Enabling Energy-Efficient Inference for Self-Attention Mechanisms in Neural Networks

Qinyu Chen[1], *Member, IEEE*, Congyi Sun[2], Zhonghai Lu[3], *Senior Member, IEEE*, Chang Gao*[,4], *Member, IEEE*
[1] Institute of Photonic Chips, University of Shanghai for Science and Technology, Shanghai, China
[2] School of Electronic Science and Engineering, Nanjing University, Nanjing, China
[3] KTH-Royal Institute of Technology, Stockholm, Sweden
[4]Institute of Neuroinformatics, University of Zürich and ETH Zürich, Zurich, Switzerland

***Abstract*—The study of specialized accelerators tailored for neural networks is becoming a promising topic in recent years. Such existing neural network accelerators are usually designed for convolutional neural networks (CNNs) or recurrent neural networks have been (RNNs), however, less attention has been paid to the attention mechanisms, which is an emerging neural network primitive with the ability to identify the relations within input entities. The self-attention-oriented models such as Transformer have achieved great performance on natural language processing, computer vision and machine translation. However, the self-attention mechanism has intrinsically expensive computational workloads, which increase quadratically with the number of input entities. Therefore, in this work, we propose an software-hardware co-design solution for energy-efficient self-attention inference. A prediction-based approximate self-attention mechanism is introduced to substantially reduce the runtime as well as power consumption, and then a specialized hardware architecture is designed to further increase the speedup. The design is implemented on a Xilinx XC7Z035 FPGA, and the results show that the energy efficiency is improved by 5.7x with less than 1% accuracy loss.**

***Index Terms*—Workload balance, SNNs, VLSI.**

I. Introduction

OVER the past decade, Deep Neural Networks (DNNs) have achieved significant performance across various domains. However, DNNs are computational-intensive and usually have tremendous parameters, prohibitively expensive when deployed on mobile devices. In order to reduce the energy cost and improve the throughput, many dedicated hardware accelerators were designed [1]–[3]. However, most of them are focused on popular neural networks such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), which are not fully compatible with emerging neural network primitives such as attention mechanisms.

The attention mechanism is emerging as one of the most influential neural network primitives. It allows neural networks to determine what is relevant to the current input through similarity computation, and decide where to attend. As one specific case of attention mechanisms, the self-attention mechanism is used to obtain the relations among the inputs. Since Transformer architecture [4] integrated with the self-attention mechanism achieved notable improvements for sequence to sequence tasks, self-attention-oriented NLP models (e.g., BERT [5], RoBERTa [6]) have presented record-breaking results for various NLP tasks, including question answering and language inference. The success of the self-attention mechanism in the NLP domain have also led researchers to investigate its adaption to computer vision [7], [8].

The power of the self-attention mechanism comes with costly computations. The dense matrix-matrix multiplications are required to compute the similarity across all search targets, and thus the amount of computations quadratically increases with the number of the search targets. In many self-attention-oriented models, the self-attention mechanism accounts for a significant portion of time and power consumption [9], which becomes a limiting factor for hardware deployment, especially on resource-hungry devices. To address this problem, some dedicated accelerators related to the attention mechanism are recently proposed. FTRANS [10] can be potentially utilized to accelerate self-attention mechanisms, however, it is designed for end-to-end Transformer, which lacks targeted optimization for self-attention mechanisms. A^3 [11] and ELSA [9] are dedicated accelerators for attention mechanisms that apply approximation scheme, which can avoid an exhaustive search thus reduce the computations. However, A^3 requires an expensive preprocessing step (i.e., sorting all columns of the key matrix), and ELSA introduces a complex computation pipeline that needs some extra operations (e.g., hash computation and hamming distance computation).

In this work, we propose a power-efficient architecture for the self-attention mechanism with a low-overhead approximation method. The main contributions are as follows:

- Based on the observation that irrelevant relations can be removed, a prediction-based approximation method with low overhead for computing similarity is proposed, substantially reducing the computational cost.
- To benefit from the proposed approximation method, a dedicated accelerator is designed.
- The accelerator is implemented on a Xilinx XC7Z035 FPGA, and the results show that the energy efficiency is improved by 5.7x with less than 1% accuracy loss.

II. Background and Motivation

As shown in Fig. 1, the required computations of the self-attention mechanism primarily include matrix-matrix multiplications and softmax normalization. For n input entities, three different dense matrix with $n \times d$ dimensions are provided: the query matrix (Q), the key matrix (K) and the value matrix (V). First, the mechanism computes the dot product similarity. By multiplying the query matrix with the transposed key matrix, a score matrix S with $n \times n$ dimensions is obtained, where s^{ij} denotes the similarity between the ith query and jth key. Then, softmax normalization is applied on each row of the

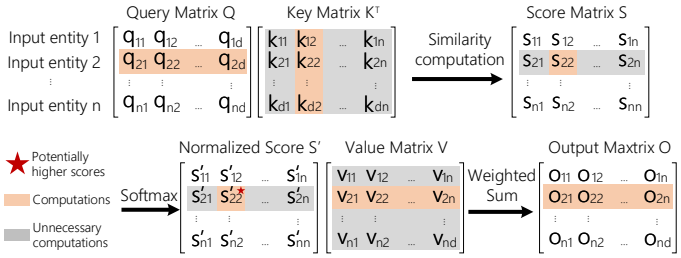

Fig. 1. The computations of self-attention mechanism, taking input entity 2 as an example.

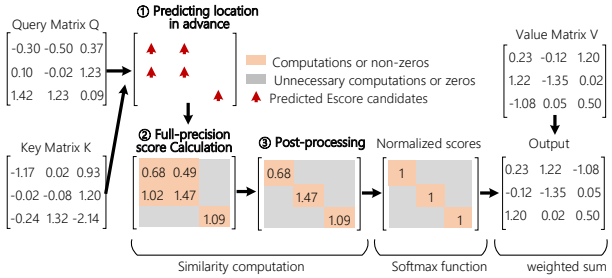

Fig. 2. Illustration of the proposed prediction-based approximate self-attention mechanism.

score matrix. Finally, the normalized score matrix is used as weights to retrieve the weighted sum from the value matrix. The result is an output matrix with $n \times d$ dimensions, where ith row represents the outcome of the ith input entity.

Since all the input matrices are dense, typically the self-attention mechanism is implemented by a dense matrix-matrix multiplication (n^2d multiply-accumulates (MACs)) followed by a softmax function again followed by another dense matrix-matrix multiplication (n^2d MACs). This implementation obtains the result correctly, but too costly. Actually, the nature of the self-attention mechanism is essentially a content-based approximate search, which indicates that not all the elements of these matrices contributes equally to the output. Most scores after softmax function will become near-zero. Most of the computations that are related to the near-zero values have very little impact on the final output. Those scores potentially with larger values are regarded as effectual scores (EScores). Our intuition is to avoid such unnecessary computations using approximate computing. If we can predict those potential larger scores approximately in advance, it is possible to skip the unnecessary calculations and greatly improve the performance. The exact dot product similarity is only computed when it is predicted to be an EScore.

III. Architecture Design and Data Processing

A. Prediction-based Approximate Self-Attention

The self-attention mechanism is composed of similarity computation, softmax normalization and weighted sum. Similarity computation as an important component of the self-attention mechanism is to compute scores by calculating the inner product between a query vector and a key vector. The main idea behind our proposed approximation scheme is to **search for EScores** in the similarity computation module, and then execute the rest two modules only using the selected EScores (Fig. 2). The procedure of searching for EScores mainly have three steps: ❶ **predicting** the location of EScore candidates in advance, ❷ **calculating** the full-precision inner product results of those EScore candidates, ❸ **post-processing** the selected EScore candidates to obtain the EScores.

❶ During prediction, we are not interested in their specific numerical number of these scores, and only care about the relative magnitudes among these scores. Therefore, there is no need to conduct the full-precision computation when predicting the relationship between those scores. Since the data value is mainly determined by the higher-order bits,

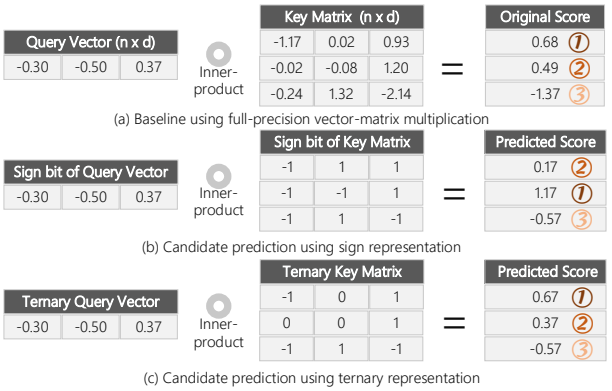

Fig. 3. Demonstration of predicting location of EScore candidates in the similarity computation module, using a 1×3 query vector and a 3×3 key matrix, (a) baseline using vector multiplication, (b) prediction using sign representation, (c) prediction using ternary representation.

the lower-order bits can be neglected during the prediction stage. The extreme case is to utilize the sign bits (i.e., -1 or +1) for inner product calculation. Fig. 3(a) describes an example that a query vector is multiplied with a key matrix, the original scores are obtained as the baseline. It shows that the first key vector in the matrix has the largest similarity (0.68), indicating that the first key vector is likely to be the candidate. Fig. 3(b) shows the candidate search prediction process when sign-bit representation is used. It is observed that the sign-bit representation will cause some obvious errors in some cases. Some near-zero (e.g., 0.02 and -0.02) items which have little impact on the inner-product result will be represented as +1 or -1, leading to a bigger gap with the correct number. Instead of using sign representation, in this work, we quantize keys to ternary representation (i.e., -1, 0, +1). The ternary representation primarily has two advantages. On one hand, the near-zero items can be represented as zeros, canceling errors brought by the enlarged gap. On the other hand, the zeros induce sparsity in the inputs, thus reducing the overheads of the prediction stage in terms of latency and energy consumption. As demonstrated in Fig. 3(c), EScore candidates prediction under the ternary representation has the same result as that under full-precision. Top-k positive scores will be predicted as the EScore candidates according to the prediction results, where hyperparameter k can be configured by the users. ❷ Once Top-k scores are predicted to be the EScore candidates, the full-precision inner product will be

executed between the query vector and those corresponding key vector candidates to obtain the full scores. ❸ For further reducing the cost, these key vector candidates should be re-selected based on their full scores. Previous work [11] adopts a dynamic post-scoring method to filter out the scores for next steps. They compare a full score for a particular row with the top-scoring rows score, and if the difference is larger than a pre-defined threshold, such score is blocked from the next steps. However, this dynamic selection will cause the different number of EScores in each row of the weight matrix, which will bring workload unbalance for later weight-value matrix-matrix multiplication. In this case, considering hardware workload balance and simplicity, we directly select Top-q scores as EScores according to their full scores, where hyperparameter q can be configured by the users.

Computational cost analysis. Given a query matrix, a key matrix and a value matrix (with n vectors where each vector has d dimensions), the computation of the baseline self-attention mechanism mainly includes: 1) n^2d full-precision MACs for query-matrix-key-matrix multiplication, and 2) n^2d full-precision MACs for score-matrix-value-matrix multiplication. After applying the proposed approximation method, the computation of the approximate self-attention mechanism contains: 1) sn^2d additions for prediction, where s is the sparsity ratio induced by ternary quantization, 2) nkd full-precision MACs for calculating full-scores of candidate Escores, where k is smaller than n, 3) nqd full-precision MACs for calculating weighted sums, where q is further smaller than n.

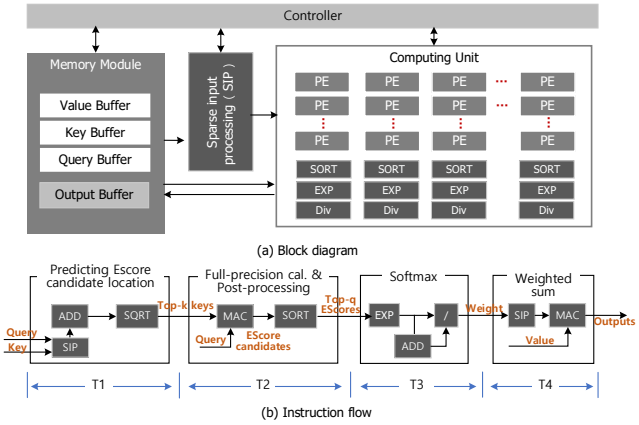

Fig. 4. The proposed accelerator: (a) block diagram, (b) instruction flow.

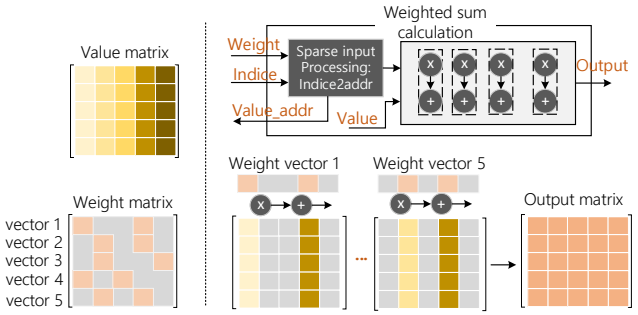

Fig. 5. The architecture of the weighted sum computation module.

B. Hardware accelerator with Approximation Support

For benefiting from the approximation, a specialized hardware accelerator should be co-designed with this approximation method. The accelerator as a specialized functional unit for the proposed efficient self-attention mechanism can be integrated with other accelerators or processors. Once the accelerator receives the instruction from the host, and input data are ready, the accelerator will be initialized to execute the instruction, and then the outputs will be written to the memory. As shown in Fig. 4 (a), the proposed accelerator is mainly composed of on-chip memory, a controller, and a computing unit. The memory prepares the input matrix consumed by computing resources and stores the generated outputs. The computing unit contains a processing element (PE) array, Exp tables, and sorting modules. The self-attention mechanism is applied on the accelerator with four instructions, including: ❶ EScore candidate location prediction, ❷ full-precision calculation and post-processing, ❸ softmax normalization, and ❹ weighted sum calculation. The controller manages the overall execution, updates the accelerator state, and decodes the instructions from the host.

Stage ❶: EScore candidate location prediction module is designed to identify the set of potentially relevant key vectors, and then transfer the indices of these key vectors to the next stage. First, the keys are quantized to ternary representation using comparators. Those large positive keys will be quantized to 1, large negative keys will be quantized to -1, and those keys near zero will be rounded to 0 by comparing with a pre-defined threshold. Those queries corresponding with non-zero keys will be fetched from memory and accumulated. The accumulation results from a query vector and a key matrix will be sent to the sorting unit to select Top-k values.

Stage ❷: Full-precision calculation module is composed of adders and multipliers. It is designed to compute the full-precision inner-product of the selected Top-k scoring key vectors and the query vector. The full-precision scores will be sent to the sort module for post-processing. Every subset of k full-precision scores will be sorted, and Top-q scores will be selected as EScores.

Stage ❸: Softmax normalization module is designed to normalize the scores by applying the softmax function. It contains a lookup table to perform the exponent computation. Once the exponent of the score is obtained, it will be accumulated to get the softmax denominator. The divider will perform the division to normalize each score. The obtained normalized score matrix is used as a weight matrix for the weighted sum computation.

Stage ❹: Weighted sum computation module takes the weight matrix and value matrix as inputs to compute the final outputs. It's worth noting that the weight matrix is sparse due to the previous EScore search. For improving energy efficiency, a sparse matrix-matrix multiplication design should be explored. As shown in Fig. 5, a sparse input processing unit is proposed to convert the indices of EScores to the address of values which correspond with the EScores. Each MAC is responsible for processing a vector-matrix multiplication.

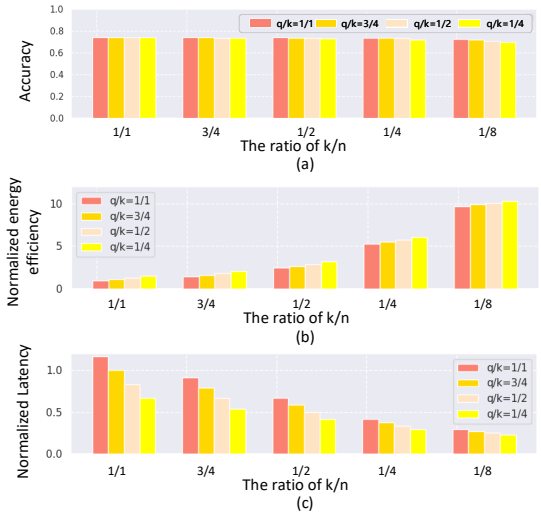

Fig. 6. Accuracy with respect to the ratio of q/k and k/n.

Since the number of EScores is the same in each weight vector, the workload is balanced.

IV. Experimental Results

In this work, we evaluate our approximation method with Google BERT model [5], which utilizes the self-attention mechanism. We use an open-source Tensorflow implementation for Bert (mini) [12] as the baseline, and study the task of the Natural Language Inference Corpus (MNLI) [13]. To estimate the impact on the accuracy of our approximation method, we replace the original self-attention mechanism with our approximate self-attention module.

We first analyze the impact of the approximation method on accuracy by changing the ratio of k/n and the ratio of q/k (Fig. 6 (a)). It shows that the larger the ratio of k/n and the ratio of q/k are, the higher the accuracy is. Since the most important feature of this design is its ability to accelerate the self-attention mechanism efficiently, we also compare the energy efficiency of performing a single self-attention operation across the accelerator with different configurations by changing the ratio of k/n and q/k (Fig. 6 (b)). The most aggressive configuration ($k/n = 1/8, q/k = 1/4$) in our evaluation shows a 4% accuracy loss but with a large energy efficiency improvement (10.3x). A sweet point is the configuration ($k/n = 1/4, q/k = 1/2$) with 0.6% accuracy loss with 5.7x energy efficiency improvement. This configuration also leads to a 3x reduction in latency (Fig. 6 (c)). The proposed accelerator is synthesized and implemented on XC7Z035 FPGA. The simulation results show that the accelerator can run up to 200MHz, consuming 3.3 W. It takes 21.12 us to finish the calculation of approximated self-attention mechanism. The resource utilization is summarized in Table I.

TABLE I
XC7Z035 FPGA resource utilization of Skydiver

Metrics	LUT	FF	DSP	BRAM
Avaliable	171900	343800	900	500
Used	123624	118704	0	140
Percentage	71.9%	34.5%	0%	28.0%

V. Conclusion

The self-attention oriented neural networks have achieved great performance for various tasks such as natural language processing. However, current hardware accelerators are often dedicated to CNNs or RNNs. The self-attention mechanism as an emerging neural network primitive has heavy computational workloads, which also has demand of being accelerated. In this work, we propose a software-hardware co-design solution for energy-efficient self-attention inference. A prediction-based approximate self-attention mechanism is introduced to substantially reduce the runtime as well as power consumption, and then a specialized hardware architecture is designed to further reduce the latency. LISA is implemented on a Xilinx X-C7Z035 FPGA, and the results show that the energy-efficiency is improved by 5.7x with less than 1% accuracy loss.

References

[1] C. Gao, D. Neil, E. Ceolini, S. C. Liu, and T. Delbruck, "Deltarnn: A power-efficient recurrent neural network accelerator," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, 2018, pp. 21–30.

[2] S. Yan, Z. Liu, Y. Wang, C. Zeng, Q. Liu, B. Cheng, and R. C. Cheung, "An fpga-based mobilenet accelerator considering network structure characteristics," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2021, pp. 17–23.

[3] D. Kadetotad, S. Yin, V. Berisha, C. Chakrabarti, and J.-s. Seo, "An 8.93 tops/w lstm recurrent neural network accelerator featuring hierarchical coarse-grain sparsity for on-device speech recognition," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 55, no. 7, pp. 1877–1887, 2020.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems (NIPS)*, 2017, pp. 5998–6008.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[6] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[8] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 213–229.

[9] T. J. Ham, Y. Lee, S. H. Seo, S. Kim, H. Choi, S. J. Jung, and J. W. Lee, "Elsa: Hardware-software co-design for efficient, lightweight self-attention mechanism in neural networks," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 692–705.

[10] B. Li, S. Pandey, H. Fang, Y. Lyv, J. Li, J. Chen, M. Xie, L. Wan, H. Liu, and C. Ding, "Ftrans: energy-efficient acceleration of transformers using fpga," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2020, pp. 175–180.

[11] T. J. Ham, S. J. Jung, S. Kim, Y. H. Oh, Y. Park, Y. Song, J.-H. Park, S. Lee, K. Park, J. W. Lee, and D.-K. Jeong, "A3: Accelerating attention mechanisms in neural networks with approximation," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020, pp. 328–341.

[12] R. Google, "Tensorflow code and pre-trained models for bert," 2018. [Online]. Available: https://github.com/google-research/bert

[13] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint arXiv:1704.05426*, 2017.