Hardware Friendly Transformer Optimization with Dynamic Attention Matrix Fusion

Qingyao Yang[1,2], Xiaoqin Wang[1,2], Shushan Qiao[1,2] and Yumei Zhou[1,2]
[1]*Institute of Microelectronics of the Chinese Academy of Sciences, Chaoyang, Beijing, China*
[2]*School of Integrated Circuits, University of Chinese Academy of Sciences, Huairou, Beijing, China.*
Corresponding Author: Email:

***Abstract*—The multi-head self-attention (MHSA) is the core component of transformer, where dynamic matrix multiplications (DMM), particularly $Q \times K^T$ and $A' \times V$, pose significant challenges for hardware acceleration. To reduce DMM MACs, this paper proposes a Dynamic Attention Matrix Fusion (DAMF) method, which optimizes DMM from the attention algorithm. For $Q \times K^T$, a quadratic form fusion of $W^Q W^K$ weight matrices and an SVD approximation is introduced, transforming DMM into fewer scalar operations and eliminating the linear transformations for QK generation. For $A' \times V$, this paper propose approximating softmax using a Maclaurin series and power-of-2 as a shift factor, replacing $A' \times V$ with a hardware-friendly shift operation. Experimental results show that the proposed DAMF method does not cause significant accuracy loss in BERT-base. Additionally, compared to MHSA with the same configuration, DAMF reduces parameters by 1.99 times, DMM MACs by 284 times, and total MACs by 2.21 times.**

***Index Terms*—Transformer, SVD, Softmax, MultiHead Self-attention, Hardware-Software Co-design**

I. Introduction

Transformer is a highly performant deep neural network (DNN) [1], widely applied in natural language processing [2]–[5] and computer vision tasks [6]–[8]. The multi-head self-attention (MHSA) mechanism [9], which forms the core structure of the transformer, is designed to improve the model's ability to focus on different parts of the input sequence simultaneously. As the number of parameters and computational demands of transformers continue to increase, there is a growing need for specialized hardware accelerators to ensure efficient inference.

Dynamic matrix multiplication (DMM) in MHSA poses significant challenges for the design of transformer accelerators, especially those based on compute-in-memory (CIM) [10]. In MHSA, static matrix multiplication (SMM) involves the multiplication of activation values and weights, such as the linear transformations that generate Q, K, and V. In contrast, the operands of DMM are intermediate values, such as $Q \times K^T$ and $A' \times V$. The hardware design challenges posed by DMM can be summarized in the following two aspects below, also shown in Fig 1

- Additional Memory Access. In CIM macro, weights and inputs of DMM are both generated during runtime. This results in redundant memory access [11], [12] or requires a transpose buffer [13]–[15] to handle intermediate data efficiently.
- Much more power consumption. For instance, in MulTCIM [16], during the inference of BERT-base, the $Q \times K^T$ and $A' \times V$ account for 58.16% of the total power consumption. However, these computations represent only 4.06% of the total MACs.

Existing research on optimizing DMM has primarily focused on designing dynamic computing engines [12], [16], [17] and wordline-feeding [18] to achieve high-speed writing and data transposition. However, hardware-level approaches alone cannot directly reduce the MACs of DMM.

Based on the existing facts, this study examines the data flow within MHSA, aiming to merge and approximate DMM-related computations. This approach results in reduced computations and parameters, which is significantly inspiring for alleviating memory access and power consumption challenges.

Optimization for $Q \times K^T$. The attention score calculation includes two steps: the linear transformations to generate Q and K, and the DMM $Q \times K^T$. This paper proposes a method to combine these steps into a single matrix operation with quadratic form. By using singular value decomposition (SVD) on this quadratic matrix, we approximate the computation with the singular vector of the largest singular value. As a result, calculating a single attention score only requires two vector inner products and one scalar multiplication, significantly reducing DMM MACs.

Optimization for $A' \times V$. The matrix A' represents attention weights, derived from the normalization and softmax of attention scores. This paper introduces a Maclaurin series and power-of-2 approximation [19], [20] to approximate softmax as shift factors. This approach replaces the DMM $A' \times V$ with shift operations, while eliminating the exponential and division computations introduced by softmax, making it more hardware-friendly.

The above strategies constitute the dynamic attention matrix fusion (DAMF) method proposed in this paper, with the following contributions:

- Introduction a Q, K weight matrices fusion with a quadratic form and SVD approximation. This fusion eliminates both the Q, K generation and the $Q \times K^T$.
- Proposed an approximation of $A' \times V$ with Maclaurin series and power-of-2 method, replacing the softmax and $A' \times V$ with shift operations.

Identify applicable funding agency here. If none, delete this.

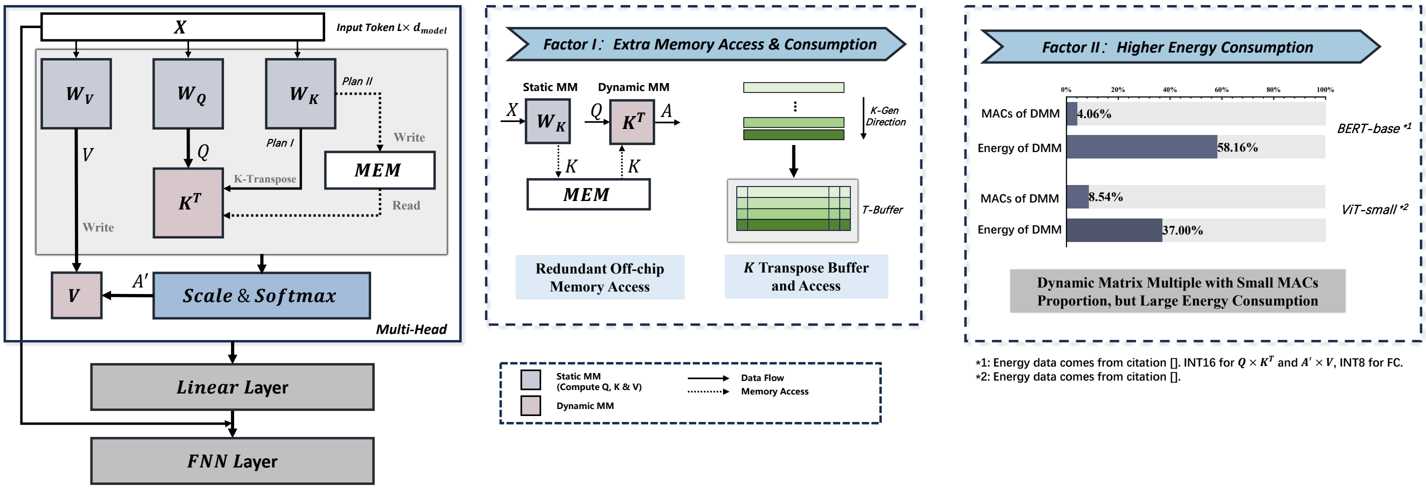

Fig. 1. The challenges caused by DMM.

- Experiment on BERT-base and the GLUE dataset demonstrates the effectiveness of the DAMF method, significantly reducing the parameters and DMM MACs without substantial accuracy loss.

II. Relative Works

In MHSA, the input sequence is linearly transformed into multiple sets of queries Qs, keys Ks, and values Vs, each corresponding to a different head. These sets are processed independently, allowing each head to focus on different parts of the sequence or capture various features.

In a single head, the attention scores are computed by taking the dot product of the Q and K vectors, followed by a softmax operation to obtain attention weights, which are subsequently used to weigh the V vectors. Then the outputs from all heads are then concatenated and linearly transformed to produce the final output.

Mathematically, for an input sequence of token length L and model dimensionality d_{model}, X is the input token, the multi-head attention mechanism can be expressed as follows.

The linear projection of Q, K, V, where W^{Qi}, W^{Ki}, W^{Vi} are the weight matrices for the $i-th$ head, and n_{head} is the number of attention heads.

$$Q^i = XW^{Qi}, \quad K^i = XW^{Ki}, \quad V^i = XW^{Vi} \tag{1}$$

The attention output of single head as follow, where $d_{head} = \frac{d_{model}}{n_{head}}$ is the dimensionality of each head.

$$\text{Attention}_i = \text{softmax}\left(\frac{Q^i K^{iT}}{\sqrt{d_{head}}}\right) V^i \tag{2}$$

Denote $A = Q^i K^{iT}$, and $A' = \text{softmax}(\frac{A}{\sqrt{d_{head}}})$.

Concatenation and final linear projection as follow, where W^O is the output projection matrix

$$\text{MultiHead} = \text{Concat}(\text{Attention}_1, \cdot\cdot, \text{Attention}_{n_{head}})W^O \tag{3}$$

III. Method

A. DAMF for $Q \times K^T$

The overview of $Q \times K^T$ dynamic attention matrix fusion can be summarized in Fig 2.

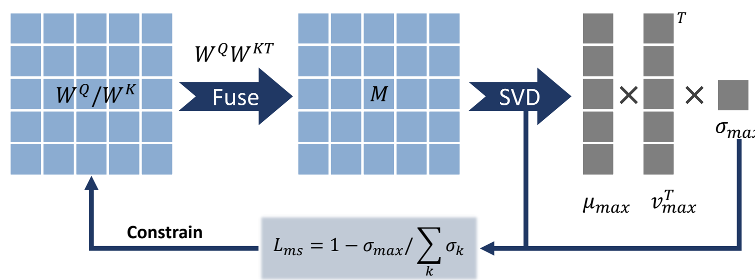

Fig. 2. The overview of DAMF for $Q \times K^T$.

According to formula (1) (2), a score in the attention score matrix A can be expressed as (4). The m, n is the index of input token, and A_{mn} represents the element in matrix A

$$\begin{aligned} A_{mn} &= (X_m W^Q)(X_n W^K)^T \\ &= X_m W^Q W^{KT} X_n^T \end{aligned} \tag{4}$$

Let $M = W^Q(W^{KT})$, the formula (4) can be further simplified as an expression in quadratic form:

$$A_{mn} = X_m M X_n^T \tag{5}$$

To simplify matrix computations, an approximating method for the matrix M by SVD decomposition, which can be expressed as formula (6), where U and V are orthogonal matrices, and Σ is a diagonal matrix containing the singular values of M.

$$M = U\Sigma V^T \tag{6}$$

M can be further expressed as a weighted sum of outer products of left and right singular vectors, as shown in formula

(7). Here, u_k, v_k is the $k-th$ column vector in U, V (left and right singular vector respectively), and σ_k is the $k-th$ singular value in Σ.

$$M = \sum_k \sigma_k \mu_k v_k^T \tag{7}$$

When the singular values in Σ are sparse, M can be approximated by the singular vectors corresponding to the largest singular values. Formula (5) can be simplified to expression (8).

$$\begin{aligned} A_{mn} &= \sigma_{max} X_m (\mu_{max} v_{max}^T) X_n^T \\ &= \sigma_{max} (X_m \mu_{max})(X_n v_{max})^T \end{aligned} \tag{8}$$

Where $\sigma_{max}, \mu_{max}, v_{max}$ represent the maximum singular value and the singular vectors.

Due to the normalization before softmax involving the division of maximum value of A, the multiplication by σ_{max} in the formula (8) can be omitted.

To induce sparsity in the singular values of Σ, a maximum singular value constraint is proposed, as shown in formula (9). Adding this loss function during training can increase the importance of the maximum singular value among all, thereby improving the approximation accuracy.

$$L_{MS} = 1 - \sigma_{max} / \sum_k \sigma_k \tag{9}$$

Fig 3 shows the data flow comparison between $Q \times K^T$ in original MHSA and the DAMF optimization. The linear transformations and $Q \times K^T$ are replaced by two vector inner products and two scalar multiplications.

With DAMF optimization, the DMM only involves a small amount of scalar operations. Besides, for the CIM units generating Q, K, it only needs to store the singular vectors with fewer parameters, instead of the W^Q, W^K matrix.

B. DAMF for $A' \times V$

After generation of attention score matrix A, it is normalized by its maximum value A_{max} before inputting into Softmax.

$$Norm(A_{mn}) = A_{mn}/A_{max} \tag{10}$$

Softmax is a crucial component in MHSA, and can be expressed as (11). Here, it transfers each row vector in the normalized attention score matrix A into a probability vector, emphasizing the maximum value within. This operation, followed by matrix product with V, highlights significant components in V.

$$A'_{mn} = softmax(Norm(A_{mn})) = \frac{e^{A_{mn}/A_{max}}}{\sum_n e^{A_{mn}/A_{max}}} \tag{11}$$

According to the expansion of MacLaurin series:

$$e^x = 1 + x + \frac{x^2}{2!} + \cdot\cdot\cdot\cdot\cdot\cdot \frac{x^2}{n!} + \mathcal{O}(x^n) \tag{12}$$

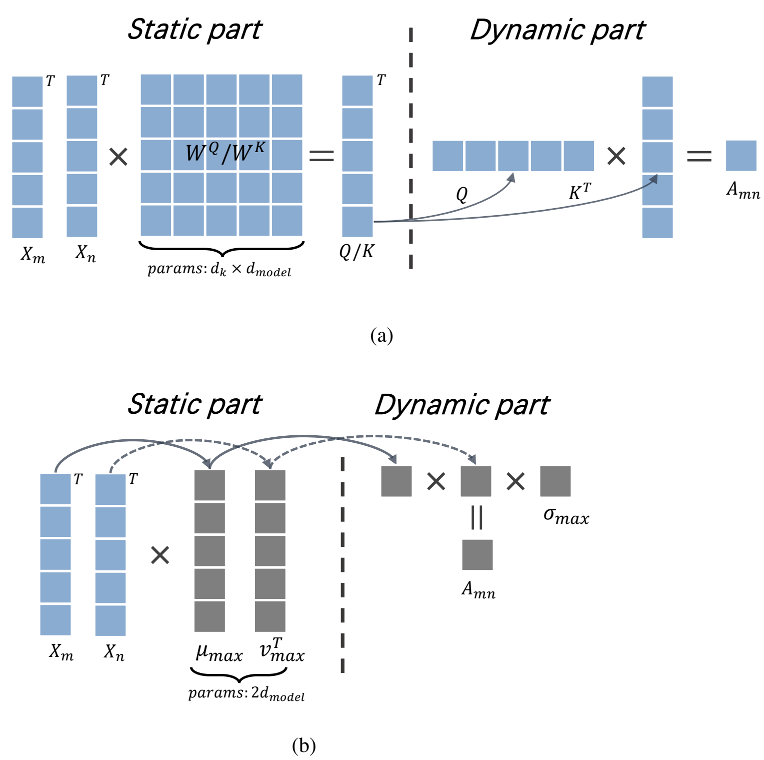

Fig. 3. $Q \times K^T$ comparison between original MHSA and DAMF. (a) $Q \times K^T$ data flow in original MHSA. (b) $Q \times K^T$ data flow with DAMF.

Let $e^{A_{mn}/A_{max}} \approx 1 + A_{mn}/A_{max}$, formula (11) can be expressed as:

$$\begin{aligned} A'_{mn} &= \frac{1 + A_{mn}/A_{max}}{\sum_{n=1}^{L} (1 + A_{mn}/A_{max})} \\ &= \frac{A_{max} + A_{mn}}{LA_{max} + \sum_{n=1}^{L} A_{mn}} \end{aligned} \tag{13}$$

According to formula (13), $A' \times V$ can be represented in the form of element-wise multiplication:

$$A'_{mn} V_{nm} = \frac{V_{nm} A_{max} + V_{nm} A_{mn}}{LA_{max} + \sum_{n=1}^{L} A_{mn}} \tag{14}$$

Then the attention score A_{mn}, A_{max} can be rounded to the nearest power-of two value 2^I.

$$A_{mn} \approx 2^{I_{mn}}, \ A_{max} \approx 2^{I_{max}} \tag{15}$$

Similarly, the denominator in formula (14) is approximated by the nearest power-of-two.

$$LA_{max} + \sum_{n=1}^{L} A_{mn} \approx 2^{I_d} \tag{16}$$

By replacing the power-of-two with hardware-friendly shift operations, A_{mn}, A_{max} can be approximated as three shifts as (17). The shift factor I_{mn}, I_{max}, I_d is derived from the softmax approximation.

$$A'_{mn}V_{nm} \approx (V_{nm} << I_{max} \ + \ V_{nm} << I_{mn}) >> I_d \quad (17)$$

The overview of DAMF optimization for $A' \times V$ is demonstrated as Fig 4.

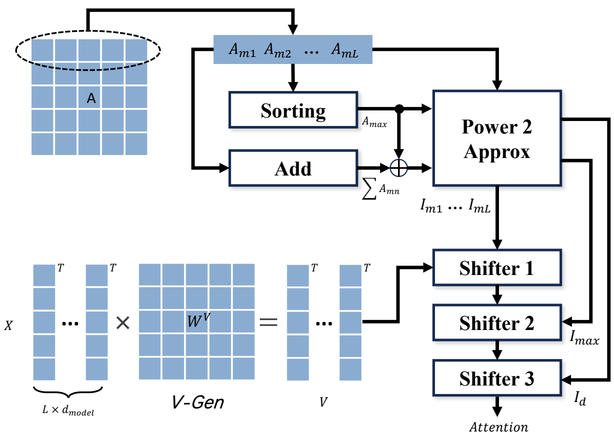

Fig. 4. The overview of DAMF for $A' \times V$.

IV. Experiment

A. Set up

In this experiment, BERT-base [21] is selected as the baseline, and the proposed DAMF is applied to it. To comprehensively evaluate the effectiveness of the proposed method, we compared it against various versions of BERT-base, including TinyBERT [22], SkipBERT [23], BERT-PKD [24], and BERTsmall. These models are designed and widely used to reduce the parameters, computations, and processing latency of BERT.

The proposed method and the comparison methods are validated on the GLUE corpus [25], which includes tasks such as SST-2, MRPC, QQP, and RTE. These tasks cover a range of types, including single-sentence tasks, similarity and paraphrase tasks, and inference tasks.

B. Accuracy Loss

The experiment compared the percentage accuracy loss of all methods relative to BERT-base across the four datasets. The results, shown in Table I, indicate that the proposed method achieved the smallest accuracy loss on SST-2 and MRPC. On the RTE dataset, both the proposed method and TinyBERT achieved an approximately 5% improvement in task accuracy.

C. Parameters and MACs

The total number parameters and computations of a single MHSA layer across all methods are compared in Table II. The input token length is set to 128. In this experiment, TinyBERT achieved the lowest parameters and MACs due to its minimal hidden layer dimensions. The proposed method, however, achieved the fewest DMM MACs, while reducing the DMM proportion in total MACs to 0.02%. This represents a reduction of approximately 481 times compared to TinyBERT.

TABLE I
THE COMPARISON OF ACCURACY LOSS ON FOUR TASKS

Methods	Acc. Loss ↓*1			
	SST-2	MRPC	QQP	RTE
TinyBERT*2	0.96%	1.01%	**-0.14%**	**-4.97%**
SkipBERT*2	2.78%	3.37%	1.40%	4.22%
BERT-PKD*2	4.39%	7.09%	1.40%	6.17%
BERTsmall*3	4.06%	6.19%	4.35%	6.93%
BERT+DAMF*4	**0.28%**	**0.69%**	4.11%	-4.79%

*1 Percentage accuracy loss compared to BERT-base.
*2 The result of TinyBERT, SkipBERT and BERT-PKD from reference [23].
*3 The result of BERTsmall from https://github.com/google-research/bert.
*4 The proposed method applied on BERT-base.

TABLE II
THE COMPARISON OF TOTAL PARAMETERS AND MACS IN MHSA

	Param. /M*1	Dyn. MACs/M*1	MACs/M*1	Dyn. Ratio
TinyBERT*2	**1.49**	5.31	**55.15**	9.63%
SkipBERT*3	9.00	12.78	314.77	4.06%
BERT-PKD*3	9.00	12.78	314.77	4.06%
BERTsmall*4	4.00	8.52	142.74	5.97%
BERT+DAMF*3	4.51	**0.03**	153.39	**0.02%**

*1 The total parameters and MACs are evaluated at token size 128, with single MHSA.
*2 TinyBERT with $d_{model} = 312, n_{head} = 12$.
*3 SkipBERT, BERT-PKD and ours with $d_{model} = 768, n_{head} = 12$, same as BERT-base.
*4 BERTsmall with $d_{model} = 512, n_{head} = 8$.

Unlike TinyBERT and BERTsmall, the proposed DAMF method does not change the number of attention heads (n_{head}) or the hidden layer dimensions (d_{model}). Compared to SkipBERT and BERT-PKD, which have the same hyperparameter settings, DAMF achieves a more significant reduction in DMM MACs, approximately 426 times lower. Additionally, DAMF exhibits much smaller accuracy loss across the four tasks compared to SkipBERT and BERT-PKD.

V. Conclusion

The proposed DAMF optimizes $Q \times K^T$ and $A' \times V$ from an algorithmic perspective. Firstly, by leveraging W^Q, W^K weight matrix fusion and SVD approximation, it merges the linear transformation and $Q \times K^T$ into vector inner products and scalar operations, reducing both the parameters and DMM computations. Secondly, it approximates the softmax as shift factors based on the Maclaurin series and power-of-two, and replaces the entire $Q \times K^T$ with hardware-friendly shift operations.

Experimental results show that the proposed method applied to BAER-base does not introduce significant accuracy loss. Compared to MHSA with the same configuration, it reduces the parameter count by 1.99 times, DMM MACs by 284 times, and total MACs by 2.21 times.

REFERENCES

[1] S. Islam et al., "A comprehensive survey on applications of transformers for deep learning tasks," in *Expert Syst. Appl.*, vol. 241, p. 122666, May 2024.

[2] W. Fedus, B. Zoph, and N. Shazeer, "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity," in *J. Mach. Learn. Res.*, vol. 23, no. 120, pp. 1-39, 2022.

[3] L. Dong, S. Xu, and B. Xu, "Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition," in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 5884-5888.

[4] T. Afouras, J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, "Deep Audio-Visual Speech Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 8717-8727, Dec. 2022.

[5] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai, "ASTER: An Attentional Scene Text Recognizer with Flexible Rectification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2035-2048, Sep. 2019.

[6] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling Vision Transformers," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 1204-1213.

[7] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for Semantic Segmentation," in *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 7242-7252.

[8] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *Int. Conf. Learn. Representations (ICLR)*, Oct. 2020.

[9] A. Vaswani et al., "Attention is All you Need," in *Advances in Neural Information Processing Systems, Curran Associates, Inc.*, 2017.

[10] H. You, W. Li, D. Shang, Y. Zhou, and S. Qiao, "A 1–8b Reconfigurable Digital SRAM Compute-in-Memory Macro for Processing Neural Networks," in *IEEE Trans. Circuits Syst. I: Regular Papers*, pp. 1–13, 2024.

[11] R. Guo et al., "CIMFormer: A Systolic CIM-Array-Based Transformer Accelerator With Token-Pruning-Aware Attention Reformulating and Principal Possibility Gathering," in *IEEE J. Solid-State Circuits*, pp. 1–13, 2024.

[12] F. Tu et al., "TranCIM: Full-Digital Bitline-Transpose CIM-based Sparse Transformer Accelerator With Pipeline/Parallel Reconfigurable Modes," in *IEEE J. Solid-State Circuits*, vol. 58, no. 6, pp. 1798–1809, Jun. 2023.

[13] M. Huang, J. Luo, C. Ding, Z. Wei, S. Huang, and H. Yu, "An Integer-Only and Group-Vector Systolic Accelerator for Efficiently Mapping Vision Transformer on Edge," in *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 70, no. 12, pp. 5289–5301, Dec. 2023.

[14] F. Tu et al., "A 28nm 15.59µJ/Token Full-Digital Bitline-Transpose CIM-Based Sparse Transformer Accelerator with Pipeline/Parallel Reconfigurable Modes," in 2022 *IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Feb. 2022, pp. 466–468.

[15] T. Li, F. Zhang, X. Fan, J. Shen, W. Guo, and W. Cao, "Unified Accelerator for Attention and Convolution in Inference Based on FPGA," in 2023 *IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2023, pp. 1–5.

[16] F. Tu et al., "MulTCIM: Digital Computing-in-Memory-Based Multimodal Transformer Accelerator With Attention-Token-Bit Hybrid Sparsity," in *IEEE J. Solid-State Circuits*, vol. 59, no. 1, pp. 90–101, Jan. 2024.

[17] S. Sridharan, J. R. Stevens, K. Roy, and A. Raghunathan, "X-Former: In-Memory Acceleration of Transformers," in *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, vol. 31, no. 8, pp. 1223–1233, Aug. 2023.

[18] J.-W. Su et al., "A 28nm 64Kb Inference-Training Two-Way Transpose Multibit 6T SRAM Compute-in-Memory Macro for AI Edge Chips," in 2020 *IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Feb. 2020, pp. 240–242.

[19] F. Spagnolo, S. Perri, and P. Corsonello, "Aggressive Approximation of the SoftMax Function for Power-Efficient Hardware Implementations," in *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 69, no. 3, pp. 1652–1656, Mar. 2022.

[20] Y. Zhang et al., "Base-2 Softmax Function: Suitability for Training and Efficient Hardware Implementation," in *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 69, no. 9, pp. 3605–3618, Sep. 2022.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *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)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[22] X. Jiao et al., "TinyBERT: Distilling BERT for Natural Language Understanding," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He, and Y. Liu, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 4163–4174.

[23] J. Wang, K. Chen, G. Chen, L. Shou, and J. McAuley, "SkipBERT: Efficient Inference with Shallow Layer Skipping," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 7287–7301. doi: 10.18653/v1/2022.acl-long.503.

[24] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient Knowledge Distillation for BERT Model Compression," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4323–4332.

[25] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, T. Linzen, G. Chrupała, and A. Alishahi, Eds., Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355.