FACT: FFN-Attention Co-optimized Transformer Architecture with Eager Correlation Prediction

Yubin Qin*
qyb20@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Yang Wang*
wangyang_imec@mail.tsinghua.edu.cn
Tsinghua University
Beijing, China

Dazheng Deng
ddz20@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Zhiren Zhao
zhaozr21@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Xiaolong Yang
yangxl21@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Leibo Liu
liulb@mail.tsinghua.edu.cn
Tsinghua University
Beijing, China

Shaojun Wei
wsj@mail.tsinghua.edu.cn
Tsinghua University
Beijing, China

Yang Hu
hu_yang@mail.tsinghua.edu.cn
Tsinghua University
Beijing, China

Shouyi Yin†
yinsy@mail.tsinghua.edu.cn
Tsinghua University
Beijing, China

ABSTRACT

Transformer model is becoming prevalent in various AI applications with its outstanding performance. However, the high cost of computation and memory footprint make its inference inefficient. We discover that among the three main computation modules in a Transformer model (QKV generation, attention computation, FFN), it is the QKV generation and FFN that contribute to the most power cost. While the attention computation, focused by most previous works, only has decent power share when dealing with extremely long inputs. Therefore, in this paper, we propose FACT, an efficient algorithm-hardware co-design optimizing all three modules of Transformer. We first propose an *eager prediction* algorithm which predicts the attention matrix before QKV generation. It further detects the unnecessary computation in QKV generation and assigns mixed-precision FFN with the predicted attention, which helps improve the throughput. Further, we propose FACT accelerator to efficiently support eager prediction with three designs. It avoids the large overhead of prediction by using log-based add-only operations for prediction. It eliminates the latency of prediction through an out-of-order scheduler that makes the eager prediction and computation work in full pipeline. It additionally avoids memory access conflict in the mixed-precision FFN with a novel diagonal storage pattern. Experiments on 22 benchmarks show that our FACT improves the throughput of the whole Transformer by 3.59× on the geomean average. It achieves an enviable 47.64× and 278.1× energy saving when computing attention, compared to previous attention-optimization-only SOTA works ELSA and Sanger. Further, FACT achieves an energy efficiency of 4388 GOPS/W performing the whole Transformer layer on average, which is 94.98× higher than Nvidia V100 GPU.

CCS CONCEPTS

• **Hardware** → **Application specific integrated circuits**; **Application specific processors**; • **Computing methodologies** → *Natural language processing*; *Computer vision*; Machine learning algorithms.

KEYWORDS

transformer, hardware accelerator, efficient computing, algorithm-hardware co-design, neural network

ACM Reference Format:
Yubin Qin, Yang Wang, Dazheng Deng, Zhiren Zhao, Xiaolong Yang, Leibo Liu, Shaojun Wei, Yang Hu, and Shouyi Yin. 2023. FACT: FFN-Attention Co-optimized Transformer Architecture with Eager Correlation Prediction. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA '23), June 17–21, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3579371.3589057

1 INTRODUCTION

Transformer model [60] has been under the spotlight in recent years due to is outstanding breakthroughs in various artificial intelligence (AI) fields, such as natural language processing (NLP) [4, 10, 27, 33, 44–46, 51, 53, 54, 60], computer vision [5, 11, 30, 34, 35, 43, 48, 71, 73], recommendation systems [6, 57], etc. Its enviable performance can be attributed to the *attention mechanism*, first introduced in [60]. It allows the model to capture correlations between contexts (a.k.a tokens) among the input and assign different weights to each token accordingly. Although the global correlation aggregation mechanism enhances the capability of the model, it introduces large power and latency costs. Taking ImageNet classification [9] as an example, the ViT-G/14 model [71] achieves 2.91% higher accuracy than typical CNN model BiT (ResNet-152×4) [26]. However, such accuracy improvements come with 2.36× higher

*Both authors contributed equally to this research.
†Corresponding author.

ISCA '23, June 17–21, 2023, Orlando, FL, USA

ACM ISBN 979-8-4007-0095-8/23/06.
https://doi.org/10.1145/3579371.3589057

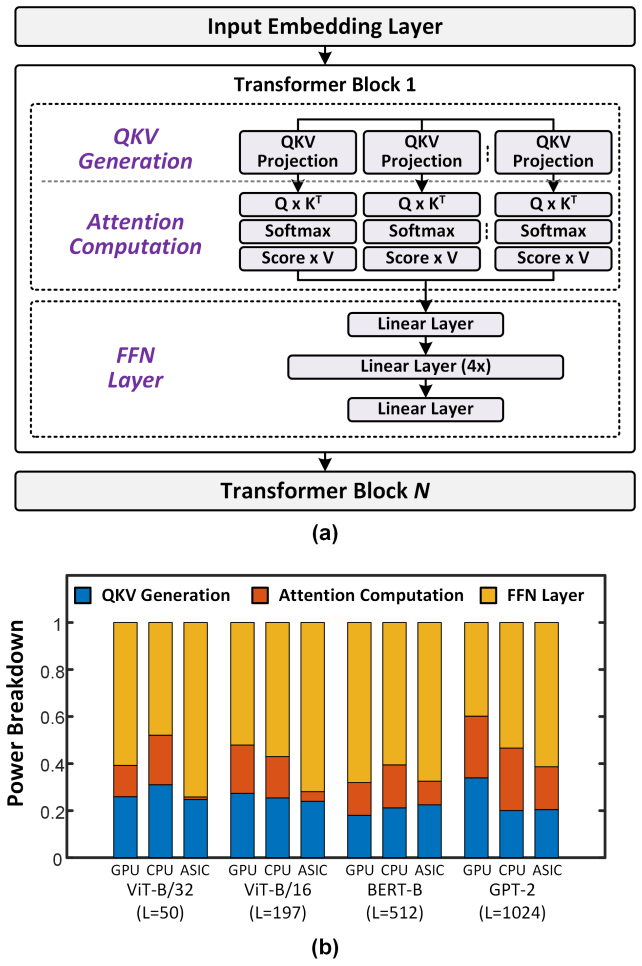

Figure 1: (a) Basic components of a Transformer model. (b) Power breakdown of different Transformer models on Nvidia V100 GPU, XEON Platinum 8168 CPU, and 28nm NVDLA Processor.

energy consumption (evaluated on NVDLA processor with 28nm CMOS technology, same below). Such high costs raise challenges for Transformer inference, making it urgent for an efficient processor.

Figure 1(a) shows the components of a Transformer model and the inference power breakdown analysis. Typically, a Transformer model comprises several identical Transformer layers (TLs) connected in sequence. Each TL has two main parts: *self-attention* and *feed-forward network (FFN)*. The self-attention can be further divided into two components: *QKV generation* and *attention computation*. These computation modules all contribute to the power consumption of Transformer, and their power breakdown varies when the input token length changes. As shown in Figure 1(b), when the token length is relatively small, such as 50 in ViT-B/32, the QKV generation and FFN costs the 21.9%, 60.8% on GPU, and 23.1%, 70.7% on ASIC. For long tokens, such as 1024 in GPT-2, the two parts cost 58.4% of total power. Therefore, the optimization of attention computation is helpful for some large models, while

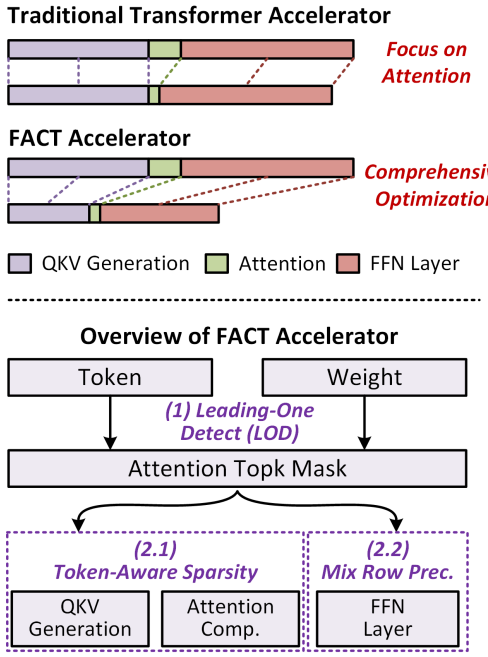

Figure 2: Overview of FACT accelerator and comparison of design concept with previous accelerators.

optimizing the QKV generation and FFN is compulsory for the majority of Transformer models.

Unlike previous works using attention's redundancy to optimize the attention computation, **our insight is to predict such redundancy at the beginning of a TL to optimize all three modules: QKV generation, attention computation, and FFN.** We propose an eager prediction (EP) mechanism to achieve this. Eager means: it predicts the attention matrix early before QKV generation. Figure 2 shows its features and two steps. **Step 1**: it predicts the attention matrix based on input tokens, W_Q, and W_K before QKV generation. It then obtains the mask that indicates important attention computation based on row-wise top-k operation. **Step 2**: it reduces the computation in QKV generation and FFN from two perspectives. From the inter-head perspective, it finds the redundancy of the QKV matrix in each head based on the attention mask. For one thing, for an attention row dominated by one element, it does not need the Q tensor to compute the attention score. For another, the KV tensor is non-contributing if the corresponding column in attention matrix is not selected by any row-wise top-k. From the intra-head perspective, it aggregates the row-wise top-k results of all heads and assigns different quantization precision to each row of tokens in the FFN computation. Intuitively, tokens with more top-k selections contribute more to this Transformer layer, while the opposite means that the token has less impact on the output. EP effectively reduces the computation of all three modules of Transformer by over 10× within negligible accuracy loss since it removes the less contributive tokens based on the attention matrix.

Despite the outstanding acceleration from algorithm analysis, a naive implementation of EP suffers from three challenges and

thus cannot effectively accelerate Transformer inference. First, EP is an aggressive cross-stage prediction: it first predicts the Q,K matrices and then predicts the result of $Q \times K^T$. It must achieve high prediction accuracy within low power overhead to ensure efficiency improvement. Second, the cross-stage prediction suffers from large latency. The PE array has to wait until the prediction of three matrices is complete. Third, previous sequential data storage pattern fails to satisfy the need to store the row-wise mixed-precision data with uniform patterns in FFN.

We proposed a dedicated hardware accelerator, FACT, to support EP effectively. Compared to naive implementation, which only has a limited 26.6× energy saving over Nvidia V100 GPU, FACT improves its performance with three innovations. Evaluated on 22 benchmarks, FACT achieves an average energy efficiency of 4388 GOPS/W, which is 94.98×, 3.91×, and 22.86× higher than Nvidia V100 GPU, state-of-the-art accelerators ELSA and Sanger. Overall, the innovations are listed as follows:

- We analyze the power breakdown of Transformer inference in detail. We point out that optimizing QKV generation and FFN layers is the key to efficiency.
- We propose an eager prediction (EP) method to estimate the attention early before the QKV generation. Its result is used to remove redundancy in QKV generation and reduce precision of unimportant rows in FFN.
- For the challenge of EP precision and power overhead, we propose an efficient EP unit with a one-hot adder tree. It carries out EP with leading-one detection and add-only operation, which greatly reduces the power overhead.
- For the challenge of EP latency, we design an out-of-order QKV generation scheduler based on KV-differential sequence. It maximizes hardware utilization and eliminates the latency of EP.
- For the challenge of row-wise mixed precision storage, we design a novel diagonal storage pattern for the mixed precision data readouts in FFN. It fully avoids memory access of unnecessary bits by shifting addresses in a diagonal pattern between SRAM banks.

This paper is arranged as follows: Section II describes the basic modules and algorithm of a Transformer model, and presents our design motivation based on a detailed power breakdown analysis. Section III presents our EP mechanism and shows how to extend the estimated attention to QKV and FFN acceleration. Section IV shows the architecture of the FACT accelerator and presents three hardware innovations for efficiency. The comprehensive experiment results, discussion of related work, and conclusions are in Sections V, VI, and VII, respectively.

2 BACKGROUND AND MOTIVATION

2.1 Basic Modules of Transformer Layer

A Transformer model is composed of several TLs with attention mechanism. The input token of a TL goes through three main computation modules, as Algorithm 1 shows. The first is the QKV generation (line 2-4). In this stage, the input token is projected to three spaces which are termed query (Q), key (K), and value (V). Specifically, it takes an $L \times D_e$ input token matrix (L stands for token length and D_e stands for the model's embedding dimension), and

Algorithm 1 Attention Mechanism

Input: $T \in \mathbb{R}^{L \times Dim}$: input token
Input: $W_Q, W_K, W_V \in \mathbb{R}^{h \times Dim \times D}$: weight of projection
1: **for** $i \in 0, 1, \ldots, h-1$ **do**
2: $\quad Q[i] \leftarrow T \cdot W_Q[i]$
3: $\quad K[i] \leftarrow T \cdot W_K[i]$
4: $\quad V[i] \leftarrow T \cdot W_V[i]$
5: $\quad Attention[i] \leftarrow Q[i] \cdot K[i]^T$
6: $\quad Attention\_score[i] \leftarrow softmax(Attention[i]/\sqrt{D})$
7: $\quad E[i] \leftarrow Attention\_score[i] \cdot V[i]$
8: **end for**
9: $E_{output} \leftarrow FFN(Concat(E[i]))$
Output: $E_{output} \in \mathbb{R}^{L \times Dim}$

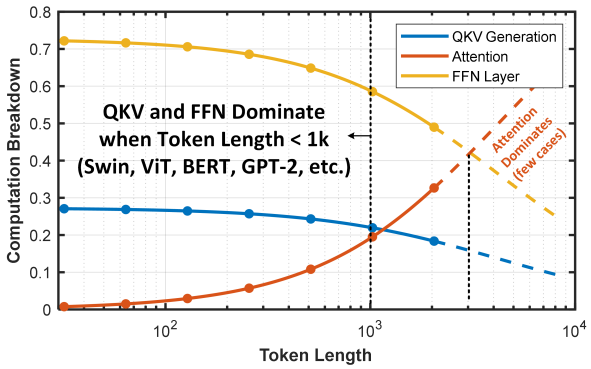

Figure 3: Computation breakdown of various token length with a fixed embedding dimension of 768.

use three linear projection layer to generate three $L \times D$ matrices. Normally, the D is D_e/h, which represents the multi-head structure. Then, the second stage is attention computation (line 5-7). It takes the $L \times D$ Q and K matrices and performs matrix multiplication to get the $L \times L$ attention matrix, which represents the correlation of each token pair. Subsequently, each row of the attention matrix is applied with *softmax* function, which normalizes the value into probabilities (attention score). For example, the value in i^{th} row and j^{th} column of the attention matrix represents the correlation from token i to token j. The higher the score is, the more contextually related the two tokens are. Finally, the score is used to weighted sum the V matrix to get an $L \times D$ output embedding. The last stage is the FFN, which are simple linear layers. After the output embeddings of all the heads are ready, the FFN takes them as input and performs linear projection. In the majority of Transformer models, the FFN is a bottleneck structure, which first projects the $L \times D_e$ input embedding into a high dimension $L \times 4D_e$, and then downgrades the intermediate result back to $L \times D_e$.

2.2 Computational Properties Analysis

According to Algorithm 1, the complexity of the three main computation modules can be confirmed. The QKV generation and FFN have a complexity of $O(LD_e^2)$, and that of attention computation is

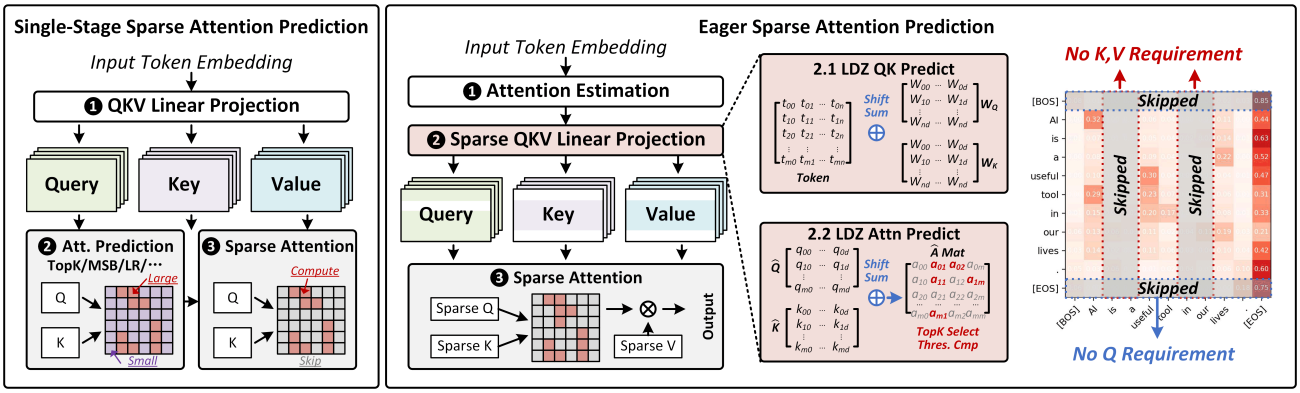

Figure 4: Procedures of eager prediction and comparison with traditional single-stage prediction.

Table 1: Summary of SOTA Transformer Accelerators

Accelerator	Optimization		
	Attention		FFN Layer
	QKV Gen.	Attention	
A^3[17]	×	√	×
ELSA[18]	×	√	×
DOTA[42]	×	√	×
Sanger[37]	×	√	×
BESAPU[64]	×	√	×
EdgeBERT[58]	×	√	×
SpAtten[63]	×	√	√*
FACT	√	√	√

* SpAtten reduces the FFN computation by token pruning.

$O(L^2D_e)$. Figure 3 shows the computation breakdown of the three modules with various token lengths under the same typical D_e of 768 (typical value for Vanilla Transformer [60], BERT[10], ViT[11], T5[46], etc.). For most of the prevalent Transformer models and tasks, the token length L would not exceed 1k, in which case the QKV generation and FFN account for over 70% of computation and are dominant. When the token length increase to over 3k, the attention computation might become the bottleneck (indicated by the dashed line in Figure 3). However, it is far from a common case for two reasons. First, most NLP Transformers have an upper limit for the token length L, such as 512 (BERT [10]), 1024 (GPT-2 [45]), 2048 (GPT-3 [4]), and the CV models have fixed token lengths, such as 49 (Swin Transformer [35]) and 256 (ViT-H/14 [71]), all having dominant QKV and FFN rather than attention. Second, although some models support long input, such as Longformer [3], LongT5 [16], Big Bird [70], these works already have algorithm optimization in terms of the $O(L^2D_e)$ attention computation. E.g., the Longformer and Big Bird use sparsity to reduce the complexity of attention computation to $O(LlogLD_e)$, much smaller than that in BERT model. Therefore, it can be concluded that optimizing the attention computation works when Transformer takes long input, while optimizing the QKV generation and FFN is essential for all Transformer models.

Table I lists the state-of-the-art Transformer accelerators and their optimizations. As far as we are concerned, most accelerators focus on attention computation. For example, Sanger, DOTA, A^3 [17, 37, 42] uses strategies to predict the value of attention matrix, and computes the large values with high precision. ELSA [18] uses random projection of Q matrix for a similar purpose. [64] uses approximate computation to reduce the computation cost of small values of attention matrix. However, as the previous analysis shows, these works cannot effectively optimize most of the Transformer models with decent token lengths. SpAtten and EdegBERT [58, 63] are not designed for solving the power bottleneck, but their methods (cascaded token pruning / early termination) can somewhat alleviate the problem. These two methods are also restricted to certain models, which is less versatile. This phenomenon motivates us to propose an efficient inference engine for general Transformer with the linear layer fully optimized. To the best of our knowledge, FACT is the first work that explicitly accelerates all three modules in Transformer model.

3 EAGER PREDICTION MECHANISM

This section introduces our algorithm optimization: EP, and its extension to acceleration in QKV generation and FFN. We first show a cross-stage correlation prediction mechanism that finds important attention scores at the very beginning of a TL. Further, we propose two methods to extend the attention prediction result to eliminate unnecessary computation of QKV generation and FFN.

3.1 Eager Correlation Prediction

Many previous works utilize the redundancy in attention score computation [17, 18, 37]. The attention matrix ($A = Q \cdot K^T$) goes

through a row-wise *softmax* function to obtain the attention score S. Each value in a row of the S mat is probability – being positive values and summing up to one. Therefore, if there exists a few large probabilities in the S, the rest are very small and can be safely skipped since they have little impact on the output. A naive intuition is to predict where the small values in the A matrix are, and skip the related unnecessary computations, which is applied in previous accelerators, as abstracted in Figure 4(a). However, only computation of $Q \cdot K^T$ can be saved since predicting the A matrix requires Q and K projection at least. As described in Section II.B, generating the QK matrices causes much more computation and power than the $Q \cdot K^T$, leading to suboptimal improvement. To solve this dilemma, we propose EP with cross-stage log-based inner-product estimation, which can reduce not only attention score computation but also the QKV linear projection.

Algorithm 2 Eager Prediction with Correlation Estimation

Input: $Token(T), Weight(W_Q, W_K, W_V)$
Input: k, r ▷ Top-K ratio and FFN threshold
Output: M^A, M^Q, M^{KV} ▷ Sparsity mask
Output: M^{FFN} ▷ Low precision mask for FFN

1: $E^T \leftarrow LOD(T)$
2: $E^{WQ} \leftarrow LOD(W_Q)$
3: $E^{WK} \leftarrow LOD(W_K)$
4: $\hat{Q_{i,j}} \leftarrow \sum_k Sgn(T_{i,k} W^Q_{k,j}) << (E^T_{i,k} + E^{WQ}_{k,j})$
5: $\hat{K_{i,j}} \leftarrow \sum_k Sgn(T_{i,k} W^K_{k,j}) << (E^T_{i,k} + E^{WK}_{k,j})$
6: $\hat{A_{i,j}} \leftarrow \sum_k Sgn(\hat{Q_{i,k}} \hat{K^T_{k,j}}) << (\hat{Q_{i,k}} + \hat{K^T_{k,j}})$
7: $M^A \leftarrow \hat{A_i} \geq top\_k(\hat{A_i}, k)$
8: $M^Q \leftarrow max(\hat{A_i}) - 2^{nd}\_max(\hat{A_i}) \leq Thr$
9: $M^{KV} \leftarrow \prod_k M^{EA}_{k,i}$
10: $M^{FFN} \leftarrow \sum M^{EA} \leq L \times C \times k \times r$

Figure 4 depicts our method. The process of EP has two branches: the prediction branch (process 1) and the computation branch (process 2 and 3). The prediction branch takes the token T and weight W_Q,W_K as the input. Its goal is to predict an approximate value of A matrix, named $\hat{A}$. Then use the predicted results to obtain the attention values and Q, K, Vs whose computation can be skipped. This information is sent to the computation branch in the form of bit-mask. The computation branch works in pipeline (Section IV.C) to perform attention computation based on the results of the prediction branch, with all the unnecessary values skipped.

The process of the prediction branch is detailed described in Algorithm 2. For simplicity, the bias of linear projection is omitted. It modifies the attention mechanism (Line 2-5 in Algorithm 1). It first predicts estimated projection results, $\hat{Q}, \hat{K}$. Then use the two estimated matrices to obtain the $\hat{A}$. However, even using low-precision matrix multiplication (e.g. using half-precision with MSBs only) causes significant power consumption in the prediction branch. A low-power prediction is a must. Note that it is the relative value of the inputs that affect the *softmax* results. As long as one can predict the interval in which each value of A matrix lies, the relative value can then be obtained and the unimportant attention can be confirmed. Based on this observation, we propose a log-based

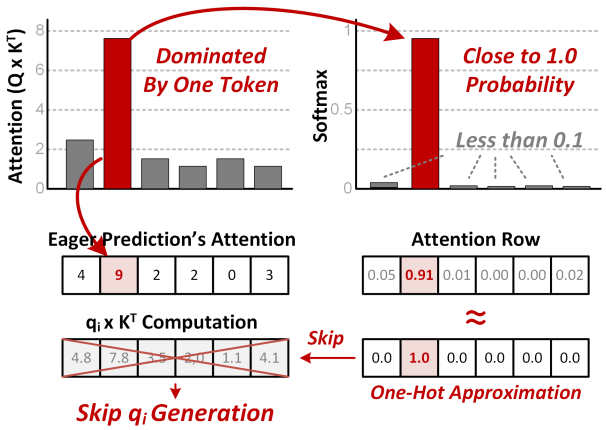

Figure 5: Query sparsity with one-hot softmax probability approximation.

multiplication-free prediction. It first transforms the data into log-domain by using a leading-one detector (LOD), and then substitutes the costly multiplication with low-power shift-and-add operations. Specifically, an INT-type number α can be decomposed as follows:

$$\alpha = Sign \times 2^{(W-LO-1)} \times M \tag{1}$$

W is the bit-width of the INT number (e.g. 8 for INT8). LO is the leading-zero count of α, which is the continuous number of repetitions of the sign bit. M is a mantissa lying between 1 and 2, like the floating-point number does. Obviously, the larger the LO of an INT number, the smaller its absolute value is. I.e., LO can be an approximation of an INT number. Based on this decomposition, the multiplication of two INT-type numbers α and β can be written as follows:

$$\begin{aligned} \alpha \times \beta = & XOR(Sign_\alpha, Sign_\beta) \\ & \times 2^{(W_\alpha + W_\beta - (LO_\alpha + LO_\beta) - 2)} \\ & \times (M_\alpha \times M_\beta) \end{aligned} \tag{2}$$

Since the bit width W is fixed for certain operands, the sum of LO can approximately characterize the magnitude of the product of two numbers. Combined with shifting and sign bit, the result of matrix multiplication can thus be predicted.

EP first performs leading-one detection on token T and weight W_Q and W_K before the prediction (Line 1-3 in algorithm 2). Then the corresponding LOs are shifted and accumulated to obtain estimated query and key projection $\hat{Q}, \hat{K}$ with a one-hot accumulator (section IV.B). The same procedure is carried out for the estimated attention $\hat{A}$ (Line 6). In our experiment, using the first "1" (a.k.a. leading one) can achieve over 90% hit rate for predicting the top-k result of the A matrix. It is not worthwhile to detect attention with more LO bits since using n LO bits for prediction requires 2^n adders. Then, a parallel top-k filtering is performed to each row of $\hat{A}$ to identify the keys to which each query should attend. The non-top-k values are regarded as small attention scores with their computation skipped.

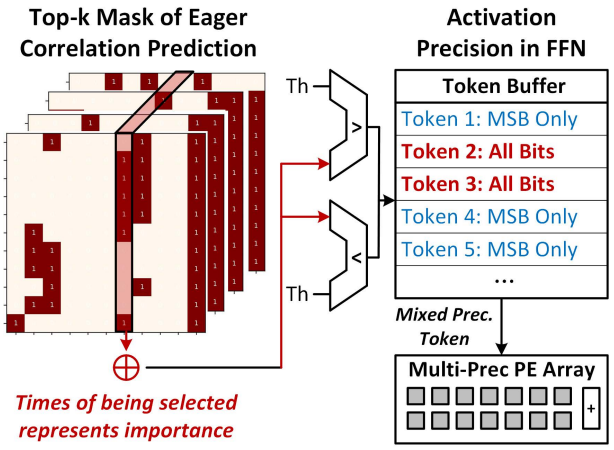

Figure 6: Token-wise mixed precision in FFN.

3.2 Attention-distribution-aware QKV Generation

The EP obtains the location of effective attention at the very beginning of a layer. The next step is to extend the sparsity in attention to accelerate the QKV generation and FFN layers. Strategies are used to predict the sparsity in QKV and reduce FFN computation in separate ways.

KV sparsity. It derives directly from the top-k result of the predicted $\hat{A}$ matrix. If a column in the $\hat{A}$ matrix has no values selected by the top-k, the key tensor related to this column is no longer required and can be safely pruned. Similarly, since the V matrix is multiplied by the attention matrix, the row with the same index in V matrix has no effect on the output, either, and can be safely removed.

Q sparsity. The sparsity of Q matrix cannot be obviously obtained from the top-k result of the $\hat{A}$ matrix because the top-k is a row-wise operation. I.e., it selects important attention for each row, and all the queries are required in this case.

To reduce the computation of Q generation, we propose a softmax-distribute-aware pruning based on value difference. Figure 5 shows this method. The *softmax* function exaggerates the difference in the values of the inputs. If the input has one token apparently larger than the rest, its output probability would be dominating and very close to 1.0. This phenomenon is quite common in Transformer for two reasons. 1. Intuitively, each row in the attention matrix represents a contextual correlation from one token to all the tokens. The token pair with semantic association tend to have a larger attention value than others. 2. Many Transformer models add special tokens to the input based on its function. For example, in BERT model, [BOS], [EOS] token are used to indicate the begin and end of a sentence, and [MASK] token represents the place to fill in a word in the language modeling task. So is the [CLS] token in ViT and many other models. Hence, when the $\hat{A}$ matrix is obtained from EP, the difference between the 1^{st} and 2^{nd} value of each row is compared to a threshold (we choose 3 based on experiments). If the former is larger, EP regards this row as being dominated by the largest token and directly uses a one-hot tensor as the *softmax* result where the largest token is assigned with 1.0 probability. In this way, the QK generation and attention computation related to this row can be fully skipped, and all that is needed is to copy the corresponding V tensor as the output.

3.3 Token-wise Mixed Precision FFN computation

Analysis in section II shows that the FFN layer is another main source of computation power. To alleviate this bottleneck, we propose a token-wise mixed precision to assign full precision to the important tokens while reducing the precision for the rest, based on EP result. Figure 6 shows its procedure.

EP finds out the essential keys and values that are highly associated with each query by top-k operation. When a token's key is selected, it can be viewed as contributive to the attention process. Otherwise, it is not. In other words, the number of a token's key being selected by the top-k in EP reveals the token's significance. Supposing that the top-k ratio is k% and the TL has h heads. Then on average, each token is selected $t = L \times h \times k\%$ times. We use a hyper-parameter r as a threshold to judge the contribution of a token. If a token's key is selected more than $r \times t$ times, it s regarded as an important one and its FFN is computed with full quantization bit width. Otherwise, the token's FFN operation is carried out with low precision (only 4MSBs of token embedding are used in this work).

4 ARCHITECTURE AND HARDWARE INNOVATION

4.1 Overall Architecture

To efficiently accelerate Transformer model with EP, we design a dedicated hardware accelerator, FACT. It is designed to accelerate the whole model, including QKV, attention computation and FFN. When the host processor is carrying out Transformer tasks, it prepares the input and weight in the external memory. FACT fetches these data for computation and returns the final outputs back to the memory. Figure 7 shows FACT's overall architecture. It contains six main modules: on-chip SRAM storage, an eager prediction unit, an out-of-order computation scheduler, a PE array, an auxiliary processing unit (APU), and a chip controller.

Operation Overview. The FACT accelerator is responsible for almost all three components of a TL, including 1) QKV generation, 2) attention computation, and 3) FFN layers. It performs attention head by head. The FACT accelerator takes the token embedding and projection weight as the input. While loading the input from external memory, the APU unit starts the preprocessing of input, such as layer normalization, and stores the results in the token buffer for computation. Once the input token embedding is ready, FACT starts the EP to estimate the attention correlation by log-based add operation in the prediction unit. It generates the top-k mask and sparse computation map of processes 1) and 2). As long as part of computation control is generated from EP, the scheduler controls the PE array to start QKV generation. The results are stored in the temp buffer in an out-of-order sequence (section IV.C). The procedure of attention computation starts after 1) finishes the QKV generation of one head. During 2), the PE array fetches Q and K

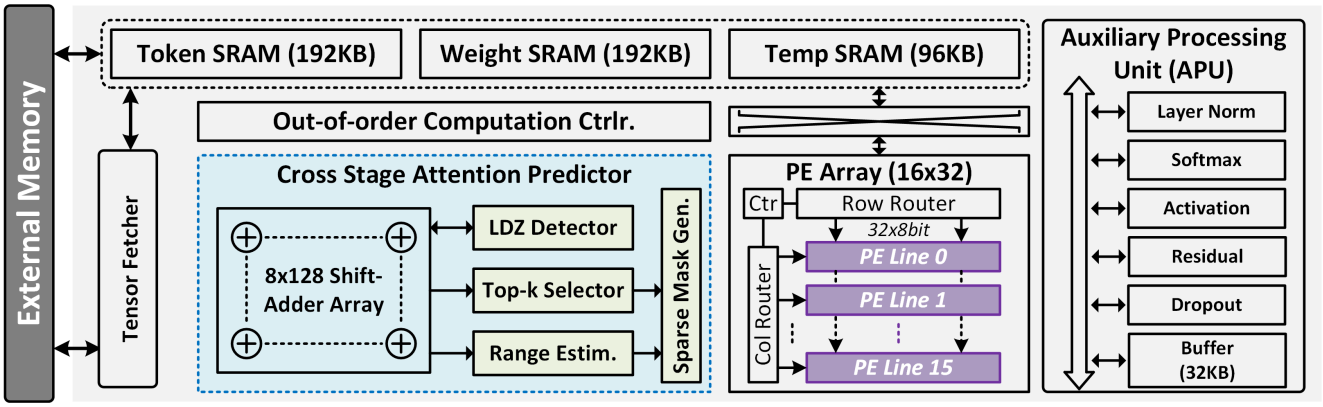

Figure 7: Overall architecture of FACT accelerator.

matrices from the temp buffer and performs sparse attention computation based on the top-k mask. The APU works simultaneously to perform the row-wise *softmax* and returns the attention scores back to the weight buffer. The process of 1) and 2) continues until all the heads have their attention output embeddings computed. Finally, FACT fetches the concatenated tensor from external memory and performs 3) with mixed precision.

4.2 EP Unit with One-hot Adder Tree

For energy-efficient EP, we designed a processing unit, as shown in Figure 8 . It consists of 4 main parts. Two FIFOs of depth 8 and width 64b and 1024b are used to cache the operands of matrix computation to be predicted. The sliced matrices are sent to the zero-eliminator, where all computations containing zeros are removed. Then two arrays of cascaded LO detectors finding the LO for the input values. If the value is positive, the detector outputs its LO number. Otherwise, the detector outputs the LO of the input's opposite number. The output is 5-bit data, where the 4 LSBs are for the LO count, and the MSB is the sign bit of the input. The LO detection results are fed to a 8x128 adder array, which performs weight-stationary add to estimate the multiplication result for at most 8 rows of input.

The input of this prediction unit supports two types of precision: 8bit and 16bit. This is because EP needs to predict the value of QK and attention. In the former case, the input is 8bit token embedding and weight, and the EP unit works in 8bit input mode. And the bit width of the output of this process is not determined. Its output is truncated to at most 16 bits of data - depending on the distribution of the output data, and then the output prediction of attention is performed. To support 8/16bit mixing, we use a cascaded LOD. Each detector consists of two 8-bit detectors connected in series. When the input is 8-bit, the two detectors output data independently. When the input is 16-bit, the detector in charge of 8 MSB transmits the output result of the last level to the other detector and use it as one of the input signals for the AND gate.

We design a counter-based adder tree for the one-hot data in EP. Note that after the add of Q's LO and K's LO, the results should be accumulated. However, this accumulation is costly, whose area overhead is almost equal to half of an FP16 adder. We observe that

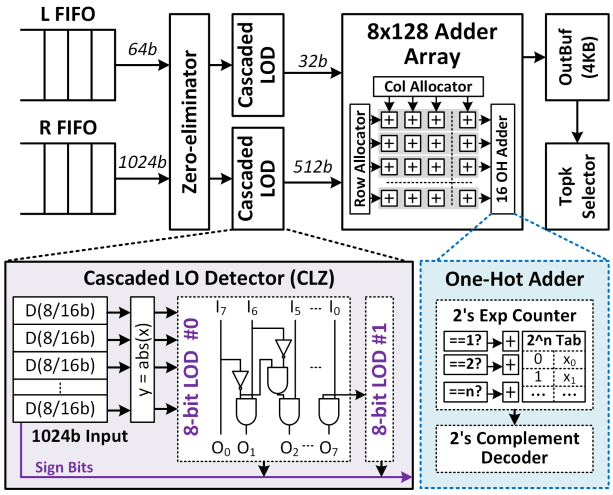

Figure 8: Eager prediction unit with cascaded LOD and one-hot adder tree.

the input is one-hot – the place of "1" represents the sum of two LOs. We use parallel counters to count the number of occurrences of each LO in the input. The count num indicates the exponent of 2 in the arithmetic result. Then a decoder is applied to turn the count into 2's complement.

4.3 Out-of-order QKV Generation Scheduler

The EP algorithm estimates the attention correlation and finds the location of large values without QKV. Thus, it sparsifies the QKV generation and the attention computation. However, there is a latency problem with this process – the PE array maintains in idle until the result of EP is generated. However, this latency cannot be ignored because EP is a cross-stage prediction: it first estimates the QK and then the attention value. This problem leads to a decrease in PE utilization, compromising the efficiency gain of sparsity utilization.

To solve this utilization problem, we utilize the Transformer's in-sensitiveness with regards to the token sequence in attention

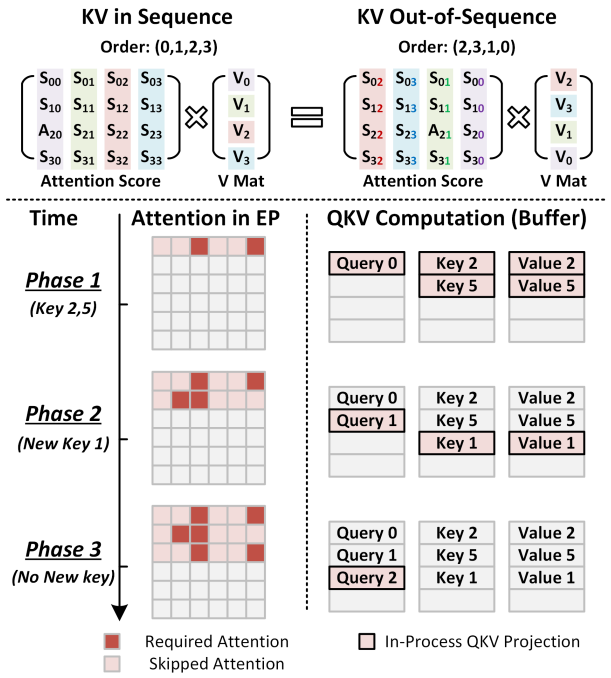

Figure 9: Out-of-order QKV generation schedule based on KV-differential computation sequence.

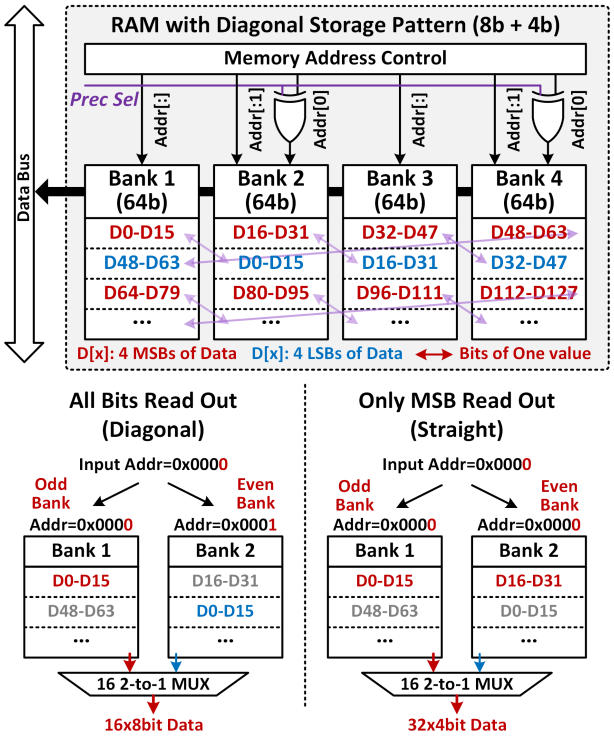

Figure 10: Data in SRAM with Diagonal Storage Pattern and two read-out modes.

computation and propose an out-of-order computation scheme, as shown in Figure 9. The output of attention score computing is multiplying attention score S with value matrix V. The data in the i^{th} row and j^{th} column of S represents the directed correlation of the i^{th} token's query to the j^{th} token's key. Supposing that the sequence of tokens in the key matrix and value matrix were swapped in the same pattern, the column of S would still match the token sequence in V, and would not change the output.

Based on this feature, FACT applies a fully pipelined computation flow with KV-differential order. The EP unit predicts the attention matrix row by row. Once one row of S is estimated, the keys and values required by this row are confirmed. Then the PE array directly starts KV projection computation and stores the result in the temp buffer. From then on, whenever a new row of attention is estimated, PE calculates the extra required KV. In the example in Figure 9, after phase 1 of prediction, the 2^{nd} and 5^{th} key and value are required and the PE starts computing. After phase 2 of prediction, the 1^{st} and 2^{nd} are required. However, token 2 has already been projected, and the PE only needs to compute the KV for the 1^{st} token. Since the KV data in the buffer are stored in the sequence of differential requirements, this computation sequence is named KV-differential order. In this way, the PE keeps working along with the EP and needs no waiting for the prediction result.

4.4 Diagonal Storage Pattern for Mixed Precision FFN

EP divides the computation of FFN into two modes: high-precision computation using full quantization bit-width and low-precision computation. In FACT accelerator, the precision of the two computations is INT8 and INT4(MSB), respectively. However, there is a hardware utilization problem here: since there is no way for EP to predict the top-k selection results of all heads until the last head's prediction is made. By then, however, the computation of all heads except the last head has already finished, and the tokens are stored in the same order as input tokens. Therefore, when calculating the FFN, all data are stored in SRAM as INT8. For the unimportant tokens, only 4 MSBs are required for computation. This feature contradicts the SRAM rule of accessing one whole row at a time, leading to a waste of memory access.

To solve this problem, we propose a special SRAM storage rule: diagonal storage pattern, as shown in Figure 10. Each data is first divided into 4MSBs (red) and 4LSBs (blue). The traditional storage rule is to store all bits of data in the same address, e.g. D0-D31 in address 0. But this results in only half the effective SRAM bandwidth when only MSBs of D0-D31 are required. The diagonal storage pattern uses a staggered storage method. For example, bank 0 of address 0 stores the MSBs of D0-D15, and their LSBs are stored in address 1 of bank 1. Address 0 of bank 1 is used to store the MSBs of D16-D31, and so on. In this type of storage, the MSBs and LSBs of one data are connected diagonally in space. When reading data, if full precision is needed, the memory address controller inverts the last bit of all even SRAM banks. In this way, both MSBs and LSBs of one data are read. Conversely, all banks get the same access address when only MSBs are needed. 16 2-to-1 MUXes are applied at the output edge of SRAM to adjust the sequence. In this way, the

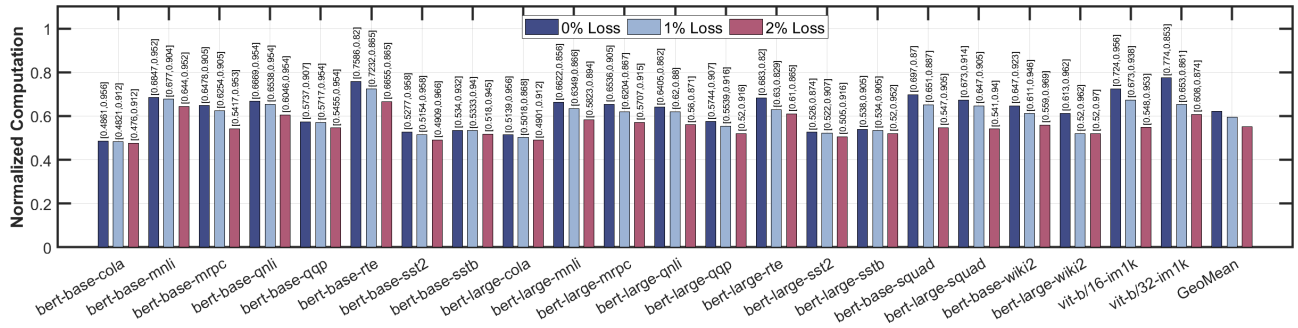

Figure 11: Computation reduction of EP with different accuracy loss tolerance. [X,Y] refers to the reduction of the whole computation and attention.

PE could fetch mixed-precision data in the same way as fetching normal single-precision data, with no memory footprint and cycle waste.

4.5 Other Modules

Memory. FACT has a total of 512KB SRAM storage, including 192KB token buffer, 192KB weight buffer, 96KB temp buffer and 32KB APU buffer. The token buffer can store a token matrix with 768 embedding dimension and 256 token lengths, which covers most Transformer cases. For tasks with longer tokens, FACT splits the task into several working phases. The weight buffer is able to store all the W_Q, W_K, W_V matrices of one attention head in QKV linear projection, and the rest is used to store the temp data in EP. The temp buffer is used to store QKVs in the generation stage and intermediate results. In FFN, the token buffer stores the input and weight buffer stores the weight. The result is stored in temp buffer and sent to external memory.

Auxiliary Processing Unit (APU). In order to fully support Transformer computation, FACT needs to support some functions other than matrix multiplication, such as softmax, dropout, layer norm, and residual connection, and we designed an arithmetic unit in FACT to implement these functions. The APU internal function modules and 32KB of storage are interconnected using a 512b data bus. The internal buffer is used to store input and intermediate results such as the sum of exponent in softmax computation or mean and var values in layer normalization.

5 EVALUATION

5.1 Experiment Setup

We evaluate the performance of FACT with several typical Transformer models and tasks. For NLP tasks, we select the BERT-base and BERT-large models [10] pre-trained on BookCorpus & English Wikipedia datasets and finetune them on 8 GLUE text classification tasks [62] (excluding WNLI due to its adversarial set split), Stanford Question Answering Dataset (SQuAD) v1.1 [47], and the WikiText-2 dataset [39]. Then we evaluate the inference performance. The maximum sequence length for BERT is 128/384/512 for GLUE, SQuAD, and Wiki-2, respectively. For CV tasks, we selected the ViT-B/32 and ViT-B/16 [11] for ImageNet-1k classification [9]

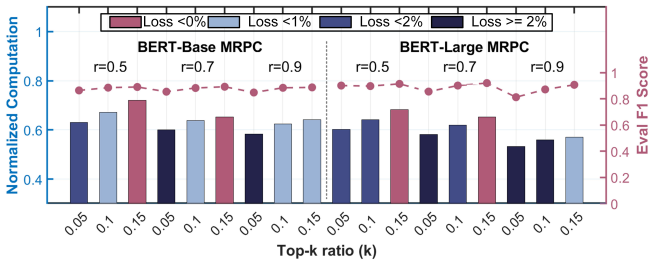

Figure 12: Computation reduction and accuracy of different (k,r) settings of EP.

by finetuning the checkpoint of ImageNet-21k. We utilize the open-source Huggingface [65] and PyTorch [40] libraries to construct the models. For all models, the finetuning lasts 5 epochs using AdamW optimizer with a learning rate of 3e-5 and cosine scheduling.

We perform RTL design for the FACT accelerator and synthesize it using Synopsys Design Compiler with 28nm 1P8M CMOS technology. Subsequently, we use DCG and ICC to complete the chip's placement and routing, generating the chip's layout and netlist. Based on the netlist, we utilize VCS and PT for post-layout simulation to obtain the chip's power consumption and latency. The data used in the simulation, such as inputs and weights, is generated by PyTorch and represents the actual data used during model inference.

5.2 Accuracy Evaluation

In EP, there are two hyperparameters that impact the accuracy of the model: top-k ratio k and row-wise mix-precision ratio r. A larger k or a smaller r means more aggressive acceleration and comes with a potential loss in accuracy. We selectively use grid search (GS) or successive halving (SH) methods to find the feasible hyperparameters during training according to the training cost of different tasks. Initially, we do a binary search and fine-tune all models under the condition of r=0 to determine the boundary of k. We find that k=0.25 ensures no accuracy loss when using EP. Based on this, we set a hyperparameter search space of k=0.05 to 0.25 with a step of 0.05 and r=0.1 to 1 with a step of 0.1 for a total of 50 models. For tasks with low training cost, such as MRPC, SST2, RTE, etc., GS is utilized by each time randomly selecting 4 sets of (k, r)

from the search space for finetuning and recording the accuracy. Note that GS does not traverse the entire search space, instead using a skipping strategy. If it finds a model with parameters (k_i, r_i) having no accuracy degradation, GS automatically skips all models with $(k > k_i, r < r_i)$ since they apparently are lossless but have less acceleration. Similarly, it skips models with $(k < k_i, r > r_i)$ when (k_i, r_i) denoting over 3% loss. For tasks with large training costs, SH is used. It starts with training all the models in the search space. After each epoch, half of the (k,r) sets with lower accuracy are removed. This process is repeated until only 16 sets remain. The skipping strategy in GS also applies here.

Figure 11 shows the computation savings and accuracy metrics of EP on several typical tasks with 0%/1%/2% error tolerance. Different end-to-end metrics are used for evaluation, including F1 for MRPC, QQP, SQuAD, accuracy for SST-2, QNLI, MNLI, RTE, ImageNet-1k, Matthews correlation for CoLA, Pearson correlation for STS-B, and perplexity for WikiText-2 language modeling. EP can on average reduce computation by 37.8%/40.6%/44.9% with 0%/1%/2% accuracy loss. Figure 12 shows the sensitivity analysis of k and r respectively based on BERT-Base and BERT-Large (MRPC) by changing each parameter independently. It shows that a smaller k or a larger r effectively reduces the computation, while introducing loss in accuracy. When $r \leq 0.7$, it is highly possible to perform EP without degradation, and over 85% of attention and its corresponding unnecessary QKV and FFN can be safely skipped. When $r \geq 0.9$, the setting of k should be careful because the loss might go beyond expectation.

5.3 Performance Evaluation

To evaluate the performance of FACT, we first use PyTorch to generate the processing data for FACT. Then we use PT to simulate FACT's behavior, which reveals the time consumption of computation and on-chip data preparation. Additionally, this simulation shows FACT's request for external memory. We use Ramulator to simulate the DDR4 behavior based on the request signal from FACT, and calculate the throughput. The baseline is the vanilla and EP versions of the same model running on a 32GB Nvidia V100 GPU with Intel XEON Platinum CPU. To compare FACT with GPU, we emulate 250 FACT accelerators arranged in 10 clusters working at 500MHz, which has a peak performance of 125TOPS@INT8, the same as that of V100 GPU. The inter-core bandwidth is 900GB/s, which is the same as V100. In our experiment, FACT requires a maximum inter-core bandwidth of 304GB/s when computing attention with k=0.05 with an average utilization in attention computation of 81.9%. Hence, the performance of 250 FACTs can be seen as linearly scalable.

Figure 13 presents the inference speed of V100 GPU and FACT accelerator using EP with different loss tolerance. It also involves a raw FACT which works without EP. EP enables an 1.07-1.51× of speed up on GPU with its sparsity. However, the GPU cannot fully exploit the EP results because it cannot handle high sparsity or fine-grained INT4/8 mix precision effectively. Nor can it run the LOD-based prediction efficiently. Comparably, FACT owns an average of 85.5% PE utilization due to its Transformer-oriented dataflow, which fully pipelines the one-hot prediction, MAC computation, and other functions such as softmax, resulting in almost double

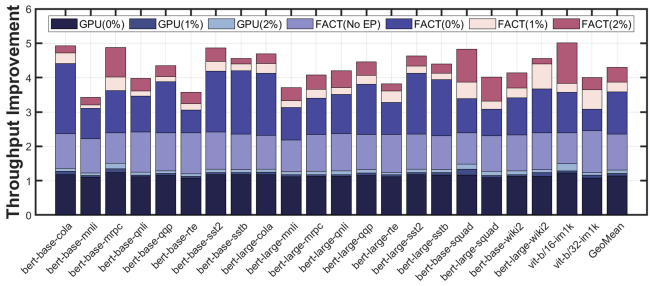

Figure 13: Throughput improvement of EP on GPU and FACT. X% in the brackets indicates accuracy loss tolerance.

Table 2: Area and Power breakdown of FACT at 500MHz

Module	Area[mm^2]	Power[mW]
PE Array	2.76	162.93
Eager Prediction - Adder	0.23	16.04
Eager Prediction - Others	0.02	1.59
APU - Softmax	0.32	13.65
APU - Others	0.24	10.30
SRAM Buffer	2.23	110.28
Others	0.22	22.28
Total	**6.03**	**337.07**

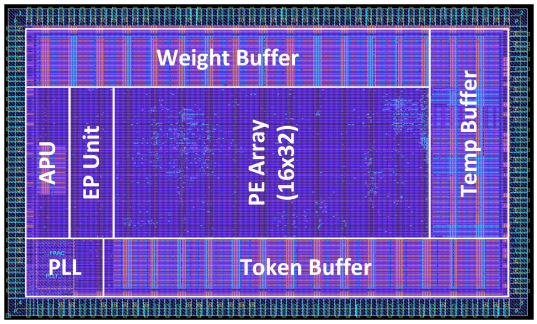

Figure 14: Post-layout of FACT accelerator

utilization than GPU. Further, the diagonal SRAM storage pattern is tailored to the INT4/8 mix precision, which totally avoids extra latency. Overall, FACT achieves an 3.59×/3.87×/4.30× of inference speed up with 0%/1%/2% accuracy degradation.

5.4 Area and Energy Evaluation

We complete the synthesis, placement and routing of FACT accelerator based on 28nm CMOS technology. Figure 14 shows its layout. We obtain its area based on ICC results and its working power based on VCS and Verdi simulation. Table II shows the power and area breakdown of FACT. It has a total area of 6.03 mm^2. The EP unit only takes 4.23% and 5.23% of area and power, which is a small

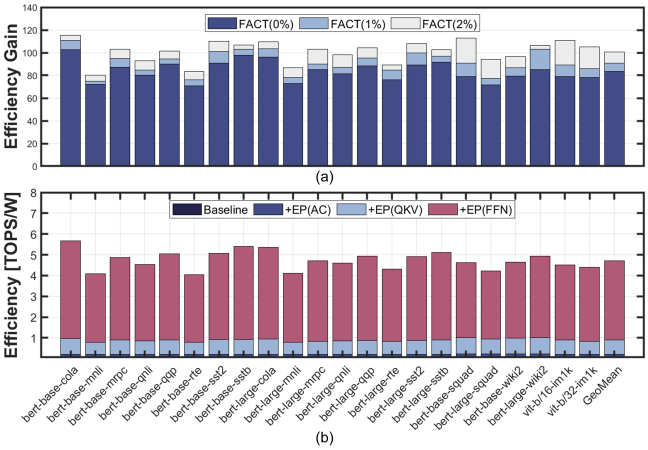

Figure 15: (a) Energy efficiency gain of FACT over V100 GPU. (b) Energy reduction breakdown of different modules with FACT.

overhead. This is because we apply only add operation for prediction, which avoids hardware costs. When working at 500MHz, the power is only 337mW. In comparison, the mobile SOC Snapdragon 8 Gen 1 consumes 11.2W at peak performance, which means only 3% energy overhead is required to embed FACT to mobile SOC to enable efficient Transformer inference.

Figure 15 (a) shows the end-to-end energy efficiency improvement of FACT over V100 GPU on various NLP and CV tasks. On geomean average, FACT achieves 83.4×, 90.7× and 100.5× higher energy efficiency compared to V100 GPU (FP16) with 0%/1%/2% accuracy loss. Figure 15 (b) shows the energy saving breakdown of FACT accelerator from QKV generation, attention computation, and FFN. FACT is able to reduce attention by over 90%, and it introduces 3.63× efficiency improvement from the aspect of the whole Transformer. This is because the computation amount of attention is not dominating. FACT further improves the efficiency by 4.05× by skipping unnecessary QKV generation based on EP results. Additonally, FACT accelerates the FFN process by turning the computation related to trivial tokens into INT4, where the processing unit is designed to handle 4 INT4 MACs in one INT8 MAC unit simultaneously. This mixed-precision improves the efficiency by 5.65× since FFN usually accounts for the most computation.

5.5 Comparison with Other Accelerators

ELSA [18] and Sanger [37] are state-of-the-art Transformer accelerators. They find effective attention computation by orthogonal projection of Q and K (ELSA) or low precision $Q \times K^T$ (Sanger). We compare their performance with our FACT accelerator. The fundamental difference between FACT and the two works is FACT can accelerate all three computation modules in Transformer, but ELSA and Sanger do not optimize the QKV and FFN computation and requires external processor to handle these. Table III compares the characteristics and metrices of these three processors. The energy efficiency of FACT (processing all QKV, attention and FFN, averaged over 22 benchmarks) is 4388 GOPS/W, which is 3.91× and 22.86× greater than ELSA and Sanger (only for attention computation). Additionally, the energy efficiency of FACT in attention computation can reach 15.2 TOPS/W with k=0.05, which is 9.5× and 40.32× higher than the counterparts (with technology and frequency normalized).

Table 3: Summary and Comparison of FACT and SOTA Works

	ELSA	Sanger	**This Work**
Acceleration for	Att. only	Att. only	**All**
Technology[nm]	40nm	55nm	**28nm**
Frequency[Hz]	1G	500M	**500M**
Area[mm^2]	1.26	16.9	**6.03**
Throughput[GOPS]	1090	529	**928**
Energy Effi.[GOPS/W]	1120	192	**4388**

6 RELATED WORKS AND DISCUSSION

Efficient Transformer Accelerator. Since the introduction of the Transformer model, its extremely high performance and great computational effort have attracted widespread attention. There have been many works [12, 13, 17, 18, 22, 32, 37, 42, 63, 67, 68] proposing methods to improve the energy efficiency and speed of Transformer inference. However, most of these works focus on the attention computation of the Transformer. This is because attention computation has a quadratic complexity with the input and it becomes a bottleneck when dealing with very long inputs. E.g., [42] accelerates Transformer tasks from Long-Range-Arena[59] with over 1k token length. However, for the vast majority of applications of the Transformer, it is the QKV generation and FFN, which these works ignore, that dominate power and time. In this case, FACT outperforms the previous works greatly. For cases with extremely long token sequences when attention dominates the computation, FACT still works with EP, which effectively reduces the attention computation by detecting the essential correlations.

Neural network accelerator with sparsity and quantization support. There are very many ASIC or FPGA accelerators [2, 8, 13–15, 19–21, 23, 28, 29, 31, 32, 36, 38, 41, 49, 50, 52, 56, 61, 66] that utilize sparsity in order to optimize the performance of neural network (NN) inference. They are able to support sparse convolutional or linear computation to improve speed and energy efficiency. Most of the works focus on how to exploit pre-trained static sparse weights. Unlike these works, FACT's EP is an output dynamic sparsity method. Further, EP prediction is a unique cross-stage method for Transformer because only the relative value of $Q \times K^T$ can indicate the importance of a value, making the traditional magnitude-based sparsity methods not applicable. Similarly, there has been a lot of work focusing on how quantization [1, 55, 68, 69, 72] can be used to improve the inference performance of NNs. However, few notices that it is feasible to adjust the quantization precision of input activation dynamically throughout the inference, based on each tensor's contextual significance. FACT assigns different quantization precisions to tensors of different importance through the attention mechanism.

Extensiveness of FACT and Eager Prediction The design concept of FACT and EP is via predicting the redundancy before computation, thus skipping unnecessary computation. It is possible to use EP to predict the negative results of convolutional outputs, which would become zero after ReLU function in CNN. We can thus use FACT to accelerate CNN by regarding sparse convolution operations as sparse FFN. Besides, embedding EP units with traditional sparse CNN accelerators such as [24] and [25] works as well.

FACT is also compatible with newly-proposed Transformer models. This is because FACT accelerates all three computation blocks. It achieves considerable acceleration regardless of the token lengths. Further, to our knowledge, it is the weight dimension and depth of the model, rather than token length, that more advanced model focuses on, such as the ViT (22B) [7] and GPT-3 [4]. It further emphasizes the necessity for QKV and FFN optimization, which can be effectively handled by FACT. Besides, FACT supports more mixed precision to cater to various precision requirements with decent modifications. For example, to support INT4/8/16, it only requires reconfiguring the PE, and adjusting the SRAM storage pattern from group-in-2 to group-in-4.

7 CONCLUSION

We propose FACT, an algorithm-hardware co-design to comprehensively accelerate Transformer inference by optimizing its QKV generation, attention computation, and FFN layers. We first propose EP mechanism, which prunes the unnecessary computation early before QKV generation of a transformer layer. It also finds out trivial tokens whose precision can be reduced in FFN. Then, we propose an efficient hardware to support inference with EP with an out-of-order scheduler and diagonal storage pattern. FACT achieves considerable speed up and energy savings over traditional hardware platforms and SOTA accelerators. On average, it is 94.88×, 3.91×, and 22.86× more energy saving than Nvidia V100 GPU, ELSA, and Sanger.

ACKNOWLEDGMENTS

This work was supported in part by NSFC Grant 62125403; in part by the Science and Technology Innovation 2030 - New Generation of AI Project under Grant 2022ZD0115201; in part by Beijing Municipal Science and Technology Project Grant Z221100007722023; in part by the National Key Research and Development Program under Grant 2021ZD0114400; in part by Beijing National Research Center for Information Science and Technology; and in part by the Beijing Advanced Innovation Center for Integrated Circuits.

REFERENCES

[1] Renzo Andri, Beatrice Bussolino, Antonio Cipolletta, Lukas Cavigelli, and Zhe Wang. 2022. Going Further With Winograd Convolutions: Tap-Wise Quantization for Efficient Inference on 4x4 Tiles. In *55th IEEE/ACM International Symposium on Microarchitecture, MICRO 2022, Chicago, IL, USA, October 1-5, 2022*. IEEE, 582–598.

[2] Bahar Asgari, Ramyad Hadidi, Tushar Krishna, Hyesoon Kim, and Sudhakar Yalamanchili. 2020. ALRESCHA: A Lightweight Reconfigurable Sparse-Computation Accelerator. In *IEEE International Symposium on High Performance Computer Architecture, HPCA 2020, San Diego, CA, USA, February 22-26, 2020*. IEEE, 249–260.

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *CoRR* abs/2004.05150 (2020). arXiv:2004.05150

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-End Object Detection with Transformers. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12346)*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer, 213–229.

[6] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. 2021. Transformers4Rec: Bridging the Gap between NLP and Sequential / Session-Based Recommendation. In *RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021*, Humberto Jesús Corona Pampín, Martha A. Larson, Martijn C. Willemsen, Joseph A. Konstan, Julian J. McAuley, Jean Garcia-Gathright, Bouke Huurnink, and Even Oldridge (Eds.). ACM, 143–153.

[7] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd van Steenkiste, Gamaleldin F. Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Patrick Collier, Alexey Gritsenko, Vighnesh Birodkar, Cristina Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetić, Dustin Tran, Thomas Kipf, Mario Lučić, Xiaohua Zhai, Daniel Keysers, Jeremiah Harmsen, and Neil Houlsby. 2023. Scaling Vision Transformers to 22 Billion Parameters. https://doi.org/10.48550/ARXIV.2302.05442

[8] Chunhua Deng, Yang Sui, Siyu Liao, Xuehai Qian, and Bo Yuan. 2021. GoSPA: An Energy-efficient High-performance Globally Optimized SParse Convolutional Neural Network Accelerator. In *48th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2021, Valencia, Spain, June 14-18, 2021*. IEEE, 1110–1123.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[12] Hongxiang Fan, Thomas Chau, Stylianos I. Venieris, Royson Lee, Alexandros Kouris, Wayne Luk, Nicholas D. Lane, and Mohamed S. Abdelfattah. 2022. Adaptable Butterfly Accelerator for Attention-based NNs via Hardware and Algorithm Co-design. In *55th IEEE/ACM International Symposium on Microarchitecture, MICRO 2022, Chicago, IL, USA, October 1-5, 2022*. IEEE, 599–615.

[13] Chao Fang, Aojun Zhou, and Zhongfeng Wang. 2022. An Algorithm–Hardware Co-Optimized Framework for Accelerating N:M Sparse Transformers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 30, 11 (2022), 1573–1586. https://doi.org/10.1109/TVLSI.2022.3197282

[14] Ashish Gondimalla, Noah Chesnut, Mithuna Thottethodi, and T. N. Vijaykumar. 2019. SparTen: A Sparse Tensor Accelerator for Convolutional Neural Networks. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2019, Columbus, OH, USA, October 12-16, 2019*. ACM, 151–165.

[15] Sumanth Gudaparthi, Sarabjeet Singh, Surya Narayanan, Rajeev Balasubramonian, and Visvesh Sathe. 2022. CANDLES: Channel-Aware Novel Dataflow-Microarchitecture Co-Design for Low Energy Sparse Neural Network Acceleration. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2022, Seoul, South Korea, April 2-6, 2022*. IEEE, 876–891.

[16] Mandy Guo, Joshua Ainslie, David C. Uthus, Santiago Ontañón, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. LongT5: Efficient Text-To-Text Transformer for Long Sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz (Eds.). Association for Computational Linguistics, 724–736.

[17] Tae Jun Ham, Sungjun Jung, Seonghak Kim, Young H. Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, Jae W. Lee, and Deog-Kyoon Jeong. 2020. A^3: Accelerating Attention Mechanisms in Neural Networks with Approximation. In *IEEE International Symposium on High Performance Computer Architecture, HPCA 2020, San Diego, CA, USA, February 22-26, 2020*. IEEE, 328–341. https://doi.org/10.1109/HPCA47549.2020.00035

[18] Tae Jun Ham, Yejin Lee, Seong Hoon Seo, Soosung Kim, Hyunji Choi, Sung Jun Jung, and Jae W. Lee. 2021. ELSA: Hardware-Software Co-Design for Efficient, Lightweight Self-Attention Mechanism in Neural Networks. In *Proceedings of the 48th Annual International Symposium on Computer Architecture* (Virtual Event, Spain) *(ISCA '21)*. IEEE Press, 692–705.

[19] Edward Hanson, Shiyu Li, Hai Helen Li, and Yiran Chen. 2022. Cascading structured pruning: enabling high data reuse for sparse DNN accelerators. In *ISCA '22: The 49th Annual International Symposium on Computer Architecture, New York, New York, USA, June 18 - 22, 2022*, Valentina Salapura, Mohamed Zahran, Fred Chong, and Lingjia Tang (Eds.). ACM, 522–535.

[20] Kartik Hegde, Hadi Asghari Moghaddam, Michael Pellauer, Neal Clayton Crago, Aamer Jaleel, Edgar Solomonik, Joel S. Emer, and Christopher W. Fletcher. 2019. ExTensor: An Accelerator for Sparse Tensor Algebra. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2019, Columbus, OH, USA, October 12-16, 2019*. ACM, 319–333.

[21] Reza Hojabr, Ali Sedaghati, Amirali Sharifian, Ahmad Khonsari, and Arrvindh Shriraman. 2021. SPAGHETTI: Streaming Accelerators for Highly Sparse GEMM on FPGAs. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2021, Seoul, South Korea, February 27 - March 3, 2021*. IEEE, 84–96.

[22] Seongmin Hong, Seungjae Moon, Junsoo Kim, Sungjae Lee, Minsub Kim, Dongsoo Lee, and Joo-Young Kim. 2022. DFX: A Low-latency Multi-FPGA Appliance for Accelerating Transformer-based Text Generation. In *55th IEEE/ACM International Symposium on Microarchitecture, MICRO 2022, Chicago, IL, USA, October 1-5, 2022*. IEEE, 616–630.

[23] Konstantinos Kanellopoulos, Nandita Vijaykumar, Christina Giannoula, Roknoddin Azizi, Skanda Koppula, Nika Mansouri-Ghiasi, Taha Shahroodi, Juan Gómez-Luna, and Onur Mutlu. 2019. SMASH: Co-designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2019, Columbus, OH, USA, October 12-16, 2019*. ACM, 600–614.

[24] Sanghoon Kang, Donghyeon Han, Juhyoung Lee, Dongseok Im, Sangyeob Kim, Soyeon Kim, Junha Ryu, and Hoi-Jun Yoo. 2021. GANPU: An Energy-Efficient Multi-DNN Training Processor for GANs With Speculative Dual-Sparsity Exploitation. *IEEE Journal of Solid-State Circuits* 56, 9 (2021), 2845–2857. https://doi.org/10.1109/JSSC.2021.3066572

[25] Sangyeob Kim, Juhyoung Lee, Sanghoon Kang, Jinmook Lee, Wooyoung Jo, and Hoi-Jun Yoo. 2021. PNPU: An Energy-Efficient Deep-Neural-Network Learning Processor With Stochastic Coarse–Fine Level Weight Pruning and Adaptive Input/Output/Weight Zero Skipping. *IEEE Solid-State Circuits Letters* 4 (2021), 22–25. https://doi.org/10.1109/LSSC.2020.3041497

[26] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. 2019. Large Scale Learning of General Visual Representations for Transfer. *CoRR* abs/1912.11370 (2019). arXiv:1912.11370

[27] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[28] Jonathan S. Lew, Yunpeng Liu, Wenyi Gong, Negar Goli, R. David Evans, and Tor M. Aamodt. 2022. Anticipating and eliminating redundant computations in accelerated sparse training. In *ISCA '22: The 49th Annual International Symposium on Computer Architecture, New York, New York, USA, June 18 - 22, 2022*, Valentina Salapura, Mohamed Zahran, Fred Chong, and Lingjia Tang (Eds.). ACM, 536–551.

[29] Gang Li, Weixiang Xu, Zhuoran Song, Naifeng Jing, Jian Cheng, and Xiaoyao Liang. 2022. Ristretto: An Atomized Processing Architecture for Sparsity-Condensed Stream Flow in CNN. In *55th IEEE/ACM International Symposium on Microarchitecture, MICRO 2022, Chicago, IL, USA, October 1-5, 2022*. IEEE, 1434–1450.

[30] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. 2022. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). PMLR, 12888–12900.

[31] Shiyu Li, Edward Hanson, Xuehai Qian, Hai (Helen) Li, and Yiran Chen. 2021. ESCALATE: Boosting the Efficiency of Sparse CNN Accelerator with Kernel Decomposition. In *MICRO '21: 54th Annual IEEE/ACM International Symposium on Microarchitecture, Virtual Event, Greece, October 18-22, 2021*. ACM, 992–1004.

[32] Zheng Li, Soroush Ghodrati, Amir Yazdanbakhsh, Hadi Esmaeilzadeh, and Mingu Kang. 2022. Accelerating attention through gradient-based learned runtime pruning. In *ISCA '22: The 49th Annual International Symposium on Computer Architecture, New York, New York, USA, June 18 - 22, 2022*, Valentina Salapura, Mohamed Zahran, Fred Chong, and Lingjia Tang (Eds.). ACM, 902–915.

[33] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692

[34] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. 2022. Swin Transformer V2: Scaling Up Capacity and Resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 11999–12009. https://doi.org/10.1109/CVPR52688.2022.01170

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 9992–10002. https://doi.org/10.1109/ICCV48922.2021.00986

[36] Zhi Gang Liu, Paul N. Whatmough, Yuhao Zhu, and Matthew Mattina. 2022. S2TA: Exploiting Structured Sparsity for Energy-Efficient Mobile CNN Acceleration. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2022, Seoul, South Korea, April 2-6, 2022*. IEEE, 573–586.

[37] Liqiang Lu, Yicheng Jin, Hangrui Bi, Zizhang Luo, Peng Li, Tao Wang, and Yun Liang. 2021. Sanger: A Co-Design Framework for Enabling Sparse Attention Using Reconfigurable Architecture. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture* (Virtual Event, Greece) *(MICRO '21)*. Association for Computing Machinery, New York, NY, USA, 977–991.

[38] Mostafa Mahmoud, Isak Edo, Ali Hadi Zadeh, Omar Mohamed Awad, Gennady Pekhimenko, Jorge Albericio, and Andreas Moshovos. 2020. TensorDash: Exploiting Sparsity to Accelerate Deep Neural Network Training. In *53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2020, Athens, Greece, October 17-21, 2020*. IEEE, 781–795.

[39] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[40] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

[41] Julian Pavon, Iván Vargas Valdivieso, Adrián Barredo, Joan Marimon, Miquel Moretó, Francesc Moll, Osman S. Unsal, Mateo Valero, and Adrián Cristal. 2021. VIA: A Smart Scratchpad for Vector Units with Application to Sparse Matrix Computations. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2021, Seoul, South Korea, February 27 - March 3, 2021*. IEEE, 921–934.

[42] Zheng Qu, Liu Liu, Fengbin Tu, Zhaodong Chen, Yufei Ding, and Yuan Xie. 2022. 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* (Lausanne, Switzerland) *(ASPLOS '22)*. Association for Computing Machinery, New York, NY, USA, 14–26.

[43] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8748–8763.

[44] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2019. Improving Language Understanding by Generative Pre-Training. (2019).

[45] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[46] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67.

[47] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, Jian Su, Xavier Carreras, and Kevin Duh (Eds.). The Association for Computational Linguistics, 2383–2392.

[48] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 10674–10685.

[49] Alexander Rucker, Matthew Vilim, Tian Zhao, Yaqi Zhang, Raghu Prabhakar, and Kunle Olukotun. 2021. Capstan: A Vector RDA for Sparsity. In *MICRO '21: 54th Annual IEEE/ACM International Symposium on Microarchitecture, Virtual Event, Greece, October 18-22, 2021*. ACM, 1022–1035.

[50] Fazle Sadi, Joe Sweeney, Tze Meng Low, James C. Hoe, Larry T. Pileggi, and Franz Franchetti. 2019. Efficient SpMV Operation for Large and Highly Sparse Matrices

using Scalable Multi-way Merge Parallelization. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2019, Columbus, OH, USA, October 12-16, 2019*. ACM, 347–358.

[51] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019). arXiv:1910.01108

[52] Jong Hoon Shin, Ali Shafiee, Ardavan Pedram, Hamzah Abdel-Aziz, Ling Li, and Joseph Hassoun. 2022. Griffin: Rethinking Sparse Optimization for Deep Learning Architectures. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2022, Seoul, South Korea, April 2-6, 2022*. IEEE, 861–875.

[53] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *CoRR* abs/1909.08053 (2019). arXiv:1909.08053

[54] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zheng, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. 2022. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model. *CoRR* abs/2201.11990 (2022). arXiv:2201.11990

[55] Zhuoran Song, Bangqi Fu, Feiyang Wu, Zhaoming Jiang, Li Jiang, Naifeng Jing, and Xiaoyao Liang. 2020. DRQ: Dynamic Region-based Quantization for Deep Neural Network Acceleration. In *47th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2020, Valencia, Spain, May 30 - June 3, 2020*. IEEE, 1010–1021.

[56] Nitish Kumar Srivastava, Hanchen Jin, Jie Liu, David H. Albonesi, and Zhiru Zhang. 2020. MatRaptor: A Sparse-Sparse Matrix Multiplication Accelerator Based on Row-Wise Product. In *53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2020, Athens, Greece, October 17-21, 2020*. IEEE, 766–780.

[57] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (Eds.). ACM, 1441–1450.

[58] Thierry Tambe, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul N. Whatmough, Alexander M. Rush, David Brooks, and Gu-Yeon Wei. 2021. EdgeBERT: Sentence-Level Energy Optimizations for Latency-Aware Multi-Task NLP Inference. In *MICRO '21: 54th Annual IEEE/ACM International Symposium on Microarchitecture, Virtual Event, Greece, October 18-22, 2021*. ACM, 830–844.

[59] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long Range Arena : A Benchmark for Efficient Transformers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008.

[61] Sumit Walia, Bachu Varun Tej, Arpita Kabra, Joydeep Devnath, and Joycee Mekie. 2022. Fast and Low-Power Quantized Fixed Posit High-Accuracy DNN Implementation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 30, 1 (2022), 108–111. https://doi.org/10.1109/TVLSI.2021.3131609

[62] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[63] Hanrui Wang, Zhekai Zhang, and Song Han. 2021. SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2021, Seoul, South Korea, February 27 - March 3, 2021*. IEEE, 97–110.

[64] Yang Wang, Yubin Qin, Dazheng Deng, Jingchuan Wei, Yang Zhou, Yuanqi Fan, Tianbao Chen, Hao Sun, Leibo Liu, Shaojun Wei, and Shouyi Yin. 2022. An Energy-Efficient Transformer Processor Exploiting Dynamic Weak Relevances in Global Attention. *IEEE Journal of Solid-State Circuits* (2022), 1–16. https://doi.org/10.1109/JSSC.2022.3213521

[65] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45.

[66] Dingqing Yang, Amin Ghasemazar, Xiaowei Ren, Maximilian Golub, Guy Lemieux, and Mieszko Lis. 2020. Procrustes: a Dataflow and Accelerator for Sparse Deep Neural Network Training. In *53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2020, Athens, Greece, October 17-21, 2020*. IEEE, 711–724.

[67] Amir Yazdanbakhsh, Ashkan Moradifirouzabadi, Zheng Li, and Mingu Kang. 2022. Sparse Attention Acceleration with Synergistic In-Memory Pruning and On-Chip Recomputation. In *55th IEEE/ACM International Symposium on Microarchitecture, MICRO 2022, Chicago, IL, USA, October 1-5, 2022*. IEEE, 744–762.

[68] Ali Hadi Zadeh, Isak Edo, Omar Mohamed Awad, and Andreas Moshovos. 2020. GOBO: Quantizing Attention-Based NLP Models for Low Latency and Energy Efficient Inference. In *53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2020, Athens, Greece, October 17-21, 2020*. IEEE, 811–824.

[69] Ali Hadi Zadeh, Mostafa Mahmoud, Ameer Abdelhadi, and Andreas Moshovos. 2022. Mokey: enabling narrow fixed-point inference for out-of-the-box floating-point transformer models. In *ISCA '22: The 49th Annual International Symposium on Computer Architecture, New York, New York, USA, June 18 - 22, 2022*, Valentina Salapura, Mohamed Zahran, Fred Chong, and Lingjia Tang (Eds.). ACM, 888–901.

[70] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

[71] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. 2022. Scaling Vision Transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 1204–1213. https://doi.org/10.1109/CVPR52688.2022.01179

[72] Yongwei Zhao, Chang Liu, Zidong Du, Qi Guo, Xing Hu, Yimin Zhuang, Zhenxing Zhang, Xinkai Song, Wei Li, Xishan Zhang, Ling Li, Zhiwei Xu, and Tianshi Chen. 2021. Cambricon-Q: A Hybrid Architecture for Efficient Training. In *48th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2021, Valencia, Spain, June 14-18, 2021*. IEEE, 706–719.

[73] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2021. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.