DTQAtten: Leveraging Dynamic Token-based Quantization for Efficient Attention Architecture

Tao Yang[1], Dongyue Li[1], Zhuoran Song[1], Yilong Zhao[1], Fangxin Liu[1], Zongwu Wang[1], Zhezhi He[1] and Li Jiang[1,2,3]
[1]Shanghai Jiao Tong University, Shanghai, China, [2]Shanghai Qi Zhi Institute, Shanghai, China
[3]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University

Abstract*—Models based on the attention mechanism, i.e. transformers, have shown extraordinary performance in Natural Language Processing (NLP) tasks. However, their memory footprint, inference latency, and power consumption are still prohibitive for efficient inference at edge devices, even at data centers. To tackle this issue, we present an algorithm-architecture co-design with dynamic and mixed-precision quantization, DTQAtten. We present empirically that the tolerance to the noise varies from token to token in attention-based NLP models. This finding leads us to quantize different tokens with mixed levels of bits. Thus, we design a compression framework that (i) dynamically quantizes tokens while they are forwarded in the models and (ii) jointly determines the ratio of each precision. Moreover, due to the dynamic mixed-precision tokens caused by our framework, previous matrix-multiplication accelerators (e.g. systolic array) cannot effectively exploit the benefit of the compressed attention computation. We thus design our accelerator with the variable-speed systolic array (VSSA) and propose an effective optimization strategy to alleviate the pipeline-stall problem in VSSA without hardware overhead. We conduct experiments with existing attention-based NLP models, including BERT and GPT-2 on various language tasks. Our results show that DTQAtten outperforms the previous neural network accelerator Eyeriss by 13.12× in terms of speedup and 3.8× in terms of energy-saving. Compared with the state-of-the-art attention accelerator SpAtten, our DTQAtten achieves at least 2.65× speedup and 3.38× energy efficiency improvement.

***Index Terms*—transformers; domain-specific accelerator; dynamic quantization; algorithm-architecture co-design**

I. Introduction

Natural Language Processing (NLP) has witnessed rapid progress in recent years driven by the attention mechanism. Attention-based models such as Transformers [1], BERT [2], and GPT-2 [3] provide significant performance improvements over convolutional neural networks and recurrent neural networks. Nevertheless, the high accuracy is at the cost of the increasing demand for computation power.

Quantization methods have been effective techniques for reducing the inference workload in deep neural networks [4], [5]. Nowadays, several works have introduced quantization in attention-based NLP models. Q-BERT [6] proposes to use group-wise Hessian information to quantize BERT models to low precision. I-BERT [7] quantizes all the embeddings and executes matrix multiplication (MM) with INT8 multiplication and INT32 accumulation, which is easier for hardware implementation. SpAtten [8] employs both the pruning and quantization methods on BERT [2] and GPT-2 [3] models. The authors introduce applying both token-wise pruning and layer-wise quantization jointly in a model. However, there are two drawbacks in current quantization methods for attention-based NLP models: (i) the quantization granularities are coarse-grained to achieve sufficient computational cost reduction; (ii) quantization and pruning are generally performed as two independent steps, which makes it difficult to search the global optimal solution for model compression with an acceptable accuracy [4].

To address these issues, we first show empirically that tokens in attention-based NLP models show different tolerance to noise. Improper quantization on these tokens might seriously deteriorate the overall performance. Specifically, to obtain a high compression ratio with preserved accuracy, we can relax the quantization strength for more robust tokens, while quantization for less tolerant tokens should be restricted. Based on the finding, we propose a fine-grained token-based mixed-precision quantization framework for attention-based NLP models. Our quantization method dynamically tracks the token tolerance and adjusts the precision of token feature vectors, i.e. Query (Q), Key (K), Value (V) vectors in each block of attention-based NLP models. Moreover, we treat pruning as a corner case of quantization where the tokens are quantized to 0-bit. Thus, quantization and pruning are integrated as a unified optimization problem, and we jointly search for the global solution of model compression.

The mixed-precision tokens resulting from our quantization framework challenge the efficiency of existing hardware. It is hard for existing architectures to capture the dynamically distributed low-precision tokens and effectively convert these low-precision calculations into performance improvement. We develop a hardware architecture based on variable-speed systolic array (VSSA) [5] for our quantization method and propose a dedicated optimization strategy to solve the low PE efficiency problem in VSSA by reordering and clustering the tokens with the same precision. The optimization strategy ensures that most of the calculation iterations in VSSA only involve a pair of precisions. Thus, no pipeline stall occurs in these iterations, which ensures the high efficiency of computing resources.

This work was partially supported by the National Key Research and Development Program of China (2018YFB1403400), National Natural Science Foundation of China(Grant No. 61834006). The author Tao Yang is supported by Wu Wen Jun Honorary Doctoral Scholarship, AI Institute, Shanghai Jiao Tong University. Corresponding author: Li Jiang.

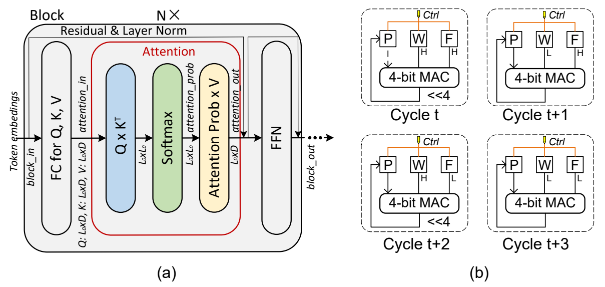


Fig. 1. (a) An attention-based block in NLP models. (b) PE of 4-bit MAC unit can perform 8bit-8bit MAC mode taking four cycles.

In summary, we present an algorithm-architecture co-design for attention-based NLP models, namely DTQAtten. Our contributions are threefold.

- We propose a token-based mixed-quantization framework for attention-based NLP models. This framework dynamically decides quantization precision levels for tokens and jointly search for the ratio of each precision.
- We design a hardware architecture to support the inference of our quantization framework for attention-based NLP models. We propose a lightweight optimization strategy to reduce pipeline stalls caused by the dynamically distributed mixed-precision tokens.
- We validate the effectiveness of our design through extensive experiments on NLP tasks. Results show that DTQAtten achieves 7.94×, 4.41×, 2.65× speedup improvement upon the state-of-the-art attention-based accelerators MNNFast [9], A^3 [10], and SpAtten [8]. Our design also outperforms these designs in terms of area cost and energy efficiency.

II. BACKGROUND

A. Attention-Based NLP Models

The recent attention-based models [11], pre-trained from large unlabeled data (e.g., BERT [2] and the GPT family [1]) have achieved a significant accuracy improvement on a wide range of NLP tasks. Although the connectivity among layers and the organizations of these networks are different, they are all based on the same basic attention-based block. Fig. 1(a) shows the architecture of the block. The inputs of the block are the *embeddings* of the tokens. In the block, *embeddings* are linearly transformed to Query (Q), Key (K) and Value (V) vectors. Then, Q, K, V vectors are processed by the attention layer. In attention layer, the attention probabilities (*attention_prob*) are produced by employing softmax on $Q \times K^T$. The attention output (*attention_out*) is obtained by multiplying the *attention_prob* with V. A residual layer adds the *attention_out* with *embeddings* and execute normalization. Furthermore, a Feed-Forward Network (FFN) containing three Fully-Connected (FC) layers is applied. Finally, another residual operation is conducted and outputs *block_out*.

B. Neural Networks Pruning and Quantization

Pruning and quantization are previously introduced in compressing convolutional neural networks. Due to the prevailing redundancy in model weights, removing unimportant weights [4] or representing activations and weights with low bit precision [5] can reduce computing overhead while maintaining high accuracy performance. In attention-based NLP models, pruning and quantization are executed on activation vectors (including Q, K, and V vectors), for they take a large amount of the memory and consume most of the computational overhead. For pruning on attention-based NLP models, MnnFast [9] prunes Value (V) vector locally in attention computation. A^3 [10] first sorts each dimension of the key vectors among all keys and uses a predefined ratio of elements in the keys to multiply with queries to get partial attention scores. For quantization on attention-based NLP models, Fully 8-bit [12] and I-BERT [7] propose 8-bit quantization schemes for BERT. Q-BERT [6] quantizes activation vectors with a group granularity using second-order Hessian information. Only a few works concerning both pruning and quantization to compress attention-based NLP models, SpAtten [8] first prunes the unimportant tokens in the sentence and then implements a layer-wise quantization on the model. However, the quantization for attention-based NLP models in these works are still coarse-grained [6]–[8], [12] and conducted separately from pruning [8].

C. Variable-speed Systolic Arrays

Systolic array is nowadays a popular architecture used in various AI accelerators (e.g. Google TPU [13] and Gemmini [14]) to accelerate MM. It features by friendliness for very large scale integration (VLSI) with high speed and low cost. However, traditional systolic array can only work with “one” precision. DRQ [5] proposes variable-speed systolic array (VSSA) to support multi-precision (4bit and 8bit) convolution. 4-bit MAC unit is used as basic PE in VSSA. As shown in Fig. 1(b), each PE has two registers W and F holding a weight value and a feature value, respectively. A register P stores the partial result. Each PE can perform 8bit-8bit MAC mode by taking four cycles in a timing-multiplexing manner. In each cycle, the PE extracts the higher 4 bits (H) or lower 4 bit (L) of the weight value and the feature value from W and F, then choose whether to shift the MAC result accordingly and store the result into the P register. For the strict dataflow requirement in systolic array architecture, the pipeline stall occurs when the adjacent PEs execute MACs with different precisions. The stall cases will be discussed in detail in Section IV-B. The high stall ratio incurs the low computational efficiency of VSSA.

III. DYNAMIC TOKEN-BASED QUANTIZATION

A. Different Tolerance to Noise of Tokens

In this section, we show empirically that the tokens with different importance scores show different tolerance to noise. We take a widely used task (qnli) from the GLUE [1] benchmark on BERT-Base [2] as a case study. In attention layer, relevances between pairs of tokens are measured by $Q \times K^T$ and then normalized by the *softmax* function [11]. Naturally, the more the token is relevant to others, the more important it is. Thus we can calculate an importance score for each token by accumulating the soft-maxed probabilities across all tokens as shown in Fig. 2(a).

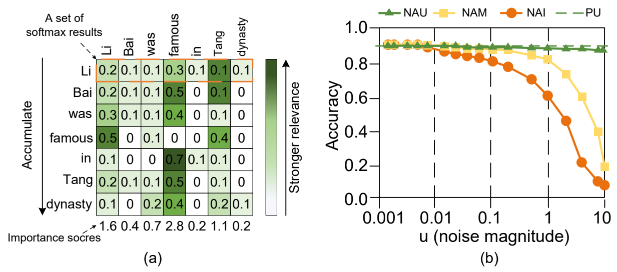


Fig. 2. (a) Attention probabilities are summed over each column to get importance scores. (b) BERT-Base accuracy on task qnli [1] with noise added on tokens with different importance scores.

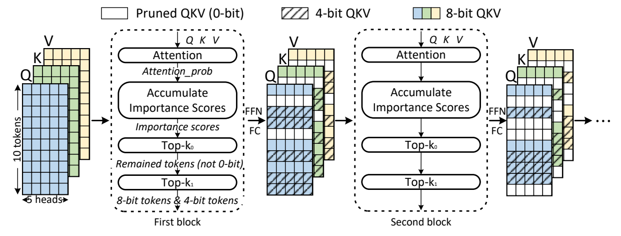


Fig. 3. Dynamic token-based quantization: tokens are quantized to 8-bit, 4-bit and 0-bit in the inference according to their important scores.

With the importance scores, we sort the tokens by their importance scores and classify the tokens of each layer into three segments using the threshold ratio of 15% and 85%. In the three segments, segment 0 contains the most important 15% tokens. Segment 1 contains the middle 70% and segment 2 contains the most unimportant 15% tokens. We then add Gaussian noise to the tokens in different segments and measure the accuracy of BERT with the noise magnitude u. The results are shown in Fig. 2 (b), where "NAI", "NAM", "NAU" mean only add noise to segments 0, 1 and 2, respectively. We can observe that "NAI" curve decreases rapidly with the growth of u, while the accuracy of "NAU" remains stable while increasing u. The results indicate that (i) tokens in different segments show varied tolerance to noise, and (ii) token's tolerance to noise is inversely proportional to its importance degree. Furthermore, in order to explore the noise tolerance boundary of the unimportant tokens, we add an experiment to prune the tokens in segment 2 directly shown as "PU" in Fig. 2 (b). The accuracy of "PU" is 90.1% which is comparable to the accuracy of baseline 90.3%. The results further indicate that (iii) pruning unimportant tokens has a negligible effect on the model accuracy. Given the above findings, achieving a high compression ratio with maintained accuracy requires restricting the quantization impact (equivalent to noise) for the important tokens and relaxing the quantization for the unimportant tokens.

B. Dynamic Token-based Quantization Algorithm Overview

Based on the experimental findings, we propose our token-based mixed-quantization algorithm. We show the overall logic of our algorithm in Fig. 3. We use attention probabilities to calculate the importance score of each input token dynamically in each attention block. We then use a two-level top-K engine to split the tokens into three segmentation according to their importance scores. The first top-k engine selects the top k_0 tokens to remain in the next block and prune other tokens. Then, for the top k_0 tokens, we use the second top-k engine to further select the top k_1 tokens to be quantized with high-precision. The other $(k_0 - k_1)$ tokens are quantized with low-precision. In this paper, we define 8-bit as high-precision, which has been proved to be sufficient for typical attention-based NLP models [6], [8], [12], [15]. We define 4-bit as low-precision. Particularly, the pruned tokens in the first top-k engine are treated as 0-bit tokens, making our method a unified quantization algorithm.

C. Design Space Exploration

In our proposed quantization method, the tokens need to be divided into 0-bit, 4-bit and 8-bit parts according to their importance scores in each layer. The ratio of each part in each layer forms a large design space. Using conventional search methods such as grid search will cause unbearable time consumption. Thus, we apply a Bayesian optimization method [16] to execute the search process. The targeted optimization problem is constructed concerning both the accuracy and the computational performance, which is formalized as:

$$\textbf{minimize} \quad \mathcal{L}\left(R\right) = \mathcal{L}_{en} + \lambda * \mathcal{L}_{ops} \qquad (1)$$

where R is the hyper-parameter vector composed of the ratio factors in each layer, $\mathcal{L}_{en}$ is the cross-entropy loss, and $\mathcal{L}_{ops}$ is the penalty term for computational overhead. λ is a coefficient to balance the accuracy and performance. The $\mathcal{L}_{ops}$ we used in our paper can be formulated as:

$$\mathcal{L}_{ops} = \sum_i BOPs(r_{0b\_i}, r_{4b\_i}, r_{8b\_i}) / \sum_i FLOP(L_i) \qquad (2)$$

Where r_{0b_i}, r_{4b_i} and r_{8b_i} are the ratios of 0-bit part, 4-bit part and 8-bit part of layer L_i in quantized model. $BOPs$ can calculate the number of bit operations (BOPs) [17] according to these ratios. $FLOP(L_i)$ denotes the number of float point operations (OPs) of layer L_i in the original model. The final $\mathcal{L}_{ops}$ means the average bit operation number for each OP.

IV. Architecture Design

A. Architecture Overview

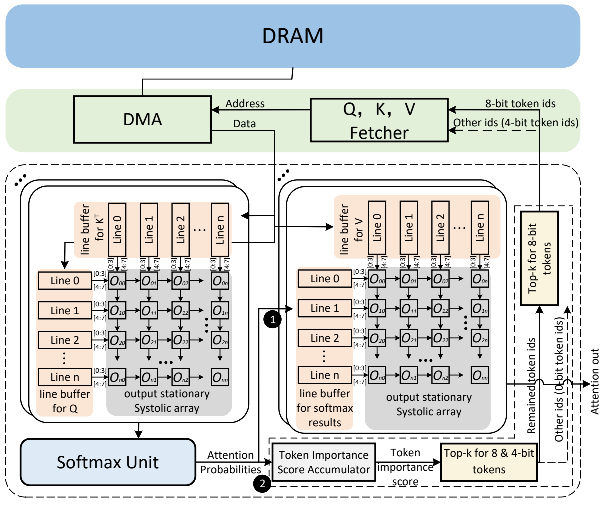
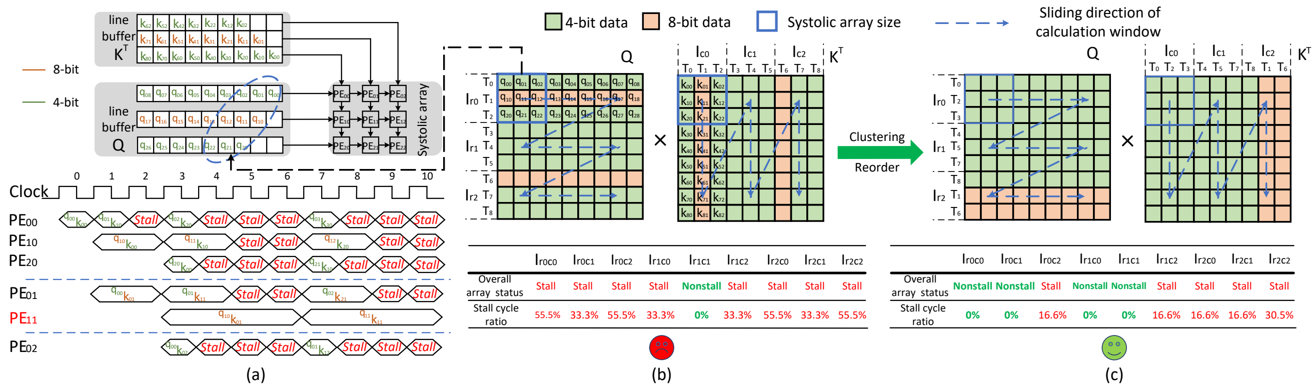


Fig. 4. Overview of the hardware architecture in DTQAtten.

We design an accelerator to support our dynamic token-based quantization. The overview of the architecture is shown



(b)

	I_{r0c0}	I_{r0c1}	I_{r0c2}	I_{r1c0}	I_{r1c1}	I_{r1c2}	I_{r2c0}	I_{r2c1}	I_{r2c2}
Overall array status	Stall	Stall	Stall	Stall	Nonstall	Stall	Stall	Stall	Stall
Stall cycle ratio	55.5%	33.3%	55.5%	33.3%	0%	33.3%	55.5%	33.3%	55.5%

(c)

	I_{r0c0}	I_{r0c1}	I_{r0c2}	I_{r1c0}	I_{r1c1}	I_{r1c2}	I_{r2c0}	I_{r2c1}	I_{r2c2}
Overall array status	Nonstall	Nonstall	Stall	Nonstall	Nonstall	Stall	Stall	Stall	Stall
Stall cycle ratio	0%	0%	16.6%	0%	0%	16.6%	16.6%	16.6%	30.5%

Fig. 5. (a) The dataflow and the executing timing of VSSA in iteration I_{r0c0}. (b) The executing window sliding in $Q \times K^T$ and the stalling cases when the input tokens are in normal order. (c) The executing window sliding in $Q \times K^T$ and the stalling cases after clustering and reordering the tokens.

in Fig. 4. Variable-speed systolic arrays (VSSA) are used to support mix-precision (4-bit and 8-bit) MM with an output-stationary style. In each layer, the DMA first fetch the Q, K, and V vectors of a valid token id and store them into the corresponding line buffers. Then, the two matrices Q and K^T are feed into the systolic PE array in a step-wise style as shown in Fig. 5(a) to meet the dataflow requirement in the systolic array. A softmax unit then processes the result of $Q \times K^T$ to get attention probabilities. After that, the attention probabilities are broadcasted to two modules. Module ❶ is the systolic array for $Attention\_prob \times V$, which can produce the final results of the attention. Module ❷ consists of a token importance score accumulator and two top-k engines. The token importance score accumulator accumulates the attention probabilities to achieve the importance score of each token. The two top-k engines with high computational parallelism split the input tokens into 8-bit, 4-bit and 0-bit segments according to the importance scores as described in Sec. III-B. The token ids of 4-bit and 8-bit produced by the two top-k engines are sent to the Q, K, V fetcher which can calculate the physical start address and the length of the Q, K, V vectors of corresponding tokens. With the address and data length, the DMA can then prefetch the Q, K, V vectors for the next layer.

The two top-K engines in ❷ are both of $O(n)$ time complexity [8]. When there are hundreds of input tokens, it will take a long period to sort these tokens. However, in our architecture, the executions of ❷ and ❶ are in parallel. Therefore, the executing time of ❷ can be hidden because of the large calculating quantity of the MM in ❶.

B. Low PE Efficiency Problem

As shown in Fig. 5 (b), to compute $Q \times K^T$ with 9 tokens and 9 channels using a 3×3 VSSA, we will temporally load part of the two matrices to the VSSA. Specifically, in each iteration, a tile of Q (include three rows) and a tile of K^T (include three columns) are loaded into VSSA to generate a 3×3 results. By Loading Q and K^T in the direction of the arrow, the whole process is divided into 9 iterations as shown in the table. The VSSA supports mixed-precision MM (4-bit and 8-bit) by flexibly controlling the calculation mode of each PE. However, the calculation of each PE should be synchronized with others for the strict dataflow requirement in systolic array, which causes pipeline stalls on some of the PEs. We take the first iteration I_{r0c0} shown in Fig. 5 (a) as an example to explain the seriousness of the pipeline stalls. It can be indicated from the figure that the overall throughput is restricted by the "critical path" on PE_{11} which is responsible for the inner product between the second row of I_{r0} and the second column of I_{c0}. Each pipeline step in PE_{11} takes 4 cycles to perform 8bit-8bit MAC mode. Other PEs performing 8bit-4bit MAC mode or 4bit-4bit MAC mode have to stall to synchronize with the "critical path". As our statistics table shown in Fig. 5 (b), eight of the nine iterations meet the stall problem. Here, we use the stall cycle ratio to express the under-utilization degree in PEs of each iteration:

$$R[iter_{id}] = \frac{\sum_{p=0}^{PE\_NUM} stall\_cycle[iter_{id}][p]}{PE\_NUM \times max\_cycle[iter_{id}]} \qquad (3)$$

where PE_NUM is the number of PEs in the systolic array, $stall\_cycle$ is the stall cycle number of each PE in each iteration, and max_cycle is the operating cycles of the PE with the "critical path". The table in Fig. 5 (b) shows the stall cycle ratios of all the 9 iterations. The average 39.5% stall cycle ratio indicates a low efficiency of the computing resources.

C. PE Efficiency Optimization Strategy

Facing the under-utilization of the PEs, we propose an efficient scheduling strategy to optimize hardware efficiency. We cluster the tokens of the same precision and move the tokens of low-precision to the front. Correspondingly, the Q and K of the tokens are clustered and reordered as shown in Fig. 5 (c). Then the systolic array executes the same nine iterations following the direction of the arrow to complete the entire calculation process. The PE utilization of this method in each iteration is shown in the table in Fig. 5 (c). From the results, we can figure out that our optimization strategy brings two benefits: (i) the number of stall iterations is decreased. The clustering method reduces the number of tiles that contain both 4-bit tokens and 8-bit tokens. Pipeline Stall will not occur if the loaded two tiles of Q and K^T both have single precision; (ii) the stall cycle ratio of each stall iteration is also decreased. Only one of Q tile or K^T tile contains both 4-bit token and 8-bit token in most stall iterations (I_{r0c2}, I_{r1c2}, I_{r2c0} and I_{r2c1}). In these iterations, the 4-bit $\times$ 8-bit operations have the longest pipeline step of 2 cycles, while the 4-bit $\times$ 4-bit operations have the shortest pipeline step of 1 cycle. Thus, only 1 extra cycle is wasted in each pipeline step on the stalled PE, contributing to the overall low stall cycle ratios.

The results of the attention layer with the reordered input tokens can be expressed as follows:

$$\begin{aligned} Attention\_out^{*} &= Softmax[(A \times Q) \times (K^{T} \times A^{T})] \times (A \times V) \\ &= A \times Softmax[Q \times K^{T} \times (A^{T} \times A)] \times V) \\ &= A \times [Softmax(Q \times K^{T}) \times V] \\ &= A \times Attention\_out \end{aligned} \tag{4}$$

where elementary matrix A and A^{T} responsible for row and column reordering are reciprocal pair. Taking A out of the $softmax$ function or moving the A into the $softmax$ function will not affect the results. We can draw that reordering the input tokens results in a reordered $Attention\_out$ without impacting the output values. The following Feed-Forward Network (FFN) will not change the order of $Attention\_out$. Thus, the input tokens to the next Attention-based block are also ordered by A.

Similarly, reordering the tokens impacts the row order and the column order in $Attention\_prob$ of each block as follows:

$$\begin{aligned} Attention\_prob^{*} &= Softmax[(A \times Q) \times (K^{T} \times A^{T})] \\ &= A \times Softmax(Q \times K^{T}) \times A^{T} \\ &= A \times Attention\_prob \times A^{T} \end{aligned} \tag{5}$$

as the token importance scores are achieved by accumulating the $Attention\_prob$ in each column, the left A has no impact on importance scores. The right A^{T} makes the order of importance scores matches that of the input tokens to the next attention-based block. Correspondingly, the token ids obtained by sorting the importance scores are also based on the order caused by A. Thus, we can sequentially store the block results of ordered input tokens on DRAM and then use the token ids directly in the next block to fetch tokens with no extra hardware overhead.

V. EXPERIMENTS

A. Experimental Setup

To valid our quantization algorithm, we conduct experiments on attention layers of BERT-Base [2], BERT-Large [2], GPT-2-Small [3] and GPT-2-Medium [3]. We fine-tune the BERT models on nine tasks from GLUE [1] and SQuAD V2.0 [1]. We fine-tune the GPT-2 models on language modeling task, including Wikitext-2/-103 [1], and 1BW [18].

For computational performance, we compare DTQAtten with two kinds of accelerators: **1) NN-based accelerators**, including Eyeriss-based [19] design (with 16-bit models) and Systolic-based design (with models quantized by our proposed method, but mapped on VSSA without our optimization strategy). In 40nm TSMC technology library, the area occupation of a 16-bit MAC unit is almost 16× larger than a 4-bit MAC unit. Therefore, with a similar area budget, Eyeriss-based design has a total of 224 16-bit MAC units, Systolic-based design and DTQAtten have 3168 ($\approx 16 \times 18 \times 11$) 4-bit MAC units. **2) Attention accelerators**, including A^{3} [10], MNNFast [9] and SpAtten [8]. For a fair comparison, we also restrict all four designs with a similar area budget of PEs. We use RTL-Verilog to implement our architecture and synthesize

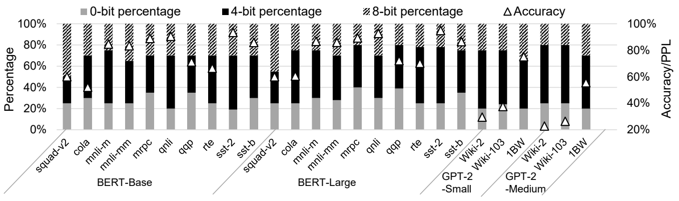


Fig. 6. Comparisons of the model accuracy and percentages of 8/4/0-bit tokens on different datasets.

RTL using Synopsys DC on 40nm TSMC library to estimate latency, area and power with target 1ns clock period (1GHz). Besides, we use CACTI [20] to estimate the energy and area of SRAMs and FIFOs.

B. Quantization Algorithm Performance

In Fig. 6, we show the percentages of 0-bit (pruning), 4-bit and 8-bit of the four models on each task and the corresponding accuracies. As our exploration method in Section III-C, we use a interval of 5% to sample from 0% to 100% as the search space for r_{0b_i}, r_{4b_i}, r_{8b_i} and search for 20 iterations using Bayesian optimization method [16] in each task. λ in Equation 1 is set as 0.37 (for BERT) and 0.24 (for GPT-2). We can observe from Fig. 6 that compressed BERT-Base and compressed BERT-Large show negligible accuracy degradation (within 1%) on the nine datasets compared with that of the original models, except 1.7% for squad-v2. Particularly, for some tasks, the compressed models show even better performance. Compressed BERT-Base achieves a 0.2% accuracy improvement on task mprc (87.9% vs. 88.1%), and compressed BERT-Large shows a 0.3% accuracy improvement on task mnli-mm (82.7% vs. 83.0%). The average percentages of the 0-bit, 4-bit and 8-bit parts in BERT are 28%, 41.4% and 30.6%, respectively. GPT-2 exhibits no accuracy loss on each dataset except a negligible 0.3% PPL enhancement for GPT-2-small on task 1BW (56.3% vs 56.0%). The average percentages of the 0-bit, 4-bit and 8-bit parts in these two GPT networks account for 21.8%, 52.0% and 26.2%, respectively.

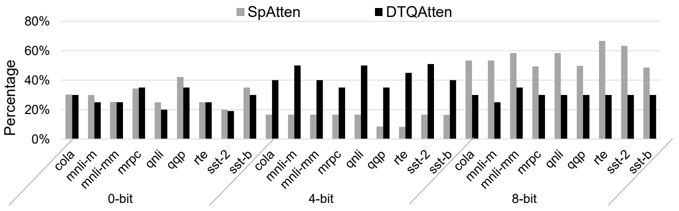


Fig. 7. Comparisons of 0-bit/4-bit/8-bit ratios between DTQAtten and SpAtten [8].

We further compare the performance of our quantization method with SpAtten [8], the state-of-the-art 2-step compression method for attention-based model. We reproduce the compression method of SpAtten, in which we prune the unimportant tokens and then implements a layer-wise quantization (4-bit/8-bit) according to each layer's attention probability distribution. For a fair comparison, we gradually increase the pruning ratio (0-bit) and then the percentage of 4-bit layers in SpAtten until the accuracy performance is equal to or lower than the corresponding ones of DTQAtten shown in Fig. 6. Finally, the comparison of the ratios of 0-bit, 4-bit and

8-bit using the two methods are shown in Fig. 7. It can be figured that although the average 0-bit ratio in SpAtten (29.8%) is a little bit higher than that in DTQAtten (26.3%), the average ratio of the 4-bit part in SpAtten (18.9%) is much lower than that in DTQAtten (40.5%). Thus, DTQAtten has a totally higher compression ratio. Two factors contribute to the better compression effect of DTQAtten: (i) we exploit a finer-grained quantization with token-wise granularity, which helps to remove the redundancy more accurately; (ii) compared to the 2-step method which greedily searches the optimal pruning ratio first and then searches the optimal low-bit ratio based on the first step, the 1-step DTQAtten only needs to search in a unique design space, thus has a larger opportunity to find the global optimal solution for model compression.

C. Computational Performance and Energy Consumption

Fig. 8 shows the average speedup and energy cost comparisons of DTQAtten with NN-based accelerator on attention layers. Where the implementations of Eyeriss-based design without model compression are normalized to 1. Systolic-based design achieves 8.17×, 9.16×, 5.67×, 6.06× speedup and reduce 56%, 68%, 49%, 53% energy cost on each model. The speedup and reduced energy cost mainly come from the low computational overhead of the models compressed by our proposed quantization algorithm. DTQAtten achieves 13.57×, 16.42×, 10.67× and 11.8× speedup and reduce 71%, 79%, 71%, 74% energy cost for each model. The low hardware efficiency problem in Systolic-based design is solved to a great extent in DTQAtten, which contributes to a better speedup ratio. Besides, Combining the advantages of (i) low-precision PEs; (ii) fewer data transferred between DRAM and on-chip buffer; (iii) topological advantage of systolic array; and (iv) lower inference latency, DTQAtten costs less energy than the other two designs.

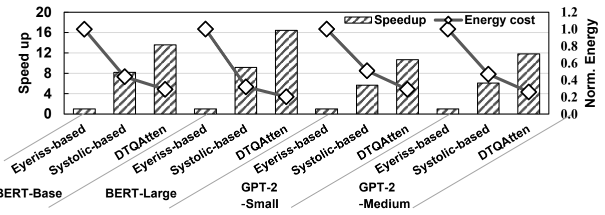


Fig. 8. Performance comparisons with each NN-based accelerator.

The comparisons among MNNFast, A^3, SpAtten and DTQAtten are shown in Tab. I. These four attention accelerators all explore the sparsity in attention layer. However, only DTQAtten considers fine-grained token-based quantization. DTQAtten achieves the highest throughput (7.94× compared with MNNFast) and the smallest area cost (67% of the area cost in A^3) during the four designs. The high throughput is contributed by (i) the high compression ratio using our quantization method, which reduces a large amount of computational overhead and alleviates lots of bandwidth pressure for accelerators; (ii) our accelerator and optimization strategy can efficiently convert the low-precision computing into low latency, which is contrasted to SpAtten that only takes advantage of the bandwidth benefits of the low-precision data.

TABLE I
COMPARISONS AMONG FOUR ATTENTION ACCELERATORS

	MNNFast	A^3	SpAtten	**DTQAtten**
Technology	FPGA (28nm)	ASIC(40nm)	ASIC (40nm)	**ASIC (40nm)**
Frequency	1GHz (projected)	1GHz	1GHz	**1GHz**
Area (mm^2)	-	2.08mm^2	1.55mm^2	**1.41mm^2**
Throughput (GOP/s)	120 (1×)	221 (1.8×)	360 (3.0×)	**952.8 (7.94×)**
Energy Effi. (GOP/j)	120 (1×)	269 (2.2×)	382 (3.2×)	**1298.4 (10.82×)**
Area Effi. (GOP/s/mm2)		106 (1×)	238 (2.2×)	**678.4 (6.4×)**

The smallest area overhead is mainly because the budget of the 4-bit PEs in our design is smaller (about 71% of the area of the 12-bit PEs in SpAtten) than the high-precision PEs in other designs. Besides, the energy cost of our design is also the lowest.

VI. CONCLUSION

In this paper, we find that different tokens in attention-based NLP models show different tolerance to noise. Inspired by the similarity between noise and quantitative loss, we propose to dynamically quantize tokens with different precision (0-bit, 4-bit and 8-bit) according to their importance level to reduce the computation complexity without losing accuracy. Moreover, we propose a hardware architecture with an optimization strategy to exploit our quantization algorithm efficiently. The experiments show that our algorithm-architecture co-design DTQAtten outperforms other attention accelerators in accuracy, performance and energy efficiency.

REFERENCES

[1] D. W. Otter *et al.*, "A survey of the usages of deep learning for natural language processing," *TNNLS*, 2021.
[2] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.
[3] A. Radford *et al.*, "Language models are unsupervised multitask learners," in *OpenAI Blog*, 2019.
[4] Wang *et al.*, "Apq: Joint search for network architecture, pruning and quantization policy," in *CVPR*, 2020.
[5] Z. Song *et al.*, "Drq: Dynamic region-based quantization for deep neural network acceleration," in *ISCA*, 2020.
[6] S. Shen *et al.*, "Q-bert: Hessian based ultra low precision quantization of bert," in *AAAI*, 2020.
[7] S. Kim *et al.*, "I-bert: Integer-only bert quantization," *ICML*, 2021.
[8] Wang *et al.*, "Spatten: Efficient sparse attention architecture with cascade token and head pruning," in *HPCA*, 2021.
[9] H. Jang *et al.*, "Mnnfast: A fast and scalable system architecture for memory-augmented neural networks," in *ISCA*, 2019.
[10] T. J. Ham *et al.*, "A^3: Accelerating attention mechanisms in neural networks with approximation," in *HPCA*, 2020.
[11] Vaswani *et al.*, "Attention is all you need," in *NIPS*, 2017.
[12] Lin *et al.*, "Towards fully 8-bit integer inference for the transformer model," in *IJCAI*, 2020.
[13] Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *ISCA*, 2017.
[14] H. Genc *et al.*, "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *DAC*, 2021.
[15] O. Zafrir *et al.*, "Q8BERT: quantized 8bit BERT," *CoRR*, 2019.
[16] J. Snoek *et al.*, "Practical bayesian optimization of machine learning algorithms." *NIPS*, 2012.
[17] H. Yu *et al.*, "Search what you want: Barrier panelty NAS for mixed precision quantization," 2020.
[18] C. Chelba *et al.*, "One billion word benchmark for measuring progress in statistical language modeling," 2013.
[19] Y.-H. Chen *et al.*, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," 2017.
[20] "Cacti 6.0: A tool to model large caches," *Bragantia*, 2009.