug, S.; Kobayashi, S.; Akram, Y.; Sacramento, J.; Pascanu, R. Attention as a hypernetwork. *arXiv preprint arXiv:2406.05816* **2024**.
9. Dherin, B.; Munn, M.; Mazzawi, H.; Wunder, M.; Gonzalvo, J. Learning without training: The implicit dynamics of in-context learning. *arXiv preprint arXiv:2507.16003* **2025**.
10. Roy, A.; Saffar, M.; Vaswani, A.; Grangier, D. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics* **2021**, *9*, 53–68.
11. Tay, Y.; Bahri, D.; Yang, L.; Metzler, D.; Juan, D.C. Sparse sinkhorn attention. In Proceedings of the International conference on machine learning. PMLR, 2020, pp. 9438–9447.
12. Sun, Z.; Yang, Y.; Yoo, S. Sparse attention with learning to hash. In Proceedings of the International Conference on Learning Representations, 2021.
13. Ge, S.; Zhang, Y.; Liu, L.; Zhang, M.; Han, J.; Gao, J. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801* **2023**.
14. Shi, L.; Zhang, H.; Yao, Y.; Li, Z.; Zhao, H. Keep the cost down: A review on methods to optimize LLM's KV-cache consumption. *arXiv preprint arXiv:2407.18003* **2024**.
15. Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.V.; Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860* **2019**.
16. Beltagy, I.; Peters, M.E.; Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* **2020**.
17. Zaheer, M.; Guruganesh, G.; Dubey, K.A.; Ainslie, J.; Alberti, C.; Ontanon, S.; Pham, P.; Ravula, A.; Wang, Q.; Yang, L.; et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems* **2020**, *33*, 17283–17297.
18. Petrick, F.; Rosendahl, J.; Herold, C.; Ney, H. Locality-sensitive hashing for long context neural machine translation. In Proceedings of the Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022), 2022, pp. 32–42.
19. Liu, M.; Rabbani, T.; O'Halloran, T.; Sankaralingam, A.; Hartley, M.A.; Huang, F.; Fermüller, C.; Aloimonos, Y. HashEvict: A Pre-Attention KV Cache Eviction Strategy using Locality-Sensitive Hashing. *arXiv preprint arXiv:2412.16187* **2024**.
20. Pappagari, R.; Zelasko, P.; Villalba, J.; Carmiel, Y.; Dehak, N. Hierarchical transformers for long document classification. In Proceedings of the 2019 IEEE automatic speech recognition and understanding workshop (ASRU). ieee, 2019, pp. 838–844.
21. Nawrot, P.; Tworkowski, S.; Tyrolski, M.; Kaiser, Ł.; Wu, Y.; Szegedy, C.; Michalewski, H. Hierarchical transformers are more efficient language models. *arXiv preprint arXiv:2110.13711* **2021**.
22. Liu, Y.; Lapata, M. Hierarchical transformers for multi-document summarization. *arXiv preprint arXiv:1905.13164* **2019**.
23. Ye, Z. The Sleep Mechanism of LLMs. *Preprints* **2025**. https://doi.org/10.20944/preprints202508.0071.v2.

 doi:10.20944/preprints202508.1313.v2

24. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE transactions on information theory* **1967**, *13*, 21–27.
25. Liu, A.; Feng, B.; Wang, B.; Wang, B.; Liu, B.; Zhao, C.; Dengr, C.; Ruan, C.; Dai, D.; Guo, D.; et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434* **2024**.