Exploring Approximation and Dataflow Co-Optimization for Scalable Transformer Inference Architecture on the Edge

Liu He*, Yujin Wang*, Zongle Huang*, Shupei Fan*, Chen Tang*, Shuyuan Zhang*, Luchang Lei*, Huazhong Yang*†, Yongpan Liu*†, Hongyang Jia*†
*Department of Electronic Engineering, Tsinghua University, Beijing, China
†BNRist, Tsinghua University, Beijing, China
hjia@tsinghua.edu.cn

2024 IEEE 37th International System-on-Chip Conference (SOCC) | 979-8-3503-7756-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/SOCC62300.2024.10737793

***Abstract*—Transformer-based neural networks (NNs) prevail in today's artificial intelligence applications, including autonomous driving, natural language processing and generative modeling, showing superior accuracy and generalization over traditional deep-learning models. However, the quadratic scaling computation and complex dataflow in the self-attention set challenges to the efficient deployment of Transformer-based NNs on edge and edge-server devices, where the latency of single-batch inference is a critical concern. The lack of data parallelism necessitates exploring more dimensions in tensor parallelism, more specifically, sequence parallelism in transformer inference for strong scaling in domain-specific accelerator (DSA) design, which is non-trivial due to the temporal dependency of the max-finding in softmax operators. This work formulates these challenges into an on-chip buffering problem, and then puts forward a hardware-software co-design approach exploiting max-finding-free approximation for softmax operators, which removes the blocking of the inference pipeline and thus alleviates the on-chip buffering pressure. An example architecture design shows up to 2.83× and 28.02× speedup, over the baseline DSA designs respectively, with negligible algorithmic performance loss.**

***Index Terms*—Transformer, SW-HW Co-design, Approximate Softmax, Scalability**

I. Introduction

Transformer-based neural networks (NNs) have been the backbones of recent artificial intelligence (AI) applications, including autonomous driving [1], [2], natural language processing (NLP) [3], [4] and computer vision (CV) [5], [6]. The exceeding performance of Transformer NNs comes at the cost of extraordinarily high complexity in computation and storage, for both training and inference. Efficient Transformer inference is a key concern for edge and edge-server computing platforms. However, executing Transformer-based NNs at the desired speed is still challenging for state-of-the-art general-purpose edge-server compute devices, necessitating customized architecture and dataflow design [7]–[9].

Performing efficient Transformer inference with stringent constraints in edge scenarios, such as low data ingestion rate and limited off-chip bandwidth, poses critical challenges to architecture design and task scheduling. Unlike cloud inference, which may potentially take advantage of batching input and larger buffering resources, the inference at the edge usually handles single-batch input and prioritizes latency in metrics for the overall system. Losing the knob of batch-level parallelism motivates aggressive exploitation of all the other parallelisms in Transformer NNs, including pipeline parallelism and tensor parallelism. Among those, sequence parallelism, which segments the sequence dimension in Transformer operators and maps them to separate processing engines, has the potential to greatly improve inference efficiency and hardware scalability.

However, the native dataflow of self-attention poses difficulties in exploiting sequence parallelism. This is mainly due to the max-finding step in the softmax operator requiring buffering of the complete rows in the attention matrix, whose size is proportional to the sequence length. Holding multiple attention rows for sequence-parallel execution escalates the buffering issue. Simplification of the individual softmax function, i.e., quantization [10] and piecewise-linear approximation [11], does not guarantee solving this pipeline-blocking issue. Instead, co-optimization of algorithm approximation error and hardware parallelization scheme is mandatory to achieve scalable and low-latency Transformer inference on the edge. This work proposes an end-to-end architecture enabling fine-grained sequence parallelism with fully low-bit-width softmax execution achieving negligible accuracy loss. It reduces the buffering demands of the attention matrix and associated memory-access overhead, which significantly improves the system-level compute density. The key contributions are:

1) At the algorithm level, this work exploits a linear-after-saturation approximate softmax scheme to eliminate the max-finding step and enable fully low-bit-width self-attention execution with negligible accuracy drop.
2) At the micro-architecture level, this work proposes a hybrid dense-sparse compute core for the linear approximate dataflow, which enables layer fusion between matrix multiplications (MMs) in self-attention with distributed softmax evaluation and greatly reduces memory access for the attention matrix.
3) At the system architecture level, this work gives an end-

979-8-3503-7756-9/24/$31.00 ©2024 IEEE

Authorized licensed use limited to: Tsinghua University. Downloaded on April 23,2025 at 03:04:16 UTC from IEEE Xplore. Restrictions apply.

to-end architecture exploiting fine-grained sequence parallelism with reduced intermediate data movement and buffering requirements on the attention matrix, showing strong scaling to the number of compute cores.

Compared to two baselines domain-specific accelerator (DSA) designs sharing architectural ideas with a state-of-the-art Transformer accelerator [9], the proposed solution achieves up to 2.83× and 28.02× speedup with 1.31× and 3.87× energy efficiency, respectively.

II. Related Work

A. Optimizations on Softmax Kernel

Many pioneer works [10]–[13] focus on accelerating heavily non-linear softmax kernel. The post-processing unit (PPU) in [10] is co-optimized with a base-2 exponential function, enabling hardware-friendly shifting-based softmax evaluation. Using a principle of Taylor expansion, [11] exploits piece-wise-linear approximation of the exponential function with look-up tables (LUTs). The online normalization scheme in [12] combines max-finding and denominator accumulation into one pass to reduce local memory access, but still requires row buffering of the attention matrix. The scope of these efforts primarily confines in optimizing the softmax function itself while overlooking the significant memory access overhead and hardware storage demands imposed by system scale-up at large sequence lengths.

B. Optimizaitons on Transformer Dataflow

Many studies also focus on optimizing Transformer dataflow using GPUs and DSAs. FlashAttention [14] decouples the temporal dependency in softmax and enables sequence parallelism while reducing access to GPU high-bandwidth memory (HBM), achieved by updating the max value and recomputing softmax denominators. However, this approach introduces recomputation overhead. FlashDecoding++ [15] incorporates further optimizations to FlashAttention by utilizing a unified maximum value in the softmax operation, thereby minimizing frequent data updates. However, it utilizes a full-precision (FP32) implementation of softmax, which cannot directly translate to DSAs, especially when considering the complex recompute dataflow benefiting from the programmability of GPU platforms. DOTA [8], a work of DSA also addresses sequence (token) parallelism, however, it confines sequence parallelism within a core, thus constraining scalability. This paper discusses sequence parallelism to explore strong scaling across the Transformer processing core array.

III. Background and Motivation

A typical Transformer encoder block comprises self-attention layers and linear-projection layers. The self-attention consists of two MM layers with dynamic matrices generated from preceding linear layers, and a softmax layer inserted between them. As shown in Fig. 1 left, it receives three matrices Q, K, V, and $Q \cdot K^T$ is firstly computed to produce attention matrix X whose size is quadratic to the sequence length. Then it applies X to the softmax unit to obtain the attention score A, which is then multiplied by V to produce the attention output O. A standard softmax operation is denoted in Algorithm 1, where s is the sequence length. The self-attention module, especially the softmax kernel, dominates Transformer computation as sequence length increases, illustrated in Fig. 1 right.

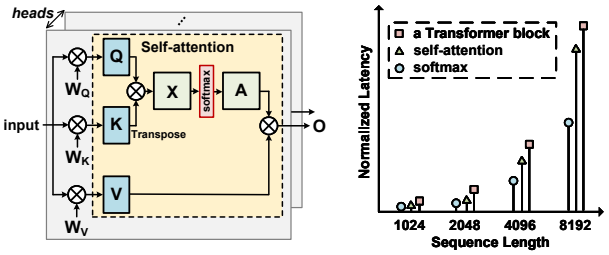

Fig. 1: Illustration of Transformer attention module (left) and the normalized execution latency of a Transformer block, self-attention, and softmax kernel respectively, when sequence length increases (right).

As discussed later, the regular form of softmax introduces excessive intermediate data movement and infeasible attention matrix buffering requirements when exploiting sequence parallelism in edge DSAs.

A. *Intermediate Data Movement in Softmax*

The first bottleneck arises due to data movement issues at the level of the core. Standard softmax involves an element-wise subtraction of $x_{i,\max}$ from the vector X_i prior to the exponential function to prevent data overflow, which usually requires a full-precision format as well. However, the three sequential steps of softmax (Algorithm 1 Step1-3) require scanning through the entire row of the attention matrix, resulting in excessive memory access. This necessitates low-bit-width softmax computation with the fusion of MM layers, hiding excessive intermediate attention matrix access.

Algorithm 1 Standard Softmax on vector X_i

$x_{i,\max} \leftarrow x_{i,1}$
$\text{denominator} \leftarrow 0$
for $j \leftarrow 1, s$ **do Step1**
  $x_{i,\max} \leftarrow \max(x_{i,\max}, x_{i,j})$
end for
for $j \leftarrow 1, s$ **do Step2**
  $\text{denominator} \mathrel{+}= \exp(x_{i,j} - x_{i,\max})$
  $y_{i,j} \leftarrow \exp(x_{i,j} - x_{i,\max})$
end for
for $j \leftarrow 1, s$ **do Step3**
  $y_{i,j} \leftarrow \frac{y_{i,j}}{\text{denominator}}$
end for

B. *On-chip Buffering for Sequence Parallelism*

The inter-row independence in the attention matrix helps sequence parallelism, which enables further compute scaling after exploiting the parallelization of attention heads. Fig. 2

Authorized licensed use limited to: Tsinghua University. Downloaded on April 23,2025 at 03:04:16 UTC from IEEE Xplore. Restrictions apply.

illustrates the partitioning of one head in the attention matrix multiplication. Each core is responsible for multiplying a group of sub-matrices, namely $Q_{\mathrm{x-y}} \cdot K^T = X_{\mathrm{x-y}}$ and $A_{\mathrm{x-y}} \cdot V = O_{\mathrm{x-y}}$. The temporal dependency of max-finding prohibits $A \cdot V$ from pipelining before completing rows of the attention matrix.

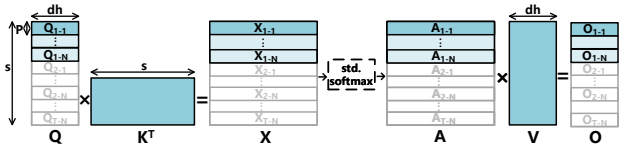

Fig. 2: Partition of one attention head for sequence parallelism, with sequence length s, head embedding dimension dh, partition length P, sequence parallelism N, and iterations for computing one head T.

Storing those pipeline intermediate data in bandwidth-limited and energy-inefficient off-chip DRAM causes severe under-utilization. Meanwhile, the proportional increase to both sequence length and parallelism on buffering requirements prohibits storing intermediate data fully on-chip, which otherwise could make SRAM dominate the area and degrade system-level compute density. This creates a dilemma between achieving high sequence parallelism for strong scaling and maintaining a feasible hardware configuration at the edge. This challenge, originating from max-finding operators, necessitates a holistic redesign of both the algorithm and the end-to-end architecture.

IV. Computing Scheme of Approximate Softmax

Buffering of entire rows of attention matrix comes from (1) finding the maximum value and (2) accumulating the denominator for normalization, both of which require global information across full rows. To address these issues, the proposed approximate softmax computing scheme omits the max-finding operation and defers normalization, employing two major techniques as elaborated in the subsequent subsections.

A. Saturation-Approximate Softmax

To eliminate the max-finding step, which only contributes to prevent exponentiation overflow, the proposed softmax employs a conditional linear approximation for the exponentiation operator. It uses a pre-compiled threshold: values below this threshold pass through an FP8 exponential function, while larger values are approximated by the tangent line at the threshold point, as shown in Fig. 3. The linear approximation incorporates only affine transformation, whose computing results are held as tuples of input integer, shared scaling factor and bias, waiting for fusing with subsequent linear operations. This approximation scheme is defined in Equation Eq.1 as well as the mathematical format of saturation-approximate softmax (SA-softmax) in Eq.2.

$$\mathrm{SA}-\exp(X_i) = \begin{cases} e^{x_{i,j}}, & x_{i,j} \le x_{\mathrm{th}} \\ k \cdot x_{i,j} + b, & x_{i,j} > x_{\mathrm{th}} \end{cases} \quad j = 1, ..., s \tag{1}$$

$$\mathrm{SA}-\mathrm{softmax}(X_i) = \begin{cases} \dfrac{e^{x_{i,j}}}{\sum \mathrm{SA}-\exp(X_i)}, & x_{i,j} \le x_{\mathrm{th}} \\ \dfrac{k \cdot x_{i,j} + b}{\sum \mathrm{SA}-\exp(X_i)}, & x_{i,j} > x_{\mathrm{th}} \end{cases} \quad j = 1, ..., s \tag{2}$$

where $k = \lambda \cdot e^{x_{\mathrm{th}}}$ is the tangent slope of the exponent function at point $(x_{\mathrm{th}}, e^{x_{\mathrm{th}}})$ multiplies a hyperparameter λ, and b can be calculated as $e^{x_{\mathrm{th}}} - k \cdot x_{\mathrm{th}}$. This monotonic approximation introduces negligible error since softmax tends to emphasize larger numbers to 1, creating a one-hot style representation, which squeezes the absolute error for each exponentiation.

B. Fusing Softmax with Linear Layer

The elimination of the max-finding releases the pipeline between layers in self-attention, thus enabling efficient layer fusion for intermediate reduction. Fig. 3 illustrates the fusion of SA-softmax operation with the subsequent V-MM. The proposed dataflow contains two data paths: (1) X_{dense} data path with input values below x_{th} and (2) X_{sparse} data path with values exceeding x_{th} that are marked for linear approximation, and a binary vector X_{mask} that denotes the position of elements in X_{sparse}. For the X_{sparse} data path, we perform V-MM before the linear transformation of X_{sparse}. This order considers bit widths and potential overflow. For both data paths, the normalization step (division by the denominator) is postponed until after the V-MM, which is handled by PPU. This decouples the second temporal dependency in the standard softmax, thereby unblocking the attention pipeline and enabling the tiled execution of $Q \cdot K^T$ and $X \cdot V$. The overall operations for the modified softmax fused with subsequent MM can be expressed as shown in Eq.3.

$$o_{i,j} = \frac{e^{X_{\mathrm{dense}-i}} \cdot V_j + k \cdot (X_{\mathrm{sparse}-i} + \frac{b}{k}) \cdot V_j}{\sum e^{X_{\mathrm{dense}-i}} + \sum (k \cdot X_{\mathrm{sparse}-i} + b)}, \mathrm{i,j=1,\ldots,s} \tag{3}$$

where

$$x_{\mathrm{dense}-i,j} = \begin{cases} x_{i,j}, & \text{if } x_{i,j} \le x_{\mathrm{th}} \\ -\infty, & \text{otherwise} \end{cases} \tag{4}$$

$$x_{\mathrm{sparse}-i,j} = \begin{cases} x_{i,j}, & \text{if } x_{i,j} > x_{\mathrm{th}} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

V. Memory-Efficient Dense-Sparse Hybrid Architecture

Fig. 4 illustrates the overall architecture and mapping of this work. The proposed computational system comprises an array of Transformer-processing clusters (TrPCs), which implement head parallelism in the attention layer. Each TrPC consists

Authorized licensed use limited to: Tsinghua University. Downloaded on April 23,2025 at 03:04:16 UTC from IEEE Xplore. Restrictions apply.

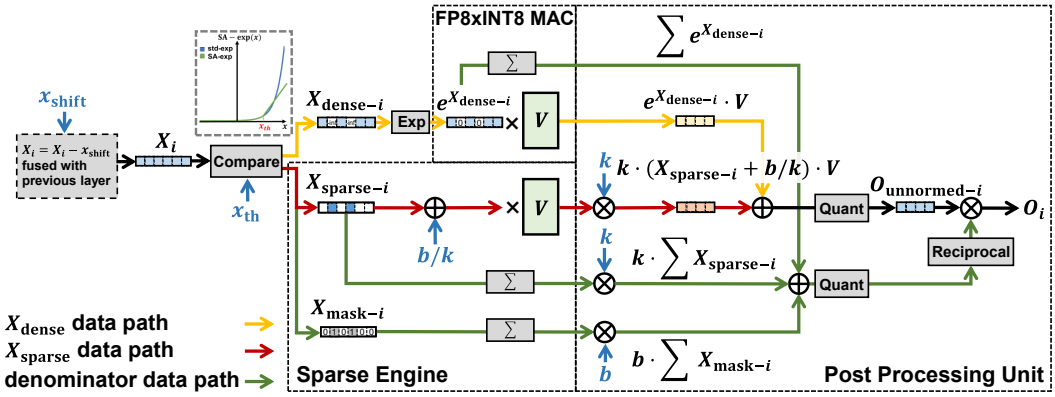

Fig. 3: Computation map of SA-softmax and layer fusion with subsequent V-MM. X_i is the result of previous $Q \cdot K^T$ layer. For simple illustration, a vector is used instead of a matrix as input.

of an array of Transformer-processing elements (TrPEs) and a shared memory. Two TrPEs form a TrPE-Pair to enable pipelined attention dataflow, as illustrated in Fig. 4 (b). Sequence parallelism is implemented among TrPE-Pairs.

To implement a fully-pipelined attention module and therefore relieve the storage requirements of the attention matrix X, we deploy $Q \cdot K^T$ in TrPE1 of Fig. 4 and the saturation-approximate softmax together with $X \cdot V$ is executed in TrPE2. The attention matrix tile transferred from TrPE1 to TrPE2 is of size $P \cdot R$ as illustrated in Fig. 4 (a). As Fig. 4 (c) shows, data from the output buffer of TrPE1 first goes through the Comparator of TrPE2 and diverges into two data paths. The dense data path goes through LUTs for exponentiation, $FP8 \times INT8$ MAC for multiplication with V, and then stored in the output buffer. The sparse path goes through the Sparse Engine before being stored in the output buffer. Finally, the dual paths converge in PPU for normalization and the final result is calculated.

A. Building Blocks of TrPE

1) $FP8 \times INT8$ ***MAC Array****:* As discussed in Sec. IV-A, we select FP8 precision for the values in the dense path after exponentiation. To support this, we design a $FP8 \times INT8$ MAC, which implements minor modifications to an ordinary MAC array. We insert a shifter after the 8-bit multiplier to expand the result to the correct value. The adder tree is limited to 24 bits for area savings without compromising model accuracy.

2) ***Sparse Engine****:* The Sparse Engine is designed to support the calculation of the sparse data path. The micro-architecture of the Sparse Engine is shown in the bottom middle part of Fig. 4 (c). It includes a 16-wide adder-multiplier-accumulator (AMA) for matrix multiplication and a 16-wide 16-bit accumulator (ACC) for denominator calculation.

3) ***Post Processing Unit****:* The Post Processing Unit is responsible for merging dense and sparse data paths and conducting normalization operations. The PPU contains a 16-wide INT16 multiplier and a 16-wide INT24 adder, with each input selected from 4 data signals. The denominator goes through the INT8 Reciprocal Unit and then multiplies by the numerator. The PPU is designed to be fully pipelined.

B. Scalability

The scalability of the entire system can be achieved by increasing the number of TrPE-Pairs within a TrPC and the number of TrPCs within the system. This augmentation enhances both sequence parallelism and head parallelism, thereby improving the architecture's performance in scenarios involving long-sequence lengths and large model sizes. In such cases, segments of the attention matrix can be stored locally within TrPE-Pairs, eliminating the need for communication with shared memory or off-chip DRAM during end-to-end attention execution. Our proposed approach especially facilitates on-chip execution of the attention module with limited on-chip memory requirements.

VI. Evaluation

In this section, we present both detailed algorithm experiments and architecture performance breakdown for low-latency edge inference of Transformer NNs.

A. Experimental Setup

1) ***Models and Datasets****:* The experiment comprises evaluations of typical Transformer NNs. The RoBERTa-Base model [16] on GLUE benchmark [17] covers edge NLP cases. For CV tasks, DeiT [18] and Swin Transformer [19] models are evaluated on the ImageNet-1K dataset [20].

2) ***Hardware Baseline****:* The baseline DSAs borrow the core-level micro-architecture from [9] with shifting-based softmax scheme described in [12], the local buffering SRAMs are aligned to our TrPE as shown in Tab. I. We design two baselines to explore the intermediate data movement and storage problem. DSA_S baseline hard-codes the global SRAM buffer in TrPCs to the peak buffering requirement set by the maximum sequence length, eliminating off-chip attention access, while, DSA_D aligns the size of the global SRAM buffer to our proposed architecture, showing the impact of DRAM offloading. Both our design and the baselines were configured to have the same silicon area.

Authorized licensed use limited to: Tsinghua University. Downloaded on April 23,2025 at 03:04:16 UTC from IEEE Xplore. Restrictions apply.

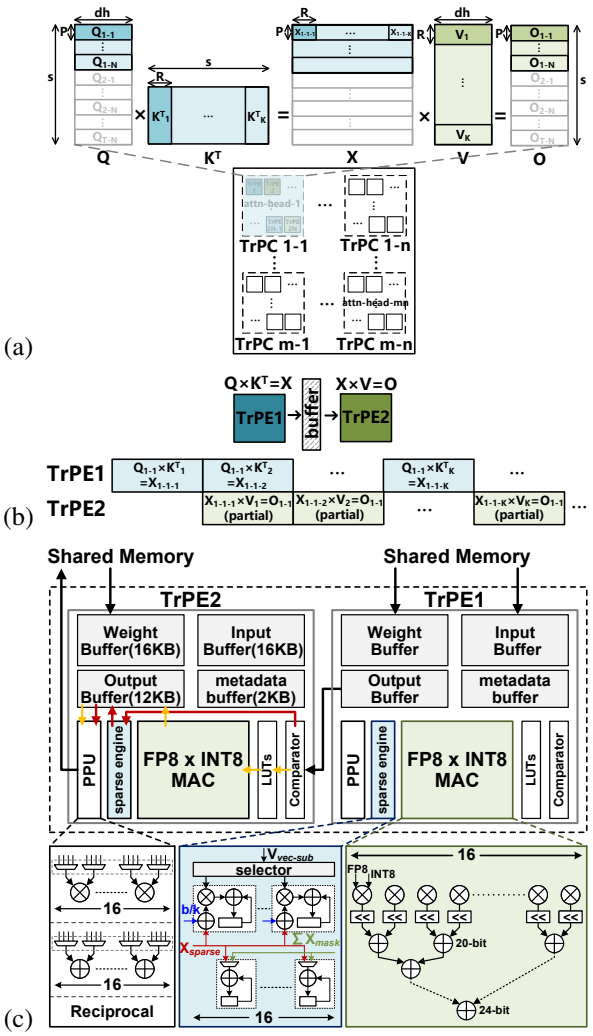

Fig. 4: (a) Mapping of attention module utilizing head parallelism and sequence parallelism. Since the two layers in the attention module are fused, X is used to denote both the attention matrix and the attention score. (b) Pipeline of the dataflow for one attention head with layer fusion of $Q \cdot K$ and $X \cdot V$. The modified softmax operation is excluded for simplicity. (c) Micro-architecture of TrPE.

TABLE I: TrPE Configuration for Evaluation

Comparator	Width	16
LUTs	Depth	256
$FP8 \times INT8$ MAC	Number of MAC lane	16
	Width of MAC lane	16
Sparse Engine	Number of ACC	16
	Number of AMA	16
PPU	Number of INT16 multiplier	16
	Number of INT24 adder	16
	Number of INT8 reciprocal	16
Buffer	Input buffer	16KB
	Weight buffer	16KB
	Output buffer	12KB
	Metadata buffer	2KB
Mapping parameter	P	64
	R	64

*3) **Software and Hardware Experiment Methodologies**:* We first evaluate standard $INT8$ quantized models exploiting widely used repositories, including I-BERT [21] for NLP models and FQ-ViT [22] for CV models, where traditional $FP32$ softmax operators are firstly kept as a comparison. We set hyperparameter $\lambda = 5$ and $P = 1\%$. Quantization-aware fine-tuning (QAT) helps recover the accuracy to the traditional $INT8$ NLP model, while CV tasks require post-training quantization (PTQ) only, confirming our design introduces minimal accuracy loss.

All functional modules in our TrPE and baseline cores are implemented in RTL and synthesized with Synopsys Design Compiler using 28nm CMOS technology. The whole system architecture takes silicon sizes around 200mm^2 and a clock frequency of 800MHz. The SRAMs in the core and cluster levels are synthesized with a foundry-provided high-density single-port compiler, while the cluster's buffer are made of 64KB blocks. The energy consumption of every module was estimated based on PrimeTimePX.

B. Algorithm Performance

Tab. II and Tab. III demonstrate the accuracy comparison of NLP and CV tasks, respectively. Evaluation of the RoBERTa-Base model confirms only a 0.30% accuracy drop on average compared to the baseline INT8 model. While this number increases to 0.94% on average for CV tasks, since PTQ is used instead of QAT in the NLP task. Our work reserves inference accuracy in all test cases while eliminating high-bit-width softmax computation.

TABLE II: Algorithm Accuracy on RoBERTa-Base model

GLUE Scores	RTE	SST-2	MNLI	QNLI	CoLA	QQP	MRPC	STS-B	Avg. drop
FP32	78.7	94.8	87.8	92.6	83.9	91.4	89.0	90.4	-
INT8+FP32 softmax	76.9	95.2	87.2	92.4	83.4	91.4	89.5	90.5	0.26 (cmp. to FP32)
This work	77.6	94.4	86.6	92.2	83.1	90.9	89.5	89.8	0.30 (cmp. to INT8)

TABLE III: Algorithm Accuracy on ImageNet-1K Dataset

Model	DeiT-T	DeiT-S	DeiT-B	Swin-T	Swin-S	Swin-B	Avg. drop
FP32	72.21	79.85	81.85	81.35	83.20	83.60	-
INT8+FP32 softmax	71.61	79.17	81.20	80.51	82.71	82.97	0.65 (cmp. to FP32)
This work	69.99	78.06	80.11	79.66	82.36	82.34	0.94 (cmp. to INT8)

C. Speedup

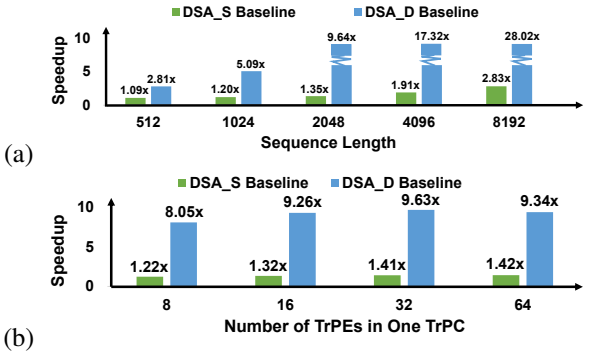

Fig. 5: (a) Speedup on Transformer layer with 768 embedding dimension for our design at different sequence lengths. (b) Scaling up the system with more TrPEs.

Authorized licensed use limited to: Tsinghua University. Downloaded on April 23,2025 at 03:04:16 UTC from IEEE Xplore. Restrictions apply.

Fig. 5 (a) illustrates the speedup of our design over the baseline architectures across sequence lengths ranging from 512 to 8192, while Fig. 5 (b) examines the number of TrPEs within a TrPC that exploit sequence parallelism with a total sequence length of 2048 and a fixed total area. For evaluation, we use a ViT [5] layer with an embedding dimension of 768. The speedup over the DSA_S baseline is attributed to our design's high compute density at the micro-architecture level, as DSA_S relies on a larger shared memory within the core cluster to hold all intermediate attention data. In contrast, the speedup over DSA_D is achieved by our design's elimination of DRAM access for intermediate data during attention execution.

Fig. 5 (a) and (b) demonstrate the weak and strong scaling capabilities of our design, respectively, due to sequence parallelism with reduced attention buffering and memory access. These findings highlight that both spilling data to off-chip DRAM and inflating on-chip buffer size degrade computational capability, whereas our design offers a promising solution for low-latency Transformer inference in edge-server applications.

D. Energy efficiency

Fig. 6 compares the energy consumption between the baseline DSAs and our design. Comparison to the DSA_S baseline shows the benefit from layer-fusion execution by reducing local high-bit-width intermediate data access in standard softmax, while the significant energy reduction to the DSA_D baseline is mainly from the reduction of DRAM access. The energy-efficiency gain also scales with sequence length, achieving $1.31\times$ and $3.87\times$ to the DSA_S and DSA_D baselines, respectively, with a sequence length of 4096.

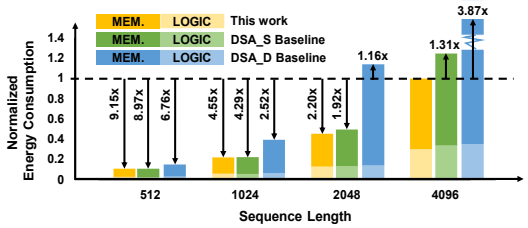

Fig. 6: Energy consumption of a Transformer layer over two baselines as sequence length grows.

VII. Conclusion

This work demonstrates an efficient Transformer inference architecture on the edge exploiting flexible sequence parallelism for latency-sensitive applications. It enables low-bit-width layer-fusion execution of self-attention by eliminating the max-finding step in softmax with saturation-linear approximation. Customized dense-sparse-hybrid datapath at core-level together with system-level architecture with sequence-length/parallelism insensitive buffering demand guarantee strong scaling of the compute system. For comparisons with the state-of-the-art DSA design, this work achieves up to $2.83\times$ and $28.02\times$ speedup with $1.31\times$ and $3.87\times$ energy efficiency, respectively, under two baseline settings.

Acknowledgment

This work was supported in part by NSFC Grant 62374101. The corresponding author is Hongyang Jia.

References

[1] Z. Li *et al.*, "Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," in *European conference on computer vision*. Springer, 2022, pp. 1–18.
[2] T. Liang *et al.*, "Bevfusion: A simple and robust lidar-camera fusion framework," *Advances in Neural Information Processing Systems*, vol. 35, pp. 10 421–10 434, 2022.
[3] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[4] H. Touvron *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.
[5] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
[6] Y. Rao *et al.*, "Dynamicvit: Efficient vision transformers with dynamic token sparsification," *Advances in neural information processing systems*, vol. 34, pp. 13 937–13 949, 2021.
[7] H. Wang *et al.*, "Spatten: Efficient sparse attention architecture with cascade token and head pruning," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021, pp. 97–110.
[8] Z. Qu *et al.*, "Dota: detect and omit weak attentions for scalable transformer acceleration," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 14–26.
[9] B. Keller *et al.*, "A 95.6-tops/w deep learning inference accelerator with per-vector scaled 4-bit quantization in 5 nm," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 4, pp. 1129–1141, 2023.
[10] M. Wang *et al.*, "A high-speed and low-complexity architecture for softmax function in deep learning," in *2018 IEEE asia pacific conference on circuits and systems (APCCAS)*. IEEE, 2018, pp. 223–226.
[11] Y. Gao *et al.*, "Design and implementation of an approximate softmax layer for deep neural networks," in *2020 IEEE international symposium on circuits and systems (ISCAS)*. IEEE, 2020, pp. 1–5.
[12] J. R. Stevens *et al.*, "Softermax: Hardware/software co-design of an efficient softmax for transformers," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 469–474.
[13] W. Wang *et al.*, "Sole: Hardware-software co-design of softmax and layernorm for efficient transformer inference," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.
[14] T. Dao *et al.*, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 344–16 359, 2022.
[15] K. Hong *et al.*, "Flashdecoding++: Faster large language model inference on gpus," *arXiv preprint arXiv:2311.01282*, 2023.
[16] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.
[17] A. Wang *et al.*, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.
[18] H. Touvron *et al.*, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.
[19] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.
[20] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
[21] S. Kim *et al.*, "I-bert: Integer-only bert quantization," in *International conference on machine learning*. PMLR, 2021, pp. 5506–5518.
[22] Y. Lin *et al.*, "Fq-vit: Post-training quantization for fully quantized vision transformer," *arXiv preprint arXiv:2111.13824*, 2021.

Authorized licensed use limited to: Tsinghua University. Downloaded on April 23,2025 at 03:04:16 UTC from IEEE Xplore. Restrictions apply.