DOTA: Detect and Omit Weak Attentions for Scalable Transformer Acceleration

Zheng Qu*
zhengqu@ucsb.edu
UC Santa Barbara
United States

Liu Liu*
liu_liu@ucsb.edu
UC Santa Barbara
United States

Fengbin Tu
fengbintu@ucsb.edu
UC Santa Barbara
United States

Zhaodong Chen
chenzd15thu@ucsb.edu
UC Santa Barbara
United States

Yufei Ding
yufeiding@cs.ucsb.edu
UC Santa Barbara
United States

Yuan Xie
yuanxie@ece.ucsb.edu
UC Santa Barbara
United States

ABSTRACT

Transformer Neural Networks have demonstrated leading performance in many applications spanning over language understanding, image processing, and generative modeling. Despite the impressive performance, long-sequence Transformer processing is expensive due to quadratic computation complexity and memory consumption of self-attention. In this paper, we present DOTA, an algorithm-architecture co-design that effectively addresses the challenges of scalable Transformer inference. Based on the insight that not all connections in an attention graph are equally important, we propose to jointly optimize a lightweight Detector with the Transformer model to accurately detect and omit weak connections during runtime. Furthermore, we design a specialized system architecture for end-to-end Transformer acceleration using the proposed attention detection mechanism. Experiments on a wide range of benchmarks demonstrate the superior performance of DOTA over other solutions. In summary, DOTA achieves 152.6× and 4.5× performance speedup and orders of magnitude energy-efficiency improvements over GPU and customized hardware, respectively.

CCS CONCEPTS

• **Computer systems organization** → **Neural networks**; • **Computing methodologies** → **Machine learning approaches**.

KEYWORDS

Transformer Acceleration, SW-HW Co-design, Sparse Architecture

ACM Reference Format:
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 (ASPLOS '22), February 28 – March 4, 2022, Lausanne, Switzerland*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3503222.3507738

*Both authors contributed equally to this research.

ASPLOS '22, February 28 – March 4, 2022, Lausanne, Switzerland

ACM ISBN 978-1-4503-9205-1/22/02.
https://doi.org/10.1145/3503222.3507738

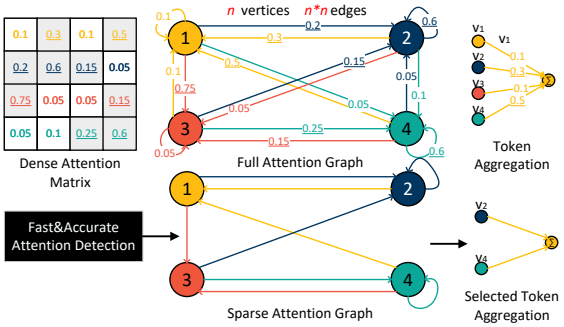


Figure 1: The graph perspective of self-attention.

1 INTRODUCTION

In recent years, Transformer Neural Networks have drawn a surge of interests from the deep learning community. Lots of Transformer models have been proposed and demonstrated superior performance over traditional Deep Neural Networks (DNNs) [12, 52]. The use of Transformers has spanned over a wide range of application domains including language understanding [4, 41, 43], image processing [5, 11, 38], and generative modeling [7, 42, 46].

The key to the stunning performance of Transformers is the self-attention [52] mechanism. Self-attention relates different positions of a sequence by evaluating the pair-wise importance between the input tokens. This process can be understood as creating and aggregating over an *attention graph* to compute the output feature of each token. As shown in Figure 1, the dense 4×4 attention matrix is interpreted as the Laplacian representation of a graph with each vertex corresponding to one token. Therefore, the attention graph is directed and complete, where the edge weights are assigned by the attention weights, i.e., SoftMax probabilities. The output of self-attention is one-step aggregation over the attention graph. Each vertex collects features from its incoming vertices and performs weighted sum to update its own feature vector.

Despite the significant representational power on sequence modeling, standard self-attention mechanism cannot scale with long sequences. As shown in Figure 1, using full attention graphs unavoidably incurs quadratic complexity in both computation and memory consumption with regard to sequence length. Thus, deploying Transformers for long-sequence modeling tasks is challenging for current hardware platforms.

We observe that a significant portion of edges in attention graphs are weak connections with negligible contributions to the output, as illustrated in Figure 1. In other words, only a small portion of connections are important to deliver long-range representations. Therefore, instead of creating a full attention graph, we propose to detect and omit weak connections and skip the unnecessary computations. Furthermore, after attention detection, feature aggregation only need to be performed over the selected strong connections.

However, weak connection detection is difficult as inaccurately omitting strong connections will not only disturb the immediate attention output but also cascade to subsequent Transformer layers, degrading the overall model performance. Unlike graph pruning or weight sparsity commonly used in traditional DNNs, each attention graph need to identify its unique strong connections at runtime depending on the inputs. Therefore, the cost of the detection mechanism is critical to deliver overall performance speedup and energy saving. Prior work fails to deliver a satisfying solution for weak attention detection, as they either suffer from poor detection quality [18], or from hardware inefficiency [17, 22, 27, 48, 50].

To tackle the problem, we introduce DOTA, an algorithm-architecture co-design that reduces the cost of self-attention and boosts the performance and efficiency of long-sequence Transformer inference. We first design and train a lightweight Detector along with the Transformer model. The training process is formulated as a joint optimization problem to minimize both attention detection loss and model loss. We adopt low-rank transformation and low-precision computation to reduce the overhead of attention detection, and we rely on the joint optimization process to ensure the attention detection quality.

Furthermore, we explore architecture support to translate the theoretical savings to real performance speedup and energy reduction. We address three system-level challenges through our architecture design. Firstly, to support large Transformer models with various configurations, we need to effectively disassemble the algorithm and identify the essential components. Prior work designs accelerators for specific component like self-attention block [17, 18]. Instead, DOTA provides an efficient abstraction of the model and presents a unified architecture to support all components, achieving better area- and energy-efficiency. Besides, we further analyze different levels of parallelism on top of the proposed abstraction and present a scalable system architecture (Section 4.1). Secondly, low-precision computation is essential to the cost of the attention detection mechanism. To support multi-precision computations, we design a Reconfigurable Matrix Multiplication Unit (RMMU) that can be dynamically orchestrated to satisfy the throughput requirements of different computation precision (Section 4.2). Finally, when computing the attention output with the sparse attention graph, DOTA outperforms prior work by adopting the Token-Parallel dataflow with software-enabled workload balancing and hardware-enabled out-of-order execution. These techniques can further improve system performance and energy-efficiency.

In summary, our work makes the following contributions:

- We propose to detect sparse attention graphs to compute for self-attention, which can significantly reduce both computation complexity and memory consumption.
- We present a trainable Detector to effectively select important attention connections. The proposed method achieves both hardware efficiency and detection accuracy, yielding an adequate trade-off between computation savings and model quality.
- We design DOTA, a scalable inference system that addresses three hardware challenges of executing long-sequence Transformer models with attention detection.
- DOTA improves the performance and energy-efficiency of Transformer inference. On average, DOTA achieves 152.6× and 4.5× performance speedup and orders of magnitude energy-efficiency improvements over GPU and state-of-the-art customized accelerator, respectively.

2 BACKGROUND AND MOTIVATION

Firstly, we introduce the preliminaries of Transformer model architecture and the challenges of serving long sequences. Secondly, we find that weak connections exist in attention graph and can be omitted without hurting performance. Finally, we present the opportunity to detect weak connections for more computation savings and the need for architecture support.

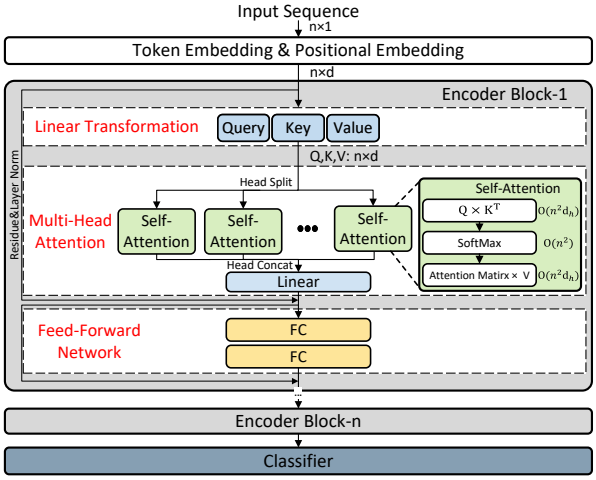


Figure 2: Transformer model architecture.

2.1 Preliminaries of Transformer

A typical Transformer model is composed of stacked encoder (decoder) blocks as shown in Figure 2. At the beginning, the input sentence with n tokens is first transformed into an embedding matrix $X \in \mathbb{R}^{n \times d}$. Then, the input embedding matrix is processed by blocks of encoders. We split each encoder into three stages, namely Linear Transformation, Multi-Head Attention, and Feed-Forward Network (FFN). In the transformation stage, we multiply the input with three weight matrices to obtain Query (Q), Key (K), and Value (V) as

$$Q, K, V = XW_Q, XW_K, XW_V \tag{1}$$

After linear transformation, the attention weights $A \in \mathbb{R}^{n \times n}$ is defined as

$$A = SoftMax(\frac{QK^T}{\sqrt{d_k}}) \tag{2}$$

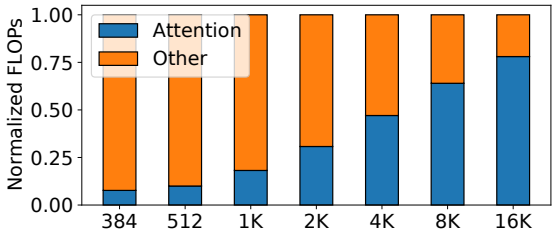


Figure 3: Breakdown of attention operations vs. other operations when scaling sequence length.

where SoftMax(·) is computed row-wise. Finally, the output values are generated by multiplying attention weights A with the projected values V as

$$Z = AV. \tag{3}$$

The output of the Multi-Head Attention is added with the encoder's input through a residue connection, and a layer normalization is applied afterwards. Finally, a Feed-Forward Network (FFN) containing two fully-connected (FC) layers, followed by another residual connection and layer normalization is applied to generate the output of the encoder. As presented in Figure 2, the same encoder structure is repeated and stacked for multiple times in a single Transformer. Usually, a classifier is added at the end to make predictions.

2.2 Weak Connections in Attention

Transformer-based models equipped with the self-attention mechanism are promising in a wide range of applications that need long sequence modeling capabilities. However, the quadratic computational complexity of self-attention, w.r.t. sequence length, hinders the deployment of Transformers. Hence, we are in need of scalable acceleration for Transformers, especially for emerging sequence modeling workloads.

To better illustrate the scaling challenge of Transformers, we show the comparison of self-attention, as in Eq. 2 and Eq. 3, vs. linear transformations as in Eq. 1 and FFN in terms of floating-point operations (FLOPs). One characteristic of self-attention operations is that these are parameter-free general matrix-matrix product (GEMM), other than GEMM routines as in linear transformations and FFN, where one matrix is parameterized and commonly referred as the weight matrix. As shown in Figure 3, the parameter-free attention GEMM operations become the bottleneck when scaling sequence length.

The root cause of the quadratic complexity in self-attention is the fully-connected attention graph, where all pairs of sources and targets are computed. With that, we post the hypothesis that not all connections in attention graphs are equally important and contribute significantly to attention outputs. In other words, there exist extremely sparse attention graphs with weak connections omitted that can achieve performance on par with full attention graphs.

To test our hypothesis, we experiment on a pre-trained Transformer model. Specifically, we regard small values in attention graph (Eq. 2) as weak connections and remove these values while keep the rest of the model intact. The remaining important connections are determined by row-wise top-k search after obtaining full attention graphs. Retention ratio is used to indicate the portion of preserved connections. We use BERT-large as the baseline model, and evaluate it on the SQuAD v1.0 dataset with a total number of 10,570 samples. Table 1 gives the performance results in terms of F1 accuracy. As shown by the table, we can omit almost 90% of the connections in the attention graphs with negligible performance degradation.

Table 1: Transformer evaluation accuracy results, i.e., SQuAD F1 scores, when omitting different portion of attentions.

Retention	full	20%	15%	10%	5%
F1 Score	91.4	91.4	91.3	91.1	90.2

2.3 Detect and Omit Weak Connections

Although omitting attention connections are effective, we still need to compute all the attention scores and the follow-up SoftMax normalization to obtain the attention weights for omission. Putting it in another way, the parameter-free multiplications and the SoftMax operations in Eq. 2 that result in weak connections in attention graph A are wasted. Hence, we propose an efficient method to first detect weak connections prior to the computations in Eq. 2, such that we can save computations in the attention bottleneck. The underlying principle is that the weak connections with small values in attention weights A will also have small values in raw attention scores $S = QK^T$ correspondingly.

The detection mechanism needs to be both efficient and accurate. On the one hand, approximation methods such as angular distance approximation [23] can be efficiently implemented in hardware but cannot provide accurate detection of weak connections, especially when scaling sequence length. On the other hand, inefficient detection could offset the computational saving. At last, although omitting weak connections is straightforward, it can cause the workload imbalance issue and irregular data accessing. Our DOTA features an algorithm-architecture co-design to support efficient and accurate detection for weak attention connections and provides architecture support for computation saving from omission.

3 WEAK ATTENTIONS DETECTION

Given the challenge of efficient and accurate detection of weak attentions, we propose a learning-based method that can effectively detect the relative importance of connections in attention graphs. As shown in Figure 4, our detection method uses low-rank linear transformations to estimate attention scores. Then, from estimated scores, we can use top-k selection to generate bit-masks with zeros indicating the weak attentions. The low-rank transformations of query and key are from the optimization of estimation loss. We further propose model adaptation to improve performance from degradation via jointly optimizing model loss and estimation loss with weak connection omission enabled.

3.1 Low-Rank Linear Transformation

We introduce a pair of low-rank linear transformations for query and key as in

$$\tilde{Q}, \tilde{K} = XP\tilde{W_Q}, XP\tilde{W_K} \tag{4}$$

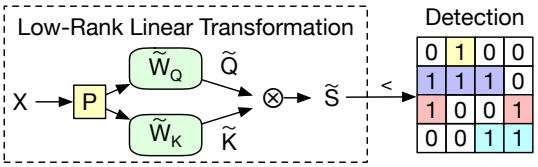


Figure 4: Weak attention detection from estimated attention scores computed by low-rank linear transformations.

, where $P \in \sqrt{\frac{3}{k}} \cdot \{-1, 0, 1\}^{d \times k}$ is a sparse random projection matrix [1] to reduce the dimensions of input feature X. Therefore, $\tilde{W_Q}$ and $\tilde{W_K}$ both contain $k \times k$ parameters, where k is much smaller than d. Since the estimated attention scores, as in $\tilde{S} = \tilde{Q}\tilde{K}^T$, are only used to select weak attentions based on relative importance, the low-rank transformations can afford low-precision computations, such as INT4 fixed-point arithmetic. The quality of low-rank transformation is determined by reduced rank k and the compute precision.

We use the mean squared error (MSE) as the estimation loss to optimize low-rank transformation parameters, as in

$$L_{MSE} = \frac{1}{B}||S - \tilde{S}||_2^2 = \frac{1}{B}||QK^T - \tilde{Q}\tilde{K}^T||_2^2 \quad (5)$$

, where B is the mini-batch size. Here we omit the scaling factor for simplicity.

With estimated scores $\tilde{S} = \tilde{Q}\tilde{K}^T$, we can select the important connections by comparing the scores with thresholds. An attention connection can only be preserved if the score is larger than the threshold. The threshold value can be determined by top-k searching or tuning from the validation set.

3.2 Model Adaptation with Joint Optimization

When attention scores are masked out to generate sparse attention graphs, the remaining important attention weights are scaled up as the denominator in SoftMax becomes small. The disturbed attention weights will degrade model quality. As a countermeasure, we propose to fine-tune model parameters with constraints of weak attention omission, referred as model adaptation. With adaptation, the model evaluation accuracy can recover to be on par with full attention baselines, while the computational costs are significantly reduced.

Given a pre-trained model, our method jointly optimize the model parameters and the low-rank transformation parameters by minimizing the loss function as

$$L = L_{Model} + \lambda L_{MSE}. \quad (6)$$

After joint optimization, the low-rank transformation parameters are adjusted to capture the relatively important attention connections. Besides, the original model parameters are also adapted to the sparse attention graph to compute for outputs. Therefore, we are able to recover the model performance after introducing aggressive weak attention omission. During inference, the sparse attention graph can benefit all three sub-layers of the self-attention block, reducing the computational costs of attention weights, softmax function, and attention outputs.

3.3 Intuitive Explanation

Our method estimates attention scores with a low-rank matrix $\tilde{S}$. When training the model with loss function in Eq. 5, the gradient from L_{MSE} will be passed to both the low-rank $\tilde{S}$ and the original attention score S. Intuitively, this loss function not only makes $\tilde{S}$ a better estimation of S, but also makes S easier to be estimated by a low-rank matrix, i.e., by reducing the rank of S. On the other hand, the loss L_{Model} guarantees the rank of S to be high enough to preserve the model accuracy. In other words, the joint optimization of L_{Model} and L_{MSE} implicitly learns a low-rank S with the rank learned depending on the difficulty of the task.

Our design brings two advantages. First, the rank of S will be automatically adjusted to tasks with different difficulty. Hence, our method can potentially achieve higher accuracy on difficult tasks compared with fixed-rank approximation methods. Second, as the rank of $\tilde{S}$ only implicitly influences the rank of S, the final result is less sensitive to the hyper-parameter k.

4 DOTA SYSTEM DESIGN

We present DOTA's hardware system, which is capable of performing scalable Transformer inference by efficiently utilizing the detected attention graph. We specifically address three system-level challenges. First, long-sequence Transformer models involve large GEMM/GEMV computations with configurable hidden dimensions. Therefore, to effectively execute different Transformer models, we need to disassemble the algorithm and identify the essential components. We provide abstraction of the model that helps us to design a scalable and unified architecture for different Transformer layers, achieving good area- and power-efficiency. (Section 4.1). Second, apart from implementing normal precision arithmetics, DOTA also needs to support low-precision computations required by the attention detection. Instead of separately implementing all the arithmetics, a reconfigurable design would be preferred as it can dynamically balance the computation throughput of multi-precision computations. (Section 4.2). Finally, to efficiently compute over the detected attention graph, we should tackle the workload imbalance and irregular memory access caused by attention sparsity (Section 4.3).

4.1 Overall System Architecture

We use Figure 5 to illustrate the overall system architecture of DOTA, and explain how it execute a single encoder block. Running decoders can be considered as a special case of encoder with strict token dependency. As depicted by the figure, DOTA processes one input sequence at a time. Different input sequences share the same weights while requiring duplicated hardware resources to be processed in parallel. Therefore, we can scale-out multiple DOTA accelerators to improve sequence-level parallelism.

For each encoder, we split it into three GEMM stages namely Linear Transformation, Multi-Head attention, and FFN. The GEMM operations in different stages need to be computed sequentially due to data dependency, while each GEMM can be cut into multiple chunks and processed in parallel. Therefore, as shown by Figure 5, we locate 4 compute Lanes in the DOTA accelerator and dedicate each Lane to the computation of one chunk. For example, during Transformation stage, each Lane contains a fraction of

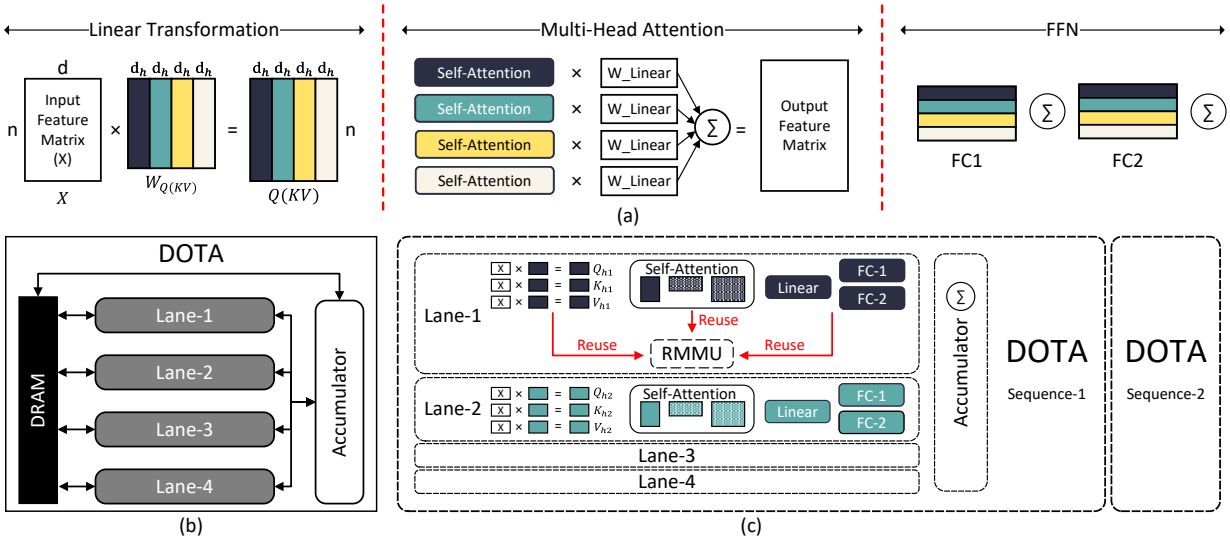


Figure 5: DOTA system design. (a) The abstraction of a single encoder block. We divide each encoder into three sequential stages. Each stage contains multiple GEMM operations that can be further cut into chunks (represented by different colors) and mapped to different compute Lanes. (b) Overall system design of DOTA. Each compute Lane communicates with off-chip DRAM for input feature. The intermediate results are summed up in the Accumulator. (c) Computation mapping between the algorithm and hardware. Each DOTA accelerator processes one input sequence, and each Lane computes for one chunk (color).

weight W_Q, W_K, W_V and generates a chunk of QKV. We make the chunk's size equal to the attention head size h_d. Thus, for Multi-Head Attention, each Lane can directly use the chunks previously generated by itself to compute for self-attention, keeping the data local during execution. Finally, the FC layers in the FFN stage can be orchestrated in a similar way.

As we can see, different compute Lanes share the same input at the beginning of a encoder, whereas the weights and intermediate results are unique to each Lane. Therefore, we avoid data exchanging as well as intermediate matrix split and concatenation among the Lanes. An exception of the above discussion is that, at the end of Multi-Head attention and each FC layers in FFN, we need to accumulate the results generated by each Lane. In DOTA, this is handled by a standalone Accumulator. We locate four Lanes in one DOTA accelerator because 4 is the least common multiple of the attention head numbers across all the benchmarks we evaluated. More Lanes can be implemented for higher chunk-level parallelism.

Inside each Lane, as shown in Figure 6, there is an SRAM buffer, a Reconfigurable Matrix Multiplication Unit (RMMU), a Detector for attention selection, and a Multi-Function Unit for special operations such as Softmax and (De)Quantization. As discussed above, one large RMMU is utilized to execute all different-precision GEMM operations in each stage. Specifically, RMMU first computes low-precision (IN2/4) estimated attention score. The low-precision results are sent to the Detector to be compared with preset threshold values for attention selection. Besides selecting important attentions to be calculated later, the Detector also contains a Scheduler to rearrange the computation order of these important attention values. We incorporate this reordering scheme to achieve balanced computation and efficient memory access (Section 4.3).

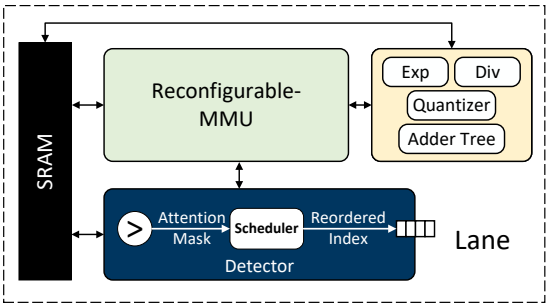


Figure 6: Architecture of each compute Lane.

After obtaining the reordered attention selection results, RMMU starts to compute the attention output under FX16 precision (equation 2, 3). In order to avoid overflow during the computation, we need to dequantize the FX16 computation results of $Q * K$ into floating-point numbers before applying the softmax function. This is done in the Multi-Function Unit, and scaling factors are stored in the global SRAM buffer, which is accessible to the MFU. Thus, the exponent and division are done using floating-point arithmetic. The softmax results are quantized again to keep the consecutive computation ($A * V$) still in fixed-point format.

4.2 Reconfigurable Matrix Multiplication Unit

As presented in Figure 6, each compute Lane contains a Reconfigurable Matrix Multiplication Unit (RMMU) which supports MAC operation in different precision. Low-precision computation occurs during the attention detection. Naively, we can support this feature with separate low-precision arithmetic units, but with the cost of

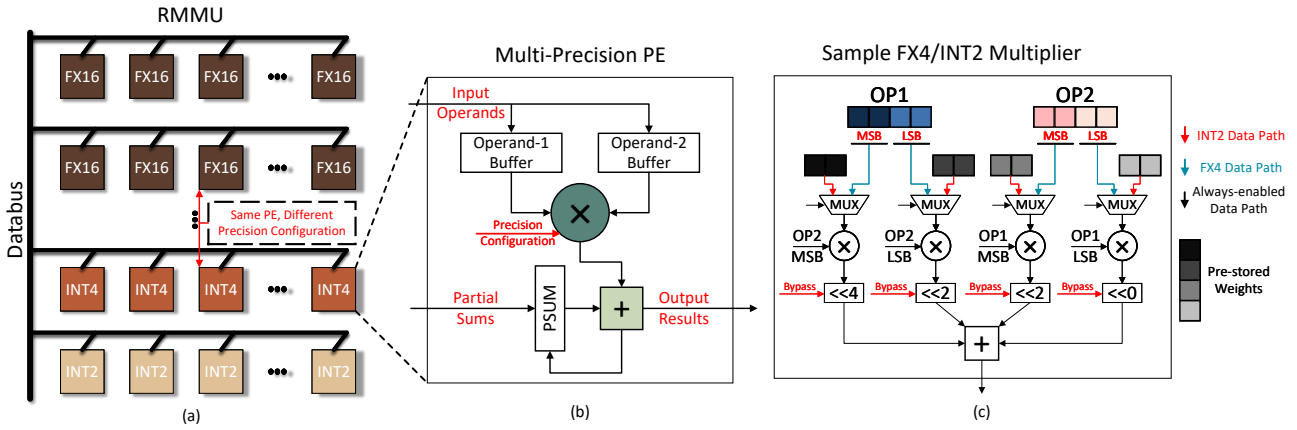


Figure 7: Design of the Reconfigurable Matrix Multiplication Unit. (a) RMMU is composed of a 2D PE array, where each row can be configured to a specific computation precision. (b) Each PE is a multi-precision MAC unit. (c) A sample FX4/IN2 multi-precision multiplier. The key is to build up high precision multiplication data path with low precision multipliers. In low precision mode, we split and multiply the input operands with pre-stored weights and perform in-multiplier accumulation. Therefore, the computation throughput is quadratically improved while input/output bit-width are kept the same as high precision mode.

extra resources to implement all supported precision levels. Besides, the decoupled design can only provide constant computation throughput for each precision, but the ratio of attention detection with respect to the other parts of the model varies from benchmark to benchmark. Thus, we need to dynamically control the computation throughput of attention detection and computation to achieve better resource utilization and energy-efficiency.

To tackle this problem, we present RMMU as shown in Figure 7. The key idea is to design computation engine with configurable precision. As we can see from Figure 7, RMMU is composed of a 32× 16 2-D PE array, where each PE is a fixed-point (FX) MAC unit. The PE supports FX16, INT8, INT4, and INT2 computations. FX16 is used for important attention computation and the rest are for attention detection. The RMMU can be configured to different precision at a row-wise granularity. Therefore, we can flexibly control how many rows of PE use FX16 for computation and how many rows adopt low precision to balance the computation throughput.

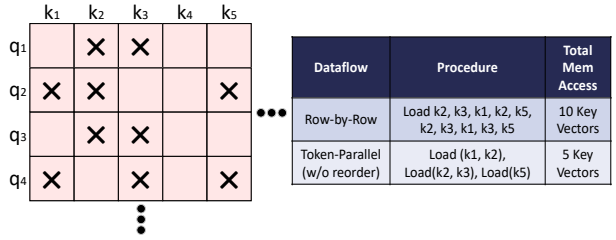
We design the multi-precision multiplier based on two common knowledge of computing arithmetic. Firstly, a fixed-point multiplier is essentially an integer multiplier, only with a different logical explanation of the data. Secondly, we can use low-precision multipliers as building blocks to construct high-precision multipliers [49]. Without loss of generality, we present the implementation of an FX4/INT2 multiplier in Figure 7 (c). As we can see, each operand is divided into MSBs and LSBs and then sent to an INT2 multiplier. A INT2 multiplier takes one fraction from each operands and generates a 4-bit partial sum. Therefore, we need four INT2 multipliers to generate all the required partial sums. The four partial sums are shifted and accumulated to give the final 8-bit result. On the other hand, if the multiplier is in INT2 computation mode, the four INT2 multipliers is able to provide four times higher computation throughput. Note that, we need 16-bit input and 16-bit output each cycle to facilitate all the INT2 multipliers. However, an FX4 multiplication only requires half the bit-width (8-bit for input/output). We address this problem by keeping half the input stationary in the multiplier, and accumulate the INT2 multiplication results before sending them out. Therefore, the input bit-width is the same as FX4 computation while the output consumes 6-bit instead of 16-bit. In other words, when working on INT2 data, we utilize the multiplier as a tiny input-stationary MAC unit which can perform 4 INT2 multiplications and accumulations each cycle.

To summarize, we implement multi-precision PEs in the RMMU and ensure a scalable computation throughput when using the low-precision data. Our final design implements FX-16 multiplier built up from low-precision INT multipliers as discussed above.

4.3 Token-Parallel Dataflow for Sparse Attention Computation

After RMMU generates estimated attention scores, we use the Detector unit to select important attention connections. Specifically, as depicted in Figure 6, the Detector loads estimated attention scores from SRAM and compare them with preset thresholds. A binary mask is generated after the comparison, with 1s representing the selected connections. The Scheduler further processes the binary mask to rearrange the computation order for each token, and stores the reordered connection IDs in the Queue. Later, RMMU will load Key and Value vectors according to these IDs to compute the attention output. Multiple tokens are processed in parallel, each corresponding to one row of the attention matrix. We name this Token-parallel dataflow, which can improve Key/Value data reuse and reduce total memory access. In this subsection, we use three different examples to demonstrate the benefits, challenges, and our solutions to compute the attention output with the detected attention graph and Token parallelism.

Token-Parallel Dataflow. As shown by the example in Figure 8, the 4×5 matrix is the sparse attention graph with important connections marked with crosses. Prior work process each Query (Token) one by one, meaning that the attention weights and output are



Dataflow	Procedure	Total Mem Access
Row-by-Row	Load k2, k3, k1, k2, k5, k2, k3, k1, k3, k5	10 Key Vectors
Token-Parallel (w/o reorder)	Load (k1, k2), Load(k2, k3), Load(k5)	5 Key Vectors

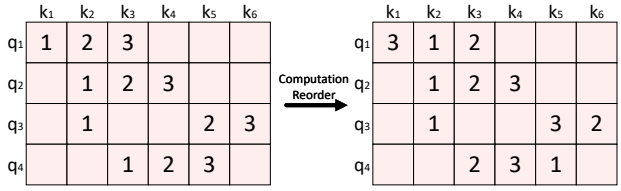
Figure 8: Token-level parallelism reduces key/value vector memory access.

computed row by row. As a result, we need to load ten keys from the memory, even though only four different keys are required. On the contrary, processing all four queries in parallel, as shown in Figure 8, significantly reduces the total memory accesses because some key vectors can be loaded once and shared by multiple rows. This example shows that exploring token-level parallelism benefits memory accessing when attention weight matrix has such row-wise localities. We observe similar locality in real attention graphs. On one hand, there are usually some important tokens in one sentence that attend to multiple tokens. On the other hand, a token is likely to attend to its neighbor tokens within a certain window size. We perform design space exploration (see Section 5.5) and find that processing four queries in parallel is a good trade-off point for hardware resources consumption and memory access savings. Thus, in DOTA, each Header processes four query vectors in parallel.

Workload Balancing. One challenge of parallel token processing is the workload imbalance issue among different rows. Figure 8 shows that different queries may have various numbers of important key vector pairs, which may further cause resource under-utilization and performance degradation. One solution is to let early-finished PEs switch to the processing of other queries. However, this will generate extra inter-PE communications as well as query reloading. Therefore, we tackle this problem directly from algorithm perspective without affecting the underlying hardware. Specifically, we add a constraint to force all the rows in the attention matrix to have the same number of selected attention connections. This constraint ensures that each vertex in the selected sparse attention graph have same number of incoming edges. We will further prove in Section 5.2 that the added constraint has negligible influence on model accuracy.

Out-of-Order Execution. Finally, we propose hardware-enabled out-of-order execution to further improve key/value reuse and reduce total memory access. As shown in Figure 9, suppose all four queries have balanced workload and are processed in parallel. With left-to-right computation order, we first compute (q_1, k_1), (q_2, k_2), (q_3, k_3), (q_4, k_3), and then (q_1, k_2), (q_2, k_3), (q_3, k_5), (q_4, k_4), and finally (q_1, k_3), (q_2, k_4), (q_3, k_6), (q_4, k_5). Consequently, some originally shared keys will have to be reloaded and the locality is broken. In this example, the required total memory access is 11 vectors, which is only one vector less compared with no parallelism.

To address this problem, we design a locality-aware scheduling algorithm to reorder the computation of each query. As shown in Figure 9 and 10, we start with issuing the keys that are shared by most queries. When scheduling partially shared keys like k_2, we also need to schedule computations for the unassigned query,



Dataflow	Procedure	Total Mem Access
Token-Parallel w/o Out-of-order Execution	Load (k1,k2,k3), Load(k2,k3,k5,k4), Load(k3,k4,k6,k5),	11 Key Vectors
Token-Parallel w/ Out-of-order Execution	Load (k2,k5), Load(k3,k6), Load(k1,k4,k5)	7 Key Vectors

Figure 9: Even with token parallelism, the computation order of each row still matters and affects total memory access.

which is q_4. To do so, we first look for keys that belong to q_4 alone. If not found, we move on to keys shared by q_4 and another query, and so on. In this example, there are no key vectors that are owned by q_4. Therefore, we go to the second best choice, which is k_5. Thus, in the first round, we schedule k_2 for $q_{1,2,3}$ and k_5 for q_4. Although this breaks the locality of q_5, the greedy search ensures overall minimal memory access. Besides, since each query is scheduled for exactly one connection at each round, and they have same total connections, this ensures the synchronization of each rows and maximizes resource utilization and performance. The complete scheduling algorithm is presented in Algorithm 1. Note that, the scheduling only needs to be performed once, and the generated computation order is reused for computing attention output using attention weights A and Value matrix V.

Algorithm 1 Locality-Aware Scheduling Algorithm.

Require: A set of buffers B that store the selected connection IDs for query q_1, q_2, q_3, q_4. e.g., B_{0110} stores IDs that are required by q_2 and q_3.

Ensure: A computation order that achieves optimal Key and Value data reuse.

```
Issue all the IDs in B_1111 (required by all 4 queries)
while B_1110 is not empty do
    Issue an ID in B_1110
    if B_0001 is not empty then
        Issue an ID in B_0001
    else
        Search and Issue an ID in B_xxx1
        Move the issued ID from B_xxx1 to B_xxx0
    end if
end while
Repeat 2-10 for all the other buffers.
```

We design a Scheduler to implement the scheduling algorithm. As shown in Figure 10, the Scheduler first stores each connection ID in the corresponding buffer according to the 4-bit binary mask generated after threshold comparison. For example, according to Figure 9, '1' is stored in *buff-1000*, '2' is stored in *buff-1110*. Then, the Scheduler starts issuing computations from *buff-1111*. Besides, when k_5 is scheduled for q_4 during the step-1, '5' will be moved to *buff-0010*, meaning that now it only belongs to q_3. We use a

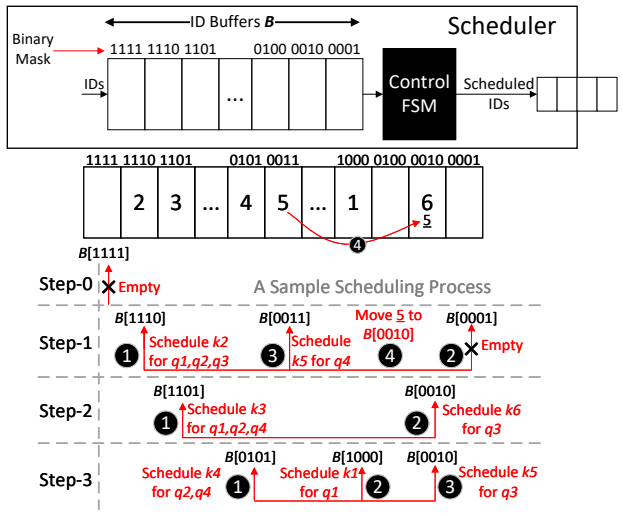


Figure 10: Design of the Scheduler and the scheduling process of Figure 9.

Finite-State Machine to implement the condition statements and control logic.

In summary, we explore token-level parallelism with software-enabled workload-balancing and hardware-enabled out-of-order execution to efficiently compute the attention output. The proposed strategy can be generalized and used in other applications with the same two-step matrix multiplication chain as shown in equation 7. (SoftMax is optional.)

$$O = (Q * K) * V = A * V \tag{7}$$

More importantly, even with out-of-order execution, the final result is automatically generated in a regular order. Because the irregular computation only affects the intermediate matrix A, which is completely consumed during the computation. In contrast, exploring same reordering in CNN would require a crossbar-like design to correctly store the output result [31].

4.4 System Design Completeness

Decoder Processing For decoders, since the input tokens have to be processed sequentially, the core operation would be GEMV and the performance is memory-bounded. DOTA reduces total memory access by efficiently filtering out majority of the attention connections.

Memory Modules The on-chip memory is implemented as banked SRAM module that can be configured to store different types of data. We implement a custom simulator to obtain the capacity and bandwidth requirement of the SRAM module. We facilitate each Lane with a 640KB SRAM (10 64KB banks). Therefore, DOTA has a total on-chip SRAM capacity of 2.5MB. The bandwidth requirements of embedding layer and decoders are significantly higher than other layers. Therefore, we make sure the SRAM bandwidth meets the need of the computation-bounded layers, while leaving embedding and decoder to be memory-bounded.

Table 2: Configurations, Power, and Area of DOTA under 22nm Technology and 1GHz Frequency.

Hardware Module		Configuration	Power(mW)	Area(mm^2)
Lane		4 Lanes per accelerator	2878.33	2.701
Lane	RMMU	32*16 FX-16	645.98	0.609
	Filter	Token Paral. = 4	9.13	0.003
	MFU	16 Exp, 16 Div 16*16 Adder Tree	60.73	0.060
Accumulator		512 accu/cycle	139.21	0.045
DOTA (w/o SRAM)		2TOPS	3017.54	2.746
SRAM		2.5MB	0.51(Leakage)	1.690

5 EVALUATION

In this section we present the evaluation results of DOTA.

5.1 Evaluation Methodology

Benchmarks. Our experiments include series of representative Transformer benchmarks with challenging long-sequence tasks. We first run BERT (large) [12] on question answering task (QA) using the Stanford Question Answering Dataset (SQuAD) [45] v1.1 with a sequence length of 384. To scale our evaluation to longer sequences, we further select three tasks from Long-Range-Arena [51] (LRA), which is a benchmark suite tailored for long-sequence modeling workloads using Transformer-based models. Specifically, the first benchmark performs image classification on CIFAR10 [28], where each image is processed as a sequence length of 1K. The second task is a text classification problem built on the IMDb reviews dataset [33] with a sequence length of 2k. The third task aims to identify if two papers in the ACL Anthology Network [40] contain a citation link. The papers are modeled as 4k input sequences to the Transformer model. Finally, we use GPT-2 [42] to evaluate causal language modeling (LM) on Wikitext-103 [34] using sequences of 4K length.

Software Experiment Methodology. We implement our attention detection mechanism on top of each baseline Transformer, and jointly optimize the model with attention selection enabled. We study the effectiveness of our method by evaluating the model performance in terms of accuracy or perplexity with respect to the retention ratio of the sparse attention graph. Besides, we further compare DOTA's accuracy with state-of-the-art algorithm-hardware co-design (ELSA [18]) and pure software Transformer models presented in LRA [51].

Hardware Experiment Methodology. The system configuration and consumption of DOTA is shown in Table 2. We implement DOTA in RTL, and synthesize it with Synopsys Design Compiler using TSMC 22nm standard cell library to obtain power and area statistics. The power and area of SRAM module are simulated by CACTI [35]. We implement a custom simulator for performance and energy-efficiency evaluation. The simulator is integrated with the software implementations of the Transformer models. We further

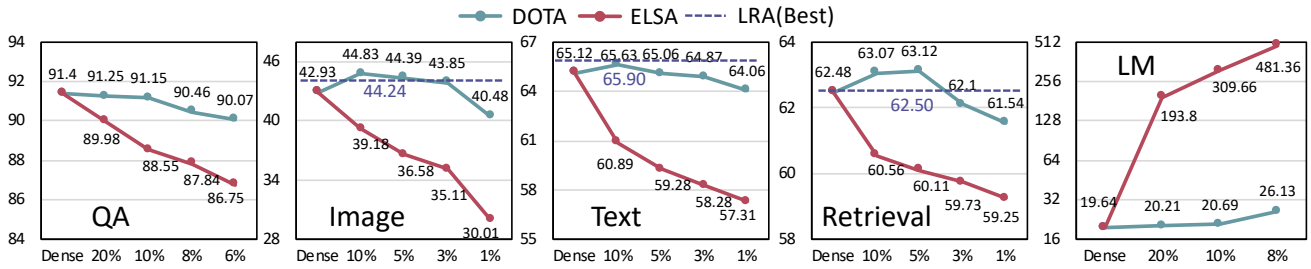


Figure 11: Model accuracy of DOTA comparing with dense baseline and ELSA under different retention ratios across the benchmarks. The performance metric of GPT-2 is perplexity score, the lower the better. The other dataset uses accuracy, the higher the better. The purple line indicates the best results provided by the LRA benchmark.

conduct design space exploration to search for optimal system design choices.

Hardware Baselines We quantitatively compare DOTA with NVIDIA V100 GPU and ELSA [18], while qualitatively discuss the difference between DOTA and other customized hardware (See Section 6). When comparing with GPU, we scale up DOTA's hardware resource to have a comparable peak throughput (12 TOPS) as V100 GPU (14 TFLOPS). The energy consumption of DOTA is also re-simulated for fair comparison. When comparing with ELSA's performance, we extend and validate our simulator to support ELSA's dataflow. Then, we re-synthesize DOTA with the same data representation, computation resources and technology node as ELSA to compare the energy-efficiency.

5.2 Algorithm Performance

We present the model accuracy of DOTA in Figure 11, and compare it with dense Transformer model as well as other software baselines. For DOTA, we first add the row-wise attention connection constraint and then select optimal quantization precision and dimension reduction factor (σ) based on design space exploration (Section 5.5). For ELSA, our implementation delivers aligned results on QA compared with the original paper, and we extend it to other datasets.

As we can see, across all the tested benchmarks, DOTA is able to achieve comparable or slightly higher model accuracy compared with the dense baseline, while selecting only 3 ~ 10% of the attention connections. Furthermore, DOTA significantly outperforms ELSA in accuracy-retention trade-offs. For example, on QA task with 1.5% of accuracy degradation interval, DOTA delivers 3.3× higher reduction ratio by keeping 6% of the connections, while ELSA needs to keep 20%. The gap becomes even larger on long-sequence benchmarks, which indicates that our detection method is more scalable with long sequence. Furthermore, we also provide leading results given by the LRA [51] benchmarks on image classification, text classification, and document retrieval tasks. As shown in the figure, DOTA achieves on-par or better accuracy than LRA's leading results with 5% to 10% of retention ratio.

5.3 Speedup

Figure 12 presents the speedup of DOTA over the baselines. We evaluate both stand along attention block as well as the end-to-end performance improvements. We provide two versions of DOTA by setting the accuracy degradation of DOTA-C (Conservative) to be less than 0.5%, and limiting the degradation of DOTA-A (Aggressive) within 1.5%. As for ELSA, although it fails to reach the above accuracy requirement, we follow the original setting [18] and set the retention ratio to be 20% for performance evaluation.

As we can see, comparing with GPU, DOTA-C achieves 152.6× and 9.2× average speedup on attention computation and Transformer inference, respectively. On the other hand, DOTA-A achieves on average 341.8× and 9.5× speedups at the cost of a slightly higher accuracy degradation. The speedup mainly comes from three aspects. Firstly, DOTA benefits from highly specialized and pipelined datapath. Secondly, the attention detection mechanism significantly reduces the total computations. Finally, the Token-parallel dataflow with workload balancing and out-of-order execution further improves resource utilization.

The end-to-end speedup is lower than that of attention computation, since the proposed detection method is tailored to the cost reduction of self-attention blocks. We add another baseline by assuming the accelerator always works at its peak throughput, and the attention computation has a ignorable cost. Combining this peak throughput assumption and Amdahl's law [3], we can derive the theoretical speedup upper bound for DOTA. As we can see, the real performance of DOTA is relatively close to the upper bound by virtue of the extremely small retention ratio and hardware specialization. We only compare DOTA and ELSA on attention computation performance, because ELSA does not support end-to-end Transformer execution. As we can see from Figure 12 (b), on average, DOTA-C is 4.5× faster than ELSA and DOTA-A is 10.6× faster. This improvements mainly come from lower retention ratio and Token-parallel dataflow.

The latency breakdown in Figure 12 (c) delivers two key messages. Firstly, the latency of attention estimation is negligible compared with the overall consumption. Therefore, the Detector is both accurate and hardware efficient as we expected. Secondly, with the proposed detection method and system architecture, the cost of attention has been significantly reduced. The new performance bottleneck is Linear computation, which can be optimized with weight pruning and quantization. These classic NN optimization techniques can be fluently transplanted on DOTA, because our system is designed on top a GEMM accelerator with multi-precision arithmetic support and sparse computation dataflow. Overall, DOTA delivers scalable Transformer inference acceleration.

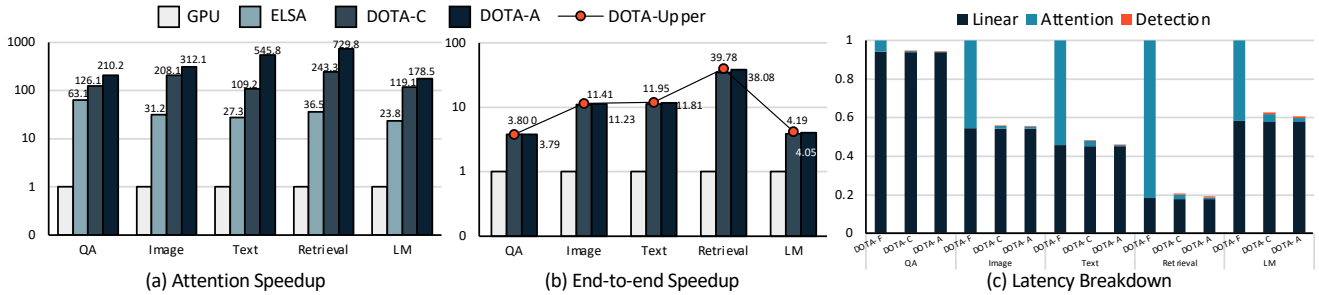


Figure 12: (a) Speedup of DOTA over GPU and ELSA on attention block. (b) End-to-end speedup over GPU. Red dots indicate the theoretical performance upper-bound of an accelerator. (c) Normalized latency breakdown of DOTA. DOTA-F means to compute the *Full* attention graph with DOTA without detection and omission. DOTA-C (Conservative) and DOTA-A (Aggressive) both adopt attention detection, while DOTA-C allows for an accuracy degradation less than 0.5% and DOTA-A allows for 1.5%.

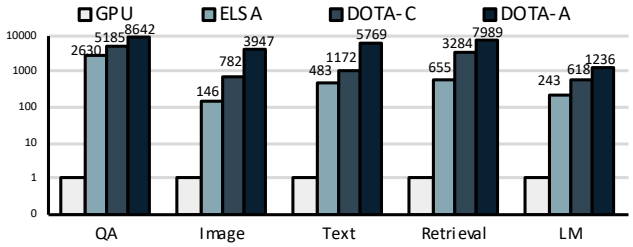


Figure 13: Energy-efficiency comparisons.

5.4 Energy-Efficiency

As shown in Table 2, each DOTA accelerator consumes a total power of 3.02W. RMMU and Accumulator are the two major contributing factors to the dynamic power consumption, whereas SRAM and RMMU together occupies the most chip area. We compare DOTA's energy-efficiency with GPU and ELSA. The results are shown in Figure 13. As we can see, DOTA-C achieves 618~5185× and 1.97~5.14×energy-efficiency improvements over GPU and ELSA, while DOTA-A achieves 1236~8642× and 3.29~12.20× improvements over these two baselines. The energy saving mainly comes from two parts. Firstly, despite the attention estimation overhead, the proposed attention detection largely reduces overall cost of attention computation and memory access. Secondly, both external memory access and on-chip SRAM access are saved to a large extent. On one hand, the hardware specialization helps improve intermediate data reuse between the pipeline stages. On the other hand, Token-parallel dataflow effectively utilizes attention connection locality to improve Key/Value data reuse. The energy breakdown of DOTA exhibits similar pattern as the latency breakdown. That is, with effective attention reduction, FC-layer consumes around 84.9~99.3% of the total energy cost, while attention detection only consumes 0.11~0.34%. This further illustrates the efficiency of the proposed algorithm-hardware co-design.

5.5 Design Space Exploration

We search and select optimal architectural settings for DOTA through design space exploration.

Dimension Reduction Scale As discussed above, the dimension reduction scale σ directly affects the size of the input and weight matrices involved in attention detection. Therefore, a small σ can effectively control the overhead of attention estimation, but the Detector's performance will also be limited. We experiment on the Text classification benchmark, fixing the retention ratio and quantization precision while only adjusting the scale values. The results are shown in Figure 14 (a).

As we can see, for Text classification, the scale factor can be as small as 0.2 without affecting the overall model accuracy. Therefore, the hidden dimension in approximation is *floor*(64*0.2)=12, compared with the original dimension 64. Besides, σ is a hyper-parameter which does not influence the underlying hardware. Therefore, each benchmark can use its own optimal σ value.

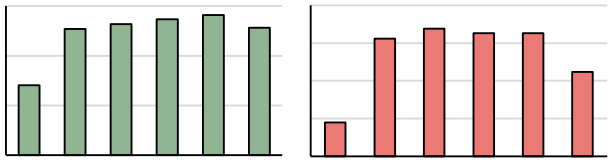


Figure 14: Influence of (a) dimension reduction factor σ and (b) quantization precision on overall model accuracy using Text classification benchmark. Retention ratio = 10%.

Precision of Attention Detection Another factor that affects the attention detection cost is the choice of quantization precision. Furthermore, the precision also influences the design complexity of RMMU. For each benchmark, we fix σ and retention ratio and sweep over different quantization precision. Figure 14 (b) presents the experiment results on Text classification benchmark. As we can see, the quantization precision could be as low as 2-bit with negligible accuracy degradation. After our experiments, we found that INT4 is a safe precision for all the benchmarks, while some can tolerate INT2 computations. Therefore, our final RMMU design supports INT2, INT4, and INT8 apart from FX16. INT8 computation is required when X, $\tilde{W_Q}$, and $\tilde{W_K}$ are INT4 data. As the estimated $\tilde{Q}$ and $\tilde{K}$ will be in INT8 precision.

Token Parallelism Our token-parallel dataflow leverages locality among important attention distribution to improve memory access. Higher parallelism increases data reuse and reduces total memory access, but also results in growing size of the Scheduler unit.

Therefore, we aim to find an optimal trade-off point that achieves lowest overall energy consumption. Figure 15 shows the case on Text classification benchmark with Retention ratio to be 10%. The left axis indicates the normalized memory access cost of Key and Value, while the right axis is the required number of buffers in the Scheduler. Figure 15 mainly delivers two key messages. Firstly, as shown by the solid blue bar, leveraging row-parallelism does help reduce memory accesses, but increasing the parallelism has diminishing returns. This is because attention distribution exhibits certain but only a limited degree of locality. Secondly, increasing row-parallelism causes exponential growth in scheduling overhead. In Figure 15, this is shown by the red line (buffer requirement) and the dotted blue bar (scheduling energy consumption). After summing up the memory cost (solid blue bar) and scheduling cost (dotted blue bar) together, we choose the shortest one because it represents the sweetest spot with the lowest total energy consumption. As we can see, parallelism 4 has the lowest total height, which means 4 is the best setting for Text classification. We also evaluate on other benchmarks and most benchmarks have an optimal parallelism to be or around 4. Therefore, we choose 4 as the final setting in DOTA.

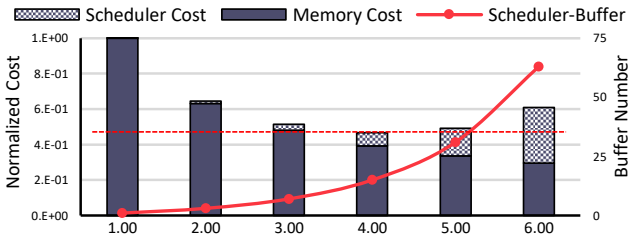


Figure 15: Key/Value memory access (left axis) and Scheduler buffer requirement (right axis) with different Token parallelism. The hatched area is the projected cost of Scheduler.

6 RELATED WORK

In this Section, we mainly discuss related work on efficient Transformer models from the algorithm perspective and hardware accelerators for Transformers and Self-Attention. For general DNNs, quantization and low-precision support have been proposed [16, 19–21, 36]. While sharing the high-level similarity, our method focuses on attention operations that are not parameterized. Hence, those methods applied on model parameters are not applicable to our scenario. Our work is in the scope of dynamic pruning on attentions as we discussed and compared with other related work. Approximation for DNNs is also a line of related work [2, 29, 32, 44]. Finally, hardware accelerators for DNNs are related to executing the non-attention components of Transformers [6, 8–10, 13–15, 24–26, 30, 47, 55].

6.1 Efficient Transformer Models

Recent studies propose efficient variants of Transformer models to mitigate the quadratic memory complexity of long sequence modeling [27, 48, 50]. However, these methods are impractical for efficient inference as they focus on training memory footprint reduction while trading off more computations for clustering or grouping.

Another line of work exploit static or fixed sparse patterns in attention, such as local windows, block-wise, dilated, or a combination of static patterns [11, 39, 54]. However, as discussed in Section 2, the sparse attention graphs are inherently dynamic depending on input sequences. Hence, these approaches lack the capability of capturing dynamic sparse attentions.

6.2 Attention and Transformer Accelerators

There have been a few recently proposed work targeting the acceleration of attention and Transformer. MnnFast [22] skips the computation of specific value vectors if its attention weights is lower than the threshold. This method can only benefit the attention output computation rather than attention weights computation. A^3 [17] is the first work to apply approximation to the attention weights for computation reduction. However, A^3 involves a sorting-based preprocessing phase that needs to be done outside the accelerator, causing inevitable performance and energy overhead. ELSA [18] improves the approximation method by directly using sign random projection to estimate the angle between query and key vectors. Although the approximation becomes much more hardware friendly, the detection accuracy and model quality is hurt. DOTA addresses all of the above limitations by simultaneously concerning detection accuracy and efficiency. In terms of hardware design, prior work only implements attention block with no token parallelism, while DOTA supports end-to-end inference acceleration with Token-parallel dataflow to improve system performance.

SpAtten [53] proposes cascade token pruning and head pruning to reduce the cost of both self-attention block and subsequent layers in the Transformer model. The proposed method can be regarded as adding structured sparsity constraints to the attention matrix, as it directly removes several rows and columns. Based on our visualization and experiments, we believe that despite a certain degree of locality, such constraint is not flexible enough to capture the irregularly distributed attention connections. As for hardware design, SpAtten supports both decoder and encoder processing, but it is also mostly tailored to attention acceleration with very few discussions on end-to-end execution.

Finally, OPTIMUS [37] proposes a GEMM architecture to accelerate Transformer inference. It focuses on accelerating sequential decoding process and proposes technique to maintain resource utilization. Although OPTIMUS avoids computing redundant attention weights, such redundancy is due to naturally existed token dependency, rather than the weak connections we discussed in this work. Thus, the self-attention still has quadratic cost and OPTIMUS does not scale on longer sequences.

7 CONCLUSION

In this work, we address the challenge of scalable Transformer inference. Specifically, we first propose algorithm optimization to reduce the quadratic cost of self-attention mechanism. Our method efficiently detects and omits weak connections in attention graphs to skip the corresponding computations and memory accesses. Furthermore, we provide system-level support for end-to-end large Transformer model inference. We first effectively abstract the Transformer model to design a scalable and unified architecture. Then,

we implement the proposed attention detection method with efficient hardware specialization techniques. Our final evaluation results sufficiently demonstrate the effectiveness of the proposed algorithm and system design.

ACKNOWLEDGMENTS

This work was supported in part by NSF 2124039. Use was made of computational facilities purchased with funds from the National Science Foundation (OAC-1925717) and administered by the Center for Scientific Computing (CSC). The CSC is supported by the California NanoSystems Institute and the Materials Research Science and Engineering Center (MRSEC; NSF DMR 1720256) at UC Santa Barbara. We would also like to thank our shepherd and all the reviewers, who offered valuable suggestions to help us improve our paper.

REFERENCES

[1] Dimitris Achlioptas. 2001. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 274–281. https://doi.org/10.1145/375551.375608

[2] Vahideh Akhlaghi, Amir Yazdanbakhsh, Kambiz Samadi, Rajesh K Gupta, and Hadi Esmaeilzadeh. 2018. Snapea: Predictive early activation for reducing computation in deep convolutional neural networks. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 662–673. https://doi.org/10.1109/ISCA.2018.00061

[3] Gene M Amdahl. 2013. Computer architecture and amdahl's law. *Computer* 46, 12 (2013), 38–46. https://doi.org/10.1109/MC.2013.418

[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. arXiv:2005.14165 [cs.CL]

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European Conference on Computer Vision*. Springer, 213–229. https://doi.org/10.1007/978-3-030-58452-8_13

[6] Srimat Chakradhar, Murugan Sankaradas, Venkata Jakkula, and Srihari Cadambi. 2010. A dynamically configurable coprocessor for convolutional neural networks. *ACM SIGARCH Computer Architecture News* 38, 3 (2010), 247–257. https://doi.org/10.1145/1815961.1815993

[7] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. 2020. Generative Pretraining From Pixels. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 1691–1703. https://proceedings.mlr.press/v119/chen20s.html

[8] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. 2014. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM Sigplan Notices* 49, 4 (2014), 269–284. https://doi.org/10.1145/2541940.2541967

[9] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. 2014. Dadiannao: A machine-learning supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 609–622. https://doi.org/10.1109/MICRO.2014.58

[10] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. 2017. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits* 52, 1 (2017), 127–138. https://doi.org/10.1109/ISSCC.2016.7418007

[11] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating Long Sequences with Sparse Transformers. arXiv:1904.10509 [cs.LG]

[12] 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, Volume 1 (Long and Short Papers)*. 4171–4186. https://doi.org/10.18653/v1/n19-1423

[13] Zidong Du, Robert Fasthuber, Tianshi Chen, Paolo Ienne, Ling Li, Tao Luo, Xiaobing Feng, Yunji Chen, and Olivier Temam. 2015. ShiDianNao: Shifting vision processing closer to the sensor. In *ACM SIGARCH Computer Architecture News*, Vol. 43. ACM, 92–104. https://doi.org/10.1145/2749469.2750389

[14] Clément Farabet, Berin Martini, Benoit Corda, Polina Akselrod, Eugenio Culurciello, and Yann LeCun. 2011. Neuflow: A runtime reconfigurable dataflow processor for vision. In *2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops 2011)*. IEEE, 109–116. https://doi.org/10.1109/CVPRW.2011.5981829

[15] Mingyu Gao, Jing Pu, Xuan Yang, Mark Horowitz, and Christos Kozyrakis. 2017. Tetris: Scalable and efficient neural network acceleration with 3d memory. *ACM SIGOPS Operating Systems Review* 51, 2 (2017), 751–764. https://doi.org/10.1145/3037697.3037702

[16] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep learning with limited numerical precision. In *International conference on machine learning*. PMLR, 1737–1746.

[17] Tae Jun Ham, Sung Jun 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. A3: Accelerating Attention Mechanisms in Neural Networks with Approximation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 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 *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 692–705. https://doi.org/10.1109/ISCA52012.2021.00060

[19] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research* 18, 1 (2017), 6869–6898.

[20] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2704–2713. https://doi.org/10.1109/CVPR.2018.00286

[21] Shubham Jain, Swagath Venkataramani, Vijayalakshmi Srinivasan, Jungwook Choi, Pierce Chuang, and Leland Chang. 2018. Compensated-DNN: Energy efficient low-precision deep neural networks by compensating quantization errors. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 1–6. https://doi.org/10.1109/DAC.2018.8465893

[22] Hanhwi Jang, Joonsung Kim, Jae-Eon Jo, Jaewon Lee, and Jangwoo Kim. 2019. Mnnfast: A fast and scalable system architecture for memory-augmented neural networks. In *Proceedings of the 46th International Symposium on Computer Architecture*. 250–263. https://doi.org/10.1145/3307650.3322214

[23] Jianqiu Ji, Jianmin Li, Shuicheng Yan, Bo Zhang, and Qi Tian. 2012. Super-bit locality-sensitive hashing. In *Advances in neural information processing systems*. 108–116.

[24] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ACM, 1–12. https://doi.org/10.1145/3079856.3080246

[25] Patrick Judd, Jorge Albericio, Tayler Hetherington, Tor M Aamodt, and Andreas Moshovos. 2016. Stripes: Bit-serial deep neural network computing. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 1–12. https://doi.org/10.1109/LCA.2016.2597140

[26] Duckhwan Kim, Jaeha Kung, Sek Chai, Sudhakar Yalamanchili, and Saibal Mukhopadhyay. 2016. Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 380–392. https://doi.org/10.1109/ISCA.2016.41

[27] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rkgNKkHtvB

[28] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

[29] Dongwoo Lee, Sungbum Kang, and Kiyoung Choi. 2018. ComPEND: Computation Pruning through Early Negative Detection for ReLU in a deep neural network

accelerator. In *Proceedings of the 2018 International Conference on Supercomputing*. 139–148. https://doi.org/10.1145/3205289.3205295

[30] Daofu Liu, Tianshi Chen, Shaoli Liu, Jinhong Zhou, Shengyuan Zhou, Olivier Teman, Xiaobing Feng, Xuehai Zhou, and Yunji Chen. 2015. Pudiannao: A polyvalent machine learning accelerator. In *ACM SIGARCH Computer Architecture News*, Vol. 43. ACM, 369–381. https://doi.org/10.1145/2694344.2694358

[31] Liu Liu, Zheng Qu, Lei Deng, Fengbin Tu, Shuangchen Li, Xing Hu, Zhenyu Gu, Yufei Ding, and Yuan Xie. 2020. DUET: Boosting Deep Neural Network Efficiency on Dual-Module Architecture. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 738–750. https://doi.org/10.1109/MICRO50266.2020.00066

[32] Zhenhong Liu, Amir Yazdanbakhsh, Taejoon Park, Hadi Esmaeilzadeh, and Nam Sung Kim. 2018. SiMul: An algorithm-driven approximate multiplier design for machine learning. *IEEE Micro* 38, 4 (2018), 50–59. https://doi.org/10.1109/MM.2018.043191125

[33] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. 142–150.

[34] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843* (2016).

[35] Naveen Muralimanohar, Rajeev Balasubramonian, and Norman Jouppi. 2009. Cacti 6.0: A tool to model large caches. *HP Laboratories* (01 2009).

[36] Eunhyeok Park, Dongyoung Kim, and Sungjoo Yoo. 2018. Energy-efficient neural network accelerator based on outlier-aware low-precision computation. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 688–698. https://doi.org/10.1109/ISCA.2018.00063

[37] Junki Park, Hyunsung Yoon, Daehyun Ahn, Jungwook Choi, and Jae-Joon Kim. 2020. OPTIMUS: OPTImized matrix MUltiplication Structure for Transformer neural network accelerator. *Proceedings of Machine Learning and Systems* 2 (2020), 363–378.

[38] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *International Conference on Machine Learning*. PMLR, 4055–4064.

[39] Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen tau Yih, Sinong Wang, and Jie Tang. 2020. Blockwise Self-Attention for Long Document Understanding. arXiv:1911.02972 [cs.CL]

[40] Dragomir R Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. 2013. The ACL anthology network corpus. *Language Resources and Evaluation* 47, 4 (2013), 919–944. https://doi.org/10.1007/s10579-012-9211-2

[41] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. (2018).

[42] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[43] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683* (2019).

[44] Arnab Raha, Swagath Venkataramani, Vijay Raghunathan, and Anand Raghunathan. 2016. Energy-efficient reduce-and-rank using input-adaptive approximations. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25, 2 (2016), 462–475. https://doi.org/10.1109/TVLSI.2016.2586379

[45] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. arXiv:1606.05250 [cs.CL]

[46] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-Shot Text-to-Image Generation. arXiv:2102.12092 [cs.CV]

[47] Brandon Reagen, Paul Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, José Miguel Hernández-Lobato, Gu-Yeon Wei, and David Brooks. 2016. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 267–278. https://doi.org/10.1109/ISCA.2016.32

[48] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics* 9 (2021), 53–68. https://doi.org/DOI:10.1162/tacl_a_00353

[49] Hardik Sharma, Jongse Park, Naveen Suda, Liangzhen Lai, Benson Chau, Vikas Chandra, and Hadi Esmaeilzadeh. 2018. Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Network. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. 764–775. https://doi.org/10.1109/ISCA.2018.00069

[50] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020. Sparse sinkhorn attention. In *International Conference on Machine Learning*. PMLR, 9438–9447. http://proceedings.mlr.press/v119/tay20a.html

[51] 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 *International Conference on Learning Representations*. https://openreview.net/forum?id=qVyeW-grC2k

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 6000–6010.

[53] Hanrui Wang, Zhekai Zhang, and Song Han. 2021. SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 97–110. https://doi.org/10.1109/HPCA51647.2021.00018

[54] 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.). https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html

[55] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. 2015. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 161–170. https://doi.org/10.1145/2684746.2689060